



SPRINGFIELD PUBLIC SCHOOLS
ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Springfield Public Schools Springfield, New Jersey 07081

App Creators Grade 7

Curriculum Guide

2022

This curriculum may be modified through varying materials, techniques, and strategies as per an individual student's IEP.

*Springfield Board of Education
Approved July 25, 2022*



Springfield Public Schools Springfield, New Jersey

**Dr. Rachel Goldberg
Superintendent of Schools**

**Erica Scudero
Assistant Superintendent**

Curriculum Developed By:

Tim Kielty ~ Middle School Principal

Gregory Salmon ~ STEM Supervisor

Christina Rodriguez ~ Math Teacher

Springfield Board of Education
PO Box 210
Springfield, New Jersey 07081

"Cultivating compassionate and extraordinary learners"

Rachel Goldberg
Superintendent of Schools
(973) 376-1025

Michelle Calas
School Business Administrator/
Board Secretary
(973) 376-1025

Resolution: School Government
Introduced By: Mr. Miller
Seconded By: Mrs. Gamarekian
Adopted: July 25, 2022

Curriculum Adoption

To adopt the following curriculum and courses for the 2022-2023 school year:

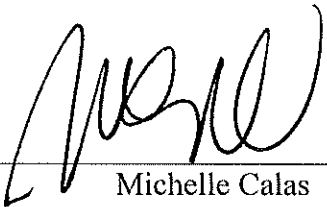
- Revised Elementary Curriculum due to revised state standards.
 - K-5 Social Studies
- Revised Middle School Curriculum due to revised state standards.
 - Grades 6-8 Social Studies
- Revised High School Curriculum due to revised state standards.
 - AP Macroeconomics
 - AP European History
 - AP US Government & Politics
 - AP US History
 - Contemporary Global Studies
 - Diversity Studies
 - Holocaust & Genocide
 - Sociology
 - US History to 1900
 - World History
 - World History Honors
 - 20th Century & Beyond
- Revised Middle School Courses due to new standards in Computer Science
 - PLTW Gateway: Computer Science for Innovators and Makers
 - PLTW Gateway: App Creators
- Revised High School Courses due to new standards in Computer Science
 - Introduction to Java
 - AP Computer Science Principles
 - Advanced Computer Science - Data Structures: Java
- New Elementary Courses
 - Kindergarten Computer Science
 - Grade 1 Computer Science
 - Grade 2 Computer Science
 - Anticipated - moving PLTW Launch down to Grade 3
 - PLTW Launch Grade 4
 - PLTW Launch Grade 5
- New Middle School Elective Course
 - Forensics

- New Elective Courses at FMG
 - Creative Writing
 - Drama/Theater/Playwriting
 - Journalism
 - Mock Trial
 - Model UN
 - Mysteries in History
 - Project Citizen
- New ELA Courses at FMG
 - Accelerated Communication Arts Grade 7
 - Accelerated Communication Arts Grade 8
- New Elective Courses at JDHS:
 - African American Studies

I, Michelle Calas, School Business Administrator/Board Secretary of the Springfield Board of Education, hereby certify that the extract from the minutes of a meeting of the Board of Education duly called and held on July 25, 2022 has been compared by me with the original minute book of said Board of Education, and is a true, complete and exact copy therefor, as far as the same relates to the subject matter referred to said contract.

IN WITNESS THEREOF, I have hereunto set my hand and seal of the Springfield Board of Education, this 26th of July 2022.

SEAL



Michelle Calas
School Business Administrator/Board Secretary



Table of Contents

New Jersey Student Learning Standards: Pages 4-5

PLTW App Creators Course Outline: Page 6

App Creators General Preparations and Considerations: Pages 7-9

Lessons: Pages 10-85

PLTW Curriculum Framework: Pages 86-100



New Jersey Student Learning Standards

8.1 Computer Science

Computing Systems

People interact with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form.

- 8.1.8.CS.1: Recommend improvements to computing devices in order to improve the ways users interact with the devices.
- 8.1.8.CS.4: Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems.

Impacts of Computing

Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and, in turn, computing influences new cultural practices.

- 8.1.8.IC.1: Compare the trade-offs associated with computing technologies that affect individual's everyday activities and career options.

Data & Analysis

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, so the need to process data effectively is increasingly important. Data is collected and stored so that it can be analyzed to better understand the world and make more accurate predictions.

- 8.1.8.DA.4: Transform data to remove errors and improve the accuracy of the data for analysis.
- 8.1.8.DA.5: Test, analyze, and refine computational models.
- 8.1.8.DA.6: Analyze climate change computational models and propose refinements.

Algorithms & Programming

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems.



- 8.1.8.AP.1: Design and illustrate algorithms that solve complex problems using flowcharts and/or pseudocode.
- 8.1.8.AP.2: Create clearly named variables that represent different data types and perform operations on their values.
- 8.1.8.AP.3: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- 8.1.8.AP.4: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- 8.1.8.AP.5: Create procedures with parameters to organize code and make it easier to reuse.
- 8.1.8.AP.6: Refine a solution that meets users' needs by incorporating feedback from team members and users.
- 8.1.8.AP.7: Design programs, incorporating existing code, media, and libraries, and give attribution.
- 8.1.8.AP.8: Systematically test and refine programs using a range of test cases and users.
- 8.1.8.AP.9: Document programs in order to make them easier to follow, test, and debug.

Career Readiness, Life Literacies, and Key Skills (Standard 9):

- 9.2.8.CAP.5 - Develop a personal plan with the assistance of an adult mentor that includes information about career areas of interest, goals and an educational plan.
- 9.2.8.CAP.10 - Evaluate how careers have evolved regionally, nationally, and globally.
- 9.2.8.CAP.12 - Assess personal strengths, talents, values, and interests to appropriate jobs and careers to maximize career potential.
- 9.2.8.CAP.19 - Relate academic achievement, as represented by high school diplomas, college degrees, and industry credentials, to employability and to potential level
- 9.4.8.CT.1 - Evaluate diverse solutions proposed by a variety of individuals, organizations, and/or agencies to a local or global problem, such as climate change, and use critical thinking skills to predict which one(s) are likely to be effective (e.g., MS-ETS1-2).
- 9.4.8.IML.4 - Ask insightful questions to organize different types of data and create meaningful visualizations.
- 9.4.8.IML.7 - Use information from a variety of sources, contexts, disciplines, and cultures for a specific purpose (e.g., 1.2.8.C2a).
- 9.4.8.TL.3 - Select appropriate tools to organize and present information digitally.
- 9.4.8.TL.5 - Compare the process and effectiveness of synchronous collaboration and asynchronous collaboration.



Have you ever wondered how mobile apps are created?

Students learn and apply computational thinking and technical knowledge and skills to create mobile apps. Students also acquire and apply skills pertaining to the design process, problem solving, persistence, collaboration, and communication.

Go beyond being an app consumer and become an app creator!

App Creators introduces students to the field of computer science and the concepts of computational thinking, through the creation of mobile apps. Students are challenged to be creative and innovative, as they collaboratively design and develop mobile solutions to engaging, authentic problems. Students experience the positive impact of the application of computer science to society as well as other disciplines, particularly biomedical science.

The unit provides students opportunities for self-expression. Teams identify a personal or community problem of interest to them that can be solved with a mobile app solution. The problem can address issues such as health and wellness, the environment, school culture, emergency preparedness, education, community service—the options are endless!

App Creators Lesson Summary

Lesson 1	Let's Create an App!
Lesson 2	Taking It to the Next Level
Lesson 3	The App Challenge

Lesson 1: Let's Create an App!

Students are introduced to the concept of pair programming, app development, and the MIT App Inventor development tool. They learn about the Model-View-Controller (MVC) design pattern, app graphical design, event-driven programming, debugging, and algorithm creation using variables and conditional logic. They create engaging biomedical science apps and fun interactive games that apply these concepts and use basic user interface features, media, and animation.

Lesson 2: Taking It to the Next Level

Students further explore the concepts investigated in Lesson 1 and build upon their skills to use data in mobile applications. They create algorithms using loops to streamline repetition and iterate through lists, and create procedures to abstract the details of a task and reduce redundancy. They learn to organize and store persistent data collected from user input and device sensors.

Lesson 3: The App Challenge

Students apply all of the knowledge and skills they have acquired to design and create a mobile app solution for a personal or community problem. They apply the design process and computational thinking skills to decompose the problem into smaller modules. Following user-centered design principles, they design and create an appropriate user interface and program the app to produce the desired behavior.



App Creators

General Preparation and Considerations

FEEDBACK ON ACTIVITIES, PROJECTS, AND PROBLEMS

Thank you for being a PLTW App Creators teacher! Below are the links to two surveys that you can use to provide feedback about your teaching experience.

The top link should be used to provide feedback for each lesson.

The bottom link is to provide feedback on the course overall, and should be completed at the conclusion of the course. Your feedback is confidential and helps inform improvements to PLTW curriculum and professional development.

 Lesson Feedback

 PLTW Gateway Teacher Survey

COURSE PREFACE

PLTW App Creators introduces students to the field of computer science and the concepts of computational thinking, through the creation of mobile apps. Students are challenged to be creative and innovative, as they collaboratively design and develop mobile solutions to engaging, authentic problems. Students experience the positive impact of the application of computer science to society as well as to other disciplines, particularly biomedical science.

The unit provides students opportunities for self-expression. Teams identify a personal or community problem of interest to them that can be solved with a mobile app solution. The problem can address issues such as health and wellness, the environment, school culture, emergency preparedness, education, community service—the options are endless!

COURSE MATERIALS

- PLTW Gateway Notebook
- Computer with internet access and MIT App Inventor 2 software installed
- Android™ tablet with MIT AI2 Companion installed (not needed in Activity 1.2 or 1.4)
- Google account (not needed in Activity 1.2 or 1.4)

COURSE RESOURCES

-  App Inventor Setup Guide
-  App Inventor Troubleshooting Guide
-  Design Process
-  Pair Programming Graphic

NATIONAL AND STATE STANDARDS ALIGNMENT

For the most up-to-date Standards Alignment, please visit [PLTW's Alignment to Standards](#).



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Safety Considerations

- Using MIT App Inventor requires students to log in to the App Inventor website with Google accounts. Be aware of your school protocols for using the internet at school, and check whether you might need to have parents or guardians sign any agreements for terms of use.
- App Inventor has a very large and active online community where students and teachers can share projects on the Gallery page. The App Inventor team works diligently to keep the gallery content student-appropriate; however, it is important to be aware of the possibility of finding some inappropriate content online. To report inappropriate content to the App Inventor team, access the specific app's detail page in the Gallery, then click the **REPORT** button (next to DESCRIPTION and SHARE).
- If students use the internet to access online photo repositories for their challenge activities, make sure that the sites they visit are safe and appropriate.

Ethical Considerations

This unit requires students to work online and use Google accounts. This is a good opportunity to teach students about the ethical use of online resources. Students should not share user names and passwords, they should not try to hack into other student accounts, and should not copy other people's code (plagiarism) or digital assets.

Student Accommodations

- Students with visual needs may benefit from a [Google Chrome browser extension](#) to convert the App Inventor website to high-contrast color schemes. Other web browsers may have similar extensions to assist with converting the programming environment to high-contrast color schemes.
- If students get overwhelmed with reading programs that are lengthy or have many event handlers, you can suggest that they collapse the event handlers that they are not working on. This will free up space in the *Blocks Viewer* and allow them to focus on the blocks they are working on.
- For students with certain visual impairments, using the browser's zoom feature to zoom in on the course content, including the code, can be helpful.

Unit Preparation

To prepare your classroom environment before you begin this unit, complete the following steps.

1. Set up classroom computers with internet access and be sure the App Inventor software is installed. Refer to the App Inventor Setup Guide for more information.
2. Set up Google accounts for students to use.
 - If your school uses Google Apps for Education, students can use their school Google accounts with App Inventor.
 - If you don't use Google Apps for Education, consider the following options:
 - Create generic class accounts. Remember to make the account name something easy for all students to remember—a recommended name is the combination of school name, course name, and a unique number, like SchoolName_AppCreators_1. You can recycle these accounts every term for reuse by the next group of students.
 - Have students create Google accounts (please check with your school first). This is a good opportunity to reinforce online safety and privacy by asking students to avoid using any personal or identifying information when creating accounts and working online.
3. Decide where students will save their files for this unit. Some schools use network drives; some use cloud-based storage; others ask students to use their personal flash drives. If you need to contact your IT department for assistance, make sure to do so before the first day of class. Regardless of where the files will reside, guide students to organize their files under one main folder for this unit. For every activity in which students are asked to download any type of file, have them create a new subfolder with that activity number and save the corresponding files in that subfolder to keep their files organized properly.
4. When students use pair programming, consider asking them to share the App Inventor .aia file with their partners at the end of each class. This will guarantee access to the code in case a student is absent the following class.
5. Before you facilitate an APB within App Inventor, determine how you will facilitate student sharing of the mobile devices and how they will deploy and test their apps on both the mobile device and the emulator.
6. It is recommended that you work with your IT department to make sure that any time students use the internet to search for information or media files to use in Project 2.5 or Unit Problem, they are only able to access sites that are safe and appropriate. Both student computers and tablets need to be set up accordingly.

Unit Facilitation

- For each APB that involves students developing in App Inventor, encourage students to save their work using the **Projects > Checkpoint** option after each section is completed. Checkpoints automatically add a numbered suffix to the project name to represent the version number of the project. This is good practice for two reasons: It teaches students to track versions of their code, and it provides a way for students to revert back to a previous version of their project.
- Glossary terms that are introduced or reinforced within the APB are displayed in **bold**. You may choose to have students define a few of the terms in their PLTW Gateway Notebook or as a means of formative assessment.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

- The [App Inventor Features](#) page includes a list of all the App Inventor terms that are introduced in the unit. You may choose to use that list for a quick class review at the end of a APB or as a means of formative assessment.
- At the end of each APB, remind students to reflect on what they have learned and how they might apply this knowledge or skill to the end-of-unit problem.

General APB Preparation

1. For each APB, determine what you would like your students to do with their completed code. Options include having them export and submit a copy of their App Inventor Application file (.aia), download their blocks as an image (a right-click menu option in the *Blocks Editor*) and submit that to you, write a summary and reflection about their work, or simply explain the code to you. Communicate your plan to your students when facilitating the APB.
2. For each APB, determine how you will present and ask any formative assessment questions/tasks in the curriculum. You may choose which questions/tasks you have students complete, decide which questions you require students to turn in for a grade, and determine which questions/tasks to use as a class discussion. You can present these questions/tasks in a variety of manners, such as exit tickets, graphic organizers, concept maps, discussions, and think-pair-share. For additional information, please refer to the PLTW Formative Assessment Teacher Resource, accessible within your PLTW App Creators Professional Development track.
3. For each APB that uses pair programming, determine how you will have students self-assess and peer-assess their collaboration. You may have them use the collaboration section of the project rubrics to help identify what proficient and advanced collaboration looks like. Determine how you will monitor, facilitate, and assess this collaboration and help students be successful. For additional information, please refer to the Pair Programming Teacher Resource accessible within your PLTW App Creators Professional Development track.
4. For every activity that does not require pair programming, consider what you want collaboration to look like in your classroom and how you will facilitate it. Consider pairing students together and having them use each other as "elbow partners." This means that the students can give each other feedback, ask each other questions, and offer their partner suggestions when needed. Determine how much work you want students to complete individually versus how far you want students to go in using this collaboration. If you choose to use this strategy, consider whether you will randomly assign partners, allow students to select a partner, or you will match up students according to ability.
5. For all projects and the problem or any time you ask students to provide constructive feedback to each other, determine how you want this to work in your classroom. All projects and the problem include a Peer Feedback Form to help students provide specific and constructive feedback to each other. Consider how you can help guide students through this process and share examples of what to do and what not to do when providing peer feedback. Determine whether you will use this form with any other activities throughout the unit.

PLTW Gateway Notebook

For each APB, determine where you would like your students to document their work, answer formative assessment questions, and answer conclusion questions. Which items you collect and assess is up to you. The curriculum assumes students will document all work in their PLTW Gateway Notebooks. However, where and how students document their work is up to you. For example, you may choose to use other documentation tools, such as a digital notebook, a Word document to be shared with you, or an online collaboration service (such as Google Drive). Communicate your plan to your students when facilitating the APB.

Professional Development Opportunities

Professional development opportunities are available to you from the Professional Development page of myPLTW.org. To begin learning, visit myPLTW.org and select the **Professional Development** icon.

myPLTW.org



Community Forums

The Community tool allows users to create posts, ask questions, and share files within their Community groups. As a PLTW teacher, you will automatically be subscribed to Community groups based on the course(s) for which you are certified.

As a Community group member, you have the opportunity to post comments, tag posts, bookmark, and follow other members of your group. The Community environment is provided to facilitate a professional learning community among teachers who teach the same course across the country. Use this forum to ask questions, share best practices and lessons learned, and provide instructional support to your community of professionals.

To access the Community tool, visit myPLTW.org and select the **Community** icon.

myPLTW.org





Lesson 1

Let's Create an App – Quick Guide

Preface

Students are introduced to the concept of pair programming, app development, and the MIT App Inventor development tool. They learn about the Model-View-Controller (MVC) design pattern, app graphical design, event-driven programming, debugging, and algorithm creation using variables and conditional logic. They create engaging biomedical science apps and fun interactive games that apply these concepts and use basic user interface features, media, and animation.

Essential Questions

1. In what ways does technology affect people's lives?
2. How do computer scientists design and develop mobile apps that meet desired needs?
3. What does effective teamwork look like?

Activity, Project, or Problem	Days
Activity 1.1 The App Revolution	2
Activity 1.2 Amazing Algorithms	1
Activity 1.3 The Germ Guide App	2
Activity 1.4 Coding with Conditions	1
Activity 1.5 Decision Time	2
Activity 1.6 Bug Blasters	1
Activity 1.7 Game Time	4
Project 1.8 Build-A-Body	5
Lesson 1 Summative Assessment	1
Total	19 Days



Activity 1.1

The App Revolution

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVES

LO

- LO 1.3A Apply computational thinking to solve problems.
- LO 2.4D Design and develop a mobile app solution.
- LO 2.6A Analyze the implications of computing in society.
- LO 3.4A Abide by professional, legal and ethical standards when using digital resources.
- LO 3.5A Identify the variety of careers related to engineering, biomedical sciences, and/or computer science.

KNOWLEDGE AND SKILLS

KS

- KS 1.3A1 Recognize that computational thinking can be applied in multiple disciplines.
- KS 2.4D1 Describe the components of a mobile app.
- KS 2.4D5 Deploy an app using the emulator or a mobile device.
- KS 2.6A1 Identify ways that computing has changed or is changing society.
- KS 2.6A2 Give examples of technologies that assist or extend human capabilities.
- KS 2.6A3 Describe possible positive and negative social impacts of computing.
- KS 3.4A2 Describe the importance of protecting personal information.
- KS 3.5A2 Explore a variety of careers related to engineering, biomedical sciences, or computer science.

ESSENTIAL QUESTIONS

EQ

- EQ 1 In what ways does technology affect people's lives?
- EQ 1.1 How do computer scientists design and develop mobile apps that meet desired needs?

RESOURCES

-  A1_1_MysteryApp_TR.aia
-  Deploying to the Emulator
-  Activity 1.1 Answer Key



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Preparation

- Have the PLTW Gateway Notebooks ready for distribution.
- Refer to the [Unit Preparation](#) section for details regarding common preparation needs across multiple APBs.

Activity Time: 2 Days

Possible Student Misconceptions

- Computer science equates to computer literacy—the understanding of using basic computer functions.
- Computer scientists are geniuses and not everyone can become one.
- Computer science is just programming or coding.
- Computer scientists or software engineers just make video games.
- Mobile applications are only used for games, social networking, and entertainment, not for solving real problems.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Welcome to App Creators

In the App Creators course, students are introduced to technological advancements and how they have changed our lives. They will learn about computer scientists and their role in the creation of technology, and they will get an idea of what this unit and the activity will be about.

KWL Chart

Students begin the activity thinking about what computer science means to them. Ask students what they picture when they think of a computer scientist and their work. This will help them activate prior knowledge. Also ask students to describe any prior experience they have had with computer science. Allow students time to reflect on their thoughts about computer science and computer scientists.

You may choose to implement the Know-Want to Know-Learned (KWL) chart as a class or individually. This could be done on large chart paper with sticky notes or in students' PLTW Gateway Notebooks. Student responses would be placed under the K and W columns. This will help you identify any misconceptions students have and allow you to determine any prior knowledge and experience students have coming into your classroom.

If you plan to implement a flipped class, students can complete their response to this prompt outside of class.

After students have completed their responses, invite them to share their thoughts as you lead the class in a short discussion. Encourage students to share without correcting possible misconceptions they have about computer science or computer scientists. You may choose to ask them additional questions to get them thinking deeper about computer science.

Document any identified misconceptions and plan to return to these original misconceptions throughout the unit, to allow students to reflect and discuss how their original perceptions might have changed.

Reflect on the Video

Lead a classroom discussion about the Unit Introduction video. Begin asking students clarifying questions about computer science to get them to rethink possible misconceptions identified by the entries in their KWL charts. For example:

- Besides developing computer games, what else might be part of computer science?
- How do you think computer science impacts our daily lives?

Managing Digital Assets

It is important to teach students to organize their digital assets. It is recommended that you have them create a folder for this unit on a school/teacher-preferred storage drive. For example, some schools provide network drives for students to save their files on or recommend an external flash drive. Regardless of what storage device is used, students should organize their files in an obvious folder structure. A good approach is to create a folder for the unit and then a subfolder for each APB as needed.

Explore App Inventor

- Each student will need a Google account to create projects in MIT App Inventor. Please refer to the [Unit Preparation](#) section.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

- As part of managing their projects effectively, it is important for students to learn to name their projects properly so they are easy to identify. In this unit, the file naming convention is to use the APB number, followed by the project descriptive name, followed by the student initials. Spaces and periods are not allowed in most software development tools including App Inventor; therefore, the use of underscores is preferred. For example, A1_1_MyFirstApp_JP.
- It is important to train the students to become independent learners, and part of that is to explore and to read the App Inventor component descriptions. You can share [The MIT App Inventor Library: Documentation & Support](#) page for them to further their knowledge and get help.

Components Pane

- Note that the *Components* list is displayed in a hierarchical structure, also known as a “tree” structure. This structure is used to show how components are embedded within others.
- All visible components, like the buttons and images, appear in the Components pane under the *Screen1* component which is automatically added to the app when it is created. They also appear on the screen in the *Viewer*.
- Components that are not visible, like *Sound* and *TextToSpeech*, appear in the Components pane, but do not appear on the screen in the *Viewer*. Instead, they appear in the *Non-visible components* list at the bottom of the *Viewer*. Although non-visible components are not visible to the user, the programmer uses them to add special features when programming the app.

Properties Pane

- It is important to allow students some time to explore the properties in App Inventor. They will be diving in more deeply in later activities, but this activity is a good primer that allows them to explore the general features of App Inventor.
- Properties are attributes that affect the way a component looks and sometimes the way it behaves. For example, the width and height properties determine what the dimensions will be for the component. If you choose **Fill parent**, the component size will adjust to fill the component that it resides in. For example, a button that resides in a screen will adjust to fill the screen, which is a good option if you want your button to fill the height and width of the screen, regardless of the device size. App Inventor auto-adjusts to fit any device screen size.
- An example of a property that controls how a component behaves is the *Enabled* property. This controls whether the component (such as a button) is enabled for clicking or not. Another example is the property *Scrollable* on a *Screen* component. This determines whether a scroll bar will be available on the side to allow the user to scroll.
- Most of these properties can be modified using code as well, allowing the programmer to control the component's look and behavior during runtime (which is the time when the app is actually running).

Blocks Editor

- Students will be adding some code to explore how code relates to the components.
- There is no expectation that students need to understand the code yet. The goal is that they understand that the *Designer* view is where programmers design the app (add components, lay them out as they want), while the *Blocks* view is where they program these components to actually do something! The blocks in the *Blocks* view determine what will happen when a user clicks buttons, swipes the screen, and tilts or shakes the device.

Discussion: Have students explore the *Built-in drawers* and have them share their observations. You can share the following with the students as part of this discussion.

As described in the *Blocks Editor* screenshot, the *Built-in drawers* contain general blocks that can be used to define how the app behaves. They are not tied to a specific component. These *Built-in blocks* are available regardless of which components are in your project. Examples of such blocks are:

- **Control blocks** Used for making decisions in a program, to pause the execution of a program, and to repeat certain blocks, as well as any other type of block that affects the flow of execution of the program.
- **Logic blocks** Used to execute logical operators, like checking whether a condition is true or false, or combining multiple conditions with “and” and “or” logical operators.
- **Math blocks** Used for executing mathematical operations like +, -, *, /, picking a random number, and comparing two numbers.
- **Colors blocks** Used to specify a component's color.

Other types of *Built-in Blocks* are not mentioned here. Students will learn more about these and what they do later in this unit.

Component-specific Drawer



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Discussion: Have students explore the *Component-specific* drawers and have them share their observations. As part of this discussion, you can share the following with the students.

Unlike the *Built-in drawers*, *Component-specific drawers* are directly tied to the component selected. The blocks in these drawers provide ways to access and update the properties of the specific component and execute functionality on the component. For example:

- A *Button* will have blocks like *When Button.Clicked* to detect when the button is clicked.
- A *CheckBox* will have blocks like *When CheckBox.Changed* to detect when the checkbox is clicked and its value changes.
- A *SoundPlayer* will have blocks like *Play* and *Pause* to play and pause an audio component.

Explore the Mystery App

The Mystery App shows the students another way to create a project—this time by importing the source code of an existing App Inventor project (.aia file).

- Walk students through the steps to download the source file and save it in their App Creators folder (preferably in a subfolder specific to this activity).
- Some students might try to double-click on the file after it's downloaded, which won't do anything, since it is an App Inventor Application (.aia) source file and will only run when imported into the App Inventor development environment as a project.
- Walk students through the steps to browse and select the project file they have just downloaded.

Hints

- The import process might take time. If it times out or fails, go back to the My Projects screen and check whether the project was actually created. If not, try again.
- Because App Inventor auto-saves projects, if students accidentally modify the app, they will need to delete the project and re-import it to continue with this activity.

Deploying to a Physical Device

The preferred and recommended way to deploy and test apps in this unit is with a physical device. If you do not have enough devices, students can share and take turns using them. Another backup option is the App Inventor Emulator. For more information, refer to [Deploying to the Emulator](#).

Many activities in this unit direct students to deploy their apps to their mobile devices. Help facilitate student sharing of the devices. While one student deploys their app to the device, the other student can deploy their app to the emulator.

- It is highly recommended that students deploy their apps to their device using Wi-Fi versus using a USB cable. USB cables should only be used if Wi-Fi is not available. Note that for live testing to work, both your computer and your device must be connected to the same Wi-Fi network.
- Refer to the [App Inventor Troubleshooting Guide](#) for additional help with troubleshooting. Additionally, App Inventor has a lot of [troubleshooting information](#) that students can refer to when deploying their apps to their devices.
- If there are issues, the easiest first thing to try is to have students click **Reset Connection** from the Connect menu and then click **AI Companion** to restart the connection.

Reminders

- Encourage exploration, observation, and analysis whenever possible.
- Comments are very important to use and a good habit to start as early as their first app in this unit. It is one of the ways computer scientists document their work.
- Encourage students to become independent learners by reading through support pages and help forums online. The App Inventor Help options available on code blocks and *Palette* components provide helpful information.
- Please refer to the note about the Gallery content found in the unit level [Safety Considerations](#).
- Have students bookmark the links for App Inventor, documentation, troubleshooting, and any page they find useful to them. Help them organize their bookmarks into a new App Inventor folder of bookmarks.



The App Creators Unit Problem

Encourage students to think creatively of ideas that they are interested in. If they struggle in coming up with ideas, the following themes might help them get started:

1 Problem category: Personal interest

Design and develop an app that addresses a personal interest of yours.

☐ [Questions to consider](#)

2 Problem category: Health and wellness

Chronic disease, emerging infectious disease, physical inactivity, poor nutrition, and lack of access to proper healthcare are all major problems in the United States.

☐ [Questions to consider](#)

3 Problem category: Environment

Natural resource depletion, climate change, environmental degradation, and pollution are environmental issues that are of great concern to our communities.

☐ [Questions to consider](#)

4 Problem category: Emergency preparedness

Natural disasters, disease outbreak, and other public emergencies affect millions of Americans each year. Preparing for an emergency incident plays a vital role in ensuring that people have the necessary supplies and equipment, know where to go, and know how to keep themselves safe when an emergency occurs.

☐ [Questions to consider](#)

5 Problem category: Community service

A community's members are the community's richest resource. They provide knowledge about their community and dedicate energy and time to helping their community thrive. Unfortunately, many communities have a hard time getting people involved.

☐ [Questions to consider](#)

6 Problem category: Education

In many communities in the United States, students are falling behind in math and reading. Often, there is a large achievement gap between students from advantaged versus disadvantaged backgrounds.

☐ [Questions to consider](#)

7 Problem category: School culture

Students today often feel disconnected from their school environment, have been bullied in the past year, have low self-esteem, or generally feel left out.

☐ [Questions to consider](#)



Activity 1.2

Amazing Algorithms

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVES

LO

LO 2.2A Analyze algorithms that accomplish a task.

KNOWLEDGE AND SKILLS

KS

KS 2.2A1 Create an algorithm or sequence of steps to accomplish a task.

KS 2.2A2 Identify different algorithms that can be used to solve the same problem.

KS 2.2A3 Analyze, break down, and explain the logic of an algorithm.

ESSENTIAL QUESTION

EQ

EQ 1.3 What does effective teamwork look like?

RESOURCES

 1.2 Map Print

 1.2 Cutouts Print

 Activity 1.2 Answer Key



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Preparation

If you did not purchase the PLTW App Creators kits, print and cut out the [1.2 Map Print](#) and bus [1.2 Cutouts Print](#). Students will need the obstacle cutouts for Activity 1.4 Coding with Conditions.

Safety Considerations

Make sure students do not hide objects in air vents, in or near emergency equipment (eye wash stations, fire equipment), behind heavy furniture, in places where confidential documents are held, in places that will require ladders or standing on desks to access, behind fragile objects, in animal cages, or in any spaces that that would prevent students from recovering the objects.

Student Accommodations

Students with mobility and visual disabilities may need assistance with hiding and finding their object.

Activity Time: 1 Day

Possible Student Misconceptions

- Computers can think or perform actions independently.
- There is only one algorithm to solve a problem.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Introduction to Algorithms

Students will be introduced to the process of algorithmic thinking. Start by asking students what they know about algorithms.

You may choose to provide students with more introduction than what the APB provides. You may choose to lead the students in an activity where you give directions to draw an object or complete a task. You can choose the amount of directions and the depth. Students can compare their drawing or task outcome to what was intended for them to accomplish. They can discuss any issues they encountered for the activity.

Algorithmic Thinking

Discussions about the Video:

- Option for discussion before the video:
 - What do students think algorithmic thinking involves?
- Options for discussion after the video:
 - The various algorithms they use in their daily lives that are not listed in the text or discussed in the video.
 - How computer programs and apps use algorithms to carry out their functions.

Without correcting possible misconceptions they have about algorithms, encourage students to share their thoughts.

You may ask them additional probing questions to get them to think deeper and to expand on their current understanding. Students will explore how to organize these thoughts throughout the lesson and later reflect on what they've learned compared to their initial thoughts.

- After the video, lead students into a discussion about the process of creating algorithms. Review the steps for creating a volcano for a science project, as shown in the flowchart.

Figure 1. Volcano Flowchart

- For extra practice, choose a task that students can relate to, such as making lunch or getting ready for sports practice.
 - Use a numbered list to list the steps of the task. Discuss the importance of creating steps that are clear and accurate.
 - After listing the steps, use the symbols referenced in the chart to create a linear flowchart as a class.

Hide and Seek



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

- Students will step into the role of a computer scientist to write an algorithm that will locate an object that they have hidden in the classroom. Allow each student time to hide an object in a place that can be found by another student safely. You may ask the other students to close their eyes and put their heads down while the student is hiding the object.
- After they have written and reviewed their algorithms, they will exchange them with other students, and then the search is on for the hidden object. You may choose to divide the students into small groups or let all the students go at the same time to find their objects. The students must use only the provided written instructions, and if they cannot find the objects, they should return the algorithms to the owners for revision.
- In case you have concerns about Hide and Seek not being suitable for your classroom setup, here are some alternatives that you can consider using:
 - Ask the students to draw a simple drawing and write the step-by-step algorithm for how to replicate that drawing on a piece of paper. Students swap algorithms and follow the instructions, then compare the drawings they had originally made to the ones their partners made. What went wrong? How can the algorithm be improved?
 - If you have access to building blocks, preferably colored ones, you can have students sit in pairs, facing away from one another, and have one student build a structure with the blocks and give instructions to the other on how to replicate that structure. Student can then compare the structures with the ones their partners made. What went wrong? How can the algorithm be improved?

Navigation

- For this part of the activity, students will work in pairs to write an algorithm that will navigate a school bus to a museum. You may choose to provide the small scale version of the map for the students to paste or tape into their PLTW Gateway Notebook.
- Students will find there are many routes for the bus to take to reach its destination. Students will learn that there are many algorithms they can use to accomplish the same task. After completing the activity, you can lead a discussion on the possible misconception that there is only one algorithm for a task.
- Make sure that students understand that a turnLeft navigation means that the bus turns 90 degrees to the left, in its current position without moving forward or backward. The turnRight navigation functions in the same manner.
- After they write and review their algorithm, students will swap with another team for them to test it. If the bus does not reach its destination, the writers should revise their algorithm and try again.

Note: Students should save their algorithms. They will use it again in Activity 2.1 Keep Me In the Loop.



Activity 1.3

The Germ Guide App

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVES

LO

- LO 1.2B Apply user-centered design principles when creating a solution.
- LO 2.3A Develop and apply abstractions.
- LO 2.4A Analyze the structure and functionality of a program.
- LO 2.4B Create programs by developing and testing code in a modular, incremental approach.
- LO 2.4C Adapt or improve existing code.
- LO 2.5A Identify methods in which electronic devices communicate with each other.
- LO 3.2A Communicate effectively for specific purposes and settings.
- LO 3.3A Demonstrate the ability to manage multiple resources throughout a project.

KNOWLEDGE AND SKILLS

KS

- KS 1.2B2 Describe the importance of involving prospective users early and often during the design process.
- KS 1.3A1 Recognize that computational thinking can be applied in multiple disciplines.
- KS 2.3A1 Explain the purpose of an abstraction.
- KS 2.4A1 Identify and describe the high-level structures of a program, such as user interface components, data components, event handlers, and procedures.
- KS 2.4A2 Describe the appropriate code blocks and conventions used in the programming language.
- KS 2.4A3 Identify possible events that can occur during runtime and select the appropriate event handler blocks to respond to these events.
- KS 2.4B2 Design and develop a program by breaking a large plan into smaller modules using procedures and event handlers (if applicable).
- KS 2.4B3 Test code frequently as it is being developed.
- KS 2.4C2 Improve readability of code by creating or improving documentation, using descriptive variable and procedures names, and using comments.
- KS 2.4D1 Describe the components of a mobile app.
- KS 2.4D2 Navigate and use an app development software to design, develop, and test a mobile app.
- KS 2.4D3 Design an app's user interface using appropriate tools.
- KS 2.4D4 Select and use appropriate features to meet the desired functionality of an app, such as user interface components, layout arrangements, media objects, drawing and animation, sensors, location awareness, accessing phone features, and data storage.
- KS 2.4D5 Deploy an app using the emulator or a mobile device.
- KS 2.5A1 Recognize various ways of transferring data from one electronic device to another, such as USB connection, Wi-Fi, or Bluetooth.
- KS 2.5A2 Use a systematic process to identify the source of a problem within individual and connected devices.
- KS 2.6A2 Give examples of technologies that assist or extend human capabilities.
- KS 3.2A2 Use accurate and appropriate terminology.
- KS 3.3A2 Manage digital files appropriately.
- KS 3.5A2 Explore a variety of careers related to engineering, biomedical sciences, or computer science.

ESSENTIAL QUESTIONS

EQ

- EQ 1 In what ways does technology affect people's lives?
- EQ 1.1 How do computer scientists design and develop mobile apps that meet desired needs?

RESOURCES

-  A1_3_GermGuide_TR.aia
-  Activity 1.3 Answer Key



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Preparation

- Determine how students will connect their tablets to the AI2 software—through Wi-Fi or USB cables.
 - For live testing to work over Wi-Fi, both the computer and the device must be connected to the same Wi-Fi network, so check to make sure they are connected to the same network.
 - Using Wi-Fi is the easiest way to connect and is highly recommended, but you may also have students connect using a USB cable. If you choose to have students connect using a USB cable, make sure to have these available for students.
- Consider how you will facilitate the distribution and sharing of mobile devices for student app testing.
- Review the source code file. It contains the completed code for this activity and contains comments explaining each event handler.
- Review the [App Inventor Troubleshooting Guide](#) to prepare for any connection issues that might occur with students.

Activity Time: 2 Days

Possible Student Misconceptions

- All computer programs are linear and are a set series of instructions performed in a predetermined order.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Why an App?

In this activity, students will be introduced to the basic app architecture and explore the basic principles of app development. They will learn about event-driven programming and begin to build their first app in MIT App Inventor.

Although students will work individually through this activity and Activity 1.5 Decision Time, encourage students to collaborate with each other as they work. For example, students should work with a partner if they have questions or need help. Create a general classroom policy that students should ask a peer for help before they ask you for help. Students should ask their partners to check their work and provide feedback. Consider asking more advanced students to help those who are new to computer science.

Discussion: Lead a discussion where students share what they know about app development and how apps work. Document their ideas. You can revisit these initial ideas as they work through the unit and see how their ideas change as they learn more.

App Design and Development

Background Information: M-V-C Pattern

[Show](#)

Consider having students view the App Design and Development video as a class. You can pause the video throughout so that they can take notes.

Discussion: After students have viewed the video, lead a class discussion about what they learned. Make sure students are comfortable with the new terminology presented in the video, such as user interface (UI) and end user, and make sure students understand the different layers of an app's architecture.

Reiterate the importance of working closely with the client and end users when developing an app.

Design Time!

Instruct students to review the design requirements and design plans for the app individually.

Make sure students understand the design requirements and design sketch. Display the design sketch for all students to see and work with the class to trace what will happen when each button is pressed.



Background Information: Diseases

[Show](#)

Events

Event-driven Programming

Event-driven programming is different from traditional software programming. In traditional software programming, the program is a series of instructions that are performed in a predetermined order, similar to a recipe. In mobile app development, event-driven programming is used, which means that the flow and order of the instructions is determined by events. This means that all of the app's activity occurs in response to an event.

The idea of event-driven programming may be a new concept even to those students who have prior computer science experience. A common student misconception is that all computer programs are linear and are a set series of instructions performed in a predetermined order. Make sure to reinforce this idea of event-driven programming with students, as it is fundamental to how apps are programmed.

Make sure students understand that event-driven programming is all about cause and effect. Use some real life examples of events and their triggered responses to help students understand this new terminology. For example, when you touch a hot stove, you pull your hand away. The event was touching a hot stove, the response was pulling your hand away.

- Have students discuss apps that they've used and share how they interact with the app. As they discuss, they should identify the events that occur during runtime of the apps.
- Have students create a simple graphic organizer, similar to the one shown below:

Term	Definition	Example
Event-driven programming		

- Direct students to create a definition for event-driven programming in their own words and describe an example in their graphic organizers.

Event Handlers

- Ask the students to describe one event handler, using a flowchart, for the Germ Guide App. Remind students that they will use the format learned in [Activity 1.1 Amazing Algorithms](#) for creating flowcharts.
 - The Begin step will be represented with an oval.
 - The commands will be represented with rectangles.
 - The Begin oval will represent the event (because the event triggers the algorithm to run).

Students should refer to the design requirements and plan for the events that will occur in the Germ Guide App. Example event handler flowcharts might say:

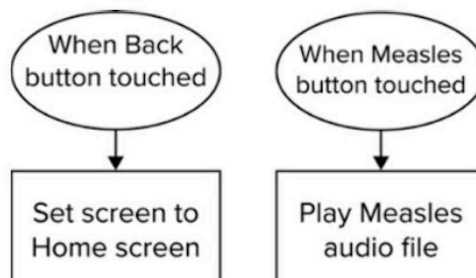


Figure 1. Germ Guide App Event Handler Flowcharts

These statements will eventually translate to the following event handlers in App Inventor:

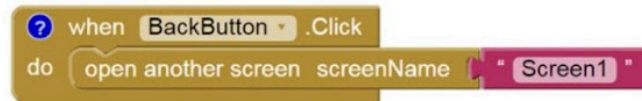


Figure 2. BackButton Event Handler

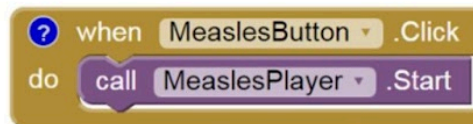
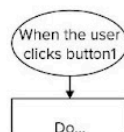


Figure 3. MeaslesButton Event Handler

Explore some of the different event types that can trigger responses in App Inventor.



Begin Building

In this section, students will program the app to play audio clips when the disease buttons are touched. In Activity 1.5 Decision Time, students will add code so that the audio files don't play over each other during runtime and will add conditional statements so that the audio can be started and paused.

- Students should download the source code for the starter app. The source code contains some already programmed screens and all of the media images and audio files that are necessary to build the app.
- To deploy the app to a device:
 - In App Inventor, click **Connect** and select **AI Companion**. This will cause a QR code to appear.
 - Scan the QR code with your device. This will deploy the app to your device. Note that for live testing to work, both your computer and your device must be connected to the same Wi-Fi network. Using Wi-Fi is the easiest way to connect, but you may also connect using a USB cable.

Reminders

- Remind students to save their file to the appropriate folder, using the organization structure they determined in the previous activity.
- Walk around the room and help students navigate around in App Inventor.
- Remind students that the event handler blocks in the *Blocks Editor Viewer* are each the controller layer behind a button on the Home screen.
 - Students will see the *CrowdingButton.clicked*, the *VectorButton.clicked*, and the *WaterButton.clicked* event handlers in the *Blocks Editor Viewer*.
- Refer to the [Activity 1.1 App Revolution](#) to help students navigate within the App Inventor environment and for more information on camelcase, if needed.

Background Information: Abstraction

[Show](#)

To make sure students have a handle on the concept of abstraction, share some common every day examples of abstractions with students and then ask them to describe an example of abstraction in their daily lives.

Students will further explore the idea of abstraction in Lesson 2, when they create procedures.

Optional Extension



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

A new App Inventor source code file with all of the screens prebuilt, will be available for students to use in the Decision Time activity, but if there is additional time during this activity or you have advanced students who are ahead, you may have students build the UI for the remaining screens.

The Vectors screen and Water screen should look the same as the Crowding screen. For example, the Vector screen should look like:

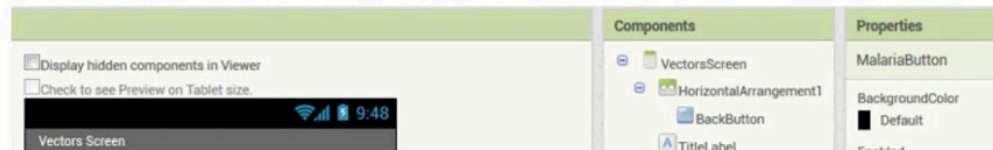


Figure 4. Germ Guide App - Vector-borne Diseases UI

Use the *Properties* panes for the components of the Water screen, shown below, to guide students if they need additional help.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Properties	Properties	Properties	Properties	Properties	Properties
	VerticalArrangement1	MeaslesButton	MeaslesButton	MeaslesPlayer	MeaslesPlayer

Figure 5. Germ Guide App - Properties Panels

- Have students test their updated apps on their devices to see whether the UI on all of the screens looks okay.
 - They can touch each of the buttons on *Screen1* to navigate to the other screens.
 - They can use the Back button to navigate back to *Screen1* from any of the other screens.



Activity 1.4

Coding with Conditions

Teacher Resources

[View Student Course](#)

KNOWLEDGE AND SKILLS

KS

- KS 1.3A1 Recognize that computational thinking can be applied in multiple disciplines.
- KS 2.2A1 Create an algorithm or sequence of steps to accomplish a task.
- KS 2.2A2 Identify different algorithms that can be used to solve the same problem.
- KS 2.2A5 Create simple algorithms that involve variables, conditionals, operators, or logic.

ESSENTIAL QUESTION

EQ

- EQ 1.3 What does effective teamwork look like?

RESOURCES

-  1.2 Cutouts Print
-  Activity 1.4 Answer Key



Preparation

If you did not purchase the PLTW App Creators kits, cut out the traffic signs for the maze (you may reuse the blank maze and bus cutouts from the Amazing Algorithms activity). Consider laminating the mazes, bus cutouts, and traffic sign cutouts to be reused. Alternatively, you may choose to have students draw traffic signs on their mazes instead of using cutouts, but you will have to instruct students to do this instead of placing the signs on their maps.

Activity Time: 1 Day

Possible Student Misconceptions

- Algorithms are only linear, meaning they only include commands to be followed sequentially.
- Computers don't need input to make decisions.
- There is only one possible algorithm to solve a problem.
- All programming is done using software.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Conditional Statements

In this activity, students will continue to practice thinking algorithmically. They will build upon the knowledge they learned in Activity 1.1 Amazing Algorithms, and they will learn how to use conditional statements to make decisions within an algorithm.

Discussion: Ask students to think about the algorithms they created in the Amazing Algorithms activity. Remind them that these algorithms were all linear, meaning that they needed to be followed in a simple step-by-step sequence. Ask students:

- What would happen to the algorithm if a decision had to be made?
- How would that change the algorithm?
- Would the algorithm still be followed in a linear sequence?

Students should start to realize that if an algorithm includes a decision, it will no longer be a single linear progression, and instead the algorithm will have multiple paths (also called branches). This means that if the answer to the question is true, there will be one path, and if the answer to the question is false, there will be a different path.

Algorithms with Conditional Logic

- Students are asked to think of Yes/No questions that they use in their everyday life. Have students share their questions with a partner and then open it up to the class. Make sure that all questions posed are Yes/No questions and discuss how the answers to these questions would result in different actions. Make sure students see that these questions and their subsequent actions are examples of conditional statements. Example conditional statements are: "Is it raining?" and "Is it cold out?"
- Direct students to take notes as they watch the video. You may choose to play the video for the class or have them view it individually.

Discussion: Lead the students into a discussion on the process of creating algorithms with conditions. Explain the role of the rhombus in the flowchart.



Suggestion for Struggling Students: Guide students to create a simple algorithm, in a flowchart format, for packing their backpack for school. Have them include a conditional statement for determining whether or not they need to pack their gym clothes for that day. An example flowchart for this algorithm is shown below. Alternatively, you may show students this algorithm, and have them dissect the logic of the conditional statement.

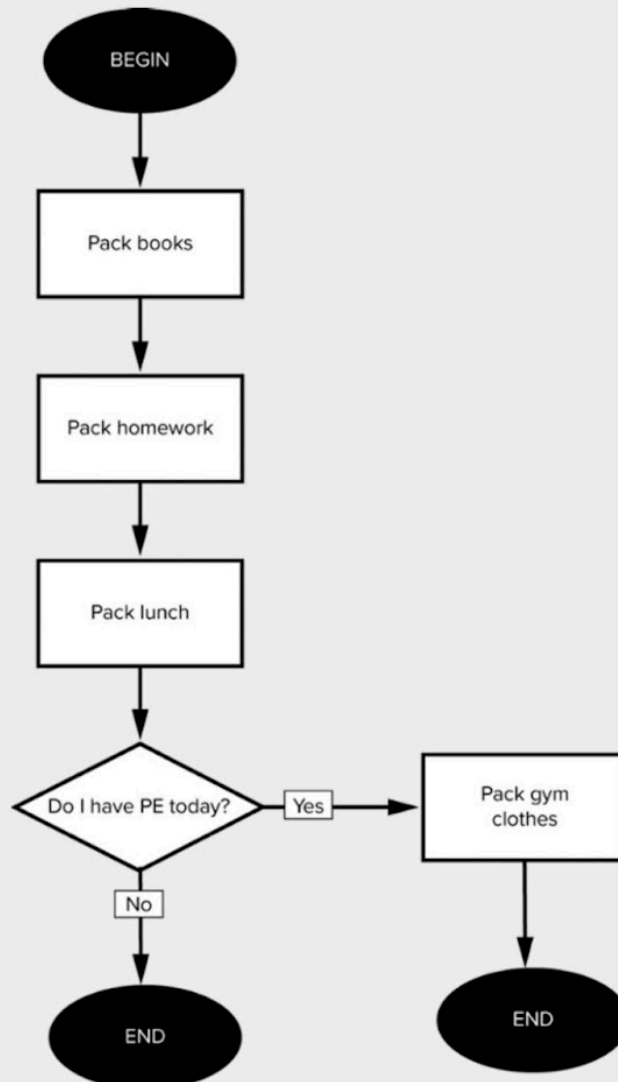


Figure 1. Backpack Flowchart

Field Trip Navigation

- For this part of the activity, students will work individually to write an algorithm with conditional statements that will autonomously navigate a school bus from the school to the museum. The algorithms that students write will have to account for any obstacle, such as Stop Signs and No Right Turn or No Left Turn signs that may be located randomly along the route.
- You may choose to provide small-scale versions of the maps for the students to paste or tape into their PLTW Gateway Notebooks.
- Direct students to think about the navigation commands available to them. Point out to the class that the moveForward command will move the bus forward until it reaches the end of the street or there is a traffic sign. (This is different from the moveForward command in the Amazing Algorithms activity.)
- Have students discuss the possible questions they might want to use for their algorithm. Some examples include:



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

- Can I turn left?
- Can I turn right?
- Is there a stop sign?
- Have I reached the museum?

Note that the question "Have I reached the museum?" requires the students to incorporate a step that has them repeat all of the steps of their algorithm until the bus has reached the destination. The concept of "repeating until" is called a *loop*. In programming, loops are used to allow for repetition within an algorithm.

Students will further explore loops and the idea of repetition in code in Lesson 2. The idea of repetition is included in this activity to plant the seed and lay the foundation for student learning of this concept.

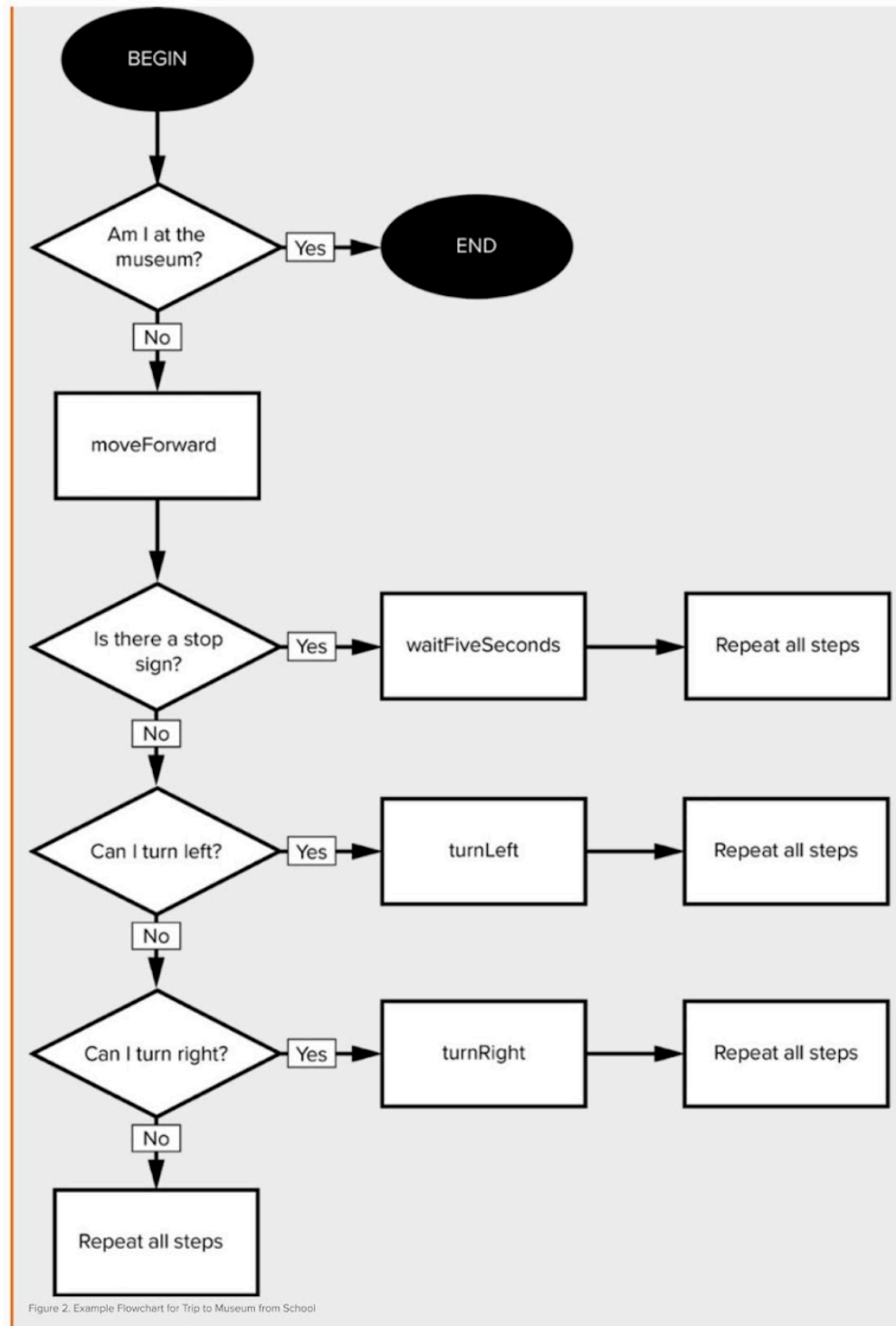
Universal Test

- After students write and test their algorithms on the example maze, if time permits, they will test it on a blank maze where they have placed at least three traffic signs in different locations. If the bus does not reach its destination, the students should revise their algorithm and try again.
- Some students will not finish working on their algorithms and will not be able to successfully create an algorithm that works with any map. This is okay. The purpose of this task is for students to become comfortable writing conditional statements. If students are able to successfully write a conditional statement in their algorithm, then they are meeting the fundamental learning goal of this activity.
- Before the end of class, stop students from working, even if they are not finished, and share an example of a successful algorithm that works universally. Students should see from the example that programming all of the conditional statements for each situation that may be encountered allows the autonomous bus to make the necessary decisions to navigate to the museum.



Example Algorithm

1. Begin
2. Have I reached the museum?
 If Yes, then End
 If No, go to the next step
3. moveForward
4. Is there a stop sign?
 If Yes, waitFiveSeconds then go back Step 2
 If No, go to the next step
5. Can I turn left?
 If Yes, turnLeft then go back Step 2
 If No, go to the next step
6. Can I turn right?
 If Yes, turnRight then go back Step 2
 If No, go to the next step





SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Discussion: Have students discuss their answers to the reflection questions. Listen to student discussions and ask clarifying questions as needed, to gauge student understanding.

Have students discuss the similarities and differences in their algorithms and their strategy for navigating the school bus. This is a great way to reinforce the idea that different algorithms can achieve the same task.



Activity 1.5

Decision Time

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVES

LO

- LO 2.4B Create programs by developing and testing code in a modular, incremental approach.
- LO 3.2B Document a process according to professional standards.

KNOWLEDGE AND SKILLS

KS

- KS 1.3A1 Recognize that computational thinking can be applied in multiple disciplines.
- KS 2.2A3 Analyze, break down, and explain the logic of an algorithm.
- KS 2.2A5 Create simple algorithms that involve variables, conditionals, operators, or logic.
- KS 2.4A2 Describe the appropriate code blocks and convention used in the programming language.
- KS 2.4B1 Plan a program using appropriate strategies such as natural language or flowcharting.
- KS 2.4B3 Test code frequently as it is being developed.
- KS 2.4D2 Navigate and use an app development software to design, develop and test a mobile app.
- KS 2.4D4 Select and use appropriate features to meet the desired functionality of an app, such as user interface components, layout arrangements, media objects, drawing and animation, sensors, location awareness, accessing phone features, and data storage.
- KS 2.4D5 Deploy an app using the emulator or a mobile device.
- KS 3.2B1 Present data and information through a variety of accepted means such as: graphs, charts, images, video, schematics, code, 3D models, or simulations.
- KS 3.3A2 Manage digital files appropriately.

ESSENTIAL QUESTION

EQ

- EQ 1.1 How do computer scientists design and develop mobile apps that meet desired needs?

RESOURCES

-  A1_5_GermGuide_TR.aia
-  Activity 1.5 Answer Key



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Activity Time: 2 Days

Possible Student Misconceptions

- Algorithms are all linear, meaning they only follow one set, sequential path.
- Computers don't need input to make decisions.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Using Conditionals

In this activity, students will dive back into using App Inventor to finish their Germ Guide App. They will use what they learned in the Activity 1.4 Coding with Conditions about conditional statements and apply this logic to their apps.

Although students will continue to work individually through this activity and finish their Germ Guide App, students should work with a partner if they have questions or need help. Remind students that they should ask a peer for help before they ask for your help. Students should ask their partner to check their work and provide feedback. Consider asking more advanced students to help those who are new to computer science.

Opening your App

- For students who completed the UIs for all screens on their own, you can direct them to open **Projects > My projects** and select the **GermGuide** app that they started in The Germ Guide activity.
- The rest of the students will need to download the new source code that contains these screens already completed for them, so that all students are starting this activity at the same point.
- To deploy the app to a device:
 - In App Inventor, click **Connect** and select **AI Companion**. This will cause a QR code to appear.
 - Scan the QR code with your device. This will deploy the app to your device. Note that for live testing to work, both your computer and your device must be connected to the same Wi-Fi network. Using Wi-Fi is the easiest way to connect, but you may also connect using a USB cable.

Making Decisions

Remind students about what they learned in Activity 1.4 Coding with Conditions. Conditional statements check whether a condition is true, and then perform one action if it is true, and perform a different action if it's false.



Suggestion for Struggling Students: If students are still struggling with conditional behavior, have them practice writing conditional if-else statements.

For example, have students create a sample conditional statement for a game that continues playing until the score reaches 50, and when the score reaches 50, it displays "winner" on the screen.

Display or draw the following blank flowchart for students.

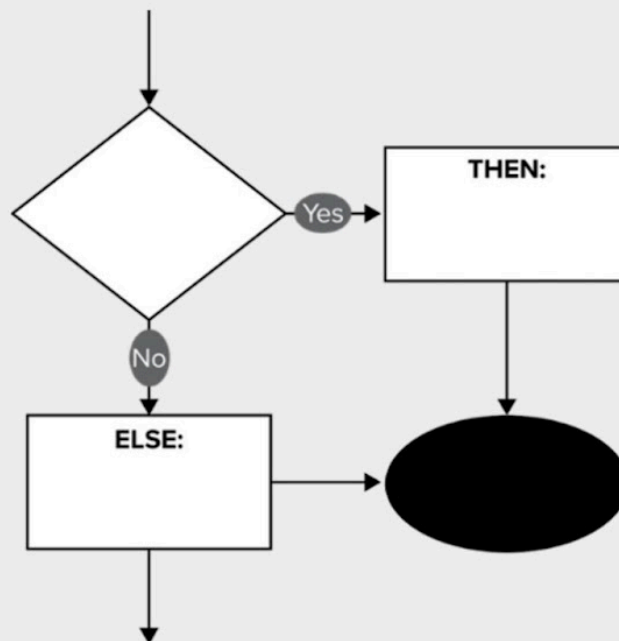


Figure 1. Conditional Statements Template

Discuss how the flowchart can be filled in for the game. Use the filled-in flowchart below as a guide.

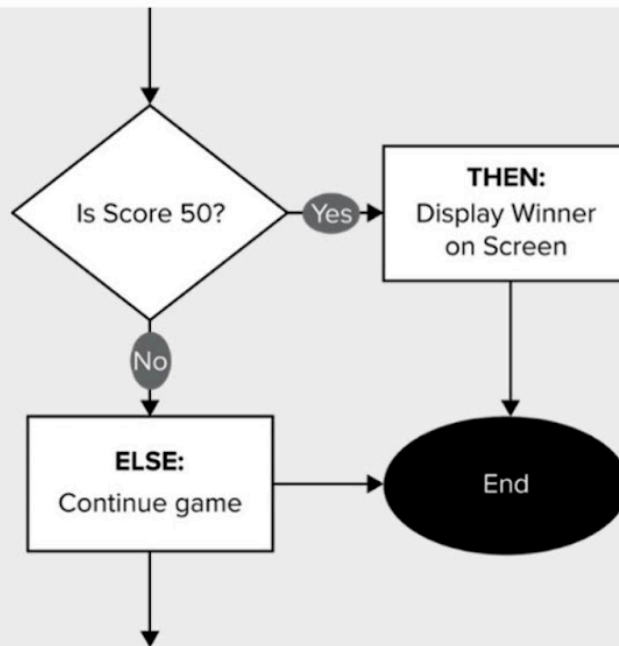


Figure 2. Example Conditional Flowchart

Examples: The following are example *if* and *if else* blocks that you can share with students to give them a point of reference. Note that they correspond with the algorithm shown in the flowchart above.

Example *if* block:

- Is score 50?
 - If yes, then display "Winner"
 - If no, do nothing

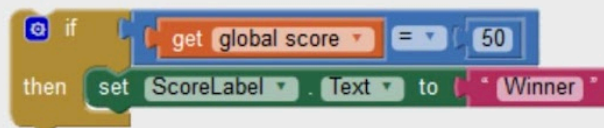


Figure 3. Example If-then Block

Example *if-else* block:

- Is score 50?
 - If yes, then display "Winner"
 - If no, continue game



Figure 4. Example If-then-else Block



Code This Behavior in App Inventor

- Remind students that any time they are coding, they will do the coding in the *Blocks Editor* in App Inventor. They will then use either the emulator or the Companion app on a physical device to test the app.
- Walk around the room checking students' work as they complete the code for this behavior.
- Once students have completed their code, walk students through the conditional statement in the flowchart they created and relate it to each block of code they added to their event handler.
- Have students identify that the *if MeaslesPlayer.isPlaying* block corresponds with the rhombus from the flowchart that asks the question, "Is the MeaslesPlayer playing?"

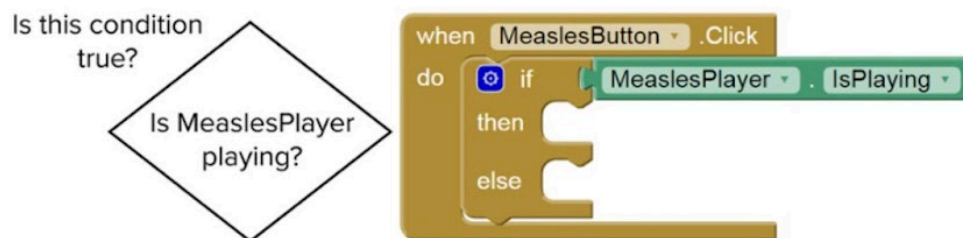


Figure 5. MeaslesPlayer Code

- Make sure students connect the remaining portions of the algorithm to the sections of the flowchart:



Figure 6. Pause the MeaslesPlayer

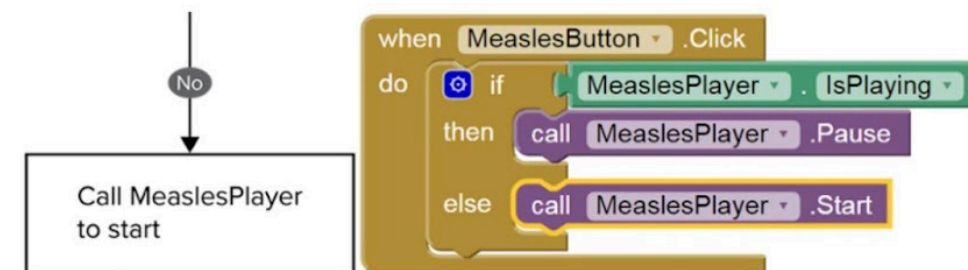


Figure 7. Start the MeaslesPlayer

Completing the Pause Behavior

Check student work for this section to see whether they were able to program their Meningitis button to pause the audio when clicked. If they were not, determine where they stumbled and have them work backward to figure out how they can fix their error(s).

Completing the Audio

- Allow for students to share their ideas. Consider grouping students together in small groups to brainstorm a solution.
- If students need help, direct them to explore the different blocks available in the *MeningitisPlayer* drawer and hover over these blocks to get more information on their function. Encourage students to keep trying and test the code when they add it, until they can get it to work.
- You may end the activity after this section and direct students to complete the end-of-activity questions. If you do this, make sure to show students how to use the *Backpack* feature as a copy-and-paste function within App Inventor.

Optional Extension: Using the Backpack



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

- See [additional information](#) on the App Inventor website about the *Backpack* feature.
- Depending on time and ability level, students may play around with the completed app and make changes to the UI and add additional functionality of their choosing.



Activity 1.6

Bug Blasters

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVES

LO

- LO 1.1A** Describe and analyze moments within a problem-solving process where persistence, iteration, and the positive aspect of failure played an important role in gaining understanding about a problem or unexpected observation.
- LO 1.3A** Apply computational thinking to solve problems.
- LO 2.4B** Create programs by developing and testing code in a modular, incremental approach.
- LO 3.1A** Collaborate effectively on a diverse team.
- LO 3.2A** Communicate effectively for specific purposes and settings.

KNOWLEDGE AND SKILLS

KS

- KS 2.2A4** Locate and debug errors within an algorithm.
- KS 2.4B4** Debug programs using a variety of strategies to isolate and identify problems, including analyzing error messages, analyzing variable values line by line, generating output, or commenting out code.
- KS 3.1A1** Describe how diverse perspectives in collaboration typically produce the best results in a process.
- KS 3.1A2** Apply team norms to encourage productivity and define how a team will function and measure its success.
- KS 3.2A1** Communicate to meet the needs of the audience and be appropriate to the situation.

ESSENTIAL QUESTIONS

EQ

- EQ 1.1** How do computer scientists design and develop mobile apps that meet desired needs?
- EQ 1.2** What does effective teamwork look like?

RESOURCES

-  [A1_6_StudyBuddy_FixMe_TR.aia](#)
-  [Debug Steps Reference](#)
-  [Pair Programming Poster](#)
-  [1.6 Study Buddy App Description](#)
-  [Activity 1.6 Answer Key](#)



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Preparation

- You can choose to print the [Debugging Steps Reference](#) and have students attach it in their PLTW Gateway Notebooks for future reference.
- Students will use pair programming when working on the second part of the activity (Blast the Bug) to apply debugging strategies they learned in this activity to debug the Study Buddy app. Prepare a list of partners to assign for the activity. For more information on facilitating pair programming in your classroom and what best practices are for effectively facilitating it, refer to the notes and the pair programming video for teachers.
- You may choose to print the [Pair Programming graphic](#) and hang it on your classroom wall for reference.

Activity Time: 1 Day

Possible Student Misconceptions

- Programs always work the first time.
- Computer scientists work alone all the time.
- Programming is much better when done independently.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Programming with Persistence

This activity stresses the fact that programming requires persistence and iteration—it is expected that students will find bugs and debug their programs over and over again! It is important to stress these learning objectives. A common misconception students have about computer science is that good programmers are able to get their programs to work on their first try. Reinforce that this is not true, and that having the willingness to keep trying to figure out what went wrong is what makes a good programmer.

You can mention that computer scientists, in fact, spend a lot of time going through multiple stages of testing their programs:

1. They build and test their code incrementally to isolate bugs early on.
2. Every time they complete a feature of the app, they test that feature in isolation (known as unit testing). This falls in line with the idea of using an incremental approach to programming, which is discussed in detail in this activity.
3. Then they test the rest of the app to make sure the new feature did not somehow affect the app as a whole (known as regression testing).
4. Once they complete their app, they have the client/user test it to make sure it meets the client's needs (known as user acceptance testing).

Steps for Debugging Code

In this activity, students will learn about each of the [debugging steps](#) except the Code trace/Variable trace, which will be introduced in the next activity.

Explore Debugging Features

Reminder: Always reinforce an incremental approach to developing code. Every time students add a couple of blocks, have them test their work. This allows students to isolate bugs more easily and in less time.

Right-click Menu

Students will explore the following commands in the right-click menu:

- Add Comment
- Do It
- Collapse/Expand Block
- Disable/Enable Block

Students are asked to make observations or answer questions. You can have students answer the questions in the activity, in their PLTW Gateway Notebooks, or they can pause and think about these questions as they work through the steps.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Note: If you get the opportunity to explain the other commands to the students (or they have time to explore), here is a quick reminder of what the commands do:

- **Duplicate** makes another copy of the highlighted block.
- **Inline/External Inputs** are advanced commands, out of scope for this unit.
- **Add to Backpack** adds the highlighted block to the backpack for later reuse. Anything that is in the backpack can be accessed by clicking on the backpack icon on the top right of the *Blocks Viewer*, and can be dragged/dropped into the program. Great for reusing code.
- **Delete** removes the highlighted block and all its content.
- **Help** takes you to the MIT App Inventor blocks support page.

Add Comment

Commenting code is a useful habit that students should be encouraged to always do whenever programming an app. It is especially helpful when the code becomes more complicated.

Students should know how to add comments by now, having used them in previous activities.

Do It

In addition to the message on the join block, other examples of blocks that return a result are mathematical and logical operators.

Disable/Enable Block

Some students might have previous experience with text-based programming, like *Python*® or *Java*®. Disabling a block is just like commenting out code. For example, the following piece of code has lines 2 and 4 commented out, which means the program will skip them when the program runs. They are *not* part of the active program.

```
x=10
//X=20
print X
//print "hello!"
print "goodbye!"
```

When the program runs, it will print "10" for X and "goodbye!"; the other two lines are commented out.

The same logic applies for disabled blocks in App Inventor.

Collapse/Expand Block

Students might develop a possible misconception that if the code is collapsed, then it is no longer part of the program. Make sure to emphasize that this is not true. If the code is still enabled, then it is still part of the program!

Collapsing code does not directly help debug a program, but it does help make it more readable when the program gets complicated and contains many event-handler blocks. Students may want to collapse code blocks when they know they are not actively working on them to help keep their focus on the code that they are currently developing.

Other Useful Commands

Encourage students to use these commands to keep their code organized and make it easier to find blocks.

You may want to have students use the **Download Blocks as Image** command to submit their code to you as a checkpoint during an activity, project, or the problem, to keep a record of their code, or to share code with other students.

Pair Programming

Using the pair programming collaboration approach here is intended to introduce the students to the approach and get students acclimated with how it works. It will be applied again in future APBs.

Before you do this activity with students, take the following steps to get ready:

1. Prepare the list of partners to pair up for pair programming.
2. Watch this video about pair programming facilitation.



Pair Programming Teacher's Perspective video

3. Preview the two videos found in the course.

You can choose to have students switch roles every time they find a bug or every 10 minutes, whichever comes first.

Blast the Bugs

Note: If students use an emulator for this activity, they will not be able to test and fix the camera function.

The Study Buddy App

- You may choose to print the requirements for this app for students to add to their PLTW Gateway Notebooks and check off the requirements as they fix them in the code.
- Encourage each pair of students to communicate among themselves and share ideas about the bug and how to solve it.

Prompting Questions: While students are debugging their programs, observe their progress and facilitate as needed. Make sure you *do not* give them any answers.

You may ask them prompting questions such as:

1. What does the program do?
2. Have you thoroughly read the app description?
3. What features have you tested?
4. Have you identified any features that did not work as expected?
5. Did you read all the comments to better understand the program's purpose and logical sequence? Did you find any problems or inconsistencies?
6. Have you isolated the bugs? How do you plan to fix or debug them?

An Error: If students use the *Do It* command on blocks that require values to be set during runtime by their event handlers, then they will get an error that states *"variable is not bound in the current context"*. This means that when debugging in App Inventor's *Blocks Editor*, the variable being used in the block does not have a value. It only gets its value during the runtime of the app. So students will not benefit from the *Do It* command in event handlers like *Touched*, *Dragged*, and *PositionChanged*, because these handlers depend on runtime values, which do not exist in the context of the *Blocks Editor*.

Study Buddy Bug Descriptions and Fixes



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

When debugging the Study Buddy app, students will encounter the following bugs.

- Before any code is fixed in the *Touched* event handler, whenever the user taps the canvas, a circle is drawn in the wrong spot.

Reason: The *centerX* value is being set to the *get y* value.

Fix: Change *centerX* to *get x*.

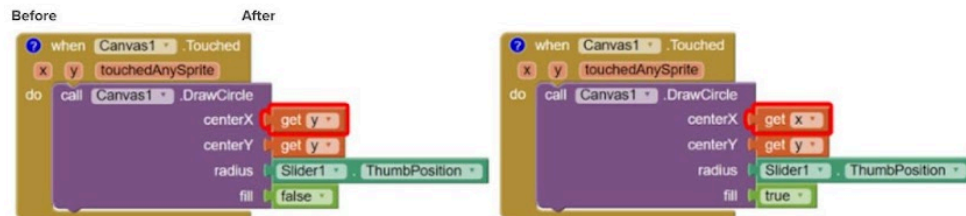


Figure 1. *centerX* Value

- The circle that is drawn is NOT filled with the chosen color.

Reason: The *fill* value is set to *false*.

Fix: Change *fill* to *true*.

- Before any code is fixed in the *Dragged* event handler, when the user tries to draw a line, the line does not follow their fingertip. It draws lines in strange directions.

Reason: The *x1* and *y1* slots have their values swapped.

Fix: Place the *get prevX* in the *x1* slot and the *get prevY* in the *y1* slot.

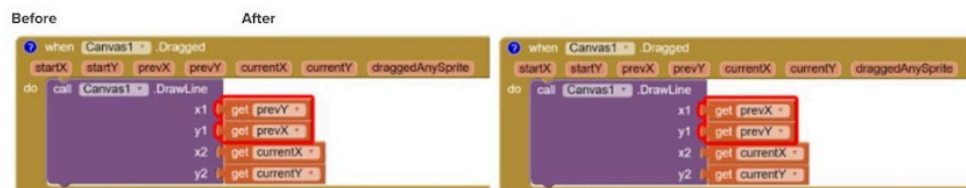


Figure 2. *x1 = prevX* and *y1 = prevY*

- The three buttons for setting the color have incorrect colors selected.

Fix: Click the color value and select the correct color from the palette.

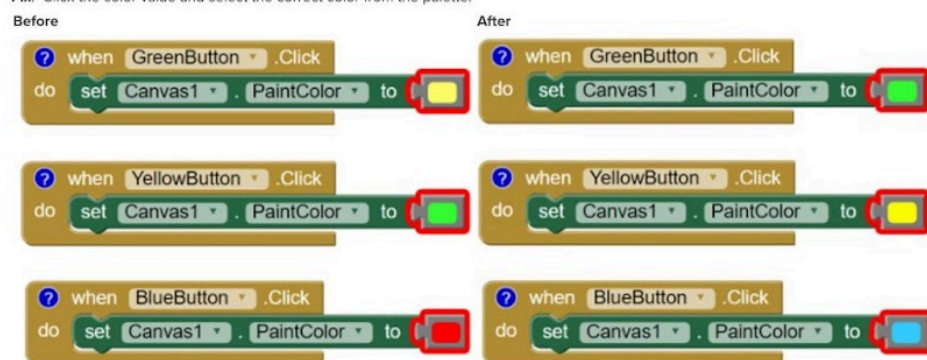


Figure 3. Fixing Button Colors

- The Clear button does not do anything.

Reason: The *Clear* block is disabled.

Fix: Right-click and enable it.





SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS



Figure 4. Enabling the ClearButton

6. The Take Picture button does not do anything.

Reason: The *TakePictureButton.Click* event handler has no code in it.

Fix: Drag the *Camera1.TakePicture* block that is next to it and place it into the event handler.



Figure 5. Camera1.TakePicture Block

7. The Take Picture will then launch the device camera, but nothing happens after you take the picture.

Reason: The *Camera.AfterPicture* event handler has no code.

Fix: Drag the *Canvas.BackgroundImage* block that is next to it and place it into the event handler.

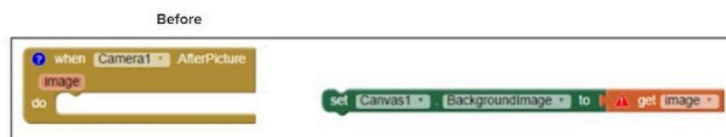


Figure 6. Canvas.BackgroundImage Block



Figure 7. Camera1 Event Handler

Instruct each team to compare debugging work and resolutions with another team. To help teams describe their work, you may choose to ask them some of the prompting questions listed above.

Pair Programming: To help you when pairing students for Project 1.8, reflect on how students did at each role.

Ask students to write down what they liked about being the Driver role versus Navigator role, what worked well, and what could be improved in their Driver/Navigator relationship.

Extra Challenge

If students finish early, have both partners each add bugs to a program and swap so that the partner has to debug the code!



Activity 1.7

Game Time

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVE

LO

LO 2.1B Create and store data during the execution of a program.

KNOWLEDGE AND SKILLS

KS

KS 1.3A3 Decompose a problem into smaller parts.

KS 2.1B1 Describe ways data can be stored on a digital device.

KS 2.1B2 Store, access, and update data stored in variables (temporary data).

KS 2.4A4 Trace a program and deduce the values that variables will have after the code is executed.

KS 2.4B4 Debug programs using a variety of strategies to isolate and identify problems, including analyzing error messages, analyzing variable values line by line, generating output, or commenting out code.

KS 3.2A1 Communicate to meet the needs of the audience and be appropriate to the situation.

ESSENTIAL QUESTIONS

EQ

EQ 1.2 How do computer scientists design and develop mobile apps that accomplish desired tasks?

RESOURCES

[A1_7_GermFighter_TR.aia](#)

[A1_7_GermFighter_TR_withExtensions.aia](#)

[Code Trace Table Blank](#)

[1.7 Germ Fighter Game Design Requirements](#)

[Activity 1.7 Answer Key](#)



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Preparation

Students will use pair programming when working on this activity. Prepare a list of partners to assign for the activity. Refer back to [Activity 1.6 Teacher Resources](#) for more information on facilitating pair programming in your classroom.

Activity Time: 4 Days

Possible Student Misconceptions

- Variables are used in computer programming the same way they are used in mathematics.
- Variables can only be numeric.
- The equal sign represents a mathematical equation.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Becoming a Game Developer

Although students will work through this activity individually, each student should work with a partner, if the student has questions or needs help. Remind students that they should ask peers for help before they ask for your help. Students should ask their partners to check their work and provide feedback. Consider asking more advanced students to help those who are new to computer science.

Discussion: Lead a short class discussion about the games they have played and what these games have in common. The goal is to get the students to realize that a score, time, lives, and health are common features, and all are tracked by variables as they will learn later in this activity.

Pair Programming

Before you do this activity, prepare the list of partners you want to pair up for pair programming.

Discussion: Have a quick discussion about their experience of pair programming in the previous activity to make sure they are ready to apply it again in this activity.

Reinforce the benefits of pair programming, such as:

- Two heads are better than one (writing and reviewing code at the same time)
- Peer coaching and help
- Collaboration skills
- Communication skills

Track a Variable

Work with the class on the example algorithm and trace table and make sure they understand how to use the trace table.

Algorithm:

X = 0
Y = 15
X = X + 2
Y = Y + X



X	Y
0	
0	15
2	15
2	17

Figure 1. Variable Tracing Chart Answer Key

Have the students identify the lines of the algorithm that initialize the x and y variables, and which ones update them. Are the update statements incrementing or decrementing the variables?

The goals of this exercise are to:

- Show that the value of a variable can change during the execution of a program or algorithm.
- Reinforce that a line, such as " $X = X + 2$ " does not represent a math equation (that would be an impossible math equation!). This code is an assignment statement. That means we are assigning the variable X a new value, which is equal to the previous value of X plus 2.

Important: It is very important to address the possible misconceptions that variables are the same as what they have learned in math, and that the equal sign means an equation!

The Germ Fighter Game

- Mention to the students that it is good practice to write the requirements of an app by identifying the app's main components and their functions. When designing a game, students should also determine how the game is won and lost.
- This activity will prompt the students to switch pair programming roles for each section. Make sure that students are switching as prompted. A good strategy when pair programming is to have the activity displayed on one computer, while programming in App Inventor on the other. This allows the Navigator to read through the directions as the Driver works on the program build.

Reminders:

- Remind students to save their files using the file structure determined in the first activity.
- Importing a project might take time. If it times out or fails, go back to the My Projects screen and check whether the project was actually created. If not, then try again.
- Determine where you want students to deploy the app to, the emulator or the mobile device.

View Layer

The screens of mobile devices range in size. If the height of the canvas suggested in this step is too big or too small for the device, students can try another value to make the game fill the screen just right, that is, without affecting the horizontal arrangements that contain the labels and buttons.

Controller Layer

The Start Button

- For more information about the properties of the *ImageSprite*, click the question mark icon on the component and select **More Information** to go to the MIT App Inventor library.
- Make sure that if they change the *Visible* checkbox, they click it **again** so it does not affect the app's functionality. In other words, the *Visible* checkbox should be selected (checked = True). Editing the property is just a workaround for the students to refresh their app in their emulator or device without having to reset the connection and reconnect.



The Reset Button

Important: It is very possible that students will think of the grid as an XY plane. Make sure you address that by referring to the illustrated grid that shows the origin (0, 0) in the top-left corner and all values of x and y being positive. Use the questions below to formatively assess their understanding of the grid.

Canvas Grid

1. Project the image of the x,y grid and ask students to identify the location of the following:

- (500, 0)
- (0, 200)
- (-50, 100) Does not exist! (no negative values on the canvas grid)
- (50, -100) Does not exist! (no negative values on the canvas grid)
- (300, 500)

Assume each cell in the grid represents 50 x 50 pixels.

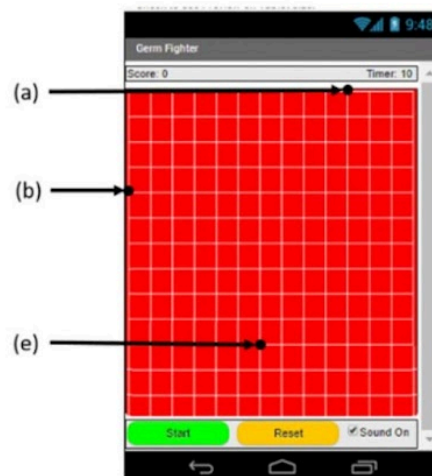


Figure 2. UI Canvas Grid

2. What if you want to move something to the center of the canvas, but don't know how many pixels in height or width the canvas is?

We can use the canvas properties for height and width and divide each by 2 to get the midpoint of each. So we set the x value to "canvas.width/2" and the y value to "canvas".

WhiteBloodCellSprite Behavior

- Direct the students to explore all of the event handlers available to them for this sprite in the *WhiteBloodCellSprite* drawer. They can hover over the blocks to get a description of each.
- Note that "prevX" and "prevY" are the values of the previous location and "currentX" and "currentY" are the values of the new location of the fingertip. As the user drags the sprite, this event is triggered over and over until the user stops dragging. So these values are constantly changing as long as the sprite is being dragged.
- The reason we keep the y at the same current value, is that we do not want the *WhiteBloodCellSprite* to move up and down the canvas. Instead, we want it to stay at the same vertical location and only move left and right.

GermSprite Behavior



Important: Students might think that there is only one way to solve a problem in coding. That is not necessarily true. It is important that students realize there are many ways to solve the same problems. Show them the following two options to compare the *if* statement in each. This activity will use the solution on the top.

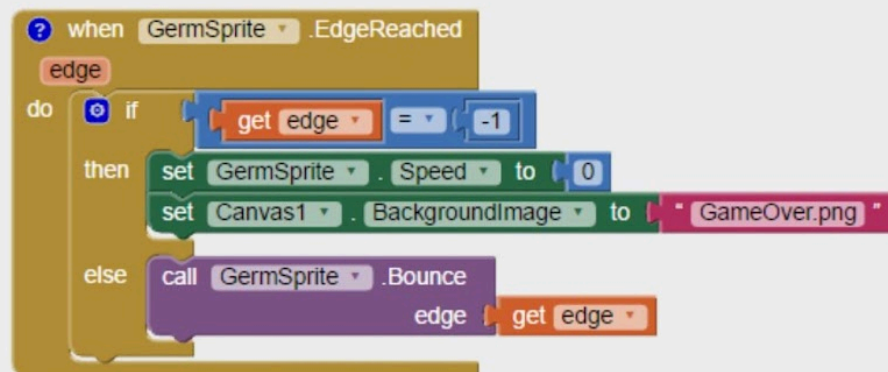


Figure 3. GermSprite.EdgeReached Event Handler - Option 1

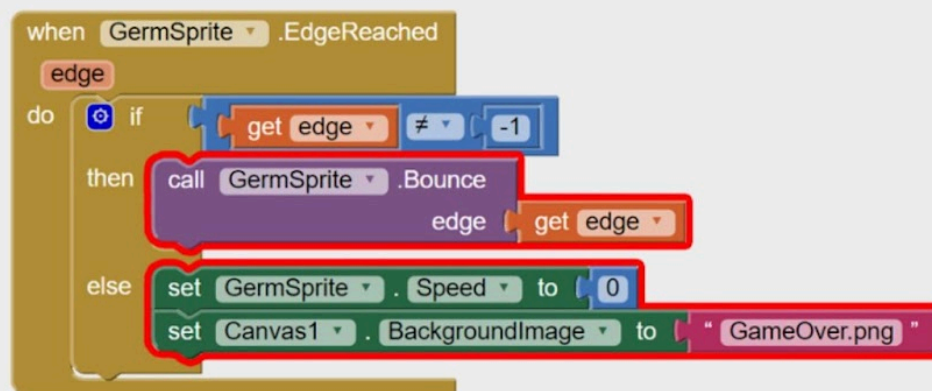


Figure 4. GermSprite.EdgeReached Event Handler - Option 2

- Allow students a few minutes to try to build this event handler. If they struggle, remind them of what should be done by asking them questions, such as:
 - What is the event that triggers the bounce?
 - In which cases do we want the sprite to bounce off the edge, and in which cases do we let it go and end the game?
- Reinforce incremental testing. If students have bugs in their code, it is a good opportunity for them to practice persistence as they work with their partners to debug their code.
- To ensure students do not fall behind, you may choose to guide them through some of the code. Avoid sharing solutions unless really necessary.

Programming the Score

- It is important for students to understand that variables need to be initialized before the program uses them. In App Inventor, you are forced to initialize them upon creation. However, in other languages like C or Java®, if you do not initialize a variable, you cannot predict what type of value is assigned to it by default! Therefore, it is good practice to always initialize variables to the values you'd like them to start out with.
- This section contains exercises similar to the ones students did at the beginning of the activity, where they traced values of variables. Incrementing (or decrementing) variables by 1 is a very common practice in computing, since it is used for scores, timers, number of lives in a video game, object counters, and so on.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Note: To increment and decrement variables in a program, use the addition or subtraction blocks found in the *Math* drawer.

Reminder: Remind students that the *join* block joins two text strings together. So even though the value of the score is a number, it converts that number into a text data type and it joins it to the string "Score: "

- You can give students a blank trace table if they need to trace their code or variables as they test their app.

Programming the Timer

- Recall that some components in App Inventor, like the *ImageSprite* and the *Clock*, have an *Interval* property. The *Interval* property determines how frequently the internal clock of the component should tick. The *Clock* component is a special component, because it already comes with built-in blocks that relate to its timer (such as the *When Clock1.Timer* event handler).

Note: A shortcut is to right-click on the score initialization block and select **Duplicate**. Then rename the variable to *timeLeft* and change the value to "10".

- Give students time to work on this. They should be able to do it all, since they have done very similar work earlier in this activity. Remind them of the code they used for updating the score as well as the code for programming the loss scenario.
- It is important for students to understand that finding and fixing errors is normal and expected when programming. It requires patience, persistence, and iteration (many trials!)

Hints:

1. Review the code in the event handler that is behaving incorrectly.
2. Check whether any code is missing or incorrect.
3. Trace any variables that seem to be incorrect and find the code that might be causing the error.
4. Fix one thing at a time, then test again.



Project 1.8

Build-A-Body

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVES

LO

- LO 1.1A** Describe and analyze moments within a problem-solving process where persistence, iteration, and the positive aspect of failure played an important role in gaining understanding about a problem or unexpected observation.
- LO 1.1B** Demonstrate creativity and courage to take risks in proposing designs.
- LO 1.2A** Apply an iterative process to solve a problem or create an opportunity that can be justified.

KNOWLEDGE AND SKILLS

KS

- KS 1.1A1** Recognize that identifying complex problems, defining them clearly, and proposing solutions can be difficult and requires persistence and iteration.
- KS 1.1B1** Develop solutions employing nontraditional techniques; novel combinations of artifacts, tools, and techniques; and exploration of personal curiosities throughout a creative process.
- KS 1.2A2** Explain the goal of any design process.
- KS 1.2A3** Generate ideas and/or build upon existing ideas to innovate.
- KS 1.2A2** Iteratively test throughout the design process to ensure that a solution meets the requirements.
- KS 1.2B1** Investigate the types of interactions between users and a proposed solution.
- KS 1.3A2** Choose appropriate computational practices when solving a problem.

ESSENTIAL QUESTIONS

EQ

- EQ 1.1** In what ways does technology affect people's lives?
- EQ 1.2** How do computer scientists design and develop mobile apps that meet desired needs?
- EQ 1.3** What does effective teamwork look like?

RESOURCES

-  [A1_8_BuildABody_TR_alloptions.aia](#)
-  [A1_8_BuildABody_TR_decrement.aia](#)
-  [A1_8_BuildABody_TR_timer.aia](#)
-  [A1_8_BuildABody_TR_organinfo.aia](#)
-  [A1_8_BuildABody_TR_solution.aia](#)
-  [Project 1.8 Answer Key](#)
-  [Project 1.8 Sample Solution](#)
-  [1.8 Build-A-Body Rubric](#)
-  [1.8 Build-A-Body Peer Feedback Form](#)
-  [1.8 Build-A-Body App Design Requirements](#)



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Preparation

- Print a [Project 1.8 Peer Feedback Form](#) for every group of students.
- Review the [Project 1.8 Sample Solution](#) to acclimate yourself with one way to incrementally build an app solution that meets the given design requirements/criteria for this project.
- Students will use pair programming to collaboratively design and build their apps. Determine how you will group the students. For more information on facilitating pair programming in your classroom, refer to the [Activity 1.6 Teacher Resources](#).
- Review the [Project 1.8 Build-A-Body Rubric](#) and determine how you will use the rubric to assess student work and collaboration. Review the PD resources on rubrics.

Project Time: 5 Days

Possible Student Misconceptions

- I have to be an expert programmer to successfully develop an app.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

The Design Process

This project is estimated to take five instructional days. Use the chart below as a guide for pacing your students for this project. Students are expected to complete certain steps of the design process for their app solutions on each day. If students are taking longer than is expected, develop a plan to help get them back on track to meet the deadline. This may mean students working outside of class to catch up or you providing the groups with helpful hints to move their work along.

Day	Milestones
1	Review the design process and rubric, download starter code, and complete design sketches.
2	Fill out the event chart and create a flowchart for the algorithm of each event handler identified in the chart. Review and explain the event handler charts and flowcharts before moving on to Day 3.
3	Build the app using the starter code.
4	Test the code and make necessary changes.
5	Assess the apps using the rubric and peer feedback.

Discussion: Lead a discussion with students about the application of computer science to other fields, such as biomedical science. Ask students to share computer science solutions for biomedical science problems. Examples to share include:

- Pacemaker—a device that deliver electric current to the heart when it is beating irregularly
- Electronic monitoring devices (heart monitor, blood glucose monitor, and oxygen monitor)

Review the steps of the design process. Throughout the project, students are guided through each step of the design process and given prompts and tasks to help them complete each step.

1. **Define the problem.** As part of this step, designers validate the problem or opportunity by asking questions, such as:
 - Who says it's a problem?
 - What are the needs and wants?
 - Are there any prior solutions to this problem?
2. **Generate concepts.** When generating concepts, designers research past solutions and brainstorm new ideas. When brainstorming, designers capture all the ideas they can think of without critiquing each other.
3. **Design a solution.** Designers create a detailed design using technical drawings and models. The drawings and models provide the technical information necessary to produce a solution.
4. **Build and test.** Designers build and test their solution. They test its performance, usability, and durability, and they collect and analyze the test data.
5. **Evaluate the solution.** Designers test the solution's effectiveness, reflect on their design, and recommend any potential improvements. They will go back to step 4 if needed to improve the solution.
6. **Present the solution.** Present the final solution to the stakeholders and clients.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Define the Problem

In this stage of the design process, students will read a design brief for the problem that they will be solving.

- Give students time to think about the problem and design statement for this project. Ask students to think about how they can apply what they have learned and done so far in the unit to create an app to solve this problem.
- Review the design requirements/criteria and deliverables with students and have students review the [rubric](#) for this project. Go through each category of the rubric and discuss the expectations with students. For example, remind students that they must document each step of the design process in their PLTW Gateway Notebooks, include all of the required design requirements/criteria in their apps, and use programming best practices—such as adding comments to their code—because all of these categories will be assessed using the rubric.

Design a Solution

View Layer

- Students will download the source code for the starter app. The source code contains all of the media images and includes some of the required components preprogrammed.
- Students should refer to the design requirements/criteria and start sketching the components needed for the app. They should label the components and describe their intended function. They should also identify any user interaction with each component (if applicable) and the behavior that results from the interactions.
- All students need the following components, regardless of which additional feature they decide to implement:
 - Reset button
 - Score label
 - Horizontal arrangement and/or vertical arrangement for organizing the button and label
 - Brain sprite (provided)
 - Digestive system sprite (provided)
 - Urinary system sprite (provided)
 - Respiratory system (with heart) sprite (provided)

Reminders:

- Remind students to save their file to the appropriate folder.
- Remind students that their app needs to meet all of the design requirements/criteria.

Controller Layer

- Using the list of design requirements/criteria, students should start thinking about how to achieve the desired behavior for their app.
- Students should plan their design by filling out the event handler chart and creating a flowchart for the algorithm of each event handler identified in the chart. Example event handler chart and flowcharts are in the [Project 1.8 Answer Key](#).

TouchUp versus CollidedWith event handler

The table provides descriptions of both event handlers for comparison.

Event handler	Description
<i>TouchUp</i> (number x, number y)	When the user stops touching the sprite (lifts finger after a <i>TouchDown</i> event), provides the (x,y) position of the touch, relative to the upper left of the canvas.
<i>CollidedWith</i> (component other)	Handler for <i>CollidedWith</i> events, called when two sprites collide.

It is possible that students will initially choose the *CollidedWith* event handler for checking whether the organ/body system are colliding with their placement dots. The problem with using the *CollidedWith* event handler is that it will trigger even when the user is still dragging the organ/body system sprites, so it can get triggered multiple times while the user is dragging the organ/body system sprite—once for every time the organ/body system sprite collides with any other sprite. That can cause three issues:

1. The logic in the event handler will force the organ/body system sprite back to its original place every time it touches any sprite that is not its correct placement.
2. It will affect the score keeping, resulting in adding a point every time the organ/body system sprite collides with its placement dot, even if the player is still dragging it.
3. It will also affect the score keeping, resulting in subtracting a point every time the organ/body system sprite collides with any other sprite.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

All of these issues are caused because the *CollidedWith* event is triggered at the collision of the organ/body system sprite with any sprite, even while the player is still dragging it.

To solve these issues, use the *TouchUp* event. This event is only triggered when the player lifts his/her fingertip after dragging the organ/body system sprite. That works perfectly for this app, because we want the app to check whether the organ/body system sprite is at the correct placement dot when the player lifts their fingertip off the screen—not while still dragging the organ/body system sprite! That way, the app will only check the placement after the organ/body system sprite collides with the correct placement dot, then lock it in place, and add a point to the score. Otherwise, the organ/body system sprite will return to its original (x, y) position and subtract a point from the score.

Build and Test Solution

Important: To make sure everyone is on track, review and discuss all the event handlers and flowcharts with the class before moving on.

- Guide the students through the development as they use pair programming to build their app. Students will begin with starter code that has the body outline on the canvas. The body outline will have the correct organ/body system placements (dots) for matching the organ/body system.
 - Students need to complete the view layer:
 - Complete the UI design and layout of the app per their app sketches.
 - Name their components properly and according to correct naming conventions (descriptive names and in camelcase).
 - Set the properties as needed for size, color, font, label, and so on.
- Students need to complete the Controller layer. As students prepare to begin work on the Controller layer, they need to fill out the xy coordinates table for the sprites and the corresponding original and placement positions.
 - To retrieve the (x,y) coordinates of the original location of each sprite, students can click the sprite in the *Designer* and check the x and y properties in the Properties pane.
 - As for the location on the body, they can check the coordinates of the placement sprites (black dots) that are already placed on the body to help identify the correct location.
- Advise students to start developing with one organ/body system. Once they figure out how to meet the design requirements/criteria for that organ/body system, they can adapt the code for the other organ/body system. For example, if students program the digestive system sprite first, they can build the code for the different requirements, and once functioning correctly, they can reuse the code for the other body system sprites.
- If students are struggling with their code, offer hints, but do not provide answers. Have the students think about the components that the user is interacting with and explore the events and blocks that are available in App Inventor for these components.
- Guide students by providing support or resources, not by providing the solution to their problem. For example:
 - Users have to touch the sprite and move the sprite with their finger. Which block captures this event? The *when dragged* event block.
 - Users will drag the sprite to the human body outline, and when they think the sprite is in the correct location, they will lift up their finger. Which block captures this event? The *touch up* event.
 - How will the app know whether the sprite was released on the correct placement dot? Use the *if collidedWith* conditional statement to detect whether the sprite is released on the correct placement dot.
 - What code block should students use to make the sprite move to a specific location, specifically, back to its original location on the screen? The *moveTo x, y* block.
 - To program the score function of the app, have students reflect on their work in the game they created for Activity 1.7 Game Time. Ask them whether the score is a static number that stays the same throughout the game or a number that will change as the game progresses. If it is a number that will change, ask students what they can use as a container to hold this type of dynamic information. Lead them to think about variables and how they can set up the score variable for this app.

Evaluate the Solution

- Students should share their apps with another team for testing. The testers will determine whether the apps function as specified in the design brief and provide written feedback on the [Peer Feedback Form](#). If the apps are not functioning or need modifications, the developers can make necessary changes to update the apps.
- If time permits, students may add additional features to their apps.

Present the Solution

- Student apps and documentation should be assessed using the [Project 1.8 Build-A-Body Rubric](#). Consider having students use the rubric to do a self-assessment.
- After submission, students should answer the reflection questions, and then as a class, they can discuss their responses.

LESSON 1 INTERIM ASSESSMENT

The App Creators Lesson 1 Interim Assessment is located in the Unit Level Resources, Assessment Page. The assessment is a starting point for you to use with your students. Feel free to edit this assessment to best meet your needs and the needs of your students. Use the Lesson 1 Interim Assessment Answer Key, to evaluate student work.



SPRINGFIELD PUBLIC SCHOOLS
ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS



Lesson 2

Taking It to the Next Level – Quick Guide

Preface

In Lesson 2, students further explore the concepts investigated in Lesson 1 and build upon their skills to use data in mobile applications. They create algorithms using loops to streamline repetition and iterate through lists, and create procedures to abstract the details of a task and reduce redundancy. They learn to organize and store persistent data collected from user input and device sensors.

Essential Questions

1. In what ways does technology affect people's lives?
2. What do programming best practices look like?
3. How do diverse perspectives of team members impact a solution?

Activity, Project, or Problem	Days
Activity 2.1 Keep Me in the Loop	2
Activity 2.2 Programming with Procedures	1
Activity 2.3 Playing with Lists	2
Activity 2.4 Disease Tracker	3
Activity 2.4 Extension Persistent Data	1
Project 2.5 Fitness Challenge App	6
Lesson 2 Summative Assessment	1
Total	16 Days



Activity 2.1

Keep Me in the Loop

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVE

LO

LO 2.3A Develop and apply abstractions.

KNOWLEDGE AND SKILLS

KS

KS 1.3A Apply computational thinking to solve problems.

KS 2.2A Identify what has been made more general by an abstraction and identify what details have been hidden or removed.

ESSENTIAL QUESTIONS

EQ

EQ 2.1 What do programming best practices look like?

EQ 2.2 How do diverse perspectives of team members impact a solution?

RESOURCES

[A2_1_KeepMeTheLoop_TR.aia](#)

[A2_1_KeepMeTheLoop_TR_withExtension.aia](#)

[Activity 2.1 Answer Key](#)



Preparation

Students will use the algorithms they created to navigate the bus to the museum in Activity 1.2 Amazing Algorithms.

Activity Time: 2 Days

Possible Student Misconceptions

- Algorithms displayed in a flowchart must show every step individually.
- Algorithms are only linear.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Repeating Tasks

Introduce the idea of reducing redundancy. Students can discuss activities that have many repetitive steps. Ask the students how excess repetition in a set of instructions, such as the 200 steps instructing a baker to add an egg to the batter, may make it hard to follow.

Algorithms with Repetition

Students explore reducing redundancy in algorithms and then use it to learn about loops. They revisit the algorithm and map from Activity 1.2 Amazing Algorithms. They start with a small portion of the algorithm for the bus's route as an example for learning about loops.

Students should be aware that there are different methods to reduce redundancy in computer programs. Methods include loops, which are used to repeat a set of instructions for a specified number of times or indefinitely. Procedures are also used to reduce redundancy. A procedure is a set of code that performs an action. In this activity, students learn to build loops to reduce the use of redundant code. They also use procedures to navigate the bus to the museum.

Loops in App Inventor

Students will use the *for each from to by* control block in App Inventor to loop through code to reduce the redundancy in the program. To read a description, click the popitp beside each of the parts of the block in the image.

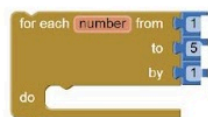


Figure 1. for each number from-to-by Block

When using this loop, students will need to determine how many times the *moveForward* step should be repeated to move the bus from one point on the map to another.

Building an App

- Students download and load the source code into App Inventor. The prebuilt app has the background and bus sprite already loaded. The procedures for the commands for the bus movements (*moveForward*, *turnRight*, and *turnLeft*) are also included in the prebuilt app. The navigation blocks are collapsed and should not be altered by students, otherwise the bus may not move in the correct direction.
- Similarly, code for adjusting the canvas size and the bus size are provided in the procedure called *adjustComponentSizes*. The code in this procedure should NOT be edited by students, because it ultimately affects the *moveForward* procedure. Note that the program uses a *TracingLabel* component to trace the values of the screen size, canvas size, and the bus size and adjust them as needed based on the tablet's screen size. If you have time at the end of this activity, you may like to show the trace code as an example of using a label for tracing. The code is currently disabled but can be enabled for testing.
- As students complete this exercise, remind them of pair programming roles. Ensure that the Driver and Navigator in each student team are adhering to their defined roles. If not, review the pair programming strategy with students. Consider having them switch every 15 minutes or each component. You may choose to have students watch the pair programming video again outside of class.



- Lead a discussion about procedural abstraction. Ask students to think of reasons that procedural abstraction might come in handy. Reinforce the idea that the purpose of a procedure is to reuse code and to help decompose the program into smaller parts of code—each of which accomplish a task.

Background Information: Procedural Abstraction

Show

- The three collapsed blocks and the expanded *turnRight* block are shown below.

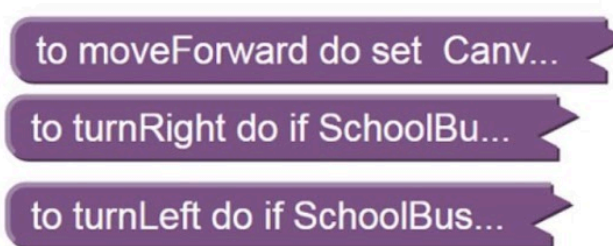


Figure 2. Navigation commands

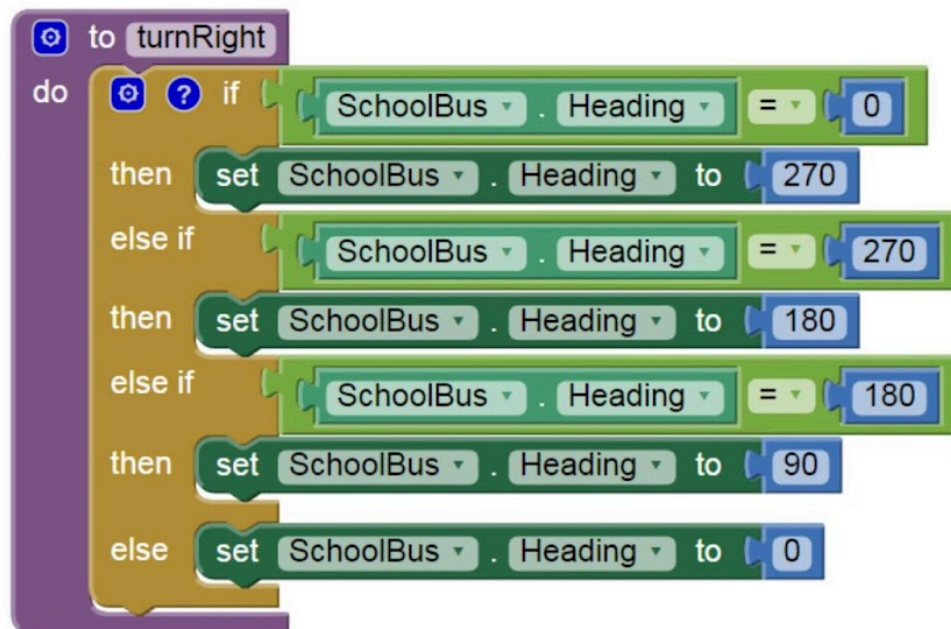


Figure 3. turnRight Code

- The bus will turn based on the orientation of the bus. The bus is initially pointed at the 0 degree direction (north).
- Students will need to make sure their app has the required buttons (Start and Reset) to trigger the program to work.
- The bus has three different routes it can use to travel to the school. Students can build their app using their algorithm from Activity 1.2 Amazing Algorithms or they can choose a different route for the bus to travel.
- Before building the app, students should review their algorithm flowchart first to make sure it directs the bus from the school to the museum using the map from Activity 1.2 Amazing Algorithms. Next they should add loops to reduce the redundancy.
- You can guide the students on the steps for building their app, or you can allow them to figure out the best way to proceed themselves.
- Students should build an app with the least amount of redundancy to simplify the code for the bus travel. There will be many solutions to the bus traveling to the museum, so students can choose their favorite route for the bus to take. Remind students to build and test their app one set of steps at a time and fix errors as they arise. Students should click the Reset button after each runtime to restart the program.



Optional: Extension

As an extension, students can program the red line that traces the bus's route to line up in the center of the road for the journey from the school to the museum.

Have students compare their code with another group. Suggest the following questions for reflection.

1. Did you choose the same route for the bus to travel?
2. Which route was shorter?
3. Which route has the least amount of redundancy?
4. Can you identify all the loops in the program?



Activity 2.2

Programming with Procedures

Teacher Resources

[View Student Course](#)

KNOWLEDGE AND SKILLS

KS

KS 2.3A3 Define and use procedures that hide the complexity of a task and can be reused to solve a similar task.

ESSENTIAL QUESTION

EQ

EQ 2.1 What do programming best practices look like?

RESOURCES

- [A2_2_Charades_TR.eia](#)
- [2.2 and 2.3 Charades Game Design Requirements](#)
- [Activity 2.2 Answer Key](#)



Preparation

Determine how you will facilitate student sharing of the mobile devices and how they will deploy and test on both the mobile device and the emulator.

Ethical Considerations

Whether communicating in a virtual or physical environment, students should demonstrate positive digital citizenship skills. The words or phrases students use in the charades app should be appropriate and not offensive or insensitive to others. Students are responsible for the digital content and programs they create.

Activity Time: 1 Day

Possible Student Misconceptions

- Variables can only hold one value at a time.
- Computer/computing variables are the same as mathematical variables.
- Variables are only numeric.
- The equal sign represents a mathematical equation.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

The Charades Game

Although students will continue to work individually through this activity, students should work with a partner when they have questions or need help. Remind students that they should ask a peer for help before they ask for your help. Students should ask their partners to check their work and provide feedback. Consider asking more advanced students to help those who are new to computer science.

Understanding the Requirements

Discussion: Project the game description for students; as a class, discuss each requirement together.

Exploring the Starter Code

Students will:

- Explore the code that is prebuilt in the app.
- Determine the current functionality that is set in the code.
- Answer questions to assess their ability to analyze and understand the code that was prebuilt.

Depending on time and your students' level of understanding, you may choose to go through this entire section of this activity as a class, step by step, to make sure all students understand the code equally, or you may have students independently work through each step and assess their understanding. How much time you spend or how deep you go is up to you. The goal of this section is to make sure the students all understand the code that is prebuilt.

Reminder: Remind students to save their files using the file structure you determined in the previous lesson.

Discussion: Lead a class discussion working through all existing code, to make sure students understand what code is prebuilt and what behavior each set of blocks code for. It is up to you to determine how deep you want students to go.

Background Information: Clock/timer Functionality

In the *Designer* view, you will note that the `TimerEnabled` property is not checked.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

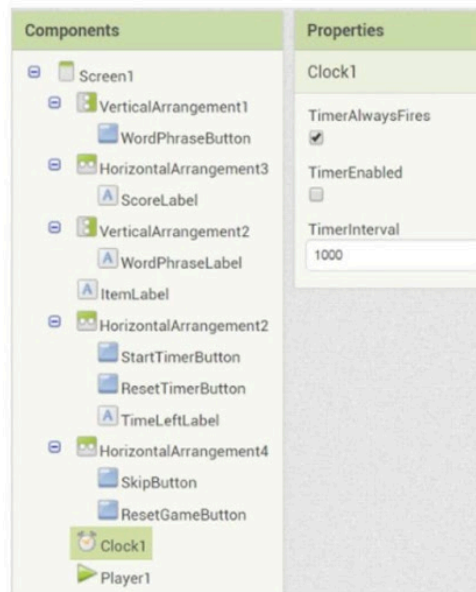


Figure 1. Clock1 Properties

This means that the timer does not fire until it is enabled in the code. The code that enables the timer is shown below.

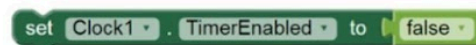


Figure 2. Code to Enable Clock

In this Charades program, you will see that the timer is enabled in the *when StartTimerButton.Click* event handler.



Figure 3. TimerEnabled true

When the timer is enabled, it triggers the *when.Clock1.Timer* event handler to run.

Look at the following event handlers:

- *when ResetTimerButton.click*
- *when ResetGameButton.click*
- *when Clock1.Timer*

Why do they all contain the following code?

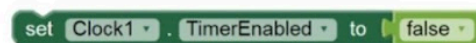


Figure 4. TimerEnabled false

What would happen if they did not contain this code?

Deploying the App



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Students will be using the accelerometer sensor built into the mobile device, so it is necessary that they deploy their apps to the mobile device when testing this functionality. However, because students are working individually on this app, depending on the number of mobile devices you have available, students may also have to deploy the app to the emulators at different points during the activity. Determine how you will facilitate student sharing of the mobile devices and how they will deploy and test on both a mobile device and the emulator.

Reducing Redundancy

- Students learned about loops and procedures in Activity 2.2 Keep Me in the Loop. To reduce the redundancy in this program, they will write a procedure that is composed of the redundant blocks of code. Then, whenever this code is required, you can call the procedure instead of repeating all of these commands again.
- Review with students the concept of abstraction and what they learned in the Keep Me in the Loop activity about how procedures are a form of abstraction.
- The program students will write for this activity will get lengthy, so it is necessary for students to annotate their code with comments as well as collapse any event handlers that they have finished programming.
- Students will complete the Charades app in the next activity.



Activity 2.3

Playing with Lists

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVES

LO

LO 2.1A Collect, process, and analyze real or simulated data.

KNOWLEDGE AND SKILLS

KS

- KS 1.3A2 Choose appropriate computational practices when solving a problem.
- KS 2.1A2 Collect, process, and interpret data to gain insight on a problem and draw conclusions.
- KS 2.1B3 Store, access, and update data stored in a list (temporary data).
- KS 2.2A5 Create simple algorithms that involve variables, conditionals, operators, or logic.
- KS 2.2A6 Create algorithms to perform repetitive or reusable tasks.
- KS 2.2A3 Analyze, break down, and explain the logic of an algorithm.
- KS 2.3A2 Identify what has been made more general by an abstraction and identify what details have been hidden or removed.
- KS 2.3A3 Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks.
- KS 2.4A1 Identify and describe the high-level structures of a program, such as user-interface components, data components, event handlers, and procedures.
- KS 2.4A2 Describe the appropriate code blocks and conventions used in the programming language.
- KS 2.4B2 Design and develop a program by breaking a large plan into smaller modules using procedures and event handlers (if applicable).
- KS 2.4B3 Test code frequently as it is being developed.
- KS 2.4D2 Navigate and use an app development software to design, develop, and test a mobile app.
- KS 2.4D4 Select and use appropriate features to meet the desired functionality of an app, such as user interface components, layout arrangements, media objects, drawing and animation, sensors, location awareness, accessing phone features, and data storage.
- KS 2.4D5 Deploy an app using the emulator or a mobile device.

ESSENTIAL QUESTION

EQ

EQ 2.1 What do programming best practices look like?

RESOURCES

-  A2_3_Charades_TR.aia
-  2.2 and 2.3 Charades Game Design Requirements
-  Activity 2.3 Answer Key



Preparation

Determine how you will facilitate student sharing of the mobile devices and how students will deploy and test their apps on both the mobile device and the emulator.

Student Accommodations

If there are students in your class who are unable to do the physical acting, students may use the charades app to play Pictionary™ (where players draw the words or phrases) or Catch Phrase® (where players give verbal hints of the words or phrases).

Activity Time: 2 Days

Possible Student Misconceptions

- Variables can hold only one value at a time.
- Computer/computing variables are the same as mathematical variables.
- Variables are only numeric.
- The equal sign represents a mathematical equation.
- In computer programming, the order of the items in a list doesn't matter.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Introducing the Model Layer

The model layer of an app is the layer that contains and manages this information. For more information on the different layers of an app's architecture, refer to the App Design video presented in Activity 1.3 The Germ Guide App. Make sure to review all three layers with students.

Discussion: Lead a discussion with students about apps they are familiar with and how these apps process and maintain data. For example:

- A social media app processes the user's list of friends.
- A shopping app processes the user's information and information from a supplier's inventory database.
- A game app processes lists of characters and high scores.
- A notes app processes a list of user-entered information.
- A step counter app processes the data received from a sensor.

Understanding Lists

Instruct students to take notes as they watch the Introduction to Lists video.

Discussion: Lead a discussion about what was introduced in the video. Ask students what type of list they will use in their charades app, a static list or a dynamic list? Have students discuss their answer.

- In the case of the charades app, because the words or phrases will be prebuilt into the program, it is considered a static list.
- Students will explore and use dynamic lists in [Activity 2.4 Disease Tracker](#).

Creating a List

- The App Inventor website provides [List blocks information](#) that details each block available and how each block works.
- Remind students that any data, or information, that you want to have access to in a program needs to be stored in a variable. To create a word/phrase list in their app, they first need to create a variable for this list.

Shaking Things Up

- Because the emulator cannot sense shaking, direct students to use the mobile device to test the shaking behavior for this section.
- Review the purpose of the different property values for the accelerometer sensor with students, as shown below.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Component Name	Property	Value
AccelerometerSensor1	Enabled	checked
	MinimumInterval (ms)	2000
	Sensitivity	moderate

Figure 1. Accelerometer Properties

Note: Direct students to review the code they built in their Germ Fighter app for ideas for how to increment the score.

- Have students identify what functionality has not been checked off in their event handler charts. They should identify that they still have to program the following behaviors:
 - Display a new random item from the wordPhraseList when the device senses shaking.
 - Reset the timer every time the app advances to a new word or phrase.
 - Advance to a new word or phrase whenever the timer gets to 0.
 - Advance to a new word or phrase whenever the user touches the Skip button.

Playing Charades: If time permits, students may want to use their charades app to play a couple of rounds of the game. If you allow students to do this, make sure that students conduct themselves in a safe and respectful manner. Consider having students swap tablets so that they can play each other's games, where they don't know the items in the list of each other's apps.

Have students determine the rules of the game before play, such as:

Are players allowed to hold up their fingers to indicate the number of syllables in the answer?

Are players allowed to silently reply to questions?

Can players make gestures to indicate that a guess is close?

Can players point at or tug on an earlobe to indicate that the word "sounds like"?

If there are students in your class who are unable to do the physical acting, students may use the app to play Pictionary™ (where players draw the words or phrases) or Catch Phrase™ (where players give verbal hints of the words or phrases).

Housekeeping Challenge

If you are running out of time and students cannot complete this optional challenge independently, you may show students the solution code and work them through the logic or how this procedure would work.



Activity 2.4

Disease Tracker

Teacher Resources

[View Student Course](#)

KNOWLEDGE AND SKILLS

KS

KS 2.1A1 Understand that computers enable rapid processing of information.

KS 2.1A2 Collect, process, and interpret data to gain insight on a problem and draw conclusions.

ESSENTIAL QUESTIONS

EQ

EQ 2.1 What do programming best practices look like?

EQ 2.2 How do diverse perspectives of team members impact a solution?

RESOURCES

-  [A2_4_DiseaseTracker_TR.aia](#)
-  [A2_4_DiseaseTracker_TR_withChallenges.aia](#)
-  [2.4 Disease Tracker App Design Requirements](#)
-  [Activity 2.4 Answer Key](#)



Preparation

Students will use pair programming to collaboratively design and build their apps. For more information on facilitating pair programming in your classroom, refer to [Activity 1.6 Teacher Resources](#).

Activity Time: 3 Days

Possible Student Misconceptions

- Variables are used in computer programming the same way they are used in mathematics.
- Variables can only be numeric.
- Variables can only hold one value at a time.
- Values in lists are fixed and cannot be changed.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Data Collection

- You can access flu activity surveillance data from the CDC at: <https://www.cdc.gov/flu/weekly/fluactivitysurv.htm>
- Consider showing students this data and relating their work on this app to what is happening in the real world.
- Additional information on how the CDC monitors flu data is available at: <https://www.cdc.gov/flu/about/qa/disease.htm>.

The Disease Tracker App

- Students will refer to the design requirements frequently and check them off as they complete their development.
- Remind students of pair programming practices. Both students should answer questions and draw flowcharts.

View Layer

- In step 4, students might be able to guess what the new components are used for or might read the description of the components in the *Palette*.
- Determine where you want students to deploy the app to—the emulator or a mobile device.

Dynamic Lists

Discussion: Watch the Dynamic Lists video as a class or have students view the video individually. Use the questions in the activity to discuss what students learned from the video.

Controller and Model Layers

- The purpose of using the event handler chart is to bridge the students' understanding between the app requirements presented earlier in the activity and what needs to be done in the program to meet these requirements.
- Give students time to complete the Add and the Display Data algorithms.
- For the flowcharts, students do not have to include every single line of the potential code. They should focus on the big-picture tasks. You may choose to lead a class brainstorm to get students collaborating on identifying the big picture tasks.

Lists

- Remind students that there are multiple ways to solve the same problem in computing. This activity uses two separate lists to maintain the data for the months and the amounts. Another way to approach this is to use a list of lists but that could be too confusing for middle school students!
- Remind students that variables are placeholders in a device or computer's memory used to store data in a program.
- Explain to the students that the length of the list will change during the execution of the app because the user will be adding values to both lists (months and amounts) with the data entry process.

Add Button

Tell students that in computing, we refer to the values we set on a block as "parameters". These values are said to be "passed in" to the block for it to use during execution. Examples of parameters in App Inventor are the values in the right-most column of the blocks table.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Hint: An easy way to drag a "get" or "set" block for a variable is to:

- Hover your mouse over the variable name in the initialization block.
- Then drag the "get" or "set" block from the pop-up list.

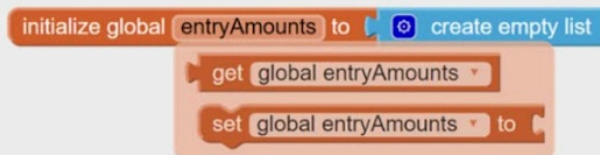


Figure 1. Hover Mouse to Display Options

Show Data Button

- Reinforce the concept of abstraction using procedures. Someone reading the code will understand what the *ShowDataButton.Click* event handler does very easily, because all of the blocks in the event handler are named according to their function. The exact details of how each procedure will execute is hidden, or abstracted.
- Remind students that procedure names, like variable names, should be descriptive and written in camelcase with a lowercase first letter.
- A "procedure call" is the phrase used when a block of code calls a procedure, such as *call displaySum*.

Creating the displaySum Procedure

- Note that algorithms can be solved in different ways. The solution presented is just one of the ways, and it is the one used in this activity.
- Make sure to stop the class and go through the flowchart in this section to ensure students really understand how the average is going to be calculated.

Example: You can help them visualize this procedure by picturing the variable *sum* as a container that starts out with nothing (0). Then you loop through the items in the list, and you add the value of each of the items to the cumulative sum. If they need further explanation, share the following example with the students:

If the list *entryAmounts* contains four values:
[50, 100, 80, 40]

Then *sum* starts out equal to 0, but becomes the following after each iteration of the loop.

Initial value: $\text{sum} = 0$
Iteration 1: $\text{sum} = \text{sum} + 50 = 0 + 50 = 50$
Iteration 2: $\text{sum} = \text{sum} + 100 = 50 + 100 = 150$
Iteration 3: $\text{sum} = \text{sum} + 80 = 150 + 80 = 230$
Iteration 4: $\text{sum} = \text{sum} + 40 = 230 + 40 = 270$

- Remind students that they used another loop in Activity 2.1 Keep Me in the Loop, which was the *for each number from to*. That loop was great for repeating a task a specific number of times based on the range identified in the loop header.
- Note that the *for each number from to* loop could have been used here as well, but the *for each item in list* loop is much more appropriate for traversing a list without having to deal with numbers and indexes.

Creating the displayAverage Procedure

- The sum of all of the items is equal to the global *sum* variable. The total number of items is the same as the length of the *entryAmounts* list, which is a built-in block available in the *Lists* drawer.

Creating the displayDataList Procedure

- Discuss with the students the benefit of using the *for each number from to* loop and make sure they understand the explanation provided in step 33 about the benefits of the two *for each* loops. The point of using the *for each number* loop is to use the number variable in the loop header as the index to access the items in both lists at the same index, since they correspond to one another (month and amount).
- Explain to students that renaming the variable in the loop header is optional and does not affect the functionality of the loop. However, because the term *index* is commonly used in computing to reference an item in a list, it is good practice to rename the variable in the loop header to "index"



since it will be used to access the items in the lists.

Important: It is important for students to understand that finding and fixing errors is normal and expected when programming. It requires patience, persistence, and iteration (many trials!).

Housekeeping Challenges

The Housekeeping Challenges are optional, if time permits.

Delete Challenge

The Delete Challenge is optional, if time permits.

Direct the students to explore the blocks in the *displayList* drawer. They can hover over each blocks to get a description of the block.



Activity 2.4 Extension

Persistent Data

Teacher Resources

[View Student Course](#)

KNOWLEDGE AND SKILLS

KS

KS 2.1B4 Store, access, and update data stores in a database or file (persistent data).

RESOURCES

-  [A2_4_Ext_DiseaseTrackerPersistentData_TR_withHousekeeping.aia](#)
-  [A2_4_Ext_DiseaseTrackerPersistentData_TR_withMedia.aia](#)



Activity Time: 1 Day

Possible Student Misconceptions

- Available data storage is unlimited.
- Data stored in variables is available the next time the app is run.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Big Data

This is an optional extension activity for students to explore if time permits. It is highly recommended to do this activity, because not only is *TinyDB* useful for storing information permanently, but it is also a great way to share data across multiple screens of an app. Refer to the solution code for the media challenge. It uses *TinyDB* to store the media objects and contains two screens.

All notes about this section are included in the student course. Solution code is provided in the Resources section of this page.

Media Challenge

Solution code for this activity including the Media Challenge is available in [A2_4Ext_DiseaseTracker_PersistentData_TR_withMedia.aia](#).

Housekeeping Challenges

Solution code for this extension activity including the housekeeping code is available in [A2_4Ext_DiseaseTracker_PersistentData_TR_withHouse-keeping.aia](#).



Figure 1. Code to Prevent the Runtime Error

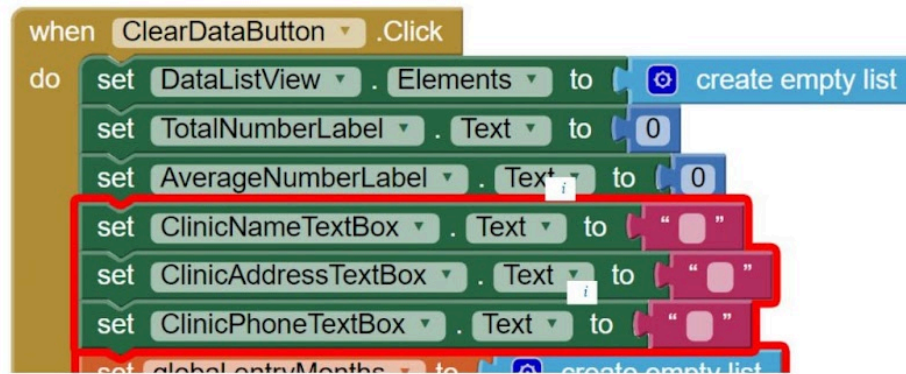


Figure 2. Clear the Data Lists



Project 2.5

Fitness Challenge App

Teacher Resources

[View Student Course](#)

ESSENTIAL QUESTIONS

EQ

EQ 2.1 What do programming best practices look like?

EQ 2.2 How do diverse perspectives of team members impact a solution?

RESOURCES

-  [A2_5_FitnessChallenge_TR.aia](#)
-  [A2_5_FitnessChallenge_TR_withExtensions.aia](#)
-  [2.5 Fitness Challenge App Rubric](#)
-  [2.5 Peer Feedback Form](#)
-  [Project 2.5 Answer Key](#)
-  [Project 2.5 Sample Solution](#)
-  [2.5 Fitness App Challenge Requirements](#)



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Preparation

- Print a [Project 2.5 Peer Feedback Form](#) for every team of students.
- Review the [Project 2.5 Sample Solution](#) to acclimate yourself with one way to incrementally build an app solution that meets the given design requirements/criteria for this project. Note that this sample solution uses the Disease Tracker app as starter code.
- Students will use pair programming to collaboratively design and build their apps. Determine how you will group the students. For more information on facilitating pair programming in your classroom, refer to [Activity 1.6 Teacher Resources](#).
- Review the [Project 2.5 Fitness Challenge App Rubric](#) and determine how you will use the rubric to assess student work and collaboration. Review the PD resources on rubrics.

Safety Considerations

If students are using online photo repositories for their challenge activities, make sure that the sites they visit are safe and appropriate.

Ethical Considerations

In this app, students will ask their users to enter some basic, personal demographic information about themselves, such as name and age. Any information that can be used to potentially identify a specific individual is considered personally identifiable information (PII) and needs to be protected. PII can include information such as a person's name, address, school name, e-mail address, social security number, telephone number, picture, and birth date. Lead a discussion with students about personally identifiable information, and the importance of protecting this type of information.

The data collected for this app does not need to be protected, because the information will not go anywhere, and in most cases, won't even be saved in the device's memory. However, it is important to use this opportunity to discuss with students the importance of protecting users' personal information in the digital age. Make sure students understand that whenever app developers and publishers create apps that require personally identifiable information, they need to follow proper protocols and any data security and privacy laws that are in place to protect people—for both confidentiality and security reasons. For example, there are laws designed to give patients specific rights regarding their personal health information, there are laws used to prosecute hackers when personal information has been stolen, and there are even laws that protect students' privacy in regard to any PII available in education records.

This is also a good opportunity to discuss the importance of students being safe when online. Students should be careful whenever posting any personally identifiable information in a public place, such as online and in social media.

Student Accommodations

Students should include physical challenges in their apps that are interesting and appropriate for themselves. If there are students in your class who are limited in their physical abilities, those students should include in their apps only physical activity challenges that are appropriate to their own abilities. For example, a student with a physical impairment that affects their legs might include all activities that can be done sitting down. If any students are unable to do physical activities at all, they may alter the app requirements to include nonphysical activities that are interesting to them, such as reading a book or doing an art project.

Project Time: 6 Days

Possible Student Misconception

- I have to be an expert programmer to successfully develop an app.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Staying Healthy

This project is estimated to take six instructional days. It is expected that students will complete certain steps of the design process for their app solutions on each day. Monitor the teams' progress throughout. If students are taking longer than expected, develop a plan to help get them back on track to meet the deadline. This may mean working outside of class to catch up or providing the teams helpful hints to move their work along.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Day	Milestones
1	Read and discuss introductory information Complete Design Process Step 1: Define the Problem/Step 2 Generate Concepts Complete Design Process Step 3: Design a Solution – View Layer
2	Complete Design Process Step 3: Design a Solution – Controller Layer
3–5	Complete Design Process Step 4: Build and Test
6	Complete Design Process Step 5: Evaluate the Solution Complete Design Process Step 6: Present the Solution

Background Information: Chronic Diseases

[Show](#)

Throughout the project, students will be guided through each step of the design process with prompts and tasks to help them complete each step. Students should document all work on each step in their PLTW Gateway Notebooks. Review the following steps of the design process with your students.

- 1. Define the problem.** As part of this step, designers validate the problem or opportunity by asking questions, such as:
 - Who says it's a problem?
 - What are the needs and wants?
 - Are there any prior solutions to this problem?
- 2. Generate concepts.** When generating concepts, designers research past solutions and brainstorm new ideas. When brainstorming, designers capture all the ideas they can think of without critiquing each other.
- 3. Design a solution.** Designers create a detailed design using technical drawings and models. The drawings and models provide the technical information necessary to produce a solution.
- 4. Build and test.** Designers build and test their solution. They test its performance, usability, and durability, and they collect and analyze the test data.
- 5. Evaluate the solution.** Designers test the solution's effectiveness, reflect on their design, and recommend any potential improvements. They will go back to step 4 if needed to improve the solution.
- 6. Present the solution.** Present the final solution to the stakeholders and clients.

Define the Problem

In this step of the design process, students read a design brief for the problem that they are solving.

- Give students time to think about the problem and design statement for this project. Ask students to think about how they can apply what they have learned and done so far to create an app to solve this problem.
- Review the design requirements/criteria and deliverables with students. Have students review the rubric for this project. Go through each category of the rubric and discuss the expectations with students. For example, remind students that they must document each step of the design process in their PLTW Gateway Notebooks, include all of the required design requirements/criteria in their apps, and use programming best practices—such as adding comments to their code and not using redundant code in their program, because all of these categories will be assessed using the rubric. Direct students to periodically refer to the rubric as they work, to gauge their own progress. For your own information, review the PD resources on rubrics.

Design Brief

- For the display of activity challenges design requirements, students have to include a picture, an audio file, and/or a video demonstrating how to complete the given challenge. Students can use their tablets to record the audio files or videos. For challenge images, students can use online photo repositories; make sure they cite the source of their image in their documentation. Students may want to spend a lot of time creating these media assets, so be sure to help them with their time management and don't allow them to waste too much time. Consider requiring students to complete their code before they begin work on these media elements or work on their media creation outside of class. If students create videos, make sure that the videos are very short—less than five seconds long, because App Inventor won't upload larger files.
- For demographic information, students are discouraged from asking for height, weight, and gender, because they are questions that can be sensitive for some students. Make sure students are respectful and do not use these questions.

Design a Solution

View Layer



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Review the prompts in step 3 with students. As students work on this step of the design process, circulate around the room and make sure students are using these prompts to guide their design work. When they design the UI for their apps, remind students to use user-centered design practices, where they design the solution with the end user in mind and find ways to improve the user experience.

Controller Layer

Students should think about the code they created in both the Charades and Disease Tracker apps to see whether there is any code that they want to reuse or modify for this app. Students can go into their saved projects (in My projects) in App Inventor to get ideas. Students can also refer to any flowcharts presented in earlier activities to help with their algorithm planning, specifically the algorithm flowcharts presented in the Disease Tracker activity.

Build and Test

Pair Programming. Students should switch Driver and Navigator roles often. Direct students to switch roles every time they begin to program a new event/event handler or design requirement. This should allow students to perform both roles at least once during a class period.

Remind students to review the collaboration criteria in the rubric as they use pair programming to build and test their apps. You may consider using this section of the rubric as a peer evaluation tool to help students informally assess and refine their collaboration skills.

Starter Code. Students may choose to use their Disease Tracker apps as starter code for this app, as many of the design requirements for that app are similar to the design requirements for this app. This is okay and can even be encouraged. Computer scientists often look to adapt or improve existing code, because there is no reason to duplicate work when it isn't necessary.

Suggestions for Struggling Students:

- If students are struggling with their code, offer hints, but do not provide answers. Ask students to think about the event handlers they are using and the blocks that will accomplish the behavior that they desire. Guide students by providing support or resources, not by providing the solution to their problem. For example, prompt students to think about the behavior they're trying to create with their code and investigate all of the blocks available for the corresponding event handler, to get ideas. Prompt students to think about whether they've ever coded similar behaviors in previous apps, and how they handled those behaviors.
- If students' code is not working properly, direct them to review and use the debugging strategies they've learned.
- If students need help, they can refer to the General Programming Help and Hints resource.

As students are building their apps, circulate around the room to assess that students are using programming best practices and building their apps incrementally.

Evaluate the Solution

- Students should share their apps with another team for testing. The testers will determine whether the app functions as specified and provide the development team with written feedback on the Peer Feedback Form. Review this form with students so they know what kind of feedback they should provide to each other. This form isn't intended to be used for assessment, but as a useful tool to help guide students in providing constructive feedback on each other's work. You may choose to have teams simply discuss their feedback, using the prompts on the form, versus writing out their responses.
- If the apps are not functioning properly or need modifications, the developers can make necessary changes to update the apps.
- If time permits, students may add additional features to the apps and complete any additional housekeeping tasks.

Present the Solution

- Student apps and documentation will be assessed using the [Project 2.5 Fitness Challenge App Rubric](#). Consider having students use the rubric to do a self-assessment of their work and final product.
- Consider having students complete a gallery walk where they can go around the room and try out each other's apps. This will give them the chance to see how different teams approached the same problem in different ways.
- After submission, students should answer the reflection questions individually, and then they can discuss their responses as a class.

LESSON 2 INTERIM ASSESSMENT

The App Creators Lesson 2 Interim Assessment is located in the Unit Level Resources, Assessment Page. The assessment is a starting point for you to use with your students. Feel free to edit this assessment to best meet your needs and the needs of your students. Use the Lesson 2 Interim Assessment Answer Key, to evaluate student work.



SPRINGFIELD PUBLIC SCHOOLS
ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS



Lesson 3

The App Challenge – Quick Guide

Preface

Students apply all of the knowledge and skills they have acquired to design and create a mobile app solution for a personal or community problem. They apply the design process and computational thinking skills to decompose the problem into smaller modules. Following user-centered design principles, they design and create an appropriate user interface and program the app to produce the desired behavior.

Essential Questions

1. In what ways does technology affect people's lives?
2. How do you express yourself and your creativity through computer science?
3. How do diverse perspectives of team members impact a solution?

Activity, Project, or Problem	Days
Problem 3.1 The Great App Challenge	9
Unit Summative Assessment	1
Total	10 Days



Problem 3.1

The App Challenge

Teacher Resources

[View Student Course](#)

LEARNING OBJECTIVES

LO

- LO 1.1B Demonstrate creativity and courage to take risks in proposing designs.
- LO 1.2B Apply user-centered design principles when creating a solution.
- LO 3.4B Consider accessibility and equity when designing products, creating solutions, and collaborating with others.
- LO 3.5B Describe the role, connections between disciplines, and impact of engineering, biomedical science, and/or computer science on society.

KNOWLEDGE AND SKILLS

KS

- KS 1.1B1 Develop solutions employing nontraditional techniques; novel combinations of artifacts, tools, techniques; and exploration or personal curiosities throughout a creative process.
- KS 1.2A1 Identify a problem and justify the pursuit of a solution to the problem.
- KS 1.2B1 Investigate the types of interactions between users and a proposed solution.
- KS 1.2B2 Describe the importance of involving prospective users early and often during the design process.
- KS 1.2B3 Recognize that understanding human interaction with a solution can improve the design and extend the abilities of humans.
- KS 3.4A1 Describe property rights of digital media and software to facilitate sharing and avoid infringement.
- KS 3.4B1 Describe and give examples of universal design strategies that increase accessibility to appropriate resources.
- KS 3.5B1 Demonstrate personal responsibility and initiative for independent learning, keeping in mind that technology and processes will always evolve.

ESSENTIAL QUESTIONS

EQ

- EQ 1 In what ways does technology affect people's lives?
- EQ 3.1 How do you express yourself and your creativity through computer science?
- EQ 3.2 How do diverse perspectives of team members impact a solution?

RESOURCES

-  3.1 Peer Feedback Form
-  3.1 The Great App Challenge Rubric
-  3.1 Great App Challenge Programming Requirements



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Preparation

- Print a [Problem 3.1 Peer Feedback Form](#) for every team of students.
- Students will use pair programming to collaboratively design and build their apps. Determine how you will pair the students. You might consider not only grouping students by ability, but also consider grouping students by common interests. Refer to [Activity 1.6 Teacher Resources](#) for more information on facilitating pair programming in your classroom.
- Review the [Problem 3.1 The Great App Challenge Rubric](#) and determine how you will use the rubric to assess student work and collaboration. Review the PD resources on rubrics.
- If you want to give the option of students packaging up and installing their apps on their devices, note that the devices need to be set up to allow for the installation of an app from an unknown source. Refer to the [App Inventor instructions](#) under Sharing your app for others to install on their phone/tablet (.apk file).
- Review the options in the Present the Solution section. If you are consider inviting members of the school administration or hosting an App Fair, make sure to plan accordingly.

Safety Considerations

If students are doing any research or using online photo repositories, make sure that the sites they visit are safe and appropriate.

Time: 9 Days

Possible Student Misconceptions

- You have to be an expert programmer to successfully develop an app.
- Computer science can only be used to solve global problems and not those in your community.
- When creating code, it will function properly the first time tested.
- Coding is best done as an individual.
- Pair programming is a "second-best" strategy for when there aren't enough computers.
- There is only one possible algorithm to solve a problem.
- You cannot learn from failure; persistence is a waste of time.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Problem Introduction

This project is estimated to take nine instructional days. Students are expected to complete certain steps of the design process for their app solutions on each day. Monitor teams' progress throughout. If students are taking longer than expected, develop a plan to help get them back on track to meet the deadline. This may mean working outside of class to catch up or providing the groups with helpful hints to move their work along. Consider what works best for your classroom and your students.

Facilitate students' learning throughout their iterative design process during this problem. However, the purpose of this unit problem is to provide an open-ended problem where students can demonstrate the knowledge and skills they've obtained throughout this unit. While you may provide students with some prompting questions during their work, you might refrain from giving the same level of facilitation and support you did during the Fitness Challenge App project. Use this unit problem to comprehensively assess your students' learning.

Day(s)	Milestones
1–2	Read and discuss and introductory information Complete Design Process Step 1: Define the Problem Complete Design Process Step 2: Generate Concepts
3	Complete Design Process Step 3: Design a Solution
4–7	Complete Design Process Step 4: Build and Test
8	Complete Design Process Step 5: Evaluate the Solution
9	Complete Design Process Step 6: Present the Solution

- Consider having students sign each other's PLTW Gateway Notebooks as a pseudo contract to hold them accountable for upholding their team norms.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

- Review the rubric and determine whether you will assess students on the Presentation criterion. This section of the rubric is optional; whether you assess students on their presentation skills is up to your discretion. To help you determine the format you will expect for student presentations and determine whether you will assess students on their presentations, review the Present the Solution section below about options for student presentations for this problem.
- Direct students to periodically refer to the rubric to gauge their own progress as they complete their work.
- Have students review the rubric for this project. Go through each category of the rubric and discuss the expectations with students. For example, remind students that they must document each step of the design process in their PLTW Gateway Notebooks, include all of the required design requirements/criteria in their apps, and use programming best practices—such as adding comments to their code and not having any redundant code in their program—as all of these categories will be assessed using the rubric. Make sure students understand whether they will be assessed on the Presentation criterion described in the rubric.

Design Process

Have students document all ideas for each stage of the design process in their PLTW Gateway Notebooks. Each team can present these responses as part of their final design solution to demonstrate their work. Students should individually write their team's responses and show their work in their own PLTW Gateway Notebooks.

Throughout the project, students will be guided through each step of the design process and given prompts and tasks to help them complete each step. Students should document all work on each step in their PLTW Gateway Notebooks. Students will follow the design process steps to guide their work through this problem. Each stage of the process is outlined in the APB headings. Review the following steps of the design process with your students.

1. **Define the problem.** As part of this step, designers validate the problem or opportunity by asking questions, such as:
 - Who says it's a problem?
 - What are the needs and wants?
 - Are there any prior solutions to this problem?
2. **Generate concepts.** When generating concepts, designers research past solutions and brainstorm new ideas. When brainstorming, designers capture all the ideas they can think of without critiquing each other.
3. **Design a solution.** Designers create a detailed design using technical drawings and models. The drawings and models provide the technical information necessary to produce a solution.
4. **Build and test.** Designers build and test their solution. They test its performance, usability, and durability, and they collect and analyze the test data.
5. **Evaluate the solution.** Designers test the solution's effectiveness, reflect on their design, and recommend any potential improvements. They will go back to step 4 if needed to improve the solution.
6. **Present the solution.** Present the final solution to the stakeholders and clients.

Define the Problem

In this step of the design process, students will identify and define the problem they will be solving. Make sure students know what is expected of them with this problem, including the minimum app programming requirements and expectations outlined in the rubric.

Ask students to think about how they can apply what they have learned and done so far in the unit to create an app to solve this problem.

For students who are stuck in identifying a problem they are interested in solving, the following are some ideas for apps for each problem category.

- 1 **Problem category: Personal interest**
 - An app that organizes your music
 - An app that tracks sports stats
 - An app that helps track chores, allowance, savings/spending, babysitting or dog-walking jobs
- 2 **Problem category: Health and wellness**
 - An app that displays the USDA's MyPlate graphic that depicts the recommended amounts of different food groups. Users can track the foods they're eating, and as they eat them, the items on the plate fill up until they meet the recommended amount.
 - An app that helps people find farmers markets in their area
 - A fitness or sports app
- 3 **Problem category: Environment**
 - An app that organizes a recycling project that maintains general information about recycling in a town, such as drop-off location, contact information, and the lists of items (by category) that are recyclable
 - An app where students can sign up for a cleanup day
 - An educational app about conservation (water, recycling, sustainable food and products)
- 4 **Problem category: Emergency preparedness**
 - An app that helps people create and track their inventory of emergency supplies for their home



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

- An educational app about different natural disasters that can occur in their area and how they can prepare for these disasters, such as with drills

5 Problem category: Community service

- An app where students can sign up to volunteer locally, such as at a senior center or a soup kitchen
- An app where students can track their volunteer work and include a description of each volunteer opportunity and the corresponding number of hours

6 Problem category: Education

- An app that helps preschool-aged students practice literacy or math skills. For example, it can include letter tracing or passages that are read aloud.
- Trivia or quiz game that can be used to help someone study for a test

7 Problem category: School culture

- An educational app that teaches about bullying
- An app that allows users to enter their interest and be given information about clubs or groups they can join that are connected with that interest.

Design Brief

The purpose of this modified design brief is to make sure that students take the time to clearly identify the end user of their app, the problem they are trying to solve with their app, briefly describe their justification for choosing this problem, and define the basic design requirements for their app.

For this learning experience, it isn't necessary for students to do research for their problem justification. At this level, it is important that students think about why they're choosing the problem they're choosing and can explain their reasoning. It is okay if their justification is as simple as that their problem is interesting to them and something that they personally feel passionate about.

Generate Concepts

- Facilitate team discussions as student pairs discuss the questions for each of their design solution ideas. Ensure students are discussing all possible ideas from each team member.
- Refer students to the minimum app programming requirements and their identified design requirements (from their Design Briefs), and review available resources as needed, if they have concerns or questions while considering each design idea.
- Observe students as they generate concepts and discuss their design solution ideas and answer questions about their designs. Facilitate their discussions as needed and note their collaboration in this early stage of the design process. These observations can be compared later to their teamwork abilities toward the end of the process, and can be assessed using the rubric.

Design a Solution

You may have students read the design considerations independently, in pair programming teams, or as a class. If your schedule allows it, you may assign this reading to students before you begin this problem, as an out-of-class or flipped-class activity.

View Layer

- Discuss with students the importance of designing apps that are safe for users and can secure their data. Universal design is not the same as an accommodation. Apps that are universally designed meet the accessibility needs of all users without having to make special accommodations for a specific group of users.
- Review the view layer prompts with students. As students are working on this step of the design process, circulate around the room and make sure students are using these prompts to guide their design work. When designing the UI for their apps, remind students to use user-centered design practices, where they are designing the solution with the end user in mind and finding ways to improve the user experience.

Controller / Model Layer

Students should think about the code they created throughout this unit to see whether there is any code that they want to reuse or modify for this app. Students can go into their saved projects (in *My projects*) in App Inventor to get ideas. Students can also refer to any flowcharts presented in earlier activities to help with their algorithm planning.

- **Event handler chart.** The purpose of the event handler chart is to help students determine what events need to be handled and what each event handler should include in the program to meet their design plans.
- **Flowcharts.** For this problem, to save on time, students are not required to create a flowchart for every event handler for their app. Students should identify any event handlers that are new or unfamiliar to them, or that they think might be hard to program, and should create the flowcharts for just these.

Build and Test



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Pair Programming. Students should switch Driver and Navigator roles often as they develop and test their app. Direct students to switch roles every time they begin to program a new event, event handler or design requirement. This should allow students to perform both roles at least once during a class period.

Remind students to review the collaboration criteria for the rubric as they use pair programming to build and test their apps. You may consider using this section of the rubric as a peer-evaluation tool to help students informally assess and refine their collaboration skills.

Starter Code. Students may choose to use an app they've previously developed as starter code for this app. This is okay. Computer scientists often look to adapt or improve existing code, because there is no reason to duplicate work when it isn't necessary. However, make sure to encourage student creativity. Do not allow students to simply take an app they've already created and make small tweaks. If they use another app as their starter code, they should make enough changes to the app to ensure their app solution is unique and of their own creation.

Note: As students are building their apps, circulate the room to assess that students are using programming best practices and are building their apps incrementally. Document your observations for later use when using the rubric to assess students' work products.

- If students are struggling with their code, offer hints, but do not provide answers. Ask students to think about the event handlers they are using and the blocks that will accomplish the behavior that they desire. Guide students by providing support or resources, not by providing the solution to their problem. For example, prompt students to think about the behavior they're trying to create with their code and investigate all of the blocks available for the corresponding event handler to get ideas. Prompt students to think about whether they've coded similar behaviors in previous apps, and how they handled those behaviors.
- Determine how you want students to cite any intellectual property that they are using. Consider any school policies and remind students to document their sources in the corresponding format in their PLTW Gateway Notebooks.
- If students' code isn't working properly, direct them to review and use the debugging strategies they've learned.
- If students need help, they can refer to the General Help and Hints resource in Problem 3.1. This page includes links to resources available on App Inventor's website that can help students learn new features they might want to include in their apps.

Evaluate the Solution

- Students should share their apps with another team for testing. The testers will determine whether the apps function as specified in the team's design brief and provide the development team with written feedback on the Peer Feedback Form. Review this form with students so they know what kind of feedback they should be providing to each other. This form isn't intended to be used for assessment, but as a useful tool to guide students in providing constructive feedback on each other's work. You may choose to have teams discuss their feedback, using the prompts on the form, versus writing out their responses.
- Additionally, consider having students share their apps with individuals outside of class to get feedback from various people.
- If the apps are not functioning properly or need modifications, the students can make necessary changes to update the apps.
- If time permits, students may add additional features to the apps and complete any additional User-Centered Design: Housekeeping tasks.
- Remind students that, like the debugging steps for fixing code, the design process is iterative. Encourage student teams to persevere and look for opportunities to improve their app rather than dwell on mistakes or issues that arise.

Present the Solution

- Student apps and documentation will be assessed using the [Problem 3.1 The Great App Challenge Rubric](#). Consider having students use the rubric to do a self-assessment of their work and final product.
- Consider the size of your class and the class time you have available for the presentations. Determine what level of detail you want to require students to include in their presentations, and determine the format you want students to use to present their apps. For example, students may use presentation software and formally present the apps to the audience, or you may have them create poster displays and present their apps as a gallery walk.

Presentation Options: You might have each team present to school administrators or members of the community. Alternatively you might host an App Fair where students can showcase their work for parents and members of the community. Students can display posters or create short videos to describe their design process for developing their app and showcase their final app solutions. Keep in mind that these presentation methods will take additional time to complete, but can be a wonderful opportunity to showcase students' work with your community and parents.

Note that there are a variety of app development student competitions available for students to enter, if they are interested.

Conclusion

- After students reflect on the two final questions, discuss them as a class. Ask students to share their thoughts and explain what they learned throughout the unit.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

UNIT SUMMATIVE ASSESSMENT

The App Creators Unit Summative Assessment is located in the Unit Level Resources, Assessment Page. Students will take the End-of-Unit Assessment. The assessment is a starting point for you to use with your students. Feel free to edit the assessment to best meet your needs and the needs of your students. Use the App Creators Summative Assessment Answer Key, to evaluate student work.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS



PLTW Curriculum Framework – App Creators (2017-2018)
Lesson 1: Let's Create an App!

Desired Results (stage 1)		
Meaning		
ESSENTIAL QUESTIONS: <i>Students will keep considering ...</i> - Q1 – In what ways does technology affect people's lives? - Q2 – How do computer scientists design and develop mobile apps that meet desired needs? - Q3 – What does effective teamwork look like?		
Meaning	Acquisition	
Domains/Understandings <i>Domains are key understandings and long-term takeaways that go beyond factual knowledge into broader and more conceptual comprehensions.</i> <i>Domains are areas of expertise that an employer in a specific field may seek.</i> <i>Students will develop understandings about...</i> "I will be able to address real-world challenges because I understand..." <i>Students will understand that ...</i>	Learning Objectives <i>Objectives articulate what skills students need to be able to do. (The learning objectives will become targets of assessment.)</i> <i>Objectives are functions that directly relate to the workplace or in an applied academic setting.</i> <i>Students will use knowledge and skills to...</i> "In the workplace or academic setting, I will need to know and be able to..."	
D1.1 Mindset: Ethics, analytical thinking, creativity, persistence, iteration, and the positive role of failure are important mindsets and habits of action. They are developed over time in problem solving processes, inquiry, and computational thinking.	LO1.1A: Describe and analyze moments within a problem solving process where persistence, iteration, and the positive aspect of failure played an important role in gaining	KS1.1A1: Recognize that identifying complex problems, defining them clearly, and proposing solutions can be difficult and requires persistence and iteration.

© 2017 Project Lead The Way, Inc.
 Curriculum Framework → PLTW Gateway → 1.1 – App Creators → Page 1

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

	understanding about a problem or unexpected observation.	
	LO1.1B: Demonstrate creativity and courage to take risks in proposing designs.	KS1.1B1: Develop solutions employing non-traditional techniques; novel combinations of artifacts, tools, techniques; and exploration of personal curiosities throughout a creative process.
D1.2 Problem Solving Process and/or Design Process: Many disciplines, including engineering, computer science, and biomedical science, use an iterative, systematic process to solve problems.	LO1.2A: Apply an iterative process to solve a problem or create an opportunity that can be justified.	KS1.2A2: Explain the goal of any design process. KS1.2A3: Generate ideas and/or build upon existing ideas to innovate. KS1.2A4: Iteratively test throughout the design process to ensure that a solution meets the requirements.
	LO1.2B: Apply user-centered design principles when creating a solution.	KS1.2B1: Investigate the types of interactions between users and a proposed solution. KS1.2B2: Describe the importance of involving prospective users early and often during the design process.
D1.3 Computational Thinking: Computational thinking is the thought processes involved in formulating problems and solutions that can be effectively carried out by a computer or electronic device. Common concepts of computational thinking include: the use of algorithms, abstraction, problem decomposition, and data analysis and processing.	LO1.3A: Apply computational thinking to solve problems.	KS1.3A1: Recognize that computational thinking can be applied in multiple disciplines. KS1.3A2: Choose appropriate computational practices when solving a problem. KS1.3A3: Decompose a problem into smaller parts.
D2.1 Data: With the aid of computational power, a tremendous quantity of data can quickly and efficiently processed and analyzed to help solve a problem.	LO2.1B: Create and store data during the execution of a program.	KS2.1B1: Describe ways data can be stored on a digital device. KS2.1B2: Store, access, and update data stored in variables (temporary data).
D2.2 Algorithms: A wide range of professionals use algorithms, a sequence of steps used to solve a problem. This can be accomplished with or without coding.	LO2.2A: Analyze and create algorithms.	KS2.2A1: Create an algorithm or sequence of steps to accomplish a task. KS2.2A2: Identify different algorithms that can be used to solve the same problem. KS2.2A3: Analyze, break down, and explain the logic of an algorithm.

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → 1.1 – App Creators → Page 2

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

		KS2.2A4: Locate and debug errors within an algorithm. KS2.2A5: Create simple algorithms that involve variables, conditionals, operators, or logic.
D2.3 Abstraction: An abstraction generalizes a concept, allowing people to ignore some of the details. Abstractions make it easier to handle complexity by focusing on the relevant pieces of the big picture and hiding the details.	LO2.3A: Develop and apply abstractions.	KS2.3A1: Explain the purpose of an abstraction.
D2.4 Programming: Creating a set of statements processed by an electronic device to perform a task.	LO2.4A: Analyze the structure and functionality of a program.	KS2.4A1: Identify and describe the high-level structures of a program, such as user interface components, data components, event handlers, and procedures. KS2.4A2: Describe the appropriate code blocks and convention used in the programming language. KS2.4A3: Identify possible events that can occur during runtime and select the appropriate event-handler blocks to respond to these events. KS2.4A4: Trace a program and deduce the values that variables will have after the code is executed.
	LO2.4B: Create programs by developing and testing code in a modular, incremental approach.	KS2.4B1: Plan a program using appropriate strategies such as natural language or flowcharting. KS2.4B2: Design and develop a program by breaking a large plan into smaller modules using procedures and event handlers (if applicable). KS2.4B3: Test code frequently as it is being developed. KS2.4B4: Debug programs using a variety of strategies to isolate and identify problems, including analyzing error messages, analyzing variable values line-by-line, generating output, or commenting out code.

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → 1.1 – App Creators → Page 3

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS
ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

	LO2.4C: Adapt or improve existing code. LO2.4D: Design and develop a mobile app solution.	KS2.4C1: Find code relevant to a problem and extend or apply it to a new purpose. KS2.4C2: Improve readability of code by creating or improving documentation, using descriptive variables and procedure names, and using comments. KS2.4D1: Describe the components of a mobile app. KS2.4D2: Navigate and use an app development software to design, develop and test a mobile app. KS2.4D3: Design an app's user interface using appropriate tools. KS2.4D4: Select and use appropriate features to meet the desired functionality of an app, such as user interface components, layout arrangements, media objects, drawing and animation, sensors, location awareness, accessing phone features, and data storage. KS2.4D5: Deploy an app using the emulator or a mobile device.
D2.5 Computer Systems: Effectively harnessing computing power requires an understanding of how computers work.	LO2.5A: Identify methods in which electronic devices communicate with each other.	KS2.5A1: Recognize various ways of transferring data from one electronic device to another, such as USB connection, WiFi, or Bluetooth. KS2.5A2: Use a systematic process to identify the source of a problem within individual and connected devices.
D2.6 Social Impacts of Computing: Computer science solutions have global impacts in economic, environmental, and societal contexts.	LO2.6A: Analyze the implications of computing in society.	KS2.6A1: Identify ways that computing has changed or is changing society. KS2.6A2: Give examples of technologies that assist or extend human capabilities. KS2.6A3: Describe possible positive and negative social impacts of computing.

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → 1.1 – App Creators → Page 4

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

D3.1 Collaboration: Effective collaboration requires an ability to function within teams and is often necessary for successful problem solving, experimentation or design work.	LO3.1A: Collaborate effectively on a diverse team.	KS3.1A1: Describe how diverse perspectives in collaboration typically produce the best results in a process. KS3.1A2: Apply team norms to encourage productivity and define how a team will function and measure its success. KS3.1A3: Recognize individual strengths when defining roles and responsibilities.
D3.2 Communication: Effectively communicating orally and in writing for a given audience is an essential skill for success in all fields.	LO3.2A: Communicate effectively for specific purposes and settings.	KS3.2A1: Communicate to meet the needs of the audience and be appropriate to the situation. KS3.2A2: Use accurate and appropriate terminology.
	LO3.2B: Document a process according to professional standards.	KS3.2B1: Present data and information through a variety of accepted means such as: graphs, charts, images, video, schematics, code, 3D models, or simulations.
D3.3 Project Management: The discipline of carefully projecting or planning, organizing, motivating and controlling resources to achieve specific goals and meet specific success criteria.	LO3.3A: Demonstrate the ability to manage multiple resources throughout a project.	KS3.3A1: Devise, prioritize, and execute a plan to solve a problem, including short-term and long-term objectives when appropriate. KS3.3A2: Manage digital files appropriately.
D3.4 Professionalism and Ethics: Successful computer scientists typically exhibit specific personal and professional characteristics that lend themselves to the creative, collaborative, and solution driven nature of the profession.	LO3.4A: Abide by professional, legal and ethical standards when using digital resources.	KS3.4A2: Describe the importance of protecting personal information.
D3.5 Career Awareness: It is important to prepare a flexible education plan that matches your interests, knowing that you can change or modify that plan as you discover more about career opportunities.	LO3.5A: Identify the variety of careers related to engineering, biomedical sciences, and/or computer science.	KS3.5A2: Explore a variety of careers related to engineering, biomedical sciences, or computer science.



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS



PLTW Curriculum Framework – App Creators (2017-2018)

Lesson 2: Taking it to the Next Level!

Desired Results (stage 1)		
Meaning		
ESSENTIAL QUESTIONS: <i>Students will keep considering ...</i> - Q1 – In what ways does technology affect people's lives? - Q2 – What do programming best practices look like? - Q3 – How do diverse perspectives of team members impact a solution?		
Meaning	Acquisition	
Domains/Understandings <i>Domains are key understandings and long-term takeaways that go beyond factual knowledge into broader and more conceptual comprehensions.</i> <i>Domains are areas of expertise that an employer in a specific field may seek.</i> <i>Students will develop understandings about...</i> "I will be able to address real-world challenges because I understand..." <i>Students will understand that ...</i>	Learning Objectives <i>Objectives articulate what skills students need to be able to do. (The learning objectives will become targets of assessment.)</i> <i>Objectives are functions that directly relate to the workplace or in an applied academic setting.</i> <i>Students will use knowledge and skills to...</i> "In the workplace or academic setting, I will need to know and be able to..."	Knowledge and Skills <i>Knowledge and skills include the essential facts and basic concepts that a student should know and be able to recall in order to perform the competency.</i> <i>Knowledge and skill statements are foundational to the performance of a skill.</i> <i>Student will know and be able to...</i> "After I learn the information, I will be able to use my knowledge and skills to..."
D1.1 Mindset: Ethics, analytical thinking, creativity, persistence, iteration, and the positive role of failure are important mindsets	LO1.1A: Describe and analyze moments within a problem solving process where persistence, iteration, and the positive aspect of failure	KS1.1A1: Recognize that identifying complex problems, defining them clearly, and proposing

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → Lesson 2 – App Creators → Page 1

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

and habits of action. They are developed over time in problem solving processes, inquiry, and computational thinking.	played an important role in gaining understanding about a problem or unexpected observation.	solutions can be difficult and requires persistence and iteration.
D1.2 Problem Solving Process and/or Design Process: Many disciplines, including engineering, computer science, and biomedical science, use an iterative, systematic process to solve problems.	LO1.2A: Apply an iterative process to solve a problem or create an opportunity that can be justified.	KS1.2A3: Generate ideas and/or build upon existing ideas to innovate. KS1.2A4: Iteratively test throughout the design process to ensure that a solution meets the requirements.
D1.3 Computational Thinking: Computational thinking is the thought processes involved in formulating problems and solutions that can be effectively carried out by a computer or electronic device. Common concepts of computational thinking include: the use of algorithms, abstraction, problem decomposition, and data analysis and processing.	LO1.3A: Apply computational thinking to solve problems.	KS1.3A1: Recognize that computational thinking can be applied in multiple disciplines. KS1.3A2: Choose appropriate computational practices when solving a problem. KS1.3A3: Decompose a problem into smaller parts.
D2.1 Data: With the aid of computational power, a tremendous quantity of data can quickly and efficiently processed and analyzed to help solve a problem.	LO2.1A: Collect, process, and analyze real or simulated data.	KS2.1A1: Understand that computers enable rapid processing of information. KS2.1A2: Collect, process and interpret data to gain insight on a problem and draw conclusions. KS2.1A3: Compare methods of data representation.
	LO2.1B: Create and store data during the execution of a program.	KS2.1B1: Describe ways data can be stored on a digital device. KS2.1B2: Store, access, and update data stored in variables (temporary data). KS2.1B3: Store, access, and update data stored in a list (temporary data). KS2.1B4: Store, access, and update data stored in a database or file (persistent data) [optional].
D2.2 Algorithms: A wide range of professionals use algorithms, a sequence of steps used to solve a problem. This can be accomplished with or without coding.	LO2.2A: Analyze and create algorithms.	KS2.2A2: Identify different algorithms that can be used to solve the same problem. KS2.2A3: Analyze, break down, and explain the logic of an algorithm. KS2.2A5: Create simple algorithms that involve variables, conditionals, operators, or logic.

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → Lesson 2 – App Creators → Page 2

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

D2.3 Abstraction: An abstraction generalizes a concept, allowing people to ignore some of the details. Abstractions make it easier to handle complexity by focusing on the relevant pieces of the big picture and hiding the details.	LO2.3A: Develop and apply abstractions.	KS2.2A6: Create algorithms to perform repetitive or reusable tasks. KS2.3A1: Explain the purpose of an abstraction. KS2.3A2: Identify what has been made more general by an abstraction and identify what details have been hidden or removed. KS2.3A3: Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks.
	LO2.4A: Analyze the structure and functionality of a program.	KS2.4A1: Identify and describe the high-level structures of a program, such as user interface components, data components, event handlers, and procedures. KS2.4A2: Describe the appropriate code blocks and convention used in the programming language. KS2.4A3: Identify possible events that can occur during runtime and select the appropriate event-handler blocks to respond to these events. KS2.4A4: Trace a program and deduce the values that variables will have after the code is executed.
D2.4 Programming: Creating a set of statements processed by an electronic device to perform a task.	LO2.4B: Create programs by developing and testing code in a modular, incremental approach.	KS2.4B1: Plan a program using appropriate strategies such as natural language or flowcharting. KS2.4B2: Design and develop a program by breaking a large plan into smaller modules using procedures and event handlers (if applicable). KS2.4B3: Test code frequently as it is being developed. KS2.4B4: Debug programs using a variety of strategies to isolate and identify problems, including analyzing error messages, analyzing variable values line-by-line, generating output, or commenting out code.

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → Lesson 2 – App Creators → Page 3

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

	LO2.4C: Adapt or improve existing code.	KS2.4C1: Find code relevant to a problem and extend or apply it to a new purpose. KS2.4C2: Improve readability of code by creating or improving documentation, using descriptive variables and procedure names, and using comments.
	LO2.4D: Design and develop a mobile app solution.	KS2.4D2: Navigate and use an app development software to design, develop and test a mobile app. KS2.4D3: Design an app's user interface using appropriate tools. KS2.4D4: Select and use appropriate features to meet the desired functionality of an app, such as user interface components, layout arrangements, media objects, drawing and animation, sensors, location awareness, accessing phone features, and data storage. KS2.4D5: Deploy an app using the emulator or a mobile device.
D3.1 Collaboration: Effective collaboration requires an ability to function within teams and is often necessary for successful problem solving, experimentation or design work.	LO3.1A: Collaborate effectively on a diverse team.	KS3.1A1: Describe how diverse perspectives in collaboration typically produce the best results in a process. KS3.1A2: Apply team norms to encourage productivity and define how a team will function and measure its success. KS3.1A3: Recognize individual strengths when defining roles and responsibilities.
D3.2 Communication: Effectively communicating orally and in writing for a given audience is an essential skill for success in all fields.	LO3.2A: Communicate effectively for specific purposes and settings.	KS3.2A1: Communicate to meet the needs of the audience and be appropriate to the situation. KS3.2A2: Use accurate and appropriate terminology.
	LO3.2B: Document a process according to professional standards.	KS3.2B1: Present data and information through a variety of accepted means such as: graphs, charts, images, video, schematics, code, 3D models, or simulations.

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → Lesson 2 – App Creators → Page 4

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

D3.3 Project Management: The discipline of carefully projecting or planning, organizing, motivating and controlling resources to achieve specific goals and meet specific success criteria.	LO3.3A: Demonstrate the ability to manage multiple resources throughout a project.	KS3.3A1: Devise, prioritize, and execute a plan to solve a problem, including short-term and long-term objectives when appropriate. KS3.3A2: Manage digital files appropriately.
D3.4 Professionalism and Ethics: Successful computer scientists typically exhibit specific personal and professional characteristics that lend themselves to the creative, collaborative, and solution driven nature of the profession.	LO3.4A: Abide by professional, legal and ethical standards when using digital resources.	KS3.4A2: Describe the importance of protecting personal information.

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → Lesson 2 – App Creators → Page 5
Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS



PLTW Curriculum Framework – App Creators (2017-2018)

Lesson 3: Great App Challenge

Desired Results (stage 1)		
Meaning		
ESSENTIAL QUESTIONS: <i>Students will keep considering ...</i> - Q1 – In what ways does technology affect people's lives? - Q2 – How do you express yourself and your creativity through computer science? - Q3 – How do diverse perspectives of team members impact a solution?		
Meaning	Acquisition	
Domains/Understandings <i>Domains are key understandings and long-term takeaways that go beyond factual knowledge into broader and more conceptual comprehensions.</i> <i>Domains are areas of expertise that an employer in a specific field may seek.</i> <i>Students will develop understandings about...</i> “I will be able to address real-world challenges because I understand...” <i>Students will understand that ...</i>	Learning Objectives <i>Objectives articulate what skills students need to be able to do. (The learning objectives will become targets of assessment.)</i> <i>Objectives are functions that directly relate to the workplace or in an applied academic setting.</i> <i>Students will use knowledge and skills to...</i> “In the workplace or academic setting, I will need to know and be able to...”	Knowledge and Skills <i>Knowledge and skills include the essential facts and basic concepts that a student should know and be able to recall in order to perform the competency.</i> <i>Knowledge and skill statements are foundational to the performance of a skill.</i> <i>Student will know and be able to...</i> “After I learn the information, I will be able to use my knowledge and skills to...”
D1.1 Mindset: Ethics, analytical thinking, creativity, persistence, iteration, and the positive role of failure are important mindsets and habits of action. They are developed over time in problem solving processes, inquiry, and computational thinking.	LO1.1A: Describe and analyze moments within a problem solving process where persistence, iteration, and the positive aspect of failure played an important role in gaining	KS1.1A1: Recognize that identifying complex problems, defining them clearly, and proposing solutions can be difficult and requires persistence and iteration.

© 2017 Project Lead The Way, Inc.
 Curriculum Framework → PLTW Gateway → Lesson 3 – App Creators → Page 1
 Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

	understanding about a problem or unexpected observation.	
	LO1.1B: Demonstrate creativity and courage to take risks in proposing designs.	KS1.1B1: Develop solutions employing non-traditional techniques; novel combinations of artifacts, tools, techniques; and exploration of personal curiosities throughout a creative process.
D1.2 Problem Solving Process and/or Design Process: Many disciplines, including engineering, computer science, and biomedical science, use an iterative, systematic process to solve problems.	LO1.2A: Apply an iterative process to solve a problem or create an opportunity that can be justified.	KS1.2A1: Identify a problem and justify the pursuit of a solution to the problem. KS1.2A2: Explain the goal of any design process. KS1.2A3: Generate ideas and/or build upon existing ideas to innovate. KS1.2A4: Iteratively test throughout the design process to ensure that a solution meets the requirements.
	LO1.2B: Apply user-centered design principles when creating a solution.	KS1.2B1: Investigate the types of interactions between users and a proposed solution. KS1.2B2: Describe the importance of involving prospective users early and often during the design process. KS1.2B3: Recognize that understanding human interaction with a solution can improve the design and extend the abilities of humans.
D1.3 Computational Thinking: Computational thinking is the thought processes involved in formulating problems and solutions that can be effectively carried out by a computer or electronic device. Common concepts of computational thinking include: the use of algorithms, abstraction, problem decomposition, and data analysis and processing.	LO1.3A: Apply computational thinking to solve problems.	KS1.3A1: Recognize that computational thinking can be applied in multiple disciplines. KS1.3A2: Choose appropriate computational practices when solving a problem. KS1.3A3: Decompose a problem into smaller parts.
D2.1 Data: With the aid of computational power, a tremendous quantity of data can quickly and efficiently processed and analyzed to help solve a problem.	LO2.1B: Create and store data during the execution of a program.	KS2.1B2: Store, access, and update data stored in variables (temporary data). KS2.1B3: Store, access, and update data stored in a list (temporary data).

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → Lesson 3 – App Creators → Page 2

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

		KS2.1B4: Store, access, and update data stored in a database or file (persistent data) [optional].
D2.2 Algorithms: A wide range of professionals use algorithms, a sequence of steps used to solve a problem. This can be accomplished with or without coding.	LO2.2A: Analyze and create algorithms.	KS2.2A1: Create an algorithm or sequence of steps to accomplish a task. KS2.2A3: Analyze, break down, and explain the logic of an algorithm. KS2.2A4: Locate and debug errors within an algorithm. KS2.2A5: Create simple algorithms that involve variables, conditionals, operators, or logic. KS2.2A6: Create algorithms to perform repetitive or reusable tasks.
D2.3 Abstraction: An abstraction generalizes a concept, allowing people to ignore some of the details. Abstractions make it easier to handle complexity by focusing on the relevant pieces of the big picture and hiding the details.	LO2.3A: Develop and apply abstractions.	KS2.3A3: Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks.
D2.4 Programming: Creating a set of statements processed by an electronic device to perform a task.	LO2.4A: Analyze the structure and functionality of a program.	KS2.4A1: Identify and describe the high-level structures of a program, such as user interface components, data components, event handlers, and procedures. KS2.4A3: Identify possible events that can occur during runtime and select the appropriate event-handler blocks to respond to these events.
	LO2.4B: Create programs by developing and testing code in a modular, incremental approach.	KS2.4B1: Plan a program using appropriate strategies such as natural language or flowcharting. KS2.4B2: Design and develop a program by breaking a large plan into smaller modules using procedures and event handlers (if applicable). KS2.4B3: Test code frequently as it is being developed. KS2.4B4: Debug programs using a variety of strategies to isolate and identify problems, including analyzing error messages, analyzing

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → Lesson 3 – App Creators → Page 3

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

		variable values line-by-line, generating output, or commenting out code.
	LO2.4C: Adapt or improve existing code.	KS2.4C1: Find code relevant to a problem and extend or apply it to a new purpose. KS2.4C2: Improve readability of code by creating or improving documentation, using descriptive variables and procedure names, and using comments.
	LO2.4D: Design and develop a mobile app solution.	KS2.4D2: Navigate and use an app development software to design, develop and test a mobile app. KS2.4D3: Design an app's user interface using appropriate tools. KS2.4D4: Select and use appropriate features to meet the desired functionality of an app, such as user interface components, layout arrangements, media objects, drawing and animation, sensors, location awareness, accessing phone features, and data storage. KS2.4D5: Deploy an app using the emulator or a mobile device.
D2.5 Computer Systems: Effectively harnessing computing power requires an understanding of how computers work.	LO2.5A: Identify methods in which electronic devices communicate with each other.	KS2.5A1: Recognize various ways of transferring data from one electronic device to another, such as USB connection, WiFi, or Bluetooth. KS2.5A2: Use a systematic process to identify the source of a problem within individual and connected devices.
D3.1 Collaboration: Effective collaboration requires an ability to function within teams and is often necessary for successful problem solving, experimentation or design work.	LO3.1A: Collaborate effectively on a diverse team.	KS3.1A1: Describe how diverse perspectives in collaboration typically produce the best results in a process. KS3.1A2: Apply team norms to encourage productivity and define how a team will function and measure its success. KS3.1A3: Recognize individual strengths when defining roles and responsibilities.

© 2017 Project Lead The Way, Inc.
Curriculum Framework → PLTW Gateway → Lesson 3 – App Creators → Page 4

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

D3.2 Communication: Effectively communicating orally and in writing for a given audience is an essential skill for success in all fields.	LO3.2A: Communicate effectively for specific purposes and settings.	KS3.2A1: Communicate to meet the needs of the audience and be appropriate to the situation. KS3.2A2: Use accurate and appropriate terminology.
	LO3.2B: Document a process according to professional standards.	KS3.2B1: Present data and information through a variety of accepted means such as: graphs, charts, images, video, schematics, code, 3D models, or simulations.
D3.3 Project Management: The discipline of carefully projecting or planning, organizing, motivating and controlling resources to achieve specific goals and meet specific success criteria.	LO3.3A: Demonstrate the ability to manage multiple resources throughout a project.	KS3.3A1: Devise, prioritize, and execute a plan to solve a problem, including short-term and long-term objectives when appropriate. KS3.3A2: Manage digital files appropriately.
D3.4 Professionalism and Ethics: Successful computer scientists typically exhibit specific personal and professional characteristics that lend themselves to the creative, collaborative, and solution driven nature of the profession.	LO3.4A: Abide by professional, legal and ethical standards when using digital resources.	KS3.4A1: Describe property rights of digital media and software to facilitate sharing and avoid infringement. KS3.4A2: Describe the importance of protecting personal information.
	LO3.4B: Consider accessibility and equity when designing products, creating solutions, and collaborating with others.	KS3.4B1: Describe and give examples of universal design strategies that increase accessibility to appropriate resources.
D3.5 Career Awareness: It is important to prepare a flexible education plan that matches your interests, knowing that you can change or modify that plan as you discover more about career opportunities.	LO3.5B: Describe the role, connections between disciplines, and impact of engineering, biomedical science, and/or computer science on society.	KS3.5B1: Demonstrate personal responsibility and initiative for independent learning, keeping in mind that technology and processes will always evolve.

© 2017 Project Lead The Way, Inc.
 Curriculum Framework → PLTW Gateway → Lesson 3 – App Creators → Page 5

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria, VA: ASCD



SPRINGFIELD PUBLIC SCHOOLS

**ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY
GLOBAL SKILLS**

IEP AND 504 ACCOMMODATIONS AND MODIFICATIONS FOR EARLY CHILDHOOD AND ELEMENTARY SCHOOL STUDENTS:

Large print textbooks
Textbooks for at-home use
Additional time for assignments
Review of directions
Use of mnemonics
Have student restate information
Concrete examples
Adaptive writing utensils
Support auditory presentations with visuals
Use of a study carrel
Assistance in maintaining uncluttered space
Weekly home-school communication tools (notebook, daily log, phone calls or email messages)
Space for movement or breaks
Study sheets and teacher outlines
Extra visual and verbal cues and prompts
Math sheets with highlighted instructions
Graph paper to assist in organizing or lining up math problems
Use of computers and calculators
Books on tape
Graphic organizers
Quiet corner or room to calm down and relax when anxious
Preferential seating
Alteration of the classroom arrangement
Reduction of distractions
Answers to be dictated
Hands-on activities
Use of manipulatives
No penalty for spelling errors or sloppy handwriting
Follow a routine/schedule
Alternate quiet and active time
Teach time management skills
Rest breaks
Verbal and visual cues regarding directions and staying on task
Agenda book and checklists
Adjusted assignment timelines
Visual daily schedule



SPRINGFIELD PUBLIC SCHOOLS

**ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY
GLOBAL SKILLS**

Varied reinforcement procedures
Immediate feedback
Work-in-progress check
Personalized examples

TESTING AND ASSESSMENT ACCOMMODATIONS AND MODIFICATIONS:

Answers to be dictated
Frequent rest breaks
Additional time
Oral testing
Untimed tests
Choice of test format (multiple-choice, essay, true-false)
Alternate ways to evaluate (projects or oral presentations instead of written tests)
Accept short answers
Open-book tests
Read test and directions to student
Provide study outline prior to tests
Highlight key directions
Test in alternative site
Use of calculator
Extra credit option
Preview test procedures
Simplified test wording; rephrased test questions and/or directions

GENERAL ACCOMMODATIONS AND MODIFICATIONS:

Use of alternative books or materials on the topic being studied
Computerized spell-check support
Word bank of choices for answers to test questions
Provision of calculator and/or number line for math tests
Reworded questions in simpler language
Projects instead of written reports
Highlighting important words or phrases in reading assignments
Modified workload or length of assignments/tests
Modified time demands
Modified grades based on IEP

BEHAVIOR ACCOMMODATIONS MODIFICATIONS:



SPRINGFIELD PUBLIC SCHOOLS

**ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY
GLOBAL SKILLS**

Breaks between tasks
Cue expected behavior
Daily feedback to student
Have contingency plans
Use de-escalating strategies
Use positive reinforcement
Use proximity/touch control
Use peer supports and mentoring
Model expected behavior by adults
Have parent sign homework
Have parent sign behavior chart
Set and post class rules
Chart progress and maintain data



SPRINGFIELD PUBLIC SCHOOLS

**ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY
GLOBAL SKILLS**

IEP AND 504 ACCOMODATIONS AND MODIFICATIONS FOR MIDDLE AND HIGH SCHOOL STUDENTS:

Large print textbooks
Textbooks for at-home use
Additional time for assignments
A locker with adapted lock
Review of directions
Review sessions
Use of mnemonics
Have student restate information
Provision of notes or outlines
Concrete examples
Adaptive writing utensils
Support auditory presentations with visuals
Weekly or bi-weekly home-school communication
Space for movement or breaks
Study sheets and teacher outlines
Extra visual and verbal cues and prompts
Lab and math sheets with highlighted instructions
Graph paper to assist in organizing or lining up math problems
Use of tape recorder for lectures
Use of Interactive notebook
Use of a scribe
Use of computers and calculators
Books on tape
Graphic organizers
Quiet corner or room to calm down and relax when anxious
Preferential seating
Alteration of the classroom arrangement
Reduction of distractions
Answers to be dictated
Hands-on activities
Ability to type all assignments
Follow a routine/schedule
Teach time management and organizational skills
Rest breaks
Verbal and visual cues regarding directions and staying on task
Agenda book and checklists
Daily check-in with case manager or special education teacher
Adjusted assignment timelines
Visual daily schedule



SPRINGFIELD PUBLIC SCHOOLS

**ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY
GLOBAL SKILLS**

Varied reinforcement procedures
Immediate feedback
Work-in-progress check
Personalized examples

TESTING AND ASSESSMENT ACCOMMODATIONS AND MODIFICATIONS:

Answers to be dictated
Frequent rest breaks
Additional time
Oral testing
Choice of test format (multiple-choice, essay, true-false)
Alternate ways to evaluate (projects or oral presentations instead of written tests)
Accept short answers
Open-book or open-note tests
Read test and directions to student
Provide study guides prior to tests
Highlight key directions
Test in alternative site
Use of calculator
Extra credit option
Pace long-term projects
Preview test procedures
Simplified test wording; rephrased test questions and/or directions

GENERAL ACCOMMODATIONS AND MODIFICATIONS:

Allow outlining, instead of writing for an essay or major project
Use of alternative books or materials on the topic being studied
Computerized spell-check support
Word bank of choices for answers to test questions
Provision of calculator for math tests
Film or video supplements in place of reading text
Reworded questions in simpler language
Projects instead of written reports
Highlighting important words or phrases in reading assignments
Modified workload or length of assignments/tests
Modified time demands



SPRINGFIELD PUBLIC SCHOOLS

**ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY
GLOBAL SKILLS**

Pass/no pass option
Modified grades based on IEP

BEHAVIOR ACCOMMODATIONS FOR STUDENTS:

- Breaks between tasks
- Cue expected behavior
- Daily feedback to student
- Have contingency plans
- Use de-escalating strategies
- Use positive reinforcement
- Use proximity/touch control
- Use peer supports and mentoring
- Model expected behavior by adults
- Have parent sign homework
- Set and post class rules
- Chart progress and maintain data



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY
GLOBAL SKILLS

Accommodations and Modifications for ESL Students

- Give tests orally
- Give instructions/directions in writing and orally
- Assign a peer tutor, same language or English speaking
- Allow use of bilingual/translation dictionary
- Provide summaries of text or study guides of particular chapters
- Allow errors in speaking
- Allow errors in writing
- Accept writing in first language
- Highlight key vocabulary
- Reduce amount of work required
- Assess comprehension through demonstration or other alternative means (gestures, drawings)
- Rephrase questions, directions, and explanations
- Reduce multiple choices on assessments
- Provide study guides and/ or outlines
- Shorten reading assignments
- Allow extended time for test/project completion
- Use of audio books
- Use of graphic organizers
- Use reduced text, so that print is not dense
- Use visuals/pictures throughout lesson/unit/assignment
- Adapt homework to reflect language proficiency/home support
- Adapt homework to reflect home support
- Provide hands-on activities and explanations
- Allow extended time to answer questions, and permit drawing, as an explanation
- Use of translation dictionaries to locate words in the native language
- Use anecdotal records and portfolio of work as forms of assessment
- Use story retellings to assess comprehension