Final Design Project

1. Synopsis

This is the culminating project for the Middle School App Inventor curriculum. It is intended as a way to assess learning of computational thinking concepts, practices, and perspectives in the curriculum, and as a way to showcase gained skills and knowledge. Students will demonstrate their knowledge of computational thinking concepts by incorporating sequences, parallelism, repetition, conditionals, naming, operators, data structures, and events into an app of their choosing. They will also apply computational thinking practices of reusing and remixing, being incremental and iterative, abstraction, testing and debugging, and algorithmic thinking to successfully create the app. Students will work in pairs for the final project.

2. Learning Objectives

After completing this project, students will be able to:

1. Ideate and design an app that can be used to solve a problem.
2. Apply computational thinking concepts such as sequences, events, repetition, conditionals, parallelism, operators, data structures, and naming.
3. Employ computational thinking practices such as reusing and/or remixing code, decomposing tasks into subtasks, being iterative and incremental, and testing.
4. Work collaboratively to design and build a project.
5. Increase personal interest and confidence in computing by creating a complex app from the initial ideation and design stages.

3. Mapping with the Computational Thinking Framework

These tables show the alignment of this unit with the intended learning outcomes for the computational thinking framework. The entries in the tables indicate the expected relevance of the unit to each outcome:

✓✓✓ : High relevance

✓✓ : Some relevance

✓ : Low relevance

Computational Thinking Concepts

Final Project		
1. Sequences	✓✓✓	Students must sequence blocks correctly to make their app work.
2. Events	✓✓✓	Events such as button clicks will be included.
3. Repetition	✓✓	Possible repeated user interactions or loops may be included in the app.
4. Conditionals	✓✓	Possible conditionals may be included in the app.
5. Parallelism	✓✓	ImageSprites may act in parallel in an app.
6. Naming	✓✓	Students should use descriptive names for components and variables.
7. Operators	✓✓	Mathematical operators may be used in the app.
8. Manipulation of data and elementary data structures	✓✓✓	Lists may be used in the app.

Computational Thinking Practices

L3AIFP: App Inventor Final Project		
1. Reusing and remixing	✓✓✓	Students are encouraged to reuse and remix components and blocks from previous App Inventor units.
2. Being incremental and iterative	✓✓✓	Students design their app, receive feedback, code, and receive feedback again.
3. Abstracting and modularizing	✓✓	Students may use procedures as a means of abstraction in their app.
4. Testing and debugging	✓✓✓	Students should test their app to ensure it works.
5. Algorithmic thinking	✓✓	Students may need to use more complex logic in their app.

Computational Thinking Perspectives

L3AIFP: App Inventor Final Project		
1. Expressing	✓✓✓	Students may express their creativity by designing and making their own app.
2. Questioning	✓✓✓	Students should learn more about technology when building an app from scratch.
3. Connecting	✓✓	Students connect their app to their real life and healthy habits.
4. Computational identity	✓✓	Students gain more confidence as a creator of apps.
5. Digital empowerment	✓✓	Students are empowered by making an app that can be used to help others.

4. Mapping with the CSTA Standards

This table shows the alignment of this unit with the intended learning outcomes to the CSTA CS Standards. The entries in the tables indicate the expected relevance of the unit to each outcome:

2-CS-02	Design projects that combine hardware and software components to collect and exchange data. [C] CS: Hardware & Software [P] Creating (5.1)	Students may build an app to collect and exchange data on mobile devices.
2-AP-11	Create clearly named variables that represent different data types and perform operations on their values. [C] AP: Variables [P] Creating (5.1, 5.2)	Students should use variables to store data.
2-AP-12	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. [C] AP: Control [P] Creating (5.1, 5.2)	Students will use conditionals, and perhaps compound conditionals and nested loops.
2-AP-13	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. [C] AP: Modularity [P] Computational Problems (3.2)	Students design and implement the app over 10 class periods, breaking down the implementation into achievable steps.
2-AP-14	Create procedures with parameters to organize code and make it easier to reuse. [C] AP: Modularity [P] Abstraction (4.1, 4.3)	Students may implement procedures in their project.
2-AP-15	Distribute tasks and maintain a project timeline when	A timeline will be kept, and depending on the

	collaboratively developing computational artifacts. [C] AP: Program Development [P] Collaborating (2.2)	tasks, distribution of tasks may occur.
2-AP-16	Seek and incorporate feedback from team members and users to refine a solution that meets user needs. [C] AP: Program Development [P] Collaborating (2.3), Inclusion (1.1)	Peer feedback is used throughout the unit.
2-AP-17	Incorporate existing code, media, and libraries into original programs, and give attribution. [C] AP: Program Development [P] Abstraction (4.2), Creating (5.2), Communicating (7.3)	Students may reuse and remix project created in previous units.
2-AP-18	Systematically test and refine programs using a range of test cases. [C] AP: Program Development [P] Testing (6.1)	Students test and refine in stages.

Three tables follow, outlining the alignment Computational Thinking Concepts, Practices, and Perspectives in this project.

Computational Thinking Concepts

App Inventor Final Project	
1. Sequences	Students must sequence blocks correctly to make their app work.
2. Events	Events such as button clicks, camera picture taking, and pedometer step walking will be included.
3. Repetition	Students may use loops to iterate through a list.
4. Conditionals	Possible conditionals may be included in the app.
5. Parallelism	Depending on the app, multiple users could use the app in parallel.
6. Naming	Students should use descriptive names for components and variables.
7. Operators	Mathematical operators may be used in the app.
8. Manipulation of data and elementary data structures	TinyDB, CloudDB and/or lists may be used in the app.

5. Learning Prerequisites

Students should have completed all units of the curriculum, so they should have a complete understanding of the computational thinking concepts needed to fulfill the final project requirements.

6. Lesson Plan (45 minutes x 10)

This unit consists of ten 45 minutes lessons. Teachers may shorten or extend this time, depending on school schedules.

Lesson 1

<table>
<tr><th>Time</th><th>Activity</th></tr>
<tr><td>10 min</td><td>
Introduction to the Final Project

Explain to the students they will be designing an app around a particular problem. Teachers may choose a particular theme, such as the environment, school, community, family, the elderly, or may leave it more open.
Remind students of components and concepts they have learned and may incorporate in their apps.

<table>
<tr><th>Components</th><th>Concepts</th></tr>
<tr><td>Map
Camera
TextToSpeech
TinyDB
CloudDB
Accelerometer
Canvas
ImageSprite/Ball</td><td>Sequences
Events
Conditionals
Parallelism
Naming
Operators
Manipulation of data and elementary data structures</td></tr>
</table>
<ol start="3">
Students will work in pairs on the final project. Remind students of the concept and rules of pair programming.
Ask students to form groups of two (pairs).

</td></tr>
<tr><td>30 min</td><td>
Brainstorming and Planning

Students will work with their partners using the Pair Programming model, using the Final Project Design Worksheet to detail the project.

</td></tr>
</table>

	2. Check and comment on each group's worksheet if necessary, and assist the groups who are having trouble coming up with a workable plan for their app. 3. Demonstrate how to fill out the "To-Do Checklist"in the final part of the Final Project Design Worksheet and ask students to complete that page of the worksheet.
5 min	**Wrap-up** 1. Check in that all groups have a feasible plan. 2. Remind student groups to make sure their plan is something they can complete in 8 lessons.

Lesson 2

Time	Activity
5 min	**Introduction to Lesson** Check in again with student groups to make sure they have a completed Design Worksheet. Discuss giving constructive feedback, and receiving and using feedback given by others.
10 min	**Peer Feedback** Ask student groups to present and share their Design Worksheets with another group and write down their feedback and the corresponding changes.
5 min	**Update Design Worksheets** Student groups should update their Design Worksheets based on feedback from their partner group. Ask for one or two groups to share with the class what updates they are making.
25 min	**Code the first feature of the app** Student groups work on their apps, based on their Design Worksheet, . 1. Remind students they are using the Pair Programming model, and to take turns being driver/navigator. 2. Code the first element of the app, based on their *To-Do Checklist*. 3. Test and debug the app using MIT AI2 Companion.

Lesson 3 - 5

Time	Activity
5 min	**Review of progress** 1. Check the progress of students' apps 2. Explain the target of this lesson: work on the code of one more element in app.
35 min	**Code New Elements in App** 1. Remind students to modify their app design based on feedback from their partner group in Lesson 2. 2. Students work on their apps based on the *To-Do Checklist* for this lesson. 3. Students test and debug the app using MIT AI2 Companion.
5 min	**Wrap-up** Check in with groups to make sure they have completed a new section of their To-do List.

Lesson 6

Time	Activity
5 min	**Review of progress** 1. Check the progress of students' apps 2. Explain the target of this lesson: peer feedback on current implementation of app.
15 min	**Peer Feedback** Student groups meet with a second group (this should be a different group than the one they met with in Lesson 2) and share their apps with each other. Groups provide verbal and written feedback using the Feedback Worksheet to the other group.
10 min	**Update Design Worksheet** 1. Students share Feedback Worksheets with the other group. 2. Groups update their Design Worksheet based on feedback from their partner group.
15 min	**Coding Activity** Student groups continue coding their app.

Lessons 7-9

Time	Activity
5 min	**Review of progress** 1. Check the progress of students' apps 2. Explain the target of this lesson: work on the code of one more element in app.
35 min	**Code New Elements in App** 4. Remind students to modify their apps based on feedback from their partner group in Lesson 2. 5. Students work on their apps, based on the "To-Do Checklist" for lesson 3. 6. Test and debug the app using MIT AI2 Companion.
5 min	**Wrap-up** Check in with groups to make sure they have completed a new section of their To-do Lists.

Lesson 10

Time	Activity
10 min	**Introduction to Final Lesson** Explain that students will share their apps with the class. Give students 5-10 minutes to prepare their presentation.
30 min	**App Sharing** Students share apps with the class. Depending on the size and classroom logistics, this can be: 1. Meeting with other groups in a round-robin style where they share with each other. 2. Group presentations in front of the class. 3. Students install apks on phones and groups rotate around the room to try each other's apps. 4. Students set up around the classroom to demonstrate their apps and classmates visit to learn about them. You could invite other classes to see the apps too.
5 min	**Wrap-up** Final discussion on apps and completion of surveys.

7. Assessment

Below is the mapping table of assessment modes and the corresponding Learning Objectives from Section 2 above.

Assessment Modes	UILO
1. Teacher Assessment: Final project with assessment rubrics	$LO_{1,2}$
2. Self-assessment: Self-reflection questions	$LO_{4,5}$
3. Peer Assessment: Feedback Worksheet	LO_{1}

Teacher Assessment

This checklist should be used to note students' use of **Computational Thinking Concepts** in their final projects. Level 0 indicates that the concept is not included. Level 1 indicates that the concept is included but incorrectly and/or inappropriately used. Level 2 indicates the concept is used correctly and appropriately. Select one level for each item by checking the appropriate box using a tick (✓). Students will not be assessed on whether they include these concepts or not. This is merely informational.

Computational Thinking Concepts			
Level	2	1	0
1. Events and Sequences			
2. Conditionals			
3. Naming / Variables			
4. Operators (mathematical and/or logical)			
5. Procedures			
6. Repetition (loops)			
7. Elementary Data Structures (lists)			
8. Manipulation of data (storage in TinyDB and/or CloudDB)			

This rubric assesses the student's use of Computational Thinking Concepts (e.g. Does the final project include correct and proper use of concepts learned in the curriculum?).

Level	5	4	3	2	1
Include Computational Thinking Concepts (10%)	Includes varied, appropriate, and accurate computation-al thinking concepts in the app.	Includes varied, appropriate, and mostly accurate computa-tional thinking concepts in the app.	Includes appropriate, and mostly accurate computational thinking concepts in the app.	Includes appropriate, and sometimes accurate computational thinking concepts in the app.	Does not include any computational thinking concepts in their app.

This rubric assesses the student's implemented solution (e.g. Does the final project address the problem/theme?).

Level	5	4	3	2	1
Implements a solution (30%)	Implements a solution in a manner that addresses, thoroughly and in depth, multiple contextual factors.	Implements a solution in a manner that addresses multiple contextual factors.	Implements a solution in a manner that addresses the problem but ignores relevant contextual factors.	Implements the solution in a superficial manner that does not directly address the problem statement.	Does not implement any solutions.

This rubric assesses the logical design of the student's project (e.g. Does it have well-designed programming logic? Is the interface design of the App Inventor project appropriate?).

Level	5	4	3	2	1
Programming logic (20%)	Code blocks are appropriate, logical, efficient, and correct for the proposed solution.	One or two instances where code blocks do not work, are inappropriate, or could be more efficient.	Three to five instances where code blocks do not work, are inappropriate, or could be more efficient.	More than five instances where code blocks do not work, are inappropriate, or could be more efficient.	Code blocks are incorrect or incomplete and do not solve the problem.
User Interface (20%)	User interface is advanced, with use of horizontal and/or vertical layouts and multiple components, arranged in a logical and aesthetically pleasing way.	User interface is advanced, using multiple components, arranged in a logical and aesthetically pleasing way.	User interface is basic, with limited components, arranged in a logical and aesthetically pleasing way.	User interface is basic, with limited components, arranged in a non-logical way.	User interface makes it difficult for the user to run the app.

This rubric assesses the **Creative Thinking** Components of "*Innovative Thinking"* and "*Connecting, Synthesising, Transforming."*

Level	**5**	**4**	**3**	**2**	**1**
Innovative Thinking (10%)	Extend a novel or unique idea, question, format or product to create new or boundary-crossing knowledge.	Create a novel or unique idea, question, format or product.	Experiment with creating a novel or unique idea, question, format or product.	Reformulate a collection of available ideas.	Does not formulate any ideas.
Connecting, Synthesising, Transforming (10%)	Transform ideas or solutions into entirely new forms.	Synthesise ideas or solutions into a coherent whole.	Connect ideas or solutions in novel ways.	Recognise existing connections amongst ideas or solutions.	Does not recognise any connections amongst ideas or solutions.

This rubric assesses how students reflect upon and evaluate the process and outcomes. (Note: This rubric is based on the student's self-assessment below.)

Level	**5**	**4**	**3**	**2**	**1**
Reflect upon and evaluate the process and outcomes (based on student reflection below) (10%)	Review the quality of the process and outcomes, with thorough and specific consideration of the need for further work.	Review the quality of the process and outcomes, with sufficient consideration of the need for further work.	Review the quality of the process and outcomes, with some consideration of the need for further work.	Review the quality of the process and outcomes superficially, with inadequate consideration of the need for further work.	Does not review any process or outcomes.

Remark: The above rubrics for creative thinking and problem solving skills are adapted from the Rubrics for Generic Intended Learning Outcomes (GILOs) of The Education University of Hong Kong.

Self-assessment

Self-assessment allows students to reflect on their learning performance. Please ask students to answer the following questions in the CoolThink@JC Web Portal:

- Look back at your initial design. How does your completed final project differ from your initial design? Why did you make these changes?
- What else would you want to do to improve this finalized project? (You will not have to code these improvements - just describe them.)
- If you had the chance to redo this project, what would you do differently?
- Did you like working with another person? Do you feel you were a good partner, and respected and encouraged your partner in the project?
- What was your role as a partner in this project? What did you do and what did your partner do?

Peer Assessment

Students will make comments on another group's apps based on the Feedback Worksheet.

8. Screen Design and Code

The App Inventor Final Project is self-directed so there is no screen design and code.

Appendix 1
Final Project Teacher's Guide: Lesson 1

Learning Objectives

At the end of this lesson, students should be able to:

1. Brainstorm ideas for apps they can implement.
2. Develop a design for a new app and plan the detailed layout and components needed.
3. Decompose a complex development project into a set of tasks that can be accomplished in sequence.
4. Share their final project ideas with their peers, and give and receive helpful feedback.
5. Work collaboratively to design a mobile app.

Lesson Outline

Introduction and Discussion (10 minutes)

1. Briefly introduce the final project. Tell the students that this project is their opportunity to demonstrate their computational thinking knowledge by creating a mobile app of their own design and imagination. The final project will be an app that solves a particular problem. Teachers can decide whether to impose a particular theme, such as the environment, school, community, family, the elderly, or leave it more open.

2. Explain that students should consider how they can include components and concepts within their app, according to the table below.

New Components	**Concepts**
• Map • Camera • Pedometer • TextToSpeech • SpeechRecognizer • TinyDB • CloudDB • Accelerometer	• Sequences • Events • Repetition • Conditionals • Parallelism • Naming • Operators • Manipulation of data and elementary data structures

3. Students will work in pairs on this project. There are many different strategies that teachers can use to group students; you may choose to put a high-achieving student with a student who struggles, or you might put students of a similar level together. Each has its own advantages and disadvantages. Explain that students will be working together with a partner to design and code their project. They will use the pair programming model to code their project. Remind them of the pair programming model and ask them if they can remember the table of rules:

DO	**DON'T**
• Be respectful; • Talk to one another about the work; • Explain what you are doing; • Think ahead and make suggestions; • Switch roles often.	• Be a bossy navigator; • Grab the driver's mouse/keyboard.

4. Have students form their groups.

Brainstorming and Planning (30 minutes)

1. Generate a brainstorming session on the board with ideas for apps that might address a problem. It can help to set a particular theme, like the environment, school, community, family. Ask students for ideas. Explain that in brainstorming, no ideas are thrown away. All ideas are valid.
2. Hand out and go over the *Final Project Design Worksheet*. The student teams will write brief descriptions of their final projects and the steps needed to make them happen. They will also draw sample screenshots, describe the user interface, and list necessary components.
3. Demonstrate how to fill out the *To-Do Checklist* in the final part of the *Final Project Design Worksheet* and ask students to complete that page of the worksheet.
4. Check and comment on each team's design to make sure they have a reasonable project plan.

Wrap-up (5 minutes)

1. Check in that all groups have a feasible plan.
2. Remind student groups to make sure their plan is something they can complete in 8 lessons.

Appendix 2
Final Project Teacher's Guide: Lesson 2

Learning Objectives

At the end of this lesson, students should be able to:

1. Share their projects with each other and give helpful feedback.
2. Incorporate feedback provided by others.
3. Work collaboratively to implement an app design step-by-step.

Lesson Outline

Lesson Introduction (5 minutes)

1. Check in again with student groups to make sure they have a completed Design Worksheet.
2. Discuss giving constructive feedback, and receiving and using feedback given by others. The expectation is that each group will provide some feedback. They must have some comments and suggestions to give to their peers.
3. Discuss how to receive feedback well. Students are expected to listen carefully to feedback and possibly incorporate changes to their design based on the feedback.

Peer Feedback (10 minutes)

Have each student group pair up with another group to take turns sharing their Design Worksheet and giving verbal feedback. One member of each pair should take notes on their Design Worksheet, writing down what their peers have said.

Update Design Worksheets (5 minutes)

Student groups should update their Design Worksheets based on feedback from their partner groups. Ensure that each group has filled out the feedback section of their Design Worksheet, and ask each group what they are changing, based on their peers' feedback. Ask for one or two groups to share with the class the updates they are making.

Code first feature of app (25 minutes)

Student groups work on their apps based on their Design Worksheets.

1. Code the first element of the app, based on their *To-do Checklist*.
2. Test and debug the app using MIT AI2 Companion.

Appendix 3
Final Project Teacher's Guide:
Lesson 3, 4, 5, 7, 8, 9

Lessons 3, 4, 5 and 7, 8, 9 should all follow the same format. Students work on their apps, following the To-do Checklists in their Design Worksheets.

Learning Objectives

At the end of this lesson, students should be able to:

1. Follow the computational thinking practice of being incremental and iterative in developing their apps step-by-step.
2. Employ the computational thinking practice of testing and debugging to ensure their apps work as expected.
3. Work collaboratively to develop and test an app.

Lesson Outline

Review of Progress (5 minutes)

1. Check progress of students' apps.
2. Remind students to update their apps based on feedback they may have received in Lesson 2 from their partner group.
3. Explain target of this lesson: work on the code of at least one more element in the app.

Code New Elements in App (35 minutes)

1. Remind students to modify their app design based on feedback from their partner group in Lesson 2.
2. Students work on their apps. based on the *To-Do Checklist* for this lesson.
3. Test and debug the app using MIT AI2 Companion.

Wrap-up (5 minutes)

Check-in with groups to make sure they have completed a new section of their To-do List.

Appendix 4
Final Project Teacher's Guide: Lesson 6

Learning Objectives

At the end of this lesson, students should be able to:

1. Give positive and constructive feedback to their peers and also accept and possibly integrate feedback from others.
2. Follow the computational thinking practice of being incremental and iterative in developing their apps step-by-step.
3. Work collaboratively to develop and test an app.

Lesson Outline

Review of Progress (5 minutes)

1. Check the progress of students' apps
2. Explain the target of this lesson: peer feedback on current implementation of app. After 5 lessons working on their app, students should have at least a partially working app where they can show its current features to their peers.

Peer Feedback (15 minutes)

Student groups meet with a second group, different from the previous one, and share their apps. Groups provide verbal and written feedback using the Feedback Worksheet to the other group.

Update Design Worksheet (10 minutes)

1. Students share Feedback Worksheets with the other group. If needed, they may add verbal explanation to what they have written.
2. Each group updates their Design Worksheet based on feedback from their partner group

Coding Activity (15 minutes)

Student groups continue coding their apps. Remind students that they should update their *To-do Checklist,* since it should change based on the feedback from their peers.

Appendix 5
Final Project Teacher's Guide: Lesson 10

Learning Objectives

At the end of this lesson, students should be able to:

1. Explain how their apps work and how they were made.
2. Give positive and constructive feedback to their peers and also accept feedback from others.

Lesson Outline

Introduction to Final Lesson (10 minutes)

Explain that students will share their apps with the class. Give students 5-10 minutes to prepare their presentation.

App Sharing (30 minutes)

Students share apps with the class. Depending on the size and classroom logistics, this can be:

1. Meeting with other groups in a round-robin style where they share with each other.
2. Group presentations in front of the class.
3. Students install apks on phones and groups rotate around the room to try each other's apps.
4. Students set up like exhibitors at a science fair and students visit to learn about the apps. You could invite other classes to see the apps too.

Wrap-up (5 minutes)

Final discussion of apps and completion of surveys.