

85-738 EGIA Individual Project

App Development for High School Students

Author: Christina Ma (qianoum)

Instructors: Sharon Carver (sc0e), Lauren Herckis (lrhercki)

18th December, 2020

Table of Content

CS education: the importance and challenges	3
Why should students learn CS?	3
What are some challenges in CS education?	3
Domain Challenge	3
Learning Challenge	3
A proposed solution: afterschool CS program	4
Project-based learning and near-peer mentoring	4
The proposed unit: app development for high school students	4
Similar educational interventions	5
Learner Profile: who are the students?	6
Learner characteristics	6
Developmental level	6
Prior experience	6
Individual differences	7
Learners in context	8
Learning Goals: what will students learn?	9
Conceptual Knowledge	10
Cognitive Level (students will be able to identify or define ...)	10
Metacognitive Level (students will learn to ...)	10
Procedural Skills	11
Cognitive Level (students will be able to ...)	11
Metacognitive Level (students will learn to ...)	12
Dispositions	13
Cognitive Level (students will gain ...)	13
Metacognitive Level (students will learn to ...)	13
Assessment: how well do students learn?	14
Overview of Assessment Evidence	14
Use of Assessment	14
Design Justification	15
Specific Assessments Overview	16
Specific Assessments Rubrics	19
Project proposal	20
Storyboard development	22
User survey development	22

Code review	24
Project presentation	25
Workshop exit survey	26
Instruction: how should we teach?	28
General Description	28
Classroom Climate	28
Weekly Routines	28
Key Design Elements	29
Instruction and Assessment Timeline	30
Design Justification	32
Specific Activities	33
Collaboration agreement & design practices	33
Project proposal, Storyboard, User survey (development & revision)	34
App development & programming practices (basics / tailored)	34
Project presentation guideline & practice	36
Evaluation: how could we improve?	37
Research to Evaluate Educational Impact	37
Research Questions and Hypotheses	37
Experimental Design	37
Experimental groups & control group	38
Design quality assessment	38
Research to Evaluate Educational Implementation	39
Weekly workshop checklist	40
Conclusion: peer feedback and project reflection	42
Peer Feedback	42
Project Reflection	43
Self-Assessment of the Project PRODUCT	43
Self-Assessment of the Project PROCESS	44
Resources	46
Personal Background	46
Educational Materials	46
Professional Contact	46
Meeting Notes with Contacts	46
Appendix A: Selective Standards from CSTA K-12 Computer Science Standards (2017)	49
References	51



CS education: the importance and challenges

Why should students learn CS?

Computer Science (CS) is a rapidly developing field in both industry and academia, and the fundamental concepts in CS including algorithmic thinking and data analysis become core to students in the 21st-century. Learning CS practices and abilities not only provide students enough tech literacy to survive the technological evolution, but it also enables them to become creative thinkers and problem-solvers to tackle their own individual questions and also the challenges for their generation and society.

The rigorous thinking habits and problem-solving abilities learned in CS projects will be applicable to any field of study, so these are highly transferable skills. And students will also become interdisciplinary thinkers and be able to use technological tools to facilitate other disciplines that they care about.

Additionally, no one can work solely on their own in this complicated world. Real-world CS applications always involve collaboration; therefore, practicing team skills in a collaborative programming project is especially beneficial to students. This can also help foster equity and open access to CS and connect very diverse students together.

What are some challenges in CS education?

Domain Challenge

Incorporating CS courses into K-12 education has gained more and more popularity; however, many schools suffer from the lack of instructors and professional development resources (Yadav et al., 2016).

Learning Challenge

Equity is a staggering issue in CS education, as students from lower social-economic status and minority groups suffer from the lack of resources and prior exposure (Vakil, 2018), and girls can be severely impeded by stereotype threat (Master et al., 2017).

Furthermore, many CS courses become disconnected to students' intrinsic motivation and interests. For example, according to the account of some high school students that I tutored before, some pre-college CS courses focus more on specific programming languages or theoretical constructs, which is hardly fun for students.

A proposed solution: afterschool CS program

Project-based learning and near-peer mentoring

Definitions:

- **Near-Peer Mentoring:** an instructional design model where mentors are proximal in age (near-peer) but more skilled (Clarke-Midura et al., 2018).
- **Project-Based Learning:** an educational instruction method that fosters learning through problem solving and application of knowledge in real-world settings (Merritt et al., 2017).

There are existing educational outreach organizations that offer near-peer mentoring in project-based learning experiences, and it could possibly be one solution to the challenges discussed above: the near peers can be supplementary instructors for students who don't have access to CS education resources in their own schools, and projects are a good way to expose students to real-life problems and allow them to apply the CS knowledge they learned.


Various research showed that project-based learning (PBL) improves students' learning in different disciplines (Dole et al., 2017; Merritt et al., 2017), and it also reduces the performance gap (Holmes & Hwang, 2016). However, PBL is not often implemented in schools' curriculum because of the additional costs on resources, time, and teachers' training (Menzies et al., 2016). Therefore, afterschool programs may have more flexibility to implement this strategy.

On the other side, near-peer mentoring was shown to have a positive effect on middle school students because of role modeling (Clarke-Midura et al., 2018), so there may be an opportunity to combine these two methods together to tackle the learning challenges in CS.

The proposed unit: app development for high school students

In this project, I proposed a 10-hour educational intervention unit on a particular application of CS for a specific group of learners: app development for high school students. The topic of app development introduces a variety of interdisciplinary opportunities in all subject areas that a high school teaches and beyond, and it allows students to apply their knowledge in CS and other school subjects of their interests to solve any real-world problems that they care about.

This app development unit will be situated as an after-school program during weekends, with 1-hour instructional time each week for a span of 10 weeks. High school students of grade 11-12 will come to CMU campus for the weekly workshops (or meet remotely through zoom under pandemic). High school students will be grouped into a team of 4-5 and paired with 1-2 CMU



students as project advisors. This unit is designed to be implemented in a real-world context as a project in an educational outreach student organization at CMU: [Project Ignite](#), which holds 10-weeks workshops for local Pittsburgh high schoolers each spring, and app development is a popular project that's often offered in Project Ignite. The following learner profile, goal setting, assessment, instructional, and evaluation design of this unit are all based on this context.

Similar educational interventions

Some comparable counterparts of this unit under the formal school environment are a light-weight term project at the end of an introductory CS course, or a brief, less in-depth version of a CS product design course. Therefore, with some small tweaks on this unit (like compressing the 10 weeks into 4 weeks), it can be used as a group project assignment in an introductory CS course in college, or in an advanced CS course in high school.

The [term-project](#) of CMU's introductory CS course 15112 requires 40 hours of individual work in a span of 4 weeks, note that this course is also offered during summer with registrations of high school upperclassmen. This unit, of course, should be much less intensive (only a 2~3-hrs devotion is expected per week, including the 1-hour instructional time in the 10 weeks) and it relies heavily on team collaborations.

The [course](#) *Entrepreneurship and Product Design 1* in Winchester Thurston is a year-long course open to students from grades 10-12, with a prerequisite of a prior "trimester" CS course. This unit's time span is much reduced (a 10 weeks project rather than a year-long course), so the business or entrepreneurship component would be minimal, and students will be focused on the design and programming sides of app development.



Learner Profile: who are the students?

Learner characteristics

Developmental level

Typically developed high school students at grades 11-12 (age 16-18) should be able to think abstractly and analogically, reason logically in inductive and deductive ways, and grasp the scientific process of problem definition and hypothesis testing (Berger, 2003). Thus, students have the ability to handle a more demanding curriculum in high school and develop their own understanding of complex scientific, philosophical, moral, and social concepts.


By mid-adolescence, students can rationally reason, make decisions, and plan under “cold” cognitive conditions. Because of more mature brains, high school students have improved memory and executive control capabilities, so they are better at time management and organization than the younger teenagers. (Armstrong, 2016).

Students at this age are capable of perspective-taking and understanding others, and they also start to care about the world beyond themselves and their future life and goals (Berger, 2003). However, they may still have a more self-centered focus and firmly believe in their own ideas, which may pose challenges to collaboration.

The physical development of students around this age is near adulthood. Students also show more independence and begin making important decisions on their own, but they may develop concerns or confusion about their future (Berger, 2003). They also have an intense desire for social connections and a strong need for peer approval; therefore, their social development puts a strong emphasis on the role of peers and friendships (Armstrong, 2016).

Prior experience

Students should have completed algebra I, be comfortable with generic CS concepts, be able to code in at least one imperative language (any of JavaScript, Python, C, C++, and etc. suffice), and have experiences using online resources including but not limited to development communities and repositories (stackoverflow, python library documentation, GitHub, etc.). Students should have experiences developing surveys (in any discipline) to collect people’s feedback or opinions, understand how to avoid leading questions, and perform analysis on the data to extract the common patterns in the responses. Moreover, some presentation experiences are needed, as students should know how to make good presentations (be familiar with tools like powerpoint, google slides, and general principles like provide less words and good visuals, fairly attribute



individual contributions in teamwork, and etc.) Students should also have team work experiences, and be able to characterize their own collaboration and working styles.

Basic exposure to CS and design ideas is required for learning the general app development programming practices, but no pre-existing knowledge in app development is needed. Furthermore, students should have individual problem-solving skills and basic communication, teamwork, and reflection dispositions, but they don't necessarily need collaborative experiences for programming projects specifically.

Students who have more experience programming may be able to pick up the app development techniques more quickly, but they're also likely to fall under the misconceptions of programming as an individual task and writing code in their own habits without considering team collaboration. So it will be important for them to learn how to write readable, modularized, and portable code and practice code documentation for teamwork.


Students may be more used to the lecturing style of teaching and the norms in a formal school context and these norms may prevent them from taking more active control of the collaboration process in this less familiar informal after-school project context. Also, students may not be used to a common collaboration style (which is full of take-turn communication and compromises) in their cultural norms, so it will probably confuse them if the new norms aren't explicitly discussed or if their convention wasn't included. Furthermore, students may knowingly or unknowingly suffer from stereotype threats with regard to their social groups, which could make them less comfortable expressing themselves.

Individual differences

Learners from different backgrounds are likely to have different interests and programming abilities, some may be more attracted by the design side, some may want to spend more time on the technical side. Also, students may have different comfort levels in collaboration and unique ideas in terms of what they want to do with the app, so this unit will allow them to improve their team communication and collaboration skills and compromise to achieve their individual goals.

Since this unit incorporates team collaboration, it will allow students to contribute in the directions of their interests and collaborate with others with different backgrounds and fields of expertise. Besides programming itself, the app development process would allow students to gain experience in prototype development, user experience and market research, visual and user design, or data collection, analysis, and visualization.

There's a lot of flexibility to adjust the instructional emphasis based on individual differences. To fully support teams consisting of students from various backgrounds, pre-entry surveys can be used to assess students' abilities and interests and help balance the team composition, and



instructions will also be tailored to meet students' needs (e.g., add advanced features like move apps onto phones if students all have a strong programming background, or add interdisciplinary components to foster the intrinsic motivation for students from different backgrounds).

Learners in context

Students who participated in Project Ignite, the afterschool program in which this unit is designed to be implemented, likely come from the greater Pittsburgh area, especially high schools that are physically closed to the CMU campus because of the transportation limitation. Project Ignite has collaborated with a variety of Pittsburgh high schools, including public schools, private schools, catholic schools, charter schools, magnet schools, and home-school. Some specific examples include Upper St. Clair, Mt. Lebanon, Peters Township, Taylor Allderdice, Winchester Thurston, SciTech, Penn Hills, Pittsburgh Obama 6-12, Central Catholic, and Academy of Excellence.

Since some prior exposure to computational thinking is required, it's likely that this project would not be able to reach the communities that are most at-risk and can't provide secondary school students with early resources in CS. But it still includes a student body that comes from a broad range of communities which has various values and priorities, and a broad range of high schools which teaches with different practices and terminologies.

Therefore, as app development necessarily involves a lot of collaboration, it's important to build healthy norms in the teams, create a sense of belonging, and make students comfortable with the norms of after-school projects with highly interactive and collaborative instructional styles. Specifically, instructors should make sure that a natural critical learning environment is fostered, the project team's norm is collaboratively constructed so that "all students are listened to, respected, and viewed as valuable contributors to the learning process." (Gal-Ezer et al., 2014) Particular attention is needed for populations that are likely under stereotype threats.

For example, students may not have personal laptops coming into the program, which will be inconvenient for them to make significant contributions during the weekdays. Project Ignite can lend laptops to students to enable more equitable access to the program, but an additional laptop tutorial on general setup and basic coding pipeline may be needed.

Furthermore, students may have learned the required prior knowledge about CS and design in different means and terminologies, it's important to establish a common ground for the students and be explicit about the definition of important concepts and how we will refer to them. Although we've specified some prerequisite requirements for this program, to better personalize the assessment and instruction plans based on students' prior knowledge and skills and accommodate for the diverse student body, a pre-entry survey should be used to get a better sense of a particular group of students in the program and group the project teams.

Learning Goals: what will students learn?

* denotes standards adopted from CSTA K-12 Computer Science Standards (2017), [Appendix A](#);

** denotes standards adopted from Barr & Stephenson (2011).

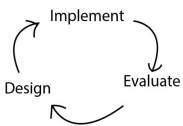
112 denotes standards adopted from the [term-project](#) of CMU's introductory CS course 15-112

Conceptual Knowledge	Procedure Skills	Dispositions
Cognitive Goals <ul style="list-style-type: none"> ● I-A-1 Storyboard prototype ● I-A-2-a MVP ● I-A-2-b Iterative Design ● I-A-3-a-(1) usability ● I-A-3-a-(2) readability ● I-A-3-a-(3) robustness ● I-A-3-b App development tools and languages ● I-A-3-c Version control and design documentation 	Cognitive Goals <ul style="list-style-type: none"> ● II-A-1-a-(1) Project Description ● II-A-1-a-(2) Timeline Plan ● II-A-1-a-(3) Structural Plan ● II-A-1-a-(4) Resource List ● II-A-1-b storyboard ● II-A-1-c-(1) Survey Question ● II-A-1-c-(2) Survey Format ● II-A-1-c-(3) Feedback Analysis ● II-A-2-a use existing resources ● II-A-2-b-(1) usability ● II-A-2-b-(2) readability ● II-A-2-b-(3) robustness ● II-A-3-a-(1) collaborative design practices ● II-A-3-a-(2) programming ● II-A-3-a-(3) communication ● II-A-4-a slide deck ● II-A-4-b present to the public 	Cognitive Goals <ul style="list-style-type: none"> ● III-A-1 responsibility ● III-A-2 confidence to complexity ● III-A-3 persistence ● III-A-4 independence ● III-A-5 politeness and patience when helping
Metacognitive Goals <ul style="list-style-type: none"> ● I-B-1 prioritize design or programming principles ● I-B-2 evaluate design or coding products ● I-B-3 adjust design or programming tools choice 	Metacognitive Goals <ul style="list-style-type: none"> ● II-B-1 plan timeline ● II-B-2 monitor team MVP progress and adjust design and plan ● II-B-3 monitor and adjust on communication and collaboration ● II-B-4 evaluate design and app development process ● II-B-5 monitor and adjust user survey development process ● II-B-6 evaluate and adjust project presentation 	Metacognitive Goals <ul style="list-style-type: none"> ● III-B-1 evaluate growth mindset ● III-B-2 monitor and reflect responsibility ● III-B-3 balance help-seeking & independence

I. Conceptual Knowledge

A. Cognitive Level (students will be able to **identify** or **define** ...)

1. Storyboard prototype design in app development (define) (*3A-AP-13)
Storyboard: a storyboard demonstrates how users interact with the app with a series of actions with ≥ 6 panels, ≥ 3 of which demonstrate features of the app ([Figure 1](#)) (112)
2. Fundamental principles in user design (define)
 - a) Aim for a Minimal Viable Product (MVP) (*3A-AP-18)
MVP: a product with just enough features to be usable by early customers who can then provide feedback for future product development. The MVP for an app would just be something interactable on a specific mobile device.
 - b) Incorporate an iterative design procedure (*3B-AP-17)
Iterative Design: a design methodology based on a cyclic process of developing, testing, analyzing, and refining a product. For this project, it means collecting and evaluating feedback from team members, project advisors, or other users for at least once and incorporating them into the next iteration of design.
3. Common practices and tools for collaborative app development (identify)
 - a) Collaborative coding practices (*3B-AP-15, 3B-AP-21)
 - (1) Usability: modularization and reuse
 - (2) Readability: code comment, consistent variable naming
 - (3) Robustness: unit testing
 - b) App development tools and languages (*3B-AP-24)
E.g., IOS app: Xcode, Swift or Android app: Android Studio, Java, XML
 - c) Version control and design documentation tools (*3A-AP-22, 3B-AP-20)
E.g., GitHub, Terminal, Colab, Jupyter Notebook; Google doc, slide



B. Metacognitive Level (students will learn to ...)

1. Assess the importance, evaluate the effectiveness of, and prioritize different design or programming principles with regard to their MVP idea, e.g., asking questions like which operating system should we use for our app, should we aim for more iterations of user feedback or spend more time fleshing out the app itself, etc. (*3B-DA-07)
2. Evaluate the strengths and weaknesses of design or coding products (e.g., prototype quality or degree of modularization) (*3B-AP-23)
3. Reflect on how suitable are the design or programming tools to their projects and adjust by switching to alternatives if necessary (*3A-AP-15)

II. Procedural Skills

A. Cognitive Level (students will be able to ...)

1. Design and iteratively develop a MVP of app of their interest (*3A-AP-16)
 - a) Collectively write a project proposal, which include at least (112)
 - (1) Project Description: describe the bare minimum goals and functionalities for the MVP of their app
 - (2) Timeline Plan: describe a feasible timeline for completing major features of their MVP and the iterative stages to include user feedback
 - (3) Structural Plan: describe how the finalized team project will be organized in different functions and/or files, and how team members will divide the job and collaborate
 - (4) Resource List: a list of all external modules/ hardware/ technologies they are planning to use; similar projects online and how some functionalities or algorithms that are expected to be involved in their app can be built on them
 - b) Develop a storyboard prototype of their app ([Figure 1.](#)) (*3A-AP-13)
Use paper and pen or digital prototyping tools like figma if students are very comfortable with the concept
 - c) Iteratively collect and analyze user feedback (*3A-AP-19)
 - (1) Survey Question: design a user survey with ≥ 3 questions about the app's main features (based on the storyboard prototype or a preview demo depending on the stage of development) and ≥ 3 questions about design decisions, difficulties, or disagreements during design process
 - (2) Survey Format: using google form or other tools, using the appropriate format (e.g., multiple choice, yes/no, rating, short answer, etc.)
 - (3) Feedback Analysis: distribute the survey, interpret the collected user feedback, update the team design document with the summary, and translate it into to-do lists for each team member to implement users' suggestions
2. Program the app of their interest based on existing resources
 - a) Adapt existing algorithmic solutions to solve their problems (APIs, libraries, GitHub Repo, open-source projects) (*3B-AP-14, 3B-AP-16)
 - b) Improve the code usability, readability, and robustness (*3A-AP-21)
 - (1) Usability: better modularize and reuse the code
 - (2) Readability: add or change to more consistent comment and variable names
 - (3) Robustness: perform unit testing and debug (*3B-AP-21)
3. Collaborate on and manage a complex app development project

a) Adhere to agreed collaborative practices (*3A-AP-22, 3A-AP-23)

(1) Design: document design decisions in team's google drive at least once per week

(2) Programming: backup, version-control, and document code in team's GitHub repository at least once per week

(3) Team communication: clearly express intentions, politely give constructive criticism, and carefully listen to others

4. Present and communicate their MVP (*3B-IC-25)

a) Create a presentation slide deck for their MVP including the development process and its social influence to general public

b) Present as a group in front of the audience including their peers, friends, family members, teachers, and etc.

B. Metacognitive Level (students will learn to ...)

1. Plan on their timeline and team roles to develop the MVP of their app

2. Evaluate and monitor their process and progress toward MVP, and adjust their plans or designs accordingly

3. Evaluate and monitor the effectiveness of their communication or collaborative procedure and adjust their communication style or collaborative agreement accordingly

4. Self-assess and evaluate their familiarity and work efficiency with the design and app development process

5. Evaluate and monitor the effectiveness of their user survey development, audience selection, distribution methods, and data analysis, and adjust their targeted focus, adoption of tools, or design of questionnaires accordingly

6. Evaluate, reflect, and adjust their presentation slides design and ways of communicating their product and conveying their message



Figure 1. [Storyboarding](#), a way to develop physical prototype of mobile apps

III. Dispositions

A. Cognitive Level (students will gain ...)

1. Responsibility and care as a community member and team collaborator (**)
2. Confidence in dealing with complexity (**)
3. Persistence in working with difficult or open-ended problems (**)
4. Independence in critical thinking and problem solving (**)
5. Politeness when seeking help and patience when helping others

B. Metacognitive Level (students will learn to ...)

1. Evaluate and monitor their own confidence towards a difficult problem, and adjust to growth mindset if necessary
2. Monitor and reflect on how responsible are they in teamwork and adjust if they fail to meet the team's agreement
3. Plan on the balance of help-seeking and independent endeavor in the problem-solving processes, monitor, reflect, and adjust according to problem's difficulty and limitations on time and resources

Assessment: how well do students learn?

Overview of Assessment Evidence

Performance Tasks: <ul style="list-style-type: none">● Project proposal (week 1-7)<ul style="list-style-type: none">○ I-A-2-b, I-A-3-c, I-A-3-b○ II-A-1-a-(1), II-A-1-a-(2), II-A-1-a-(3), II-A-1-a-(4), II-A-2-a,○ II-B-1, II-B-2, II-B-3, II-B-4● Storyboard development (week 2-7)<ul style="list-style-type: none">○ I-A-1○ II-A-1-b, II-B-4● User survey development (week 2-3, and optionally week 6-7)<ul style="list-style-type: none">○ I-A-2-b○ II-A-1-c-(1), II-A-1-c-(2), II-A-1-c-(3), II-B-5● Code review (week 4-5)<ul style="list-style-type: none">○ I-A-3-a-(1), I-A-3-a-(2), I-A-3-a-(3), I-B-2○ II-A-2-b-(1), II-A-2-b-(2), II-A-2-b-(3)● Presentation (week 9-10)<ul style="list-style-type: none">○ I-A-2-a○ II-A-4-a, II-A-4-b, II-B-6	Other Evidence: <ul style="list-style-type: none">● Observe that students are ...<ul style="list-style-type: none">○ Communicating with each other in respectful and efficient manner (after week 1) [II-A-3-a-(1), II-B-3, III-A-1, III-A-5]○ Updating design documentation Google Doc promptly (after week 1) [II-A-3-a-(2), III-A-1]○ Updating programming progress GitHub promptly and with proper comment during development (after week 1) [II-A-3-a-(3), III-A-1, III-A-2, III-A-3, III-A-4]○ Addressing unexpected challenges and difficulty in algorithm or design (after week 1) [I-B-3, II-B-2, II-B-4]● When asked informally, students are able to ...<ul style="list-style-type: none">○ Answer what is their MVP with their own words (after week 2) [I-A-2-a]○ Describe how are user surveys incorporated as part of their iterative design process (after week 3) [I-A-2-b]○ Justify why are they choosing a specific programming language or design tools for their app (after week 5) [I-B-1] Self-Assessment / Reflection <ul style="list-style-type: none">● Workshop exit surveys (every week after workshop) [III-B-1, III-B-2, III-B-3]
--	--

Use of Assessment

Since this unit is going to be implemented in an after school program, no grade or ranking will be given, everyone will have a project completion note, and the team's final MVP project will be the reward that students get. The rubric and assessment results will be used to help project advisors adjust the pace of the workshop, keep track of the students' work, and to provide feedback to students, so that we can maximize students' learning gains.

Design Justification

All of the cognitive conceptual, procedural, or dispositional goals are framed around the heart of learning -- understanding. Understanding is not just about the knowledge and skills, but it's the ability to thoughtfully and actively do the work with discernment, to self-assess, justify, critique such doings, to transfer, and to create new knowledge (Wiggins, Wiggins & McTighe, 2005, p.41).

In order to check students' understanding of core conceptual knowledge or procedural skills, it's important to see whether students can apply the learned knowledge and transfer it to new contexts, so whether students would be able to transfer the knowledge and skill effectively onto their own team project is the crucial part of the assessment design. Therefore, in this unit, a series of authentic performance-based tasks, including [Storyboard development](#), [User survey development](#), [Code review](#), and [Project presentation](#), are all designed to assess students' understanding.

These are authentic challenges since the context of the assessment tasks is realistic: students need to develop the app to address real-world problems of their concerns, and they will “do” the storyboard, user survey, code, and presentation parts of the app development process, which is “complex and multistage” in nature. Furthermore, these tasks are distributed across the 10 weeks to enable “perform-feedback-revise-perform cycles,” so that there will be abundant opportunities for students to “rehearse, practice, consult resources, and get feedback on and refine performances and products.” (Wiggins, Wiggins & McTighe, 2005, p.154).

To facilitate this feedback loop, [Workshop exit surveys](#) can be used: they not only provide students opportunities to regularly reflect on their learning, but they also offer timely feedback that allow instructors to deal with students' misconceptions during the program and push for true understanding (Wiggins, Wiggins & McTighe, 2005, p.195).

Targeting at the metacognitive goals, students would need to improve their metacognitive abilities to assess, evaluate, plan, monitor, and adjust, which are the 5 stages of the self-directed learning cycle proposed in the book *How Learning Works* (Ambrose et. al, 2010, p.193). One of the effective strategies that this book suggested is to let students submit staged progress and plan for their work, which is the purpose of [Project Proposal](#) (Ambrose et. al, 2010, p.207). [Exit surveys](#) are also helpful tools for self-reflection and assessment.

The validity of these assessments are ensured as the cognitive and metacognitive learning goals are aligned with the design, the reliability is strengthened by the inter-rater checks between the two project advisors per team, and equity is enforced as each assessment provides students with multiple opportunities to improve their work and receive feedback, and each unmet criteria is framed as “not yet met” to encourage a growth mindset and prompt advisors to help.

Specific Assessments Overview

Cognition (Content)	Observation (Format, Tasks Yield Evidence)	Interpretation (Rubric, Criteria)
Project proposal [Formative] I-A-2-b, I-A-3-c, I-A-3-b II-A-1-a-(1), II-A-1-a-(2), II-A-1-a-(3), II-A-1-a-(4), II-A-2-a, II-B-1, II-B-2, II-B-3, II-B-4	<p>Students will create a project proposal as a team in the google doc that includes the following 4 components.</p> <ol style="list-style-type: none"> 1. Project Description 2. Timeline Plan 3. Structural Plan 4. Resource List <p>The rubric will be provided to scaffold the proposal writing and structure the assessment; a list of potential resources (e.g., common modules for app development) should be compiled by instructors and made available. The instructor will review the proposal to assess students' understanding of some key knowledge including iterative design principle and MVP, and dispositions for team communication and collaboration. Feedback will be provided during subsequent workshops and students need to further revise the proposal.</p>	<p>Completeness:</p> <ul style="list-style-type: none"> • Does the proposal contain some valid descriptions in all 4 components? <p>Feasibility:</p> <ul style="list-style-type: none"> • Does the timeline cover all 10 weeks and specify each member's roles with justifications of their abilities? • Does the resource list cover the things they'll need in order to develop the app according to their description of their app-specific functionalities (e.g., if they want to do something with computer vision, have they included openCV)? • Is the MVP they specified accomplishable in 10 weeks based on their job division and timeline plan? <p>Clarity:</p> <ul style="list-style-type: none"> • Does the project description and structural plan clearly describe what they will do in the end?
Storyboard development [Formative] I-A-1 II-A-1-b, II-B-4	<p>Students will create a storyboard with ≥ 6 panels, and ≥ 3 of those should demonstrate features of the app. Some paper storyboard examples like Figure 1 will be given as demonstration. Feedback will be provided during subsequent workshops and students need to further revise the storyboard.</p>	<p>Completeness:</p> <ul style="list-style-type: none"> • Does the storyboard contain ≥ 6 panels with ≥3 of them illustrating the app's feature? <p>Clarity:</p> <ul style="list-style-type: none"> • Is the user interaction illustration clear and intuitive?
User survey development [Formative] I-A-2-b II-A-1-c-(1),	<p>Students will create a user survey using google form or other tools, with ≥ 3 questions about the app's main features (based on the storyboard or a code demo), and ≥ 3 questions about design</p>	<p>Completeness:</p> <ul style="list-style-type: none"> • Does the survey include ≥ 3 + 3 questions for the app's main features and design decisions? <p>Reflection:</p>

II-A-1-c-(2), II-A-1-c-(3), II-B-5	<p>decisions, difficulties, or disagreements during the design process, with appropriate format for the questions. Students will also distribute the survey to their friends or other members, interpret the collected user feedback, and update the team design document with the summary and to-do list. Feedback will be given throughout the weeks, when students come up with survey questions, decide on a user sample, discuss the user data, and optionally implement another iteration of user feedback collection.</p>	<ul style="list-style-type: none"> Have students collect the survey and analyze user feedback to incorporate user feedback into design? <p>Quality:</p> <ul style="list-style-type: none"> Is the format of the questions (e.g., multiple choice, yes/no, rating, short answer, etc.) consistent with the question they asked? Is the user sample accurately representing the potential users of their app?
<p><u>Code review</u> [Summative]</p> I-A-3-a-(1), I-A-3-a-(2), I-A-3-a-(3), I-B-2 II-A-2-b-(1), II-A-2-b-(2), II-A-2-b-(3)	<p>Students will be seeing code excerpts from existing online resources and build the functionalities of their own app based on other projects. Instructors will be observing whether the students divide the code into appropriate modules, add comments, adapt to a consistent variable naming scheme, and plan unit tests for their code. To scaffold the process, instructors should compile excerpts with imperfect style and point out its usability, readability, and robustness issues. A sample unit test that covers all edge cases will be performed afterwards for reference and also to check for students' programming products.</p>	<p>Usability (on a percentage scale):</p> <ul style="list-style-type: none"> How many of those large functions have students broken or modularized into small chunks of functions that are ≤ 30 lines each? <p>Readability (on a percentage scale):</p> <ul style="list-style-type: none"> How many flaws in styles have students spotted and corrected? <p>Robustness (on a percentage scale):</p> <ul style="list-style-type: none"> How many edge cases have the test code that students developed cover?
<p><u>Project presentation</u> [Summative]</p> I-A-2-a II-A-4-a, II-A-4-b, II-B-6	<p>Students will be creating a slide deck for their project and MVP, and they will be presenting the slide as a team to a general public audience. They should talk about what their MVP is and how they arrived at it, including their project's goal, social influence, timeline, iterative design based on user research, trade-offs, and reflections to the original plan they made during the process. They should also discuss their collaboration efforts like job division and roles, and attribute individual ideas and</p>	<p>Completeness:</p> <ul style="list-style-type: none"> Does the team complete the MVP they specified? <p>Communication:</p> <ul style="list-style-type: none"> Is the social influence and the purpose of their app clearly communicated? Is the user survey data and feedback used as evidence to justify their app design? <p>Collaboration:</p> <ul style="list-style-type: none"> Does the team communicate individuals' unique contributions

	<p>contributions.</p> <p>Guidelines on presentation slide creation will be discussed in the group (to refresh students' prior knowledge).</p>	<p>to the project?</p> <ul style="list-style-type: none"> Does the team do the job they agreed to do in the project proposal in a responsible and collaborative manner, e.g., update the design & programming documentation regularly?
<p>Workshop exit surveys [Formative] III-B-1, III-B-2, III-B-3</p>	<p>Students will complete one exit survey after each workshop to reflect on 3 main questions about their learning, and some other questions to provide feedback to instructors:</p> <ol style="list-style-type: none"> What are the biggest take-aways you learned from today's workshop? What are the most confusing concepts you found in today's workshop? What might you improve on or prepare for in order to better make use of the workshop's time next week? <p>Students will submit the response to their project advisors in Google Form or paper (remote or in-person).</p>	<p>Completeness:</p> <ul style="list-style-type: none"> Does the student submit responses to all questions? <p>Reflection:</p> <ul style="list-style-type: none"> Does the students' summary on take-aways correctly capture the key concepts (e.g., what is iterative design, MVP, good programming practices, etc. depending on the week) and show a fair understanding? Does the students' reflection on what they can improve on showcases their metacognitive abilities?
<p>Observation: Updating programming progress [Formative] II-A-3-a-(3), III-A-1, III-A-2, III-A-3, III-A-4, II-B-4</p>	<p>Students will need to promptly and use meaningful comments to document their code, and instructors will review their GitHub push history every week to see if they are making proper progress and completing the tasks they agreed to do. Students should clean-up and organize their group repository at least once per week as a team, and they should regularly commit to backup their work during development.</p>	<p>Minimal Completeness:</p> <ul style="list-style-type: none"> Have the student push to their GitHub team repository at least once per week? <p>Adequate Frequency:</p> <ul style="list-style-type: none"> Have students commit to GitHub at least once per hour during the development stage with readable comments on meaningful progress (change \geq 30 lines of code)?

Specific Assessments Rubrics

The team's [Project Proposal](#), [Storyboard](#), and [User Survey](#) should be evaluated using the rubrics in the following sections, which question sections could be made available to students as guidelines, with the assessment criteria (grey area) only visible to advisors, as the questions in rubrics should be used as a guideline to scaffold the discussion and collaborative process, not a limitation on students' creativity or different ways to achieve their product.

For all of these assessments, specific rubrics items are designed to align with the goals, which advisors can review together with the rubrics to enforce the validity of these assessments, that is, each student is assessed based on their individual contributions and group work is assessed with accurate attribution to individual work. Advisors should discuss the rubric items as a team to highlight the boxes they agree upon (in person), or record their assessment using separate google docs each time based on the template rubrics below (remote), which will ensure the reliability of these assessments.

These rubric items and prompts the “Not yet” box are there to help advisors discuss the reflection questions, and work together to generate feedback that they need to provide to students in the beginning of subsequent workshops (as a part of the built-in step 2 & 3 in the [weekly routines](#)). Specifically, advisors should provide students with feedback targeted at the weaknesses in different tasks, such as

- The timeline plan may not be feasible in 10 weeks because this function of the app is unlikely to be implemented in 1 week
- The 4th and 5th questions in your user survey may need to be changed, can you think of ways to improve them (e.g., should be open-ended instead of closed-ended)

But advisors can always ask general guiding questions to scaffold this metacognitive reflection process (assessment as instruction idea from [Big Ideas Synthesis](#)):

- What have you done in XX that's really effective, and why?
- What have you done in XX that's not working, and why?

Therefore, students would have multiple opportunities to further revise their project products like proposal, storyboard, and user survey for 4-5 times each based on advisors feedback and group reflections (please refer to the [Instruction and Assessment Timeline](#) section for specific feedback timelines), which is a practice that fosters equity of these assessment designs on different types of learners.

Project proposal

Advisors can describe project proposal as

- To scaffold your iterative MVP development, your team will create a project proposal in a google doc. You'll need to at least include a Project Description, Timeline Plan, Structural Plan, and Resource List. And here we just listed some guiding questions to help you get started (rubrics).

They can further explain what they mean by MVP, iterative design, and these 4 components as described in the [goal description](#). They should also make it clear that it's an iterative process and students aren't expected to produce a fully developed plan in week 1:

- Try to come up with some ideas on what you want to do, but don't worry if you don't know what would be a feasible timeline or where to find good resources, we'll help you with that based on what you want to do with your app. You'll get a more concrete idea as we talk more about app development as well and you'll have until week 7 to refine your project proposal.

Project Proposal Rubrics		Week:	Team:	Advisors:
Component	Questions	Yes: met criteria		Not yet: how to help?
Project Description I-A-2-b, II-A-1-a-(1)	Completeness: Does the proposal contain valid descriptions of this component?	Yes: students described their goals and MVP in a Project Description section.		No: how might you help students better frame their MVP next week?
	Clarity: Does the description of MVP clearly describe what they will do in the end?	Yes: students' descriptions are understandable to non-team members.		No: what needs to be further clarified about their MVP? How would you discuss it next week?
Timeline Plan II-A-1-a-(2), II-B-1	Completeness: Does the proposal contain valid descriptions of this component?	Yes: the timeline plan cover all 10 weeks of workshops		No: which weeks' plans are unclear? How may you help them better develop a sense of timeline?
	Feasibility: Is the MVP they specified accomplishable in 10 weeks based on their timeline plan?	Yes: every week's breakdown is reasonable and accomplishable based on students' abilities.		No: which weeks' plans are less feasible and why? How may you help them to better scope their MVP and distribute weekly work?

Structural Plan I-A-3-c, I-A-3-b, II-A-1-a-(3)	Completeness: Does the proposal contain valid descriptions of this component?	Yes: each member's roles and the organizational format of their MVP are specified.	No: how may you help each student figure out how they could make unique contributions? How can they improve the organizational structure?
	Clarity: Does the structural plan clearly describe what they will do in the end?	Yes: their description of their final product organization is understandable.	No: what is causing the confusion or in clarity? How may you help them better frame their structure?
	Feasibility: Is the MVP they specified feasible based on their job division with justifications of their abilities?	Yes: their described job division would enable them to accomplish their MVP in 10 weeks.	No: how may you help them to learn about each other's strengths and weaknesses to optimize their job divisions?
Resource List II-A-1-a-(4), II-A-2-a	Completeness: Does the proposal contain valid descriptions of this component?	Yes: they've started to compile a list of helpful resources.	No: how might you point them towards helpful resources and good strategies to search for resources?
	Feasibility: Does the resource list cover the things they'll need to develop the app?	Yes: they listed all necessary resources for their MVP-specific functionalities	No: what have they left out? (e.g., if they want to do something with computer vision, they need openCV)
General II-B-1 II-B-2 II-B-3 II-B-4	Reflection (after 1st week): Have students addressed the feedback from the previous week in all components?	Yes: students addressed all feedback discussed and improved on each section.	No: what have they missed and why they missed it? How might you emphasize or effectively communicate your advice next week?

Storyboard development

Advisors can describe storyboard development as

- To scaffold your MVP development, your team will create a storyboard on paper or digital platform (show examples like [Figure 1](#), and explain key terms like storyboard as described in the [goal description](#) if the students are not familiar with it). You'll need to at least include at least 6 panels with more than 3 showcasing major app features. And here we just listed some guiding questions (rubrics) to help you get started.

Again, advisors should also make it clear that it's an iterative design process and students aren't expected to produce a fully developed plan in week 1:

- Don't worry if you don't know what would be a feasible feature for your app, we'll help you with that based on the goal of your app. You'll develop more concrete ideas as we talk more about app development and you'll have until week 7 to refine your storyboard.

Storyboard Development Rubrics		Week:	Team:	Advisors:
Questions	Yes: I-A-1, II-A-1-b, II-B-4	Not yet: how to help?		
Completeness: Does the storyboard contain ≥ 6 panels with ≥ 3 of them illustrating the app's feature?	Yes: students included all necessary components of the storyboard.	No: what's the point of confusion or difficulty? How might you help students better understand the idea of storyboarding next week?		
Clarity: Is the user interaction illustration clear and intuitive?	Yes: students' descriptions are understandable to non-team members.	No: what needs to be further clarified in their storyboard? How would you discuss it next week?		
Reflection (after 1st week): Have students addressed the feedback from the previous week?	Yes: students addressed all feedback about the storyboard in discussion and improved this section.	No: what have they missed and why they missed it? How might you emphasize or better communicate your advice next week?		

User survey development

Advisors can describe user survey development as

- To make your MVP app more usable and apply the iterative design principles, your team will create user surveys using google forms or other tools of your choice (explain key terms like [iterative design](#) and [user survey](#) to activate students' [prior knowledge](#)). You'll need to at least include at least 3 questions about the app's main features based on your

storyboard or app demo and at least 3 questions to help your design decisions. For example, if you don't know whether to use a button or to use a slider, ask the users!

- Here we just listed some guiding questions (rubrics) to help you get started (activate student's [prior knowledge](#) for data collection and analysis).
- More questions: What are some common format of user survey questions? Who's your target audience? What would you do after you've collected the data? How would you use user surveys to iteratively improve your MVP?

Advisors should also mention that they may do the user surveys for 1 or 2 rounds depending on their progress and development of MVP:

- You'll do the user research from week 2 to 3 and if the app development went well, another round on week 6 to 7. Usually in a design process you'll want more iterations of user feedback and improvement, but given our limited time, you should think about how you would best use the time and put it into your timeline plan of the project proposal.

User Survey Development Rubrics		Week:	Team:	Advisors:
Questions	Yes: I-A-2-b, II-A-1-c, II-B-5	Not yet: how to help?		
Completeness: Does the survey include $\geq 3 + 3$ questions for the app's main features and design decisions?	Yes: students included all necessary components of the user survey.	No: what's the point of confusion or difficulty? How might you help students better understand the idea of user research next week?		
Quality: Is the format of the questions consistent with the question they asked?	Yes: students' use of question formats are appropriate (e.g., multiple choice, yes/no, rating, short answer, etc.)	No: what causes the inconsistency (e.g., a misunderstanding on how to ask open questions)? What are some better ways to frame their questions and how would you discuss it with students?		
Quality: Is the user sample accurately representing the potential users of their app?	Yes: students identify the appropriate group of users for their MVP.	No: what's inaccurate of their user target based on their MVP goals? How may you help them to find the right group?		
Reflection (after 1st week): Have students addressed the feedback from the users and advisors?	Yes: students collected the survey and analyzed user feedback to incorporate user feedback into design.	No: what have they missed and why they missed it? How might you emphasize or better communicate your advice next week?		

Code review

Code excerpts with purposeful impairment on the usability, readability, and robustness will be provided, and students will submit their revised code with a unit test to GitHub. The code needs to be compilable, but the usability, readability, and robustness can then be assessed independently (e.g., it may be only 30% robust but 90% usable). Since this activity requires individual work and has a reference solution (the unmodified version of code), the advisors can split the work and switch to assess different students in different weeks.

Advisors will review individual students' revisions using the following rubrics, record assessments in google docs, and provide targeted feedback to individual students to help them better understand the concept of code usability, readability, and robustness. This activity will be conducted during step 4 of [weekly routines](#) on week 4 and 5 to provide multiple opportunities.

Code Review Rubrics		Week:	Student:	Advisor:
Questions	90%: met goals I-A-3-a, I-B-2, II-A-2-b	60%: what went wrong?	30%: how to help?	
Usability: How many large functions have students broken into small chunks?	90% of the code is modularized into reusable functions \leq 30 lines each.	What's the main point of confusion or difficulty?	How may you help students understand modularization and the importance of code reuse?	
Readability: How many flaws in styles have students spotted and corrected?	90% of the code functions have consistent variable naming and readable comments.	What's preventing them from converting all functions into readable formats?	How may you help students understand the concept and importance of code readability?	
Robustness: How many edge cases have the test code that students developed cover?	90% of the bugs were caught by their unit test, or 90% test cases passed their code.	Which part of the code have the students failed to cover with their unit tests?	How may you help students understand the concept and importance of unit testing and write better test cases?	

Project presentation

Students will practice their presentation during the week 10 workshop, in which both advisors will assess their practice presentation and also the team's summative work in the past 9 workshops. Advisors should record their assessment results in google docs and provide immediate feedback after students' presentation in the group discussion (step 3 of [weekly routines](#)).

Advisors should talk about students' strengths and highlight their efforts and learning gains during this collaborative process; the weaknesses of their projects should also be discussed (e.g., advisors can selectively use some guiding questions in the "Not yet" box to scaffold the group discussion), but this should more be a group reflective moment to identify rooms and suggest strategies for improvements, and further motivate students and help them develop metacognitive abilities.

Final Project Rubrics	Week: 10 Team:	Advisors:
Questions	Yes: met goals I-A-2-a, II-A-4-a, II-A-4-b, II-B-6	Not yet: how to improve?
Completeness: Does the team complete the MVP they specified?	The team has completed the MVP they specified in their project proposal.	What's the main difficulty? What're some lessons learned? What should be avoided, what's effective, what could have been done better?
Communication: Is the social influence and the purpose of their app communicated?	The team has clearly and effectively conveyed the importance of the problem their app targeted at.	What's their original goal for MVP? How does it help to improve society? What's ineffective in communication and how to improve?
Communication: Is the user survey data presented to justify their app design?	The team showcased how they incorporate user feedback with evidence of design adjustments.	What are some key designs of this MVP that address users needs? How to better visualize their collected user data and iterative design?
Collaboration: Does the team communicate individuals' unique contributions?	The team has each member talk about their own contributions to the project and job divisions.	How have they divided the jobs? Who has done which part of the work? How to better discuss and attribute individual and group work?
Collaboration: Does the team do the job in the project proposal in a responsible and collaborative manner?	The team has updated the design & programming documentation regularly, members do their agreed part of work promptly.	What's the main difficulty and obstacle in their collaboration? How they addressed it and how may they do better? What collaborative skills and dispositions have they learned?

Workshop exit survey

After each workshop, students will complete and submit a workshop exit survey (paper form if in-person, Google Form if remote) adopted from the 1-minute essay idea and also the Figure 11.8 Weekly Feedback Form in *Understanding by Design* (Wiggins, Wiggins & McTighe, 2005).

Project advisors should choose one of the 2 questions for question 1-2 in the same box based on their feeling of the workshop or simply alternate different versions across the weeks. For example, if the workshop went smoothly, they may choose V1, and if they feel like students really struggle with the concepts, they may choose V2 to probe what went wrong.

Advisors should review the survey questions 1 & 3 with the rubrics (only available to advisors) to gauge students' metacognitive abilities to reflect their learning, their understanding of the key concepts, and muddy points or questions that need to be addressed, so that they can provide clarifications in the beginning of the next workshop (which is a built-in procedure in the [weekly routines](#)). For questions 2 & 4, advisors should use them as feedback to iteratively improve their instruction and adjust their lesson plans accordingly.

Workshop Exit Survey		Week:	Name:
Question 1: <ul style="list-style-type: none">• [V1] What are the biggest take-aways you learned from today's workshop?• [V2] What are the most confusing concepts you found in today's workshop?			
Question 2: <ul style="list-style-type: none">• [V1] What worked the best for you this week? What activity, assignment, technique, tool, or discussion was the most interesting or helped you learn the most? Why?• [V2] What didn't work for you this week? What activity, assignment, or discussion was the most confusing, boring, or unhelpful? Why?			
Question 3: What might you improve on or prepare for in order to better make use of the workshop's time next week?			
Question 4: Please answer Yes or No to the statements below. Please suggest how we might improve and further help you for any No response.			
We were given enough freedom in how to go about achieving our goals.		Yes	No
We were given enough scaffolding so that I'm clear about where we are coming from, where we would go next and how exactly we could achieve it.		Yes	No

Workshop Exit Survey Rubrics			Week:	Student:	Advisors:
Question	Yes: III-B-1, III-B-2, III-B-3	No: what to do?			
Does the student submit responses to all of the questions?	Yes: completed in time and write meaningful responses (not just N/A's).	No: how might you motivate students to complete exit surveys?			
Does the students' summary on take-aways correctly capture the key concepts of the week and show a fair understanding?	Yes: students can explain in their own words on concepts such as iterative design, MVP, good programming practices, etc.	No: what's missed or misunderstood? How would you discuss it in next week's workshop?			
Does the students' reflection on what they can improve on showcases their metacognitive abilities?	Yes: students accurately identify weaknesses or inefficiencies and propose better strategies.	No: what are some problems they noticed or failed to identify? What are some strategies to solve them?			

Instruction: how should we teach?

General Description

Classroom Climate

If the workshops are in-person, students will be using CMU classrooms or library study rooms open for reservation, where they have access to all sorts of multimedia device like projector, whiteboard, individual laptops (sponsored or students' own devices), and convenient tech support; they would also have access to the makerspace or laboratory resources on campus so if their app include interdisciplinary components of robotics, they will have the resource to do so. It's important to have abundant maker resources at hand if needed, as illustrated in the *Making* chapter of *The ABCs of How We Learn* (Schwartz, Tsang & Blair, 2016). These classrooms or conference rooms would have the right capacity for 5-7 people and students can sit around the table to establish an atmosphere of equal contribution and interactive collaboration in the workshops, rather than a lecturing style.

If workshops are remote, students are assumed to have necessary digital devices like laptop, microphone, and camera. They will join the zoom rooms for weekly workshops, and breakout rooms can be used to enable multi-threads discussion in the meantime, so advisors can provide more targeted and personal feedback when needed and give students more chances to talk .

Advisors are encouraged to bring students for a walk on campus as a break (if in person), or play some ice-breaker online games (if remote), which allow students to chat more casually with peers and develop a sense of belonging and facilitate modeling (Clarke-Midura et al., 2018).

Weekly Routines

The overall routines for each week would be a 1 hr workshop during the weekend, and expected 3 hrs of work during the following weekdays. Students should do the job they agreed to do during the workshop collaboratively, and they are encouraged to use email or social media to communicate with each other. The routine for each workshop would be:

1. [5 minutes] Greeting & Overview
2. [10 minutes] Individual updates on prior week's work & questions & feedback from project advisors on students' progress, assessment tasks, and misconceptions
3. [10 minutes] Group discussions and reflections on the revisions and next steps
4. [10 minutes] Advisors' brief tutorial or activities of important new concepts, tools, or practices (basics in earlier workshops and tailored to the team's MVP in later workshops)

5. [5 minutes] Break and casual conversations
6. [15 minutes] Group work on the next tasks & MVP, advisors provide guidance alongside
7. [5 minutes] Closing, Summary & Workshop Exit Surveys

Besides the greeting and break, the 2nd step is a coaching style that supports practice with feedback, the 3rd step is open exploration, the 4th step here involves more direct instruction, and the 6th step is facilitation via guided discovery.

Key Design Elements

The book *Understanding by Design* proposed an useful framework (WHERE TO) for key elements in instructional design (Wiggins, Wiggins & McTighe, 2005, p.28); the instruction activities of this unit have all these elements implemented, and the design is also aligned with the learning principles in *How Learning Works* (Ambrose et. al, 2010).

<ul style="list-style-type: none"> • Where & Why? The project proposal that students would develop is an instruction that guides the purpose finding and navigation process for students, since when they discuss, specify, receive feedback on, and modify their MVP and timeline, they will gain a clearer and clearer idea about where does the project come from and where will it go.
<ul style="list-style-type: none"> • Hook: The project proposal would hook and hold students' interests since they will be coming up with an app of their design to address the problem they care about.
<ul style="list-style-type: none"> • Equip: All of the instructional pieces, either directed, facilitated, exploratory, or coached, such as programing tutorial, user feedback analysis, proposal writing, and stroyingboard development, will equip students to explore and experience the iterative process of app development and help them complete their MVP.
<ul style="list-style-type: none"> • Rethink / Reflect / Revise: The feedback sections and exit tickets in each workshop are opportunities for students to receive timely feedback, practice metacognitive abilities, and refine their work.
<ul style="list-style-type: none"> • Evaluate: The development and constant refinement of their proposal, the collection of user feedback, and the project presentation at the end of all workshops are all opportunities for students to evaluate their work.
<ul style="list-style-type: none"> • Tailor to Context & Learner Characteristics: The direct instructions on app development & programming practices will be tailored around the tools that the students would find helpful, and the exploratory or guided discovery will all be highly tailored around the team's project & MVP.
<ul style="list-style-type: none"> • Organize to Optimize: There will be a break by 40 minutes of workshops to enable sustained engagement, and the routines are all very interactive to maximize effective learning.

Instruction and Assessment Timeline

Goal Alignment	Instruction	Assessment
Week 1 I-A-1, I-A-2-b, I-A-3-c, I-A-3-b, II-A-1-a-(1),(2),(3),(4), II-A-1-b, II-A-2-a, II-B-1,-2,-3,-4 III-B-1,-2,-3	During Workshop: <ul style="list-style-type: none"> • Collaboration Agreement & Design Practices • Project Proposal for MVP • Storyboard Development After Workshop: <ul style="list-style-type: none"> • Project Proposal Assignment • Storyboard Assignment 	During Workshop: <ul style="list-style-type: none"> • Observation (communication) After Workshop: <ul style="list-style-type: none"> • Exit Survey
Week 2 I-A-1, I-A-2-b, II-A-1-a-(1),(2),(3),(4), II-A-1-b, II-A-1-c-(1),(2),(3), II-A-2-a, II-B-1,-2,-3,-4 III-B-1,-2,-3	During Workshop: <ul style="list-style-type: none"> • Proposal & Storyboard Revision • User Survey Development • App Development & Programming Practices (basics) After Workshop: <ul style="list-style-type: none"> • User Survey Assignment • MVP Development 	During Workshop: <ul style="list-style-type: none"> • Observation (communication) • Project Proposal Feedback • Storyboard Feedback After Workshop: <ul style="list-style-type: none"> • Exit Survey
Week 3 I-A-2-b, II-A-1-a-(1),(2),(3),(4), II-A-1-c-(1),(2),(3), II-A-2-a, II-B-1,-2,-3,-4,-5 III-B-1,-2,-3	During Workshop: <ul style="list-style-type: none"> • User Survey Revision & Analysis Discussion • Proposal & Storyboard Revision After Workshop: <ul style="list-style-type: none"> • MVP Development 	During Workshop: <ul style="list-style-type: none"> • Observation (communication, design, programming) • Project Proposal Feedback • Storyboard Feedback • User Survey Feedback After Workshop: <ul style="list-style-type: none"> • Exit Survey
Week 4 I-A-3-a-(1),(2),(3), I-B-2, II-B-1,-2,-3, II-A-2-b-(1),(2),(3), II-A-3-a-(3), III-A-1,-2,-3,-4 III-B-1,-2,-3	During Workshop: <ul style="list-style-type: none"> • App Development & Programming Practices (tailored) After Workshop: <ul style="list-style-type: none"> • MVP Development 	During Workshop: <ul style="list-style-type: none"> • Observation (communication, design, programming) • Code Review After Workshop: <ul style="list-style-type: none"> • Exit Survey
Week 5 II-A-3-a-(3), II-B-1,-2,-3,-4, III-A-1, III-A-2, III-A-3, III-A-4	During Workshop: <ul style="list-style-type: none"> • Proposal Revision • App Development & Programming Practices (tailored) 	During Workshop: <ul style="list-style-type: none"> • Observation (communication, design, programming) • Project Proposal Feedback • Code Review

III-B-1,-2,-3	After Workshop: <ul style="list-style-type: none"> • MVP Development 	After Workshop: <ul style="list-style-type: none"> • Exit Survey
Week 6 I-A-2-b, II-A-1-c-(1),(2), (3), III-B-1,-2,-3,-5	During Workshop: <ul style="list-style-type: none"> • App Development & Programming Practices (tailored) • (optional) User Survey 2nd Iteration After Workshop: <ul style="list-style-type: none"> • MVP Development • (optional) User Research Implementation 2nd Iteration 	During Workshop: <ul style="list-style-type: none"> • Observation (communication, design, programming) • User Survey Feedback After Workshop: <ul style="list-style-type: none"> • Exit Survey
Week 7 I-A-2-b, II-A-1-c-(1),(2),(3), II-B-5 III-B-1,-2,-3	During Workshop: <ul style="list-style-type: none"> • User Survey Revision & Analysis Discussion • Proposal & Storyboard Revision After Workshop: <ul style="list-style-type: none"> • MVP Development 	During Workshop: <ul style="list-style-type: none"> • Observation (communication, design, programming) • Project Proposal Feedback • Storyboard Feedback • User Survey Feedback After Workshop: <ul style="list-style-type: none"> • Exit Survey
Week 8 II-A-3-a-(3), III-A-1, III-A-2, III-A-3, III-A-4 III-B-1,-2,-3	During Workshop: <ul style="list-style-type: none"> • App Development & Programming Practices (tailored) • MVP Finalizing & Polishing After Workshop: <ul style="list-style-type: none"> • MVP Development 	During Workshop: <ul style="list-style-type: none"> • Observation (communication, design, programming) After Workshop: <ul style="list-style-type: none"> • Exit Survey
Week 9 I-A-2-a, II-B-6 III-B-1,-2,-3	During Workshop: <ul style="list-style-type: none"> • Project Presentation Guideline • MVP Finalizing & Polishing After Workshop: <ul style="list-style-type: none"> • MVP Development 	During Workshop: <ul style="list-style-type: none"> • Observation (communication, design, programming) After Workshop: <ul style="list-style-type: none"> • Exit Survey
Week 10 II-A-4-a, II-A-4-b, II-B-6 III-B-1,-2,-3	During Workshop: <ul style="list-style-type: none"> • Project Presentation Practice After Workshop: <ul style="list-style-type: none"> • Project Presentation Preparation 	During Workshop: <ul style="list-style-type: none"> • Observation (communication, design, programming) • Project Presentation Feedback After Workshop: <ul style="list-style-type: none"> • Exit Survey

Design Justification

The whole unit is centered around project-based learning, and students would be coming up with their own idea of an app, so that they have high intrinsic motivation. Since the specific problem that the students' app will address depends on their own interests, students have the internal drive to solve the challenges they encounter in app development and are naturally motivated to learn more, so the instructional activities of this unit could be tailored to spur students' intrinsic motivation. Also, the highly discussion-based [weekly routines](#) are designed to keep students' sustained engagement, which involve the highest level of cognitive engagement, i.e., interactive learning, according to the ICAP framework (Chi & Wylie, 2014).

The instructions all focus on facilitating students' authentic understanding of app development, so numerous practice, feedback, reflection, and revision opportunities are built-in. Regular feedback loops address students' misconceptions and enable them to improve metacognitive abilities. Although this unit is collaborative in nature, the group is so small (4-5 students per team) that different students' needs, interests, and capabilities can receive abundant attention from project advisors. Therefore, the sufficient discussion time and small size of project teams allow advisors to provide personalized feedback and be responsive to individual variability in learning strategies and challenges.

According to Barron et al. (1998), good project-based learning in schools include the following four elements: (1) learning-appropriate goals; (2) learning resources; (3) multiple opportunities for feedback and revision; and (4) a social environment that supports original production and interaction. This unit's instructional activities are not only aligned with the learning objectives, but they also provide ample built-in opportunities for timely feedback, reflections, goal setting, existing app development resources for appropriation, and a peer- and near-peer-interaction environment to foster collaboration, innovation, and the dispositional skills of accepting failures and constructive criticisms. These designs are also consistent with the major elements that facilitate learning identified in the *Making* chapter of *The ABCs of How We Learn* (Schwartz, Tsang & Blair, 2016).

Additionally, the [weekly routine](#) of practice with coaching (2nd step), exploration (3rd step discussion), and then direct instruction and explanations (4th step) is consistent with the *Just-In-Time Telling* idea, as students would be able to accumulate experiences and understand the contexts of the problems before the instructors provide high-level structures for the phenomena, present diverse *Worked Examples*, or explain the elegant algorithmic solutions of the problem (Schwartz, Tsang & Blair, 2016).

Furthermore, this discussion routine enables advisors to build on students' prior knowledge and promptly address misconceptions as reflected by their responses to last week's exit surveys

(Ambrose et. al, 2010), where the [workshop exit survey](#) reflects the “assessment as instruction” idea, as it’s not only an assessment tool for instructors, but also a learning opportunity for students to practice their metacognitive skills (Wiggins, Wiggins & McTighe, 2005).

Also note that the [weekly routine](#) involves an overview and a summary of the workshop in the beginning and the end, which follows the nice whole-part-whole learning circle that facilitates understanding and knowledge organization (Wiggins, Wiggins & McTighe, 2005). And the greeting in the beginning and the casual conversation break in the middle of workshops can contribute to a welcomed and inclusive norm and help build a sense of community (Ambrose et. al, 2010).

Specific Activities

Collaboration agreement & design practices

In Week 1 (1st workshop), use step 1-2 of the [routine](#) to introduce the general structure of workshops, give a high level overview of the 10 weeks and goals, and let students get to know each other and break the ice with games like 2 truths 1 lie. Additionally, since there is such a large focus on collaboration in the project, advisors should use step 3 guided discovery to activate students’ [prior knowledge](#) on collaboration and design practices (activate prior knowledge idea from [Big Ideas Synthesis](#)); for example, advisors can ask guiding questions like the below ([some](#) adopted from EGIA):

- What’s each member’s collaboration, working, and communication style?
- What contributions will each member make?
- What practices will make everyone comfortable? What may make one uncomfortable?
- Best times of day to collaborate? Best way to communicate?
- Any commitments that the team universally agree on?
- What’re some important concepts and practices in design?
- What makes a design successful? What makes a design process smooth?

Note that students may come from very different backgrounds and have different working styles, make sure the individual differences are discussed and respected, and the team should establish and document a clear norm on how to collaborate (e.g., put into a google doc).

In step 4 of the [routine](#), advisors can combine direct instruction with guided discovery to explain the use of design documentation and iterative design principles if students didn’t discuss them during step 3, for example:

- It’s important to keep track of the design process and document everyone’s ideas and tasks clearly in a collaborative design process. Can anyone think of some benefits of it?

Project proposal, Storyboard, User survey (development & revision)

These three components of the workshops are the clear illustration of the assessment as instruction idea ([Big Ideas Synthesis](#)), as the working development and the feedback loop of these components are both important assessment tools and instructional activities in the meantime. As discussed with the specific rubrics in the [previous sections](#), the instructions given around these components are primarily practice & feedback and guided discovery.

Project advisors should tailor the feedback according to students' project and with the help of rubrics. The group discussion format can also nicely accommodate students with mildly different experience level, as students who understand the concepts better can consolidate their knowledge in explanations (Wiggins, Wiggins & McTighe, 2005), and students who are less comfortable with the idea of timeline organization, storyboard development, or user survey analysis will be able to explore and understand the contexts before receiving the explanation and learning the conceptual framework of design (*Just-In-Time Telling*, Schwartz, Tsang & Blair, 2016).

Next semester will be the first time that this unit is being offered in Project Ignite, so we wouldn't have sample students' work for these 3 components. But students can always search for online resources and templates to build more intuition on what a good (proposal, storyboard, survey) looks like, which could be an open exploration activity in step 3 or 6 of the [routine](#).

App development & programming practices (basics / tailored)

The app development and programming practices activity will be direct instruction combined with practice & feedback and guided discovery, it should be tailored according to the time of the workshop and students' needs. For example, advisors can start off providing more direct instruction as a basic overview of app development languages, tools, platforms, the ideas of code usability, readability, and robustness, and with the [code review](#) assessment, advisors can incorporate some guided discovery and practice & feedback to help students discover the problems in code design, such as asking questions like:

- What may be some potential problems with this code?
- Remember the usability, readability, and robustness issue we just talked about? Could you try to explain your understanding of these concepts in your own words? (explanation and active learning idea in [Big Ideas Synthesis](#))
- (if students encounter a specific bug in their code) What have you tried to debug? What do you think could potentially go wrong? What are you confused about?

They may increase the level of difficulty or introduce some advanced functionality for their MVP if they have a group of students with more programming skills or in later half of the workshops.

For direct instructions, advisors can consult existing resources to create slides and talk students through some important concepts. If students decide to develop an iOS app, advisors can use Apple's [handbook](#) for XCode and Swift, or adopt the lecture notes of CMU's student-taught course [98-222](#) (Introduction to iOS development) for demo examples and glossary explanations.

Advisors can project (if in person) or share (if remote) their screens so that students can see a *Worked Example* (Schwartz, Tsang & Blair, 2016), which will help students see the exact visual experiences they'll have interacting with the app development platforms.

For example, an advisor can explicitly explain how to use breakpoint to debug in XCode by saying the following, while presenting the screen like in Figure 2.

- Programmers use breakpoints to check values or perform other debugging operations, where a breakpoint is an intentional stopping or pausing place in a program.
- We can create a breakpoint by clicking here (mouse pointing) on the left of the line where you want execution to pause. In this case, we added the breakpoint to the line `var names = ["Tammy", "Cole"]`

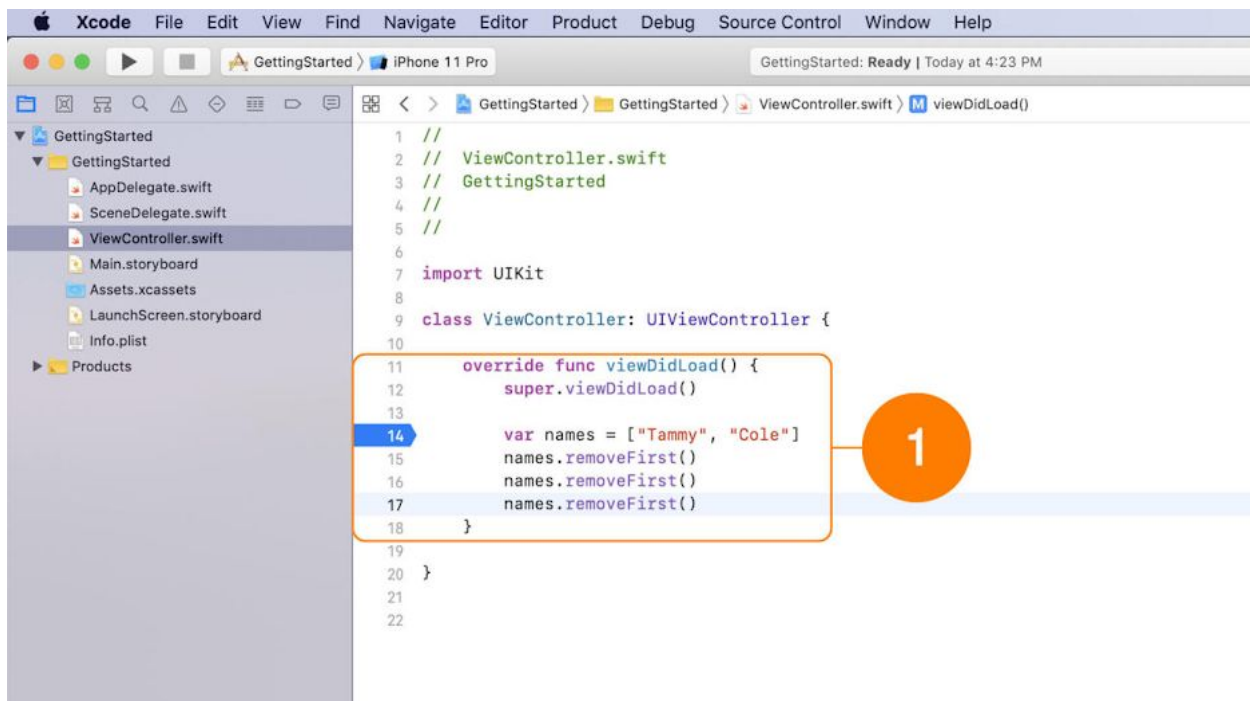


Figure 2. [Breakpoint](#), an intentional stopping or pausing place in a program 



Project presentation guideline & practice

Since we assume some [priori knowledge](#) of presentation in students, the advisors only need to use guided discovery and practice & feedback with the [rubrics](#) to help students refresh the important concepts in presentation in more systematic ways.

Advisors can provide feedback in terms of the slides deck they created, the organizational structure of the presentation, and/or the way they describe their product or each member's contributions, for example:

- Could you provide an overview of the whole presentation before you go into the specifics? The preview can better prepare the audience for your following talk and create a mental framework to help them understand your project.
- You should synthesize the words on slides a little bit more, and use bullet points instead of paragraphs. Remember that the design principles for presentation slides is “less is more,” and it shouldn't occupy too much attention from the audience.

Evaluation: how could we improve?

Research to Evaluate Educational Impact

Research Questions and Hypotheses

As described in the [Design Justification](#) section, workshop exit surveys are both effective as assessment and instructional tool: they potentially provide students with more opportunities of reflection, which means more practices with and thus improvements on their metacognitive abilities; they offer students chances of directly providing feedback and communicate with instructors to shape their learning experiences, so it gives students more senses of control of their learning and higher esteem as persons with equal power, and thus increases their level of engagement during workshops and satisfaction of their project.

But how beneficial are these surveys in practice? Some of the above benefits depend on instructors' feedback or responsiveness to the surveys, so the influence of exit surveys may be mitigated by instructors' responses (e.g., if instructors simply ignore the surveys students filled, the engagement and satisfaction benefits may be lost). Therefore, we proposed the following research questions and hypotheses:

Research Questions:

1. Does the use of [workshop exit surveys](#) improve students' metacognitive abilities, level of engagement, and satisfaction with their project?
2. Are workshop exit surveys more effective when instructors are responsive to the surveys?

Hypotheses:

1. The use of workshop exit surveys improves students' metacognitive abilities, level of engagement, and satisfaction with their project.
2. The positive effects of workshop exit surveys on students' metacognition, engagement, and satisfaction increase when instructors are responsive to the surveys.

Experimental Design

Independent variables (active ingredient): (1) the use of workshop exit survey, and (2) instructors' responsiveness to the surveys.

Dependent variables: students' (1) metacognitive abilities, (2) engagement, and (3) satisfaction with project.

Experimental groups & control group

Since each project team only has 4-6 students and it's impractical to make all project teams focus on one topic, the statistical power will be too small to produce significant results if we split the experiment and control groups by project teams. As Project Ignite can recruit 50-60 students each year and hold 7-10 project teams, students in each team will be randomly assigned with stratification into each of the following 3 conditions, so in total we have about 15-20 students in each condition.

1. **Treatment group SO** (survey only): individual emails of exit surveys will be sent after workshops to students in this group, their responses will not be available to advisors, and the discussion will be led by an advisor who hasn't reviewed any surveys and whose workshop checklist will have the 7th step (exit survey reflection) removed.
2. **Treatment group SR** (survey and responsive instruction): exit surveys will be sent to students after workshops, and their responses will only be reviewed by the advisor in each team who will lead the discussion for the SR group and be reminded to address students' feedback.
3. **Control group CR**: students in this group will not receive any exit surveys, and the discussion will be led by the same advisor as the SO group.

Design quality assessment

Validity

In-person setting will have low ecological validity for this experiment, since separating the team into individual rooms is very unnatural, but having parallel discussion thread in the same room can interfere each other and introduce many more confounders (Nathan & Alibali, 2010), so this experiment will be implemented in a remote format, where breakout rooms can easily separate the group with a rationale of providing more individual attention and parallel discussion (step 2 of the [weekly routines](#)). However, the sample size may still be too small to be representative of the general population or to really have stratified random sampling, which may harm the external validity.

Besides all assessments designed in the [Assessment](#) section, established inventories could be used to assess students' metacognitive abilities, engagement, and satisfaction (Puzziferro, 2008; Pellas, 2014) to ensure the validity of this evaluation research.

A [weekly workshop checklist](#) will also be used to ensure that the workshop exit surveys are being distributed and collected in SO and SR groups, and SR group advisors are specifically prompted to provide feedback.



Reliability

The existing self-assessed inventories should have high test-retest reliability, and inter-rater reliability can be checked by having multiple researchers observe and assess students' metacognitive abilities, engagement, and satisfaction. However, a double-blinded procedure will be hard to implement here since with the counterbalancing design, advisors could have guessed the research questions, which may harm the reliability of this experiment.

Confounding variables (covariates)

A counterbalanced design should be used so that advisors will switch for the discussion team they lead each week to control for the instructional styles of advisors themselves. But the individual differences of instructors and students in terms of responsiveness and feedback styles may still confound the effect of workshop exit surveys (e.g., students may explicitly tell the instructors their feedback without the surveys, or instructors may address students' feedback without reading the surveys just by experiences or anticipation).

To account for this problem, we can have another form to keep track of the number of feedback attempts advisors provided and the number of students' survey responses questions being addressed during the discussion section. We can analyze these data to see if it reflects our hypotheses, i.e., these numbers are higher in the SR group than both SO and CR groups.

Research to Evaluate Educational Implementation

To ensure both instructors and students followed or participated in the proposed instruction, assessment, and evaluation research design as anticipated, we require project advisors to collaboratively complete the *weekly workshop checklist* below. The executive board members of Project Ignite, specifically the President and Vice President of Projects, should also sit in several workshops randomly to evaluate the implementation, especially of the proposed research.

The forms can be either electronic or in paper, depending on the evaluators' preferred form of data collection, but an e-copy will be made and uploaded to the organization's google drive for future references. The advisors should partially fill out the weekly workshop checklist during the workshop to keep track of time and comments (advisors should collaborate, e.g., one may lead the conversation while the other record in the checklist) and complete it immediately after the workshop (while the students are doing workshop exit surveys) to prevent memory decay.

The E-board should observe the workshop unobtrusively and take the note quietly to avoid interference to the activities, and feedbacks should be provided to advisors in a separately scheduled progress checking meeting (e.g., in the 3rd and 6th week), in which the advisors, President, and Vice President of Projects will review all previously compiled forms together.

Weekly workshop checklist

Note that instructors will complete this individually, in order to remove the 7th step exit survey reflection section for the advisors leading the discussion for the SO and CR groups next week, so that only the advisor leading the SR group will be prompted to address students' surveys and provide targeted feedback.

Workshop Checklist	Team:	Week:	Date:	Advisors:	[SR]
Weekly Routines & Logistics					✓ done
[5 minutes] Greetings, overview, attendance taking (google sheet)					
[10 minutes] Individual updates, questions, feedback Students' questions (name): <ul style="list-style-type: none"> • • Advisors' feedbacks: <ul style="list-style-type: none"> • • 					
[10 minutes] Group discussion, reflections, revisions, next steps Students' reflections (name): <ul style="list-style-type: none"> • • Students' proposed next steps (name): <ul style="list-style-type: none"> • • 					
[10 minutes] Brief tutorial or activities (content): <ul style="list-style-type: none"> • • Additional comments (e.g., difficult to digest, request to slow-down/speed-up): <ul style="list-style-type: none"> • • 					
[5 minutes] Break, casual conversations					
[15 minutes] Work on the next tasks & MVP Students' questions (name): <ul style="list-style-type: none"> • • Additional comments (e.g., discover misconceptions, unclear about directions): <ul style="list-style-type: none"> • • 					

[5 minutes] Closing, summary, workshop exit surveys distributed and collected Useful students' feedback from surveys: <ul style="list-style-type: none"> • • How would you address them in next workshops: <ul style="list-style-type: none"> • • 	
Reflections & Ratings (1 the lowest - 5 the highest)	1-5
Quality of participation (if > 3, what engaged students the most? if ≤ 3, where did you lose students? What would you do differently?) [Q 1, 2, 4 of the workshop exit survey may be partial evidence for engagement] <ul style="list-style-type: none"> • • 	
Other notes (What went well? What went wrong? What to improve?) <ul style="list-style-type: none"> • • 	

This checklist above is the version that SR group advisors will receive, as the “students’ questions,” “advisors’ feedback,” “useful students’ feedback from surveys,” and “how would you address them in next workshops” prompts all trigger responsive feedback. The versions for CR and SO groups are as below (only the grey sections changes). Specifically, the CR and SO group will have the grey sections replaced with a simple checkbox, CR will have the workshop exit survey line completely removed, and the SO group will only need to check the box for “closing, summary, workshop exit surveys distributed and collected.”

[15 minutes] Individual updates, questions	[SO]
[5 minutes] Closing, summary, workshop exit surveys distributed and collected	

[15 minutes] Individual updates, questions	[CR]
[5 minutes] Closing, summary	

Conclusion: peer feedback and project reflection

16 weeks, 150+ hours, 8 drafts, I've finally come to this 52 pages document. I have received tons of feedback, enormous help, and also made countless revisions, and now is the time for conclusion and reflection for my individual project in EGIA Fall 2020!

Peer Feedback

Rubric	Suggestions [from Daniel & Mandy]
Learners:	Elaborate on addressing different learners' needs. How would you split students into teams, based on their own preferences, or having the instructor assign students with high CS skills and low skills into one group, or having students who have different interests/skills into one group?
Goals:	Link goals with particular tasks to themselves later in the document Add metacognitive level procedural goals on how students can <ul style="list-style-type: none">• self-assess their familiarity in using certain design and CS apps;• self-evaluate on their processes in making prototype (did they follow each step in MVP);• self-evaluate on the effectiveness of methods they choose for user test (self-reflect on what methods they choose and why -- e.g. was Google form the best choice, and why);• self-evaluate on presentation (whether they are confident in introducing the product etc).
Practicality:	Including a small section on how to work in a group prior to group assignments can improve cohesion (EGIA CollabU).
Innovation:	The blue->red->green color scheme (page 6) might benefit from some color changes (https://coolors.co/generate)
Others:	Give a definition of the near-peer mentoring first, spelling error, etc.

Project Reflection

Self-Assessment of the Project PRODUCT

- How well aligned are your goals, assessment and instruction?
 - My goal, assessment, and instruction are aligned around the same set of activities including proposal, storyboard, user survey, code review, presentation, etc.
- How did your age level focus impact the design, compared to similar units that have been or could be designed for younger and / or older age levels?
 - I assumed a lot of prior knowledge in high school students of age 16-18 compared to younger students, like collaboration, presentation, programming, design experiences, so many of the times the workshops are structured to help them activate these prior knowledge. But they still have a lot to learn and may benefit from direct instructions (compared to college students who may have already taken CS class with project components), so my design also has some of those aspects built-in.
- In what ways does your design explicitly and thoroughly exemplify course principles (i.e., utilize the big ideas)?
 - I addressed almost all of the big ideas in my project. The project's developmental process and organizational structure themselves exemplified alignment and iterative & backward design, and the use of both in-person and remote format touched on education technology.
 - I also mentioned ideas of understanding, misconceptions, transfer, authentic tasks, individual differences, prior knowledge, belonging, motivation, growth mindset, metacognition, self-directed learning cycle, assessment as instruction, assessment reliability & validity, self-evaluation, WHERE TO, active learning, just-in-time telling, deliberate practice, feedback loop, making, worked example, and evaluation research.
 - I referred to these big ideas using either direct citation from source reading or linked to our team's big idea synthesis resource document.
- How clearly have you described all five sections of your design so that it would be practically applicable and user-friendly for an educator?
 - I structured my project clearly into 5 sections and each section has a very descriptive naming, so just by reading the table of contents, an educator user would be able to know where to check if they have questions with specific



phases, or if they read everything from the beginning they'd be able to know how to implement things in practice, as examples and tables are provided all along.

- What resources and/or prior knowledge would be required of an educator implementing your design?
 - The educator would need to know how to develop an app themselves (and all related concepts of storyboarding, user survey, team collaboration, presentation) to the degree that they can explain important concepts, give instructive feedback, and help students build their projects. They also need to have access to electronic devices like laptops that's necessary for communication and code demo.
- What are the innovative aspects of your design?
 - Content-wise, I use near-peer mentoring and project-based learning extensively, and I put it into a practical, after school program context, where the primary instructional method is practice & feedback and directed exploration, and my audience are novice educators (CMU students as project advisors, and I wrote a lot of basics and theories with abundant references to fit their needs), all of which are all unique aspects.
 - Format-wise, I included a lot of in-text hyperlinks to help users navigate across the document and also more clearly see the alignment idea, which is also innovative.
- How did you incorporate peer feedback to enhance your project product?
 - I [summarize](#) the feedback from peers and implement them selectively based on the cost-and-effect payoff. E.g., I feel like the priority relations of my current color palette is clear, and making that change could potentially take longer than necessary, so I didn't change it. And implementing the pre-entry survey idea is not quite feasible given the limited time, so I also put it as my future plans. But I do incorporate all the other feedback to some extent.

Self-Assessment of the Project PROCESS

What were the strengths and weaknesses of your individual project design process?

- What challenges did you face as you worked through the project this semester?
 - Time constraint and misunderstanding. I was having a very obscure idea about what the document would look like before the sample projects were posted, so I wasted a lot of time writing things that are less useful or thinking about what to write, and working on big idea synthesis and my individual project at the same time is a lot of work.

- 
- How did you overcome them and/or why do some remain?
 - After the sample projects were posted I got much better ideas about what I should do. And the time problem is unsolvable. It's less about working smarter vs. harder, I just need days instead of hours to write all out. Or I may have worked in a wrong way that I didn't realize.
 - How did the experience of giving and receiving peer feedback impact your project process?
 - Seeing other people's work and receiving feedback from fresh eyes help me see my own strengths and weaknesses and help me make changes on my final draft.
 - What are your next steps, either with respect to this project if you plan to continue it, or with respect to other projects that could benefit from this approach?
 - For this project, I'll probably expand on the section of pre-entry survey and project groupings. I thought about implementing it initially, but it doesn't quite fit into my assessment and/or evaluation sections. Project Ignite as a student organization actually did implement pre-survey ideas in our application and team formation stages and I proposed some interesting things to facilitate this process, but I just didn't formally write it out in this document.
 - I can possibly also write the Methods section separately to talk about subjects, procedure, materials, timeline, data collection & scoring in a more formal way following the research paper format.
 - For other prospective projects, the step development process I used is indeed very helpful, and the backward design idea can be applied to more than educational design areas.
 - The next time you have an opportunity to begin a new project, how do you plan to proceed differently than you have on this project?
 - I'll directly use google doc to start my project and get settled with a template, so I don't need to waste time on the ineffective designs of less helpful softwares.
 - I'll also ask for samples earlier in the developmental process (probably not when I write the background and learners profile steps since I also want to start fresh without being swayed / intimidated by fully-developed sample work, but definitely before the goals so that I can see how to write and organize goals).
- 

Resources

Personal Background

I learned CS as a student, worked in a CS learning platform to create exercises for high school students before, and tutored on a project involving programming components in my student organization.

Educational Materials

- Khan Academy (good resource from basics to advance)
- CS Academy (CMU-student initiated, gamy, graphics-based intro for HS students)
- Code Academy (higher level, more professional, not free)
- CS for All (help schools offer quality CS instructions and address equity issues)

Professional Contact

- Mark Stehlik, CS professor who knows about education practices of high schoolers and supports programming for social good
- [Erin Cawley](#), the Program Manager of CMU's [CS Academy](#)
- Leigh Ann Sudol DeLyser, former EGIA student who got a self-defined CS Education PhD, run CSforAll now
- Girls Who Code, SCS4All, Women@SCS groups at CMU, Project Ignite

Meeting Notes with Contacts

- Friday 9/25 11:30 AM EST with Mark Stehlik
- Wednesday 11/25 2pm EST with Erin Cawley

Questions:

1. What's most needed by the grade 11-12 students?
Get a sense of the learner profile to see if you've got that in the right ballpark.
2. What're some priorities for the knowledge vs. procedure vs. disposition goals related to app development?
3. What might you find necessary as metacognition goals?
4. What works well for instruction, and how it can be assessed? (details of what work best for instruction and assessment)

5. What's a reasonable sequence and scope design? How can it be most useful and easily digestible for novice teachers?
6. What's a reasonable amount of content to be taught in 10hrs of instructional time in 10 weeks? What can be the expected product from a group of students?
7. Get students to collaborate in the distance learning situation?
8. What're some key remaining research questions?
9. What're some of the most important things in the evaluation research of instructions?
10. Any known & popular inventories to assess students' (1) metacognitive abilities, (2) engagement, and (3) satisfaction?


Notes:

- Biggest challenge: CS is not required everywhere, so commonly we have a mixed classroom, students may have exposure at home or just started.
 - Solution: differentiating curriculum to fit all students (hardest), or allowing for students don't walk away with same skill (hard to set goals), or similar to ETC project-based learning, have different teams match with diverse strength areas, set goals for the class & groups, assess on the group rather than individual level
- Metacognitive goals: building team dynamic and problem solving skills -> growth mindset
 - How to access: e.g., pre- and post- self reflection & survey on skill, prior knowledge, expectation of working on app, most likely work in collaboration with others or self, how many hours anticipate working with others
- Pre- and post surveys: prior knowledge coming in with, understanding of goals and hoping to get, self-efficacy, confidence in skill, so we can get really good visual of where they come from and how they end up
- Exit survey: want it short! 3 questions: grab key concept & confusing & yes/no, alternate in weeks to get all answers
- How useful are rubrics? Teaching people how to grade things, rules -> no more than 5 pts scale, start with goal, level 5 & all objectives met so full score, e.g., 3 categories: group dynamic & code & design, clearly, execute properly & comment & easy to use, 4 would be clear enough but still execute properly, etc. easy user experiences nice animation
- Rubrics can limit students' creativity, so don't provide rubrics as grading thing, instead as guide to help teachers know how to prompt students, what should a finished example include, guiding questions to prompt students to think & go through
- Grade 11-12 college level CS 1 course, prepare students for materials like that, problem-solving & app development, UX design skills, Accessible platforms for coding & app development & design, not learning to code for 10 weeks

- Evaluating focus: e.g. on understanding process of app development or functional programming skill, nice presentation vs. code executing properly. Needs a variety of survey questions: formative or summative
- Students can struggle with what they could make initially, so provide them with end goal that they are building, but don't let them come up with creative solutions right away, need 2-3 weeks, more follow an iterative design project idea
- Challenge of remote setting: access good internet & resources on devices, and class collaboration & warm up, relationship building
- Making sure there's time for students to get to know each other: virtual icebreakers, more related to design, follow in instagram?
- Breakout rooms 2 minutes & come back together, for class relationship building sake, instructors better not in rooms, but policy/legality may need instructors in rooms
 - Key: give a prompt & task to come back to full room with
- Making sure to use a running journal to keep track of group work

Appendix A: Selective Standards from CSTA K-12 Computer Science Standards (2017)


3A-AP-13	Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
3A-AP-15	Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
3A-AP-16	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.
3A-AP-18	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
3A-AP-19	Systematically design and develop programs for broad audiences by incorporating feedback from users.
3A-AP-21	Evaluate and refine computational artifacts to make them more usable and accessible.
3A-AP-22	Design and develop computational artifacts working in team roles using collaborative tools.
3A-AP-23	Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.
3B-DA-07	Evaluate the ability of models and simulations to test and support the refinement of hypotheses.
3B-AP-14	Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
3B-AP-15	Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.
3B-AP-16	Demonstrate code reuse by creating programming solutions using libraries and APIs.
3B-AP-17	Plan and develop programs for broad audiences using a software life cycle process.
3B-AP-20	Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project.
3B-AP-21	Develop and use a series of test cases to verify that a program performs according to its design specifications.



3B-AP-23	Evaluate key qualities of a program through a process such as a code review.
3B-AP-24	Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.
3B-IC-25	Evaluate computational artifacts to maximize their beneficial effects and minimize harmful effects on society.

References

- Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010). *How learning works: Seven research-based principles for smart teaching*. John Wiley & Sons.
- Armstrong, T. (2016). *The power of the adolescent brain: Strategies for teaching middle and high school students*. ASCD.
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?. *Acm Inroads*, 2(1), 48-54.
- Berger, K. S. (2003). *The developing person through childhood and adolescence* (6th ed.). Worth Publishers.
- Brigid J. S. Barron, Schwartz, D., Vye, N., Moore, A., Petrosino, A., Zech, L., The Cognition and Technology Group at Vanderbilt. (1998). Doing with Understanding: Lessons from Research on Problem- and Project-Based Learning. *The Journal of the Learning Sciences*, 7(3/4), 271-311.
- Clarke-Midura, J., Poole, F., Pantic, K., Hamilton, M., Sun, C., & Allan, V. (2018, February). How near-peer mentoring affects middle school mentees. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 664-669).
- Computer Science Teachers Association (2017). CSTA K-12 Computer Science Standards, Revised 2017. Retrieved from <http://www.csteachers.org/standards>.
- Dole, S. , Bloom, L. , & Doss, K. K. (2017). Engaged Learning: Impact of PBL and PjBL with Elementary and Middle-Grade Students. *Interdisciplinary Journal of Problem-Based Learning*, 11(2).
- Gal-Ezer, J., & Stephenson, C. (2014). A tale of two countries: Successes and challenges in K-12 computer science education in Israel and the United States. *ACM Transactions on Computing Education (TOCE)*, 14(2), 1-18.
- Holmes, V. L., & Hwang, Y. (2016). Exploring the effects of project-based learning in secondary mathematics education. *The Journal of Educational Research*, 109(5), 449-463.
- K-12 Computer Science Framework Steering Committee. (2016). *K-12 computer science framework*. ACM.
- Puzziferro, M. (2008). Online Technologies Self-Efficacy and Self-Regulated Learning as Predictors of Final Grade and Satisfaction in College-Level Online Courses, *The American Journal of Distance Education*, 22(2), 72-89, DOI: 10.1080/08923640802039024
- Master, A., Cheryan, S., Moscatelli, A., & Meltzoff, A. N. (2017). Programming experience promotes higher STEM motivation among first-grade girls. *Journal of experimental child psychology*, 160, 92-106.

- 
- Menzies, V., Hewitt, C., Kokotsaki, D., Collyer, C., & Wiggins, A. (2016). Project-Based Learning: evaluation report and executive summary.
- Merritt, J. , Lee, M. , Rillero, P. , & Kinach, B. M. (2017). Problem-Based Learning in K–8 Mathematics and Science Education: A Literature Review. *Interdisciplinary Journal of Problem-Based Learning*, 11(2)
- Michelene T. H. Chi & Ruth Wylie (2014) The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes, *Educational Psychologist*, 49:4, 219-243, DOI: 10.1080/00461520.2014.965823
- Nathan, M. J., & Wagner Alibali, M. (2010). Learning sciences. *Wiley Interdisciplinary Reviews: Cognitive Science*, 1(3), 329-345.
- Pellas, N. (2014). The influence of computer self-efficacy, metacognitive self-regulation and self-esteem on student engagement in online learning programs: Evidence from the virtual world of Second Life. *Computers in Human Behavior*, 35, 157-170.
- Schwartz, D. L., Tsang, J. M., & Blair, K. P. (2016). *The ABCs of how we learn: 26 scientifically proven approaches, how they work, and when to use them*. W W Norton & Co.
- The Igniters. (2020, December 5). [Big Ideas Synthesis Final Draft](#). In *Educational Goal, Instruction, and Assessment, Fall 2020*.
- Vakil, S. (2018). Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. *Harvard Educational Review*, 88(1), 26-52.
- Wiggins, G., Wiggins, G. P., & McTighe, J. (2005). *Understanding by design*. Ascd.
- Wikipedia contributors. (2020, November 7). [Minimum viable product](#). In *Wikipedia, The Free Encyclopedia*.
- Wikipedia contributors. (2020, September 20). [Iterative design](#). In *Wikipedia, The Free Encyclopedia*.
- Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education*, 26(4), 235-254.