Journal of
Information
Systems
Education

Volume 35
Issue 4
Fall 2024

Teaching Tip
A Screencast-Based Assessment Technique for a Mobile App Development Course

Leigh Jin, Tai-Yin Chi, and Brenda Mak

Recommended Citation: Jin, L., Chi, T.-Y., & Mak, B. (2024). Teaching Tip: A Screencast-Based Assessment Technique for a Mobile App Development Course. *Journal of Information Systems Education*, 35(4), 408-421. https://doi.org/10.62273/AVUW9529

Article Link: https://jise.org/Volume35/n4/JISE2024v35n4pp408-421.html

Received:	September 11, 2023
First Decision:	October 12, 2023
Accepted:	May 20, 2024
Published:	December 15, 2024

ISSN: 2574-3872 (Online) 1055-3096 (Print)

Journal of Information Systems Education, 35(4), 408-421, Fall 2024
https://doi.org/10.62273/AVUW9529

Teaching Tip
A Screencast-Based Assessment Technique for a Mobile App Development Course

Leigh Jin
Tai-Yin Chi
Brenda Mak
Lam Family College of Business
San Francisco State University
San Francisco, CA 94132, USA
jinlei@sfsu.edu, tchi@sfsu.edu, bmak@sfsu.edu

ABSTRACT

This paper presents the Screencast-Based Assessment Technique (SBAT) for a mobile app development curriculum in the information systems discipline. SBAT was implemented as a midterm take-home exam in which students design and develop an app project based on their own interests, passions, or ambitions. In addition to coding, students must create screencasts to document the coding process and explain key programming concepts. In Spring 2023, we collected interview feedback and course evaluations from students. Our findings indicate that SBAT supported critical thinking skills such as analyzing, evaluating, and creating in four key areas: idea generation, code development, concept explanation, and video production. In addition, students were motivated to connect their own passions with the learning outcomes of the course. Overall, we found that this assessment approach facilitated deeper internalization and integration of knowledge and helped students learn better than traditional paper-based exams.

Keywords: Mobile application development, Learning assessment, Bloom's taxonomy, Learner-centered education, Programming languages, IS curriculum

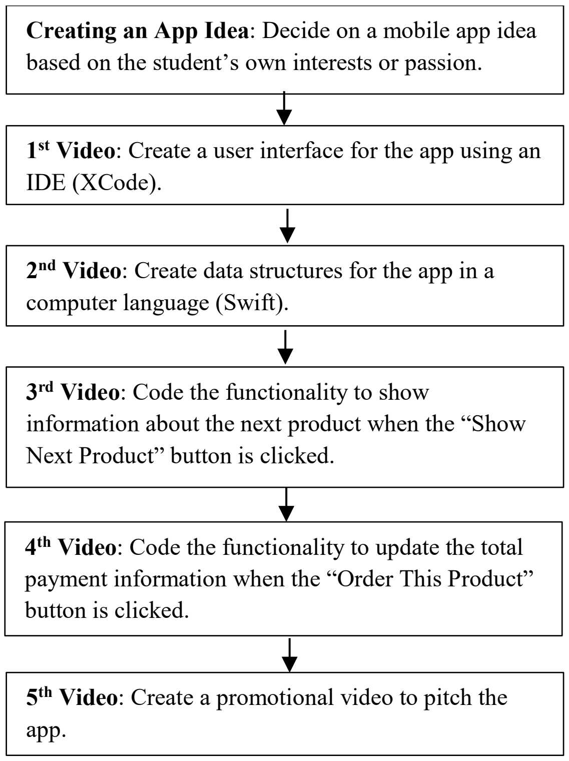
1. INTRODUCTION

App design and development are now among the most in-demand skills, presenting students with new career pathways in a global app economy (Natanson, 2022). Since 2008, Apple's App Store ecosystem has generated $155 billion USD in revenue for app developers. Fortune 500 companies are building custom apps for everything from supply chain management to customer support (Apple Newsroom, 2020, 2022). In 2021 alone, more than 2 million new apps were launched (Mandel & Shapiro, 2022). The app economy is a powerful force in stimulating business growth. As such, skills in app development not only equip students for the job market but also empower them to be entrepreneurs realizing their own business ideas.

Effective teaching involves aligning learning objectives, instructional activities, and assessments (Eberly Center, Carnegie Mellon University, n.d.). The learning goals of an app development curriculum require students to create app innovations and develop functional mobile apps. To achieve these goals, the instructional activities require techniques that are beyond memorizing and understanding (Bloom, 1956; Krathwohl & Anderson, 2010). Specifically, students need to apply high-level learning skills to analyze problems and create new mobile apps (Anderson & Krathwohl, 2001). The "assessment for learning" approach suggests that assessment should be carefully designed to promote students' learning rather than to rank students (Black et al., 2004; William, 2011). Bengtsson (2019) notes that take-home exams may be the preferred choice of assessment method for supporting students' learning at higher levels of Bloom's taxonomy skills.

This paper proposes a teaching assessment tool, the Screencast-Based Assessment Technique (SBAT), to assess an individual student's skills in a midterm take-home exam. First, the student designs and develops an individual app project based on their own interests, passions, or ambitions. Second, the student produces a series of screencasts to document the coding process and explain the important concepts and procedures, enabling the instructor to examine the student's level of understanding and ability to apply app development skills. Third, the student develops a video to pitch the app idea through storytelling and demonstrating the app, allowing them to showcase the app in a meaningful context. Figure 1 summarizes the SBAT requirements for students.

The remainder of this paper is organized as follows. Section 2 reviews the learning theories that guide the design of SBAT. Section 3 discusses SBAT implementation in an iOS app development curriculum. Section 4 presents evidence of effectiveness based on student interviews and survey responses. Section 5 concludes the paper with a discussion of the limitations of SBAT, potential improvements for the assessment activities, and suggestions for future research.



Creating an App Idea: Decide on a mobile app idea based on the student's own interests or passion.

1st Video: Create a user interface for the app using an IDE (XCode).

2nd Video: Create data structures for the app in a computer language (Swift).

3rd Video: Code the functionality to show information about the next product when the "Show Next Product" button is clicked.

4th Video: Code the functionality to update the total payment information when the "Order This Product" button is clicked.

5th Video: Create a promotional video to pitch the app.

Figure 1. SBAT Requirements for Students

2. BACKGROUND AND LITERATURE REVIEW

2.1 Bloom's Taxonomy of Learning

Bloom's taxonomy of learning provides the foundation for developing our Screencast-Based Assessment Technique (SBAT) presented in this paper. Bloom (1956) created a learning taxonomy to promote higher forms of learning in education. The taxonomy includes applying and evaluating concepts, processes, procedures, and principles rather than just remembering facts. Learners should have acquired new knowledge, skills, or attitudes after a learning event. In a revised version of the taxonomy, Anderson and Krathwohl (2001) defined six major categories of cognitive learning processes from low to high: remembering, understanding, applying, analyzing, evaluating, and creating (see Table 1).

Bloom's Taxonomy has been applied in various disciplines for different educational activities (Sobral, 2021). Masapanta-Carrión and Velázquez-Iturbide (2018) studied how it has been adopted in the computer science field, especially in specifying the learning objectives for a course and evaluating students' performance. Johnson and Fuller (2006) found that it is challenging to apply the two highest levels of Bloom's Taxonomy (evaluating and creating) in programming courses.

As suggested by Anderson and Krathwohl (2001), the *evaluating* and *creating* skills facilitate the discovery and construction of new knowledge; therefore, they are more effective but also more difficult to acquire than skills like *remembering* and *understanding*. In this study, we use Bloom's Taxonomy as a general guideline to design the assessment activities associated with the iOS app development curriculum in a mobile app development course. Specifically, we would like to embed the higher dimensions of the taxonomy into these assessment activities.

Category	Description
Remembering	Recalling or retrieving previously learned information
Understanding	Comprehending the meaning, translation, interpolation, and interpretation of instructions and problems; stating a problem in one's own words
Applying	Using a concept in a new situation or unprompted use of an abstraction; applying what was learned in the classroom to novel situations in the workplace
Analyzing	Separating material or a concept into component parts so that its organizational structure may be understood; distinguishing between facts and inferences
Evaluating	Making judgments about the value of ideas or materials
Creating	Building a structure or pattern from diverse elements; putting parts together to form a whole, with emphasis on creating a new meaning or structure

Table 1. Revised Bloom's Taxonomy (Anderson & Krathwohl, 2001)

2.2 Intrinsic Motivation Theory and Learning by Doing

In addition to Bloom's Taxonomy, the SBAT design also draws on intrinsic motivation theory. Ryan and Deci (2000) define intrinsic motivation as "the doing of an activity for its inherent satisfactions rather than for some separable consequence" (p. 56). According to Ryan and La Guardia (2000), taking interest in novelty and creatively applying techniques or methods are fundamental ways for a person to grow in knowledge and skills; therefore, this natural motivational tendency is critical for the cognitive, social, and physical development of a human being. In education, even when students are extrinsically coerced to engage in instructional activities, what they learn and how effectively they learn are influenced by their level of intrinsic motivation. It is important to design an intrinsically interesting learning environment that promotes challenge, inspires curiosity, and stimulates the learner's sense of choice and control (Malone & Lepper, 2021). Numerous education studies have shown that intrinsic motivation not only supports students' fundamental psychological needs but also facilitates enjoyable learning experiences. In addition, students are more likely to put in more effort to attain desired learning outcomes when they are intrinsically motivated (Fischer et al., 2019; Goldman et al., 2017; Ryan & Deci, 2020; Weimer, 2013). In this paper, we incorporated intrinsic motivation in the assessment activities by challenging students to choose app ideas inspired by their own interests, curiosity, and ambitions.

Learning by Doing (LBD) is another guiding principle that we considered in our SBAT design. Initially grounded in activity theory (Jonassen & Rohrer-Murphy, 1999; Vygotsky & Cole, 1978; Zurita & Nussbaum, 2007), the underlying principle of LBD is that deep learning happens when students are actively engaged in productive learning activities (Bedenlier et al., 2020; Iftikhar et al., 2022; Kirschner et al., 2006; Mekonnen, 2020; Szalai et al., 2021; Vescan, 2019). In other words, an efficient way to teach someone to master something is to encourage the person to learn by doing it. LBD helps learners internalize knowledge, adapt the skills in a new context, and create new experiences or products using the knowledge.

Project-Based Learning (PBL) is a form of Learning by Doing. According to David (2008), PBL is beneficial because real-world problems capture students' interest and intrinsically motivate them to apply new knowledge in a problem-solving context. Blumenfeld et al. (1991) argue that the key to PBL is a relevant question or problem that serves to structure and direct students' LBD actions. Instructors also play an important role in facilitating PBL by framing problems, structuring tasks, and assessing students' learning performance (David, 2008). Although PBL's core characteristics and advantages are generally known, there are many ways this instructional strategy is actually used in practice (Aditomo et al., 2013; Mentzer et al., 2020; Nainggolan et al., 2020; Rozal et al., 2021). The most successful projects are those in which students took the most initiative (Weimer, 2013), were the most resourceful (Bell, 2010), and worked the hardest (Freire, 2018). In addition, researchers have emphasized the importance of individual learning and teaching by templates when implementing PBL in information systems curricula (Al-Imamy et al., 2006; Newby & Nguyen, 2010; Yadin & Or-Bach, 2010).

2.3 Student-Centered Assessment for Learning

Assessment plays a crucial role in influencing students' behavior and their willingness, desire, and capacity to learn in the classroom (Brown et al., 1997; Rust, 2002; William, 2011). Traditional assessment methods often encourage students to memorize isolated information without deep learning, and the tendency to over-emphasize the classroom testing function of evaluation may produce undesirable consequences, including reduced intrinsic motivation, greater anxiety, and lowered self-efficacy for learning (Crooks, 1988; William, 2011).

The shift from a teacher-centered to a learner-centered paradigm profoundly affects the approach to assessment (Saulnier et al., 2008; Weimer, 2013). "Assessment for learning" is defined as "any assessment for which the first priority in its design and practice is to serve the purpose of promoting students' learning" (Black et al., 2004, p. 10). Its purpose differs from the traditional assessment method of ranking students.

William (2011) highlighted some important guidelines for implementing student-centered assessments. In general, assessment for learning tasks should help to connect students with learning outcomes, engage them in remedial actions for improvement, and motivate them intrinsically as self-regulated learners. According to Bengtsson (2019), take-home exams are the preferred choice of assessment method for supporting students' learning. Compared to traditional proctored, in-class, closed-book, pen-and-paper exams, take-home exams promote higher levels of Bloom's taxonomy skills, impose less stress, and allow students time for reflection. When they are thoughtfully designed, take-home tests provide a unique opportunity to turn an assessment into a learning activity.

The above research perspectives greatly influenced the design of our SBAT activity as a project-based take-home exam. In the course that is the setting for this study, each student is required to create a mobile commerce app that is unique and relevant to themselves by modifying the template of an exemplar project provided by the instructor. Screencast submissions are required to illustrate and explain in detail the app development process. This assessment for learning approach supports higher levels of Bloom's taxonomy skills by motivating students to connect their own interests with the learning outcome of this course. This approach also reinforces self-regulated learning by allowing students to discover and correct their own mistakes during the app development process.

In the next section, we briefly introduce iOS app development and the Swift programming language which comprise the main curriculum delivered in the class. The SBAT is conducted as a take-home midterm exam after the basic iOS app development skills are taught. Accordingly, we present the relevant user interface objects and programming concepts introduced in class prior to the implementation of SBAT.

3. SBAT IMPLEMENTATION IN AN IOS APP DEVELOPMENT CURRICULUM

SBAT was implemented in a mobile app development course in the Information Systems major at a university on the US West Coast. This elective course meets in person for a one-and-half-hour class twice a week for 16 weeks. In Spring 2023, 24 students were enrolled in a single section. Most of them had taken an introduction to programming course prior to joining this class. SBAT was implemented as a midterm take-home exam and contributed to 30% of their overall grade.[1] At the beginning of week 9 of the course, the SBAT exam was made available to students. It required each student to submit screencasts to illustrate how they developed the app of their choice. They were given one week to complete five screencast videos for the exam. Each video ranges from 5 to 15 minutes.

The curriculum teaches the Swift programming language, which is a general-purpose programming language for developing iOS, tvOS, watchOS, and macOS applications across Apple's platforms. Building on Swift, the UIKit framework supports crucial features for creating iOS apps, including UI architecture, event-handling, and interaction management. Apple also provides Xcode as an Integrated Development Environment (IDE) that enables students to create, edit, debug, and test their apps in a single IDE. It features a visual design editor (Interface Builder) that graphically connects the visual interface components onscreen with Swift code. The built-in auto code completion and error fix suggestion features make it an intelligent IDE. Because students can visually test their apps through a simulator or on a physical device such as an iPhone or iPad, the instant feedback and "wow factor" often encourage and engage them during the software development process. In our course, we adopted the *Develop in Swift Fundamentals* textbook published by Apple Education (2022), which is free from Apple Books.

3.1 Preparation for SBAT Implementation: Course Elements and Learning Outcomes

The iOS User Interface (UI) elements, coding practices, and project templates were introduced to students between week 1 and week 8. First, the instructor demonstrated how to create simple apps with basic UI elements in Interface Builder without any coding. Table 2 summarizes the UI elements introduced prior to the assessment.

Second, important coding concepts like *IBOutlet* and *IBAction* were introduced. Table 3 summarizes these two important coding practices that are unique to the UIKit framework in iOS app development. Basic Swift programming concepts (variables, functions, arrays, and loops) as well as object-oriented topics (structures and classes) were also reviewed.

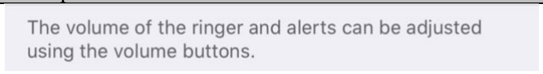
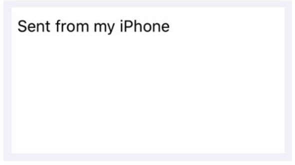
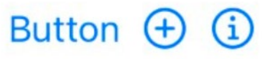
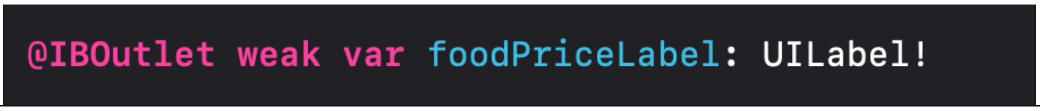
UI Element	Description	Example
UILabel	Displays static text to relay information to the user.	 The volume of the ringer and alerts can be adjusted using the volume buttons. Apple Education, 2022, p. 249
UIIMageView	Displays an image or an animated sequence of images. It can be configured to scale the image to fit or fill the space while maintaining the image's original aspect ratio.	Apple Education, 2022, p. 250
UITextView	Used to accept and display multiple lines of text, such as the body of an email message. The text font, text color, and alignment can be configured.	 Sent from my iPhone Apple Education, 2022, p. 251
UIButton	Performs certain actions when it is tapped or selected. The content of a button typically consists of a text label, an image, or both.	 Button Apple Education, 2022, p. 260

Table 2. UI Elements Introduced in the Course

Coding Practice	Description and Example
IBOutlet	Signals the establishment of a connection that links a user interface component in Interface Builder (e.g., UILabel) to a variable in the source code of a Swift file. In the figure, the foodPriceLabel is an *IBOutlet* variable that is connected with a text label in the Interface Builder.  `@IBOutlet weak var foodPriceLabel: UILabel!`

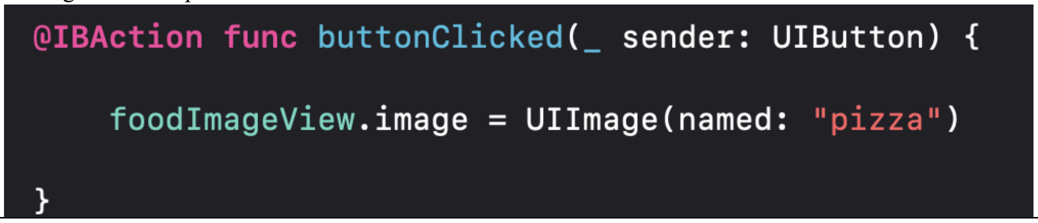
IBAction	Signals the creation of a relationship between an interactive visual component in Interface Builder (e.g., UIButton) and a method in the source code of a Swift file. In the figure, the buttonClicked() is an *IBAction* method that is connected with a button in the Interface Builder. This method will display the "pizza" image in a UIImageView component whenever this button is clicked.  `@IBAction func buttonClicked(_ sender: UIButton) {`
	`    foodImageView.image = UIImage(named: "pizza")`
	`}`

Table 3. Coding Practices Introduced Prior to SBAT Implementation

Third, a template app named "FoodOrder" was introduced to students over multiple class sessions. This is a simple one-screen app that allows users to view and order different dishes offered by a restaurant. The "Show Next Dish" button displays different dishes with updated image and price information. The "Order This Dish" button enables users to order the dish currently displayed on the screen. The itemized products and total payment information are updated when the button is clicked. The user interface and functionality of this app are illustrated in Figure 2.

The assignment assessed by SBAT required students to develop apps of their own choice that are similar to the template app, which essentially simulates the consumer experience of ordering products. The apps submitted by students could deliver similar functionality, but they should deliver different types of products or services that are not related to restaurant dishes. The learning outcomes introduced prior to SBAT implementation are listed in Table 4; these are also the learning outcomes we intended to assess with SBAT.

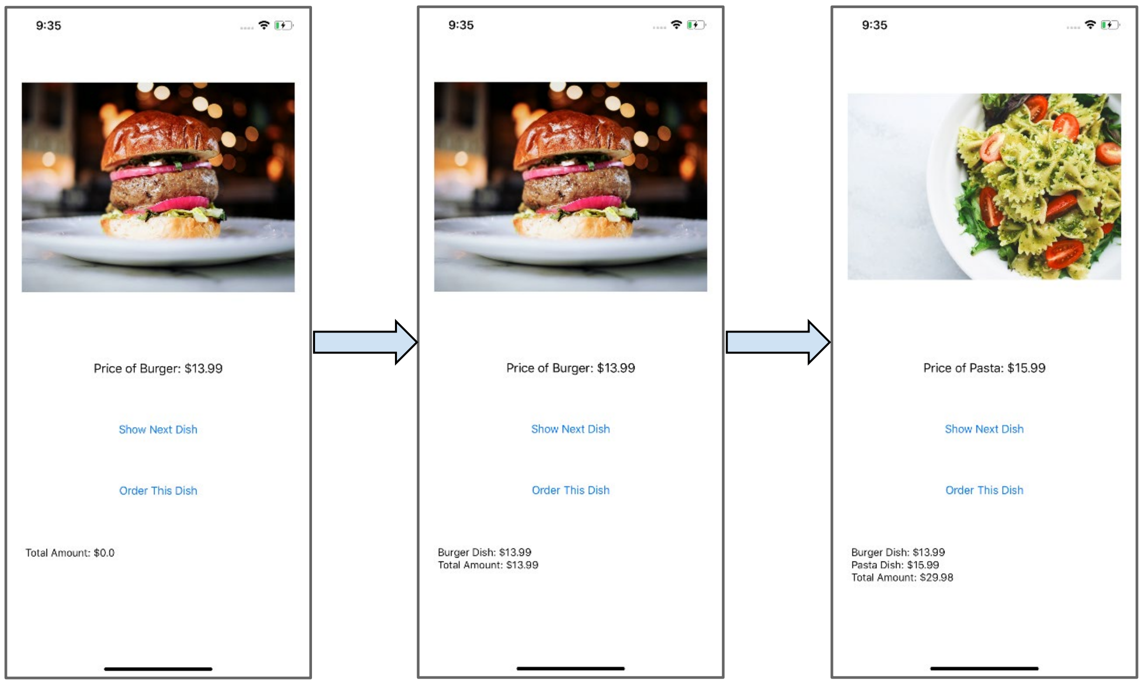


Figure 2. FoodOrder Template App: User Interface and Functionality

Skill Category	Learning Outcomes
General Swift Programming [S]	• S.1 - Creating variables, arrays, and structs • S.2 - Using control statements and loops • S.3 - Implementing functions to process data
Specific to the UIKit Framework [U]	• U.1 - Defining @IBOutlet variables to connect UI components with Swift variables • U.2 - Implementing @IBAction functions to control UI components
Specific to the Xcode IDE [X]	• X.1 - Using Xcode to create, edit, debug, and test apps • X.2 - Creating labels, images, text fields, and buttons through Interface Builder

Table 4. Learning Outcomes Introduced Prior to SBAT Implementation

3.2 Screencast-Based Assessment Technique (SBAT) Implementation

3.2.1 Student Tasks. As summarized in Figure 1, after they decide on the app they want to develop, students need to create five screencast videos[2] and narrate them in their own voices to explain the relevant programming logic and concepts. The length of each video should not exceed 15 minutes. This design helps to separate a long development process into smaller, more manageable tasks so that students will have a clear focus when explaining the programming logics in each screencast. It is also easy to identify mistakes and make corrections in the videos because they are relatively short. Students are given five days to complete all screencast videos during the midterm exam week. The instructions for creating the five videos are summarized below.

1. *Video 1: Creating the App User Interface*: Create a screencast to demonstrate and explain the process of creating the app user interface, including visual components such as labels, images, buttons, and texts.

The video should capture the visual component creation and layout process step-by-step in Xcode.

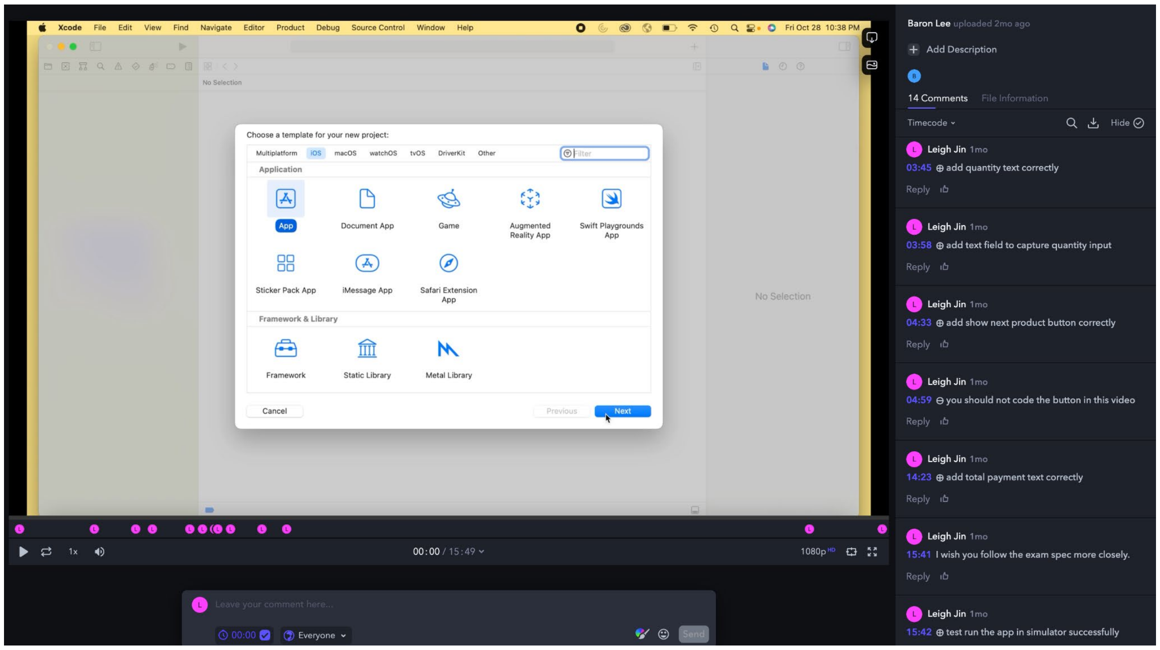
2. *Video 2: Setting up the Data Structure*: Create a screencast to demonstrate and explain the process of setting up the data structure, including constants, variables, structs, and arrays. The video should capture the data structure setup process line-by-line in Xcode.
3. *Video 3: Coding the Behavior of the "Show Next Product" Button*: Create a screencast to demonstrate and explain the process of coding the behavior of the "Show Next Product" button. The video should capture the implementation of the function that controls the button to display the next product information line-by-line in Xcode.
4. *Video 4: Coding the Behavior of the "Order This Product" Button*: Create a screencast to demonstrate and explain the process of coding the behavior of the "Order This Product" button. The video should capture the implementation of the function that controls the button to update the total payment information line-by-line in Xcode.
5. *Video 5: Creating a Promotional Video*: Create a video to promote and market the app. This may include a personal story or the motivation that inspired the app. The video should also demonstrate how the app can fulfill users' mobile commerce needs in specific contexts. Alternatively, family members or friends could be enlisted to provide testimonials.

More detailed SBAT specifications are presented in Appendix A. Note that the learning outcomes should be clearly identified in the detailed instructions. For example, in Screencast 01, an image view to show the product should be created, and in Screencast 03, the @IBAction function should be coded to display the next product information. Creating the image view and implementing the @IBAction functions are both required learning outcomes, listed as X.2 and U.2 in Table 4.

3.2.2 Instructor Grading in SBAT. Frame.io, a cloud-based collaboration platform operated by Adobe, was used to assist in the submission and grading of the screencast videos (https://frame.io/customers). In frame.io, a video can be stopped at any frame so that reviewers can provide frame-specific feedback. This frame-based comment feature is particularly helpful for evaluating screencast videos. After students upload their videos to frame.io, an instructor can view them and stop anywhere to comment on coding mistakes or tasks the students failed to complete. Positive comments can also be made if students perform exceptionally well in explaining coding concepts or deliver additional app features beyond what is required. Figure 3 illustrates an example of providing feedback in frame.io.

After text comments are made in frame.io, a simple rubric can be applied to provide a numeric evaluation of each screencast video. As presented in Table 5, our SBAT grading rubric includes three criteria: Video Requirements (30%), Coding Correctness (40%), and Communication Effectiveness (30%). Instructors who teach different IS courses could adjust these criteria to fit their own curriculum needs accordingly.

The combination of text comments and numeric grading provides a more comprehensive and constructive evaluation of students' work. The text comments help to justify the numeric grade. With 10-20 comments provided per screencast, students have a better understanding of what mistakes they made at exactly which points in the coding process, making it easier for them to take actions to correct these errors. Finally, positive comments about what they did well in the process encourage students to continue to learn and improve their coding skills.



Baron Lee uploaded 2mo ago
Add Description
14 Comments File Information
Timecode Hide

Leigh Jin 1mo
03:45 ⊕ add quantity text correctly
Reply

Leigh Jin 1mo
03:58 ⊕ add text field to capture quantity input
Reply

Leigh Jin 1mo
04:33 ⊕ add show next product button correctly
Reply

Leigh Jin 1mo
04:59 ⊖ you should not code the button in this video
Reply

Leigh Jin 1mo
14:23 ⊕ add total payment text correctly
Reply

Leigh Jin 1mo
15:41 I wish you follow the exam spec more closely.
Reply

Leigh Jin 1mo
15:42 ⊕ test run the app in simulator successfully

Figure 3. Example of Providing Feedback in frame.io

Criteria	Excellent 30 points	Fair 20 points	Poor 10 points
Video Requirements (30%)	Video submission demonstrates coding line-by-line, with interpretations in the student's own voice, within the time limit as specified.	Some requirements are unfulfilled.	The majority of requirements are unfulfilled.
Coding Correctness (40%)	All functionality and features are implemented correctly without any syntax or runtime errors.	Functionality and features are implemented with minor syntax or runtime errors.	Functionality and features are implemented with major syntax or runtime errors.
Communication Effectiveness (30%)	All required programming concepts are explained clearly, with sufficient coding comments included.	Some programming concepts are not explained, and some coding comments are missing.	Most programming concepts are not explained, and the majority of coding comments are missing.

Table 5. SBAT Screencast Grading Rubric Example

4. EVIDENCE OF EFFECTIVENESS

SBAT was implemented in an information systems mobile app development course in Spring 2023. Students developed functional iOS apps that offered products and services in arts, collectables, electronics, games, jewelry, music, sports, and social media. After the SBAT midterm exam, students were given the opportunity to provide feedback through face-to-face or online Zoom interviews with the instructor. As incentive, the participating students were offered 30 points (3%) extra credit for their effort. In total, 13 students signed up. These semi-structured interviews lasted 15-30 minutes. We include the interview questions in Appendix B. The interview transcripts were analyzed and coded[3] to capture students' experiences with the higher learning levels of Bloom's Taxonomy, including analyzing, evaluating, and creating. Below, we briefly report our findings related to four activities essential to the SBAT: idea generation, code development, concept explanation, and video production.

4.1 Idea Generation

At the beginning of the exam, students must analyze different options and evaluate the pros and cons of each app idea before making the decision on which app to build:

- [C. L.]: I could sell shoes or Apple products, but I thought that was just too easy. I can literally just copy and paste your code, so there is no fun to that … I remember reading somewhere the older you get, the less time you spend with other people. I also had experience with that because my grandparents passed away, and I realized that I never really got to spend much time with them. So I decided to create an app that allows people to connect and socialize with their grandparents.

Many students created PowerPoint slides to help promote their app ideas in their pitch videos. In these slides, students often built a business case around their apps. This by-product of the process made the app idea more complete:

- [R. Y.]: How many PowerPoints have we made throughout our time in school, like hundreds of them? But if you apply it to something creative, explaining the reason why you make the app, it just adds more value to our main deliverable, which is the [computer] code … Having the creativity part makes me want to do it [my app].

4.2 Code Development

To be able to create their own code to implement their app ideas, students needed to first analyze and evaluate what was missing from their understanding. In addition to reviewing class materials, students reported that learning by coding and repetition helped them to fill their knowledge gaps:

- [C. L.]: I was really lost in understanding functions and structures [earlier]. So the midterm was really helpful because I had to build it myself, it made me sit down and see how this was translating to that. It really solidified my understanding.
- [S. V.]: I coded everything beforehand, and then, when I was doing the recording, I made another project and coded it again just so that it [the video] would flow easily … So recording the code is really nice, it allows me to demonstrate what I've learned.

Correcting coding mistakes is a very important part of learning programming. During the process, students must analyze the mistake, evaluate the potential solutions, and recreate the code to remove the errors. Both students and teachers can benefit from recorded videos to better understand the correction process:

- [P. D.]: Seeing what I recorded myself doing, I was like looking at myself from a mirror view and seeing what I was doing wrong with my code, and I would go back and change the code and modify my video.
- [S. S.]: You can see the way they are thinking in the video, right? So when they make that tiny mistake, they notice it after, they can actually go back and correct it. That is a huge part of learning, right? You can see them make that adjustment live, it is great for the grader as well.

4.3 Concept Explanation

One of the important aspects of SBAT implementation requires students to explain key programming concepts with their own narratives in their screencast videos. We observed that our students analyzed and evaluated their communication style carefully. Some of them even wanted to narrate as if they were teaching Swift programming to their peers:

- [J. M.]: Programming is a super complicated topic. I wanted to explain it in our words and in our verbiage rather than from a professor to a student. So I try to put myself in my classmates' shoes, like, how would I want to hear this in a simpler way?
- [A. Z.]: The explaining part taught me the most; I just think that I am teaching someone. I have to understand it completely and make it really straightforward in order to be able to teach it.

To be better prepared, some students created detailed scripts before recording their videos. Even though this is not part of the SBAT requirements, they went out of their way to make their screencast more effective:

- [A. D.]: Doing everything off the top of your head is insanely difficult on the fly for a 15-minute video. I wrote a script for myself on the function of each line that I coded, just to cut it down and condense everything [when I explain].
- [Z. W.]: The fluidity of my speaking is mainly from preparation. I was reading off a script. After I coded, I really thought through why I did it in the code. Once I understood how I wanted to speak about the code, I made my script … Every video took around 5 tries each to get to the point where I liked it.

4.4 Video Production

Creating screencast videos to do live coding while explaining key programming logic is challenging. To improve screencast quality and watching experience, some students created additional features in their video production:

- [P. D.]: I was just playing around with QuickTime. I learned that I can screen-record [my code] and video-record [my headshot] simultaneously, so I just did that. I thought everybody would do it that way. I also taught two other students how to do it.
- [Z. W.]: I was watching the video back; I was like, if I were in my audience, I really wouldn't want to listen to me ramble with no background music, so I added some background music using Capcut, that is a pretty big app used by every TikTok influencer right now.

Despite overwhelmingly positive feedback, a couple of interviewees indicated that SBAT took more time to complete than they expected. Since it was the first time they were creating and editing screencasts, they had to learn video production skills, which could be challenging. They also encountered some difficulties in uploading their videos to frame.io due to limited internet access from home.

Following the grading rubric specified in Table 4, all 5 videos were graded for each student. After the points were summed and weighted, numeric exam grades on the midterm were assigned to 24 students. About 79% of students received a grade in the 81-100 range. This is consistent with the suggestion that take-home exams tend to increase students' chances for high grades since they have more time to reflect and to correct their errors (Bengtsson, 2019). Because the SBAT specifications clearly identify the learning outcomes and students are able to demonstrate in the screencasts how to achieve these learning outcomes step-by-step in their own voices, we believe that these numeric grades provide a valid assessment of their app development skills based on the specified learning outcomes.

To further assess the learning effectiveness of SBAT, we conducted a short, anonymous online survey in May 2023. The survey was sent to all 24 enrolled students, and 22 responses were received and analyzed. The results indicated that 77.3% of the respondents agreed that SBAT helped them to "internalize the app development concepts and techniques," and 81.8% agreed that SBAT prepared them to "continue to expand my app development skills." Overall, 81.8% of the respondents agreed that SBAT is "more effective than traditional paper-based coding exams." In summary, the anecdotal feedback from both interviews and the online survey suggests that, in the context of a mobile app development curriculum and specified learning outcomes, the SBAT is a better assessment tool for supporting learning than traditional paper-based exams.

5. DISCUSSION AND CONCLUSION

In this paper, we introduced a Screencast-Based Assessment Technique (SBAT) in a mobile app development course. The development of this SBAT as a take-home exam was guided by relevant learning theories, including Bloom's taxonomy, intrinsic motivation, learning by doing, and assessment for learning. The learners were required to apply their skills to overcome coding challenges in order to create a mobile app based on their personal interests. They then created five screencast videos to document and explain their entire app development process.

In Spring 2023, we collected interview feedback and course evaluations from students who took the course. Our findings indicate that SBAT provides students with many opportunities to exercise the higher dimensions of Bloom's Taxonomy, including analyzing, evaluating, and creating. More specifically, students had to analyze their options and evaluate the solutions in four aspects of SBAT: idea generation, code development, concept explanation, and video production. In addition to creating programming code and screencast videos, some students also created PowerPoint slides to pitch their app idea and narration scripts to explain the key programming logics. Because they were intrinsically motivated by the relevance and purpose of the project, students were willing to make their best effort to fulfill the assignment requirements. Some of them coded their apps and recorded screencast videos multiple times, while others added extra features that were above and beyond the instructions they were given. Overall, students perceived that the SBAT was more effective than a traditional paper-based programming assessment administered under a strict time limit.

From the instructor's perspective, an additional benefit of SBAT is that it has the potential to reduce cheating behaviors. Students had to code their own apps line-by-line, explain the concepts with narratives in their own voices, and record the entire coding process in screencasts. These requirements make cheating difficult.

SBAT does have limitations. First, video production requires different skills than coding, and students who have never done it before may find it intimidating and time-consuming. To alleviate this concern in implementing SBAT, the instructor can provide tutorials and other resources to introduce basic video editing skills. Second, video storage is resource intensive. In general, video files are much larger in size than text files. Even though frame.io is an excellent solution for screencast submission, it could be expensive if the institution where the class is offered does not subscribe to it. Finally, producing and grading these screencast videos takes a lot of time and effort from both students and the instructor. On average, it takes about 20 minutes to grade and comment on each screencast (100 minutes per student), so SBAT may not be suitable for large classes. However, there are potential remedies to reduce the exam time for students and grading stress for the instructor. For example, the number of learning outcomes assessed in SBAT could be reduced. In this course, if students were not required to code the "Order This Product" button or "Receipt" text view, the length of the screencasts could be shortened. SBAT could also be implemented as a team project rather than an individual project. Furthermore, student peer evaluation could be employed to reduce the grading burden on the instructor.

Beyond iOS app development, SBAT can be applied to other technical courses in the Information Systems, Computer Science, or Software Engineering disciplines, including Android app development, Python programming, web development, and database design and development. Instructors can customize the template to match the course contents and adapt the scope requirements for creating different types of screencasts or videos. In addition, the time frame for students to complete the assessment can be adjusted based on the complexity of the project-based assessment. Future researchers may wish to investigate how the time frame and video content can be adjusted for different courses and how screencast-based learning can be applied to smaller assignments, group presentations, or other forms of curriculum. More evidence on the effectiveness of its implementation should be collected and analyzed as well.

Overall, we found that the SBAT design is effective in helping students to practice the higher dimensions of learning skills such as analyzing, evaluating, and creating. Its intrinsic motivation aspect inspires students to make extra effort in completing their tasks, while its learning by doing perspective facilitates deeper internalization and integration of knowledge. Students and instructors agreed that it is an effective assessment technique to enhance learning for students studying technical subjects in information systems courses.

6. ENDNOTES

1. The course grade distribution includes attendance (10%), individual homework exercises (30%), the SBAT individual take-home midterm (30%), and a final group project (30%). For practical implementation purposes, SBAT is only used in the individual midterm.
2. In an earlier version of the SBAT design, only one coding screencast was required. However, it was difficult to assess learning outcomes because students could implement them in different orders at different points in the video. We found that the five-video design is a better format because the learning outcomes and scope boundary requirements are clearly specified for each video and it is easy to grade them.
3. The keywords/phrases were identified from interview transcripts, then coded and categorized, mapping the learning theories reviewed in Section 2.

7. REFERENCES

Aditomo, A., Goodyear, P., Bliuc, A., & Ellis, R. (2013). Inquiry-Based Learning in Higher Education: Principal Forms, Educational Objectives, and Disciplinary Variations. *Studies in Higher Education*, 38(9), 1239-1258. https://doi.org/10.1080/03075079.2011.616584

Al-Imamy, S., Alizadeh, J., & Nour, M.A. (2006). On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process. *Journal of Information Technology Education*, 5, 271-283. https://doi.org/10.28945/247

Anderson, L. W., & Krathwohl D. R. (Eds.). (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. New York: Addison Wesley Longman.

Apple Education. (2022). *Develop in Swift Fundamental (Xcode 13).* Apple Book.

Apple Newsroom. (2020, September 2). iOS App Economy Creates 300,000 New US Jobs as Developers Adapt during Pandemic. https://www.apple.com/newsroom/2020/09/ios-app-economy-creates-300000-new-us-jobs-as-developers-adapt-during-pandemic/

Apple Newsroom. (2022, May 25). New Research Highlights Job Growth, Global Success of Small Businesses and Entrepreneurs on the App Store. https://www.apple.com/newsroom/2022/05/new-research-highlights-job-growth-of-small-businesses-on-the-app-store/

Bedenlier, S., Bond, M., Buntins, K., Zawacki-Richter, O., & Kerres, M. (2020). Learning by Doing? Reflections on Conducting a Systematic Review in the Field of Educational Technology. In O. Zawacki-Richter, M. Kerres, S. Bendenlier, M. Bond, & K. Buntins (Eds.), *Systematic Reviews in Educational Research* (pp. 111-127). Springer. https://doi.org/10.1007/978-3-658-27602-7_7

Bell, S. (2010). Project-Based Learning for the 21st Century: Skills for the Future. *The Clearing House*, 83(2), 39-43. https://doi.org/10.1080/00098650903505415

Bengtsson, L. (2019). Take-Home Exams in Higher Education: A Systematic Review. *Education Sciences*, 9(4), 267-283. https://doi.org/10.3390/educsci9040267

Black, P., Harrison, C., Lee, C., Marshall, B., & William, D. (2004). Working Inside the Black Box: Assessment for Learning in the Classroom. *Phi Delta Kappan*, 86(1), 8-21. https://doi.org/10.1177/003172170408600105

Bloom, B. S. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. New York: David McKay.

Blumenfeld, P., Soloway, E., Marx, R., Krajcik, J., Guzdial, M., & Palincsar, A. (1991). Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning. *Educational Psychologist*, 26(3-4), 369-398. https://doi.org/10.1080/00461520.1991.9653139

Brown, G., Bull, J., & Pendlebury, M. (1997). *Assessing Student Learning in Higher Education.* Routledge. https://doi.org/10.4324/9781315004914

Crooks, T. J. (1988). The Impact of Classroom Evaluation Practices on Students. *Review of Educational Research*, 58(4), 438-481. https://doi.org/10.3102/00346543058004438

David, J. L. (2008, February 1). What Research Says About … Project-Based Learning. *Educational Leadership*, 65(5), 80-82. https://www.ascd.org/el/articles/project-based-learning

Eberly Center, Carnegie Mellon University. (n.d.) *Teaching Principles.* https://www.cmu.edu/teaching/principles/teaching.html

Fischer, C., Malycha, C. P., & Schafmann, E. (2019). The Influence of Intrinsic Motivation and Synergistic Extrinsic Motivators on Creativity and Innovation. *Frontiers in Psychology*, 10, Article 137. https://doi.org/10.3389/fpsyg.2019.00137

Freire, P. (2018). *Pedagogy of the Oppressed*. Bloomsbury Publishing USA. https://doi.org/10.4324/9780429269400-8

Goldman, Z. W., Goodboy, A. K., & Weber, K. (2017). College Students' Psychological Needs and Intrinsic Motivation to Learn: An Examination of Self-Determination Theory. *Communication Quarterly*, 65(2), 167-191. https://doi.org/10.1080/01463373.2016.1215338

Iftikhar, S., Guerrero-Roldán, A. E., & Mor, E. (2022). Practice Promotes Learning: Analyzing Students' Acceptance of a Learning-by-Doing Online Programming Learning Tool. *Applied Sciences*, 12(24), Article 12613. https://doi.org/10.3390/app122412613

Johnson, C. G., & Fuller, U. (2006, February). Is Bloom's Taxonomy Appropriate for Computer Science? *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006* (120-123). https://doi.org/10.1145/1315803.1315825

Jonassen, D. H., & Rohrer-Murphy, L. (1999). Activity Theory as a Framework for Designing Constructivist Learning Environments. *Educational Technology Research and Development*, 47(1), 61-79. https://doi.org/10.1007/BF02299477

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-based, Experiential, and Inquiry-based Teaching. *Educational Psychologist*, 41, 75-86. https://doi.org/10.1207/s15326985ep4102_1

Krathwohl, D. R., & Anderson, L. W. (2010). Merlin C. Wittrock and the Revision of Bloom's Taxonomy. *Educational Psychologist*, 45(1), 64-65. https://doi.org/10.1080/00461520903433562

Malone, T. W., & Lepper, M. R. (2021). Making Learning Fun: A Taxonomy of Intrinsic Motivations for Learning. In R. E. Snow & M. J. Farr (Eds.), *Aptitude, Learning, and Instruction* (Vol. 3, pp. 223-254). Routledge.

Mandel, M., & Shapiro, J. (2022, May 25). *U.S. App Economy Update.* https://www.progressivepolicy.org/publication/u-s-app-economy-update-2022/

Masapanta-Carrión, S., & Velázquez-Iturbide, J. Á. (2018, February). A Systematic Review of the Use of Bloom's Taxonomy in Computer Science Education. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 441-446). https://doi.org/10.1145/3159450.3159491

Mekonnen, F. D. (2020). Evaluating the Effectiveness of 'Learning by Doing' Teaching Strategy in a Research Methodology Course, Hargeisa, Somaliland. *African Educational Research Journal*, 8(1), 13-19.

Mentzer, K., Frydenberg, M., & Yates, D. J. (2020). Teaching Applications and Implications of Blockchain via Project-Based Learning: A Case Study. *Information Systems Education Journal*, 18(6), 57-85.

Nainggolan, B., Hutabarat, W., Situmorang, M., & Sitorus, M. (2020). Developing Innovative Chemistry Laboratory Workbook Integrated With Project-Based Learning and Character-Based Chemistry. *International Journal of Instruction*, 13(3), 895-908. https://doi.org/10.29333/iji.2020.13359a

Natanson, E. (2022, September 7). The New Frontier: Emerging Markets and the Future Global App Economy. *Forbes.* https://www.forbes.com/sites/eladnatanson/2022/09/07/the-new-frontier-emerging-markets-and-the-future-global-app-economy/?sh=6a70c3d1000e

Newby, M., & Nguyen, T. (2010). Using the Same Problem With Different Techniques in Programming Assignments: An Empirical Study of its Effectiveness. *Journal of Information Systems Education*, 21(4), 375-382.

Rozal, E., Ananda, R., Zb, A., Fauziddin, M., & Sulman, F. (2021). The Effect of Project-Based Learning Through YouTube Presentations on English Learning Outcomes in Physics. *Al-Ishlah: Jurnal Pendidikan*, 13(3), 1924-1933. https://doi.org/10.35445/alishlah.v13i3.1241

Rust, C. (2002). The Impact of Assessment on Student Learning: How Can the Research Literature Practically Help to Inform the Development of Departmental Assessment Strategies and Learner-Centred Assessment Practices? *Active Learning in Higher Education*, 3(2), 145-158. https://doi.org/10.1177/1469787402003002004

Ryan, R. M., & Deci, E. L. (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology*, 25(1), 54-67. https://doi.org/10.1006/ceps.1999.1020

Ryan, R. M., & Deci, E. L. (2020). Intrinsic and Extrinsic Motivation From a Self-Determination Theory Perspective: Definitions, Theory, Practices, and Future Directions. *Contemporary Educational Psychology*, 61, Article 101860. https://doi.org/10.1016/j.cedpsych.2020.101860

Ryan, R. M., & La Guardia, J. G. (2000). What Is Being Optimized Over Development?: A Self-Determination Theory Perspective on Basic Human Needs Across the Life Span. In S. H. Qualls & N. Abeles (Eds.), *Psychology and the Aging Revolution* (145-172). American Psychological Association. https://doi.org/10.1037/10363-008

Saulnier, B. M., Landry, J. P., Longenecker, H. E., & Wagner, T. A. (2008). From Teaching to Learning: Learner-Centered Teaching and Assessment in Information Systems Education. *Journal of Information Systems Education,* 19(2), 169-174.

Sobral, S. R. (2021). Bloom's Taxonomy to Improve Teaching-Learning in Introduction to Programming. *International Journal of Information and Education Technology*, 11(3), 148-153. https://doi.org/10.18178/ijiet.2021.11.3.1504

Szalai, C., Herbstreit, S., Novosadova, K., & Somerville, S. (2021). Learning by Doing: To Explore the Influence of Simulation on Clinical Decision-Making Approaches on Final Year Medical Students at the University of Duisburg-Essen, Germany. *MedEdPublish*, 10, Article 124. https://doi.org/10.15694/mep.2021.000124.1

Vescan, A. (2019). Does Learning by Doing Have a Positive Impact on Teaching Model Checking? *Proceedings of the 1st ACM SIGSOFT International Workshop on Education through Advanced Software Engineering and Artificial Intelligence*, 27-34. https://doi.org/10.1145/3340435.3342717

Vygotsky, L. S., & Cole, M. (1978). *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press.

Weimer, M. (2013). *Learner-Centered Teaching: Five Key Changes to Practice*. John Wiley & Sons.

William, D. (2011). What Is Assessment for Learning? *Studies in Educational Evaluation*, 37(1), 3-14. https://doi.org/10.1016/j.stueduc.2011.03.001

Yadin, A., & Or-Bach, R. (2010). The Importance of Emphasizing Individual Learning in the "Collaborative Learning Era." *Journal of Information Systems Education*, 21(2), 185-194.

Zurita, G., & Nussbaum, M. (2007). A Conceptual Framework Based on Activity Theory for Mobile CSCL. *British Journal of Educational Technology*, 38(2), 211-235. https://doi.org/10.1111/j.1467-8535.2006.00580.x

AUTHOR BIOGRAPHIES

Leigh Jin is a professor of information systems at San Francisco State University. She earned her doctoral in Computer Information Systems from Georgia State University. Her research interests include Augmented Reality Applications, Technology Innovation with Design Thinking, Enterprise Mobility, Mobile Reputation Systems, Open Source Software Adoption, and Virtual Organization/Community. She has published research papers in Information Systems related journals and conference proceedings, including *Journal of Information Systems Education*, *Information Resources Management Journal*, *Information and Organization*, *International Journal of Innovation and Technology Management*, *International Journal of Electronic Commerce*, *Journal of Information Technology Teaching Cases*, and the *Data Base for Advances in Information Systems*.

Tai-Yin Chi is an assistant professor in the Information Systems Department at San Francisco State University (SFSU). He received his Ph.D. in Information Systems and Technology from Claremont Graduate University, California, USA. Before joining the faculty at SFSU, he worked as a Lecturer for California State Polytechnic University, Pomona (2013-2016) and California State University, San Bernardino (2016-2018). His research interests are user training, virtual communities, technology-enhanced collaborative learning, innovation in IS education, gamification in education, cloud computing, and business intelligence. His current research projects focus on studying effective ways to train or engage people to learn computer skills such as programming, databases, software applications, web development, and cloud computing. He is also interested in developing and testing training modules/hands-on assignments (e.g., encryption, blockchain, and virtual machines on the cloud) so he can incorporate them into IS courses.

Brenda Mak is professor emeritus of information systems in the Lam Family College of Business at San Francisco State University. Her research interests include mobile communication, data mining, computer art education, employee retention, hospitality customer and employee management. Dr. Mak has published extensively in prestigious journals such as *European Journal of Operational Research*, *Information Management*, *Information Processing & Management*, *Computers & Graphics*, *IEEE Transactions on Systems, Man, and Cybernetics*, *Journal of Hospitality & Tourism Research*, *International Journal of Hospitality Management*, *International Journal of Business and Systems Research*, *International Journal of Mobile Communications*, *International Journal of Innovation and Technology Management*, and others.

APPENDICES

Appendix A. SBAT Screencast Requirements and Specifications

The SBAT specifications for the scope requirement of each screencast are presented below.

1. SBAT Overview
   The objective of this assessment is to create an app that simulates the experience of ordering a product of your choice. Its functionality is similar to the FoodOrder App introduced in class, but the app should be created to promote a different type of product, e.g., books, music, cameras, and groceries. A series of screencast videos need to be created and submitted to explain how you designed, developed, and tested the app in Xcode. The overall scope of the app is indicated below:
   Required Scope: The assessment app should include the visual UI components of the product image, product price label, "Show Next Product" button, "Order This Product" button, and order/payment receipt text. The app should feature the following functionality:
   (1) When the "Show Next Product" button is clicked, the product image and product price label will be updated with information about the next product.
   (2) When the "Order This Product" button is clicked, the current product in the display will be appended to an array of products to be ordered. The itemized name and price information of each ordered product will be listed in the order/payment receipt text. In addition, the total payment information will be displayed at the end of the receipt.

2. SBAT Screencast 01 Specification: Create App User Interface
   Create a screencast to demonstrate and explain the process of creating the app user interface. The screencast should capture the creation process step by step while you are building the user interface in Xcode. You should narrate (in your own voice) detailed explanations and instructions in the screencast. It should include the following details:
   (1) Create the app as a project in Xcode.
   (2) Add the following visual components to the Interface Builder:
      - An image view to show product image
      - A label to show the current product name and price
      - A button to show the next product
      - A button to order the current product displayed
      - A text view to show a list of products ordered and the total payment amount
   (3) Label the visual components with meaningful names. For example, instead of labeling the button "Show Next Dish", name it "Show Next Book".
   (4) Add at least five product images to the Asset Catalog of your app. These images should be relevant to the product category.
   (5) Explain how to use the auto layout features to lay out the visual components. The objects should be appropriately aligned, constrained, and displayed in the Interface Builder.
   (6) Run and test the app using the built-in simulator. The user interface should be demonstrated with one of the product information features (image, name, and price) laid out correctly in the simulator.

3. SBAT Screencast 02 Specification: Set Up Data Structure
   Create a screencast to demonstrate and explain the process of setting up the data structure, including the @IBOutlets, structs, arrays, and other constants and variables to be used in the app. The screencast should capture the data structure creation process line-by-line in Xcode. You should narrate (in your own voice) to provide detailed explanations and instructions in the screencast. It should include the following details:
   (1) Open the app you created in the last screencast in Xcode.
   (2) Add the following data structures:
      - @IBOutlet for the image view, text label, and text view
      - A struct that represents the data model of your product
      - An array that defines the product list or catalog. The data type of each array element should be the struct representing the product data model. The array should contain at least five elements
      - Other constants and variables as needed
   (3) Select meaningful names and data types for each data structure.
   (4) Explain (in the narrative) the purpose of each data structure and why it needs to be stored in the specific data type.
   (5) Run and test the app using the built-in simulator. The app should run without any syntax or runtime errors.

4. SBAT Screencast 03 Specification: Code the Behavior of the "Show Next Product" Button
   Create a screencast to demonstrate and explain the process of coding the behavior of the "Show Next Product" button. The screencast should capture the implementation of the @IBAction function that controls the button to display the next product in Xcode. You should narrate (in your own voice) to provide detailed explanations and instructions in the screencast. It should include the following details:
   (1) Open the app you created in the last screencast in Xcode.
   (2) Create the @IBAction function to connect the "Show Next Product" Button.

(3) Code the @IBAction function line-by-line to display the next product information when clicked. The product list array created in the previous screencast should be used to help to implement this functionality. The various attributes of a product should be referenced through the dot notation.
(4) Explain (in the narrative) the logic and steps taken to display the next product information.
(5) Run and test the app using the built-in simulator. The app should run without any syntax or runtime errors. In the test run, the next product information should be displayed correctly when the "Show Next Product' button is clicked.

5. SBAT Screencast 04 Specification: Code the Behavior of the "Order This Product" Button
Create a screencast to demonstrate and explain the process of coding the behavior of the "Order This Product" button. The screencast should capture the implementation of the @IBAction function that controls the button to update the order receipt/payment text in the app. You should narrate (in your own voice) to provide detailed explanations and instructions in the screencast. It should include the following details:
(1) Open the app you created in the last screencast in Xcode.
(2) Create the @IBAction function to connect the "Order This Product" Button.
(3) Code the @IBAction function line-by-line to update the order receipt/payment text when the button is clicked. After appending the current product to the array of orders, a loop should be implemented to list the ordered products and calculate the total payment.
(4) Explain (in the narrative) the logic and steps taken to update the order receipt/payment text. Explain the syntax of a loop statement and the details of its implementation in this specific context.
(5) Run and test the app using the built-in simulator. The app should run without any syntax or runtime errors. In the test run, the order receipt/payment information should be updated correctly when different products are ordered with the "Order This Product" button.

6. SBAT Screencast 05 Specification: Create the Promotional Video
Create a video to promote and market your app. If you have a name and a logo for the app, explain the meaning behind the design. A story or personal reason behind the creation and inspiration of the app can also be included. Explain how users can fulfill specific needs by using your app and how you can relate to their experience. You should briefly demonstrate the app to highlight its features and encourage users to download and use it. You can also involve family members or friends to provide testimonials for your app or products. Towards the end, explain what other app features you might be planning to add in the future.

Appendix B. Semi-Structured Student Interview Questions

1. What is your midterm exam app about?
2. How did you come up with this app idea?
3. How did you develop this app as required in the exam? Please describe the process and tools you used.
4. How did you explain the key coding concepts as required in the exam? Please describe the process and tools you used.
5. How did you capture and edit the screencast videos as required in the exam? Please describe the process and tools you used.
6. Compared to the traditional paper-based exam format, how do you feel about this video-based exam? Please describe the pros and cons of this exam design.
7. Do you have any suggestions on how to improve this exam?

ISSN: 2574-3872 (Online) 1055-3096 (Print)