Proceedings of the Informing Science + Information Technology Education Conference

An Official Publication of the Informing Science Institute
InformingScience.org

InformingScience.org/Publications

Online July 6-7, 2022

STUDENT OWNERSHIP OF LEARNING: A STUDENT'S EXPERIENCE

Rena Sakai*	University of Minnesota, Crookston, MN, USA	sakai017@crk.umn.edu
Christine Bakke	University of Minnesota Crookston, MN, USA	cbakke@crk.umn.edu

* Corresponding author

ABSTRACT

Aim/Purpose	This study reports the outcome of Student Ownership of Learning (SOL) through developing a shopping application. This research aims to describe embedding agile career-like experiences into software development courses in order to improve perceived educational value.
Background	Many classes consist of lectures, homework, and tests; however, most students do not remember what they learn through passive instruction. The researchers of this study believe that SOL and Scrum can be combined to guide students as they take an active and leading role in their learning.
Methodology	This study implemented SOL and Scrum to promote learning through teacher and student collaboration. Iterative development of an ill-defined and complex software project progressed through goal setting, task determination, prioritization, and timeboxing. Following Scrum, the complex project was first broken down into small units. The development followed short periods of independent work followed by meetings; each timeboxed development cycle is modeled after a Scrum sprint. Weekly instructor-student meetings emphasized planning and reflection through code review, discussions of progress and challenges, and prioritization for the next iteration. The project followed the agile philosophy of software development flow through iterative development rather than focusing on a defined end date.
Contribution	This study provides a practical guide for successful student learning based on SOL and Scrum through project details such as project successes and iterative challenges.

Accepted by Editor Eli Cohen | Received: April 29, 2022 | Revised: June 9, 2022 | Accepted: June 10, 2022.
Cite as: Sakai, R., & Bakke, C. (2022). Student ownership of learning: A student's experience. In M. Jones (Ed.), *Proceedings of InSITE 2022: Informing Science and Information Technology Education Conference*, Article 23. Informing Science Institute. https://doi.org/10.28945/4992

Findings	This study found that SOL, when combined with Scrum, can be used to provide a career-like software development experience. Student perceptions reflect regular interactions with a subject matter expert for the development of a complex software project increased willingness to learn, helped clarify goals, and advanced development of independent programming skills.
Recommendations for Practitioners	Practitioners can share this research with faculty members from different faculties to develop the best solutions for SOL using Scrum.
Recommendations for Researchers	Researchers are encouraged to explore different disciplines and different perspectives where SOL and Scrum methods might be implemented to increase active learning through teamwork or project-based learning.
Impact on Society	This study is beneficial for creating or redesigning a course to include career-like experiences. Readers can understand that the high level of engagement and achievement achieved through SOL and Scrum are the driving forces for project success.
Future Research	Practitioners and researchers can expand the current body of knowledge through further exploration of Scrum and SOL in educational settings where the emulation of real career experiences is desired. Future research examining best practices, tools, and methods for embedding complex software development projects into programming courses would benefit instructional faculty in many technical disciplines.
Keywords	Student Ownership of Learning, Scrum, Flutter, agile, iterative development, career-like experiences

INTRODUCTION

In 2021, according to *Statcounter GlobalStats* (2021), the market share of desktops was 57%, of mobiles was 40%, and of tablets was 3%. Since 40% of people are using mobiles worldwide, Flutter development is worth considering. One of Flutter's many benefits is to allow the creation of both Android and IOS using a single code base. This is convenient for developers and users as the system is developed once for the same look and feel by both Android and IOS users.

When an outside client approached one of the instructors about a student developed shopping application, a junior and senior level student each agreed to development. Both students worked as a team on the project for two months, after which the senior level students dropped this optional project due to the weight of many other tasks and heavy senior courses. The junior-level student decided they could extend the project through their senior year, so that this case study follows an agile software development by a single student, a client, and an advisor over the course of three semesters.

This paper explores the challenges encountered in an independent study setting where a multiple platform app is developed using Flutter. The challenge was to develop using a newer programming language that was known to lack traditional learning materials. In order to efficiently develop a Flutter app in an independent research environment, the research team developed implemented: (a) weekly meetings, (b) regular feedback, and (c) a running work log. Flutter and Dart are the tools used, but the methodology for this study is agile software development.

Agile was developed by professional software engineers for use in both large and small team-based projects. Studies have been published on a broad range of agile topics such as voluntary, self-organized, and cross-functional teams in time-intensive academic settings (Zhang & Dorn, 2012). Agile's most popular framework, Scrum, has also been examined in the context of project-based learning (Saadé & Shah, 2016) and semi-capstone experiences (Magana et al., 2018; Maxim et al., 2017). This

study follows the case of an independent honors student developing software as a creative works research project.

The study focused on two goals in order to increase ownership of learning. The first goal was to examine the students' value for learning the content of the course while self-managing their learning. The second goal was to examine student perceptions of value associated with learning a new programming language. The biggest problem was the lack of guiding resources. Flutter is a relatively new framework and documentation is less developed than that for older programming languages; when searching for online resources the student found there were noticeably fewer Q&A posts for Flutter development and few how-to videos. Specifically, the study focused on the following goals for increasing student ownership of learning:

1. Increasing students' value of software development through a positive attitude toward the project.
2. Increasing SOL by allowing students to manage their learning.

When learning to create a project, a broader knowledge of the framework and semantics was gleaned through resources such as Napoli (2020) and Huynh (2021), and a book called *Beginning Flutter: a Hands-on Guide to App Development* (Napoli, 2020) was found to provide beneficial support. While the study focused on learning Flutter, additional benefits were seen through exposure to programming terminology and development skills, including widget trees, animation development, and creating an app's navigation. The supplemental text also covered creating scrolling lists and effects, building layouts, applying interactively, writing platform-native code, and saving data with local persistence. Although the project did not include a database, the student gained a foundational knowledge of how one might add firebase and firestone to a project. During development, the student also encountered gamification, which is the act of presenting people with practical tasks, increasing the difficulty of progressing, organizing, and engaging them emotionally. Further, Huynh (2021) was referenced to learn about flashcard mobile applications with gamification capabilities that could be optimized for the user experience.

One of the greatest challenges for providing authentic software development experiences in an academic setting is the time it takes to build everything from scratch. To address this, the student supplemented coding with templates, which sped up development and reduced coding errors. The use of templates has been documented to shorten timelines for development by providing a quick and hands-on introduction to development tools, basic programming, and application development and integration process in a manner similar to the jump-start provided by sample code in a course textbook (Akopian et al., 2013).

Implementation of project-based learning methods such as Student Ownership of Learning (SOL) led students toward ownership of their college learning experience, and they are more likely to succeed in achieving their academic goals. When a student is given increased opportunities for personal voice, more individual choices, and authentic learning experiences and assessments, they will not only thrive (*Weekly tip: Ownership of learning*, 2019) but also develop a more positive attitude towards a course (Corritore & Love, 2020).

LITERATURE REVIEW

The team decided to use Flutter during the mobile shopping application project because it has many benefits. In 2016, Google released an application-level framework for mobile apps called "Flutter" which allows simultaneous app development of Android and IOS by sharing the same base code. Flutter uses the programming language "Dart", developed by Google, to replace the scripting language JavaScript which is built into all web browsers. Dart is a live development environment that increases productivity allowing programmers to check functionality in real-time. It analyzes the technical aspects of Flutter, the techniques for building the user interface (UI), and the mechanisms for

managing data and the internal state (Palumbo, 2021). Flutter has an advantage in making applications when people are trying to develop different kinds of operational systems, especially mobile applications.

Flutter is a good choice for developing mobile applications in the business context (*Flutter apps in production*, n.d.). The expressiveness of the framework and Dart allows for quick programming of complex interface elements (Palumbo, 2021). Dart is a language that makes it easy to describe UI processing including asynchronous processing that does not block the UI, and collection description to make it easy to define the UI (Tashildar et al., 2020). Traditionally, app development has been complex due to differences in widgets and the need to address numerous native languages. The Flutter architecture simplifies the app development process by providing tools that allow programs to code an app using a common language, which can then be deployed on different platforms. Flutter is able to develop UI easily by using Flutter features.

One of the driving forces behind Flutter and Dart is the ability to develop a single cross-platform solution, based on a single codebase (Dagne, 2019). Cross-platform apps are an ideal starting point for both new developers and experienced native developers, as it is possible to start developing apps immediately without prior knowledge of Dart. There is no longer a need for hybrid applications, as developers can now quickly create high-quality features based on native code (Enihe & Joshua, 2020). Professionals identify the most common issues associated with cross-platform app development as user experience, technical implementation, app performance, and testability (Majchrzak et al., 2017). Both Google and Facebook saw the need for cross-platform app development frameworks. Facebook's internally developed React-Native (RN) is comparable in many ways to Google's Flutter. Both were released in 2015, each has benefits and deficiencies, and both have become viable development toolkits. Some comparisons: scrolling is much smoother in Flutter and the development of identical Android and IOS apps is best achieved through Flutter. RN, with its pioneering work, has actively impacted the followers to a certain extent, also Flutter is said to have a bright future. Sophisticated designs from RN are well preserved with Flutter's evolution (Wu, 2018). From an implementation point of view, both RN and Flutter are very friendly to developers with JavaScript experience (Jozef, 2020). It is a good choice for people who are starting to create mobile applications to use Flutter because it is able to develop cross-platform as Flutter has different kinds of benefits.

Some of Flutter's newer features include Google's Navigator and Router API. The API gives people more control over the screen design and route analysis (Walker, 2021). The router handles communication with the underlying platform and assures that the appropriate page is displayed using platform channels (Ryan, 2020). This means that the platform service can be exposed in the host app code and it can be called from the Dart side or vice versa (Ravn, 2018) as the Dart package is the directory that contains the pubspec files (*Flutter packages – javapoint*, 2021). In addition, Flutter packages can include dependencies such as Dart libraries, apps, resources, tests, images, and examples (*Using packages,* 2021).

Flutter is a complete framework that combines widgets, tools, and a programming language into a single software development kit (SDK). Flutter has several advantages over other SDKs including control of every pixel on the screen to avoid the performance issues caused by JavaScript bridges (Leler, 2017) and the ability to compile all the way to native code. Another advantage of Flutter is the ability to edit code and apply changes to an already running application; this is known as a Hot Reload and can save significant time. Dart, Flutter's native programming language, is similar to JavaScript and addresses many of the front-end development needs of mobile and web applications. The Flutter SDK also includes many widgets which aid with rapid UI development and are valuable to both programmer and customer (Thomas, 2019).

Because Flutter apps use a single code base, deployed apps are perceived by users as highly similar regardless of the user's operating system. When users compared two mobile apps, one developed in

Flutter SDK and the other created with a native Android SDK, they perceived the apps to look similar but reported feeling that the native app was faster. An examination of an app that was developed on a native platform and on Flutter, resulted in a Flutter satisfaction score of 4.6 and a native satisfaction score of 5.2. When comparing UI design, users noted the apps were quite similar with the Flutter app having a satisfaction score of 5.1 and the native app having a satisfaction score of 5.2. Through comparison studies, Google reports that 70% of users prefer a faster app over two visually identical apps (Dahl, 2019).

Agile Philosophy

The concept of agile development was formally published in 2001 in an online proposal signed by 17 programmers (Varhol, 2019). The publication of the Agile Manifesto provided a foundation upon which an agile movement was born, with the initial statement still guiding agile practice today. Agile focuses on finding better methods, practices, and activities in support of quality development by valuing interaction with individuals rather than processes and tools, software that works rather than comprehensive documentation, collaboration with customers rather than contract negotiations, and response to change rather than following plans (Highsmith, 2001). Through these simple statements, an agile developer practices values that drive successful software development.

Agile development is a prevalent software development framework that encompasses many methodologies. Foundationally, agile is a flexible philosophy of software development through small cycles known as iterations (*What is agile?,* 2021). Each iteration begins with the prioritization of project goals based on stakeholder interactions. Goals are then prioritized and each cycle addresses the highest priority requirements as agreed upon by all stakeholders. Through regular cycles that place a high value on stakeholder input, a collection of prioritized functions can be knit into a large system. In this way, agile does not focus on the up-front development of a single specification document, but rather encourages starting with broad requirements, that are refined and adjusted as determined during each development cycle (*What is Agile software development?,* 2022).

The distinct and unrepeatable stages of waterfall development are in direct contrast to agile. Whereas waterfall development begins by formalizing a detailed requirements document before coding, agile emphasizes flexibility through regular stakeholder communications that encourage requirement refinement throughout the development process (*What is agile?,* 2021). Waterfall provides the programming team with a complete map of the project prior to starting so that requirements changes are not encouraged during development. Agile development places a high value on stakeholder feedback throughout the project so that the development team starts only with high-priority functions for each iteration and an overview of the project, rather than a rigid document that does not encourage requirement changes during the project (*What is Scrum?*, 2022). Agile development was designed to develop better products more efficiently and faster while giving top priority to client needs.

Scrum

Scrum is one of the representative management frameworks in agile development. Scrum is one of the software development methods focused on the methodology for teams to work together (Villavicencio et al., 2017). In Scrum, development works are called sprints, and the work is divided into periods of about one week to one month, and the results are reviewed. In the professional realm, if a programmer is new to agile, it is helpful to provide them with an agile coach; in the classroom, the role of the agile coach is played by the instructor. In measuring the impact of agile coaching on student performance, research has found that agile's most common framework, Scrum, provides a beneficial guide. Survey data showed that students who received agile coaching through checkpoint meetings gained measurable insights into Scrum internalization and problem-solving over students that did not receive agile coaching (Rodriguez et al., 2016).

The work and things to be done are listed in a list called the backlog in order of priority, and the members of the team do this (Atlassian, 2022). There are two types of backlogs: a product backlog that lists the items required for the product being developed; and a sprint backlog that lists the items that should be implemented in the sprint (Figure 1).

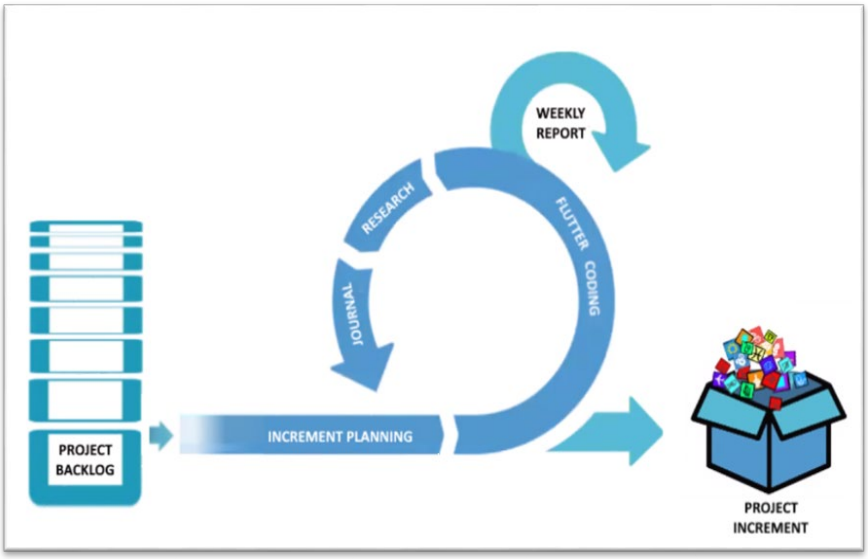


Figure 1. Academic Scrum Cycle

Regarding requirements, Scrum is the antithesis of waterfall; where waterfall compiles a large, detailed, and unchanging requirements document prior to coding, Scrum projects start with a gathering of general project requirements which are detailed and prioritized prior to each cycle. Changes to requirements in waterfall-style development are costly, whereas Scrum is designed for change and embraces clarification to requirements throughout the project. In addition to assuming that project specifications should be fluid during development, Scrum involves stakeholders in regular communication with the team as they work together to determine product backlog priorities and solve UI design problems (*What is Scrum methodology? & Scrum project management*, 2021).

Self-Selection

In academic settings, self-selection has been seen as a stressor during the beginning phase of a project, but over time team members report self-selection to be beneficial and appreciated (Md Rejab et al, 2019). This finding is similar to the report by professional agile developers that the fastest and most efficient way to form productive, small, cross-functional software development teams is with facilitated self-selection (Agile Alliance, n.d.; Mamoli & Mole, 2014). When compared, both studies reported positive findings, but the approaches were slightly different, in that professionals recommend facilitated self-selection, whereas the academic study looked at self-selection without a facilitator.

Self-selection can also be practiced through teams that self-organize tasks. It has been shown that self-organizing teams more readily adapt to changes when they are given authority to make decisions as they work together toward a shared goal (Agile Alliance, n.d). Using a bottom-up estimation and planning, self-organized teams lead the decision-making process. This is seen at the sprint level by peer balancing of workload, teams that proactively address workload, and team identification of tasks and hours needed. Agile often employs a Scrum master to provide the team with any needed education, facilitation, and guidance (Mandonca, 2016).

Student Ownership of Learning (SOL)

SOL is an applied learning framework rooted in the works of Dewey (1966). In SOL, students' skills are developed starting with an instructional shift toward facilitation, directed learning, and student engagement. It begins with a level of understanding that progresses to a level of ownership that is demonstrated by one's ability to articulate strategies and outcomes of personal learning. The apex of student ownership can be seen when teachers and students collaboratively facilitate learning through a mutual exchange of ideas and strategies (National Institute for Excellence in Teaching, 2021).

Courses that incorporate SOL increase student engagement by building on existing experiences and knowledge while encouraging student choices through project identification, goal setting, and prioritization of project tasks. When instructors encourage students to take ownership of project goals and velocity, students have been seen to demonstrate content mastery (Corritore & Love, 2020). How can such an environment be created? Students have been shown to develop expertise and confidence in their abilities through repeated interactions with projects, ideas, and important concepts.

Teachers take students further when they co-facilitate the learning process and engage them in active leadership and self-directed learning (*Learning Acceleration Resources*, 2021). Ownership of learning is typically observed in active learning settings that allow students to generate new knowledge as they become self-directed (Graus et al., 2022). To facilitate student ownership of learning, the role of the teacher must change from lecturer to coach. A teacher in this role becomes a valued resource, helping students achieve goals through active participation in decisions, choices, content application, observations, and evaluations (Chan et al., 2014; *Weekly Tip: Ownership of Learning*, 2019).

According to *Learning Acceleration Resources* (2021), "student ownership is when teachers and students co-facilitated the learning. When students are owning their learning, they are doing more than just engaging: they are actively taking a role in leading their learning. When this happens, the teacher serves more as a guide for students to take them further".

Methodology

This study employed an empirical case study design to examine the lived experiences of honor students as they learn a relatively new programming language that lacks instructional materials. An additional language challenge was faced in that the programming language was developed in English, while the student's primary language is not English. To provide a professional experience in an academic setting, project development was aligned with the agile philosophy and primarily employed Scrum practices and tools. A major hurdle was encountered at the onset when one of the two initial students dropped the project, leaving the honors student as the sole developer.

The first part of all projects focused on determining an overview of the design requirements from a real client. After the initial data collection, the students began iterative cycles of development that involved goal setting, prioritization, coding, and a reflective-planning meeting prior to the next cycle. The instructor was involved as a mentor and client liaison throughout the project, primarily serving in the role of facilitator. The rate of development in an academic setting is not as intense as that of a full-time developer, so the instructor provided both stakeholder-like feedback and coding guidance more often than the client. Project progress was tracked through a weekly journal which captured the essence of a daily Scrum stand-up meeting but on a weekly, rather than a daily basis.

Project Goals

The project goal was to create a shopping application for a client. It was decided to use a method called student ownership of learning and Scrum to develop the mobile application. Student ownership is when teacher and student work together to promote learning. When students own their learning, they do more than just engage: they are actively taking a role in leading their learning (*Learning Acceleration Resources*, 2021). Also, it was decided to use the Scrum method during student ownership

of learning. Meetings were held with the professor once a week. At that meeting, progress was reported using the work log and the next goal was set.

Project Development

At the first meeting, it was decided what functions the shopping application would have. The application includes three main items which are recipes, meals, and shopping. Each item has different functions inside. The next step was to create various UI's based on what was decided at the meeting. Then the development environment was prepared, which was installing Flutter and emulator.

A basic idea for the shopping app was decided and started to create each screen. The first thing that was needed to do was to make the bottom navigation bar. It allows users to move around to the recipe screen, meal screen, and shopping cart screen. After making it possible to switch between screens with the bottom navigation bar, the calendar function of the meal screen was started to be developed. The screen includes notifications for meal preparation times. Users can set a time and get a notification for preparing meal time.

The next step was to design subsections for store, order, and coupons on the shopping screen with icons. The store page is a page for finding a store near a user. So the Google Maps function was added to Flutter. Optical Character Recognition was added because the recipe screen needs a function to capture printed characters and handwritten characters as image data and make them editable. The last feature developed was a simple countdown timer and a gif for when the timer is over. For the time, it was necessary to run audio players on the application.

This research was conducted as part of an honor's project. The University of Minnesota Crookston Honors Program was developed to transform students' writing, critical thinking, and leadership skills. The shopping application project was researched under the guidance of a mentor. It was developed through the Scrum application by reporting progress and discussing the next goal at a weekly meeting. This method increased student value for software development through a positive attitude towards the project.

First, it was decided to use Flutter to develop the mobile shopping application because Flutter allows the creation of applications for both IOS and Android. It can operate across different platforms. Normally, when developing a smartphone application, the language is changed depending on the operating system. Therefore, if people want to create an application that supports both IOS and Android, they need to develop each one. However, if it is across platforms, the entered code will be covered to support both IOS and Android, so they can develop a system compatible with both operating systems with one development. However, Flutter has a disadvantage. Flutter is a new framework created in 2016 so it is hard to find information and examples of code.

Second, it was decided on basic functions in the shopping mobile application. The main three items in the mobile application are recipes, meals, and shopping. The recipes button includes pictures of what the dish looks like or videos that show how to cook a meal, instructions, and taking a picture of food and being able to upload it in the application. The meal button includes a calendar with some kind of organization for meals and snacks and a notification for meal preparation time based on preparation. The shopping button includes connecting with a grocery store based on geolocation, being able to order groceries, and finding coupons.

Findings

Over the course of two semesters, the student was able to develop a prototype app that included a basic app along with additional features: a bottom navigation bar, calendar, shopping page, a Google map, OCR, notification, simple count downtime, and an animated company logo-gif. Without a textbook or direct instruction, the student was able to develop a bottom navigation bar, a working calen-

dar, recipes, and shopping pages that incorporated google maps. The navigation bar provided movement back and forward between recipes, calendars, and shopping pages, whereas the calendar provided a visual scheduler for monthly or weekly appointments. The development of the shopping page morphed into three sub-categories: a store, the shopping cart, and coupons. Further, it was determined that an OCR function would be needed, which resulted in learning how to integrate and use the phone's camera. When it was realized that a shopping and recipe app would not be complete without a timer, the student expanded development to include a cooking timer and notifications.

Project discussions were often focused on the cycle's tasks; however, at times they wandered into methodologies and discussions of instructional best practices. The student frequently commented about learning a lot through the project, much more than through traditional lecture-style instruction, and often posed the question of why more instruction did not employ practical, hands-on learning. These discussions led to examinations of the differences between professional practice and academia including the differences between traditional academic models of software development. It was noted that when waterfall methods are employed in programming classes, students become experts in solving waterfall-like problems where the question is clear and the coding is straightforward and often easily found in an online forum. The student frequently commented about programming skills being greatly enhanced by the project. Because agile was employed throughout development, adjustments to the project requirements were expected and the student experience more closely mirrored a professional setting.

The project was clearly guided, with weekly goal setting, reviews, and presentations of work; a representative sample of one week's submission is included in the Appendix. The student explained her progress through code discussions, pictures, and short videos of the work. Each review meeting ended with setting goals and priorities for the next cycle, allowing the student to be part of the process while working to develop independence. Agile terminology became a natural part of the weekly review so that the student became comfortable using basic terms such as Kanban, stand-up, artifact, backlog, and Scrum master. As the project progressed, the student moved from learner to instructor, became the project expert, and began to lead the weekly discussions. By the end of the project, the instructor role was more of a coach than a lecturer, while the student often led weekly review meetings as they became emerged in the new language and the process of app development. Throughout the process, the student increased ownership through decision-making until she fully managed her own learning. In this way, both of the research goals were addressed through the development of a complex app that relied on a new programming language.

CONCLUSION

The student was able to develop a working prototype based on a real-world client proposal but did not have sufficient time to complete or deploy the app. Although the project was not completed in production, the student frequently mentioned the educational value of the project in comparison with traditional instructional methods. Flutter and Dart were released in 2017, which resulted in numerous challenges for an entry-level programmer when compared to older programming languages. Flutter development tools and resources seemed limited with few forum posts, few tutorial videos, and few well-developed instructional sites.

After reaching the prototype stage, the student felt she had gained sufficient knowledge of Flutter that she would have been able to complete the app, given more time. The core functionality and the main challenges had been completed which left relatively easy items in the backlog such as adjusting the color scheme and layout to meet client preferences, doubling a recipe, finding a way to put gif and short sounds on the simple count down timer, scheduling notifications, and converting metric or English measures. The instructor felt that of the remaining challenges there was one that might present significant challenges – that of reading an OCR picture and saving it to a new editable text file.

The instructor reflected that the project provided the student with an experience similar to an internship experience as the student encountered complex software development with minimal direct instruction. The student's feedback was positive, with the student frequently mentioning her dismay that this type of experience was uncommon in academic settings as she felt the experience was more engaging and resulted in greater learning. During the weekly review, discussions often touched on the value and challenges of student ownership, as our Scrum reports were designed for students to take the leading role in the determination of project backlog, prioritization of tasks, resource gathering, and trouble-shooting.

The student proposed three benefits of using Scrum and SOL. Firstly, the student felt that using Scrum and SOL that relied on iterative development would decrease the likelihood that students would abandon a complex project. Using regular weekly cycles that required a work log was seen as beneficial for keeping students on track, while instructor meetings provided guidance and encouragement. The reflection and goal-setting components of the weekly instructor-student interactions were highly valued. Interaction with the instructor was highly valued as an essential part of the development process through clarification, accomplishment discussions, trouble-shooting, and assistance with resource gathering. Real work experience was seen as the second benefit, as SOL and Scrum provided situations that closely mirrored development challenges that seemed likely to be faced in a career setting. The project was perceived as the opposite of traditional instruction, which involves lectures and passive learning. Once on the job, the student felt she would be expected to know how to figure out complex and challenging problems with minimal support, and that a software engineer needed to be a self-directed learner. It is good for students to experience how to learn new knowledge as it prepares them for work. A high level of engagement and ownership led to the third benefit, which the student observed to be an increase in the ability to recall learning. Incorporation of SOL, was seen to result in more realistic challenges that led to greater learning and recall, as they required engagement, research, and troubleshooting in order to resolve them. The student searches for the root cause of each problem while working to find each solution, in so doing students will become deeply engaged and remember the project long after it is over; this is a benefit that students do not get in a lecture-only class.

This research examined a way to learn that is fully engaging and adds value to a course through emersion in a project and ownership so that this study is presented to instructors, researchers, and students. Instructors may be interested in this study if they wish to explore professional practices aligned with academic projects. Researchers interested in project-based learning, professional-academic experiences, programming instruction, and student ownership of learning may find correlating studies that support increasing student engagement through the embedding of professional experiences. Researchers and faculty interested in examining hybrid delivery models that weave active learning into traditional lecture settings may also find value in this work, as similar Scrum projects have been embedded into multiple software engineering courses, with positive outcomes. Students wishing to learn a new programing language or start making a new application may find guidance and encouragement in the experiences of this project. Thus, SOL with Scrum should be incorporated whenever plausible in the academic setting.

REFERENCES

Agile Alliance. (n.d.). *Is team self-selection the obvious choice?* https://www.agilealliance.org/resources/experience-reports/is-team-self-selection-the-obvious-choice/

Agile Alliance (n.d). *What is Agile Software Development?* https://www.agilealliance.org/agile101/

Atlassian. (n.d.). *Scrum: Learn how to scrum with the best of 'em.* https://www.atlassian.com/agile/scrum

Atlassian (n.d.). *What is agile?* https://www.atlassian.com/agile

Akopian, D., Melkonyan, A., Golgani, S. C., Yuen, T. T., & Saygin, C. (2013). A template-based short course concept on Android application development. *Journal of Information Technology Education: Innovations in Practice*, *12*, 13-28. https://doi.org/10.28945/1764

Chan, P. E., Graham-Day, K. J., Ressa, V. A., Peters, M. T., & Konrad. M. (2014). Beyond involvement: Promoting student ownership of learning in classrooms. *Intervention in School and Clinic, 50*(2), 105-113. https://doi.org/10.1177/1053451214536039

Corritore, C., & Love, B. (2020). Redesigning and introductory programming course to facilitate effective student learning: A case study. *Journal of Information Technology Education: Innovations in Practice*, *19*, 91-135. https://doi.org/10.28945/4618

Dagne, L. (2019). *Flutter for cross-platform App and SDK development* [Bachelor's thesis, Metropolia University of Applied Sciences, Helsinki, Finland]. https://www.theseus.fi/bitstream/handle/10024/172866/Lukas%20Dagne%20Thesis.pdf?sequence=2&isAllowed=y

Dahl, O. (2019). *Exploring end user's perception of flutter mobile apps* [Bachelor's thesis, Malmö University, Sweden] https://mau.diva-portal.org/smash/get/diva2:1480395/FULLTEXT01.pdf

Dewey, J. (1966). *Democracy and education: An introduction to the philosophy of education*. The Free Press.

Digite (n.d.). *What is scrum methodology? & scrum project management*. https://www.digite.com/agile/scrum-methodology/

Enihe, R., & Joshua, J. (2020). Hybrid mobile application development: A better alternative to native. *Global Scientific Journals*, *8*(5), 1373-1389.

Flutter (n.d.) *Using packages*. https://flutter.dev/docs/development/packages-and-plugins/using-packages

Graus, M., van de Broek, A., Hennissen, P., & Schils, T. (2022). Disentangling aspects of teacher identity learning from reflective blogs: The development of a category system. *Teaching and Teacher Education, 111*, 103624. https://doi.org/10.1016/j.tate.2021.103624

Highsmith, J. (2001). *History: The Agile manifesto.* https://agilemanifesto.org/history.html

Huynh, T. (2021). A flashcard mobile application development with Flutter [Bachelor's thesis, Haaga-Helia University of Applied Sciences, Helsinki, Finland]. https://www.theseus.fi/bitstream/handle/10024/499862/Huynh_Tung_thesis.pdf?sequence=2&isAllowed=y

Javatpoint (n.d.). *Flutter packages.* https://www.javatpoint.com/flutter-packages

Jozef, P. (2020, December). *Flutter and react native performance overview.* https://sudolabs.io/blog/flutter-and-react-native-performance-overview/

Leler, W. M. (2017, August). *What's revolutionary about flutter.* https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514

Magana, A., Seah, Y., & Thomas, P. (2018). Fostering cooperative learning with scrum in a semi-capstone systems analysis and design course. *Journal of Information Systems Education*, *29*(2), 75-92. https://jise.org/Volume29/n2/JISEv29n2p75.pdf

Mandonca, C. (2016, March 3). *About self-organizing teams*. Scrum.org. https://www.scrum.org/resources/blog/about-self-organizing-teams

Mamoli, S., & Mole, D. (2014). Self-selecting teams part 1 - Why you should try self-selection. *Methods & Tools*. http://www.methodsandtools.com/archive/selfselectingteams.php

Majchrzak, T. A., Hansen, A. B., & Gronli, T. M. (2017). Comprehensive analysis of innovative cross-platform app development frameworks. In T. X. Bui, & R. Sprague (Eds.), *Proceedings of the 51st Annual Hawaii International Conference on System Sciences* (pp. 6162-6171). University of Hawaii at Manoa. https://doi.org/10.24251/HICSS.2017.745

Maxim, B., Acharya, S., Brunvand, S., & Kessentini, M. (2017, June). WIP: Introducing active learning in a software engineering course. *Proceedings of the 2017 Annual Meeting of the American Society for Engineering Education, Columbus, OH*, 1-12. https://peer.asee.org/wip-introducing-active-learning-in-a-software-engineering-course.pdf

Md Rejab, M., Noble, J., & Marshall, S. (2019). Agile self-selecting teams foster expertise coordination. *Interdisciplinary Journal of Information, Knowledge, and Management, 14*, 99-117. https://doi.org/10.28945/4280

Napoli, M. L. (2020). *Beginning Flutter: A hands on guide to app development.* Wiley.

National Institute for Excellence in Teaching. (2021). *Fostering student ownership through thinking and problem-solving.* https://www.niet.org/assets/Resources/student-ownership-thinking-problem-solving.pdf

Niet (n.d.). *Learning Acceleration Resources* https://www.niet.org/assets/Resources/523891961b/student-engagement-versus-student-ownership.pdf

Palumbo, D. (2021, July). *The Flutter framework: Analysis in a mobile enterprise environment* [Master's thesis, Politecnico di Torino, Turin, Italy]. https://webthesis.biblio.polito.it/19111/1/tesi.pdf

Ravn, M. (2018, August). *Flutter platform channels.* https://medium.com/flutter/flutter-platform-channels-ce7f540a104e

Rodriguez, G., Soria, A., & Campo, M. (2016). Measuring the impact of agile coaching on students' performance. *IEEE Transactions on Education*, *59*(3), 202-209. https://doi.org/10.1109/TE.2015.2506624

Ryan, J. (2020, September). *Learning Flutter's new navigation and routing system.* https://medium.com/flutter/learning-flutters-new-navigation-and-routing-system-7c9068155ade

Saadé, R. G., & Shah, S. (2016). Exploring an agile learning activity to teach agile project management. *Proceedings of Informing Science & IT Education Conference*, *Vilnius, Lithuania*, 95-101. https://doi.org/10.28945/3454

Scrum (n.d.). *What is Scrum?* https://www.Scrum.org/resources/what-is-Scrum/

Statcounter Global Stats. (2021). *Desktop vs mobile vs tablet market share worldwide.* https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet

Tashildar, A., Shah, N., Gala, R., Giri, T., & Chavhan, P. (2020). Application development using Flutter. *International Research Journal of Modernization in Engineering Technology and Science*, *2*(8), 1262-1266. https://irjmets.com/uploadedfiles/paper/volume2/issue_8_august_2020/3180/1628083124.pdf

Thomas, G. (2019, December). *What is flutter and why you should learn it in 2020.* https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/

Varhol, P. (2019, January 22). *The complete history of Agile software development.* https://techbeacon.com/app-dev-testing/agility-beyond-history-legacy-agile-development

Villavicencio, M., Narváez, E., Izquierdo, E., & Pincay, J. (2017). Learning scrum by doing real-life projects. *Proceedings of the IEEE Global Engineering Education Conference, Athens, Greece,* 1450-1456. https://doi.org/10.1109/EDUCON.2017.7943039

Walker, A. (2021, December 21). *What is an API? full form, meaning, definition, types & example.* Guru99. https://www.guru99.com/what-is-api.html

WSU (2019, January). *Weekly tip: Ownership of learning.* https://li.wsu.edu/2019/01/25/ownership-of-learning/

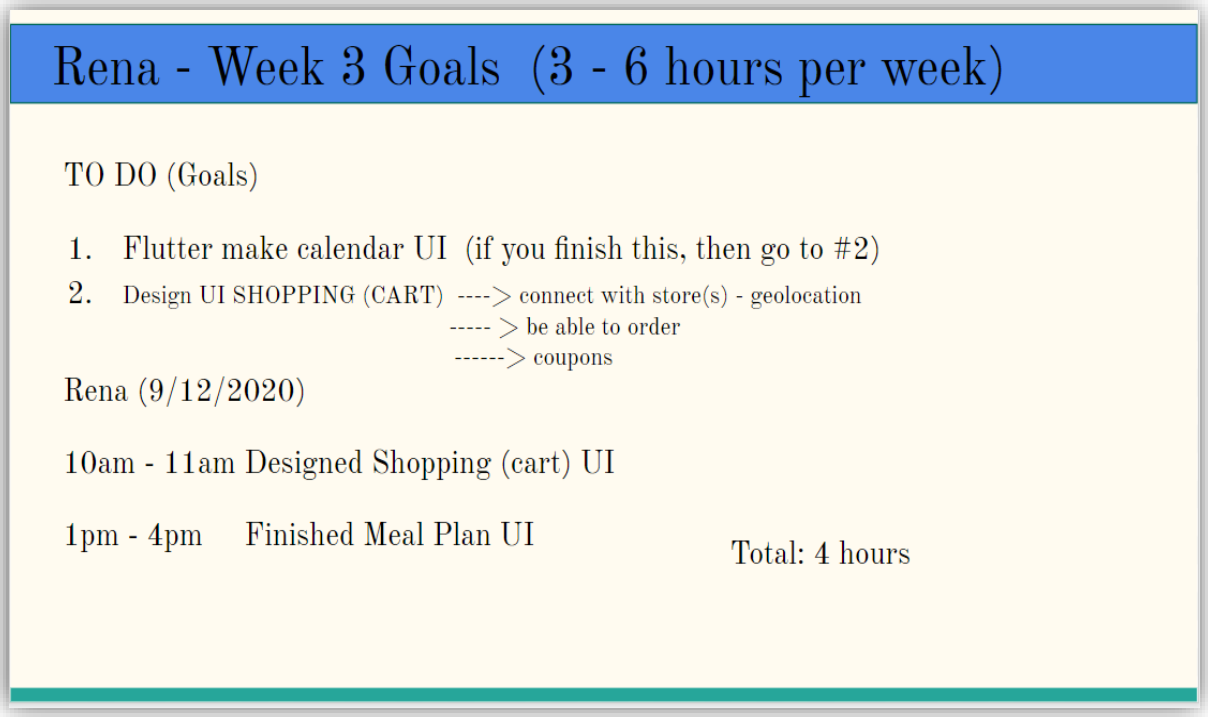
Wu, W. (2018). *React Native vs Flutter, cross-platform mobile application frameworks* (Bachelor's thesis, Metropolia University of Applied Sciences, Helsinki, Finland). https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf?sequence=1

Zhang, X., & Dorn, B. (2012). Accelerating software development through Agile practices – A case study of a small-scale, time-intensive web development project at a college-level IT competition. *Journal of Information Technology Education: Innovations in Practice*, *11*, 25-37. https://doi.org/10.28945/1545

APPENDIX

Weekly Goals	• Plan what to do by the end of next week • Continue unfinished tasks from the previous week • Write down goals for next week
Work Time	• Talk about what progress was made in the past week • Recode beginning and end work time, date, and what worked.
Resources	• Explain helpful resources • List helpful sources (URL) • Add a five-star rating for each source
Code & Problem	• Discuss problems that could not be solved • Explain the code • Demonstrate the mobile application (run the code) • Take pictures of the code • Share solutions to a challenge • Make a list of newly learned terms

Excerpts from a Weekly Scrum Report



Rena - Week 3 Goals (3 - 6 hours per week)

TO DO (Goals)

1. Flutter make calendar UI (if you finish this, then go to #2)
2. Design UI SHOPPING (CART) ----> connect with store(s) - geolocation
   -----> be able to order
   ------> coupons

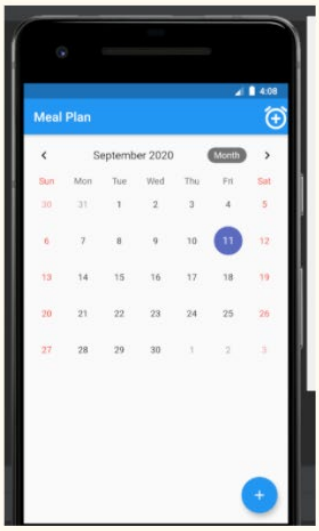
Rena (9/12/2020)

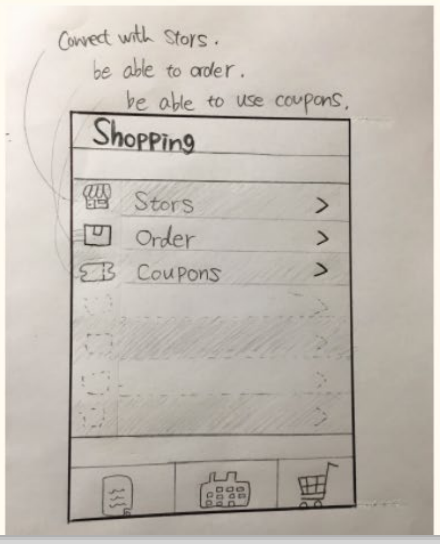
10am - 11am Designed Shopping (cart) UI

1pm - 4pm Finished Meal Plan UI

Total: 4 hours

Rena - Week 3 Goals (3 - 6 hours per week)





Code & Problem

```
import 'package:flutter/material.dart';
import 'package:meal_flutter_app/mealPlan.dart';
import 'package:meal_flutter_app/shopping.dart'; //
import 'package:meal_flutter_app/stors.dart';
import 'package:meal_flutter_app/recipes.dart';

class Nav extends StatefulWidget{
  @override
  _NavState createState() => _NavState();
}

class _NavState extends State<Nav> {
  int _seletedItem = 0; // Make value for a current
  List<Widget>_widgetOptions = <Widget>[
    recipes(), // This is for a bottom navigation ba
    Stors(), // This is for a bottom navigation bar
    Shopping (), // This is for a bottom navigation
    mealPlan(),
  ]; // <Widget>[]

  void _onItemTap(int index){ // This code tells the
    setState(() {
      _seletedItem = index;
    });
  }
```

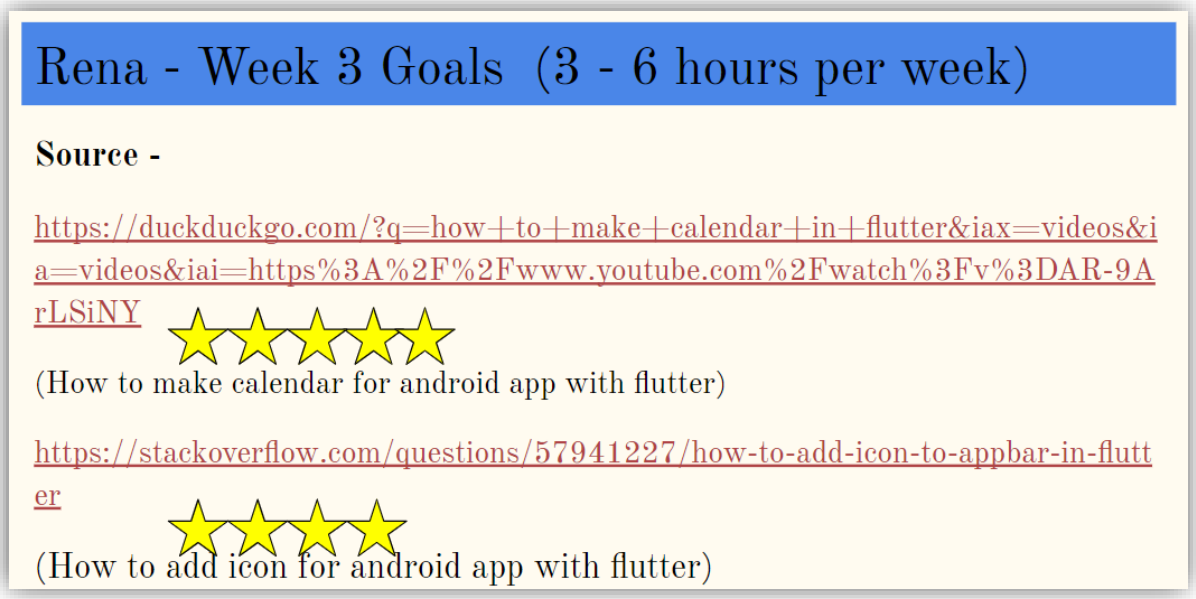
```
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: _widgetOptions.elementAt(_seletedItem),
    ), // Center
    //start bottomNavigation
    bottomNavigationBar: BottomNavigationBar(
      backgroundColor: Colors.red,
      selectedItemColor: Colors.blue,
      unselectedItemColor: Colors.blue.shade200,
      items: <BottomNavigationBarItem>[
        BottomNavigationBarItem(
          icon: Icon(Icons.note_add),
          title: Text('Recipes')), // BottomNavigationBarItem
        BottomNavigationBarItem(
          icon: Icon(Icons.store),
          title: Text('Store')), // BottomNavigationBarItem
        BottomNavigationBarItem(
          icon: Icon(Icons.shopping_cart),
          title: Text('Shopping')), // BottomNavigationBarItem
        BottomNavigationBarItem(
          icon: Icon(Icons.calendar_today),
          title: Text('Mealpaln')), // BottomNavigationBarItem

      ], // <BottomNavigationBarItem>[]
      currentIndex: _seletedItem,
      onTap: _onItemTap,
    ), // BottomNavigationBar
```

This code for a bottom navigation bar.

Dialog - a type of modal window that appears in front of app content to provide critical information or ask for a decision

Alert dialog - interrupt users with urgent information, details, or actions.

The color of the bottom navigation bar disappears when I add another button. So I asked question in flutter github.



Rena - Week 3 Goals (3 - 6 hours per week)

Source -

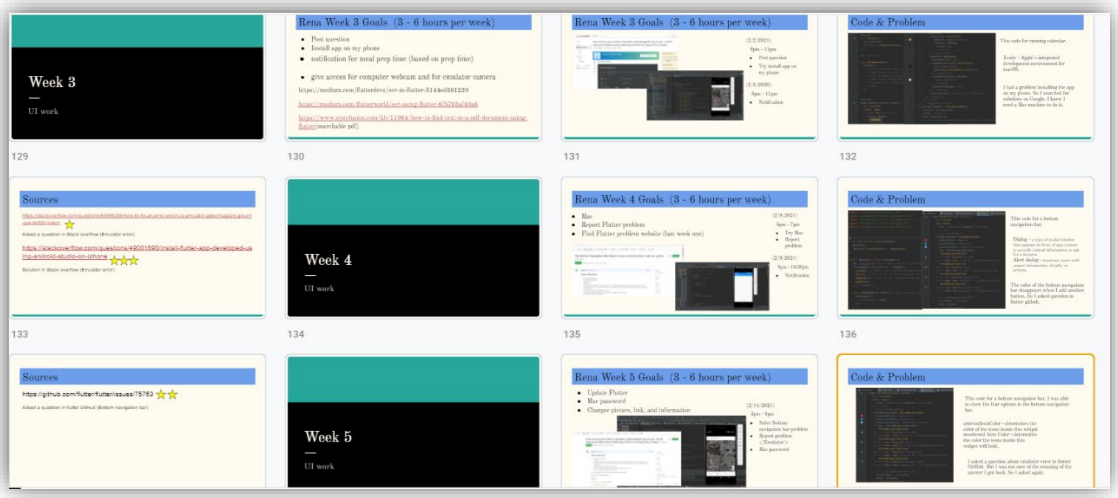
https://duckduckgo.com/?q=how+to+make+calendar+in+flutter&iax=videos&ia=videos&iai=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DAR-9ArLSiNY

(How to make calendar for android app with flutter)

https://stackoverflow.com/questions/57941227/how-to-add-icon-to-appbar-in-flutter

(How to add icon for android app with flutter)

Example showing continuous Scrum reporting



AUTHORS

Rena Sakai is an honors student at the University of Minnesota, Crookston. She is working to complete her Bachelor's degree in Software Engineering, with minors in Information Technology Management and Cybersecurity. Her research interests center around new ways to learn, innovative and immersive methods, and project-based instruction. Her previous creative works studies include Flutter programming, Student Ownership of Learning, and instructional design founded in active learning strategies.

Dr. Christine Bakke is a Software Engineering and Information Technology instructor at the University of Minnesota, Crookston. Her Information Technology PhD degree specialized in educational robotics, while her undergraduate was in Computer Science, programming, and mathematics. Her professional experience includes 18 years as an IT professional with specializations in networks, cybersecurity, database, and programming. Her research focuses on active learning techniques that combine academic and professional best practices into agile active learning experiences. Recent projects include Scrum-like development of chatbots, speech assistant software, and custom IoT devices with software.