Project title: iCons Equipment Rental APP

Team number: 724U

Bradley Stephen 20207842 - Team Lead

Charles Ko 20200207

Casper Harris 20238963

Ethan Lapid 20176605

Jesse Sugarman-Weir 20232291

Robert Yu 20216708

Project manager name: Sydney van Engelen

Faculty sponsor name: Dr. Il-Min Kim

Client name: Jacob Calderone

Date submitted: April 4th, 2022

Executive Summary

The team's objective in this project is to develop a mobile application to allow students to borrow equipment at the Integrated Learning Centre (ILC) in Beamish-Munro Hall at Queen's University in Kingston, Ontario. The app would replace the current in-person interaction and collateral exchange with the Integrated Constables (iCons). iCons are engineering students who volunteer as a resource to help undergraduate students with their studies. One of the iCons' many services is to allow students to rent equipment in the ILC. The app must allow students to view the items available and select which ones they would like to borrow, and the app should aid iCons in finding the student to deliver the equipment. The project is estimated to cost $99 per year to keep the app in the Apple App Store.

To accomplish these requirements, three design options were considered. The first option is to develop an app using XCode, Apple's programming language used to develop software for iOS. XCode has many advantages, including the ability for team members to work on the app simultaneously, an intuitive programming environment requiring only basic coding experience, and numerous premade functions to simplify the construction of a user interface. The primary disadvantage is that it is only compatible with iOS devices and would be unavailable to Android users. The second solution is to design the app in MIT App Inventor, which works similarly to XCode but uses logic blocks with selectable options rather than code, severely limiting data handling and troubleshooting capabilities. MIT App Inventor also does not allow multiple contributors to work simultaneously and is only compatible with Android systems. The

third option is to develop a website instead of an app. This would have all the capabilities of the first solution, while allowing access from iOS, Android, or a computer.

The three options were compared based on accessibility, capabilities, the prerequisite technical knowledge required by the team, cost, usability, and update flexibility. Based on these criteria, it was determined that the website would be the optimal solution, with the XCode app a close second. However, upon discussion with the client, the website option was rejected, and XCode was selected. To determine what would be the most user-friendly experience for Queen's engineering students, the team created a survey asking questions regarding the front end of the app. Front-end refers to what the user sees when interacting with the application, such as the user interface. To test the application, the team used debuggers in the XCode environment to guarantee that the application would operate as expected.

Table of Contents

Table of Contents

List of Figures and Tables

Section 1.1: Problem Statement and Scope Definition

The Integrated Constables or iCons is a student-run service that aims to assist and support undergraduate students studying at the ILC (Integrated Learning Centre) in Beamish-Munro Hall at Queen's University in Kingston, Ontario. One of its primary services is the loaning of equipment to students in the ILC. Currently, the process to rent equipment is as follows: a student approaches an iCons staff on shift and asks to rent a piece of equipment. In return, the student must leave their student card as collateral. The iCons have deemed this system to be inefficient due to difficulties tracking inventory and desire an app to streamline the process. The app should allow students to see what equipment is available and rent said equipment via the app. iCons on shift should be notified that a student has rented equipment and should be given the location to which they must deliver the equipment in the ILC. The client has also suggested that the app could link to the SOLUS accounts of student users to charge them for equipment that is damaged or not returned. Ideally, the iCons are hoping that the application is user-friendly and able to receive user feedback. Preferably, the app would be compatible with a diverse number of platforms such as Android and IOS. Possible reasons for a new system of equipment rental include computers being autonomous and more efficient than manual labor, thus making the rental process much faster. Furthermore, the process of manually adjusting the list of available equipment would be replaced by a program that automatically adjusts the list as students make orders to rent. This system would not only make the rental process much faster, but it would also limit the number of mistakes via human error due to a computer being much more efficient. Furthermore, if the team can link SOLUS to the app, the process of replacing stolen or broken equipment would be streamlined as payment would automatically be taken from the student SOLUS account rather than relying on the iCons to solve the problem manually.

The two major stakeholders are both the students and the iCons. The iCons are being directly impacted by the changes that the app will provide, as it will change the current rental system that they run. The students are also a major stakeholder as they require equipment that they do not have, so students rely on the iCons rental service to allow them to study. Students have a need for such items, and the change in the system will directly enable them to obtain them. This could help the students save money and alleviate potential stress in obtaining specified items.

There are various constraints that the team is limited by, which will impact the development of the product. The team has a limited budget of $100 which limits the software and hardware resources that can be obtained. The team is also constrained by the little programming experience. Many of the team members have never developed an app before, and with only seven weeks to develop the app, the team will be limited in what can be produced for the final product.

Due to the lengthy and complex process of developing an app and the team's minimal experience in programming, much of the allocated time will be used to learn how to program and develop an app, further emphasizing the time constraints. Given this, the team can realistically deliver an app with the following functions: the app allows one to log in as one of two profile types. One would be exclusive for iCons to log in, and the other for students where everyone can log in. Students will be able to register and log in to the app using their Queen's email. Once logged in, the students will be able to order equipment from a list of available equipment. The list will automatically change when a student orders a piece of equipment to reflect the current stock. The students will be able to enter their room number in the ILC along with their order of equipment. For iCons, the app will allow them to receive notifications of student orders, which include the equipment and location of the student. The iCons will also be able to adjust the list of available equipment manually. This project will be considered a success if it contains all the functions described below, the front end, which the user interacts with, is easy to use and comprehend, and is compatible with either iOS or Android. Furthermore, SOLUS will not be able to be linked to the app due to red tape. The ability to manage students' finances and personal information would constitute a major breach of privacy.

Section 1.2: Background Information

There are two distinct challenges in this project, creating an approachable user interface, and being a function replacement for the current equipment rental system. The back end required more research and technical information to complete. Many different software programs were found in the research phase that help do some of the functional work. Firebase is a program run by Google that provides a server to store data, user authentication, and assists with the debugging process by tracking crashes [1]. "57% of all app users have either uninstalled an app over concerns about sharing personal information or declined to install the app" [2]. Firebase provides the necessary source for security from Google databases, but it requires an integrated development environment (IDE) to be manipulated for use. The IDE chosen is Apple's XCode, this is the software environment that is the organizer of all the information [3]. XCode supports many different coding languages, but this project was written entirely in Swift.

Coding Languages allow humans to talk to the computer to use the tool in the most effective way. Key words in swift code such as, "state variables" and "bindings" connect the code to the user interface so lists, titles, and buttons can be seen and used. Lists in this case are order queues, inventory tracking, and login information [4]. XCode has a built-in graphical user interface (GUI) with iPhone size templates so the swift code can be visually represented at every stage of the app build [5]. One way to test the app is to allow new people to search through the functions of the app using the GUI to find errors. The second way to test the app is by utilizing a debugging tool. A debugging software built into XCode can be utilized to run through the app looking for possible "bugs" which would appear as an error message to any user that went through the same path [6].

The research into front end design found industry strategies to guide the user while navigating the page so it can be operated with as little user learning curve as possible. People use all applications with similar habits, so utilizing industry standards in the design will help the users feel more comfortable with the app faster. The "X" button is found in the top corner and means "exit page" and is put there universally so that when a user would like to exit the page they look straight to the top corners [7]. The "go back" button in the app is an example of this. This style of "X" button is used because it lets the user know an action has been done to get to that stage. The aesthetic of the page is a factor in the user learning curve as well, people see the aesthetic and have an instant emotional response. If the page looks more complicated than it is with patterns and images everywhere, then it is likely they will have a harder time understanding the features because of the distraction [8]. For the app to be successful it will need to bare bones simple in design and visually appear familiar and inviting.

Section 1.3: Design Solution

After much research and discussion, XCode was the chosen method for the development of the icon rental application. The reason for choosing XCode was due to members of the team being most familiar with its IDE (integrated development environment) and the programming style of Swift UI, which is an object-oriented programming language.

Table 1 - List of Software Used

Software	Description	Functions for Project
XCode	An IDE to develop software for macOS, iOS and others.	To build the basis for the phone application.
Firebase Cloud Fire store	A flexible, scalable cloud database that stores and syncs data for clients.	To help store and update data across client apps.
Firebase Authentication	A way to build secure authentication.	To help with set up and create accounts for users, and to implement security for the accounts for using the application.
Firebase Realtime Database	A cloud-hosted NoSQL database that allows the storage and synchronizing of data.	To store and sync data between users in the application.

Table 1 shows a list of the software used in this project, including Firebase and its derivatives. The software shown were either used to design, program or handle data with the implementation and maintenance of the phone app.

In the development of a mobile application, mapping out how the user interface will flow is a priority before writing the code behind it. It enables the developers to know the logic and specific target functions when programming code and presents how each objective of the mobile application is met. The mapping out of the UI was adjusted alongside the coding development. We chose to do this in the form of a flow chart, where each shape on it depicts a certain function or process.

A flow chart (Figure 3) was made with the following logic seen in Figure 1.

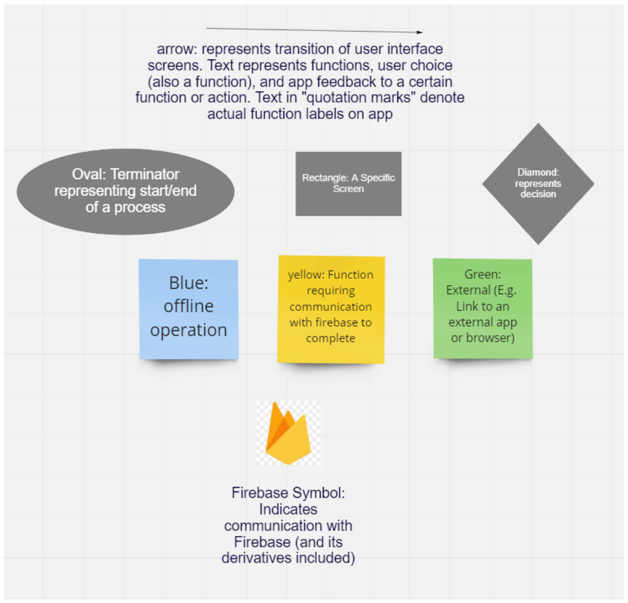

arrow: represents transition of user interface screens. Text represents functions, user choice (also a function), and app feedback to a certain function or action. Text in "quotation marks" denote actual function labels on app
Oval: Terminator representing start/end of a process
Rectangle: A Specific Screen
Diamond: represents decision
Blue: offline operation
yellow: Function requiring communication with firebase to complete
Green: External (E.g. Link to an external app or browser)
Firebase Symbol: Indicates communication with Firebase (and its derivatives included)

Figure 1: Legend for flowchart

Swift UI also offers a declarative approach to data handling, which is a programming style of building structures and elements and the logic of how the program works without describing the software control flow. Control flow is handled using the framework of Swift UI which has prebuilt data allocating functions. Examples of these functions include:

“$ReadData$ ()” which reads data from a text file, “$WriteData$ ()” Which writes data into a text file, and “$DeleteData$ ()” which deletes data from a text file. Parameters are passed into the bracket of these functions, dictating when the function should be called. Firebase is integrated into XCode, which can assist the back-end development of the app, storing data such as a student’s email address, password, username, equipment available to rent, and a timestamp. Firebase will also allow Queen’s students to sign up, sign in, sign out of the app, and give an option to students who forget their password to create a new one. This works by retrieving and storing data.

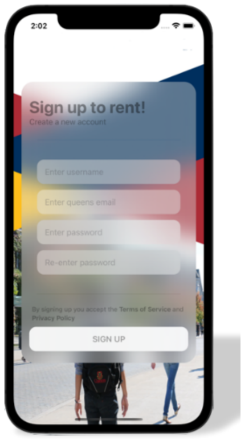


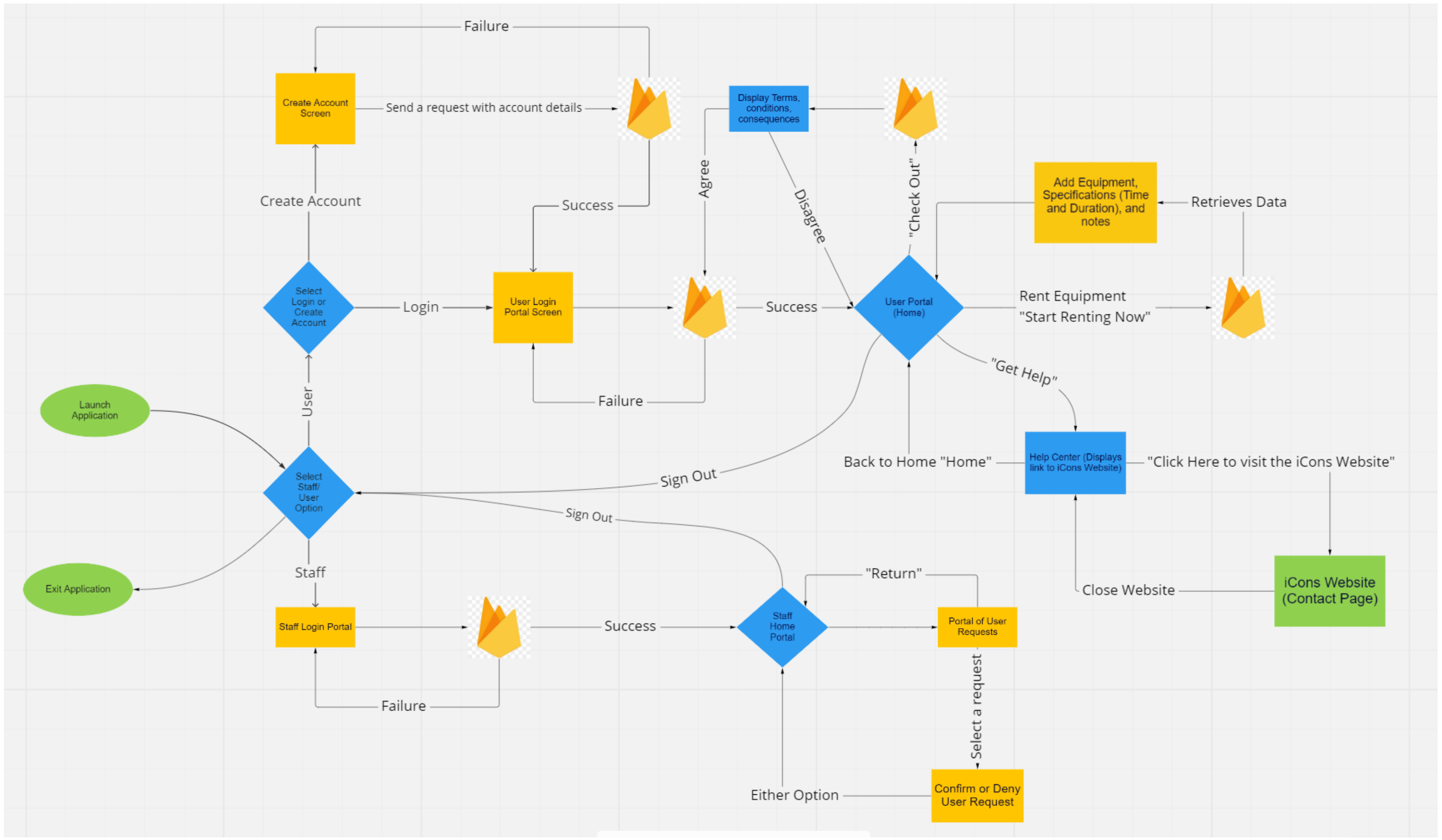
Figure 2 - The Login Portal Screen Created from XCode

A user would enter their email and password, Firebase will then store this information allowing the user to sign into the app and be authenticated by comparing email entered to email stored (this will be helpful for allowing only icons to sign into an icon account). Swift also allows for the configuration of views, which can be done by setting up a hierarchy of views. This allows the developers to easily see the relationship between the front and back-end of the app, such as the relationship between buttons and lists to the logic of the program, which is extremely important and convenient. Other buttons are also used to carried out a variety of tasks such as a "get help" button. Included with this submission is a video file which demonstrates all these functions and buttons. The video also shows where the data is saved in firebase.

Testing and evaluation is carried out in two stages. The first stage consisted of front-end and back-end testing performed separately by their respective teams. Back-end testing will consist of testing the functionality of the app as the code is being written and troubleshooting problems as they arise. Front-end testing revolves around aesthetic and usability assessments of the interface. As these are subjective judgements, they cannot be made solely by the team, as it is impossible to avoid positive or negative bias when assessing one's own work.

To impartially assess the interface, a survey was posted on the social media accounts of team members. This survey can be accessed in the **Error! Reference source not found.**The survey contained questions that respondents answered with a number between 1 and 10. The survey was created in Google Forms, as it allows forms to be easily shared as a link and can automatically format results as a spreadsheet in Google Sheets, which can in turn be converted to a file compatible with programs such as Microsoft Excel. Google Forms also carries some disadvantages. The only way to prevent the same respondent from answering multiple times is to require those filling out the survey to log in with their Google accounts. Once responses have been collected, the quantitative ratings will be averaged across the respondents and used to modify the interface design and repeat the process. Once the average rating of each category is at least 7, the interface is deemed acceptable.

The next steps taken by the client would be centered around implementing it to their current system. This includes signing the employees up manually for authentication for their respective duties and setting up an automated payment of $100 to the app developer's account to retain the app on the iOS App Store. Employees need to be signed up to make it possible for them to receive the student's orders and deliver the requested items. Implementing the iCons App to other platforms such as Android and a desktop accessible version, would better this App's functionality objective regarding accessibility, which is later discussed in decision making. As this was out of the realistic project scope, following similar methodologies used in this project to implement alternate versions is a suggested next step for the client.



Failure
Create Account Screen
Send a request with account details
Display Terms, conditions, consequences
Agree
Disagree
"Check Out"
Add Equipment, Specifications (Time and Duration), and notes
Retrieves Data
Create Account
Success
Select Login or Create Account
Login
User Login Portal Screen
Success
User Portal (Home)
Rent Equipment "Start Renting Now"
"Get Help"
Failure
Launch Application
User
Select Staff/ User Option
Sign Out
Back to Home "Home"
Help Center (Displays link to iCons Website)
"Click Here to visit the iCons Website"
Sign Out
Exit Application
Staff
"Return"
Close Website
iCons Website (Contact Page)
Staff Login Portal
Success
Staff Home Portal
Portal of User Requests
Select a request
Failure
Either Option
Confirm or Deny User Request

Figure 3 – Flowchart of the User Interface and functions of the App

Section 1.4: Conclusions

Three various solutions to automating the iCons rental service were carefully designed and evaluated to produce an optimal solution: Model 1- XCode, Model 2- MIT app inventor, Model 3 Webpage. Model 1 built in an IOS environment, Model 2 built in an android environment, Model 3 built in a Webpage html environment all provided a solution to the problem. However, each model had its pros/cons and differed in: Student accessibility, Capabilities, Perquisite knowledge, Cost, User friendly, Update flexibility. In conclusion Model 1 was chosen to be the final design proposal as it is cost effective, Capable of implementing functionality, user friendly, and accessible to most of the student body.

For the next step, if the client wishes to use students' finances as collateral for stolen or broken items, the client will need to work the Queen's University to obtain access to students' SOLUS accounts which would give the client the ability to charge students who steal or break rental items. The client can do this through Firebase by making it so that students can log into the app with one's SOLUS account.

For the app to reach production level multiple post grad students with app development and data structure experience should peer review the entirety of the code to search for data leaks bugs. Cyber security is an important factor for production level apps, as it ensures all user data is secured. Before production the app should also be launched to apples beta testing site where selected users can test out the app to ensure it is working smoothly.

To launch the application and make it available to Queens' students the app must be uploaded to the appl store where it will go under review for 90 days. If the app is found to be sufficient for production level.

Section 2.1: Conceptual Design

The conceptual design solutions are tailored to the client's specifications: an interactive mobile application that automates the process of letting students rent equipment at Beamish Munro Hall. The first model is seen in Figure 1 and was created with Apple's development software known as XCode. All three models are created using different app development software. Each model solution has its respective unique compatibilities with operating systems (OS): there are restrictions for some models on which platform the model is compatible with IOS/Android/Web browser, and which they can be built on (Mac/Windows). Nonetheless, all models provide a solution to automating the iCons current rental service process.

Model 1: XCode

Figure 2 depicts a mock-up sign-in page of the icon rental app built in XCode. The XCode framework has pre-built tools which allow for easy implementation of the design's required functionality. The advantages of the XCode model are the following: All team members can work on the project simultaneously, lesser amounts of programming knowledge are required to develop applications, it has clean layout features which allow for smooth user experience, and it achieves all required functionality within the project scope. The only disadvantage of this model is that only students with an iOS device can download the application.

Model 2: MIT app inventor

MIT app inventors' framework shares similarities with XCode but is used for developing android applications only. The main difference is that MIT app inventor works with logic blocks. The disadvantage of logic blocks is the limited functionality achievable: implementing features such as singing out equipment requires data to be fetched from a database, and handling strings of data with logic blocks are not feasible due to the lack of debugging software for it. Without being able to debug problems occurring in the app, going through mass amounts of data becomes impractical. Another downside to this model is team members would be unable to simultaneously develop the code. As an app becomes more complex, the number of blocks becomes too large to be updated.

Model 3: Webpage

Model 3 is modeled after the current student booking system for a study room at the library. Available rooms would be displayed along with time date/time slots. The advantage of creating a website is its accessibility, since it can be used even without a phone, by using the computers provided at libraries or the Queen's facilities. Additionally, web development requires minimal previous programming experience. However, this model does not meet the specification of being a mobile application format, despite it being compatible with mobile phones.

Section 2.2: Decision Making

Model 1-XCode, Model 2-MIT app inventor, and Model 3-Webpage were compared in an evaluation matrix as seen in Table 2. The criteria which evaluated each model are as follows: Student accessibility, Capabilities, prerequisite knowledge required, Cost, User friendly, and Update flexibility. Table 4 describes each of the criteria and reasoning for their weight values.

Table 2: Evaluation matrix

		Model 1 - XCode		**Model 2 - MIT app inventor**		**Model 3 - Webpage**	
Criteria	**Weight**	**Score**	**Weighted**	**Score**	**Weighted**	**Score**	**Weighted**
Student accessibility	30	4	120	2	60	5	150
Capabilities	25	5	125	3	75	4	100
Prerequisite knowledge required	15	3	45	3	45	3	45
Cost	10	4	40	4	40	5	50
User friendly	10	5	50	3	30	5	50
Update flexibility	10	5	50	3	30	4	40
Total	100	430		280		435	

Models 3 and 1 were the optimal choices for solving the problem. The team proceeded with Model 1 as it met the iCons requirement of making the rental service app based. The specific criteria were given weight values based on their level of importance within the problem scope. Making the rental services available to as many students as possible is crucial as a model that only solves the problem for a small number of people is not optimal. Thus, student accessibility was weighed the highest. Following Student accessibility, the capabilities of each model were weighted the second highest. To develop an app that meets all functional requirements within the limits of the constraints the development software needs to be capable of handling these tasks. MIT app inventor has limited capabilities to implement all the project requirements which is why it scored a 3 in this section. Prerequisite knowledge was given the third-highest weighting due to the importance of completing the project on time. Using development software that requires extensive research is not feasible given the period to complete the project. Additionally, cost, user-friendly, and update flexibility were weighted the lowest. All three models require under $100 to develop which means less emphasis is put on the cost of the project. Update flexibility is a criterion that takes place after the project is finished. Thus, more emphasis is placed on criteria taking place during the development stages. The user-friendly criteria are based on user experience and aesthetics, criteria that correlate to overall functionality are more pivotal.

Section 2.3: Implementation

In section 1.3 - design solution a general model was created for the app. The model consists of flow charts to organize the interface navigation, a list of desired functions/buttons, and mockup user interfaces. These general models were implemented into our design to produce a functioning app.

Table 3 consist of all functionality and how they are implemented.

Table 3: List of functionality and how it is implemented

Functionality	**Implementation**	**Description**
Creating a user interface	Views and Navigation View	Contents inside View are presented to user
Navigating through the Interfaces	Navigation Link	User taps on Navigation link to present a view
User Sign Up	Firebase Authentication, Sign up function	Firebase authenticates based on email/password. The function uses Boolean algebra
User Log out	Firebase Authentication, Log out function	Firebase logs out user, function puts user login at state of false.
User Create Account	Firebase Authentication, Sign Up function	Firebase stores users' data to create accounts.
Storing data	Firebase Fire store	Firebase Fire store is the apps database which holds collections of data, each collection has a set of documents
Adding data (add to cart)	Add to cart function, button	The function creates a new document in firebase and stores

		the name of the item. The button is presented to the user, when clicked it executes the function.
Deleting Data	Delete Data function, button	The function deletes a document in firebase or specific content in a document. The button is presented to the user, when clicked it executes the function.
Getting data from a user	Text Field, counter	A text field allows a user to enter in text which can later be used in the app.
Dynamic list of equipment, Shopping cart	List view / For each	A list view iterates through all documents in a collection and outputs that data in the form of a list. For each acts the same way but can be set to range 1-(n-1) like a for loop

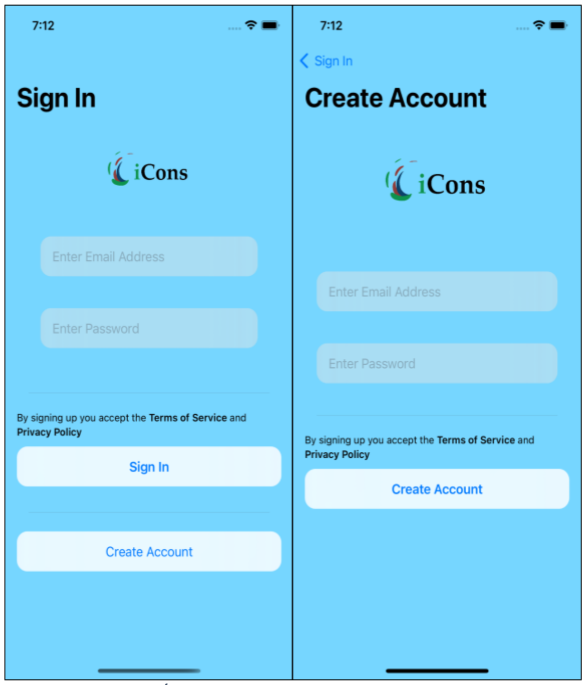
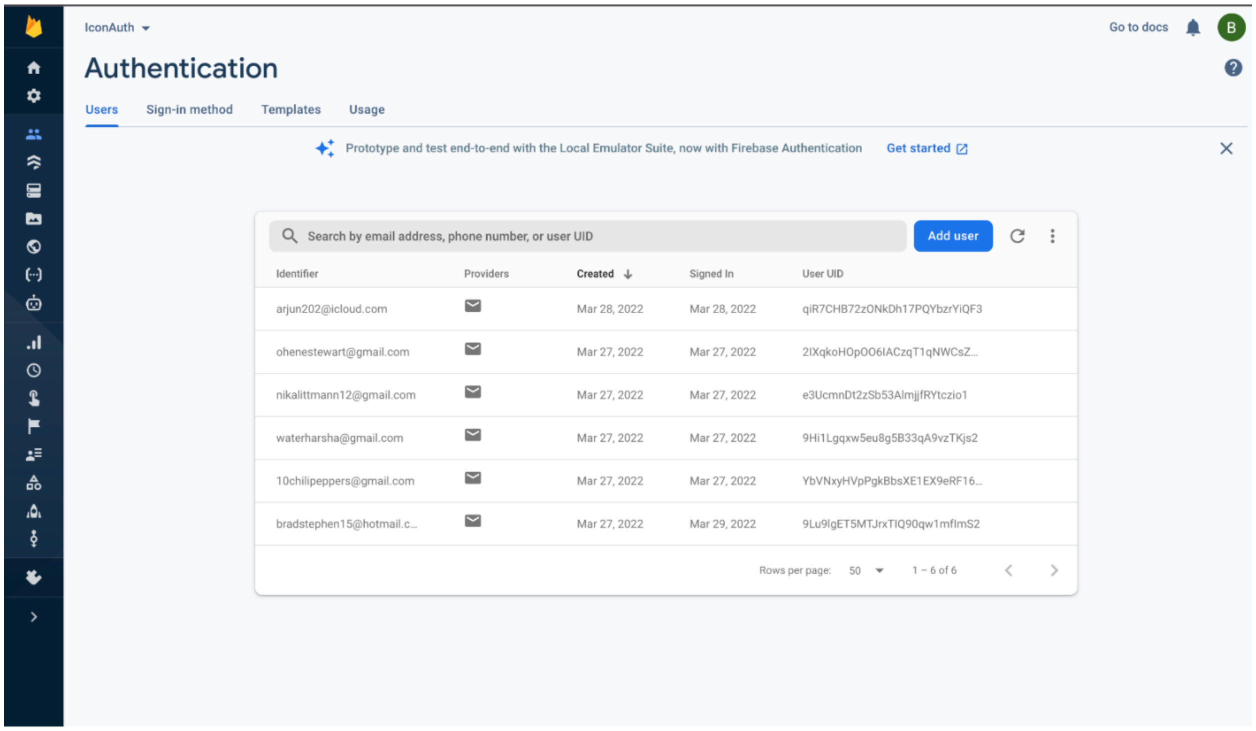


Figure 4: Sign in/ Sign up page

The First implementation was configuring and importing firebase. To achieve this, the Firebase handbook instructions were followed, refer to the appendix for the firebase configure steps. To allow users to Sign in, Sign Up, and Logout the functions as seen in Figure 4 were written to achieve these functionalities. As seen in Figure 4 the user in prompted to enter their email address and password though a text Field, there is also a button labelled “Create Account”. When the button is clicked, it executes the Sign-up function which stores the user’s information in firebase as seen in


IconAuth
Go to docs
B
Authentication
Users
Sign-in method
Templates
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
Search by email address, phone number, or user UID
Add user

Identifier	Providers	Created	Signed In	User UID
arjun202@icloud.com	✉	Mar 28, 2022	Mar 28, 2022	qiR7CHB72zONkDh17PQYbzrYiQF3
ohenestewart@gmail.com	✉	Mar 27, 2022	Mar 27, 2022	2IXqkoHOpOO6IACzqT1qNWCsZ...
nikalittmann12@gmail.com	✉	Mar 27, 2022	Mar 27, 2022	e3UcmnDt2zSb53AlmjjfRYtczio1
waterharsha@gmail.com	✉	Mar 27, 2022	Mar 27, 2022	9Hi1Lgqxw5eu8g5B33qA9vzTKjs2
10chilipeppers@gmail.com	✉	Mar 27, 2022	Mar 27, 2022	YbVNxyHVpPgkBbsXE1EX9eRF16...
bradstephen15@hotmail.c...	✉	Mar 27, 2022	Mar 29, 2022	9Lu9lgET5MTJrxTlQ90qw1mflmS2

Rows per page: 50 1 – 6 of 6

Figure 13. If the user is returning, then they are prompted to sign in as seen in Figure 4. When a user attempts to sign in, the sign in function is called and cross references the email/password entered with the email/password stored. If the user is authenticated, then the state of sign in proceeds true.

Once a user is signed in the home page will be the view shown to the user as seen in Figure 5**Error! Reference source not found.Error! Reference source not found.**. The home page is a Navigation view and contains 3 Navigations Links and 1 button. If the Log out button is clicked the Log out function in Appendix A is executed which Logs the user out and bring them to the Sign in page. The navigation link titles "Start Renting Now" Navigates the user to the Renting page when clicked as seen in **Error! Reference source not found.**. To output a list of available equipment a List view is used. It retrieves the available equipment from the "$equipment.Swift$" file as seen in Appendix A Figure 14 using the "get Data" function as seen in Appendix A Figure 15 and **Error! Reference source not found.** which displays the image, name and quantity of each equipment to the user. Next to each available equipment in **Error! Reference source not found.** is a add to cart button. When the button is clicked the "add Data" function is executed as seen in Appendix A Figure 15 and stores the item as a document in the cart collection within firebase as seen in Appendix A Figure 16 . Once the user has selected the items to rent their cart can be accessed from the home page as seen in Figure 5 by clicking the Navigation link labelled "Check out". The Check-out page as seen in **Error! Reference source not found.** retrieves the content of the user's cart from firebase using the get "Data function" and outputs the contents using a list view. The "My cart page has a delete button labelled by the minus sign which executes the delete data as seen in appendix A Figure 17 which deletes the item from the user's cart and firebase storage. The cart page also consists of two text fields which prompts the user to enter their name and room number When a user is ready to confirm their order their name and room number entered will be added to the firebase document which contains all the items in the cart, this updated document will act as a recite for the transaction. The recite is sent to the icon staff member email account to notify them of the transaction. The iCon staff members will have access to the equipment list which can updated at any time.

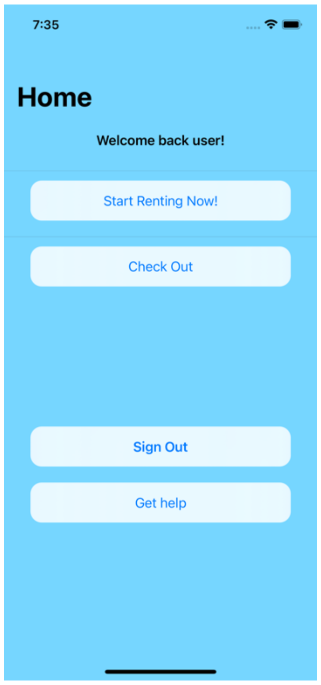


Figure 5: home page

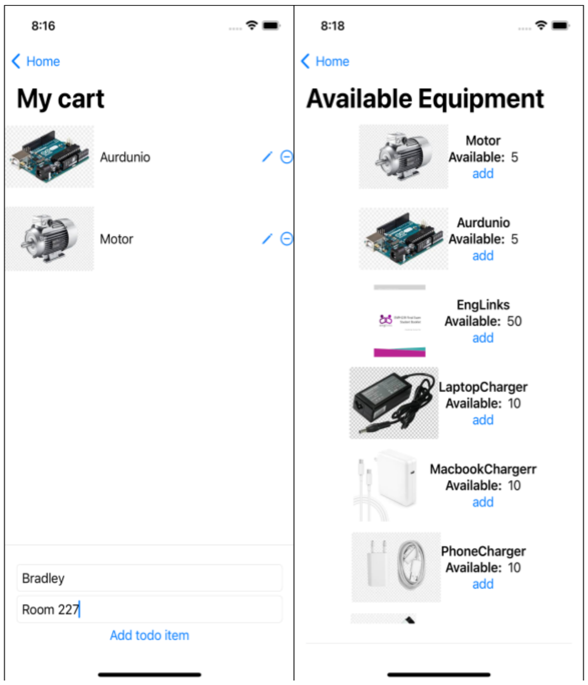


Figure 6: My cart and Rent page

To achieve accessibility and ensure user safety a help page was created as seen in Figure 7. The help page contains a hyper link which takes the user to the iCons website, from here the user can access any information regarding the iCons such as contact information. The second hyper link takes the user to the Queens social media guidelines which every user most follow.

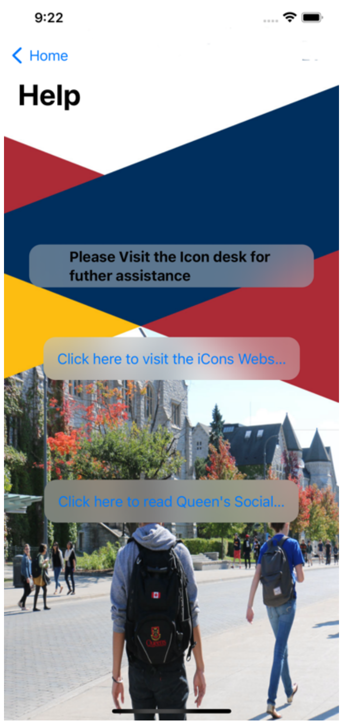

9:22
Home
Help
Please Visit the Icon desk for futher assistance
Click here to visit the iCons Webs...
Click here to read Queen's Social...

Figure 7: Help page

Section 2.4: Project Plan

The Work breakdown structure (WBS) is shown at Table 3 - Work Breakdown Structure and the Gantt chart shown in Figure 4 – Gantt Chart. The Gantt chart depicts the individual and team activities visually by showcasing their duration for the entirety of the project, which was based off the WBS and adding specific activities. While the Gantt Chart and project largely remained the same, the only difference between Phase 3 and 4 was that the dates for writing the reports for these respective phases were slightly pushed from their original dates.

If the team were to start on the project all over again, the biggest difference in the group's approach would be to ascertain the true reason why the iCons want an application and understand why the process is "inefficient". The client's solution may not always be the best. Earlier in the project, the team suggested to the client that a webpage would be the best approach to the client's problem, however, since development for the app had already begun, changing the direction of the project was rejected by the client. Perhaps if the team had proposed this solution earlier, the group would have been permitted to work on the webpage instead.

Section 2.5: Financial Analysis

The developing tool, XCode, is free and only requires a MacOS to be able to use and run. Team members that worked on the back-end development all had a computer that was able to run the MacOS: those of which that had a Windows OS used a free cross-platform virtualization software to run the MacOS to be able to use XCode. The design solution in total costed under $100, which meets the team’s budget. The budget for this project was allotted implementing the app into the iPhone App Store, since to publish an app on the App Store it is required to be subscribed to the Apple Develop program which costs $99 a month.

Financial factors did not play a complex role in deciding between options for solutions since all of them were free. Instead, the time costs and how well each solution met design specifications were highly prioritized. Since the actual implementation of the mobile app took most of the budget, the team did not explore any options outside of the free ones. The advantage of our current solution is that it has minimal variable and initial costs. However, since the app can only be developed and updated on one OS, for users without a MacOS, they would require above average computer specs to run the MacOS virtually. This restriction did not apply to our team specifically, but this may become a decisive factor financially for other future teams that may want to update and work on this mobile app.

Table 4: Firebase cost breakdown

App Indexing	Free
Analytics	Free
App Distribution	Free
App Indexing	Free

Cloud Firestore(1GiB)	Total up to 1 GiB of free storage, Then $0.108 per GiB.
Cloud Messaging (FCM)	Free
Crashlytics	Free
Dynamic Links	Free
Hosting	Free (10 GB 360 MB/day)
In-App Messaging	Free
Performance Monitoring	Free
Realtime Database	(200,000/database $5/GB $1/GB)
Remote Config	Free
Cloud Storage	$0.026/GB $0.12/GB $0.05/10,000 times $0.004/10,000 times
Test Lab	Up to 60 minutes of free time per day Then each device is charged $1 per hour Up to 30 minutes of free time per day Then each device charges $5 per hour

Table 5: Software pricing

Software products	**Cost in Canadian Dollars**
Firebase Authentication	None
Cloud Fire store	None
Firebase analytics	None
XCode IDE	None
Apple Development Program	$99 per year

	Sum in Canadian dollars = $99

Table 4 and Table 5 break down the overall cost of the project. Firebase offers free authentication for up to 10,000 users a month, Queens university is home to roughly 6000 engineers thus the model meets the requirements for free usage. Additionally, firebase offers free crash analytic software for 1 project. Cloud fire store provides 1 gigabyte of available data space. A string of data that holds the data of 1 student has a data size of 20 bytes, and a string of data that holds available equipment has a data size of 18 bytes. Assuming 10,000 users download the application, and their data must be stored: 10,000 users x 20 bytes per user data string + 20 items x 18 bytes per equipment item = 200,360 bytes of data. 1 Gigabyte is equivalent to 1 billion bytes, hence there is more than enough storage availed in the free plan.

Section 2.6: Evaluation

To reiterate the expectations from the scope of the project, a successful application for integrated constables (iCons) will include the following features: Two login paths, one for iCons and the other for students. The students log-in section will allow them to create an account with the application and sign in with the new account using their Queen's email. All login information is stored securely. Students have access to a list of available equipment that automatically updates due to a new order request. When a student is placing an order, they can specify the room number for delivery and a notification with all the order information is sent to the iCons on call. The app must help the iCons track inventory. As of now the above specifications are almost entirely implemented into the application.

The application is fully functional for the purpose of replacing the current ordering system that the iCons use with the shortcomings only affecting the ease of use. Students can log in through a separate path but are only able to create an account using an alternative Gmail account. Firebase does have a specific domain function to make using your queen's email possible, but it requires formal permission from queens. Our solution for the notifications is not the expected notification that the client described but does exist in the form of an updating scrollable list that updates for each new order. This acts as an order queue and is a standalone page for the iCons login path. The Queens email cannot be used yet so SOLUS cannot be used for billing if the equipment is damaged after a return. This means the iCons working at the time will have to sort it out and collect the student's ID card as a return incentive. The issue of not using solus is not very harmful for the success of the app because it is inevitable for the student and iCons to interact when dropping off and returning the equipment.

A group member used the iCons service to borrow white board markers for a meeting. Please see the incident report written in the appendix. Roughly, the information about when the iCon service would close was not known, this led to the group member being unable to return the markers in time. This problem was not specifically known going into the project but is solved in the app system by the information page shown to all the app users.

The client emphasized usability so the application can be integrated into use with minimal hassle and confusion. To ensure the user interface is designed to meet this standard, the graphical user interface (GUI) was tested by a group of various students who are likely to use this system in the future. Please see the link for the survey and the responses in the appendix. The survey allowed 16 people to compare two different examples of possible GUI designs and decide which one they prefer and why. When the two designs were compared directly against each other the survey showed an even split as seen in **Error! Reference source not found.**.

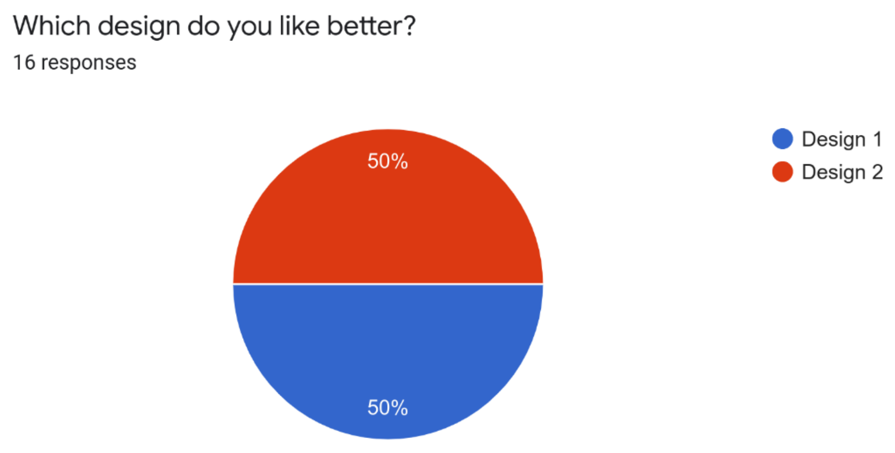


Figure 8: Comparison of two designs

The result is not helpful, but to accommodate for this possibility the follow-up questions allowed all participants to rate the understandability of each design separately, and design 1 stood out as the obvious choice.

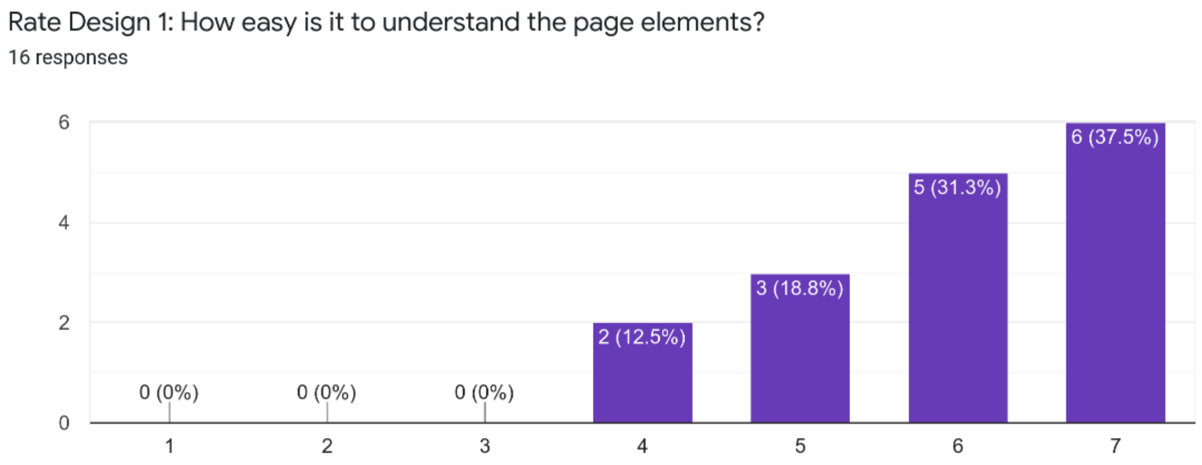


Figure 9: Comparison of 2 designs

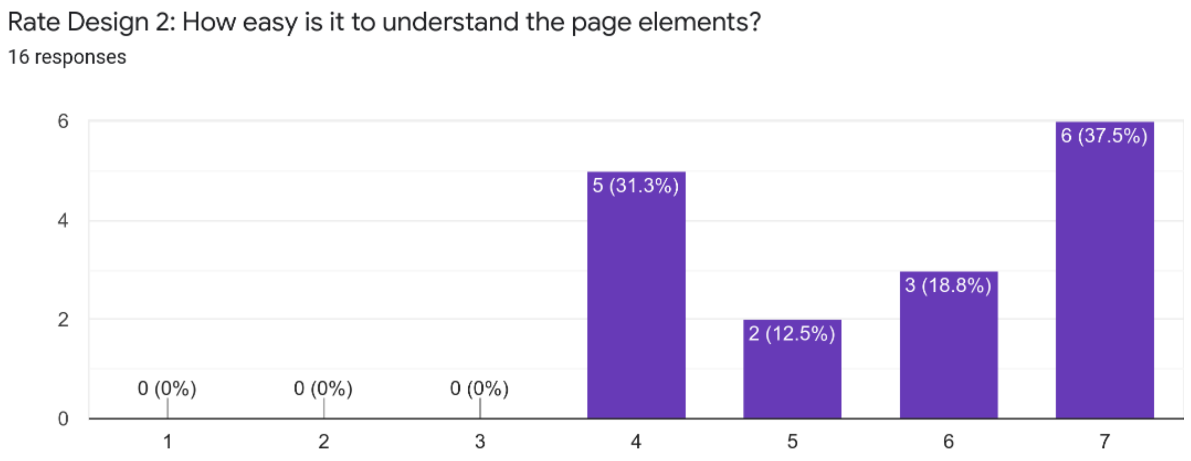


Figure 10: Design 2 feedback

The survey shows 100% of the senses group understands how the app would be used [figure 3].

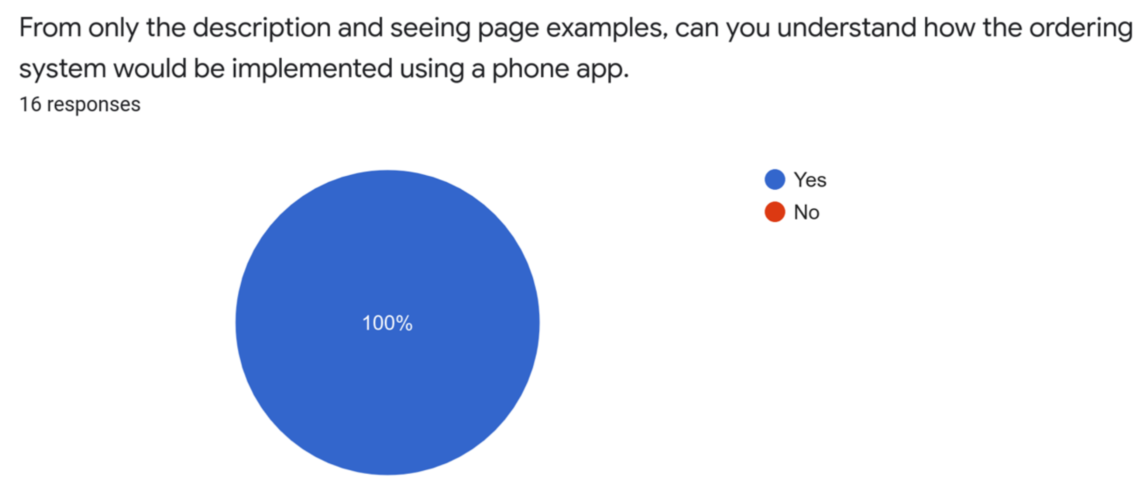


Figure 11: Feedback on implementation

The survey also collected written feedback for each design, and the reason for choosing one design over the other. The most common written response was favoring simplicity. The white background of design one allows for easy reading. A main drawback to design one is that it lacks creativity, but our project goals and research support the idea that making the app extremely simple will allow it to be used more effectively.

Now a specific quantifiable evaluation, based on each section of table 3 in the appendix. For the student accessibility category, the application meets the level four standard because the app is currently only compatible for iPhones, but iPhones are used by a large majority of the Queen's student community. For the capability category, the application meets the standard for level four and could very easily meet the standard for level five. The application functions well enough to be a beneficial addition for the iCons ordering system. The current capabilities of the team limit our ability to gain the proper permissions to reach the level five standard, but the research and exploration found it is possible with a small adjustment, so the category is given a 4.5. For the perquisite knowledge category, the application meets the level five standard which the survey proves the user does not need any technical background information to easily guide oneself through the app functions. The cost category is still a passing grade of less than 100$ which is a level four, the level five standard goes beyond the scope of this project because the one cost was 99$ to post the app on the app store and is an unavoidable cost as per the client's requirement. User friendliness can also be extrapolated by the survey which meets the level four standard because of imperfections in the chosen design as seen in figure 2. The update capabilities are flawless due to the automatic dynamic list types and therefore, fit the level five standard. The total of all the categories gives this project a score of 88% [26.5/30]. Please see table 3 in the appendix for the highlighted rubric.

In addition to the feedback obtained via the surveys, data on users' habits will be collected using FireBase throughout the app's lifetime. FireBase uses Google Analytics to allow the administrators of the app to monitor up to 500 different events such as screen changes, time spent on each screen and button clicks. This information can be used to identify areas in which users are struggling. For example, if users are frequently opening the Inventory page and then returning to the Student Portal without ordering any items, it could be indicative of a bug hindering the functionality of the interface. If a user views the privacy policy and ceases to use the app, it could mean that data gathering does more harm than good.

FireBase Analytics can also be integrated with Crashlytics, which organises and sorts crash reports according to frequency and severity, and provides potential reasons behind crashes. Using Crashlytics alongside FireBase Analytics allows app administrators to easily see who is experiencing frequent crashes and what they were doing when they occurred, and guides them through the process of determining the cause, and either debugging or alerting users, via notifications, of actions that are known to commonly lead to a crash.

References

[1] "Build Documentation | Firebase Documentation." Firebase,

[2] J. L. Boyles, A. Smith and M. Madden, "Privacy and Data Management on Mobile Devices," PewResearchCenter, Washington, D.C. 20036, 2012.

[3] "Xcode." Apple Developer, https://developer.apple.com/xcode/

[4] Apple Developer Documentation. https://developer.apple.com/documentation/swift/dictionary

[5] Apple Developer Documentation. https://developer.apple.com/documentation/xcode.

[6] Debugging Tools. https://developer.apple.com/library/archive/documentation/DeveloperTools/Conceptual/debugging_with_xcode/chapters/debugging_tools.html.

[7] "What Is User Interface Design?" The Interaction Design Foundation, https://www.interaction-design.org/literature/topics/ui-design.

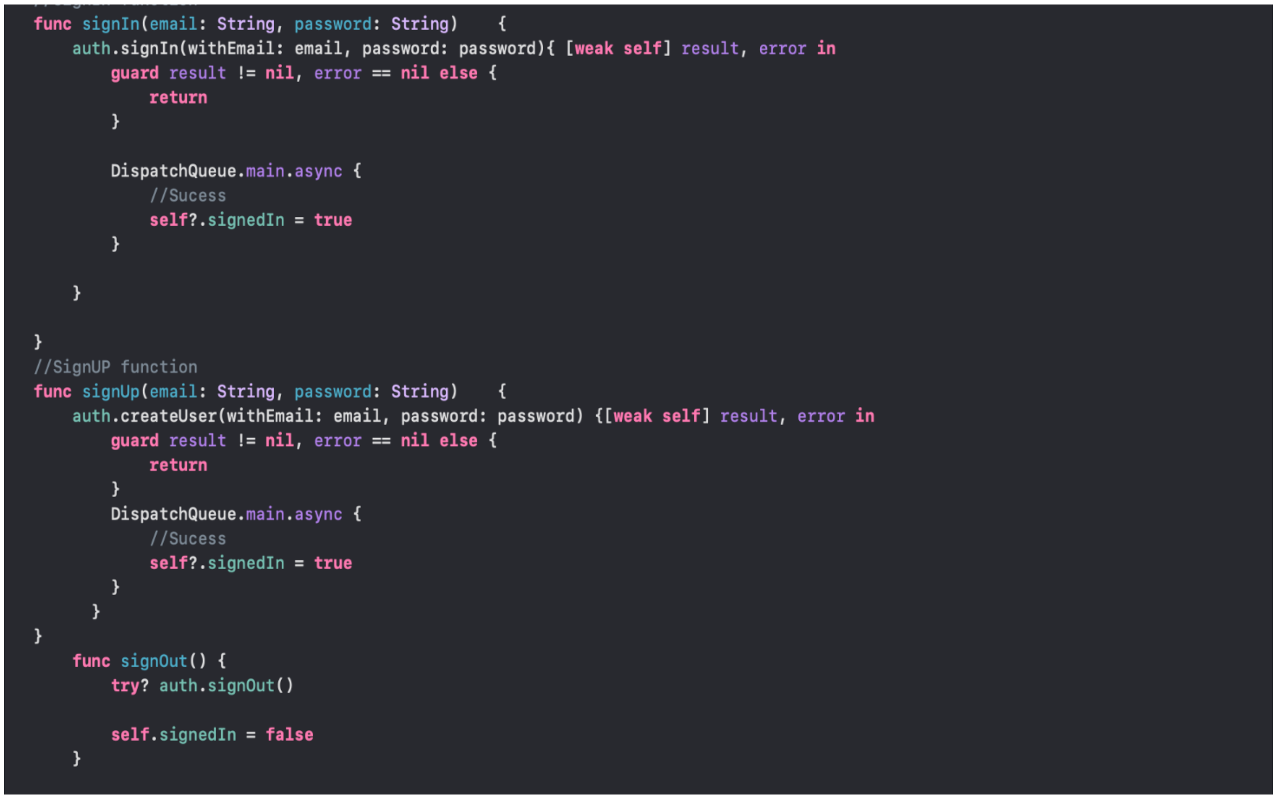
[8] G. Lindgaard, G. Fernandes, C. Dudek, and J. Brown, "Attention web designers: You have 50 milliseconds to make a good first impression!," Behaviour & Information Technology, vol. 25, no. 2, pp. 115–126, Mar. 2006.

Link to the editable survey and responses: (CLICK HERE for responses)

Link to the non-editable survey: (CLICK HERE for survey link that was sent to sample group)

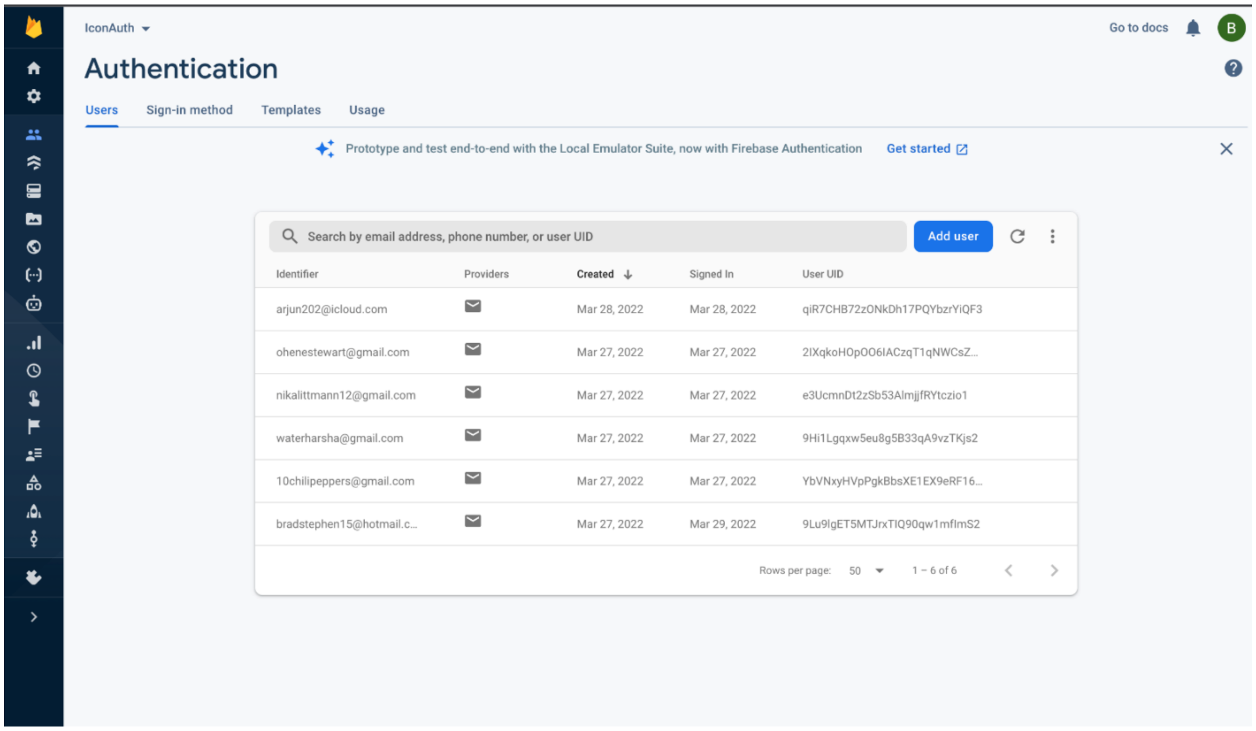
Task	Description of Activity	Activity Duration (hours)	Individual Responsible for Activity
Section 1.1	Wrote the project definition, scope, and constraints	5	Ethan Lapid
Section 1.2	Condense the Information from the research	4	Casper
Section 1.3	Write and edit the design solution	4	Charles, Bradley
Section 1.3 Flowcharts	Create and design flowchart	1	Charles
Section 1.4	Write and edit	1	Charles, Ethan
Section 2.1	Conceptual Design Solutions	2	Charles
Section 2.2	Decision Making	3	Bradley, Charles
Section 2.3	Implementation	5	Bradley
Section 2.4	Project Plan	4	Jesse, Casper, Charles, Robert, Ethan
Section 2.5	Financial Analysis	4	Bradley, Charles, Robert
Section 2.6	Made a survey and wrote evaluation referencing the survey results not the last two paragraphs	6	Casper
Appendices	Format, edit	2	All
Editing existing sections	Fixing Grammar and Spelling, edit based on feedback	3	All

Appendix A – Code and Firebase



```
func signIn(email: String, password: String)    {
    auth.signIn(withEmail: email, password: password){ [weak self] result, error in
        guard result != nil, error == nil else {
            return
        }

        DispatchQueue.main.async {
            //Sucess
            self?.signedIn = true
        }

    }

}
//SignUP function
func signUp(email: String, password: String)    {
    auth.createUser(withEmail: email, password: password) {[weak self] result, error in
        guard result != nil, error == nil else {
            return
        }
        DispatchQueue.main.async {
            //Sucess
            self?.signedIn = true
        }
      }
}
    func signOut() {
        try? auth.signOut()

        self.signedIn = false
    }
```

Figure 12: Sign in/ sign up/ log out functions



IconAuth

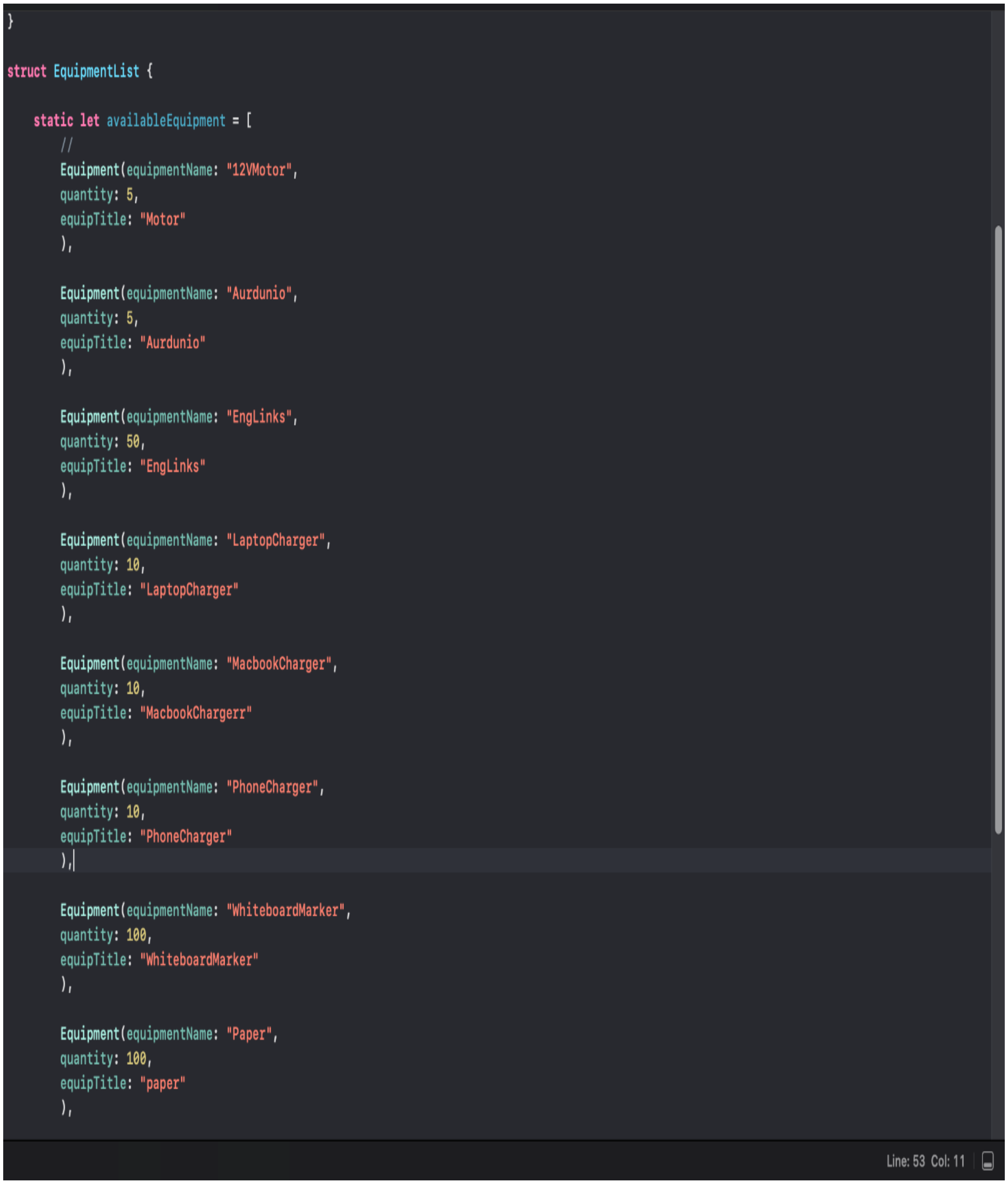
Authentication

Users
Sign-in method
Templates
Usage
Go to docs
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
Search by email address, phone number, or user UID
Add user

Identifier	Providers	Created	Signed In	User UID
arjun202@icloud.com	✉	Mar 28, 2022	Mar 28, 2022	qiR7CHB72zONkDh17PQYbzrYiQF3
ohenestewart@gmail.com	✉	Mar 27, 2022	Mar 27, 2022	2IXqkoHOpOO6IACzqT1qNWCsZ...
nikalittmann12@gmail.com	✉	Mar 27, 2022	Mar 27, 2022	e3UcmnDt2zSb53AlmjjfRYtczio1
waterharsha@gmail.com	✉	Mar 27, 2022	Mar 27, 2022	9Hi1Lgqxw5eu8g5B33qA9vzTKjs2
10chilipeppers@gmail.com	✉	Mar 27, 2022	Mar 27, 2022	YbVNxyHVpPgkBbsXE1EX9eRF16...
bradstephen15@hotmail.c...	✉	Mar 27, 2022	Mar 29, 2022	9Lu9lgET5MTJrxTIQ90qw1mflmS2

Rows per page: 50 1 – 6 of 6

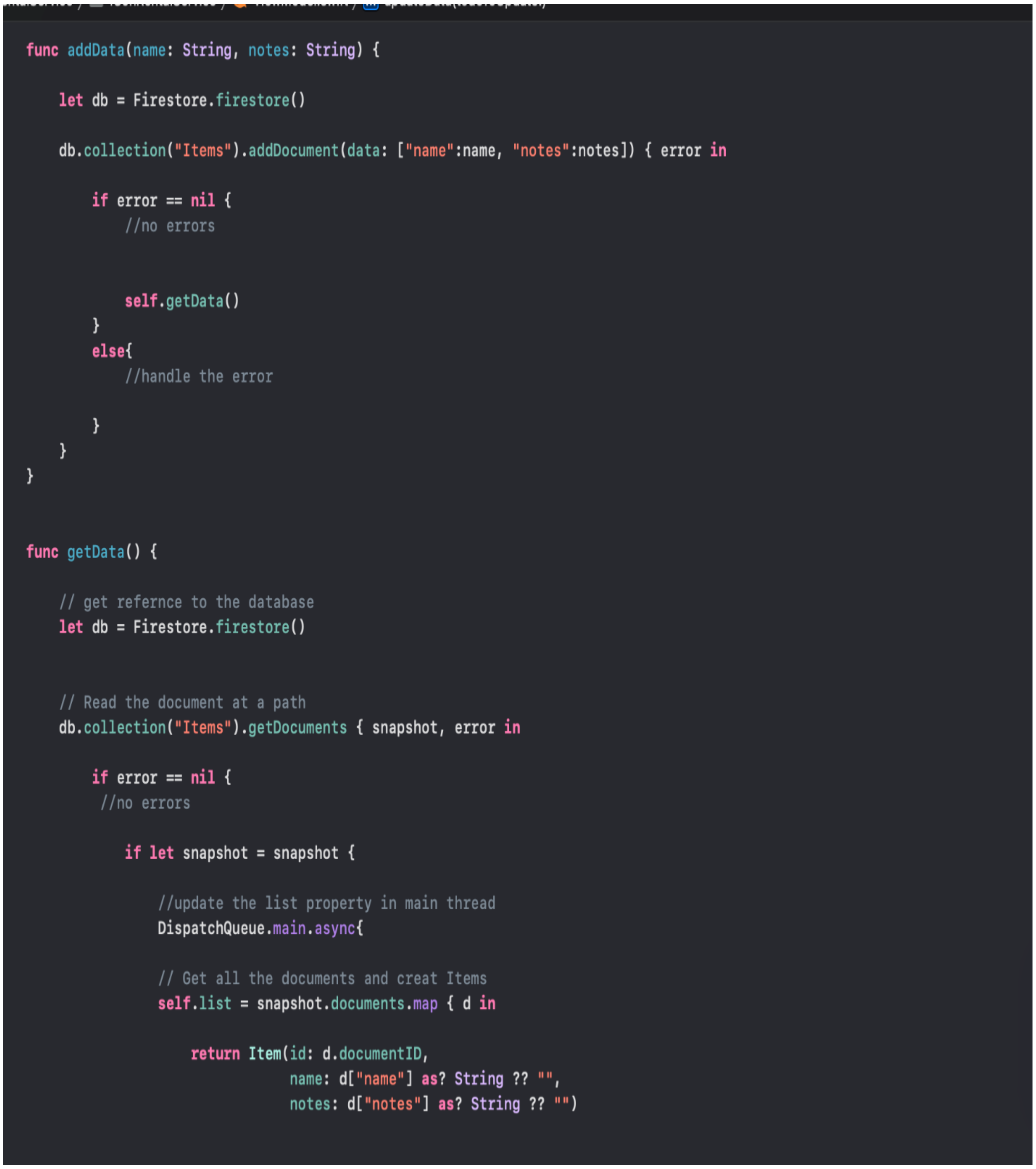
Figure 13: User information storage



```
}

struct EquipmentList {

    static let availableEquipment = [
        //
        Equipment(equipmentName: "12VMotor",
        quantity: 5,
        equipTitle: "Motor"
        ),

        Equipment(equipmentName: "Aurdunio",
        quantity: 5,
        equipTitle: "Aurdunio"
        ),

        Equipment(equipmentName: "EngLinks",
        quantity: 50,
        equipTitle: "EngLinks"
        ),

        Equipment(equipmentName: "LaptopCharger",
        quantity: 10,
        equipTitle: "LaptopCharger"
        ),

        Equipment(equipmentName: "MacbookCharger",
        quantity: 10,
        equipTitle: "MacbookChargerr"
        ),

        Equipment(equipmentName: "PhoneCharger",
        quantity: 10,
        equipTitle: "PhoneCharger"
        ),

        Equipment(equipmentName: "WhiteboardMarker",
        quantity: 100,
        equipTitle: "WhiteboardMarker"
        ),

        Equipment(equipmentName: "Paper",
        quantity: 100,
        equipTitle: "paper"
        ),
```

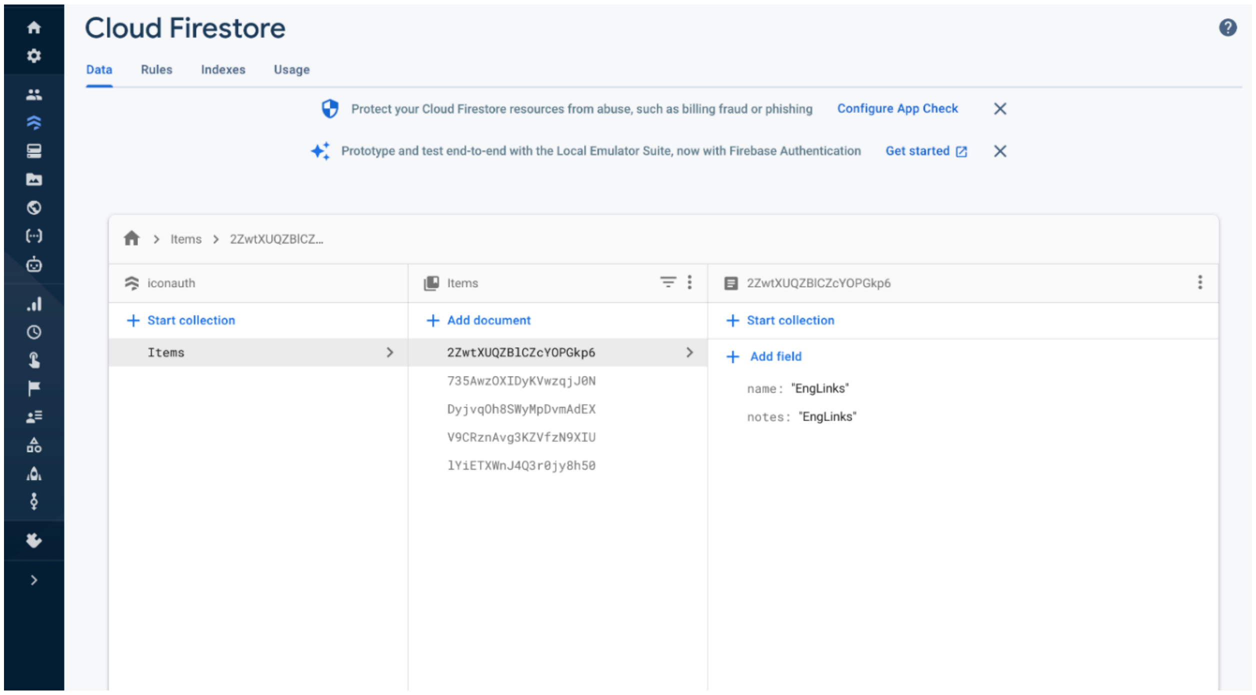
Line: 53 Col: 11

Figure 14: Equipment List file



```swift
func addData(name: String, notes: String) {

    let db = Firestore.firestore()

    db.collection("Items").addDocument(data: ["name":name, "notes":notes]) { error in

        if error == nil {
            //no errors


            self.getData()
        }
        else{
            //handle the error

        }
    }
}


func getData() {

    // get refernce to the database
    let db = Firestore.firestore()


    // Read the document at a path
    db.collection("Items").getDocuments { snapshot, error in

        if error == nil {
         //no errors

            if let snapshot = snapshot {

                //update the list property in main thread
                DispatchQueue.main.async{

                // Get all the documents and creat Items
                self.list = snapshot.documents.map { d in

                    return Item(id: d.documentID,
                                name: d["name"] as? String ?? "",
                                notes: d["notes"] as? String ?? "")
```

Figure 15: Add data/get data functions



Cloud Firestore

Data Rules Indexes Usage

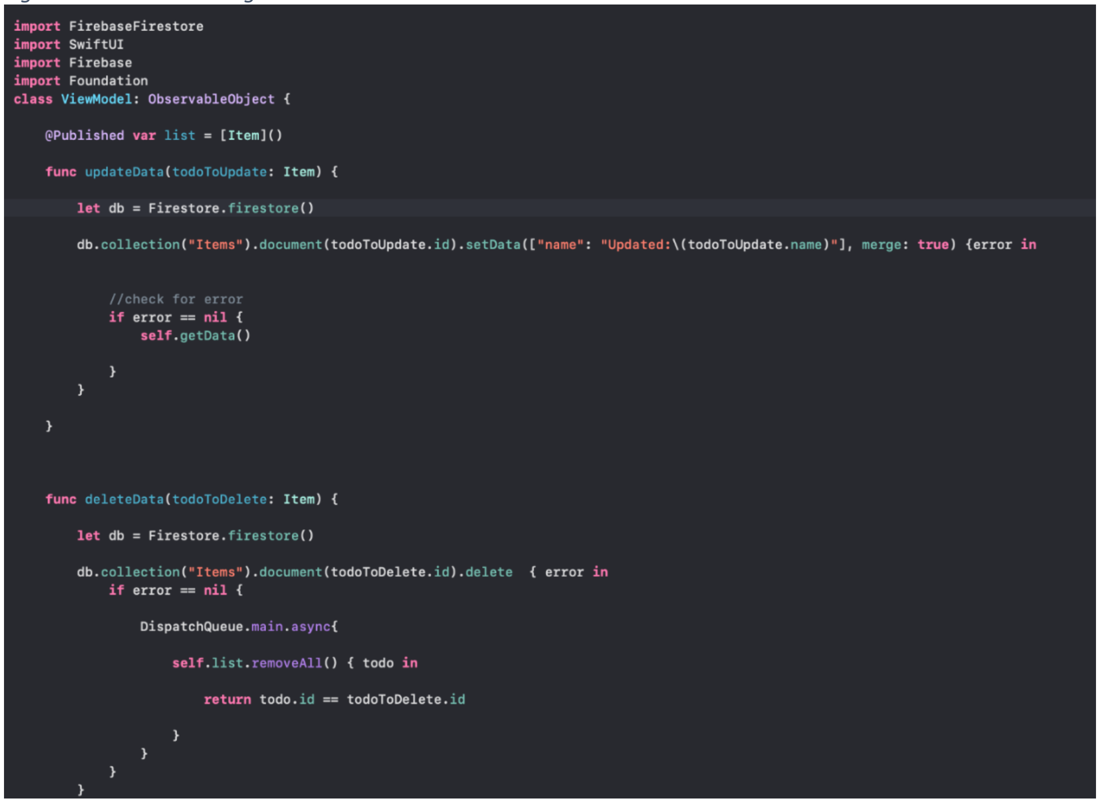
Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing **Configure App Check**

Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication **Get started**

Items > 2ZwtXUQZBlCZ...

iconauth	Items	2ZwtXUQZBlCZcYOPGkp6
+ Start collection	+ Add document	+ Start collection
Items	2ZwtXUQZBlCZcYOPGkp6	+ Add field
	735AwzOXIDyKVwzqjJ0N	name: "EngLinks"
	DyjvqOh8SWyMpDvmAdEX	notes: "EngLinks"
	V9CRznAvg3KZVfzN9XIU	
	1YiETXWnJ4Q3r0jy8h50	

Figure 16: Firebase storage



```swift
import FirebaseFirestore
import SwiftUI
import Firebase
import Foundation
class ViewModel: ObservableObject {

    @Published var list = [Item]()

    func updateData(todoToUpdate: Item) {

        let db = Firestore.firestore()

        db.collection("Items").document(todoToUpdate.id).setData(["name": "Updated:\(todoToUpdate.name)"], merge: true) {error in


            //check for error
            if error == nil {
                self.getData()

            }
        }

    }



    func deleteData(todoToDelete: Item) {

        let db = Firestore.firestore()

        db.collection("Items").document(todoToDelete.id).delete  { error in
            if error == nil {

                DispatchQueue.main.async{

                    self.list.removeAll() { todo in

                        return todo.id == todoToDelete.id

                    }
                }
            }
        }
```

Figure 17: delete data function and List view

Appendix B Evaluation Matrix Criteria

Student accessibility		Capabilities		Prerequisite knowledge	
Descriptor	**Description**	**Descriptor**	**Description**	**Descriptor**	**Description**
5= Accessible by all	Rental service can be used by all students	5= Able to implement all functionality's	This Model can achieve all requirements in the project scope	5= No prerequisite knowledge required	Requires no previous programming experience
4= Accessible by 50% +	Rental service can be used by more than 50% of students	4= Almost all functionality's can be achieved	This model is sufficient to solve the problem	4= Minimal prerequisite knowledge required	Requires little programming knowledge
3= Accessible by 50% of students	The model requires additional assistance to be used	3= Moderate functionality	This model does not solve the problem	3= Moderate prerequisite knowledge required	Requires intro-level programming knowledge
2= Accessible by few	The model can only be used by 50 students	2= Minimal functionality	This model covers only 1 requirement	2= Significant prerequisite knowledge required	This model requires 3+ years of programming experience
1= No accessibility	No students can use the model	1= No functionality	This model is not functional	1= Requires expert level knowledge	This model cannot be created by the team

Cost		User friendly		Update flexibility	
Descriptor	**Description**	**Descriptor**	**Description**	**Descriptor**	**Description**
5= Very Economic. Cost less than $75	Model is extremely cost-efficient	5= Extremely user friendly	Students have perfect experience using the app	5= Flawless update capabilities	The app can be updated by anyone with ease
4= Somewhat economic. Cost is roughly $100	The model falls under financial requirements	4= User friendly	Students have good experience using the app	4= Good Update capabilities	The app can be updated by someone familiar with the project
3= Moderate. Cost is $150 +	The model is a budget	3= Somewhat user friendly	Students have a moderate experience using the app	3= Capable of some updates	The app can be updated with some research
2= Costly. More than $200	The model is slightly above budget	2= functional not user friendly	Students face troubles using the app	2= Few updates achievable	App The applies major changes to update
1= Expensive. Cost is greater than $250	The model is not economically feasible	1= Not user friendly	App is not functional	1= Cannot update	New features cannot be added to the app

Appendix C Project Plan and additional information

My personal experience with the iCons rental service

Casper Harris 20238963

Date: 04/02/2022

Today at the group meeting in the ILC, I had an issue using the iCon service. The meeting took place in room 221 from 3:00pm to 5:30pm, but the booking time was slightly different. In the meeting I wanted to use the whiteboard, and no one had brought a whiteboard marker. I went to the iCons rental cabinet and was greeted by one of the iCons. I asked for a marker and was given two in case one was not in working condition. I had to exchange the markers with my student ID card to ensure I would bring them back. I stayed a little after we lost the room booking at 4:30pm and left around 5:30. On my way out, I stopped at the iCon cabinet to return the markers, but the room that I had originally found had been walled off and locked. I will now need to go back tomorrow to get my student card. This is a clear issue that a mobile app can fix.

Work Breakdown Structure

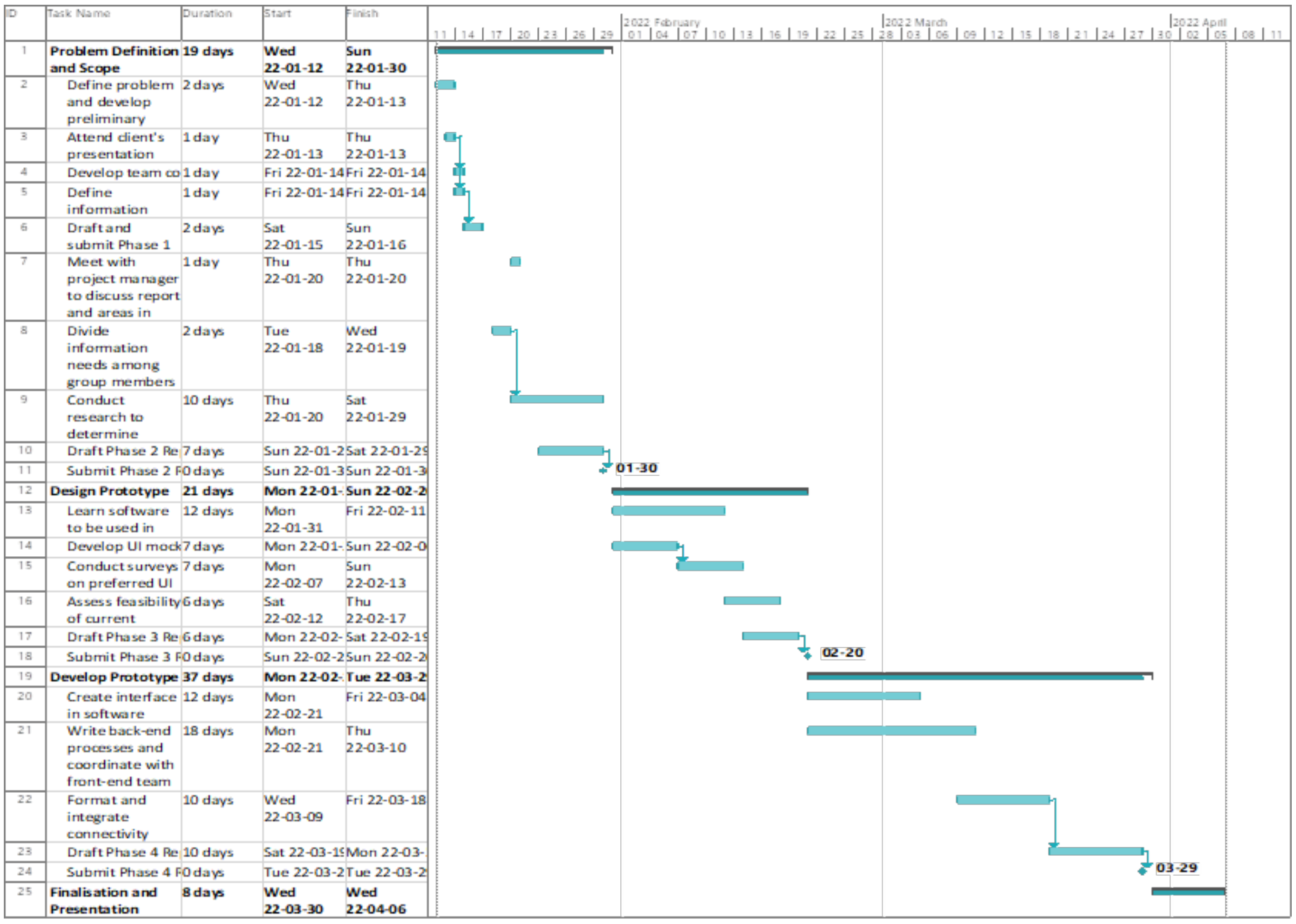
Task Number	Task Description	Expected Duration (days)	Task Leader
1	**Problem Definition and Scope**	**20**	
1.1	Define problem and develop preliminary ideas	2	Ethan
1.2	Attend client’s presentation	1	All
1.3	Develop team contract	1	Casper
1.4	Define information needs	1	Bradley
1.5	Draft and submit Phase 1 Report	2	All
1.6	Meet with project manager to discuss report and areas in which to improve	1	Bradley
1.7	Divide information needs among group members for individual research	2	Bradley
1.8	Conduct research to determine feasible scope	10	All
1.9	Draft and submit Phase 2 Report	8	All
2	**Design Prototype**	**21**	

2.1	Learn the software to be used in development	12	Charles, Ethan
2.2	Develop UI mockups	7	Jesse, Robert
2.3	Conduct surveys on preferred UI	7	Casper
2.4	Assess feasibility of current design	6	Ethan
2.5	Draft Phase 3 Report and Obtain FA Approval	6	All
3	**Develop Prototype**	**35**	
3.1	Create interface in software	11	Jesse, Robert
3.2	Write back-end processes and coordinate with front-end team	14	Bradley, Ethan
3.3	Format and integrate connectivity with email and GPS	10	Charles
3.4	Draft Phase 4 Report	10	All
4	**Finalization and Presentation**	**8**	

Task/ Description of Activity	Activity Duration (for each person) [hours]	Description of activity	Individual(s) Responsible for Activity
Making the app	24 hours	Creating a functioning rental app for icons with all required deliverables. Creating functions for each task, creating user interface for each screen, configuring and setting up database	Bradley Stephen
Section 1.3 design solution	3 hours	Writing about design solution, created all figures in this section	Bradley Stephen
Section 2.3 implementation	3 hours	Writing about implementation, created all figures in this section	Bradley Stephen

Conlusions	2 hours		Bradley Stephen
Conceptual design solutions	2 hours		Bradley Stephen
Create demonstration video	30 min		Bradley Stephen

Gantt Chart



ID	Task Name	Duration	Start	Finish
1	**Problem Definition and Scope**	**19 days**	**Wed 22-01-12**	**Sun 22-01-30**
2	Define problem and develop preliminary	2 days	Wed 22-01-12	Thu 22-01-13
3	Attend client's presentation	1 day	Thu 22-01-13	Thu 22-01-13
4	Develop team co	1 day	Fri 22-01-14	Fri 22-01-14
5	Define information	1 day	Fri 22-01-14	Fri 22-01-14
6	Draft and submit Phase 1	2 days	Sat 22-01-15	Sun 22-01-16
7	Meet with project manager to discuss report and areas in	1 day	Thu 22-01-20	Thu 22-01-20
8	Divide information needs among group members	2 days	Tue 22-01-18	Wed 22-01-19
9	Conduct research to determine	10 days	Thu 22-01-20	Sat 22-01-29
10	Draft Phase 2 Re	7 days	Sun 22-01-2	Sat 22-01-29
11	Submit Phase 2 F	0 days	Sun 22-01-3	Sun 22-01-3
12	**Design Prototype**	**21 days**	**Mon 22-01-**	**Sun 22-02-2**
13	Learn software to be used in	12 days	Mon 22-01-31	Fri 22-02-11
14	Develop UI mock	7 days	Mon 22-01-	Sun 22-02-0
15	Conduct surveys on preferred UI	7 days	Mon 22-02-07	Sun 22-02-13
16	Assess feasibility of current	6 days	Sat 22-02-12	Thu 22-02-17
17	Draft Phase 3 Re	6 days	Mon 22-02-	Sat 22-02-19
18	Submit Phase 3 F	0 days	Sun 22-02-2	Sun 22-02-2
19	**Develop Prototype**	**37 days**	**Mon 22-02-**	**Tue 22-03-2**
20	Create interface in software	12 days	Mon 22-02-21	Fri 22-03-04
21	Write back-end processes and coordinate with front-end team	18 days	Mon 22-02-21	Thu 22-03-10
22	Format and integrate connectivity	10 days	Wed 22-03-09	Fri 22-03-18
23	Draft Phase 4 Re	10 days	Sat 22-03-19	Mon 22-03-
24	Submit Phase 4 F	0 days	Tue 22-03-2	Tue 22-03-2
25	**Finalisation and Presentation**	**8 days**	**Wed 22-03-30**	**Wed 22-04-06**

Grading Rubrics

Faculty Advisor - Grading Rubric

	7-8 (outstanding)	**6 (meets expectation)**	**5 (developing)**	**4 (marginal)**	**0-3 (not demonstrated)**
Preliminary design process (Primarily Sections 1.1-1.2 & 2.1-2.2)	Thoroughly and concisely identifies and describes the presented problem; concise, thorough, relevant information summary; rigorous analysis of options; creative conceptual design proposals described and compared.	Accurately identifies the presented problem; constraints and requirements identified; appropriate information well summarized; design solutions described; supported comparison of options.	Identifies presented problem; constraints are identified but few connections to project are made; information is adequately summarized; some decisions are unsupported	Preliminary design process is missing critical constraints; some irrelevant information used; decisions are unjustified; little of preliminary process can be used to prepare for implementation.	Preliminary design process is not described, or not useful to prepare for implementation.
Project management, managing time and money (Primarily Section 2.4)	Effective project management with potential risks foreseen and mitigated; time and project budget well managed; includes thoughtful comparison with original plan.	Plans and effectively manages time and project budget. Some comparison with original plan.	Plan of time and project budget is present but incomplete or weak; little or no comparison with original plan. Some project management issues	Some time and project budget information provided but poorly presented and incomplete; no comparison with original plan. Important project management issues apparent.	No useful timeline or budget described; poorly managed project; safety
Social, environmental and financial factors (Primarily Sections 2.3 & 2.5)	Social, environmental, and safety considerations considered throughout project; financial factors fully explained with impact on stakeholders. Thorough financial analysis if relevant.	Relevant social, environmental, and financial factors considered throughout design process, including safety.	Some consideration of social, environmental and financial factors; considered in decision making.	Little consideration of social, environmental and financial factors with no clear evidence of impact on decision making.	No consideration of these factors.
Modelling and Analysis (prototype and non-prototype) (Primarily Section 2.3)	Modeling/analysis is thorough, accurate and encompasses/exceeds all required objectives; easily understood and conveys all relevant information; includes sensitivity analysis.	Model/analysis used to convey required information; meets all required objectives; incorporates relevant research.	Model/analysis is present but difficult to understand, Conveys some relevant information but missing key aspects relating to objectives	Model/analysis is poor or incomplete; does not meet all necessary objectives; does not effectively convey relevant information; model does not draw from previous research	No apparent use of modelling or analysis.

	7-8 (outstanding)	**6 (meets expectation)**	**5 (developing)**	**4 (marginal)**	**0-3 (not demonstrated)**
ONLY ONE IMPLEMENTATION RUBRIC LINE IS GRADED: Implementation -prototype, e.g. built prototype or physical object design (Primarily Sections 1.3 & 2.3) OR	Outstanding effort, analysis, and/or construction fully incorporating feedback, showing creativity and incorporating engineering science; prototype was designed for safe construction and use. Includes thorough decommissioning plans if appropriate.	Appropriate analysis, and/or construction demonstrated to implement product, process, or system; design was constructed for safe use. includes some decommissioning plans if appropriate.	Effort put into implementation but has significant room for improvement; minimal proposal feedback incorporated in implementation; design is mostly safe.	Little effort put into implementation; no feedback incorporated; many design aspects overlooked; some key safety factors overlooked.	Insufficient progress in implementation; no safety considerations.
Implementation -non-prototype, e.g. feasibility study, app development, etc. (Primarily Sections 1.3 & 2.3)	Meets expectation and includes several sophisticated effects for clarity or includes features beyond expectation; fully incorporates relevant feedback.	Creates and applies quantitative feasibility study or app using supported analysis, approximations and assumptions are supported with relevant sources; feedback is effectively applied.	Effort put into implementation but has significant room for improvement; minimal proposal feedback incorporated in implementation	Feasibility study or app has significant errors or uses invalid assumptions; does not incorporate feedback.	Insufficient progress in implementation.
Testing/Evaluation (Primarily Section 2.6)	Comprehensive and effective evaluation of design solution against quantifiable functional specifications and other objectives; with well-defended recommendations for improvement.	Compares the design solution against the project objectives and functional specifications; some recommendation for improvement.	Some factors missed in evaluating design solution; does not compare solution to all objectives; some conclusions are drawn from evaluation.	Many factors missed in evaluating design solution; does not mention expected performance of solution; no conclusions are drawn.	No evaluation of design solution.
Overall Impression	Concisely written, professional tone; clear and convincing argument; authoritative; skillful transitions; properly formatted; effective use of figures; very few or no spelling/grammar errors.	Report achieves goal using formal tone; properly formatted; generally concisely written; appropriate use of figures; few spelling/grammar errors.	Some organization problems; minor formatting errors; figures present but not used effectively; informal tone, writing not concise.	Report is mostly unorganized; insufficient thought has gone into formatting; redundancy; spelling/grammar errors; not at all concisely written.	Poorly constructed report; little to no requirements are met.