

Cobb Connect

GCISL



Mentor:

Darcie Bagott
Washington State University

Cobb Connectors

Gabriel Righi
Joshua Frey
Anthony Salvione

CptS 423 Software Design Project II

Spring 2022

II. Team Members & Bios	5
II.1. Project Stakeholders	7
II.2. Use Cases	7
II.3. Functional Requirements	14
II.4. Non-Functional Requirements	17
III. Software Design - From Solution Approach	19
III.1. Architecture Design	19
III.1.1. Overview	19
III.1.2. Subsystem Decomposition	19
Widgets	19
Description	19
Concepts and Algorithms Generated	19
Interface Description	20
View Models	20
Description	20
Concepts and Algorithms Generated	20
Interface Description	20
Controller	21
Description	21
Concepts and Algorithms Generated	21
Interface Description	21
Firebase	22
Description	22
Concepts and Algorithms Generated	22
Interface Description	22
XI.1. Data design	23
XI.2. User Interface Design	24
XII. Test Case Specifications and Results	24
XII.1. Testing Overview	24
XII.2. Environment Requirements	26
XII.3. Test Results	26
XIII. Projects and Tools used	29
XIV. Description of Final Prototype	29

[Registration]	30
Functions and Interfaces Implemented	30
Preliminary Tests	33
[Geolocation]	33
Functions and Interfaces Implemented	33
Preliminary Tests	34
[Profile Page]	35
Functions and Interfaces Implemented	35
Preliminary Tests	35
[Home Page]	35
Functions and Interfaces Implemented	35
Preliminary Tests	36
[Admin Functionality]	37
Functions and Interfaces Implemented	37
XV. Product Delivery Status	40
XVI. Conclusions and Future Work	40
XVI.1. Limitations and Recommendations	40
Navigation	40
Features	40
Hosting	41
XVI.2. Future Work	41
XVII. Acknowledgements	42
XVIII. Glossary	43
XIX. References	44
XX. Appendix A – Team Information	46
XXI. Appendix B - Example Testing Strategy Reporting	47
XXII. Appendix C - Project Management	47

I. Introduction

Describe your project background, motivation, and goals in detail.

Draw from your project description writing assignment and other summaries for this.

- I.1. Include information about your client and mentor
- I.2. Name, company, contact information
- I.3. Include a summary of what the company / client does

The Granger Cobb Institute for Senior Living (GCISL) is a WSU major which focuses on assisted living care for individuals over 55+. This concentration has a rich body of alumni that the department wants to be able to keep in contact with post-graduation. To fulfill this goal, Darcie Bagott, a program coordinator for GCISL, requested a website from the WSU CS Capstone course. This website was meant to serve as a social media site specifically for GCISL alum. This request occurred last year, and the students assigned to the task created a website called “Cobb Connect”. Cobb Connect fit all the website requirements but was not thoroughly tested or moved to WSU servers. Our goal is to finish what the previous Capstone class started, and move Cobb Connect to a WSU server, and provide quality assurance and improvements for the website.

This project is looking to create a platform for GCISL graduates, current GCISL staff, current GCISL students, and students from the School of Hospitality Business Management that are interested in the Senior Living Industry to be able to track and interact with one another. Our client wants this platform to be available as a web application and as a mobile application since many students are likely to use their phone more and it will be more accessible. The goal for our team is to take over this project from a previous team and build upon the foundation laid by the previous team and further enhance the platform. The previous team first developed the web application using Flutter and Firebase to get feedback from the client [3]. After the web application is built to the standard the client wants, the previous team was planning on using Flutter’s multi-platform capabilities to then make it very easy to build a mobile version [4]. So far only the web application has been built and there is still more work to be done on it before starting on the mobile version. Reading the previous team’s documentation and code will be important for our team to continue to add to the project. Finishing up the web application is going to be one of the first things that needs to be done and the main focus before our team can start on the mobile version.

This platform has multiple key components that the previous team has implemented. First the previous team has created different user and roles. As of right now they have graduates and faculty roles. If the user hasn’t created an account, then it will give the user a form to register them to the platform. If the user is a faculty member, then they will be able to access data analytics and have a heatmap where they can view where other users are located. Another component the previous team implemented was the ability for users to message each other. This allows for faculty members to message each other and coordinate better and for graduate students to get advice from fellow graduates or faculty, and a way to keep in touch with each other. The final major component the previous team implemented was a textbox, image upload,

and posting interface. Having the ability for faculty to post will allow everyone to be updated about the program and to coordinate with other members. In the final sprint report provided by the previous team, they have provided our team with future work to be picked up. The future work documented was the ability to have group messaging, user types, admin privileges such as deleting posts, and being able to comment and like posts. Our client has also mentioned the implementation of administrative power or features using the admin's email, having notifications being sent through the application, and adding students as a user to the application. The students that can join this app will be any student in the senior management major or minor or whoever takes the intro course for senior management. The client would also like the status of a student to change to an alumnus once the student graduates.

The initial step for our team that the client wants done is to start migrating the website that the previous team hosted on a google server to a domain from WSU to host on. The client and our team will be coordinating with Tony Burt, Director of IT at the Voiland College of Engineering and Architecture (VCEA) to facilitate this transition. The next step for our team is to send out Google Forms to a select group of individuals identified by our client. These individuals will be asked to provide their valuable feedback and opinions on the current website. Their insights will be instrumental in identifying areas for improvement, potential changes, and additional features that can enhance the platform's usability and effectiveness.

Following the collection of feedback from our selected group, our client and project team will conduct a thorough review of the responses. This review process will help us identify common themes, key suggestions, and areas of priority. From the analysis, we will collaboratively establish a new set of project requirements and objectives for the current year, aligning the project with the evolving needs and expectations of our user community. The end goal of our project is to be able to train the GCISL staff so they can learn how to use the application. The GCISL staff needs to be trained so they can train students on how to use the application.

II. Team Members & Bios

Gabriel Righi is a computer science enthusiast currently immersed in his studies with a keen interest in machine learning, and cyber-security. His past projects have ranged from developing encryption tools and cybersecurity simulations to creating intelligent chatbots. Proficient in Java, Python, SQL, and TensorFlow, Gabriel is well-versed in the intricacies of code and data structures. For this project his role is as a general manager and feature implementer. He will work to condense and make the code more efficient while adding various features.

Joshua Frey is a senior computer science student at Washington State University. He has an interest in full stack development and data science. His prior projects have included developing a basic implementation of a EXT2-Filesystem that is linux compatible and a Yelp-Database GUI application. Joshua's skills include C/C++, Python, PostgreSQL, and Git. For this project, his responsibilities include leading Beta Testing feedback and developing various features of the Cobb-Connect application.

Anthony Salvione is a senior in computer science at Washington State University. He is interested in video game making, back-end work and website creation. Prior projects required Java, C# and Python which gave him some proficiency in those languages. For this project, I help with mostly front-end responsibilities, such as the profile page and front page. Anthony will work to continue adding more front-end features.

II.1. Project Stakeholders

Granger Cobb Institute for Senior Living (GCISL):

☒ **Needs:**

The GCISL wants a platform that facilitates communication and interaction among its alumni, current staff, and students. They seek a web and mobile application that serves as a social hub for the Senior Living Industry. The platform should have user roles such as graduates, faculty, and potentially administrators. GCISL desires features like messaging, data analytics for faculty, posting capabilities, and the ability to differentiate between current students and alumni.

Darcie Bagott (Program Coordinator for GCISL):

- ☒ **Needs:** Darcie Bagott is interested in a platform that allows seamless communication and networking among GCISL alumni. She has requested the website initially and is now looking for improvements, migration to WSU servers, and additional features like administrative controls.

GCISL Alumni:

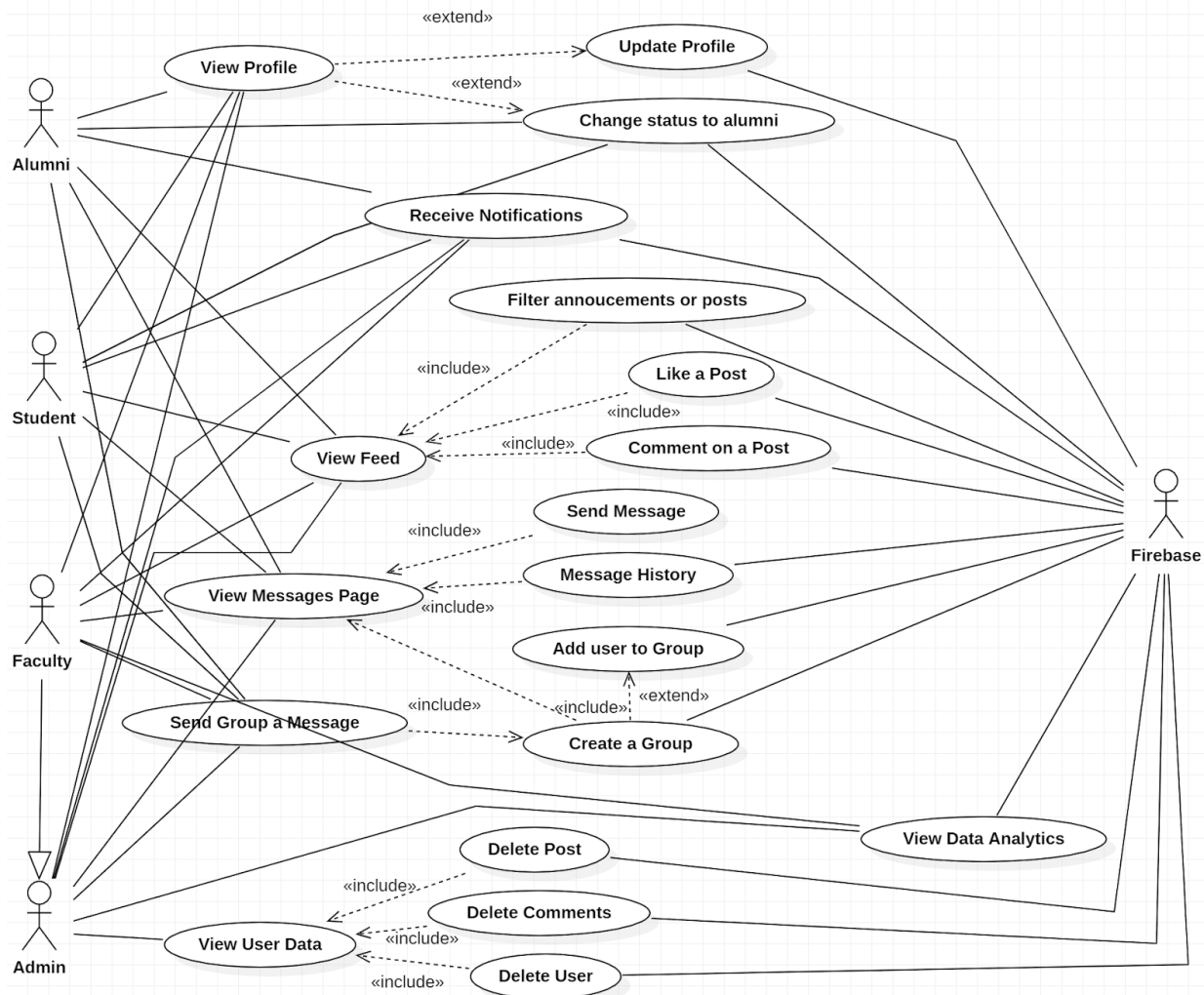
☒ **Needs:**

Alumni are interested in staying connected with each other and being informed about program updates. The platform should provide features for alumni to share experiences, advice, and collaborate with the current GCISL community.

Current GCISL Students:

- ☒ **Needs:** Current students are looking for a platform that connects them with alumni and staff, providing opportunities for advice, mentorship, and staying updated on program-related information. They may also have input on features that enhance their experience.

II.2. Use Cases



View Profile

Actors:	Student, Alumni, Faculty, Admin, Firebase
Goal:	For users to view their profile information for the application.
Preconditions:	<ul style="list-style-type: none"> ⊄ User has an account. ⊄ User is logged in.
Scenarios:	<ul style="list-style-type: none"> ⊄ A way for current faculty members to keep track of their alumni's career status.

Exceptions:	<ul style="list-style-type: none"> None
-------------	--

Edit Profile

Actors:	Student, Alumni, Faculty, Admin, Firebase
Goal:	For users to be able to change their profile information for the application.
Preconditions:	<ul style="list-style-type: none"> User has an account. User is logged in. Users must click on the profiles button to go to profiles page first.
Scenarios:	<ul style="list-style-type: none"> A way for an alumni to update their career status so faculty can keep track of their progress in their career.
Exceptions:	<ul style="list-style-type: none"> None

Change Status to Alumni

Actors:	Student, Alumni, Firebase
Goal:	If a student graduates, then they can change their status from student to alumni.
Preconditions:	<ul style="list-style-type: none"> User has an account. User is logged in. User is a student at first in order to change status. User must navigate to the profiles page first.

Scenarios:	<ul style="list-style-type: none"> ⊄ A way for students who graduated to change their status to an alumni so faculty members can then keep track of their career status and progress.
Exceptions:	<ul style="list-style-type: none"> ⊄ None

Receive a Notification

Actors:	Student, Alumni, Faculty, Admin, Firebase
Goal:	A User will be notified by a red indicator in the messages icon to show how many new messages they received.
Preconditions:	<ul style="list-style-type: none"> ⊄ User must be logged in. ⊄ User must have an account.
Scenarios:	<ul style="list-style-type: none"> ⊄ A way for every user to be up to date with their messages so communication between each other is efficient.
Exceptions:	<ul style="list-style-type: none"> · None

View Feed

Actors:	Student, Alumni, Faculty, Admin, Firebase
Goal:	Users can view the feed for posts.
Preconditions:	<ul style="list-style-type: none"> ⊄ User must have an account. ⊄ User must be logged in.
Scenarios:	<ul style="list-style-type: none"> ⊄ Allows graduates to be up to date with the program when faculty members post.

Exceptions:	<ul style="list-style-type: none"> None
-------------	--

Filter Announcements or Posts

Actors:	Student, Alumni, Faculty, Admin, Firebase
Goal:	Users can filter to either only see posts or only see announcements in their feed.
Preconditions:	<ul style="list-style-type: none"> User must have an account. User must be logged in. User must be in the home page.
Scenarios:	<ul style="list-style-type: none"> Allows for users to be able to filter between seeing only posts or only announcements.
Exceptions:	<ul style="list-style-type: none"> There are no posts or announcements, so nothing was filtered.

Liking a Post/Commenting on a Post

Actors:	Student, Alumni, Faculty, Admin, Firebase
Goal:	Users can like or comment under a post or announcement.
Preconditions:	<ul style="list-style-type: none"> User must have an account. User must be logged in. User must be at the home page where the feed is to view posts and announcements.

Scenarios:	<ul style="list-style-type: none"> When faculty members post updates about the program the faculty member can see what graduates are keeping up to date as well when they like or comment.
Exceptions:	<ul style="list-style-type: none"> None

View Messages Page

Actors:	Student, Alumni, Faculty, Admin
Goal:	User can view messages on the messages page.
Preconditions:	<ul style="list-style-type: none"> User has an account. User is logged in. User clicks on the messages icon to go to messages page.
Scenarios:	<ul style="list-style-type: none"> Everyone can see there messages to be up to date with one another.
Exceptions:	<ul style="list-style-type: none"> User sees no messages if no message has been sent to the user or the user hasn't sent out a message yet.

Send Message

Actors:	Student, Alumni, Faculty, Admin, Firebase
Goal:	Users can send a message to another user privately.
Preconditions:	<ul style="list-style-type: none"> User has an account. The other user has an account. User is logged in. User is in the messages page.

Scenarios:	<ul style="list-style-type: none"> ⊄ Allows faculty members to contact alumni to track their progress and receive feedback from them. ⊄ Allows for faculty members to better coordinate with each other for program events.
Exceptions:	⊄ None

View Data Analytics

Actors:	Faculty, Admin, Firebase
Goal:	A faculty member can view data analytics.
Preconditions:	<ul style="list-style-type: none"> ⊄ Must be a faculty user. ⊄ The faculty member is logged in. ⊄ The faculty member is in the analytics page.
Scenarios:	⊄ Faculty members can view a summary of the graduates data so they can make better decisions for their program.
Exceptions:	⊄ None

Admin Privileges (view user data, delete posts, comments, users)

Actors:	Admin, Firebase
Goal:	An admin can have administrative power of viewing or overseeing other user's posts, announcements, comments. Admin can delete users, comments, and posts.

Preconditions:	<ul style="list-style-type: none"> ⊄ Must be an admin. ⊄ Admin must be logged in.
Scenarios:	<ul style="list-style-type: none"> ⊄ If a faculty member isn't working at the program anymore the admin can delete them as a user. ⊄ Any inappropriate post or comment can be deleted.
Exceptions:	<ul style="list-style-type: none"> ⊄ None

II.3. Functional Requirements

II.2.1 Firebase Backend

i) Firebase backend

a) Unread notifications

Display to the user how many messages were sent to their account that are currently unread. This will be tracked via the firebase backend. The way it will be displayed is so that the number of new notifications will be in the same area as the app bar so it indicates how many are left.

b) Add new account to database when created

Store the information in firebase so that they will always have information saved.

A dictionary or stack will contain all of the user's information.

c) Messaging page and data

Create a clean and functional UI to hold and display all viewable messages for the user and store them in firebase so they may be viewed again later.

d) Saving feed data

Firebase will save all post data and display them when the user accesses them. The post data will include time of post, user associated with it and its contents.

i) Firebase backend

Priority Level 2

Source: Darcie and WSU Students

a) Unread notifications

Priority Level 1

Source: Darcie and WSU Students

b) Add new account to database when created

Priority Level 2

Source: Darcie and WSU Students

c) Messaging page and data

Priority Level 2

Source: Darcie and WSU Students and Administrators of the website

d) Saving feed data

Source: Darcie and WSU Students
Priority Level 1

II.2.2 Interface and Navigation

ii) Interface and navigation

a) Creating an interface to display the data

Present information in an accessible and clear way

b) Auto filling information onto the profile page

When the user is logged in and they already have an account with information connected to it, their profile page will be filled with their proper information.

c) App bar

A bar at the top of each page which allows the user to easily navigate the website

d) Settings page

Create a clean and functional UI to hold and display all of the user settings

e) Messaging page and data

Create a clean and functional UI to hold and display all viewable messages for the user and store them in firebase so they may be viewed again later.

ii) Interface and navigation

Source: Darcie and WSU Students and Capstone Counselor

Priority Level 3

a) Creating an interface to display the data

Priority Level 3

Source: Darcie and WSU Students

b) Auto filling information onto the profile page

Priority Level 3

Source: Darcie and WSU Students

c) App bar

Priority Level 3

Source: Darcie and WSU Students

d) Settings page

Priority Level 2

Source: Darcie and WSU Students

e) Messaging page and data

Priority Level 3

Source: Darcie and WSU Students

II.2.3 Main Functionality

iii) Main functionality

a) Feed page and ability to edit feed

Functionality to sort which updates and messages you want to view and a clean and functional UI for the user to navigate their updates and feed.

b) Creating a post

Certain users who have the access to post updates onto the feed, which can be viewed by all users

iii) Main functionality

Priority Level 2

Source: Darcie and WSU Students

a) Feed page and ability to edit feed

Priority Level 3

Source: Darcie and WSU Students

b) Creating a post

Priority Level 3

Source: Darcie and WSU Students

II.2.4 Network and Security

iv) Network and Security

a) Restrict access unless logged in

Some things will not be able to be viewed unless the user has access (logged in or not)

b) Hosting the website

WSU servers will host this website

c) Password hashing and encrypting

Users have their data secured (most likely provided by either firebase or WSU servers)

iv) Network and Security

a) Restrict access unless logged in

Priority Level 2

Source: Darcie and WSU Students

b) Hosting the website

Priority Level 2

Source: Darcie and WSU Students

c) Password hashing and encrypting

Priority Level 2

Source: Darcie and WSU Students

ii) Interface and navigation

a) Creating an interface to display the data

Present information in an accessible and clear way. The correct information is shown when presented with a filtered search and/or permissions are granted to see the given posts.

b) Auto filling information onto the profile page

When the user is logged in and they already have an account with information connected to it, their profile page will be filled with their proper information.

c) App bar

A bar at the top of each page which allows the user to easily navigate the website. Every tab of the website will be included in this bar. The bar will add or remove different tabs depending on the access of the user.

d) Settings page

Create a clean and functional UI to hold and display all of the user settings.

e) Messaging page and data

Create a clean and functional UI to hold and display all viewable messages for the user and store them in firebase so they may be viewed again later. Prioritize a quick way to store and sort data for these messages and data.

iii) Main functionality

a) Feed page and ability to edit feed

Functionality to sort which updates and messages you want to view and a clean and functional UI for the user to navigate their updates and feed. Different sorts of feed posts may include: Recent posts, posts flagged as football related, posts from one specific user, etc.

b) Creating a post

Certain users who have the access to post updates onto the feed, which can be viewed by all users. The users that can post include staff, administrators, school alumni and others.

iv) Network and Security

a) Restrict access unless logged in

Some things will not be able to be viewed unless the user has access (logged in or not). This is important for the website to maintain its user integrity and requires that the user creates an account if most features are to be accessed.

b) Hosting the website

WSU servers will host this website

c) Password hashing and encrypting

Users have their data secured (most likely provided by either firebase or WSU servers). Firebase contains a feature and documentation to safely secure user passwords.

II.4. Non-Functional Requirements

The app overall shall be developed using Flutter as the website frontend, and firebase as a website backend. These choices were made by the previous developers, and we will continue in the same manner. To actually implement new features suggested by the Beta Testers, we will be using the agile software methods with a series of sprints. For each sprint we will outline a series of tasks that we want to complete, and we will get them done by the end of the sprint.

Hosting

The website shall be hosted on WSU servers. Having a locally hosted website allows for quicker error diagnosis, faster service, and easier maintenance overall.

Usability

An average non-technically inclined user should have no trouble navigating the website. All UI elements should be clearly defined, and adhere to common standards for icons and usage. The website should be intuitive and easy to access for all users.

Reliability (Robustness)

The website should never go offline due to software malfunction, except in the case of server failure. The code of the website should be written robustly enough so that no individual user can crash the website through misuse.

Reliability (Scalability)

The website should be able to sustain 100 concurrent users and store the data for 10x its concurrent limit. Its storage capability should scale with the user count, and it should not need any input to increase storage space.

Performance

There should be no noticeable lag between user actions responses on the webpage. Actions that interact with the database should appear within a second of the submission or changing of any data.

Supportability

There should be at least 1 admin account that can remove user content and users in general. This admin account should be passed over to GCISL and they will moderate their app as they see fit.

Packaging

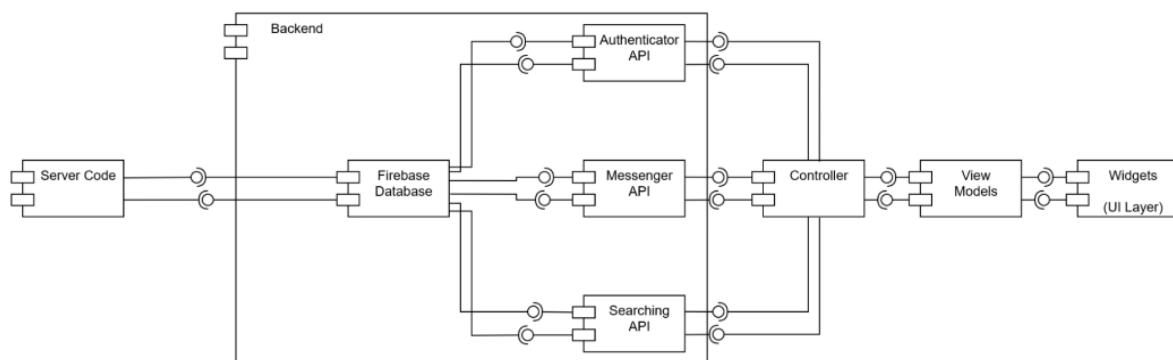
The entire app should be delivered in a docker container. This container should start the website upon instantiation, and should work on any docker system. The system itself can be installed on as many servers as necessary as it shares a cloud database.

III. Software Design - From Solution Approach

III.1. Architecture Design

III.1.1. Overview

As a group we are coming in to work on a project that has already been designed. While we will most definitely make improvements to the project overall, we have no plans to make any fundamental changes to the architecture of the system. The previous cobb connect group documented their architecture system, the same one we will be using, in their Solution Approach. If you want that information, you should go view it in the github, as we are not designing a system for our project. Since we are not actually designing a system but upgrading and maintaining one, it would not make sense for us to speak about the architecture design, except in reference to what we have changed. Parts of our description might seem similar to the prior groups, for instance we will have very similar (if not identical) subsystem decomposition [8]. This is because the prior group defined what they think the subsystems are, and we will just be stating what we will be changing in relation to those subsystems. Since this project is one continuous work it reasons that we should follow in the style of the previous documentation and try to keep as much similarity as possible, while not rehashing or copying their work.



(Since a diagram was required, this is the previous groups diagram which we aren't changing)

III.1.2. Subsystem Decomposition

Widgets

Description

The widgets represent the front end of our project. This encompasses the layout of pages, style, and overall theme of the website.

Concepts and Algorithms Generated

Dart uses forms as its method to generate css,html and javascript code. We will be using these forms to describe the layout of pages we create. These forms will handle all user interaction from button presses to keyboard input.

Interface Description

Provide a description of the subsystem interface. Explain the provided services in detail and give the names of the required services.

Services Provided:

IV. Service name: Admin Page

Service provided to: Admin User

Description: This page will display detailed information about all users on the system

V. Service name: Profile Page

Service provided to: User

Description: This page will display a user's information in a profile style

Services Required:

1. Service Name: View Model

Service Provided From: View subsystem

Needs updated information from the view model to display the most accurate data.

View Models

Description

The view model describes the model of how data will be gathered and stored in our project. When someone enters information about themselves, or makes a post about something, a model defines how the data they provide will be grouped in code.

Concepts and Algorithms Generated

The algorithm behind this lies in how we store the data in the firebase database. We will group relevant pieces of information together with unique identifiers to reference at later points.

Interface Description

Services Provided:

VI. Service name: Student Model

Service provided to: Widgets

Description: This model will help define one of the three kinds of users. Students will have reduced permissions as they are still in school so they will not need to provide certain pieces of information such as their location and job information.

VII. Service name: Alumni Model

Service provided to: User

Description: This page will help define one of three kinds of users. Alumni will be required to provide information about their current company, and location.

VIII. Service name: Faculty Model

Service provided to: User

Description: This page will help define one of three kinds of users. Faculty, or admins, will be able to control system wide information through their accounts. They will have special permissions that allow them to delete users, their posts, and other elevated actions.

Services Required:

Service Name: Controller

Service Provided From: the controller subsystem

Controller

Description

The controller defines the actual routes the user goes through to complete tasks. This will generally be in the form of web requests. The requests will generally be calls to various firebase applications to update the database

Concepts and Algorithms Generated

The algorithm behind this lies in how we store the data in the firebase database. We will group relevant pieces of information together with unique identifiers to reference at later points.

Interface Description

Services Provided:

- a. Service name: Delete (DELETE)

Service provided to: View Model, Backend - Authenticator

Description: This command will delete a post from the firebase, and remove it from the page.

b. Service name: Get Lat/Long (GET)

Service provided to: View Model, Backend - Authenticator

Description: This command gets the geolocation latitude and longitude based on the information provided to the webpage.

c. Service name: Retrieve Users (GET)

Service provided to: View Model, Backend - Authenticator

Description: This command gets all the users' information from the database.

d. Service name: Delete User (DELETE)

Service provided to: View Model, Backend - Authenticator

Description: This command removes a user from the database.

Services Required

Service Name: Firebase

Service Provided From: Datastorage

Firestore

Description

All information from the program will be stored on the firestore database.

Concepts and Algorithms Generated

Through web requests we will be sending and receiving information from the database

Interface Description.

Services Provided:

IX. Service name: Users Table

Service provided to: Controller Subsystem

Description: This database table contains all of the users information. This is notably different from the users login information, its just their account information such as firstname lastname location etc...

X. Service name: Posts Table

Service provided to: Controller Subsystem

Description: This database table contains all the information about every post made by users. It stores text, images, and user

XI. Service name: Messages Table

Service provided to: Controller Subsystem

Description: This database table contains all messages sent by users.

Services Required:

1. Service Name: Controller

Service Provided From: Controller subsystem

2. Service Name: Model

Service Provided From: Model subsystem

XI.1. Data design

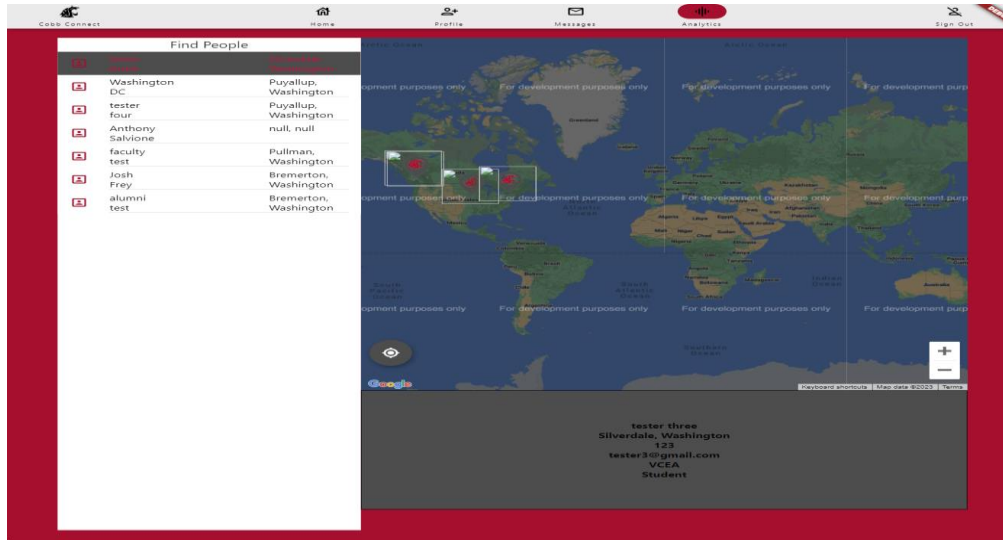
There will be three primary data groups recorded using Google Firebase: user profiles, user posts and user messaging. Through these three sets the website can store all of the appropriate data needed to present the analytics page [9]. All these data sets will be internal data structures saved permanently unless manual removal is done.

The user profile contains the bulk of the information on the user. The information included is the following: city, state and country location, company the user works for, time of creation, experience, first name, last name, usertype, latitude and longitude coordinates, phone number, company position and zip code. Depending on privacy settings and preferences some of the information will not be displayed for the user requesting it. There also should be a dataset that holds the users preferences. If they want to show something or hide something from their profile it should save to the database.

The message and post datasets require that the user has an account, so some of the information in these sets are dependent on information in the user profile. Posts are viewable to the public and contain three pieces of information: the post contents, name of the user who posted (if no name inputted yet, it will remain anonymous), and the time of creation. Lastly, the private messaging data set. Each user will have its own branch, containing all received and sent messages. Each message contains the message contents, user identification number of sender (obtained through a hash function), and the time of creation.

XI.2. User Interface Design

The primary feature of this website is the analytics map which displays all of the users and their locations using Google's API. This is where the primary interface design changes will take place. The current analytics interface design will be pasted below:



The prioritization of the interface should be in consideration of the clients. The information should be easily accessible, displayed in a neat and satisfying way and provide more specific ways to sort through information. A pulldown tab should be implemented allowing the user to filter information (such as by location, by company etc.). The information below should be displayed in a neater way [10].

The homepage of this project contains a text box to fill to post, and a list of posts underneath. We have added a filter method so you can sort posts by various factors. Tags should be added to posts as a way to sort posts easier and allow the poster to target their audience easier.

The user is expected to use the homepage, messaging and analytics page the most and this is where the user interface design efforts should be focused.

The messaging page contains two parts. The message window and the user list. The user list contains

XII. Test Case Specifications and Results

XII.1. Testing Overview

Our plan for testing is as follows. We will test the functionality that we add to the website. As we have previously mentioned, a prior group created the main body of the website. Our capstone

group is coming in and improving on that design and we are not changing any core functionality. To that end, we will not be testing core functionality as we assume the prior group has done appropriate testing in that realm.

Unit testing

For unit testing we will be using the dart test library. This allows us to have a specific setup function and run tests without having to deploy the actual website. We plan to have unit tests for the features we have integrated, such as post sorting, geolocation and our error messages. Having unit tests for our geolocation and such will ensure that when we change other parts of our code, the old parts of our code still function properly. Currently we have successful unit tests for multiple functions we have implemented. As we implement new functionality, we will need to add more tests to cover.

Integration testing

Due to how flutter works, it is difficult to test aspects of a website with code. To that end we will be manually testing the integration of each of our features. While we can test that our post sorting returns a list of properly sorted posts, we can manually check that these posts are visually properly sorted and everything is as we expect. We will do manual integration testing for all the features we add. We are up to date on integration testing. After each new feature we create, we extensively manually test that feature to ensure functionality. Our future work is continuing this process as we add more functionality.

System testing

We have beta testers working on our website. During our second sprint, we handed out a version of our website to 14 beta testers, and got their feedback. They act as a form of quality assurance for our product as they will be able to catch any bugs in our product. They can also give us valuable feedback for how to improve the product. We will have beta testers test again at a later point during our development process.

Functional testing

Our functional requirements are also tested by our beta testers. We have a list of features and pages they are supposed to check out and give feedback on. As many of our functional requirements require actual usage of the website, it can be difficult to test in an isolated environment. To this end we have manual testers, instead of repeatable code tests. Many of the dependencies required for our widgets in dart and the prior group's design decisions make code tests for this difficult. Manual testing will be required for more functionality in the future, but currently we meet all the functionality requirements and have manually tested them all.

Performance testing

Performance testing is difficult to test programmatically, so we will just be testing it via our own usage and beta testers. We also will get statistics from the CCB and make sure that the server they are hosting the website on is powerful enough to serve static webpages to a large amount of people.

User acceptance testing

Like our prior tests, this will be done by the beta testers. We plan to have another round of beta testers once we get the website hosted. These beta testers will be able to provide more feedback and ensure that all components of the website are functional before we release it to the public.

XII.2. Environment Requirements

Nothing special is needed for our testing environment. All of our unit tests test specific functions that are isolated from the rest of the code. Since we designed our functions in this way it makes them easily testable without having to invoke any other part of the program. For a couple of our tests an internet connection is required as they test features from an API.

XII.3. Test Results

Aspect being tested: **queryGeoLocation function in GeoLocationService**

Test Case 1:

- ⌘ What is being tested/how: The function should return coordinates for a valid address in the United States.
- ⌘ Expected Results: The expected latitude is 42.879179, and the expected longitude is -75.246253.
- ⌘ Observed Result: 42.879179, -75.246253
- ⌘ Test Result: Pass

Test Case 2:

- ⌘ What is being tested/how: The function should return coordinates for a valid address in Canada.
- ⌘ Expected Results: The expected latitude is 38.511316, and the expected longitude is -122.460908.
- ⌘ Observed Result: 38.511316, -122.460908
- ⌘ Test Result: Pass

Test Case Requirements:

- ⌘ The test assumes a functional GeoLocationService with the queryGeoLocation method.
- ⌘ The test environment should have network access to make real requests.
- ⌘ Valid address parameters are provided for each test case (country, city, state, zipCode).
- ⌘ The expected latitude and longitude values for each test case are known and correct.

Note:

- ⌘ Ensure that the actual and expected values are updated and accurate according to the real-world data.
- ⌘ The success of the test depends on the accuracy of the expected values and the correctness of the GeoLocationService implementation.

Aspect being tested: sortPostList function in PostSorting

Test Case:

- ⌘ **What is being tested/how:** The sortPostList function is being tested for its ability to correctly sort a list of posts based on different criteria.
 - ⌘ Test sorting by newest: The function should correctly sort the list by the newest posts based on the date.
 - ⌘ Test sorting by oldest: The function should correctly sort the list by the oldest posts based on the date.
 - ⌘ Test sorting alphabetically: The function should correctly sort the list alphabetically based on user names.
 - ⌘ Test sorting by likes: The function should correctly sort the list by the number of likes, with posts having more likes appearing first.
- ⌘ **Expected Results:**
 - ⌘ For sorting by newest, the post with the latest date ('2023-01-04') should be the first in the list.
 - ⌘ For sorting by oldest, the post with the earliest date ('2023-01-01') should be the first in the list.
 - ⌘ For sorting alphabetically, the post with the user name 'User1' should be the first in the list.
 - ⌘ For sorting by likes, the post with the most likes (User3 with [1,1,1] likes) should be the first in the list.
- ⌘ **Observed Result:**
 - ⌘ All Tests Passed
- ⌘ **Test Result:** All tests Passed

Test Case Requirements:

- ⌘ The PostSorting class with the sortPostList function is correctly implemented.
- ⌘ The PostSortOption enum is correctly defined with options for sorting.
- ⌘ The provided postList has valid post data for testing.
- ⌘ The expected values are accurate and reflect the correct sorting criteria.

Note:

- ⌘ Ensure that the actual and expected values are updated and accurate according to the real-world data.
- ⌘ The success of the test depends on the correctness of the PostSorting implementation and the accuracy of the expected values.

Aspect being tested: errorMessage method in ErrorMessages

Test Case:

- ⌘ **What is being tested/how:** The errorMessage method is being tested for its ability to correctly translate error codes into human-readable error messages.
 - ⌘ Test case 1: Invalid email - The method should return 'Invalid email' for the 'invalid-email' error code.
 - ⌘ Test case 2: Missing password - The method should return 'Missing Password' for the 'missing-password' error code.
 - ⌘ Test case 3: Invalid login credentials - The method should return 'Email or Password is Invalid' for the 'invalid-login-credentials' error code.
- ⌘ **Expected Results:**
 - ⌘ For each test case, the expected error message corresponding to the provided error code.
- ⌘ **Observed Result:**
 - ⌘ Properly returned error messages
- ⌘ **Test Result:** Pass

Test Case Requirements:

- ⌘ The ErrorMessage class is correctly implemented with the errorMessage method.
- ⌘ The provided error instances (error1, error2, error3) have valid error codes for testing.
- ⌘ The expected error messages are accurate and reflect the correct mapping from error codes to messages.

Note:

- ⌘ Ensure that the actual and expected values are updated and accurate according to the real-world data.
- ⌘ The success of the test depends on the correctness of the ErrorMessage implementation and the accuracy of the expected values.

XIII. Projects and Tools used

Flutter	Multiplatform web development
Firebase	Online database
Geocode	Online latitude longitude api

As a quick survey, let me know what languages you wrote some of the project in. This is anything you wrote yourself, not just used in libraries. This includes both programming languages and markup languages.

Languages Used in Project			
Dart			

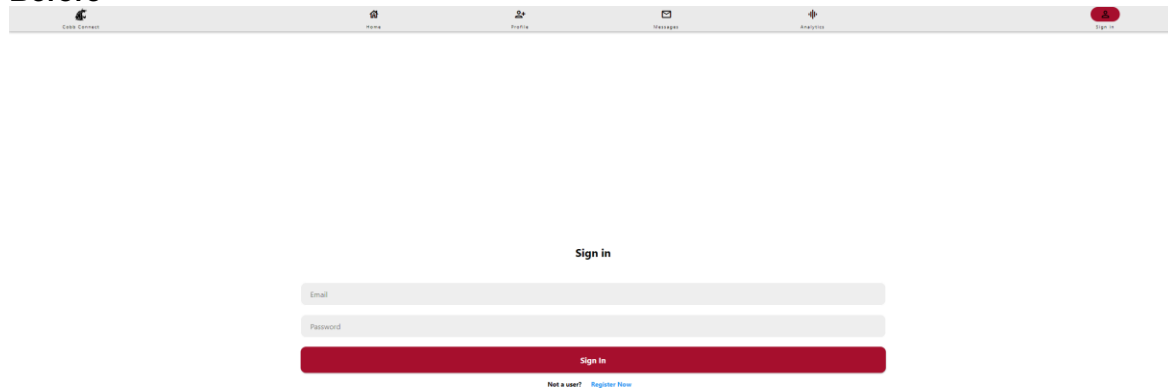
XIV. Description of Final Prototype

Since we are creating upgrades to an already existing website, our final prototype will be more about the changes we made, rather than the website itself. The prior group already described the bulk of the website, so we will not be focusing on that.

[Login Page]

For the login page, we have removed the app bar at the top, and added a WSU logo. The app bar was non functional on this page so it served no purpose to keep it there. We also received user feedback saying that the website should be more clearly WSU branded, so we added a large logo.

Before



Sign in

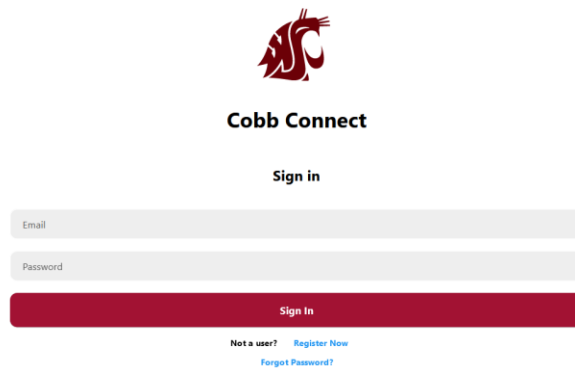
Email


Password

Sign In

Not a user? [Register Now](#)

After





Cobb Connect

Sign in

Email

Password

Sign In

Not a user? [Register Now](#)

[Forgot Password?](#)

[Registration]

Functions and Interfaces Implemented

One of the first changes we made was to registration. Firebase stores users and their data separately, so previously the way to data stored was as followed. A user would make an account, and that login information would be stored in the authentication section of firebase. Then when the user felt like it, they could add information to their profile page. This is obviously not ideal as we would want a user to add info to their profile page as they made their account. To this end we changed how registration functioned. Instead of just having a user input an

account, and then have the option to edit their profile page, we combined the register page and the edit profile page. A user can still edit their profile once they make their account, but they will have to enter preliminary account info. The information is still stored separately in firebase, but this is largely irrelevant as we only need to ensure that both authentication and user info data exists. This prevents anonymous users, and people forgetting to input their information.

Aside from changing how registration worked, we also added a tab that describes the type of user. Previously all users were considered to be alumni, but our client mentioned the possibility of having students on the site. To this end we added user types, student alum and administrator. This can be seen at the bottom of the new registration page.

Another issue that users might be having was the ability to change their password after they have registered. To fix this we have added a “forgot password” button that allows a user to reset their password.

Before

Welcome to Cobb Connect!

Register

Already a User? [Login](#)

After

Welcome to Cobb Connect!

Email

Password

Re-enter Password

First Name

Last Name

Phone Number

United States

State

City

Zip Code

Company

Position

student

Reset Password

Reset Password


Reset Password

[Return to Login](#)

Preliminary Tests

The testing we have done for this is purely functional. We have tested adding a user and ensuring all their information is still present in the database. When we add a user and query the database we can see the following

New User

email	password	created	updated	token
francis@gmail.com		Nov 15, 2023	Nov 15, 2023	wEsbKPC2bQTtrF5RhSr0PS8Alr1

User Info

419815932
city address: "Sammamish"
company: "Google"
country address: "United States"
date added: 1700116399264
email: "Francis@gmail.com"
experience: "1"
first name: "Francis"
isAdmin: false
last name: "Smith"
lat: 47.606101
long: -122.042934
phone: "1231231234"
position: "CEO"
state address: "Washington"
userType: "alumni"

You can see that the new user has been successfully added and that their user type is also present.

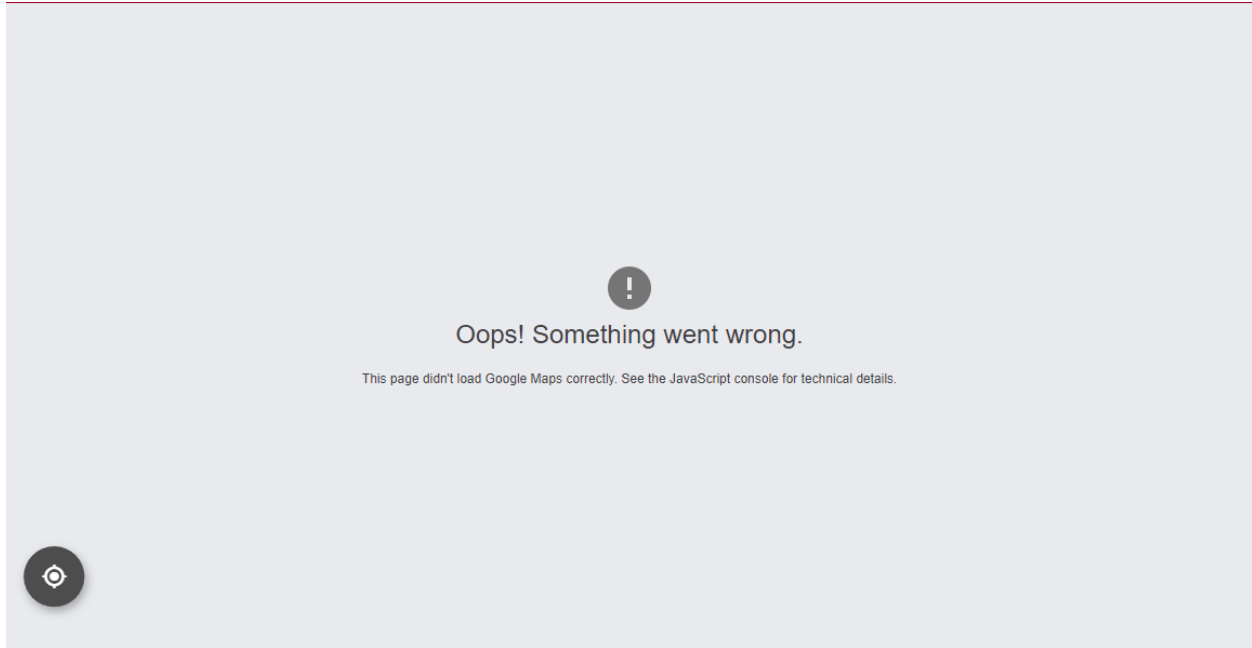
[Geolocation]

Functions and Interfaces Implemented

The previous group that implemented geolocation used a website called geocode.xyz. While this might have worked for them at the time, during testing we found this website to be inaccurate. It was putting pins for people who set their location to pullman, in washington DC. This is obviously unusable as the entire point of the analytics page is to find users that are close to each other. To fix this we have changed the service we use to query addresses. Instead of using geocode.xyz we use a website called Radar.com. This website has proved more accurate and has been accurate throughout our testing.

More importantly we also found that the google maps API key for the previous website has expired. To fix this we have acquired a new google API key and attached a WSU credit card to it. We have also remade the page to be prettier.

Previous Group Maps API:



Our Fixed API:

Find People

Address	City
tester NumberThree	Seattle, Washington
surfer dude	Waianae, Hawaii
Test Tester	Silverdale, Washington
Gabe Gaberson	Sammamish, Washington
student one	Bremerton, Washington
Butch T. Cougar	Pullman, Washington
Gabe Righi	Pullman, Washington
tester four	Silverdale, Washington
Spicy Italian Sausage	Sammamish, Washington
Anthony Salvione	Pullman, Washington
Francis Smith	Sammamish, Washington
Admin Admin	Pullman, Washington

tester five
Silverdale, Washington
1234567890
test5@gmail.com
Twitch
PM

Preliminary Tests

To test this we have written a unit test that queries our “getGeolocation” function that returns a latitude and longitude. We have confirmed that this returns accurate results for the locations we have chosen .

```
00:00 +0: queryGeolocation should return coordinates for valid address parameters
00:00 +1: queryGeolocation should throw an exception for invalid address parameters
00:00 +2: All tests passed!
```

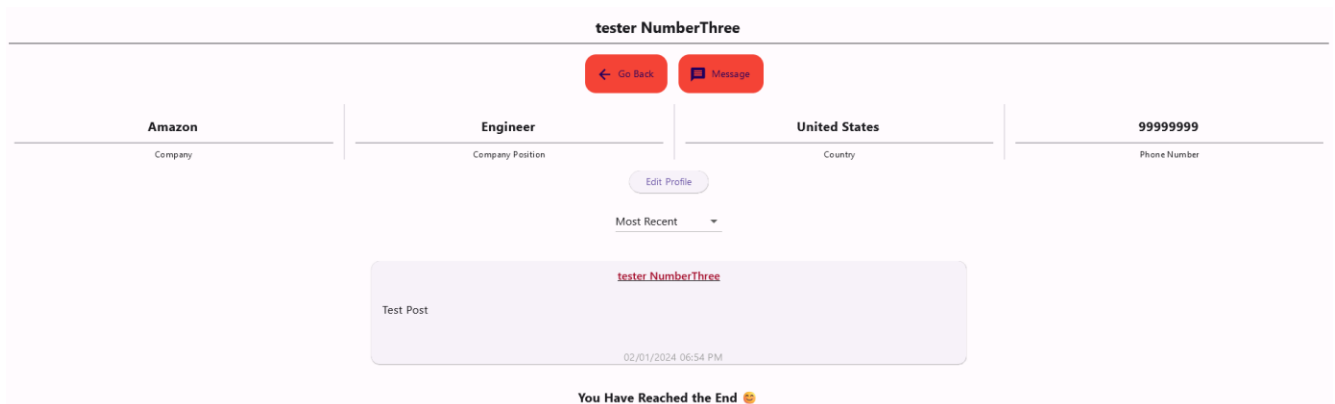
We currently do not have a test for the google maps api key. It seemed redundant to make a test just to ensure that an api key is working.

[Profile Page]

Functions and Interfaces Implemented

We added a user profile page that allows a user to see what posts they have made and information about their posts. This is useful so a user can catalog their interactions on the website. From the profile page if its someone else's profile you can message them.

After (There was no before as this is new)



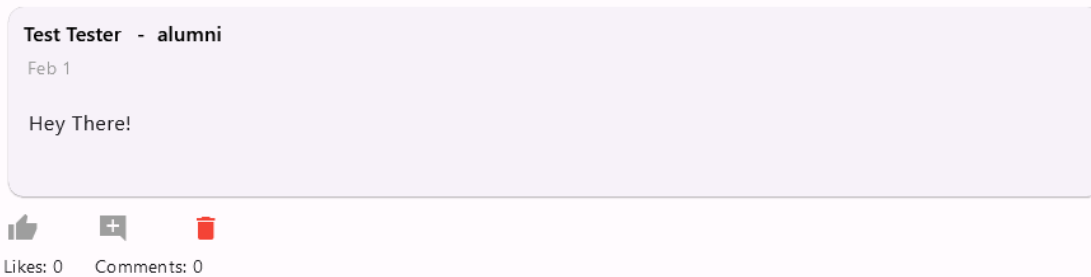
Preliminary Tests

While we have tested this functionally, we have yet to add consistent unit tests. We will add a test that checks all the posts a user has made, and the posts that are returned on the profile page. These posts lists should be identical, so this will ensure we're gathering data properly.

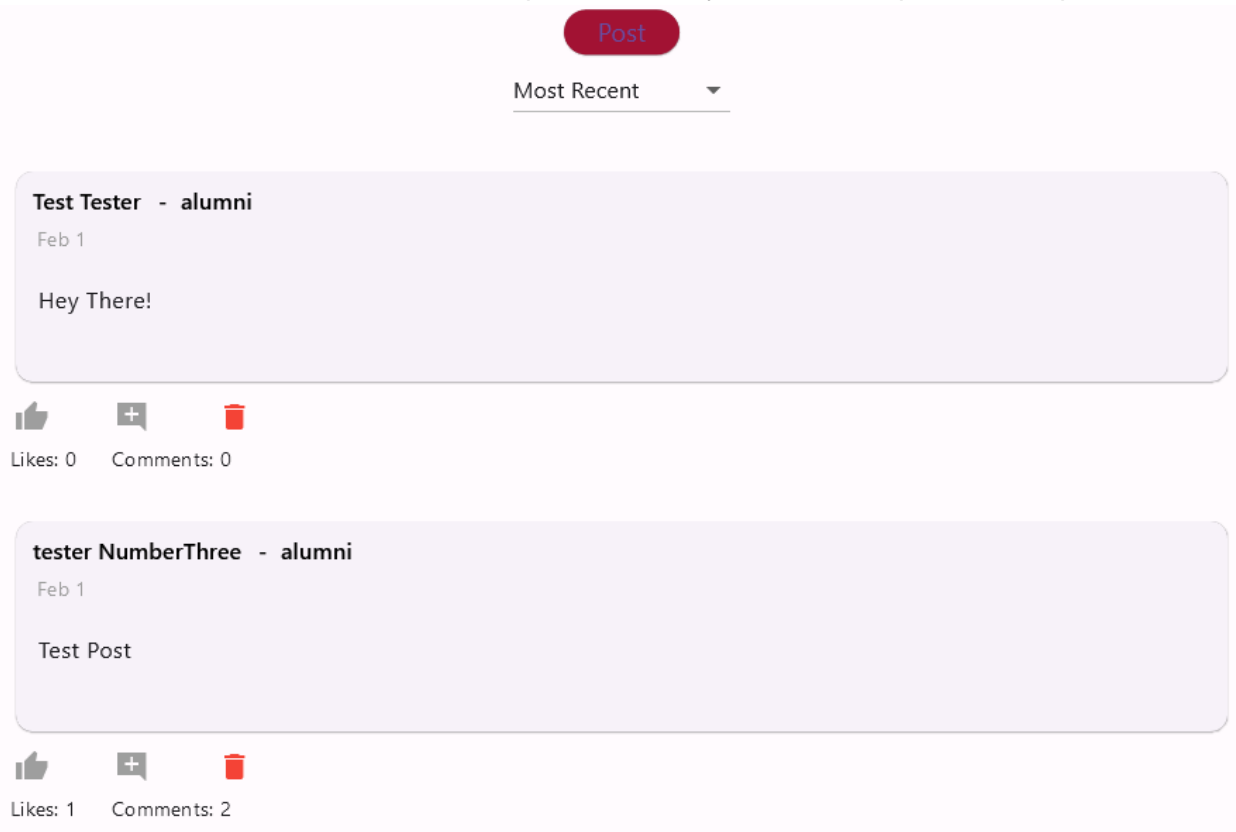
[Home Page]

Functions and Interfaces Implemented

The home page that the previous group left us was lacking in functionality. You could make posts, but you could not delete them or comment. We have added the ability to do all 3.



You can see in the picture above that the user has the ability to delete posts, and can leave likes and comments. You can see in the picture below you have the option to sort posts as well.



Preliminary Tests

We have yet to implement tests for any of the above features except sorting. It can be hard to implement testing with web environments, you need to be able to click and edit features. While we do plan on testing this, we have yet to implement the selenium web engine functionality necessary to ensure consistent tests. We have of course tested this extensively personally, we have yet to write consistent rewritable tests.

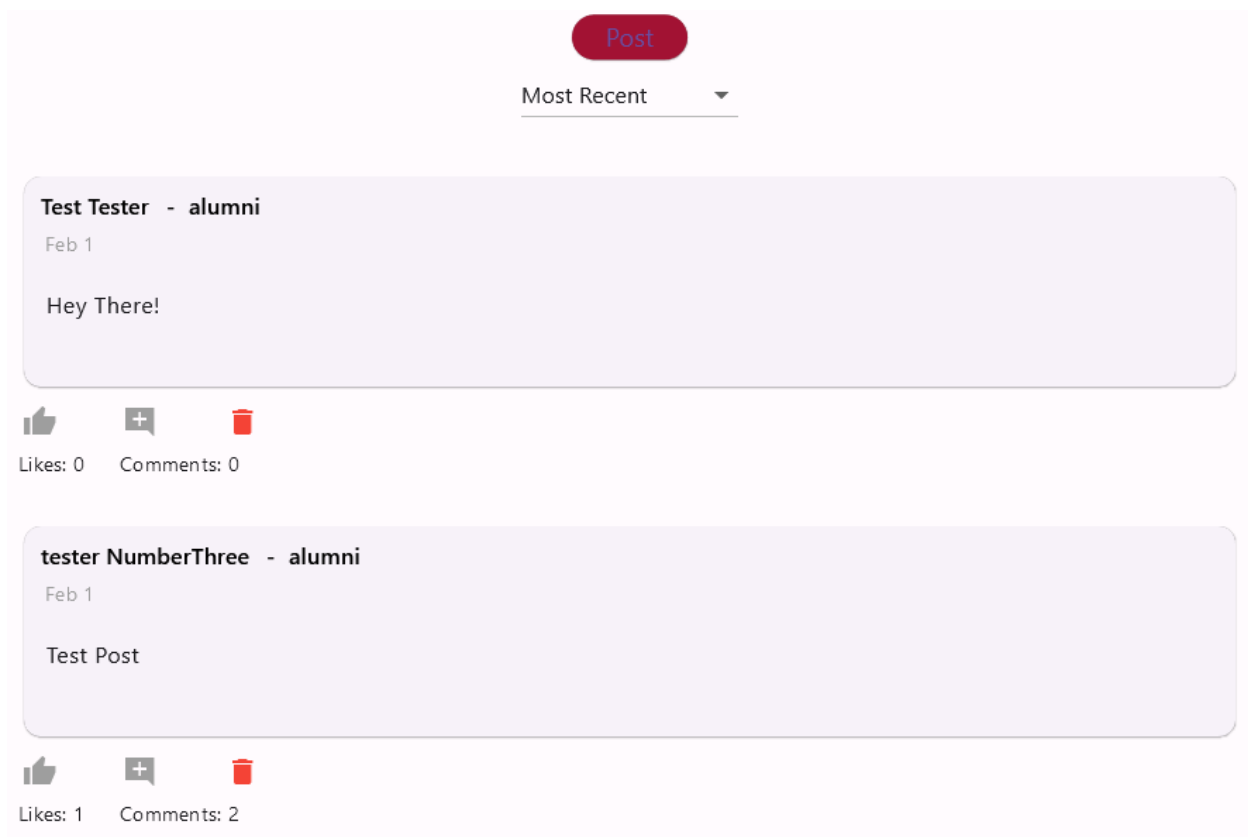
[Admin Functionality]

Functions and Interfaces Implemented

The prior group left no moderation tools for any potential admins. To this end we have created an admin account, with the email admin@wsu.edu. This account will be present on all versions of the cobb connect website. The admin has access to the Admin page, which shows all user information. The admin can delete any post, delegate other admins, and download all user data into CSV format. When delegating other admins, it means the main admin account can make other accounts able to delete posts. Currently the only thing those admins can do is delete any post, and they cannot access the user info page that the main admin account can see.

User List	
tester five Email: test5@gmail.com Phone: 1234567890 Country: United States State: Washington City: Silverdale Zipcode: 98383 Company: Twitch Position: PM Set As Admin	
tester NumberThree Email: tester3@gmail.com Phone: 99999999 Country: United States State: Washington City: Seattle Zipcode: 98004 Company: Amazon Position: Engineer Revoke Admin	
surfer dude Email: surferdude808@gmail.com Phone: 8082214905 Country: United States State: Hawaii City: Waianae	

Cobb Connect – Final Report



You can notice that the admin account has a delete option for every post. This is because they are the admin and can delete any post. In the screenshot above that you can also see that users can delegate admins. If you make someone an admin, they can also delete any post.

	A	B	C	D	E	F	G	H	I	J
1	Name	Last Name	Email	Phone	Country	City	State	Zipcode	Company	Position
2	tester	three	tester3@g	2.22E+10	United Sta	Pullman	Washingto	99163	GCISL	SE
3	Test	Tester	test@gma	1.23E+09	United Sta	Pullman	Washingto	99163	As	ASd
4	Gabe	Gaberson	test3@gm	1.23E+09	United Sta	Sammami	Washingto	98075	asd	asd
5	student	one	student@	11111111	United Sta	Bremerto	Washingto	98311	VCEA	Student
6	Gabe	Righi	gabriel.rig	1.23E+09	United Sta	Pullman	Washingto	99163	asd	asd
7	tester	four	tester4@g	3121	United Sta	Silverdale	Washingto	98383	VCEA	Student
8	Anthony	Salvione	anthony98	3.61E+09	United Sta	Pullman	Washingto	98133	WSU	Student
9	josh	frey	frey.josh.l	11111111	United Sta	Silverdale	Washingto	98383	VCEA	teacher
10	Francis	Smith	Francis@g	1.23E+09	United Sta	Sammami	Washingto	98075	Google	CEO
11	Tester	Two	tester2@g	1.11E+08	United Sta	Bremerto	Washingto	98311	VCEA	SE
12										

This is what the information looks like when downloaded.

XV. Product Delivery Status

The product is scheduled to be delivered to the client at the end of the semester, which is estimated to be around the end of May 2024. We will coordinate with the client to confirm the exact date and time of the delivery.

You can access all the client and course material related to the project on our GitHub repository, at the link <https://github.com/WSUCptSCapstone-F23-S24/gcisl-fullstackapp>. There, you will find the source code, documentation, test cases, and screenshots of the app. You can also view the live demo of the app, which is currently hosted on Firebase, a cloud-based service that provides database and storage solutions.

We are currently working on finding a more permanent and secure hosting option. A more permanent hosting spot is required for the project to be delivered and deployed successfully.

XVI. Conclusions and Future Work

XVI.1. Limitations and Recommendations

Navigation

One of the limitations of the app is the navigation between pages. The app has a navigation bar at the top of the screen, which contains links to the main pages of the app, such as Home, Inventory, Orders, Customers, and Analytics. However, the navigation bar does not provide any indication of which page the user is currently on, nor does it allow the user to go back to the previous page easily. This can cause confusion and frustration for the user, especially if they want to switch between pages frequently or compare data from different pages.

We recommend that the navigation bar should be improved to provide more feedback and functionality to the user. For example, the navigation bar could highlight the current page with a different color or a border, to show the user where they are. The navigation bar could also include a back button or a breadcrumb trail, to allow the user to go back to the previous page or navigate through the hierarchy of pages. These features would enhance the user experience and make the app more intuitive and user-friendly.

Features

Another limitation of the app is the lack of some features that could enhance the functionality and usability of the app. For example, the app does not allow the user to like or preview the comments that are posted on the inventory items or the orders. This could limit the user's engagement and interaction with the app, as well as the feedback and communication between the user and the client or the CBB team. The app also does not have a clean and fully functional analytics page, which could provide the user with more insights and data visualization on their inventory, orders, and customers. The app also does not have a profile page, which could allow

the user to customize their information and preferences, such as their name, email, password, theme, language, etc.

We recommend that the app should include these features in the future, as they would improve the functionality and usability of the app. For example, the app could allow the user to like or preview the comments by adding buttons or icons next to the comments, or by showing a pop-up window when the user hovers over the comments. The app could also improve the analytics page by adding more charts, graphs, tables, and filters, to display the data in a more meaningful and interactive way. The app could also add a profile page by adding a link to the navigation bar or a menu icon at the top right corner of the screen, which would take the user to a page where they can edit their information and preferences.

Hosting

A final limitation of the app is the hosting solution. The app is currently hosted on a WSU EECS Server, a cloud-based service that provides database and storage solutions. However, WSU EECS Server is only a temporary hosting spot for the app, and it is not secure or reliable enough for the app to be delivered and deployed successfully. WSU EECS Server also has some limitations on the amount of data and requests that it can handle, which could affect the performance and scalability of the app.

XVI.2. Future Work

There are always aspects that can improve a website. Since cobb connect is essentially a social media website, we could copy from popular social media sites and implement features that they have. Including some features that could enhance the functionality and usability of the app, such as allowing the user to like or preview the comments, improving the analytics page, adding a profile page, etc. We could also improve the navigation bar to provide more feedback and functionality to the user, such as highlighting the current page, adding a back button or a breadcrumb trail, etc. The most important future work right now is finding a permanent and secure hosting option, such as AWS, Azure, or Heroku, which would provide more features and benefits.

XVII. Acknowledgements

Special thanks for

Our Professor: Ananth Jillepalli

Our Client: Darcie Bagott/GCISL

Hosting: VCEA IT

XVIII. Glossary

Docker: A platform for packaging and deploying applications in containers

Flutter: Google's toolkit for building cross-platform mobile, web, and desktop apps.

Firebase: Mobile and web app development platform with various backend services.\

Selenium WebDriver: A tool for automating web browsers, commonly used for web application testing.

Beta Testing: Testing performed by a select group of users before a full release to gather feedback.

Load Testing: Testing the system's response under typical user loads.

Stress Testing: Testing the system's behavior under extreme conditions beyond its specifications.

Scalability Testing: Evaluating the system's ability to accommodate user growth.

Robustness Testing: Ensuring the system remains stable under various usage scenarios.

Responsiveness Testing: Testing the system's response time and resource utilization.

Acceptance Test: A test performed by end-users to compare the system to initial requirements.

Beta Testers: Users who participate in beta testing to provide feedback on the software.

XIX. References

- [1] D. Jolayemi, “Containerizing Flutter web apps with Docker,” LogRocket Blog, 19-Oct-2021. [Online]. Available: <https://blog.logrocket.com/containerizing-flutter-web-apps-with-docker/>
- [2] Firebase, “Integrate Flutter Web | Firebase Hosting,” Firebase Documentation, 27-Nov-2023. [Online]. Available: <https://firebase.google.com/docs/hosting/frameworks/flutter>
- [3] Appcircle, “Deploy a Flutter Web App to Firebase,” Appcircle Blog. [Online]. Available: <https://blog.appcircle.io/article/deploy-a-flutter-web-app-to-firebase>
- [4] LogRocket, “Migrate a Flutter mobile app to the web: Tutorial with examples,” LogRocket Blog. [Online]. Available: <https://blog.logrocket.com/how-to-migrate-a-flutter-mobile-app-to-the-web/>
- [5] “Flutter - Build apps for any screen,” Flutter Official Website. [Online]. Available: <https://flutter.dev/>
- [6] “Dart testing | Dart,” Dart Documentation. [Online]. Available: <https://dart.dev/guides/testing>
- [7] Firebase, “Firebase Hosting A complete foundation for your web app,” Firebase Documentation. [Online]. Available: <https://firebase.google.com/products/hosting/>
- [8] The Foraker Group, ‘Architectural Programming,’ 2023. [Online]. Available: <https://www.forakergroup.org/predevelopment/resources/architectural-programming/>
- [9] “S. Lai, ‘Start with data: how to incorporate data into your design process,’ UX Planet, Jan. 9, 2021. [Online]. Available: <https://uxplanet.org/start-with-data-how-to-incorporate-data-into-your-design-process-91ecba75daf8>.
- [10] “Coursera, ‘What Is UI Design? Definition, Tips, Best Practices,’ 2023. [Online]. Available: <https://www.coursera.org/articles/ui-design>.

[11] “What Is Unit Testing: Detailed Guide With Best Practices - LambdaTest,” LambdaTest. [Online]. Available: <https://www.lambdatest.com/learning-hub/unit-testing>

[12] “What is Functional Testing? Definition, Key Concepts, & Types,” Sitepoint. [Online]. Available: <https://www.sitepoint.com/functional-testing-introduction/>

[13] Making Group Chats in Flutter: 2. “How to Build Group Chat with Flutter?,” Rlogical Techsoft, Medium. [Online]. Available: <https://rlogicaltech.medium.com/how-to-build-group-chat-with-flutter-488dc3490bdf>

[14] “What is User Acceptance Testing (UAT)? with Examples,” Guru99, Software Engineering. [Online]. Available: <https://www.sitepoint.com/functional-testing-introduction/>

[15] “Best Practices for Creating Great User Guides,” Whatfix Blog, 27-Nov-2023. [Online]. Available: <https://whatfix.com/blog/user-guides/>

XX. Appendix A – Team Information

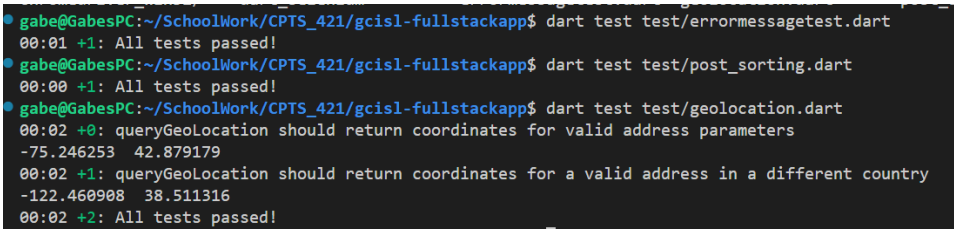
Gabriel Righi

Joshua Frey

Anthony Salvione

(We have never met in person so we don't have a team photo)

XXI. Appendix B - Example Testing Strategy Reporting

1) 

```

gabe@GablesPC:~/SchoolWork/CPTS_421/gcisl-fullstackapp$ dart test test/errormessage.dart
00:01 +1: All tests passed!
gabe@GablesPC:~/SchoolWork/CPTS_421/gcisl-fullstackapp$ dart test test/post_sorting.dart
00:00 +1: All tests passed!
gabe@GablesPC:~/SchoolWork/CPTS_421/gcisl-fullstackapp$ dart test test/geolocation.dart
00:02 +0: queryGeoLocation should return coordinates for valid address parameters
-75.246253 42.879179
00:02 +1: queryGeoLocation should return coordinates for a valid address in a different country
-122.460908 38.511316
00:02 +2: All tests passed!
  
```

XXII. Appendix C - Project Management

We meet with our client Darcie once a week. After these meetings we would also normally meet on our own on Discord. We used Discord as our primary means of communication. We used github issues and the github project board to control the versions of the website.

Sprint 4						
View 1 + New view						
Filter by keyword or by field						
Title	...	Assignees	...	Status	...	+
1 Place Edit Profile Page as function of Profile Page #207		joshua-frey-wsu		Done		
2 Make Sign-out Page and Dialog Boxes more User-friendly #206		joshua-frey-wsu		Done		
3 Bugfix on users being able to edit profile without filling in their state/city #205		joshua-frey-wsu		Done		
4 Adding to other user's profile pages #204		AnthBoss				
5 Further develop the profile page website #203		AnthBoss				
6 Fix bug of Name not updating on Posts/Comments/Replies when user updates in edit pro... #202		joshua-frey-wsu		In Progress		
7 Testing for User Registration #208		GabeRighi		In Progress		
8 Update Messages page #209		GabeRighi		Done		
9 Allow messaging from profile page. #214		GabeRighi		Done		
10 Prevent Being able to edit profile while viewing other user #216		joshua-frey-wsu		Done		
11 Have users last message displayed under user card #218		GabeRighi		Done		
12 Profile Picture #219		joshua-frey-wsu		Done		
+ You can use Control + Space to add an item						