

# **GCISL Full Stack Application**

**Project Solution Approach**

**Washington State University Granger Cobb Institute of  
Senior Living**



## **GCISL Team**

Musa Husseini, Tom Arad, Nathan Bunge

<b>I. Introduction</b>	<b>3</b>
<b>II. System Overview</b>	<b>3</b>
<b>III. Architecture Design</b>	<b>4</b>
III.1. Overview	4
III.2. Subsystem Decomposition	5
III.2.1 [Widgets]	5
a) Description	5
b) Concepts and Algorithms Generated	5
c) Interface Description	5
III.2.2 [View Models]	6
a) Description	6
b) Concepts and Algorithms Generated	6
c) Interface Description	6
III.2.3 [Controller]	7
a) Description	7
b) Concepts and Algorithms Generated	7
c) Interface Description	7
III.2.4 [Firebase]	8
a) Description	8
b) Concepts and Algorithms Generated	8
c) Interface Description	8
<b>IV. Data design</b>	<b>10</b>
<b>V. User Interface Design</b>	<b>11</b>
<b>VI. Glossary</b>	<b>11</b>
<b>VII. References</b>	<b>12</b>
<b>VIII. Appendices</b>	<b>13</b>
Appendix A	13
Image 1	13
Image 2	14
Image 3	15
Image 4	15
Image 5	16

# **I. Introduction**

The purpose of this document is to provide a solution approach to our project. This document will be discussing the architectural design, which will give an overview of our software, our subsystem decomposition which will give a description, concepts, algorithms, and an interface description that our software will use. It will also discuss the data design and how the software will maintain the data it is given. Lastly this document will also show a user interface design, which will be examples of what our product will look like.

The GCISL team has been assigned to make a web application and a mobile application that allows alumni to share their career tracks with other alumni, faculty, and students. Each user will be able to post their job history and chat with other users. The project overview section will go into more detail.

The motivation behind this project was the Granger Cobb Institute for Senior Living is a new institution in the school of hospitality looking for a way to stay connected with alumni in the field. They have data of students who have graduated from the hospitality program in the past eleven years and plan to share this application with them so faculty and alumni can reconnect. The GCISL team will provide the faculty with an application that can help connect the school with alumni.

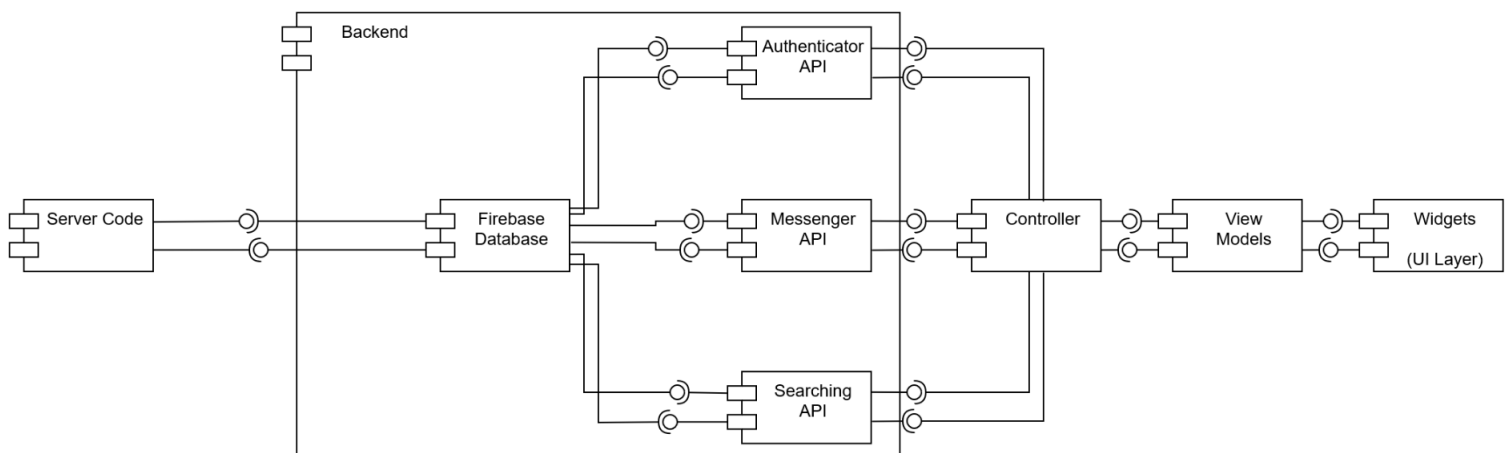
# **II. System Overview**

The GCISL program must have a robust and scalable architecture in order for the app to be functional and long lasting. It is a web based application, so we must define what is going on in the Front-End, the Back-End, and any APIs we will be using. We are mainly using Flutter for the UI and Front-End, and Firebase as the Back-End and APIs. The following sections will define what type of system architecture we used to organize these and why.

## III. Architecture Design

### III.1. Overview

We decided to use a Client-Server architecture for the GCISL program. This is because it is a web based application, so users will need to be able to view updated data that is being sent from a server. As stated before, we opted to use Flutter and Firebase as our main frameworks for our codebase, which consequently led us to create the architecture as shown in the following diagram. On the far right is the UI layer, which contains all the widgets we will define. These are all the things that the user will see on their page and interact with. The widgets themselves are only tasked with displaying information in the most well looking fashion and initiating callbacks only when necessary. Next to that is the View model, which holds state info and updates any incoming data when it is changed. The view models can be updated when users puts in an input and sends those to the controller. In the reverse setting, it takes the updates from the controller and sends them to the widgets to be displayed to the user. The controller is in charge of communicating with the different APIs to request the necessary information that the view model needs. All the APIs are part of the firebase backend. Currently, we are using the Authenticator, Messenger, and Searching API's that firebase offers [4]. Each of these aim to satisfy a different requirement, and are explained more in depth in the subsystem decomposition section. The three API's then access the Firebase Database as necessary to retrieve or add information. Finally, the Firebase Database is dictated by the defined server code. This will tell the database how to organize the data efficiently and effectively.



## III.2. Subsystem Decomposition

### III.2.1 [Widgets]

#### a) Description

The widgets are responsible for all the functions that the user will be interacting with directly. This is typically considered the front end of the project.

#### b) Concepts and Algorithms Generated

The model will be creating prototype forms that can be called and used in the application. Typically each class will be a form and will have multiple variables where the user input is stored.

#### c) Interface Description

##### Services Provided:

1. **Service name:** Registration View  
**Service provided to:** User  
**Description:** This will provide a user interface when the user is making an account
2. **Service name:** Login View  
**Service provided to:** User  
**Description:** This will provide a user interface when the user is attempting to log in
3. **Service name:** Post View  
**Service provided to:** User  
**Description:** This view will provide a view of all the job postings from users on the platform.
4. **Service name:** Messaging View  
**Service provided to:** User  
**Description:** This view will provide a user interface when users are messaging between each other.

##### 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.

### III.2.2 [View Models]

#### a) Description

The view model is responsible for defining essential components of the application. It will provide the forms that users will fill out for logging in, registration, posting job updates, or sending messages.

#### b) Concepts and Algorithms Generated

This is where a lot of the html, css, and javascript will be kept. It will create the layout of the application and control what the user sees.

#### c) Interface Description

##### **Services Provided:**

1. **Service name:** Faculty model  
**Service provided to:** Widge, Controller  
**Description:** The faculty model will define a specific type of user. This type of user will have basic attributes such as name, email, profession, etc. But this type of role will allow faculty to have access to specific data analytics on other users.
2. **Service name:** Alumni model  
**Service provided to:** Widge, Controller  
**Description:** This model is similar to the faculty model. The key difference is this type of user will be considered a “standard” user and will not have access to data analytics
3. **Service name:** Posting model  
**Service provided to:** Widge, Controller  
**Description:** The posting model is responsible for creating a form for posting messages, such as job postings. It will include attributes such as title, description, start date, end date, etc...
4. **Service name:** Messaging model  
**Service provided to:** Widge, Controller  
**Description:** This model will be similar to the posting model. The key difference with the messaging model is it is only intended for two users to interact with. There will only be a few attributes associated with it also, such as, recipient, title, and body of message.

##### **Services Required:**

**Service Name:** Controller

**Service Provided From:** the controller subsystem

### III.2.3 [Controller]

#### a) Description

The controller is responsible for defining how a user will add or complete a task. It acts as a connection between the view and database. It takes user input and decides what to do with it, making calls to the backend as necessary.

#### b) Concepts and Algorithms Generated

The controller will be in charge of managing the routes and forms that a user may go through when using the application. It will handle GET, POST, and DELETE methods as well as security for certain landing pages. It will take in the data received from the user, process it, and send the results to the model.

#### c) Interface Description

##### Services Provided:

1. **Service name:** Register (GET, POST)  
**Service provided to:** View Model, Backend - Authenticator  
**Description:** It will request data such as name, job, job title, years of experience, and any other relevant information. Once the required information is received it will add the new user into the database.
2. **Service name:** Login (GET, POST)  
**Service provided to:** View Model, Backend - Authenticator  
**Description:** It will request data from the user asking for login credentials, if there is a match in the database they'll be redirected to a success page, if not they will be prompted to login again.
3. **Service name:** Logout (GET)  
**Service provided to:** View Model, Backend - Authenticator  
**Description:** If the user is logged in they will be logged out and sent back to a login screen.
4. **Service name:** Post Job (POST)  
**Service provided to:** View Model, Backend - Database  
**Description:** It will take the information a user inputs for a job posting, create a database entry for it, and post it to the server.
5. **Service name:** Send Message (POST)  
**Service provided to:** Model, View, Database - Messaging

**Description:** This will allow users to send messages to other users.

**Services Required:**

**Service Name:** Messaging Data

**Service Provided From:** Message API

**Service Name:** Authenticating User Information

**Service Provided From:** Authenticator API

**Service Name:** Searching tool

**Service Provided From:** Searching API

### **III.2.4 [Firebase]**

**a) Description**

The database will be the place where all information is stored. Information can vary from user information, postings, messages, and analytics.

**b) Concepts and Algorithms Generated**

The database will be created to store information in different tables. Each table can have relationships with each to share information. In order to call this information there will need to be queries called upon tables and each of its columns to retrieve specific information.

**c) Interface Description**

**Services Provided:**

1. **Service name:** User information table  
**Service provided to:** Controller subsystem  
**Description:** This table will hold all relevant user information such as names, login credentials, and career information
2. **Service name:** Post table  
**Service provided to:** Controller subsystem  
**Description:** This table will contain information on each post made. Columns of this table can include but are not limited to, title, body, and comments.
3. **Service name:** Messaging table  
**Service provided to:** Controller subsystem  
**Description:** This table will keep track of all the messages sent between users. It will allow there to be message history.



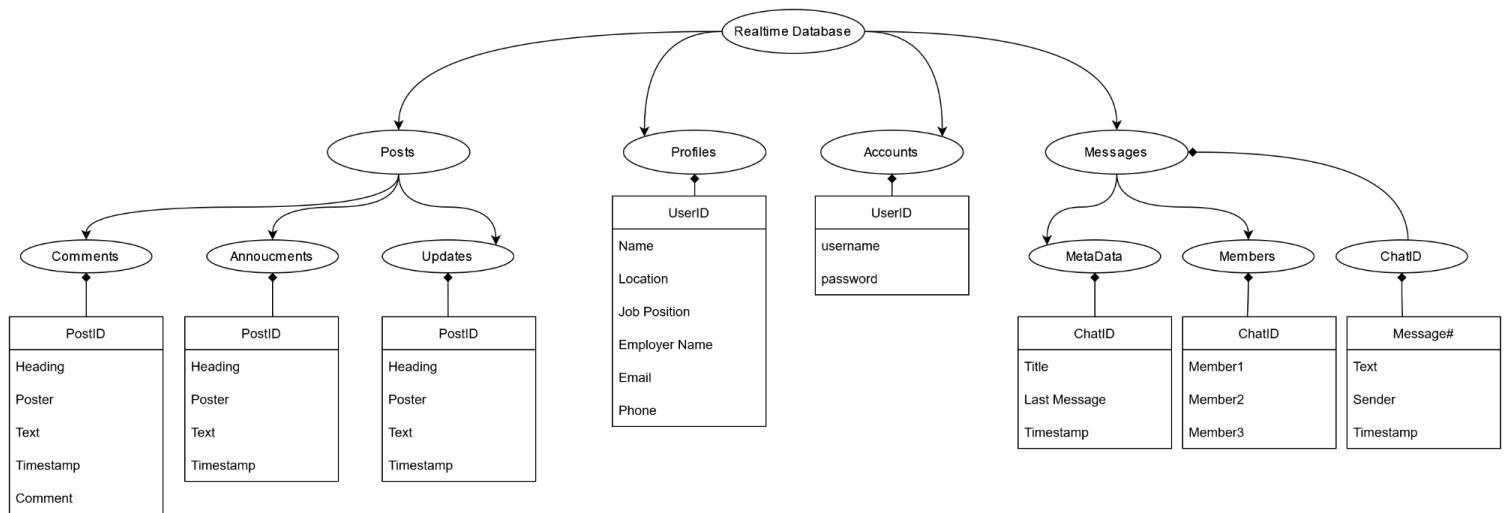
4. **Service name:** Analytic Table  
**Service provided to:** Controller, Faculty Users  
**Description:** This table will store specific data that can only be seen by faculty users.
5. **Service name:** Message API  
**Service provided to:** Controller and Database  
**Description:** This API will handle messaging between users and notifications.
6. **Service name:** Authenticator API  
**Service provided to:** Controller and Database  
**Description:** This API will handle user credentials when logging into their accounts.
7. **Service name:** Searching API  
**Service provided to:** Controller and Database  
**Description:** This API will handle search queries when getting data from the database.

**Services Required:**

1. **Service Name:** Controller  
**Service Provided From:** Controller subsystem
2. **Service Name:** Model  
**Service Provided From:** Model subsystem

## IV. Data design

We will be utilizing Firebase as our main source of API extensions and database, which means we will store our data in Firebase's default data structure format. Firebase stores data in realtime using a JSON tree [6]. As such, each of our data objects will be stored as a node in the JSON tree.



Posts: Posts will either be stored under announcements or updates. This makes searching by one or the other categories much faster than having them altogether. For example, If they were classified in one branch, then every single post would be retrieved if you requested to see only announcements. However, if they are separated, then only the announcement branch would be searched. For the same reason, comments are separated into a different branch. Comments are only displayed when you select to view them, so we don't need to retrieve them from the database every time we are viewing the main feed page.

Profiles: Profiles contain all data for analytics that are necessary to track. So far this is only Name, Work Location, JobPosition, Employer Name, Contact Email and Phone. Depending on which categories are searched more frequently than others, this may be reorganized in the future to optimize queries.

Accounts: Accounts only contain the information necessary for users to login. This is to optimize sign up/sign in times for the application.

Messaging: Contains all the data necessary to allow users to message each other and view old messages. MetaData contains the minimum data needed to display all the chat preview windows. This is separated so that the preview windows can all be loaded without having to load in all of the messages in the whole database. Members contain the members that are a part of each chat session, and ChatID contains the actual text for every message in a session.

## V. User Interface Design

The GCISL team has created a partial UI design for the GCISL web application. [Appendix A](#) contains images of this design. Upon launching the application will display [Image 1](#), the login/sign up page. This page will allow users to create and sign into their account (Satisfies Login use case). Following that the application will display the “Home” page as can be seen in [Image 2](#) it contains a feed of posts made by the other users and allows creating new posts (Feed use case). [Image 3](#) displays the “Personal Information” page” this page will allow the users to provide and edit their current information in accordance with the data the client is interested in (Profile use case). Once the client enters their information it will only be updated if they click the “save” button at the bottom of the page. This is a barebone design of what the page will look based on the client’s interests and as such it is likely to change in the future to be more visually appealing. [Image 4](#) displays the “Messages” page this page allows users to view their message history with other users as well as send new messages (Messaging use case). This page also has a lookup feature for users to find specific people they want to contact. The design for this page was inspired by a community design on figma.<sup>[2]</sup> The last image ([Image 5](#)) displays the “Analytics” page (Site Analytics use case). This page will be visible only to those with staff permissions and will contain all the personal information provided by the users. Currently the information is displayed in a table, but further discussion with the client about data visualization could lead to changes. In addition every single page will have a side menu as can be seen in all the images. Through this menu they will be able to go between the different pages of this application described above.

Some features that appear in the use case diagram do not appear in this early design. The home page should allow users with staff permissions to delete posts. This feature will be added later once the UI team gets more time to expand on the design. Our use case diagram also shows that users should be able to alter their account settings. Since these settings are not yet clear to us we have not been able to create a design which implements this option, hence this will be also added to the UI at a later point.

## VI. Glossary

**Flutter:** An open source user interface development kit

**Firebase:** A hosting service for applications

**Figma:** A web application used for interface design

**GCISL:** Granger Cobb Institute for Senior Living

**Front End:** The part of the application the user will be seeing and interacting with

**Back End:** The part of the application that runs the logic and processing

**Database:** The part of the application that stores all data

**Full Stack Application:** An application that has front end, back end, and a database working together

**API:** Stands for application programming interface. It is a type of software interface that offers a service to other pieces of software.

## VII. References

1. (Dutoit, 2010), 3<sup>rd</sup> Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice Hall, 2010.
2. "Apple Messages Template". Figma Community.  
<https://www.figma.com/community/file/1118503083462368752> (October 4, 2022)
3. "Structure Your Database" Firebase Documentaion  
<https://firebase.google.com/docs/database/web/structure-data> (September 15, 2022)
4. Firebase Extensions, Google (October 1, 2022)  
<https://firebase.google.com/products/extensions>

Cite your references here.

For the papers you cite give the authors, the title of the article, the journal name, journal volume number, date of publication and inclusive page numbers. Giving only the URL for the journal is not appropriate. You should use either IEEE or Chicago style formatting for your citations

For the websites, give the title, author (if applicable) and the website URL.

# VIII. Appendices

## Appendix A

Image 1

**Sign In**

Sign in to stay connected.





Email

Password

☐ Remember me? [Forgot Password](#)

**Sign in**

or sign in with other accounts?

Don't have an account? [Click here to sign up.](#)

Image 2

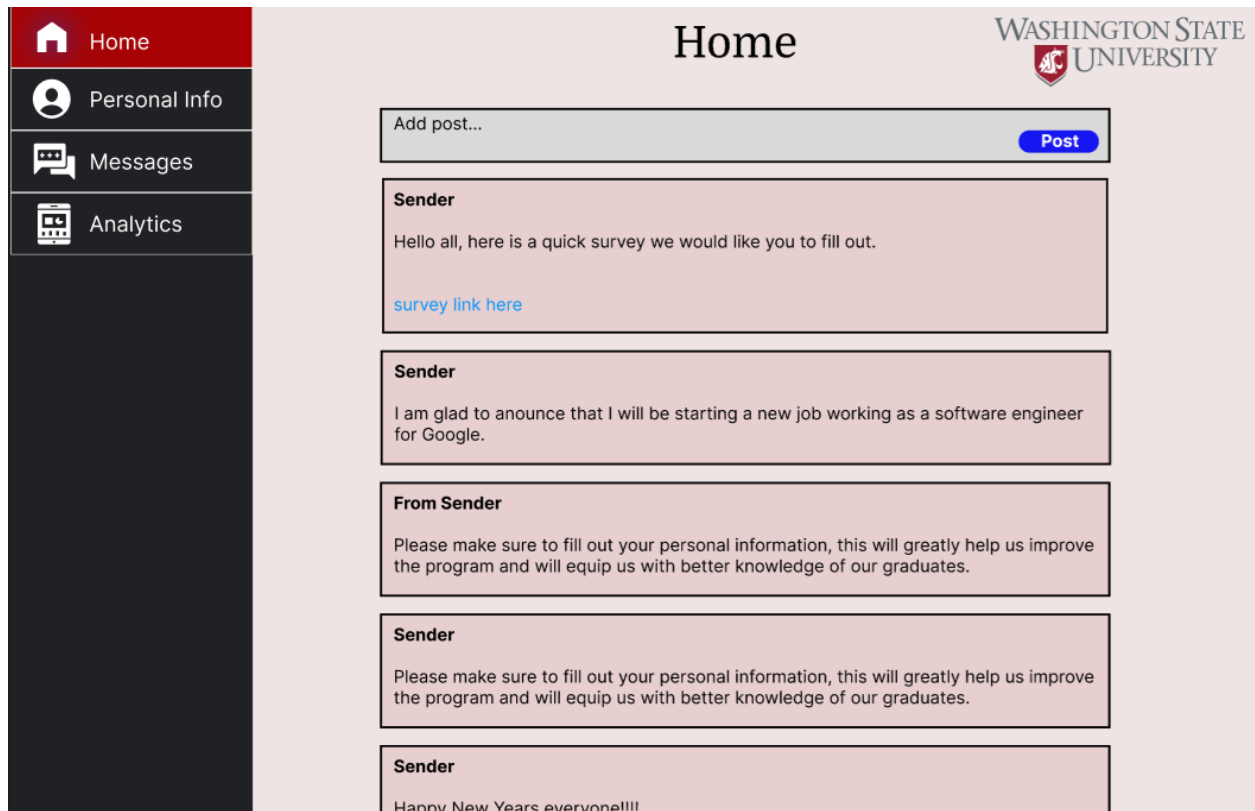


Image 3

Home

Personal Info

Messages

Analytics

# Profile Information

WASHINGTON STATE UNIVERSITY

Name (first and last)  
Enter Name Here...

Employer's Name  
Enter Employer Name Here...

Location (City, State)  
Enter Location Here...

Job Title  
Enter Job Title Here...

Employment Date  
Enter Name Here...

Email  
Enter Email Here...

Phone Number  
Enter Phone Number Here...

Save

Image 4

Home

Personal Info

Messages

Analytics

Stephen >

Let's get lunch. How about pizza?

Let's do it! I'm in a meeting until noon.

That's perfect. There's a new place on Main St I've been wanting to check out. I hear their hawaiian pizza is off the hook!

I don't know why people are so anti pineapple pizza. I kind of like it.

# Messages

Search

Stephen Yustiono 9:36 AM >  
Nice. I don't know why people get all worked up about hawaiian pizza. I ...

Erin Steed 9:28 AM >  
(Sad fact: you cannot search for a gif of the word "gif", just gives you gifs.)

Daisy Tinsley 9:20 AM >  
Maybe email isn't the best form of communication.

Zach Friedman 9:00 AM >  
Tabs make way more sense than spaces. Convince me I'm wrong. LOL.

Kyle & Aaron 8:58 AM >  
That's what I'm talking about!

Dee McRobie 8:35 AM >  
There's no way you'll be able to jump your motorcycle over that bus.


Gary Butcher 8:32 AM >  
Nathan is a hater, you can totes make that jump. Do it. Do it.

Kyle Norman Olson 8:02 AM >  
#letsgetcoffee

Dee & Daisy 7:45 AM >  
Good morning! I've got some great news, we got the grant!!!

15

### Image 5

	Analytics					
Name	Employer	Location	Job Title	Employment Date	Email	
Bob McGee	Kaiser Permanente	Sunnydale, California	Respiratory Therapist	10/02/2020	bob.mcgee@kp.org	
Robert Lamb	Providence	Seattle, Washington	Nurse Assistant	10/04/2022	robert.lamb@providence.org	