CA400 Functional Specifications

SAGE

(Decision Making App)

Joanna Talvo - 18342523
Chloe Ward - 18302716
Group Members

Dr. Annalina Caputo
Project Supervisor

11/11/2021

0. Table of contents

1. Introduction

1.1 Overview

Our project is a decision making application. Organising plans with a large number of people can often be difficult. SAGE is a progressive web application that will behave as a decision maker which will ultimately simplify the decision making process for friends, family and colleagues.

The main functionality of SAGE is to provide a group of people with an activity based upon their mutual preferences and availability. The application will offer the users with 3 categories to choose from such as activities to do, places to eat and movies to watch. It will also take into account the current day's weather and users' availability with the aid of an in app calendar to find a solution that suits all group members. It will have a swipe feature where the user can swipe right for “yes” and left for “no” to indicate whether they like or dislike an activity. This will ultimately bring us to the common agreed solution for all users - if a tie is set then all users will do a re-vote otherwise, the most voted activity will be the one chosen.

The reason behind choosing to develop an application like SAGE was from personal experience. We believe that organising plans with friends is difficult especially during a global pandemic when external factors such as weather indicate what we can or can not do. Therefore, we wanted to make this process simpler for everyone.

1.2 Business Context

The following examples could be implemented in terms of business for this application:

- Companies could use our application to plan events. It can ultimately be adapted for situations in software development such as planning poker.
- Advertisements could be displayed throughout the application such as restaurants, cafes, activities or movies in order to gain revenue from the application.
- Businesses could use it for planning social events with staff members.

1.3 Glossary

- **Progressive Web Application** - a web app that uses service workers, manifests and other web-platform features in combination with progressive enhancement to give users an experience on par with native apps.
- **Heroku** - a container-based cloud platform where developers can deploy, manage and scale their apps.
- **Firebase** - a cloud-based platform owned by Google that provides developers with tools for building their applications.

- **Firebase Authentication** - this provides methods for users to create an account and to manage their email address and passwords. It also manages user authentication and verification.
- **Firebase Cloud Storage** - this allows users to upload and share user-generated content within the application.
- **Firebase RealTime Database** - a cloud-hosted database that syncs and stores data in real time within our application.
- **API** - used for retrieving data from social media and open-timetable to be used in our application.

2. General Description

2.1 Product / System Functions

Below are the main functionalities that will be implemented into the application. Over the course of the next few months, our functionalities might change where we incorporate more or modify some. However, these are the main features of the application that we plan to implement:

The main functionalities below will at first require the user to create an account and verify their email.

- Download the application or access in the browser and create an account.
- Verification of user's email address.
- Login to the application with a password. A reset option is provided.
- Create or join a group.
- Group dashboard (which includes the members and events of that group)
- Calendar (where a user inputs their available dates)
- Options section (swipe feature)
- Side drawer (which includes user profile, notifications, events, and option to change password)

2.2 User Characteristics and Objectives

The target audience of SAGE is everyone that has an electronic device with access to the internet. We all need to make decisions in our daily lives. Whether you're planning group events or a movie to watch with friends, SAGE will simplify the process for you.

The user interface of the application will be easy to use since it will provide a user friendly experience. We will be using Shneiderman's Eight Golden Rules of interface design throughout our user interface development process to ensure the user has the best possible experience. Once a user creates an account, they will be able to gain access to all the

available features such as the homepage, user profile, group dashboard and creating an event.

The structure of the user interface will be easy to use and self explanatory for the user however we also plan to have a help section which will explain each page in detail to ensure the user has the smoothest experience when navigating through the application.

2.3 Operational Scenarios

Unregistered User

An unregistered user will not have access to the application's features. In order to achieve this, they will have to register on our registration page and fill in the details required which are their name, email address and password. Our application will verify the email address before the user can proceed to logging in.

Registered User

A registered user will have full access to the application's features. They will be able to log in using their verified email address and password.

Change password

A user will have the ability to change their password if they wish to. This can be found in the user profile.

Reset password

A user will have the ability to reset their password in a case where they can't remember their login credentials.

Edit Profile

A user will have the ability to edit their profile. The name they have provided in registration will be initialised as their username and they can change their names whenever they want.

Create a group

A user will be able to create a group and this user will act as the host of this group. The host will then add any members they want to be part of the group using their email address providing they are a registered user of the application. There will be only one host per group and only the host has the ability to add members in the group.

Join a group

A user can only join a group once the host of that group has requested them to join. The user will be notified when they have been added to a group.

Set a date

Each user in a group will be prompted with a calendar to pick 5 maximum available dates. These dates then will be calculated by our application to pick the date when the majority is available.

Pick a category and survey

The host in the group will pick a category from activities, places or movies. After selecting a category, all members of the group will then be given a brief survey to fill out which our application will base the recommendations on.

Pick options

A user will then have a choice to pick from the options our application will show. A user can swipe right for “yes” and left for “no”.

Getting notifications

A user will get notifications to inform them on certain situations such as if they have been requested to join a group, their upcoming event or to finish their survey.

2.4 Constraints

Software Development Technology

- One of the team members will be new to Next.js so they will have to devote time to learning the framework.
- We have to successfully tie the backend to the frontend.
- We will have to create an algorithm for the web application that will allow us to successfully monitor matched decisions and also find an available date for all group members.
- Successfully learn to configure a Progressive Web Application (PWA) installable.
- Ingestion of data and designing the database.

Time

- Due to the time constraint, we will focus on implementing the core functionality of the application and to ensure it works to our desired standard while also carrying out our iterative testing throughout the development process. We will then extend the core functionality by adding more features to it if we have sufficient time left.

Internet Access

- Team members must have access to the internet (wifi or data) in order to create the app and its functionality.
- The team will have to communicate with each other over the internet on certain days so in order to complete the project successfully each member needs to have access to a stable internet connection.

3. Functional Requirements

3.1 Registration

- **Description**
    - For unregistered users, it is required for them to register before being able to login. The registration form requires the users to input information such as first and last name, and password while also providing a valid email address. Once the user has submitted their registration form, a verification email link will be sent to their email address.
- **Criticality**
    - This part of the application is essential to ensure that each user is using a valid email address that they have access to. It combats people making multiple accounts with random email addresses. The verification process makes users know that the people that they are interacting with are verified which establishes a users trust in the application as random people will not be able to join their groups.
- **Technical Issues**
    - There will be no technical issues when implementing this feature.
- **Dependencies with other requirements**
    - None.

3.2 Login

- **Description**
    - Once the user has successfully registered and got verified, the user can now login using their email address and password. The registration details of the user will be automatically stored in our database. These will be used to compare and check if the information they provide matches the ones stored in our database otherwise, a prompt will appear on the screen informing the user. In the event that the user forgets their password, a “forgot password” option will be provided.
- **Criticality**
    - This function is essential for identification and security purposes. It is the entry point to establish a connection between the user and the application. In addition, it also checks if the user has provided a valid account before getting access to the application.
- **Technical Issues**
    - The input details provided by the user must match the ones we have recorded in our database. This will be done by querying the database. In the event where the user forgets their password, they will receive a new password through the email they registered with.
- **Dependencies with other requirements**
    - The user must complete all the necessary requirements in the registration process first in order to use this functionality.

3.3 Home Page

- **Description**
    - Once a user has successfully logged into the application, they will be able to view the main functionalities of the application. In the Home Page, the user will be able to view the sidebar navigation, and the list of groups that they have joined. If the user has joined no groups, then it will prompt the user to join a group.
- **Criticality**
    - This is a necessary function in the application as it allows easy navigation throughout the application since it shows the user all the groups that they have joined.
- **Technical Issues**
    - We have to find a way to make the groups visible only to the members. A user needs to be added to a group to see the group's events and members. Only the host has the ability to do so.
- **Dependencies with other requirements**
    - The user must first be able to log into the application to view the homepage.

3.4 User Profile

- **Description**
    - Upon registration, the user can now start creating their own profile when they successfully log in. The user will also be able to change the information they provided at registration at any time they want, such as to change their password.
- **Criticality**
    - This is not entirely necessary to the overall functionality of the application. However, if a user forgets their password, we want them to be able to change it from within the application instead of logging out to reset it.
- **Technical Issues**
    - No technical issues.
- **Dependencies with other requirements**
    - The user must be registered and logged in properly.

3.5 Creating a group

- **Description**
    - Once a user is successfully logged into the application, they can then begin creating a group. The host inputs the verified users email addresses to invite them to the group. The creator of the group then picks the category that will consist of the decision making survey for the group members to take.
- **Criticality**
    - This is an important feature as it will allow users to join the group and take part in the pick date, survey and pick options so that a solution can be found for their group decision making.

- **Technical Issues**
    - We must implement a way that the users will be able to see the notification pop up that they have been added to a group.
- **Dependencies with other requirements**
    - The user must have the emails of the registered users to join a group.

3.6 Joining a group

- **Description**
    - Once a group has been created and a user has been invited to the group, the user will receive a notification they have been added to a group. This will give the user access to the group's events and take part in the decision process.
- **Criticality**
    - This is essential because a user won't be able to be part of any of the events if they are not part of the group.
- **Technical Issues**
    - The user must receive a notification that they have been added to a group.
- **Dependencies with other requirements**
    - None.

3.7 Setting a date

- **Description**
    - The user will be asked to choose a maximum of 5 days including the time that they are available to take part in the activity. An in app calendar will be available to the users to select the dates.
- **Criticality**
    - Setting a date is important for the group as we need to find a date when the majority of members is free to do the activity.
- **Technical Issues**
    - We must implement a calendar in the application where we can successfully store the users availability so that we can query it from the database when finding a date that all users are free.
- **Dependencies with other requirements**
    - The user must have joined a group to participate in picking a date.

3.8 Answering survey

- **Description**
    - A user will be prompted to fill out a brief survey that the application will base the recommendations on.
- **Criticality**
    - This will allow us to provide recommendations for the user's picked options section with options that they will like based on their responses to the survey.
- **Technical Issues**

    - We must provide a survey that is discriminative to make the dataset small, so that we provide activities that are suitable for their preferences.
- **Dependencies with other requirements**
    - The user must be part of a group to gain access to the survey.

3.9 Pick Options

- **Description**
    - A user will gain access to the pick options. Here, the user will swipe right for “yes” and left for “no” on the options that we will supply them with.
- **Criticality**
    - This will allow us to collect the data and store it in our database so that we can compare it with the other users' responses in order to run the algorithm against it to find a common matched activity.
- **Technical Issues**
    - Successfully implementing a swipe feature to record what the user likes and dislikes in the recommendation system.
- **Dependencies with other requirements**
    - The user must have completed the pick date section to progress to this section.

3.10 Record choices

- **Description**
    - Once a user has completed and picked their options, they will be prompted to press a submit button. Here, their choices will be calculated and added into our database.
- **Criticality**
    - To allow the matched option to be successfully stored in our database.
- **Technical Issues**
    - We plan to add the data into the firebase realtime database so that it's constantly updating when they pick their choices.
- **Dependencies with other requirements**
    - The user must have completed the full survey to submit it so that the user's choices can be recorded in the database.

3.11 Matched Option

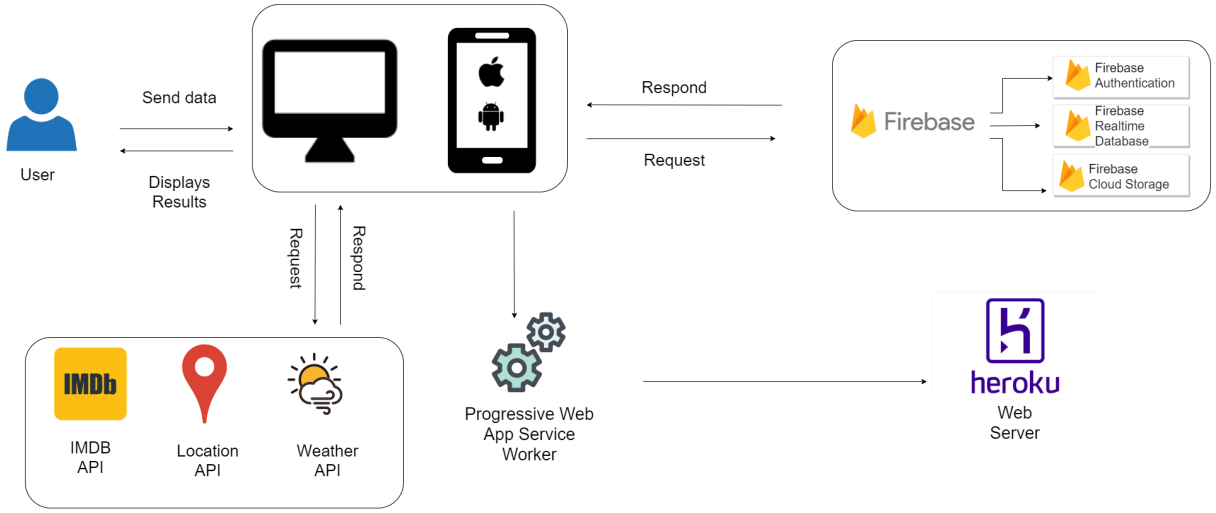
- **Description**
    - Once all users have picked their options and the time limit for completing the survey is up, the users in the group will be provided with the results. It will show the users the date when they are all free and the option that was matched among them all.
- **Criticality**
    - The matched option is vital for the application as when all the users have completed the survey and a final decision has been generated by the

application then the results will appear in the user interface for the users to see. This will allow them to see the activity results and when everyone is free to do the activity.

- **Technical Issues**
    - We have to create an algorithm that will provide us with the matched options as well as informing users about the date and time that they are all available.
    - If no date is free between all of them then the application will suggest the date when the majority is free.
    - If there is no matched option, our application will have to perform a second iteration based on all the options the members have picked. Otherwise, we will provide a random recommendation.
- **Dependencies with other requirements**
    - The user must have completed the pick options for the results to be published.

4. System Architecture

4.1 System Architecture Diagram



The system architecture of our application is shown above. The diagram consists of the user which initiates the starting of the application. The desktop and a mobile phone is shown with both Android and IOS which represents the front end which we will be using Next.js to implement. The database is also shown which will be implemented using Firebase.

- **User**
  - The user initiates the interaction between the mobile application and the server. The user can send data requests to the server and the server will respond to its request through the user's choice of device.

- **Devices**
  - The application will work on both mobile phones and desktop. This will take request data from the client and send it to the server and display the response from the server to the client.

- **Firebase**
  - Firebase has many features which we plan to take advantage of for our application, such as the Realtime Database which will allow us to store and sync data in real time. Firebase authentication will allow us to build a secure authentication system. Cloud storage will easily allow us to store and serve user content.

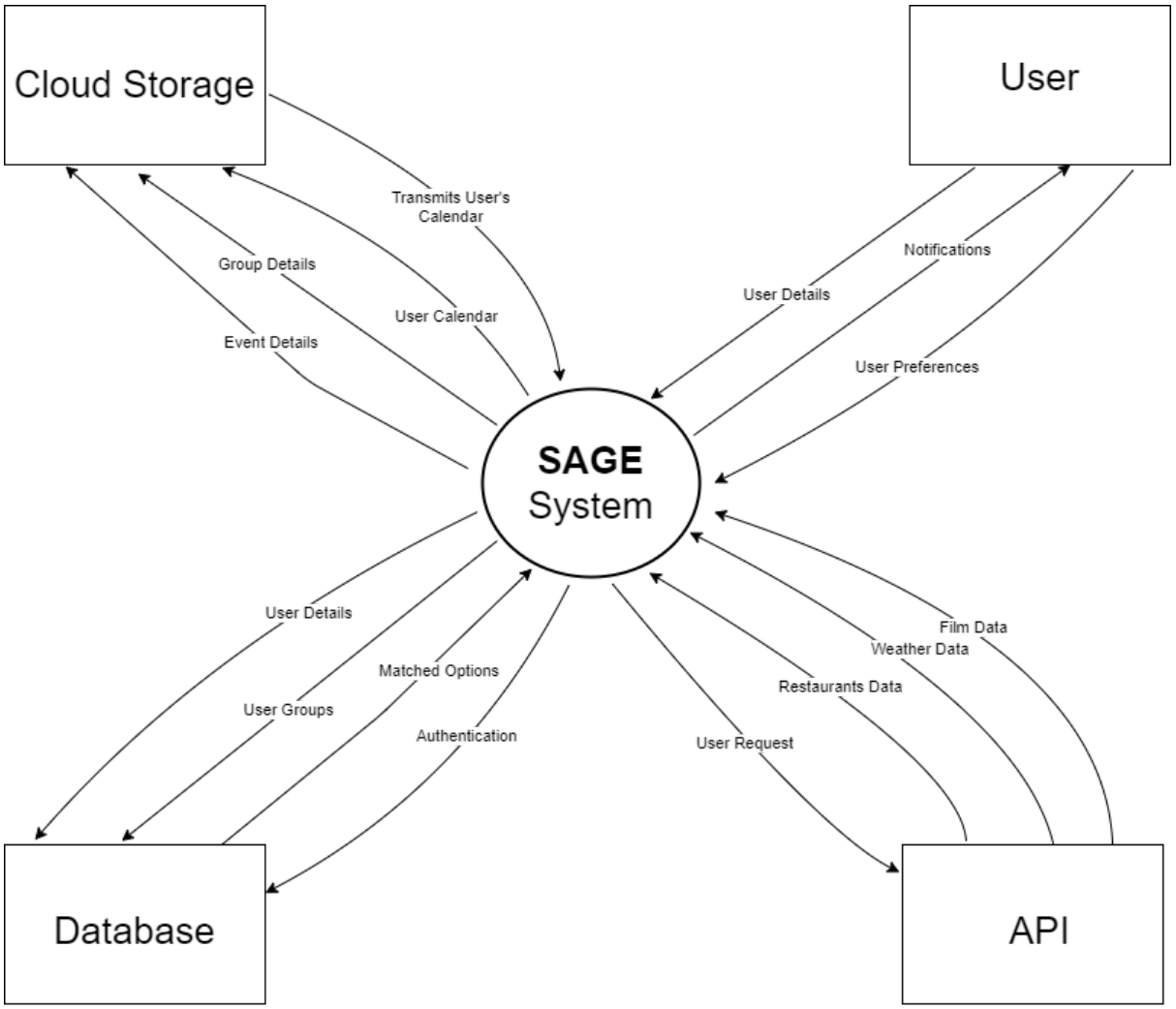
- **APIs**
  - Our application will implement various API's, such as the IMDB, a weather API and a location API. This will ultimately allow us to extend the functionality of our application by supplying users with a large variety of movies with the IMDB API and access to their current location and restaurants near them with the location API.

- **Service Worker**
  - By configuring our application as a progressive web application, it adds a new technical dynamic, the service worker is a script that runs in the user's device browser and will manage the caching etc for an application which will give the users a better experience.

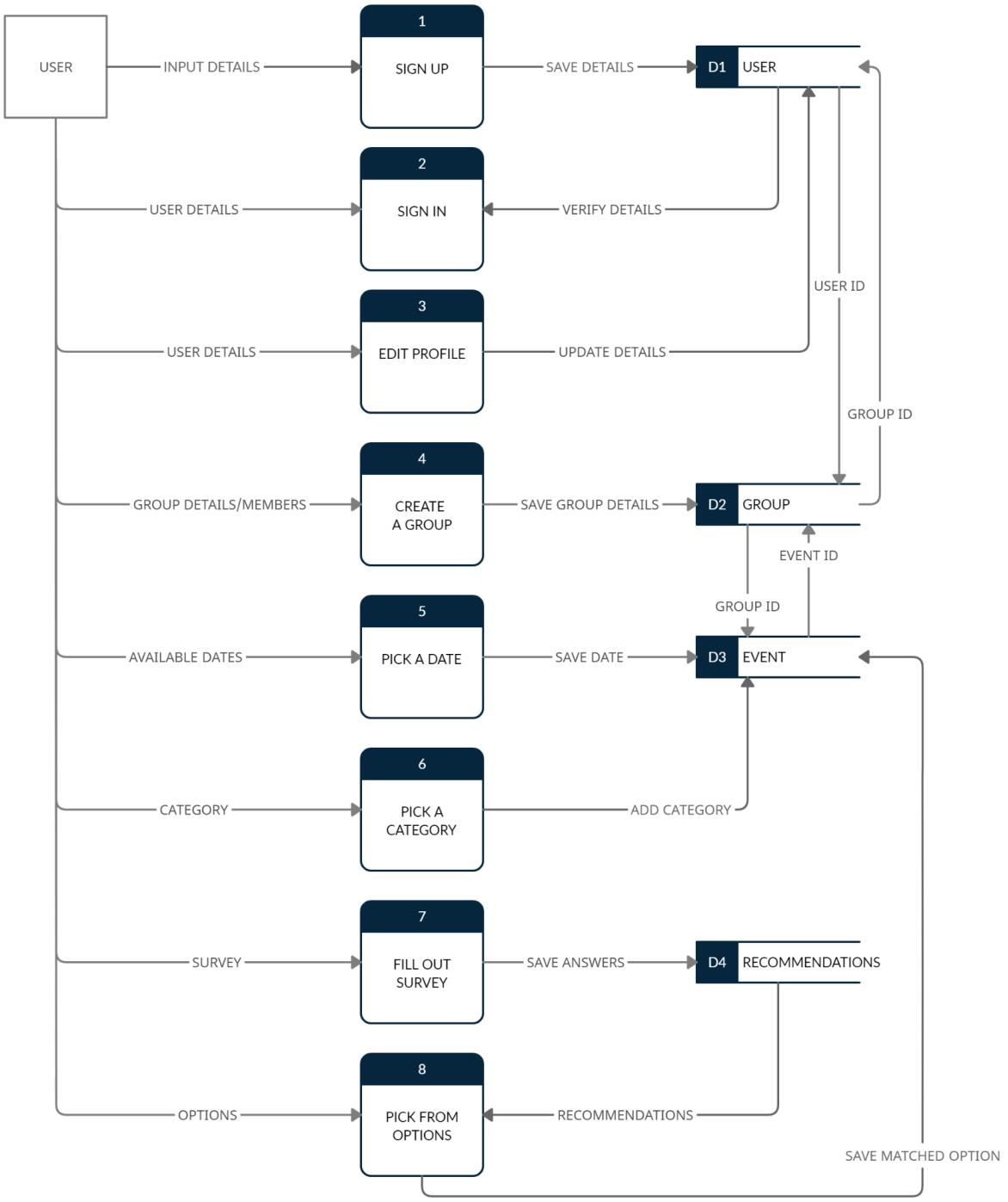
- **Heroku**
  - Heroku is a container-based cloud platform where developers can deploy, manage and scale their apps. This will be used to host our application.

5. High-Level Design

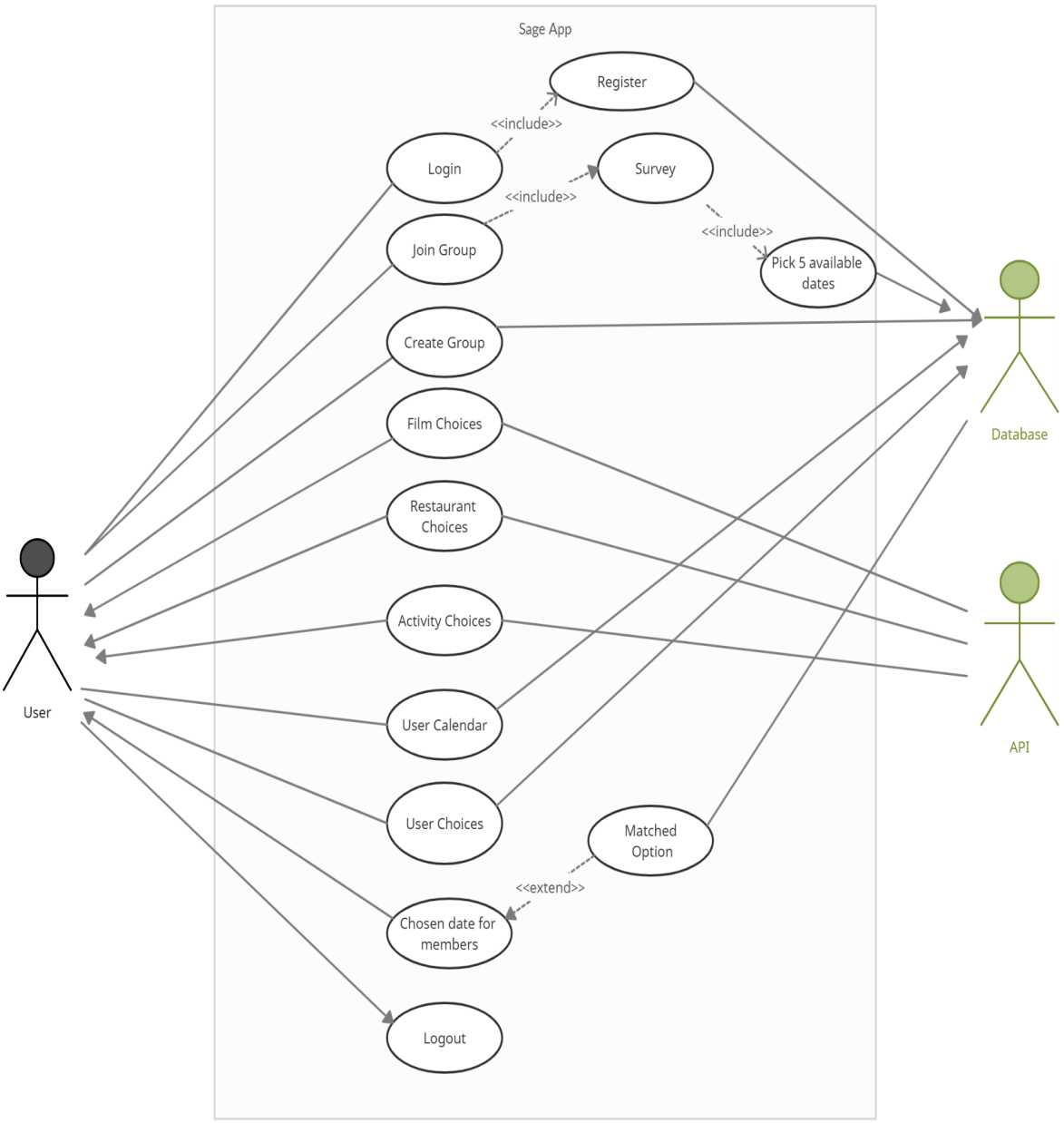
5.1 Context Flow Diagram



5.2 Data Flow Diagram



5.3 Use Case Diagram



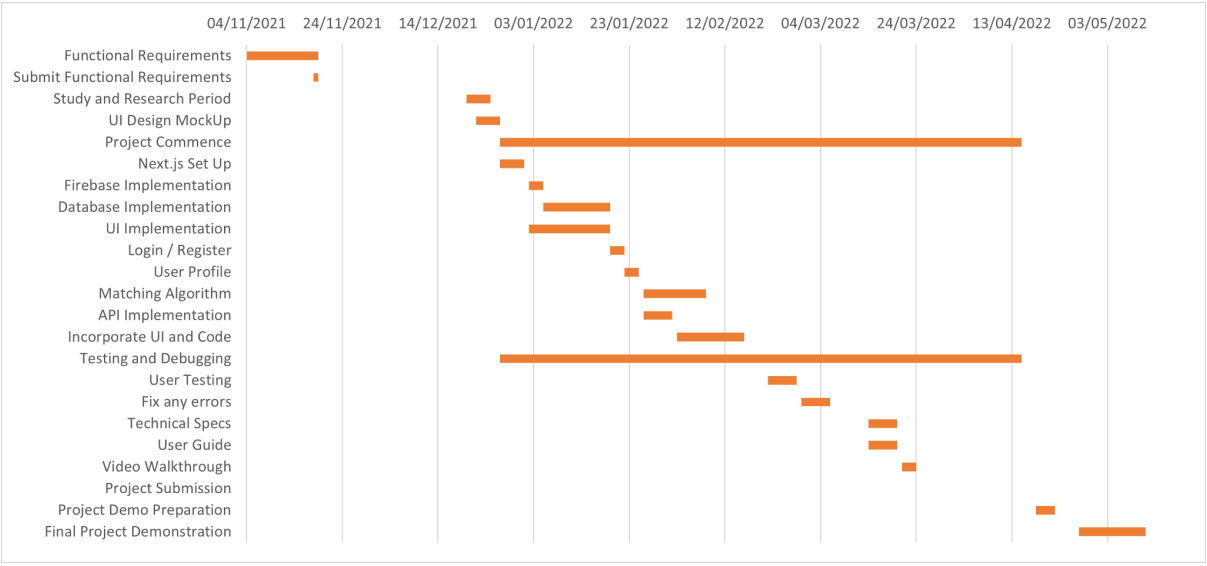
6. Preliminary Schedule

6.1 Task List

TASK	START DATE	END DATE	DURATION
Functional Requirements	04/11/2021	19/11/2021	15
Submit Functional Requirements	18/11/2021	19/11/2021	1
Study and Research Period	20/12/2021	25/12/2021	5
UI Design Mock Up	22/12/2021	27/21/2021	5
Project Commence	27/12/2021	15/04/2022	109
Next.js Set Up	27/12/2021	01/01/2022	5
Firebase Implementation	02/01/2022	05/01/2022	3
Database Implementation	05/01/2022	19/01/2022	14
UI Implementation	02/01/2022	19/01/2022	17
Login / Register	19/01/2022	22/01/2022	3
User Profile	22/01//2022	25/01/2022	3
Matching Algorithm	26/01/2022	08/02/2022	14
API Implementation	26/01/2022	01/02/2022	6
Incorporate UI and Code	02/02/2022	16/02/2022	14
Testing and debugging	27/12/2021	15/04/2022	109

User Testing	21/02/2022	27/02/2022	6
Fix any errors	28/02/2022	06/03/2022	6
Technical Specs	14/03/2022	20/03/2022	6
User Guide	14/03/2022	20/03/2022	6
Video walkthrough	21/03/2022	24/03/2022	3
Project Submission	15/04/2022	15/04/2022	0
Project Demo Preparation	18/04/2022	22/04/2022	4
Final Project Demonstration	27/04/2022	11/05/2022	14

6.1 Gantt Diagram



7. Appendices

- https://firebase.google.com/
- https://firebase.google.com/products/realtime-database
- https://firebase.google.com/products/cloud-messaging
- https://firebase.google.com/products/storage
- https://firebase.google.com/products/auth
- https://web.dev/progressive-web-apps/
- https://developers.google.com/web/ilt/pwa/introduction-to-service-worker
- https://developers.google.com/web/ilt/pwa/introduction-to-service-worker
- https://nextjs.org/
- https://devcenter.heroku.com/start
- https://developer.imdb.com/
- https://openweathermap.org/api
- https://developers.google.com/maps/documentation/places/web-service/overview
- https://cacoo.com/blog/everything-you-need-to-know-about-architectural-diagrams-and-how-to-draw-one/