Software Design Document
Inclusive Solutions

Members:

Connor Kilgore, Olive Price, Monika Beckham, and Ethan Green

Sponsor:

Susan Purrington

Mentor:

Italo Santos

Version 2.0

Feb 17, 2023

Overview:

This document details the design of our software, with a base overview of the communication between modules, and a deep dive into each individual module, along with plans for their implementation.

Table of Contents

Introduction

Navigating a world that does not always cater to one's individual needs can be challenging, particularly when it comes to access to essential public services. Accessibility requirements can vary greatly, ranging from the need for wheelchair access to baby changing facilities, with a universal expectation for safety in daily surroundings. These requests are reasonable and justified. Our initiative aims to alleviate the feeling of inadequacy by creating a secure mobile platform that enables individuals to share and access information about the accessibility of local businesses. This information will assist businesses in better understanding and addressing accessibility issues, which is critical given the current lack of awareness and the limitations of existing technology and applications. Ongoing prototypes have given us valuable insight into what exactly is required of such a project. This gives us the chance to examine key platform development challenges and address underlying issues more thoroughly.

This project falls under the crowd-sourced mobile application sector, dominated by the well-known company Yelp. This app offers customers a way to rate businesses based on information collected from other users. Yelp, as noted on its Wikipedia page, is a billion-dollar company with over 142 million monthly unique visitors. Given the high demand for quality assurance of businesses, a similar platform for ensuring basic necessities could serve just as many people.

In order to make this project possible, Inclusive Solutions is working with Welcomed Here, founded by Susan Purrington. Welcomed Here strives to provide demographic information regarding acceptance, accessibility, and safety about local businesses to the public. Because this organization is non-profit and fairly new, it has no established funding scheme as of yet. They are currently using a *business to business consultation*, meaning they hire informants to conduct assessments of certain businesses. With this information they are able to provide an inclusion roadmap and connect to further resources in the community. The end goal of Welcomed Here is to elevate the reputation of businesses that provide accessibility for all individuals.

The design of this app will include a client-side application on Android, and a server system back-end with a database. This system will allow users to find businesses near them that best meet their specified needs, as well as make reports on each location based on a wide accessibility criteria. Additionally, the application will call services from Firebase Authentication and Google Maps APIs. Features of this solution include but are not limited to:

- Intuitive user interface crafted with accessibility in mind
- Firebase Authentication for secure account creation with encrypted information
- Profile page for editing user demographic information and preferences
- Profile manager to save user data and provide personalized information
    - Stores information about user including their gender, sexual orientation, abilities, and ethnicity with accessibility preferences
    - Takes user specified preferences into account when searching and viewing business locations
- Place search page with Google Maps API and crowd-sourced information
- Option for any user to leave a review or report about any location
- Server-side database to store information about users, locations, and reviews
- Review manager to save reviews and reports and communicate accurate information on locations
- Data about locations and areas wrangled into intuitive reports for clients

Implementation Overview

To reiterate, our project will solve the issue of inadequate accommodations of local businesses by creating an Android application called "welcomed." Our development process follows an agile methodology, where we iteratively create and refine the software through close collaboration with our clients, incorporating their feedback to enhance the overall design and structure at an early stage before major changes become costly. To manage our frontend, we will develop an application on Android OS, using Android Studio as our primary IDE. The front-end language will be in Java and it will communicate with our backend to get information to and from the database, and interpret that data appropriately. The front-end will also communicate with a Firebase Authentication API for encrypted log-in information, and Google Maps API to receive necessary map data. The backend will also be programmed in Java where it will handle incoming and outgoing requests to and from clients to properly manage the database with little maintenance. The database will be coded in SQL and will store all relevant information, with the exception of passwords, which will be handled with Firebase Authentication.

The frontend is composed of various XML elements created to replicate the prototypes made during the initial design phase. These elements should implement the detailed changes advised by our client. The client side application incorporates the use of the dyslexie font, which has a clear baseline and prevents letters from being turned upside down for users with Dyslexia. The application also ensures optimal color and contrast for its theme, to be visible to users with varying degrees of color-blindness. Frontend elements have descriptions to be recognized through screen readers, and each page design passes web-accessibility guideline tests.

The backend consists of a profile and business manager, along with various other modules. Each of these modules are responsible for different tasks, including allowing users to log in or create new accounts and communicating directly with Firebase. They also work to verify if the user has set up their account and retrieve or create their information, manage a user's access on the server-side, handle reviews and business information, and communicate with the client in JSON format. We are then able to retrieve and define the model for representing place data and handle the display of place reviews and tags. Lastly, we handle Google Maps communication, providing autofill suggestions based on user input and generating a map view of nearby reviewed places.

The database for the Welcomed app stores all necessary user demographic information, including preferences saved by each user. This includes accessibility categories such as the need for easy grade ramps or baby-changing stations, which our backend managers use to display businesses containing the corresponding traits. The data is collected through various sources, including user feedback, reports, and reviews. Additionally, the database allows users to create detailed profiles with specific information about their needs and preferences. Our client provided us with specific options for user demographics, which we incorporate into our database. This ensures that businesses and places that cater to a diverse range of users are highlighted in the app. The user feedback system is an essential tool in collecting data for our database, as it provides us with detailed reviews and comments on the accessibility of businesses and places. The backend managers directly communicate with the database to retrieve and display the relevant information, making the user experience more personalized and accessible. Through this approach, the Welcomed app is able to provide a comprehensive platform that accommodates a wide range of user needs and preferences.

Architectural Overview

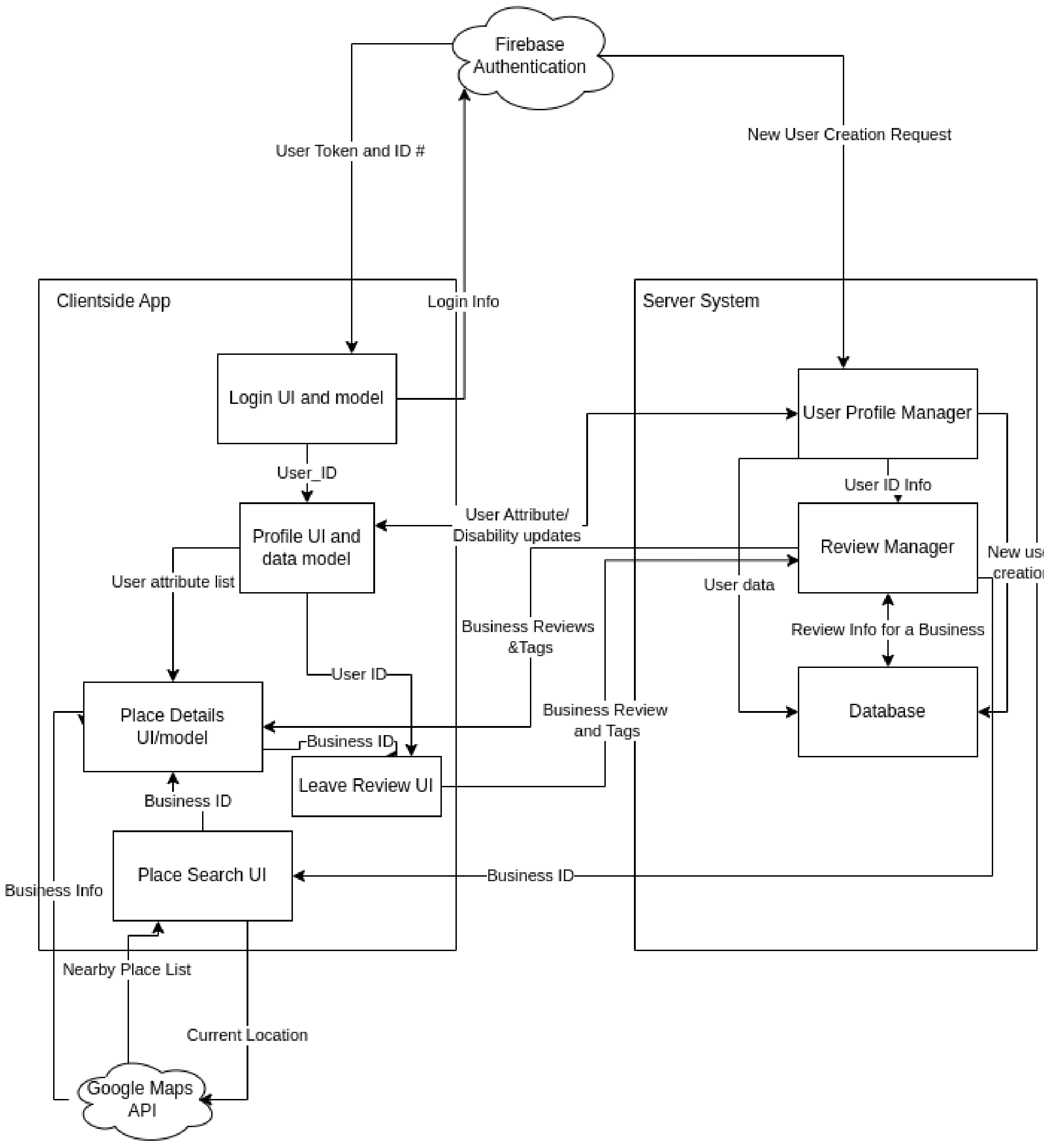


Figure 1. A diagram of our system architecture.

Our solution will use a Client-Server architecture, with the client application being responsible for displaying the data it receives from the server and sending new data from the users to the server. The server, meanwhile, is responsible for processing and storing the data to be sent to client instances to display. Our client will consist of six major modules, which will communicate with our server components that interpret the data for the database and the client.

The Login Module is responsible for allowing users to log in or create a new account with welcomed. This module is the only module to directly interact with Firebase, the rest will use this module's public attributes to get the user's authentication token and user ID. This module then passes the user ID to the Profile model to get or create the user's info.

The Profile Module takes the auth token and user ID, and checks if the user has set up their account, pulling the user's attributes, identities, and accommodation needs if they exist. If they do not exist, this module will open the Profile UI and instruct the user to select their attributes, identities, and accommodation needs, and then it will send this to the User Account Manager on the server end to store for later use. From there, the user is then able to fully use the app.

The User Profile Manager Module is responsible for establishing a user's access on the server side and for storing, retrieving and resending a user's attributes, identities, and requested business tags. It communicates with the client using the Moshi library's JSON serializer and deserializers to convert to and from JSON for receiving and sending information.

The Review Manager Module is responsible for inserting and reading reviews and business information, sending the information to and from the client and database. It receives the review as JSON from a message to the client, and it sends back a list of the reviews for a business with a given Google Maps Place ID in a similar JSON format, using the Moshi Java-Object to JSON converter

The Place Details Module is responsible for handling the display and definition of the "Place Details" that make up the core of welcomed.'s purpose. It defines the model for representing place data that we collect, as well as pulling the parts we need from Google Maps. This module also handles how we display the reviews and tags of a place to the user.

The Place Search Module is responsible for a lot of Google Maps communication for both autofilling a place based on the user's text input in the search box, and for generating a “map view” of nearby places that have been reviewed in a Google Map.

Module and Interface Descriptions

Login Module

The login module will be the primary liaison between our Firebase server, the local application, and our database server. It is responsible for creating new authentication tokens and validating existing ones. The primary goal of this module is security - ensuring that access to an individual user’s information is limited to that specific user.

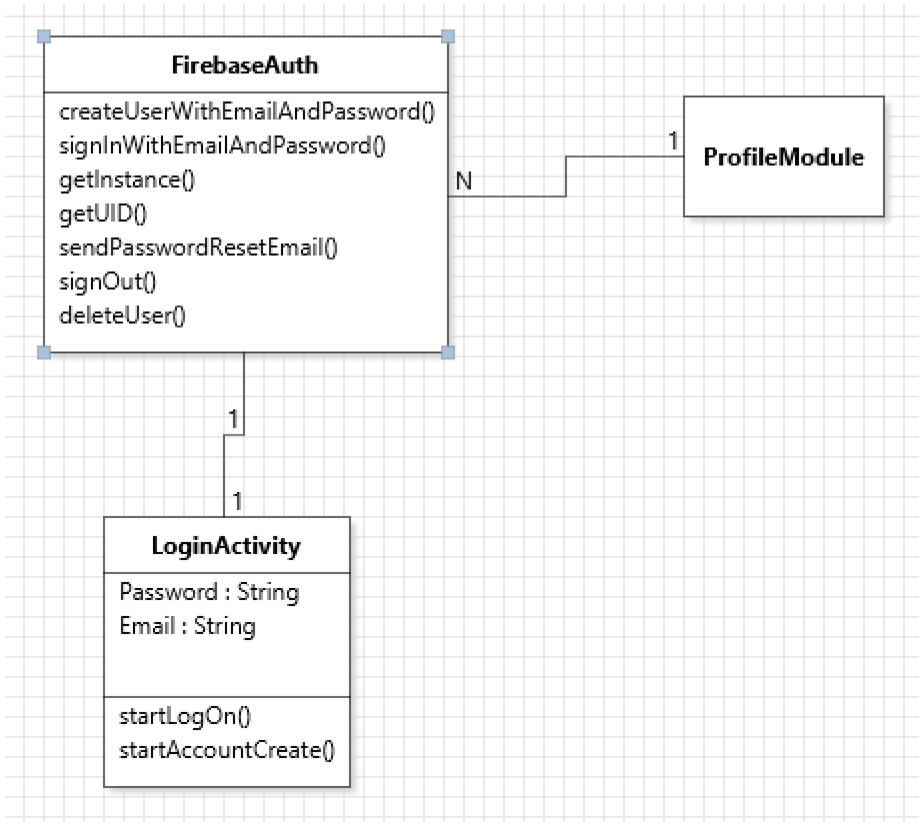


Figure 2. Diagram relating Authentication, LoginActivity, and ProfileModule.

The login module will be what handles the initial login or account creation of a given user. What this will be a majority of the time is the user typing in login credentials to the front facing interface and pressing a “log-in” button that confirms it to be passed to the signInWithEmailAndPassword public method in an instantiated FirebaseAuth object personal to that session. This will then check against our account database on the Firebase console. If no exceptions are raised, then the login module will make available the necessary information to create the account in our database and pass responsibility on to the profile module.

Profile Module

The profile module is responsible for querying the database and profile manager module to retrieve a user's information or ask the user to input it as necessary. It essentially acts as the last gateway to the application proper to ensure a user's session is fully set up.

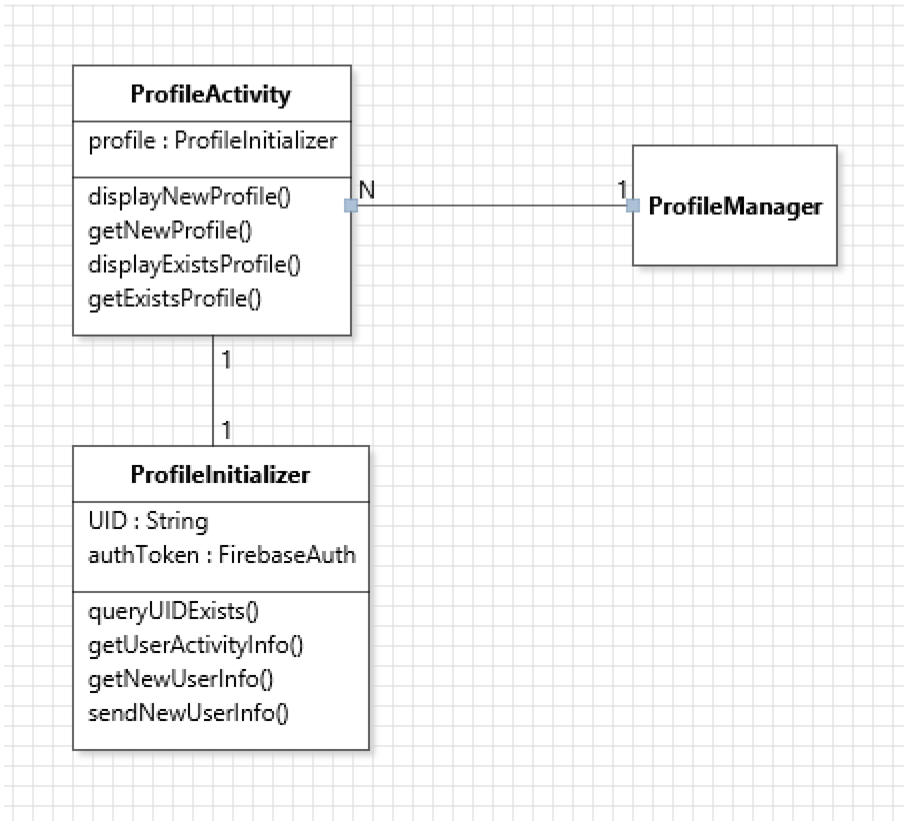


Figure 3. Diagram relating ProfileActivity, ProfileInitializer, and ProfileManager.

ProfileActivity acts as the main display for this step of the application. It uses a subclass to receive and send information to the profile manager and thus the database for the user session verified by Firebase. If a profile does exist for a user, this activity will display based on that information. Otherwise, the activity will open a list of prompts for the user to input their personalized information for their user profile and send it to the profile manager for verification.

User Profile Manager Module

The user profile manager will be responsible for communicating data between the client and the database with minimal maintenance. This will allow the profile model to send information about the user's demographics to the database, and allow the database to send back personalized information to the user based on their demographics. This will also act as a middleman between the Firebase Authentication and the database to store the encrypted data.

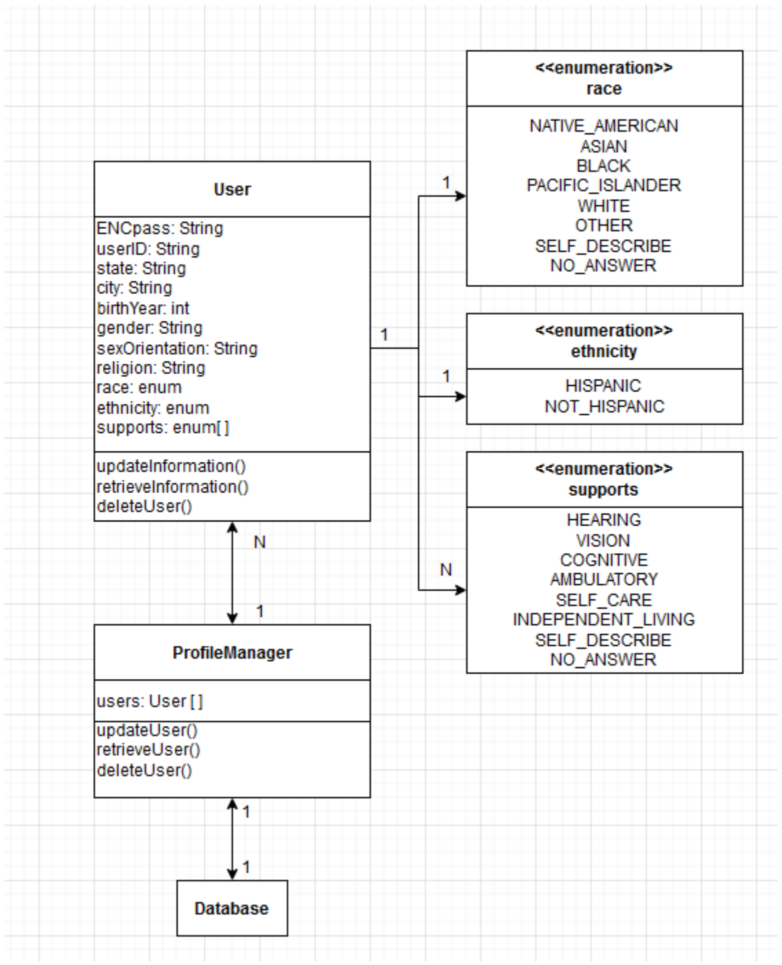


Figure 4. Entity diagram relating User to demographics with Profile Manager and Database.

The most important component of the module is the ProfileManager class which acts as an intermediary between the user and the database. A user will be able to request their information wherein the profile manager will retrieve the information from the database by returning a User. Additionally it allows the user to update their information or delete their account if requested. The class User holds a lot of information regarding demographics as well as login information. Some attributes are enumerated as shown on the right side enumerations.

Review Manager Module

The review manager is very similar to the user profile manager except that it pertains to reviews. It will communicate data between the client and the database to update and retrieve the requested information about a specific review.

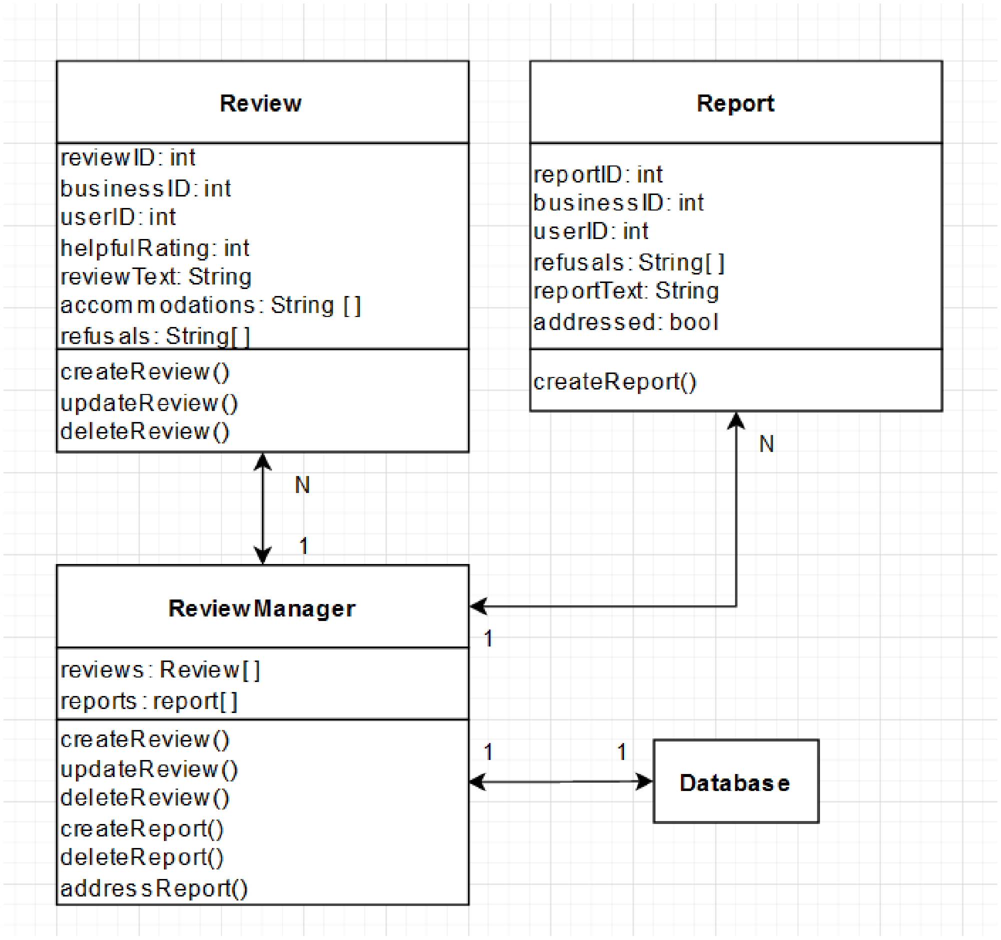


Figure 5. Entity diagram relating Review, ReviewManager, Report, and Database.

Like the user manager, the most important component is the review manager. It will act as a middleman between the client and the database to handle dataflow. It has both a review and a report class which are very similar in attributes and functions. A review can create, update and delete a review. The review manager will then send that data to the database which it can also do with reports. This will allow the place search model to check for reviews and reports on specific locations before sending the information to the client UI.

Place Details Module

The place details will be responsible for pulling up relevant information on a location selected by the client. It will grab relevant information about the location from the database and put it into a readable fashion for the user. When the information is put up, it will also be responsible for requesting information from the user to generate a personalized report of the location based on the user's demographics.

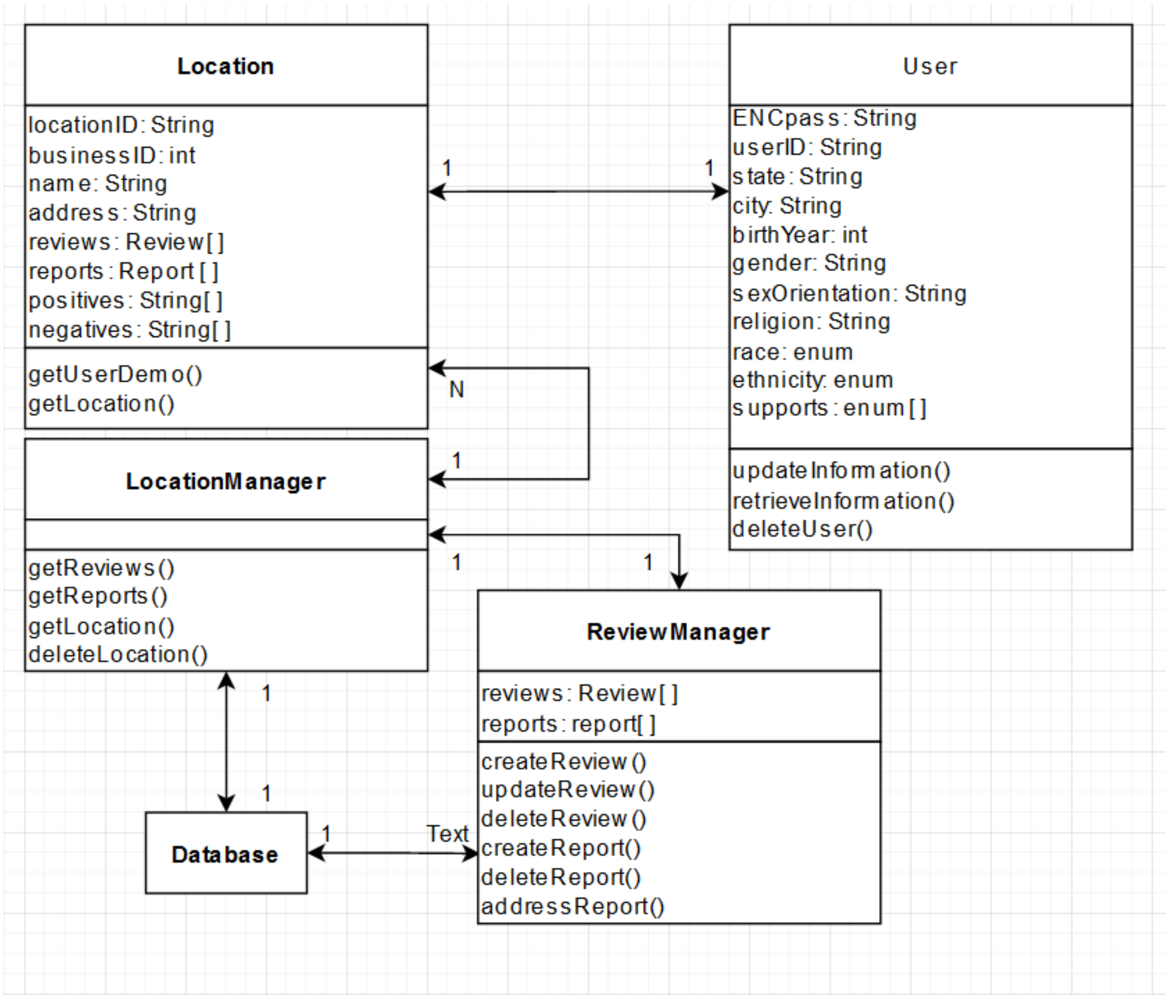


Figure 6. Entity diagram relating Location, LocationManager, User, ReviewManager and Database.

The most important component here is the location manager which is in charge of getting the relevant information to display in the location description UI. The location manager will communicate with the review manager to get the reviews and reports associated with a specified location. It will also get the location data such as its address and name. The location class will be able to communicate with a user to display data relevant to the users demographics and all the information received will be put into a UI that makes it easily understandable to a user.

Place Search Module

The place search module will handle the front end components of searches. Its main purpose will be to make the initial queries to the Google Maps API and the place details module. It will take the results of these two sources and display them in a user friendly and expected manner via a map interface.

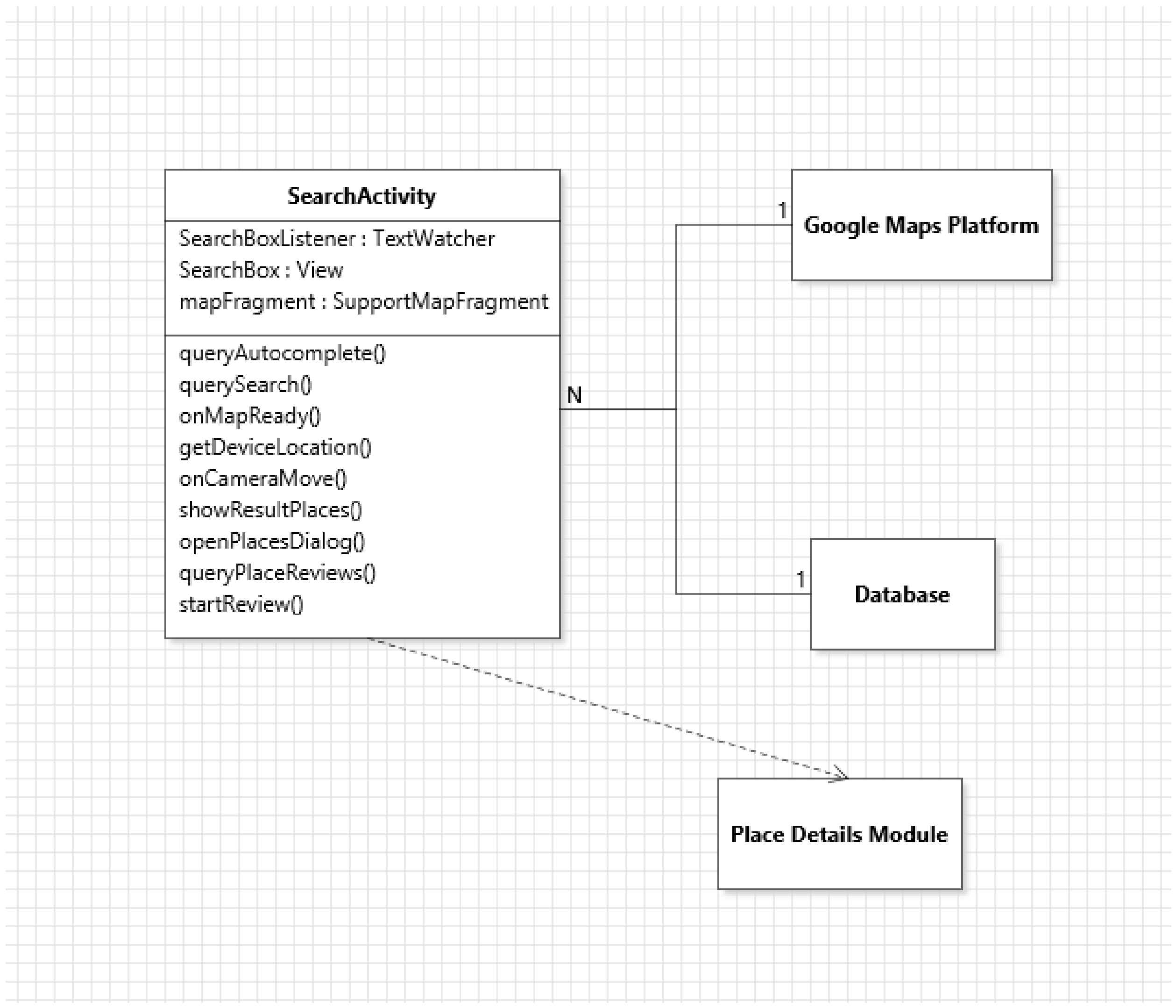
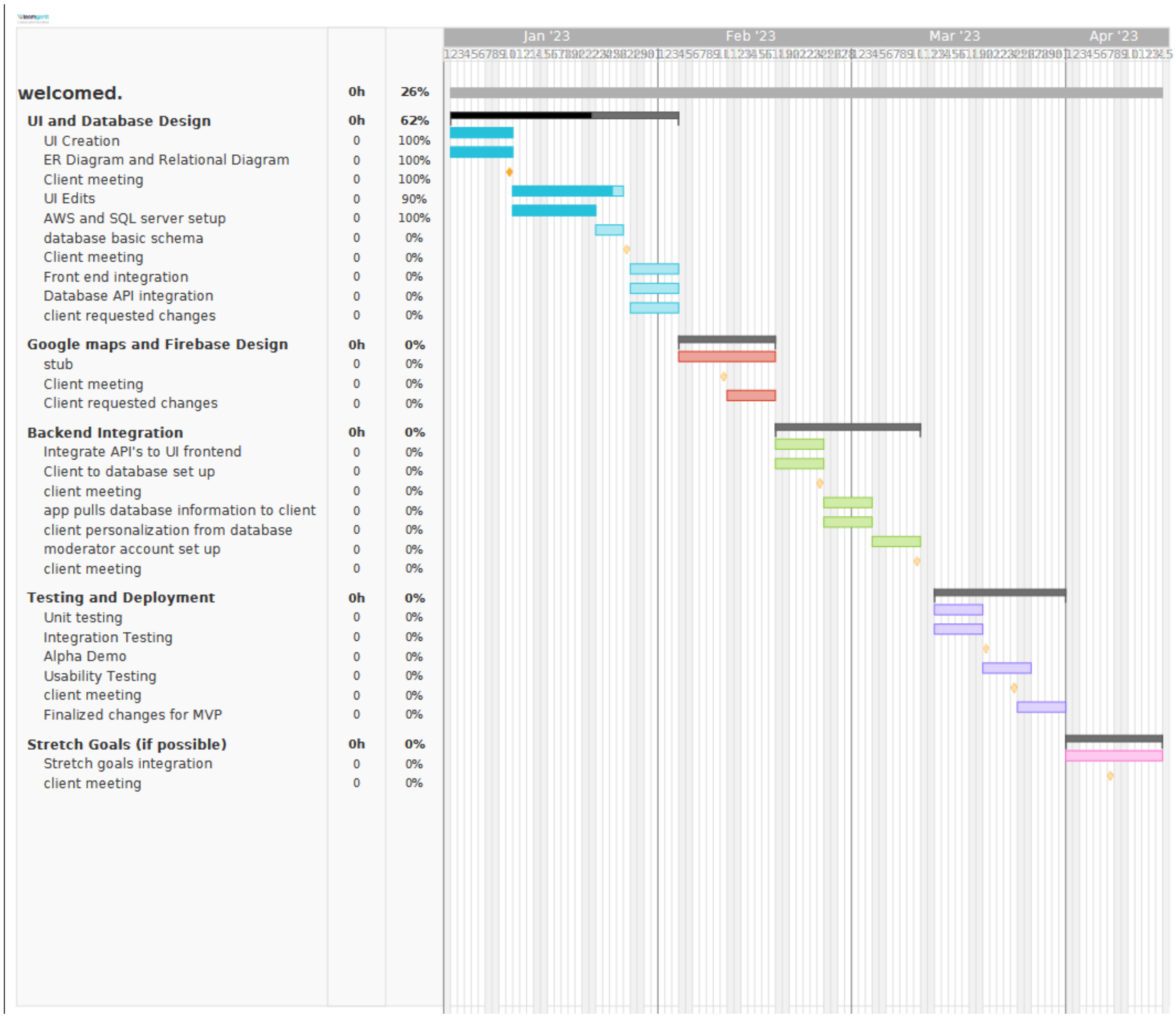


Figure 7. Diagram relating SearchActivity, Google Maps, Database, and Place Details Module.

SearchActivity will be the main controller for what is shown to the user and making the necessary queries. Worth note in particular is the SearchBoxListener, which will call queryAutocomplete any time the user edits the text box. By doing so, it can query Google’s autocomplete engine and suggest results on partial text entry. There are also various auxiliary functions which handle general expected mobile map interface operations, such as centering on a device’s location or enabling map scrolling.

Implementation Plan

Shown in the Gantt chart below is a rough outline of our planned implementation timeline for our project. As can be seen, our first priority has been to implement foundational aspects of our project such as establishing our database and creating a functional frontend.



welcomed.	0h	26%
UI and Database Design	0h	62%
UI Creation	0	100%
ER Diagram and Relational Diagram	0	100%
Client meeting	0	100%
UI Edits	0	90%
AWS and SQL server setup	0	100%
database basic schema	0	0%
Client meeting	0	0%
Front end integration	0	0%
Database API integration	0	0%
client requested changes	0	0%
Google maps and Firebase Design	0h	0%
stub	0	0%
Client meeting	0	0%
Client requested changes	0	0%
Backend Integration	0h	0%
Integrate API's to UI frontend	0	0%
Client to database set up	0	0%
client meeting	0	0%
app pulls database information to client	0	0%
client personalization from database	0	0%
moderator account set up	0	0%
client meeting	0	0%
Testing and Deployment	0h	0%
Unit testing	0	0%
Integration Testing	0	0%
Alpha Demo	0	0%
Usability Testing	0	0%
client meeting	0	0%
Finalized changes for MVP	0	0%
Stretch Goals (if possible)	0h	0%
Stretch goals integration	0	0%
client meeting	0	0%

Figure 8. A Gantt chart of our 8-week plan.

We decided to begin with front end development due to our established understanding of each module. A large portion of our functionality will depend on event listeners to be attached to various buttons, text boxes, and other such user interface elements. Furthermore, the demonstrations we developed to prove our technologies to be implemented gave us a very robust understanding of each element our team members decided to focus on - detailed in the table below. This gives us the capability to quickly implement the working features of our demonstrations into our developed application in such a way that should not conflict.

Connor	Monika	Ethan	Olive
Database	Firebase API and backend	Google Maps API and backend	User Interface designs and actualization

Creating a user-friendly interface that meets the needs and preferences of our users is a critical step in developing any successful app. For this platform, we prioritized collaboration with our client to ensure that we incorporated her feedback into the design process. This approach allows us to make adjustments and improvements to the user interface in the early stages, before implementing major changes becomes difficult and costly. In addition to designing the user-facing system, we also mapped out the full database, creating relational diagrams and basic schemas. This process helped us to determine the relationships between different components of the app and ensure that the database was structured in an organized and efficient manner. We then took the necessary steps to set up the AWS and SQL server, which forms the backbone of the database API. Google maps and Firebase are then implemented.

Once the frontend elements of our app have been fully realized and the foundation has been laid, we turn our attention to the critical task of backend integration. This complex and multifaceted process will take 4 to 5 weeks to complete and requires the seamless integration of our APIs with the frontend interface, as well as the implementation of client-to-database functionality and server-side calculations. As one of the key objectives of our backend integration is to deliver personalized recommendations to each user, this requires the analysis of user preferences to suggest businesses that match these preferences. In addition to these critical tasks, we also use this time to set up moderator accounts, which allow us to maintain the integrity and quality of our platform by moderating reviews and ensuring that they meet fair standards. In order to deliver a high-quality yet intuitive platform in our rigorous backend integration process, we must ensure that every element of our app is working together efficiently.

As we move forward in the development of our platform, we recognize that testing and deployment are absolutely critical to the success of our project. During the unit testing phase, our team tests each component of the platform individually to ensure that it is functioning as intended. This is followed by integration testing, which involves testing how these individual components work together as a whole. The alpha demo stage allows us to demonstrate the platform to our capstone class, gather feedback, and make any necessary improvements. Finally, usability testing provides us with insight into how real users may interact with our platform, allowing us to identify any issues or areas for improvement. Throughout the testing and deployment process, our team is committed to delivering a polished, user-friendly platform that meets the needs of our client. We also remain focused on our stretch goals, which represent the vision and ambition that our sponsor has for this application. With any extra time, we plan to explore new features and functionalities that will help us to continue to improve and enhance the platform over time.

Conclusion

In summary, this document has detailed our planned design implementation choices on the client side, server side, and between to accomplish our project goal of developing a sufficient and user-friendly application to source business inclusivity data for our client. We plan to leverage existing application package interfaces, our own personally developed user interface designs, and database to accomplish this goal in a timely manner and deliver a polished application to our client that can be expanded into a fully functional product. By detailing our plans in this document, we have a good foundation and guide as to how each part of our application fits together to create our anticipated product. The collaboration between Inclusive Solutions and Welcomed Here is a significant step towards making businesses more inclusive for individuals of all backgrounds and abilities. With the help of Welcomed Here, this project aims to provide a reliable and up-to-date source of information regarding the acceptance, accessibility, and safety of local businesses.

The design of the app is user-friendly, with accessibility being a top priority. The use of Firebase Authentication and Google Maps APIs ensures the security and accuracy of user information, while also providing personalized information and business recommendations based on user preferences. With a profile manager to save user data, users will be able to save their demographic information and accessibility preferences, enabling the app to suggest businesses that meet their specific needs. The app's features are numerous, including a place search page with crowd-sourced information, an option for users to leave a review or report on any location, and a server-side database to store information about users, locations, and reviews.

This app is a significant step towards inclusivity, and Inclusive Solutions is proud to be working with Welcomed Here to make this happen. Through the development and implementation of this app, we are one step closer to making businesses more welcoming and accessible to everyone, regardless of their abilities or background.