

“For The People” Detailed Design

JIC Team 2337

Ivan Alkhovik, Emily Eby, Lucas Cowden, John Gunner, Lakshman
Ravoori

Mamie Harper

Github: <https://github.com/lravoori/JIC-2337>

Table of Contents

[Terminology](#)

[Introduction](#)

[System Architecture](#)

[Data Storage Design](#)

[Component Detailed Design](#)

[UI Design](#)

Terminology

Admin Account: Account that allows those with admin usernames and passwords to access information about all users and businesses (identification removed).

Back-end: The segment of the application that the user does not interact with and handles data.

Business Layer: The business layer contains the majority of the application's logic. Once a user makes a decision or inputs via the presentation layer, this information is passed into the business layer which decides how to handle the information. For example, if a user presses a button on the login screen to request to make a new account, the presentation layer passes this into the business logic, which decides the screen needs to switch over to a new app page. It also can retrieve or send information into the database layer, which stores data on all of the accounts, businesses, etc. The business layer tends to be built across the same files as the presentation layer on a per-page basis. This allows us to keep our code organized based on which app page we should be currently looking at.

Dart: Language which utilizes libraries and pre-made features to develop web and mobile applications.

Database Layer: The section of the application which contains the database that stores all required information.

Firebase: NoSQL database where all of the user, business, and account information is stored securely. Information is stored in JSON documents.

Flutter: Platform used to display applications made using the Dart language.

Front-end: The segment of the application that the user interacts with.

Google Cloud: Utilizes Google's cloud for data storage and computing power.

JSON: JavaScript Object Notation which is used for data exchange.

NoSQL: Database in which information is not stored in a table. Instead relations and other information is stored in different ways.

Presentation Layer: This is the section of code in our software dealing with what users of the application will see. The presentation layer is where we control what is displayed on the screen of a user's phone, such as the buttons, images, and business information. Additionally, this presentation layer tends to be broken up on a per-page basis. In other words, each app page (such as the search page, registration page, login page, etc.) tends to have its own file of code associated with it. Finally, the presentation layer takes in user input through text fields, button clicks, etc. and passes it into the business layer.

SQL: Database in which information is stored in a table.

Tech Stack: Combination of front-end, back-end, database, and cloud functionalities in order to create an application.

UML: Unified Modeling Language which provides a standard for diagrams, intending to make them easier to understand.

User Account: An account that a regular user will use to interact with the app. It will not allow users to engage in admin actions and will only be able to access its own information.

Introduction

Background

Our project is For the People and we are Team 2337. For the People is an idea thought up by Mamie Harper, of Carrie's Closet, in order to provide networking and community to disenfranchised businesses (such as women, POC, LGBTQ). Currently, many of these businesses struggle to gain support from the community as well as funding and other support. Our app is intended to allow users to search for these verified businesses in order to better support them. In doing this, we will have three account types: users will search for businesses and leave reviews, businesses will input their information, and the admin will regulate the businesses and ensure the verification process is being followed. This document will cover the major architectural and design structures that are present within the proposed application. Specific components, UI, and data goals will be specifically discussed and supported using diagrams and images that demonstrate the application structure.

Document Summary

The [*System Architecture*](#) section details the stack that the team utilizes as well as details on the used Layered Architecture.

The [*Data Storage Design*](#) section explains the usage of Firebase as the database and other data concerns.

The [*Component Detailed Design*](#) section goes over static and dynamic interactions between the components in the system.

The [*UI Design*](#) section details user experience and interaction choices, and how it affects the system as a whole.

System Architecture

This section will go over our system architecture, and why we chose to use the options that we did. We decided to go with Flutter for our front end as it allows us to develop an application for both iOS and Android. It is also done in the Dart language which is similar to Java. As all of the team is familiar with Java, this was seen as a large advantage. The other options we considered were ReactNative, Android Studio, and Swift UI.

For our implementation, we decided not to utilize a separate backend and utilize Flutter for both the back end and front end. Flutter utilizes the Dart language, which is able to perform backend functions entirely on its own. Additionally, because Flutter integrates directly with Firebase, we can cut out the need for additional software to perform calls to the database.

We decided to use Firebase for our database due to the ease of interaction with both Flutter and Google Cloud. It is because of this ease of use that we are using Google Cloud rather than attempting to integrate with other Cloud systems. In addition to this, Firebase is easy to scale as the business grows, this is something that our client expressed a need to do. NoSQL is also faster and more user-friendly which would be beneficial when we pass the project off to our client who is less proficient in technology. Other NoSQL platforms we considered include MongoDB.

Static

The architecture that our system is using is Layered Architecture. We chose this because of the clear divide between the UI, the code controlling all of our main systems, and our database which is run by Firebase. With this, user interaction trickles down the layers without having to directly interface with them. Also, we thought there was a large benefit to having the database be a separate layer. This way, changes in the database later on that are anticipated will not require any of the code to change. This is extremely important as our client is anticipating a large variability of users and businesses and our database will need to be flexible with this.

Our Static System Architecture diagram (Figure 1) demonstrates how our main features will function in the Layered system. We only have a mobile application as our view. This directly interfaces with our main functionalities of User Authentication/Registration, Business

Registration, Admin Management, and Business Operations. These business layer functions will interface with User Data, Business Data, and Application Data which exists in Firebase.

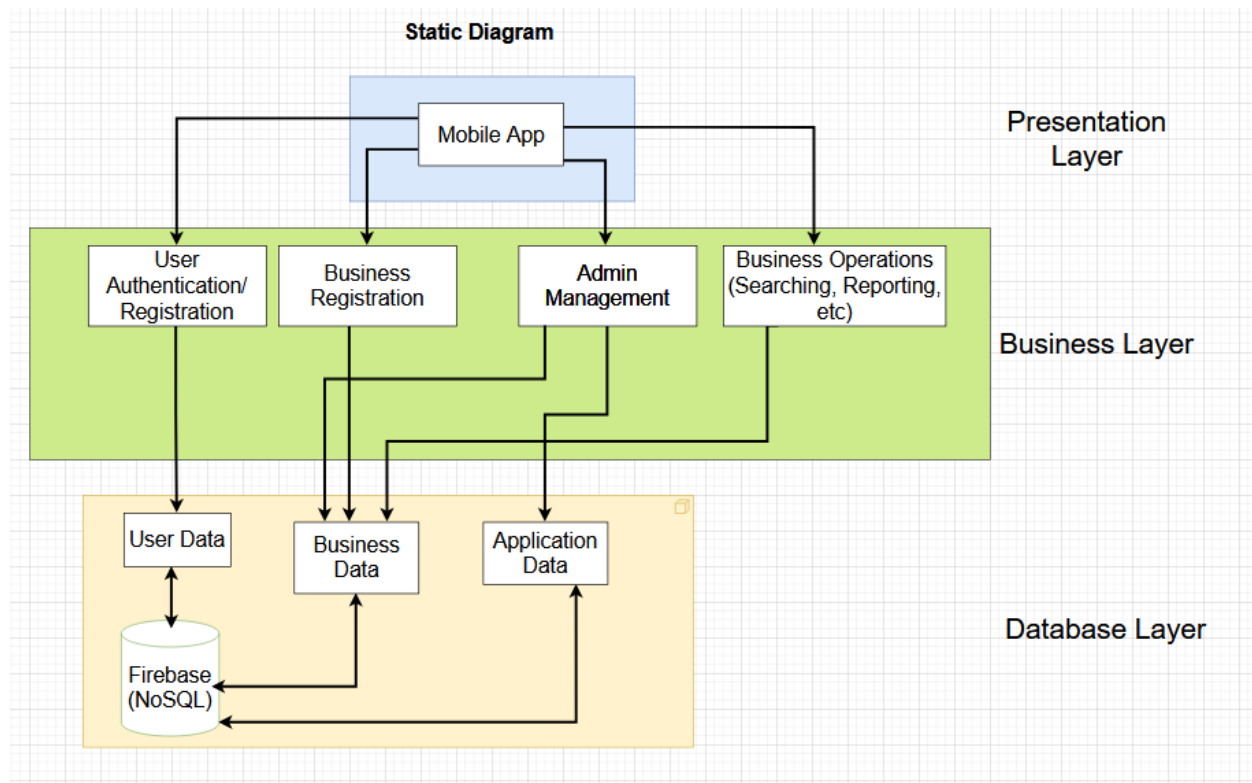


Figure 1. Static Diagram which illustrates the system structure of our application in the Layered Architecture.

Dynamic

Our dynamic diagram is intended to illustrate how this Layered Architecture will work in user interactions throughout the application. We are using a System Sequence Diagram (Figure 2) in order to detail how each of the parts are going to interact with each other during runtime. This diagram is based off of the user, interacting with each of the control components, which then get the data from Firebase.

View App Statistics

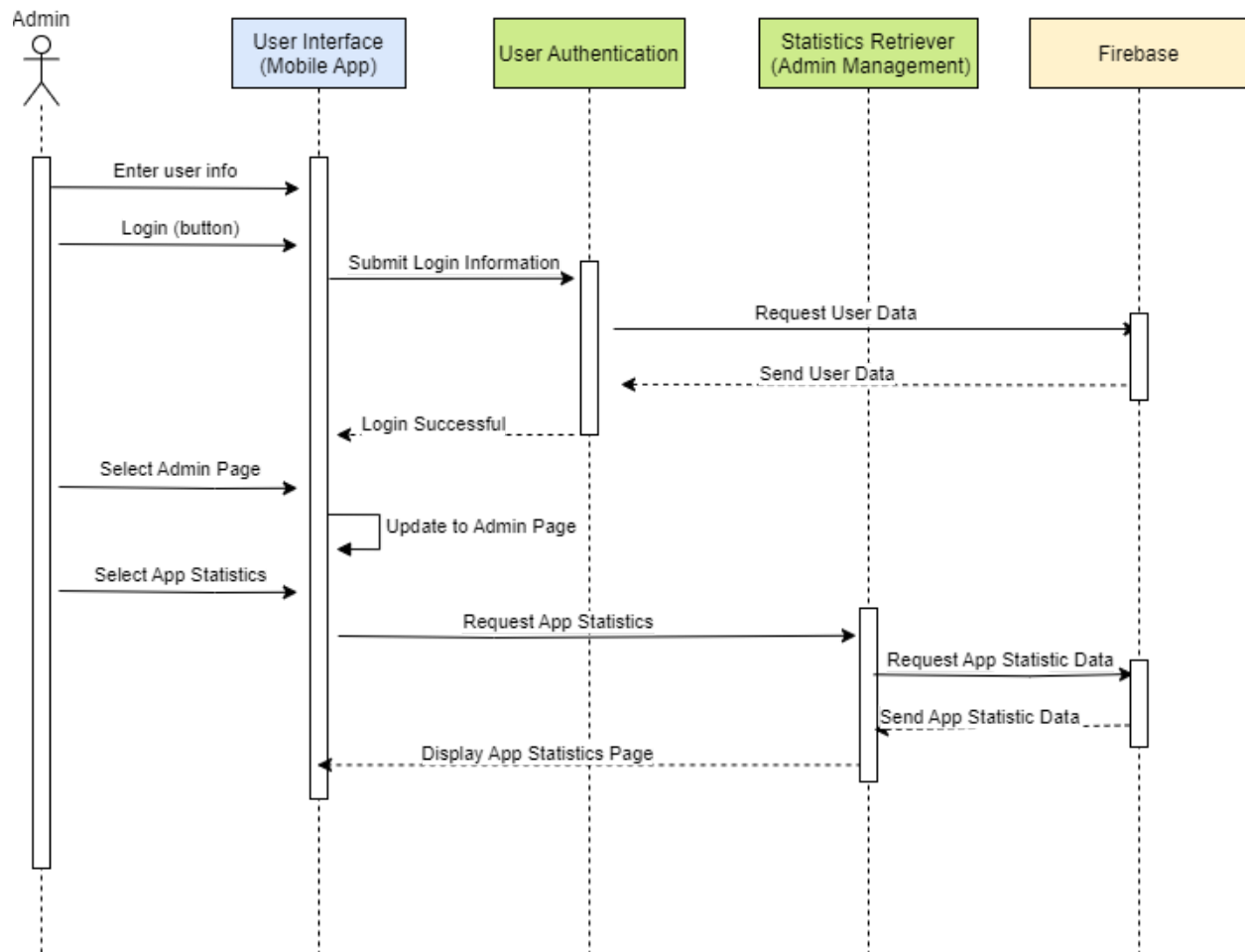


Figure 2. System Sequence Diagram illustrating how the Layered Architecture works when a user interacts with the system.

Data Storage Design

Intro

The database that we are using is Firebase. The main features that we are using with the database is Cloud Firestore which is NoSQL, user authentication, and Google Cloud Storage for image storage. These all integrate with Firebase and do not require any external plugins. Firebase also connects with Flutter and holds extensive documentation on how to use it. This was the main reasoning for utilizing Firebase as it greatly simplified our tech stack.

Database Use

Cloud Firestore is a NoSQL document-based database which creates “documents” which are identified by a key and have key-value attributes. We have databases for the businesses within the application, the users, and the authentication information. For user authentication, Firebase will store the email address and the password of the user in its dedicated authentication service database. This database will then create a UserID which will be used as the user’s identifier. Within the business and user information databases, documents will be identified by their name, which will be a unique ID, and will contain the following information:

Business Database:

Key (Name)	Document
BusinessID (autogenerated)	<pre>{ "Street Name": String "State": String "Zipcode": Integer "Owner's Name": String "Owner's Gender": String "Owner's LGBTQ+": Boolean "Owner's Race": String "Business Details": String "Category": String "Hours": String "Logo": String (file path for Cloud Storage) "Business Name": String "Phone Number": String "Website": String "Ratings": Map<String(UserID), Integer> "Verified": Boolean "Flag Count": int "FlagReasons": Map<String(Reason), Integer> "UserID": String }</pre>

User Info Database:

Key	Document
UserID (autogenerated)	<pre> { "Name": String "Gender": String "LGBTQ+": Boolean "Race": String "isAdmin": Boolean "Age": Int "BusinessIDs": { "BusinessID": String ... } }</pre>

File Use

Google Cloud Storage will be used to store various logos as images. These images will be stored as standard image formats (.jpg, .png, etc.). Once uploaded, these images will be named as "[BusinessID]_logo" which will provide identification to the stored images and uniqueness to their names. Files will not be stored on users' mobile devices.

Data Exchange

All data is transferred to the Cloud Firestore database which is pulled by each individual device or usage of the application. Users will only be able to access the data that they enter in. The Admins will be the only ones able to access app statistics. Firebase utilizes WebSocket communication protocol that uses data synchronization instead of HTTP communication. The information is stored in the Cloud Firestore database as a JSON - like format. The frontend of the application will pass the information to the database directly. The Firebase User Authentication we are using utilizes OAuth 2.0 and OpenID Connect.

Security

User authentication is done by Firebase as one of the utilized features. This is integrated into the login and logout process in order to have users authenticated and unauthenticated. This will also ensure that username and passwords are not stored as plain text in the database. Instead, a randomly generated ID will identify the user. No financial data will be collected or utilized. Personal identifiable information will include name, gender, race, if they are LGBTQ+, and age. This will be stored in the database and will only be accessible when the user is authenticated. Only this authenticated user, and admins should be able to access this information. A disclaimer that admins have access to this information will be utilized before creation of the account. In terms of business information, this is all publicly accessible so there are no known security issues.

Component Detailed Design

Introduction

The main components of our system include our front-end which is made up of business search, business information, business filtering, and user profile, as well as our backend which includes user and business databases. This aligns with our layered architecture which has the front-end, back-end, and user on separate layers.

Static

The Component Diagram shown below represents the static detailed components of the system and their interaction with each other. This diagram offers a high level overview of the system that breaks the application into multiple distinguishable components.

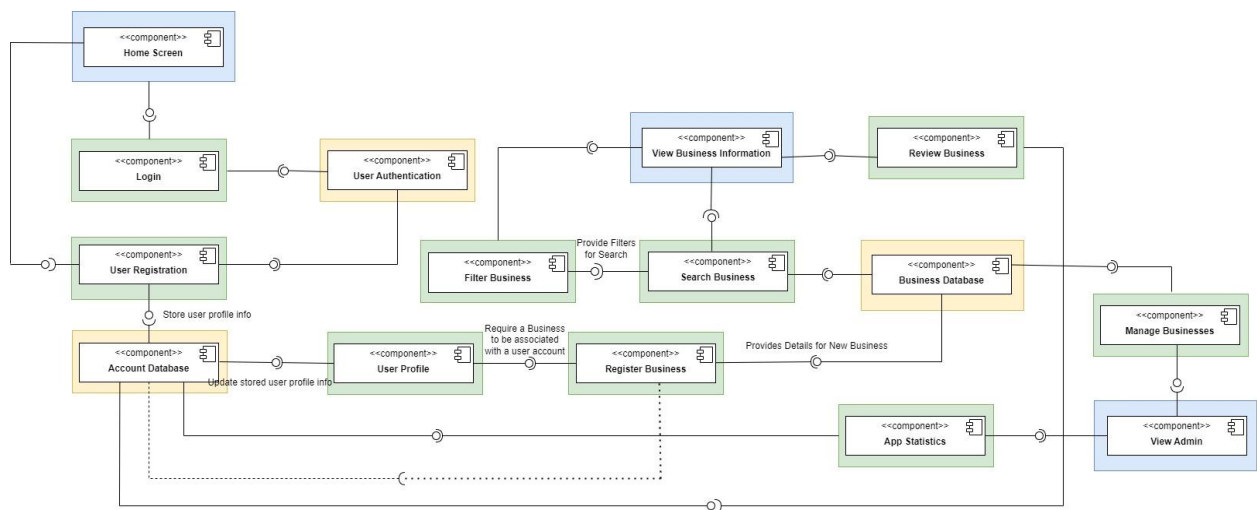


Figure 3. Static Component Diagram that shows the various components of the system and their relationships with each other.

Dynamic

In order to show the dynamic component design we have decided to utilize the Interaction Overview Diagram. This diagram illustrates how the interactions exist between the user and the system during runtime. The user is first presented with a login page where they can either register or login. If they do not have an account, they are prompted to enter their information and register. This information will be sent to the Firebase database for User Authentication. From here users will be shown filters which can be used to determine what businesses they are shown. They can then decide to view all businesses or view their account information that is stored in the database. From their account, they can then decide to create a business which directly inserts into Firebase.

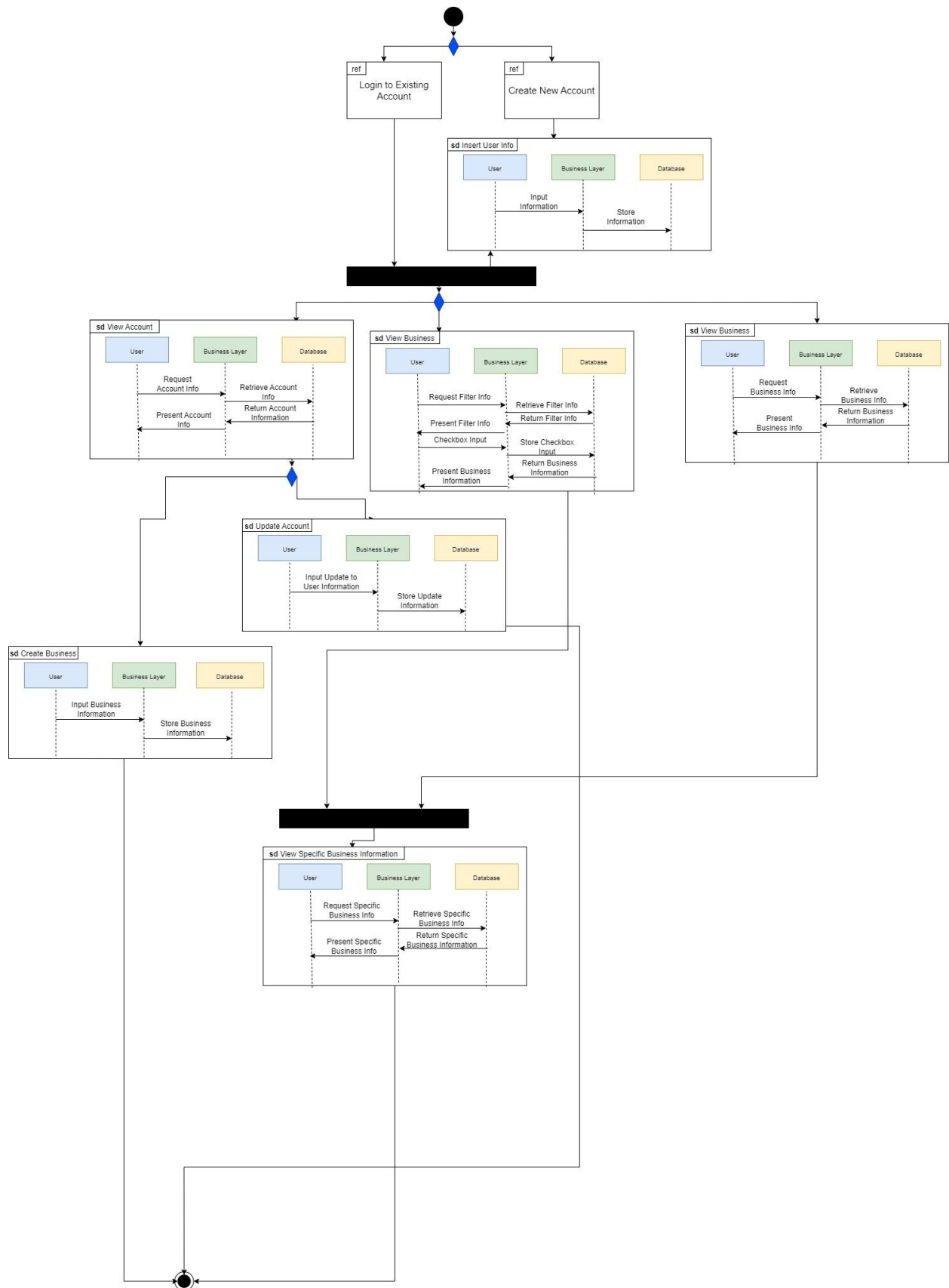


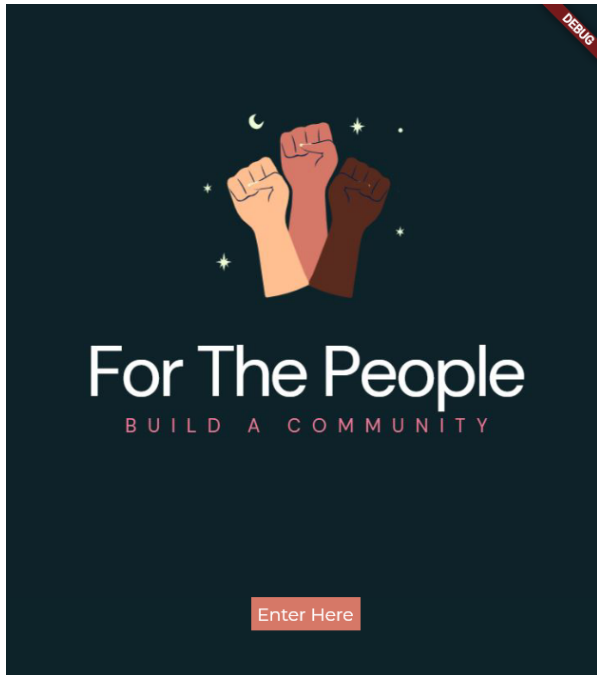
Figure 4. Dynamic (Interaction Overview) Diagram that shows user interactions with the system. Here our system is labeled as Database as this is where the information processing occurs.

UI Design

We are utilizing Flutter with the Dart language in order to complete our front end and UI design.

We have two account types: user and admin which present different screens according to the account. The major screens for each of the account types are shown below.

User:



When the user opens the application, this is what they will be presented with. Enter here is how the user logs into the app

The screenshot shows a web browser window with the title 'We The People' and the address bar displaying 'localhost:51323/#/'. The page content is a 'Sign in' form. At the top, it says 'Sign in' followed by a link 'Don't have an account? Register'. Below this are two input fields: 'Email' and 'Password'. To the right of the 'Password' field is a link 'Forgot password?'. At the bottom of the form is a 'Sign in' button. A red 'Demo' banner is visible in the top right corner of the browser window.

The screenshot shows a web browser window with the title 'We The People' and the address bar displaying 'localhost:50591/#/'. The page content is a 'Register' form. At the top, it says 'Register' followed by a link 'Already have an account? Sign in'. Below this are three input fields: 'Email', 'Password', and 'Confirm password'. At the bottom of the form is a 'Register' button. A red 'Demo' banner is visible in the top right corner of the browser window.

These are the login screens. The user has the ability to login to the application using an email and password. If one does not exist, they can choose to create a new account with their own email and password. Once the user inputs their credentials, they are moved to the information screen below.

User Registration Page

DEBUG

Please input the information to creat an account

Name

Enter your age

Gender

Ethnicity

☐ LGBTQ+?

Register

This screen is how the user will input information that will go to the database. They will be prompted to register, and they will be able to view or change this information in the account page later on in the application.

← For The People: Types of Businesses

DEBUG

Black	<input type="checkbox"/>		
Hispanic	<input type="checkbox"/>	Women	<input type="checkbox"/>
Asian	<input type="checkbox"/>	Non-Binary	<input type="checkbox"/>
Pacific Islander	<input type="checkbox"/>	LGBT+	<input type="checkbox"/>
Native American	<input type="checkbox"/>		
Middle Eastern	<input type="checkbox"/>		

Submit Filters

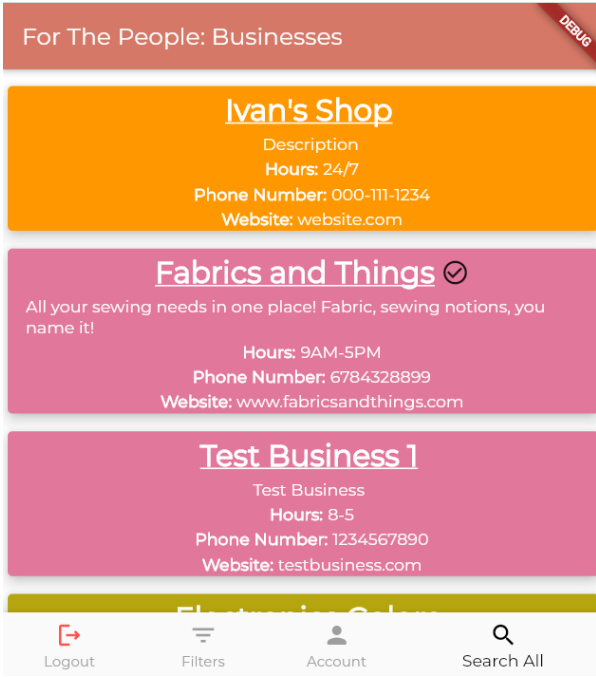
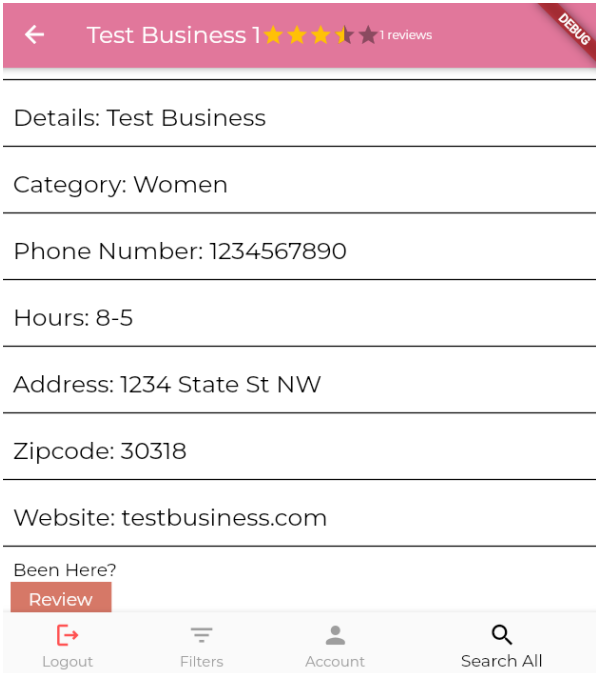
Logout

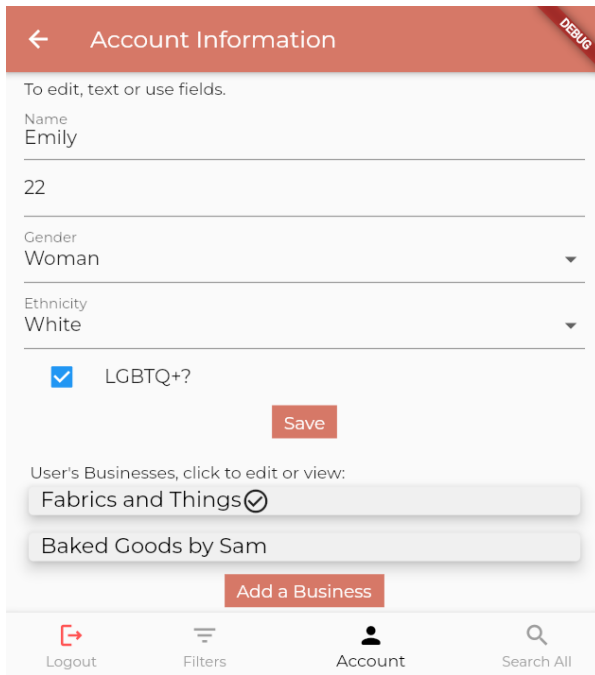
Filters

Account

Search All

This is one of the filter pages in which users can select which type of business they would like to support. This screen will appear directly after the user logs in or will be able to be reached by clicking filters on the navigation bar.

	<p>Here is the page that users will see when they are searching for businesses (after filter selection of clicking the search all button). Each business is represented by a “card” button which the user can click on to view business information. Each business is color-coded according to category.</p>
	<p>This page provides the details on the business, including its descriptions, category, contact information, location, and will display the logo. In addition, a user can review and flag the business on this page. The app-bar is the color of the category as well.</p>



← Account Information DEBUG

To edit, text or use fields.

Name
Emily

22

Gender
Woman ▼

Ethnicity
White ▼

☒ LGBTQ+?

Save

User's Businesses, click to edit or view:

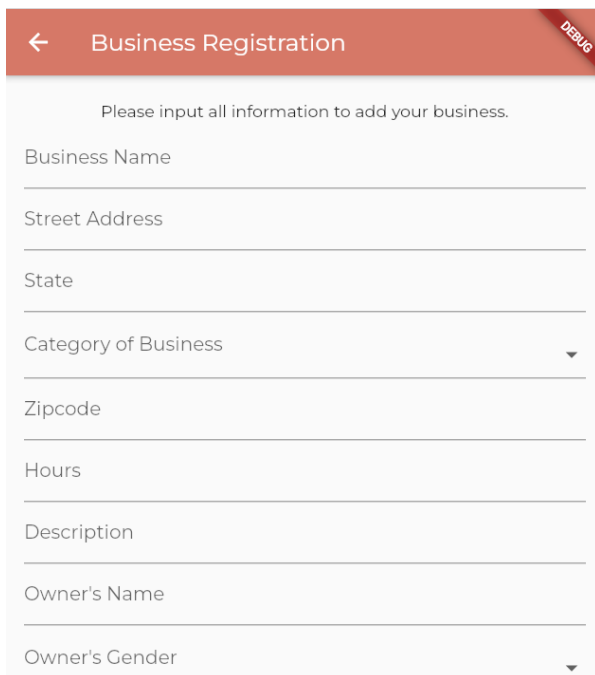
Fabrics and Things ✓

Baked Goods by Sam

Add a Business

Logout Filters Account Search All

This is the account information page. The account details for each user are displayed on this page. A user can edit their account details on this page as well by modifying their selected value and clicking the save button. The Businesses that have been created by this account are accessible on this page as well. Selecting a specific business will lead the user to the Update Business Information screen. Users can also click on the Add a Business button to be taken to the Business Registration screen where they can create a business.



← Business Registration DEBUG

Please input all information to add your business.

Business Name

Street Address

State

Category of Business ▼

Zipcode


Hours

Description

Owner's Name

Owner's Gender ▼

This is the business registration page which allows people to input their business information and allows their business to be displayed within the app. They will be able to later change this information with the update business information screen.

 Update Business Information

DEBUG

Business Name
Fabrics and Things

Street Address
555 Peachtree St.

State
GA

Category
Women

Zipcode
30318

Hours
9AM-5PM

Description
All your sewing needs in one place! Fabric, sewing notions, y

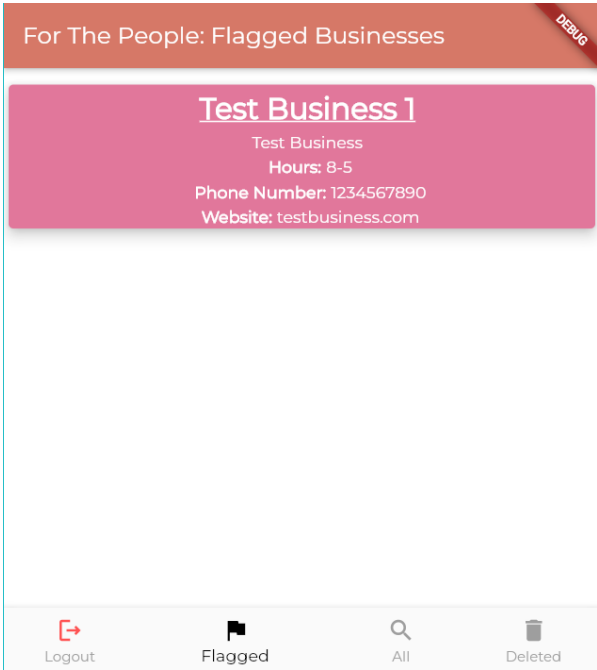
Owner's Name
Lucia

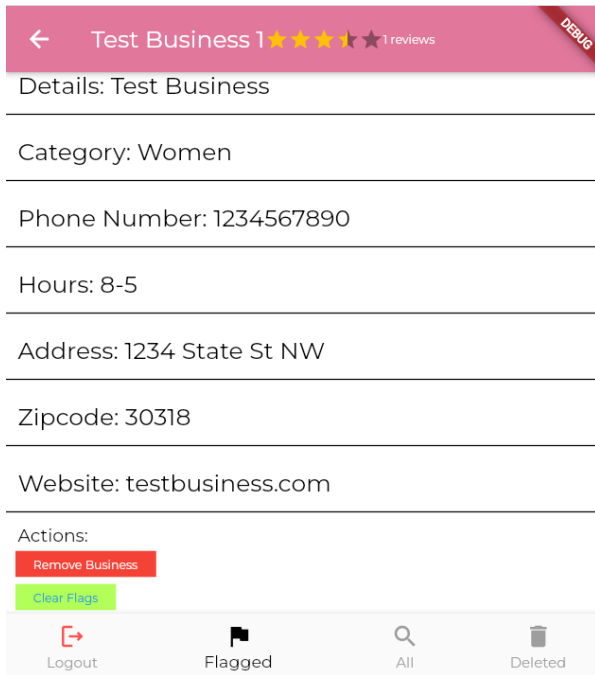
Gender
Woman

Ethnicity

This is the aforementioned update business information screen which allows the user to update any of their business information which is then displayed on the app. To modify an existing value, a user must first change their selected value and then click on the Save button.

Admin:

 <p>The mockup shows an admin interface. At the top is an orange header with the text "For The People: Flagged Businesses" and a "DEBUG" label in the top right corner. Below the header is a pink card for "Test Business 1" with details: "Test Business", "Hours: 8-5", "Phone Number: 1234567890", and "Website: testbusiness.com". At the bottom is a navigation bar with four items: "Logout" (with a red arrow icon), "Flagged" (with a black flag icon), "All" (with a magnifying glass icon), and "Deleted" (with a trash can icon).</p>	<p>This is a page for administrators. It displays a list of businesses that have been flagged by users. Each business is displayed on its own little “card” that acts as a button. Clicking or tapping on a button takes you to a page displaying a flagged business’s information.</p>
--	---



Here, the information of a flagged business is displayed. On the top of the screen, the name of the business is displayed, as well as a checkmark icon to denote if the business is verified and a series of stars displaying the average rating of the business. At the bottom of the list of the business's information are two buttons. The button titled "Remove Business" deletes the business, however a copy of the business's data is copied and stored into a separate area of the database. The button titled "Clear Flags" removes the business from the flagged business list and resets its flagged statistics. You can navigate to other pages via the navigation bar at the bottom, and by using the back button on the app bar at the top of the screen.