Localising Essentials

in the Times of Coronavirus

Design Document

Divya Donapati - 2019201084
Shivani Hanji - 2019201016
Vanalata Bulusu - 201556135
Sarigama Yerra - 201556206
Abhishek Gorisaria - 2019201007
Praveena Sidgar - 2019201042

1. INTRODUCTION

In this tough time where all of us are quarantined and struggling to get day-to-day household supplies. Here our team came up with an idea of designing an app where it gives information about nearby stores. When we need to buy some groceries but we are unaware of their availability in shops, we are not even aware of the availability of shops. In such a case we need to check each and every store until we find the product. Quarantine is all about maintaining distance and staying at home, It's not advisable to check all the stories manually. How does our application help in such cases? Our application will give the information of the availability of the store and availability of the products(sticking to the day to day needs basically). It will be simple for the user just to go through the application and check for availability of the shop and availability of things.

How will we make the information available in the application?

It's a very simple few step process. Shopkeepers should first register into the app and list the availability of the products. This will let us know the availability of the shop and also products. How to get rid of fake stores?

Here users can upvote/downvote the store.

How to update the list of items available?

As we don't mention the quantity available, once the item gets completed the shopkeeper removes it from the list and once the stock comes the shopkeeper updates the list.

Here we may get a question: why does the shopkeeper keep making efforts to use and maintain the app?

Making the available products visible to customers there is an increase in the sales. It also helps people in tough times.

2. FUNCTIONAL REQUIREMENTS

1. Management of Shop details:

   The app will store and manage the following details, for each shop registered on the app:

   a) Name of the shop
   b) Shop location
   c) The type of the shop
   d) The shop's GST number (if available)
   e) Shop status: Open or Closed
   f) Shop opening and closing times
   g) List of items available in the shop
   h) Answers to feedback questions
   i) Image of the shop/ curfew pass / essential services pass

2. Management of User details:
   The app will store and manage the following user details:

   a) Name
   b) Phone number (for verification)
   c) App account's password
   d) User type (whether it is a customer/retailer)

3. The app will allow a user to perform the following tasks without requiring the user to log in:

   a) Search for shops and items
   b) View a list of shops near the user's location which fulfil the specified criteria (i.e., shops which have the item(s) specified by the user).

   c) To see if a particular shop is open or closed
   d) To get the directions to go to a particular shop

e) Register for a new account, either as a retailer/customer

4. A user (type = customer) can perform the following tasks only if they have created an account on the app:

a) Give feedback on a shop by answering specific questions
b) Log in to account
c) Update account details

5. A user (type = retailer) can perform the following tasks only if they have created an account on the app:

a) Mark their shop's location

b) Specify the shop's opening and closing times

c) Specify the items available in the shop

d) Log in to account

e) Update account details

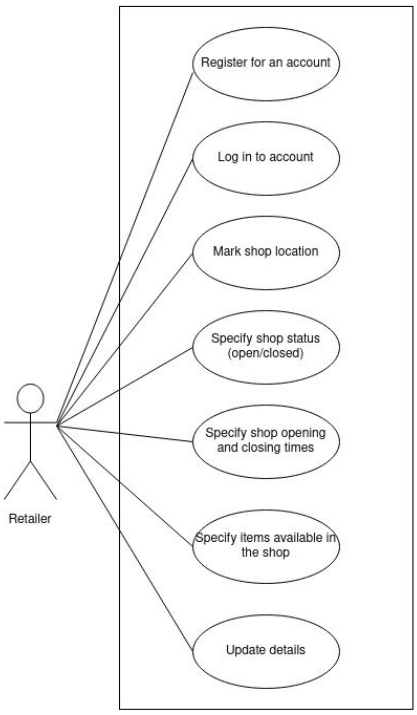
6. The app will also optionally require access to a customer's device's current location to look for shops near the customer. If permission is not granted, then the user has to specify the location manually on the map.

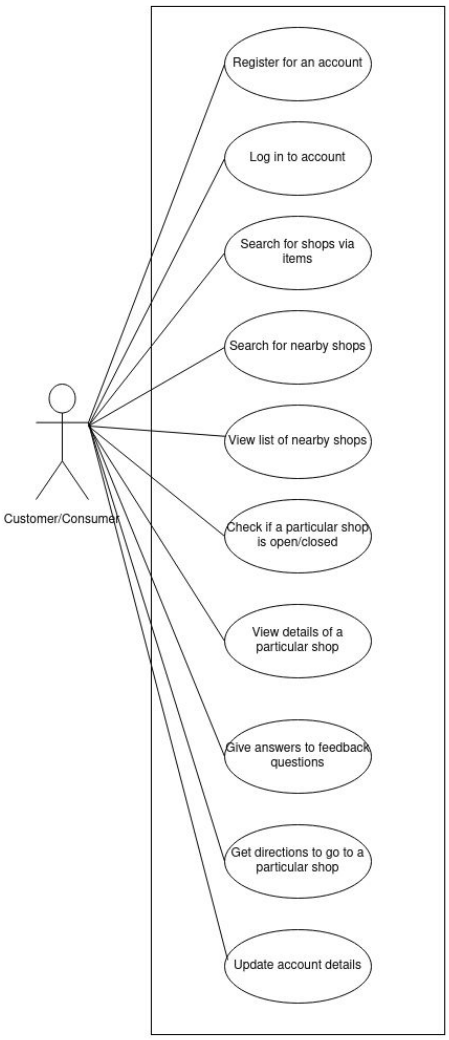
3. Non Functional Requirements

- Automatic Scaling - Firebase is built for performance and scalability. As and when there is a change in data, Firebase helps in the calculation of the minimum set of updates needed to keep all your clients synchronized. Additionally, the API functions of firebase are designed in order to scale horizontally with the size of data being synchronized. It handles the scaling operations. Your application will scale from its first user to its first million users without any change in the code.

- Availability of Firebase is >= 99.99% according to the Service Level Agreement.

Use Case Diagram:





Use Case Descriptions:

Use Case Name:	Register for an account
Actors involved:	Customer/Consumer, retailer
Preconditions:	The app must be installed on the user's phone
Postconditions:	A new user account will be created and securely stored in the database
Flow of activities:	1. The user launches the app and is shown the home screen 2. The user presses the button for account registration 3. The user chooses either the customer or the retailer option 4. The user enters their name and phone number a. If the user selected the retailer option, they can also optionally upload an image of their shop / curfew pass / essential services pass, and also specify their GST number if it is available 5. The phone number (and GST number, if available) is used for verification 6. If the phone number is verified successfully, the account is created 7. User data will be stored in the database in a secure way
Exception conditions:	If the phone number is not verified successfully, then an error message will be shown and the user will be asked to retry registration

Use Case Name:	Mark shop location
Actors involved:	retailer
Preconditions:	The user must have an account of type = 'retailer'
Postconditions:	1. The shop location will be stored in the database and will be associated with the retailer who specified it 2. It will be used to provide search results to users
Flow of activities:	1. The retailer logs in to their account on the app 2. The retailer presses a button for entering the shop location 3. The retailer will be asked if the current device location can be accessed by the app, and if so, then whether it is to be used as the shop's location 4. If the retailer agrees to both of these conditions, then the current device location will be detected automatically and stored as the shop location. Else, the retailer has to specify the location manually on the map.

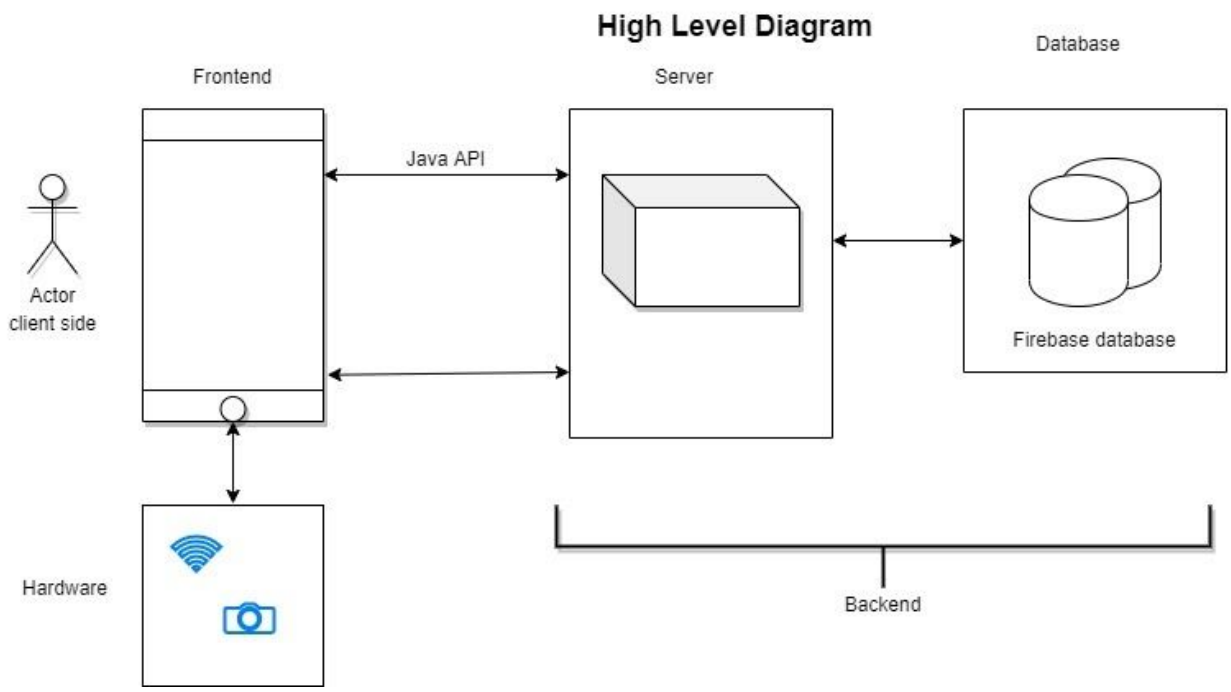
Use Case Name:	Specify shop opening and closing times
Actors involved:	Retailer
Preconditions:	The user must have an account of type = ‘retailer’
Postconditions:	The shop’s opening and closing times will be shown to users
Flow of activities:	1. The retailer logs in to their account on the app 2. The retailer presses a button for entering timing information 3. The retailer enters the shop opening and closing times which are subsequently saved
Exception conditions:	The closing time must be chronologically greater than the opening time, else an error message will be shown.

Use Case Name:	Specify items available in the shop
Actors involved:	retailer
Preconditions:	The user must have an account of type = ‘retailer’
Postconditions:	The list of items available in the shop will be shown to users
Flow of activities:	1. The retailer logs in to their account on the app 2. The retailer presses a button for entering item information 3. The retailer is shown a drop-down list with predefined general items/categories of items, from which items can be selected. There will also be an option to type names of other items/categories of items, which are not present in the list.

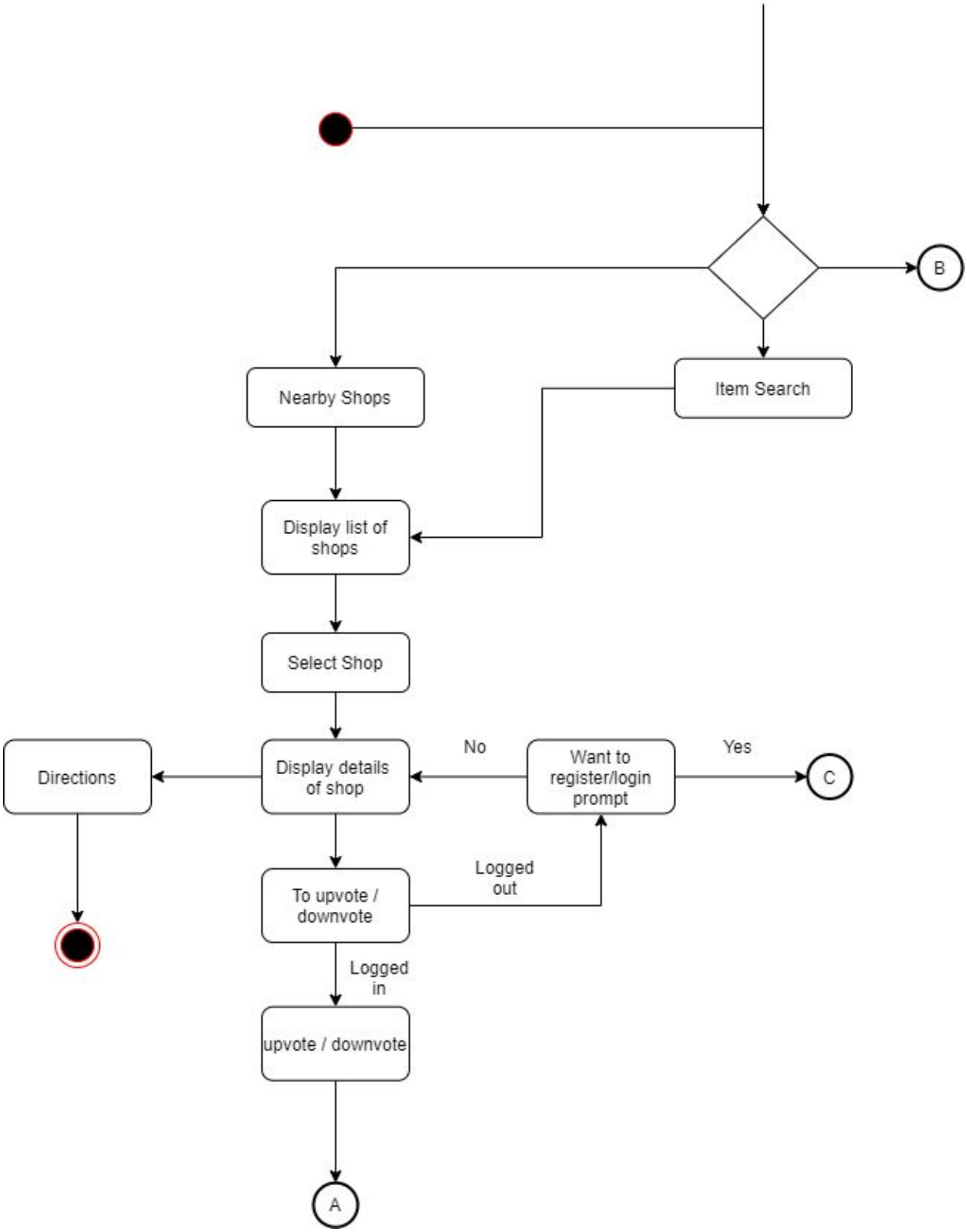
Use Case Name:	View details of a particular shop
Actors involved:	Customer/Consumer
Preconditions:	The app must be installed on the user’s phone
Flow of activities:	1. The customer logs in to their account on the app 2. The customer searches for nearby shops, either by specifying a type of shop, or a certain item/type of item 3. The customer is shown a list of shops near the customer’s current location 4. The customer selects one of the shops in the list 5. The customer can view details of the selected shop (whether open/closed, its timings, location, items available, etc.)

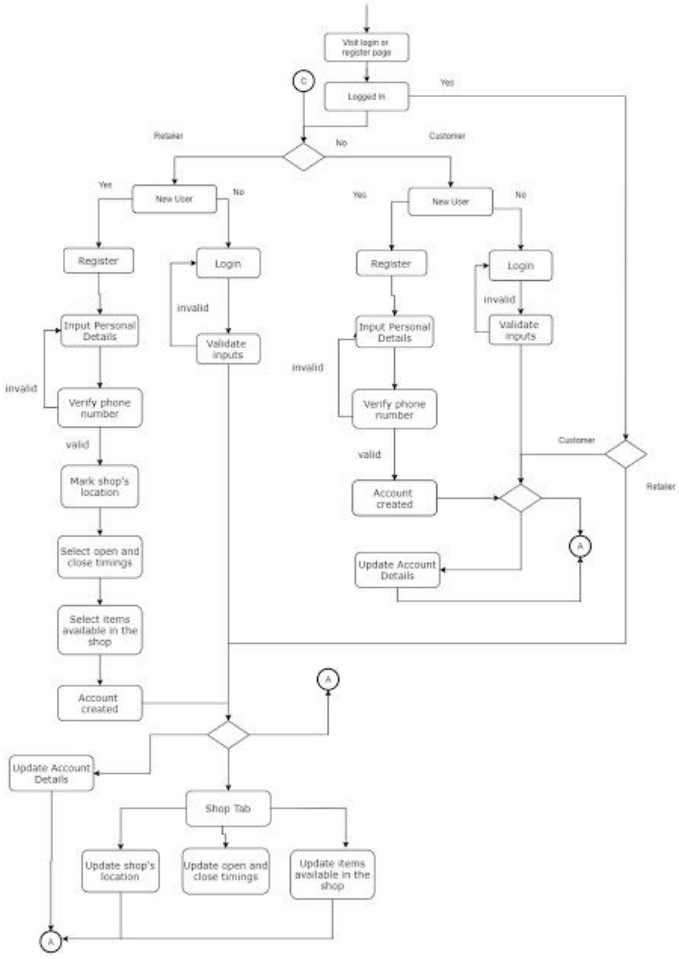
Use Case Name:	Give answers to feedback questions
Actors involved:	Customer/Consumer
Preconditions:	The user must have an account of type = ‘customer’
Postconditions:	Feedback given by the user will be stored in the database, and would be used to help other users know about factors such as availability of items in a shop
Flow of activities:	1. The customer logs in to their account on the app 2. The customer searches for nearby shops by specifying a certain item/type of item 3. The customer is shown a list of shops near their current location 4. The customer selects one of the shops in the list 5. The customer will be provided an option to answer some questions related to the selected shop as feedback, which can be answered if the customer has visited the shop (eg. whether the items indicated were actually available or not)

4. HIGH LEVEL DIAGRAM

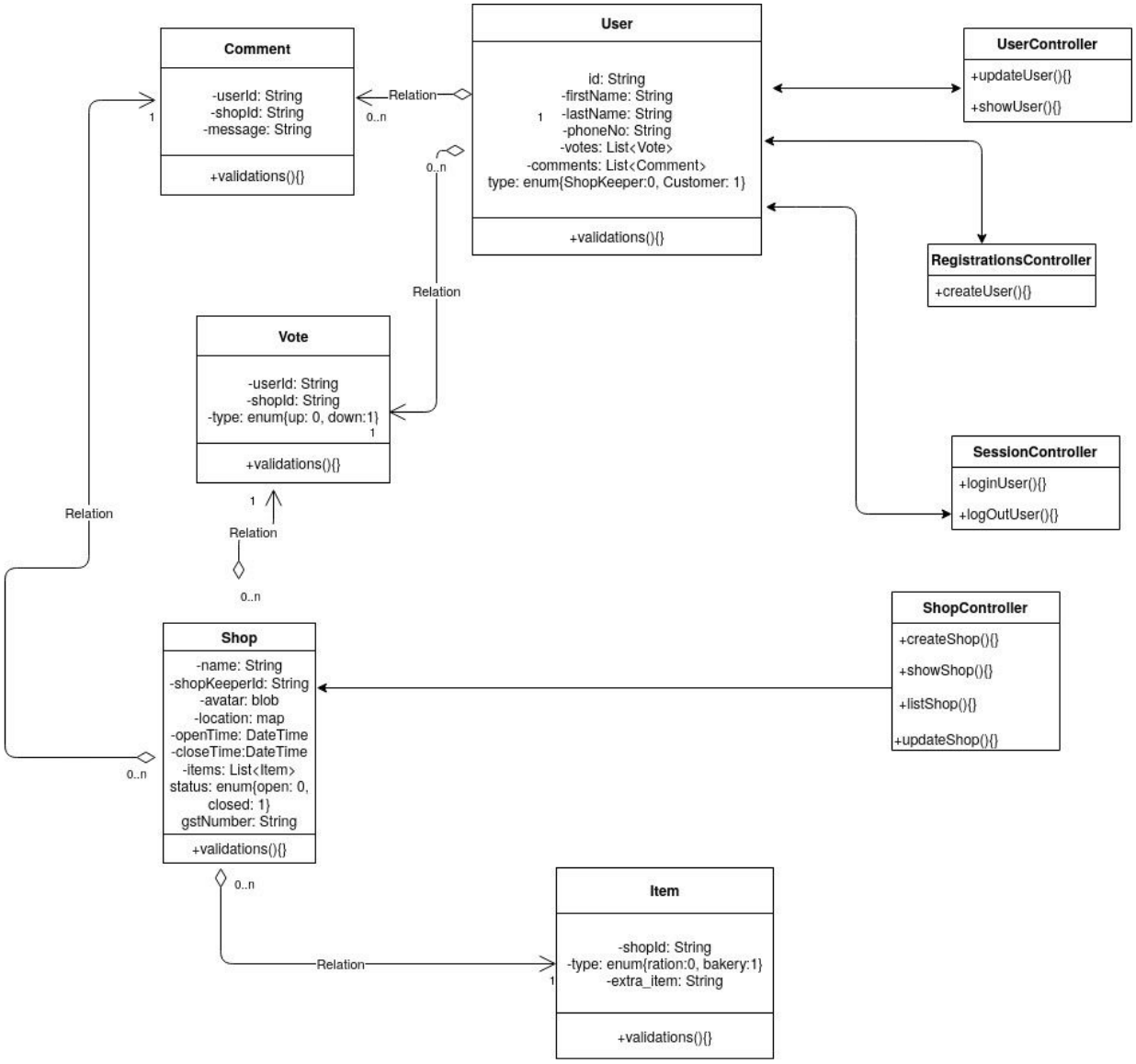


5.ACTIVITY_DIAGRAM




Visit login or register page
C
Logged In
Yes
No
Retailer
Customer
New User
Yes
No
Register
Login
invalid
Input Personal Details
Validate inputs
invalid
Verify phone number
valid
Mark shop's location
Select open and close timings
Select items available in the shop
Account created
A
Update Account Details
Shop Tab
Update shop's location
Update open and close timings
Update items available in the shop
Customer
Retailer

6. CLASS DIAGRAM



7. Design Pattern to be used

1)Builder design pattern for User Creation since, user may have optional field.

8. Technology

- Google API
- Firebase (Database and Server)
- Android (XML and java)

Google API

Google APIs is a set of application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Third-party apps can use these APIs to take advantage of or extend the functionality of the existing services.
We use Google API in the following way

Mapping location : We make use of the maps service provided by Google API to keep track of the location of shops. There are client libraries in various languages which allow us to use Google APIs from within their code.

Firebase (Database and Server)

Advantages: Firebase provides a real-time database and back-end as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The company provides client libraries that enable integration with Android, JavaScript, Java. With the help of libraries we can easily make use of the Firebase database.

Android (XML and java)

An Android app is made up of two parts: the front end and the back end. The front end is the visual part of the app that the user interacts with, and the back end, which contains all the code that drives the app. The front end is written using XML. it is very similar to HTML. There is at least one XML layout file for each activity (or several if you are supporting multiple device sizes), as well as layout files for custom views. We can even use java in the frontend. The back end is written in Java. You can use the Java

standard library in addition to the Android libraryThis gives us access to a ton of pre-made objects, and the APIs are thoroughly documented online by Oracle and Google.

Why Android?

Most of the shopkeepers use Android based smart phones, if they use a smart phone (around >95%).

Why firebase?

- When you have a short development time it would be best to go with firebase. Firebase helps in creating a prototype in a very short period. It helps to cut down the development time and avoids messing with servers and data storage to a great extent.

- Queries with limited sorting and filtering functionality can be performed with the firebase database. Cloud firestore assures automatic scaling and can handle 1 million concurrent connections and 10,000 writes/second. If your application's scalability and concurrency level fall within this category, you can opt for the cloud firestore.

- When your applications require a minimal level of integration with legacy systems or third-party services, firebase will be the right choice. Firebase also becomes an ideal choice when your application does not require heavy data processing or any form of complex user authentication requirements.

- Firebase is a good choice if you plan to either write a brand-new application or rewrite an existing one from scratch. Additionally, firebase helps in the easy storing and retrieval of dynamic content. If you decide to develop the application without any form of custom coding the backend, firebase makes this easy. For example, the chat app or a video conferencing application where the backend is not required in managing the things.

9. The reasons for various design decisions and the alternatives considered:

1. Why we decided to ask the user to answer feedback questions: Because this way, in case items are not actually available in the shop, this can help other users know about it. It helps to validate if the retailer is genuine or not. We considered the alternatives of letting the users upvote/downvote, or to write comments, but ruled out these alternatives because the main purpose of our app is to ensure that people get the things that they require, and not to serve as a platform for rating shops.

2. Why we decided to let users search for shops by items: Because usually the first thing that would come to a user's mind is the item(s) that they want to buy, and also, some items may be available in different kinds of stores. So this way, a user can see all the stores (which may be of different types, eg. grocery stores as well as medical stores) which have the item(s) that the user requires. The other alternative we considered was to let the user search for shops by 'shop type', but that would not provide the advantages which searching by item provides.

3. Why we decided to ask for the user's phone number (and GST number for shopkeepers): In order to verify the user's identity, especially that of shopkeepers, so that the possibility of fake accounts is reduced

10 . Testing

1)Test Cases Types - Pass Case Scenario , Fail Case Scenario ,Edge Case Scenario

Annotations used in Junit -

BeforeEach - The method annotated under this gets executed before each and every test method.

BeforeAll - The method annotated under this gets executed before all the test methods and is executed only once at the start. AfterEach - The method annotated under this gets executed after each and every test method.

AfterAll - The method annotated under this gets executed after all the test methods and is executed only once at the end.

2) Use libraries Junit 4.1

Follow this for Junit Assertions - https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html 3) Use Mockito library for mocking, networks calls, java class creation etc.

Mocking & Isolated Test Case -

Mocking means to create “mock” objects that will replace ‘get’ method of class B and simulate its behaviour inside the test of ‘add’ method which belongs to class A. It is done when there is dependencies among classes

Mockito is a popular open source framework

and we will use it to create mock objects in the coming videos.

4) TDD

Test Driven Development Test->Code->test->code->test->test->code...

Writing test cases that fail

Writing code just enough to pass the failed test case Making changes to write the code more effectively Test cases should be independent.

Test cases should not have shared resources

Refactoring -

https://refactoring.guru/refactoring/techniques
https://sourcemaking.com/refactoring/refactorings

11. Extended Feature

In the future version of the app with the help of google API we can use multiple languages.