Lab5

for

TransitEase

Version 3 approved

Prepared by Ashwin, Dave, Jun Heng, Jonathan

Nanyang Technological University

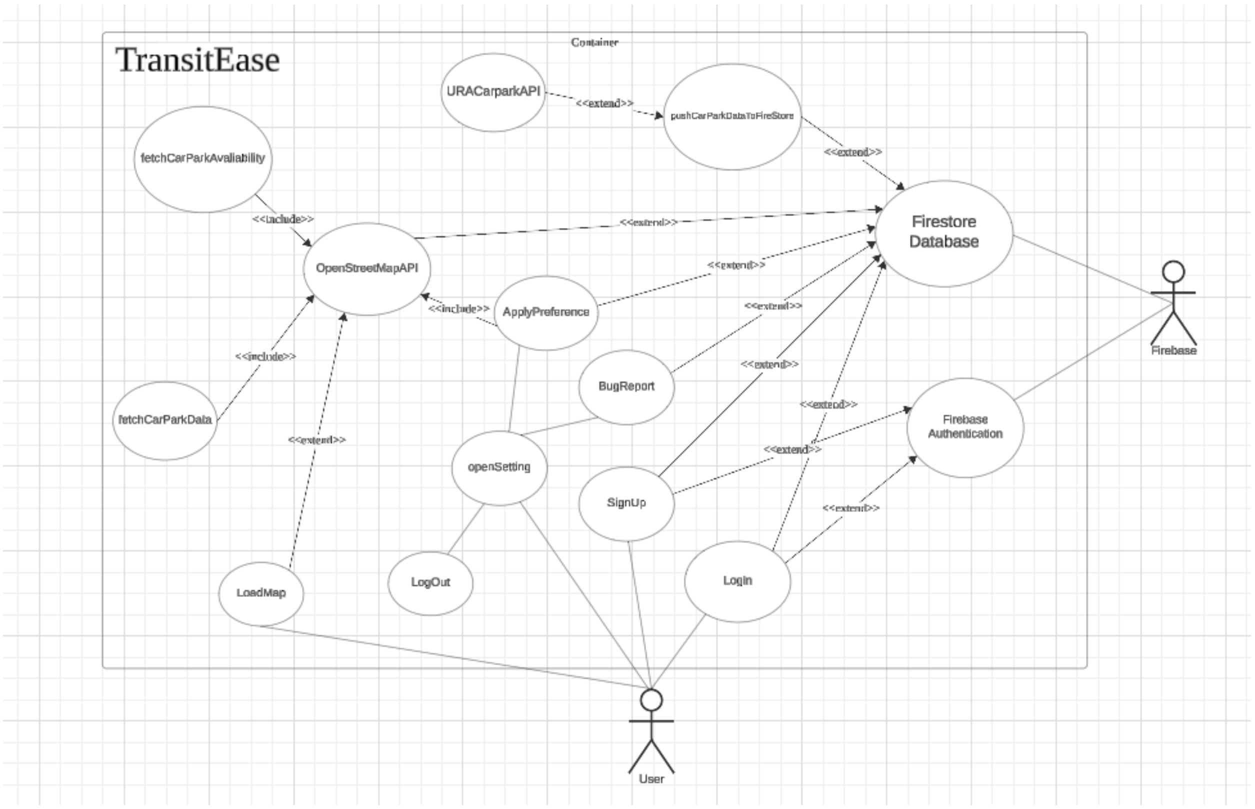
10/11/24

Revision History

Name	**Date**	**Reason For Changes**	**Version**
Ashwin			3.0
Dave			3.0
Jonathan			3.0
Goh Jun Heng			3.0

1.Table Of Contents

2. Complete Use Case Diagram



3. Use Case Descriptions

1. LogInEmail

Use Case ID:	1		
Use Case Name:	LogInEmail		
Created By:	Dave Goh	Last Updated By:	Dave Goh
Date Created:	17/09/24	Date Last Updated:	17/09/24

Actor:	User
Description:	Allows users to log in using an existing account via email credentials.
Preconditions:	The user must have already registered an account.
Postconditions:	The user gains access to the app's main interface upon successful login.
Priority:	High
Frequency of Use:	1
Flow of Events:	1. User selects "Login with Email" in the user interface. 2. User enters an email and password. 3. System validates credentials with Firebase authentication. 4. Users are granted access upon successful authentication.
Alternative Flows:	NIL
Exceptions:	1. Incorrect password.

	2. Username does not exist.
Includes:	Firebase authentication API
Special Requirements:	Integration with Firebase Authentication.
Assumptions:	User has already created an account before.
Notes and Issues:	None

2. LoginWGoogle

Use Case ID:	2		
Use Case Name:	LoginWGoogle		
Created By:	Ashwin Suresh	Last Updated By:	Ashwin Suresh
Date Created:	17/09/24	Date Last Updated:	17/09/24

Actor:	User
Description:	Enables users to log in using their Google account credentials.
Preconditions:	The user must have a valid Google account.
Postconditions:	The user gains access to the app's main interface upon successful login.
Priority:	High

Frequency of Use:	Once per session
Flow of Events:	1. User selects "Login with Google" in the user interface. 2. User is prompted to grant app permissions to access their Google account. 3. System authenticates user via Google authentication. 4. User logs in successfully.
Alternative Flows:	NIL
Exceptions:	User denies app permission to access their Google account.
Includes:	
Special Requirements:	Integration with Google Authentication.
Assumptions:	User has a valid and accessible Google account.
Notes and Issues:	None

3. CreateAccount

Use Case ID:	3		
Use Case Name:	CreateAccount		
Created By:	Ashwin Suresh	Last Updated By:	Ashwin Suresh
Date Created:	17/09/24	Date Last Updated:	17/09/24

Actor:	User
Description:	Allows new users to create an account using their email.

Preconditions:	The email must not be registered in the system.
Postconditions:	A new account is created, and the user can log in with their credentials.
Priority:	High
Frequency of Use:	Typically used once by new users
Flow of Events:	1. User selects "Create Account" in the user interface. 2. User enters a valid email, password, and confirms the password. 3. System checks for existing account using Firebase API. 4. Account is created successfully if email is unique.
Alternative Flows:	NIL
Exceptions:	1. Email already exists. 2. Passwords do not match.
Includes:	Firebase API
Special Requirements:	Integration with Firebase.
Assumptions:	1. User provides a valid email. 2. User inputs matching passwords.
Notes and Issues:	None

4. LoadMap

Use Case ID:	4		
Use Case Name:	LoadMap		
Created By:	Ashwin Suresh	Last Updated By:	Ashwin Suresh
Date Created:	17/09/24	Date Last Updated:	17/09/24

Actor:	User
Description:	Loads the map interface showing the user’s current location.
Preconditions:	User must have granted location permissions.
Postconditions:	The map displays the user’s location.
Priority:	High
Frequency of Use:	Every time the map is loaded
Flow of Events:	1. User is prompted to grant location permission if not already granted. 2. User grants permission. 3. Map is loaded and centred on the user's current location. 4. User's location is indicated on the map.
Alternative Flows:	NIL
Exceptions:	User denies location permission. No network connectivity.
Includes:	TomTomAPI

Special Requirements:	Location permission must be granted
Assumptions:	The user has internet access
Notes and Issues:	None

5. QueryNearbyCarpark

Use Case ID:	5		
Use Case Name:	QueryNearbyCarpark		
Created By:	Dave Goh	Last Updated By:	Dave Goh
Date Created:	17/09/24	Date Last Updated:	17/09/24

Actor:	System
Description:	Queries and displays nearby carparks based on the user’s location.
Preconditions:	Network connectivity and location permissions are granted.
Postconditions:	The user can view nearby carparks with details.
Priority:	High
Frequency of Use:	Frequently used when searching for parking.

Flow of Events:	1. System queries URA Carpark API for nearby carpark information based on the user's location. 2. Car Parks are displayed on the map in proximity to the user's location. 3. Car Park information such as distance, rate, EV charging capability, and capacity are displayed.
Alternative Flows:	NIL
Exceptions:	1. No network connectivity. 2. Location permissions are not granted.
Includes:	1. URA Carpark API 2. TomTom API
Special Requirements:	Integration with URA and TomTom APIs.
Assumptions:	User has internet access.
Notes and Issues:	None

6. ReportBug

Use Case ID:	6		
Use Case Name:	ReportBug		
Created By:	Ashwin Suresh	Last Updated By:	Ashwin Suresh
Date Created:	17/09/24	Date Last Updated:	17/09/24

Actor:	User
Description:	Allows users to submit a bug report.
Preconditions:	User must be logged in.
Postconditions:	Bug report is saved in the database for review.
Priority:	High
Frequency of Use:	Used occasionally.
Flow of Events:	1. User navigates to the settings menu. 2. User selects the "Report Bug" option. 3. User enters the bug description in a text field. 4. User submits the bug report. 5. Report is stored in the database.
Alternative Flows:	NIL
Exceptions:	1. Text field contains invalid characters. 2. Report exceeds 1000 characters. 3. No network connectivity.
Includes:	Firebase API
Special Requirements:	Integration with Firebase.
Assumptions:	User has internet access.
Notes and Issues:	None

7. ApplyPreferences

Use Case ID:	7		
Use Case Name:	ApplyPreferences		
Created By:	Dave Goh	Last Updated By:	Dave Goh
Date Created:	17/09/24	Date Last Updated:	17/09/24

Actor:	User
Description:	Allows users to modify their app preferences.
Preconditions:	Users must be on the preferences page.
Postconditions:	Changes are saved to the user’s profile in the database.
Priority:	High
Frequency of Use:	Used frequently to adjust preferences.
Flow of Events:	1. User navigates to the preferences page. 2. Users modify their preferences. 3. System updates the changes in the user database.
Alternative Flows:	NIL
Exceptions:	No network connectivity.
Includes:	None
Special Requirements:	None

Assumptions:	User has internet access
Notes and Issues:	None

8. Logout

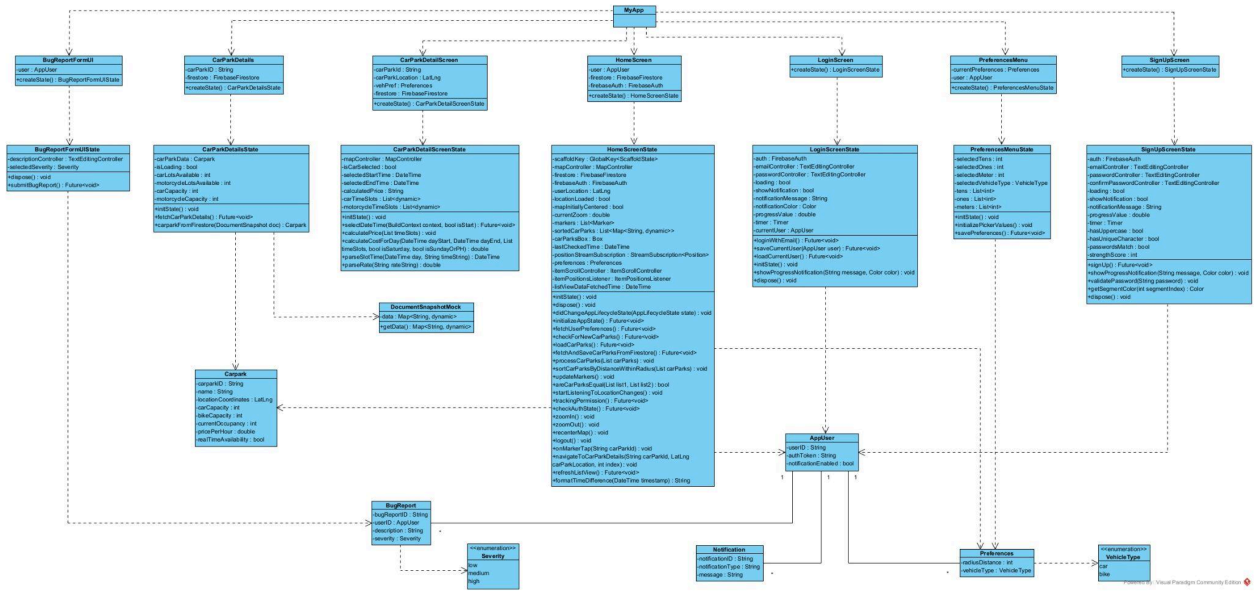
Use Case ID:	8		
Use Case Name:	Logout		
Created By:	Dave Goh	Last Updated By:	Dave Goh
Date Created:	17/09/24	Date Last Updated:	17/09/24

Actor:	User
Description:	Logs the user out of the app.
Preconditions:	The user must be logged in.
Postconditions:	User is redirected to the login screen.
Priority:	High
Frequency of Use:	Frequently used at the end of a session.
Flow of Events:	1. User selects "Logout." 2. Application logs out the user and redirects them to the login page.
Alternative Flows:	NIL
Exceptions:	No network connectivity.

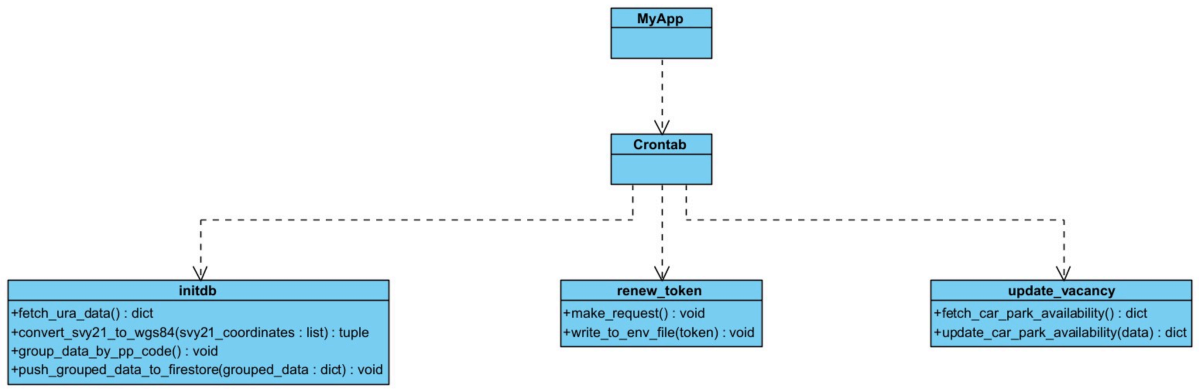
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4. Design Model

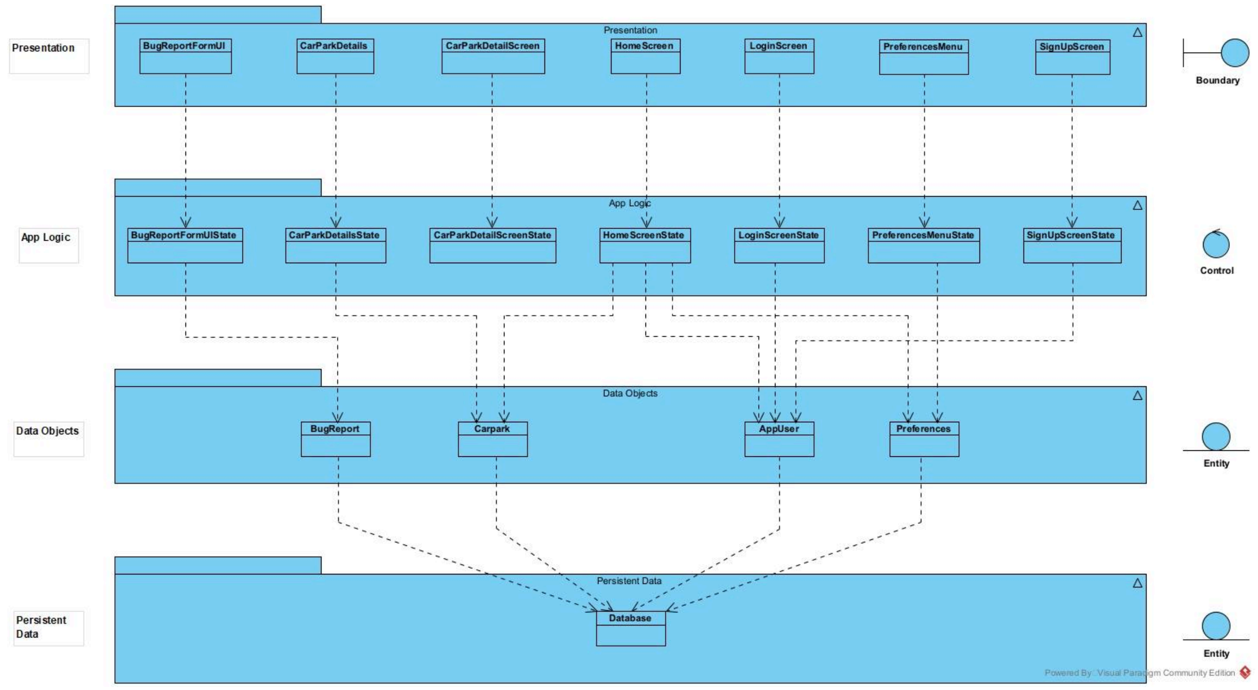
4.1 Class Diagram for FrontEnd


MyApp
BugReportFormUI
-user : AppUser
+createState() : BugReportFormUIState
CarParkDetails
-carParkID : String
-firestore : FirebaseFirestore
+createState() : CarParkDetailsState
CarParkDetailScreen
-carParkId : String
-carParkLocation : LatLng
-vehPref : Preferences
-firestore : FirebaseFirestore
+createState() : CarParkDetailScreenState
HomeScreen
-user : AppUser
-firestore : FirebaseFirestore
-firebaseAuth : FirebaseAuth
+createState() : HomeScreenState
LoginScreen
+createState() : LoginScreenState
PreferencesMenu
-currentPreferences : Preferences
-user : AppUser
+createState() : PreferencesMenuState
SignUpScreen
+createState() : SignUpScreenState
BugReportFormUIState
-descriptionController : TextEditingController
-selectedSeverity : Severity
+dispose() : void
+submitBugReport() : Future<void>
CarParkDetailsState
-carParkData : Carpark
-isLoading : bool
-carLotsAvailable : int
-motorcycleLotsAvailable : int
-carCapacity : int
-motorcycleCapacity : int
+initState() : void
+fetchCarParkDetails() : Future<void>
+carparkFromFirestore(DocumentSnapshot doc) : Carpark
CarParkDetailScreenState
-mapController : MapController
-isCarSelected : bool
-selectedStartTime : DateTime
-selectedEndTime : DateTime
-calculatedPrice : String
-carTimeSlots : List<dynamic>
-motorcycleTimeSlots : List<dynamic>
+initState() : void
+selectDateTime(BuildContext context, bool isStart) : Future<void>
+calculatePrice(List timeSlots) : void
+calculateCostForDay(DateTime dayStart, DateTime dayEnd, List timeSlots, bool isSaturday, bool isSundayOrPH) : double
+parseSlotTime(DateTime day, String timeString) : DateTime
+parseRate(String rateString) : double
HomeScreenState
-scaffoldKey : GlobalKey<ScaffoldState>
-mapController : MapController
-firestore : FirebaseFirestore
-firebaseAuth : FirebaseAuth
-userLocation : LatLng
-locationLoaded : bool
-mapInitiallyCentered : bool
-currentZoom : double
-markers : List<Marker>
-sortedCarParks : List<Map<String, dynamic>>
-carParksBox : Box
-lastCheckedTime : DateTime
-positionStreamSubscription : StreamSubscription<Position>
-preferences : Preferences
-itemScrollController : ItemScrollController
-itemPositionsListener : ItemPositionsListener
-listViewDataFetchedTime : DateTime
+initState() : void
+dispose() : void
+didChangeAppLifecycleState(AppLifecycleState state) : void
+initializeAppState() : Future<void>
+fetchUserPreferences() : Future<void>
+checkForNewCarParks() : Future<void>
+loadCarParks() : Future<void>
+fetchAndSaveCarParksFromFirestore() : Future<void>
+processCarParks(List carParks) : void
+sortCarParksByDistanceWithinRadius(List carParks) : void
+updateMarkers() : void
+areCarParksEqual(List list1, List list2) : bool
+startListeningToLocationChanges() : void
+trackingPermission() : Future<void>
+checkAuthState() : Future<void>
+zoomIn() : void
+zoomOut() : void
+recenterMap() : void
+logout() : void
+onMarkerTap(String carParkId) : void
+navigateToCarParkDetails(String carParkId, LatLng carParkLocation, int index) : void
+refreshListView() : Future<void>
+formatTimeDifference(DateTime timestamp) : String
LoginScreenState
-auth : FirebaseAuth
-emailController : TextEditingController
-passwordController : TextEditingController
-loading : bool
-showNotification : bool
-notificationMessage : String
-notificationColor : Color
-progressValue : double
-timer : Timer
-currentUser : AppUser
+loginWithEmail() : Future<void>
+saveCurrentUser(AppUser user) : Future<void>
+loadCurrentUser() : Future<void>
+initState() : void
+showProgressNotification(String message, Color color) : void
+dispose() : void
PreferencesMenuState
-selectedTens : int
-selectedOnes : int
-selectedMeter : int
-selectedVehicleType : VehicleType
-tens : List<int>
-ones : List<int>
-meters : List<int>
+initState() : void
+initializePickerValues() : void
+savePreferences() : Future<void>
SignUpScreenState
-auth : FirebaseAuth
-emailController : TextEditingController
-passwordController : TextEditingController
-confirmPasswordController : TextEditingController
-loading : bool
-showNotification : bool
-notificationMessage : String
-progressValue : double
-timer : Timer
-hasUppercase : bool
-hasUniqueCharacter : bool
-passwordsMatch : bool
-strengthScore : int
+signUp() : Future<void>
+showProgressNotification(String message, Color color) : void
+validatePassword(String password) : void
+getSegmentColor(int segmentIndex) : Color
+dispose() : void
DocumentSnapshotMock
-data : Map<String, dynamic>
+getData() : Map<String, dynamic>
Carpark
-carparkID : String
-name : String
-locationCoordinates : LatLng
-carCapacity : int
-bikeCapacity : int
-currentOccupancy : int
-pricePerHour : double
-realTimeAvailability : bool
AppUser
-userID : String
-authToken : String
-notificationEnabled : bool
1
1
1
BugReport
-bugReportID : String
-userID : AppUser
-description : String
-severity : Severity
*
<<enumeration>>
Severity
low
medium
high
Notification
-notificationID : String
-notificationType : String
-message : String
*
Preferences
-radiusDistance : int
-vehicleType : VehicleType
*
<<enumeration>>
VehicleType
car
bike
Powered By: Visual Paradigm Community Edition

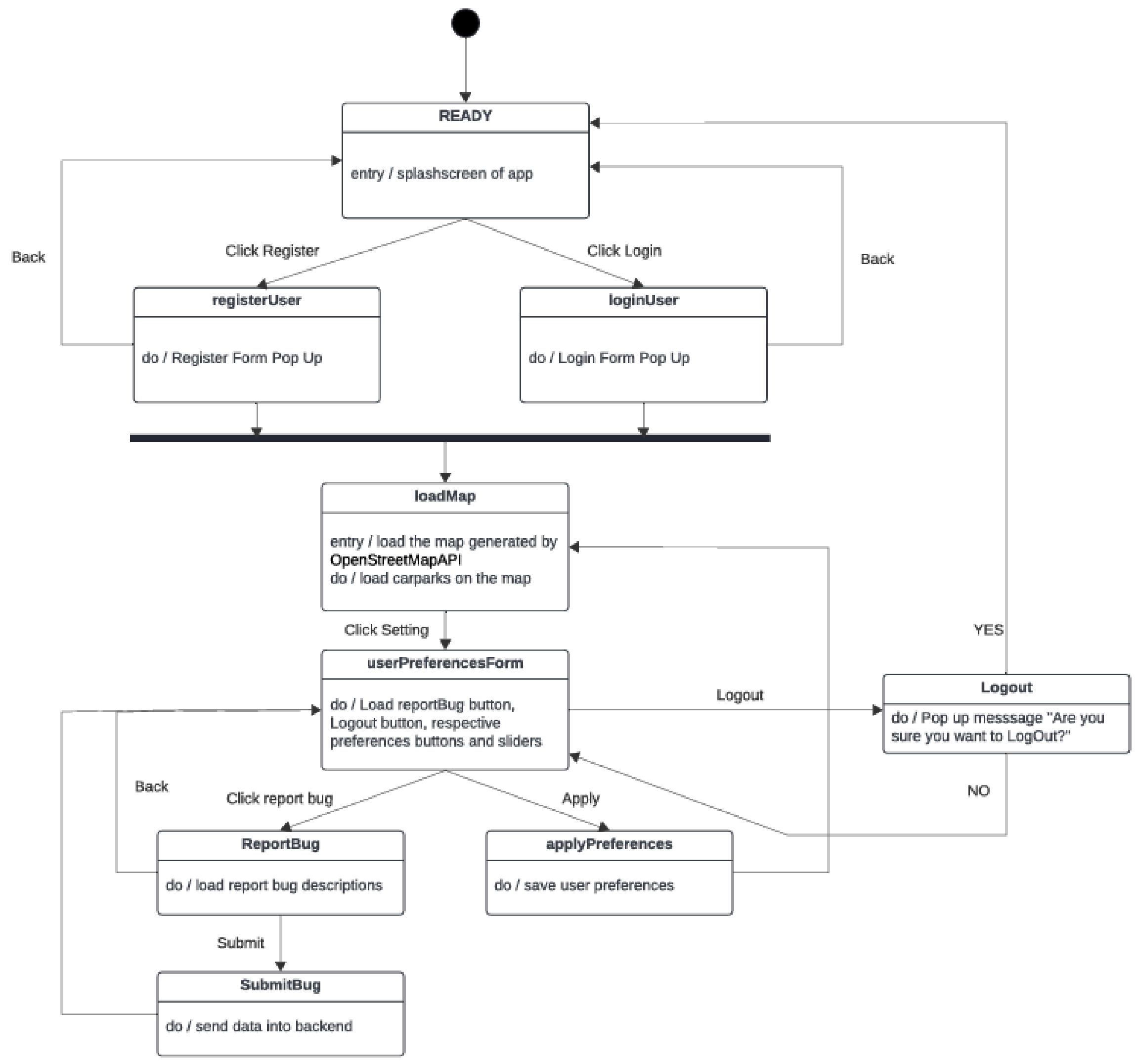
4.2 Class Diagram for BackEnd


MyApp
Crontab
initdb
+fetch_ura_data() : dict
+convert_svy21_to_wgs84(svy21_coordinates : list) : tuple
+group_data_by_pp_code() : void
+push_grouped_data_to_firestore(grouped_data : dict) : void
renew_token
+make_request() : void
+write_to_env_file(token) : void
update_vacancy
+fetch_car_park_availability() : dict
+update_car_park_availability(data) : dict

4.3 System Architecture Diagram

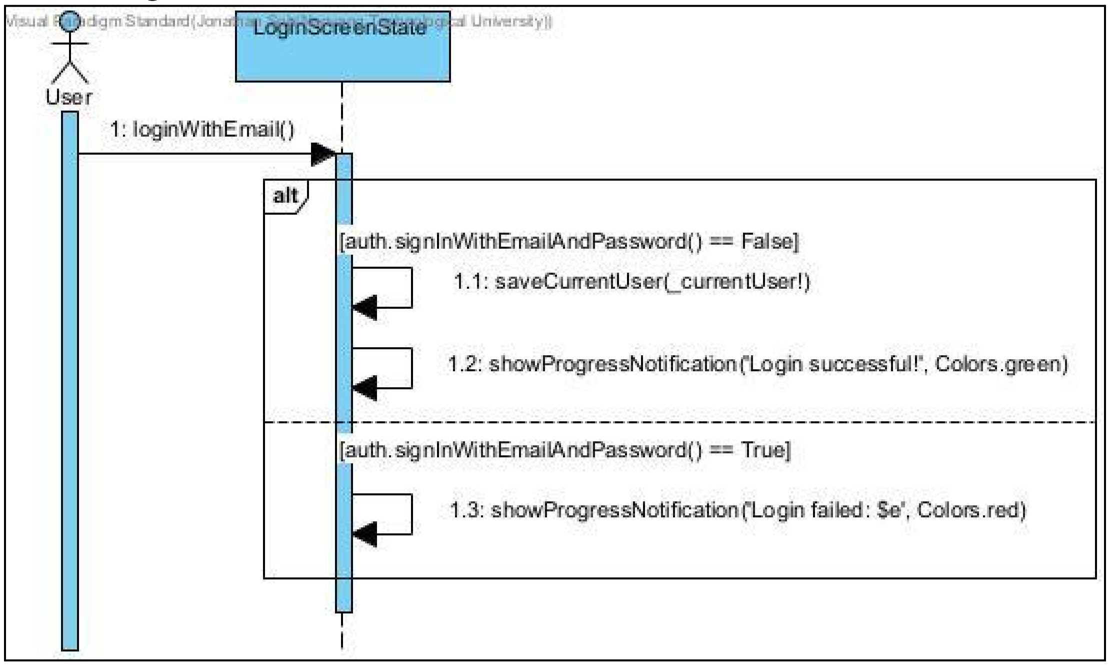


4.4 Dialog Map

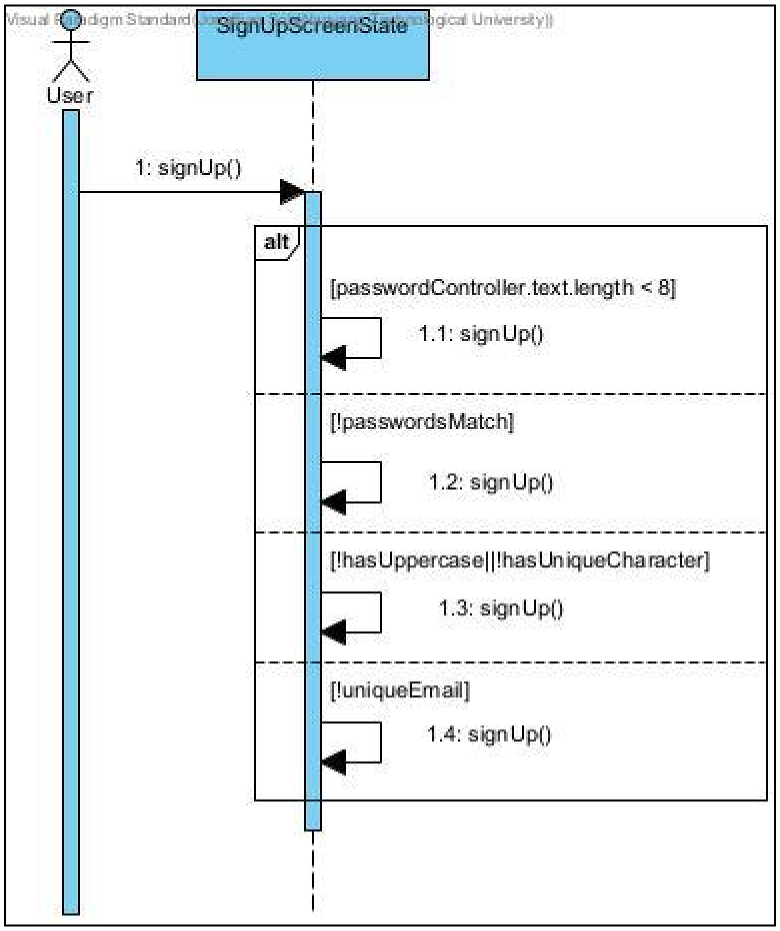

READY
entry / splashscreen of app
Back
Click Register
Click Login
Back
registerUser
do / Register Form Pop Up
loginUser
do / Login Form Pop Up
loadMap
entry / load the map generated by OpenStreetMapAPI
do / load carparks on the map
Click Setting
YES
userPreferencesForm
do / Load reportBug button, Logout button, respective preferences buttons and sliders
Logout
Logout
do / Pop up messsage "Are you sure you want to LogOut?"
Back
Click report bug
Apply
NO
ReportBug
do / load report bug descriptions
applyPreferences
do / save user preferences
Submit
SubmitBug
do / send data into backend

4.4 Sequence Diagram for Use Cases for FrontEnd

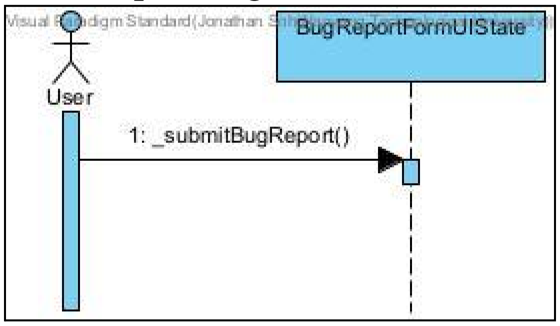
4.4.1. Login


Visual Paradigm Standard(Jonathan (Nanyang Technological University))
LoginScreenState
User
1: loginWithEmail()
alt
[auth.signInWithEmailAndPassword() == False]
1.1: saveCurrentUser(_currentUser!)
1.2: showProgressNotification('Login successful!', Colors.green)
[auth.signInWithEmailAndPassword() == True]
1.3: showProgressNotification('Login failed: $e', Colors.red)

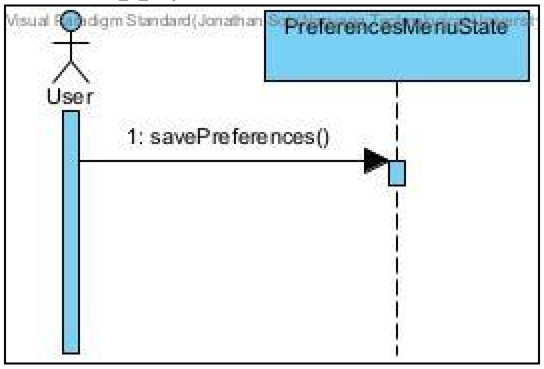
4.4.2 SignUp



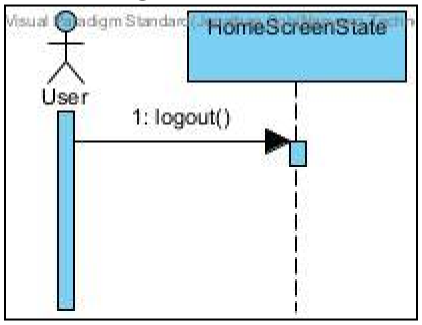
4.4.3 ReportBug



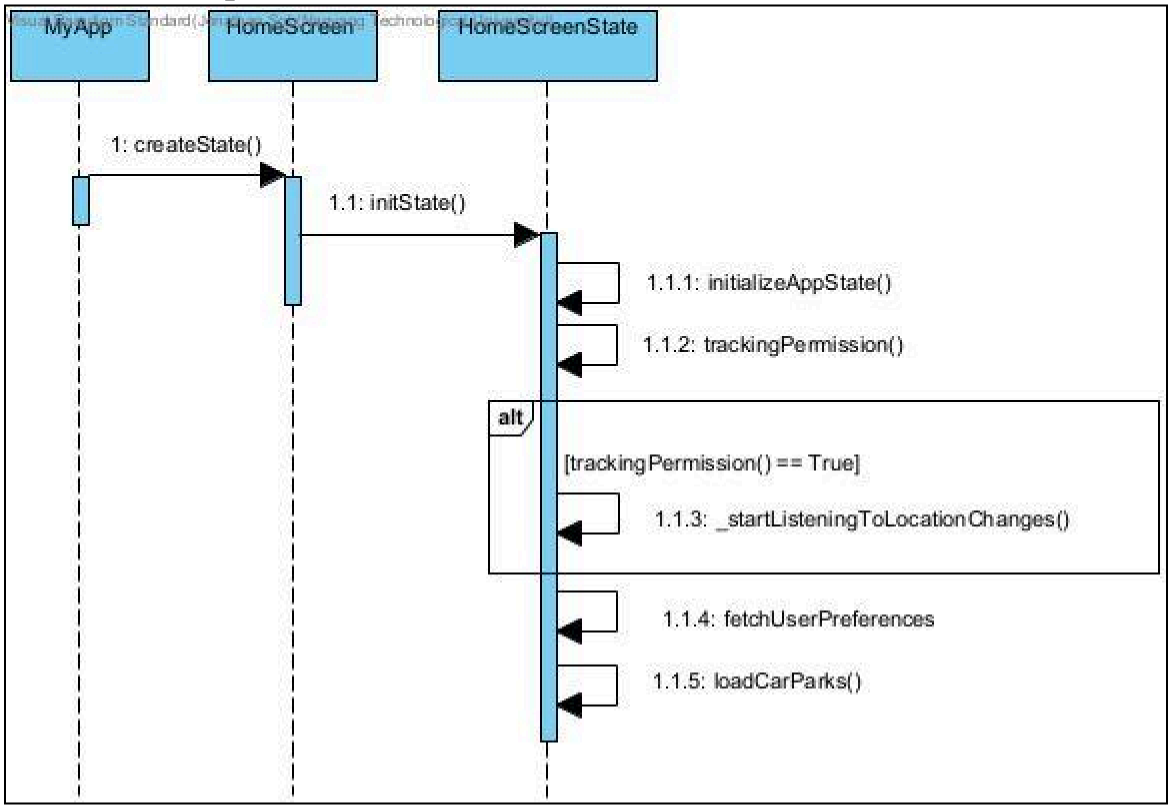
4.4.4 ApplyPreference



4.4.5 LogOut



4.4.6 LoadMap


MyApp
HomeScreen
HomeScreenState
1: createState()
1.1: initState()
1.1.1: initializeAppState()
1.1.2: trackingPermission()
alt
[trackingPermission() == True]
1.1.3: _startListeningToLocationChanges()
1.1.4: fetchUserPreferences
1.1.5: loadCarParks()

5. Key Design Issue

- API handling required a lot of computational power due to the sheer amount of data given out per call. Furthermore, the API requires re-validation every 24 hours which meant to combat this issue, we offloaded the scheduling and computational requirements into a Docker container that handles parsing, validation and writing to the database.

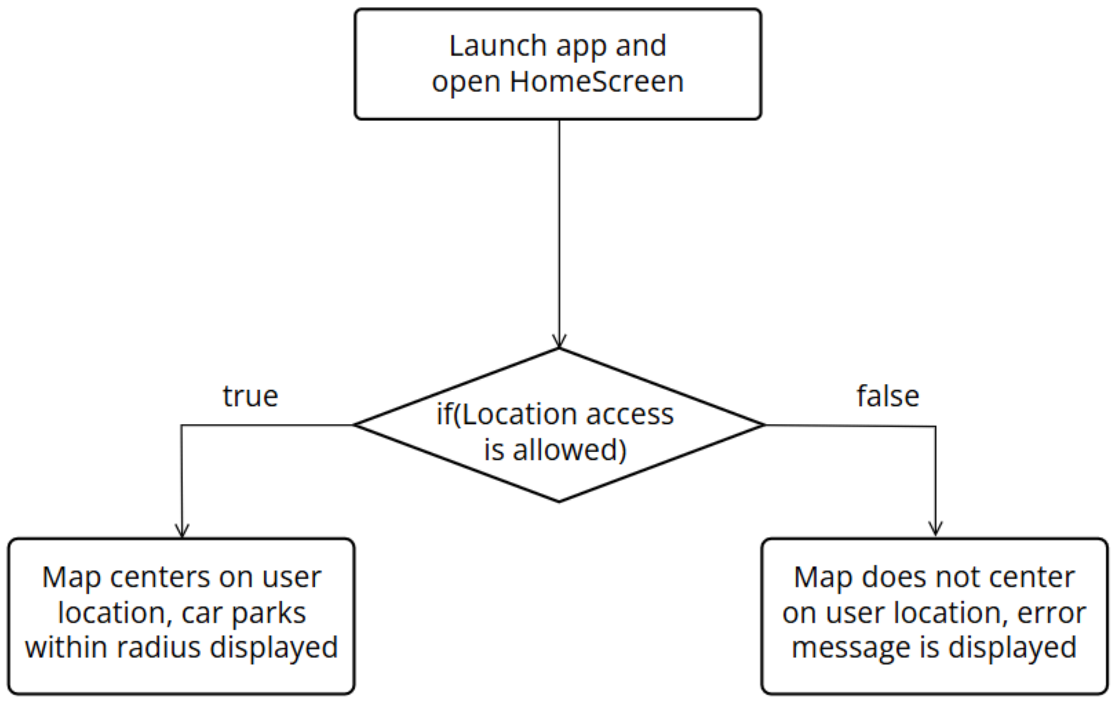
- API formatting for location was in SVY21 format, which is localized to Singapore. However, our Application required a conversion to WGS84 for every instance of a carpark, which required further computational power.

6. Application Skeleton

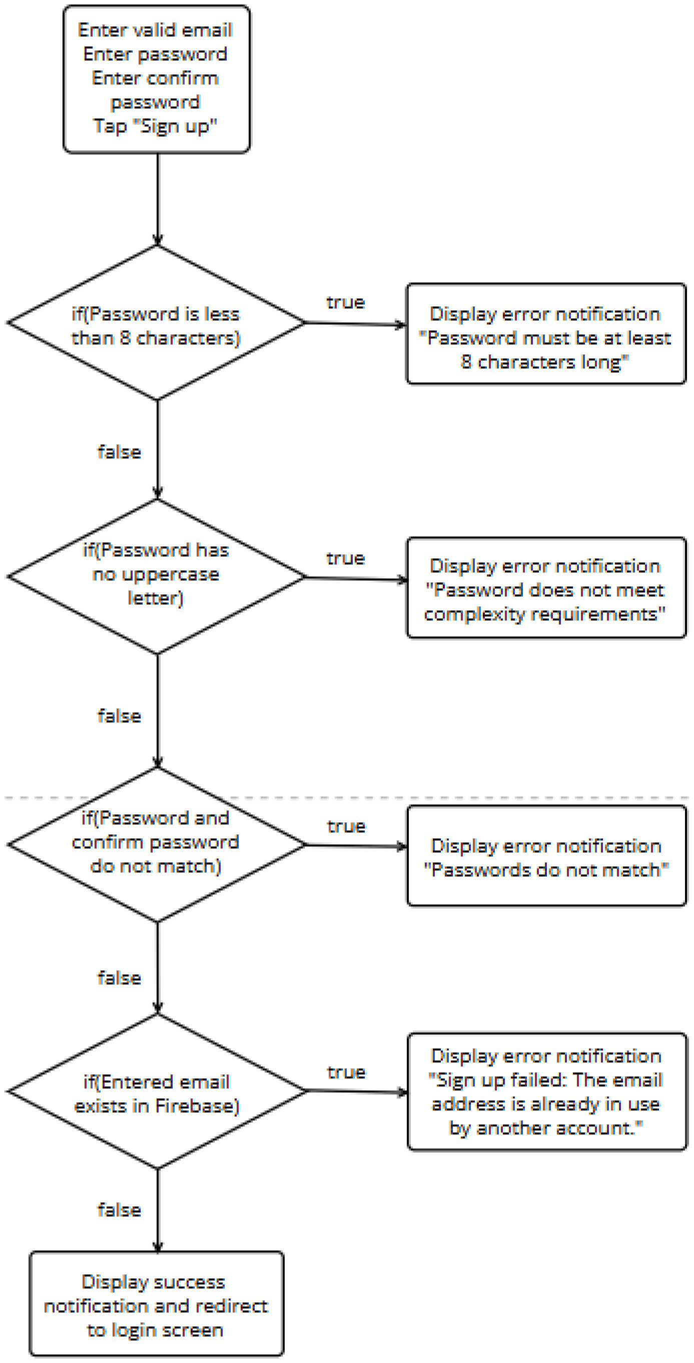
The application skeleton is located in Lab3/app_skeleton.

7. Testing FlowChart

7.1 Car Park Data Loading and Display


Launch app and open HomeScreen
if(Location access is allowed)
true
false
Map centers on user location, car parks within radius displayed
Map does not center on user location, error message is displayed

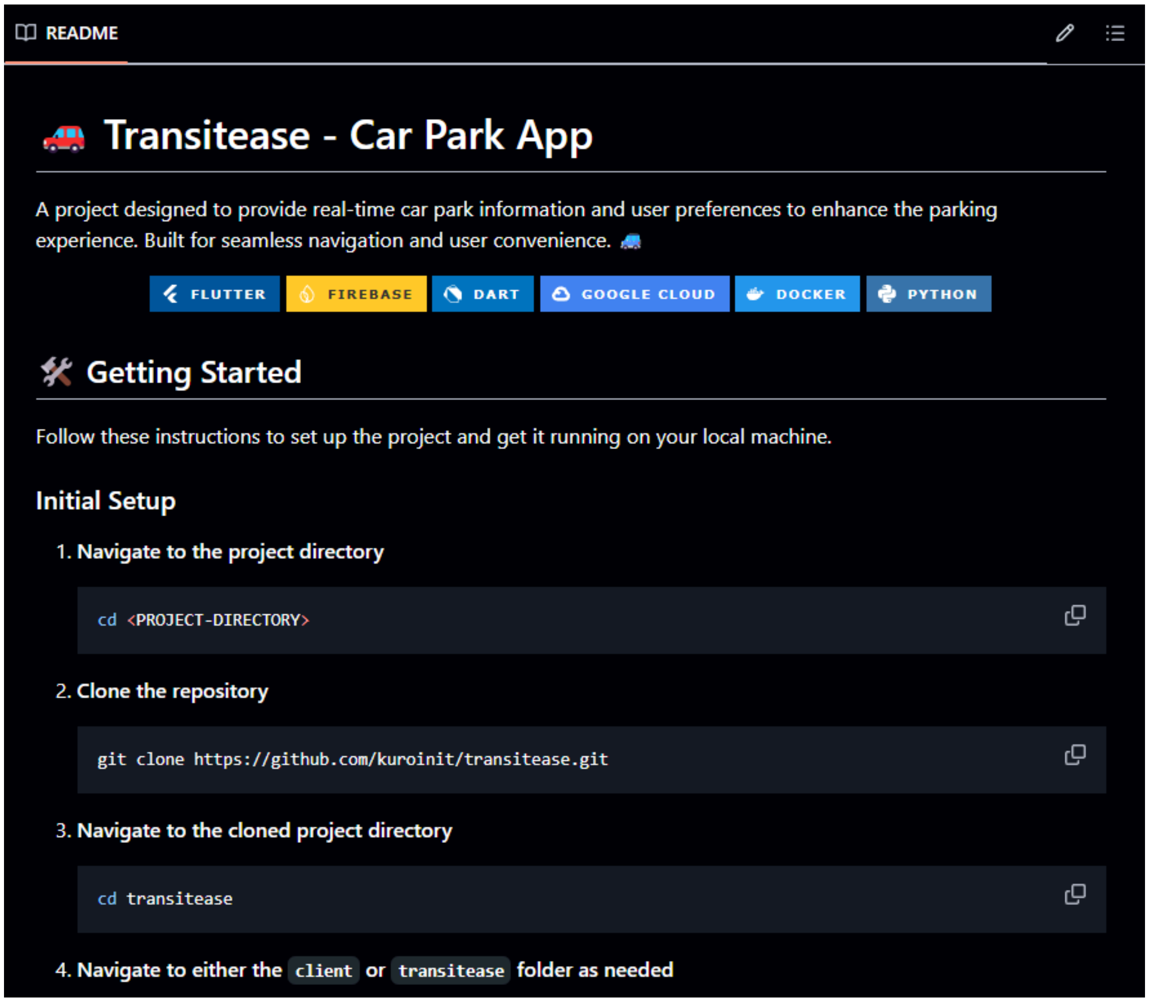
7.2 User Registration



Enter valid email
Enter password
Enter confirm password
Tap "Sign up"
if(Password is less than 8 characters)
true
Display error notification "Password must be at least 8 characters long"
false
if(Password has no uppercase letter)
true
Display error notification "Password does not meet complexity requirements"
false
if(Password and confirm password do not match)
true
Display error notification "Passwords do not match"
false
if(Entered email exists in Firebase)
true
Display error notification "Sign up failed: The email address is already in use by another account."
false
Display success notification and redirect to login screen

8. README

The README on github covers the description of our Car Park App - TransitEase. Additionally, it covers the step by step for easy integration by other developers. This ensures the robustness and the extensiveness of integration by other developers. Diagram below shows our README on github.


README

🚗 Transitease - Car Park App

A project designed to provide real-time car park information and user preferences to enhance the parking experience. Built for seamless navigation and user convenience. 🚙

🛠️ Getting Started

Follow these instructions to set up the project and get it running on your local machine.

Initial Setup

1. **Navigate to the project directory**

```
cd <PROJECT-DIRECTORY>
```

2. **Clone the repository**

```
git clone https://github.com/kuroinit/transitease.git
```

3. **Navigate to the cloned project directory**

```
cd transitease
```

4. **Navigate to either the `client` or `transitease` folder as needed**