Integration Testing

1. Testing Objective

To verify that the `home_screen.dart` and `signup_screen.dart` file functions as expected, including loading the map, password validation and user validation.

2. Testing Scope

The test scope for this project includes verifying key functionalities of the `home_screen.dart` and `signup_screen.dart` files in the Car Park app, specifically focusing on:

1. **Password Validation**: Testing password strength and complexity requirements in `signup_screen.dart` to ensure users meet the necessary criteria for creating secure passwords. This includes checks for:
    - Minimum length (8 characters)
    - Inclusion of uppercase letters
    - Presence of unique characters (e.g., symbols)
    - Matching passwords in both `Password` and `Confirm Password` fields
    - Handling cases where a user already exists in Firebase
2. **User Authentication and Registration**: Ensuring correct behavior for user authentication, including handling errors when attempting to register with an email that already exists in Firebase.
3. **Car Park Data Loading and Display**: Testing the functionality of `home_screen.dart` to confirm successful data retrieval and display of car park locations on a map, as well as handling location access permissions.

3. Test Cases for Integration testin

3.1 Car Park Data Loading and Display

Test Case ID	LoadCarParks-1A
Description	Load and display car parks from Firebase on the map – Positive Test Case
Priority	High
Prerequisite	Stable Internet connection and Firebase data available
Post-Requisite	None

Step No.	Action	Input	Expected Output	Actual Output	Device	Status	Comments
1	Launch app and open `HomeScreen`	-	Map and car park markers displayed	Map and car park markers displayed	iOS Simulator	Pass	Map loaded successfully
2	Allow location access	-	Map centers on user location, car parks within radius displayed	Map centers on user location, car parks within radius displayed	iOS Simulator	Pass	Car parks loaded successfully

Negative Test Cases

1. **Scenario**: Deny location access.
    - **Expected Output**: Map does not center on user location; display error message.

3.2. User Registration with Valid Credentials

Test Case ID	RegisterUser-1A
Description	Successful registration with valid email, matching passwords, and required complexity
Priority	High
Prerequisite	Stable internet connection; Firebase authentication set up
Post-Requisite	User should not be registered

Test Execution Steps

No.	Action	Input	Expected Output	Actual Output	Device	Status	Comments
1	Enter valid email	`user@example.com`	Email is accepted	Email is accepted	iOS Simulator	Pass	-
2	Enter password with required complexity	Password with uppercase, unique character, and 8+ characters	Password validation checklist shows green checks	Password validation checklist shows green checks	iOS Simulator	Pass	-
3	Enter matching confirm password	Same as password	Confirm password shows checkmark	Confirm password shows checkmark	iOS Simulator	Pass	-
4	Tap "Sign up"	-	Displays success notification and redirects to login screen	Displays success notification and redirects to login screen	iOS Simulator	Pass	Registration successful

3.3. Registration with Weak Password

Test Case ID	RegisterUser-2A
Description	Attempt to register with a password that does not meet complexity requirements
Priority	High
Prerequisite	Stable internet connection; Firebase authentication set up
Post-Requisite	User should not be registered

Test Execution Steps

Step No.	Action	Input	Expected Output	Actual Output	Device	Status	Comments
1	Enter valid email	`user@example.com`	Email is accepted	Email is accepted	iOS Simulator	Pass	-
2	Enter password missing complexity (e.g., no uppercase or unique character)	`password123`	Checklist displays red icons for missing criteria	Checklist displays red icons for missing criteria	iOS Simulator	Pass	-
3	Enter matching confirm password	Same as password	Confirm password shows checkmark, but password does not meet requirements	Confirm password shows checkmark, but password does not meet requirements	iOS Simulator	Pass	-
4	Tap "Sign up"	-	Displays error notification "Password does not meet complexity requirements"	Displays error notification "Password does not meet complexity requirements"	iOS Simulator	Pass	Password complexity validation works correctly

3.4. Registration without Unique Character

Test Case ID	RegisterUser-2A
Description	Attempt to register with a password that does not meet complexity requirements
Priority	High
Prerequisite	Stable internet connection; Firebase authentication set up
Post-Requisite	User should not be registered

Test Execution Steps

No.	Action	Input	Expected Output	Actual Output	Device	Status	Comments
1	Enter valid email	`user@example.com`	Email is accepted	Email is accepted	iOS Simulator	Pass	-
2	Enter password missing complexity (e.g., no uppercase or unique character)	`password123`	Checklist displays red icons for missing criteria	Checklist displays red icons for missing criteria	iOS Simulator	Pass	-
3	Enter matching confirm password	Same as password	Confirm password shows checkmark, but password does not meet requirements	Confirm password shows checkmark, but password does not meet requirements	iOS Simulator	Pass	-
4	Tap "Sign up"	-	Displays error notification "Password does not meet complexity requirements"	Displays error notification "Password does not meet complexity requirements"	iOS Simulator	Pass	Password complexity validation works correctly

3.5. Registration with Short Password

Test Case ID	PasswordValidation-1A
Description	Attempt to register with a password that is less than 8 characters
Priority	High
Prerequisite	User is on the `SignUpScreen`; Firebase authentication set up
Post-Requisite	User should not be registered

Test Execution Steps

No.	Action	Input	Expected Output	Actual Output	Device	Status	Comments
1	Enter password less than 8 characters	`Pass1!`	Password checklist shows a red icon for "Minimum 8 characters"	Password checklist shows a red icon for "Minimum 8 characters"	iOS Simulator	Pass	Length validation works
2	Tap "Sign up"	-	Displays error notification "Password must be at least 8 characters long"	Displays error notification "Password must be at least 8 characters long"	iOS Simulator	Pass	-

3.6. Password Without Uppercase Character

Test Case ID	PasswordValidation-2A
Description	Attempt to register with a password that lacks an uppercase character
Priority	Medium
Prerequisite	User is on the `SignUpScreen`; Firebase authentication set up
Post-Requisite	User should not be registered

Test Execution Steps

No.	Action	Input	Expected Output	Actual Output	Device	Status	Comments
1	Enter password without uppercase character	`password 1!`	Password checklist shows a red icon for "Contains uppercase letter"	Password checklist shows a red icon for "Contains uppercase letter"	iOS Simulator	Pass	Uppercase validation works
2	Tap "Sign up"	-	Displays error notification "Password does not meet complexity requirements"	Displays error notification "Password does not meet complexity requirements"	iOS Simulator	Pass	-

3.7. Password and Confirm Password Do Not Match

Test Case ID	PasswordValidation-4A
Description	Attempt to register with a password and confirm password that do not match
Priority	High
Prerequisite	User is on the `SignUpScreen`; Firebase authentication set up
Post-Requisite	User should not be registered

Test Execution Steps

No.	Action	Input	Expected Output	Actual Output	Device	Status	Comments
1	Enter password and confirm password that do not match	Password: `Password1!`, Confirm Password: `Password2!`	Password checklist shows a red icon for "Passwords match"	Password checklist shows a red icon for "Passwords match"	iOS	Pass	Matching validation works
2	Tap "Sign up"	-	Displays error notification "Passwords do not match"	Displays error notification "Passwords do not match"	iOS	Pass	-

3.8 . User Already Exists in Firebase

Test Case ID	PasswordValidation-6A
Description	Attempt to register with an email that already exists in Firebase
Priority	High
Pre-Requisite	Existing Firebase user with the same email as the one being registered
Post-Requisite	User should not be registered again, and an error message should be displayed

Test Execution Steps

No.	Action	Input	Expected Output	Actual Output	Device	Status	Comments
1	Enter an existing email in Firebase	`existing_user@example.com`	Email is accepted, but Firebase will check if the user already exists	Email is accepted	iOS	Pass	-
2	Enter valid password that meets all criteria	`Password1!`	Password checklist shows green icons for all criteria	Password checklist shows green icons for all criteria	iOS	Pass	-
3	Tap "Sign up"	-	Displays error notification "Sign up failed: The email address is already in use by another account."	Displays error notification "Sign up failed: The email address is already in use by another account."	iOS	Pass	Firebase correctly identifies duplicate user

Unit Testing

`carparkFromFirestore` Tests

These tests ensure that `carparkFromFirestore` accurately parses Firestore data into a `Carpark` object, handling various scenarios of data completeness and correctness.

1. **Test Case 1: Complete Data Parsing**
    - **Description**: Verify that `carparkFromFirestore` correctly parses and creates a `Carpark` object when all data fields are complete.
    - **Expected Outcome**: The returned `Carpark` object should contain accurate values for `carparkID`, `name`, `locationCoordinates`, `carCapacity`, `bikeCapacity`, and `realTimeAvailability`.
    - **Input**:
        i. Firebase Document Data

```
'carpark': {
          'ppCode': 'TEST01',
          'ppName': 'Test Carpark',
          'geometries': {
            'coordinates': [103.8198, 1.3521]
          },
          'vehCat': {
            'Car': {'parkCapacity': 100},
            'Motorcycle': {'parkCapacity': 50}
          },
        },
        'availability': {
          'C': {'lotsAvailable': 75},
          'M': {'lotsAvailable': 30}
        },
```

    - **Expected Output**:
        i. Carpark object with:
            - `carparkID`: "TEST01"
            - `name`: "Test Carpark"
            - `locationCoordinates`: `LatLng(1.3521, 103.8198)`
            - `carCapacity`: 100
            - `bikeCapacity`: 50
            - `realTimeAvailability`: true

2. **Test Case 2: Handling Missing Coordinates**
    - **Description:** Verify that `carparkFromFirestore` sets default coordinates if `coordinates` data is missing.
    - **Expected Outcome:** The returned `Carpark` object should have coordinates set to `LatLng(0, 0)`.
    - **Input:**
        i. FireBase Document Data:

```
'carpark': {
          'ppCode': 'TEST01',
          'ppName': 'Test Carpark',
          'geometries': {},
          'vehCat': {
            'Car': {'parkCapacity': 100},
            'Motorcycle': {'parkCapacity': 50}
          },
        },
```

    - **Expected Output:**
        i. Carpark object with:
            - `carparkID`: "TEST01"
            - `name`: "Test Carpark"
            - `locationCoordinates`: `LatLng(0, 0)`
            - `carCapacity`: 100
            - `bikeCapacity`: 50

3. **Test Case 3: Malformed Coordinates Handling**
    - Description: Check that `carparkFromFirestore` sets coordinates to `LatLng(0, 0)` if coordinates are malformed or `null`.
    - Input:
        i. Carpark object with:

```
'carpark': {
          'ppCode': 'TEST01',
          'ppName': 'Test Carpark',
          'geometries': {
            'coordinates': [null, null]
          },
          'vehCat': {
            'Car': {'parkCapacity': 100},
            'Motorcycle': {'parkCapacity': 50}
          },
        },
```

- **Expected Output:**
  i. Carpark object with:
     - `carparkID`: "TEST01"
     - `name`: "Test Carpark"
     - `locationCoordinates`: `LatLng(0, 0)`
     - `carCapacity`: 100
     - `bikeCapacity`: 50

4. **Test Case 4: Vehicle Categories Handling**
   - **Description**: Ensure that `carparkFromFirestore` assigns `0` as `carCapacity` and `bikeCapacity` when vehicle categories are invalid.
   - Input:
     i. Carpark object with:

```
'carpark': {
          'ppCode': 'TEST01',
          'ppName': 'Test Carpark',
          'geometries': {
            'coordinates': [103.8198, 1.3521]
          },
          'vehCat': {'Car': 'invalid', 'Motorcycle':
'invalid'},
        },
```

   - **Expected Output:**
     i. Carpark object with:
        - `carparkID`: "TEST01"
        - `name`: "Test Carpark"
        - `locationCoordinates`: `LatLng(0, 0)`
        - `carCapacity`: 0
        - `bikeCapacity`: 0

`calculateGeohashRange` and `haversineDistance` Tests

Test Description

These unit tests ensure the accuracy of geohash range calculations and distance measurements between latitude and longitude points. Each test is focused on verifying the output of individual functions without other dependencies.

calculateGeohashRange Tests

1. **Test Case 1: `calculateGeohashRange` Returns a Valid Geohash Range**
    - **Description**: Verify that `calculateGeohashRange` returns a list of two non-empty geohashes based on the provided latitude, longitude, and radius.
    - **Expected Outcome**: The output is a `List<String>` with two non-empty geohashes.
    - **Example Input**:
        i. Latitude: 1.3521
        ii. Longitude: 103.8198
        iii. Radius: 500 meters
    - **Expected Output**: `geohashRange` list with:
        i. `geohashRange.length`: 2
        ii. `geohashRange[0]`: Non-empty string
        iii. `geohashRange[1]`: Non-empty string

haversineDistance Tests

2. **Test Case 1: `haversineDistance` Calculates Distance Between Two Points**
    - **Description**: Ensure `haversineDistance` calculates the distance accurately between two latitude/longitude points.
    - **Expected Outcome**: The calculated distance should be within a reasonable range of the expected distance.
    - **Example Input**:
        i. `lat1`: 1.3521
        ii. `lon1`: 103.8198
        iii. `lat2`: 1.3525
        iv. `lon2`: 103.8202
    - **Expected Output**: Distance close to 56 meters with a tolerance of ±10 meters.

3. **Test Case 2: `haversineDistance` Calculates Zero for Identical Point**
    - **Description:** Verify that `haversineDistance` returns `0` when the two points have identical latitude and longitude values.
    - **Expected Outcome:** The calculated distance should be `0`.
    - **Example Input:**
        i. `lat1`: 1.3521
        ii. `lon1`: 103.8198
        iii. `lat2`: 1.3521
        iv. `lon2`: 103.8198
    - **Expected Output:** Distance equals `0`.

All Test Files for Unit Testing are under Lab Deliverables\lab_4\test.

1. Folder Contains
    a. Both Unit Testing Scripts
    b. Test results in test_results.txt
    c. Photographic proof of tests passing
    d. This document