

SC2006



PROJECT SKYSCAPE

TEAMWON

Li Yaohao
Lim Zhi Yong
Kieran Liow Jiun Kiat
Jin Ying
Kiersten Yeo Shu Xian
Yoon Sang Won

Table of Contents

01

Introduction

02

Use Case &
Functionalities

03

SE Practices

04

Live Demo

05

Conclusion

1. INTRODUCTION

NEVER MISS THE GOLDEN HOUR AGAIN.

About Skyscape

Purpose

To predict the quality of sunsets on any given day and bring everyone closer to nature's spectacle.

Target Audience

For casual users who occasionally enjoy sunsets, and also photography enthusiasts and nature lovers.

Introduction

SkyScape!

Never miss the golden hour again.

Email Address



Password



Don't have an account yet? Click [here](#) to register!

Login

About Skyscape

Motivation

- Skyscape is modelled after social media applications whereby users are able to interact with other users.
- Drawing inspiration from activity-tracking apps like "Strava," Skyscape places a strong emphasis on community involvement.
- This communal aspect is instrumental in retaining users.

Introduction



Community Stories

STRAVA

About Skyscape



- Extensive widget library and reactive framework streamline UI development
- Hot reload feature for faster development
- Declarative UI approach

2. USE CASE & FUNCTIONALITIES

NEVER MISS THE GOLDEN HOUR AGAIN.

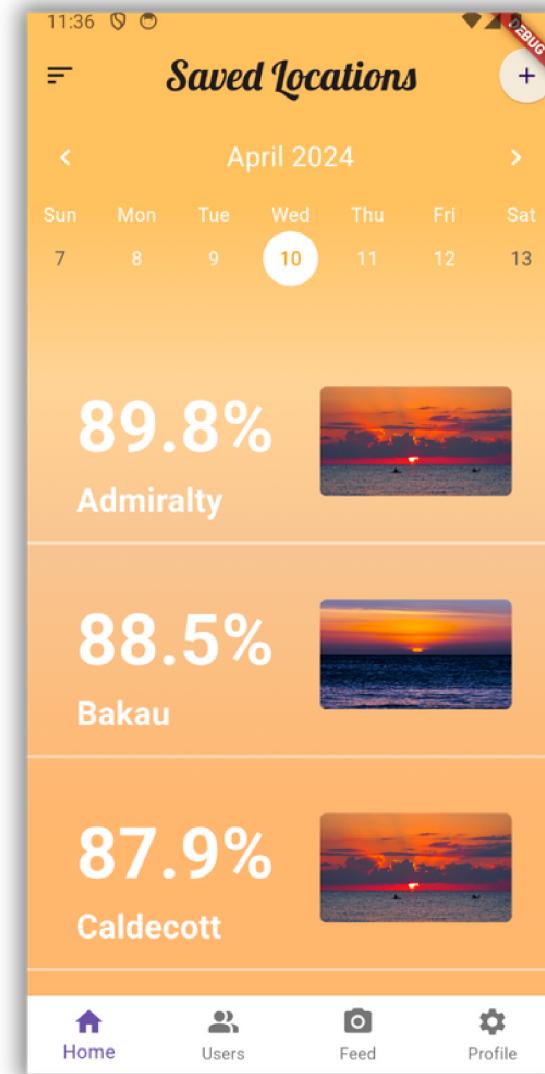
Skyscape

Brief Overview



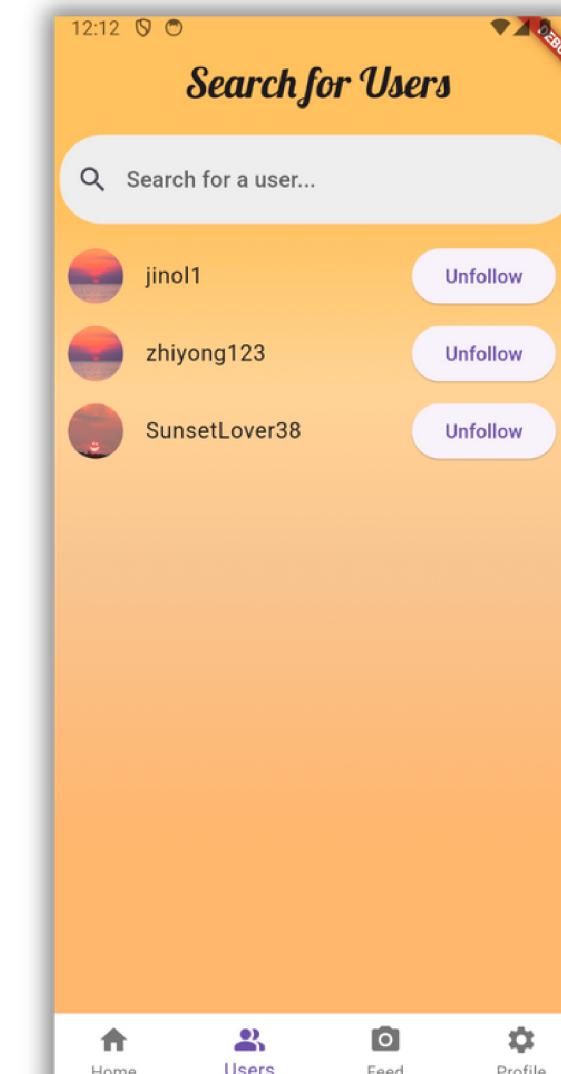
Quality of sunsets

Users can view the predicted quality of sunsets of any location on a chosen date on the home page



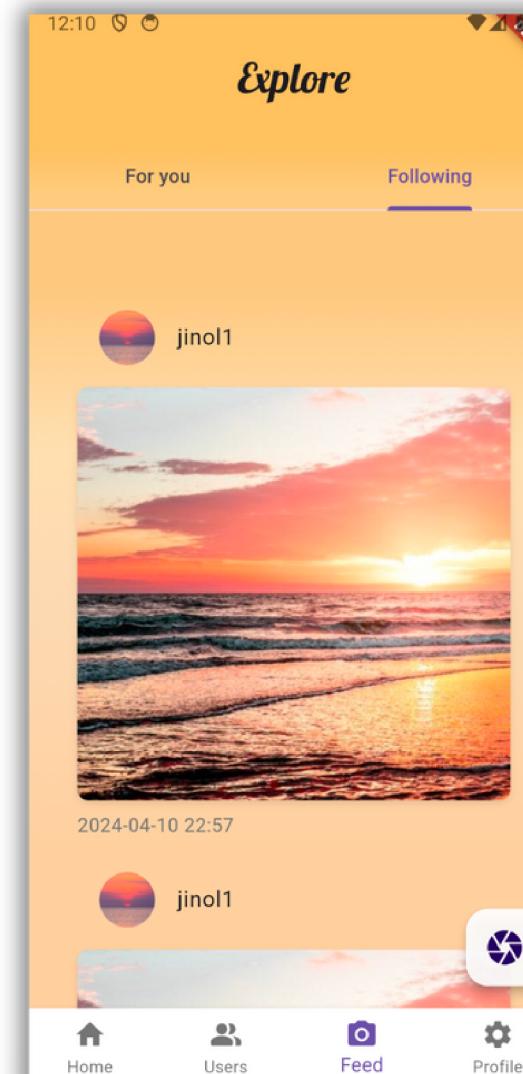
Follow Users

Users can follow other users to view the sunset images that they have uploaded

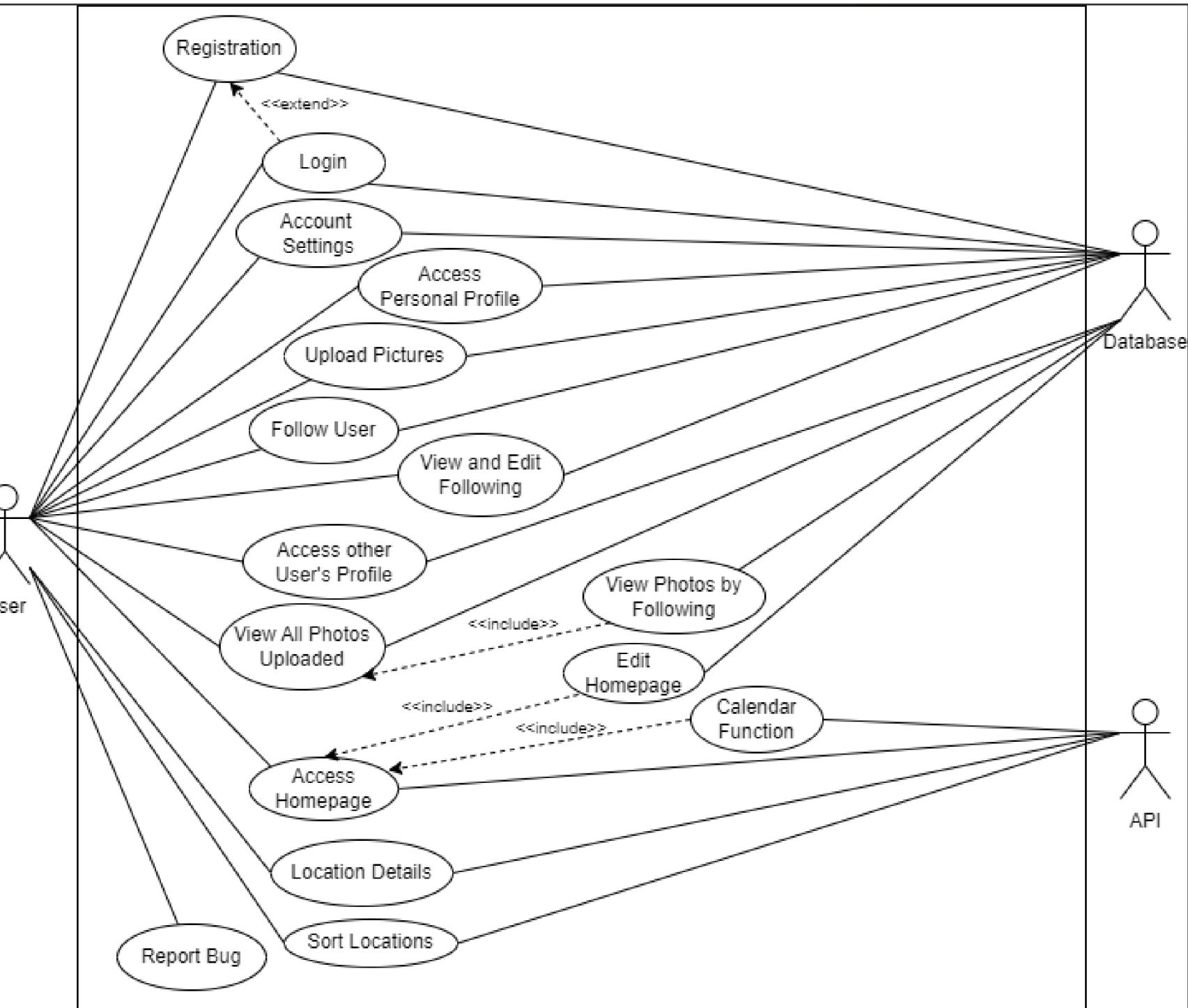


Upload Pictures

Users can upload pictures of sunsets that they have taken for memory and to share with other users



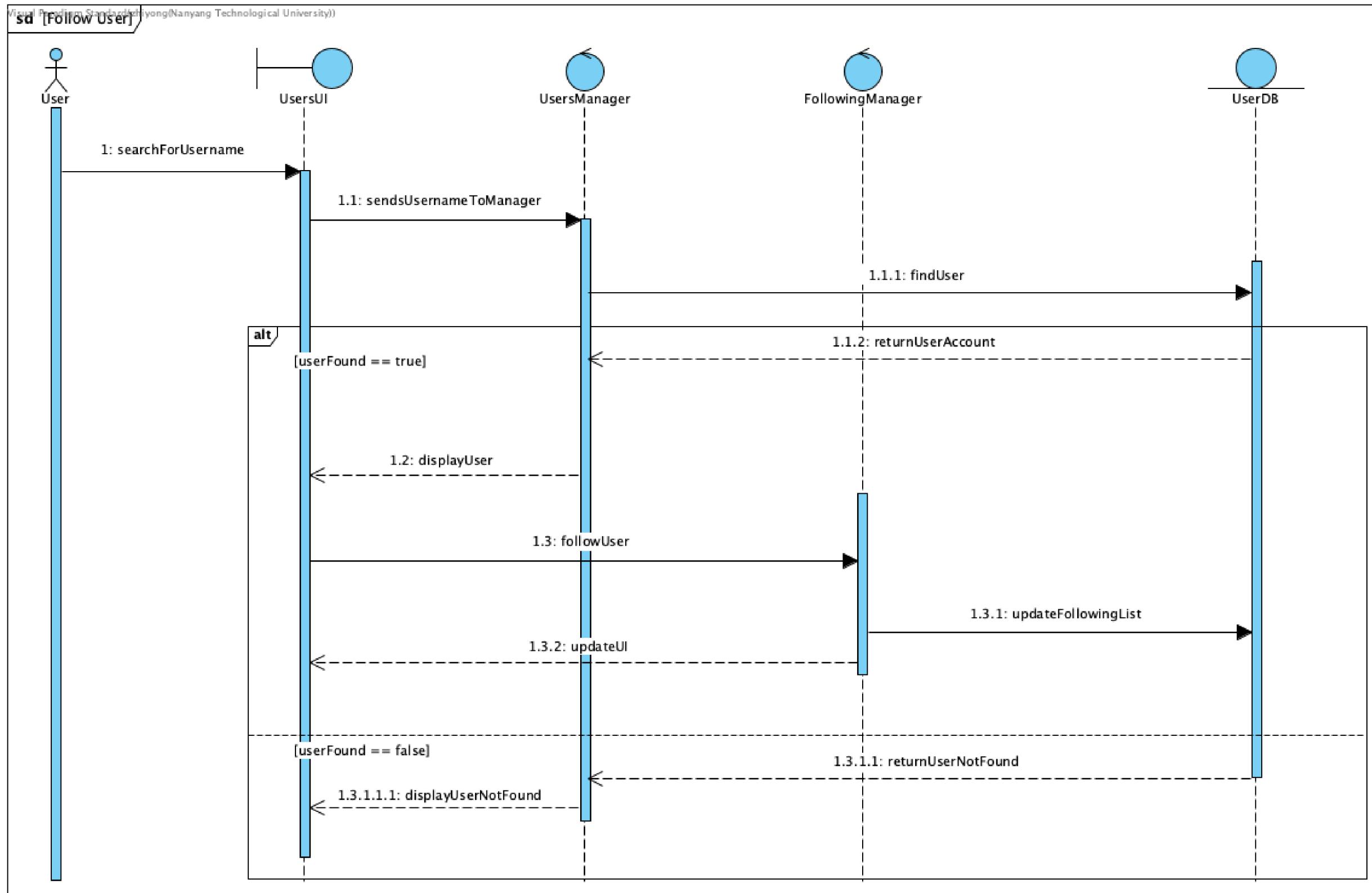
Complete Use Case Diagram



Use Case Diagram

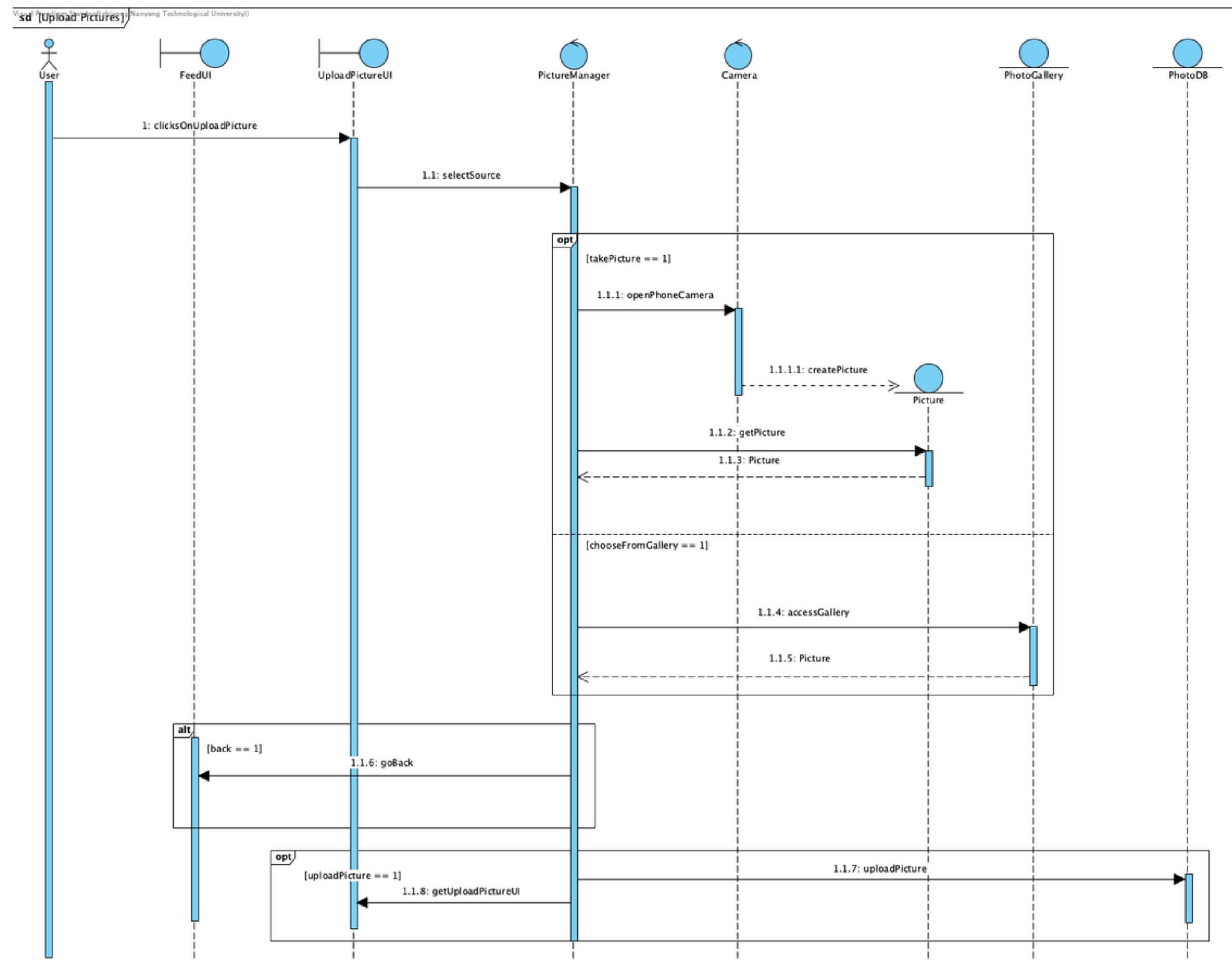
Main Functionalities

- Login/Register
- View Homepage
- View Location Details
- View Feed
- Upload Pictures
- Follow User
- View Other's Profile
- Report Bugs



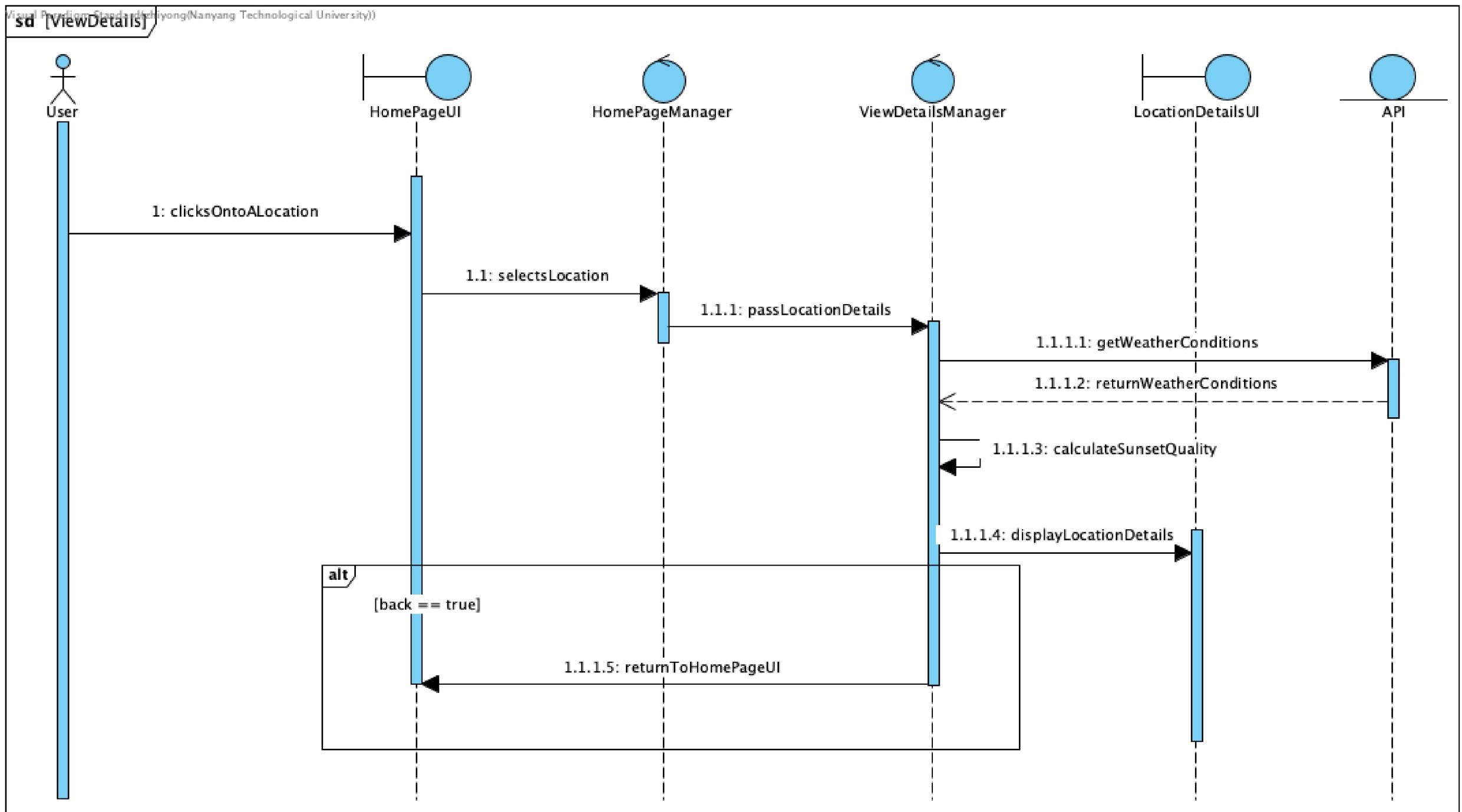
Follow User Functionality

User is able to search the username of other users he wishes to find and follow that user



Upload Pictures Functionality

User is able to select a photo from his gallery or take a photo using the phone camera and upload the photos to the feed for others to see



View Detail Functionality

User is able to view the weather details of the location he chooses as well as the calculated sunset quality with the information obtained from API

3. GOOD SE PRACTICES

NEVER MISS THE GOLDEN HOUR AGAIN.

**3 Layered
Architecture**

**DRY - Don't
Repeat Yourself**

**SRP - Single
Responsibility
Principle**

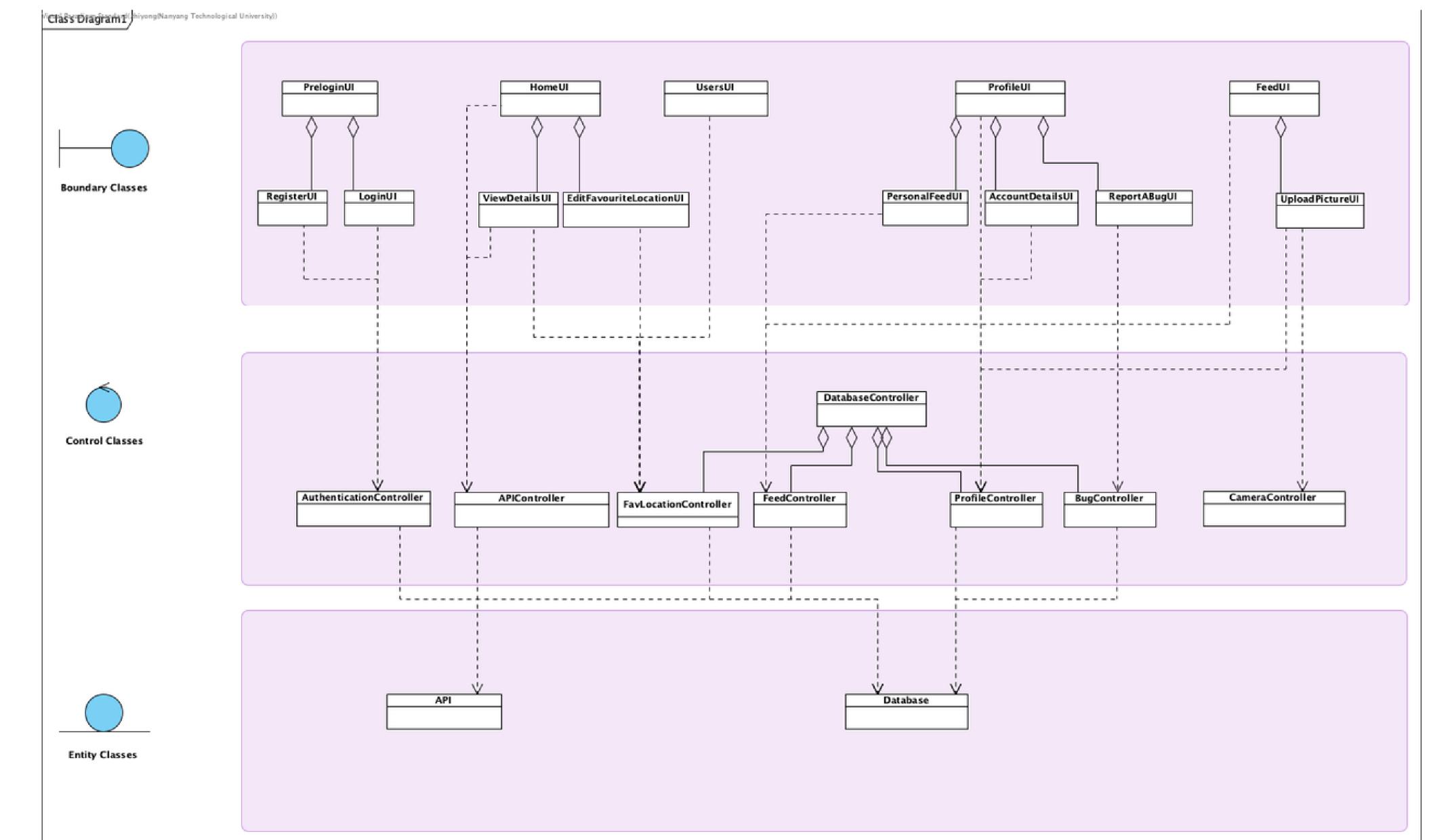
**Black box
Testing**

3 Layered Architecture

A well-established software application architecture that organizes applications into three logical and physical computing tiers.

Benefits

1. Accelerated Development
2. Improved Scalability
3. Loose Coupling



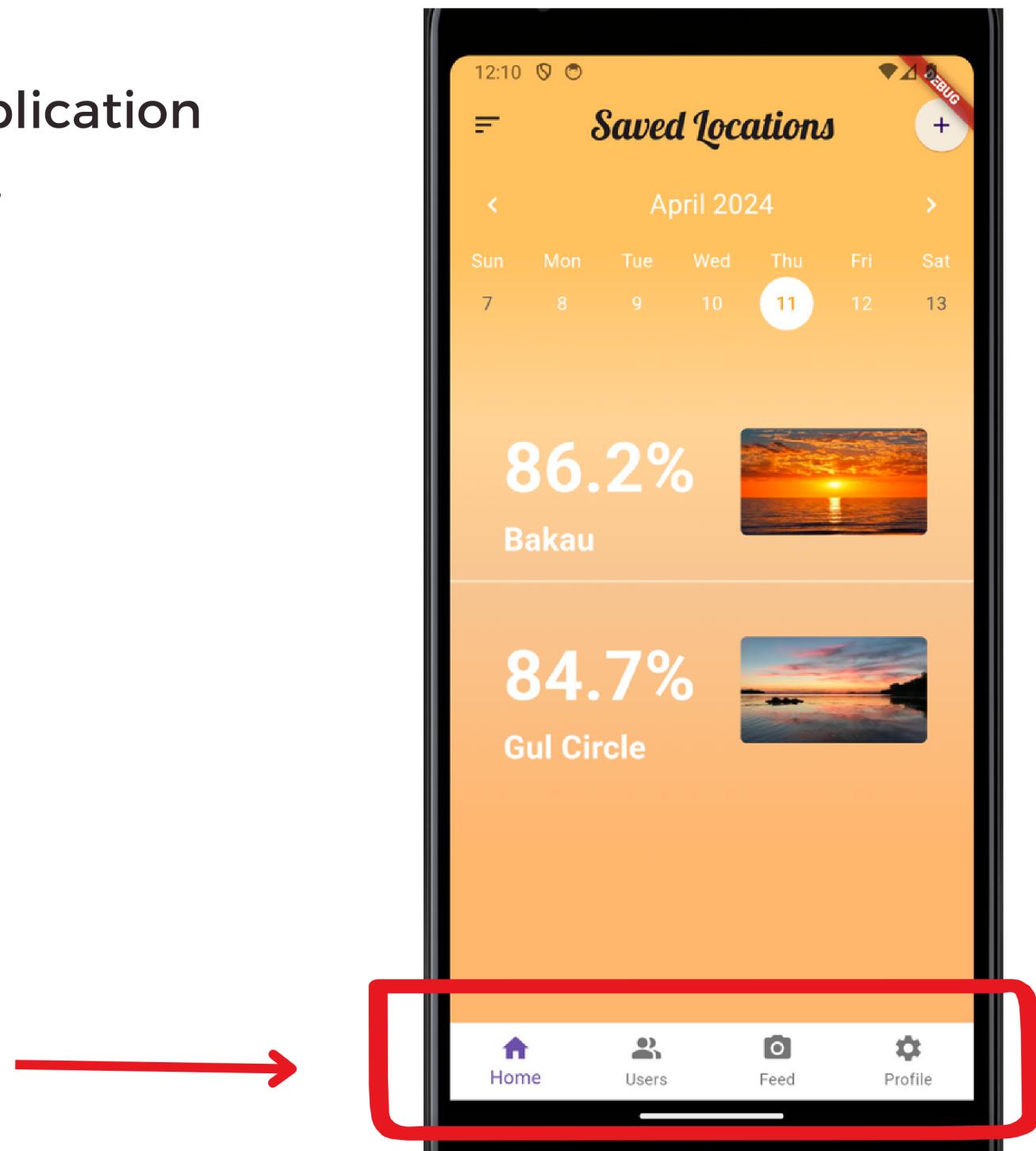
DRY

A principle of software development that aims at **reducing the repetition** of patterns and code duplication in favor of abstractions and avoiding redundancy.
(Muldrow, 2020)

Benefits

1. Improved Code Readability
2. Better Performance
3. Better Maintainability

DRY - Don't Repeat Yourself



**Build the body around
the fixed template
rather than include the
template on individual
bodies.**

```
index: currentIndex,
children: [
  buildHomeScreen(),
  SearchUsers(),
  FeedPage(),
  ProfileMainWidget(),
],
), // IndexedStack
), // Container
// ),
bottomNavigationBar: BottomNavigationBar(
  backgroundColor: Color.fromARGB(255, 255, 255, 255),
  type: BottomNavigationBarType.fixed,
  currentIndex: currentIndex,
  onTap: (index) => setState(() => currentIndex = index),
  items: const [
    BottomNavigationBarItem(
      icon: Icon(Icons.home),
      label: 'Home',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon(Icons.people_alt_rounded),
      label: 'Users',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon(Icons.camera_alt),
      label: 'Feed',
    ), // BottomNavigationBarItem
    BottomNavigationBarItem(
      icon: Icon(Icons.settings),
      label: 'Profile',
    ), // BottomNavigationBarItem
  ],
), // BottomNavigationBar
```

**DRY - Don't
Repeat Yourself**

**Implementation for the
Bottom Navigation Bar**

A class should have only **one reason to change**, in other words, a class should have a single responsibility or concern. (Kant, 2023)

Benefits

1. Simplifies Debugging
2. Improved Code Ownership

SRP - Single Responsibility Principle

```
Future<List<String>> getFavouritedLocations() async {
  DocumentSnapshot snapshot = await userCollection.doc(uid).get();
  if (snapshot.exists) {
    List<dynamic> locations = snapshot.get('favouritedLocations');
    return locations.map((location) => location.toString()).toList();
  }
  return [];
}

Future<void> sortFavouritedLocations(List<String> locationList) async {
  return await userCollection.doc(uid).update({
    'favouritedLocations': locationList,
  });
}

Future<String> findUsernameFromUID(String userid) async {
  DocumentSnapshot snapshot = await userCollection.doc(userid).get();
  if (snapshot.exists) {
    String username = snapshot.get('username');
    return username;
  }
  return "No user found.";
}

Future<String> findUIDFromUsername(String username) async {
  QuerySnapshot snapshot = await userCollection.where('username', isEqualTo: username).get();
  if (snapshot.docs.isNotEmpty) {
    String userid = snapshot.docs.first.id;
    return userid;
  }
  return "No user found";
}
```

Black box Testing

A good software engineering practice is to test the software using equivalence class testing to **find bugs**

Benefits

1. Directs tester to a very small subset of test inputs that can highly likely find bugs

Login Functionalities

Test ID	Test Case Description	Input: Email	Input: Password	Expected Output	Actual Output	Pass/Fail
EC1	Verify successful login with valid email and password	sunsetlover72@gmail.com	SUNSETlover72	User is logged in successfully	User is logged in successfully	Pass
EC2	Verify error message when email is invalid	sunsetlover72	SUNSETlover72	Error message "Invalid email or password"	Error message "Invalid email or password"	Pass
EC3	Verify error message when password is invalid	sunsetlover72@gmail.com	sunsetlover72	Error message "Invalid email or password"	Error message "Invalid email or password"	Pass
EC4	Verify error message when email is unregistered	sunsetlover73@gmail.com	SUNSETlover73	Error message "Invalid email or password"	Error message "Invalid email or password"	Pass
EC5	Verify error message when email is empty		SUNSETlover72	Error message "Enter an email"	Error message "Enter an email"	Pass
EC6	Verify error message when password is empty	sunsetlover72@gmail.com		Error message "Enter a valid password"	Error message "Enter a valid password"	Pass

Login Functionalities

Equivalence class testing

Testing only one invalid input for each test case, allowing us to learn the error handling or behavior for each type of invalid input

Test ID	Test Case Description	Input: Email	Input: Password	Expected Output	Actual Output	Pass/Fail
EC1	Verify successful login with valid email and password	sunsetlover72@gmail.com	SUNSETlover72	User is logged in successfully	User is logged in successfully	Pass
EC2	Verify error message when email is invalid	sunsetlover72	SUNSETlover72	Error message "Invalid email or password"	Error message "Invalid email or password"	Pass
EC3	Verify error message when password is invalid	sunsetlover72@gmail.com	sunsetlover72	Error message "Invalid email or password"	Error message "Invalid email or password"	Pass
EC4	Verify error message when email is unregistered	sunsetlover73@gmail.com	SUNSETlover73	Error message "Invalid email or password"	Error message "Invalid email or password"	Pass
EC5	Verify error message when email is empty		SUNSETlover72	Error message "Enter an email"	Error message "Enter an email"	Pass
EC6	Verify error message when password is empty	sunsetlover72@gmail.com		Error message "Enter a valid password"	Error message "Enter a valid password"	Pass

4. GOOD TRACEABILITY PRACTICES

NEVER MISS THE GOLDEN HOUR AGAIN.

**Version traceability**

Usage of Github/Git to help in maintaining a history of changes to code files.

**Design traceability**

Code followed UI mockups and documentations done previously

**Requirements traceability**

Code followed closely to Use case descriptions, as well as functional and non functional requirements

Skyscape

The screenshot shows a GitHub Desktop interface with the following details:

- Current Repository:** Skyscape
- Current Branch:** main
- Last fetched:** 8 minutes ago

An update message indicates an available update for GitHub Desktop.

The main area displays a list of changes and a detailed code diff for a file named `lib/screens/home/home.dart`. The commit message is "Updated the sort button".

Changes: 1

History: An updated version of GitHub Desktop is available and will be installed at the next launch. See [what's new](#) or [restart GitHub Desktop](#).

Updated the sort button

Constraints added for bugreport

Message popup on update details

Color changes in bugreport

changed profile to personal feed

Updated the sort button

changes to sort function

changes to feed UI to upload fas...

UI standardisation for main pages

View details parameters

Sort function implemented

Scrollable appbar

UI changes

changed font size for viewdetails air...

Revert "Revert "Merge branch 'main'..."

Revert "Merge branch 'main' of https://github.com/kringyfam/skyscape"

Merge branch 'main' of https://github.com/kringyfam/skyscape

fixed the password error UI

Photos function fully implemented

feed page working perfectly now...

lib/screens/home/home.dart

```
@@ -18,7 +18,7 @@ import 'package:skyscape/screens/home/viewdetails.dart';
class Home extends StatefulWidget {
    const Home({Key? key}) : super(key: key);
}
@Override
State<Home> createState() => _HomeState();
}

@@ -138,22 +138,22 @@ class _HomeState extends State<Home> {
}

Future<void> _sortLocations() async {
    setState(() {
        if (_isSortedAlphabetically) {
            if (input == 0) {
                // Sort by quality of sunset
                favouredLocationNames.sort((a, b) => allValues[b]?[5].compareTo(allValues[a]?[5]));
            }
            _isSortedAlphabetically = false;
        } else {
            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(duration: Duration(seconds: 1), content: Text('Sorted by Quality of Sunset')),
            );
        }
    });
    favouredLocationNames.sort((a, b) => allValues[b]?[5].compareTo(allValues[a]?[5]));
}
_isSortedAlphabetically = false;
ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(duration: Duration(seconds: 1), content: Text('Sorted by Quality of Sunset')),
);
}

} else {
    // Sort by alphabetical order
    favouredLocationNames.sort();
    _isSortedAlphabetically = true;
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(duration: Duration(seconds: 1), content: Text('Sorted by Alphabetical Order')),
    );
}
favouredLocationNames.sort();
_isSortedAlphabetically = true;
ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(duration: Duration(seconds: 1), content: Text('Sorted by Quality of Sunset')),
);
```

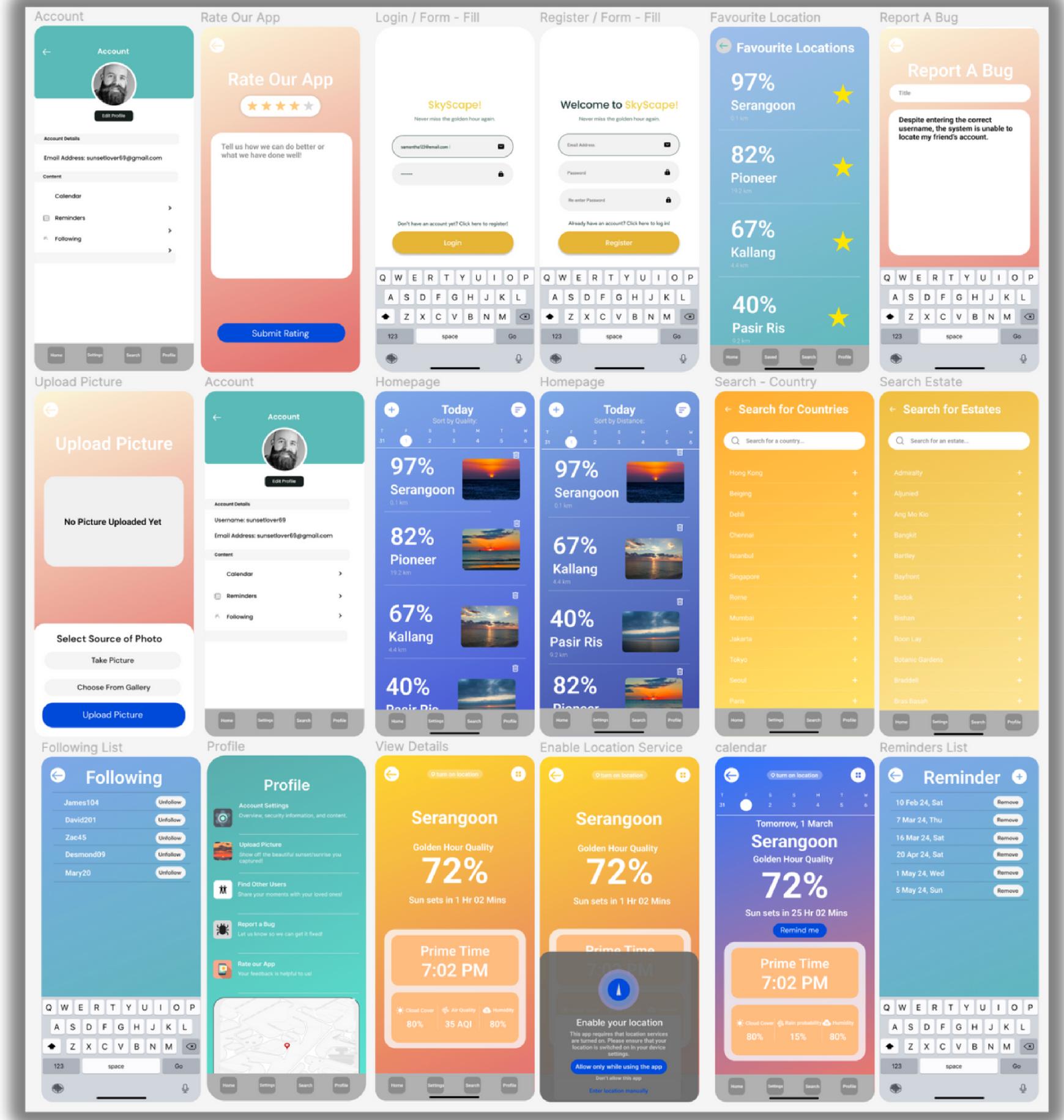
Traceability

Version Traceability

GitHub/Git helps with version control in traceability by maintaining a history of changes to code and related artifacts.

This enabled our team to track the changes in design, functions and implementations over time.

Skyscape



Traceability

Design Traceability

Following UI mockups during coding helps with design traceability by ensuring that the implemented code accurately reflects the intended user interface and design decisions.

Also facilitates alignment between the design and implementation stages of software development.

1. Use Case Descriptions

1.1. Use case of Registration

Use Case ID:	REG1
Use Case Name:	Registration
Created By:	Li Yaohao
Date Created:	30/01/24
Date Last Updated:	

1.2. Use case of Login

Use Case ID:	LOG1
Use Case Name:	Login
Created By:	Li Yaohao
Date Created:	06/02/2024
Date Last Updated:	

1.3. Use case of Access Homepage

Use Case ID:	HOME1
Use Case Name:	Access Homepage
Created By:	Li Yaohao
Last Updated By:	
Created Date:	30/01/24
Date Last Updated:	

1.5. Use case of Calendar Function

Use Case ID:	CALENDAR1
Use Case Name:	Calendar Function
Created By:	Li Yaohao
Date Created:	06/02/2024
Date Last Updated:	

1.8. Use case of Upload Pictures

Use Case ID:	POST1
Use Case Name:	Upload Pictures
Created By:	Lim Zhi Yong
Last Updated By:	
Created Date:	06/02/2024
Date Last Updated:	

1.9. Use case of Follow User

Use Case ID:	FOL1
Use Case Name:	Follow User
Created By:	Lim Zhi Yong
Date Created:	06/02/2024
Date Last Updated:	

1.11. Use case of Account Settings

Use Case ID:	ACC1
Use Case Name:	Edit Account Details
Created By:	Lim Zhi Yong
Last Updated By:	
Created Date:	06/02/2024
Date Last Updated:	

1.12. Use case of Report Bug

Use Case ID:	REPI
Use Case Name:	Report Bug
Created By:	Lim Zhi Yong
Date Created:	06/02/2024
Date Last Updated:	

1.16. Use case of Access Personal Profile

Use Case ID:	PPF1
Use Case Name:	Access Personal Profile
Created By:	Li Yaohao
Last Updated By:	
Created Date:	28/03/2024
Date Last Updated:	

Actor: User (initiating)

Description: User can report a bug that has been found in the system.

Preconditions: A bug has been found in the system.

Postconditions: Nil

Priority: High

Frequency of Use: Low

Flow of Events:

- Upon entering the system, the user will click on the navigation bar located at the top of the page.
- The user will click on the "Report a Bug" link.
- The user will enter the details of the bug.
- The system will log the bug and send an email to the administrator.

Alternative Flows: POST1-AF-S1 If the user does not have a profile, they will be prompted to create one.

Exceptions: Nil

Includes: Nil

Special Requirements: Nil

Non-Functional Requirements:

Usability	<ul style="list-style-type: none"> The user must be able to choose between multiple languages (English, Chinese, Malay, Thai). The user interface must be intuitive and easy to use for users from different levels of technical expertise. The system should be able to handle multiple users simultaneously without impacting performance.
Reliability	<ul style="list-style-type: none"> After a system reboot, the system must be able to recover all data within 2 minutes.

Functional requirements

- The system must have a registration function.
 - The registration form must collect basic user information.
 - The form must consist of fields for username, password, email, and location.
 - The user must be able to upload a profile picture.
 - The form must validate input fields to ensure they meet specific criteria (e.g., minimum length, alphanumeric).
- The system must verify user login.
 - The system must verify the provided credentials against the database.
 - If successful, the user must be logged in and directed to their personal profile page.
- The system must verify user account settings.
 - The user must be able to edit their account details.
 - The system must update the database with the new information.
 - The user must receive a confirmation message upon successful update.
- The system must verify user reporting of bugs.
 - The user must be able to report a bug.
 - The system must log the bug and notify the administrator.
 - The user must receive a confirmation message upon successful reporting.
- The system must verify user access to personal profiles.
 - The user must be able to view their own profile.
 - The system must display the user's profile information, including their name, email, and uploaded photos.
 - The user must be able to edit their profile information.

Requirements Traceability

Following use case descriptions and functional and non-functional requirements during coding helps with requirement traceability by ensuring that the implemented code directly addresses and fulfills the specified functionalities.

This establishes a clear link between the software's capabilities and its intended purpose.

5. DEMONSTRATION

NEVER MISS THE GOLDEN HOUR AGAIN.

