NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

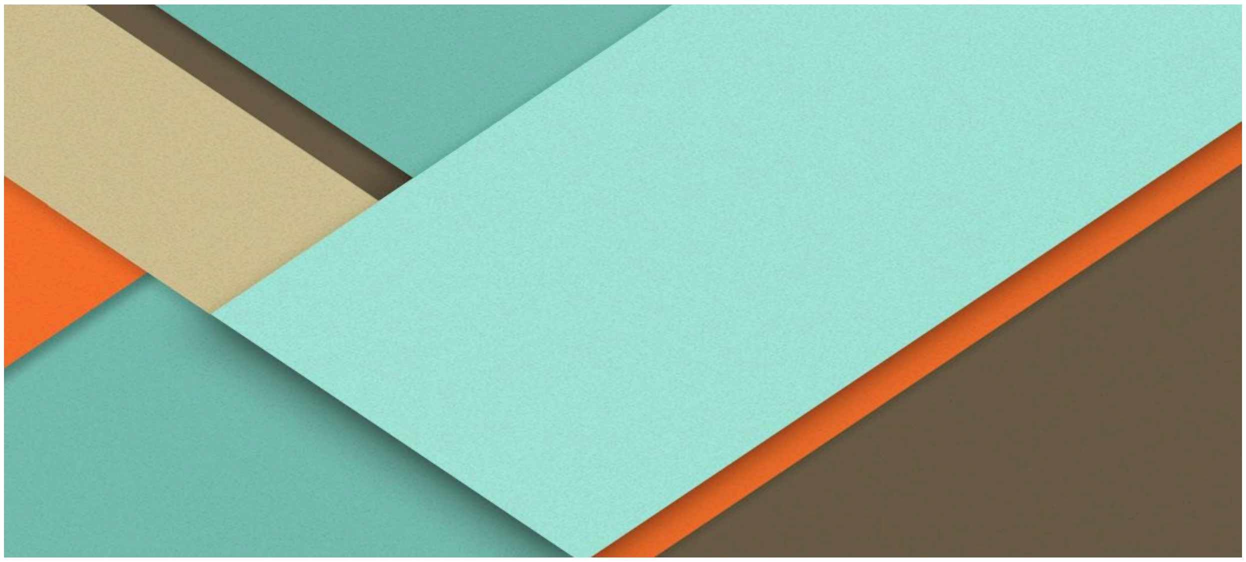
SC2006 Software Engineering

Software Requirement Specification Report

Tutorial Group: SCSA

Delete Save

Name	Matriculation Number
Celyna Teo	U2221037F
Joy Kuan	U2221261D
Jden Goh	U2222711B
Neo Wei	U2123800F
Tian Yidong	U2220492B
Kenneth Lemuel	U2222928H



Software Requirement Specification - EatAlready

Prepared by Team DeleteSave
Group 4

Celyna Teo (U2221037F)

Joy Kuan (U2221261D)

Jden Goh (U2222711B)

Neo Wei (U2123800F)

Tian Yidong (U2220492B)

Kenneth Lemuel (U2222928H)

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This Software Requirement Specification (SRS) document aims to outline the requirements for the development and deployment of our team's web application, serving as a guide to all stakeholders. This document will include system features, limitations, functional and non-functional requirements, as well as our use case models and design models. Our team will develop a Web Application, EatAlready, which aims to help users create itinerary recommendations in a restaurant's vicinity. The project will be complete when we can provide relevant recommendations based on the user's selected restaurant location. The project will support store owners in promoting their store to the intended target audience, and at the same time benefit our users to find locations that match their interests. This EatAlready web application will be built using Django and SQLite for the backend, and Google Maps API.

1.2 Document Conventions

This Software Requirement Specification (SRS) document will follow the standard typographical conventions, using headings and subheadings to organize content. Appendix A contains the list of definitions for the special terms used during this report.

Heading 1	Times New Roman, Bold, Size 20
Heading 2	Times New Roman, Bold, Size 16
Heading 3	Times New Roman, Bold, Underlined, Size 14
Normal Text	Size 12
Font	Times New Roman font, Black color

1.3 Intended Audience and Reading Suggestions

The following section outlines the intended readers of this Software Requirement Specification (SRS) document. The stakeholders include EatAlready users, the project manager, and the web application's development and testing team.

This documentation will begin with the introduction, followed by Functional and Non-functional requirements. It then showcases the User Interface and concludes with the appendix, which contains the data dictionary and analysis models.

1.4 Product Scope

The EatAlready web application will allow users to be able to register and log into accounts, select a restaurant based on a chosen area, and add activities near the selected restaurant to an itinerary.

1.5 Reference

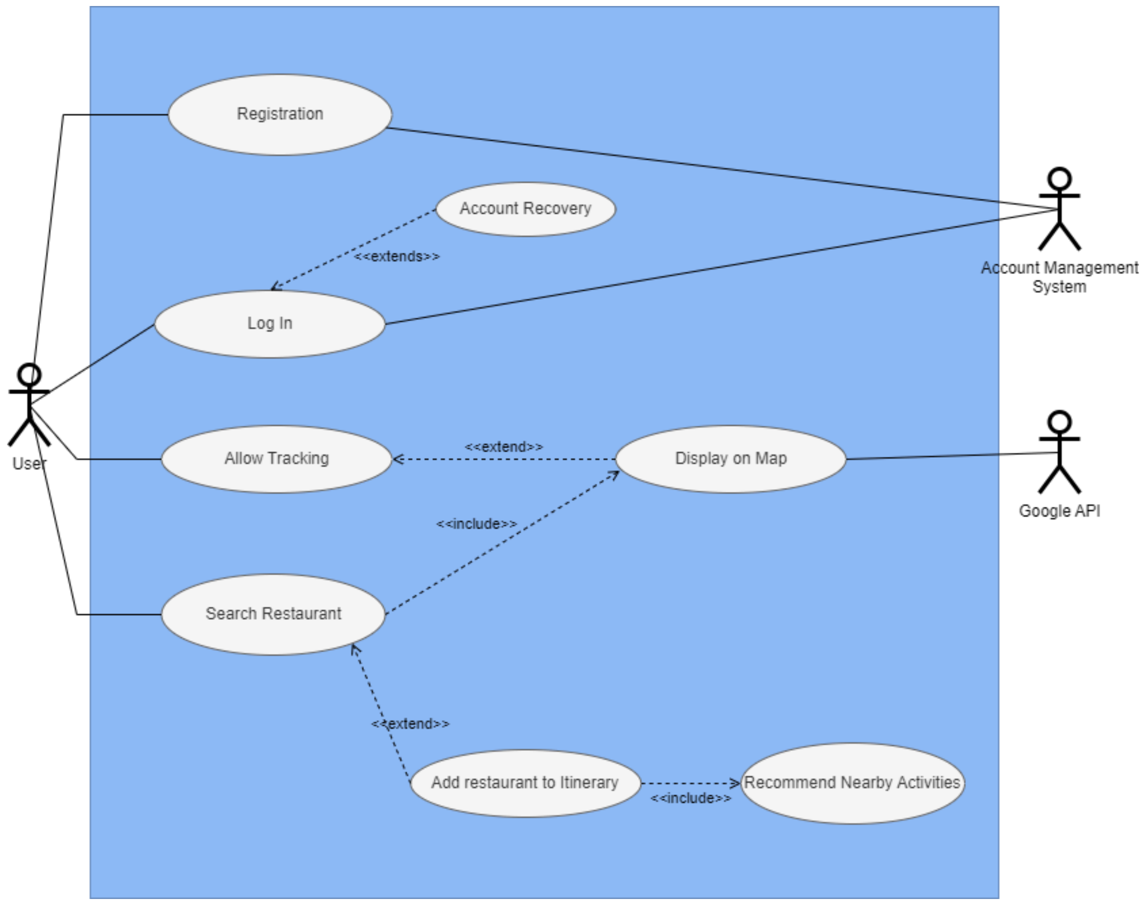
No external materials were used in this report.

2. Overall Description

2.1 Product Perspective

EatAlready is a web application built using Django for the backend and Google Maps API for location service. This application serves as a self-contained product, aiming to aid users in creating an itinerary centered around their selected restaurant choice.

2.2 Product Functions



EatAlready must minimally be able to:

- Register new users
- Log users in with their username and password
- Allow users to enable location tracking on their device
- Allow users to search for restaurants/activities based on location
- Add restaurant/activities to the itinerary

2.3 User Classes and Characteristics

Target users are those who are planning to do activities near a selected restaurant's vicinity and want to create an itinerary for it.
Admins are those who are responsible for overseeing EatAlready's operations and managing the user accounts.

2.4 Operating Environment

EatAlready works on a standard web environment, typically compatible with common web browsers such as Safari and Firefox. The backend of the application is built on the Django framework, which is compatible with any modern operating system like Windows, Linux, and macOS. It requires internet access to fetch information from external services like Google Maps API.

2.5 Design and Implementation Constraints

EatAlready is currently only available in English, which may come as a constraint for users who are not familiar with the language.
EatAlready relies on Google Maps API for location services, location tracking, fetching information on places, and retrieving nearby activities. Thus, the accuracy and the correctness of the information provided by the Google Maps API will impact the reliability of our EatAlready application. Inaccurate labelling of places on the Google Maps API would therefore lead to inaccurate information being presented on the EatAlready application.

2.6. User Documentation

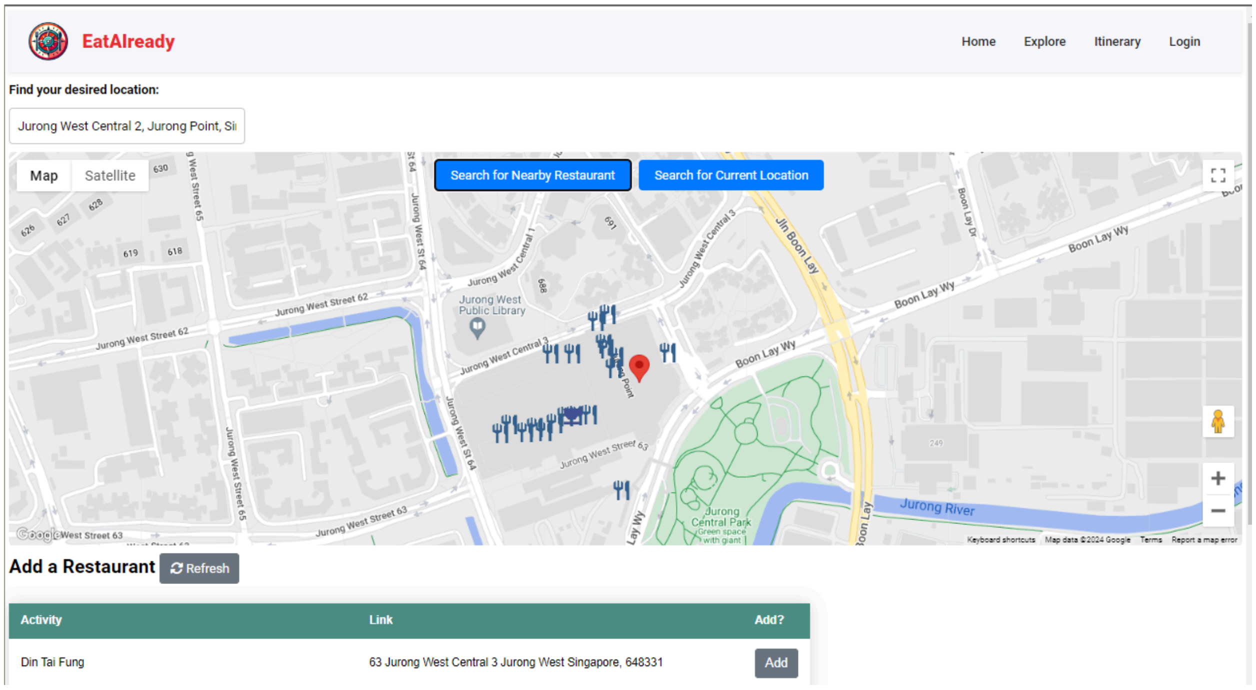
EatAlready's source code and design models are available on GitHub (https://github.com/softwarelab3/2006-SCSA-Delete-Save.git).

3. External Interface Requirements

3.1. User Interface

Homepage

Upon entering the web application, users will be presented with the homepage and prompted to enable location tracking. Users will be able to navigate through the various links on the navigation bar. Users will be able to use their current location, if they enable location tracking, or enter a desired location in the search bar above Google Maps. After the user inputs a location, the map will automatically move to the location with the destination marker placed at the center of the page. Users will then be able to "Search for Nearby Restaurant".



Find your desired location:

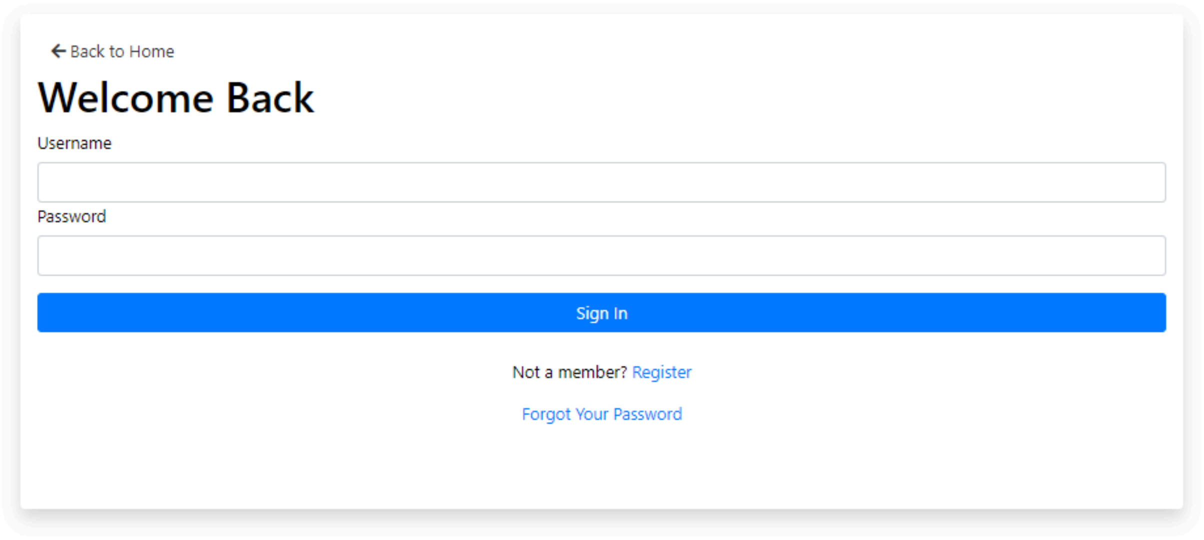
Jurong West Central 2, Jurong Point, Si

Add a Restaurant

Activity	Link	Add?
Din Tai Fung	63 Jurong West Central 3 Jurong West Singapore, 648331	Add

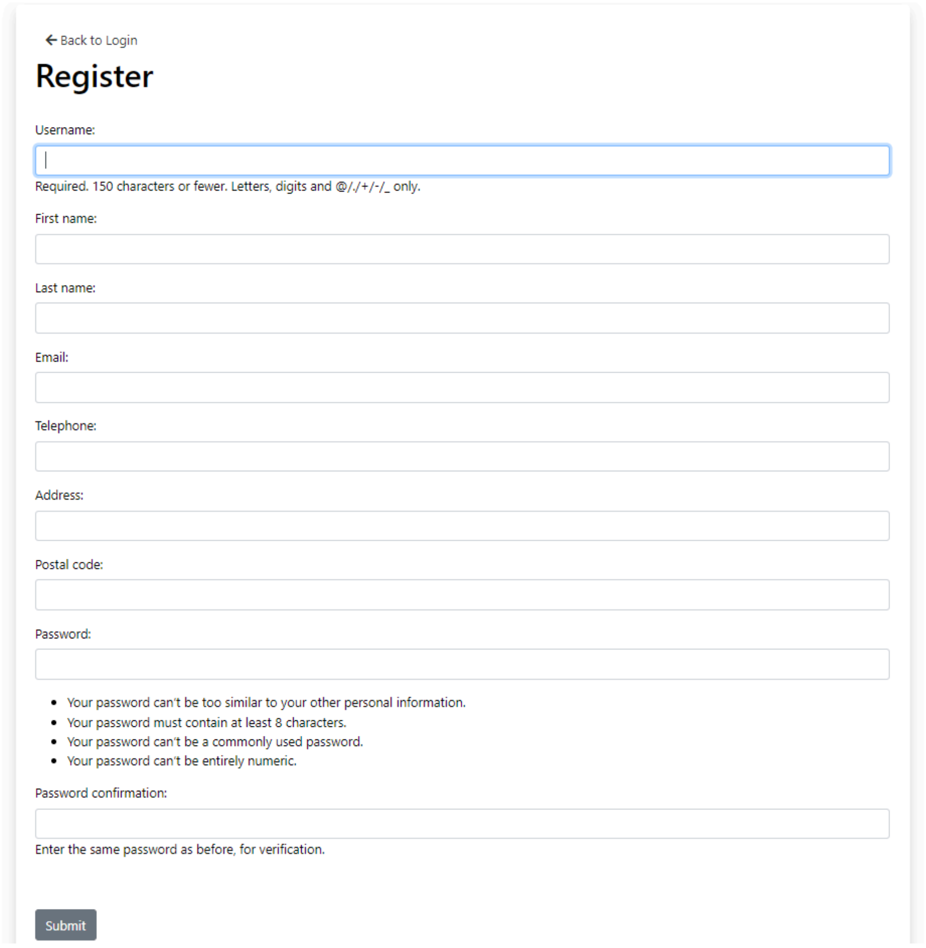
Login Page

Users can click “Login” on the homepage’s navigation bar and be brought to the login page. Users with an existing account can enter their username and password to log in. If the user is new and does not have an account, they can click “Register” to create an account. If the user forgets their password, they can click “Forget Your Password” to retrieve it.



Registration Page

New users can create an account on this page. They will be prompted to enter details such as Username, first name, last name, email, telephone, address, postal code, and password. Fields such as username, email, and telephone will be checked to ensure that it is not registered yet.



← Back to Login

Register

Username:

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name:

Last name:

Email:

Telephone:

Address:

Postal code:

Password:

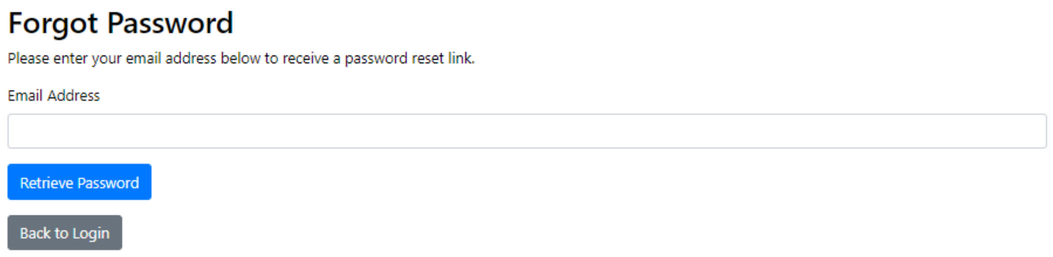
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

Account Recovery

Existing users can utilize this page to recover their account's password. They will be prompted to enter their email which allows our system to send a reset password email.



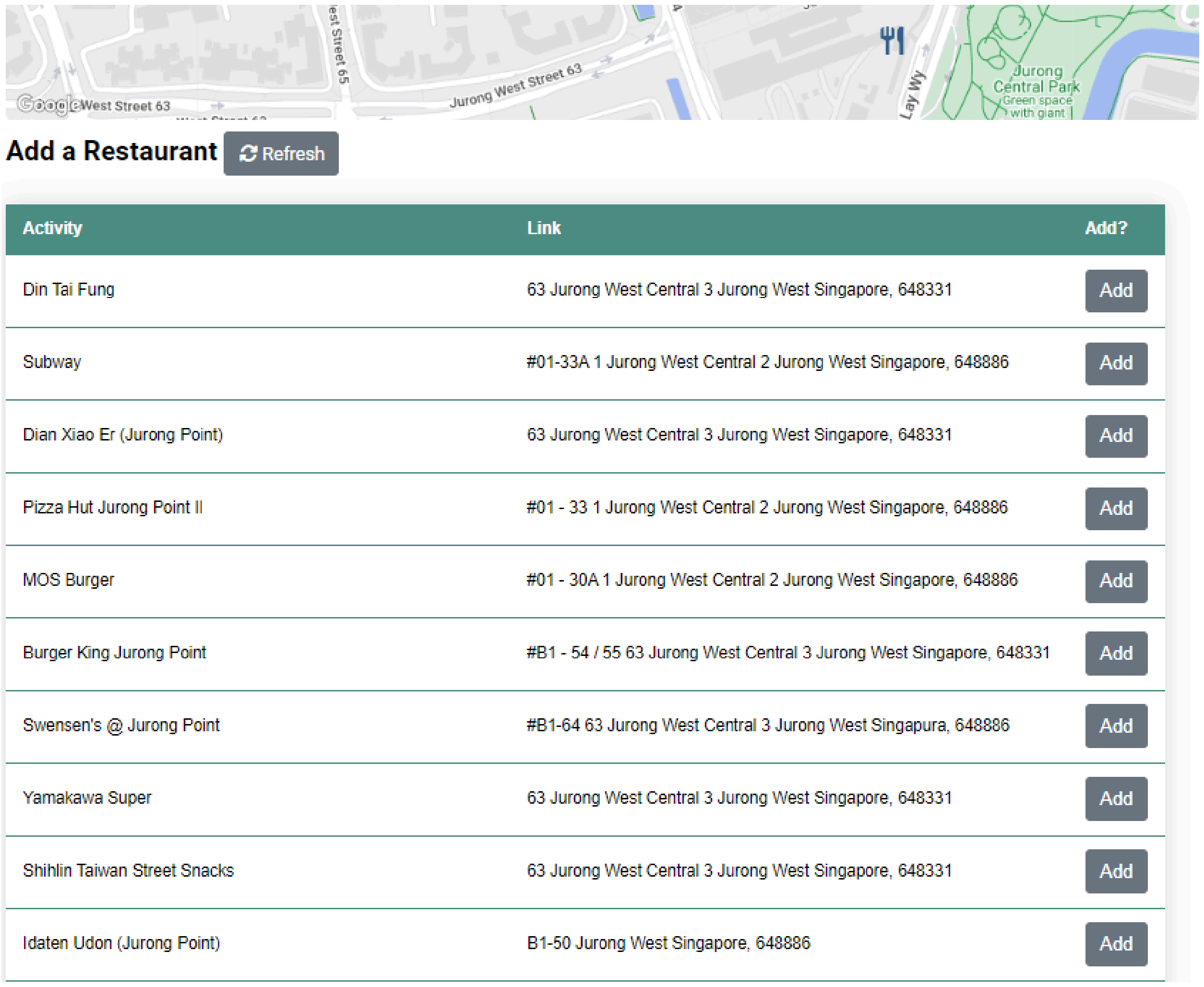
Forgot Password

Please enter your email address below to receive a password reset link.

Email Address

Restaurant List

Users can retrieve a list of nearby restaurants based on their current location or the location they entered in the search bar. Users can choose a restaurant to add to their itinerary list.

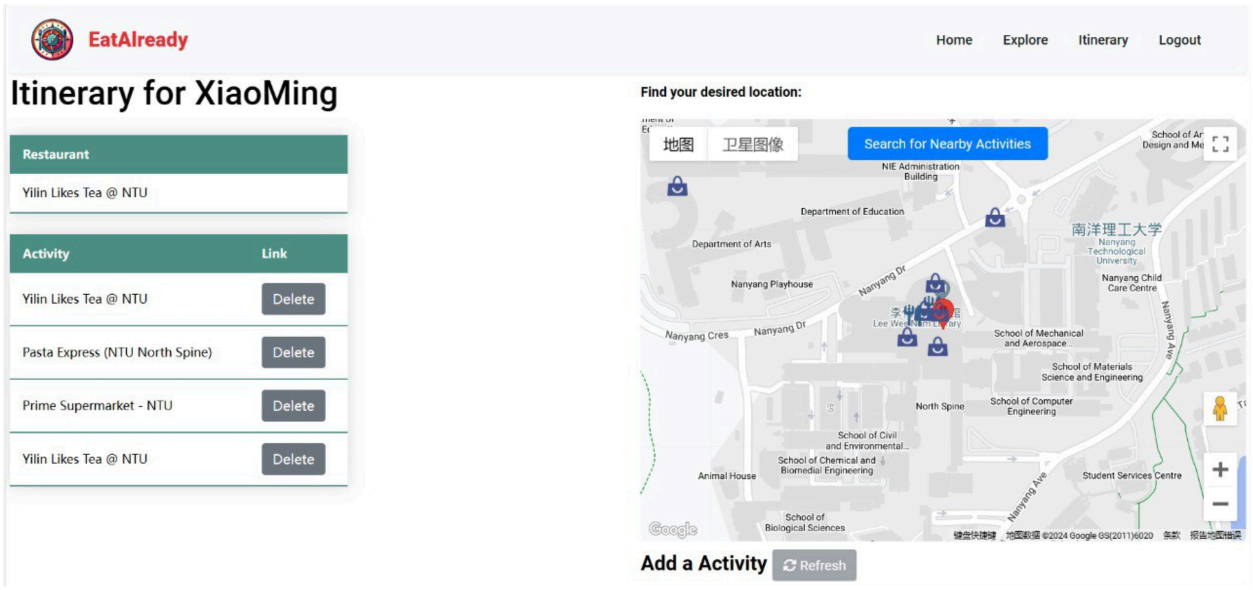


Add a Restaurant

Activity	Link	Add?
Din Tai Fung	63 Jurong West Central 3 Jurong West Singapore, 648331	Add
Subway	#01-33A 1 Jurong West Central 2 Jurong West Singapore, 648886	Add
Dian Xiao Er (Jurong Point)	63 Jurong West Central 3 Jurong West Singapore, 648331	Add
Pizza Hut Jurong Point II	#01 - 33 1 Jurong West Central 2 Jurong West Singapore, 648886	Add
MOS Burger	#01 - 30A 1 Jurong West Central 2 Jurong West Singapore, 648886	Add
Burger King Jurong Point	#B1 - 54 / 55 63 Jurong West Central 3 Jurong West Singapore, 648331	Add
Swensen's @ Jurong Point	#B1-64 63 Jurong West Central 3 Jurong West Singapura, 648886	Add
Yamakawa Super	63 Jurong West Central 3 Jurong West Singapore, 648331	Add
Shihlin Taiwan Street Snacks	63 Jurong West Central 3 Jurong West Singapore, 648331	Add
Idaten Udon (Jurong Point)	B1-50 Jurong West Singapore, 648886	Add

Itinerary Page

After the user adds a restaurant to the itinerary list, they can click "Itinerary" on the navigation bar to enter the itinerary page. The left of the page will consist of the selected restaurant and the user's itinerary, whereas the right of the page will consist of Google Maps and the list of nearby activities around the selected restaurant. Users can choose to add/ delete activities from the itinerary list.



Itinerary for XiaoMing

Restaurant
Yilin Likes Tea @ NTU

Activity	Link
Yilin Likes Tea @ NTU	Delete
Pasta Express (NTU North Spine)	Delete
Prime Supermarket - NTU	Delete
Yilin Likes Tea @ NTU	Delete

3.2 Hardware Interfaces

EatAlready is a Web Application primarily made for computer devices.

Since the application must run over the internet, all the hardware shall require to connect to the internet will be required for hardware interface for the system. For example, Modem, WAN – LAN, Ethernet Cross-Cable.

3.3 Software Interfaces

The system shall communicate with the Django administration system for account management.

The system shall communicate with GoogleAPI to utilise location information and to obtain nearby restaurants and activities.
The system shall communicate with the itinerary system to add and remove activities from a User’s itinerary.

3.4 Communication Interfaces

The system shall use the HTTP protocol for communication over the internet and the intranet communication will be through the TCP/IP protocol suite.

4. Functional Requirements

Functional Requirement FR-001: Account Management

Functional Requirement FR-001.1: User Registration

Description:

1. The system must allow users to register an account with the system.
2. Users should provide the necessary information and system to validate and store the data.

Acceptance Criteria:

1. Users must provide the registration page with fields for information such as username, name, email, phone number, address, postal code, password, and password confirmation.
2. The system should validate that all usernames, email addresses, and phone numbers are unique and in a valid format.
3. The system to check passwords adheres to a specified strength policy (E.g. Minimum length, Combination of letters and numbers, etc) and matches with the password confirmation field.
4. Upon successful registration, the system will redirect users to the login page.
5. System to prompt feedback to users on the success or failure of the registration process.

Priority: High

Preconditions: None

Postconditions:

1. A new user account is created in the system.
2. The user returns to the login page.

Functional Flow

1. The user accesses the registration page from the home page.
2. The user fills in the required information, including username, email address, and password.
3. The user submits the registration form.
4. The system validates the information from the user.
5. If validation is successful, the system creates a new user account and redirects to the login page.

Inputs:

- User-provided information (username, email address, password)

Processes:

- Validate the uniqueness and format of the email address.
- Validate the strength and format of the password.
- Create a new user account in the system.
- Redirect to the login page.

Outputs

- The user is redirected to the login page
- Feedback message to indicate the success or failure of the registration process.

Error Handling

- Displays appropriate error messages for duplicate fields or weak passwords.
- Displays appropriate error messages for empty mandatory fields.

Performance Requirements:

- The user registration process should not take more than 1 minute.

Usability Requirements:

- The user interface should be straightforward and user-friendly.

Dependencies:

- Username, email, and phone number were checked with the database to ensure no duplicates.

Functional Requirement FR-001.2: User Access (Login)

Description:

- The system must provide a secure and user-friendly login process for registered users to access the system.

Acceptance Criteria:

1. Users must be able to access the login page from the home page.
2. The system should validate the entered credentials (username, email address, password) against the stored user data.

3. If the entered credentials are valid, the system should grant access to the user's account.
4. After a successful login, the system should redirect the user to the home page.
5. In case of invalid credentials, the system should display the appropriate error messages

Priority: High

Preconditions:

6. The user must be a registered member of the system.
7. The user must have an active and confirmed account.

Postconditions:

1. Upon successful login, the user gains access to the system.

Functional Flow

1. The user navigates to the login page from the home page.
2. The user enters their username/email and password.
3. The system validates the entered credentials.
4. If validation is successful, the user is redirected to the home page.
5. If validation fails, the appropriate messages are displayed.

Inputs:

- User-provided information (username/email address, password)

Processes:

- Validate entered username and password against stored user data.
- Redirect the user to the Home page upon successful login.

Outputs

- Access granted message upon successful login
- Error messages for invalid credentials.

Error Handling

- Displays appropriate error messages for invalid credentials.

Performance Requirements:

- The login process should not take more than 10 seconds.

Usability Requirements:

- The user interface should be straightforward and user-friendly.

Dependencies:

- Secure storage and retrieval of user credentials.

<u>Functional Requirement FR-001.3: Account Recovery</u>

Description:

- The system must provide a secure and user-friendly account recovery process for registered users to recover their accounts.

Acceptance Criteria:

1. Users must be able to access the forget password page from the login page.
2. The system should validate the entered credentials (email) against the stored data.
3. If the entered credentials are valid, the system should send a password reset email to the user’s email.
4. The system should display the appropriate error messages in case of invalid credentials.
5. The user must be able to access the password reset page from the link in the password reset email with mandatory fields (new password, confirm password).
6. The system to check passwords adheres to a specified strength policy (E.g. Minimum length, Combination of letters and numbers, etc) and
7. Upon successful password reset, the system should update the new password over the old stored user password.
8. System to prompt feedback to users on the success or failure of the password reset process.

Priority: High

Preconditions:

1. The user must be a registered member of the system.

Postconditions:

1. The user receives a password reset email.

Functional Flow

1. The user navigates to the login page from the home page.
2. The user clicks on the “Forget password” hyperlink.
3. The system redirects the user to the “Forget Password” page.
4. The user enters their username/email.

5. The system validates the username/email.
6. If validation is successful, the password reset email is sent to the email.
7. If validation fails, the appropriate messages are displayed.
8. The user navigates to the reset password page from the link in the email.
9. The user enters their new password and confirmation password.
10. The system to check passwords adheres to a specified strength policy (E.g. Minimum length, Combination of letters and numbers, etc) and matches with password confirmation field.
11. If validation is successful, users will be redirected to the login page.
12. If validation fails, the appropriate messages are displayed.

Inputs:

- User-provided information (username/email address)

Processes:

- Validate entered username and password against stored user data.
- A password reset email is to be sent to the user's email.

Outputs

- Email sent message upon password reset.
- Feedback message on password reset email sent.

Error Handling

- Displays appropriate error messages for invalid password strengths.

Performance Requirements:

- The reset process should not take more than 1 minute.

Usability Requirements:

- The user interface should be straightforward and user-friendly.

Dependencies:

- Secure storage and retrieval of user credentials.
- Email-sending to send out password reset emails.

Functional Requirement FR-002: Location Tracking

Functional Requirement FR-002.1: Allow Tracking of account location

Description:

- The system should provide a secure channel for the users to share their location information.

Acceptance Criteria:

1. Users should be able to decide if they want to allow tracking of their device's location in the system.
2. The system should be able to differentiate between accounts that allow or disable location-tracking information.
3. Upon successfully updating “Allow tracking”, the system should update the user’s information in the database.
4. Users should see additional functionality that considers the device’s current location if enabled.
5. If disabled, users should only see functionality based on the location they inputted.

Priority: Medium

Preconditions:

1. The user must be a registered member of the system.
2. The user must have an active and confirmed account.

Postconditions:

1. Enabled location tracking for the user.

Functional Flow:

1. The user navigates to the settings/profile Page.
2. Click on the “Enable Location Tracking” button.

Inputs:

- NA

Processes:

- Update the user’s preference in the database.

Outputs:

- Feedback message on enabling location tracking.

Error Handling:

- Display an appropriate message if "Allow Tracking" is not enabled and no initial location is given by the user.

Performance Requirements:

- The location-enabling process should take at most 1 minute.

Usability Requirements:

- The user interface should be straightforward and user-friendly

Dependencies:

- Secure storage and retrieval of user's data/preferences.

Functional Requirement FR-002.2: Display Location on Map

Description:

- The device's current location or the user's search input to be displayed on the map interface.

Acceptance Criteria:

1. The user's device location is accurately displayed on the Map interface.
2. Error messages are displayed appropriately when there are issues with location tracking or map rendering.
3. Users are prompted to grant permission to access location data if "Allow Tracking" is disabled.

Priority:

- High

Preconditions:

- The user must have enabled the "Allow tracking" setting or inputted an initial location.

Postconditions:

- The device's current location is displayed on the map.

Functional Flow:

- The user logs in and navigates to the home page.
- The user device's current location will appear on the map interface.

Inputs:

- The user must be a registered member of the system.
- The user must have an active and confirmed account.

Processes:

- Display the device's location on the map interface.

Outputs:

- Marker of device location on the map interface

Error Handling:

- Display an appropriate message if "Allow Tracking" is not enabled and the user gives no initial location.

Performance Requirements:

- The location display process should take at most 1 minute to load.

Usability Requirements:

- The map interface should be straightforward and user-friendly.

Dependencies:

- The "Allow Tracking" setting is to be enabled.

Functional Requirement FR-002.3: Search for a nearby restaurant

Description:

- The system must allow users to search for nearby restaurants based on the keyed-in location. Users should be able to view relevant information about each restaurant and make informed decisions.

Acceptance Criteria:

1. Users can search for restaurants based on location.
2. Search results provide information about each restaurant, including contact details, opening hours, and average ratings.
3. Users can view restaurants on a map interface
4. Users can choose to add the restaurant to the itinerary.

Priority:

- High

Preconditions:

- NA

Postconditions:

- Restaurant location to be displayed on the map interface.
- Selected Restaurants will be added to the User's Itinerary.

Functional Flow:

1. The user inputs the restaurant name in the search bar.
2. A list of restaurants with similar names and properties will appear.
3. The user clicks on the "add" button on one of the restaurants returned from the search results to add it to the itinerary.

Inputs:

- User-provided information (location).

Processes:

- Validate user-provided information with the API.

Outputs:

- Restaurant location to be displayed on the map interface.
- Button to add Restaurant to the itinerary.

Error Handling:

- Display an appropriate message if no result is returned from the API.

Performance Requirements:

- The location display process should take at most 1 minute to load.

Usability Requirements:

- The search process should be straightforward.

Dependencies:

- User-provided information
- Google API (Map, Places)

Functional Requirement FR-001.9: View restaurant List

Description:

- Users shall be able to view a list of restaurants using the Google Places API

Acceptance Criteria:

1. The system shall send a request to the Google Places API to retrieve a list of restaurants based on a location input by the user.
2. The system shall display the list of restaurants along with relevant information (name, address)

Priority:

- High

Preconditions:

- NA

Postconditions:

- A list of restaurants is displayed based on the filters chosen.

Functional Flow:

1. Users enter a location.
2. The system requests the Google Place API to retrieve a list of restaurants near the input location.
3. The system receives and processes the response from the Google Places API.
4. The system displays the list of restaurants.

Inputs:

- User-provided information

Processes:

- Querying the Google Places API to retrieve a list of restaurants based on the filters selected, parsing and displaying the results

Outputs:

- Lists of restaurants with relevant information (name, address)

Error Handling:

- Display an appropriate message if there is an error in retrieving the restaurant list from the Google Places API, prompting users to try again.

Performance Requirements:

- The restaurant list should be retrieved and displayed within 30 Seconds

Usability Requirements:

- The restaurant list should be presented in a user-friendly format, allowing users to easily understand.

Dependencies:

- Integration with the Google Places API for retrieving restaurant data.

Functional Requirement FR-003: Itinerary

Functional Requirement FR-003.1: Adding Restaurant to Itinerary

Description:

- Users shall be able to add a selected restaurant to their itinerary for future references

Acceptance Criteria:

1. Users can select a restaurant from the search results or enter the restaurant details manually.
2. The system shall provide an option to add the selected restaurant to an itinerary.
3. Each added restaurant shall be associated with an itinerary and displayed in list format.
4. Users can view, edit, or remove restaurants from their itinerary.

Priority:

- Medium

Preconditions:

- The user must be logged in and have access to the itinerary feature.

Postconditions:

- After adding a restaurant to the itinerary, the user will see it listed in their itinerary.

Functional Flow:

1. Users search for a restaurant.
2. Users select a restaurant.
3. Users click on the "Add" button.
4. The system adds the restaurant to the itinerary.

Inputs:

- User-provided information (restaurant name).
- User-action Input (Click)

Processes:

- Adding restaurants to the itinerary

Outputs:

- Confirmation message of a successful addition to the itinerary.

Error Handling:

- Display an appropriate message if adding the same restaurant to the itinerary.

Performance Requirements:

- Adding a restaurant should be completed within 5 seconds.

Usability Requirements:

- The option to add a restaurant to the itinerary should be clearly visible and intuitive.

Dependencies:

- User authentication system to ensure that only currently logged-in users can add restaurants to their itinerary.

Functional Requirement FR-003.2: Search Nearby Activities

Description:

- Users shall be able to receive recommendations for nearby activities using the Google Places API.

Acceptance Criteria:

1. Users can click on a “Search Nearby Activity” Button.
2. The system shall send a request to the Google Places API to retrieve a list of nearby activities.
3. The system shall display a list of recommended activities to the user.

Priority:

- Medium

Preconditions:

- The user must have selected a restaurant.

Postconditions:

- The user is recommended a list of nearby activities to do.

Functional Flow:

- Users select a restaurant.
- The user clicks on the “Recommend Nearby Activity” button.
- The system sends a request to the Google Places API to retrieve activities based on the selected restaurant’s location.
- The system receives and processes the response from the Google Places API.

- The system displays a list of recommended activities to the user.

Inputs:

- User-provided information
- User-action Input

Processes:

- Querying the Google Places API to retrieve nearby activities based on the selected restaurant's location.

Outputs:

- Displaying a list of nearby activities.

Error Handling:

- Display an appropriate message if there is an error in retrieving the activity list from the Google Places API, prompting users to try again.

Performance Requirements:

- The process of retrieving and displaying nearby activities should be completed within 10 seconds.

Usability Requirements:

- The button should be prominently displayed and easily accessible to users.

Dependencies:

5. Integration with the Google Places API for retrieving nearby activities.

6. Other Nonfunctional Requirements

Usability	The user interface should be organized and simple for users to navigate. Any buttons on the page should be easily recognizable. The user's credentials must be stored securely.
Reliability	The app must not be down for over 31 seconds in a year (99.9999% availability). After the app reboots, the full system functionality must be restored within 5 minutes.
Performance	The app must be able to load and respond to the user's prompt within 30 seconds. Locations tracked and provided should be accurate with a maximum error of 20m. The application should be able to redirect the User to external websites or applications within 15 seconds.
Supportability	The application must be supported on mobile phones. The system must support 500 Users simultaneously on the application at the same time.
Security	The password must consist of 6 to 20 characters. The password must contain at least one uppercase letter, at least one lowercase letter, and at least one number.

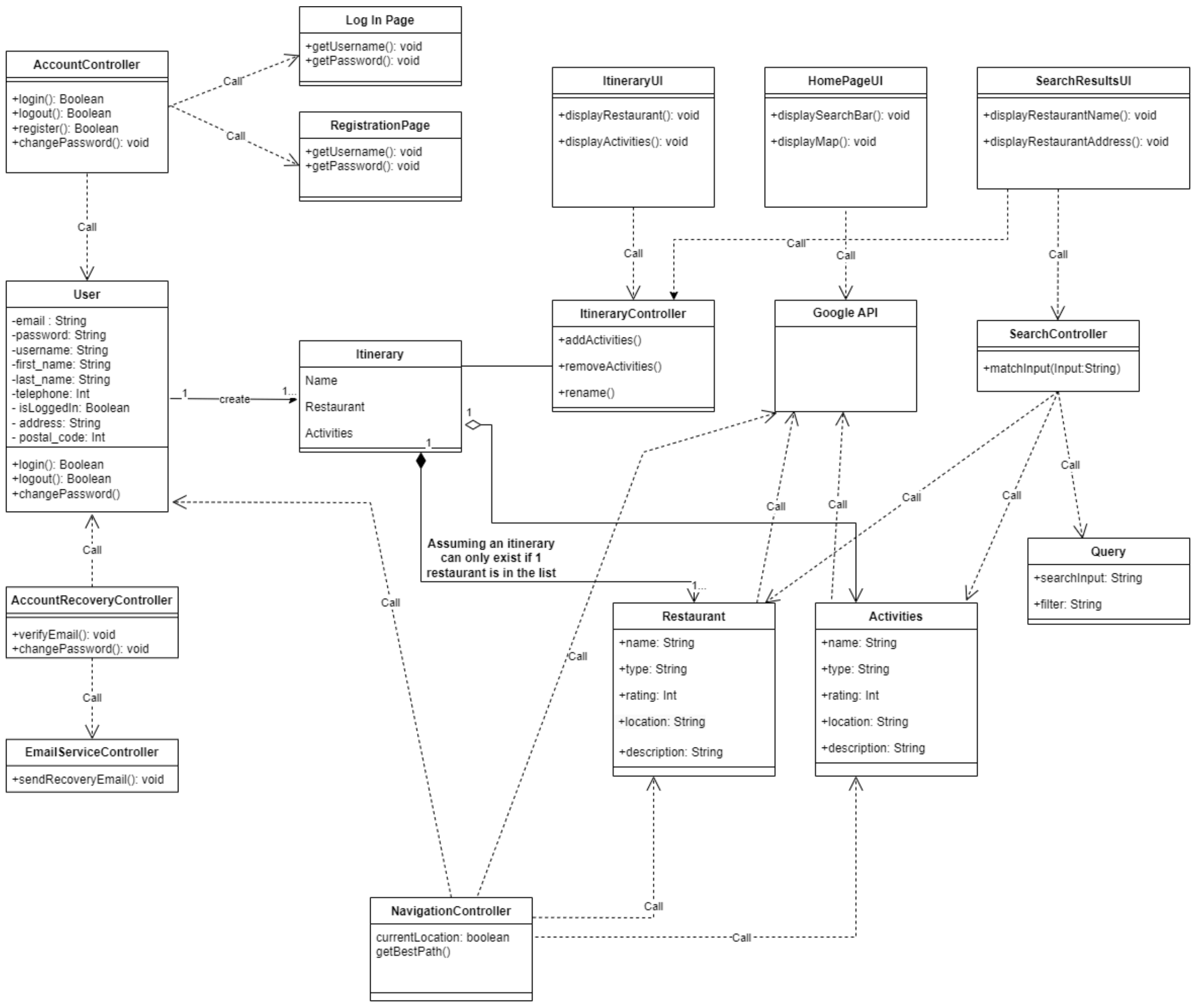
Appendix

Appendix A: Data Dictionary

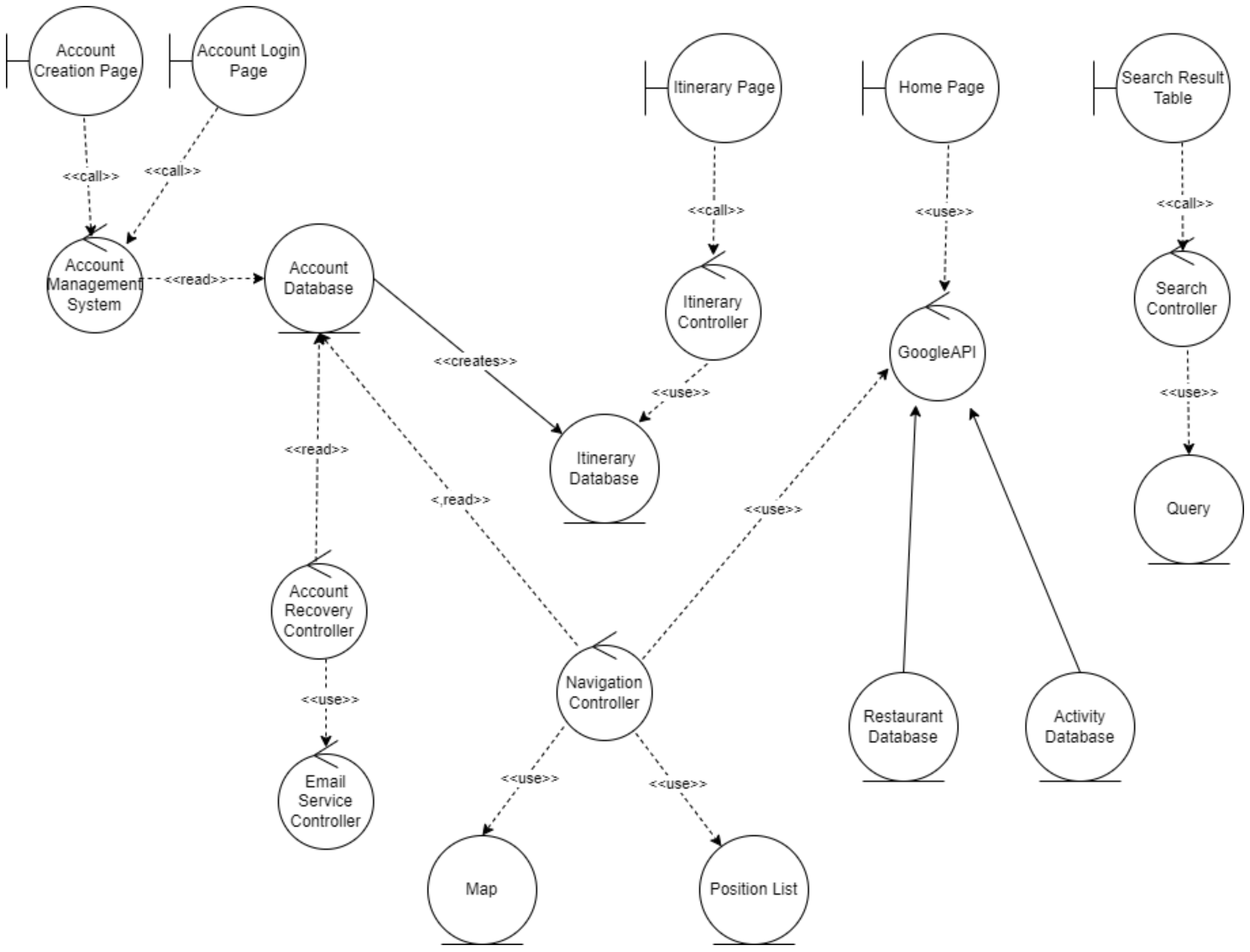
Term	**Definition**
Planner/ Itinerary	A plan or schedule of events, activities, or tasks
Recommendation	Suggestions or advice provided by the system to the user based on the selected event or based on their preferences
Event	An activity or task that the user wants to plan
Itinerary	A list of places(restaurant/activities) the user plans to go
Review	User-generated feedback towards a restaurant or activity to provide insights for other users and influence recommendation
Category/Tag	A classification or label assigned to events to group them on a common theme (restaurant/activity)
User Profile	Information about an individual user (preferences, saved items, and historical data)
Collaboration	The ability for multiple users to contribute or share itineraries
Notification	Alerts or messages sent to users to inform them about upcoming events, changes, or other updates
Preferences	User-specified settings and choices that will influence recommendations and planning process (based on preferred categories, time constraints, or locations)
Location	A specified area associated with events or activities (location-based recommendation)
Feedback	Input from users regarding their satisfaction with the recommendation
Search	The functionality allows users to find events or activities based on keywords, categories etc
History	A record of past events, activities, and user interactions with the system, used to enhance future recommendations and planning
Authentication	Verification process of identifying users to ensure secure access to their profiles and data

Appendix B: Conceptual Diagrams

UML Class Diagram

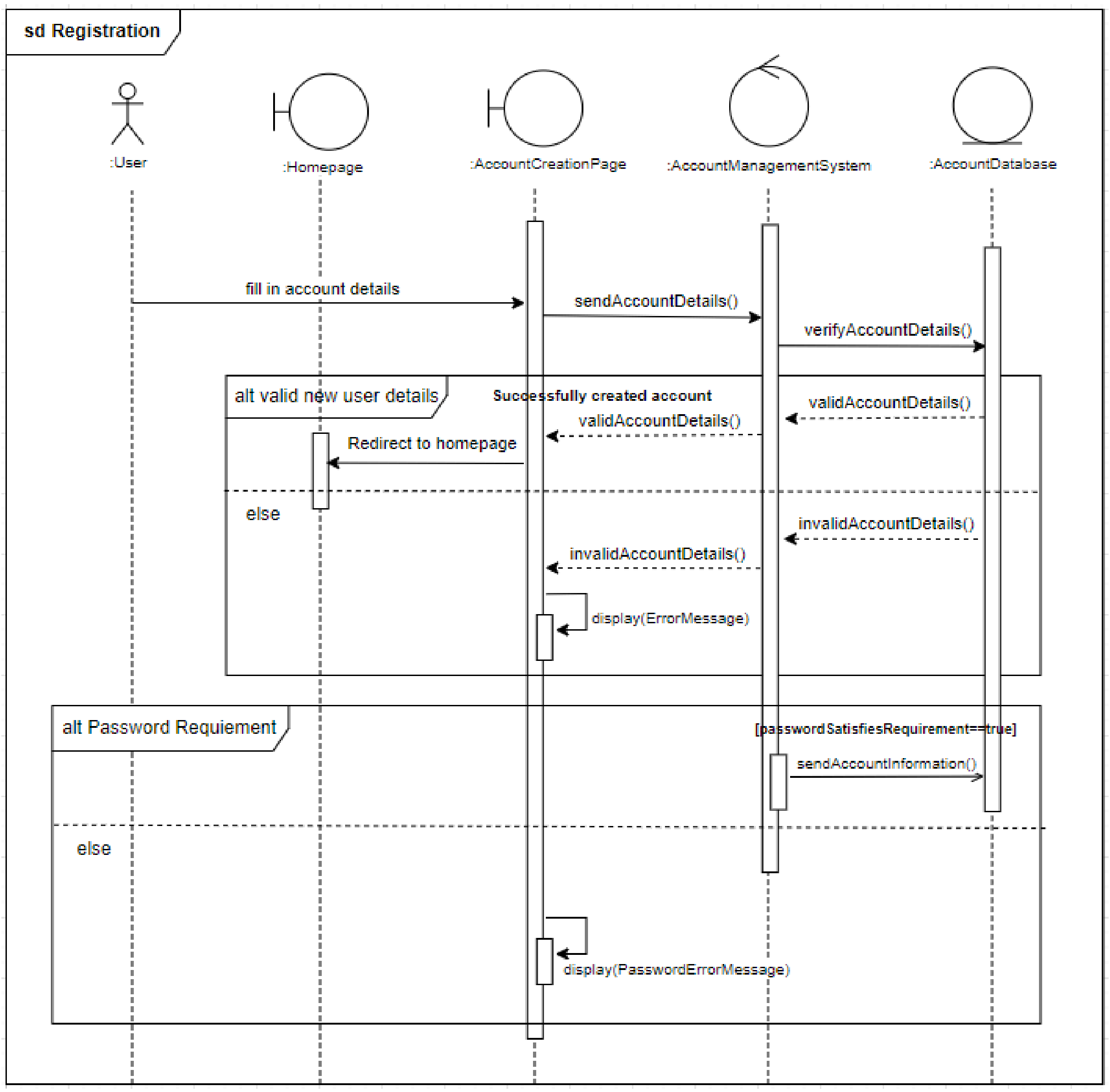


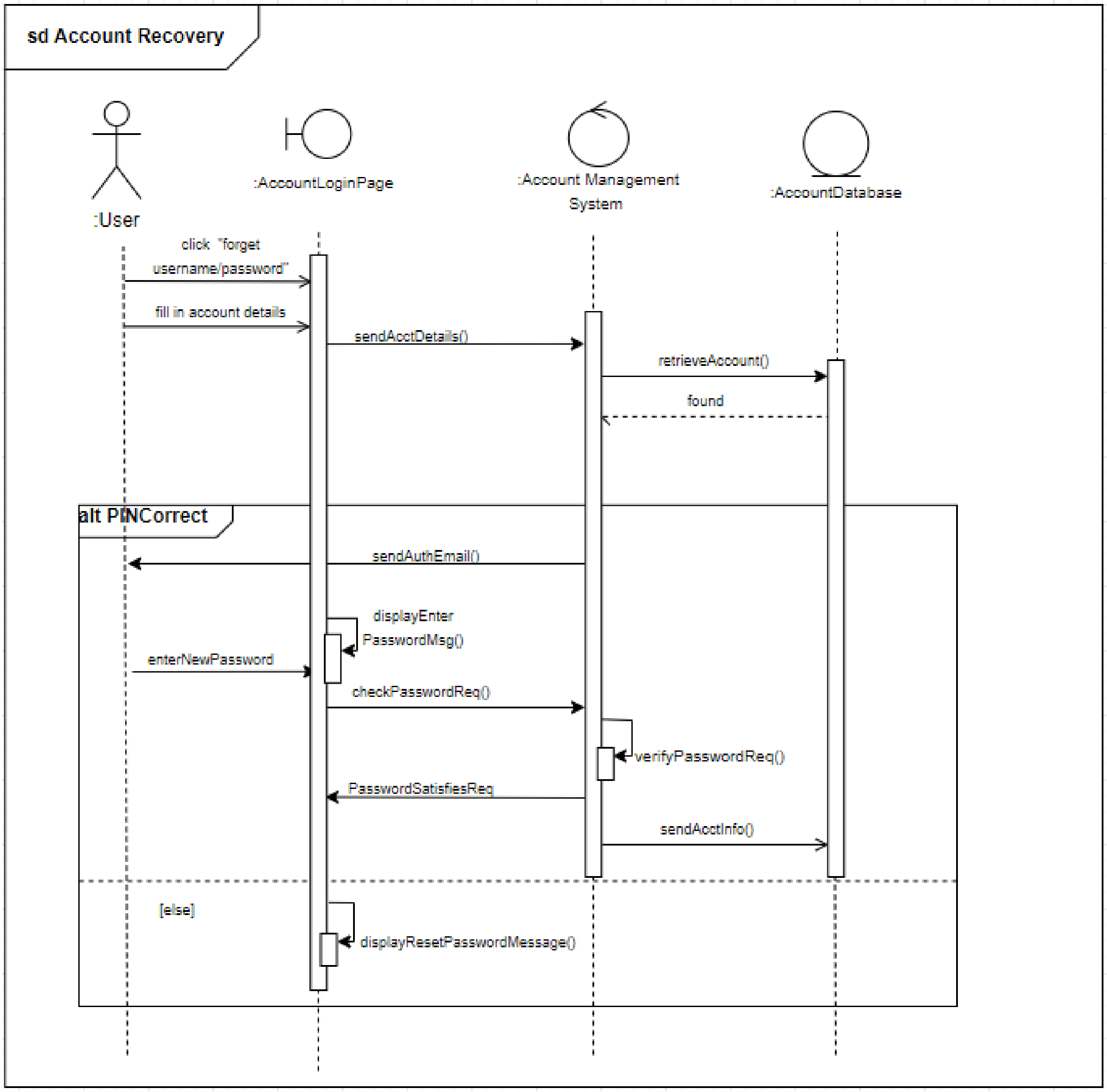
Stereotype Relationship Diagram

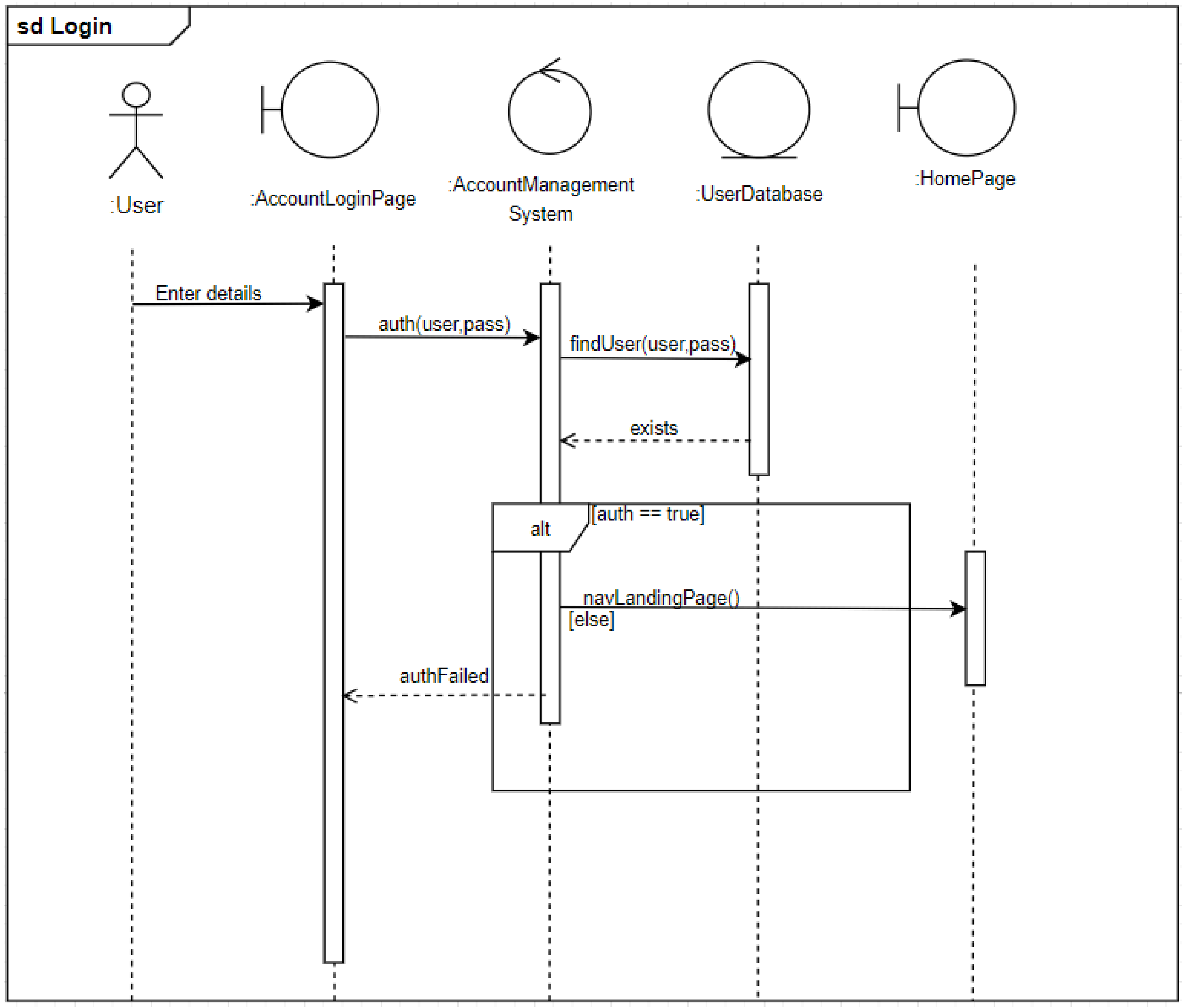


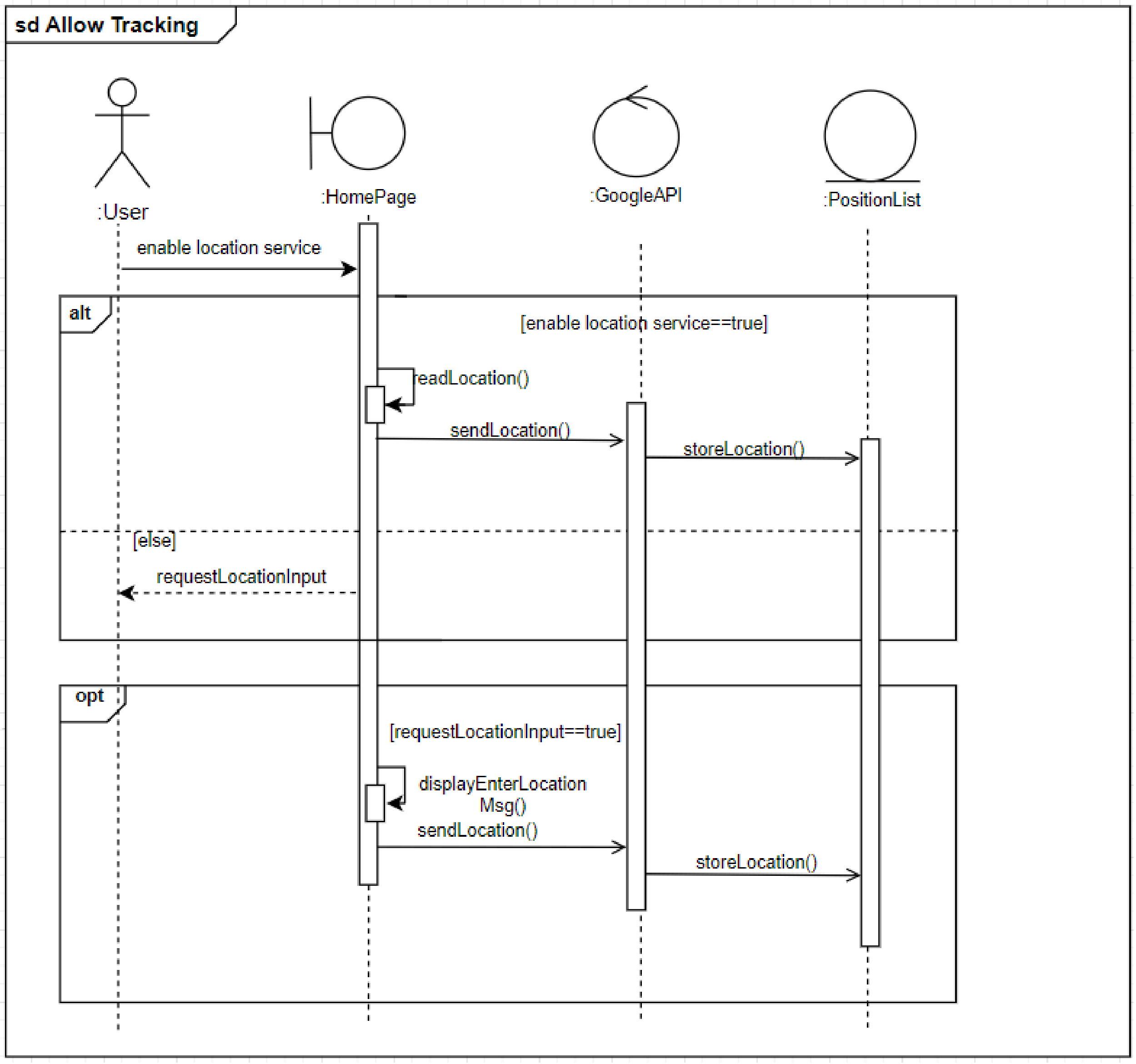
Appendix C: Dynamic Models

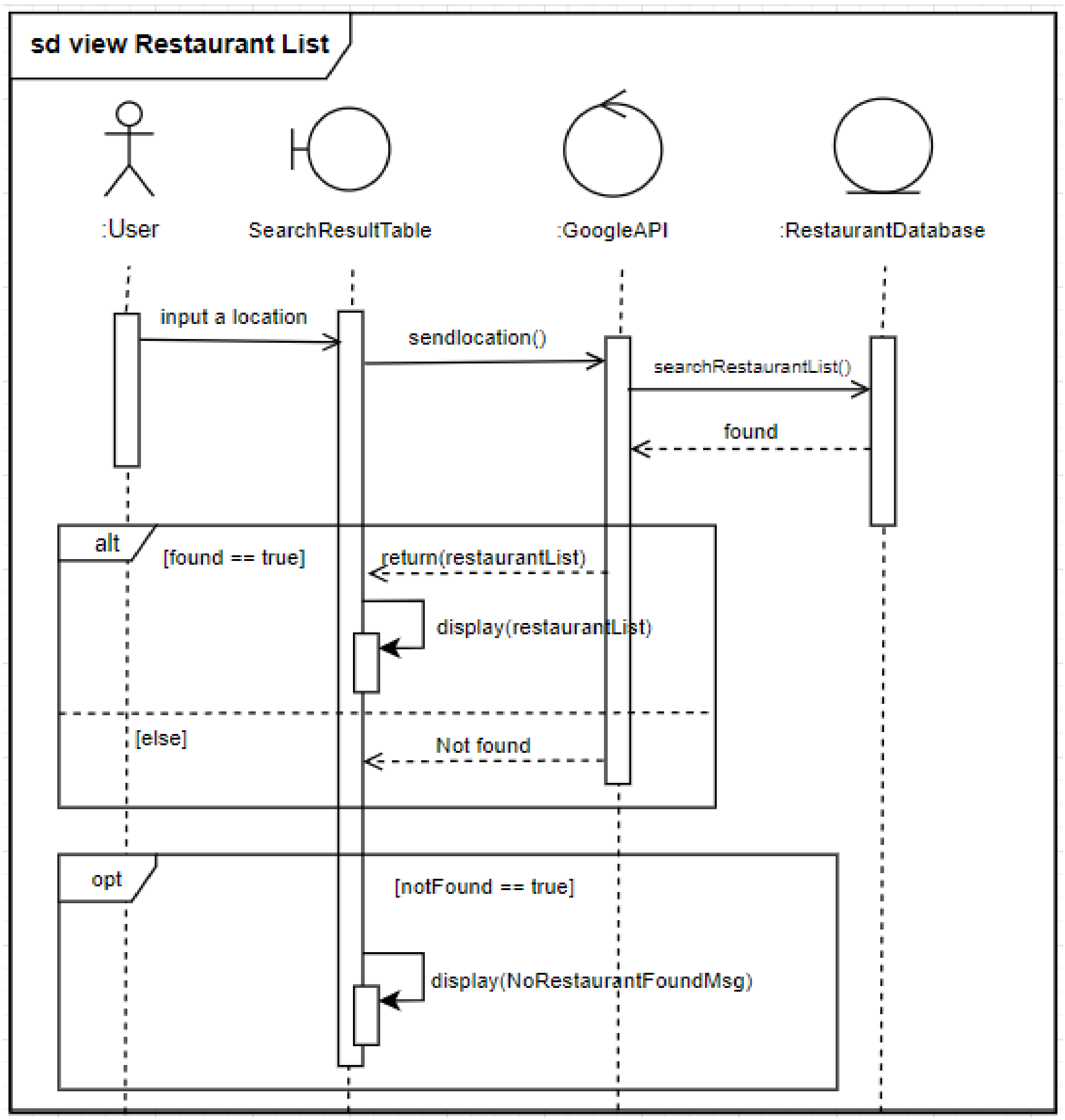
<u>Sequence Diagram</u>

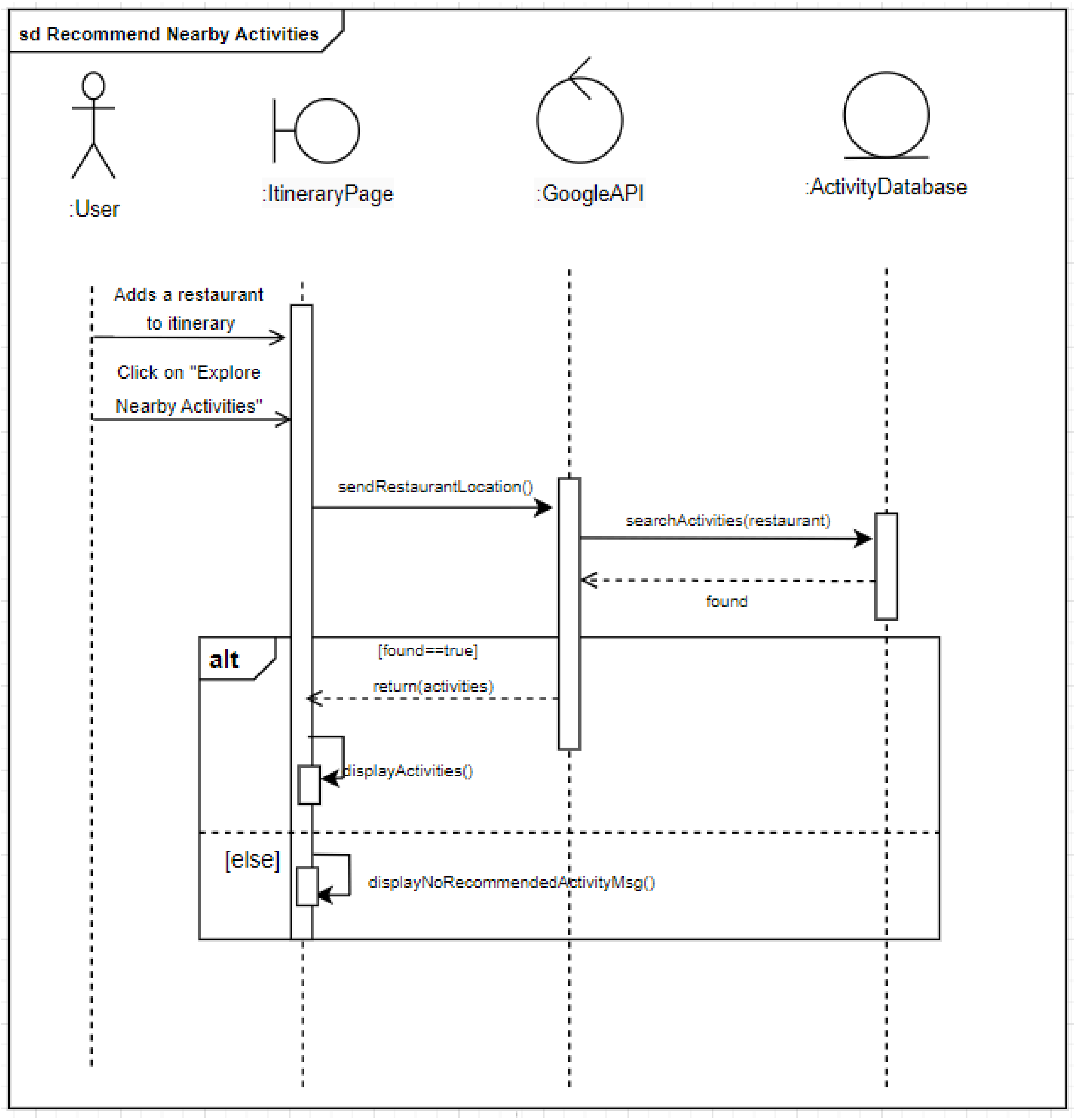



sd Account Recovery
:User
:AccountLoginPage
:Account Management System
:AccountDatabase
click "forget username/password"
fill in account details
sendAcctDetails()
retrieveAccount()
found
alt PINCorrect
sendAuthEmail()
displayEnter PasswordMsg()
enterNewPassword
checkPasswordReq()
verifyPasswordReq()
PasswordSatisfiesReq
sendAcctInfo()
[else]
displayResetPasswordMessage()

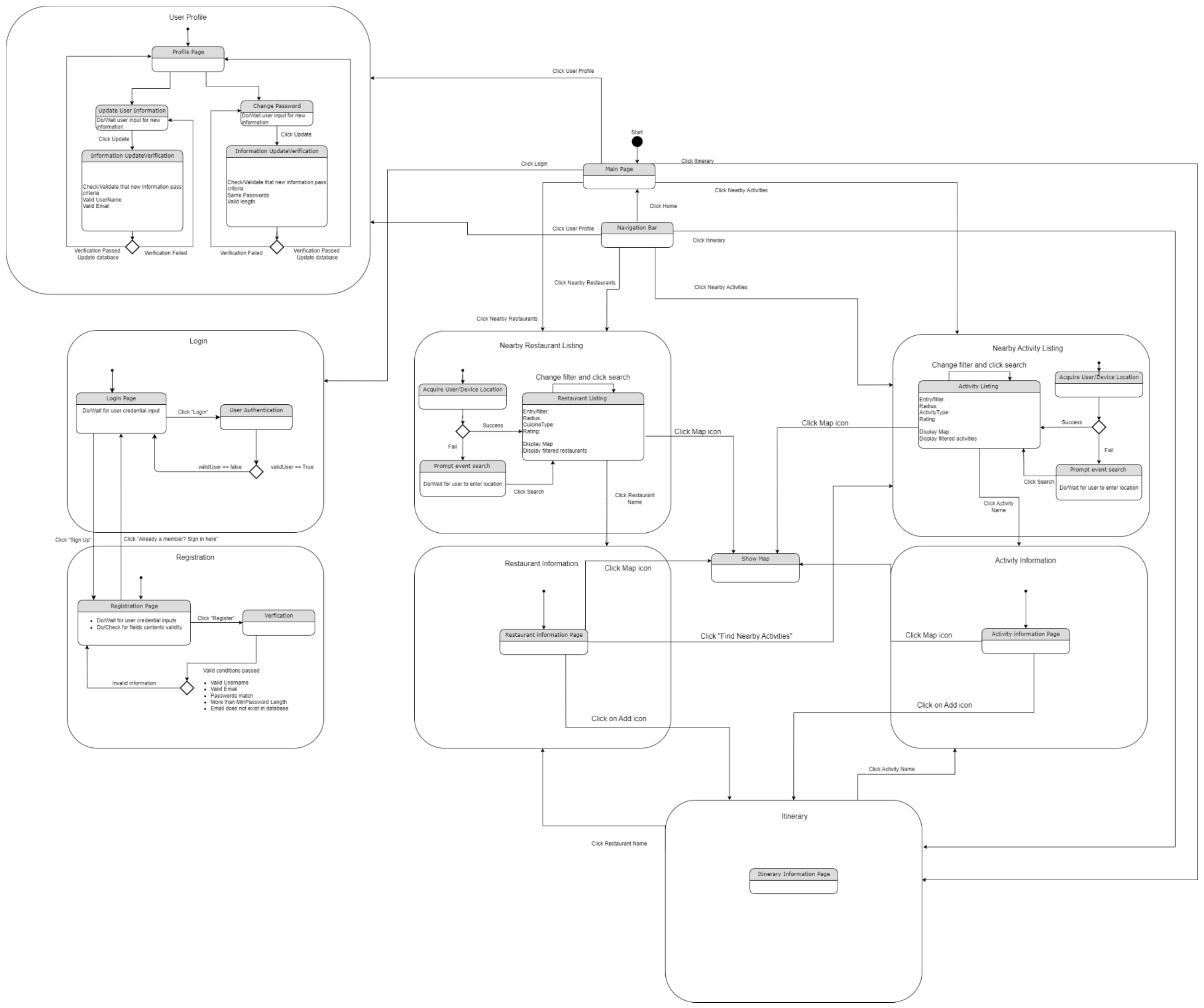

sd Login
:User
:AccountLoginPage
:AccountManagement System
:UserDatabase
:HomePage
Enter details
auth(user,pass)
findUser(user,pass)
exists
alt
[auth == true]
navLandingPage()
[else]
authFailed


sd Allow Tracking
:User
:HomePage
:GoogleAPI
:PositionList
enable location service
alt
[enable location service==true]
readLocation()
sendLocation()
storeLocation()
[else]
requestLocationInput
opt
[requestLocationInput==true]
displayEnterLocation Msg()
sendLocation()
storeLocation()

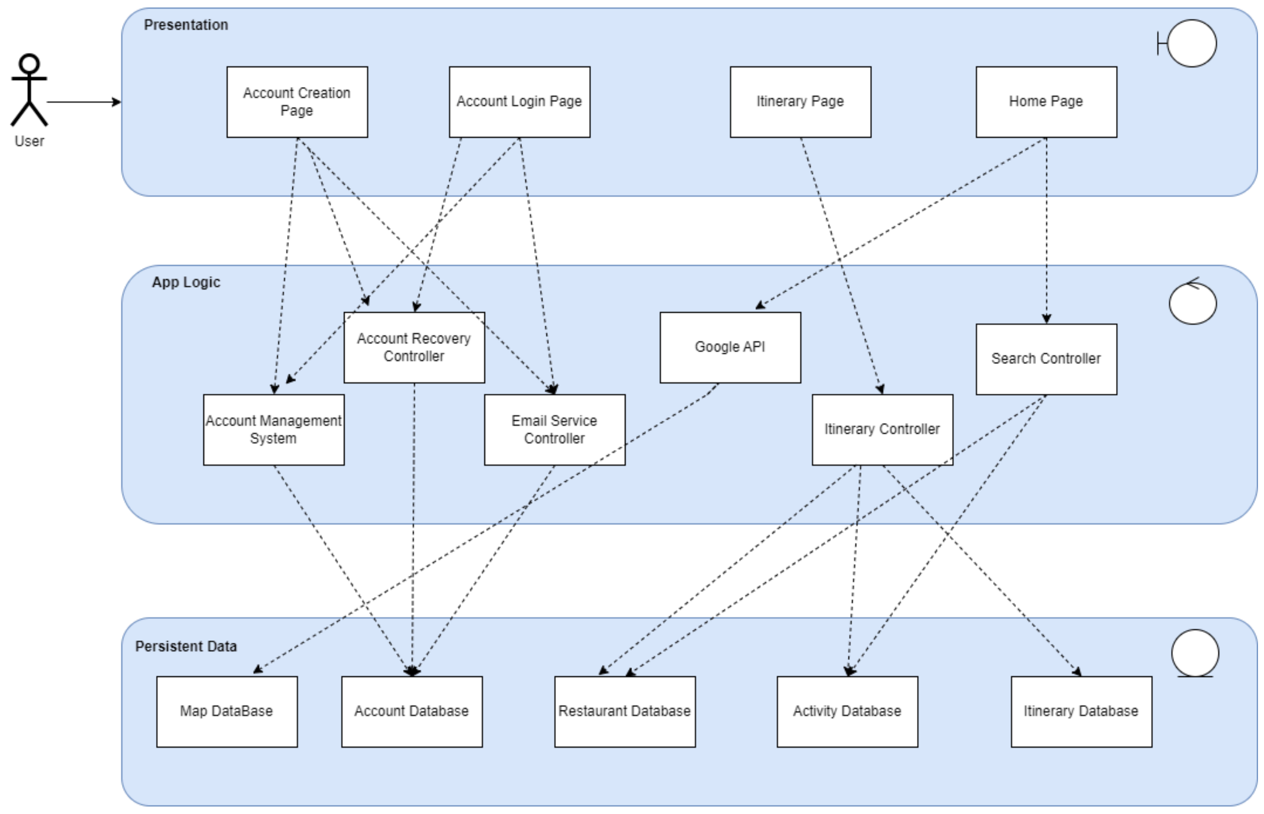

sd view Restaurant List
:User
SearchResultTable
:GoogleAPI
:RestaurantDatabase
input a location
sendlocation()
searchRestaurantList()
found
alt
[found == true]
return(restaurantList)
display(restaurantList)
[else]
Not found
opt
[notFound == true]
display(NoRestaurantFoundMsg)


sd Recommend Nearby Activities
:User
:ItineraryPage
:GoogleAPI
:ActivityDatabase
Adds a restaurant to itinerary
Click on "Explore Nearby Activities"
sendRestaurantLocation()
searchActivities(restaurant)
found
alt
[found==true]
return(activities)
displayActivities()
[else]
displayNoRecommendedActivityMsg()

Dialog Map


User Profile
Profile Page
Update User Information
Do/Wait user input for new information
Click Update
Information UpdateVerification
Check/Validate that new information pass criteria
Valid UserName
Valid Email
Verification Passed
Update database
Verification Failed
Change Password
Do/Wait user input for new information
Click Update
Information UpdateVerification
Check/Validate that new information pass criteria
Same Passwords
Valid length
Verification Failed
Verification Passed
Update database
Click User Profile
Start
Click Login
Main Page
Click Itinerary
Click Nearby Activities
Click Home
Click User Profile
Navigation Bar
Click Itinerary
Click Nearby Restaurants
Click Nearby Activities
Click Nearby Restaurants
Login
Login Page
Do/Wait for user credential input
Click "Login"
User Authentication
validUser == false
validUser == True
Click "Sign Up"
Click "Already a member? Sign in here"
Registration
Registration Page
Do/Wait for user credential inputs
Do/Check for fields contents validity
Click "Register"
Verfication
Invalid information
Valid conditions passed:
Valid Username
Valid Email
Passwords match
More than MinPassword Length
Email does not exist in database
Nearby Restaurant Listing
Acquire User/Device Location
Success
Fail
Prompt event search
Do/Wait for user to enter location
Click Search
Change filter and click search
Restaurant Listing
Entry/filter:
Radius:
CuisineType:
Rating:
Display Map
Display filtered restaurants
Click Map icon
Click Restaurant Name
Nearby Activity Listing
Change filter and click search
Activity Listing
Entry/filter:
Radius:
ActivityType:
Rating:
Display Map
Display filtered activities
Acquire User/Device Location
Success
Fail
Prompt event search
Do/Wait for user to enter location
Click Search
Click Activity Name
Click Map icon
Show Map
Restaurant Information
Click Map icon
Restaurant information Page
Click "Find Nearby Activities"
Click on Add icon
Activity Information
Click Map icon
Activity information Page
Click on Add icon
Click Activity Name
Itinerary
Click Restaurant Name
Itinerary Information Page

Appendix D: System Architecture



Appendix E: Black Box Testing

Registration Functionality

- Password Requirement Testing

Input Description	**Test Input**	**Expected Result**	**Actual Result**
> 8 characters, not entirely numerical	AdvPass8921	Accepted	Accepted
< 8 characters	Pw1234	Rejected	Rejected
Entirely numerical	12345678	Rejected	Rejected
Commonly used password (Noun)	flower	Rejected	Rejected
Commonly used password (Noun)	money	Rejected	Rejected
Commonly used password (Proper noun)	alexandra	Rejected	Rejected
Password too similar to personal information	Username: test2 Password: test2user	Rejected	Rejected

Username Requirement Testing

Input Description	**Test Input**	**Expected Result**	**Actual Result**
< 150 characters, letters and digits only, used one of accepted special characters “@/./+/-/_”	TestUser.123	Accepted	Accepted
< 150 characters, letters only	TestUser	Accepted	Accepted
< 150 characters, digits only	123	Approval	Accepted
< 150 characters, accepted special characters only “@/./+/-/_”	.	Approval	Accepted

Unaccepted special character “!”	TestUser!123	Rejected	Rejected
>150 characters	TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser100	Accepted	Accepted (Textbox automatically cuts off at 150 characters, hence it is within the acceptable range of username)
150 characters	TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser10TestUser10 TestUser10	Accepted	Accepted

Account Recovery Functionality

- Forget Password

Input Description	**Test Input**	**Expected Result**	**Actual Result**
Enter existing email	testuser2@gmail.com	Password reset link has been sent to email.	Password reset link has been sent to email.
Enter non-existing email	nottesteruser2@gmail.com	Email does not exist.	Email does not exist.
Click on the password reset link. Enter password following requirements.	kNflOaWP1239!	Password will be reset. User redirected to login page.	Password will be reset. User redirected to login page.

Search Functionality

- Looking for nearby restaurant

Input Description	**Test Input**	**Expected Result**	**Actual Result**
Activate current location	Click “Allow” to share user’s location	Map will move to current user’s location	Map will move to current user”s location
Disable location sharing, user keys in starting location	Location: Jurong point	Map will move to Jurong Point	Map will move to Jurong Point
Disable current location, dont key in any other location	No input to location	No changes will happen, map stays at its original location	No changes will happen, map stays at its original location

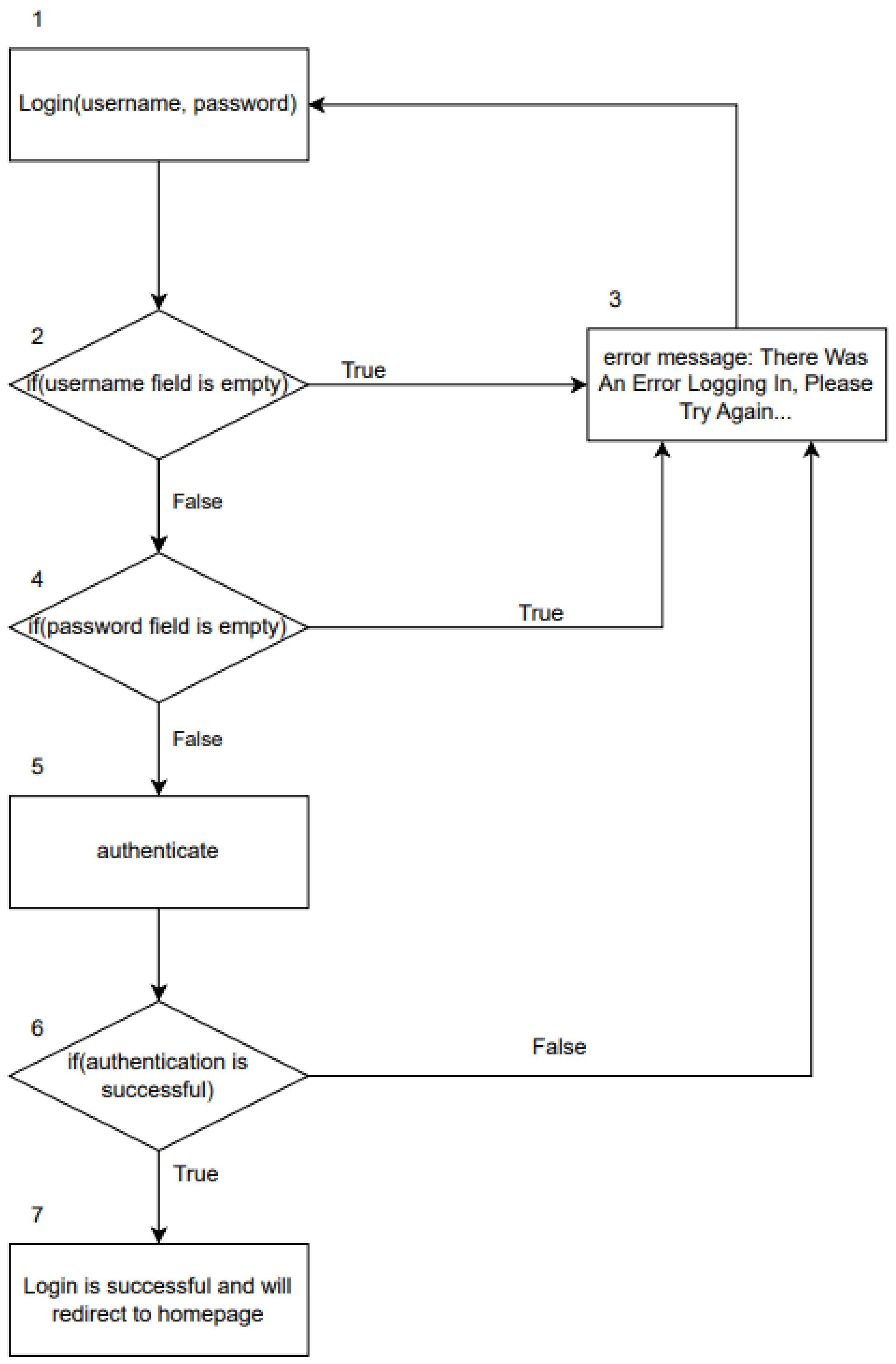
Itinerary Functionality

- Adding an activity to an itinerary (Control flow testing)

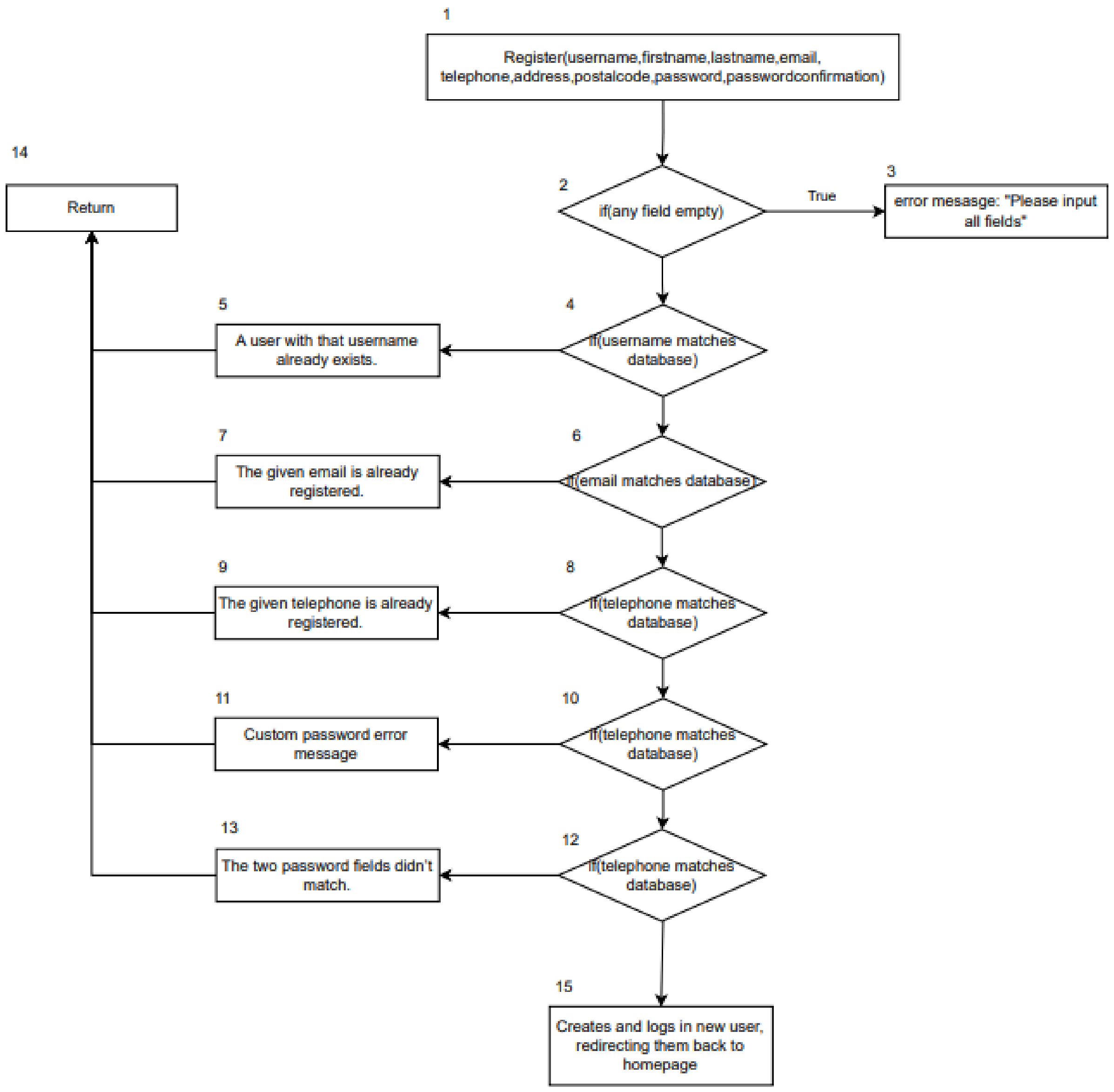
Input Description	**Test Input**	**Expected Result**	**Actual Result**
No selected restaurant has been added to the itinerary	Click on “search nearby activities”	No nearby activities will be displayed	No nearby activities will be displayed

Appendix F: White Box Testing

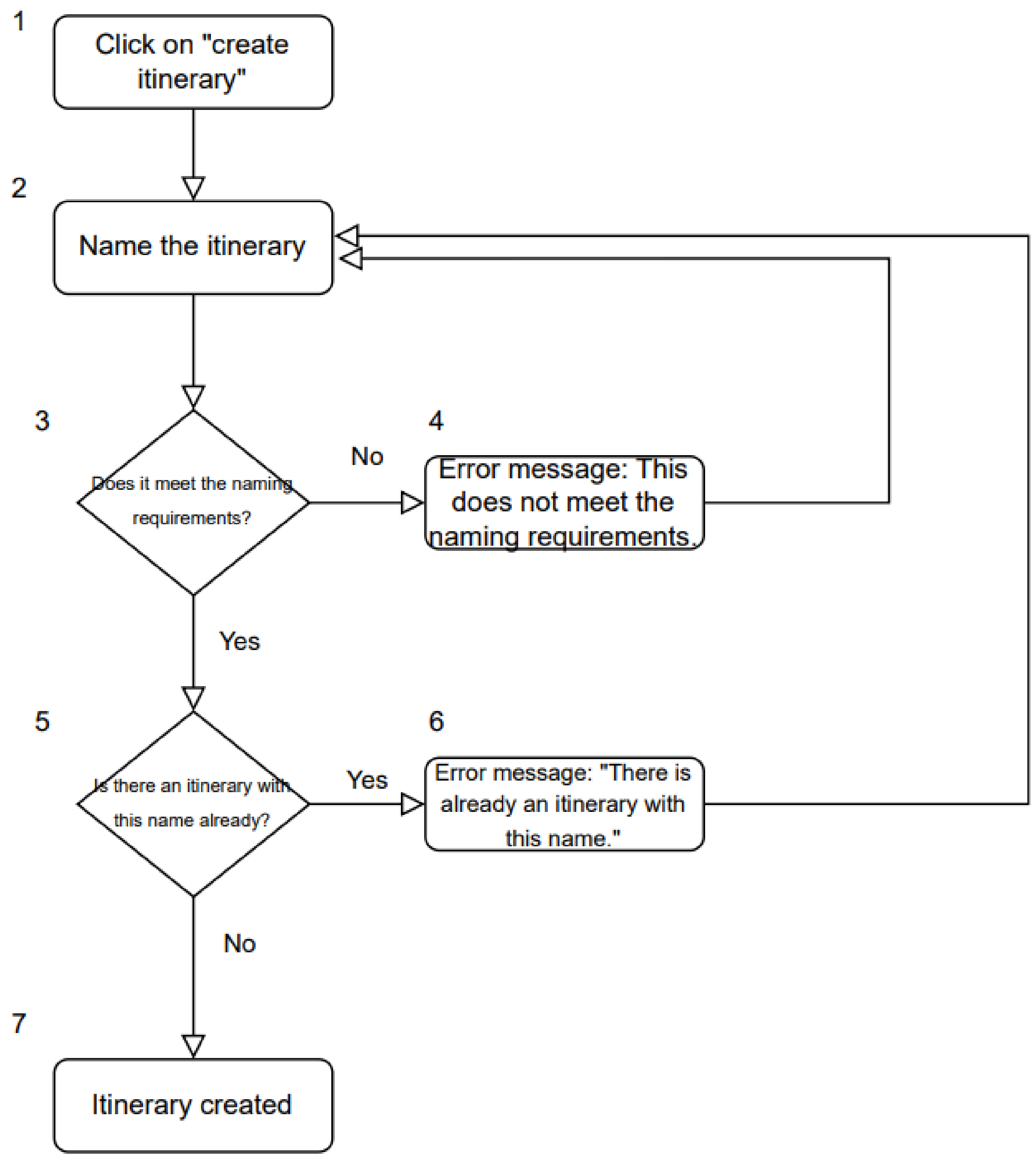
Login


1
Login(username, password)
2
if(username field is empty)
True
3
error message: There Was An Error Logging In, Please Try Again...
False
4
if(password field is empty)
True
False
5
authenticate
6
if(authentication is successful)
False
True
7
Login is successful and will redirect to homepage

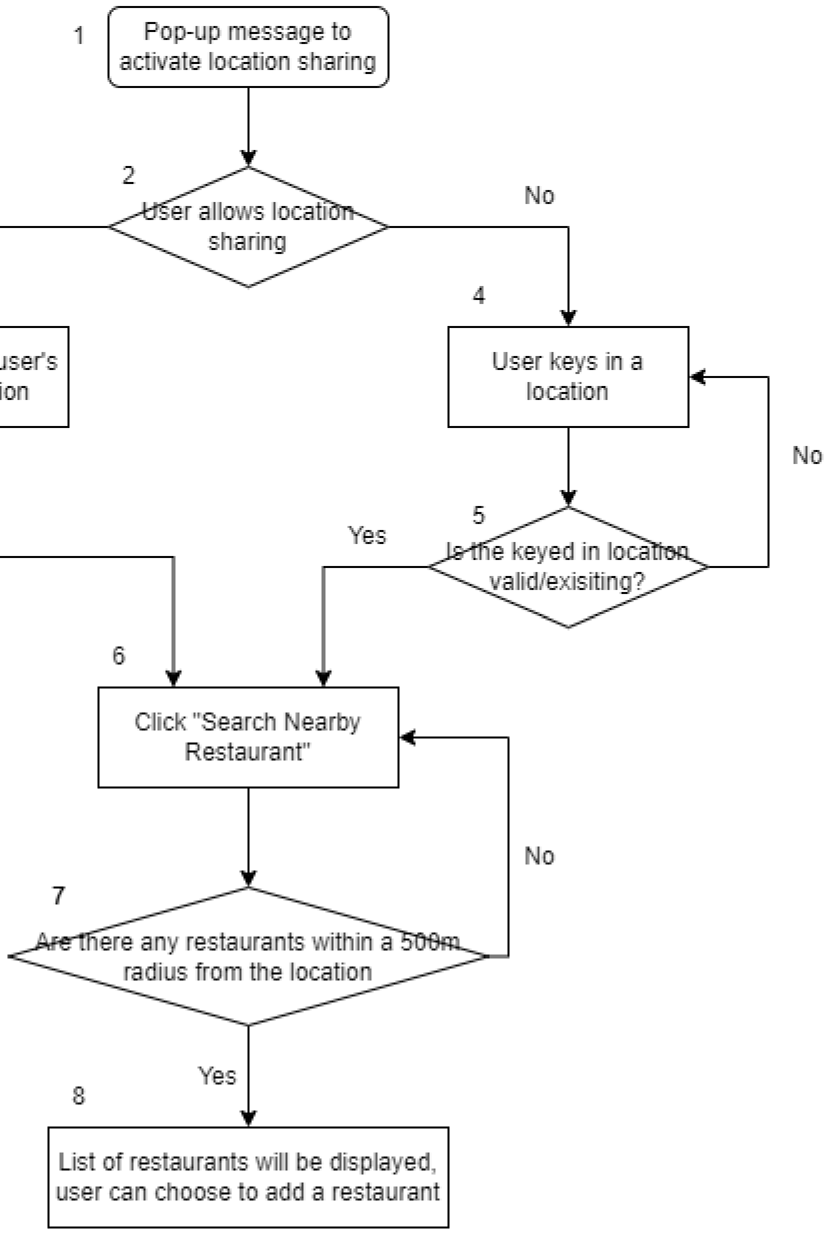
Registration


1
Register(username,firstname,lastname,email, telephone,address,postalcode,password,passwordconfirmation)
2
if(any field empty)
True
3
error mesasge: "Please input all fields"
14
Return
4
if(username matches database)
5
A user with that username already exists.
6
if(email matches database)
7
The given email is already registered.
8
if(telephone matches database)
9
The given telephone is already registered.
10
if(telephone matches database)
11
Custom password error message
12
if(telephone matches database)
13
The two password fields didn't match.
15
Creates and logs in new user, redirecting them back to homepage

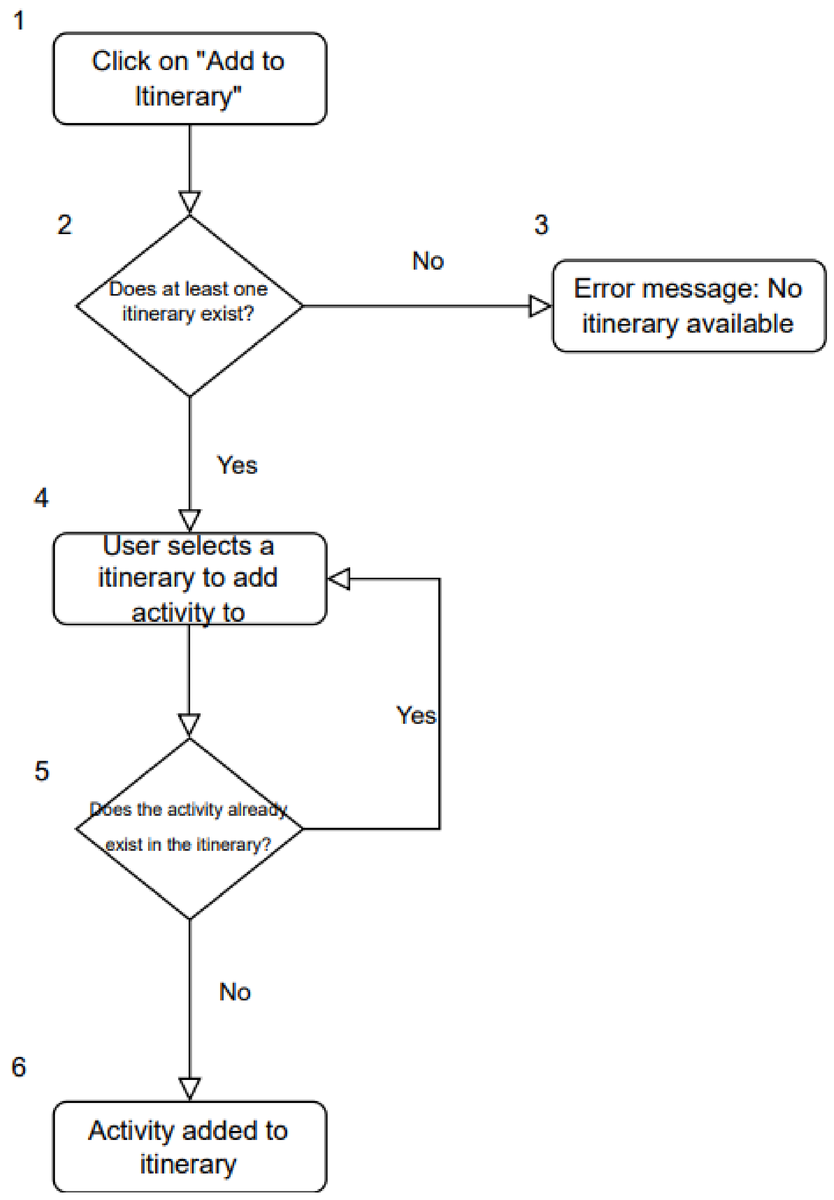
Create Itinerary


1
Click on "create itinerary"
2
Name the itinerary
3
Does it meet the naming requirements?
No
4
Error message: This does not meet the naming requirements.
Yes
5
Is there an itinerary with this name already?
Yes
6
Error message: "There is already an itinerary with this name."
No
7
Itinerary created

Search and Add Restaurant


1
Pop-up message to activate location sharing
2
User allows location sharing
Yes
No
3
Map moves to user's current location
4
User keys in a location
No
5
Yes
Is the keyed in location valid/exisiting?
6
Click "Search Nearby Restaurant"
No
7
Are there any restaurants within a 500m radius from the location
Yes
8
List of restaurants will be displayed, user can choose to add a restaurant

Add Activity


1
Click on "Add to Itinerary"
2
Does at least one itinerary exist?
No
3
Error message: No itinerary available
Yes
4
User selects a itinerary to add activity to
Yes
5
Does the activity already exist in the itinerary?
No
6
Activity added to itinerary