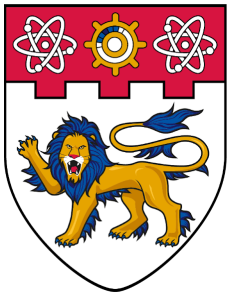


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

School of Computer Science and Engineering (SCSE)

SC20006 Software Engineering

Software Requirement Specification

for Homey

Lab Group/ Group No.: A29/Group 1
Team Name: NGNRS

Team Members:

Crystal Cheong Yu Qing	U2121134A
Chan Ming Han	U2120477J
Ang Jun Koon	U2122362K
Cheong Chi Hian	U2122966L
Ng Mu Rong	U2121323F

Table of Contents

Revision History

Date	Name	Reason For Changes	Version
27/01/2023	Crystal Cheong	- Initialised documentation	0.1.0
27/01/2023	Crystal Cheong	- Added first draft of Use Case diagram and Use Case descriptions - Drafted Functional and Non-Functional requirements	0.2.0
29/01/2023	Crystal Cheong, Ming Han, Chi Hian, Mu Rong	- Updated Use Case diagram and Use Case descriptions - Added Data Dictionary - Included Initial UI Mockups	0.3.0
14/03/2023	Crystal Cheong	- Finalised Use Case, Dialog Map and Class diagrams - Updated documentation on Purpose, Scope, Initial UI Mockups and Functional requirements - Added documentation on Architecture Design, Non-Functional and Interface requirements	0.4.0
17/03/2023	Crystal Cheong	- Added documentation on Application Skeleton	0.4.1
25/03/2023	Mu Rong	- Included new and updated existing sequence diagrams with alternate flows	0.4.2
29/03/2023	Chi Hian, Jun Koon	- Completed Testing (Black-box & White-box) analysis	0.4.3
03/04/2023	Crystal Cheong	- Final formatting for submissions - Updated exported images	0.4.3.1
03/04/2023	Crystal Cheong, Ming Han, Chi Hian, Mu Rong, Jun Koon	- Document submission sign-off	1.0.0

1. Product Description

1.1. Purpose

Homey seeks to provide a comprehensive and convenient online platform for homebuyers, sellers, and renters to search for properties, view property listings and connect with agents. The web application is designed to simplify the property search process and make it easy for users to find their dream home or investment property.

With access to a vast database of property listings, Homey offers users a one-stop-shop for all their real estate needs, from finding the right property to connecting with experienced agents who can guide them through the buying, selling or renting process. The goal of Homey is to provide a user-friendly and efficient real estate search experience, helping users to make informed decisions and find the perfect property to suit their needs.

1.2. Scope

The Homey web application offers a range of features and services to facilitate the buying, selling, and renting of properties, including:

(1) Property search: Users can search for properties based on their location, budget, and other criteria, and view detailed information and images of each property.

(2) Property listings: Homey provides access to a vast database of property listings, complete with detailed information, high-quality images, and pricing information.

(3) Agent connection: Users can connect with experienced agents who can provide guidance and assistance throughout the buying, selling, or renting process.

(4) User account management: Users can create and manage their accounts, save property searches, and receive updates on new listings and properties that meet their criteria.

1.3. Users and Stakeholders

Users

- Homebuyers
- Home sellers
- Renters
- Real estate investors
- Real estate agents

Stakeholders

- Homey development team
- Homey management team
- Real estate agents and agencies partnering with Homey
- Potential investors in Homey
- Advertisers using Homey to promote properties or services.

1.4. Assumptions and Constraints

Assumptions

Users should have a device that is able to connect to the internet

Constraints

The web application is only available in English, language location support may be available in the future iterations

1.5. Initial UI Mockups

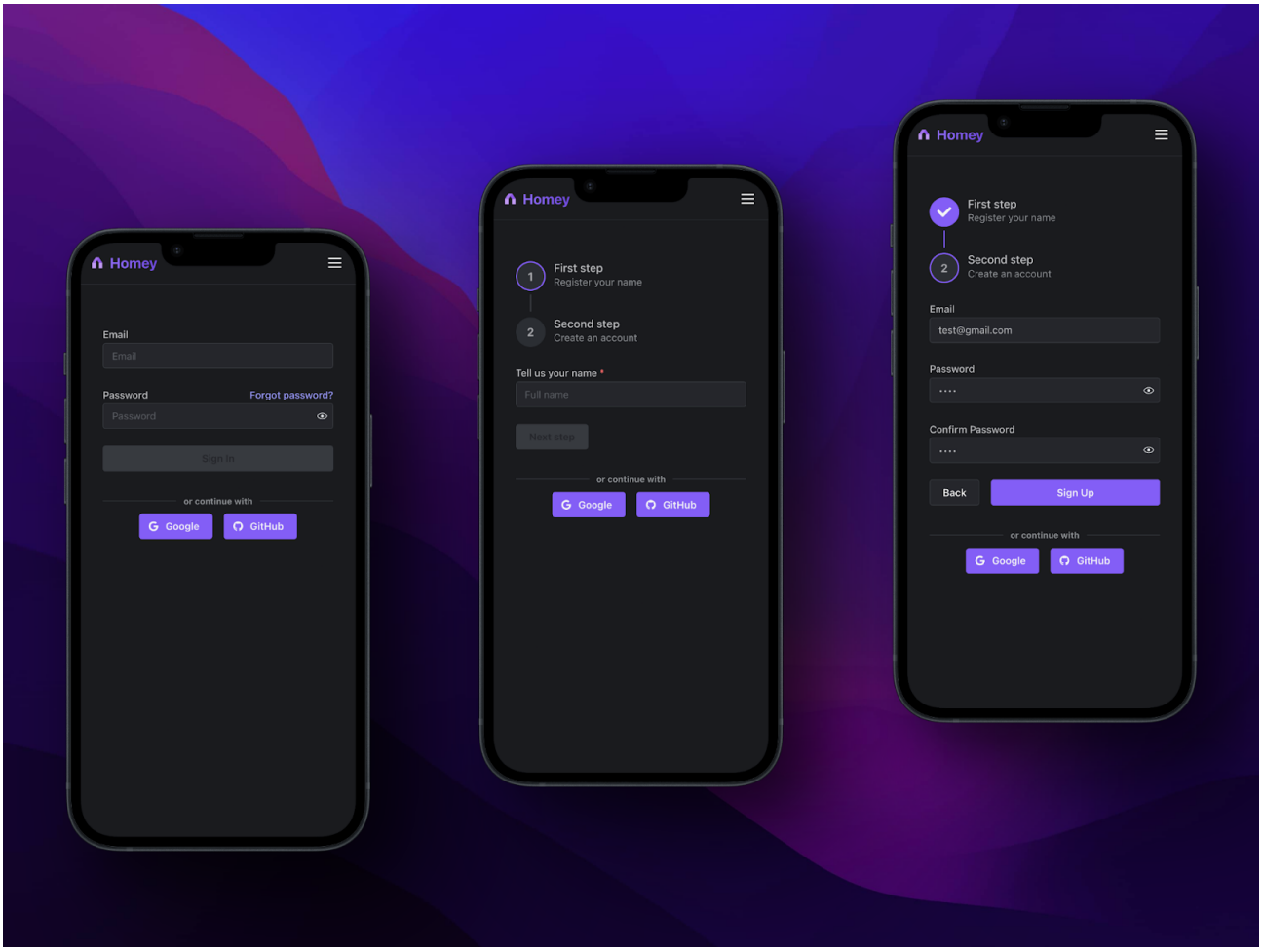


Figure 1.5.1, Sign In and Sign Up pages (mobile 320px)

Figure 1.5.1 shows the mobile mockups of the Sign In and Sign Up screens. The design is consistent with the overall aesthetic of the application, featuring the brand colour scheme and intuitive layout. Both screens offer the option for the user to authenticate via any one of the supported external providers.

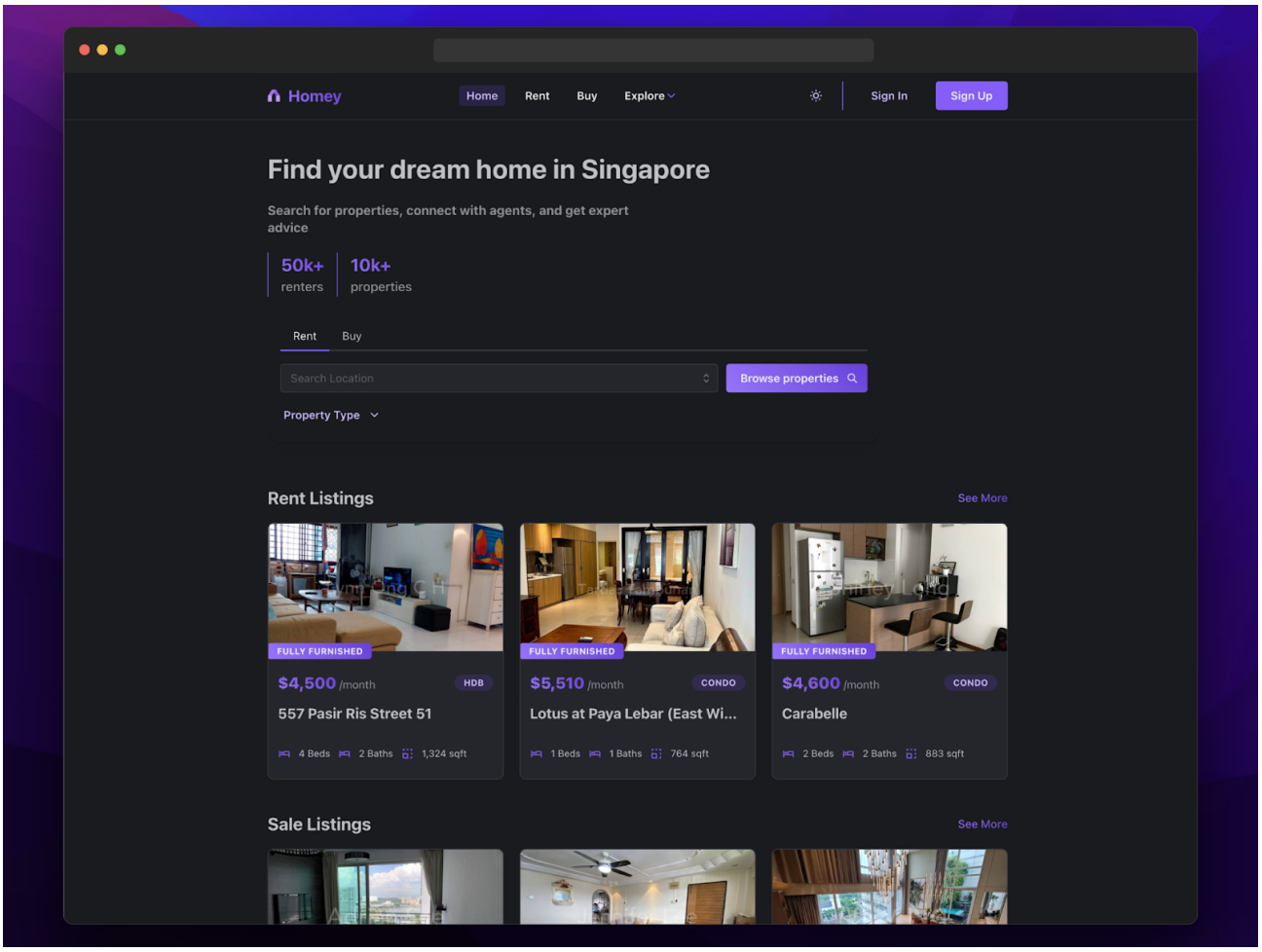


Figure 1.5.2, Landing page (desktop 1024px)

Figure 1.5.2 shows the landing page of Homey which features a large hero section prominently displayed at the top of the page. This section includes a multiselect location input field and a dropdown with property types to search listings. Below the hero section, the page showcases some of the featured listings currently available for either rent or sale, along with eye-catching photos and a brief description of each property.

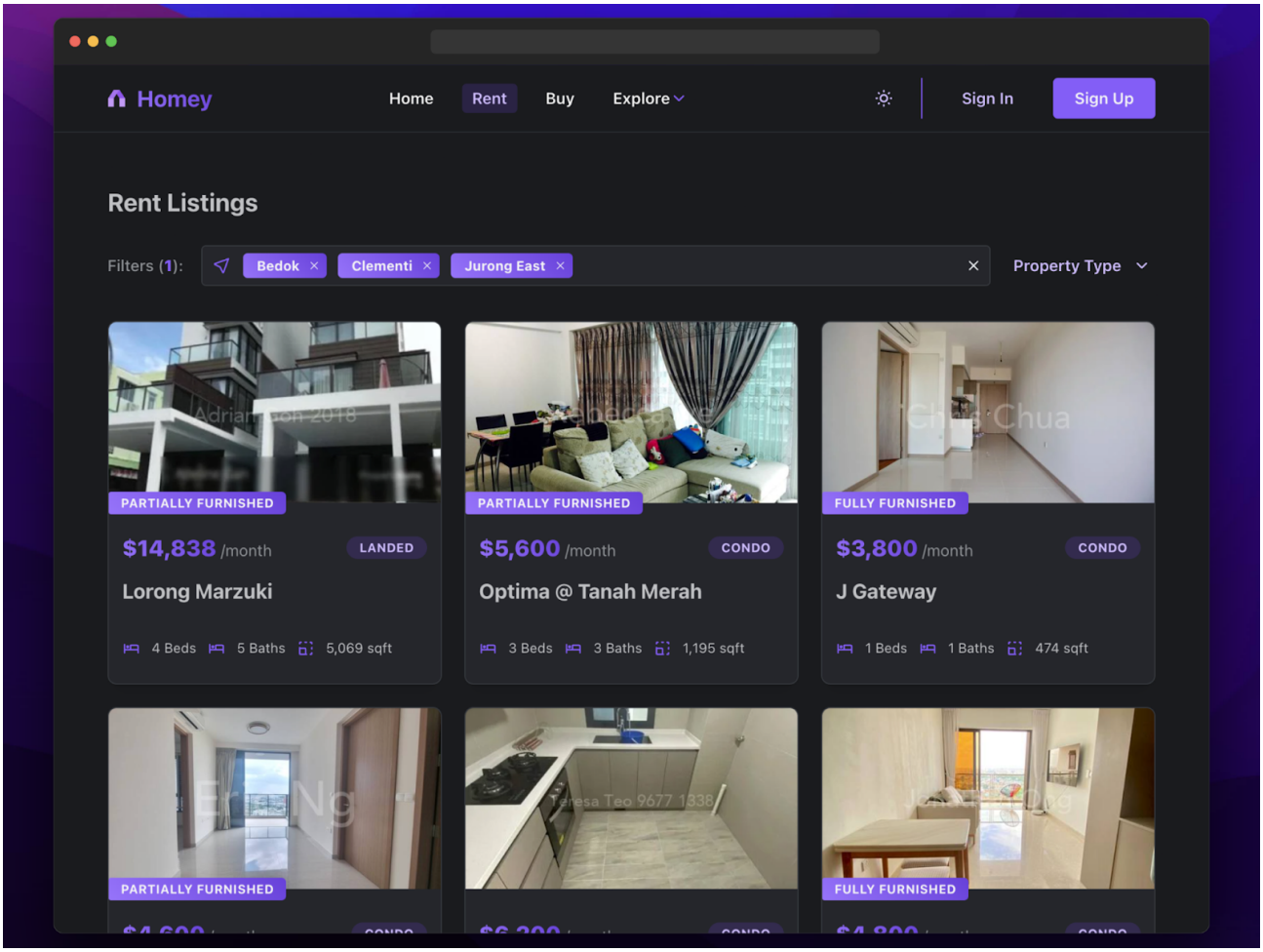


Figure 1.5.3, Listings page with search filters (desktop 1024px)

The Listings page of the application is a key feature that allows users to search for available properties based on their preferences. The page is designed with a clean and intuitive layout that displays property listings in a grid format with 3 columns. Each property listing includes a photo, the property type (HDB, Condo or Landed) and at most 3 locations.

The 3-column grid layout of the listings page allows users to easily compare and contrast different properties, while the search filters and sorting options provide a user-friendly way to refine their search and find the properties that best fit their needs.

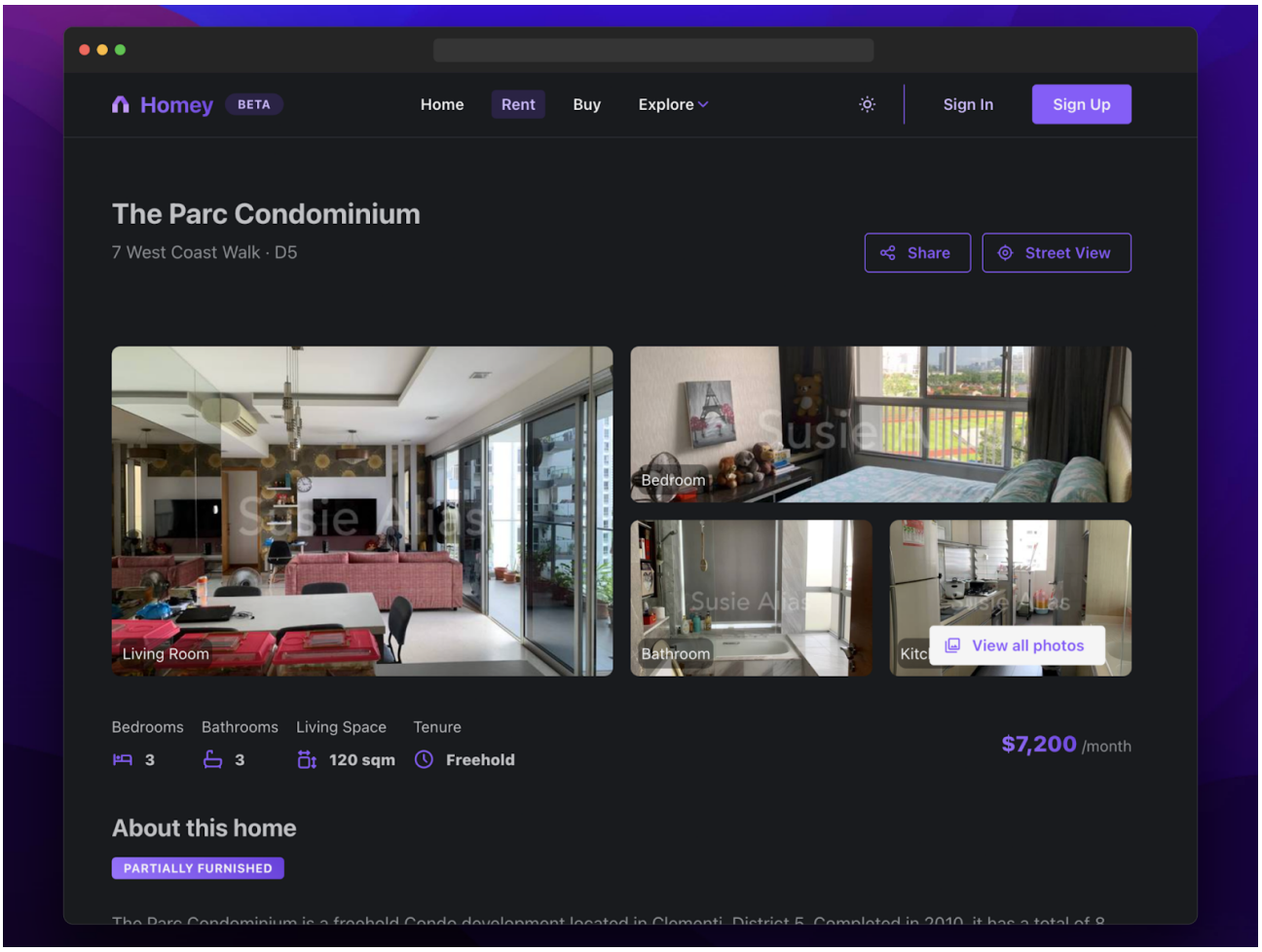


Figure 1.5.4, Listing page (desktop 1024px)

The listing page of the application features a detailed display of information about a particular property. The top of the page includes a large, high-quality photo of the property, along with basic details such as the property type (HBD, Condo, Landed), location, and description.

Below the main photo, the page displays additional information about the property, such as the number of bedrooms and bathrooms, square footage, and any additional amenities or features. The page also includes a detailed description of the property, highlighting key features and providing additional context about the property's history and location.

At the bottom of the page, users can find information about the real estate agent or agency representing the property, along with contact information and the option to schedule a showing or request more information about the property. Overall, the listing page provides a comprehensive and detailed view of a particular property, allowing users to fully evaluate and consider whether it meets their needs and preferences.

2. Functional Requirements

2.1. Account

Index / Functionality	Requirements
2.1.1 Sign Up	1. The user should be able to register for a new account 1.1. The required information includes: 1.1.1. Name 1.1.2. Email Address 1.1.3. Password & Confirm Password 1.2. The system should validate that all required fields have the sufficient inputs. 1.3. The system should validate that the input email address is unique and not part of an existing account. 1.3.1. If the input email address is not unique, system should display an error message 1.4. The system should validate that Password and Confirm Password matches. 1.4.1. If either input password or confirm password does not match, the system should display an error message. 1.5. The system should use the Argon2 hashing algorithm to perform salt-hashing on the input password before storing into the database. 2. Alternatively, the user should be able to register for a new account via supported external provider(s) 2.1. Supported externals providers includes: 2.1.1. Google 3. Upon successful sign up, the system should automatically sign in with the newly created account. 4. The system should create a new Session token consisting of the account information 5. The system should redirect the user to the landing page.
2.1.2 Sign In	1. The user should be able to sign in with their email and password 1.1. The system should validate that all required fields have the sufficient inputs. 1.2. The system should validate that the input email address is part of an existing account 1.3. The system will perform reverse salt-hashing on the database-stored password to compare with the input password.

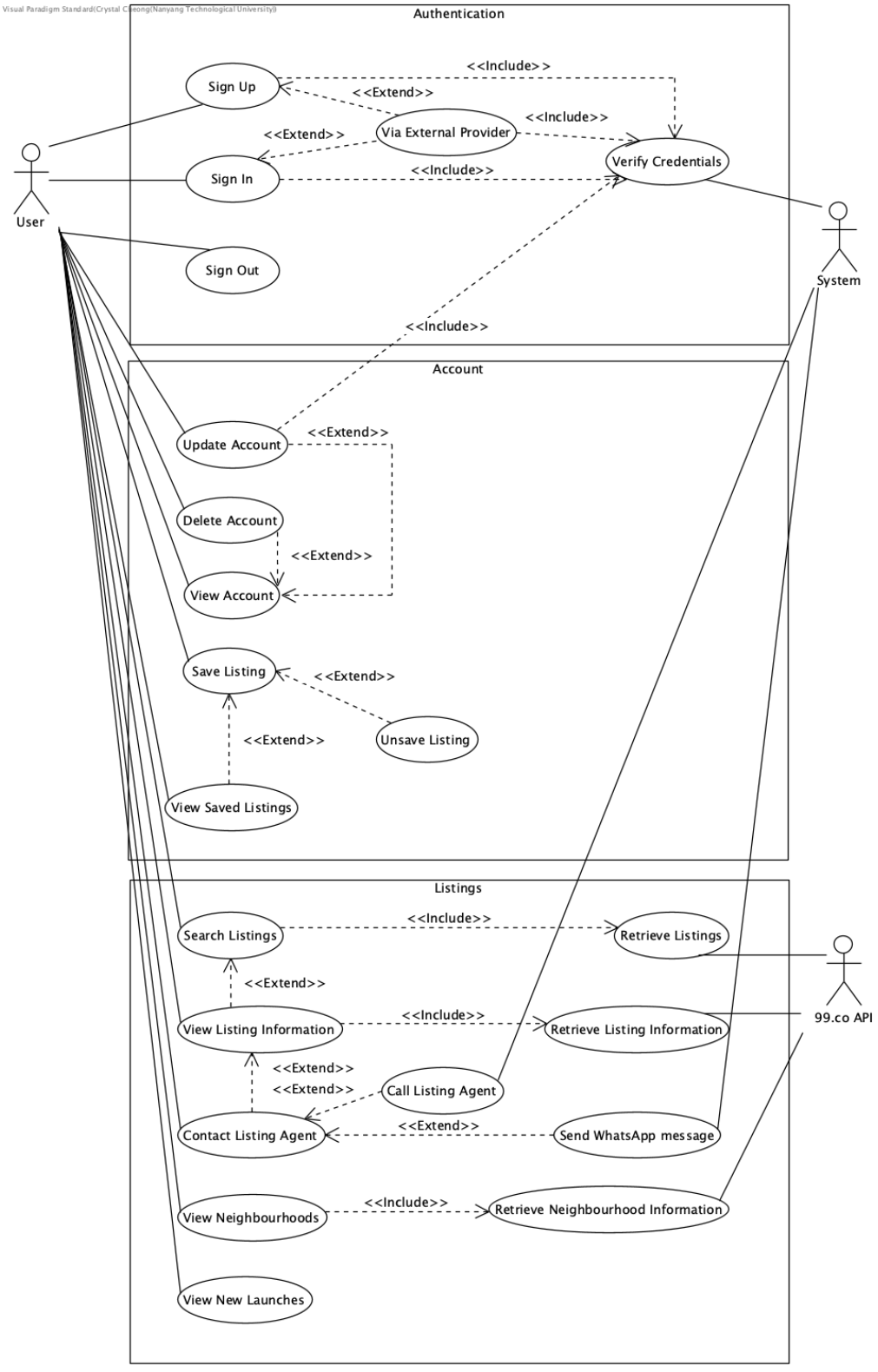
	2. Alternatively, the user should be able to sign in via support external provider(s). 2.1. Supported externals providers includes: 2.1.1. Google 3. Upon successful sign in, the system should create a new Session token consisting of the account information 4. The system should redirect the user to the landing page.
2.1.3 Sign Out	1. The user should be able to sign out 2. The system should destroy the current Session token associated with the user's account 3. Upon successful sign out, the system should display `Sign In` and `Sign Up` buttons in place of the account information in the navigation bar.
2.1.3 Update Account Information	1. The user should be able to view their account information 2. The user should be able to edit their account information 2.1. Editable fields includes: 2.1.1. Name 2.2. The system should validate that at least one of the editable fields has input value 2.3. The system should validate that the input value differs from the current account information 2.4. Upon successful validation, the user should be able to click on the `Save Changes` button to submit their changes 2.5. The system should perform an update operation on the database record with the matching account email address to update the account information 2.6. If unsuccessful, the system should display an error message. 2.7. The system should display the updated account information.
2.1.4 Delete Account	1. The user should be able to request for account deletion 2. Upon the user clicking the `Delete Account` button, the system will perform a delete operation on the database record with the matching account email address. 3. If unsuccessful, the system should display an error message. 4. The system will destroy the current Session token associated with the deleted account.
2.1.5 Save Listing	1. The user must log in first in order to save their favourite listing 2. The user should be able to save their favourite listing 3. The user should be able to view all their saved listings

	4. The user should be able to unsave specific saved listings

2.2. Listings

Index / Functionality	Requirements
2.2.1 Search	1. The user should be able to search for property listings based on filters 1.1. Filters includes: 1.1.1. Location 1.1.2. Property Type 1.1.2.1. HDB 1.1.2.2. Condo 1.1.2.3. Landed 1.2. The system should retrieve the location input from a multiselect field that allows at most 3 locations. 1.3. The system should provide autocompletion for the location multiselect input field 1.4. The system should be able to query for property listings based on the given filters 1.5. If no listings are available based on the filters, the system display an error message
2.2.2 View Listing Information	1. The system should be able to query for the specific listing with the route parameters. 1.1. Route parameters includes: 1.1.1. Listing ID 1.1.2. Cluster ID 2. The system should be able to retrieve the specific listing information from the 99.co API 3. The system should be able to display information about the specific listing. 3.1. Information includes: 3.1.1. Address Name 3.1.2. Listing Photos 3.1.3. Listing Description 3.1.4. Listing Agent Information 3.1.5. Listing Agent Contract Information 3.1.6. Embedded Google Map view of the listing location 3.2. The system should display a `Save` button for authenticated users to save the listing

3. Use Case Diagram



4. Use Case Descriptions

4.1. Account

Use Case ID:	11		
Use Case Name:	Sign In		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the sign-in authentication process.
Preconditions:	- Device must be connected to the internet - User account must already exist in the database
Postconditions:	- User is able to save listings - User is able to view account information - Account name is shown in the navbar
Priority:	
Frequency of Use:	
Flow of Events:	1. User enters their email and password into the appropriate fields 2. User clicks on the `Sign In` button 3. System validates the account by matching the entered credentials with the database 4. System authenticates the user to sign-in successfully and is redirected to the Home page
Alternative Flows:	AF1: Sign with External Provider (Google) 1. Return to Step 4 AF2: Invalid credentials 1. System displays error message “Sign In Failed” 2. Return to Step 1 AF3: Invalid email input 1. System displays error message “Invalid Email”
Exception:	EX1: User not logged in to their External Provider (Google) account 1. System will prompt the user to input their External Provider (Google) account credentials to be authenticated
Includes:	1. Validate Credentials
Special Requirements:	

Assumptions:	
Notes and Issues:	

Use Case ID:	12		
Use Case Name:	Sign Up		
Created By:	Crystal Cheong	Last Updated By:	Ng Mu Rong
Date Created:	29/01/2023	Date Last Updated:	25/03/2023

Actor:	User, System
Description:	This use case describes the sign-up authentication process.
Preconditions:	- Device must be connected to the internet - User account must not already exist in the database
Postconditions:	- User is able to save listings - User is able to view account information - Account name is shown in the navbar
Priority:	
Frequency of Use:	
Flow of Events:	1. User enters their name and clicks `Next` 2. User enters their email and password into the appropriate fields 3. System validates the inputs before enabling the submission 4. User clicks on the `Sign Up` button 5. System validates the email address to ensure that its unique 6. System authenticates the user to sign-in successfully and is redirected to the Home page
Alternative Flows:	AF1: Sign with External Provider (Google) 1. Return to Step 5 AF2: Invalid credentials 1. System displays error message “Sign Up Failed” 2. Return to Step 1 AF3: Invalid email input 1. System displays error message “Invalid Email”
Exception:	EX1: User not logged in to their External Provider (Google) account 1. System will prompt the user to input their External Provider (Google) account credentials to be

	authenticated
Includes:	1. Validate Credentials
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	13		
Use Case Name:	Verify Credentials		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the session authentication process.
Preconditions:	- Device must be connected to the internet
Postconditions:	
Priority:	
Frequency of Use:	
Flow of Events:	1. System will check for the current authentication status and skip to step 5 if the active session is already authenticated 2. System will create a new User account in the database if one does not already exist and query the authenticated User account if it exists. 3. System will create a new Session using the account 4. System will also a create a new JSON Web Token (JWT) for the aforementioned Session 5. System will emit the current authenticated status and Session data
Alternative Flows:	AF1: User is already authenticated 1. Skip to step 5 AF2: User account is updated 1. System will query for the updated User account in the database 2. Proceed to step 3 AF3: User is signed out 1. System will destroy the authenticated Session and JWT

	token
Exception:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	14		
Use Case Name:	Via External Provider		
Created By:	Crystal Cheong	Last Updated By:	Ng Mu Rong
Date Created:	29/01/2023	Date Last Updated:	25/03/2023

Actor:	User, System
Description:	This use case describes the external provider authentication process.
Preconditions:	- Device must be connected to the internet - User has a supported external provider account - System supports at least one external provider
Postconditions:	- User is successfully authenticated
Priority:	
Frequency of Use:	
Flow of Events:	1. User selects `Continue with Google` button 2. User is already logged into their Google account 3. System will validate the account availability with respect to the database records 4. On the first time, the User will be prompted to grant permission for System to connect using User's Google account 5. Upon granting permission, User will be signed-in successfully using their Google account and is redirected to the Home page
Alternative Flows:	AF1: Invalid credentials 1. System displays error message "Sign Up Failed" 2. Return to Step 1
Exception:	EX1: User not logged in to their External Provider (Google)

	account 1. System will prompt the user to input their External Provider (Google) account credentials to be authenticated
Includes:	1. Validate Credentials
Special Requirements:	
Assumptions:	
Notes and Issues:	Supported External Providers: - Google

Use Case ID:	15		
Use Case Name:	Sign Out		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the sign-out process
Preconditions:	- Device must be connected to the internet - User account must already exist in the database - User is currently authenticated
Postconditions:	- User is signed out
Priority:	
Frequency of Use:	
Flow of Events:	1. User clicks on `Sign Out` in the Account submenu 2. System will destroy the current Session and JWT token associated to the User 3. System will redirect to the Home page
Alternative Flows:	
Exception:	
Includes:	1. Verify Credentials
Special Requirements:	
Assumptions:	

Notes and Issues:	

Use Case ID:	16		
Use Case Name:	Update Profile		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the process of updating the User account
Preconditions:	- Device must be connected to the internet - User account must already exist in the database - User is currently authenticated
Postconditions:	- User account has been updated accordingly
Priority:	
Frequency of Use:	
Flow of Events:	1. User clicks on `Account Settings` in the Account submenu and redirected to the Account page 2. User enters input into the appropriate fields 3. System validates the inputs before enabling the submission 4. User clicks on `Save Changes` 5. System serialises the updated Account information into the database and updates the current Session token 6. System displays the updated Account information
Alternative Flows:	AF1: Input is no different from current 1. System prevents form submission
Exception:	
Includes:	1. Validate Credentials
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	17		
Use Case Name:	View Profile		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the process of viewing the user account
Preconditions:	- Device must be connected to the internet - User account must already exist in the database - User is currently authenticated
Postconditions:	
Priority:	
Frequency of Use:	
Flow of Events:	4. User clicks on `Account Settings` in the Account submenu and redirected to the Account page 5. System displays the updated Account information
Alternative Flows:	
Exception:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	18		
Use Case Name:	Delete Account		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the process of deleting the user

	account
Preconditions:	- Device must be connected to the internet - User account must already exist in the database - User is currently authenticated
Postconditions:	- User will be signed out - User's account is permanently deleted
Priority:	
Frequency of Use:	
Flow of Events:	1. User clicks on `Account Settings` in the Account submenu and redirected to the Account page 2. User clicks on the `Delete Account` button 3. System will sign out the current User before deleting the account from the database
Alternative Flows:	
Exception:	
Includes:	1. Verify Credentials
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.2. Listings

Use Case ID:	21		
Use Case Name:	Save Listing		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the process of saving a listing
Preconditions:	- Device must be connected to the internet - User account must already exist in the database - User is currently authenticated
Postconditions:	- Listing is saved and tagged to the User
Priority:	
Frequency of Use:	
Flow of Events:	1. System displays some property Listings and checks for the Session authentication status. 2. If authenticated, the System will display the bookmark icon on every Listing card 3. User clicks on the bookmark icon on one of the Listing cards 4. On click, System queries the database for the matching Property value and creates one if it does not already exists 5. System creates a new UserSavedProperty consisting of the UserID of the current User and PropertyID of the Listing 6. Upon completion, the System displays a toast notification that the Listing has been saved successfully
Alternative Flows:	AF1: Listing has already been saved 1. System will delete the UserSavedProperty record in the database with the matching UserID and ListingID
Exception:	
Includes:	1. Verify Credentials 2. View Listing Information
Special Requirements:	
Assumptions:	

Notes and Issues:	

Use Case ID:	22		
Use Case Name:	Unsave Listing		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the process of unsaving a listing
Preconditions:	- Device must be connected to the internet - User account must already exist in the database - User is currently authenticated - User account has at least one Saved Listing
Postconditions:	- Listing is unsaved and untagged to the User
Priority:	
Frequency of Use:	
Flow of Events:	1. User clicks on `Saved Listings` in the Account submenu and is redirected to the Saved page 2. User clicks on the filled bookmark icon on one of the Listing cards 3. System deletes the UserSavedProperty record in the database with the matching UserID and ListingID 4. System removes the Listing card from the Saved page
Alternative Flows:	
Exception:	
Includes:	1. View Saved Listings
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	23
Use Case Name:	View Saved Listings

Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the process of viewing saved listings
Preconditions:	- Device must be connected to the internet - User account must already exist in the database - User is currently authenticated - User account has at least one Saved Listing
Postconditions:	- User is able to view their Saved Listings
Priority:	
Frequency of Use:	
Flow of Events:	1. User clicks on `Saved Listings` in the Account submenu and is redirected to the Saved page 2. System shall query the database for all UserSavedProperty records with the matching UserID 3. System returns the data to the Saved page to be displayed
Alternative Flows:	AF1: User has no saved listings - System shall display the error message “No saved listings found”
Exception:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	24		
Use Case Name:	Search Listings		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System, 99.co API

Description:	This use case describes the process of searching listings
Preconditions:	- Device must be connected to the internet
Postconditions:	- System displays Listings with the matching filter values
Priority:	
Frequency of Use:	
Flow of Events:	1. User selects at least one location or property type 2. User clicks on `Browse Properties` and is redirected to the Listings page 3. System will invoke a request to 99.co API to retrieve property listings 4. Upon a successful response from 99.co API, System will filter through the results using the User input values to display the matching Listings
Alternative Flows:	
Exception:	
Includes:	1. Retrieve Listings
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	25		
Use Case Name:	Retrieve Listings		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	System, 99.co API
Description:	This use case describes the process of retrieving listings from the 99.co API
Preconditions:	- Device must be connected to the internet - Request was invoked from a frontend component
Postconditions:	- System returns a list of Listings
Priority:	

Frequency of Use:	
Flow of Events:	1. On its first query, the System shall query the Redis cache to retrieve cache data 2. If empty or not on first query, the System shall query the 99.co API endpoint for more listings data 3. Upon successful response of the first query, the System shall cache the results data before returning it to the frontend components to be displayed
Alternative Flows:	AF1: Response data is empty 1. Skip to step 4 AF2: Response is unsuccessful 1. System shall display the error message “Unable to fetch listings”
Exception:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	26		
Use Case Name:	View Listing Information		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the process of viewing a specific listing information
Preconditions:	- Device must be connected to the internet
Postconditions:	- System displays specific listing information
Priority:	
Frequency of Use:	
Flow of Events:	1. User clicks on any Listing card and is redirected to Listing page 2. System shall display Listing details such as: ○ Address Name

	○ Sale Price / Rental Price ○ No. of Bedrooms, No. of Bathrooms ○ Living Area ○ Contact Options (Call / WhatsApp)
Alternative Flows:	AF1: Invalid ListingID 1. System shall display the error message “Listing not found”
Exception:	
Includes:	1. Retrieve Listing Information
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	27		
Use Case Name:	Retrieve Listing Information		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	System, 99.co API
Description:	This use case describes the process of retrieving a specific listing information
Preconditions:	- Device must be connected to the internet
Postconditions:	- System returns data about a specific Listing
Priority:	
Frequency of Use:	
Flow of Events:	1. System shall query the 99.co API endpoint to retrieve listing information of the given ListingID 2. Upon successful response, the System shall cache the specific listing as `currentListing` in the browser’s local storage before returning it to the frontend components to be displayed
Alternative Flows:	AF1: Invalid ListingID 1. System shall display the error message “Listing not found” AF2: Listing data is empty

	1. System shall display the error message “Listing not found”
Exception:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	28		
Use Case Name:	Contact Listing Agent		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

Actor:	User, System
Description:	This use case describes the process of contacting a listing agent
Preconditions:	- Device must be connected to the internet - User is viewing a Listing page
Postconditions:	- User is able to send a message to a Listing agent
Priority:	
Frequency of Use:	
Flow of Events:	1. User clicks on any Listing card and is redirected to Listing page 2. System shall display Listing details such as: ○ Address Name ○ Sale Price / Rental Price ○ No. of Bedrooms, No. of Bathrooms ○ Living Area ○ Contact Options (Call / WhatsApp) 3. User clicks on the WhatsApp option and is redirected to WhatsApp prompt to `Continue to Chat` 4. User selects `Continue to Chat` button and sends the preset message on WhatsApp
Alternative Flows:	AF1: User did not select `Continue to Chat` button

	1. Flow is aborted and message would not be sent to listing agent
Exception:	
Includes:	1. Viewing Listing Information
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	29		
Use Case Name:	View Neighbourhoods		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

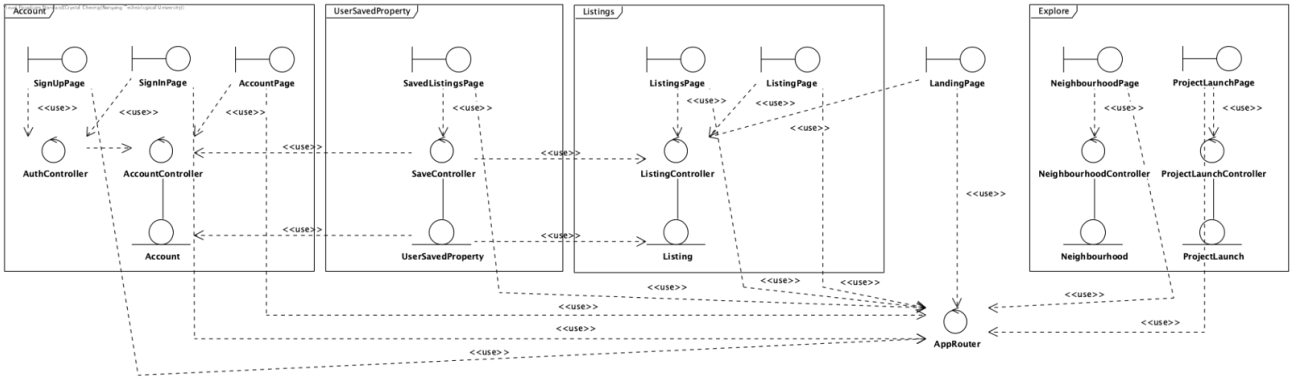
Actor:	User, System, 99.co API
Description:	This use case describes the process of viewing neighbourhoods
Preconditions:	- Device must be connected to the internet
Postconditions:	- System displays information about a specific Neighbourhood in Singapore
Priority:	
Frequency of Use:	
Flow of Events:	1. User selects `Neighbourhoods` under the Explore submenu and is redirected to the Neighbourhoods page 2. System displays a grid of neighbourhoods in Singapore 3. User clicks on one of the Neighbourhood cards and is redirected to the Neighbourhood page 4. System shall query the 99.co API endpoint and display the following neighbourhood information: ○ MRT Stations ○ Educational Institutions ○ Shopping Malls
Alternative Flows:	
Exception:	
Includes:	

Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	30		
Use Case Name:	View New Launches		
Created By:	Crystal Cheong	Last Updated By:	
Date Created:	29/01/2023	Date Last Updated:	

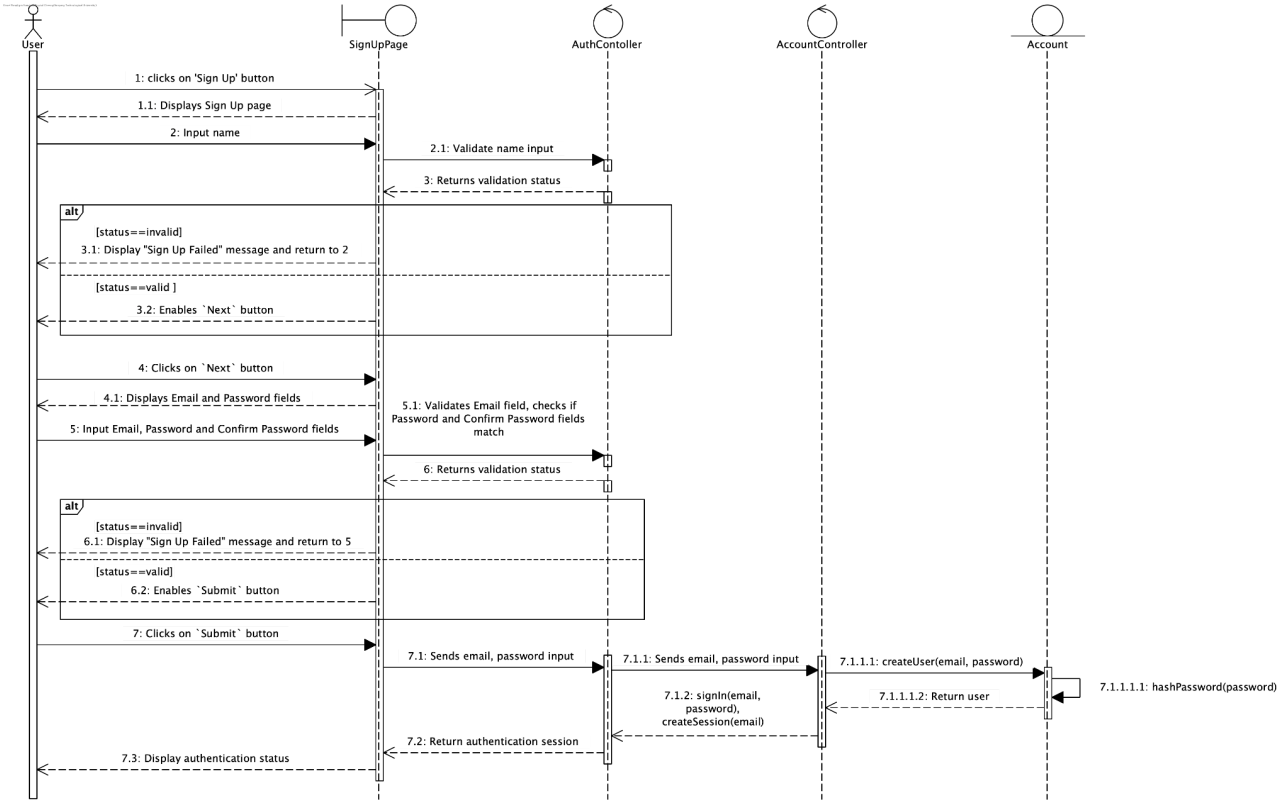
Actor:	User, System, 99.co API
Description:	This use case describes the process of viewing new condo launches
Preconditions:	- Device must be connected to the internet
Postconditions:	- System displays a grid of new condo launches in Singapore
Priority:	
Frequency of Use:	
Flow of Events:	1. User selects `New Launches` under the Explore submenu and is redirected to the New Launches page 2. System displays a grid of new launches in Singapore
Alternative Flows:	
Exception:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

5. Class Diagram

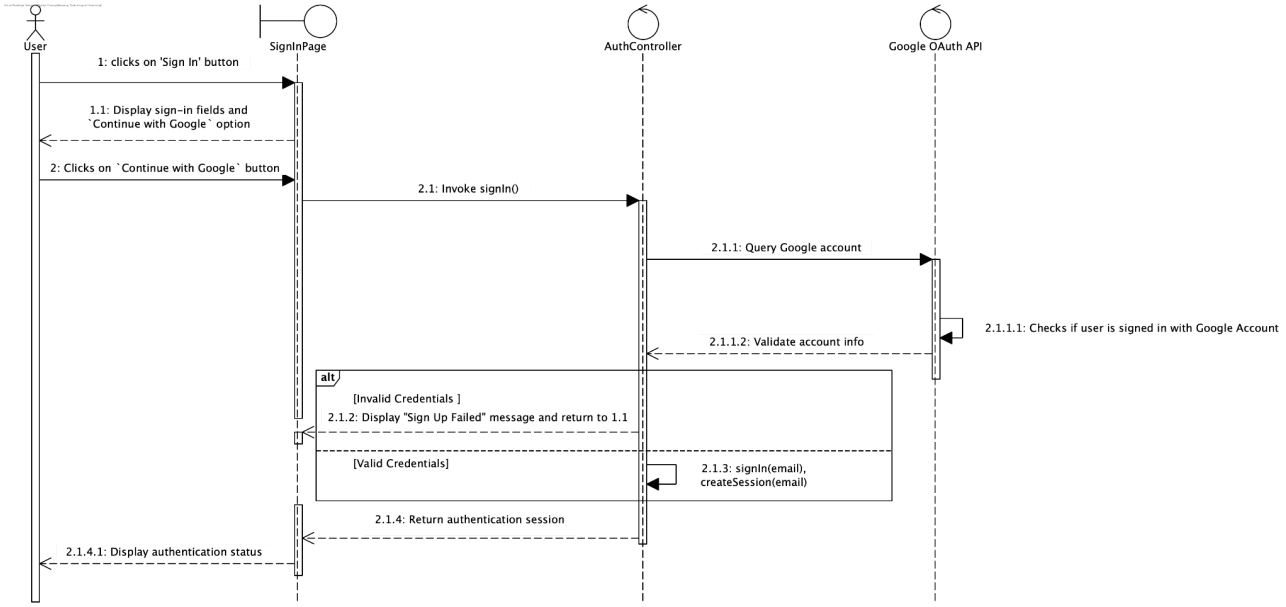


6. Sequence Diagrams

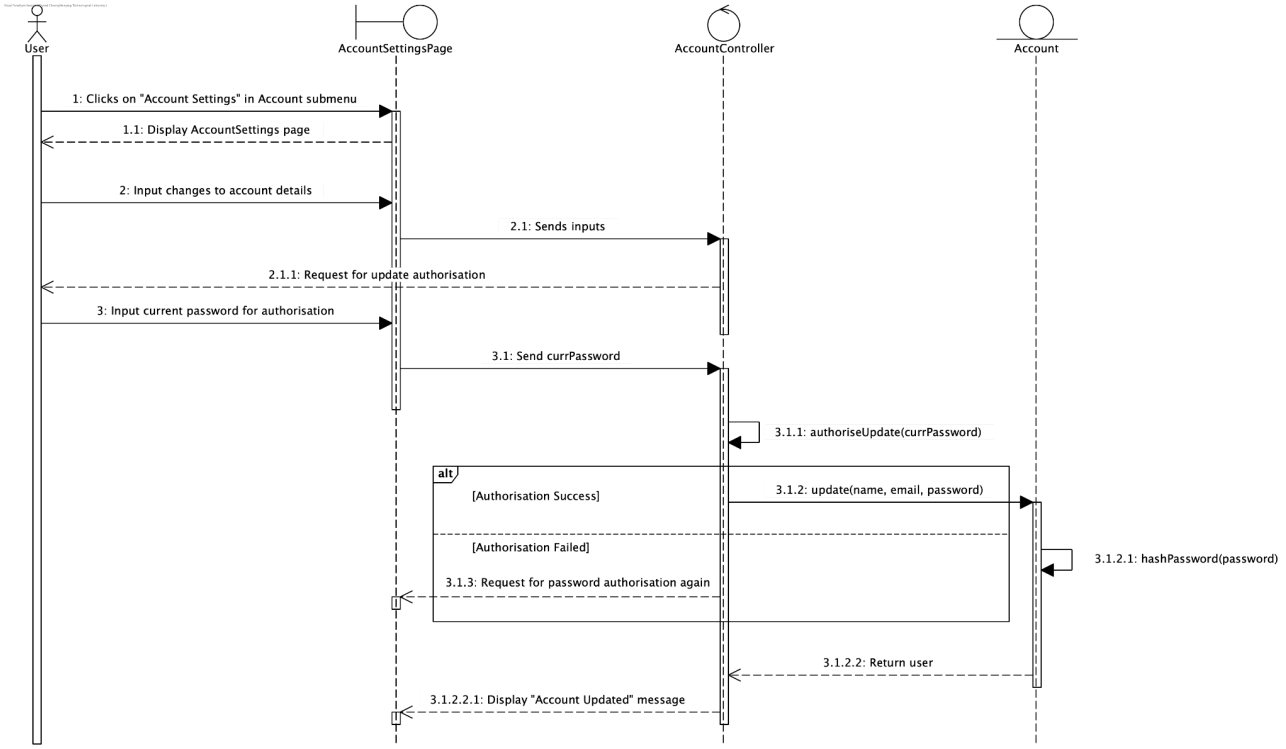
6.1. Sign Up



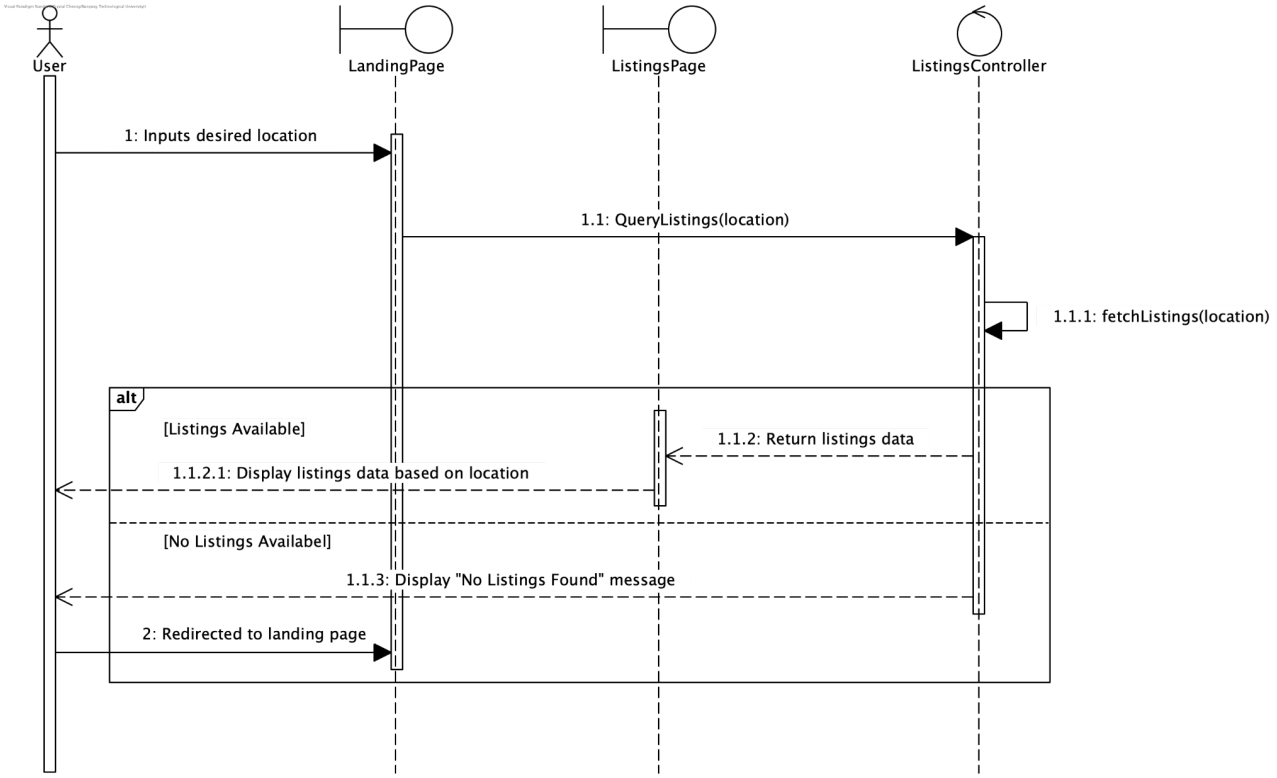
6.2. Sign In with Google



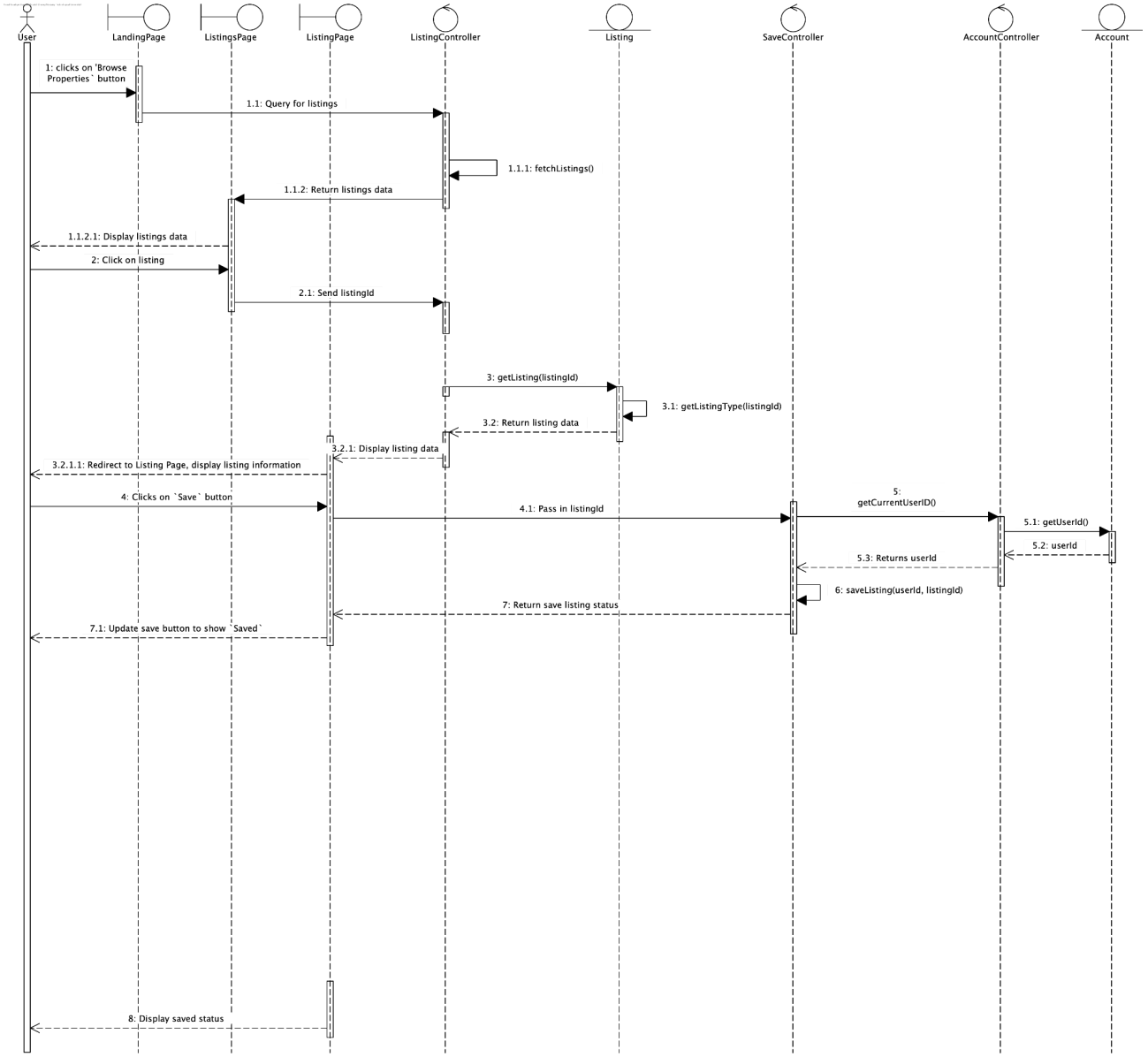
6.3. Update Account



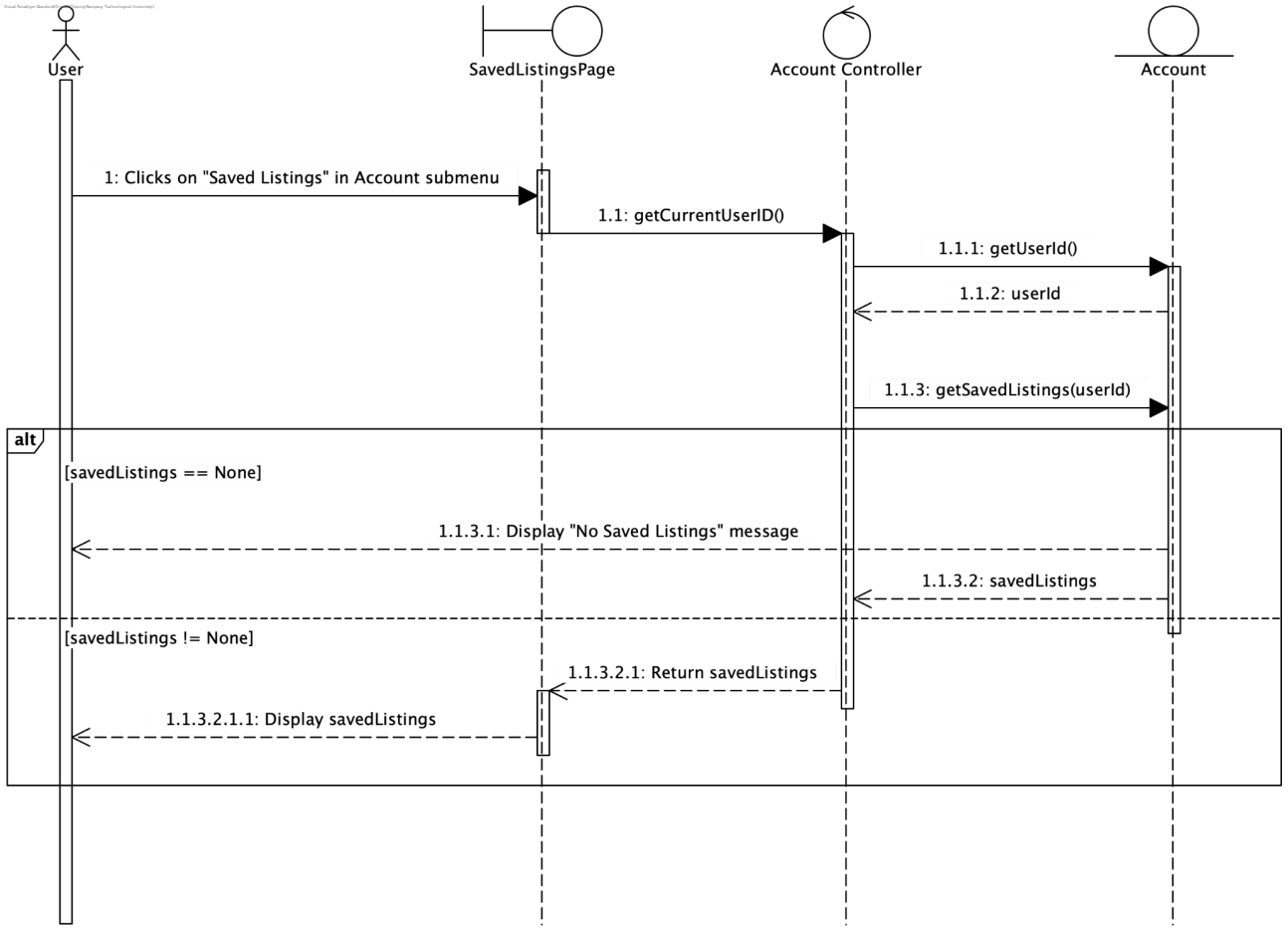
6.4. Search Listing



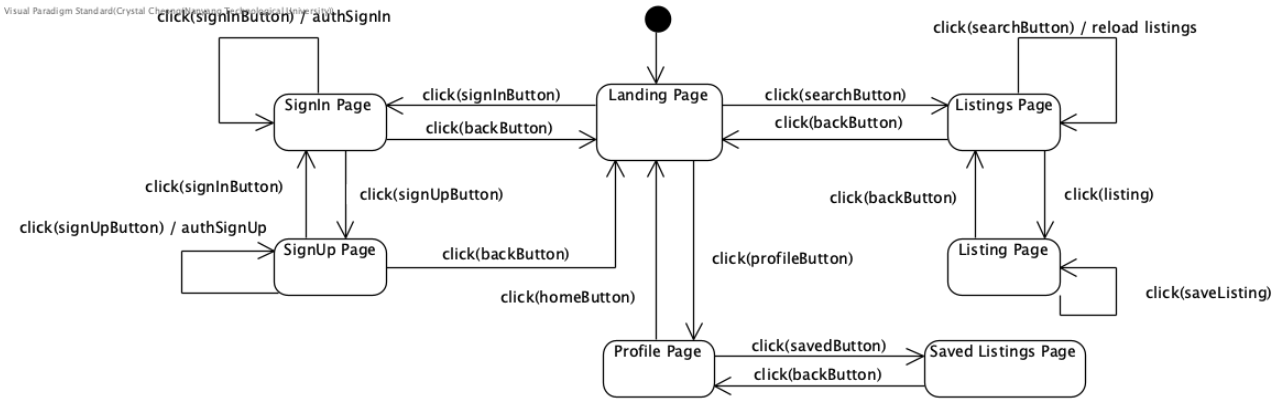
6.5. Save Listing



6.6. View Saved Listings



7. Dialog Map



8. Non-Functional Requirements

8.1. Usability

8.1.1. The system should have a user-friendly interface that is easy to navigate and understand.

8.1.2. The system should provide clear and concise instructions for using its features.

8.1.3. The system should be designed to provide a consistent user experience across all devices.

8.2. Performance

8.2.1. The system should load quickly and efficiently, with minimal delay or buffering.

8.2.2. The system should be able to handle high traffic loads during peak usage times.

8.2.3. The system should be optimised for fast search results and property listings.

8.3. Security

8.3.1. The system should provide secure user account management, with strong password policies and encryption of sensitive data.

8.3.2. The system should protect against malicious attacks such as SQL injection, cross-site scripting, and other common web application vulnerabilities.

8.3.3. The system should be regularly tested for security vulnerabilities and updated as necessary to ensure continued security.

8.3.4. The system will mask the password field in order to prevent any potential shoulder
8.3.5. surfing.
8.3.6. The system should use the Argon2 hashing algorithm to perform salt-hashing on all passwords before storing into the database.
8.3.7. Upon sign-in, the system will perform reverse salt-hashing on the database-stored password to compare with the input password.

8.4. Extendibility

8.4.1. The system should be designed to easily incorporate new features and functionality as needed.
8.4.2. The system should be able to integrate with other systems and services as required.
8.4.3. The system should be designed to scale and accommodate growing user numbers and increasing amounts of data.

9. Interface Requirements

9.1. User

9.1.1. The web application should have a user-friendly interface that is easy to navigate, with intuitive menus and buttons.
9.1.2. The search function should be prominent and easy to use, with clear filters and sorting options.
9.1.3. The website should have a consistent design and branding throughout all pages.

9.2. Hardware

9.2.1. The website should be designed to be compatible with a range of devices, including desktop computers, laptops, tablets, and mobile phones.
9.2.2. The application should be optimised for different screen sizes and resolutions.
9.2.3. The website should be designed to load quickly, even on slower internet connections.

9.3. Software

9.3.1. The website should be compatible with different web browsers, including Chrome, Firefox, Safari, and Edge.

9.3.2. The website should be designed to be responsive, adapting to different device types and screen sizes.

9.4. Communication

9.4.1. The website should be designed to be accessible through standard internet protocols, such as HTTP and HTTPS.

9.4.2. The website should be secure, with measures in place to protect user data and prevent unauthorised access.

9.4.3. The website should be designed to handle large amounts of traffic and user interactions.

10. Architecture Design

10.1. System Architecture Design

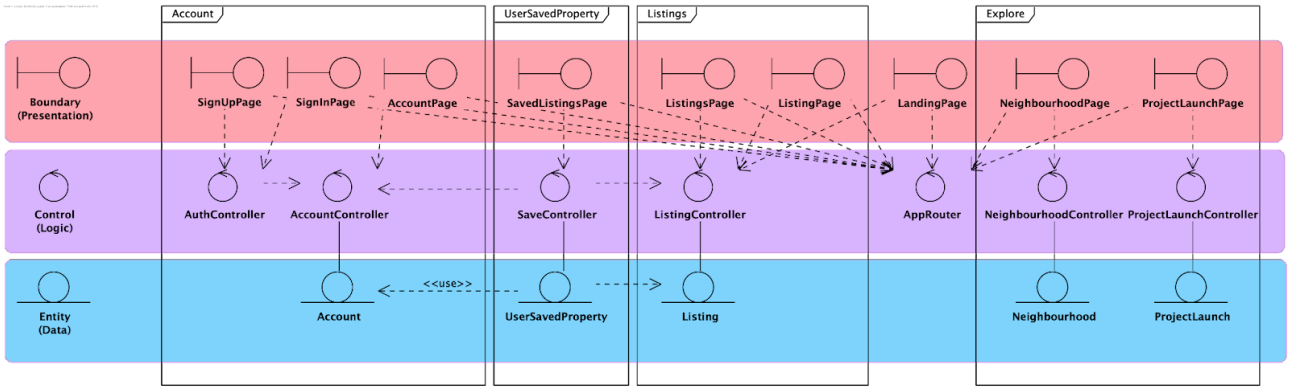


Figure 10.1.1, Stereotyped-form of System Architecture

The system architecture of Homey is designed to be scalable and robust, the general flow of dependencies trend downward from presentation, logic, to data layer. At the heart of the system is the logic layer, which is responsible for managing user interactions, serving content, and handling user data.

The web application is built on NextJS, a Javascript web framework. As such, it relies on an app router to route between pages. The router uses a combination of URL paths and React components to define the routes and content of each page. When a user clicks a link or enters a URL, the router checks the defined routes and loads the appropriate component, updating the content on the page. Hence, it is used by every boundary element.

10.2. Tech Stack

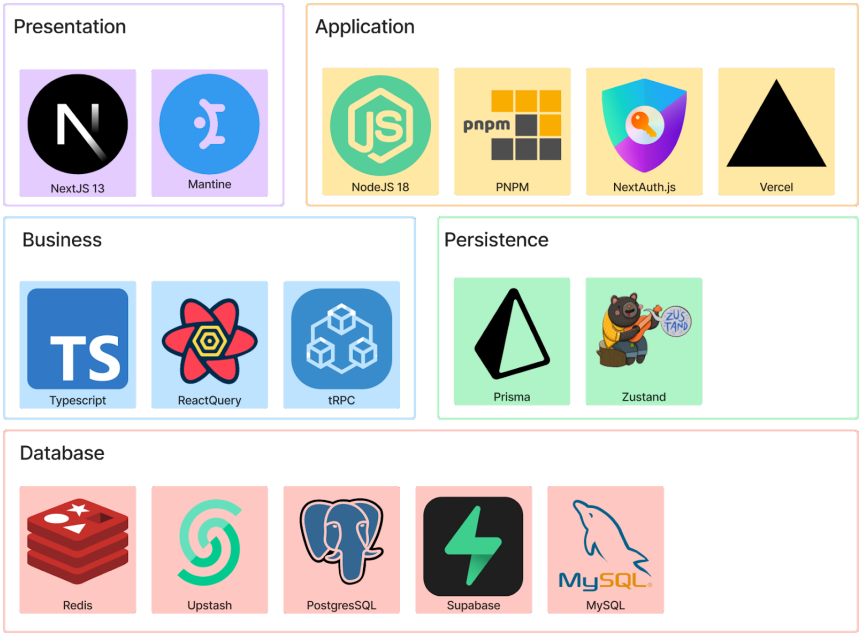


Figure 10.1.2, Tech stack in System Architecture

Presentation layer:

- NextJS 13 framework: used for building user interfaces and managing state
- HTML, CSS, and JavaScript: used for styling and interactivity
- Mantine: used for responsive design and UI components

Application layer:

- Node.js 18: used for server-side scripting and managing incoming requests
- PNPM: used for managing dependencies
- NextAuth.js: used for authentication
- Vercel: used for web hosting and deploying the application to production

Business layer:

- Typescript: used for scripting
- React Query: used for external queries
- tRPC: used to enforce type-safety for API queries

Persistence layer:

- Prisma: used for data modelling and schema syntax
- Zustand: used for application state management

Database layer:

- Redis: used for caching frequently accessed data
- Upstash: used for hosting Redis database
- PostgresQL, MySQL: used for storing and retrieving the data
- Supabase: used for hosting and scaling the PostgresQL database

10.3. Design Pattern

10.3.1. Data Fetching Pattern

Problem

Commonly used in web applications that involve fetching data from an external source (such as an API or database) and displaying it in the user interface. In NextJS applications, there are several challenges to data fetching, including:

- Slow initial load times: Fetching large amounts of data can result in slow initial load times, which can frustrate users and lead to high bounce rates.
- Inefficient re-fetching: Inefficient data fetching can lead to redundant and unnecessary API calls, which can slow down the application and put unnecessary strain on server resources.
- Complexity of data management: Managing data in a large-scale application can be complex, especially when dealing with multiple data sources and complex data relationships.

Solution

To address the above-mentioned challenges, the adoption of Data Fetching pattern is required.

```
const allListings: Map<ListingType, Listing[]> =
  useNinetyNineStore.use.listings();
const updateListings = useNinetyNineStore.use.updateListings();

const [
  { isFetching: isFetchingRentListings },
  { isFetching: isFetchingSaleListings },
] = api.useQueries((t) =>
  ListingTypes.map((listingType) =>
    t.ninetyNine.getListings(
      {
        listingType,
      },
      {
        enabled: !(allListings.get(listingType) ?? []).length,
        onSuccess(data) {
          if (!data.length) return;
          updateListings(listingType, data as Listing[]);
        },
      }
    )
  )
);
```

Figure 10.2.3.1, Data fetching in the LandingPage component

Firstly, the useQueries hook from tRPC is used to fetch two listing queries for rent and sale listings data. The useQueries hook can be used to fetch a variable number of queries at the same time using only one hook call. In this case, the hook is only enabled when there is either no available rent or sale listings data. Upon successful response from the API query, the listings state is updated with the listings data.

```
<Property.Grid
  key={`grid-${type}-${idx}`}
  listings={allListings.get(type) ?? []}
  isLoading={isTypeLoading}
  maxViewableCount={isTablet ? 4 : 3}
  allowSaveListing={isAuth}
  showMoreCTA
/>
```

Figure 10.2.3.2, Displaying the listings data in the LandingPage component

The listings data stored in the app state management is then utilised in the render function to be displayed by the LandingPage component.

10.4. Application Skeleton

The source code of the application adheres to the application skeleton of a typical Next.js project. By following this structure, developers can easily manage their code and assets, as well as create a scalable and maintainable application.

The notable key files and directories includes:

Key Files & Directories	Purpose / Functionality
`public` folder	This directory contains static assets such as images and fonts.
`src` folder	Contains the source code of the application, segmented into different subfolders such as components, pages, styles, etc.

`components` folder	Contains reusable UI components that are used across the application, such as buttons, forms, and navigation bars.
`pages` folder	This directory contains the different pages of the application. Each file in this directory represents a route in the application and is responsible for rendering the content of that route.
`styles` folder	This directory contains global styles that are used across the entire application.
`utils` folder	This directory contains helper functions and utilities that are used across the application.
`node_modules` folder	This directory contains all the dependencies of the application.
`package.json` file	This file contains the project's metadata and dependencies.
`next.config.js` file	This file contains the configuration settings for Next.js, such as the environment variables and build options.
`.env` file	This file contains credentials in key-value format for services used by the application.

11. Testing

11.1. Black-box

11.1.1. Sign In

Test Scenarios

ID	Scenario	Expected Output	Actual Output
TS-11-1	Account does not exist	System displays an error message ‘Sign In Failed, User does not exist.’	System displays an error message ‘Sign In Failed, User does not exist.’
TS-11-2	Login with valid account username and password	The system displays the main page for user to continue the operation	The system displays the main page for user to continue the operation
TS-11-3	Login without valid credentials	The system prompts the user of invalid credentials	The system prompts the user of invalid credentials
TS-11-4	Any of the fields are empty	System does not allow clicking of ‘Sign In’ button	System does not allow clicking of ‘Sign In’ button
TS-11-5	Login using Google	The system displays the main page for user to continue the operation	The system displays the main page for user to continue the operation
TS-11-6	Login using Github	The system displays the main page for user to continue the operation	The system displays the main page for user to continue the operation

Unit Cases

ID	Username	Password	Expected Output	Actual Output
TC-11-1	test1@gmail.com	test	Error: ‘Sign In Failed, User does not exist.’	Error: ‘Sign In Failed, User does not exist.’
TC-11-2	test@gmail.com	test	The system displays the main page for user to continue the operation	The system displays the main page for user to continue the operation
TC-11-3	test	test	Error: ‘Invalid email’	Error: ‘Invalid email’
TC-11-4	test@gmail.com	(Empty)	Unable to click on object ‘Sign In’	Unable to click on object ‘Sign In’

TC-11-5	(Empty)	test	Unable to click on object ‘Sign In’	Unable to click on object ‘Sign In’
TC-11-6	test@gmail.com	123	Error: ‘Sign In Failed, Invalid account credentials.’	Error: ‘Sign In Failed, Invalid account credentials.’

11.1.2. Sign Up

Test Scenarios

ID	Scenario	Expected Output	Actual Output
TS-12-1	Name field is empty or invalid	System prevents moving on to the next step	System prevents moving on to the next step
TS-12-2	Register with valid name, email and password	The system displays the main page for user to continue the operation	The system displays the main page for user to continue the operation
TS-12-3	Register with existing account email	The system prompts the user that user already exists	The system prompts the user that user already exists
TS-12-4	Register with invalid name	The system prompts the user of invalid credentials	The system prompts the user of invalid credentials
TS-12-5	Register with mismatch password	The system prompts the user of mismatch credentials	The system prompts the user of mismatch credentials
TS-12-6	Register with invalid email	The system prompts the user of invalid credentials	The system prompts the user of invalid credentials
TS-12-7	Email or password field is empty	System does not allow clicking of ‘Sign Up’ button	System does not allow clicking of ‘Sign Up’ button

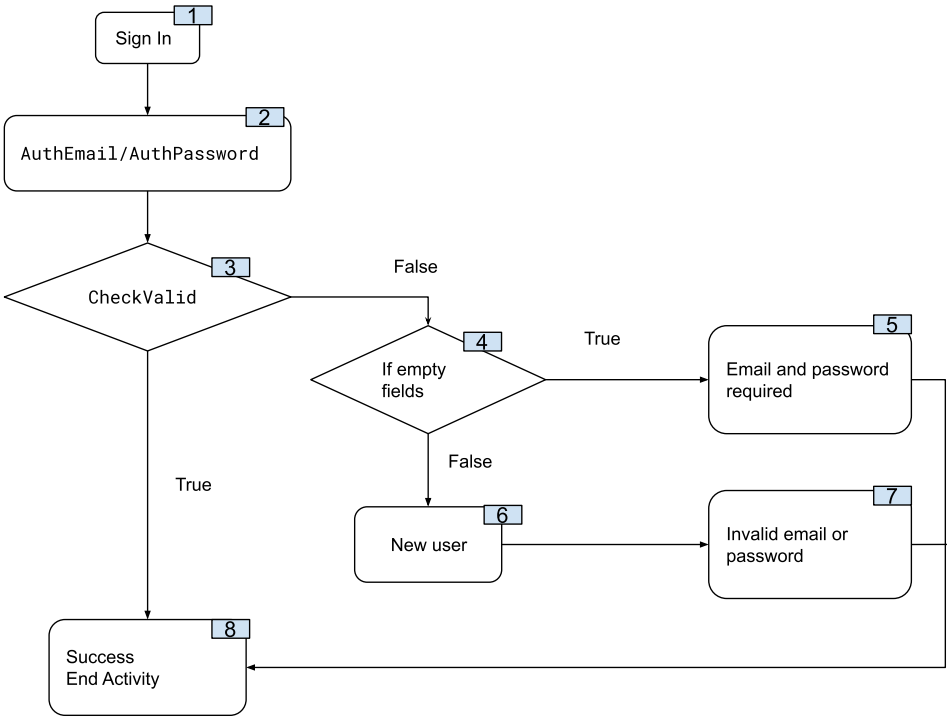
Unit Cases

ID	Name	Email	Password	Confirm Password	Expected Output	Actual Output
TC-12-1	123	-	-	-	Error: Invalid Name	Error: Invalid Name
TC-12-2	(Empty)	-	-	-	Unable to click on object ‘Next step”	Unable to click on object ‘Next step’

TC-12-3	testtwo	test2@gmail.com	test	test	The system displays the main page for user to continue the operation	The system displays the main page for user to continue the operation
TC-12-4	test	test@gmail.com	test	test	Error: Sign Up Failed User already exists	Error: Sign Up Failed User already exists
TC-12-5	testfour	test4@gmail.com	test	testing	Error: Passwords do not match	Error: Passwords do not match
TC-12-6	testfive	test5@.com	(Empty)	(Empty)	Error: Invalid Email	Error: Invalid Email
TC-12-7	testfive	(Empty)	test	test	Unable to click on object 'Sign Up"	Unable to click on object 'Sign Up'
TC-12-8	testfive	test5@gmail.com	(Empty)	(Empty)	Unable to click on object 'Sign Up"	Unable to click on object 'Sign Up"

11.2. White-box

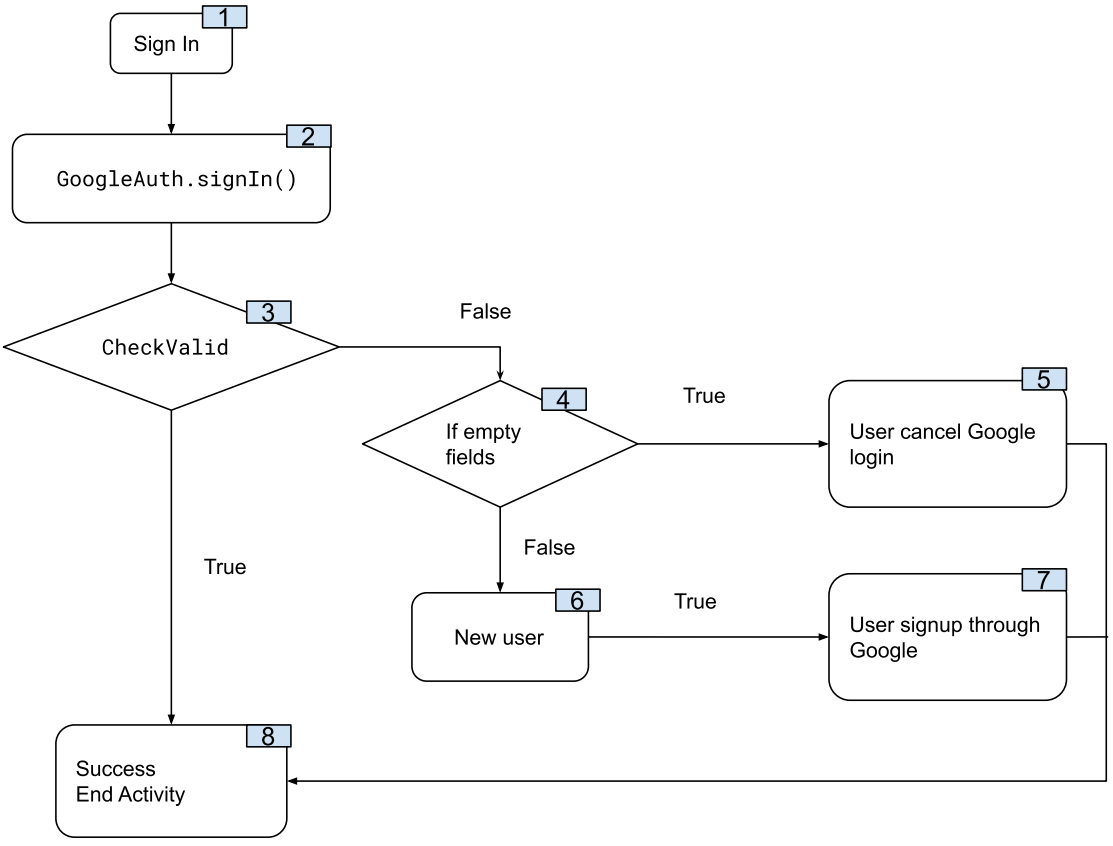
11.2.1. Sign In



Cyclomatic Complexity = 3

No.	Test Cases	Expected Basis Path	Actual Basis Path
1	Email: test@gmail.com Pw: test	1, 2, 3, 8	1, 2, 3, 8
2	Email: (Empty), Pw: (Empty) -> Email: test@gmail.com, Pw: test	1, 2, 3, 4, 5, 8	1, 2, 3, 4, 5, 8
3	Email: testhundred@gmail.com Pw: test	1, 2, 3, 4, 6, 7, 8	1, 2, 3, 4, 6, 7, 8

11.2.2. Sign In with Google

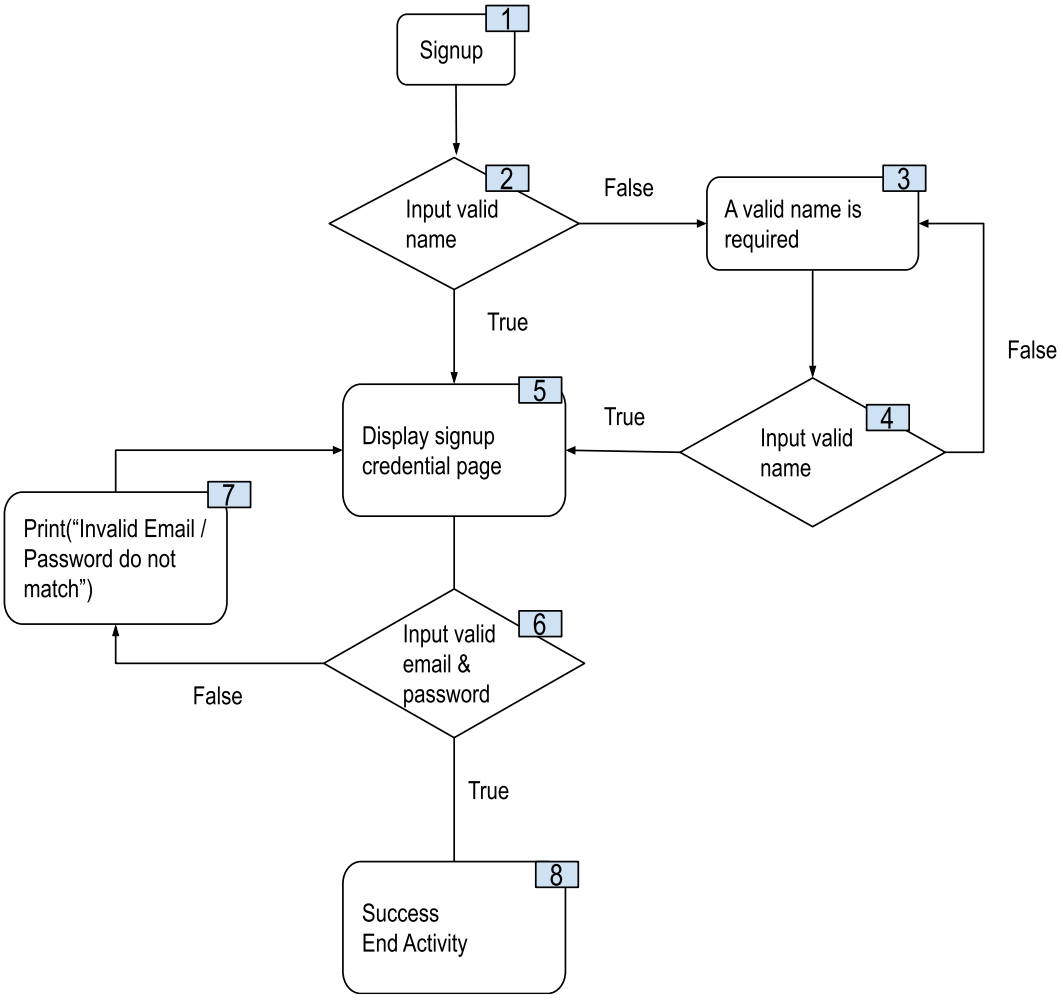


Cyclomatic Complexity = 3

No.	Test Cases	Expected Basis Path	Actual Basis Path
1	Login with valid GAccount	1, 2, 3, 8	1, 2, 3, 8
2	No email entered	1, 2, 3, 4, 5, 8	1, 2, 3, 4, 5, 8

3	Create new GAccount and Login	1, 2, 3, 4, 6, 7, 8	1, 2, 3, 4, 6, 7, 8

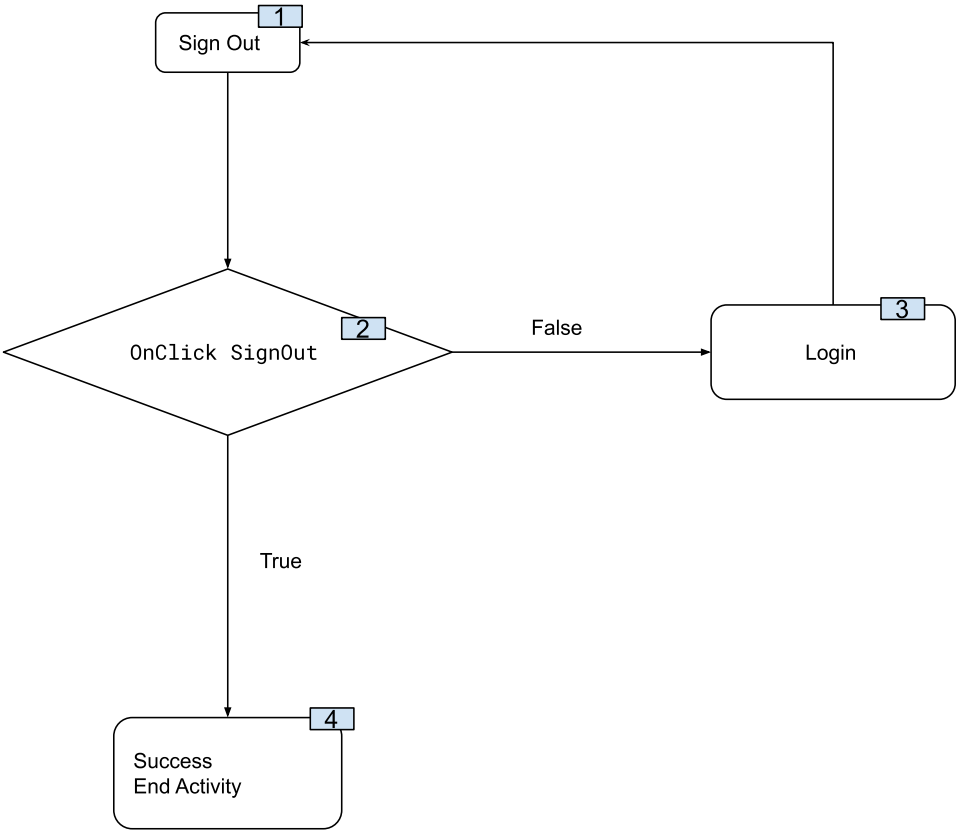
11.2.3. Sign Up



Cyclomatic Complexity = 4

No.	Test Cases	Expected Basis Path	Actual Basis Path
1	testtwo -> test2@gmail.com	1, 2, 5, 6, 8	1, 2, 5, 6, 8
2	123 -> testthree -> test3@gmail.com	1, 2, 3, 4, 5, 6, 8	1, 2, 3, 4, 5, 6, 8
3	123 -> 456 -> testfour -> test4@gmail.com	1, 2, 3, 4, 3, 4, 5, 6, 8	1, 2, 3, 4, 3, 4, 5, 6, 8
4	testfive -> test5@.com -> test5@gmail.com	1, 2, 5, 6, 7, 5, 6, 8	1, 2, 5, 6, 7, 5, 6, 8

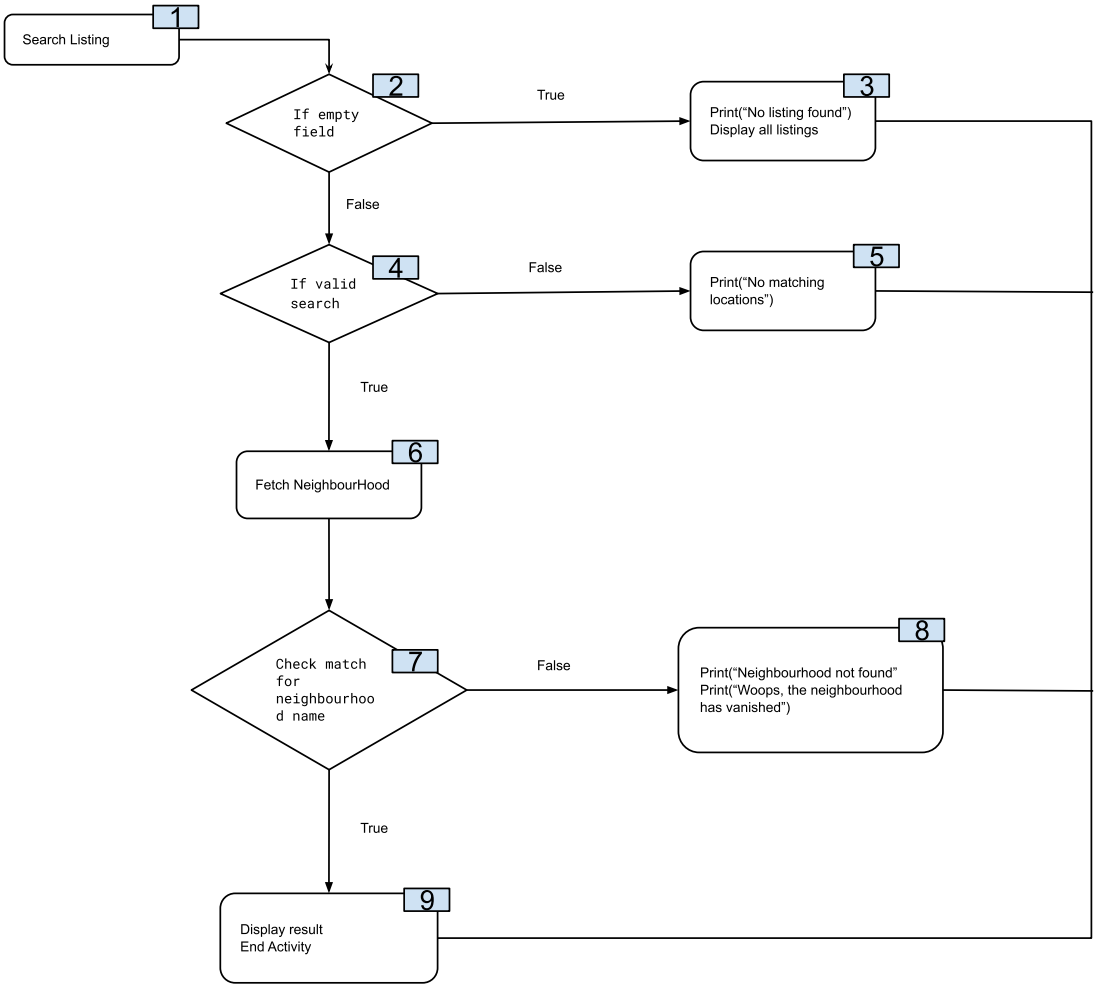
11.2.4. Sign Out



Cyclomatic Complexity = 4

No.	Test Cases	Expected Basis Path	Actual Basis Path
1	Hover over ‘Profile’ -> click ‘Logout’	1, 2, 4	1, 2, 4
2	User not logged in, sign out button not visible, User logs in, sign out button visible, clicks on sign out button	1, 2, 3, 1, 2, 4	1, 2, 3, 1, 2, 4

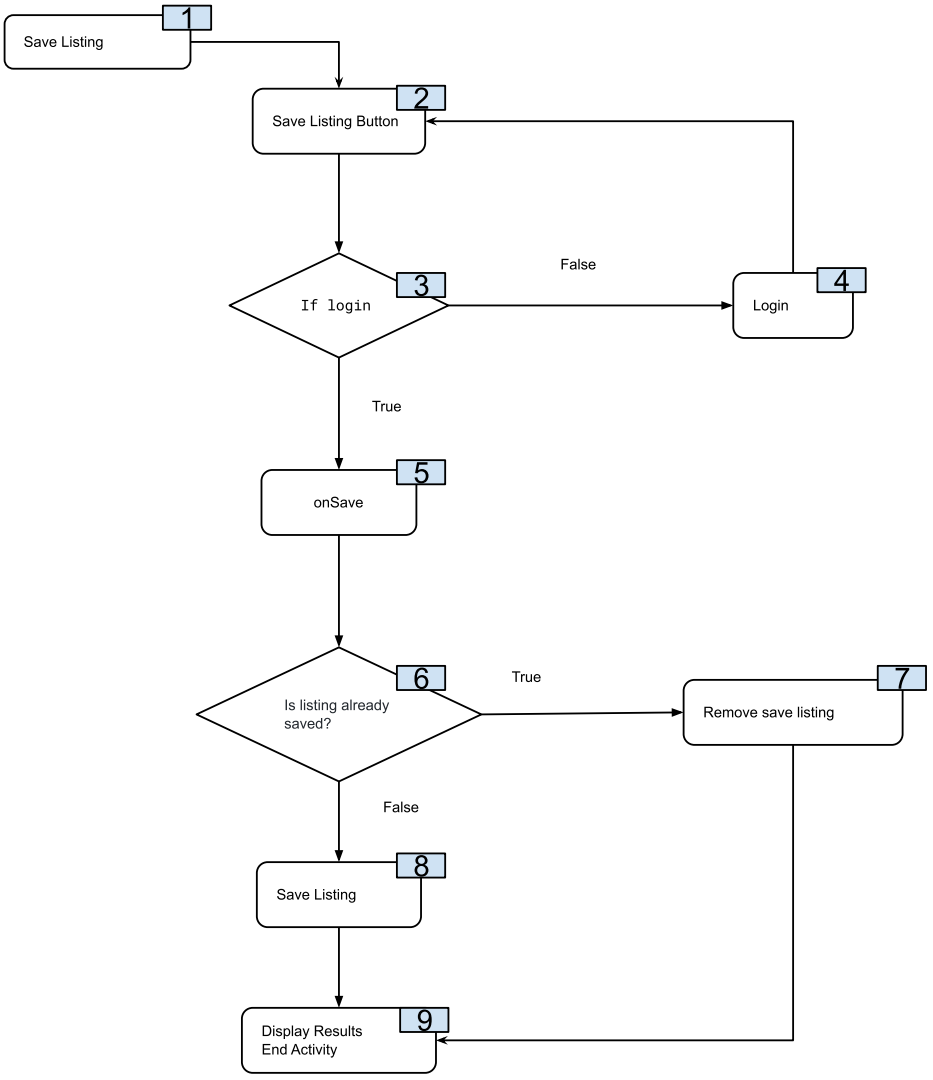
11.2.5. Search Listings



Cyclomatic Complexity = 4

No.	Test Cases	Expected Basis Path	Actual Basis Path
1	Search “Bedok”	1, 2, 4, 6, 7, 9	1, 2, 4, 6, 7, 9
2	No Input in search bar	1, 2, 3, 9	1, 2, 3, 9
3	Search “test”	1, 2, 4, 5, 9	1, 2, 4, 5, 9
4	Search “Aljunied”	1, 2, 4, 6, 7, 8, 9	1, 2, 4, 6, 7, 8, 9

11.2.6. Save Listing



Cyclomatic Complexity = 3

No.	Test Cases	Expected Basis Path	Actual Basis Path
1	User is logged in, clicks save listing button on unsaved listing	1, 2, 3, 5, 6, 8, 9	1, 2, 3, 5, 6, 8, 9
2	User not logged in, unable to save listings, logs in, clicks save listing button on unsaved listing	1, 2, 3, 4, 2, 3, 5, 6, 8, 9	1, 2, 3, 4, 2, 3, 5, 6, 8, 9
3	User is logged in, clicks save listing button on saved listing	1, 2, 3, 5, 6, 7, 9	1, 2, 3, 5, 6, 7, 9

12. Appendix

12.1. Data Dictionary

Term	Definition
System	Refers to the Homey web application
Listing	A property that is listed for rent or sale
User	A user is a person who is using the Homey application
Saved Listing	A listing that has been marked as a favourite by the user with an account
Salt	A salt is a random generated string that is used as an additional input to a one-way function that 'hashes' a password in cryptography.
Authenticated User	A user that is signed in and has their credentials verified to be true.