Software Requirements Specification

for

<PeePal>

Version 1.0 approved

Prepared by <Soong Jun Shen, Adam Soh Shi Jie, Joyce Lee Jia Xuan, Liew Jia Wei, Tan Kai Hooi>

<SC2006-FDAB-P1>

<15/04/2025>

Copyright © 2002 by Karl E. Wiegers. Permission is granted to use, modify and distribute this document.

Table of Contents

1. Introduction **1**

1.1 Purpose **1**

1.2 Document Conventions **1**

1.3 Intended Audience and Reading Suggestions **1**

1.4 Project Scope **2**

1.5 References **3**

2. Overall Description **3**

2.1 Product Perspective **3**

2.2 Product Features **4**

2.2.1 Use Case Diagram 4

2.2.2 Use Case Diagram 5

2.2.3 Class Diagram 6

2.3 User Classes and Characteristics **7**

2.4 Operating Environment **7**

2.5 Design and Implementation Constraints **8**

2.5.1 Frontend Constraints 8

2.5.2 Backend Constraints 8

2.5.3 Language Constraints 9

2.6 User Documentation **9**

2.6.1 README Files 9

2.6.2 Backend API Documentation 10

2.7 Assumptions and Dependencies **10**

2.7.1 Assumptions 10

2.7.2 Dependencies 11

3. System Features **13**

3.1 PeePal Features **13**

3.1.1 PeePalAutoDeleteSystem 13

3.2 User Authentication Features 14

3.2.1 AccountAuthentication 14

3.2.2 CreateAccount 14

3.2.3 LoginAccount 16

3.3 User Features 17

3.3.1 ToiletMenu 17

3.3.2 NavigateToilet 18

3.3.3 ReviewToilet 19

3.3.4 ReportReview 20

3.3.5 ReportNonExistence 20

3.3.6 UpdateToilet 21

3.3.7 UserMenu 22

3.3.8 ViewToiletOnMap 23

3.3.9 AddToilet 24

3.3.10 FavouritesToilet 25

3.3.11 UserProfile 26

3.3.12 SearchToilet 26

3.3.13 NearbyToilet 27

4. External Interface Requirements 28

4.1 User Interfaces 28

4.1.1 Style Guides 28

4.1.2 UI Mockups 31

4.1.3 Common Components 37

4.2 Hardware Interfaces 39

4.2.1 Device Types 39

4.2.2 Data and Control Interactions 39

4.2.2 Communication Protocols 40

4.3 Software Interfaces **40**

4.4 Communications Interfaces **41**

5. Other Nonfunctional Requirements **42**

5.1 Performance Requirements **42**

5.2 Safety Requirements **43**

5.3 Security Requirements **43**

5.4 Software Quality Attributes **43**

6. Other Requirements **44**

6.1 Internationalisation Requirements **44**

6.2 Legal Requirements **44**

6.3 Reuse Objectives **44**

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This document outlines the software requirements for PeePal, Version 1.0. This SRS outlines the functionalities, features, and constraints of the platform, aiming to provide a clear and comprehensive framework for development. The scope of this SRS encompasses the entire mobile platform, detailing the subsystems for toilet management, and public toilet indexing and navigation. This SRS is integral for guiding the development, testing, and maintenance of the platform, ensuring alignment with the project's objectives.

1.2 Document Conventions

This SRS follows standard documentation conventions to ensure clarity and consistency. Key conventions include:

- Font Styles: 'Arial' for body text, 'Times New Roman' for headers.
- Highlighting: Bold for key terms, italics for emphasis.
- Requirement Prioritization: Higher-level requirements have inherited priority levels, cascading down to detailed requirements. Each requirement statement is explicitly labeled with a priority level (High, Medium, Low).
- Requirement Identification: Each requirement is uniquely identified for easy reference.
- Versioning: Changes in requirements through different versions are tracked for historical reference and future revisions.

1.3 Intended Audience and Reading Suggestions

This document is structured to cater to a wide array of stakeholders and should be read differently depending on the reader’s role:

● **Project Managers and Stakeholders:** Begin with the Introduction to understand the purpose and scope, then proceed to the Overall Description for a high-level overview of the system.

● **Developers:** After the introductory sections, delve into the System Features for detailed descriptions of the platform's functionalities and the External Interface Requirements for integration specifics.

● **Testers:** Focus on the System Features for test case development and refer to the Nonfunctional Requirements for performance and security testing guidelines.

● **User Experience Designers:** The User Interface sections within the External Interface Requirements will be of particular interest, providing details on the interaction between the system and its users.

● **Technical Writers:** The entire document is relevant, with emphasis on User Documentation and Assumptions and Dependencies to ensure accurate and comprehensive user guides and help documentation.

● **Quality Assurance:** Refer to the System Features for functionality, Other Nonfunctional Requirements for performance benchmarks and the Other Requirements for additional specifications.

All readers are encouraged to review the document in its entirety, as each section builds on the information presented in the preceding ones, providing a complete understanding of the PeePal's requirements.

1.4 Project Scope

PeePal is a mobile application designed to help users locate and navigate to nearby public toilets in Singapore, using crowdsourced data. The primary objective of the platform is to provide a convenient, user-friendly solution for finding public toilets, with additional features that allow users to:

- Filter toilets based on specific amenities and preferences, such as bidets, showers, sanitizer availability, and accessibility for people with disabilities (OKU-friendly).
- Navigate to selected toilets.

- Rate and review toilets.
- Report non-existent or problematic toilets.
- Save their favorite locations for future use.

1.5 References

UI Inspiration:

https://www.airbnb.com.sg/

OOP Principles:

https://khalilstemmler.com/articles/object-oriented/programming/4-principles/

2. Overall Description

2.1 Product Perspective

This SRS details the requirements for PeePal, a mobile application designed to provide an end-to-end solution for public toilet indexing and navigation. PeePal is not part of an existing product family, nor does it replace any current systems. Instead, it introduces an approach to address the specific need for convenient and reliable public toilet location services. The platform functions independently, leveraging crowdsourced data and integrating with external technologies via APIs. A system architecture diagram (below) illustrates the system's interconnections and external interfaces.

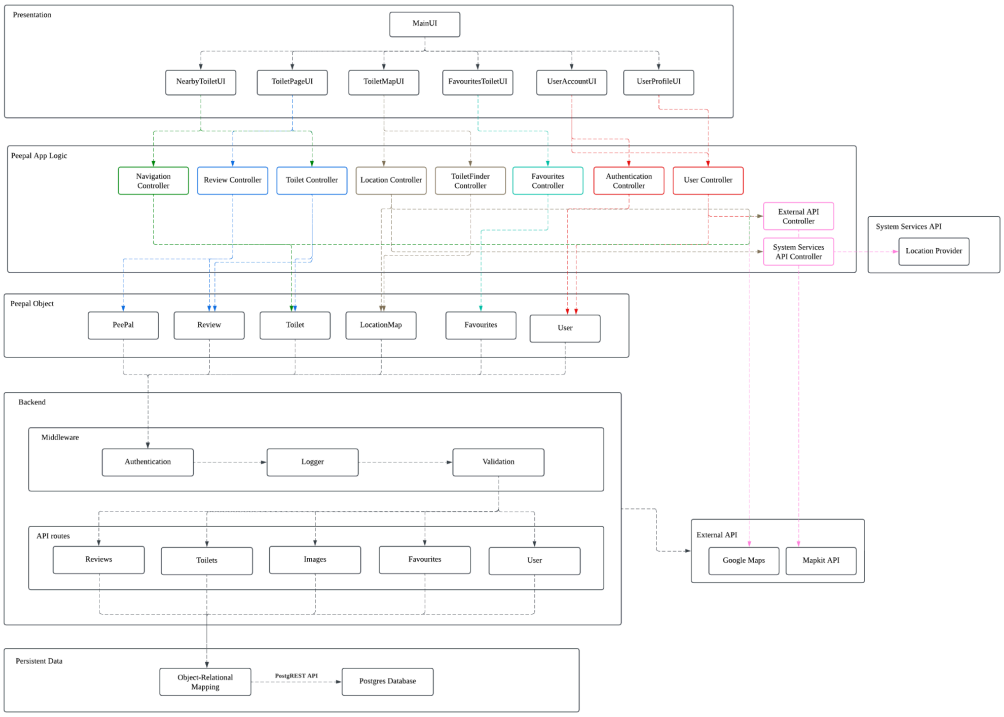


Figure 2.1.1: System Architecture Diagram

2.2 Product Features

2.2.1 Use Case Diagram

The following use case diagram (Figure 2.2.1) depicts the key product functionalities that PeePal includes.

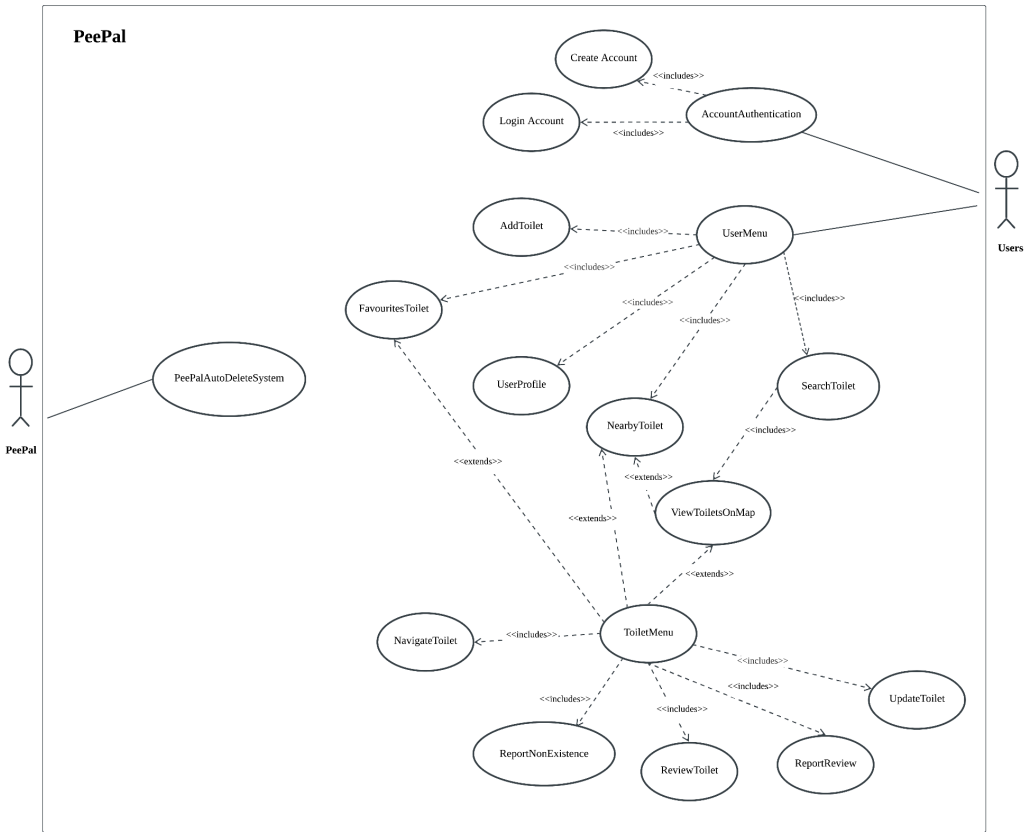


Figure 2.2.1: Use Case Diagram

2.2.2 Use Case Diagram

PeePal allows users to perform the following functions:

1. Authentication
    - **Sign Up**: Users can create an account.
    - **Login**: Users can log into their existing account.
    - **Account Authentication**: Users' credentials are validated during login to access their account.

2. User Menu
    - **User Profile**: Users can manage their profile.
    - **Search Toilet**: Users can search for toilets.
    - **Nearby Toilet**: Users can view toilets located near them.
    - **View Toilets on Map**: Users can view the locations of toilets on a map.

3. Toilet Management
    - **Add Toilet**: Users can add new toilets to the system.
    - **Update Toilet**: Users can update details of existing toilets.
    - **Report Non-Existence**: Users can report if a toilet listed is no longer available.
    - **Review Toilet**: Users can leave reviews for toilets.
    - **Report Review**: Users can report inappropriate reviews.

4. Favourites
    - **Favourites Toilet**: Users can mark toilets as their favourites for easy access later.
5. Navigate
    - **Navigate Toilet**: Users can navigate to their selected toilet’s location.
6. System Management
    - **PeePal Auto Delete System**: A System feature for auto-deleting outdated or unverified toilets from the database.

2.2.3 Class Diagram

The following Class Diagram (Figure 2.2.3) shows how different classes interact with each other, as well as the key functionalities that PeePal performs (refer to the controllers in the middle row section).

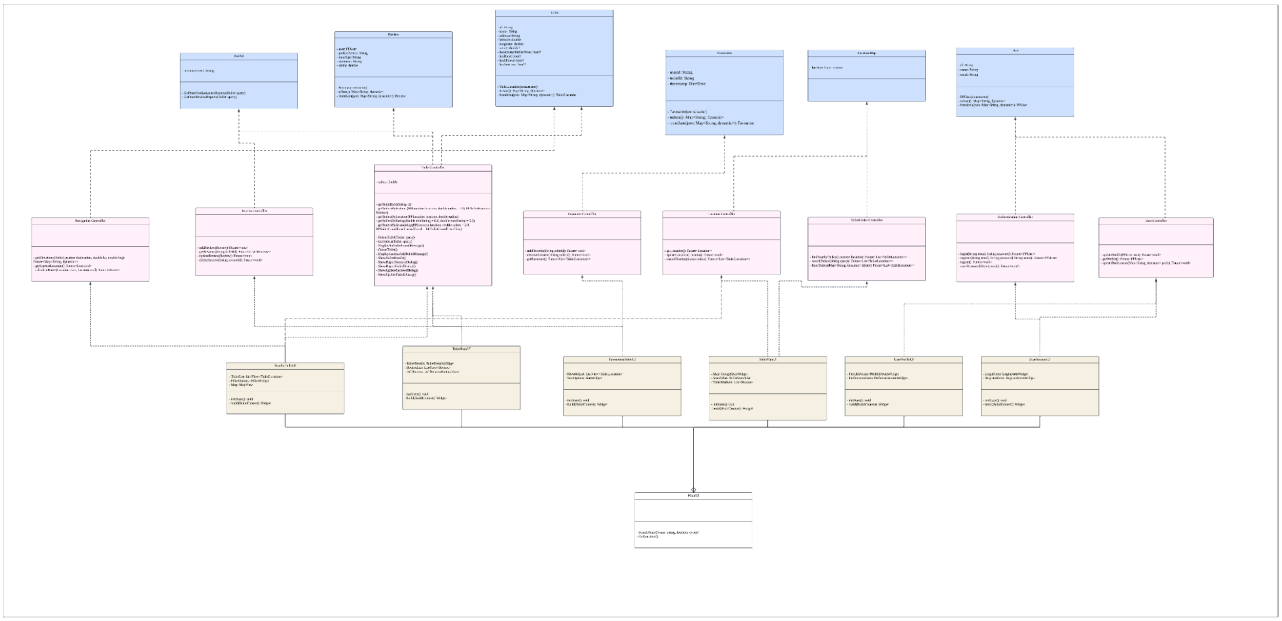


Figure 2.2.2: Class Diagram

2.3 User Classes and Characteristics

The primary user classes for PeePal include:

- **Users**: Individuals who interact with the application to perform tasks such as finding toilets, adding new toilets, viewing toilet details, updating existing information and submitting reviews. They may have varying levels of technical familiarity but primarily seek an intuitive, easy-to-navigate interface.

2.4 Operating Environment

The Peepal platform will be a mobile application, optimized for both iOS and Android devices. It will support the latest versions of mobile operating systems, utilizing Google Maps API and MapKit API for location services and map rendering. The application will be hosted on cloud services for scalability and reliability, integrating with external APIs such as MinIO for image storage. The backend will use PostgreSQL for database storage, ensuring robust data management and efficient queries.

2.5 Design and Implementation Constraints

The development of PeePal will be subject to several constraints that will guide the design and implementation of the software.

2.5.1 Frontend Constraints

2.5.1.1 Framework (Frontend)

The user interface of the Peepal application will be developed using Flutter. Flutter was chosen due to its ability to target both iOS and Android platforms with a single codebase, enhancing development speed and maintainability.

2.5.1.2 Code Formatting

Code formatting and consistency on the frontend will be enforced using Dart's built-in formatter for Flutter code and Prettier for any non-Dart code. This ensures that the code adheres to best practices, maintains readability and reduces errors during development.

2.5.2 Backend Constraints

2.5.2.1 Framework (Backend)

The backend server will be developed using Hono (TypeScript), chosen for its minimalistic and high-performance features, enabling rapid development of RESTful APIs.

2.5.2.2 Database

The database for Peepal will be a PostgreSQL relational database. SQL will be used for database interactions due to the structured and relational nature of the data, ensuring data integrity and efficient query handling.

2.5.2.3 Code Formatting

Code formatting and consistency on the backend will be enforced using ESLint for TypeScript code. This ensures uniform coding styles, proper linting and adherence to best practices, streamlining the development process and facilitating code reviews.

2.5.3 Language Constraints

- All software documentation, interfaces and user interactions will be provided in English.
- The choice of English ensures accessibility for a broader user base, as it is widely recognized as the primary or secondary language among the target audience.

2.6 User Documentation

2.6.1 README Files

2.6.1.1 Main Repository

A README file for the main repository will provide an overview of the Peepal project, detailing its architecture, setup instructions for the development environment, deployment steps and a high-level overview of the repository's structure.

2.6.1.2 Frontend Repository

The frontend repository will include a README file with instructions specific to the Flutter setup, using Flutter SDK, Xcode (iOS development), Android Studio (Android development) and installation.

2.6.1.3 Backend Repository

The backend repository will contain a README file with instructions on setting up the Hono backend server, configuring the environment, handling dependencies and integrating with the PostgreSQL database.

2.6.2 Backend API Documentation

Peepal will utilize Swagger/OpenAPI for interactive backend API documentation. This documentation will be auto-generated from the backend source code and will include details about:

- API routes
- Expected input parameters
- Authentication requirements
- Response models

The documentation will serve as a resource for frontend developers integrating with the backend and any third-party services that may need to interact with Peepal.

2.7 Assumptions and Dependencies

2.7.1 Assumptions

It is assumed that the Peepal platform will have continuous access to the internet for both frontend and backend operations.

2.7.2 Dependencies

2.7.2.1 Front-end Libraries

The user interface of PeePal depends on numerous front-end libraries:

Library	Purpose
Flutter	To create a cross-platform mobile application.
logging	To log messages throughout the application
flutter_bloc	To implement the BLoC pattern for state management.
provider	For dependency injection and managing app-wide state.
dio	For HTTP requests and networking. Low-level API to interact with the backend.
maps_toolkit	For map and geographical calculations and operations.
apple_maps_flutter	To integrate the Apple Maps view into the application.
geolocator	To access platform specific location services.
rxdart	For better usage of Dart streams and reactive programming.
cached_network_images	To implement image caching to prevent multiple requests to the same image resource.
image_picker	For enabling image upload features (e.g., toilet photos).

2.7.2.2 Back-end Libraries

The backend of PeePal depends on numerous back-end libraries:

Library	Purpose
Hono	A high-performance backend framework for developing REST APIs.
PostgreSQL	Relational database used for storing data (users, toilets, reviews).
Drizzle ORM	TypeScript ORM for database management.
MinIO	Object storage for image upload and management.
JWT	For secure, stateless user authentication.
bcrypt	To securely hash user passwords.
Zod	For input validation to ensure data integrity and security.

2.7.2.3 External Services

- Third-party APIs:
    - **Google Maps API**: For location-based services, retrieving toilets location.
    - **MapKit API**: For location-based services, real-time map display and route calculations.
    - **MinIO**: For secure image storage, handling uploads, downloads and image resizing.
- Cloud Hosting Services:
    - DigitalOcean or AWS (for hosting the backend services and PostgreSQL database).

Peepal relies on the availability and reliability of these external services to provide essential features like location tracking, image storage and mapping functionalities.

3. System Features

<Please refer to the 'Use Case Diagram and Descriptions.pdf' and 'Sequence Diagram.pdf' for the complete use case diagram, detailed use case descriptions and sequence diagrams for each use case.>

3.1 PeePal Features

3.1.1 PeePalAutoDeleteSystem

3.1.1.1 Description and Priority

The system automatically removes flagged reviews and non-existent toilets when the number of reports exceeds a predefined threshold (i.e. 3 in our current system).

Priority: High

3.1.1.2 Stimulus/Response Sequences

1. The system will retrieve the number of reports on non-existent toilets and inappropriate reviews.
    a. If a toilet was reported (non-existent) by more than or equal to 3 times, the system will delete the toilets from the database.
    b. If a review was reported by more than or equal to 3 times, the system will delete the review.
    c. If a review contains vulgar language, inappropriate content, or violates community guidelines, the system will censor the review.

3.1.1.3 Functional Requirements

REQ-1: App must successfully delete the toilet from the database if it was reported more than or equal to 3 times.

REQ-1.1: Deleted toilet must not appear in the Nearby Toilet page.

REQ-1.2: Deleted toilet must not appear in the search result.

REQ-1.3: Deleted toilet must not appear on the map.

REQ-2: App must successfully delete the review under the respective toilets, if it was reported more than or equal to 3 times.

REQ-2.1: Deleted review must not appear in the Toilet Details page.

3.2 User Authentication Features

3.2.1 AccountAuthentication

3.2.1.1 Description and Priority

User must authenticate their user accounts on PeePal by either logging in if they are existing users or creating a new account if they are first-time users.

Priority: High

3.2.1.2 Stimulus/Response Sequences

1. Users enter PeePal by clicking on the app.
2. Users are prompted with an account authentication menu, either Login or Sign up.
    a. If the users choose Login, the system will invoke the LoginAccount use case to request that users enter their login details.
    b. If users choose Sign Up, the system will invoke the CreateAccount use case to request that users enter their sign-up details.

3.2.1.3 Functional Requirements

REQ-1: App must successfully display the account authentication menu.

REQ-2: App must successfully navigate to the CreateAccount page if users click the ‘Sign-Up’ link.

REQ-3: App must integrate the LoginAccount and CreateAccount features.

3.2.2 CreateAccount

3.2.2.1 Description and Priority

Users can create a new user account if they are first-time users to PeePal.

Priority: Medium

3.2.2.2 Stimulus/Response Sequences

1. At the LoginScreen, people without an account with PeePal can click the 'Sign Up' link to create an account with PeePal.
2. The system will direct the users to the CreateAccountScreen and prompt them to enter required information into the specified field.
3. The users select the 'Create Account' button to confirm their inputs.
4. The system will then create an account for the user and store the user's information into the database.

3.2.2.3 Functional Requirements

REQ-1: App must allow the users to select the 'Sign Up' link.

REQ-2: App must prompt the users to enter required informations.

REQ-2.1: App must display the input field for "Full Name".

REQ-2.2: App must display the input field for "Email".

REQ-2.3: App must display the input field for "Password".

REQ-2.3.1: App must display an error message if the password entered is not at least 8 characters long.

REQ-2.3.2: App must hide/show the password when toggle the visibility icons.

REQ-2.4: App must display the dropdown field for "Gender" to select either Male, Female, or Others.

REQ-2.5: App must display an error message if any of the input fields have no inputs.

REQ-3: App must display a “Create Account” button for the users to select once all input has been keyed in.

REQ-4: App must create an account for the user once all information have been approved and verified.

3.2.3 LoginAccount

3.2.3.1 Description and Priority

Users can log into his/her PeePal account using his/her email and password.

Priority: High

3.2.3.2 Stimulus/Response Sequences

1. Users who choose to login with email and password, will have to key in their email address and password into the input fields.
2. Users will then click the “Log In” button.
3. The system will validate the input data.
4. Once authentication is successful, the users will be directed to the HomeScreen.

3.2.3.3 Functional Requirements

REQ-1: App must display the option to log in with email and password.

REQ-2: App must prompt the users to enter required informations.

REQ-2.1: App must display the input field for “Email”.

REQ-2.2: App must display the input field for “Password”.

REQ-2.2.1: App must display an error message if the password entered is not at least 8 characters long.

REQ-2.2.2: App must hide/show the password when toggle the visibility icons.

REQ-3: App must display a “Log In” button for the users to click once all required informations has been entered.

REQ-3.1: App must log users in successfully if users have an existing PeePal account and the inout data are all correct.

REQ-3.2: App must direct users to HomeScreen after successful login.

REQ-3.3: App must display error message if users do not have an existing PeePal account, or the input data are incorrect.

3.3 User Features

3.3.1 ToiletMenu

3.3.1.1 Description and Priority

This feature enables the users to select and perform various activities under the toilet page.

Priority: Medium

3.3.1.2 Stimulus/Response Sequences

1. The user will be able to view the information details of the selected toilets and the reviews under the comment section.
2. The user can select and perform various activities such as: NavigateToilet, ReviewToilet, ReportReview, UpdateToilet and ReportNonExistence.
3. The user can select an option now.
    a. User selects NavigateToilet.
        i. The system will invoke NavigateToilet use case, where the system will direct user to navigation page and guide user to the destination.
    b. User selects ReviewToilet.
        i. The system will invoke ReviewToilet use case, where the user can comment their reviews on their selected toilets.
    c. User selects ReportReview.
        i. The system will invoke ReportReview use case, where the user can report any inappropriate reviews under the comment section.
    d. User selects UpdateToilet.

i. The system will invoke UpdateToilet use case, where the user can suggest changes to the toilet's details and update it.

e. User selects ReportNonExistence.

i. The system will invoke ReportNonExistence use case, where the user can report the toilet if they found out the toilet doesn't exist in the real world.

3.3.1.3 Functional Requirements

REQ-1: App must successfully display the information details of the toilet.

REQ-2: App must display the reviews by other users under the comment section.

REQ-3: App must integrate the NavigateToilet, ReviewToilet, ReportReview, UpdateToilet and ReportNonExistence features.

3.3.2 NavigateToilet

3.3.2.1 Description and Priority

This feature enables the users to navigate to their selected toilet location.

Priority: High

3.3.2.2 Stimulus/Response Sequences

1. The user will be able to view the summary details of the selected toilet card.
2. The user can clicks on the 'Navigate" button.
3. The system will retrieve the user's current location.
4. The system will calculate the optimal route from the user's current location to the selected toilet location.
5. The system will display the navigation instructions on the navigation page and guide user to the destination using map displayed.

6. The system will display a successful message to the user once the destination is reached.

3.3.2.3 Functional Requirements

REQ-1: App must successfully display the summary details of the selected toilet card.

REQ-2: App must successfully display the “Navigate” button for user to select.

REQ-2.1: App must successfully direct user to the navigation page after the user click on the “Navigate” button.

REQ-2.2: The system must successfully render the Apple Map with the optimal route displayed on the map.

REQ-2.3: The system must successfully display the navigation instructions below the map.

REQ-2.3.1: The system must highlight the instruction to the user at every turning point of the route.

REQ-3: App must successfully relocate the map position with user’s current location at the center when user click on the “Center Current Location” icons.

REQ-4: App must successfully display successful message after the user reached the destination.

3.3.3 ReviewToilet

3.3.3.1 Description and Priority

This feature enables users to review the selected toilet.

Priority: Medium

3.3.3.2 Stimulus/Response Sequences

The user can click on the “Review” icons at the bottom right of the toilet page.
The user can enter the required information in the input field.

a. The user can enter the rating for the selected toilet.
b. The user can enter his/her comment for the selected toilet in the text box.
c. The user can upload the toilet's images.

The user can click on the 'Confirm" button after all required information has been entered.

The system will display the newly added comment under the comment section.

3.3.3.3 Functional Requirements

REQ-1: App must successfully display the "Review" icons under the selected toilet page.

REQ-2: App must successfully direct the user to the comment page.

REQ-3: App must successfully reflect the newly added comments under the comment section of the selected toilet page.

3.3.4 ReportReview

3.3.4.1 Description and Priority

This feature enables users to report a chosen review which deemed to be inappropriate.

Priority: High

3.3.4.2 Stimulus/Response Sequences

1. The user can click on the "Report Review" link to report any inappropriate reviews in the comment section.
2. The system will pop up a window and prompt the user to confirm if the user wants to report the reviews.
3. The system will display a successful dialogue message.
4. The review will be deleted if the review is reported more than or equal to 3 times.

3.3.4.3 Functional Requirements

REQ-1: App must successfully display the "Report Review" link in the review.

REQ-2: App must successfully increment the report count by 1 if the user confirms to report the review.

REQ-3: App must successfully delete the review under the comment section of the toilet if the report count is more than or equal to 3 times.

3.3.5 ReportNonExistence

3.3.5.1 Description and Priority

This feature enables users to report a chosen toilet that has been discovered it does not exist in the real world.

Priority: High

3.3.5.2 Stimulus/Response Sequences

1. The user can click on the “Report Non-Existent Toilet” link to report the toilet under the toilet page.
2. The system will pop up a window and prompt the user to confirm if the user wants to report the toilet.
3. The system will display a successful dialogue message.
4. The toilet will be deleted if the toilet is reported more than or equal to 3 times.

3.3.5.3 Functional Requirements

REQ-1: App must successfully display the “Report Non-Existent Toilet” link in the toilet page.

REQ-2: App must successfully increment the report count by 1 if the user confirms to report the toilet.

REQ-3: App must successfully delete the toilet if the report count is more than or equal to 3 times.

REQ-3.1: App must successfully delete the toilet from the search result.

REQ-3.2: App must successfully delete the toilet marker from the map.

REQ-3.3: App must successfully delete the toilet card from Nearby Toilet (if necessary).

REQ-3.4: App must successfully delete the toilet from Favourites Toilet (if necessary).

3.3.6 UpdateToilet

3.3.6.1 Description and Priority

This feature enables users to suggest changes and update the details of an existing toilet.

Priority: High

3.3.6.2 Stimulus/Response Sequences

1. The user can click on the “Suggest Changes” button on the toilet page.
2. The system will display the toilet’s information with editable fields: bidet availability, accessibility, sanitiser availability and shower availability with “Confirm Edits” buttons.
3. The user may toggle the button to apply changes.
4. The system will display a successful dialogue message.
5. The system will reflect the changes in the toilet page after the user confirms his/her changes.

3.3.6.3 Functional Requirements

REQ-1: App must successfully display the “Suggest Changes” link in the toilet page.

REQ-2: App must successfully direct the user to edit the toilet’s details.

REQ-3: App must successfully reflect the changes on the toilet page after the user clicks on the “Confirm Edits” button.

3.3.7 UserMenu

3.3.7.1 Description and Priority

This feature enables users to select and perform various activities within the app system.

Priority: High

3.3.7.2 Stimulus/Response Sequences

1. After the user is successfully logged in, the user will be directed to HomeScreen.
2. The user will be prompted to allow permission for location.
3. The system will display a navigation bar at the bottom of HomeScreen to navigate among: HomeScreen(Nearby Toliets), SearchToilet, AddToilet and FavouritesToilet.
4. There is a profile icon at the top right of the HomeScreen to navigate to UserProfile.
5. The user can select an option now.
    a. User selects HomeScreen.
        i. The system will invoke the NearbyToilet use case, where the top 5 nearest toilets are displayed.
    b. User selects SearchToilet.
        i. The system will invoke the SearchToilet use case, where the user can search for the toilet and look up the toilet's location on the map displayed.
    c. User selects AddToilet.
        i. The system will invoke the AddToilet use case, where the user can create a new toilet discovered by the user..
    d. User selects FavouritesToilet.
        i. The system will invoke the FavouritesToilet use case, where the user can look up their favourite toilets.
    e. User selects UserProfile.
        i. The system will invoke the UserProfile use case, where the user can look up their user information.

3.3.7.3 Functional Requirements

REQ-1: App must display the navigation bar for the user to select from the following available activities: HomeScreen(Nearby Toilets), SearchToilet, AddToilet and FavouritesToilet.

REQ-1.1: App must integrate NearbyToilet, SearchToilet, AddToilet and FavouritesToilet features.

REQ-1.2: App must navigate the user to the activity page based on their selected activity.

3.3.8 ViewToiletOnMap

3.3.8.1 Description and Priority

This feature enables the system to display the nearby toilet on the map based on the input location or the user's current location.

Priority: High

3.3.8.2 Stimulus/Response Sequences

1. The system will retrieve the user's current location or the input location.
2. The system will relocate the map position by placing the location at the centre.
3. The user can select any toilet on the map and the system will invoke the ToiletMenu use case.

3.3.8.3 Functional Requirements

REQ-1: App must successfully display the map and relocate the position based on the user's location or the input location in the search bar.
REQ-2: App must successfully display all the toilet markers near the location.
REQ-3: App must successfully display the respective toilet card and its toilet menu.
REQ-4: App must integrate the ToiletMenu feature.

3.3.9 AddToilet

3.3.9.1 Description and Priority

This feature enables the user to add a new toilet discovered by the user.

Priority: High

3.3.9.2 Stimulus/Response Sequences

1. The system will display the Add New Toilet page.
2. The system allows the user to select the location of the toilet using Apple Maps and drop a marker.
3. The system will prompt the user to enter the toilet's details, such as name, rating and amenities availability.
4. The system will save the new toilet data.
5. The system will confirm that the toilet has been added successfully.

3.3.9.3 Functional Requirements

REQ-1: App must successfully display the map and allow the user to drop the marker at the location intended.

REQ-2: App must prompt the users to enter the required information.

REQ-2.1: App must display the input field for "Full Name".

REQ-2.2: App must display an error message if the name field has no input.

REQ-3: App must display the successful dialogue message to confirm the new toilet has been added.

REQ-3.1: New toilet must be added to the database.

REQ-3.2: New toilet must have a marker on the map.

REQ-3.3: New toilet must appear in the search result.

REQ-3.4: New toilet must be displayed in the Nearby Toilet if it is near the user's current location.

3.3.10 FavouritesToilet

3.3.10.1 Description and Priority

This feature enables the system to display a favourite list of toilets of the user.

Priority: High

3.3.10.2 Stimulus/Response Sequences

1. The system will retrieve the user's favourite toilets from the database.

2. The system will display the user's favourite toilet list.
3. The user can select any toilet on the favourite list and the system will invoke the ToiletMenu use case.
4. If the user toggles the "Favourites" icon, the toilet will be removed from the list.

3.3.10.3 Functional Requirements

REQ-1: App must successfully display the user's favourite toilet list.

REQ-2: App must successfully remove the toilet from the list if the user un-favourite the toilet.

3.3.11 UserProfile

3.3.11.1 Description and Priority

This feature enables the system to display user information, that is username and email.

Priority: High

3.3.11.2 Stimulus/Response Sequences

1. The system will display the user information at the UserProfile page, that is his/her username and email.
2. If the user choose to logout, user can click on the "Log Out" button.
    a. The system will pop-up a window to confirm if the user want to log out from his/her account.
    b. If user choose to log out, the user can click on the "Confirm" button and the system will direct the user to the LoginPage.
    c. If user choose to cancel, the user can click on the "Cancel" button and the system will halt the logout process to remain at the current page.

3.3.11.3 Functional Requirements

REQ-1: App must successfully display the user's information details.

REQ-2: App must successfully direct the user to the LoginPage if the user choose to log out.

REQ-3: App must successfully remain at the current page if the user choose to log out.

3.3.12 SearchToilet

3.3.12.1 Description and Priority

This feature enables the user to search for the toilets in any location.

Priority: High

3.3.12.2 Stimulus/Response Sequences

1. User can enter the toilet location he/she want to search for.
2. If the toilet exists, the search result will display the toilet.
3. If the user clicks on it, the system wll invoke ViewToiletsOnMap use case and display the toilet location on the map.
4. The system will display a summary details of the toilet for user.

3.3.12.3 Functional Requirements

REQ-1: App must successfully display the search result (if the toilet exists).

REQ-2: App must successfully display the location marker on the toilet that user is searching.

REQ-3: App must successfully display the summary details of the toilet card.

3.3.13 NearbyToilet

3.3.13.1 Description and Priority

This feature enables the system to display the the informations of the nearest 5 toilets around the user's current location in the card format.

Priority: High

3.3.13.2 Stimulus/Response Sequences

1. The system will ask for user's location permission.
2. The system will find theuser's nearest 5 toilets.
3. The system will display the information in the card format at HomeScreen (Nearby Toilet).
4. The user can swipe the toilet card to look for their preferred toilet near them.
5. The user can select any toilet card and the system will invoke the ToiletMenu use case, direct user to the toilet page.
6. If the user click on the "Navigate" button, the system will direct the user to the SearchToilet page and invoke the ViewToiletsOnMap use case.

3.3.13.3 Functional Requirements

REQ-1: App must successfully display the nearest 5 toilets in the card format.

REQ-2: App must successfully allow the user to swipe the card to the left/right.

4. External Interface Requirements

4.1 User Interfaces

4.1.1 Style Guides

4.1.1.1 Theme Colours

Neutral

#ffffff
White

Primary

#448aff
≈ Dodger Blue

Secondary

#fef7ff
≈ White Pointer

#f7f2fa
≈ White Lilac

#f2f2f7
≈ Athens Gray

Tertiary

#ffc106
≈ Amber

Link

#38843a
≈ Goblin

Dark

#000000
Black

#4c4c4c
≈ Tundora

#bdbdbd
≈ Silver

#dadcdf
≈ Iron

#eeeeee
≈ Gallery

4.1.1.2 Typography

Mazzard H

- Text (XS) - 12px

Mazzard H

- Text (Small) - 16px

Mazzard H

- Text (Medium) - 18px

Mazzard H

- Text (Large) - 24px

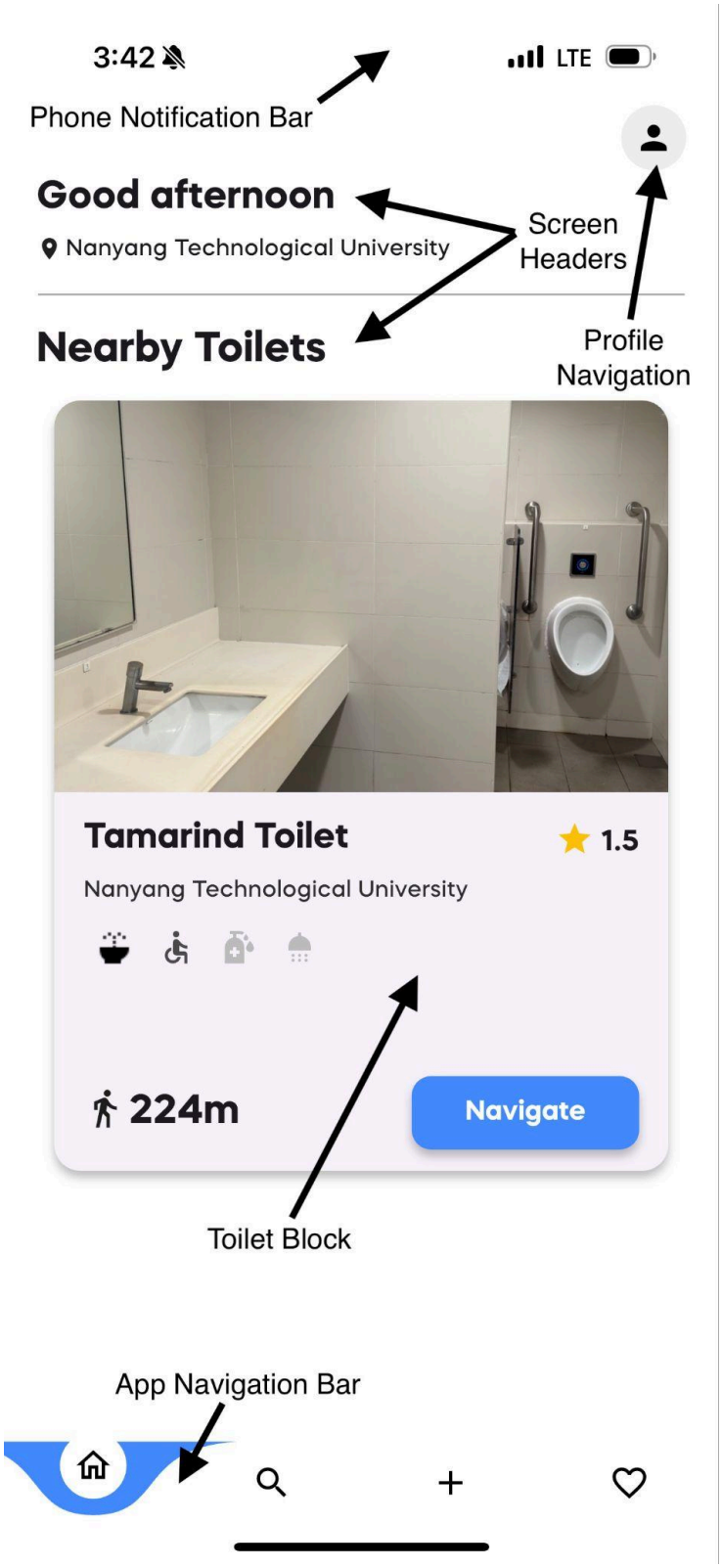
Mazzard H

- SubHeading - 36px

Mazzard H

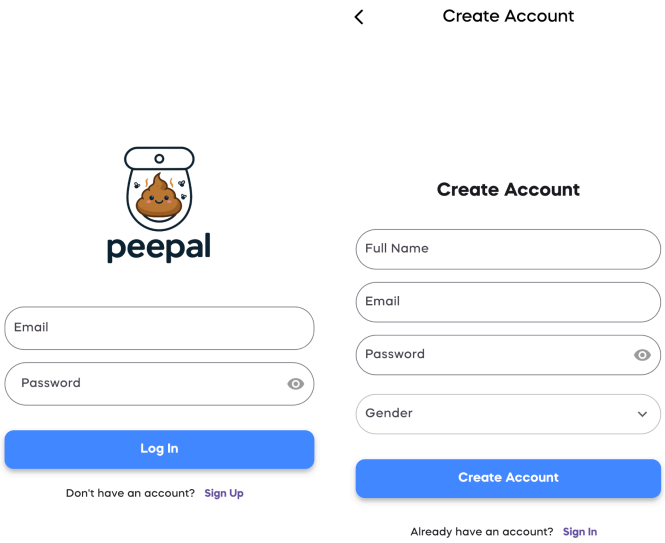
- Heading - 48px

4.1.1.3 App Screen Template

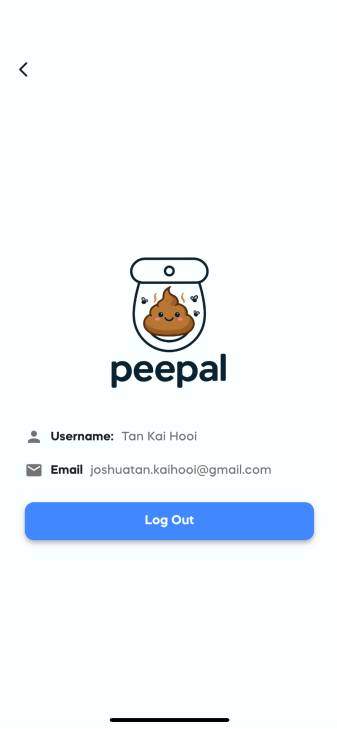


4.1.2 UI Mockups

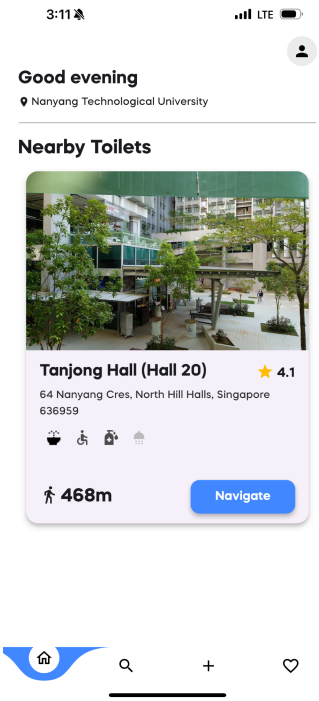
4.1.2.1 Authentication (Login + Sign-up)



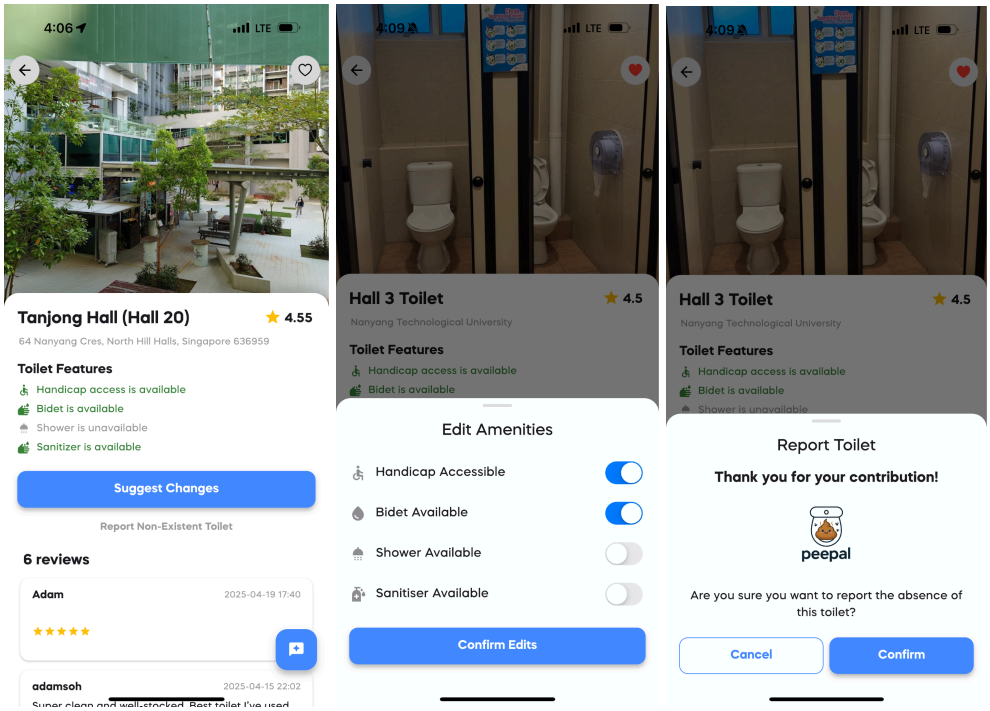
4.1.2.2 User Profile



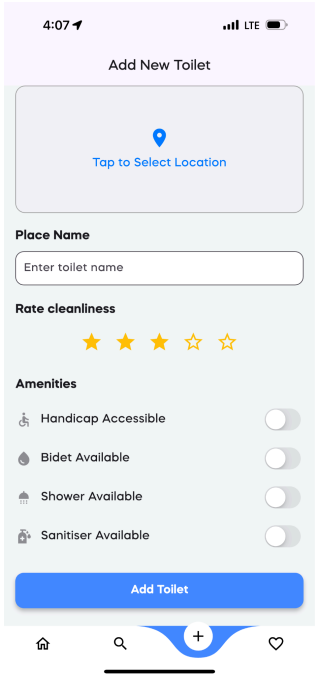
4.1.2.3 Home Screen



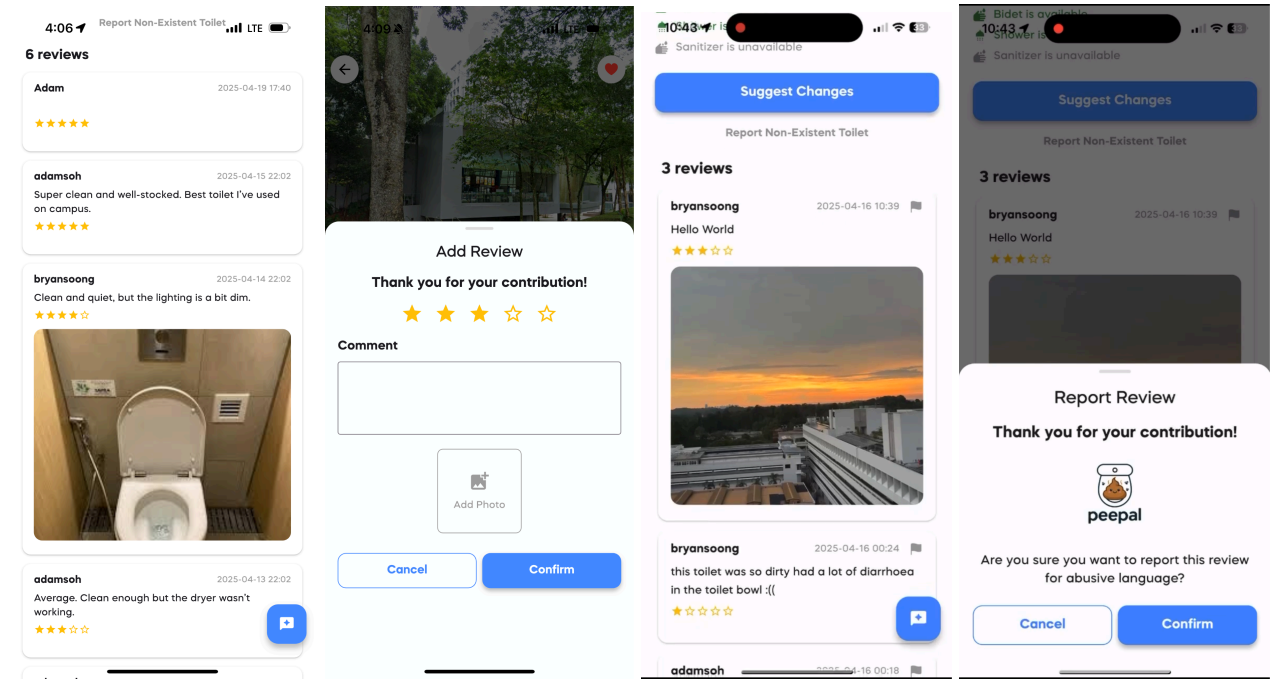
4.1.2.4 Toilet Page (Details + Suggest Changes + Report Non-Existence)



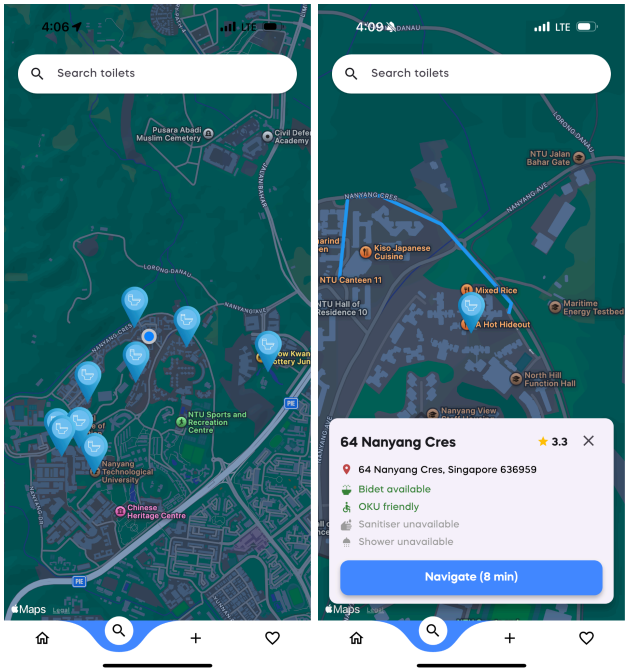
4.1.2.4.1 Add New Toilet



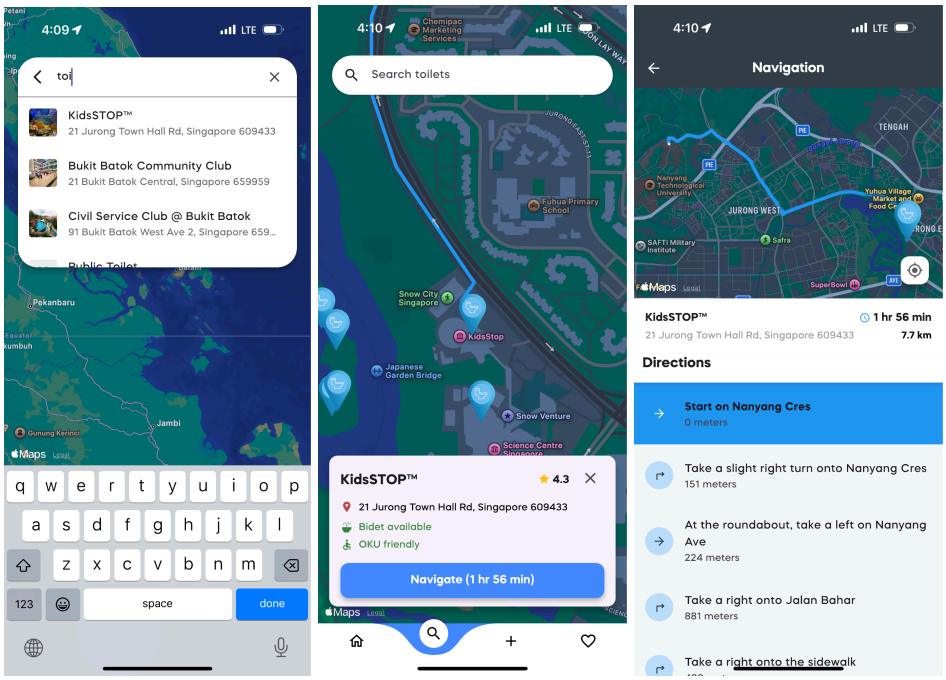
4.1.2.5 Review System (Review + Report Review)



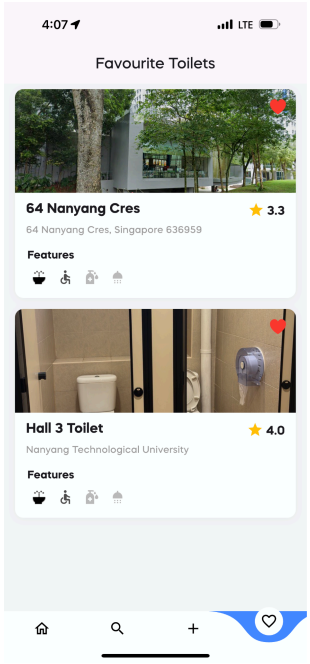
4.1.2.6 Map Functionality



4.1.2.7 Navigation System (Search + Navigate)

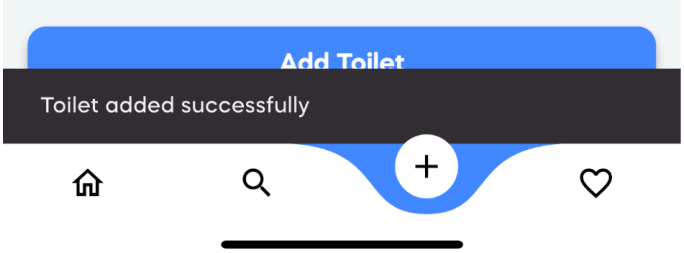


4.1.2.8 Favourites



4.1.2 Success/Error Message Display (Notification)

The notification of a success/error message will be used whenever a success/error event is triggered. For example, on a successful creation of a new toilet (Fig. 4.1.2.a), or a failed login attempt (Fig. 4.1.2.b).



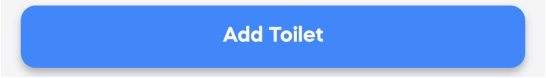
4.1.2.a: Success Message

4.1.2.b: Error Message

4.1.3 Common Components

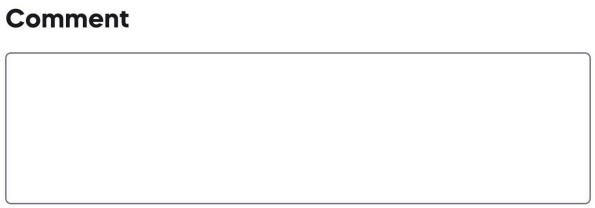
4.1.3.1 Form Inputs

Form Text Input



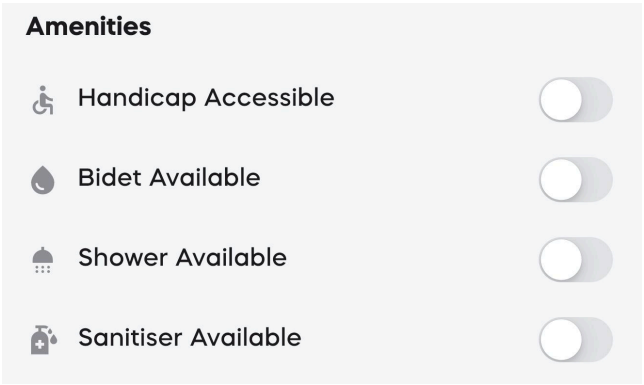
Form Submit Button

Form Image Upload Input



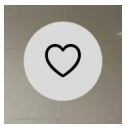
Form Text Area Input

Form Rating Picker Input



Form Option Picker Input

4.1.3.2 Miscellaneous



Favourites Button

Screen Header

App Navigation Bar

4.2 Hardware Interfaces

The software system is primarily designed as a mobile application, requiring minimal direct hardware interfacing beyond standard device capabilities. Below are the logical and physical characteristics of the hardware interfaces relevant to the software:

4.2.1 Device Types

The application is intended to be compatible with a range of devices, including:

- Tablets (iOS, Android)
- Smartphones (iOS, Android)

4.2.2 Data and Control Interactions

User interaction with the software is facilitated through standard hardware interfaces:

- Input Devices:

    - Virtual on-screen keyboard
    - Touchscreens (tap, swipe, pinch)
    - GPS/location sensor for location-based features
- Output Devices:
    - Displays (various resolutions and aspect ratios supported)
- Network Interfaces:
    - Wi-Fi and cellular data for internet connectivity

4.2.2 Communication Protocols

The system utilises standard communication protocols for data exchange and device interaction:

- HTTPS: All web and API traffic is secured via HTTPS (port 443).
- HTTP: Non-secure communication (port 80) is redirected to HTTPS where possible.
- RESTful API: All client-server communication follows REST principles over HTTP/HTTPS.
- Platform Services: Mobile apps may interface with device services (e.g., location, notifications) via standard OS APIs.

4.3 Software Interfaces

The application interfaces with several software components and platforms to ensure broad compatibility and maintainability.

4.3.1 Mobile Platforms

- Mobile application is compatible with iOS (latest and recent versions) and Android (API level 21+), built using Flutter.

4.3.2 Databases

- SQL-based database systems, PostgreSQL with PostGIS extension, is accessed via backend ORM or direct queries.

- Data models are defined in the backend and mapped to database tables.

4.3.3 Backend Tools

- Node.js (version 18.x or higher) for server-side runtime.
- TypeScript (version 4.9.5 or higher) for static typing and maintainable code.
- Hono web framework for API routing and middleware.
- ESLint and Prettier for code formatting and linting.
- Vitest for backend testing.

4.3.4 Frontend Tools

- Flutter (latest stable version) for cross-platform mobile application development.
- Dart (latest stable version) as the programming language for Flutter.
- Material Design and Cupertino widgets for native look and feel.

4.3.5 RESTful APIs

- RESTful API endpoints exposed by the backend for all client-server communication.
- API documentation provided via OpenAPI/Swagger or similar tools.
- Endpoints support standard HTTP methods (GET, POST, PUT, DELETE) and return appropriate status codes.

4.3.6 Data Sharing

- All data exchanged between frontend, backend and database is formatted as JSON.
- Data persistence is managed through the SQL database using defined data models.
- Authentication tokens (JWT) are used for secure data sharing between client and server.

4.4 Communications Interfaces

The system leverages several communication interfaces to ensure secure, efficient and responsive interactions between users, the mobile client and backend services:

- HTTP/HTTPS:
    - All communication between the client mobile application and backend services occurs over HTTP/HTTPS protocols. HTTPS is enforced to ensure data confidentiality and integrity during transmission.
- API Communication:
    - The primary mode of communication between frontend and backend is through RESTful APIs.
    - All API requests and responses utilize the JSON format for data interchange.
    - API documentation and testing are provided via OpenAPI/Swagger UI (for Node.js/Express).
- Electronic Forms:
    - User input (such as registration, login, reviews and feedback) is submitted through secure HTTPS forms in the mobile application, ensuring that user data is protected during transmission.
- Network Protocols:
    - All network communications are based on standard internet protocols, including TCP/IP, to guarantee compatibility and reliability across diverse network environments.
- Security and Encryption:
    - TLS (Transport Layer Security) is enforced for all HTTP-based communications (including REST APIs and WebSockets).
    - Authentication and authorization are managed using secure JWT tokens over HTTPS.
- Message Formatting:
    - All API responses follow standardized JSON schemas for consistency and ease of parsing.
    - Error and status messages adhere to a uniform structure, facilitating robust client-side error handling.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1) Each page should load in less than 5 seconds, including logging in and authentication.
2) The mobile application should be scalable enough to support at least 250,000 users at the same time while maintaining optimal performance.
3) Errors that could arise from synchronisation issues, like two or more users suggesting a change to the same toilet, should be kept to a maximum of no more than 100 times per month.
4) The total error rate of users adding toilets, deleting toilets, leaving reviews, reporting reviews and suggesting changes to toilets should be kept below 2 per cent.

5.2 Safety Requirements

1) The application will not require the sharing of any personal information, such as location data between users to ensure the safety of all users.

5.3 Security Requirements

1) User authentication will always be necessary before using other features of the app.
2) Backend API services should not be accessed by unauthenticated users.
3) The systems should implement password hashing for storing the users' passwords to protect users' accounts.
4) There should be no tolerance for data leaks. The database should be stored securely by limiting access to the database to key developers.
5) The users' personal information shall be kept confidential and shall not be disclosed to other users of the application or the public, in line with the PDPA.

5.4 Software Quality Attributes

1) The code will have relevant code comments indicating the purpose of the following block of code, to allow for interpretability, code reusability and future maintenance.

2) The API endpoints in the backend server will have documentation to indicate the endpoint url, required request body and expected response body.
3) The application should only crash or fail to start at most 1% of the time.

6. Other Requirements

6.1 Internationalisation Requirements

1) Future releases of PeePal will consider multi-language support such as Chinese, Malay, Japanese and many more to cater to a broader audience in the world.
2) Data and time representations should be in Singapore timezone (GMT +8).

6.2 Legal Requirements

1) Compliance with PDPA when handling users' sensitive data.

6.3 Reuse Objectives

1) Components developed for PeePal should be modular and reusable within the project and also for potential future projects.
2) APIs should be designed with industry standards to facilitate external usage of our platform through APIs.

Appendix A: Glossary

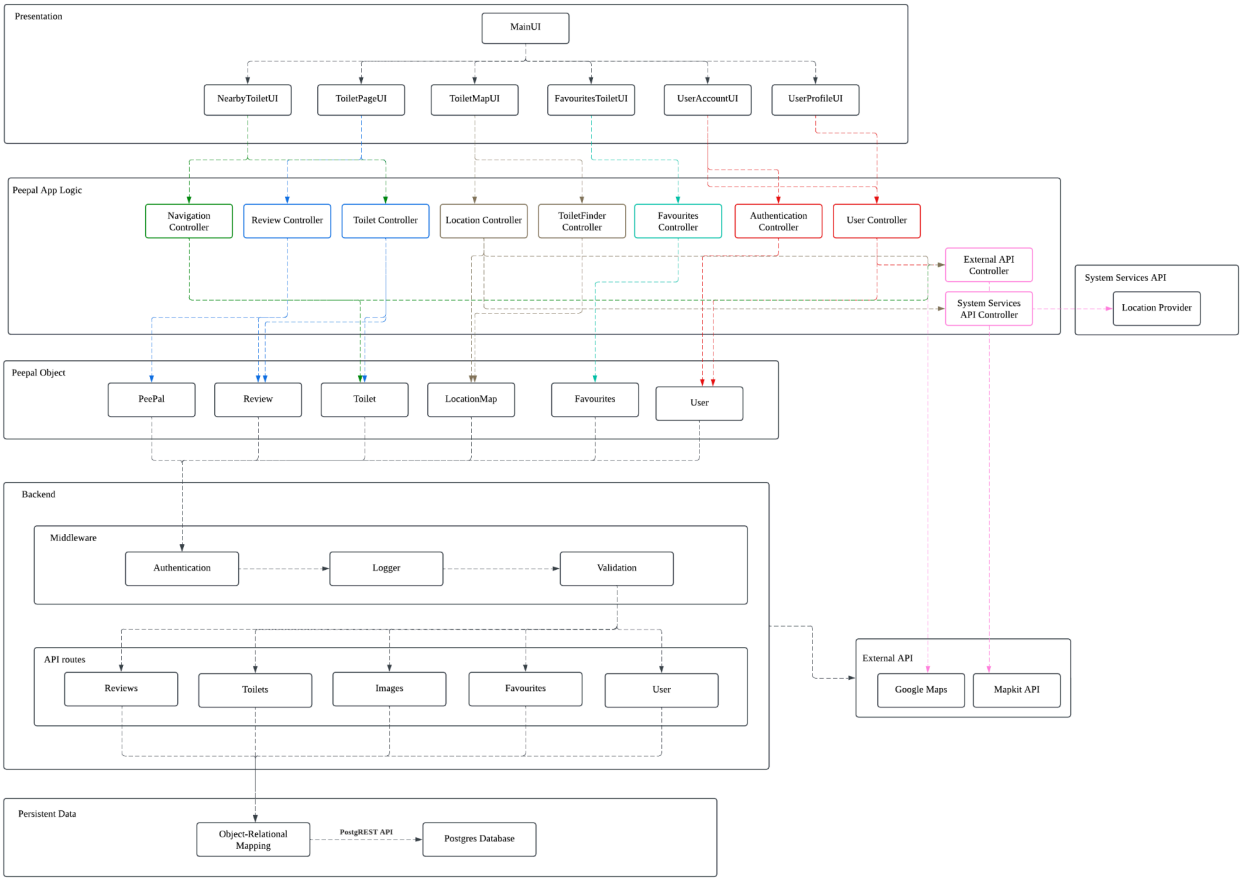
Term	*Definition*
API	Application Programming Interface. A set of rules and specifications that software programs can follow to communicate with each other.
Backend	The server-side portion of the application that handles data storage, processing, and logic.
BLoC	Business Logic Component. A design pattern used for state management, especially in Flutter applications.
Cupertino	A set of widgets for Flutter applications that mimic the look and feel of Apple's iOS design language.
Dart	The programming language used to develop Flutter applications.
ESLint	A tool used for linting TypeScript code to maintain code quality and consistency.
Flutter	A UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.
Frontend	The client-side portion of the application that the user interacts with directly.
GMT+8	Singapore Time Zone, which is 8 hours ahead of Greenwich Mean Time.
Hono	A lightweight web framework for building backend servers, used for developing REST

	APIs.
HTTP	Hypertext Transfer Protocol. An application layer protocol for transmitting hypermedia documents, such as HTML.
HTTPS	Hypertext Transfer Protocol Secure. A secure version of HTTP that uses encryption for secure communication.
iOS	Apple's mobile operating system.
JSON	JavaScript Object Notation. A lightweight data-interchange format.
JWT	JSON Web Token. A standard for securely representing claims between two parties.
MapKit API	A framework provided by Apple for integrating maps into iOS applications.
Material Design	Google's design system for building bold, digital experiences.
MinIO	An object storage server compatible with Amazon S3 cloud storage service.
MySQL	A relational database management system.
Node.js	A JavaScript runtime built on Chrome's V8 JavaScript engine.
OKU-friendly	Facilities that are accessible to people with disabilities (Orang Kurang Upaya is Malay term).
OOP	Object-Oriented Programming.

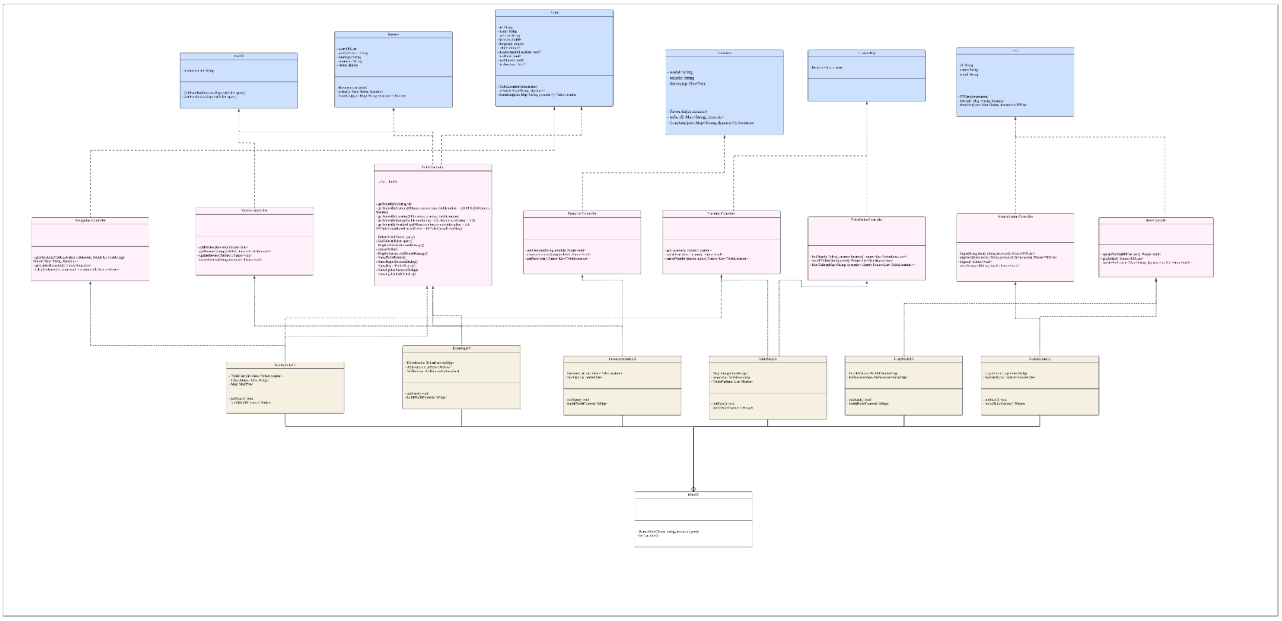
OpenAPI	A specification for machine-readable interface definitions for describing, producing, consuming, and visualizing RESTful web services.
ORM	Object-Relational Mapping. A technique for converting data between incompatible type systems using object-oriented programming languages.
PDPA	Personal Data Protection Act.
PostgreSQL	An open-source relational database management system.
PostGIS	An extension that extends the capabilities of the PostgreSQL relational database by adding support for storing, indexing, and querying geospatial data
Prettier	A code formatter that supports various programming languages.
REST API	Representational State Transfer Application Programming Interface. An architectural style for designing networked applications.
README	A standard text file introducing and explaining a project.
SRS	Software Requirements Specification. A document that describes what the software will do and how it will be expected to perform.
SQL	Structured Query Language. A domain-specific language used in programming and designed for managing data held in a relational database

	management system.
Swagger	A set of open-source tools for designing, building, documenting, and consuming RESTful web services.
TCP/IP	Transmission Control Protocol/Internet Protocol. The fundamental communication protocols that underpin the Internet.
TLS	Transport Layer Security. A cryptographic protocol designed to provide communications security over a computer network.
TypeScript	A strongly typed programming language that builds on JavaScript, giving you better tooling at any scale.
UI	User Interface. The means by which the user and a computer system interact.
Vitest	A blazing fast unit test framework powered by Vite.
WebSocket	A communications protocol providing full-duplex communication channels over a single TCP connection.
WSS	WebSocket Secure. The secure version of the WebSocket protocol.
Zod	A TypeScript-first schema declaration and validation library.

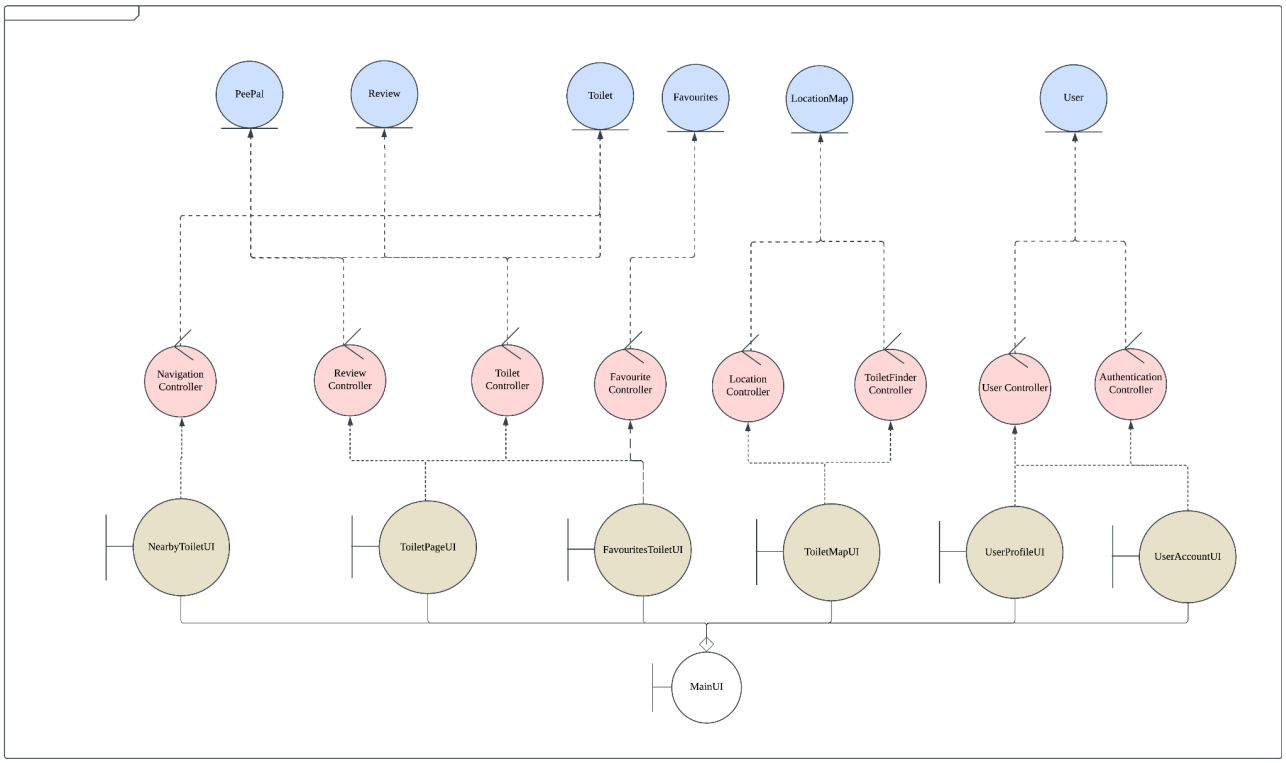
Appendix B: Analysis Models



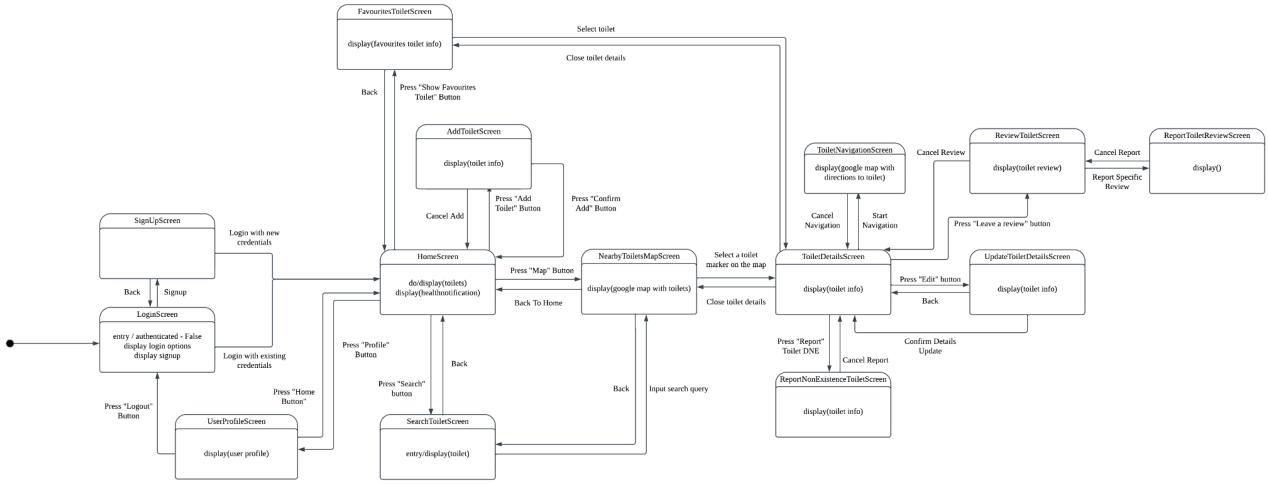
Architecture Diagram



Class Diagram



Stereotype Class Boundary Diagram


FavouritesToiletScreen
display(favourites toilet info)
Select toilet
Close toilet details
Back
Press "Show Favourites Toilet" Button
AddToiletScreen
display(toilet info)
Cancel Add
Press "Add Toilet" Button
Press "Confirm Add" Button
ToiletNavigationScreen
display(google map with directions to toilet)
Cancel Navigation
Start Navigation
ReviewToiletScreen
display(toilet review)
Cancel Review
Press "Leave a review" button
ReportToiletReviewScreen
display()
Cancel Report
Report Specific Review
SignUpScreen
Login with new credentials
Back
Signup
LoginScreen
entry / authenticated - False
display login options
display signup
Login with existing credentials
HomeScreen
do/display(toilets)
display(healthnotification)
Press "Map" Button
Back To Home
NearbyToiletsMapScreen
display(google map with toilets)
Select a toilet marker on the map
Close toilet details
ToiletDetailsScreen
display(toilet info)
Press "Edit" button
Back
UpdateToiletDetailsScreen
display(toilet info)
Confirm Details Update
Press "Report" Toilet DNE
Cancel Report
ReportNonExistenceToiletScreen
display(toilet info)
Press "Profile" Button
Press "Home Button"
Press "Logout" Button
UserProfileScreen
display(user profile)
Press "Search" button
Back
SearchToiletScreen
entry/display(toilet)
Back
Input search query

Dialog Map

Appendix C: To Be Determined List

There are no TBD items for this version of Software Requirement Specification.