Flixtyle

TEAM #3 FLIXTYLE

DESIGN

SPECIFICATION

Team 3

May 19th, 2019

Flixtyle

Software Design Specification

Introduction to Software Engineering-41

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Initial	05/19/2019	Initial version	0.1

1. Preface

1.1 Objective

The purpose of this document is to outline the system design and structure specifications of the “Flixtyle” application. The document includes architecture of each system using UML, API design, Database Design and Testing plan. By reading this document, reader can understand how to work the system and relationship of its components.

1.2 Readership

This document is written for project manager, project team and development team. Also, this document is for the end users and other various stakeholders who is involved in the support, maintenance of the system. In other words, the readers of this document are all members of the development and maintenance teams of the “Flixtyle” application which introduced in this document. It is recommended to know most of the terms in the glossary section before reading through the document.

1.3 Document Structure

1.3.1 Preface

The ‘Preface’ defines the reader of this document and introduces the structure of the document. Moreover, it describe the purpose and outline of each table of contents when introducing structures.

1.3.2 Introduction

The ‘Introduction’ defines describes the diagrams and tools used in this document. Furthermore, it describes the scopes of the “Flixtyle” application which introduced in this document.

1.3.3 System Architecture

The ‘System Architecture’ defines general description of the systems that our team wants to develop. In addition, the overall structure of each system is represented as a block diagram. Each relationship and how it is actually used is described as a Package Diagram and Deployment Diagram.

1.3.4 Hearts System

The ‘Hearts System’ describes how user can manage their Heart list. Users can heart their favorite items through the discovery and search process. These favorite items are stored in the user database and user can be checked in the user personal account by the account system. Moreover, Users can manage items from the heart list by adding or deleting them. The document expresses and describes of the Hearts System structure by using Class Diagram, Sequence Diagram, and State Diagram.

1.3.5 Discovery System

The ‘Discovery System’ describes how to get user preference. Since the recommendation system is a customized service according to the user's preference, the system must have to investigate user's preference data by Discovery System. User can represent their preferred items through adding or not adding to likes. These items are stored in the recommendation database and used to create the recommendation item list. Additionally, the Heart system allows users to store their favorite items in the heart list. The document expresses and describes of the Discovery System structure by using Class Diagram, Sequence Diagram, and State Diagram.

1.3.6 Recommendation System

The “Recommendation System” describes the system that recommends items to user as a list. Recommendation system makes recommendation item list by using Discovery System to select the preferred item for users and saving these items in Recommendation database. Items of recommendation depends on the actual data handed by the user. The document expresses and describes of the Recommended System structure by using Class Diagram, Sequence Diagram, and State Diagram.

1.3.7 Classification System

The “Classification System” describes how the system classify items for user and system. Classification system sort categories according to the type of items and store them in the database. This classification system not only makes programs easy to manage items, but also users can efficiently search items. The document expresses and describes of the Classification System structure by using Class Diagram, Sequence Diagram, and State Diagram.

1.3.8 Account Management System

The “Account Management System” describes how to manage user account. It is about how the user can either sign up or log in using User database Account Management System compares the user values to the Account database, and user can enter to user account if it matches. If user don't have an account, user can sign up and Account database save the user profile that system need by default. Additionally, the Account Management System allows users to freely change user profiles or check user’s Heart list. The document expresses and describes of the Account Management System structure by using Class Diagram, Sequence Diagram, and State Diagram.

1.3.9 API

The “API” describes the interfaces between the different systems and components. The interfaces are going to be implemented following the RESTful API standard. RESTful is a stateless API that uses JSON to communicate. Javascript Object Notation(JSON) is a human readable text format that a standard in the industry. JSON can be parsed and edited by many programming languages out of the box without needing to add any external libraries.

1.3.10 Database

The “Database” definess database diagram that the project will be based on. The designed database is represented by the ER diagram, and the data of the program is structured and stored through the created ER diagram.

1.3.11 Testing plan

The “Testing plan” describes the Test Policy and Test Case. The purpose of this chapter is establishes a preliminary plan for system testing to determine if the system is operating as intended to identify and analyze the system for defects after completion of the system. This part is divided into testing policy and Testing case. The Testing Policy sets the criteria for testing and The test case gives a set of more detailed input and output.

1.3.12 Development plan

The “Development plan” describes the programming language and the IDE that is going to be used. Also, it explains the version of programming languages and management tool.

1.4 Glossary

★ Seller
The affiliates to the system who sells products on the Flixtyle service.

★ User
Customers who use Flixtyle services to purchase products.

★ Fashion item
Wearable items. Limited to clothing, hats, swimming suits

★ Product
A fashion item the seller has registered to the system

★ Recommendations
Products which are shown to relevant customers and acts as an advertisement to the user

★ Discovery
The process of verifying whether the user is actually interested in the products the system predicts.

★ Save
The act of adding a product to the “Heart” List, so that the user will be able to quickly find the product in the future.

★ “Heart” List
The list of the products that a user has saved.

★ Swipe
The act of swiping across the touchscreen.

★ Authentication
The process of verifying the user’s identity.

★ Request to the server
A message sent with the HTTP protocol to the server with information from client.

★ Filtering
The process of separating unwanted items from a list of items, where these unwanted items are decided from the options.

★ SNS
Social networking service, for example Facebook, Twitter.

2. Introduction

2.1 Objective

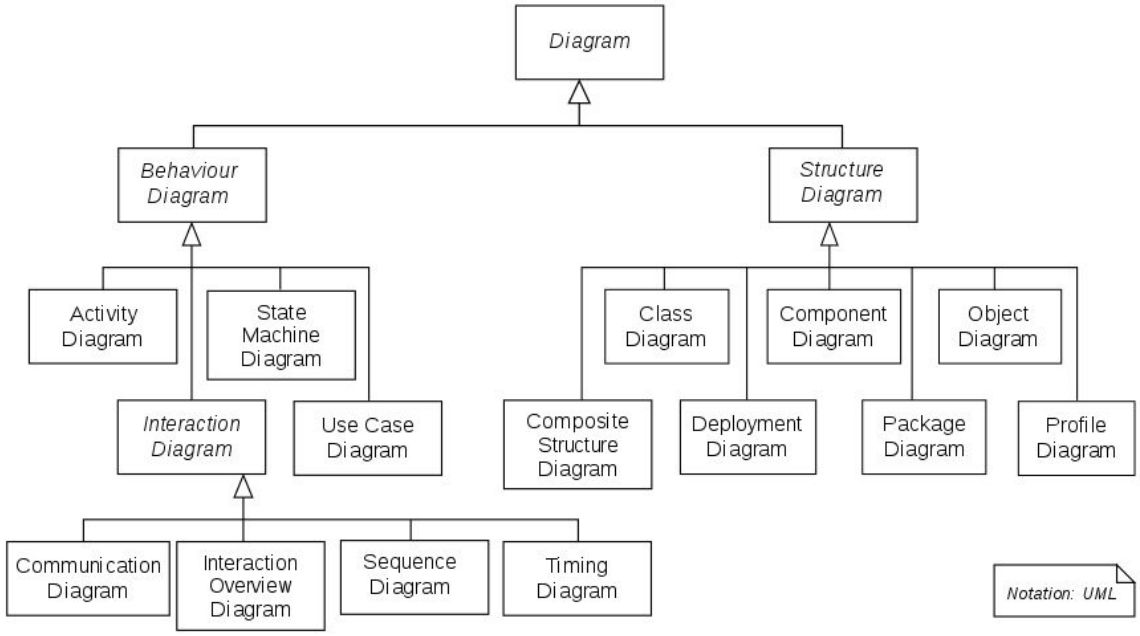
This chapter will explain the diagrams used in this document, state the applied tools and outline the project scope.

2.2 Applied diagrams

A. UML

This document will feature several diagrams from the Unified Modelling Language (UML). UML is a general-purpose modelling language made for standardizing the visualization of a software system. The modelling language was developed in 1994-1995, by Ivar Jacoson, Grady Booch and James Rumbaugh and has evolved steadily since.

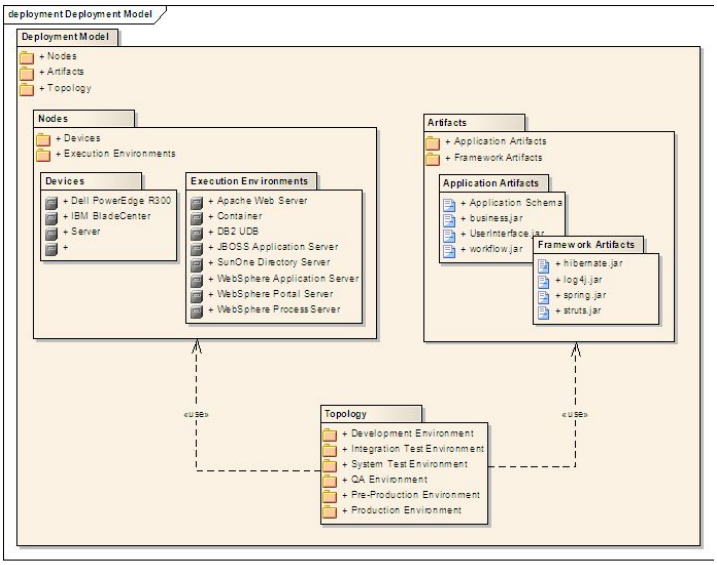
One of the main reasons behind UML's success, is how the language offers intuitive ways to visualize a software system's architecture, including system elements like activities, components and their interactions. UML can be divided into two different views, static and dynamic. The static (or structural) view describes the static structure of the system using objects, attributes and relationships. Class, deployment and package diagrams belong to this view. The dynamic view, on the other hand, emphasizes the dynamic behaviour of the system. It does this by describing interactions among objects and how these changes the internal states of the various objects. Important diagrams belonging to this view are state and sequence diagrams. A general overview of the various UML diagrams is provided below.



B. Package diagram

Package diagrams aim to depict the dependencies between the packages that make up a model. A package is a general-purpose element for organizing model elements and diagrams into groups. Elements can be classes, interfaces, objects, tables or other packages. Generally, these models aim to describe the relationships between major elements in the system (as opposed to component diagrams). Package diagrams may also include use cases.

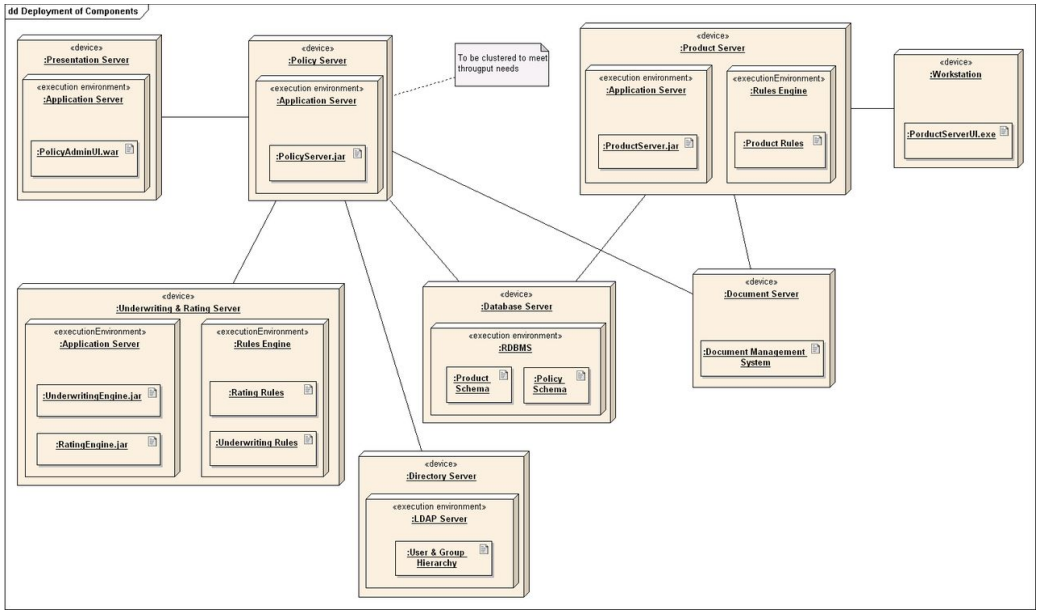
An example of a package diagram can be seen below.



C. Deployment diagram

Deployment diagrams aim to model the physical of artefacts (software components) on nodes (hardware components) in the system. The purpose of these diagrams is to provide stakeholders with an overview of how the system will be implemented physically. This can help engineers determine what hardware components they need, what criteria these components should fulfil and the dependencies between the hardware components. This is very valuable to identify potential fault propagation.

Deployment diagrams consist of nodes, shown as boxes. These nodes represent physical hardware devices or a virtual execution environment. Each node has a number of artifacts (elements), describing the functions, classes etc. that will run in that particular device or environment. An example of a deployment diagram can be seen below.

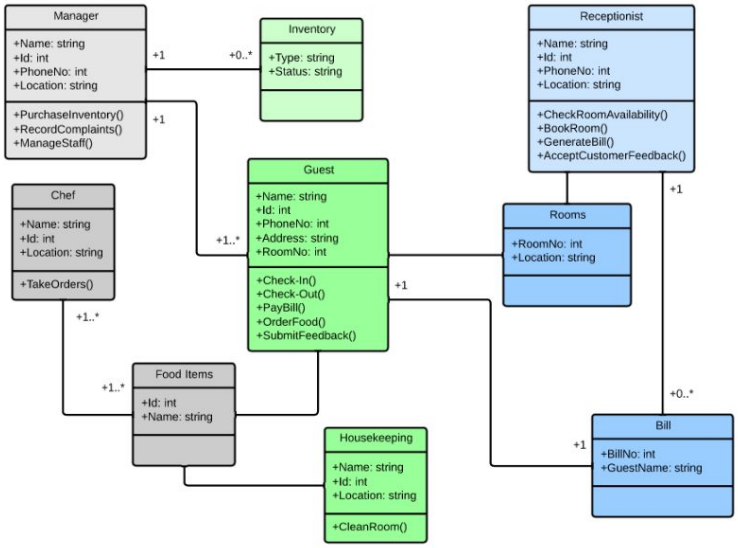


D. Class diagram

Class diagrams are structural diagrams that aim to describe the classes of a software system, their attributes (i.e. variables, functions) and the relationships among objects. Class diagrams are one of the most detailed UML diagrams and provide a relatively easy conversion from modelling to actual code.

The system classes are represented as boxes divided into three compartments. The top compartment simply contains the name of the class. The middle contains variables in a form that describes their names and type. The bottom compartment contains the class' operations (functions), these are represented with an intuitive name and input parameters. This way, if the class, variables and functions are given intuitive names, one can easily identify and understand the role a given class will play in the system.

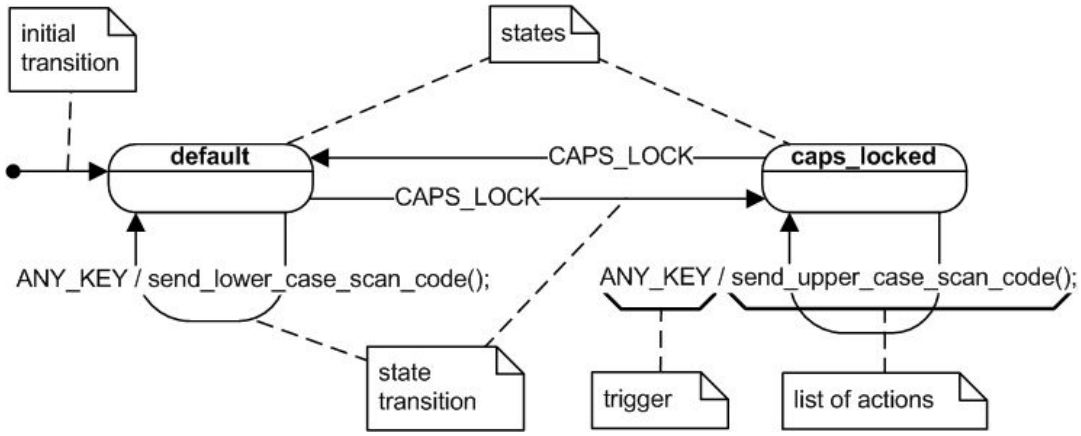
Furthermore, these classes are then grouped together in a class diagram that describes the static relationship between them. The class-boxes are connected with lines, indicating which external classes they interact with and in what fashion (one-to-one, one-to many etc.). An example of a class diagram can be seen below.



E. State diagram

The UML state diagrams are dynamic diagrams that aim to illustrate the various states of a software system and what events will lead to a specific state. State diagrams are built up by nodes denoting states and edges (connectors) denoting state transitions. The states are represented as rounded rectangles labeled with the relevant state name. The state transitions are represented as arrows, labeled with the triggering event. Note that a state diagram should be cyclic, so that the system won't be stuck in a state that it's impossible to transition from. These diagrams provide a graphical abstraction that may prevent programming errors, as event-based state transition often lead to.

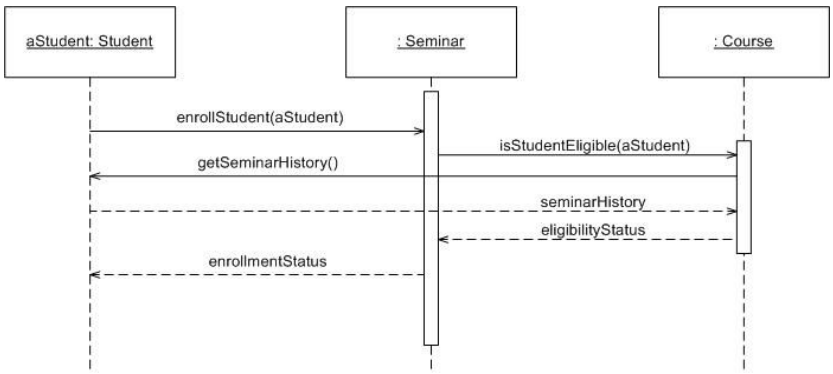
Below, an example of a state diagram showing the states of a keyboard is shown.



F. Sequence diagram

Sequence diagrams are dynamic diagrams showing object interactions, sequenced by time in a top-to-bottom fashion. This provides a graphical illustration of how the messages (queries) flow between the objects when an event happens.

Parallel vertical lines (called lifelines) illustrates objects and horizontal arrows illustrates the messages being transmitted between them, in the order that they occur. When an event occurs, a query may be asked from the leftmost lifeline (object) to the lifeline to its right. This process continues rightwards until the response to the query has been found. This could be updating a table in the database or a request for information. Thereafter, the object producing the response will send the information to the left, back to the original querying object. This message can contain a simple boolean specifying that a table was updated successfully or specific information. Below, a sequence diagram describing a call for enrolling a student in a class is shown.



2.3 Applied tools

We will use Google's Firebase for our back-end database. Firebase is an easy-to-use mobile and web application database, built on Google infrastructure. This allows applications to scale

automatically, in addition to provide lots of support and analytic tools. More about this in chapter 10.

Android Studio will be used for the main app development. This is an Java-based IDE for developing android applications, made by JetBrains. It supports several useful features like integrated Android environments for testing, drag-and-drop features for UI-design and Gradle-based build support.

We will use the Node.js platform to provide a framework for our application. Node.js is an event-based asynchronous I/O framework built on Google's V8 JavaScript Engine. This way, we can make use of running JavaScript on both the client and server side and that way benefit from the lack of context switching, in addition to non-blocking and event-driven queries. Our server only needs to perform GET or POST calls when an event occurs.

To provide concurrent documentation of the development process, we have mainly used Google Docs. This text-editor allows all the team members to work on a single document simultaneously.

In addition, Microsoft's PowerPoint will be used for presenting our system. PowerPoint is a very easy-to-use software for creating presentation slides and graphs.

2.4 Project scope

The system mainly consists of two parts: backend database server and the frontend graphical user interface (GUI). The backend runs algorithms to classify users based on their likes and recommend liked items from other users of the same group. The application's GUI will consist of four main pages; the login page, the account page, the discovery page and the recommendations page.

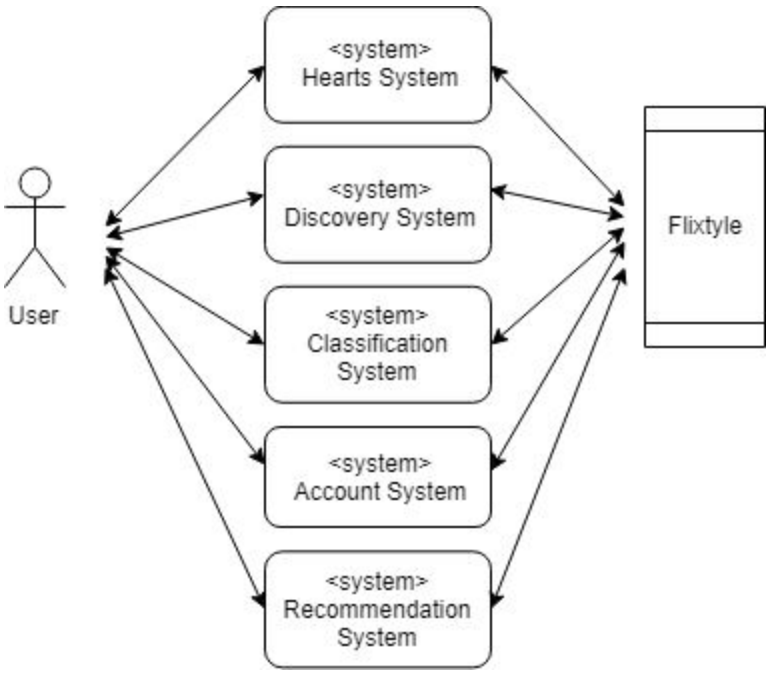
After login, the user will be presented to the three main functional pages. The account page, the discovery page and the recommendation page. The account page will contain functionality for changing the password, gender, age and country of residence, in addition to an option to link accounts (if the user has logged in previously through another authentication method).

3. System Architecture

3.1 Objectives

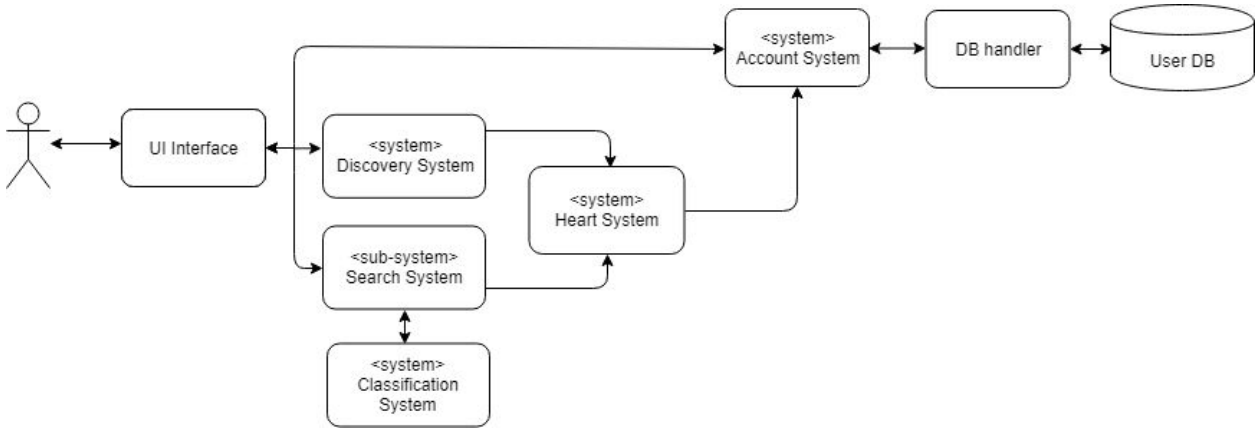
System Architecture will provide general description of the systems that our team wants to develop. In addition, the overall structure of each system is represented as a block diagram. Each relationship and how it is actually used is described as a Package Diagram and Deployment Diagram.

3.2 System Organization



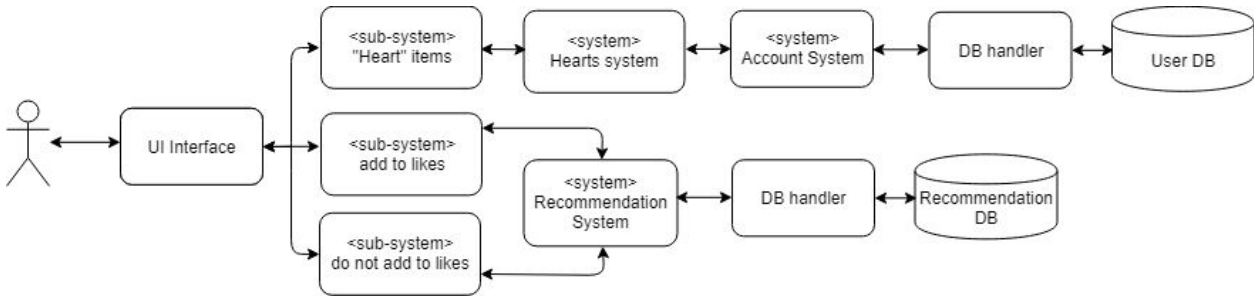
To provide the Flixtyle service, it has five systems overall. Each system stores and updates information in a database using firebase, and recalls datas from the database by the request of the user or system.

3.2.1 Hearts System



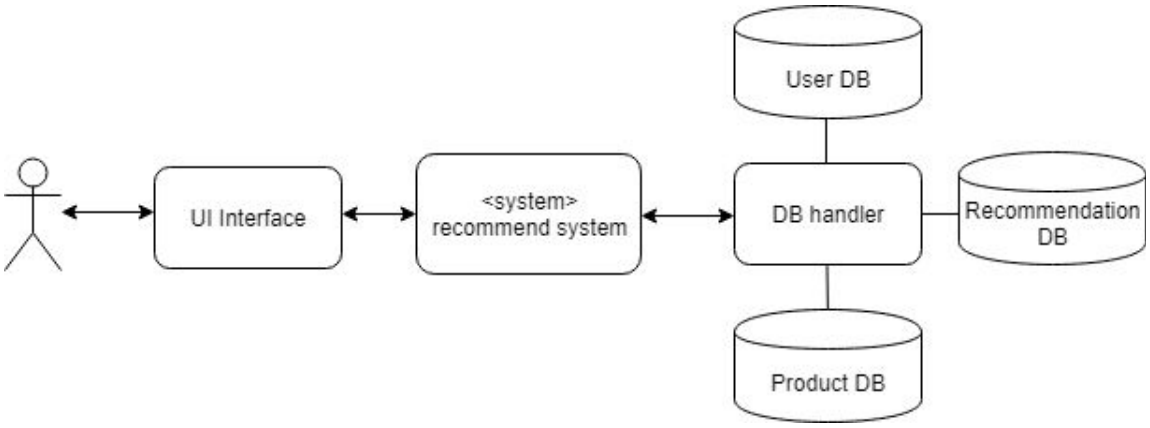
Users can heart their favorite items through the discovery and search process. These favorite items are stored in the user database and user can be checked in the user personal account by the account system. Moreover, Users can manage items from the heart list by adding or deleting them.

3.2.2 Discovery System



In the Discovery System, user can represent their preferred items through adding or not adding to likes. These items are stored in the recommendation database and used to create the recommendation item list. Additionally, the Heart system allows users to store their favorite items in the heart list.

3.2.3 Recommendation System

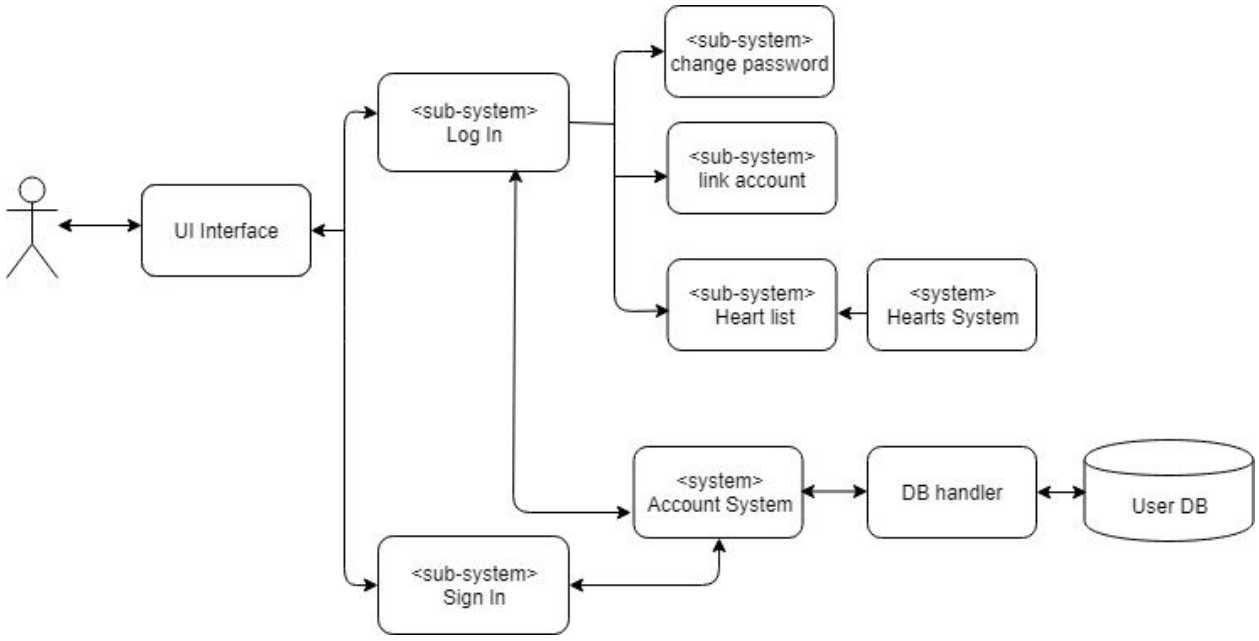


Recommendation system makes recommendation item list by using Discovery System to select the preferred item for users and saving these items in Recommendation database. Items of recommendation depends on the actual data handed by the user.

3.2.4 Classification System

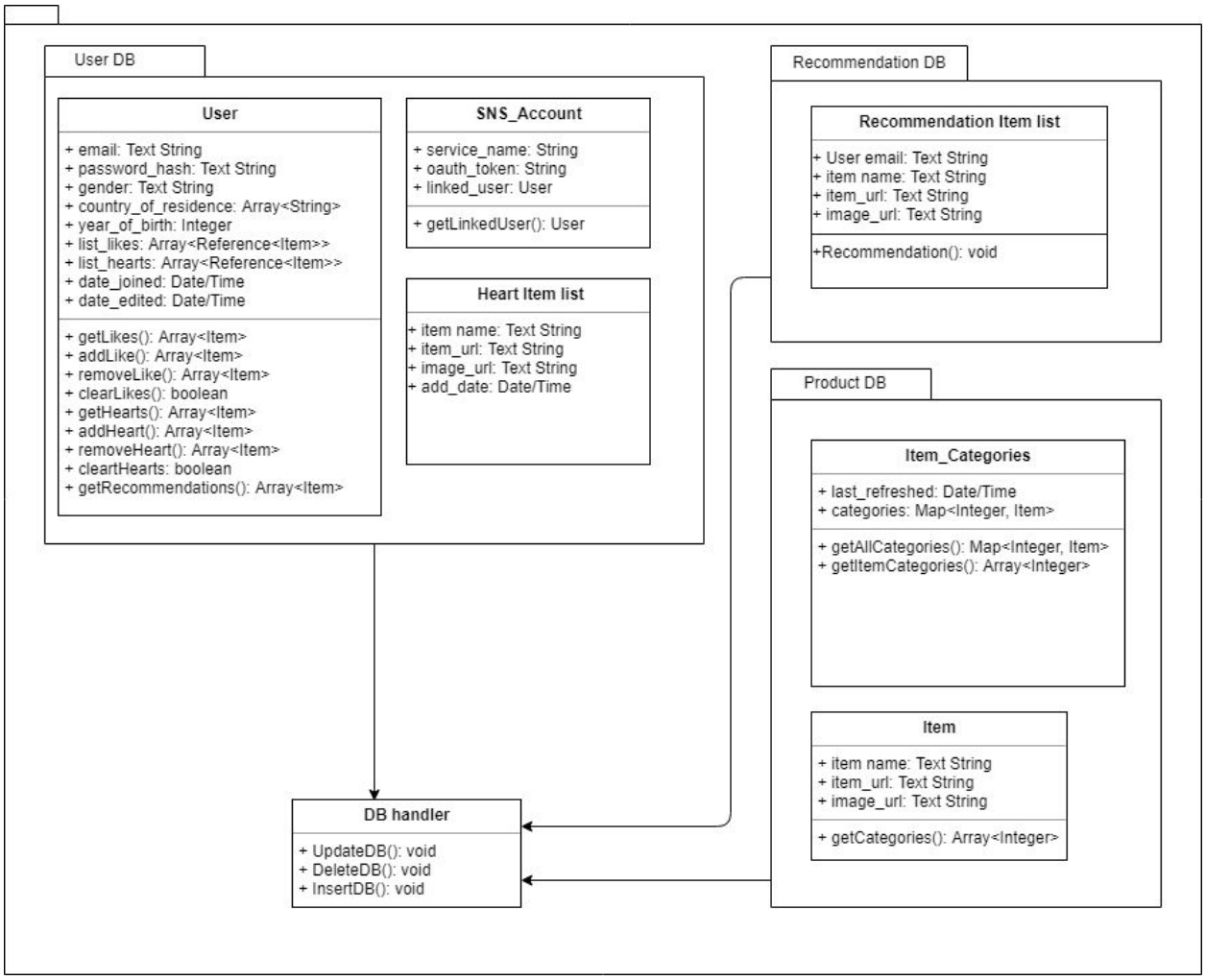
Classification system sort categories according to the type of items and store them in the database. This classification system not only makes programs easy to manage items, but also users can efficiently search items.

3.2.5 Account Management System

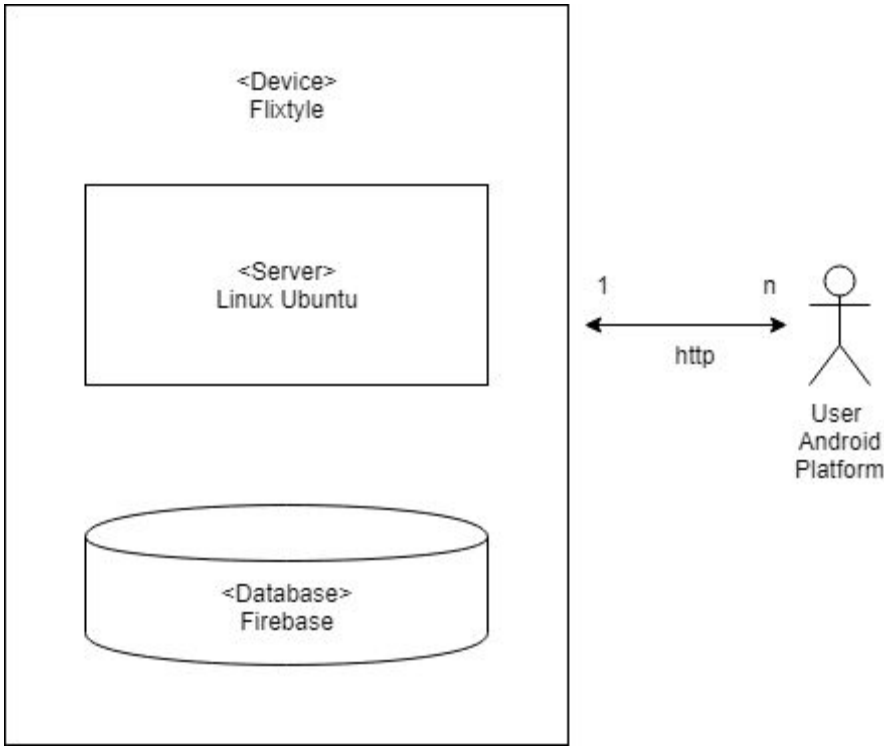


Account Management System compares the user values to the Account database, and user can enter to user account if it matches. If user don't have an account, user can sign up and Account database save the user profile that system need by default. Additionally, the Account Management System allows users to freely change user profiles or check user’s Heart list.

3.3 Package diagram



3.4 Deployment diagram

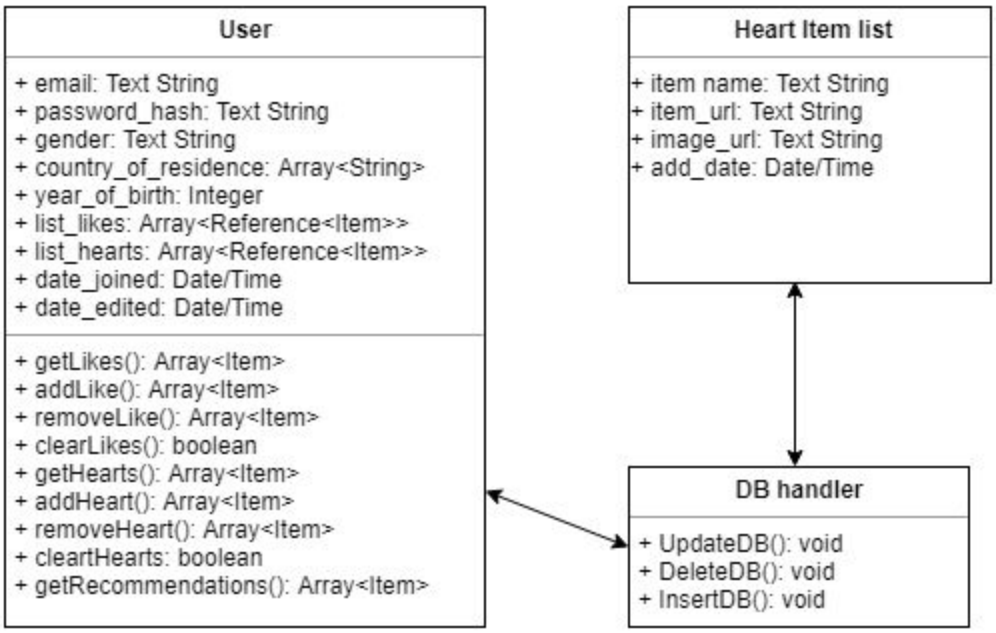


4. Hearts System

4.1 Objective

This application presents various products to users. Let's call Hearts what the user has expressed his preference for the product. Here we describe the Hearts System, which adds or manages the list of Hearts.

4.2 Class Diagram



4.2.1 User

4.2.1.1 Attributes

list_hearts	List of items the user has “Hearted”, contains reference to the item object and also the date of when the user “Hearted” the item.

4.2.1.2 Methods

getHearts()	Getter for user’s hearts.
addHeart()	Appends an item to the user’s “Heart” list and saves the resulting list.
removeHeart()	Pops an item from the user’s “Heart” list and saves resulting list.
clearHearts()	Sets the user’s “Heart” list to an empty array.

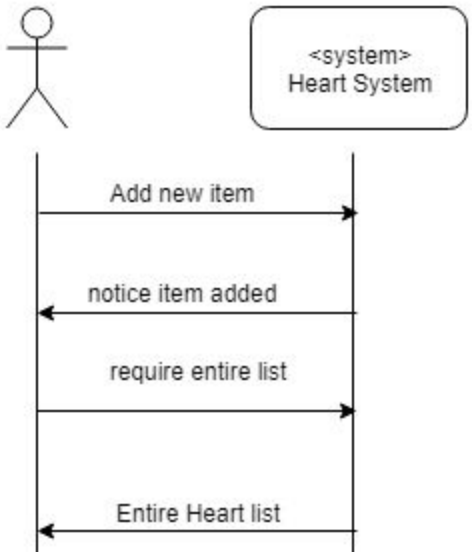
4.2.2 DB handler

4.2.2.1 Methods

updateDB()	When user choose new ‘heart’ item, them add that to heart list.
DeleteDB()	Delete heart item from list
insertDB()	When new account created make new heart list for new user.

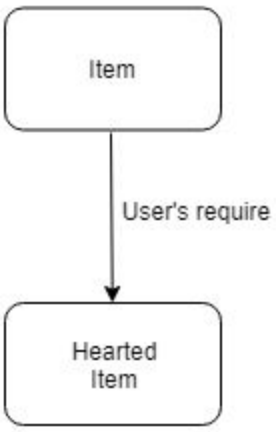
4.3 Sequence Diagram

4.3.1 User



4.4 State Diagram

4.4.1 Items

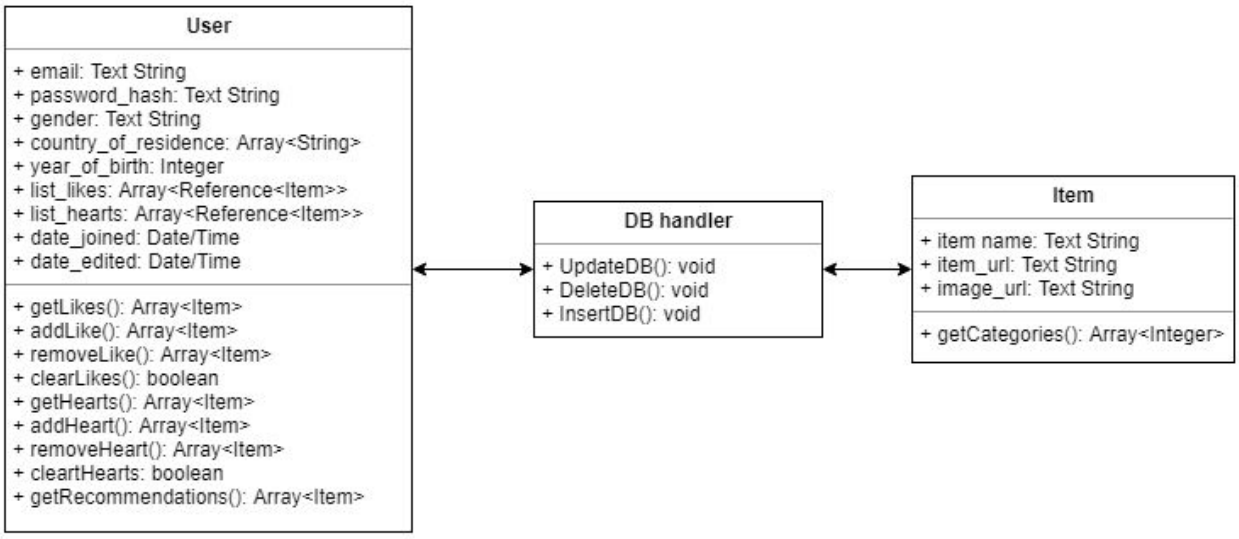


5. Discovery System

5.1 Objective

Present various products to users. The user can enjoy efficient shopping through the Recommended products.

5.2 Class Diagram



5.2.1 User

5.2.1.1 Attributes

email	User's email. In the form of (.^+)@(.^+)\.(.^+)
password_hash	Hashed and salted user password for authentication
gender	Users preferred gender for clothes. Male/Female/Both
country_of_residence	Users country of residence.
year_of_birth	User's age. Greater than 1900 and lower than the current year.
list_likes	List of items the user has liked, contains reference to the item object and also the date of when the user liked the item.
list_hearts	List of items the user has "Hearted", contains reference to the item object and also the date of when the user "Hearted" the item.
date_joined	Datetime of when the user's account was created.
date_edited	Datetime of when the user's account was last edited.

5.2.1.2 Methods

getLikes()	Getter for user's likes.
addLike()	Appends an item to the user's likes and saves the resulting list..
removeLike()	Pops an item from the user's likes and saves resulting list..
clearLikes()	Sets the user's like list to an empty array.
getHearts()	Getter for user's hearts.
addHeart()	Appends an item to the user's "Heart" list and saves the resulting list.
removeHeart()	Pops an item from the user's likes and saves resulting list.
clearHearts()	Sets the user's "Heart" list to an empty array.
getRecommendations()	Returns a list of recommendations based on the user's likes and the classification system's classifications.

5.2.2 DB handler

5.2.2.1 Methods

updateDB()	Update the DB for show new items

5.2.3 items

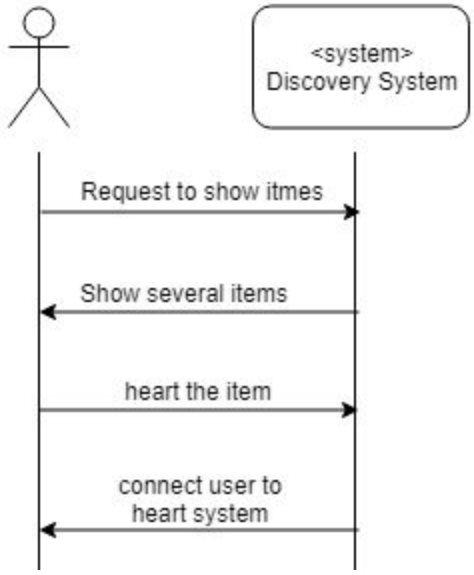
5.2.3.1 Attributes

item_name	The Unique name of item
item_url	Shopping Mall that selling the item's url.
image_url	The item's image describe item's shape.

5.2.3.2 Methods

getCategories()	Get the item's category from database.

5.3 Sequence Diagram

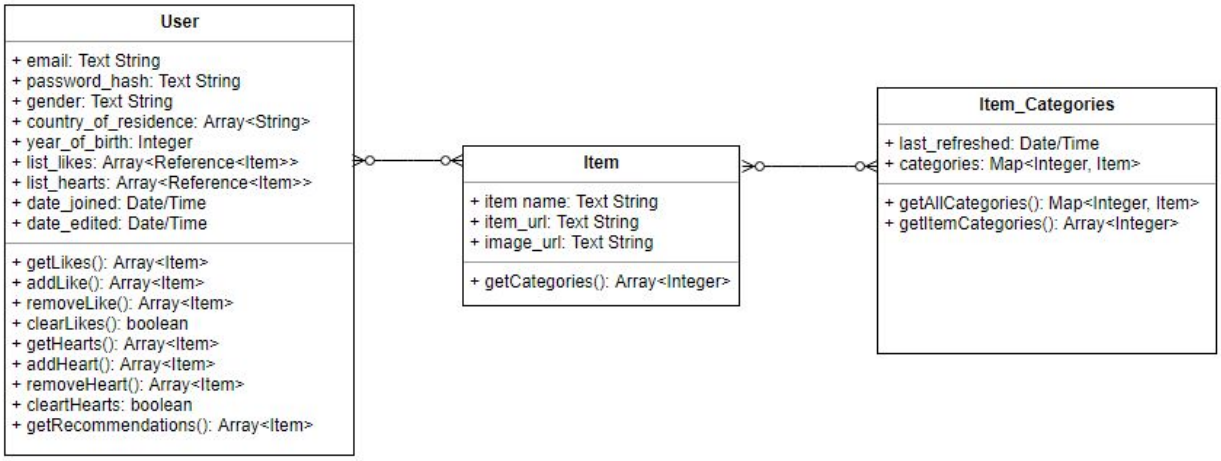


6. Recommendation System

6.1 Objective

The user will interact with this system when on the main page of the application. This chapter describes the function of the recommendation system through class- and sequence diagrams.

6.2 Class Diagram



6.2.1 User

6.2.1.1 Attributes

email	User's email. In the form of (.^+)@(.^+)\.(.^+)
password_hash	Hashed and salted user password for authentication
gender	Users preferred gender for clothes. Male/Female/Both
country_of_residence	Users country of residence.
year_of_birth	User's age. Greater than 1900 and lower than the current year.
list_likes	List of items the user has liked, contains reference to the item object and also the date of when the user liked the item.
list_hearts	List of items the user has "Hearted", contains reference to the item object and also the date of when the user "Hearted" the item.
date_joined	Datetime of when the user's account was created.
date_edited	Datetime of when the user's account was last edited.

6.2.1.2 Methods

getLikes()	Getter for user's likes.
addLike()	Appends an item to the user's likes and saves the resulting list..
removeLike()	Pops an item from the user's likes and saves resulting list..
clearLikes()	Sets the user's like list to an empty array.
getHearts()	Getter for user's hearts.
addHeart()	Appends an item to the user's "Heart" list and saves the resulting list.
removeHeart()	Pops an item from the user's likes and saves resulting list.
clearHearts()	Sets the user's "Heart" list to an empty array.
getRecommendations()	Returns a list of recommendations based on the user's likes and the classification system's classifications.

6.2.2 Item

6.2.2.1 Attributes

item_name	Name of an item
item_url	URL where the item was scraped from.
image_url	URL of the item's image.

6.2.2.2 Methods

getCategories()	Returns the item's classified categories.

6.2.3 Item Categories

6.2.3.1 Attributes

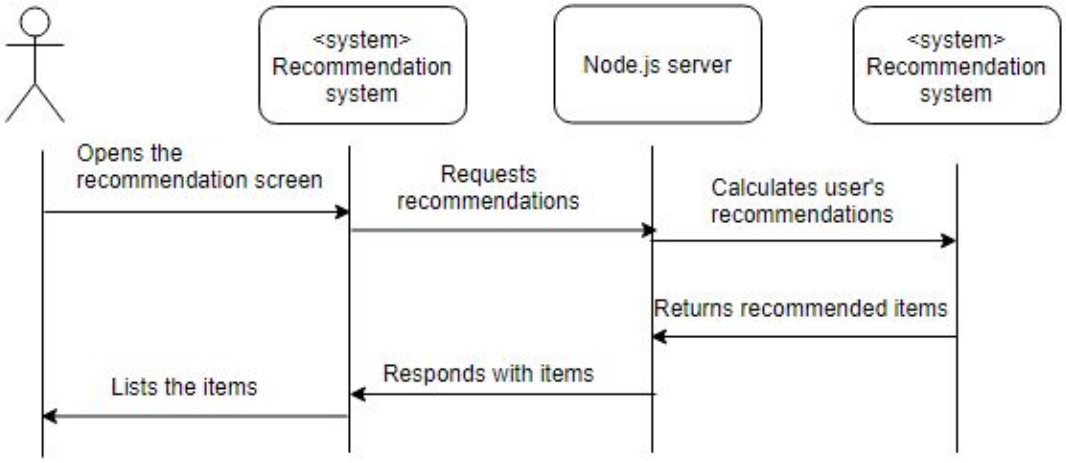
last_refreshed	Name of an item
categories	URL where the item was scraped from.

6.2.3.2 Methods

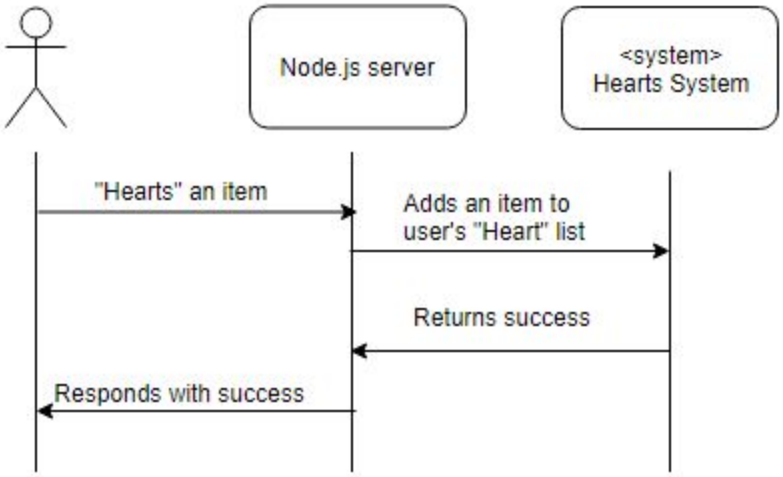
getAllCategories()	Returns all categories and the items.
getItemCategories()	Returns a specific item's categories.

6.3 Sequence Diagram

6.3.1 Show recommendations function



6.3.2 Add to “Hearts” list function



7. Classification System

2.1 Intended Audience and Reading Suggestions

Thi.

8. Account Management System

2.1 Intended Audience and Reading Suggestions

9. API Design

9.1 Objective

The objective is to describe the interfaces between the different systems and components. The interfaces are going to be implemented following the RESTful API standard. RESTful is a stateless API that uses JSON to communicate. Javascript Object Notation(JSON) is a human readable text format that a standard in the industry. JSON can be parsed and edited by many programming languages out of the box without needing to add any external libraries.

9.2 Variables

Name	Type	Explanation
email	Text String	The user's email
password_hash	Text String	The user's password
gender	Text String	The user's gender
country_of_residence	Array<String>	The user's country of residence
year_of_birth	Integer	The user's year of birth
list_likes	Array<Reference<Item>>	The user's list of likes
list_hearts	Array<Reference<Item>>	The user's list of hearts

date_joined	Data/Time	The date the user joined
item_name	Text String	Name of the item in the website
item_url	Text String	URL of the item in the website

9.3 Request and Responses

9.3.1 Request item thumbnail

Type	URL	Value	Response
GET	/items/{item_id}/thumbnail	item_id	Image file(.jpg, .png)

9.3.2 Request item website URL

Type	URL	Value	Response
GET	/items/{item_id}/viewpage	item_id	item_url

9.3.3 Create account request

Type	URL	Value	Response

POST	/accounts/create	email password gender country_of_residence year_of_birth	<Success> with session ID

9.3.4 View account request

Type	URL	Value	Response
GET	/accounts/view	Session	email gender country_of_residence year_of_birth date_joined

9.3.5 Update account information request

Type	URL	Value	Response
POST	/accounts/patch	email password gender country_of_residence year_of_birth	<Success>

9.3.6 Delete account request

Type	URL	Value	Response

POST	/accounts/delete	Session	<Success>

9.3.7 View “Hearts” list request

Type	URL	Value	Response
GET	/hearts	Session	Array<Item>

9.3.8 Add item to “Hearts” list request

Type	URL	Value	Response
POST	/hearts/add	item_id	<Success>

9.3.9 Remove item from “Hearts” list request

Type	URL	Value	Response
POST	/hearts/remove	item_id	<Success>

9.3.10 View recommendations request

Type	URL	Value	Response
GET	/recommendations	Session ID	Array<Item>

9.3.11 Request Discoverable Items

Type	URL	Value	Response
GET	/discovery/getitems	Item count: Integer	Array<Item>

9.3.12 Send item liked request

Type	URL	Value	Response
POST	/discovery/sendlike	item_id	<Success>

10. Database Design

10.1 Objective

This section shows database diagram that the project will be based on. Firebase (a Google Cloud product) will be used as the database of the application.

10.2 Data types

Cloud Firestore is a NoSQL Database. Cloud firestore stores data in documents organized in collections. The following table lists the data types supported by Cloud Firestore that are going to be used in the project.

Data Type	Sort Order	Explanation
Array	By element values	An array cannot contain another array value as one of its elements.

		Within an array, elements maintain the position assigned to them. When sorting two or more arrays, arrays are ordered based on their element values. When comparing two arrays, the first elements of each array are compared. If the first elements are equal, then the second elements are compared and so on until a difference is found. If an array runs out of elements to compare but is equal up to that point, then the shorter array is ordered before the longer array.
Date/time	Chronological	When stored in Cloud Firestore, precise only to microseconds; any additional precision is rounded down.
Integer	Numeric	64-bit, signed
Text String	UTF-8 encoded byte order	Up to 1,048,487 bytes (1 MiB - 89 bytes). Only the first 1,500 bytes of the UTF-8 representation are considered by queries.

10.3 Table

This section describes the fields per table that make up the database.

10.3.1 User

Name	Type
email	Text String
password_hash	Text String
gender	Text String

country_of_residence	Array<String>
year_of_birth	Integer
list_likes	Array<Reference<Item>>
list_hearts	Array<Reference<Item>>
date_joined	Data/Time
date_edited	Date/Time

10.3.2 SNS Account

Name	Type
service_name	String
oauth_token	String
linked_user	User

10.3.3 Item

Name	Type
item_name	Text String
item_url	Text String
image_url	Text String

10.3.4 Item Categories

Name	Type
last_refreshed	Date/Time
categories	Map<Integer,Item>

11. Testing Plan

11.1 Objectives

This section establishes a preliminary plan for system testing to determine if the system is operating as intended to identify and analyze the system for defects after completion of the system. This part is divided into testing policy, which sets the criteria for testing and test case, which gives a set of more detailed input and output.

11.2 Testing Policy

11.2.1 Development Testing

It is the concept of applying testing practices consistently throughout the software development life cycle model. This method of testing ensures detection of bugs at the right time which further ensures avoidance of any kind of risk in terms of time and cost. It basically refers to testing performed during the development process. The main goal of development testing is to evaluate the system's compliance with the specified needs. Depending on what you are testing, you will be divided into different testing levels which help to check behavior and performance for development testing. These testing levels are designed to recognize missing areas and reconciliation between the development lifecycle states. The main four testing levels are:

11.2.1.1 Component Testing

A Component is a smallest testable portion of system or application which can be compiled, liked, loaded, and executed. This kind of testing helps to test each module separately.
The aim is to test each part of the software by separating it. It checks that component are fulfilling functionalities or not. This kind of testing is performed by developers.

11.2.1.2 Integration Testing

Integration means combining. For example, In this testing phase, different software modules are combined and tested as a group to make sure that integrated system is ready for system testing.
Integrating testing checks the data flow from one module to other modules. This kind of testing is performed by testers.

11.2.1.3 System Testing

System testing is performed on a complete, integrated system. It allows checking system's compliance as per the requirements. It tests the overall interaction of components. It involves load, performance, reliability and security testing.
System testing most often the final test to verify that the system meets the specification. It evaluates both functional and non-functional need for the testing.

11.2.1.4 Acceptance Testing

Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. Acceptance testing is basically done by the user or customer. However, other stockholders can be involved in this process.

11.2.2 Release Testing

Test the final system before release to the user. Ensure that the requirements written in the requirements specification are properly reflected.

11.2.3 User Testing

The user tests the system in the user's environment.

11.3 Test Case

11.3.1 User Management System

11.3.1.1 Login

User’s action	System’s action	Success	Failure

User tries logging in with ID and password	Compare with the data stored in User DB.	Login with corresponding ID.	System alert, "The username or password does not match." Returns to login screen with login and sign up button.

11.3.1.2 Sign Up

User's action	System's action	Success	Failure
The user will fill out six fields: E-mail, password, password confirmation, gender, age, and country of residence. The email, password, and password confirmation fields will be mandatory.	Checks if passwords match and mandatory fields are filled out.	Stores user's information in the DB.	System alert, "Passwords do not match." or "Fill out mandatory fields." Returns to sign up screen.

11.3.1.2 Logout

User's action	System's action
User tries logging out.	Logout.

11.3.2 Discovery System

11.3.2.1 Swipe Left

User's action	System's action	Success	Failure
Swiping left will indicate that the user is not interested and will show a frowny face in the back.	Gets rid off information about the item.	The user can view the item on the next card and keep swiping.	Fails to go on to the next card to view the next discovery item.

11.3.2.2 Swipe Right

User's action	System's action	Success	Failure
Swiping right will indicate that the user is interested in the product and show a smiley face in the back.	Saves information that the user likes the item. Sends to recommendation page.	The user can view the item on the next card and keep swiping.	Fails to go on to the next card to view the next discovery item.

11.3.2.3 Swipe Up

User's action	System's action	Success	Failure
Swiping up will indicate that the user is very	System will save item to the "Heart" List.	The user can view the item on the next card and keep swiping.	Fails to go on to the next card to view the next discovery item.

interested in buying the product.			

11.3.3 Recommendation System

11.3.3.1 Base case

User's action	System's action	Success	Failure
User opens the recommendation view	Sends a request to the recommendation system and returns a list of recommended items	The user can view the items in the application window and scroll up and down. These items also are similar to the items the user has liked.	When the system fails to deliver the list of items to the user, be it because of connection issues or anything else. Another type of failure would be when the list doesn't reflect the user's likes.

11.3.3.2 Add to "Hearts" list

User's action	System's action	Success	Failure
User clicks on the "Heart" icon on an item	Sends a request to the "Hearts" system and saves the item to the user's "Hearts" list.	The system successfully updates the user's "Hearts" list.	When system fails to update the "Hearts" list of the user.

12. Development Plan

12.1 Objectives

Describes the programming language and the IDE that is going to be used.

12.2 Programming Language & IDE

12.2.1 Programming Language

12.2.1.1 JAVA

JAVA is used as the development programming language. JAVA is an object-oriented language and has the advantage of running on any platform. It is widely used in Android application development language and is suitable for our project.

12.2.1.2 Node.js

Node.js is used for back-end development. Using node.js requires using javascript, so we will use JavaScript for server-side programming. JavaScript is an object-based scripting programming language. Node.js will handle the networking.

12.2.2 IDE

The IDE used in the project development process is Android Studio. Android Studio is the official integrated development environment for Android and Android applications. Java and Kotlin are the programming languages used in Android Studio.

12.3 Version Management Tool

GitHub will be used for code management and more efficient development. GitHub is a web hosting service that provides extended project management support to Git, a version management system. It is currently one of the most popular web hosting services for open source project management around the world.