Recommend Laptop for Tech Newbies

Software Design Specification

2020.05.24.

Introduction to Software Engineering 41

TEAM 1 (iDecide)

Team Leader	Kyungyeon Park
Team Member	Allan Tazhenov
Team Member	Hanwoul Lee
Team Member	Hogi Min
Team Member	Sara B.Zaki
Team Member	Seungrok Yoon

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Preface

This chapter contains the readership information, readership, scope, objective of this document and the document structure of this Software Design Document for iDecide.

1.1. Readership

This Software Design Document is divided into 10 sections with various subsections. The structure of the Software Design Document can be found as listed below, in the Document Structure subsection of this SDD. In this document, Team 1 is the main reader. Additionally, professors, TAs, and team members in the Introduction to Software Engineering class can be the main readers.

1.2. Scope

This Design Specification is to be used by Software Engineering and Software Quality Engineering as a definition of the design to be used to implement the Recommendation System that recommend laptop for tech newbies.

1.3. Objective

The primary purpose of this Software Design Document is to provide a description of the technical design aspects for our laptop recommendation application, iDecide. This document describes the software architecture and software design decisions for the implementation of iDecide. It also provides an architectural overview of the system to depict different aspects of the system. It further specifies the structure and design of some of the modules discussed in the SRS document and in addition, displays some of the use cases that have been transformed into sequential and activity diagrams, including the class diagrams which show how the programming team would implement the specific module. The intended audience of this document is, but not limited to, the stakeholders, developers, designers, and software testers of the iDecide mobile application.

1.4. Document Structure

- **1. Preface**: this chapter describes readership, scope of this document, object of this system, and structure of this document.

- **2. Introduction**: this chapter describes several tools used for this document, several diagrams used in this document and the references, and object of this project.
- **3. Overall System Architecture**: this chapter describes overall architecture of the system using context diagram, sequence diagram, and use case diagram.
- **4. System Architecture - Frontend**: this chapter describes architecture of the frontend system using class diagram and sequence diagram.
- **5. System Architecture - Backend**: this chapter describes architecture of the backend system using class diagram and sequence diagram.
- **6. Protocol Design**: this chapter describes design of several protocols which used for communication of client and server.
- **7. Database Design**: this chapter describes database design using several ER diagrams and SQL DDL.
- **8. Testing Plan**: this chapter describes testing plan for our system.
- **9. Development Plan**: this chapter describes which tools to use to develop the system, constraints, assumption, and dependencies for developing this system.
- **10. Supporting Information**: this chapter describes baseline of this document and history of this document.

2. Introduction

The project is to develop and design a mobile application used for the recommendation of computer devices, specifically laptops, to those who are not well-versed in the technological field. The system should be able to determine what laptop would be best for the user by using specialized algorithms to find a match tailored to the user’s needs. This design document presents the designs used or intended to be used in implementing the project. The designs described, follow the requirements specified in the Software Requirements Specifications document prepared earlier for the project.

2.1. Objectives

In this chapter, we describe the various tools and diagrams which we have applied to this project in the design phase.

2.2. Applied Diagrams

2.2.1. UML

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it’s one of the most popular business process modeling techniques. UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems. UML can be applied to diverse application domains (e.g., banking, finance, internet, aerospace, healthcare, etc.) It can be used with all major object and component software development methods and for various implementation platforms (e.g., J2EE, .NET). It is based on diagrammatic representations of software components. As the old proverb says: “a picture is worth a thousand words”. By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

2.2.2. Use case Diagram

A cornerstone part of the system is the functional requirements that the system fulfills. Use Case diagrams are used to analyze the system’s high-level requirements. These requirements are expressed through different use cases. We notice three main components of this UML diagram: Functional requirements – represented as use cases; a verb describing an action. Actors – they interact with the system; an actor can be a human being, an organization or an internal or external application. Relationships between actors and use cases – represented using straight arrows.

2.2.3. Sequence Diagram

Sequence diagrams are probably the most important UML diagrams among not only the computer science community but also as design-level models for business application development. Lately, they have become popular in depicting business processes, because of their visually self-explanatory nature. As the name suggests, sequence diagrams describe the sequence of messages and interactions that happen between actors and objects. Actors or objects can be active only when needed or when another object wants to communicate with them. All communication is represented in a chronological manner.

2.2.4. Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Since classes are the building block of objects, class diagrams are the building blocks of UML. The various components in a class diagram can represent the classes that will actually be programmed, the main objects, or the interactions between classes and objects. The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object.

2.2.5. Context Diagram

The system context diagram (also known as a level 0 DFD) is the highest level in a data flow diagram and contains only one process, representing the entire system, which establishes the context and boundaries of the system to be modeled. It identifies the flows of information between the system and external entities (i.e. actors). A context diagram is typically included in a requirements document. It must be read by all project stakeholders and thus should be written in plain language, so the stakeholders can understand items. The objective of the system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of systems requirements and constraints. A system context diagram is often used early in a project to determine the scope under investigation. Thus, within the document. A system context diagram represents all external entities that may interact with a system. The entire software system is shown as a single process. Such a diagram pictures the system at the center, with no details of its interior structure, surrounded by all its External entities, interacting systems, and environments.

2.2.6. Entity Relationship Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. Entity Relationship diagrams are used to sketch out the design of a database.

2.3. Applied Tools

2.4.1 Microsoft PowerPoint

This is a tool that supports drawing text and figures. It is convenient to draw diagrams using various shapes. In addition, it is easy to edit because it works as full word-processor formatting (which is a tool for working with documents), graphic shapes with attached text for drawing diagrams and tables.

2.4.2 ERD Plus

This is a web-based database modeling tool that lets you quickly and easily create Entity Relationship Diagrams (ERDs), Relational Schemas (Relational Diagrams), and Star Schemas (Dimensional Models).

2.4. Project Scope

The iDecide recommendation system is established to reduce the exhausting hours of searching for the perfect laptop and to create a convenient and easy-to-use application for ordinary users, trying to find and buy an appropriate device in a short amount of time. The system is based on a relational database with AI functions. We will have a database server supporting hundreds of computer manufacturers around the world as well as thousands of devices such as laptops by various companies and from a wide variety of vendors. Above all, we hope to provide a comfortable user experience along with the best possible pricing available to the users.

2.5. References

The user of this SDD may need the following documents for reference:

- Team 5, 2019 Spring. Software Design Document, SKKU.
- Appleton, Brad. A Software Design Specification Template. N.d.
- P. Burke, K. Martin, D. Longtin. Software Design Specification for a One Runway Airport/Air Traffic Controller Simulation <https://www.academia.edu/29811493/SOFTWARE_DESIGN_SPECIFICATION_F

OR_A_ONE_RUNWAY_AIRPORT_AIR_TRAFFIC_CONTROLLER_SIMULATIO N>.

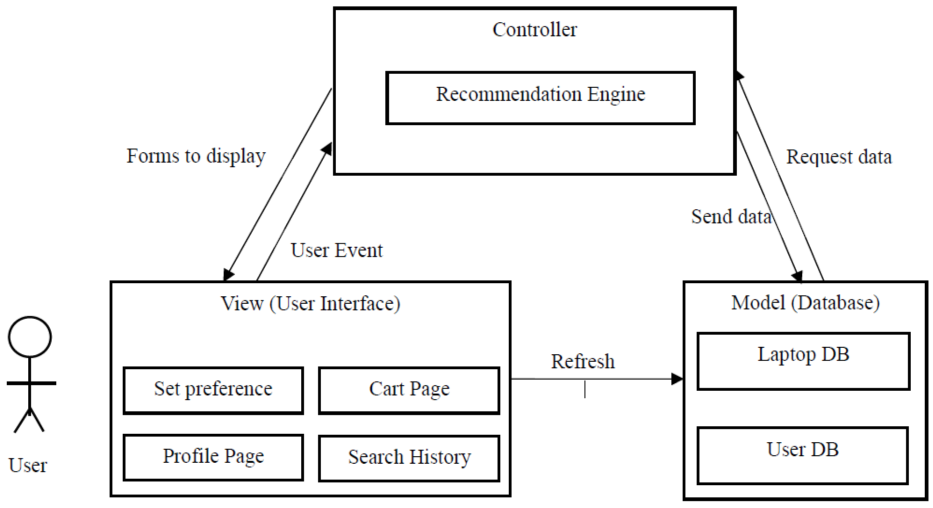
3. System Architecture – Overall

3.1. Objectives

Here in this chapter, we describe and show the organization of the system, ranging from the frontend design to the backend design of the application for the project.

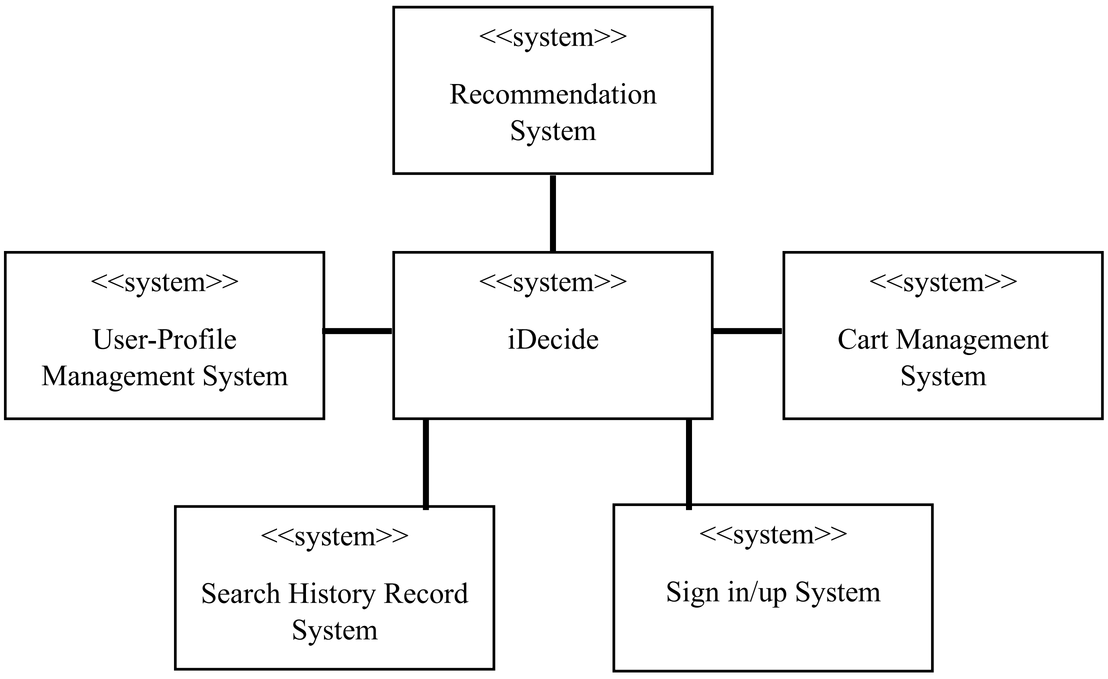
3.2. System Organization

This service is designed by applying the client - server model, frontend Application is responsible for all interactions with users, and the front-end application and back-end application send and receive data through HTTP communication based on JSON. The back-end application distributes design specification 17 requests from the front-end to the controller, obtains the required object information from the database, processes it from the database, and delivers it to the JSON format. The back-end application receives Reviews from the Review Collecting System requests every time. Use the target crawling site list and target product list to collect information on the review. Once the review information is collected, the review analysis system will deliver the minimum processed list of reviews to the Review Analysis System, and the review analysis system will use Google Natural Language API to store the necessary information in the database. The Item Ranking System and Recommendation System will then update the score rankings and recommended categories of each product using the updated Review database. Subsequently, if the user requests the information again, the updated information is delivered.



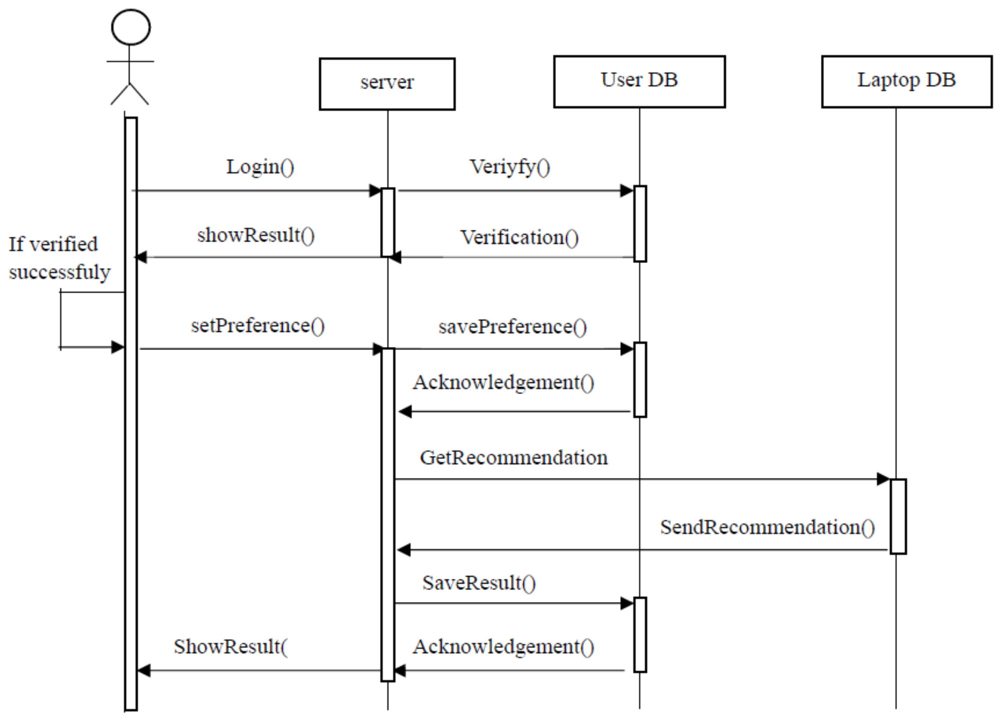
[Figure 1] Overall system architecture

3.2.1. Context Diagram


<<system>> Recommendation System
<<system>> User-Profile Management System
<<system>> iDecide
<<system>> Cart Management System
<<system>> Search History Record System
<<system>> Sign in/up System

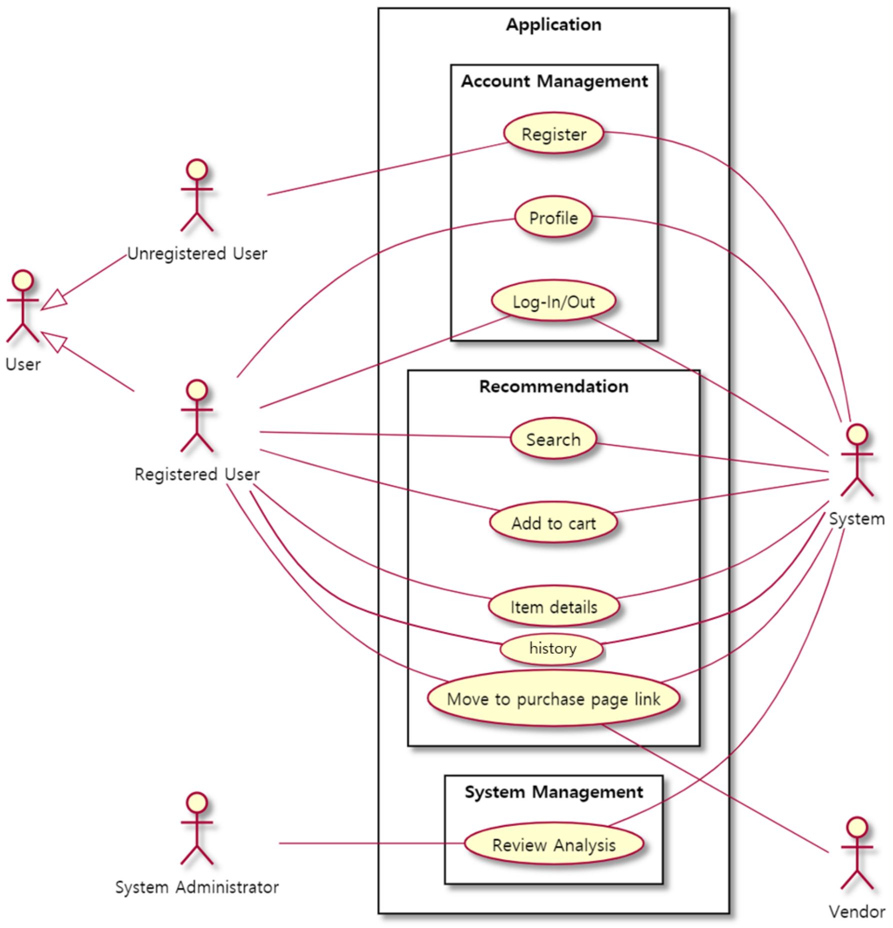
[Figure 2] Overall context diagram

3.2.2. Sequence Diagram



[Figure 3] Overall sequence diagram

3.2.3. Use Case Diagram



[Figure 4] Use case diagram

4. System Architecture – Frontend

4.1. Objectives

This chapter describes structure, attributes and function of the frontend system and describe the relation of each component.

4.2. Subcomponents

4.2.1. Profile

The profile class deals with basic user information. After user register, user must enter the profile. After user login, user can modify user profile.

4.2.1.1. Attributes

These are the attributes that profile object has.

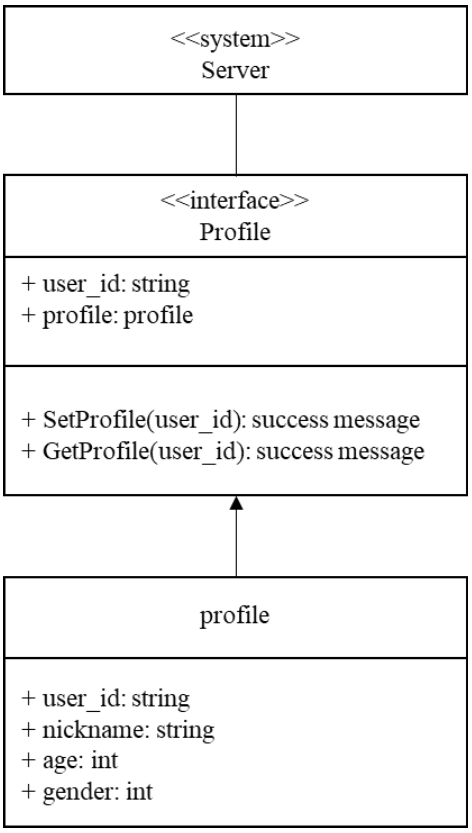
- **User id**: id of the user (email address)
- **Nickname**: nickname of the user
- **Age**: age of the user (get in date of birth)
- **Gender**: gender of the user

4.2.1.2. Methods

These are the methods that profile class has.

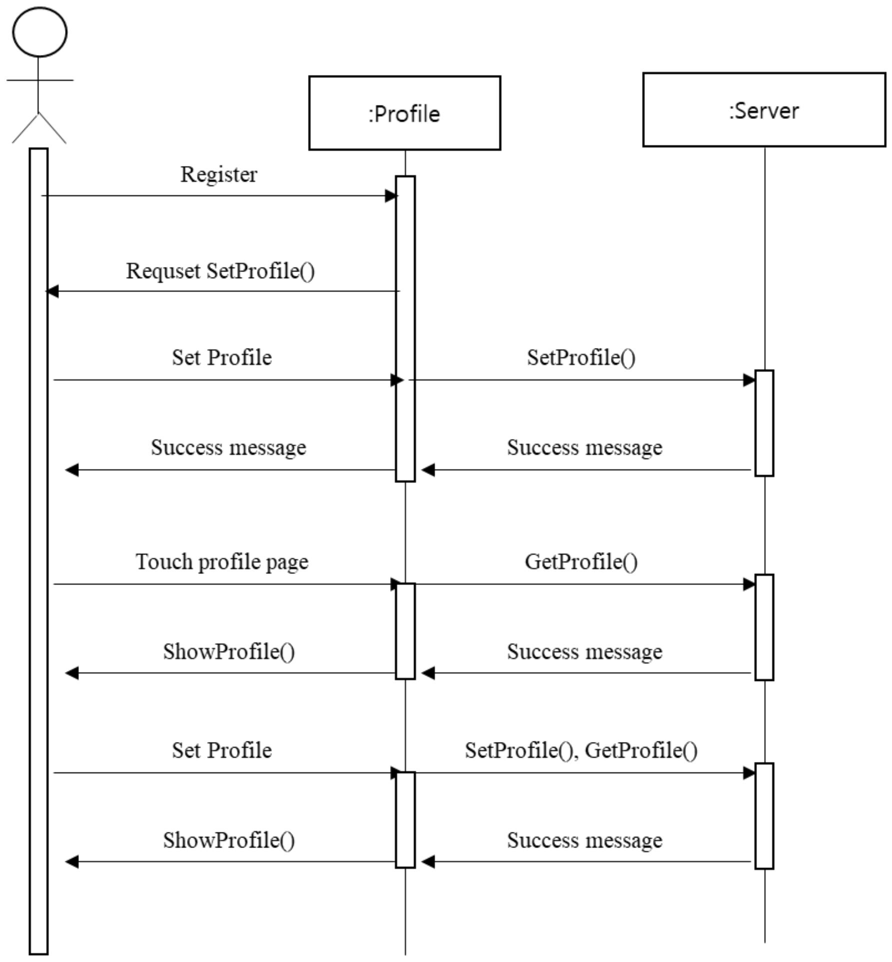
- SetProfile()
- GetProfile()

4.2.1.3. Class Diagram



[Figure 5] Class diagram – Profile

4.2.1.4. Sequence Diagram



[Figure 6] Sequence diagram – Profile

4.2.2. Search

The search class deals with user preference. this class get the preference about the laptop from user and send the preference object to recommendation system.

4.2.2.1. Attribute

These are the attributes that preference object has.

- **Budget**: budget of the user. Get this data in integer format (pick number).
- **Main purpose**: main purpose of the user. Personal or office. Get this data in integer format (pick number).
- **Usage**: usage of the user.

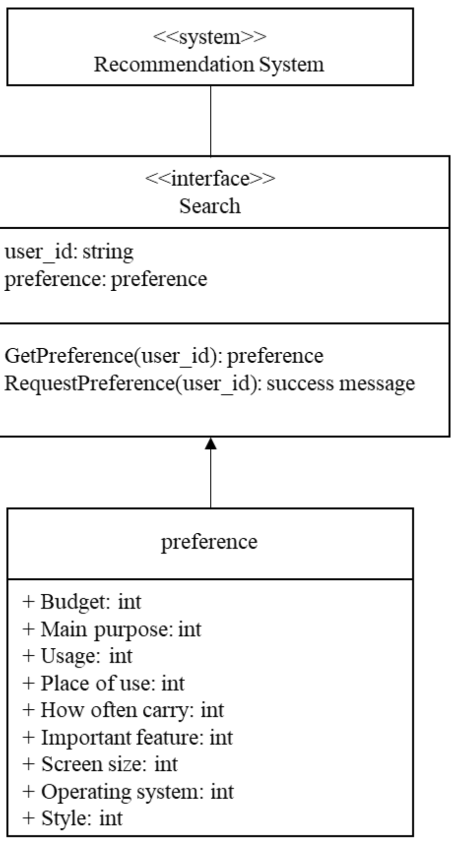
- **Place of use**: where user mainly use the laptop. Caffe, home, office, etc. Get this data in integer format (pick number).
- **How often carry**: how often user carry the laptop. Get this data in integer format (pick number).
- **Important feature**: user can select the feature in order of importance. Get this data in integer format (pick number).
- **Screen size**: screen size of the laptop that user want. 13inch, 14inch, etc. Get this data in integer format (pick number).
- **Operating system**: select main operating system. Window, IOS, Mac. Get this data in integer format (pick number).
- **Style**: style of the user. Cute, Cool, etc. Get this data in integer format (pick number).

4.2.2.2. Methods

These are the methods that profile class has.

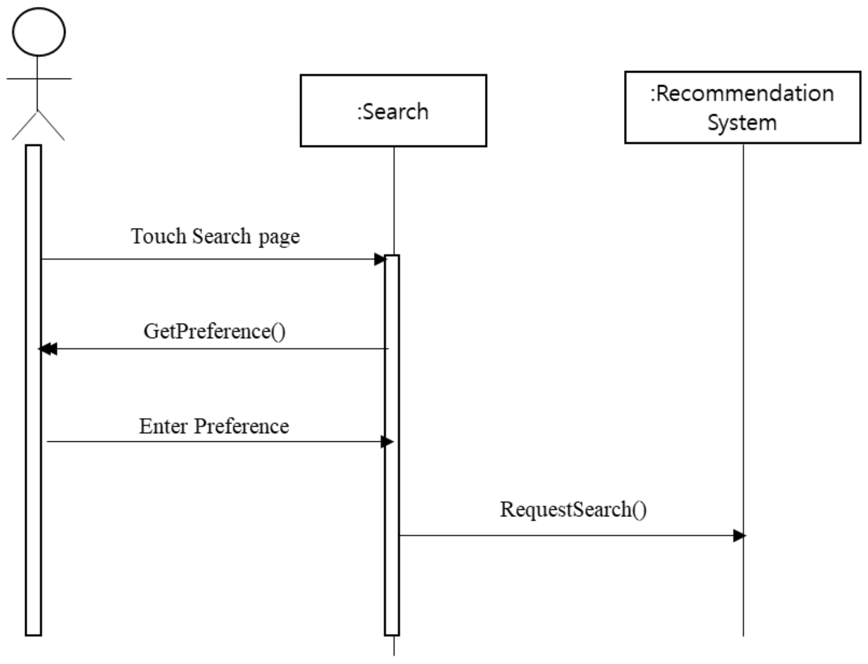
- GetPreference()
- RequestSearch()

4.2.2.3. Class Diagram



[Figure 7] Class diagram – Search

4.2.2.4. Sequence Diagram



[Figure 8] Sequence diagram – Search

4.2.3. Search Result

The search result class get the recommendation result from the recommendation system and contains recommended laptop. After search, the search result should be added to the server.

4.2.3.1. Attributes

These are the attributes that search result object has.

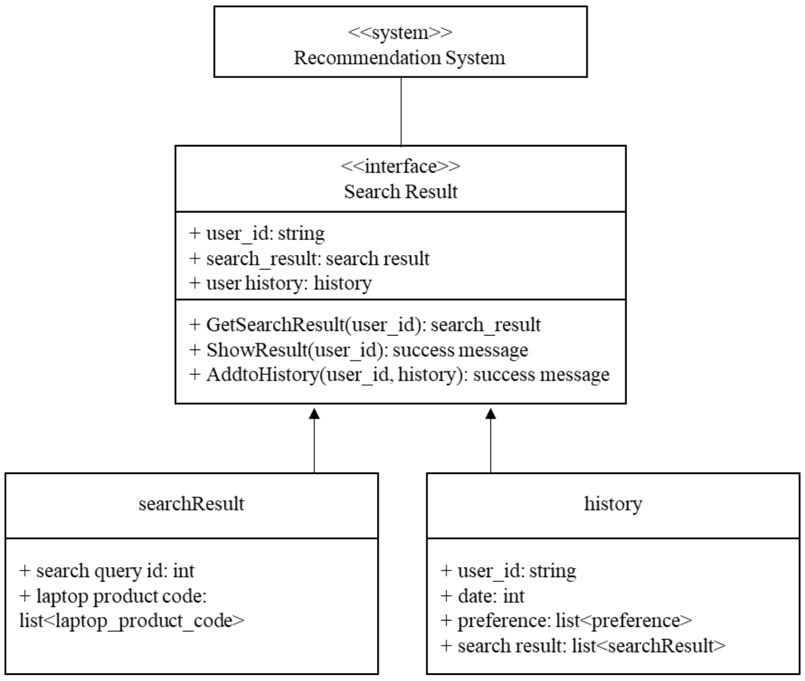
- **Search query id**: primary key for search. Using this value, system delete the search history.
- **Laptop product code**: product code of the laptop

4.2.3.2. Methods

These are the methods that search result class has.

- GetSearchResult()
- ShowResult()
- AddtoHistory()

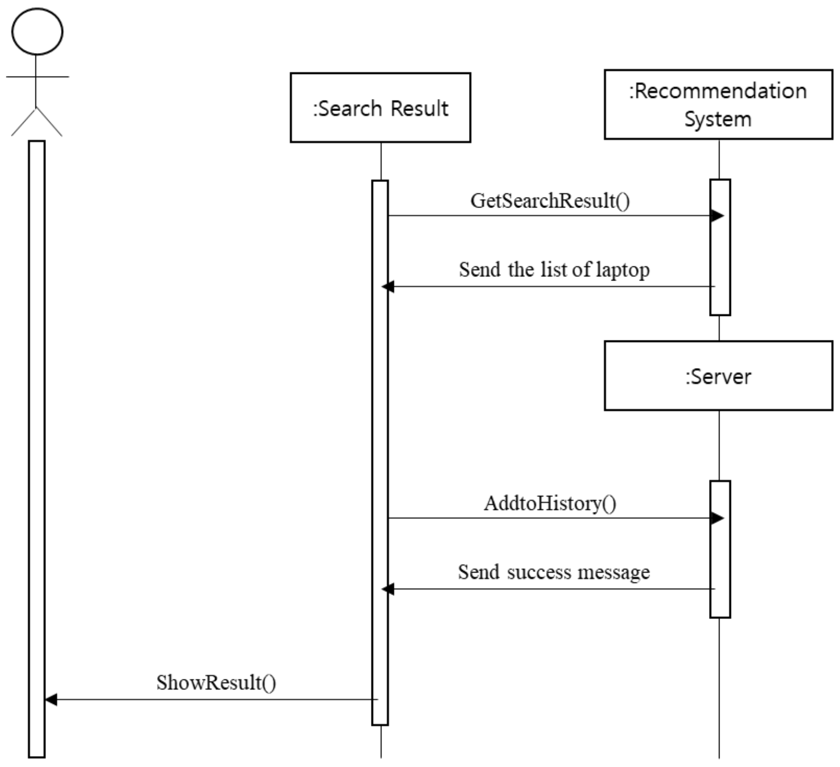
4.2.3.3. Class Diagram



<<system>>
Recommendation System
<<interface>>
Search Result
+ user_id: string
+ search_result: search result
+ user history: history
+ GetSearchResult(user_id): search_result
+ ShowResult(user_id): success message
+ AddtoHistory(user_id, history): success message
searchResult
+ search query id: int
+ laptop product code: list<laptop_product_code>
history
+ user_id: string
+ date: int
+ preference: list<preference>
+ search result: list<searchResult>

[Figure 9] Class diagram – Search result

4.2.3.4. Sequence Diagram



[Figure 10] Sequence diagram – Search result

4.2.4. History

The history class deals with the result of the search histories. This class get the search history of the user from the server and show it to user. If user search new laptop, this new search history object should be added to server. Also, if user want some history to be removed, the search history object will be removed.

4.2.4.1. Attributes

These are the attributes that search history object has.

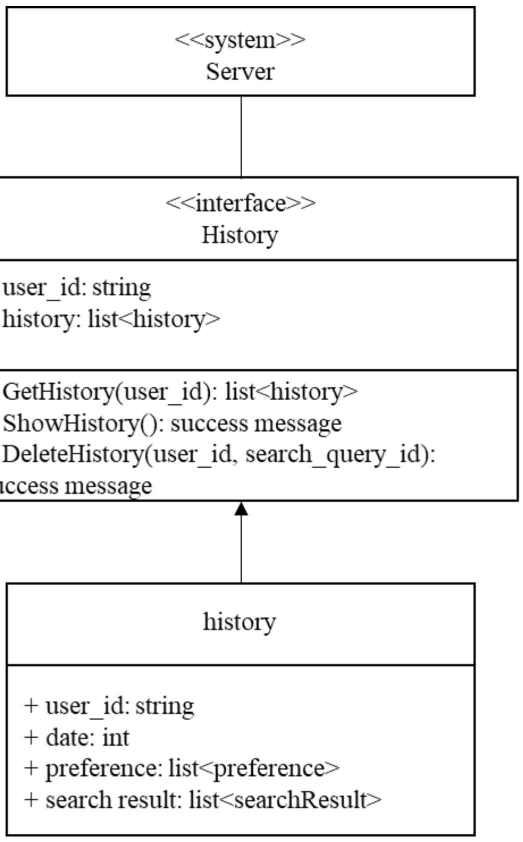
- **User id**: id of the user
- **Date**: date when user search
- **User preference**: preference of the user for that search
- **Search result**: result of the search. Contains search result.

4.2.4.2. Methods

These are the methods that search history class has

- GetHistory()
- ShowHistory()
- DeleteHistory()

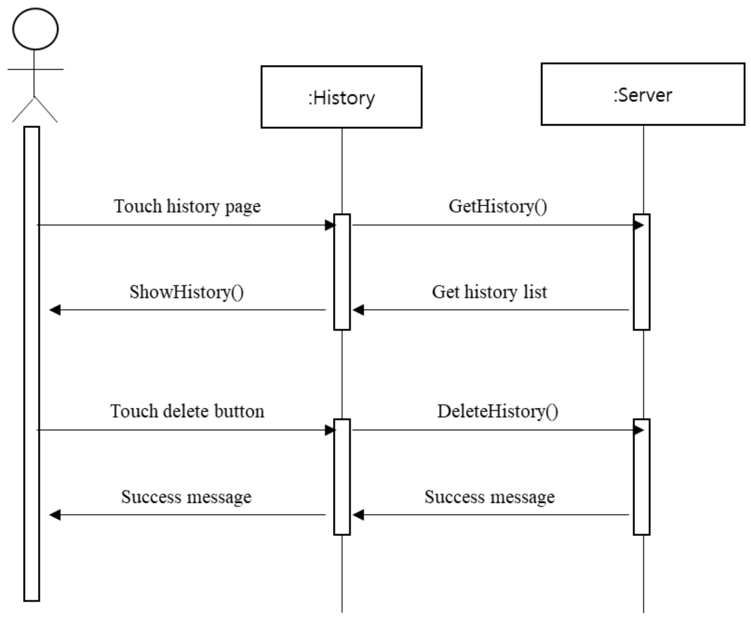
4.2.4.3. Class Diagram



<<system>>
Server
<<interface>>
History
+ user_id: string
+ history: list<history>
+ GetHistory(user_id): list<history>
+ ShowHistory(): success message
+ DeleteHistory(user_id, search_query_id): success message
history
+ user_id: string
+ date: int
+ preference: list<preference>
+ search result: list<searchResult>

[Figure 11] Class diagram – History

4.2.4.4. Sequence Diagram



[Figure 12] Sequence diagram – History

4.2.5. Cart

The cart class deals with laptop object that user added to cart. If user click the laptop object, item detail page will be loaded. Also, if user want some laptop to be removed, the laptop object will be removed from the cart.

4.2.5.1. Attributes

These are the attributes that cart object has.

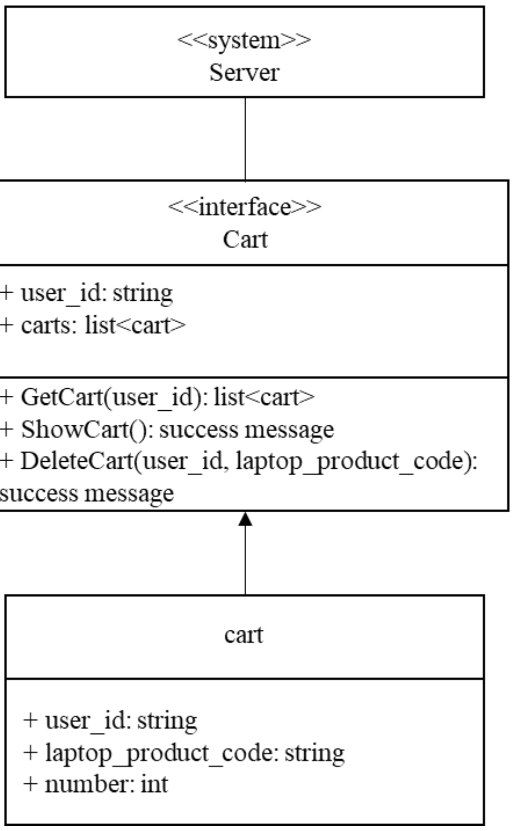
- **User id**: id of the user
- **Laptop product code**: product code of the laptop
- **number**: number for the item

4.2.5.2. Methods

These are the methods that profile class has.

- GetCart()
- ShowCart()
- DeleteItem()

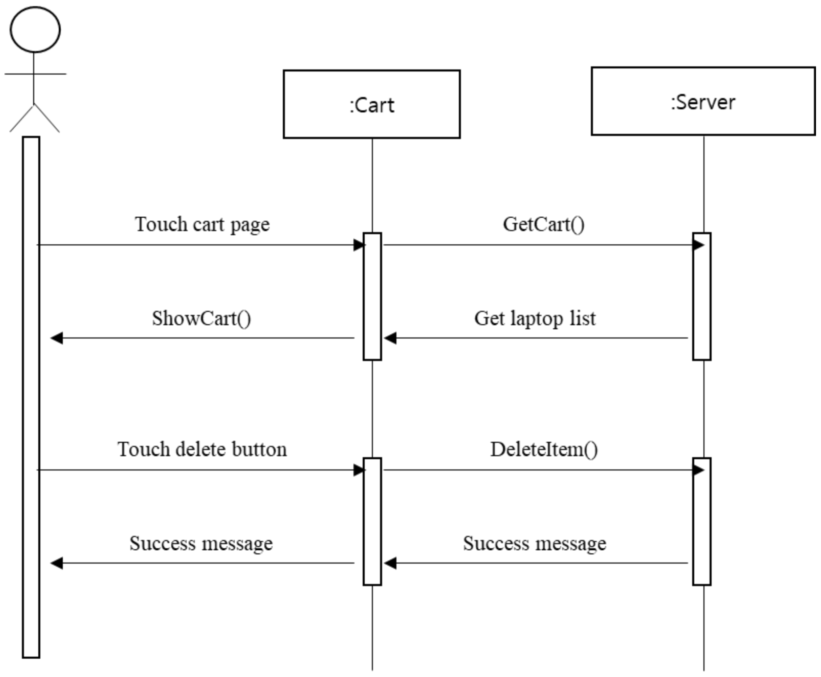
4.2.5.3. Class Diagram



<<system>>
Server
<<interface>>
Cart
+ user_id: string
+ carts: list<cart>
+ GetCart(user_id): list<cart>
+ ShowCart(): success message
+ DeleteCart(user_id, laptop_product_code): success message
cart
+ user_id: string
+ laptop_product_code: string
+ number: int

[Figure 13] Class diagram – Cart

4.2.5.4. Sequence Diagram



[Figure 14] Sequence diagram – Cart

4.2.6. Item Detail

The item detail class deal with detail of the item. After user get the recommendation of the laptop and click the item, or click the item in the cart, item detail will be shown to user.

4.2.6.1. Attributes

These are the attributes that laptop object has.

- **Laptop Product code**: product code of the laptop.
- **Manufacturer**: manufacturer of the laptop. Samsung, LG, etc.
- **Brand**: brand of the laptop. always9, galaxybook, etc.
- **Device type**: type of the laptop. 2in1, etc.
- **Display name**: name of the display.
- **CPU name**: name of the CPU. Intel i5-8250U, etc.
- **VGA name**: name of the VGA of the laptop
- **Battery capability**: size of the battery. (Wh)
- **Operating system**: operating system of the laptop.
- **Weight**: weight of the laptop. (kg)

These are the attributes that laptop review object has.

- **Laptop product code**: product code of the laptop.
- **Review**: review of the other users. Saved in list of integers. (score)

These are the attributes that vender link object has.

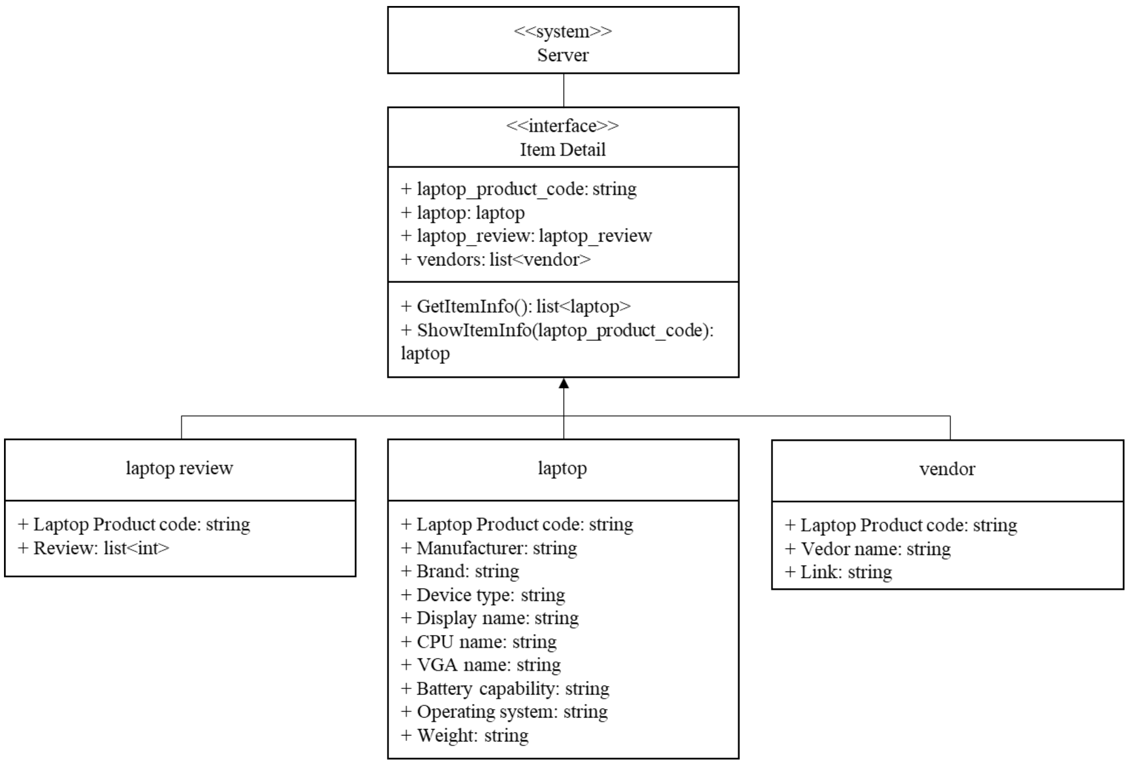
- **Laptop product code**: product code of the laptop.
- **Vender name**: name of the vendor. (G market, 11 street, etc.)
- **Link**: link for the purchase page.

4.2.6.2. Methods

These are the methods that profile class has.

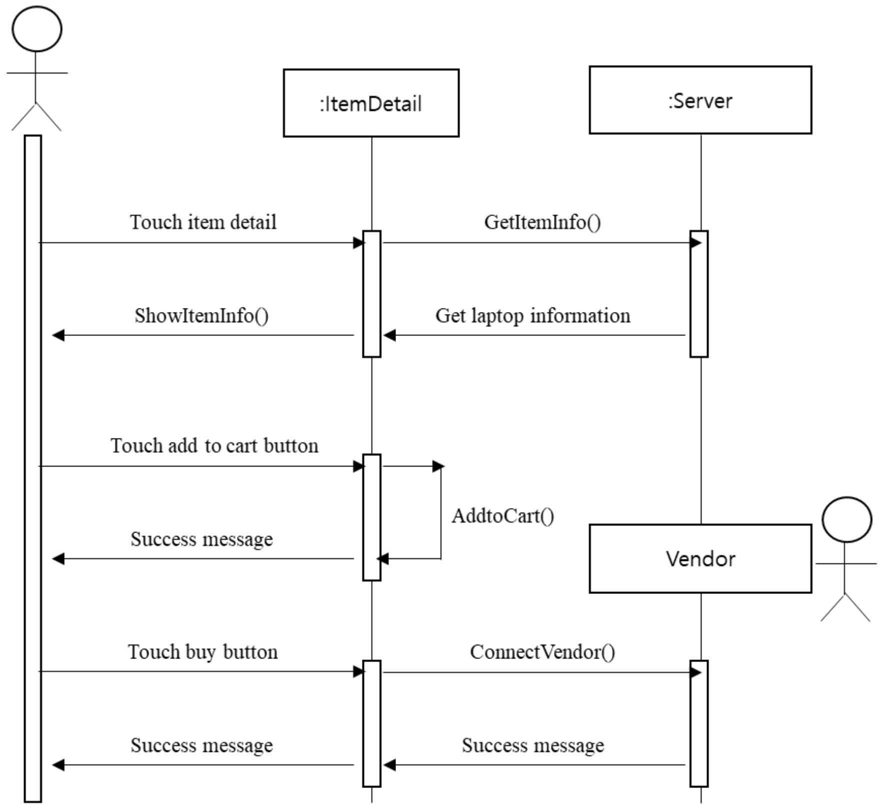
- GetItemInfo()
- ShowItemIno()
- AddtoCart
- ConnectVendor()

4.2.6.3. Class Diagram



[Figure 15] Class diagram – Item detail

4.2.6.4. Sequence Diagram



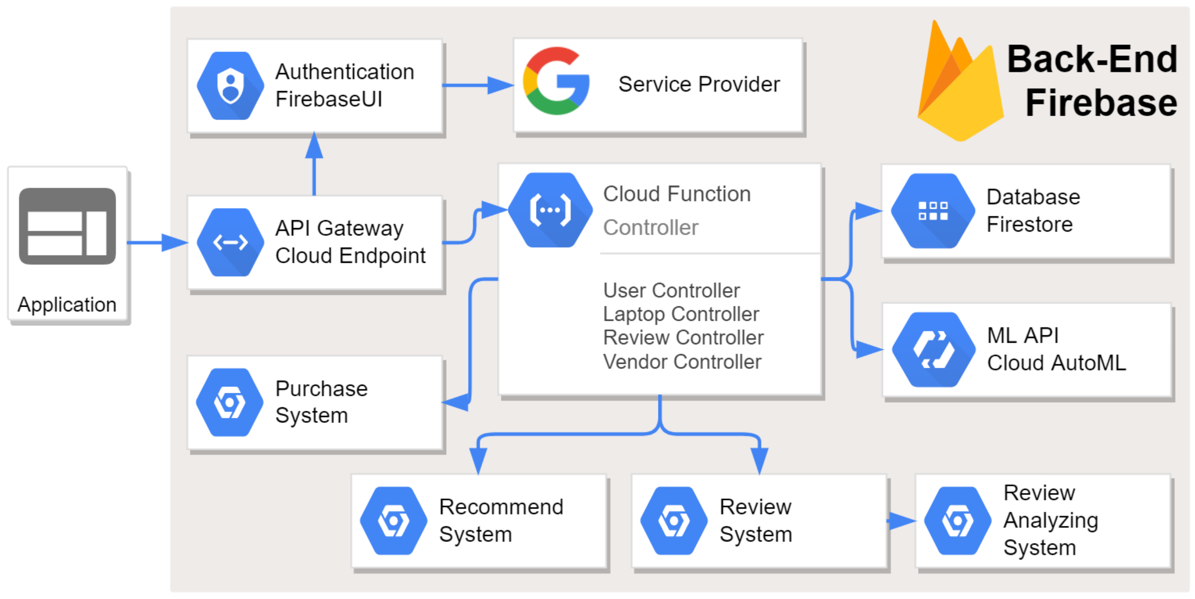
[Figure 16] Sequence diagram – Sequence diagram

5. System Architecture – Backend

5.1. Objectives

This chapter describes the structure of the back-end system include DB and API Cloud include.

5.2. Overall Architecture

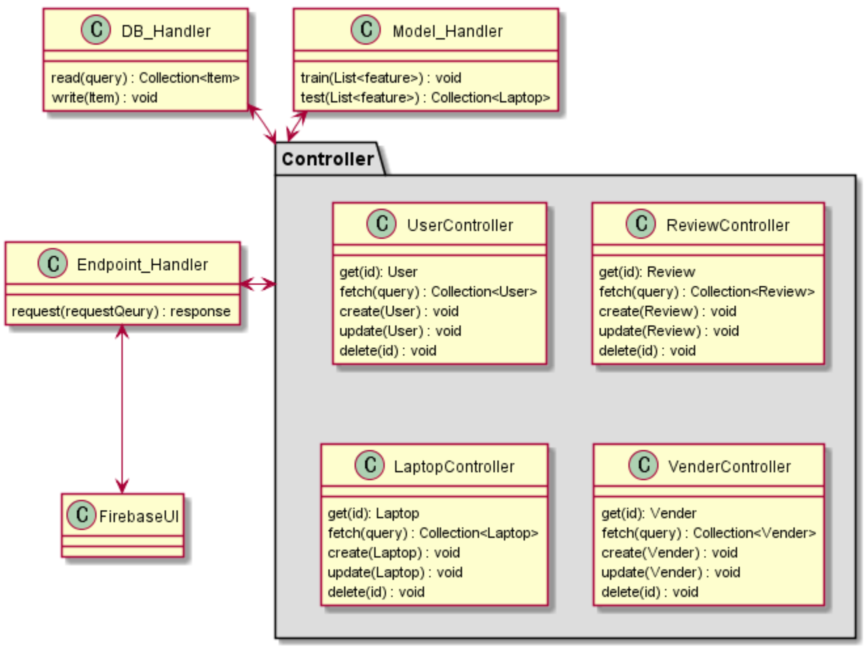


[Figure 17] Overall architecture

The overall architecture of the system is as above. The API gateway (Request Handler) receives the request from the front-end and distributes it to the appropriate cloud function (Controller / Manager). In this process, function that use external API such as authentication are handled. Other requests (processed internally) sent to the corresponding controller. The controller interacts with the cloud (Database / Cloud AutoML) and processes request.

5.3. Subcomponents

5.3.1. Cloud Function



[Figure 18] Class diagram – Cloud functions

5.3.1.1. Endpoint Handler Class

API gateway. Distribute requests from the frontend to the appropriate controller or API.

5.3.1.2. FirebaseUI

Class to implement authentication through FirebaseUI

5.3.1.3. DB_Handler Class

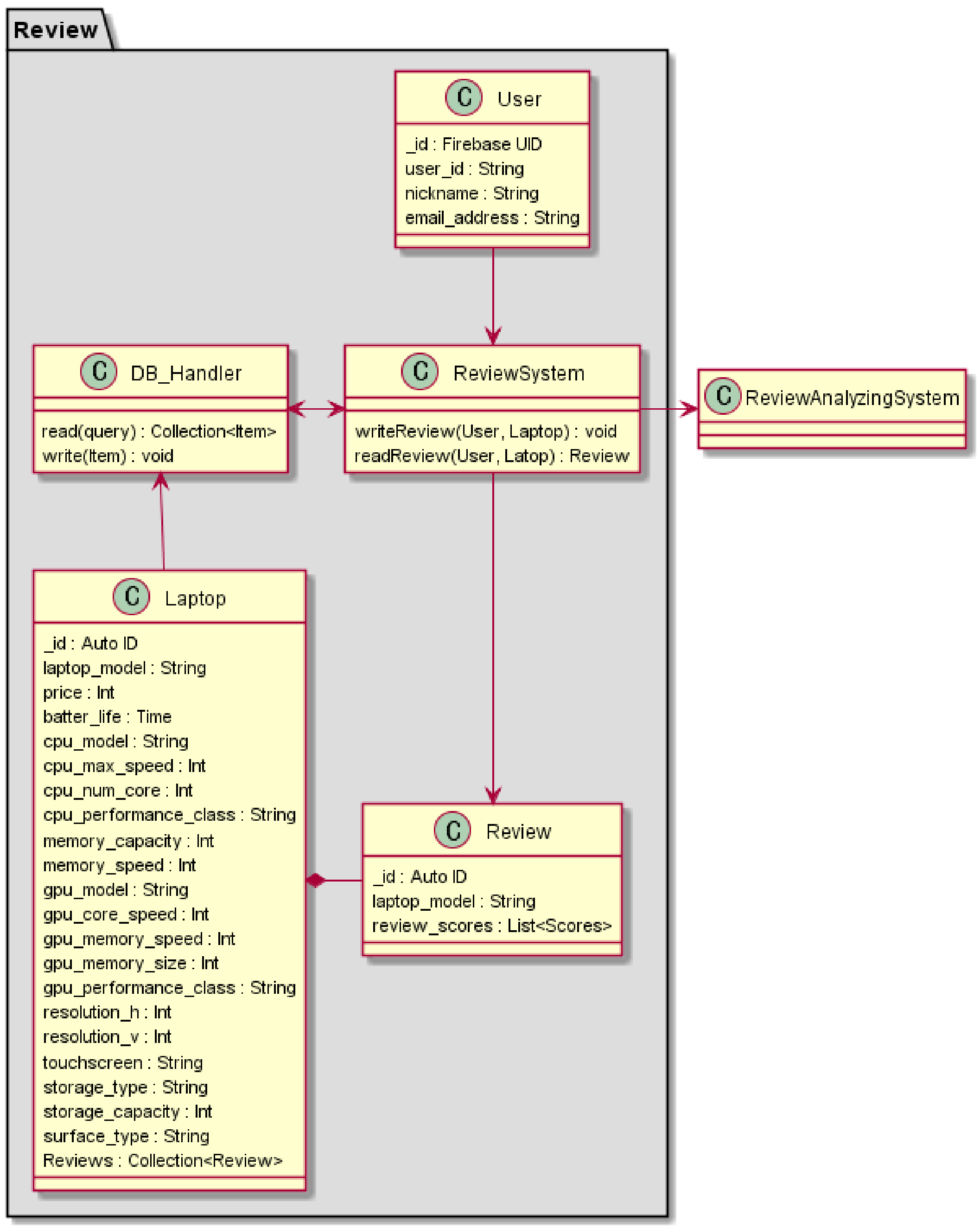
Interface to communicate with DB. The specific laptop object is fetched from the DB, and the Review object is added to the review collection of the laptop object and stored.

5.3.1.4. Model_Handler

It is an interface for machine learning. Tensorflow model is used through a custom model in the Firebase ML kit.

5.3.2. Review System

5.3.2.1. Class Diagram

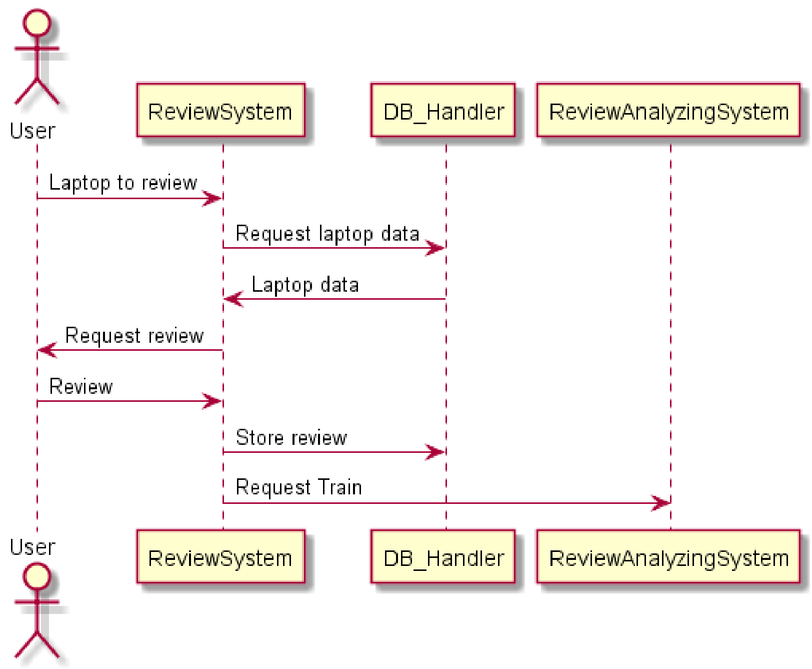


[Figure 19] Class diagram – Review system

- Class Description
  - ✓ **Review System Class**: This is the interface for controlling the review object.

When the user requests the write a review to system, the system adds the user's review to the laptop's review collection. When the user wants to see their review, it is taken from the laptop object's review collection.

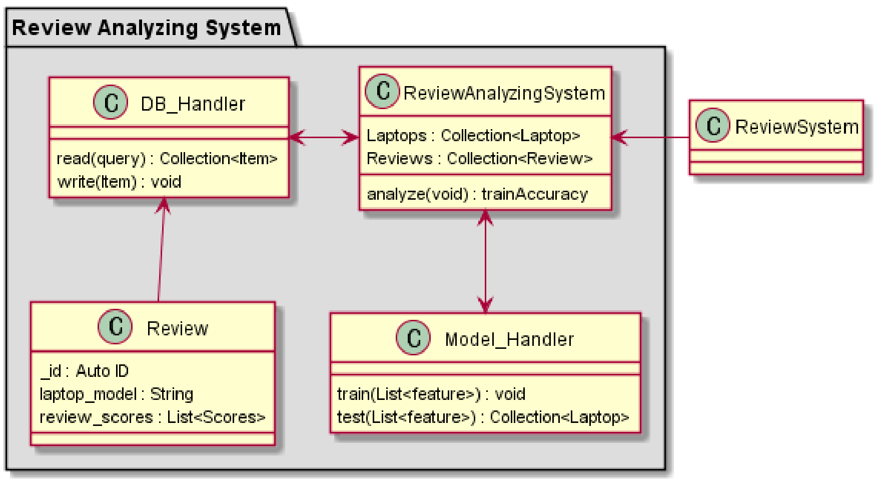
5.3.2.2. Sequence Diagram



[Figure 20] Sequence diagram – Review system

5.3.3. Review Analyzing System

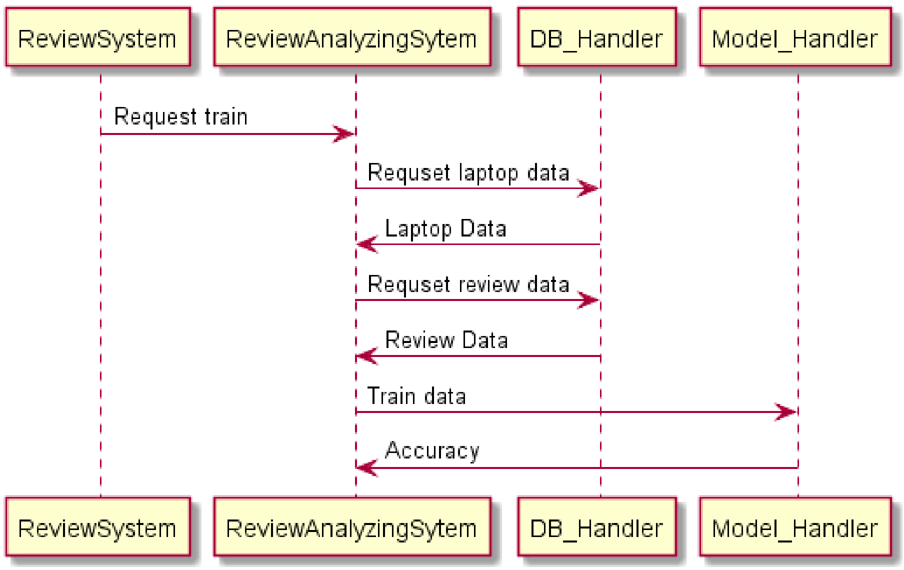
5.3.3.1. Class Diagram


Review Analyzing System
C DB_Handler
read(query) : Collection<Item>
write(Item) : void
C ReviewAnalyzingSystem
Laptops : Collection<Laptop>
Reviews : Collection<Review>
analyze(void) : trainAccuracy
C ReviewSystem
C Review
_id : Auto ID
laptop_model : String
review_scores : List<Scores>
C Model_Handler
train(List<feature>) : void
test(List<feature>) : Collection<Laptop>

[Figure 21] Class diagram – Review analyzing system

- Class Description
  - ✓ **Review Analyzing Class**: It is an interface for analyzing reviews. Called when a new review is added, request training to model_handler.

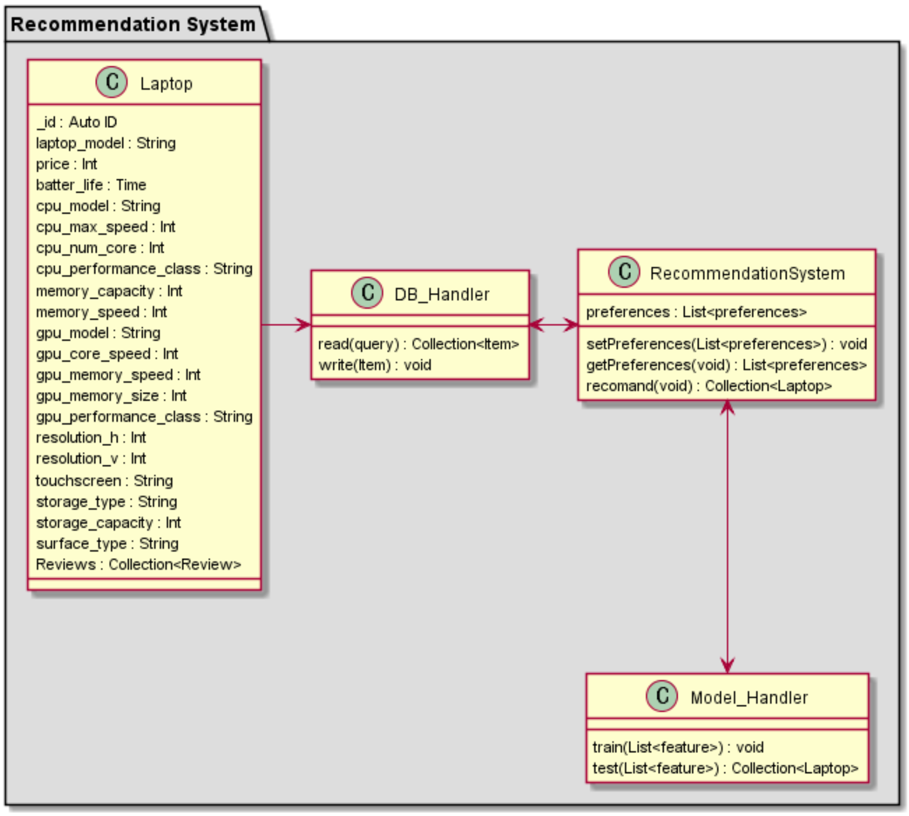
5.3.3.2. Sequence Diagram



[Figure 22] Sequence diagram – Review analyzing system

5.3.4. Recommendation System

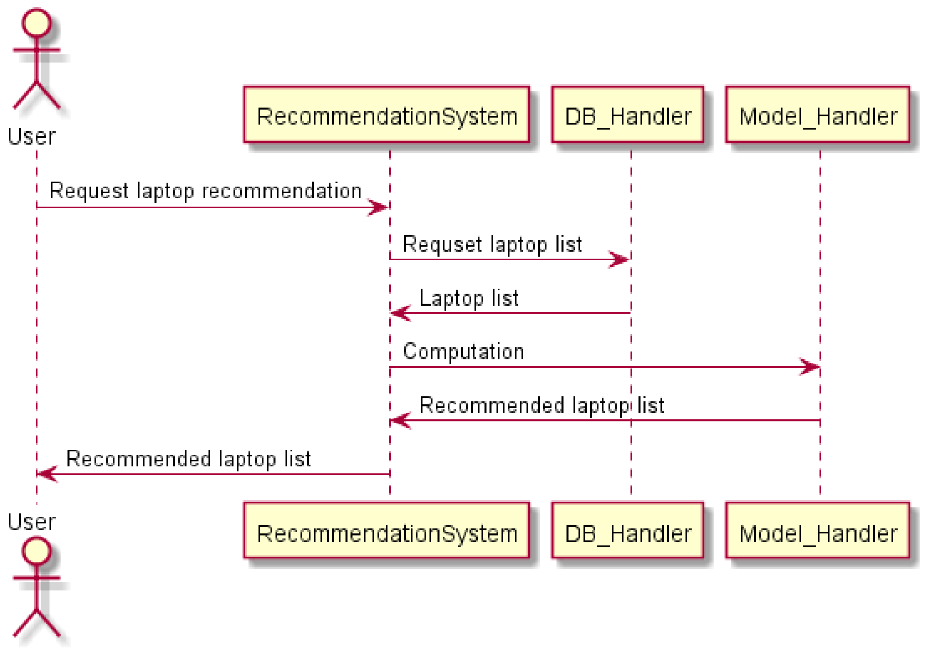
5.3.4.1. Class Diagram



[Figure 23] Class diagram - Recommendation system

- Class Description
  - ✓ **Recommendation System Class**: Interface for laptop recommendations. When a user enters a condition, a highly correlated laptop group is recommended through a model handler.

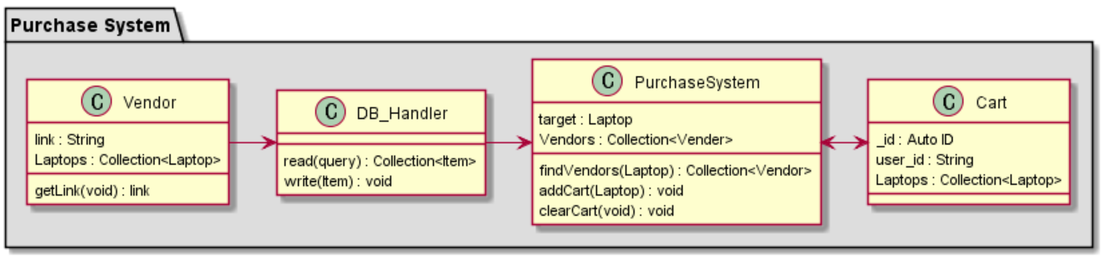
5.3.4.2. Sequence Diagram



[Figure 24] Sequence diagram – Recommendation system

5.3.5. Purchase System

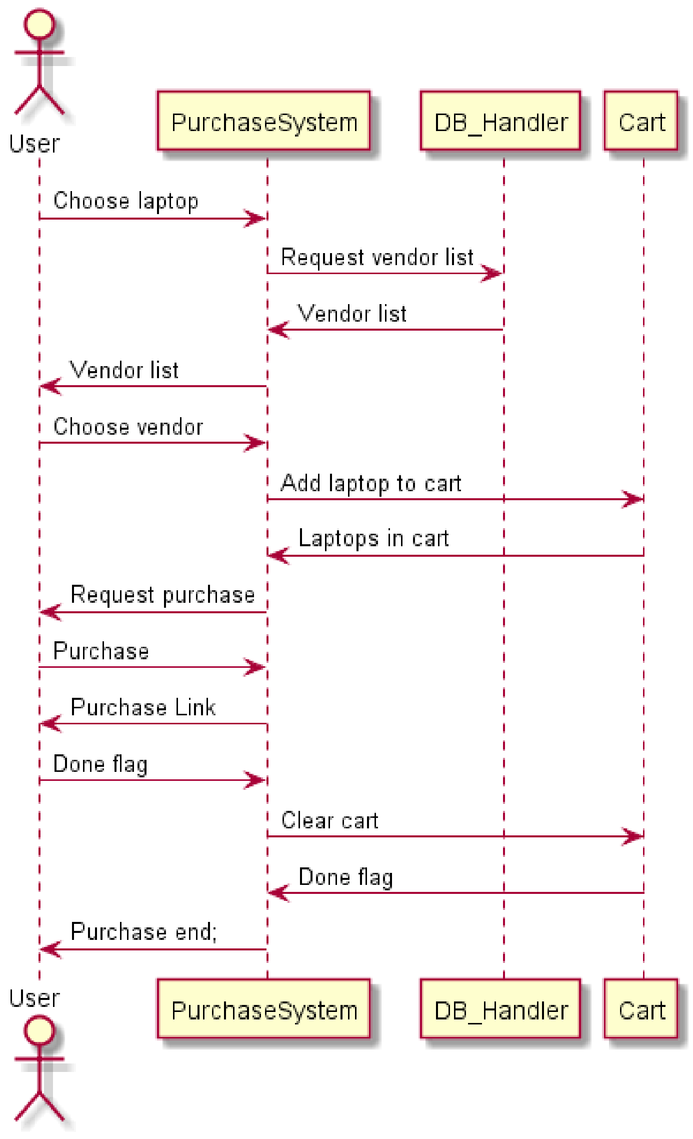
5.3.5.1. Class Diagram



[Figure 25] Class diagram – Purchase system

- Class Description
  - ✓ **Purchase System Class**: Interface for purchase. When the user enters the desired laptop, it connects to the seller selling the laptop and allows the user to put it in the cart.
  - ✓ **Cart Class**: Shopping cart for laptop

5.3.5.2. Sequence Diagram



[Figure 26] Sequence Diagram – Purchase system

6. Protocol Design

6.1. Objectives

This chapter describes what structures are used for the protocols which are used for interaction between each subsystem, especially iDecide application and the server. Also, this chapter describes how each interface is defined.

6.2. JSON

JSON(JavaScript Object Notation) is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format, with a diverse range of applications, such as serving as a replacement for XML in AJAX systems.

All Firebase Realtime Database data is stored as JSON objects. The database is like a cloud-hosted JSON tree. Unlike a SQL database, there are no tables or records. When data is added to the JSON tree, it becomes a node in the existing JSON structure with an associated key.

6.3. OAuth

OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords. This mechanism is used by companies such as Amazon, Google, Facebook, Microsoft and Twitter to permit the users to share information about their accounts with third party applications or websites.

6.4. Authentication

6.4.1. Register

- Request

[Table 1] Table of register request

Attribute	Detail	
Protocol	OAuth	
Request Body	Email	User email
	Request Token	Token for OAuth
	User	Basic information of the user

- Response

[Table 2] Table of register response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400 (Bad Request, overlap)	
Success Response Body	Access Token	Token for access
	Message	Success message
Failure Response Body	Success	Fail
	Message	Success message

6.4.2. Log In

- Request

[Table 3] Table of register request

Attribute	Detail	
Protocol	OAuth	
Request Body	Email	User email
	Request Token	Token for OAuth
	User	Basic information of the user

- Response

[Table 4] Table of register response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success	Access Token	Token for access

Attribute	Detail	
Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.5. Profile

6.5.1. Set Profile

- Request

[Table 5] Table of set profile request

Attribute	Detail	
Method	POST	
URI	/user/:id/profile	
Parameter	User	Basic information of the user
Header	Authorization	User authentication

- Response

[Table 6] Table of set profile response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 403 (Forbidden)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.5.2. Get Profile

- Request

[Table 7] Table of get profile request

Attribute	Detail	
Method	GET	
URI	/user/:id/user	
Request Body	-	-
Header	Authorization	User authentication

- Response

[Table 8] Table of get profile response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	User	User objects
Failure Response Body	Message	Empty

6.6. Search Laptop

6.6.1. Laptop Recommendation

- Request

[Table 9] Table of laptop recommendation request

Attribute	Detail	
Method	GET	
URI	/recommendation	
Parameter	Preference	Preference of the user
Header	Authorization	User authentication

- Response

[Table 10] Table of laptop recommendation response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Search_Result	Search result for the user preference that contain list of laptops
Failure Response Body	Message	Fail message

6.7. Cart

6.7.1. Get Cart

- Request

[Table 11] Table of get cart request

Attribute	Detail	
Method	GET	
URI	/user/:id/cart	
Request Body	-	-
Header	Authorization	User authentication

- Response

[Table 12] Table of get cart response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Cart	Cart objects
Failure Response Body	Message	Empty

6.7.2. Add Item to Cart

- Request

[Table 13] Table of add item to cart request

Attribute	Detail	
Method	POST	
URI	/user/:id/cart	
Parameter	Cart	Cart objects
Header	Authorization	User authentication

- Response

[Table 14] Table of add item to cart response

Attribute	Detail
Success Code	200 OK
Failure Code	HTTP error code = 400 (Bad Request, overlap)

Attribute	Detail	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.7.3. Delete Item of Cart

- Request

[Table 15] Table of delete item of cart request

Attribute	Detail	
Method	DELETE	
URI	/user/:id/cart	
Request Body	laptop_product_code	Cart object that user wants to delete

- Response

[Table 16] Table of delete item of cart response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.8. History

6.8.1. Get History

- Request

[Table 17] Table of get history request

Attribute	Detail	
Method	GET	
URI	/user/:id/history	
Parameters	-	-
Header	Authorization	User authentication

- Response

[Table 18] Table of get history response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Search_History	History objects
Failure Response Body	message	Empty

6.8.2. Add History

- Request

[Table 19] Table of add history request

Attribute	Detail	
Method	POST	
URI	/user/:id/history	
Parameter	Search_History	History object
Header	Authorization	User authentication

- Response

[Table 20] Table of add history response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400 (Bad Request, overlap)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.8.3. Delete History

- Request

[Table 21] Table of delete history request

Attribute	Detail	
Method	DELETE	
URI	/user/:id/history	
Request Body	query_id	History object that user wants to delete
Header	Authorization	User authentication

- Response

[Table 22] Table of delete history response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.9. Item

6.9.1. Get Item Information

- Request

[Table 23] Table of get item information request

Attribute	Detail	
Method	GET	
URI	/laptop/:id	
Parameters	laptop_product_code	laptop id
Header	Authorization	User authentication

- Response

[Table 24] Table of get item information response

Attribute	Detail
Success Code	200 OK
Failure Code	HTTP error code = 404 (Not Found)

Attribute	Detail	
Success Response Body	Laptop	Laptop object (Laptop name, picture, CPU, ..) Laptop_Review object
Failure Response Body	-	-

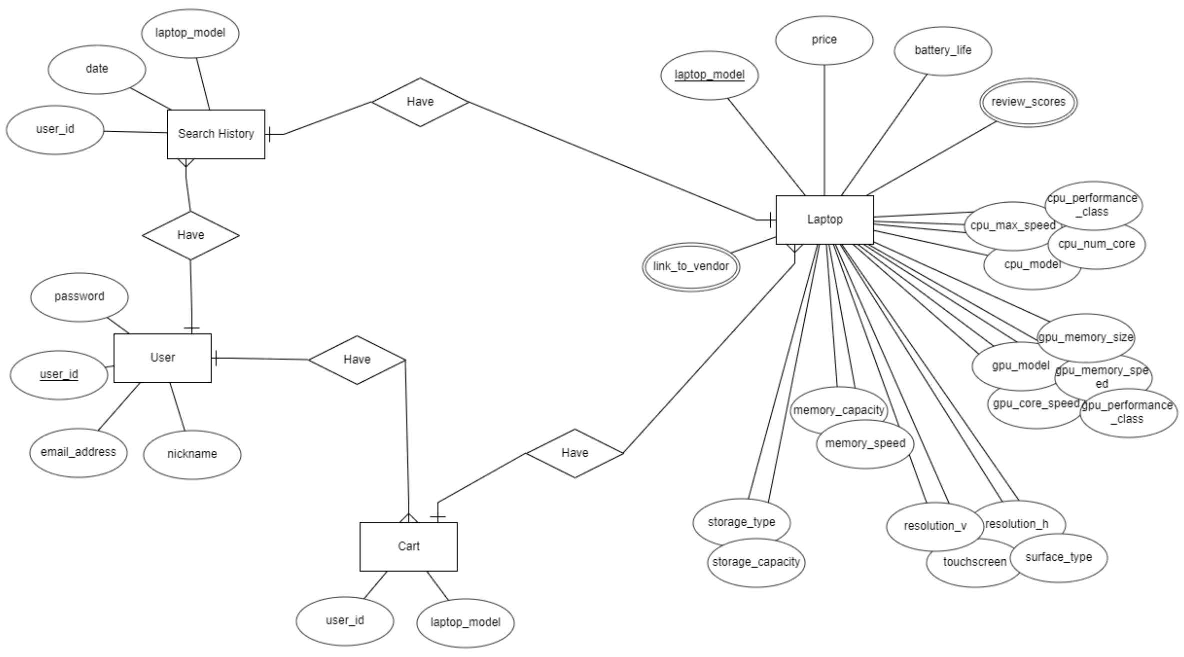
7. Database Design

7.1. Objectives

This section describes the system data structures and how these are to be represented in a database. It first identifies entities and their relationship through ER-diagram (Entity Relationship diagram). Then, it generates Relational Schema and SQL DDL (Data Description Language) specification.

7.2. ER Diagram

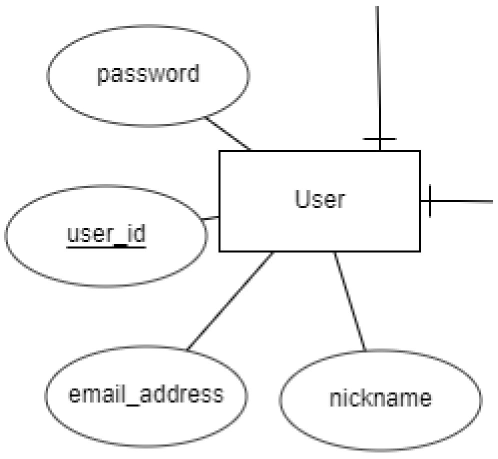
The system consists of four entities; User, Laptop, Search History, Cart. ER-diagram expresses each entity as rectangular and their relationship as rhombus. When an entity has multiple relationship with another entity, trident (three line) is used to indicate it. When an entity has just one relationship with another entity, the cross (two line) is used to indicate it. The attribute of an entity is expressed as an ellipse. The unique attribute which uniquely identifies an entity is underlined. If an entity has a number of same attributes, that attribute is expressed as ellipse with double border line.



[Figure 27] ER-diagram

7.2.1. Entities

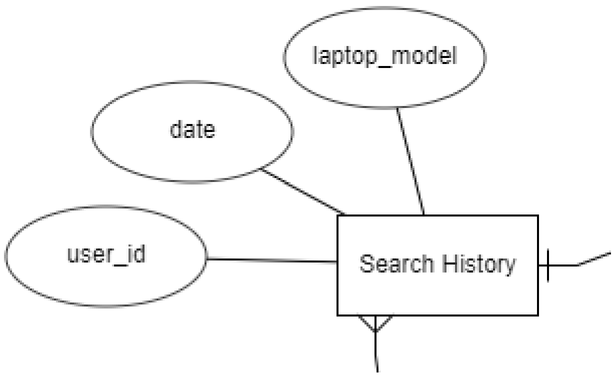
7.2.1.1. User



[Figure 28] ER diagram, Entity, User

User entity represents user of IDecide. It consists of user_id, password, nickname, email_address and user_id attribute is primary key. It can have multiple Search History and multiple Cart.

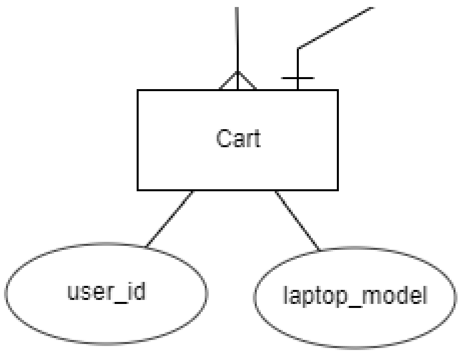
7.2.1.2. Search History



[Figure 29] ER diagram, Entity, Search History

Search History represents the search history of a user. It can have relationship with a user and a laptop and date indicates the date when user searches. There is no primary key. It can be found by the composite key which includes user_id.

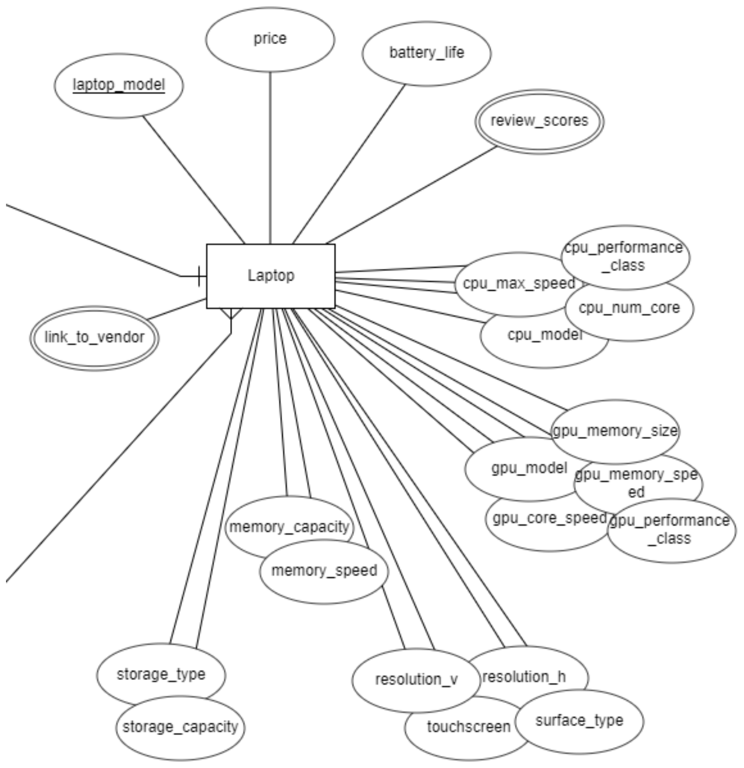
7.2.1.3. Cart



[Figure 30] ER diagram, Entity, Cart

Cart is similar to Search History. But it does not have the added date.

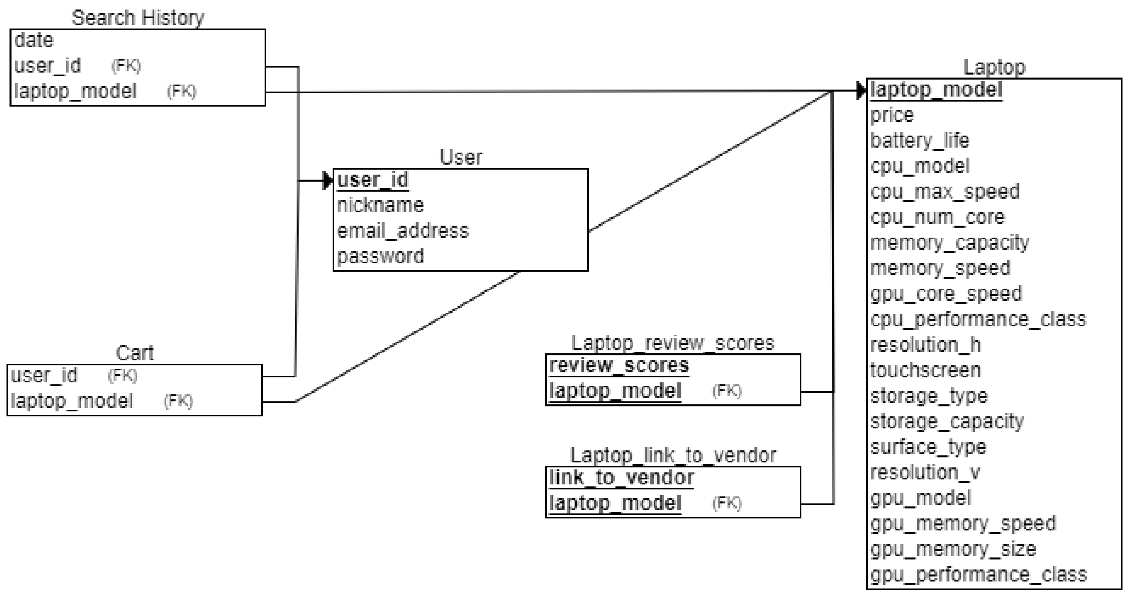
7.2.1.4. Laptop



[Figure 31] ER diagram, Entity, Laptop

Laptop includes necessary information to recommendation. The closely related attributes are tightly coupled. Slaptop_model is primary key. price is the lowest price available. battery_life is average battery life. There are cpu_x which are cpu-related attribute and gpu_x, memory_x, storage_x. resolution_v and resolution_h are vertical pixel and horizontal pixel respectively. touchscreen indicates whether manipulating the screen with touch is supported or not. surface_type can be the either value “Matte” which means good visibility in all conditions, or “Glossy” which means more visibility but poor visibility in brightly lit environments. review_scores and link_to_vendor can have multiple values.

7.3. Relational Schema



[Figure 32] Relational Schema

7.4. SQL DDL

7.4.1. User

```
CREATE TABLE User
(
  user_id INT NOT NULL,
  nickname INT NOT NULL,
  email_address INT NOT NULL,
  password INT NOT NULL,
  PRIMARY KEY (user_id)
);
```

7.4.2. Cart

```
CREATE TABLE Cart
(
  user_id INT NOT NULL,
  laptop_model INT NOT NULL,
  FOREIGN KEY (user_id) REFERENCES User(user
_id),
  FOREIGN KEY (laptop_model) REFERENCES Lap
top(laptop_model)
);
```

7.4.3. Search_History

```
CREATE TABLE Search_History
(
  date INT NOT NULL,
  user_id INT NOT NULL,
  laptop_model INT NOT NULL,
  FOREIGN KEY (user_id) REFERENCES User(user
_id),
  FOREIGN KEY (laptop_model) REFERENCES Lap
top(laptop_model)
);
```

7.4.4. Laptop

```
CREATE TABLE Laptop
(
   laptop_model INT NOT NULL,
   price INT NOT NULL,
   battery_life INT NOT NULL,
   cpu_model INT NOT NULL,
   cpu_max_speed INT NOT NULL,
   cpu_num_core INT NOT NULL,
   memory_capacity INT NOT NULL,
   memory_speed INT NOT NULL,
   gpu_core_speed INT NOT NULL,
   cpu_performance_class INT NOT NULL,
   resolution_h INT NOT NULL,
   touchscreen INT NOT NULL,
   storage_type INT NOT NULL,
   storage_capacity INT NOT NULL,
   surface_type INT NOT NULL,
   resolution_v INT NOT NULL,
   gpu_model INT NOT NULL,
   gpu_memory_speed INT NOT NULL,
   gpu_memory_size INT NOT NULL,
   gpu_performance_class INT NOT NULL,
   PRIMARY KEY (laptop_model)
);
```

7.4.5. Laptop_review_scores

```
CREATE TABLE Laptop_review_scores
(
  review_scores INT NOT NULL,
  laptop_model INT NOT NULL,
  PRIMARY KEY (review_scores, laptop_model),
  FOREIGN KEY (laptop_model) REFERENCES Lap
top(laptop_model)
);
```

7.4.6. Laptop_link_to_vendor

```
CREATE TABLE Laptop_link_to_vendor
(
  link_to_vendor INT NOT NULL,
  laptop_model INT NOT NULL,
  PRIMARY KEY (link_to_vendor, laptop_model),
  FOREIGN KEY (laptop_model) REFERENCES Lap
top(laptop_model)
);
```

8. Testing Plan

8.1. Objectives

This chapter illustrates plans on testing which has three major sub-groups of development testing, release testing, and user testing. These tests are crucial in that they detect potential errors and defects of the product and ensure flawless operation and stable release of a product to the market and customers.

8.2. Testing Policy

8.2.1. Development Testing

Development testing is carried out mainly for synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce potential risks of software development and save time and costs.

In this phase, since the software has not undergone enough tests, it can be unstable, and components can collide with each other. Therefore, static code analyzing, data flow analyzing, peer code review, unit testing need to be carried out in this stage. Through these processes, we are mainly focusing on achieving 1) performance which defines identity of our software, 2) reliability for ensuring safe and failure-free operation, and 3) security.

8.2.1.1. Performance

Since the recommendation algorithm take most time-consuming operation in our system, it is crucial for developers to shorten the time for filtering and presenting the outcome of recommendation to users. As specified in the recommendation specification, the system should give users the result within 5 seconds. We would prepare test cases on various preferences and evaluate the speed of the recommendation function, and improve the flow of code regarding the recommendation algorithm and communication with the server.

8.2.1.2. Reliability

In order for the system would operate safely without failure, the sub-components and units comprising the system should operate and be connected correctly first. Therefore, we need to undergo development testing from unit developing stage and check failure iteratively while each unit is integrated to the system.

8.2.1.3. Security

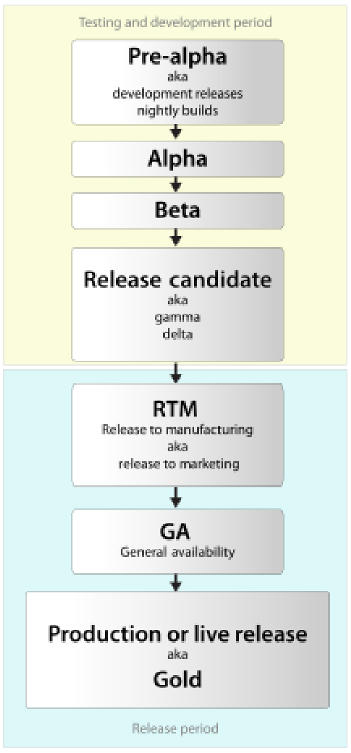
Securing information of users and the system is a crucial matter to be handled by developers. Regardless of the value of the information, it should be protected from unwanted visitors to the system.

In order to achieve security of the system, we can access a nearly finished version of the app, identify security issues and write the report through manual code review. Also, we can take advantage of other mobile app security testing services provided by Ostorlab, Appvigil, etc., which would let developers know overlooked vulnerable points in the application.

8.2.2. Release Testing

One of the most critical parts of any software development project is to release the product to the market, and customers. A technically good software can go wrong due to a wrong way of release. Therefore, release testing is inevitable for better connection between the product and the market. Release testing is testing a new version of a software/application to verify that that the software can be released in a flawless, so it doesn't have any defects on its working. It should be carried out before release.

Based on Software Release Life Cycle, testing is generally initiated from the 'Alpha' version, where a basic implementation of the software is completed. We would start development testing in Alpha version, and release Beta version for the further testing including user and release testing.



[Figure 33] Software Release Life Cycle

After Beta version is released, we would get feedback from actual users as well as developers.

8.2.3. User Testing

We should set up possible scenario and realistic situation that can proceed necessary user tests. We assume that there would be 30 users for iDecide application. After setting this situation, we would distribute Beta version of iDecide application to them and collect user reviews while carrying out our own use cases test with android emulator.

8.2.4. Testing Case

Testing case would be set according to 3 fundamental aspects of the application–function(interaction), performance, and security. We would set 5 test cases from each aspect (15 cases in total) and proceed testing on iDecide application and would make an evaluation sheet.

9. Development Plan

9.1. Objectives

This chapter illustrates the technologies and environment for the development of the application.

9.2. Frontend Environment

9.2.1. Adobe Photoshop

[Figure 34] Adobe Photoshop logo

It is a raster graphics editor developed and published by Adobe Inc. for Windows and macOS. This program would provide aesthetical layout and icons for improved user experience during our project.

9.2.2. Adobe Xd

[Figure 35] Adobe Xd logo

It is an all-in-one UX/UI solution program for designing web and mobile app. It provides interactive prototypes in various platforms, which facilitates communication among teammates and enables prompt reflection of feedbacks.

9.2.3. Android Studio

[Figure 36] Android Studio logo

It is the official integrated development environment for Google's Android operating system, and it is the most fundamental environment for developing iDecide application. Supporting various programming languages and featuring an intelligent code editor equipped with IntelliJ IDEA interface, Android Studio enables developers to make code more easily and accurately.

For frontend development, we can build a structure of the application, by setting visible actions to be followed according to user interaction in the application using XML layouts.

9.3. Backend Environment

9.3.1. Github

[Figure 37] Github logo

It provides hosting for software development version control using Git. It enables teammates to develop a single project together, resulting in easy integration of components. We are now using GitHub for developing iDecide application and controlling version of it.

9.3.2. Firebase

[Figure 38] Firebase logo

It supports development of mobile and web applications by providing various features such as cloud storage, real-time database, machine learning kit, etc.. Among them, we would use real-time database feature for managing data of users, laptops, and etc. Thanks to real-time database, data is synchronized across all the clients connected to this database. It means all clients can share a single real-time database instance and receive updated data with automated update.

9.3.3. Android Studio

[Figure 39] Android Studio logo

We can add additional features by connecting the application to external APIs, and connect XML files to activities of the application. In addition, we connect the application to DB server for the control of data.

9.4. Constraints

The system will be designed and implemented based on the contents mentioned in this document. Other details are designed and implemented by selecting the direction preferred by the developer, but the following items are observed.

- Use the technology that has already been widely proven.
- Laptop search speed should not exceed 4 seconds.
- Avoid using technology or software that requires a separate license or pays for royalty. (Exclude this provision if this is the only technology or software that the system must require.)
- Decide in the direction of seeking improvement of overall system performance.
- Decide in a more user-friendly and convenient direction
- Use open source software whenever possible
- Consider the system cost and maintenance cost
- Consider future scalability and availability of the system
- Optimize the source code to prevent waste of system resources
- Consider future maintenance and add sufficient comments when writing the source

code

- Develop with Windows 10 environment and Android Studio whose build tools version is 29.0.3
- Develop with minimum Android version 6.0 (API 23) and target Android version 29
- Emulate the system using Android version 10 (API 29)

9.5. Assumptions and Dependencies

All systems in this document are written on the assumption that they are designed and implemented based on Android devices and open source. Therefore, all contents are written based on the Android operating system with minimum API version 23 and may not be applied to other operating systems or versions.

10.Supporting Information

10.1. Software Design Specification

This software design specification was written in accordance with the IEEE Recommendation (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016).

10.2. Document History

[Table 25] Document History

Date	Version	Description	Writer
2020/05/20	0.1	Style and overview	Kyungyeon Park
2020/05/23	1.0	Addition of 8, 9	Seungrok Yoon
2020/05/23	1.1	Addition of 7	Hogi Min
2020/05/23	1.2	Addition of 6	Kyungyeon Park
2020/05/24	1.3	Addition of 1, 2.1, 2.2	Sara B.Zaki
2020/05/24	1.4	Addition of 2.3, 3	Allan Tazhenov
2020/05/24	1.5	Addition of 5	Hanwoul Lee
2020/05/24	1.6	Addition of 4	Kyungyeon Park

Date	Version	Description	Writer
2020/05/24	1.7	Revision of 2.2, 3, 9.4, 9.5	Kyungyeon Park
2020/05/24	1.8	Addition of 2.2.4, 2.2.5	Sara B.Zaki
2020/05/24	1.9	Revision of structure	Kyungyeon Park