Bilkent University

Department of Computer Engineering

Senior Design Project

Team ID: T2331

RecoModa

Detailed Design Report

Güven Gergerli 21803393 guven.gergerli@ug.bilkent.edu.tr

Hakan Gülcü 21702275 hakan.gulcu@ug.bilkent.edu.tr

Nasuh Dinçer 21702933 nasuh.dincer@ug.bilkent.edu.tr

Tarık Buğra Karali 21703937 bugra.karali@ug.bilkent.edu.tr

Zülal Nur Hıdıroğlu 21903125 nur.hidiroglu@ug.bilkent.edu.tr

Supervisor: Shervin Rahimzadeh Arashloo

Innovation Expert: Muhammed Naci Dalkıran

23.02.2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

1. Introduction

1.1 Purpose of the System

Searching for outfit combinations and finding the best-fitting clothes online can be time-consuming and challenging. According to research that investigates people's shopping behavior, almost %42.8 of the UK spend more than 7 hours online shopping, and shoppers browse different websites to find the best product. [1] RecoModa wants to optimize time spent during online shopping by a more accurate recommendation system of outfit combinations according to the user's taste, suggesting the product with the correct body size and allowing the shopper to see the products of different brands. RecoModa is a mobile-based shopping application that acts like social media. It is a platform where shoppers can follow other users and buy their combinations. Overall, it aims to decrease the waste of time as much as possible by offering every product with the correct body sizes.

The innovation planned in this application is to assist shoppers who want to browse different products of different brands with an accurate recommendation system in less time spent. The application will suggest the products with correct body sizes to the user since each brand may not obey the standardized body-size table. In this offering, combining social media and shopping applications to ease finding the best product with the correct body sizes is intended. The innovation type of RecoModa is incremental innovation because RecoModa adds new functionalities to the services currently in the market.

In this report, the design goals of RecoModa are listed below. In the following sections, the current software architecture and proposed software architecture of RecoModa are discussed. Later, subsystem services and test cases are explained in a detailed way. Lastly, consideration of various factors in engineering design and teamwork details are discussed.

1.2 Design goals

1.2.1 Usability:

- It should be easy for a new user to adapt to the application interface
- Users should operate all functionalities such as creating a post, buying product, commenting, etc. through their mobile application without any technical knowledge
- The application should suggest the products which match with user’s body size correctly
- It is essential that the application runs smoothly and efficiently in order.

1.2.2 Performance:

- The processes that need high computing power should be handled on the server side so that the program does not drain the battery.

1.2.3 Reliability:

- There should not be any program crash or data loss or any failure with payment
- If a failure occurs, the mean time to fix the failure should be minimum
- The average time between two failures should be maximum

1.2.4 Marketability:

- RecoModa should have a user-friendly user interface
- The application acts like social media, and it gives users to follow other people's combines. If influencers use the application, the range of people using this app should increase.

1.2.5 Extendability:

- The implementation of the application should allow adding new features and functionalities in the future, such as other distributing services and messaging.

1.2.6 Security:

- The sensitive information of the user, such as passwords and credit card information, should be saved in an encrypted format.
- By using a stateless session (JSON Web Token), all data does not need to be saved in a database on the server side like cookies. It only exists on the client side. It eliminates the CSRF attacks [2].

1.2.7 Scalability:

- The prototype of the application should be able to handle 50.000 users and 2000 concurrent users.

1.2.8 Maintainability:

- The RecoModa should have %90 maintainabilities in a day which means the component should be repaired maximum within a day.

1.2.9 Flexibility:

- The application should adapt to future changes and requirements such as messaging etc.

1.2.10 Modularity:

- The application should be divided into smaller modules which ensures software development manageability.

1.2.11 Aesthetics:

- The application should create attractiveness for customers and positive emotions through the design of the front end.

1.3 Definitions, acronyms, and abbreviations

- API: It stands for an application programming interface. APIs are tools that enable software programs to communicate with one another using protocols. Software applications, platforms, and services are frequently integrated using APIs.
- AWS: The term refers to Amazon Web Services. Businesses can get cloud-based services from AWS. Customers can create, deploy, and manage their applications and services in the cloud using these services.

- HTTP: It stands for HyperText Transfer Protocol. It enables clients, such as web browsers, to ask servers for resources and get the needed information back in return. The World Wide Web's functionality depends on HTTP.
- JWT: It stands for JSON Web Token. It is a standardized way of securely transmitting information between different parties in a compact and self-contained JSON object. JWT is commonly used for authentication and authorization in web applications and APIs.
- KNN: It stands for K-Nearest Neighbors. KNN is a simple machine-learning algorithm used for both classification and regression tasks. It works by finding the K nearest data points to a new input and using their labels or values to predict the label or value for the new input.
- CNN: CNN stands for Convolutional Neural Network.An image classification and object recognition task can be performed using CNN, an algorithm commonly used in deep learning.

1.4 Overview

RecoModa is a mobile shopping application with a strong recommendation system that combines the features of both shopping and social media. Like social media, users can share products or combinations with others and be followed by others. These shared combinations can be registered by the users in parts or completely, and they will be able to buy the whole combination or the parts they want with a single click with a fingerprint scan. Using information such as body information, weight, height, and shoulder width, appropriate sizes will be determined and then suggested. This information will be requested to be updated at regular intervals, and new suggestions will be made according to the changes. Suggestions that may be different in different brands will reduce the size problem. The most important feature of RecoModa is its strong personalized recommendation system. For this, every data that can be obtained from the user will be used. To enhance and elaborate the recommendation system, this data includes likes of users, shares of users followed, products previously bought, etc. There will be suggestions generated all over the system. RecoModa will constantly update its database as bots scrape data from other clothing companies.

2. Current software architecture

There are several shopping websites and Android and iOS applications available in the current system. Customers may access clothing, glassware, and other items through these applications. But, unlike RecoModa, such applications lack practical recommendation algorithms for body shapes and items.

Trendyol is one of Turkey's most popular shopping apps and has a superior suggestion system to other Turkish apps. It considers the user's prior decisions, visited brands, and so on. However, it does not propose the right product based on the user's body measurements.

Trendyol's system is highly similar to Amazon's, which is popular worldwide. It evaluates all actions to make decisions about the user's preferences, such as likes, prior purchases, and so on. Amazon, like Trendyol, does not have a mechanism for suggesting items based on body size.

3. Proposed Software Architecture

3.1 Overview

RecoModa is a mobile shopping application with a strong recommendation system that combines the features of both shopping and social media. Like social media, users can share products or combinations with others and be followed by others. These shared combinations can be registered by the users in parts or completely, and they will be able to buy the whole combination or the parts they want with a single click with a fingerprint scan. Using information such as body information, weight, height, and shoulder width, appropriate sizes will be determined, and then suggested. This information will be requested to be updated at regular intervals, and new suggestions will be made according to the changes. Suggestions that may be different in different brands will reduce the size problem. The most important feature of RecoModa is its strong personalized recommendation system. For this, every data that can be obtained from the user will be used. To enhance and elaborate the recommendation system, this data includes likes of users, shares of users followed, products previously bought, etc. There will be suggestions generated all over the system. RecoModa will constantly update its database as bots scrape data from other clothing companies.

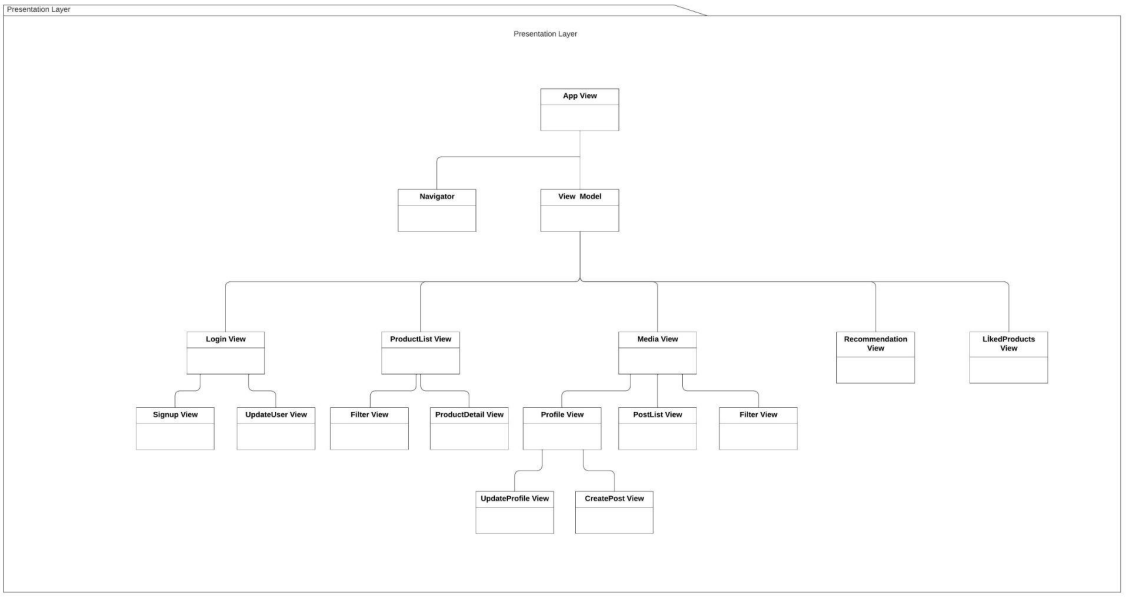
3.2 Subsystem Decomposition

According to the goals of our application's design, the RecoModa system follows the 3-Tier architecture that includes the Presentation Tier, Logic Tier, and Data Tier. Since RecoModa will be a mobile application, the 3-Tier architectural style is appropriate to reduce the interaction between subsystems. This style provides flexibility to develop and update each layer with minimum dependency on other layers. In this architectural style, the client does not have direct access to the database, so it is more difficult for a client to obtain unauthorized data, and it provides security. Because of the increased modularity, one tier can be changed or replaced more easily without affecting the other tier.

Figure[1] subsystems interact

The diagram İn the figure 1 shows how the subsystems interact with each other. The general overview of the system is given in the figure XX. The Presentation Layer represents the client-side. Every interaction which the user performs belongs to this layer (UI). The Logic Layer represents the server-side. Every backend operation belongs to this layer. The Data Layer represents the storage side of the application.

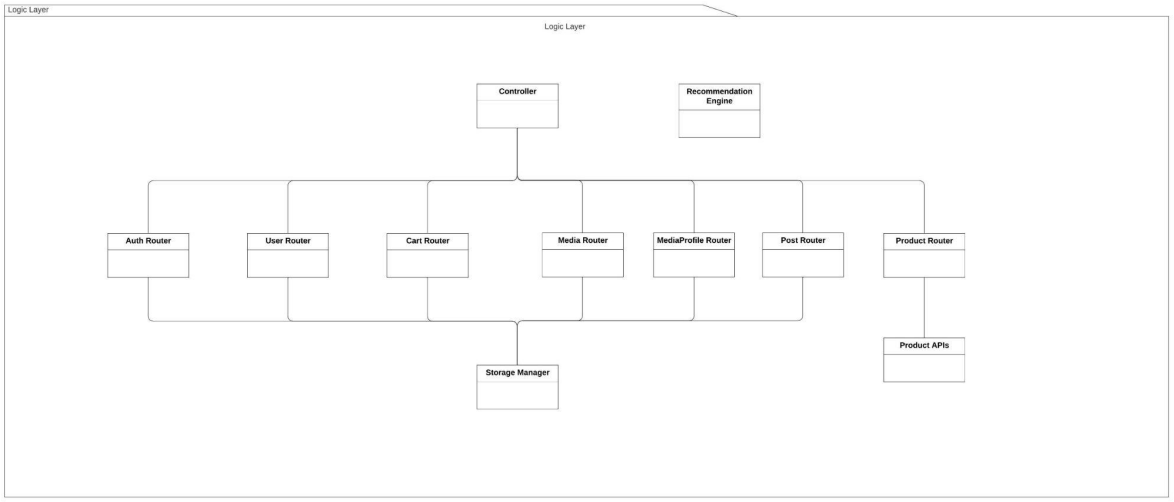
3.2.1 Presentation Layer



Figure[2] presentation layer

The Interface of the application is represented by the presentation layer, as seen in Figure 2. This layer handles all interactions with the user. With API calls, all user requests are passed directly to the logic layer. Then the Logic Layer passes the results of these calls back to the Presentation Layer.

3.2.2 Logic Layer



Figure[3] Logic Layer

The Logic Layer which is seen in Figure 3, includes all of the Presentation Layer's backend functionality. The connection between the presentation layer and the logic layer is made through API endpoints. The machine learning model for recommendation is also placed here. The data that is sent from the Logic Layer to the Data Layer via the Storage Manager.

3.2.3 Data Layer

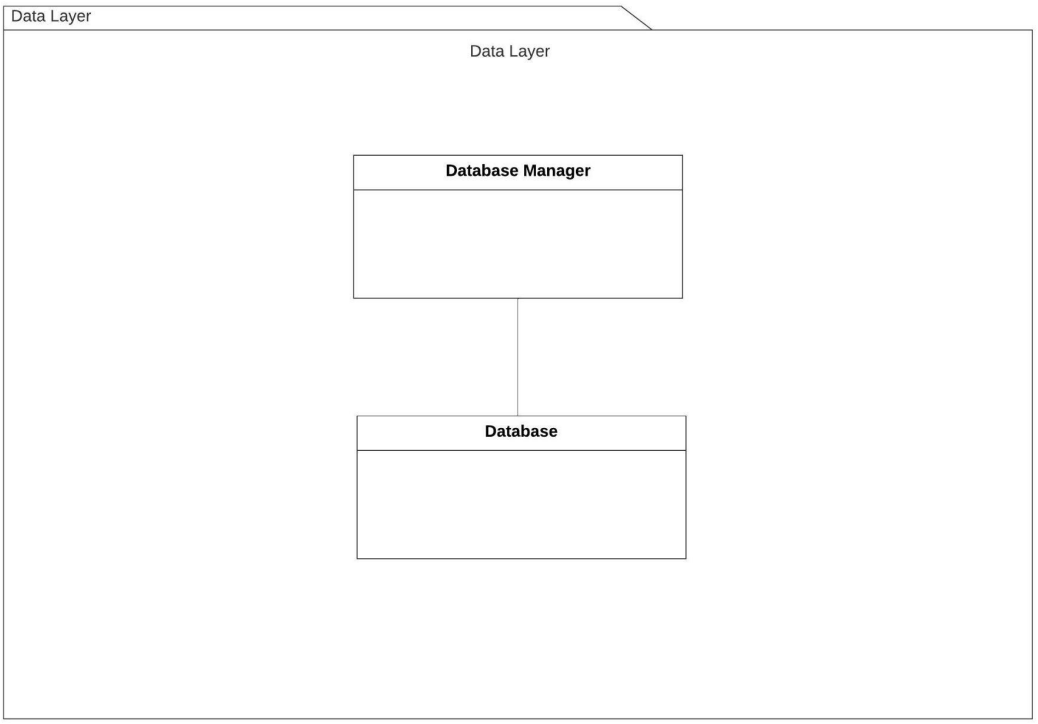
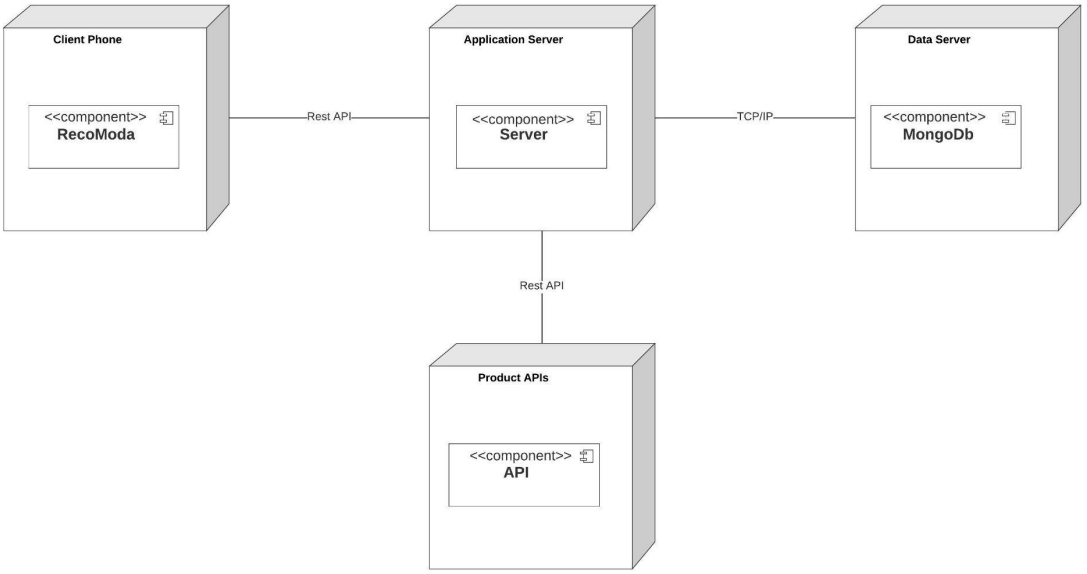


Figure [4] Data Layer

The Data Layer which is seen in Figure 4, represents the database of the application. The database contains the user information, posts, social media information and shopping cart. Since MongoDb is a real time database, the data can be retrieved or updated instantly.

3.3 Hardware/Software Mapping



Figure[5] Hardware/Software Mapping

RecoModa is a react native mobile application that works with Android-powered devices. Users can use their Android smartphones or tablets to communicate with the application's servers. Via their Android devices, users can execute the application's features.

The deployment diagram given in Figure 5 illustrates the relationship between the application's subsystems. Server subsystem includes the REST API that makes the HTTP request processing with Client Phone and external APIs. The application backend functionalities are executed in the Server subsystem. The server subsystem accesses the necessary data from the database and sends a response back to the presentation layer in order to meet the requests from the client phone. Moreover, to retrieve product information, the server sends API requests to the external Product APIs.

3.4 Persistent Data Management

In the RecoModa, users' personal data and information will be stored in MongoDB. Since MongoDB is a real-time database, users can update their information, and changes are updated instantly. User information should also be persistent after a successful login until the user logs out. This part of the persistent data management will be done using JSON web tokens and redux libraries. Both of the libraries will use the local file system of the user machine. After a successful login, token information will be stored in local storage. Then, it sends this token

information in all requests that request authorization (get, post, put, delete…). For every incoming request, the server verifies the token and checks whether the user can access it. Additionally, thanks to the "Redux" library, the user information, including social media and shopping cart information, will be saved locally on users' devices and can be instantly accessed as needed. This will improve the application's operational effectiveness. Since the products are fetched from external APIs, product information is not stored in the database, so the required field for data storage is minimized.

3.5 Access Control and Security

3.5.1 Authentication

In the RecoModa, there are two types of users which are regular users and admin. Since there are different types of users, access to the pages should be controlled. The admin pages should be accessible only to admins. An open standard called JSON Web Token (JWT) outlines a compact and self-contained method for securely transferring data between parties as a JSON object. The fact that this information is digitally signed allows for verification and security. JWTs can be signed using a public/private key pair using RSA or ECDSA or a secret (with the HMAC algorithm) [3]. In order to prevent undesired access, JSON Web Tokens are used. The JWT will be included in each request after the user logs in, enabling access to the routes, services, and resources that are authorized with that token.

3.5.2 Deep Linking

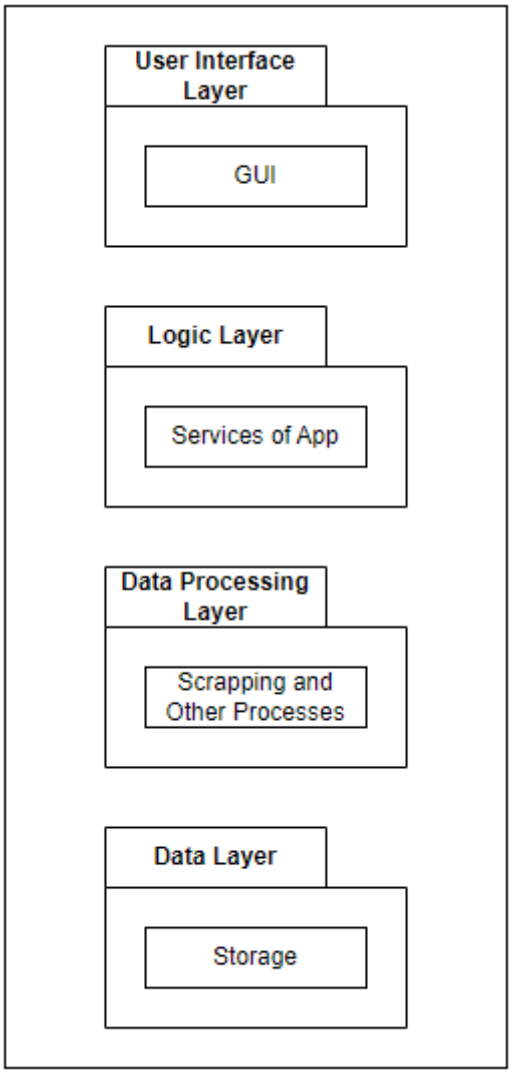
Deep linking presents a special vulnerability for mobile apps. Deep linking allows data to be sent from an external source directly to a native program [4]. Nothing prevents a malicious application from joining the same scheme and hijacking your deep link in order to gain access to the information it holds. While sending app:/products/1 is not harmful, sending tokens raises security issues. To avoid this problem, we set up a universal link (HTTP or HTTPS) login interface and use a random identifier to locally authenticate the incoming login token.

3.5.3 Sensitive Information

Anyone inspecting the app bundle might access anything contained in the code in plain text. Using sensitive information in the code can be insecure. In order to solve this problem, the dotenv library is used. Sensitive informations are defined in the .env file and their instances are used in the code. Also, before personal information is saved to the database, the information is encrypted by using the bcrypt library.

4. Subsystem services

4.1 User Interface Layer



Figure[6] User Interface Layer

The user interface layer is the layer that welcomes the user using the application and allows the user to have a more enjoyable application experience. Thanks to this layer, the user communicates with the system. In this layer, the commands coming from the user are transferred to the lower layers and the results obtained for the user are obtained with various parameters. The user reaches the results visually and the application experience is completed.

4.2 Feature Layer

4.2.1. User authentication and management

User authentication and management is a basic security layer in the RecoModa application. This layer is responsible for ensuring that only authorized

users are given access to sensitive resources and data. In this layer, the user encounters various components. In the user authentication phase, users are authenticated before granting access to the system. It can include a number of techniques such as passwords, biometric authentication or security tokens. In the access control component, it is responsible for controlling access to resources within the system based on the permissions assigned to each user. Overall, the user authentication and management layer plays a critical role in maintaining the security and integrity of computer systems and applications and must be implemented carefully and carefully to provide the highest level of protection possible.

4.2.2 Photo upload and storage

The photo upload and storage tier is a specific type of file upload and storage tier that focuses on managing the upload and storage of photos. This layer is widely used in web applications that allow users to upload and share photos, such as social media platforms, photo sharing sites, and e-commerce sites selling photo products, and will also be used in RecoModa. The photo upload and storage layer consists of several components. In the file upload process component, it is responsible for executing the upload process, which includes accepting the photo file from the user's device, verifying the file type and size, and then transferring the file to the server. The image processing component is responsible for processing the uploaded photo file, including resizing, cropping, compressing, and optimizing the image for storage and viewing. The storage management component is responsible for storing uploaded photo files locally or in a cloud-based storage service and managing access to the files. In addition to these core components, the photo upload and storage tier may also include additional features such as metadata extraction, tagging, search functionality, and social media integration. Overall, the photo upload and storage tier is a critical component of the RecoModa app that allows users to upload and share photos and should be implemented with a focus on security, scalability and usability. It is important to ensure that uploaded photos are properly verified, processed and stored to prevent unauthorized access or data loss.

4.2.3 Image processing and analysis:

The image processing and analysis layer is an important component of the RecoModa application that deals with visual data such as image and video processing applications. This layer is responsible for performing various operations on images, such as improving image quality, detecting and tracking objects, recognizing patterns and features, and analyzing image data for insights and information. The preprocessing component is responsible for preparing the image for analysis by removing noise, correcting distortions, and improving image quality. The feature extraction component is responsible for determining the key features and properties of the image, such as edges, vertices, textures, and colors. The classification and recognition component is responsible for categorizing images based on their content, such as identifying specific objects, scenes or people in the

image. In addition to these core components, the image processing and analysis layer may also include additional features such as machine learning algorithms, deep learning models, and computer vision libraries. Overall, the image processing and analysis layer is a critical component of any system that deals with visual data and should be implemented with a focus on accuracy, efficiency, and scalability. It is important to ensure that the processing and analysis techniques used are suitable for the particular application and that the data is appropriately secured and protected, and RecoModa will evaluate this data in the most appropriate way.

4.2.4 Recommendation engine:

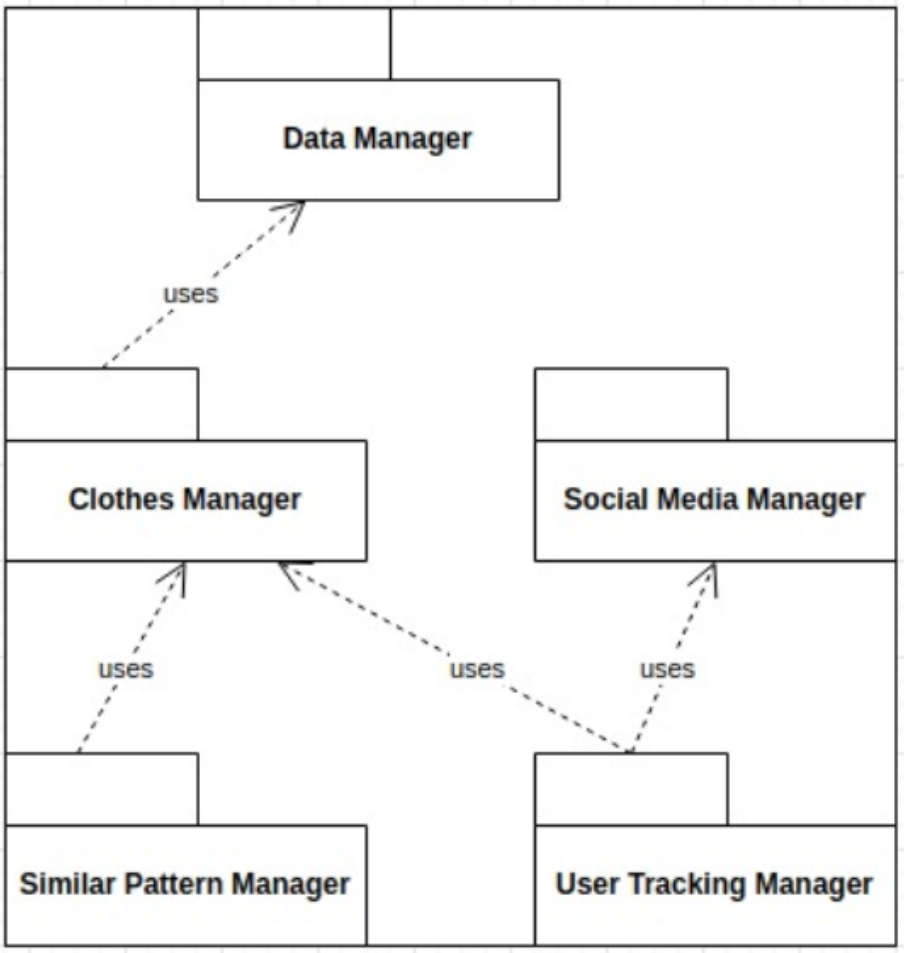
The recommendation engine layer is a component of a system designed to provide personalized recommendations to users based on their interests, behaviors, and preferences. This layer will be used in our RecoModa application to recommend products, services or content that may be of interest to the user on e-commerce, social media and content platforms. The data collection component is responsible for collecting data about the user's behavior such as browsing history, search queries, and purchase history. The data storage and processing component is responsible for storing and processing data collected from various sources such as databases, log files and APIs. The recommendation algorithm component is responsible for analyzing data and generating recommendations based on the user's preferences, behavior, and other relevant factors. In addition to these core components, the recommendation engine layer may include additional features such as machine learning models, collaborative filtering techniques, and content-based filtering methods. Overall, the recommendation engine layer is a critical component of any system aimed at providing personalized recommendations to users and should be implemented with a focus on accuracy, relevance, and user experience. It is important to ensure that the recommendations provided are tailored to the user's needs and preferences, and that the data used to generate the recommendations is appropriately secured and protected.

4.2.5 Payment and checkout:

The checkout and payment tier is a critical component of the RecoModa application that enables customers to purchase products and services safely and efficiently. This layer is responsible for managing the payment process, including collecting payment information, processing payments, and confirming orders. The payment processing component is responsible for processing the payment transaction, including authorizing the payment, transferring funds, and notifying the seller and customer of the payment status. The order confirmation component is responsible for confirming the order and providing the customer with an order summary containing information about the products or services purchased, payment amount, and delivery details. Overall, the payment and payment layer is a critical component of any e-commerce application that accepts online payments and should be implemented with a focus on security, reliability and user experience. It is

important to ensure that the payment information collected is properly secured and protected, and that the checkout process is streamlined and efficient to prevent cart abandonment and loss of sales.

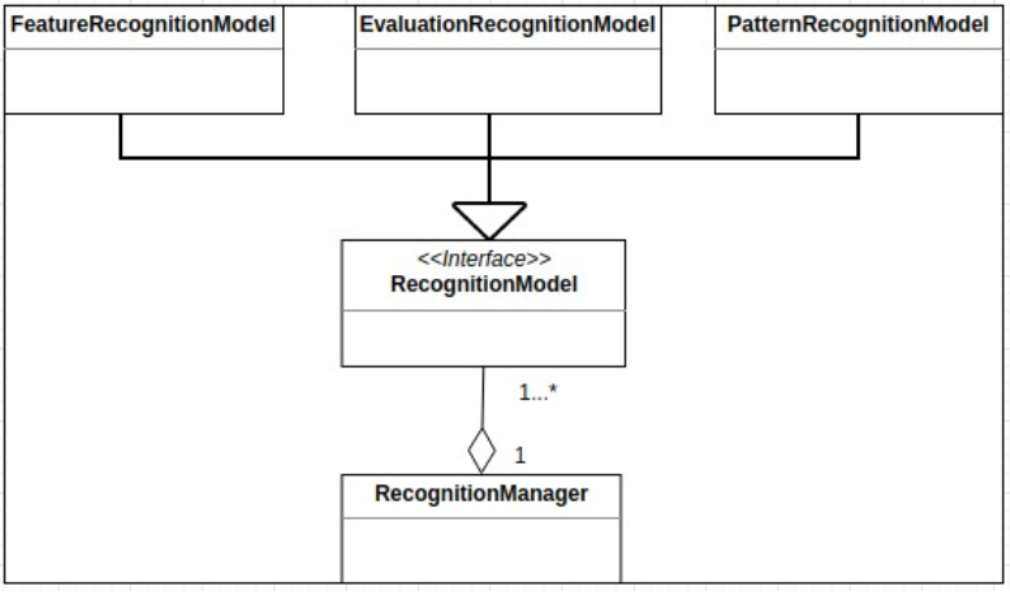
4.3 Data Processing Layer



Figure[7] Data Processing Layer

The Data Processing layer data shows how the scraped data will be processed and where it will be used. In short, the clothes manager processes the scraped data and then uses the similar pattern and user tracking manager. In addition, the user tracking manager tracks the behavior of users in the social media section.

4.3.1 Clothes Manager



Figure[8] Clothes Manager

The scraped data is basically recognized according to 3 different properties. With the Feature Recognition model, features such as price, size and category are determined. With Evaluation Recognition, features such as interpretation and scoring are determined. With Pattern Recognition, the pattern in the product is categorized and intended to be used. The features set here are used by Similar Pattern Manager and User TrackingManager. Similar Pattern uses more colors and patterns, while User Tracking uses features such as category and size.

4.3.2 Social Media Manager

Social Media Manager is related to the social media part of the application. Users can see, like and save people's posts like social media. There is also a chance to examine the clothes. So, examining the digital footprint is essential. A lot of data will be tracked, such as the posts that the users like, the products they save, the links they click and the comments they make. These will also be processed and used by the User Tracking manager.

4.3.3 Similar Pattern Manager

The information obtained about all products will be used by the similar pattern manager. It is a core part of the application. It aims to create accurate and appropriate results using machine learning. It will generate content-based results using K-Nearest Neighbors (KNN) and Convolutional Neural Network (CNN) methods. By Clothes Manager, the products will already be separated according to their properties. Therefore, methods such as searching for similarities between products in the same category will be used.

4.3.4 User Tracking Manager

User Tracking Manager works using both Social Media Manager and Clothes Manager. Since our main purpose in the application is to provide personalized results, the most important thing is to track user movements. User movements tracked in Social Media Manager will be used to determine the posts that the user will encounter. The products in the shipments work in sync with the Clothes Manager and aim to put the most similar products in front of them. In other words, the most accurate results are aimed with a model that works in a hybrid way.

4.4 Database Layer

Database Layer is actually the most basic stone of the system. It is linked to all other layers and this layer is very important for the application to work properly. It has Backup Service, Storage Service and Query services. Backup Service periodically backs up the database. Storage Service records the necessary data. Query Services ensures that the correct results are sent to the Backend. It will work very closely with Data Processing in particular. It is vital for the system to update the data at the right times and regularly, and to take the right actions. That is, synchronous communication will be established so that these can be avoided. Operations such as user verification, information changes, storing images, keeping product information, user movements are also done with the Database Layer. It can then be used by other Layers.

5. Test Cases

5.1 Unit Test

In the software development cycle, unit testing is a testing technique used to ensure that a unit of code will behave as intended and that the tested code will return the same result each and every time the test is run. White-box testing includes unit testing, which is often carried out by creating test cases that put specific methods, functions, and classes to the test. The primary components or functions checked by QA professionals using unit testing are the accuracy of the code, code coverage, implementation and maintenance of coding standards, and verification of the functionality provided by the piece of code. By executing unit tests, we identify bugs or errors at the early stages, isolate a unit of code and verify that it works properly, test software application’s functionality and reduce cost and save time.

Unit testing of RecoModa is performed using functional testing techniques. The goal of the functional testing method is to confirm that each app feature functions in accordance with the specifications of the software. In order to do this testing, sample inputs are used, outputs are recorded, and the expected outputs are

compared to the actual outputs. For unit tests, we use Jest. Jest is built into React, which makes Jest a simple, out-of-the-box solution for React Native unit testing [5]. Since Jest is a framework, we simply download it and add Jest as a dev dependency to the project. After that, we write test files and test the functions.

5.1.1 Procedure of Unit Test

1. We decide which functionality of the product needs to be tested.
2. We provide testable input data for functionality in accordance with the specifications.
3. We decide acceptable output parameters in accordance with the given specifications.
4. We execute test cases by using Jest.
5. We compare actual output from the test with the predetermined output values. This shows whether the system is performing as it expected.

5.1.2 Use Case Scenarios of Unit Test

5.1.2.1 User tries to write invalid username in username textfield.

Test Case id: 5121
Test Type: Security
Test Case Objective: Checks the username textfield.
Test Case Procedure:

- The username field requires a minimum of 6 characters, a maximum of 16 characters, numbers(0-9), letters(a-z, A-z), and special characters (only underscore, period, hyphen allowed). It cannot be left blank. The username must begin with a character. It cannot include special characters.
- Enter an invalid username to the username textfield.

Expected Result: A warning message that contains the reason why the given username is invalid should be displayed.
Priority: Critical

5.1.2.2 User tries to write valid username in username textfield.

Test Case id: 5122
Test Type: Security
Test Case Objective: Checks the username textfield.
Test Case Procedure:

- The username field requires a minimum of 6 characters, a maximum of 16 characters, numbers(0-9), letters(a-z, A-z), and special characters (only underscore, period, hyphen allowed). It cannot be left blank. The username must begin with a character. It cannot include special characters.
- Enter a valid username to the username textfield.

Expected Result: Any warning message should not be displayed.

Priority: Critical

5.1.2.3 User tries to write invalid password in password textfield.

Test Case id: 5123
Test Type: Security
Test Case Objective: Checks the password textfield.
Test Case Procedure:

- The password field requires a minimum of 6 characters, a maximum of 16 characters, numbers (0-9), letters (a-z, A-Z), and all special characters. It cannot be empty.
- Enter an invalid password to the password textfield.

Expected Result: A warning message that contains the reason why the given password is invalid should be displayed.
Priority: Critical

5.1.2.4 User tries to write valid password in password textfield.

Test Case id: 5124
Test Type: Security
Test Case Objective: Checks the password textfield.
Test Case Procedure:

- The password field requires a minimum of 6 characters, a maximum of 16 characters, numbers (0-9), letters (a-z, A-Z), and all special characters. It cannot be empty.
- Enter a valid password to the password textfield.

Expected Result: Any warning should not be displayed.
Priority: Critical

5.1.2.5 User tries to write valid credentials in login template.

Test Case id: 5125
Test Type: Security
Test Case Objective: Checks the login button.
Test Case Procedure:

- Enter a valid username.
- Enter a valid password.

Expected Result: Login button should be enabled.
Priority: Critical

5.1.2.6 User tries to write invalid credentials in login template.

Test Case id: 5126
Test Type: Security
Test Case Objective: Checks the login button.
Test Case Procedure:

- Enter an invalid username.

- Enter an invalid password.

Expected Result: Login button should be disabled.
Priority: Critical

5.1.2.7 User tries to reset password with an invalid email address.

Test Case id: 5127
Test Type: Security
Test Case Objective: Checks the password reset functionality.
Test Case Procedure:

- Enter an invalid email address.
- Click the reset password button.

Expected Result: A warning message should be displayed.
Priority: Critical

5.1.2.8 User tries to reset password with a valid email address.

Test Case id: 5128
Test Type: Security
Test Case Objective: Checks the password reset functionality.
Test Case Procedure:

- Enter a valid email address.
- Click the reset password button.

Expected Result: “Reset link is sent” message should be displayed .
Priority: Critical

5.1.2.9 User tries to create a post without an image.

Test Case id: 5129
Test Type: Functionality
Test Case Objective: Checks the create post functionality.
Test Case Procedure:

- Login to the application.
- Click the create post button.
- Enter a valid description.
- Enter a valid link or links.
- Click share button.

Expected Result: A warning message beside the image box should be displayed. Share button should be disabled.
Priority: Minor

5.1.2.10 User tries to create a post without a description.

Test Case id: 51210
Test Type: Functionality
Test Case Objective: Checks the create post functionality.
Test Case Procedure:

- Login to the application.
- Click the create post button.
- Enter valid images or an image.
- Enter a valid link or links.
- Click share button.

Expected Result: A warning message beside the description box should be displayed. Share button should be disabled.
Priority: Minor

5.1.2.11 User tries to create a post without a link.

Test Case id: 51211
Test Type: Functionality
Test Case Objective: Checks the create post functionality.
Test Case Procedure:

- Login to the application.
- Click the create post button.
- Enter valid images or an image.
- Enter a valid description.
- Click share button.

Expected Result: A warning message beside the link box should be displayed. Share button should be disabled.
Priority: Minor

5.1.2.12 User tries to create a post without a link.

Test Case id: 51212
Test Type: Functionality
Test Case Objective: Checks the create post functionality.
Test Case Procedure:

- Login to the application.
- Click the create post button.
- Enter valid images or an image.
- Enter a valid description.
- Click share button.

Expected Result: A warning message beside the link box should be displayed. Share button should be disabled.
Priority: Minor

5.1.2.13 User tries to create a post with valid credentials.

Test Case id: 51213
Test Type: Functionality
Test Case Objective: Checks the create post functionality.
Test Case Procedure:

- Login to the application.

- Click the create post button.
- Enter valid images or an image.
- Enter a valid description.
- Enter a valid link or links.
- Click share button.

Expected Result: Share button should be enabled.
Priority: Minor

5.1.2.14 User tries to increase the quantity of the product in the shopping cart.

Test Case id: 51214
Test Type: Functionality
Test Case Objective: Checks the shopping cart functionality.
Test Case Procedure:

- Login to the application.
- Click the shopping cart button.
- Click the + button placed beside the product quantity.

Expected Result: Quantity and total price should be updated.
Priority: Major

5.1.2.15 User tries to decrease the quantity of the product in the shopping cart.

Test Case id: 51214
Test Type: Functionality
Test Case Objective: Checks the shopping cart functionality.
Test Case Procedure:

- Login to the application.
- Click the shopping cart button.
- Click the - button placed beside the product quantity.

Expected Result: Quantity and total price should be updated. If quantity were 1, the product should disappear from the cart.
Priority: Major

5.1.2.16 User tries to cancel the product in the shopping cart.

Test Case id: 51215
Test Type: Functionality
Test Case Objective: Checks the shopping cart functionality.
Test Case Procedure:

- Login to the application.
- Click the shopping cart button.
- Click the x button placed beside the product quantity.

Expected Result: Total price should be updated. The product should be deleted from the shopping cart list.
Priority: Major

5.2 Integration Test

After every team member completes their part, the integration test will be executed. Integration testing guarantees that individual code units and components can function cohesively as a whole. Integration testing checks whether all the modules are properly communicating with each other. We use the bottom-up testing method during the integration tests. Using this approach, the lower-level modules are tested first, and the higher-level modules are tested later.Since it is easier to locate and identify flaws and troubleshooting takes less time, we choose the bottom-up method. For unit tests, we use Jest and Enzyme. Since Jest does not require any configuration for mock functions, again we choose Jest. In addition to Jest, we use Enzyme. Enzyme is a testing library that offers a jQuery-like API for traversing the react component tree and the ability to render React components in memory or to the DOM [6]. Since RecoModa uses external APIs, Enzyme library is suitable for our application.

5.2.1 Procedure of Integration Test

- We design test scenarios, test cases, and test scripts.
- We run tests after integration of units or modules are completed.
- We report errors and fix them.
- We retest functionalities after detected errors are fixed.
- We repeat this process until all problems are resolved.

5.2.2 Use Case Scenarios of Integration Test

5.2.2.1 User tries to log in to RecoModa with valid credentials.

Test Case id: 5221
Test Type: Security
Test Case Objective: Check the interface link between the Login and other pages.
Test Case Procedure:

- Enter login credentials and click on the Login button.
- Verify that you have navigated to the main page.

Expected Result: If successful, user information should be placed on the right of the top-bar on every page. Otherwise, a warning message should be displayed.
Priority: Critical

5.2.2.2 User tries to like a product placed on the product cart.

Test Case id: 5222
Test Type: Functionality
Test Case Objective: Check the interface link between the Detailed Product and Liked Product modules.
Test Case Procedure:

- Login to the application.
- Click on a post to see detailed post information
- Click the like button placed on the product cart.
- Verify that the liked product should be added to the liked product page.

Expected Result: If successful, the liked product should be added to the liked product page. Otherwise, there will be a warning to the user that the post has not been liked due to a connection problem.
Priority: Minor

5.2.2.3 User tries to like a product placed on the product cart.

Test Case id: 5223
Test Type: Functionality
Test Case Objective: Check the interface link between the Detailed Product and Liked Product modules.
Test Case Procedure:

- Login to the application.
- Click on a post to see detailed post information
- Click the like button placed on the product cart that is already liked.
- Verify that the liked product should be deleted from the liked product page.

Expected Result: If successful, the Liked product should be deleted from the liked product page. Otherwise, there will be a warning to the user that the post has not been removed due to a connection problem.
Priority: Minor

5.2.2.4 User tries to follow a user's media profile.

Test Case id: 5224
Test Type: Functionality
Test Case Objective: Check the interface link between the Follower List and Media Profile modules.
Test Case Procedure:

- Login to the application.
- Click on a user's media profile
- Click the follow user button placed on the user's media profile.
- Click on your own user media profile.
- Verify that the user's media profile is added to the following list on your own user media profile page.

Expected Result: If successful, the user should be added to the follower list. Otherwise, there will be a warning to the user that the post has not been removed due to a connection problem.
Priority: Minor

5.2.2.5 User tries to unfollow a user's media profile.

Test Case id: 5225
Test Type: Functionality
Test Case Objective: Check the interface link between the Follower List and Media Profile modules.
Test Case Procedure:

- Login to the application.
- Click on a user's media profile.
- Click the unfollow user button placed on the user's media profile that the user already follows.
- Click on your own user media profile.
- Verify that the user's media profile is removed from the following list on your own user media profile page.

Expected Result: That user should be deleted from the follower list.
Priority: Minor

5.2.2.6 User adds product to user's shopping cart.

Test Case id: 5226
Test Type: Functionality
Test Case Objective: Check the interface link between the Cart and Product modules.
Test Case Procedure:

- Login to the application.
- Click on the shopping page.
- Click on the product cart.
- Click on the add button at the end of the cart. Generate error if not available in stocks.
- Verify that the product is shown in the user's shopping cart.

Expected Result: The product should be added to the user's cart.
Priority: Minor

5.2.2.7 User tries to post a new post.

Test Case id: 5227
Test Type: Functionality
Test Case Objective: Check the interface link between the Post and Media Profile modules.
Test Case Procedure:

- Login to the application.

- Click on the new post button in the media.
- Fill in the required areas to create a post.
- Upload the media to the post.
- Click the share button.

Expected Result: The post should be added to the user's media profile. If some of the required areas are not filled, or the image is not uploaded, the user will be given an error message.
Priority: Moderate

5.2.2.8 User tries to delete a post from my posts page

Test Case id: 5228
Test Type: Functionality
Test Case Objective: Check the interface link between the Post and Media Profile modules.
Test Case Procedure:

- Login to the application.
- Click on a post to see detailed post information
- Click on the delete button
- Verify that deleted posts should be deleted from My Posts page.

Expected Result: If successful, deleted posts should be deleted from the my posts page. Otherwise, there will be a warning to the user that the post has not been removed due to a connection problem.
Priority: Minor

5.2.2.9 User tries to update a post in the selected post's details page

Test Case id: 5229
Test Type: Functionality
Test Case Objective: Check the interface link between the Post and Media Profile modules.
Test Case Procedure:

- Login to the application.
- Click on a post to see detailed post information
- Click on the update post button
- Verify that updated posts should be updated user's media profile.

Expected Result: If the post is updated successfully, the post should be updated in the user's media profile. Otherwise, the system should give a warning message.
Priority: Minor

5.2.2.10 User clicks the search button on the Filter page.

Test Case id: 52210
Test Type: Functionality
Test Case Objective: Check the interface link between the Product and Filter modules.

Test Case Procedure:

- Login to the application.
- Click on the products page.
- Click on the filter button.
- Select some filters available.
- Click on the search button to see filtered results.
- Verify that the results shown in the products page are filtered correctly.

Expected Result: The products that suit with given filters should be listed in the products page.
Priority: Major

5.2.2.11 User enters a keyword in the search bar and click search button.

Test Case id: 52211
Test Type: Functionality
Test Case Objective: Verify that the user can search for a product they want or want to reach by entering a keyword.
Test Case Procedure:

- Login to the application.
- Click on the products page.
- Click on the search button.
- Enter the keyword wanted to search on the search bar.
- Click on the search button to see filtered results.
- Verify that the results shown in the products page are filtered correctly.

Expected Result: Products related to the keyword written to the application by the user should be displayed on the products page.
Priority: Minor

5.2.2.12 User selects a category from the filter menu.

Test Case id: 52212
Test Type: Functionality
Test Case Objective: Verify that the user can filter products by category.
Test Case Procedure:

- Login to the application.
- Click on the products page.
- Click on the filter button.
- Select the category wanted to be displayed.
- Click on the search button to see filtered results.
- Verify that the results shown in the products page are filtered correctly.

Expected Result: Products related to the selected category should be displayed on the product page.
Priority: Minor

5.2.2.13 User searches price range from the filter menu.

Test Case id: 52213
Test Type: Functionality
Test Case Objective: Verify that the user can filter products by price range.
Test Case Procedure:

- Login to the application.
- Click on the products page.
- Click on the filter button.
- Select the price range wanted to be displayed.
- Click on the search button to see filtered results.
- Verify that the results shown in the products page are filtered correctly.

Expected Result: Products within the selected price range should be displayed on the product page.
Priority: Minor

5.2.2.14 User sorts products with price sorting

Test Case id: 52214
Test Type: Functionality
Test Case Objective: Verify that the user can sort products by price.
Test Case Procedure:

- Login to the application.
- Click on the products page.
- Click on the sort button.
- Verify that the results shown in the products page are sorted based on the selected sorting option correctly.

Expected Result: Products should be displayed in ascending or descending order based on the selected sorting option.
Priority: Minor

5.2.2.15 User adds products to wishlist

Test Case id: 52215
Test Type: Functionality
Test Case Objective: Verify that the user can add a product to their wishlist.
Test Case Procedure:

- Sign in to the app.
- Click on the products page.
- Choose a product.
- Click the Add Wish List button.
- Verify that the product added to the user's wish list.

Expected Result: The product should be added to the user's wish list.
Priority: Minor

5.2.2.16 User remove products from wishlist

Test Case id: 52216
Test Type: Functionality
Test Case Objective: Verify that the user can remove a product from the wishlist.
Test Case Procedure:

- Sign in to the app.
- Click on the wishlist page.
- Identify a product that needs to be removed.
- Click on the remove button or icon next to the product.
- Verify that the product is no longer present in the user's wish list.

Expected Result: The specified product should be successfully removed from the user's wish list.
Priority: Minor.

5.2.2.17 User view order history

Test Case id: 52217
Test Type: Functionality
Test Case Objective: Verify that the user can view the order history.
Test Case Procedure:

- Sign in to the app.
- Click on the order history page or section.
- Verify that a list of the user's past orders is displayed.
- Click on a specific order to view details.
- Verify that the details of the selected order are displayed accurately.

Expected Result: The user should be able to view their order history and the details of each order accurately.
Priority: Major.

5.2.2.18 User comment for a product

Test Case id: 52218
Test Type: Functionality
Test Case Objective: Verify that the user can leave a review for a product.
Test Case Procedure:

- Sign in to the app.
- Click on the products page.
- Choose a product to comment.
- Click on the comments tab or section for that product.
- Click on the add review button.
- Enter the required information.
- Verify that the review is successfully submitted and displayed on the product's comment section.

Expected Result: The user should be able to leave a review for a product and the review should be successfully submitted and displayed on the product's reviews section.
Priority: Moderate.

5.2.2.19 User delete comment for a product

Test Case id: 52219
Test Type: Functionality
Test Case Objective: Verify that the user can remove a review for a product.
Test Case Procedure:

- Sign in to the app.
- Click on the products page.
- Choose a product that the user already has a comment on.
- Click on the comments tab or section for that product.
- Click on the edit button.
- Click on the Delete Review button.
- Verify that the review is successfully deleted and no longer displayed on the product's comment section.

Expected Result: The review should be removed from the product page and the user should receive a confirmation message.
Priority: Moderate.

5.2.2.20 User view profile information

Test Case id: 52220
Test Type: Functionality
Test Case Objective: Verify that the user can view their profile information.
Test Case Procedure:

- Sign in to the app.
- Click on the profile page.
- Verify that the user's profile information is displayed.
- Verify that the user can edit their profile information.

Expected Result: The user should be able to view their profile information and have the option to edit it if necessary.
Priority: Minor.

5.2.2.21 User edits and updates their own profile page

Test Case id: 52221
Test Type: Functionality
Test Case Objective: Check the interface link between the Profile and Edit Profile modules.
Test Case Procedure:

- Sign in to the app.
- Click on the Profile page.

- Verify that the user's profile information is displayed.
- Verify that the user can edit their profile information.

Expected Result: The profile information should be updated and reflected in the user's profile page.
Priority: Moderate.

5.2.2.22 User checks interface link profile to settings

Test Case id: 52222
Test Type: Functionality
Test Case Objective: Check the interface link between the Profile and Settings modules.
Test Case Procedure:

- Sign in to the app.
- Click on the Profile or Account page.
- Verify that there is a link or button to access the Settings page.
- Click on the Settings link or button.
- Verify that the user is redirected to the Settings page without any errors.
- Verify that the Settings page contains relevant options to manage the user's account settings.
- Verify that there is a link or button to access the Profile page from the Settings page.
- Click on the Profile link or button.
- Verify that the user is redirected to the Profile page without any errors or issues.
- Verify that the Profile page displays the user's profile information accurately.

Expected Result: The interface link between the Profile and Settings modules should work smoothly without any errors or issues. The user should be able to access both pages easily and navigate between them. The "Settings" page should contain relevant options to manage the user's account settings, and the "Profile" page should display the user's profile information accurately.
Priority: Critical

5.2.2.23 User checks interface link profile to notification

Test Case id: 52223
Test Type: Functionality
Test Case Objective: Check the interface link between the Profile and Notifications modules.
Test Case Procedure:

- Sign in to the app.
- Click on the profile page.
- Verify that there is a link or button to access the "Notifications" page.
- Click on the notifications link or button.

- Verify that the user is redirected to the "Notifications" page without any errors or issues.
- Verify that the notifications page contains relevant options to manage the user's notification preferences, such as enabling or disabling push notifications, email notifications.
- Verify that there is a link or button to access the profile page from the notification page.
- Click on the profile link or button.
- Verify that the user is redirected to the profile page without errors or issues.
- Verify that the profile page displays the user's profile information accurately.

Expected Result: The interface link between the Profile and Notifications modules should work smoothly without any errors or issues. The user should be able to access both pages easily and navigate between them.
Priority: Critical

5.2.2.24 User checks interface link cart to checkout

Test Case id: 52224
Test Type: Functionality
Test Case Objective: Check the interface link between the Cart and Checkout modules.
Test Case Procedure:

- Sign in to the app.
- Click on the Cart icon or link in the app.
- Verify that the user is redirected to the Cart page without error.
- Verify that the Cart page displays all the items that the user has added to their cart, along with the quantity and total cost.
- Verify that there is a button or link to access the Checkout page from the Cart page.
- Click on the Checkout button or link.
- Verify that the user is sent to the Checkout page without any errors or issues.
- Verify that the Checkout page displays all the items the user has added to their cart, along with the quantity and total cost.
- Verify that the Checkout page contains a form for users to enter their shipping and billing information.
- Verify that the Checkout page displays the total cost of the items along with any applicable taxes and shipping charges.
- Verify that there is a button or link to complete the checkout process.
- Click on the Complete Checkout button or link.
- Verify that the user is redirected to a confirmation page without any errors or issues.
- Verify that the confirmation page displays the details of the user's order, such as the items purchased, the total cost, and the shipping and billing information.

Expected Result: The interface link between the Cart and Checkout modules should work smoothly without any errors or issues. The user should be able to access both pages easily and navigate between them. The Cart page should display all the items that the user has added to their cart along with the quantity and total cost, and the "Checkout" page should contain a form for the user to enter their shipping and billing information. The confirmation page should display the details of the user's order accurately.
Priority: Critical

5.2.2.25 User checks interface link notification to mark read

Test Case id: 52225
Test Type: Functionality
Test Case Objective: Check the interface link between the Notifications and Mark Read modules.
Test Case Procedure:

- Sign in to the app.
- Verify that there is a link or button to access the Notifications page.
- Click on the Notifications link or button.
- Verify that the user is redirected to the Notifications page without any errors.
- Verify that the Notifications page contains relevant options to manage the user's notification preferences.
- Verify that there is a link or button to Mark Read on the Notification modules.
- Verify that the user can select Mark Read to selected Notification modules.
- Verify that the Notification modules that are selected as Mark Read do no longer displayed on the Notification page.

Expected Result: The user should be able to access the notification page and can easily navigate through it. The selected notification should be marked as read and no longer be displayed on the notification page.
Priority: Minor

5.2.2.26 User checks interface link search to product

Test Case id: 52226
Test Type: Functionality
Test Case Objective: Check the interface link between the Search and Product modules.
Test Case Procedure:

- Sign in to the app.
- Click on the search bar or icon in the app.
- Enter a keyword or phrase to search for a product.
- Verify that the app displays a product list that matches the search query.
- Click on a product from the search results.
- Verify that the user is redirected to the Product page without any errors or issues.

- Verify that the Product page displays the details of the selected product, such as the name, description, price, and images.
- Verify that there are options to add the product to the user's cart or wish list from the Product page.
- Verify that there is a link or button to go back to the search results from the Product page.
- Click on the link or button to return to the search results.
- Verify that the user is redirected to the search results page without any errors or issues.

Expected Result: The interface link between the Search and Product modules should work smoothly without any errors or issues. The user should be able to search for a product using keywords or phrases and view a list of products that match the search query. The user should be able to click on a product from the search results and be redirected to the Product page without errors or issues. The Product page should accurately display the selected product's details, and there should be options to add the product to the user's cart or wish list. The user should be able to return to the Product page's search results page without any errors or issues.
Priority: Critical

5.2.2.27 User tries to log in to RecoModa with valid credentials.

Test Case id: 5221
Test Type: Compatibility
Test Case Objective: Check the interface link between the Login and other pages.
Test Case Procedure:

- Enter login credentials and click on the Login button.
- Verify that you have navigated to the main page.

Expected Result: If successful, user information should be placed on the right of the top-bar in 1 second.
Priority: Critical

5.2.2.28 User tries to get a recommendation.

Test Case id: 5221
Test Type: Compatibility
Test Case Objective: Check the interface link between the Recommendation page and other pages.
Test Case Procedure:

- Enter login credentials and click on the Login button.
- Verify that you have navigated to the main page.
- Click on the recommendation page.
- Click on the get recommendation button.

Expected Result: If successful, recommendation list upload in 5 seconds..
Priority: Critical

5.3 End-to-End Test

End-to-end testing verifies that every feature works as intended from the user perspective. It aims to ensure that all the parts of the app are working correctly and that the app meets the user's requirements. By applying end-to-end testing the user interface, backend functionality, server communication, and database operations are tested. We use Jest for End-to-end testing.

5.3.1 Procedure of End-to-End Test

- We determine the requirements of the RecoModa.
- We create test scenarios that users can encounter while using the RecoModa.
- We develop test cases.
- We create test data.
- We create a test environment using Jest.
- We execute the test.
- We record and analyze the test results.
- We report errors and fix them.
- We retest functionalities after detected errors are fixed.
- We repeat this process until all problems are resolved.

5.3.2 Use Case Scenarios of End-to-End Testing

5.3.2.1 Users registers and logins into the system

Test Case id: 5321
Test Type: Security
Test Case Objective: Verify users can successfully register and log in to the mobile application.
Test Case Procedure:

- Click and enter the application.
- Verify that the Login page is on.
- Click on the "Register" button.
- Verify that the Registration page is on.
- Fill in the required areas in the registration page.
- Click on the “Submit” button.
- Verify that a success message that the account is created is displayed.
- Verify that the Login page is on.

- Enter the registered username and password into the login and click on the Login account button.
- Verify that the user is redirected to the main page and logs in successfully.

Expected Result: The user should register into the application and then use their credentials to Login into the system. The user then should be redirected to the main page.
Priority: Critical.

5.3.2.2 User views and adds products to shopping carts.

Test Case id: 5322
Test Type: Functionality
Test Case Objective: Verify users can view and select products.
Test Case Procedure:

- Login to the application.
- Click on the Products page.
- Verify that a list of available products is displayed.
- Click on a product details page.
- Verify that the product details page is displayed.
- Click on the Add cart button.
- Verify that a success message is displayed and that the product is added to the shopping cart.

Expected Result: The user should view and select products with no errors.
Priority: Critical.

5.3.2.3 Users can place an order.

Test Case id: 5323
Test Type: Functionality
Test Case Objective: Verify users can place an order.
Test Case Procedure:

- Login to the application.
- Click on the cart button.
- Verify that the cart page is displayed.
- Click on the check out button.
- Fill in the shipping details and click on the next button.
- Verify that shipping information is correct and that required places are not missing.
- Fill in the payment details.
- Verify that payment details are correct.
- Click on the Place Order button.
- Fill in the code to finish the payment.
- Verify that the order is placed successfully and the message order placed successfully is displayed.

Expected Result: The user should view and select products with no errors.

Priority: Critical.

5.3.2.4 Users can view order history

Test Case id: 5324
Test Type: Functionality
Test Case Objective: Verify users can view order history.
Test Case Procedure:

- Login the application.
- Click on the Orders button.
- Verify that the list of previous orders is displayed.
- Click on a random order to view its details.
- Verify that the order details page is displayed.

Expected Result: The user should view and select products with no errors.
Priority: Major.

5.4 Database Test

The database test ensures that all the backend records should then correct. Database testing checks whether all records are stored properly, and when the data is fetched, it is executed successfully. To achieve database testing, we use an API platform, Postman. Since the backend and front end of the application are not connected yet, all records are tested through the API.

5.4.1 Procedure of Database Test

- We design test scenarios, test cases, and test queries.
- After we run the tests, we report the errors.
- The errors are fixed
- Retest the scenarios that had errors
- Repeat the whole procedure until all problems are fixed.

5.4.2 Use Case Scenarios of Database Testing

5.4.2.1 User tries to login with valid credentials.

Test Case id: 5421
Test Case Objective: Check the login function.
Test Case Procedure:

- Open Postman.
- Select the Post option.
- Type “http://localhost:5000/api/auth/login”
- Select raw-data option and enter a valid username and password.

Expected Result: The user object should return in JSON format.

Priority: Major

5.4.2.2 User tries to login with invalid credentials.

Test Case id: 5422
Test Type: Functionality
Test Case Objective: Check the login function.
Test Case Procedure:

- Open Postman.
- Select the Post option.
- Type "http://localhost:5000/api/auth/login"
- Select the raw-data option and enter the invalid username and password.

Expected Result: The error message should return.
Priority: Major.

5.4.2.3 User tries to sign up with valid credentials.

Test Case id: 5423
Test Type: Functionality
Test Case Objective: Check the login function.
Test Case Procedure:

- Open Postman.
- Select the Post option.
- Type "http://localhost:5000/api/users/signup"
- Select the raw-data option and enter valid signup credentials.

Expected Result: The user object should return in JSON format.
Priority: Major.

5.4.2.3 User tries to sign up with invalid credentials.

Test Case id: 5424
Test Type: Functionality
Test Case Objective: Check the login function.
Test Case Procedure:

- Open Postman.
- Select the Post option.
- Type "http://localhost:5000/api/users/signup"
- Select the raw-data option and enter invalid signup credentials.

Expected Result: The error message should return.
Priority: Major.

5.4.2.5 User tries to upload a valid post.

Test Case id: 5425
Test Type: Functionality
Test Case Objective: Check the upload function for a post.
Test Case Procedure:

- Open Postman.
- Select the Post option.
- Type “http://localhost:5000/api/post/upload”
- Select the form-data option and upload an image in png format.
- Select the raw-data option and enter image information

Expected Result: The post object should return in JSON format.
Priority: Major

5.4.2.6 User tries to upload a invalid post.

Test Case id: 5426
Test Type: Functionality
Test Case Objective: Check the upload function for a post.
Test Case Procedure:

- Open Postman.
- Select the Post option.
- Type “http://localhost:5000/api/post/upload”
- Select the form-data option and upload a file, not in pn png format.
- Select the raw-data option and enter image information

Expected Result: The error message should return.
Priority: Major.

5.4.2.7 User tries to like a post.

Test Case id: 5427
Test Type: Functionality
Test Case Objective: Check the like function for a post.
Test Case Procedure:

- Open Postman.
- Select Post option.
- Type “http://localhost:5000/api/post/addlike/:id”
- Instead of ‘:id’, select one of the successfully uploaded image’s id

Expected Result: The post object should return as JSON format.
Priority: Major

5.4.2.8 User tries to dislike a post.

Test Case id: 5428
Test Type: Functionality
Test Case Objective: Check the dislike function for a post.
Test Case Procedure:

- Open Postman.
- Select Post option.
- Type “http://localhost:5000/api/post/removelike/:id”
- Instead of ‘:id’, select one of the successfully uploaded image’s id

Expected Result: The post object should return as JSON format.

Priority: Major

5.4.2.9 User tries to add a comment for a post .

Test Case id: 5429
Test Type: Functionality
Test Case Objective: Check the add comment for a post.
Test Case Procedure:

- Open Postman.
- Select the Post option.
- Type "http://localhost:5000/api/post/addComment/:id"
- Instead of ':id', select one of the successfully uploaded image's id
- Select the raw-data option and enter the comment

Expected Result: The post object should return as JSON format.
Priority: Major

5.4.2.10 User tries to like a product.

Test Case id: 54210
Test Type: Functionality
Test Case Objective: Check the like function for a product.
Test Case Procedure:

- Open Postman.
- Select the Post option.
- Type "http://localhost:5000/api/product/addlike/:id"
- Instead of ':id', select one of the successfully uploaded product's id

Expected Result: The post object should return as JSON format.
Priority: Major

5.4.2.11 User tries to dislike a product.

Test Case id: 5425
Test Type: Functionality
Test Case Objective: Check the dislike function for a product.
Test Case Procedure:

- Open Postman.
- Select the Post option.
- Type "http://localhost:5000/api/product/removelike/:id"
- Instead of ':id', select one of the successfully uploaded product's id

Expected Result: The post object should return as JSON format.
Priority: Major

6. Consideration of Various Factors in Engineering Design

During the analysis phase of our project, we have considered several factors relevant to the engineering design of RecoModa.

	Effect Level	Effect
Public Health	1/10	There is no significant effect that will adversely affect the health of the person using RecoModa. However, the application allows users to use it as much as they want. This might result in extensive and unbalanced use of RecoModa, which potentially could result in a slight drop in the physical and mental health of users. In order to prevent this kind of unwanted situation, we advise users of our application not to look at the mobile phone's screen for too long because looking at the screen for a long time has a negative effect on mental and eye health. In some situations where the feedback of extensive use of the application rises, we might also consider sending notifications to users to give small breaks while using the RecoModa application.
Public Safety	8/10	The information gathered from users to be used for the application is extremely important, and its confidentiality is one of the most critical issues for us. Information thathe t we collect from users, which are credit cart information, full postal address, and body measurement information, will be encrypted and protected and will not be shared with any 3rd parties at any cost. Regardless, due to the importance of this information, we will take all necessary precautions against any cyber attack that might aim to take possession of this valuable data. We attach great importance to data privacy in our practice, and we are aware that any data breach will put our practice in legal trouble. Because of that, we are performing to implement the latest technologies that are proven successful in data privacy.

Public Welfare	5/10	RecoModa does not implement any practices that will negatively affect the well-being of the users and put them under economic pressure. Nevertheless, RecoModa, as an application, makes purchasing much more effortless. Although it saves the users time, many other users with uncontrolled shopping habits may suffer economically. To avoid these kinds of situations, we recommend that users spend according to their income and keep track of their expenses while shopping. If this does not suppress the situation, we might add an adjustable shopping limiter for users who would like to be prohibited from excessive shopping.
Global Factors	8/10	RecoModa application is set to be used anywhere in the world, and the products are served by Amazon. Since the purchases and shipping will be handled by Amazon, our application will be actively offered in all countries where Amazon is available. We will release our application in English so that our application can be used actively by large masses. On need, we will consider offering the RecoModa application in other languages, preferably all languages currently used on Amazon.
Cultural Factors	7/10	RecoModa is a social app along with a shopping app. As fashion can change through culture, we expect users to form different communities with their cultural clothing styles gradually. However, we accept the differences in cultures and fashion; we try not to ignore culture and not to become a multicultural practice for everyone. Thanks to the recommendation algorithm of RecoModa, we will continue our efforts to offer unprejudiced recommendations that are suitable for the cultural clothing styles of the users. That way, every user can be able to shape their preferences in the application freely.
Social Factors	9/10	Recomoda is an application that tries to gather all its users under the roof of fashion. Our application is a

		social media application as well as a shopping application. It is possible for people to follow each other, watch through, like, and produce content, and even turn it into an income. We set out to provide users with a free experience in the social context. We aim to enable people to communicate with each other and to develop in the field of fashion as long as the rules we regulate in this social environment.
Environmental Factors	3/10	RecoModa application uses the latest technologies in application development to maximize efficiency and minimize the workload of the servers to make it more accessible for users all over the world. Because of this, we are aiming to make our application environment-friendly and use lesser resources for the application to run. However, we do not have any applications on a greater scale that tries to center the environmental factors which make our efforts stay in a low effect level.
Economic Factors	5/10	RecoModa application has a mechanism that promotes and makes shopping easier, which thus results in an increase in shopping. We expect that our practice will have a positive effect on the major economy in the long run, as it stimulates the market and increases purchasing.

7. Teamwork Details

During the implementation of the RecoModa project, the group members were divided into three groups to work on the front-end, back-end, and data science. Although the group was divided into three, the communication between the group members continued continuously in the parts that required synchronization. Also, in need, we interchanged the groups to maximize efficiency. Detailed information about teamwork is given in the rest of this section.

7.1 Contributing and functioning effectively on the team

Güven Gergerli: Güven has played a role in the development of the machine learning model and data science of the project. He has helped in collecting and cleaning data from shopping sites such as Amazon, exploring data using various tools, and setting the data for the implementation of various machine learning models. He has also assisted in developing a data pipeline that automates the data cleaning and transformation process. He conducted research on which database could be used for the project and then worked on modeling the project database.

Hakan Gülcü: Hakan has played a role in the development of the machine learning model and data science of the project. He conducted research on which machine-learning models could be used for the project and then worked on implementing and tweaking the machine-learning model. He has also researched and developed different recommendation models based on visual data. He has helped in collecting and cleaning data from shopping sites such as Amazon, exploring data using various tools, and setting the data for the implementation of various machine learning models.

Nasuh Dinçer: Nasuh has played a role in the development of the front-end part of the RecoModa application. He has developed the required pages for the application to function. He has also played the role of merging connections between the front-end and back-end sides of the application. He also contributed to the back-end project of the application. If we talk about the technologies used by Front end, development is done on react js. Android studio and emulator are used to run tests on Android. In addition, the application will be tested over the phone.

Tarık Buğra Karali: Tarık has played a role in the development of the back-end part of the RecoModa application. He has developed the required functions for the application to function. He has handled the server-client connection of the application and created queries for the database. He has also played the role of merging connections between the front-end and back-end sides of the application. He also contributed to the front-end project of the application.

Zülal Nur Hıdıroğlu: Zülal has played a role in the development of the back-end and data science part of the RecoModa application. She has helped with the development of the required functions and queries for the application. She has researched developing automated data scraping and cleaning. She has also played the role of merging connections between the back-end and machine-learning sides of the application. She also contributed to the development of the machine learning model of the application.

7.2 Helping creating a collaborative and inclusive environment

We each have our own set of obligations. If the person in charge of a task has other duties or finds the work assigned to them challenging to do, we assist them in completing the assignment. When someone is unsure about a task, we always support one another in specifying precisely what needs to be done. To keep each other informed, we communicate the status of the tasks for which we are accountable on a regular basis. We provide each other feedback on the tasks for which we are accountable and make improvements as needed. Even if we assigned various jobs to each other, we always collected feedback from other teammates while working on a task or after completing it to verify that the tasks were completed correctly.

7.3 Taking lead role and sharing leadership on the team

We always shared the responsibilities for leadership. When we needed to organize a meeting, decide on a timetable, or accomplish something as a team in general, whoever was the most available at the time took the lead. Having our own responsibilities, we tried to give a lead to others in need and attached great importance to it. The goal is for both of us to lighten each other's load and contribute new insights into our progress. Furthermore, just because we have various leaders at different periods doesn't really indicate that the leader makes decisions independently. We tried to bring our ideas to a consensus before we proceeded to minimize the errors. Our leader's role is to organize individuals, mix diverse ideas, and develop a strategy by bringing the ideas together.

8. References

[1]Hashim, N. A., Janor, H., Sidek, F., & Nor, S. M. (2018). Online shopping: A potential of herding behavior symptom? *Business and Management Horizons*, *6*(2), 105. https://doi.org/10.5296/bmh.v6i2.14197

[2] Wikimedia Foundation. (2022, November 2). *React native*. Wikipedia. Retrieved November 13, 2022, from https://en.wikipedia.org/wiki/React_Native

[3]JWT. (n.d.). *JSON web tokens introduction*. JSON Web Token Introduction. Retrieved March 13, 2023, from https://jwt.io/introduction

[4]*Security · REACT NATIVE*. React Native RSS. (2023, February 1). Retrieved March 13, 2023, from https://reactnative.dev/docs/security

[5]*Jest*. Jest RSS. (n.d.). Retrieved March 13, 2023, from https://jestjs.io/

[6]Introduction · Enzyme. (n.d.). Retrieved March 13, 2023, from https://enzymejs.github.io/enzyme/