

EASTERN INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY
DEPARTMENT OF SOFTWARE ENGINEERING



PROJECT 2 REPORT
HAIRCUT BOOKING AND MANAGEMENT
SYSTEM

Students

Lê Chí Cường - 2131200001

Đặng Phú Tân – 2131200019

Supervisor

Trần Thị Như Quỳnh

Binh Duong, March, 2025

EASTERN INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY
DEPARTMENT OF SOFTWARE ENGINEERING



PROJECT 2 REPORT
HAIRCUT BOOKING AND MANAGEMENT
SYSTEM

Students

Lê Chí Cường - 2131200001

Đặng Phú Tân – 2131200019

Supervisor

Trần Thị Như Quỳnh

Binh Duong, March, 2025

ABSTRACT

The haircut booking and management system has undergone significant improvements, focusing on enhancing both functionality and user experience. Initially, the system was fully implemented with essential features, ensuring smooth business operations and allowing users to interact with the platform effectively. However, further refinements were needed to improve usability, accessibility, and design. In this phase, new pages and additional features have been introduced to create a more seamless and intuitive experience. Navigation has been improved to make it easier for both customers and staff to book appointments, manage services, and access important information. The user interface has been redesigned for better responsiveness across various devices, ensuring a visually appealing and user-friendly interaction. Alongside these front-end improvements, new technologies are being integrated to expand the platform's capabilities, offering smarter hairstyle recommendations, flexible scheduling options, and improved resource management. These advancements enhance security, scalability, and overall system performance, allowing businesses to operate more efficiently. With continuous innovation, the platform is evolving into a comprehensive and intelligent solution that streamlines business operations while delivering a superior experience for both customers and staff.

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to Ms. Tran Thi Nhu Quynh for his invaluable assistance in completing this project. His expertise and guidance have been crucial in helping us navigate challenges and achieve our objectives. Ms. Quynh's dedication and willingness to share his knowledge have greatly enhanced our work, and his mentorship has been a source of inspiration for the entire team. I deeply appreciate his unwavering support and commitment, which have been the primary key to the success of our project. Working with Ms. Tran Thi Nhu Quynh has been a privilege, and his contributions have made a lasting impact. Thank you for all your excellent advice and support throughout this journey.

TABLE OF CONTENTS

ABSTRACT.....	ii
ACKNOWLEDGEMENT.....	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES.....	vi
LIST OF TABLES.....	vii
LIST OF ABBREVIATIONS	viii
Chapter 1: INTRODUCTION.....	1
1.1 Project Overview	1
1.2 Project Functions	1
Chapter 2: TECHNOLOGIES	2
2.1 Java	2
2.2 Spring Boot.....	3
2.3 MySQL	4
2.4 JavaScript.....	6
2.5 ReactJS.....	7
2.6 SCSS & CSS	8
2.7 Firebase.....	10
2.8 Google Drive API	11
Chapter 3: SYSTEM ANALYSIS	13
3.1 UseCase Diagram	13
3.2 Database diagram.....	13
3.3 Database tables	14
3.4 Firebase Notification.....	21
3.5 Google Drive API.....	21
Chapter 4: IMPLEMENTATION.....	22
4.1 Database.....	22

4.2 APIs	22
Chapter 5: RESULT	26
5.1 Home page – Admin - Manager	26
5.2 Customer	29
5.3 Employee pages	30
5.4 Common pages	31
5.5 Result Summary	33
Chapter 6: CONCLUSIONS AND FUTURE WORKS	34
6.1 Conclusions	34
6.2 Limitations	34
6.3 Future works	35
REFERENCES	36

LIST OF FIGURES

Figure 1 - Java Logo	2
Figure 2 - Spring Boot logo	3
Figure 3 - MySQL logo	4
Figure 4 - JavaScript logo.....	6
Figure 5 - ReactJS logo.....	7
Figure 6 - SCSS logo	8
Figure 7 - CSS logo	9
Figure 8 - Firebase logo	10
Figure 9 - Google Drive API	11
Figure 10 – UseCase Diagram	13
Figure 11 - Database Diagram	14
Figure 12 - Home Page – Admin – Manager	26
Figure 13 - Manager dashboard	27
Figure 14 - Create/Update example Dialog	28
Figure 15 - Appointment Creating Dialog	28
Figure 16 - Appointment Editing Dialog	29
Figure 17 - Customer Home page.....	29
Figure 18 - Services/Combos Carousel Page.....	30
Figure 19 - Booking Page	30
Figure 20 - Appointment history of Employee Page	31
Figure 21 - Profile Page	31
Figure 22 - Change Password Page	32
Figure 23 - Notification Panel	32

LIST OF TABLES

Table 1 - Customer Table	15
Table 2 - Worker Table.....	16
Table 3 - Location Table.....	16
Table 4 - Appointment Table.....	17
Table 5 - Appointment_detail Table	17
Table 6 - Appointment_detail_id_service Table	17
Table 7 - Appointment_detail_id_combo Table.....	18
Table 8 - service_entity Table	18
Table 9 - Product Table	19
Table 10 - Combo Table	19
Table 11 - service_entity_product Table	19
Table 12 - Notification Table.....	20
Table 13 - Notification_user Table	20
Table 14 - Notification_user_notification Table	20
Table 15 - Token_firebase Table	20
Table 16 - firebase_user_tokens_token Table	21
Table 17- firebase_user_token.....	21

LIST OF ABBREVIATIONS

CSS	Cascading Style Sheets
API	Application Programming Interface
RESTful	Representational State Transfer
HTTPS	Hypertext Transfer Protocol Secured
HTTP	Hypertext Transfer Protocol
HTML	HyperText Markup Language
MYSQL	My Structured Query Language
SCSS	Sassy Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML
DOM	Document Object Model
JSON	JavaScript Object Notation
JSX	JavaScript Extensible Markup Language
XML	Extensible Markup Language
JVM	Java Virtual Machine
UX/UI	User Experience/User Interface
JSX	JavaScript Extensible Markup Language
SQL	Structured Query Language
CLI	Command-line Interface
ER diagram	Entity-Relationship diagram
AI	Artificial Intelligence
CRUD	Create, Read, Update and Delete

Chapter 1: INTRODUCTION

1.1 Project Overview

The haircut booking and management system is a web-based platform designed for hair salons of all sizes to efficiently manage staff, services, and appointments. Building on the previous version, several improvements have been made to enhance usability and performance. The user interface has been redesigned for a more intuitive and visually appealing experience, making navigation smoother for both customers and salon staff. Firebase notifications have been integrated, keeping users informed with important updates and reminders. Additionally, file storage has been moved to Google Drive, reducing server memory usage and improving system efficiency. These enhancements make the platform more user-friendly, responsive, and reliable while maintaining secure role-based access control for staff. With these upgrades, the system now offers a more seamless and convenient experience for managing salon operations.

1.2 Project Functions

- **Web-based platform:** A web-based platform offers accessibility, scalability, cost-effectiveness, centralized data, easy updates, cross-device usability, security, and seamless integration.
- **Customer Appointment Booking:** Customers can book appointments by choosing location, services, staff, and time, with all details securely stored in the database.
- **Role-Based Access Control:** This enhances security, simplifies management, boosts productivity, ensures compliance, reduces errors, supports scalability, provides clear accountability, and improves cost-effectiveness.
- **CRUD Operations for Administrators:** allows administrators to create, view, update, and delete records for staff, services, locations, products, and appointments.
- **Real-Time Appointment Management:** Customers can review, update, or cancel their appointments. Staff can view their schedules and appointment history in real time.
- **User Authentication:** Requires customers, managers, and staff to log in to access their respective features and dashboards.

Chapter 2: TECHNOLOGIES

2.1 Java



Figure 1 - Java Logo

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. Developed by Sun Microsystems (now owned by Oracle Corporation) in 1995, Java is widely used for building platform-independent applications due to its "write once, run anywhere" capability, which means compiled Java code can run on any device that supports Java without the need for recompilation.

Key points:

- **Platform-Independent:** Java applications are compiled into bytecode that can run on any Java Virtual Machine, making them cross-platform.
- **Object-Oriented:** Supports principles like inheritance, encapsulation, polymorphism, and abstraction, promoting reusable and modular code.
- **Robust and Secure:** Features strong memory management, exception handling, and a secure execution environment.
- **Rich Standard Library:** Extensive set of APIs and libraries for various tasks, from data structures to networking and GUI development.

Java is versatile and used in a wide range of applications, from web and mobile applications to enterprise-level and embedded systems. It is the backbone of many largescale enterprise applications and is extensively used in financial services, retail, and

healthcare sectors for building robust and scalable solutions. Its platform independence makes it ideal for developing cross-platform applications.

2.2 Spring Boot



Figure 2 - Spring Boot logo

Spring Boot is an open-source Java-based framework used to create independent Spring-based applications. It is built on top of the Spring Framework and provides a faster and more efficient way to develop microservices and web applications by simplifying the setup and development process.

Key points:

- **Convention over Configuration:** Provides default configurations to minimize boilerplate code. Defaults can be overridden.
- **Embedded Server:** Runs on an embedded server (Tomcat, Jetty, Undertow), simplifying deployment and testing.
- **Auto-Configuration:** Automatically configures based on added dependencies.
- **Starter POMs:** Simplifies dependency management with starter POMs like spring-boot-starter-web.
- **Production-Ready Features:** Includes health checks, metrics, externalized configuration, and logging via Spring Boot Actuator.
- **Spring Boot CLI:** Allows running Groovy scripts for quick prototyping and development.

- **Externalized Configuration:** Supports configuration via properties or YAML files for different environments.
- **Microservices Support:** Ideal for building microservices with simplicity and cloud service integration.
- **Spring Initializer:** Web tool for generating Spring Boot projects with necessary dependencies.
- **Security Integration:** Integrates easily with Spring Security for authentication, authorization, and protection against vulnerabilities.

Spring Boot is a Java-based framework that simplifies the development of standalone, production-grade Spring applications. It follows a convention-over-configuration approach, reducing boilerplate code and manual setup. With embedded servers like Tomcat, it allows easy deployment and testing. Key features include auto-configuration based on project dependencies, a variety of starter POMs for simplified dependency management, and production-ready features such as health checks and metrics through Spring Boot Actuator. It supports microservices, cloud-based applications, batch processing, and more. Tools like Spring Boot CLI and Spring Initializer also streamline project setup and Development.

2.3 MySQL



Figure 3 - MySQL logo

MySQL is an open-source relational database management system developed by Oracle Corporation. It is widely used for its reliability, performance, and ease of use, particularly in web-based applications.

Key features:

- **Open Source:** Free to use with a large community and extensive documentation.
- **High Performance:** Optimized for speed and efficiency, capable of handling large-scale databases.
- **Scalability:** Supports large databases, making it suitable for both small and large applications.
- **Cross-Platform:** Runs on various operating systems including Windows, Linux, and MacOS.
- **ACID Compliance:** Ensures reliable transactions with Atomicity, Consistency, Isolation, and Durability.
- **Replication:** Supports master-slave replication for high availability and load balancing.
- **External, Internal, and Inline Styles:** CSS can be applied externally through separate style sheets, internally within HTML documents using the `<style>` element, or inline directly within HTML tags.
- **Security:** Provides robust security features like user authentication, SSL support, and data encryption.
- **Integration:** Easily integrates with various programming languages and web technologies.

MySQL is widely used in web development as the backend database for dynamic websites and applications. MySQL's scalability and performance make it ideal for handling high-traffic websites, large-scale enterprise applications, and cloud-based services. It is also commonly used in data warehousing, online transaction processing and distributed applications due to its reliability and support for replication and clustering.

2.4 JavaScript



Figure 4 - JavaScript logo

JavaScript is a high-level, versatile programming language commonly used to create interactive effects within web browsers. It is an essential part of web development, alongside HTML and CSS. JavaScript allows developers to implement complex features on web pages, making them interactive and dynamic.

Key features:

- **Interpreted Language:** JavaScript is executed directly by the web browser without the need for prior compilation.
- **Event-driven:** JavaScript can respond to user actions, such as clicks, form submissions, and mouse movements.
- **Object-Oriented:** Supports object-oriented programming principles, including objects, inheritance, and polymorphism.
- **Client-Side and Server-Side:** While primarily used on the client side, JavaScript can also be executed on the server side with environments like Node.js.
- **Cross-Platform:** Runs on various devices and operating systems, making it highly adaptable.
- **Rich Ecosystem:** Extensive libraries and frameworks enhance development efficiency and capabilities.

JavaScript is essential for modern web development, enabling dynamic user interfaces, form validation, animations, and asynchronous content updates via AJAX. On the server side, platforms like Node.js use JavaScript for building scalable applications and APIs. It's also used in game and mobile app development with frameworks like React

Native, as well as in the Internet of Things, making it a key technology for both front-end and back-end development.

2.5 ReactJS

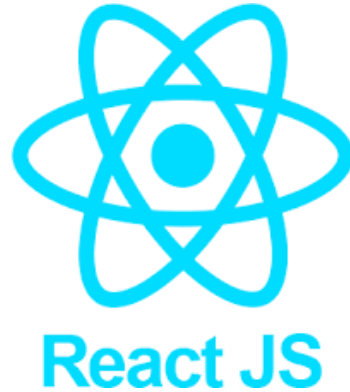


Figure 5 - ReactJS logo

ReactJS is an open-source JavaScript library for building user interfaces, particularly for single-page applications. It allows developers to create reusable UI components and manage the state of complex applications efficiently. React was developed and is maintained by Facebook.

Key features:

- **Component-Based Architecture:** React applications are built with reusable components that encapsulate logic and UI, promoting a modular and maintainable codebase.
- **Virtual DOM:** React uses a Virtual DOM to optimize rendering by updating only the necessary UI parts, improving performance.
- **Declarative UI:** React lets developers define the UI at any point, and efficiently updates it when the application state changes.
- **JSX (JavaScript XML):** React uses JSX, enabling HTML-like syntax within JavaScript, making code more readable and combining logic and markup in one file.
- **One-Way Data Binding:** React uses one-way data binding, where data flows from parent to child components via props, making the data flow predictable and easier to debug.
- **State Management:** React components manage their own state, and React provides tools to update and reflect state changes in the UI.

- **React Hooks:** Introduced in React 16.8, Hooks allow you to use state and other React features without writing a class component.
- **High Performance:** React optimizes rendering using techniques like reconciliation, updating only the components that need it, resulting in better performance.
- **React Router:** React Router enables navigation between views or components in single-page applications without reloading the page, ensuring a smooth user experience.
- **Rich Ecosystem:** React has a large ecosystem of tools, libraries, and community support, making it easier to build complex applications. Popular libraries include Redux (for state management), React Native (for mobile apps), and Next.js (for server-side rendering).

2.6 SCSS & CSS

2.6.1 SCSS



Figure 6 - SCSS logo

SCSS (Sassy CSS) is a syntax of Sass (Syntactically Awesome Stylesheets), which is a preprocessor scripting language that extends CSS (Cascading Style Sheets). SCSS allows for more powerful and dynamic styling through features like variables, nesting, and mixins, making it easier to write and manage stylesheets for large projects.

Key features:

- **Variables:** SCSS variables store values like colors and fonts, promoting reusability and simplifying style updates.
- **Nesting:** SCSS supports nested syntax, which helps to organize styles in a hierarchical manner, making it easier to read and maintain.

- **Partials and Import:** SCSS lets you break styles into partials, which can be imported into a main stylesheet, improving modularity and scalability.
- **Mixins:** SCSS mixins allow you to create reusable blocks of styles that can accept arguments. This is useful for avoiding repetitive code.
- **Inheritance (Extends):** SCSS uses the `@extend` directive to allow one selector to inherit another's properties, reducing redundancy and organizing common styles.
- **Mathematical Operations:** SCSS supports arithmetic operations like addition, subtraction, multiplication, and division, which can be applied to values like colors and sizes.
- **Color Functions:** SCSS provides built-in functions for manipulating colors, such as darkening, lightening, or mixing colors.
- **Loops:** SCSS allows the use of loops for repetitive tasks, which helps to avoid writing repetitive styles.
- **Functions:** SCSS allows users to define custom functions to perform calculations or manipulate values.
- **Built-in Functions:** SCSS has a variety of built-in functions for handling strings, colors, lists, maps, and more, allowing you to perform complex operations with ease.

2.6.2 CSS



Figure 7 - CSS logo

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a document written in a markup language like HTML. It is used to control the layout, color, font, and other visual aspects of a web page. CSS allows web developers

to separate the content (HTML) from the presentation, making it easier to maintain and update the design of a website.

Key Points:

- Separation of Concerns: CSS enables the separation of content (HTML) and presentation (CSS), making it easier to maintain and update the design of a website.
- Cascading: The "Cascading" in CSS refers to the way styles are applied to elements on a web page. Styles can be inherited from parent elements, and multiple styles can be applied to a single element.
- Responsive Design: CSS plays a crucial role in creating responsive web designs that adapt to different screen sizes and devices, ensuring a consistent and optimized user experience.
- Browser Compatibility: CSS has evolved over time, and different browsers may have varying levels of support for newer CSS features. Developers need to consider cross-browser compatibility when writing CSS.
- Preprocessors and Frameworks: There are several CSS preprocessors and frameworks that can enhance the development workflow and provide additional features.

One of the most common applications of CSS is in web development. CSS is used to style the appearance of web pages, including elements such as headings, paragraphs, images, and navigation menus. By applying CSS rules to HTML elements, web developers can control a website's layout, colors, fonts, and other visual aspects, ensuring a consistent and visually appealing user experience.

2.7 Firebase



Figure 8 - Firebase logo

Firebase is a platform developed by Google to help developers build mobile and web applications. It offers a range of cloud-based tools and services, including real-time databases, user authentication, and cloud storage.

Key Points:

- **Real-time Database:** Firebase offers a NoSQL database that allows real-time data synchronization, ideal for live updates in apps.
 - **Authentication:** Simple methods for secure user authentication, supporting email/password, social media logins, and phone number authentication.
 - **Cloud Storage:** Firebase provides scalable cloud storage for user-generated content like photos and videos.
 - **Hosting:** Fast and secure web hosting for static files with automatic SSL certificates.
 - **Analytics & Performance Monitoring:** Tools for tracking user behavior and app performance, helping developers optimize their apps.
 - **Serverless Functions:** Allows developers to run backend code without managing servers.
- Cross-platform Support: Works across iOS, Android, and web platforms.
- One of the main applications of Firebase is in mobile and web app development.

2.8 Google Drive API



Figure 9 - Google Drive API

Google Drive API is a cloud-based file management system developed by Google. It enables users to store, access, and manage files efficiently while ensuring seamless collaboration across devices and platforms.

Key points:

- **File Management:** Allows users to create, read, update, and delete files and folders programmatically.

- Access Control: Supports fine-grained permission settings for file sharing and collaboration.
- Real-time Collaboration: Enables multiple users to edit and sync documents in real-time.
- Cross-Platform Support: Works across web, Android, iOS, and desktop applications.
- Search & Indexing: Provides powerful search capabilities based on metadata and content.
- Security & Authentication: Uses OAuth 2.0 for secure access and ensures data privacy.
- Versioning: Keeps track of file changes and allows users to restore previous versions.
- Integrations: Easily connects with third-party apps and Google Workspace services.

Chapter 3: SYSTEM ANALYSIS

3.1 UseCase Diagram

Use cases play a crucial role in software development by documenting user requirements, clarifying system behaviour, facilitating communication among stakeholders, guiding system design, supporting testing and validation, enabling requirement traceability, and serving as a basis for Agile development practices. They help ensure that the resulting system meets user needs and expectations effectively.

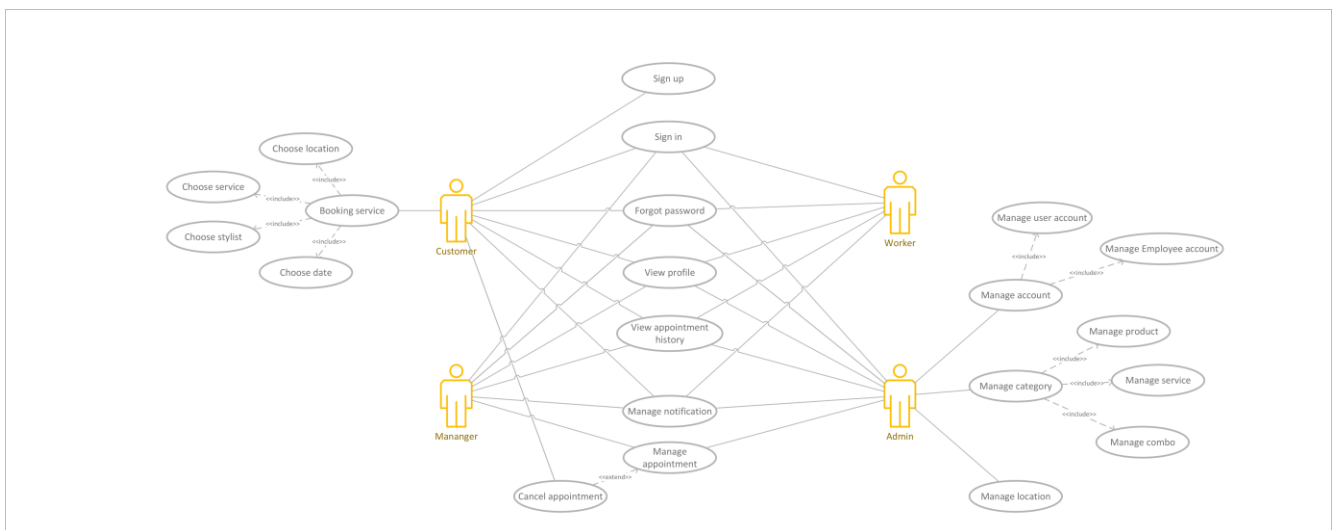


Figure 10 – UseCase Diagram

In this final project, the usecase diagram is still mostly the same, expect that now users can have access to the notifications, they can view/manage their notifications.

3.2 Database diagram

A database diagram, also known as an entity-relationship diagram (ER diagram), is a visual representation of the structure of a database. It illustrates the relationships between different entities (tables) in the database and the attributes (columns) associated with each entity.

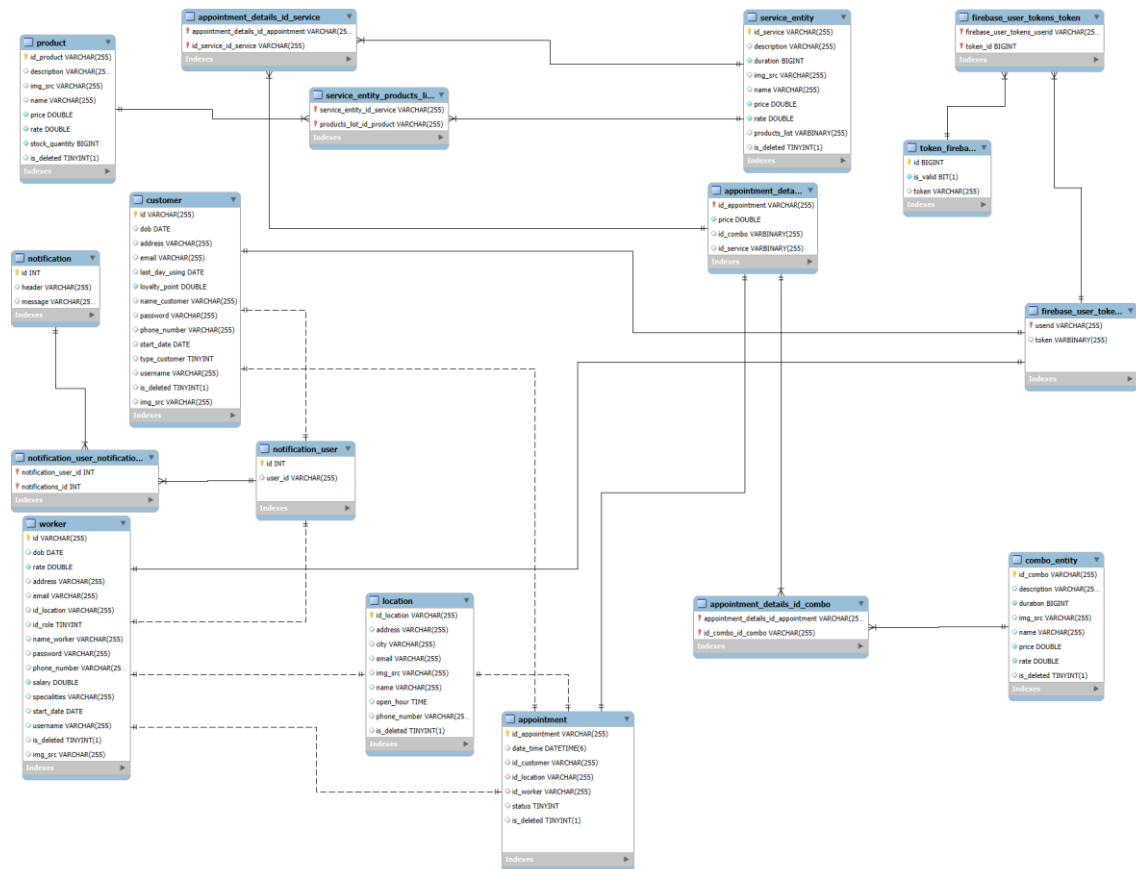


Figure 11 - Database Diagram

The database had been added a few more tables to serve new features/ functions of the web app.

3.3 Database tables

3.3.1 Customer

Column Name	Data Type	Constraint
id	varchar(255)	Primary_key
name_customer	varchar(255)	
username	varchar(255)	not null, unique

email	varchar(255)	unique
password	varchar(255)	not null
phone_number	varchar(255)	
address	varchar(255)	
img_src	varchar(255)	
loyalty_point	double	
dob	date	
start_date	date	
last_day_using	date	
type_customer	tinyint	
is_deleted	tinyint(1)	

Table 1 - Customer Table

3.3.2 Worker

Column Name	Data Type	Constraint
id	varchar(255)	Primary_key
name_worker	varchar(255)	not null
username	varchar(255)	not null, unique
email	varchar(255)	unique
password	varchar(255)	not null
phone_number	varchar(255)	not null
address	varchar(255)	

img_src	varchar(255)	
id_location	varchar(255)	foreignkey
specialities	varchar(255)	
rate	double	
salary	double	
dob	date	
start_date	date	
id_role	tinyint	
is_deleted	tinyint(1)	

Table 2 - Worker Table

3.3.3 Location

Column Name	Data Type	Constraint
id_location	varchar(255)	Primary_key
name	varchar(255)	not null
email	varchar(255)	
phone_number	varchar(255)	not null
address	varchar(255)	not null
img_src	varchar(255)	
city	varchar(255)	not null
open_hour	time(6)	
is_deleted	tinyint(1)	

Table 3 - Location Table

3.3.4 Appointment

Column Name	Data Type	Constraint
id_appointment	varchar(255)	Primary_key
id_customer	varchar(255)	foreignkey
id_worker	varchar(255)	foreignkey
date_time	datetime(6)	not null
id_location	varchar(255)	foreignkey
status	tinyint	not null
is_deleted	tinyint(1)	

Table 4 - Appointment Table

3.3.5 Appointment_detail

Column Name	Data Type	Constraint
id_appointment	varchar(255)	Primary_key, foreignkey
price	double	
is_deleted	tinyint(1)	

Table 5 - Appointment_detail Table

3.3.6 Appointment_detail_id_service

Column Name	Data Type	Constraint
id_appointment	varchar(255)	Primary_key, foreignkey
id_service	varchar(255)	Primary_key, foreignkey

Table 6 - Appointment_detail_id_service Table

3.3.7 Appointment_detail_id_combo

Column Name	Data Type	Constraint
id_appointment	varchar(255)	Primary_key, foreignkey
id_combo	varchar(255)	Primary_key, foreignkey

Table 7 - Appointment_detail_id_combo Table

3.3.8 service_entity

Column Name	Data Type	Constraint
id_service	varchar(255)	Primary_key
name	varchar(255)	not null
description	varchar(255)	
duration	long	not null
img_src	varchar(255)	
price	double	not null
rate	double	not null
is_deleted	tinyint(1)	

Table 8 - service_entity Table

3.3.9 Product

Column Name	Data Type	Constraint
id_product	varchar(255)	Primary_key
name	varchar(255)	not null
description	varchar(255)	

stock_quantity	bigint	not null
img_src	varchar(255)	
price	double	not null
rate	double	not null
is_deleted	tinyint(1)	

Table 9 - Product Table

3.3.10 Combo

Column Name	Data Type	Constraint
id_combo	varchar(255)	Primary_key
name	varchar(255)	not null
description	varchar(255)	
duration	long	not null
img_src	varchar(255)	
price	double	not null
rate	double	not null
is_deleted	tinyint(1)	

Table 10 - Combo Table

3.3.11 service_entity_product

Column Name	Data Type	Constraint
service_entity_id_service	varchar(255)	Primary_key, foreignkey
products_list_id_product	varchar(255)	Primary_key, foreignkey

Table 11 - service_entity_product Table

3.3.12 Notification

Column Name	Data Type	Constraint
id	int	Primary_key
header	varchar(255)	
message	varchar(255)	

Table 12 - Notification Table

3.3.13 Notification_user

Column Name	Data Type	Constraint
id	int	Primary_key
user_id	varchar(255)	foreignkey

Table 13 - Notification_user Table

3.3.14 Notification_user_notification

Column Name	Data Type	Constraint
notification_user_id	int	Primary_key, foreignkey
notification_id	int	Primary_key, foreignkey

Table 14 - Notification_user_notification Table

3.3.15 Token_firebase

Column Name	Data Type	Constraint
id	int	Primary_key
token	varchar(255)	not null
isvalid	bit(1)	not null

Table 15 - Token_firebase Table

3.3.16 firebase_user_tokens_token

Column Name	Data Type	Constraint
firebase_user_tokens_userid	varchar(255)	Primary_key, foreignkey
token_id	int	Primary_key, foreignkey

Table 16 - firebase_user_tokens_token Table

3.3.17 firebase_user_token

Column Name	Data Type	Constraint
userid	varchar(255)	Primary_key, foreignkey

Table 17- firebase_user_token

3.4 Firebase Notification

The firebase notification gives my web-app a better way to inform our web-app users through its provided functions. Our web-app runs with firebase to give users notifications but this project does not use firebase cloud storage to storage the old or new notifications but another table will be created to store them. This approach gives our project a better way in managing the old notifications by deleting the read notifications to save memory.

By using firebase cloud notifications, our web-app can push notifications to the users that have specific firebase tokens which will be given by firebase itself.

With this approach, we can give back to the users the notifications even if they do not accept notification on their browser.

3.5 Google Drive API

We used Google Drive API to store user images and other images. This approach helps to reduce the memory used to store them in the server.

Chapter 4: IMPLEMENTATION

4.1 Database

MySQL: We added two more tables to manage user notifications named (Notification_user_notification, notification). We then created two more tables to store user firebase tokens (firebase_user_tokens_token, Token_firebase), with this approach we can store the tokens of the users after they send them to the server.

4.2 APIs

In the previous project, the API had been built done to complete all the tasks but there are still be something new to improve and add to this APIs.

The plan is to add notification system to keep the users in touch with news. Following to structure of the APIs, we created new classes in **Controller**, **Repository**, **Entity** and **Service** to handle new features.

4.2.1 Token firebase and notification

@Configuration

```
public class FirebaseInstallization {
```

@Bean

```
public FirebaseMessaging installization() {  
    try (FileInputStream serviceAccount = new FileInputStream("./firebase_key.json")) {  
        FirebaseOptions options = new FirebaseOptions.Builder()  
            .setCredentials(GoogleCredentials.fromStream(serviceAccount))  
            .build();  
        FirebaseApp app = FirebaseApp.initializeApp(options);  
        return FirebaseMessaging.getInstance(app);  
    } catch (IOException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
    return FirebaseMessaging.getInstance();  
}
```

Before doing anything, we first configured the **FirebaseInstallization** to have access to firebase feature.

In package **Controller**, we created new class named **FirebaseController** to receive request from users.

Endpoint:

```
@RestController
@RequestMapping("/notification")
@RequiredArgsConstructor
```

@PostMapping

```
public ApiResponse<String> receiveToken(@RequestBody TokenRequest token)
```

This endpoint will be used to receive firebase tokens from users and then call the service layer to store the tokens into the database. With this approach, it gives us a way to know how to send messages or notifications to the users.

@GetMapping

```
public ApiResponse<Set<Notification>> getMyNotifications()
```

Users can call this Get endpoint to receive their notifications. In this endpoint, controller will call the service layer to give the users back their notifications. We used a class name **SecurityContextHolder** given by Spring to know which users are coming by this endpoint.

Service layer:

We used **ServicesUtils** class to be a third party class which provides middle progress to generate information required by other classes.

In **ServicesUtils** class, we created a few functions that will be used to send message to specific users.

```
public void sendNotificationsToUsers(List<String> userIds, String header, String
```

This function is responsible to send notification to a list of users.

```
public void sendNotificationsToUser(String idUser, String header, String body)
```

This function is responsible to send notification to a user identified by its userId. This function will call class **NotificationService** which will handle the last part of sending

message after this class handle all the required steps. This mainly role is to get the active **Firestore Tokens** of the user matching the idUser.

```
public void checkListReponse(BatchResponse batchResponse, List<String> tokens, String userID)
```

We also created a function to delete the firebase tokens that the notifications are sent unsuccessfully. This will use **BatchReponse** class to know which **Firestore Token** the message cannot reach then delete that token from the database.

By this approach, we can reduce the size of memory the store the firebase tokens of the users.

Notification service class:

```
public BatchResponse sendMessages(FirestoreNotification firebaseNotification, List<String> tokens)
```

This function will be the main function to send message to users. The other class will define which account users they want to send the message and retrieve a list of firebase tokens related to that list of users then this function will send their message to that list of tokens. This function will give back a **BatchResponse** which contains all the information about the sending progress.

```
public void addNotificationToUser(String header, String message, String userId)
```

This function is used to store the notification to the database. This helps the users to receive the notifications even if they are not active.

```
public APIresponse<Set<com.haircutAPI.HaircutAPI.entity.Notification>>  
getMyNotifications(Authentication authentication)
```

This function will give back the list of notification of the user who calls the endpoint.

4.2.2 File uploading Google Drive

Configuration:

To get access to google Drive, we had to do some configurations to get the access.

```
private static final JsonFactory JSON_FACTORY = GsonFactory.getDefaultInstance();  
private static final String SERVICE_ACCOUNT_KEY_PATH =  
getPathToGoodleCredentials()  
private static String getPathToGoodleCredentials();  
private Drive createDriveService() throws GeneralSecurityException, IOException
```

This configuration connects the API to the Google Drive.

Service:

```
public Images uploadImageToGoogleDrive(File file)
```

This function will store the file in the google drive and return an **Images** object.

```
public void deleteFile(String id) throws GeneralSecurityException,  
IOException
```

This function will delete the file matched the **id** in the google drive.

Chapter 5: RESULT

The haircut booking and management system is a comprehensive platform designed to streamline salon operations while providing a seamless experience for customers, employees, managers, and administrators. Customers can easily book appointments by selecting their preferred time, services, stylist, and location, while also managing their appointment history and profile. Employees have access to their schedules and profile management features, ensuring they stay updated with upcoming appointments. Managers oversee restricted operations, while administrators manage all aspects of the system through a centralized dashboard. The platform also integrates Firebase notifications to keep users informed and utilizes Google Drive for efficient file storage, reducing server load. With its well-structured role-based access and user-friendly interface, the system enhances efficiency, convenience, and organization for all users.

5.1 Home page – Admin - Manager

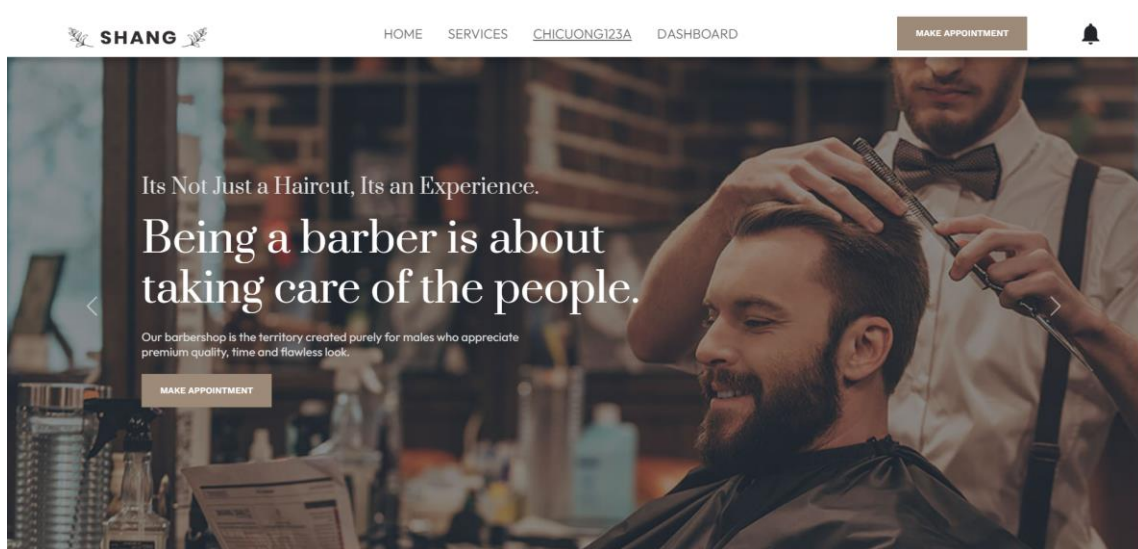


Figure 12 - Home Page – Admin – Manager

In the home page now, accounts that have permissions as admin or manager can go to dashboard by clicking the dashboard label on the nav bar.

5.1.1 Dashboard of manager

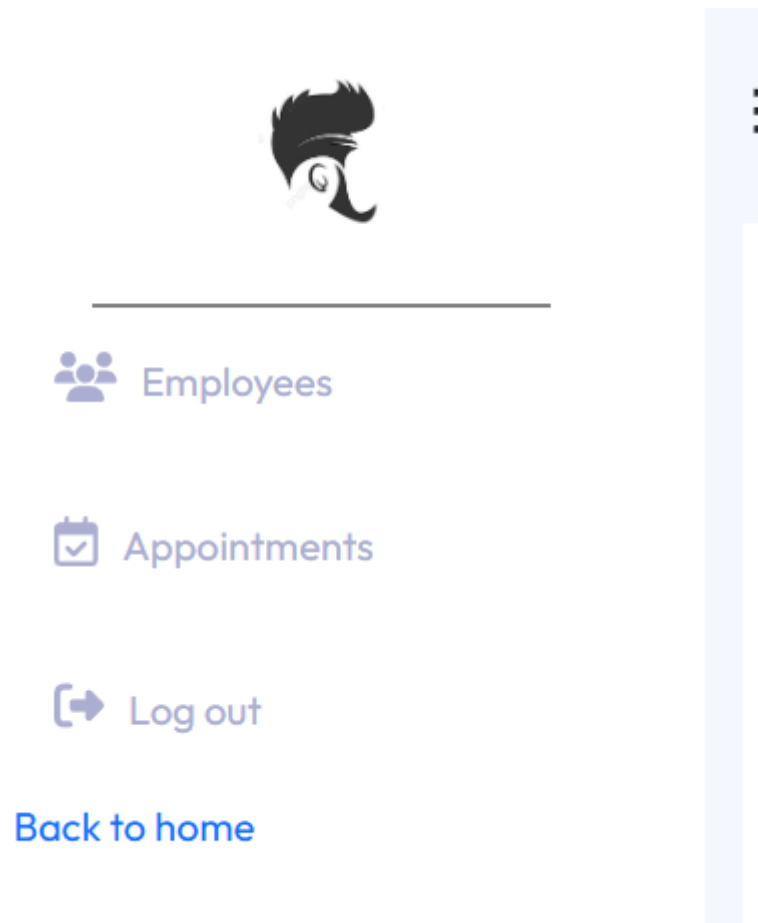


Figure 13 - Manager dashboard

Manager can only have access to Employees and Appointments. They also have limited permission like they only can see the employees and appointments that share the same location of store as them. They cannot create or edit the information of the employees but they can create and update the information of appointments.

5.1.2 Dashboard of admin

There are a few changes in the front end of the admin dashboard:

Figure 14 - Create/Update example Dialog

When click on create, edit or view, a dialog will appear to show the relevant information not navigating to other pages.

Some entities, have been updated to have a picture, will also have a choose file button to upload image.

Appointment creating, editing, viewing dialogs:

Figure 15 - Appointment Creating Dialog

The screenshot shows a web application interface. In the foreground, an 'Edit' dialog box is open, allowing an administrator to modify an appointment. The dialog includes the following fields and options:

- ID:** a113e1eb-83e9-43be-b41a-ec813887a2a2
- Worker:** a905768f-2eed-4381-a19d-f026
- Customer:** 474628c5-69fb-4ce5-b64a-4
- Status:** OVERDUE
- Date Time:** 24/01/2025 11:04
- Location:** 682cba49-0b42-4b90-91e1-e0

Below these fields are two selection lists:

- Available:** A list of services with search and selection buttons. One service is highlighted: ID: 0d25cb36-d960-4e8d-90c7-1e1dcf18fea0, Name: Combo Chăm Sóc Da Mặt Chuyên Sâu.
- Selected:** A list for services already added to the appointment.

The background shows a 'Dashboard' with a table of appointments and a sidebar with navigation icons.

Figure 16 - Appointment Editing Dialog

In the dialog of creating or editing appointment information, admin can change the location, the list of services/combos, the customer as they want.

Other entities have the same UI/UX as the appointment CRUD page.

5.2 Customer

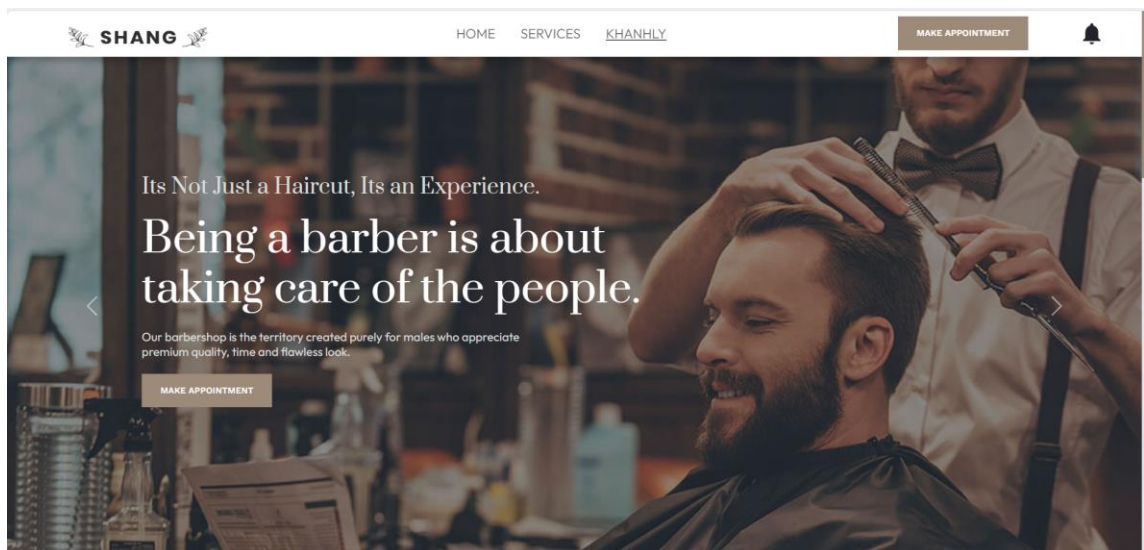


Figure 17 - Customer Home page

5.2.1 Service/Combo Carousel page

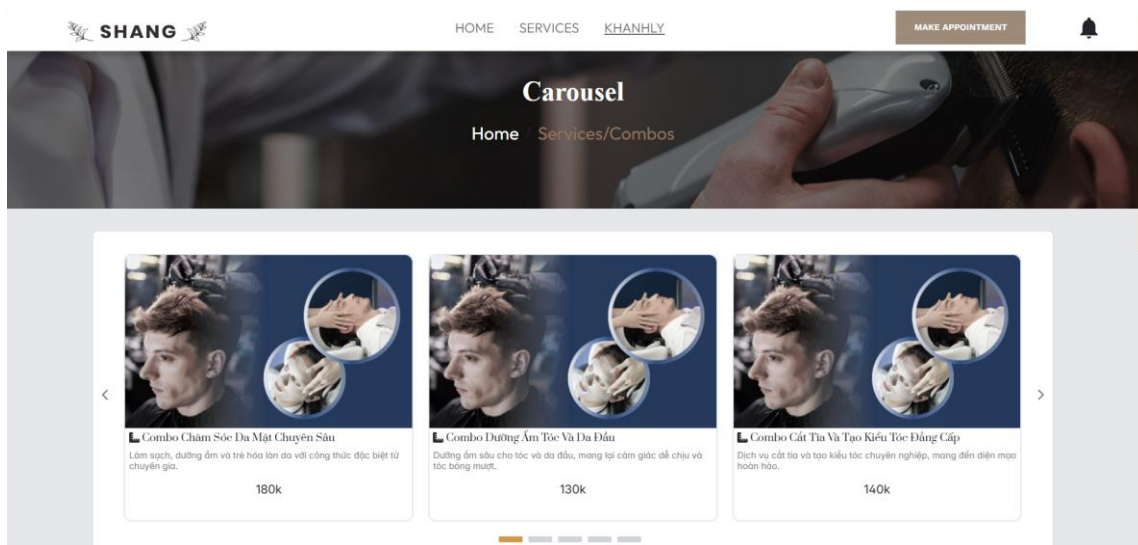


Figure 18 - Services/Combos Carousel Page

At this page, users can view all the available services and combos.

5.2.2 Booking page

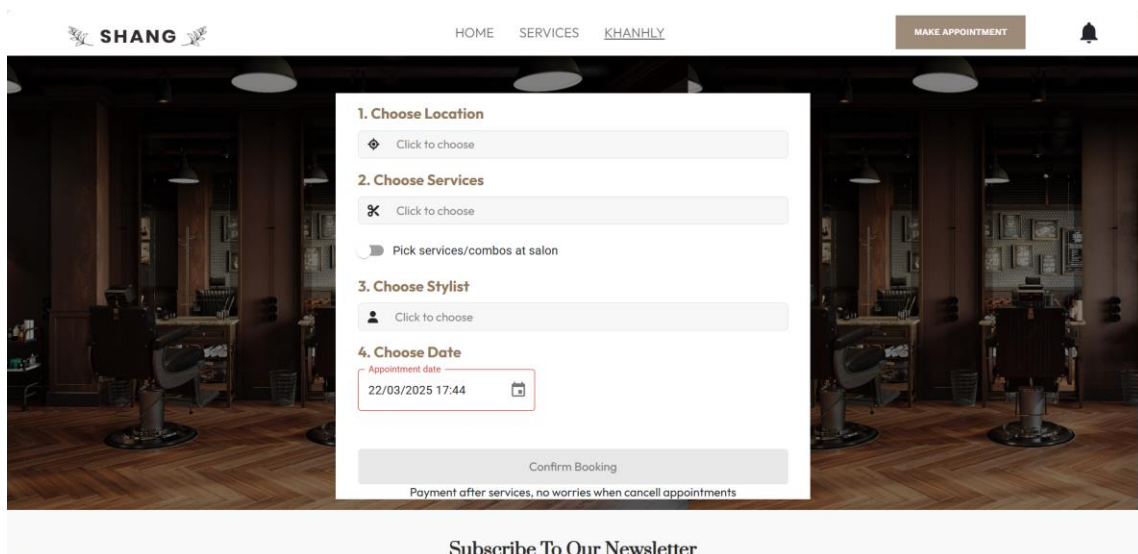


Figure 19 - Booking Page

There is only one change in this page, that is a button for customers to confirm that they do not want to book any services or combos, they want to book later at the shop.

5.3 Employee pages

Employees share the same home page and a few pages as customers.

The only different here is the Appointment history page:

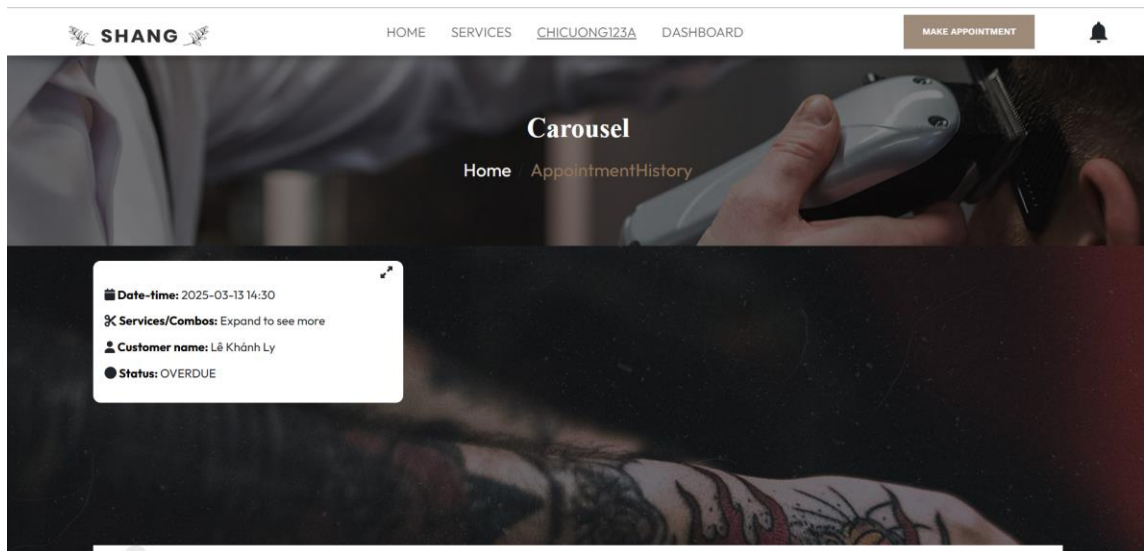


Figure 20 - Appointment history of Employee Page

Employees can see what customers they will serve. They cannot cancel the appointments but view the information.

5.4 Common pages

5.4.1 Profile page

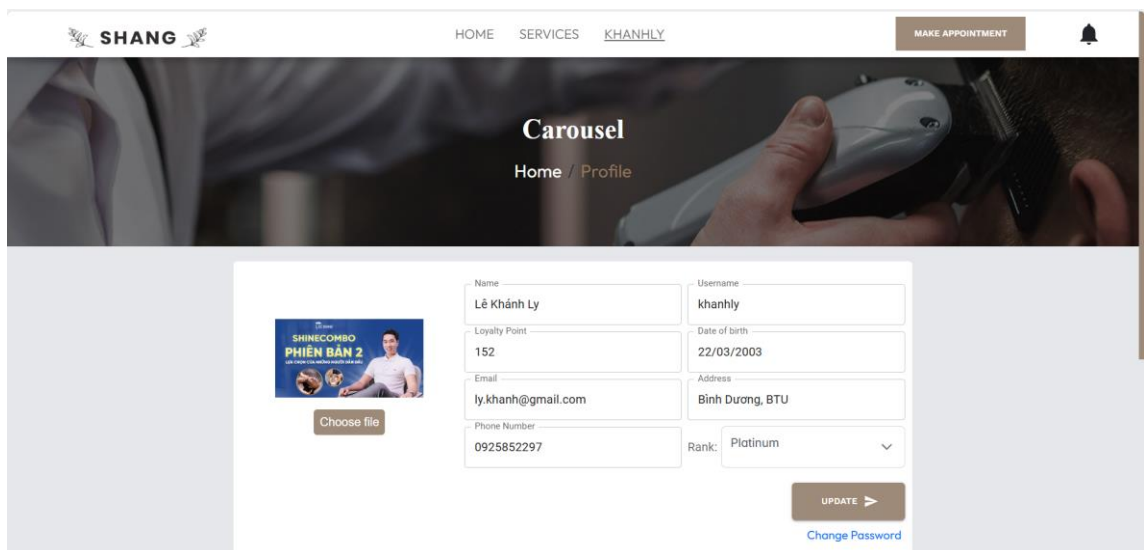
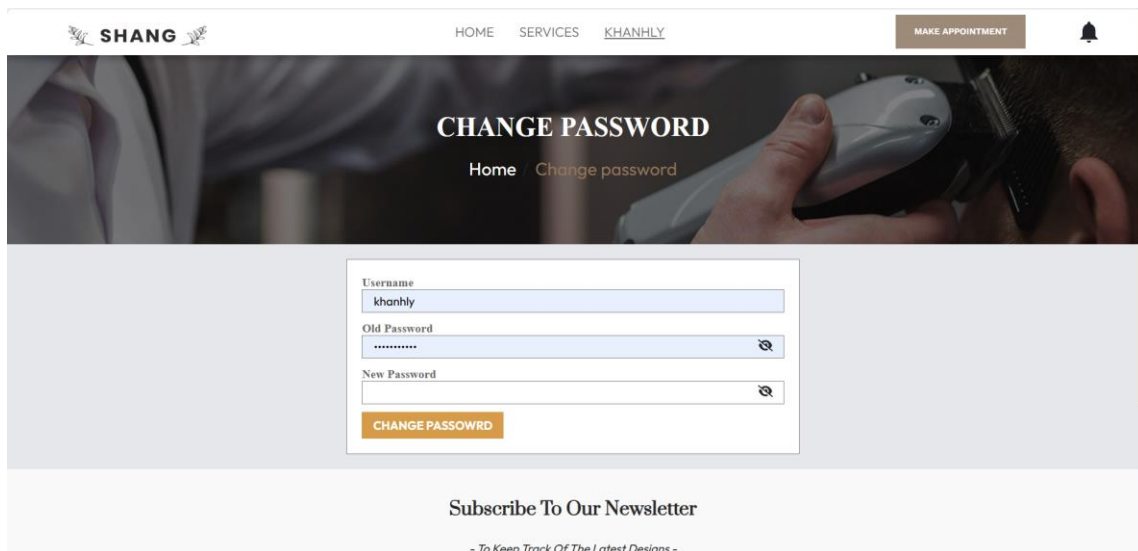


Figure 21 - Profile Page

This is the page that users can change their profile. The new here is that now users can change their avatar by uploading their own images.

5.4.2 Change password page



The screenshot shows the 'CHANGE PASSWORD' page of the SHANG website. The header includes the SHANG logo, navigation links for HOME, SERVICES, and KHANHLY, a MAKE APPOINTMENT button, and a notification bell icon. The main content area features a form with three input fields: Username (containing 'khanhly'), Old Password (masked with asterisks), and New Password (masked with asterisks). Below the form is a 'CHANGE PASSWORD' button. A secondary navigation bar shows 'Home' and 'Change password'. At the bottom, there is a newsletter subscription section with the text 'Subscribe To Our Newsletter' and a tagline '- To Keep Track Of The Latest Designs -'.

Figure 22 - Change Password Page

Users can change their passwords by given correct username and old password.

5.4.3 Notification

There are a few changes in the website. Admin, manager, employee and customer pages all have a few changes here. A notification bell had been added on the top bar.

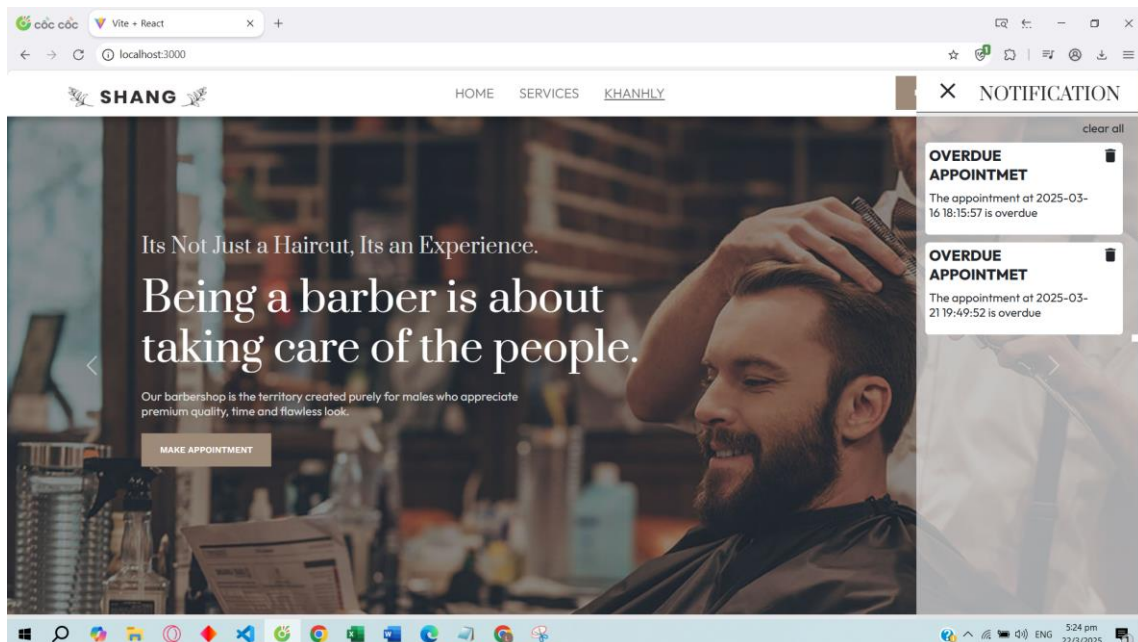


Figure 23 - Notification Panel

When users click on that bell icon, a list of notifications will appear if users have new notifications waiting for them.

Users can delete each notification or clear them all.

Admin/ Manager accounts can have a button to go to the dashboard.

5.5 Result Summary

The latest updates to the haircut booking and management system bring several key improvements compared to the previous version. The user interface has been enhanced to provide a more intuitive and visually appealing experience, making it easier for customers, employees, managers, and admins to navigate the system. Firebase notifications have been integrated, ensuring users stay informed about important updates, reminders, and salon news. Additionally, file storage has been moved to Google Drive, reducing server memory usage and improving overall system performance. These enhancements contribute to a more efficient, user-friendly, and scalable platform that better supports salon operations and customer engagement.

Chapter 6: CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

The CRUD page has undergone a significant upgrade, featuring a modernized UI/UX that enhances user experience with improved navigation and usability. The new design ensures that users can interact with the system more intuitively, reducing the time and effort required to manage data. Additionally, the page now includes enhanced functionalities that streamline operations, making CRUD processes more efficient.

One of the key improvements is the addition of a file uploading feature. Previously, the system relied on external storage links, which introduced dependencies and potential security risks. With the new implementation, files are now stored directly in an internal drive, ensuring greater control, security, and reliability. This change not only enhances data management but also reduces the risk of broken links and unauthorized access.

Furthermore, the system has been optimized to consume less memory, improving overall performance and responsiveness. This optimization ensures that the application runs smoothly even with increased data loads, making it more scalable and efficient for future expansions.

To enhance user engagement, additional features have been introduced for both customers and employees. Notifications have been integrated into the system, allowing users to stay informed about important updates, changes, and news. This ensures better communication and a more interactive experience, ultimately improving satisfaction and efficiency for all users.

6.2 Limitations

Despite the improvements, the system still has some limitations. Performance may be affected when handling extremely large datasets, requiring further optimization. The file storage feature, while more secure, may need additional backup solutions to prevent data loss. Notifications enhance communication, but reliance on real-time updates could lead to delays in certain cases. Future updates should focus on scalability, security enhancements, and refining user experience.

In the API, roles are fixed, meaning users are assigned predefined roles with specific permissions. This limits flexibility, as role customization is not supported. If new role requirements arise, modifications to the API may be needed, making it less adaptable to dynamic access control needs.

6.3 Future works

Future work will focus on enhancing role management by introducing customizable roles, allowing more flexibility in access control. Additionally, system optimization will continue to improve performance and reduce resource consumption. Expanding API functionalities, such as advanced search and filtering options, will enhance user experience. Further integration with cloud storage will ensure seamless file management. Lastly, incorporating AI-driven features, like automated recommendations and predictive analytics, will improve efficiency and decision-making.

REFERENCES

- [1] Hartman, J. (2024, November 21). *Introduction to Java*. Guru99, <https://www.guru99.com/introduction-to-java.html>
- [2] TheKnowledgeAcademy. (n.d.). *What is Java? A Complete Guide*, <https://www.theknowledgeacademy.com/blog/what-is-java/>
- [3] Bohnert, A. (2023, June 27). *What Is Java? Inside the World's Leading Programming Language*. HackerRank Blog, <https://www.hackerrank.com/blog/what-is-java-programming-language-introduction/>
- [4] Spring boot. (n.d.). Spring Boot, <https://spring.io/projects/spring-boot>
- [5] MySQL. (n.d.), <https://www.mysql.com/>
- [6] MySQL :: MySQL Documentation. (n.d.), <https://dev.mysql.com/doc/>
- [7] *JavaScript Guide - JavaScript / MDN*. (2023, May 1). MDN Web Docs, <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
- [8] *Learn JavaScript online - Courses for beginners - Javascript.com*. (n.d.), <https://www.javascript.com/>
- [9] *React – A JavaScript library for building user interfaces*. (n.d.). React, <https://legacy.reactjs.org/>
- [10] *SASS: Documentation*. (n.d.), <https://sass-lang.com/documentation/>
- [11] *W3Schools.com*. (n.d.), https://www.w3schools.com/Css/css_intro.asp
- [12] Orsini, M., & Orsini, M. (2023, July 5). *What is CSS: An Introduction to Cascading Style Sheets*. Blog Le Wagon, <https://blog.lewagon.com/skills/what-is-css/>
- [13] Lara, C. (2024, January 15). *What is CSS?* TheeDigital, <https://www.theedigital.com/blog/what-is-css>
- [14] Firebase, Google. (n.d.). Firebase, <https://firebase.google.com/>
- [15] *Google Drive API overview*. (n.d.). Google for Developers. <https://developers.google.com/drive/api/guides/about-sdk>