

GLAMOURQUEST HAIR SALON MANAGEMENT SYSTEM

Mini Project Report

Submitted by

GAYATHRY G NAIR

Reg. No.: AJC23MCA-2028

In Partial fulfillment for the Award of the Degree of
MASTER OF COMPUTER APPLICATIONS (MCA)

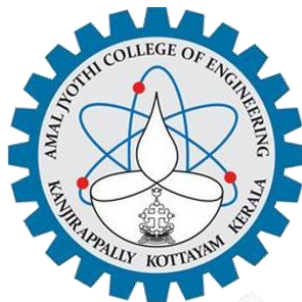


**AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS
KANJIRAPPALLY**

[Approved by AICTE, Accredited by NAAC, Accredited by NBA.
Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2024-2025

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**QLAMOURQUEST**” is the bonafide work of **GAYATHRY G NAIR (Regno: AJC23MCA-2028)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under **Amal Jyothi College of Engineering Autonomous, Kanjirappally** during the year 2024-25.

Ms. Lisha Varghese
Internal Guide

Binumon Joseph
Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

DECLARATION

I hereby declare that the project report “**GLAMOURQUEST**” is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications (MCA) from Amal Jyothi College of Engineering Autonomous during the academic year 2024-2025.

Date:
KANJIRAPPALLY

GAYATHRY G NAIR
Reg: AJC23MCA-2028

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Director (Administration) **Rev. Fr. Dr. Roy Abraham Pazhayaparampil** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev. Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Mr. Binumon Joseph** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Lisha Varghese** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

GAYATHRY G NAIR

ABSTRACT

The "GLAMOURQUEST" Hair Salon project is a web-based application built using Django to streamline salon operations. It has three main modules: User, Staff, and Admin. The User Module allows customers to manage profiles, browse services, book appointments, and make online payments. Users can view their appointment history, receive notifications, rate services, and cancel or reschedule appointments as needed. The system also sends email and SMS notifications for booking confirmations.

The Admin Module enables administrators to manage staff, services, and business performance through reports. Admins handle staff schedules and update service details. The Staff Module helps salon employees manage their profiles, view schedules, and handle appointments. Staff can update their availability and manage the services they offer. This system ensures smooth operations and a seamless experience for both customers and salon staff.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	3
2.1	INTRODUCTION	4
2.2	EXISTING SYSTEM	4
2.3	DRAWBACKS OF EXISTING SYSTEM	5
2.4	PROPOSED SYSTEM	5
2.5	ADVANTAGES OF PROPOSED SYSTEM	6
3	REQUIREMENT ANALYSIS	7
3.1	FEASIBILITY STUDY	8
3.1.1	ECONOMICAL FEASIBILITY	8
3.1.2	TECHNICAL FEASIBILITY	8
3.1.3	BEHAVIORAL FEASIBILITY	9
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	9
3.2	SYSTEM SPECIFICATION	12
3.2.1	HARDWARE SPECIFICATION	12
3.2.2	SOFTWARE SPECIFICATION	12
3.3	SOFTWARE DESCRIPTION	13
3.3.1	DJANGO	13
3.3.2	MYSQL	13
4	SYSTEM DESIGN	15
4.1	INTRODUCTION	16
4.2	UML DIAGRAM	16
4.2.1	USE CASE DIAGRAM	17
4.2.2	SEQUENCE DIAGRAM	19
4.2.3	STATE CHART DIAGRAM	20
4.2.4	ACTIVITY DIAGRAM	22
4.2.5	CLASS DIAGRAM	24
4.2.6	OBJECT DIAGRAM	26
4.2.7	COMPONENT DIAGRAM	27

4.2.8	DEPLOYMENT DIAGRAM	27
4.2.9	COLLABORATION DIAGRAM	28
4.3	USER INTERFACE DESIGN USING FIGMA	29
4.4	DATABASE DESIGN	32
5	SYSTEM TESTING	41
5.1	INTRODUCTION	42
5.2	TEST PLAN	42
5.2.1	UNIT TESTING	42
5.2.2	INTEGRATION TESTING	43
5.2.3	VALIDATION TESTING	43
5.2.4	USER ACCEPTANCE TESTING	44
5.2.5	AUTOMATION TESTING	44
5.2.6	SELENIUM TESTING	44
6	IMPLEMENTATION	56
6.1	INTRODUCTION	57
6.2	IMPLEMENTATION PROCEDURE	57
6.2.1	USER TRAINING	58
6.2.2	TRAINING ON APPLICATION SOFTWARE	58
6.2.3	SYSTEM MAINTENANCE	59
7	CONCLUSION & FUTURE SCOPE	60
7.1	CONCLUSION	61
7.2	FUTURE SCOPE	62
8	BIBLIOGRAPHY	63
9	APPENDIX	65
9.1	SAMPLE CODE	66
9.2	SCREEN SHOTS	87

List of Abbreviations

- UML - Unified Modelling Language
- ORM - Object-Relational Mapping
- MVT - Model-View-Template
- MVC - Model-View-Controller
- RDBMS - Relational Database Management System
- 1NF - First Normal Form
- 2NF - Second Normal Form
- 3NF - Third Normal Form
- IDE - Integrated Development Environment
- HTML - HyperText Markup Language
- JS - JavaScript
- CSS - Cascading Style Sheets
- API - Application Programming Interface
- UI - User Interface
- HTTP - Hypertext Transfer Protocol
- URL - Uniform Resource Locator
- PK - Primary Key
- FK - Foreign Key
- SQL - Structured Query Language
- CRUD - Create, Read, Update, Delete

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The "GLAMOURQUEST" project is designed to simplify salon management through a user-friendly web application. It offers three main sections for Customers, Staff, and Administrators. Customers can manage their profiles, browse salon services, book appointments, and make online payments. They can also view appointment history, receive notifications, and rate or reschedule services if needed, making their salon experience smooth and convenient.

For Administrators, the project provides tools to manage staff and services, update service details, and organize staff schedules, while tracking business performance through reports. Staff members can manage their profiles, update availability, and handle their appointments efficiently. Overall, the system ensures efficient operations and an enhanced experience for both salon customers and staff.

1.2 PROJECT SPECIFICATION

User Roles:

- **Customer (User Module):**
 - Registration, login, and profile management.
 - Browse services with detailed descriptions and rates.
 - Book and manage appointments (schedule, reschedule, or cancel).
 - Online payments via integrated payment gateway.
 - View appointment history and receive email/SMS notifications.
 - Rate and review services.
- **Staff (Staff Module):**
 - Profile management including personal information and work schedule.
 - View and update appointment calendar.
 - Manage service availability and preferences.
 - Update availability for bookings.
- **Administrator (Admin Module):**
 - Manage staff profiles, schedules, and roles.
 - Update and manage service offerings (descriptions, rates).
 - Generate reports on booking statistics, revenue, and performance.
 - Manage user access and security settings.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The "GLAMOURQUEST" project is a web-based system developed using Django to enhance and streamline the operational processes of a salon. The system is designed to meet the needs of three key user groups: customers, staff, and administrators. By providing an online platform for booking appointments, managing profiles, and handling payments, the system aims to offer a convenient, user-friendly experience for salon clients while ensuring efficient salon management.

The system study focuses on understanding the requirements and functionalities needed to meet the specific needs of each user group. For customers, the system provides features like service browsing, appointment scheduling, notifications, and feedback options. Administrators are equipped with tools to manage staff, services, and monitor business performance through reports. Staff members are able to handle their schedules, appointments, and service availability through their personalized dashboards. This study analyzes the system's design and functionality to ensure a seamless experience across all user roles, contributing to the salon's operational efficiency and improved customer satisfaction.

2.2 EXISTING SYSTEM

2.2.1 NATURAL SYSTEM STUDIED

The natural system studied for the "GLAMOURQUEST" project focuses on the traditional operations of a salon, where customers physically visit to book appointments, inquire about services, and make payments. Salon staff manually manage schedules and customer bookings, while administrators handle day-to-day operations like service management, staff coordination, and business performance monitoring. By studying this natural system, the project aims to automate and enhance these processes through a web-based platform, providing a more efficient and accessible solution for both customers and salon staff.

2.2.2 DESIGNED SYSTEM STUDIED

The designed system for the "GLAMOURQUEST" project automates and enhances the traditional salon operations by introducing a web-based platform with three main modules: User, Staff, and Admin. The User Module allows customers to manage profiles, browse services, book appointments, and make payments online. It also sends notifications and enables customers to rate services and manage their bookings. The Admin Module helps administrators manage staff, services, and track business performance, while the Staff Module enables employees to manage their profiles, schedules, and appointments. This system ensures smoother operations, greater convenience for customers, and better administrative control for salon management.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Time-Consuming Processes:** Customers need to visit or call the salon to book appointments, which can lead to delays and inefficiencies, especially during busy hours.
- **Limited Accessibility:** There is no online platform for customers to view services, book appointments, or make payments at their convenience. Everything must be handled in person or over the phone.
- **Manual Record Keeping:** Staff and administrators rely on paper records or basic spreadsheets to manage appointments, customer information, and staff schedules, which increases the chances of human error and data mismanagement.
- **No Automated Notifications:** Customers don't receive automatic reminders or updates about their appointments, leading to missed or forgotten appointments.
- **Lack of Real-Time Updates:** Scheduling conflicts can arise since there's no real-time system to track staff availability or appointment changes.
- **Limited Customer Feedback:** Gathering customer feedback, reviews, or ratings is often overlooked or done manually, making it difficult to assess service quality.
- **Inefficient Service Management:** Admins have to manually adjust staff schedules, service details, and pricing, which can be tedious and prone to mistakes.

2.4 PROPOSED SYSTEM

The proposed "GLAMOURQUEST" system is a web-based solution designed to overcome the inefficiencies of traditional salon operations. By automating processes, the system offers a more streamlined experience for customers, staff, and administrators. Through the **User Module**, customers can easily manage their profiles, browse services, book appointments, and make online payments. They receive automated notifications for booking confirmations and reminders, and can also rate services, view their appointment history, and reschedule or cancel bookings if needed. The **Staff Module** enables salon employees to manage their profiles, view schedules, and handle appointments, allowing them to better control their availability and service offerings. The **Admin Module** provides administrators with tools to manage staff profiles, update service details, and track business performance through comprehensive reports. This system eliminates manual processes, reduces errors, and enhances overall efficiency, leading to a better experience for both customers and salon management.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- **Efficiency and Automation:** The proposed system automates appointment bookings, notifications, and payments, reducing the need for manual processes and minimizing human error.
- **24/7 Accessibility:** Customers can browse services, book appointments, and make payments online at any time, providing greater convenience and flexibility compared to traditional methods.
- **Real-Time Updates:** Staff availability, appointment schedules, and service details are updated in real time, ensuring that customers and staff always have accurate and up-to-date information.
- **Improved Customer Experience:** Customers receive automated email and SMS notifications for appointment confirmations and reminders, allowing them to easily manage their bookings, providing a seamless and convenient experience.
- **Enhanced Service Management:** Administrators can efficiently manage staff schedules, update service offerings, and generate reports on business performance, allowing for better decision-making and optimized operations.
- **Feedback and Ratings:** The system allows customers to provide feedback and rate services, enabling the salon to assess customer satisfaction and improve service quality.
- **Reduced Scheduling Conflicts:** The system provides real-time tracking of staff availability and appointments, helping avoid double bookings and scheduling conflicts.
- **Secure Online Payments:** Integrated payment gateways enable customers to make secure online payments, improving convenience and reducing payment-related delays.
- **Data Management:** All customer, staff, and service data is stored in a centralized database, ensuring organized and efficient record-keeping and easy access to relevant information.
- **Scalability:** The system can easily scale with growing customer demand, staff expansion, or additional services, making it adaptable for future growth.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

The feasibility study for the "GLAMOURQUEST" project indicates strong economic viability, leveraging open-source technologies such as Python Django for the backend and HTML/CSS for the frontend, significantly reducing development costs and ensuring low operational expenses. Technically, the project is well-equipped with a reliable technology stack, including MySQL for efficient data management and payment gateway integration for secure online transactions, fulfilling all necessary requirements for seamless operation. Furthermore, operational feasibility is enhanced by a user-friendly interface designed to minimize resistance among customers, staff, and administrators. The system facilitates easy profile management, appointment scheduling, and service tracking, while automated notifications for appointments and payments streamline salon operations. Overall, the project demonstrates solid economic, technical, and operational feasibility, positioning it for successful implementation.

3.1.1 Economical Feasibility

The economic feasibility of the "GLAMOURQUEST" project is favorable due to its reliance on open-source technologies, which significantly reduce development costs. The backend is built using Python Django, while the frontend is developed with HTML/CSS, both of which are free to use. This approach minimizes initial investment, and ongoing operational costs are expected to remain low. Additionally, the system's potential to increase customer engagement and streamline salon operations could lead to enhanced revenue generation, making the project economically viable.

3.1.2 Technical Feasibility

The technical feasibility of the project is robust, as it employs a well-established technology stack. Python Django is chosen for backend development due to its scalability, security, and ease of use. The frontend, built with HTML/CSS, ensures compatibility across various devices and browsers. The use of MySQL as the database supports efficient data management for customer profiles, appointment bookings, and staff information. Integration with payment gateways for secure online transactions is also planned, fulfilling all technical requirements necessary for the successful operation of the salon management system.

3.1.3 Behavioral Feasibility

Behavioral feasibility assesses how well the " GLAMOURQUEST " project aligns with the needs, expectations, and behaviors of its users, including customers, staff, and administrators. The system is designed with a user-friendly interface that simplifies navigation and encourages engagement, making it accessible to individuals with varying levels of technical expertise. Customers can easily manage their profiles, book appointments, and make payments, which enhances their overall experience and satisfaction. Staff members benefit from streamlined appointment management and scheduling features, fostering better communication and collaboration within the salon. Additionally, administrators can efficiently oversee operations, manage staff, and analyze performance metrics. By focusing on user experience and facilitating smooth interactions, the project promotes a positive adoption rate among all users, thereby increasing the likelihood of successful implementation and sustained usage.

3.1.4 Feasibility Study Questionnaire

1)Project-Overview:

The "GLAMOURQUEST" project is a web-based application developed using the Django framework. It aims to streamline salon operations by offering a comprehensive system for customers, staff, and administrators. The main objectives are to enhance customer convenience through online appointment management and payments, improve salon efficiency with robust staff and service management, and provide administrators with tools for analyzing performance and managing staff.

2)System-Scope:

The system is proposed as a full-scale implementation designed for practical use in operational salons. It is not a research prototype but a complete solution intended for day-to-day use in managing salon services, appointments, and staff.

3)Target-Audience:

The primary users of the system include:

- **Customers:** Who will use the system to manage their profiles, book appointments, make payments, and provide feedback.
- **Salon Staff:** Who will manage their schedules, handle appointments, and track performance.
- **Administrators:** Who will oversee staff management, service offerings, promotions, and business analytics.

4) Modules:

1. **User Module:** Manages customer profiles, service browsing, payments, appointment history, notifications, reviews, and cancellations.
2. **Admin Module:** Handles staff management, service management, promotions, analytics, and security settings.
3. **Staff Module:** Includes profile management, schedule management, service management, appointment management, and performance tracking.

5) User Roles:

1. **Customer:** Can manage personal information, book and manage appointments, make payments, and provide feedback.
2. **Staff:** Can manage their own profiles, view and update schedules, manage services offered, and track performance.
3. **Administrator:** Can manage staff, services, promotions, analyze reports, and set security permissions.

6)System-Ownership:

The system ownership will be with [Owner's Name or Entity], which could be an individual, an organization, or a salon chain.

7)Industry/Domain:

The system is related to the beauty and personal care industry, specifically focusing on hair salons.

8) Data Collection Contacts:

- **Name:** Akshay S
- **Role:** Staff
- **Contact Information:** 8943290272

9) Questionnaire for Data Collection:

1. What are your primary needs for managing salon appointments online?
 - To streamline the booking process, reduce no-shows through automated reminders, and allow customers to easily manage their appointments.
2. How do you currently handle appointment bookings and payments?
 - Currently, appointments are booked over the phone or in person, and payments are made in cash or through card transactions at the salon.
3. What features do you find most important in a salon management system?
 - Online booking, payment processing, appointment reminders, staff scheduling, customer feedback management, and performance analytics.
4. How do you track and manage customer feedback and reviews?

- Customer feedback is collected through paper forms at the salon and manually reviewed by the staff.
- 5. What kind of promotions or discounts do you regularly offer?
 - We offer seasonal discounts, loyalty rewards, and occasional promotional packages for special events.
- 6. How do you manage and schedule staff shifts and availability?
 - Staff schedules are managed manually using a calendar and communicated via group messages or calls.
- 7. What types of reports or analytics would be most useful for your salon?
 - Booking statistics, revenue reports, staff performance metrics, and customer satisfaction ratings.
- 8. Are there any specific security concerns you have for managing staff and customer data?
 - Ensuring that customer payment information is secure, protecting customer and staff personal data, and controlling access to sensitive information within the system.
- 9. How do you currently handle cancellations and rescheduling of appointments?
 - Cancellations and rescheduling are handled manually over the phone or in person, often resulting in scheduling conflicts and errors.
- 10. What additional features or functionalities would you like to see in an online salon management system?
 - Integration with social media for marketing, automated follow-ups with customers, mobile app access for easier management, and a loyalty program system.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor: Intel Core i5

RAM: 16 GB

Hard Disk: 1 TB

3.2.2 Software Specification

Front End: HTML, CSS

Back End: Python - Django

Database: MySQL

Client on PC: Windows 7 and above

Technologies Used: JavaScript, HTML5, AJAX, jQuery, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 Django

Django is a popular and powerful open-source web framework written in Python, designed to facilitate rapid development and maintainable web applications. It follows the Model-View-Template (MVT) architectural pattern, which is similar to the Model-View-Controller (MVC) pattern. Django provides a structured and efficient way to build web applications, offering several key components and features. At its core, Django includes a robust Object-Relational Mapping (ORM) system that simplifies database interactions, allowing developers to work with Python objects instead of raw SQL queries. It also includes a URL dispatcher for mapping URLs to view functions, an automatic admin interface for managing application data, and a templating engine for creating dynamic and reusable user interfaces. Django places a strong emphasis on security, with built-in features to protect against common web vulnerabilities, including authentication and authorization systems, middleware support for global request and response processing, and compatibility with various databases.

3.3.2 MySQL

MySQL is an open-source relational database management system (RDBMS) that utilizes structured query language (SQL) for managing and manipulating data. It is known for its reliability, scalability, and ease of use, making it one of the most popular databases for web applications. MySQL supports a wide range of data types and provides powerful features such as transactions, indexing, and stored procedures. It integrates seamlessly with Django through its ORM, allowing developers to create, read, update, and delete records effortlessly while maintaining data integrity and security. MySQL's performance and robust transaction support make it an excellent choice for handling the data needs of web applications, including those developed with Django.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The initial stage of developing any engineered product or system is the design phase, which involves a creative approach. A well-crafted design plays a critical role in ensuring the successful functioning of a system. Design is defined as the process of employing various techniques and principles to define a process or system in enough detail to enable its physical realization. This involves using different methods to describe a machine or system, explaining how it operates, in sufficient detail for its creation. In software development, design is a crucial step that is always present, regardless of the development approach. System design involves creating a blueprint for building a machine or product. Careful software design is essential to ensure optimal performance and accuracy. During the design phase, the focus shifts from the user to the programmers or those working with the database. The process of creating a system typically involves two key steps: Logical Design and Physical Design.

4.2 UML DIAGRAM

Unified Modeling Language (UML) is a standardized language used for specifying, visualizing, constructing, and documenting the elements of software systems. Developed by the Object Management Group (OMG), the first draft of the UML 1.0 specification was presented in January 1997. Unlike programming languages such as C++, Java, or COBOL, UML is not a programming language but rather a graphical language that serves as a tool for creating software blueprints.

UML is a versatile visual modeling language that facilitates the representation and understanding of both software and non-software systems. While its primary application lies in modeling software systems, it can also represent processes in various contexts, such as manufacturing. UML is not directly a programming language but can be integrated with tools that generate code in different programming languages based on UML diagrams. It encompasses nine core diagrams that aid in representing various aspects of a system, including:

- Class Diagram
- Object Diagram
- Use Case Diagram
- Sequence Diagram
- Activity Diagram
- State Chart Diagram
- Deployment Diagram
- Component Diagram

4.2.1 USE CASE DIAGRAM

A use case is a crucial tool for understanding and organizing a system's requirements, particularly in contexts such as developing a product delivery website. Use cases are visually represented using "use case" diagrams in the Unified Modeling Language (UML), which provides a standardized method for modeling real-world systems and processes.

A use case diagram consists of several main elements:

- **Boundary:** This element defines the system's scope, distinguishing it from external entities.
- **Actors:** These represent individuals or entities that play specific roles within the system.
- **Interactions:** This illustrates the relationships and interactions between different actors and the system in various scenarios.

The primary purpose of use case diagrams is to document a system's functional specifications. To create an effective use case diagram, certain guidelines should be followed:

- Use clear and meaningful names for use cases and actors.
- Ensure that relationships and dependencies are well-defined.
- Include only necessary relationships to maintain the diagram's clarity.
- Utilize explanatory notes as needed to clarify essential details.

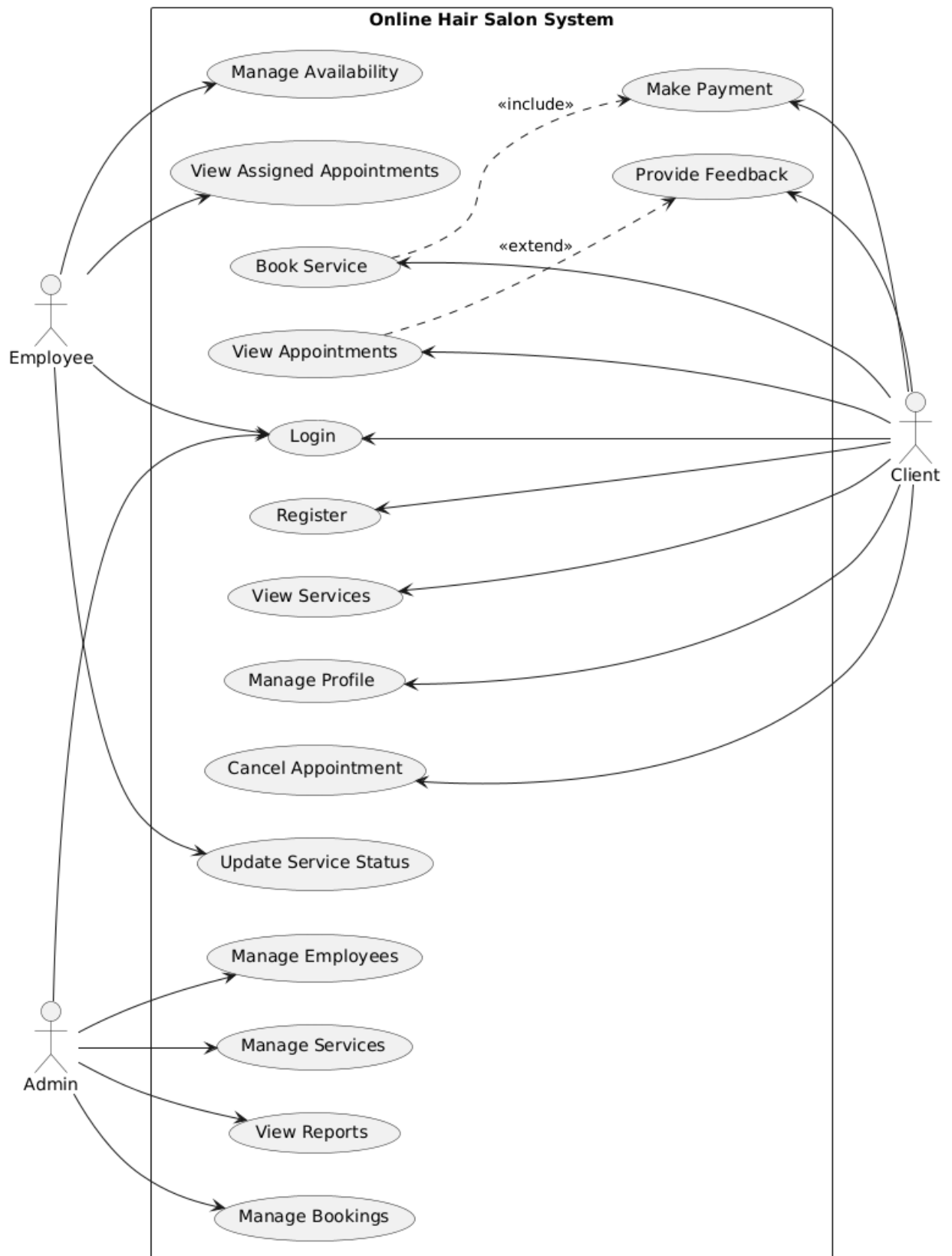


Fig: 4.2.1

4.2.2 SEQUENCE DIAGRAM

A sequence diagram illustrates the specific order in which objects interact with each other, showcasing the sequential flow of events. This type of diagram is also known as event diagrams or event scenarios. Sequence diagrams serve the purpose of elucidating how various components of a system collaborate and the precise sequence in which these actions occur. These diagrams are frequently used by business professionals and software developers to aid in understanding and depicting requirements for both new and existing systems.

Sequence Diagram Notations

1. Actors

In a UML diagram, actors represent individuals who utilize the system and its components. Actors are not depicted within the UML diagram as they exist outside the system being modeled. They serve as role-players in a story, encompassing both people and external entities. In a UML diagram, actors are represented by simple stick figures. Multiple individuals can be depicted in a diagram that portrays the sequential progression of events.

2. Lifelines

Each element in a sequence diagram is presented as a lifeline, with lifeline components positioned at the top of the diagram.

3. Messages

Communication between objects is achieved through the exchange of messages. Messages are arranged sequentially along the lifeline, with arrows representing the flow of messages, forming the core structure of a sequence diagram.

4. Guards

Guards are used within UML to denote various conditions that restrict messages if specific conditions are met. They provide software developers with insights into the rules governing a system or process.

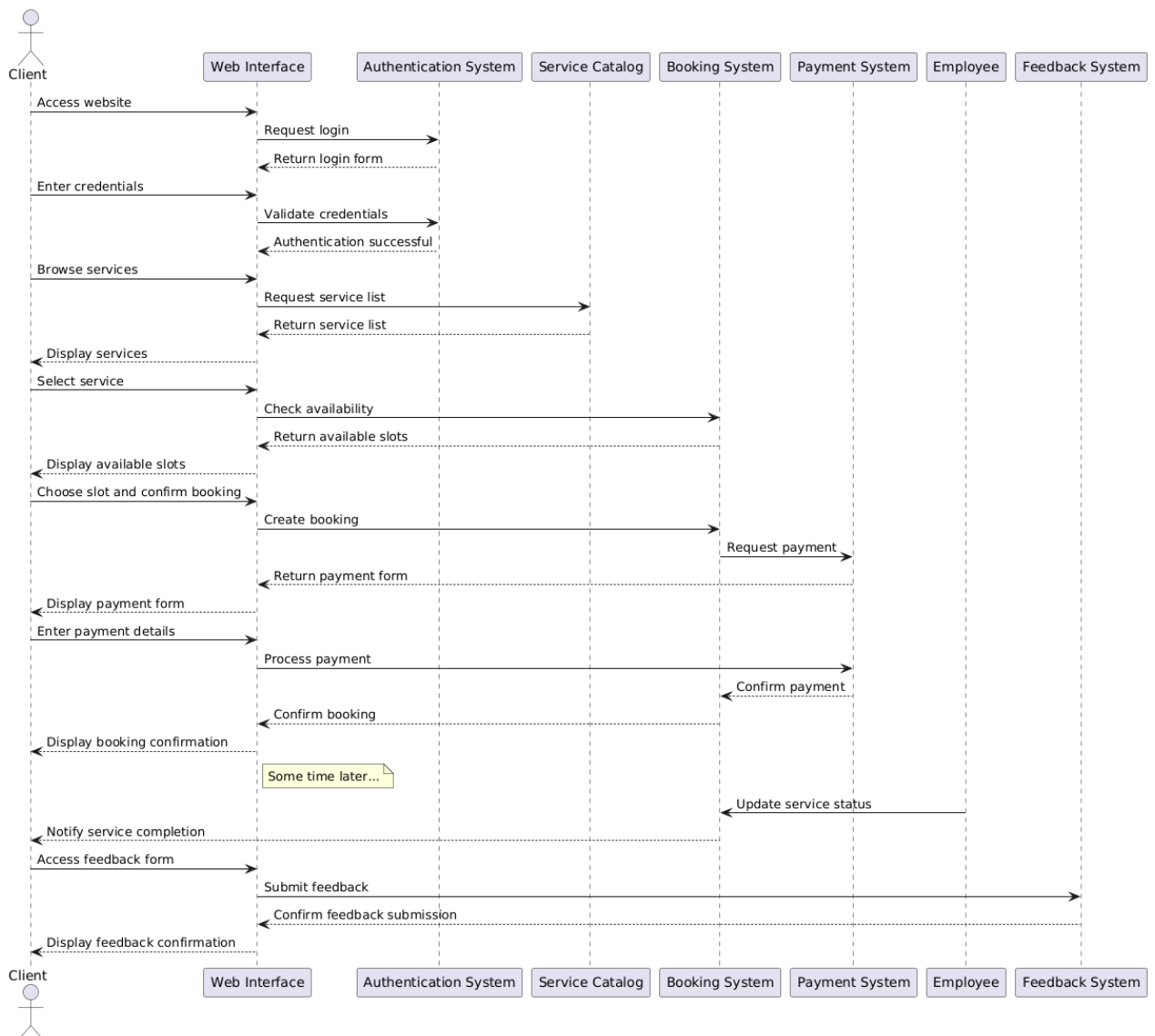


Fig: 4.2.2

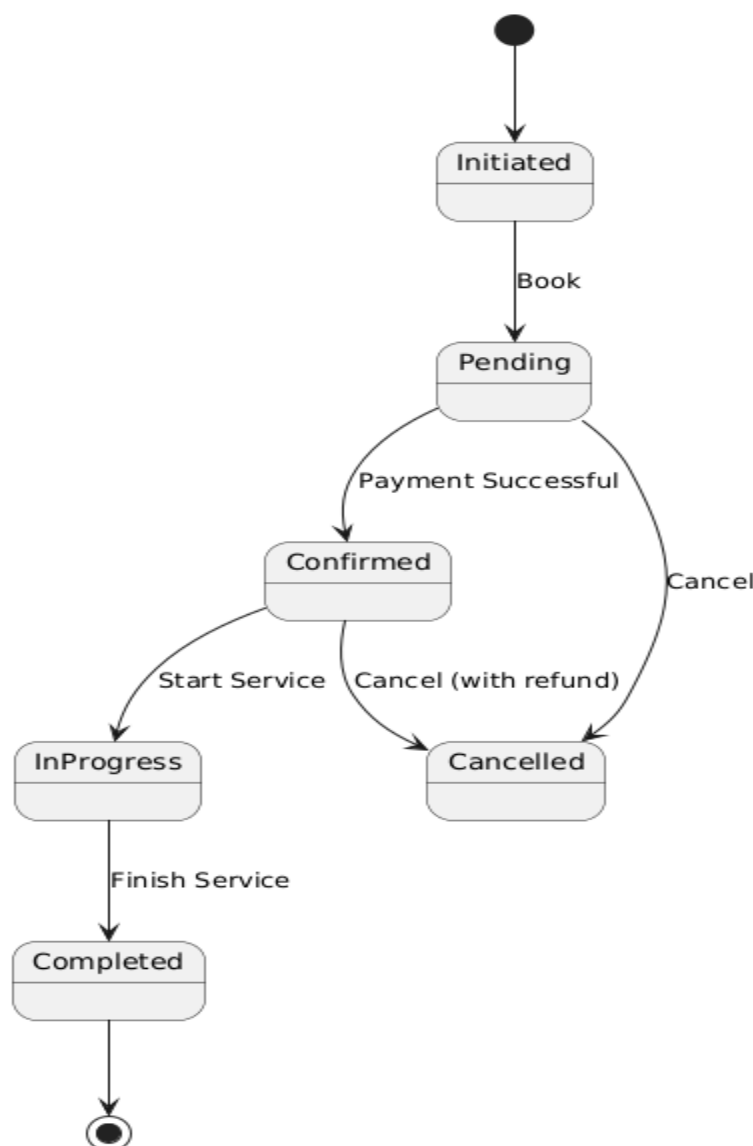
4.2.3 State Chart Diagram

A state machine diagram, also known as a state chart, visually represents the various states an object undergoes within a system and the sequence in which these states are traversed. It serves as a record of the system's behavior, illustrating the collaborative functioning of a group of entities—whether it's a team, a collection of students, a large assembly, or an entire organization. State machine diagrams are valuable for depicting the interactions of diverse components within a system, outlining how objects evolve in response to events and elucidating the various conditions that each entity or component can inhabit.

Notations within a State Machine Diagram

- **Initial State:** Symbolized by a black circle, this indicates the commencement of a process.

- **Final State:** Representing the conclusion of a process, this is denoted by a filled circle within another circle.
- **Decision Box:** Shaped like a diamond, this aids in decision-making by evaluating a guard condition.
- **Transition:** A transition occurs whenever there is a change in authority or state due to an event. Transitions are depicted as arrows with labels indicating the triggering event.
- **State Box:** This portrays the condition or state of an element within a group at a specific moment.

**Fig: 4.2.3**

4.2.4 Activity Diagram

An activity diagram is a visual representation of how events unfold simultaneously or sequentially. It aids in understanding the flow of activities, emphasizing the progression from one task to the next. Activity diagrams focus on the order in which tasks occur and can depict various types of flows, including sequential, parallel, and alternative paths. To facilitate these flows, activity diagrams incorporate elements such as forks and join nodes, aligning with the concept of illustrating the functioning of a system in a specific manner.

Key Components of an Activity Diagram

- a) **Activities:** Activities group behaviors into one or more actions, forming a network of interconnected steps. Lines between these actions outline the step-by-step sequence of events. Activities encompass tasks, controlling elements, and resources utilized in the process.
- b) **Activity Partition/Swim Lane:** Swim lanes are used to categorize similar tasks into rows or columns, enhancing modularity in the activity diagram. They can be arranged vertically or horizontally, though they are not mandatory for every activity diagram.
- c) **Forks:** Fork nodes enable the simultaneous execution of different segments of a task. They represent a point where one input transforms into multiple outputs, resembling the diverse factors influencing a decision.
- d) **Join Nodes:** Join nodes are distinct from fork nodes and employ a logical AND operation to ensure that all incoming data flows converge to a single point, promoting synchronization.

Notations in an Activity Diagram

- **Initial State:** Depicts the beginning or first step in the process.
- **Final State:** Signifies the completion of all actions with no further progress.
- **Decision Box:** Ensures that activities follow a specific path based on a decision.
- **Action Box:** Represents the specific tasks or actions to be performed in the process.

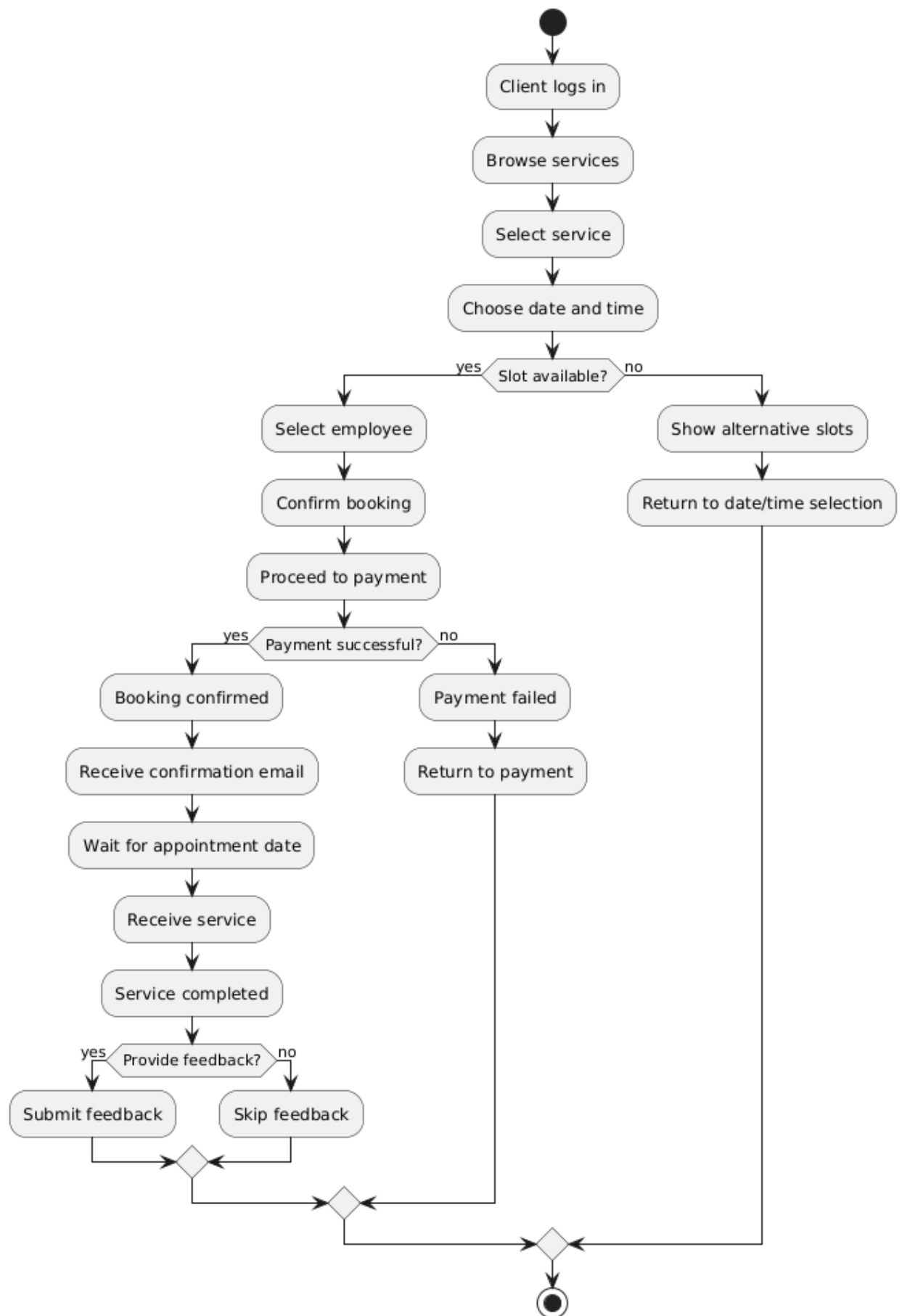


Fig: 4.2.4

4.2.5 Class Diagram

A class diagram serves as a static blueprint for an application, illustrating its components and their relationships when the system is in a dormant state. It provides insights into the system's structure, showcasing the various elements it comprises and how they interact. In essence, a class diagram acts as a visual guide for software development, aiding in the creation of functional applications.

Key Aspects of a Class Diagram

- **System_Overview:**

A class diagram offers a high-level representation of a software system, presenting its constituent parts, associations, and collaborative dynamics. It serves as an organizational framework for elements such as names, attributes, and methods, simplifying the software development process.

- **Structural_Visualization:**

The class diagram is a structural diagram that combines classes, associations, and constraints to define the system's architecture.

Components of a Class Diagram

The class diagram comprises three primary sections:

- **Upper_Section:**

This top segment features the class name, representing a group of objects that share common attributes, behaviors, and roles. Guidelines for displaying groups of objects include capitalizing the initial letter of the class name, positioning it in the center, using bold lettering, and employing a slanted writing style for abstract class titles.

- **Middle_Section:**

In this part, the class's attributes are detailed, including their visibility indicators, denoted as public (+), private (-), protected (#), or package (~).

- **Lower_Section:**

The lower section elaborates on the class's methods or operations, presented in a list format with each method on a separate line. It outlines how the class interacts with data.

UML Relationships within a Class Diagram

Relationships in a class diagram fall into several categories:

- **Dependency:**

Signifies the influence of one element's changes on another.

- **Generalization:**

Represents a hierarchical relationship where one class acts as a parent, and another serves

as its child.

- **Association:**

Indicates connections between elements.

- **Multiplicity:**

Defines constraints on the number of instances allowed to possess specific characteristics, with one being the default value when not specified.

- **Aggregation:**

An aggregation is a group that is a part of a relationship called association.

- **Composition:**

"Composition" is a stronger form of "aggregation," describing how a parent and child need each other. If one is removed, the other cannot function.

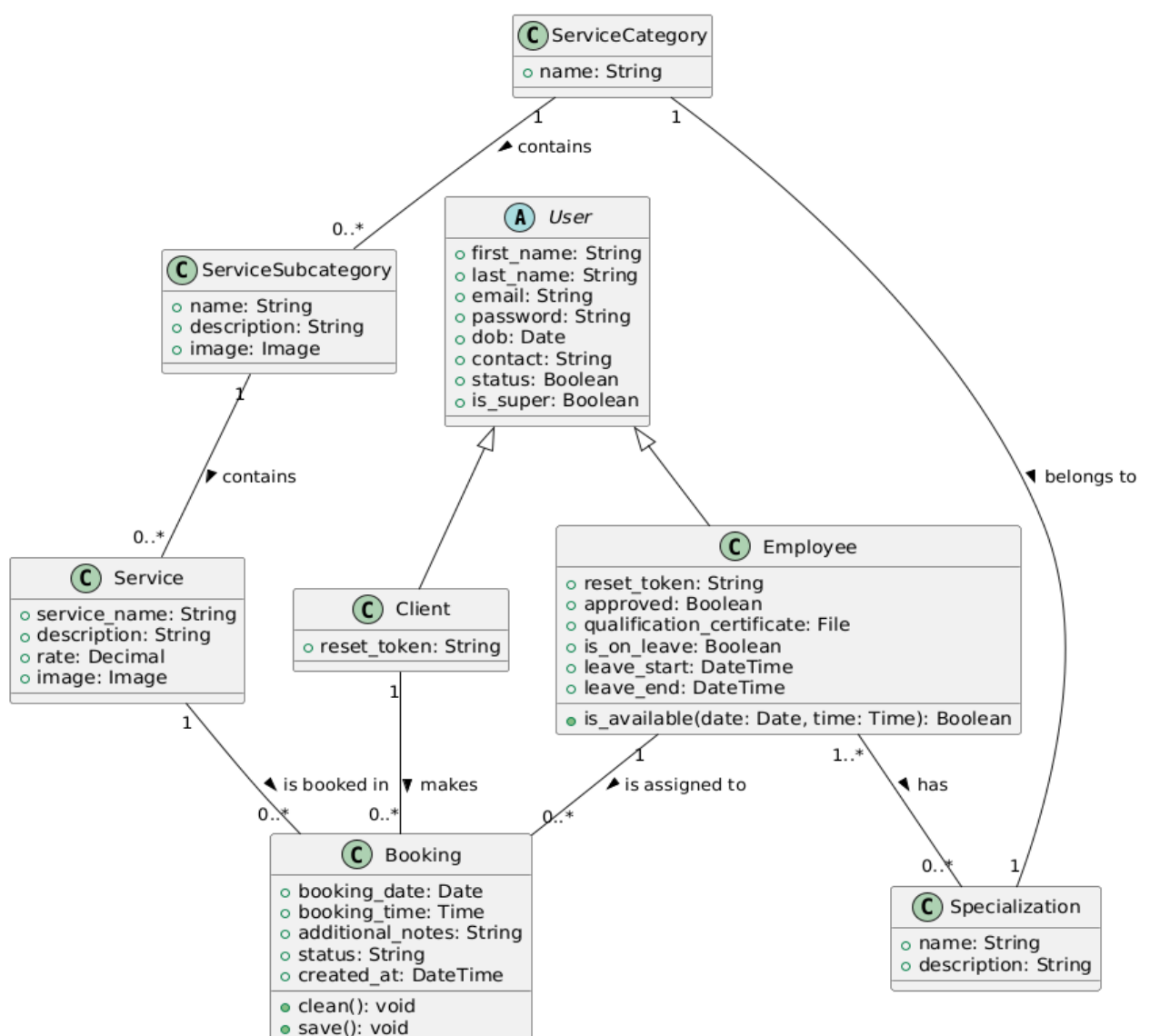


Fig: 4.2.5

4.2.6 Object Diagram

Object diagrams are derived from class diagrams and rely on them to provide a visual representation. They offer an illustration of a collection of objects related to a particular class. Object diagrams provide a snapshot of objects in an object-oriented system at a specific point in time.

Object diagrams and class diagrams share similarities, but they also have distinctions. Class diagrams are more generalized and do not portray specific objects. This abstraction in class diagrams simplifies the comprehension of a system's functionality and structure.

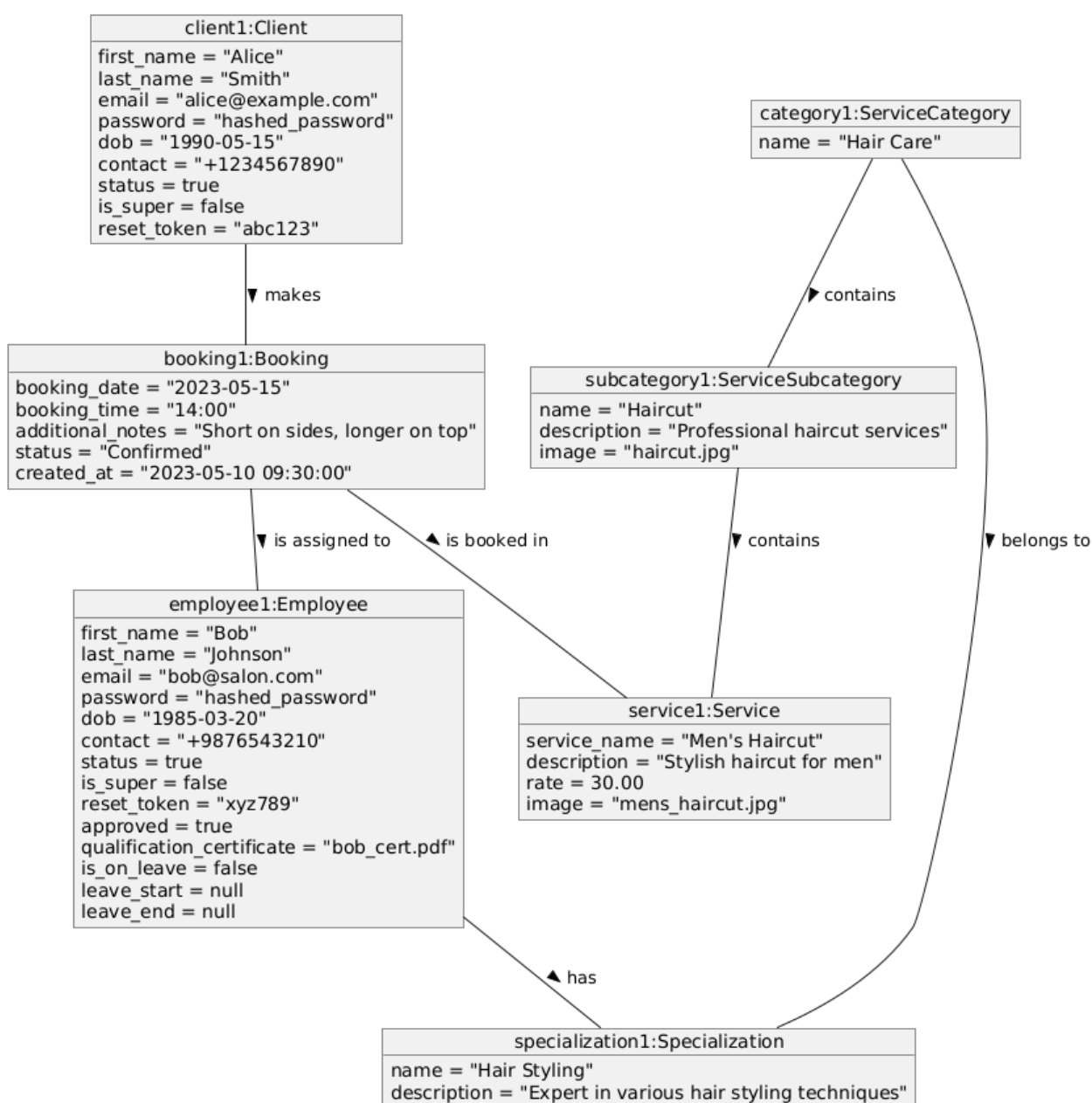


Fig: 4.2.6

4.2.7 Component Diagram

A component diagram serves the purpose of breaking down a complex system that utilizes objects into more manageable segments. It offers a visual representation of the system, showcasing its internal components such as programs, documents, and tools within the nodes. This diagram elucidates the connections and organization of elements within a system, resulting in the creation of a usable system. In the context of a component diagram, a component refers to a system part that can be modified and operates independently. It retains the secrecy of its internal operations and requires a specific method to execute a task, resembling a concealed box that functions only when operated correctly.

Notation for a Component Diagram includes:

- A component
- A node

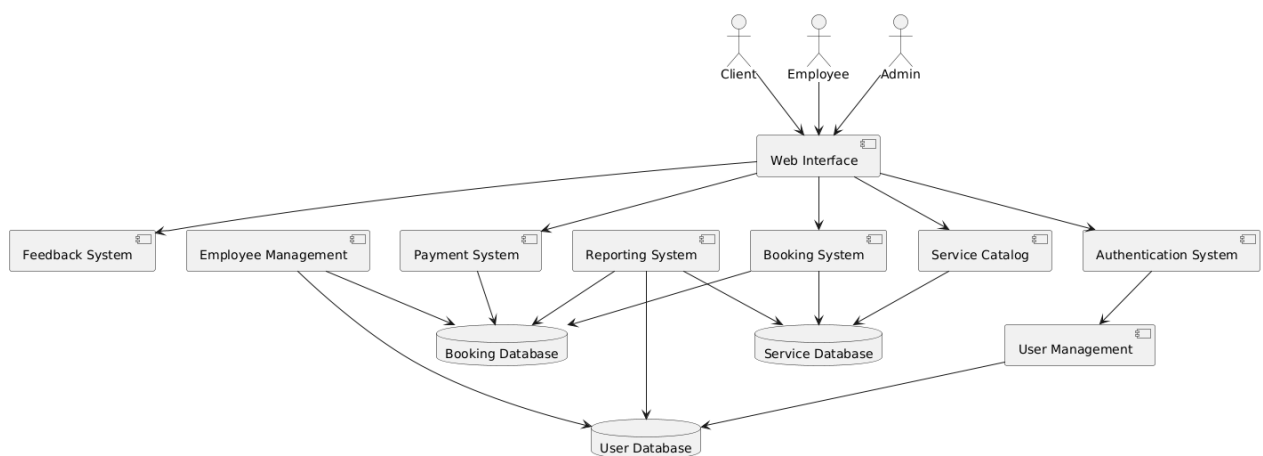


Fig: 4.2.7

4.2.8 Deployment Diagram

A deployment diagram provides a visual representation of how software is positioned on physical computers or servers. It depicts the static view of the system, emphasizing the arrangement of nodes and their connections. This type of diagram delves into the process of placing programs on computers, elucidating how software is constructed to align with the physical computer system.

Notations in a Deployment Diagram include:

- A component
- An artifact

- An interface
- A node

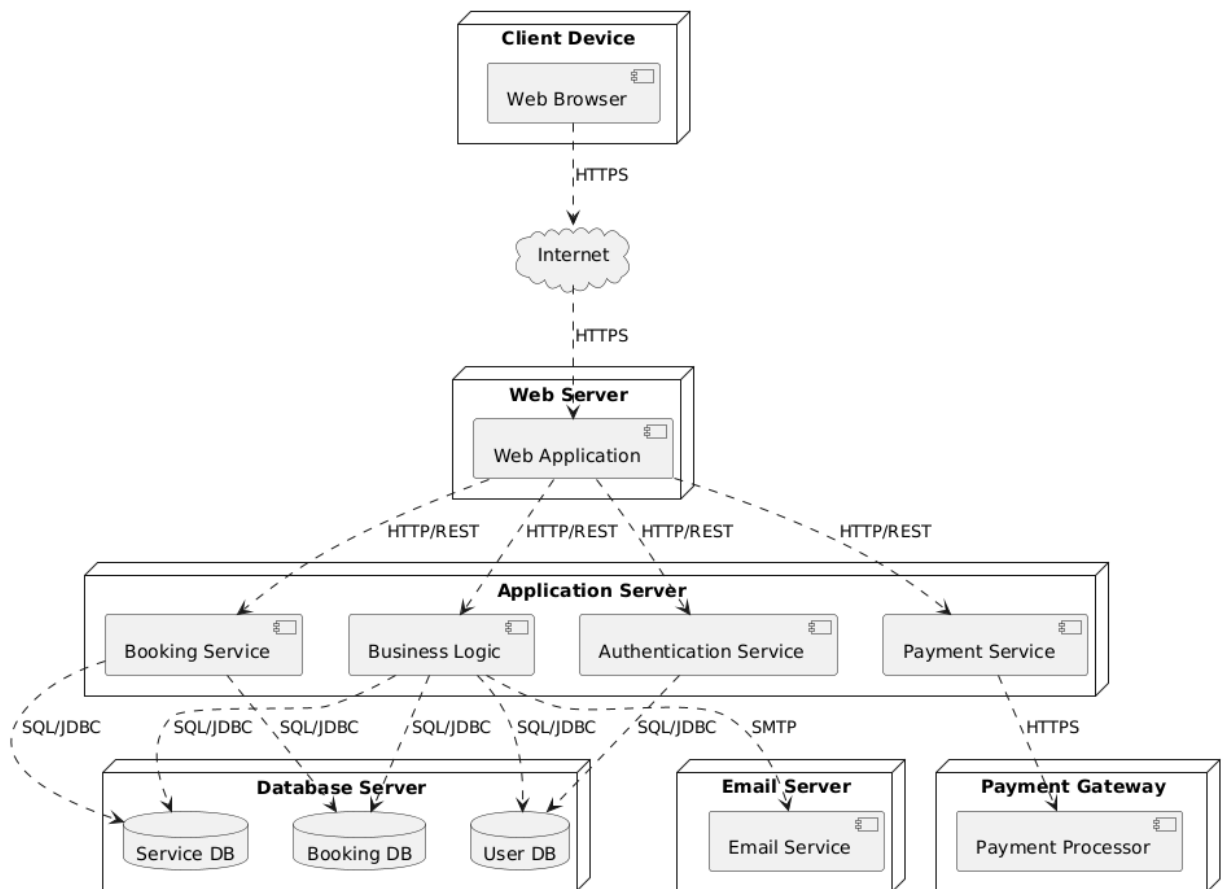


Fig: 4.2.8

4.2.9 Collaboration Diagram

A communication diagram, also known as a collaboration diagram, is a type of UML interaction diagram that shows how objects in a system interact with each other to accomplish a specific task or scenario. In this case, we're illustrating the booking process in the Online Hair Salon system.

Key elements of the communication diagram:

- Objects: Represented by boxes (Client, Web Interface, Authentication System, etc.)
- Links: Lines connecting objects, showing their associations
- Messages: Arrows on the links, showing the flow of communication
- Sequence numbers: Numbers preceding the messages, indicating the order of interactions

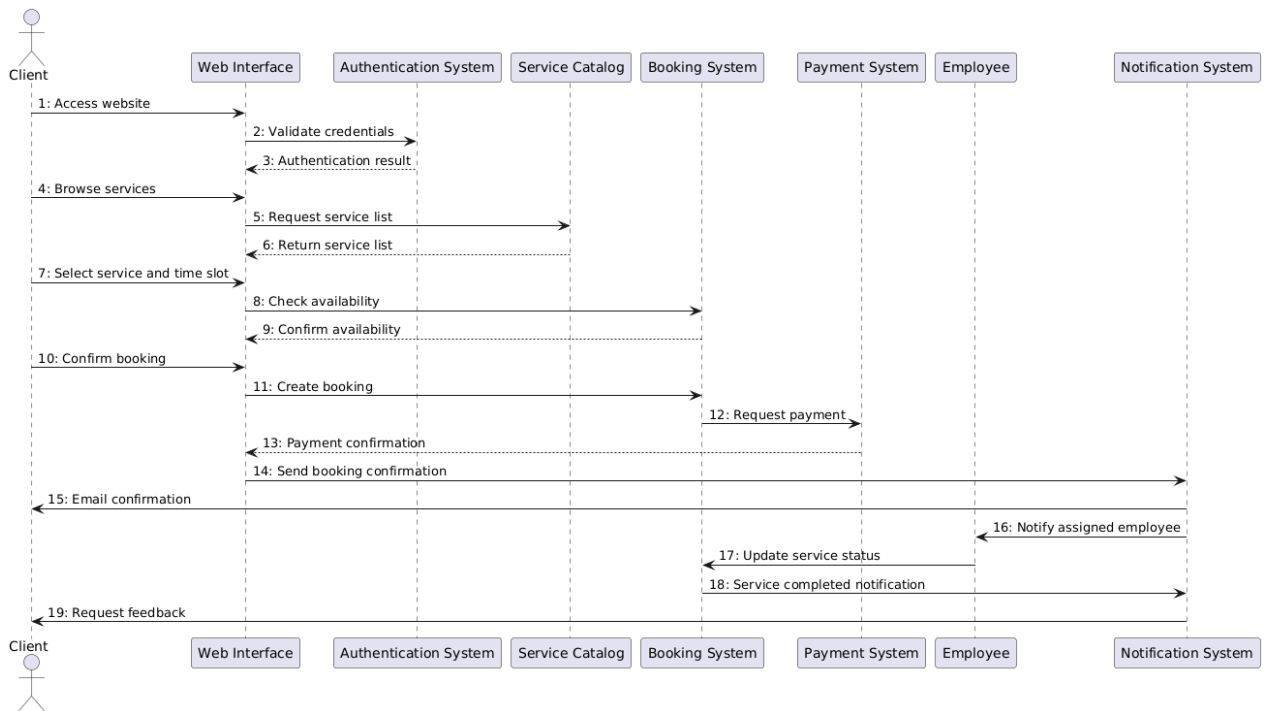


Fig: 4.2.9

4.3 USER INTERFACE DESIGN USING FIGMA

Login Page:

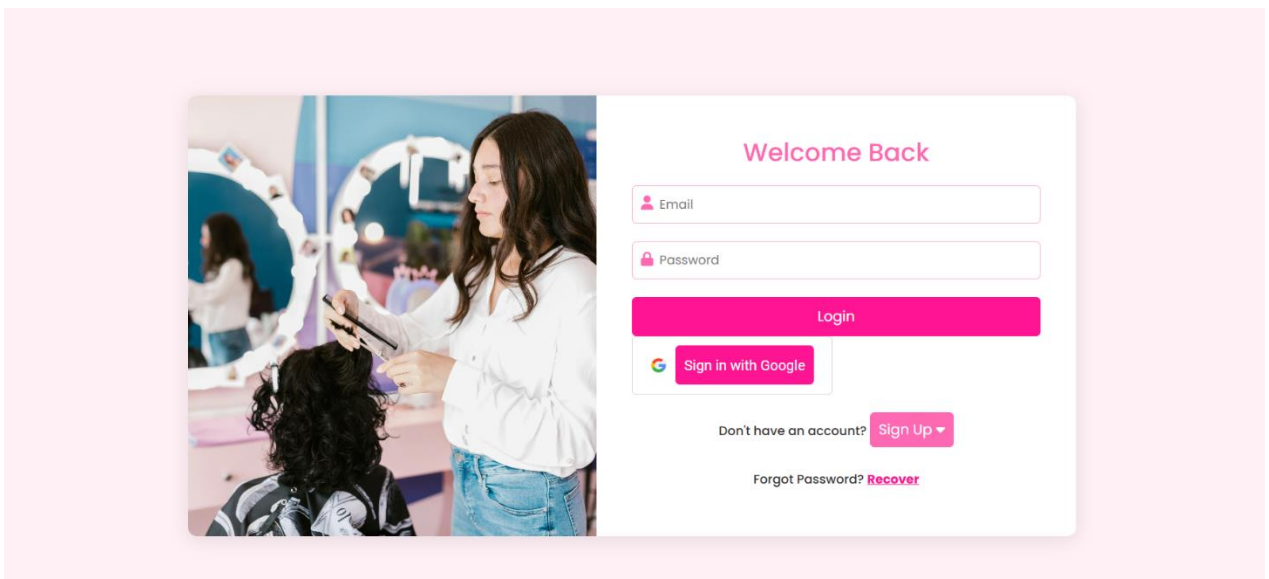
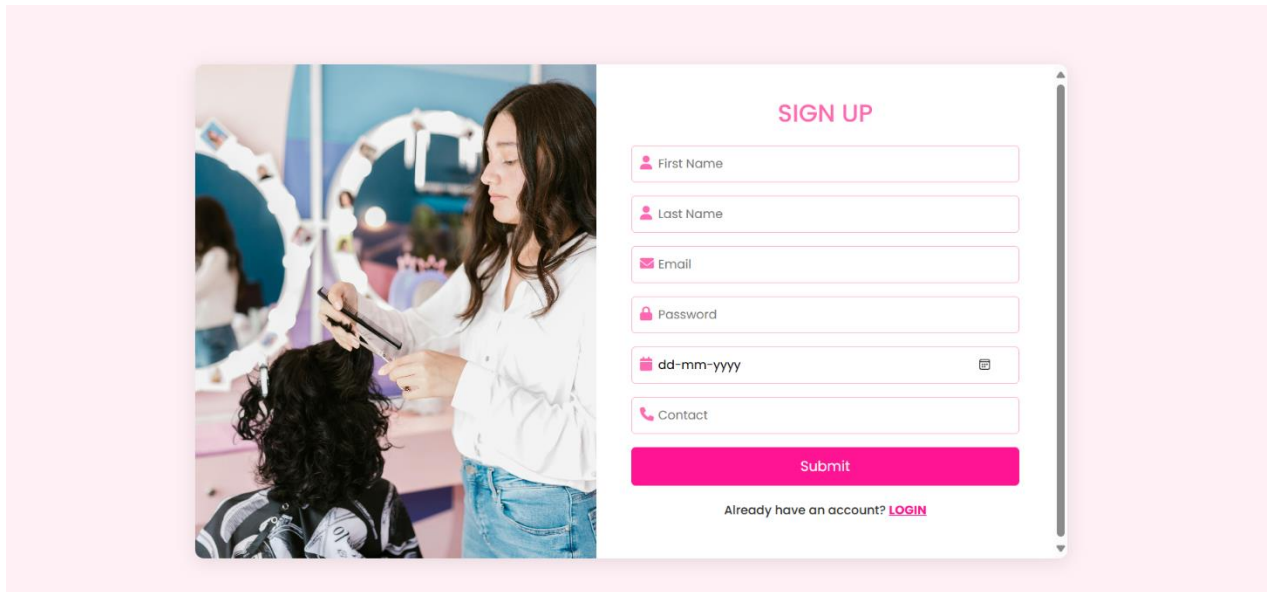


Fig: 4.3.1

Registration Page:

The registration page features a pink background. On the left is a vertical image of a woman in a white shirt styling a client's hair. On the right is a white 'SIGN UP' form with the following fields: First Name, Last Name, Email, Password, a date field (dd-mm-yyyy), and Contact. A pink 'Submit' button is at the bottom of the form, followed by the text 'Already have an account? [LOGIN](#)'.

Fig: 4.3.2

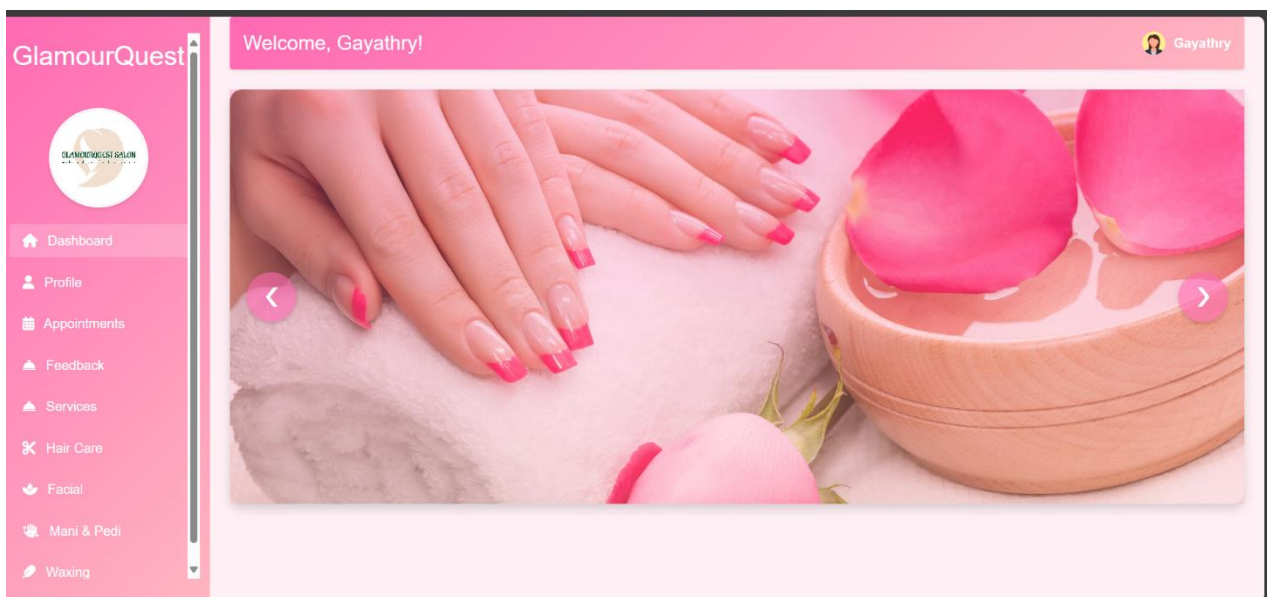

User Dashboard:

Fig: 4.3.3

Booking Page

GlamourQuest



[Dashboard](#)
[Profile](#)
[Appointments](#)
[Services](#)
[Hair Care](#)
[Facial](#)
[Mani & Pedi](#)
[Waxing](#)

Book Service

Book Service: Feather Cut

Rate: ₹1000.00

Booking date:

dd-mm-yyyy

Booking time:

08:00

Staff:

Preference


Additional notes:

Confirm Booking

Fig: 4.3.3

Feedback Page

GlamourQuest



[Dashboard](#)
[Profile](#)
[Appointments](#)
[Services](#)
[Hair Care](#)
[Facial](#)
[Mani & Pedi](#)
[Waxing](#)

Your Booking

Gayathry

Your Booking History

Service	Date	Time	Status	
Back Layer Hair Cut	Oct. 28, 2024	2 p.m.	Completed	View Feedback
V Shape Cut	Oct. 26, 2024	8 a.m.	Completed	Add Feedback

Fig: 4.3.4

4.4 DATABASE DESIGN

4.4.1 Relational Database Management System (RDBMS)

For the **Online Hair Salon** project, we can design the database structure using a Relational Database Management System (RDBMS) like MySQL, which offers robustness, scalability, and supports the relational nature of the system. Here's how the key tables, normalization principles, and constraints can be applied for the project:

Key Tables

1. User Table

- Stores general information about users, including clients and employees.
- **Columns:** user_id (Primary Key), first_name, last_name, email (unique), password, dob, contact, status, is_super, date_joined.

2. Client Table

- Contains additional details specific to clients.
- **Columns:** client_id (Primary Key, Foreign Key to User table), reset_token.

3. Employee Table

- Stores employee-specific information, including specialization and leave details.
- **Columns:** employee_id (Primary Key, Foreign Key to User table), approved, specializations (Many-to-Many relationship), qualification_certificate, is_on_leave, leave_start, leave_end.

4. ServiceCategory Table

- Manages the main categories of services.
- **Columns:** category_id (Primary Key), name.

5. ServiceSubcategory Table

- Manages subcategories for specific services under main categories.
- **Columns:** subcategory_id (Primary Key), category_id (Foreign Key to ServiceCategory), name.

6. Service Table

- Contains details of individual services offered in the salon.
- **Columns:** service_id (Primary Key), subcategory_id (Foreign Key to ServiceSubcategory), service_name, description, rate, image.

7. Booking Table

- Stores booking details created by clients.

- **Columns:** booking_id (Primary Key), client_id (Foreign Key to Client), service_id (Foreign Key to Service), booking_date, preferred_date, preferred_time, staff_id (Foreign Key to Employee), status, additional_notes.

8. Feedback Table

- Records client feedback on services.
- **Columns:** feedback_id (Primary Key), client_id (Foreign Key to Client), booking_id (Foreign Key to Booking), rating, comments, feedback_date.

9. Payment Table

Stores payment details associated with each booking.

Columns:

- payment_id (Primary Key): Unique identifier for each payment transaction.
- booking_id (Foreign Key to Booking): Links the payment to a specific booking.
- amount: The amount paid by the client.
- payment_date: Date and time the payment was made.
- payment_status: Status of the payment (e.g., Completed, Pending, Failed)

4.4.2 Normalization

To ensure data integrity and minimize redundancy, normalization up to the **Third Normal Form (3NF)** is applied to the database design:

1. First Normal Form (1NF):

- Ensure unique rows in each table, with each column containing atomic values.
- Example: The contact field in the User table stores a single contact number per record.

2. Second Normal Form (2NF):

- All non-key attributes are fully dependent on the primary key.
- Example: In the Booking table, service_id, preferred_date, and other attributes are directly related to a specific booking_id.

3. Third Normal Form (3NF):

- Remove transitive dependencies so that non-key attributes depend solely on the primary key.
- Example: In the Feedback table, feedback details relate directly to feedback_id and indirectly to booking_id, ensuring each feedback is uniquely tied to a booking.

4.4.3 Sanitization

For the Online Hair Salon project, sanitization is essential to safeguard the system against attacks like SQL injection and Cross-Site Scripting (XSS), ensuring data integrity and security. Here's how sanitization can be applied in the project:

Sanitization Techniques

1. Input Validation:

- Validate inputs to confirm they match the expected data format and type.
- Example: For fields like email in the User table, ensure that it matches a standard email format; similarly, restrict contact numbers to a specific numeric pattern.

2. Parameterized Queries:

- Use Django's ORM, which inherently protects against SQL injection by handling input as data instead of executable code.
- For complex raw SQL queries, use parameterized queries to ensure safe handling of user input.

3. Escaping Special Characters:

- Escape special characters in inputs, such as single quotes and backslashes, to prevent them from affecting SQL commands.
- Django's ORM and built-in form validation provide this protection, but any custom SQL should handle escaping appropriately.

4. Cross-Site Scripting (XSS) Protection:

- Use Django's built-in HTML sanitization features, such as `escape()` and `safe()` functions, to control how user-provided data is rendered on web pages.
- Use Django's template auto-escaping for outputs to prevent XSS attacks.

4.4.4 Indexing

In the GLAMOURQUEST Hair Salon project, indexing plays a crucial role in enhancing database performance by speeding up data retrieval for frequently accessed records.

Types of Indexes Used:

1. Primary Index:

- Automatically created on primary keys to facilitate fast lookups. For instance, the id field in tables like Client, Employee, and Service serves as the primary index, allowing efficient data retrieval by primary keys.

2. Foreign Key Index:

- Created on foreign key fields to optimize join operations and related record lookups. For example:
 - The user_id field in the Booking table, which references the Client or Employee table.
 - The subcategory_id field in the Service table, allowing quick access to services within specific subcategories.

3. Unique Index:

- Ensures unique values for specific fields and optimizes retrieval of unique records. For example:
 - The email field in the User model to prevent duplicate registrations and allow fast lookups by email.

4. Composite Index:

- Useful for optimizing searches on multiple columns. For example:
 - A composite index on date and staff in the Booking table to expedite searches for bookings on specific dates by specific staff members.

5. Full-Text Index:

- Improves search performance on text fields with large amounts of data. For instance:

- A full-text index on feedback comments in a Feedback table to enable faster text-based searches for specific phrases or keywords in user feedback.

By implementing these indexing strategies, the GLAMOURQUEST Hair Salon project can ensure optimal database performance, particularly for queries involving multiple tables or large datasets.

4.5 TABLE DESIGN

4.5.1 Tbl_Client

Primary key: **client_id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Client_id	AutoField	Primary Key	Unique Id
2	first_name	CharField (50)	Not Null	User's first name
3	last_name	CharField (50)	Not Null	User's last name
4	Email	EmailField	Unique	User's email
5	password	CharField (50)	Not Null	Hashed user password
6	Dob	DateField	Not Null	User's date of birth
7	Contact	CharField (15)	Not Null	User's contact number
8	Status	BooleanField	Default=True	User's active status

4.5.2 Tbl_Employee

Primary key: **employee_id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	employee_id	AutoField	Primary Key	Unique Id
2	first_name	CharField (50)	Not Null	User's first name
3	last_name	CharField (50)	Not Null	User's last name
4	Email	EmailField	Unique	User's email
5	Password	CharField (50)	Not Null	Hashed user password
6	Dob	DateField	Not Null	User's date of birth
7	Contact	CharField (15)	Not Null	User's contact number
8	Status	BooleanField	Default=True	User's active status
9	Approved	BooleanField	Default=False	Approval status for employee
10	qualification_certificate	FileField	Not Null	Certificate file for qualifications

4.5.3 Tbl_Specialization

Primary key: **specialization_id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	specialization_id	AutoField	Primary Key	Unique Id
2	specialization_name	CharField (50)	Unique	Specialization_name
3	Description	CharField (50)	Not Null	Description

4.5.4 Tbl_ServiceCategory

Primary key: **category_id**

Foreign key: **specialization_id**(References Table: **Tbl_Specialization**)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	category_id	AutoField	Primary Key	Unique Id
2	category_name	CharField (50)	Not Null	Name of tservice category
3	specialization_id	Int(10)	ForeignKey	Specialization associated

4.5.5 Tbl_ServiceSubcategory

Primary key: **subcategory_id**

Foreign key: **category_id**(References Table: **Tbl_ServiceCategory**)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	subcategory_id	AutoField	Primary Key	Unique Id
2	subcategory_name	CharField (50)	Not Null	Name of service category
3	Category_id	Int(10)	ForeignKey	Service category
4	image	ImageField	Not Null	Image of subcategory
5	description	TextField	Not Null	Description of subcategory

4.5.6 Tbl_Service

Primary key: **service_id**

Foreign key: **subcategory_id**(References Table: Tbl_ServiceSubcategory)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	service_id	AutoField	Primary Key	Unique Id
2	service_name	CharField (50)	Not Null	Name of the service
3	description	TextField	Not Null	Service description
4	rate	DecimalField	Not Null	Service rate in INR
5	image	ImageField	Not Null	Image of service
6	subcategory_id	Int(10)	ForeignKey	Subcategory of service

4.5.7 Tbl_Booking

Primary key: **booking_id**

Foreign key: **client_id**(References Table: Tbl_Client),

Service_id(References Table: Tbl_Service),

employee_id(References Table: Tbl_Employee),

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	booking_id	AutoField	Primary Key	Unique Id
2	booking_date	DateField	Not Null	Date of the booking
3	booking_time	TimeField	Not Null	Time of the booking
4	additional notes	TextField	Nullable	Additional nodes
5	status	CharField (20)	Not Null	Booking status
6	client_id	Int(10)	Foreign Key	Client booked the service
7	service_id	Int(10)	Foreign key	Service booked
8	Employee_id	Int(10)	Foreign Key	Assigned staff member

4.5.8 Tbl_Feedback

Primary key: **feedback_id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	feedback_id	AutoField	Primary Key	Unique Id
2	booking_id	Int(10)	Not Null	Booking linked with feedback
3	Rating	Int(10)	Not Null	Rating provided for the service
4	comment	TextField	Not Null	Comment/feedback on the service
5	created_at	DateTimeField	Auto added	Date and time of feedback

4.5.9 Tbl_Payment

Primary key: **payment_id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Id	AutoField	Primary Key	Unique Id
2	booking_id	Int(10)	Foreign key	Associated booking ID
3	client_id	Int(10)	Foreign key	Client involved in the payment
4	employee_id	Int(10)	Foreign key	Assigned employee for the service
5	service_id	Int(10)	Foreign key	Service provided in the booking
6	amount	DecimalField	Not Null	Amount paid in INR
7	payment_id	Int(10)	Foreign Key	Payment ID from Razorpay
8	order_id	CharField (100)	Not Null	Order ID from Razorpay
9	status	CharField (100)	Not Null	Status of the payment
10	created_at	DateTimeField	Auto added	Date and time of payment creation

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is crucial for verifying a program's functionality, ensuring it meets defined requirements, and upholding quality standards. Validation within testing confirms that the software performs as specified and satisfies user needs. Software testing often involves multiple techniques, such as code inspection and walkthroughs, to evaluate the program's quality and detect flaws effectively.

Key Principles and Objectives of Software Testing:

1. **Error Detection:** Testing is fundamentally aimed at identifying errors by executing the program in various scenarios.
2. **Effective Test Cases:** A good test case is designed to reveal previously unknown errors, increasing confidence in software reliability.
3. **Success through Failure:** Tests are successful if they uncover errors, as it indicates areas needing improvement and ensures correctness.

Three Aspects of Software Evaluation:

1. **Correctness Assessment:** Verifies that the program performs the intended functions accurately.
2. **Implementation Efficiency:** Assesses the program's resource usage, ensuring it performs optimally within set constraints.
3. **Computational Complexity:** Examines the efficiency of algorithms in terms of time and space, identifying potential areas for optimization.

5.2 TEST PLAN

The test plan outlines the approach, scope, resources, and schedule for testing activities. It includes identifying the testing methods, tools, and environment to ensure that each module works independently and in conjunction with other components. This project follows a structured test plan, beginning with Unit Testing and progressing to Integration, Validation, Output, and Automation Testing.

5.2.1 Unit Testing

Unit testing checks the smallest part of a software design - the software component or module. Testing important control paths within a module using the design guide to find errors. This means how difficult the tests are for each small part of a program and what parts of the program haven't been tested yet. Unit testing is a type of testing that looks at how the code works inside and can be done at the same time for different parts of the program.

Before starting any other test, we need to check if the data flows correctly between different parts of the computer program. If the information doesn't move in and out correctly, all other checks are pointless. When designing something, it's important to think about what could go wrong and make a plan for how to deal with those problems. This can mean redirecting the process or stopping it completely.

The Sell-Soft System was tested by looking at each part by itself and trying different tests on it. Some mistakes in the design of the modules were discovered and then fixed. After writing the instructions for different parts, each part is checked and tried out separately. We got rid of extra code and made sure everything works the way it should.

5.2.2 Integration Testing

Integration testing is a critical process in software development that involves constructing a program while simultaneously identifying errors in the interaction between different program components. The primary objective is to utilize tested individual parts and assemble them into a program according to the initial plan. This comprehensive testing approach assesses the entire program to ensure its proper and correct functionality. As issues are identified and resolved during integration testing, it's not uncommon for new problems to surface, leading to an ongoing cycle of testing and refinement. After each individual component of the system is thoroughly examined, these components are integrated to ensure they function harmoniously. Additionally, efforts are made to standardize all programs to ensure uniformity rather than having disparate versions.

5.2.3 Validation Testing or System Testing

The final phase of testing involves a comprehensive examination of the entire system to ensure the correct interaction of various components, including different types of instructions and building blocks. This testing approach is referred to as Black Box testing or System testing.

Black Box testing is a method employed to determine if the software functions as intended. It assists software engineers in identifying all program issues by employing diverse input types. Black Box testing encompasses the assessment of errors in functions, interfaces, data access, performance, as well as initialization and termination processes. It is a vital technique to verify that the software meets its intended requirements and performs its functions correctly.

5.2.4 Output Testing or User Acceptance Testing

System testing is conducted to assess user satisfaction and alignment with the company's requirements. During the development or update of a computer program, it's essential to maintain a connection with the end-users. This connection is established through the following elements:

- Input Screen Designs.
- Output Screen Designs.

To perform system testing, various types of data are utilized. The preparation of test data plays a crucial role in this phase. Once the test data is gathered, it is used to evaluate the system under investigation. When issues are identified during this testing, they are addressed by following established procedures. Records of these corrections are maintained for future reference and improvement. This process ensures that the system functions effectively and meets the needs of both users and the organization

5.2.5 Automation Testing

Automated testing is a method employed to verify that software functions correctly and complies with established standards before it is put into official use. This type of testing relies on written instructions that are executed by testing tools. Specifically, UI automation testing involves the use of specialized tools to automate the testing process. Instead of relying on manual interactions where individuals click through the application to ensure its proper functioning, scripts are created to automate these tests for various scenarios. Automating testing is particularly valuable when it is necessary to conduct the same test across multiple computers simultaneously, streamlining the testing process and ensuring consistency

5.2.6 Selenium Testing

Selenium is a valuable and free tool designed for automating website testing. It plays a crucial role for web developers as it simplifies the testing process. Selenium automation testing refers to the practice of using Selenium for this purpose. Selenium isn't just a single tool; it's a collection of tools, each serving distinct functions in the realm of automation testing. Manual testing is a necessary aspect of application development, but it can be monotonous and repetitive. To alleviate these challenges, Jason Huggins, an employee at ThoughtWorks, devised a method for automating testing procedures, replacing manual tasks. He initially created a tool named the JavaScriptTestRunner to facilitate automated website testing, and in 2004, it was rebranded as Selenium.

Example:**Test Case 1 – User Login****Code**

```
package Definition;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class step_definition {

    WebDriver driver=null;

    @Given("browser is open")

    public void browser_is_open() {

        System.out.println("Inside step-Browser is open");

        System.setProperty("webdriver.gecko.marionette","C:\\Users\\LENOVO\\eclipse-
workspace\\987\\src\\test\\resources\\driversgeckodriver.exe");

        driver=new FirefoxDriver();

        driver.manage().window().maximize();

    }

    @And("user is on login page")

    public void user_is_on_login_page() throws Exception {

        driver.navigate().to("http://127.0.0.1:8000/login/");

        Thread.sleep(2000);

    }

    @When("user enters username and password")

    public void user_enters_username_and_password() throws Throwable{

        driver.findElement(By.id("email")).sendKeys("gayathrignair6@gmail.com");
```

```

        driver.findElement(By.id("password")).sendKeys("Gayathry@123");
    }
    @Then("user clicks on login")

    public void user_clicks_on_login() {
        driver.findElement(By.id("login")).click();

    }

}

```

Eg.Screenshot

```

Scenario: Check login is successful with valid credentials # src/test/resources/Features/Login.feature:3
Inside step-Browser is open
    Given browser is open # Definition.step_definition.browser_is_open()
    And user is on login page # Definition.step_definition.user_is_on_login_page()
    When user enters username and password # Definition.step_definition.user_enters_username_and_password()
    Then user clicks on login # Definition.step_definition.user_clicks_on_login()

1 Scenarios (1 passed)
4 Steps (4 passed)
0m25.535s

```

Eg.Test Report

Test Case 1					
Project Name: GLAMOURQUEST					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Gayathry G Nair		
Test Priority(Low/Medium/High):Medium			Test Designed Date: 30-10-2024		
Module Name: Login Module			Test Executed By : Mrs. Lisha Varghese		
Test Title : User_Login			Test Execution Date: 30-10-2024		
Description: User has a valid email/password					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to the login page	URL: http://127.0.0.1:8000/login/	Login page should display	Login page is displayed	Pass
2	Enter valid email in email field	Email: gayathrignair6@gmail.com	User should be able to login	User logged in	Pass
3	Enter valid password in password field	Password: Gayathry@123			

4	Click the login button	N/A			
Post-Condition: The user is authenticated and redirected to their account dashboard.					

Test Case 2: Employee Accept / Decline Appointment Code

```
package Definition;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class serviceaccept{

    WebDriver driver=null;

    @Given("browser is opened")
    public void browser_is_open() {

        System.out.println("Inside step-Browser is open");

        System.setProperty("webdriver.gecko.marionette","C:\\Users\\LENOVO\\eclipse-
workspace\\987\\src\\test\\resources\\driversgeckodriver.exe");

        driver=new FirefoxDriver();

        driver.manage().window().maximize();

    }

    @And("user is on login page")
    public void user_is_on_login_page() throws Exception {

        driver.navigate().to("http://127.0.0.1:8000/login/");

        Thread.sleep(2000);

    }

}
```

```

        @When("user enters username and passwordsss")

        public void user_enters_username_and_password() throws Throwable{

driver.findElement(By.id("email")).sendKeys("fathimaps2025@mca.ajce.in");

        driver.findElement(By.id("password")).sendKeys("Fathima@123");

        }
        @Then("user clicks on loginsss")

        public void user_clicks_on_login() {
            driver.findElement(By.id("login")).click();

        }

        @And("user clicks on view appointments")

        public void user_clicks_on_view_appointments() {
            driver.findElement(By.id("view-appointments")).click();

        }

        @Then("user clicks on accept")

        public void user_clicks_on_accept() {
            driver.findElement(By.id("confirm")).click();

        }

    }
}

```

Screenshot

Scenario: Check login is successful with valid credentials # src/test/resources/Features/serviceaccept.feature:3
 Inside step-Browser is open
 Given browser is opened # Definition.serviceaccept.browser_is_open()
 And user is on login page # Definition.serviceaccept.user_is_on_login_page()
 When user enters username and password # Definition.serviceaccept.user_enters_username_and_password()
 Then user clicks on login # Definition.serviceaccept.user_clicks_on_login()
 And user clicks on view appointments # Definition.serviceaccept.user_clicks_on_view_appointments()
 Then user clicks on accept # Definition.serviceaccept.user_clicks_on_accept()

 1 Scenarios (1 passed)
 6 Steps (6 passed)
 0m8.805s

Test report

Test Case 2	
Project Name: GLAMOURQUEST	
Login Test Case	
Test Case ID: Test_2	Test Designed By: Gayathry G Nair
Test Priority(Low/Medium/High):Medium	Test Designed Date: 30/10/2024
Module Name: Appointment Management	Test Executed By : Mrs. Lisha Varghese

Test Title : Employee Login			Test Execution Date: 30/10/2024		
Description: Verifies that an authenticated user can view and accept service appointments.					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to the login page	URL: http://127.0.0.1:8000/login/	Login page should display	Login page is displayed	Pass
2	Enter valid email in email field	Email: fathimaps2025@mca.ajce.in	User should be able to login	User logged in	Pass
3	Enter valid password in password field	Password: Fathima@123			
4	Click the login button	N/A			
5	Click on view appointments	N/A	User should see a list of appointments	Appointment list displayed	Pass
6	Click on accept for an appointment	N/A	Appointment should be marked as accepted	Appointment accepted	Pass
Post-Condition: The appointment status is updated in the system, and the action is logged in the database.					

Test Case 3 – Admin Approving Employee

Code

```
package Definition;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
```

```
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class approve{

    WebDriver driver=null;

    @Given("browser is opens")

    public void browser_is_open() {

        System.out.println("Inside step-Browser is open");

        System.setProperty("webdriver.gecko.marionette","C:\\Users\\LENOVO\\eclipse-
workspace\\987\\src\\test\\resources\\driversgeckodriver.exe");

        driver=new FirefoxDriver();

        driver.manage().window().maximize();

    }

    @And("user is on login pages")

    public void user_is_on_login_page() throws Exception {

        driver.navigate().to("http://127.0.0.1:8000/login/");

        Thread.sleep(2000);

    }

    @When("user enters username and passwords")

    public void user_enters_username_and_password() throws Throwable{

        driver.findElement(By.id("email")).sendKeys("admin@gmail.com");

        driver.findElement(By.id("password")).sendKeys("admin123");

    }

    @Then("user clicks on logins")

    public void user_clicks_on_login() {

        driver.findElement(By.id("login")).click();

    }

    @Then("user clicks on manage employee")

    public void user_clicks_on_manage_employee() {

        driver.findElement(By.id("manage_employee")).click();

    }

    @And("user clicks on approve")

    public void user_clicks_on_approve() {

        driver.findElement(By.id("approval_button")).click();

    }

}
```

```

    }
}

```

Screenshot

```

Scenario: Check login is successful with valid credentials # src/test/resources/Features/approve.feature:3
Inside step-Browser is open
  Given browser is opens                                     # Definition.approve.browser_is_open()
  And user is on login pages                               # Definition.approve.user_is_on_login_page()
  When user enters username and passwords                  # Definition.approve.user_enters_username_and_password()
  Then user clicks on logins                               # Definition.approve.user_clicks_on_login()
  Then user clicks on manage employee                      # Definition.approve.user_clicks_on_manage_employee()
  And user clicks on approve                              # Definition.approve.user_clicks_on_approve()

1 Scenarios (1 passed)
6 Steps (6 passed)
0m8.382s

```

Test Report

Test Case 3					
Project Name: GLAMOURQUEST					
Login Test Case					
Test Case ID: Test_3			Test Designed By: Gayathry G Nair		
Test Priority(Low/Medium/High):High			Test Designed Date: 30/10/2024		
Module Name: Employee Management			Test Executed By : Mrs. Lisha Varghese		
Test Title : Approving an Employee			Test Execution Date: 30/10/2024		
Description: Verifies that an admin can log in and approve employees through the Employee Management module.					
Pre-Condition : Admin user has valid login credentials, and an unapproved employee exists in the system.					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to the login page	URL: http://127.0.0.1:8000/login/	Login page should display	Login page is displayed	Pass
2	Enter valid email in email field	Email: admin@gmail.com	Admin should be able to login	Admin logged in	Pass
3	Enter valid password in password field	Password: admin123			

4	Click the login button	N/A			
5	Navigate to "Manage Employee" section	N/A	Employee management page should display	Employee management page displayed	Pass
6	Click the approve button for an employee	N/A	Employee should be marked as approved	Employee approved	Pass
Post-Condition: The employee's status is updated in the system, indicating approval, and the action is logged in the database.					

Example:**Test Case 4 – Admin Add Service****Code**

```

package Definition;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

import org.openqa.selenium.support.ui.Select;

public class manageservice{

    WebDriver driver=null;

    @Given("browser is opened")

    public void browser_is_open() {

        System.out.println("Inside step-Browser is open");

        System.setProperty("webdriver.gecko.marionette","C:\\Users\\LENOVO\\eclipse-workspace\\987\\src\\test\\resources\\driversgeckodriver.exe");

        driver=new FirefoxDriver();

```

```

        driver.manage().window().maximize();
    }

    @And("user is on login page")
    public void user_is_on_login_page() throws Exception {
        driver.navigate().to("http://127.0.0.1:8000/login/");
        Thread.sleep(2000);
    }

    @When("user enters username and password")
    public void user_enters_username_and_password() throws Throwable{
        driver.findElement(By.id("email")).sendKeys("admin@gmail.com");
        driver.findElement(By.id("password")).sendKeys("admin123");
    }

    @Then("user clicks on login")
    public void user_clicks_on_login() {
        driver.findElement(By.id("login")).click();
    }

    @And("user clicks on manage services")
    public void user_clicks_on_manage_service() {
        driver.findElement(By.id("manage_service")).click();
    }

    @When("user enters service details")
    public void user_enters_service_details() throws Throwable{
        Select categoryDropdown = new
Select(driver.findElement(By.id("category"))));
        categoryDropdown.selectByVisibleText("Hair Care");

        Select subcategoryDropdown = new
Select(driver.findElement(By.id("subcategory"))));
        subcategoryDropdown.selectByVisibleText("Hair Cut");
        driver.findElement(By.id("service_name")).sendKeys("Bob Cutting");
        driver.findElement(By.id("description")).sendKeys("Good looking
");
        driver.findElement(By.id("rate")).sendKeys("800");
        driver.findElement(By.id("image")).sendKeys("E:\\hair-
coloring.jpg");
    }

    @And("user clicks on add services")
    public void user_clicks_on_add_services() {
        // Find the button element
        WebElement addButton = driver.findElement(By.id("add_service"));
    }

```

```

        // Scroll the button into view using JavaScript
        JavascriptExecutor js = (JavascriptExecutor) driver;
        js.executeScript("arguments[0].scrollIntoView(true);", addButton);

        // Add a small wait to ensure the button is clickable
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Click the button
        addButton.click();
    }
}

```

Screenshot

```

Scenario: Check login is successful with valid credentials # src/test/resources/Features/manageservice.feature:3
Inside step-Browser is open
  Given browser is opened # Definition.manageservice.browser_is_open()
  And user is on login page # Definition.manageservice.user_is_on_login_page()
  When user enters username and password # Definition.manageservice.user_enters_username_and_password()
  Then user clicks on login # Definition.manageservice.user_clicks_on_login()
  And user clicks on manage services # Definition.manageservice.user_clicks_on_manage_service()
  When user enters service details # Definition.manageservice.user_enters_service_details()
  And user clicks on add services # Definition.manageservice.user_clicks_on_add_services()

1 Scenarios (1 passed)
7 Steps (7 passed)
0m8.482s

```

Test Report

Test Case 4					
Project Name: GLAMOURQUEST					
Login Test Case					
Test Case ID: Test_4			Test Designed By: Gayathry G Nair		
Test Priority(Low/Medium/High):High			Test Designed Date: 30/10/2024		
Module Name: Service Management			Test Executed By : Mrs. Lisha Varghese		
Test Title : Admin Add Service			Test Execution Date: 30/10/2024		
Description: This test case verifies that an admin user can successfully log in and add a new service to the system by entering required service details in the "Manage Services" section.					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)

1	Navigate to the login page	URL: http://127.0.0.1:8000/login/	Login page should display	Login page is displayed	Pass
2	Enter valid email in email field	Email: admin@gmail.com	Admin should be able to login	Admin logged in	Pass
3	Enter valid password in password field	Password: admin123			
4	Click the login button	N/A			
5	Navigate to "Manage Services"	N/A	Manage services page should display	Manage services page displayed	Pass
6	Select service category	Category: Hair Care	Category is selected successfully	Category selected	Pass
7	Select service subcategory	Subcategory: Hair Cut	Subcategory is selected successfully	Subcategory selected	Pass
8	Enter service name	Service Name: Bob Cutting	Service name is entered successfully	Service name entered	Pass
9	Enter service description	Description: Good looking	Description is entered successfully	Description entered	Pass
10	Enter service rate	Rate: 800	Rate is entered successfully	Rate entered	Pass
11	Upload service image	Image path: E:\hair-coloring.jpg	Image is uploaded successfully	Image uploaded	Pass
12	Click "Add Service" button	N/A	Service should be added successfully	Service added	Pass
Post-Condition: The new service is added to the system and is available in the list of services. The action is logged in the database.					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

This section describes the comprehensive process of bringing the Online Hair Salon Platform from initial development to a fully operational state. The implementation involves configuring the application, integrating essential software and database components, and thoroughly testing to deliver a stable, user-friendly experience for salon clients, staff, and administrators.

6.2 IMPLEMENTATION PROCEDURES

The implementation procedures for the Online Hair Salon Platform outline the key steps from initial development through deployment. These include setting up both development and production environments, configuring databases, deploying the web application, and integrating advanced functionalities for user engagement. The primary procedures also involve implementing machine learning for predictive analytics, establishing secure user authentication, and enabling communication channels for notifications and live support.

Environment Setup:

- Configure the Django framework with MySQL databases in the production environment to support backend functionality and data management.
- Install necessary libraries and dependencies for frontend styling, backend processing, and specialized features like virtual try-ons and payment gateways.

Data Integration:

- Load initial data for service providers, including detailed service types and categories, as well as test data to support predictive analytics, enabling personalized recommendations for users.

Testing and Debugging:

- Conduct thorough unit and integration testing to ensure that each module—such as booking, payments, and virtual try-ons—operates seamlessly.
- Identify and address any issues, optimizing database queries and routing algorithms to ensure fast response times and smooth navigation.

Deployment:

- Deploy the application on a secure, scalable server or cloud platform configured to handle data security, manage high user loads, and scale as needed.
- Set up monitoring tools to track system health, monitor user activity, and support maintenance, ensuring consistent and reliable performance.

6.2.1 User Training

User training is a critical component in ensuring that users can effectively navigate and utilize the functionalities of the Online Hair Salon Platform. This training aims to empower users to request services, manage appointments, track their bookings, and update their personal profiles with confidence. The training program includes the following elements:

- **Guided Tours:** An introductory guide will be provided to users upon their first login, offering a walkthrough of the main features of the platform. This interactive tour will highlight key functionalities, such as booking services, viewing appointment details, and accessing payment options.
- **User Manual:** A comprehensive user manual will be made available, detailing step-by-step instructions on how to utilize the platform effectively. This document will cover various aspects, including how to register, log in, book services, and provide feedback, ensuring users have all the necessary information at their fingertips.
- **Support Access:** Users will have access to a dedicated support section within the application, which will include help articles, frequently asked questions (FAQs), and troubleshooting tips. This resource will enable users to find answers to common issues and enhance their overall experience with the platform.

6.2.2 Training on the Application Software

Once users have developed their fundamental computer skills, the next step involves training them on the Online Hair Salon application software. This training aims to provide a comprehensive understanding of the new system, covering essential areas such as navigating the interface, accessing help resources, managing errors, and resolving issues effectively. The goal is to equip users with the knowledge and skills needed to utilize the platform efficiently. Recognizing that different user groups have varied needs, the training is tailored for specific roles within the organization, ensuring that each group receives relevant information. The training focuses on the following key user categories:

- **Admin Training:** This training module is designed for administrators who oversee the platform's operations. It will cover topics such as managing service requests, accessing and interpreting analytics dashboards, overseeing communication channels, and ensuring smooth operations within the system. Administrators will gain the necessary skills to effectively manage the platform and address any issues that may arise.

- **Service Provider Training:** This module is tailored for service providers who will be responsible for managing their availability, updating their profiles, and tracking their assignments within the platform. Training will include best practices for managing bookings, responding to customer inquiries, and maintaining up-to-date service information. Service providers will be equipped with the tools they need to deliver exceptional service to clients.

6.2.3 Hosting

Hosting refers to the process of storing and serving website files on a remote server, which can be accessed by visitors over the internet. When you create a website, you need to have it hosted on a server so that it can be available for others to view. There are various types of hosting options available such as shared hosting, dedicated hosting, VPS hosting, cloud hosting, etc. The choice of hosting depends on the size of the website, its traffic volume, and the level of control and flexibility required by

Render

Render is a cloud hosting service that provides a streamlined platform for deploying web applications, APIs, static sites, and databases. It simplifies the process of hosting applications by automating tasks such as server setup, scaling, and load balancing, which allows developers to focus on building their applications rather than managing infrastructure. Render supports various languages and frameworks, including Django, and offers seamless integration with Git for continuous deployment

Procedure for hosting a website on render:

Step 1: Login to your render account

Step 2: Create a new project

Step 3: Create a requirements.txt file which will include all your dependencies to be installed.

Step 4: Upload the content which is to be hosted into GitHub

Step 5: Change the setting of your project to include the host also and then deploy.

Hosted Website: Render

Hosted Link: <https://online-hair-salon.onrender.com>

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The GLAMOURQUEST Hair Salon Platform has been designed as a comprehensive digital solution for modernizing and enhancing the salon experience. Through a combination of user-friendly features, robust service management, and seamless appointment booking, the platform caters to the needs of both clients and salon staff. Clients can easily explore available services, book appointments, and receive personalized recommendations, while administrators and service providers benefit from efficient tools for managing appointments, tracking client preferences, and updating service offerings.

The platform also integrates machine learning models for predictive analytics, providing valuable insights into client behavior and optimizing resource allocation. Additionally, secure user authentication and real-time communication channels contribute to a safe and interactive user experience, promoting trust and engagement. The implementation of payment processing and feedback systems further enhances convenience for clients, allowing them to seamlessly complete transactions and share their experiences.

In conclusion, the Online Hair Salon Platform serves as a powerful tool that not only improves operational efficiency but also elevates customer satisfaction. By streamlining salon management and enhancing the client journey, the platform paves the way for a more connected, personalized, and user-centric salon experience. Through continuous updates and feedback-driven improvements, this digital transformation initiative promises to keep up with evolving client expectations and industry trends.

7.2 FUTURE SCOPE

The GLAMOURQUEST Hair Salon Platform presents numerous opportunities for future growth, with various enhancements that can significantly improve user experience and operational efficiency. A key area for expansion is incorporating advanced AI models to offer more personalized recommendations based on client preferences, booking history, and emerging trends. This could enable more tailored suggestions for styles and products, fostering deeper client engagement. Additionally, the development of a mobile application for both iOS and Android could provide clients with greater accessibility, enabling on-the-go booking, real-time notifications, and easy access to profiles and feedback.

Building on the platform's virtual try-on feature, integrating Augmented Reality (AR) would allow clients to visualize hairstyles or colors in real-time, supporting informed decision-making. Expanding payment options to include digital wallets and buy-now-pay-later services would offer greater flexibility and convenience, enhancing the booking and payment process. Furthermore, the platform's analytics capabilities could be broadened to include performance metrics like client retention rates, popular services, and predictive analytics, aiding in inventory management and promotional planning.

Implementing loyalty programs and automated marketing through CRM tools would promote client retention by providing personalized offers, reminders, and rewards. Adding multilingual support could also attract a broader clientele by enhancing accessibility. Security remains a priority, and future improvements may focus on advanced data privacy measures like multi-factor authentication and enhanced encryption protocols to ensure the safety of client information. Finally, integrating staff training resources within the platform could facilitate continuous professional development, ensuring that salon staff are up-to-date with the latest techniques and trends. By pursuing these advancements, the platform can stay competitive, meet evolving client expectations, and streamline salon operations for sustained growth.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley Professional.
- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill.
- Django Documentation (2023). *Comprehensive reference for Django framework functionalities and best practices for backend development*. Retrieved from: <https://docs.djangoproject.com/en/stable/>
- Python Documentation (2023). *Guide and reference for programming concepts, libraries, and tools in Python*. Retrieved from: <https://docs.python.org/3/>
- MySQL Documentation (2023). *Resource for managing data storage and retrieval in Django applications*. Retrieved from: <https://dev.mysql.com/doc/>
- Bootstrap Documentation (2023). *Utilized for designing responsive frontend elements in the project*. Retrieved from: <https://getbootstrap.com/docs/>
- Stripe API Documentation (2023). *Reference for integrating secure online payment processing within the platform*. Retrieved from: <https://stripe.com/docs>

WEBSITES:

- Django Project. *Tutorials, documentation, and community resources for Django development*. Retrieved from: <https://www.djangoproject.com/>
- Python Package Index (PyPI). Retrieved from: <https://pypi.org/>
- Booking.com. Retrieved from: <https://www.booking.com/>
- HomeSalon. *Industry reference for online salon management solutions*. Retrieved from: <https://www.homesalon.in/>
- Razorpay (2023). *Guide for integrating payment solutions for Indian markets*. Retrieved from: <https://razorpay.com/>

CHAPTER 9

APPENDIX

9.1 Sample Code

user_dashboard.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>GlamourQuest Dashboard</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">
  <style>
    :root {
      --primary-color: #FF69B4; /* Hot Pink */
      --secondary-color: #FFB6C1; /* Light Pink */
      --accent-color: #FF1493; /* Deep Pink */
      --text-color: #333;
      --bg-color: #FFF0F5; /* Lavender Blush */
      --card-bg: #ffffff;
    }
    body {
      font-family: 'Poppins', sans-serif;
      background-color: var(--bg-color);
      color: var(--text-color);
    }
    .sidebar {
      height: 100vh;
      background: linear-gradient(135deg, var(--primary-color), var(--secondary-color));
      color: white;
      padding-top: 20px;
    }
```

```
.sidebar-sticky {
  position: sticky;
  top: 0;
  height: calc(100vh - 48px);
  padding-top: .5rem;
  overflow-x: hidden;
  overflow-y: auto;
}

.sidebar .nav-link {
  color: white;
  margin-bottom: 10px;
  transition: all 0.3s;
}

.sidebar .nav-link:hover {
  background-color: rgba(255,255,255,0.1);
  transform: translateX(5px);
}

.sidebar .nav-link.active {
  background-color: rgba(255,255,255,0.2);
}

.main-content {
  padding: 20px;
}

.header {
  background: linear-gradient(135deg, var(--primary-color), var(--secondary-color));
  color: white;
  box-shadow: 0 4px 6px rgba(0,0,0,0.1);
  padding: 15px 20px;
  margin-bottom: 1rem;
  border-radius: 10px;
}

.header h1 {
  margin: 0;
  font-size: 1.5rem;
```

```
}  
.header .profile-dropdown .dropdown-toggle {  
  color: white;  
}  
.header .profile-dropdown .dropdown-menu {  
  background-color: var(--card-bg);  
}  
.header .profile-dropdown .dropdown-item:hover {  
  background-color: var(--secondary-color);  
  color: white;  
}  
.profile-dropdown .dropdown-toggle::after {  
  display: none;  
}  
.card {  
  border: none;  
  box-shadow: 0 4px 6px rgba(0,0,0,0.1);  
  transition: all 0.3s;  
}  
.card:hover {  
  transform: translateY(-5px);  
  box-shadow: 0 6px 8px rgba(0,0,0,0.15);  
}  
.scrolling-background {  
  height: 500px;  
  background-size: cover;  
  background-position: center;  
  transition: background-image 1s ease-in-out;  
  border-radius: 15px;  
  box-shadow: 0 6px 12px rgba(0,0,0,0.15);  
  margin-bottom: 2rem;  
  position: relative;  
}  
.scrolling-background::before {
```

```
        content: "";
        position: absolute;
        top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background: linear-gradient(rgba(255,105,180,0.3), rgba(255,182,193,0.3));
        z-index: 1;
    }
    .logo-container {
        text-align: center;
        padding: 20px 0;
    }
    .logo {
        width: 120px;
        height: 120px;
        border-radius: 50%;
        object-fit: cover;
        border: 3px solid white;
        box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    }
    .arrow {
        position: absolute;
        top: 50%;
        transform: translateY(-50%);
        font-size: 2rem;
        color: white;
        background-color: rgba(255, 105, 180, 0.6); /* Pink shade with transparency */
        padding: 15px;
        cursor: pointer;
        z-index: 2;
        transition: all 0.3s ease;
        border-radius: 50%;
        width: 60px;
```

```
    height: 60px;
    display: flex;
    align-items: center;
    justify-content: center;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
  }

  .arrow:hover {
    background-color: rgba(255, 105, 180, 0.8); /* Darker pink on hover */
    transform: translateY(-50%) scale(1.1);
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
  }

  .arrow-left {
    left: 20px;
  }

  .arrow-right {
    right: 20px;
  }
</style>
</head>
<body>
  <div class="container-fluid">
    <div class="row">
      <!-- Sidebar -->
      <nav id="sidebar" class="col-md-3 col-lg-2 d-md-block sidebar collapse">
        <div class="position-sticky sidebar-sticky">
          <h2 class="text-center mb-4">GlamourQuest</h2>
          <div class="logo-container">
            
          </div>
          <ul class="nav flex-column">
```

```
<li class="nav-item">
  <a class="nav-link active" href="{ % url 'client_dashboard' % }">
    <i class="fas fa-home me-2"></i> Dashboard
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="{ % url 'client_profile' % }">
    <i class="fas fa-user me-2"></i> Profile
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="{ % url 'client_current_bookings' % }">
    <i class="fas fa-calendar-alt me-2"></i> Appointments
  </a>
</li>
<li class="nav-item"></li>
  <a class="nav-link" href="{ % url 'client_bookings' % }">
    <i class="fas fa-concierge-bell me-2"></i> Feedback
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="{ % url 'client_services' % }">
    <i class="fas fa-concierge-bell me-2"></i> Services
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="{ % url 'hair_care_services' % }">
    <i class="fas fa-cut me-2"></i> Hair Care
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="{ % url 'facial_services' % }">
    <i class="fas fa-spa me-2"></i> Facial
  </a>
```

```

    </li>
    <li class="nav-item">
        <a class="nav-link" href="{ % url 'mani-pedi-services' % }">
            <i class="fas fa-hand-sparkles me-2"></i> Mani & Pedi
        </a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="{ % url 'waxing-services' % }">
            <i class="fas fa-feather me-2"></i> Waxing
        </a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="{ % url 'service_history' % }">
            <i class="fas fa-feather me-2"></i> Service History
        </a>
    </li>
</ul>
</div>
</nav>

<!-- Main content -->
<main class="col-md-9 ms-sm-auto col-lg-10 px-md-4">
    <!-- Header -->
    <header class="header d-flex justify-content-between align-items-center mb-4 p-3 bg-
white rounded shadow-sm">
        <h1 class="h3 mb-0">Welcome, { { client.first_name } }!</h1>
        <div class="profile-dropdown dropdown">
            <a href="#" class="d-flex align-items-center text-decoration-none dropdown-
toggle" id="dropdownUser1" data-bs-toggle="dropdown" aria-expanded="false">
                
                <strong>{ { client.first_name } }</strong>
            </a>
            <ul class="dropdown-menu dropdown-menu-end text-small shadow" aria-

```



```

labelledby="dropdownUser1">
    <li><a class="dropdown-item" href="{ % url 'client_profile'
% }">Profile</a></li>
    <li><a class="dropdown-item" href="{ % url 'client_update' % }">Update
Profile</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="{ % url 'logout' % }">Sign out</a></li>
</ul>
</div>
</header>

<!-- Scrolling background with arrows -->
<div class="scrolling-background">
    <div class="arrow arrow-left" onclick="changeBackground(-1)">&#10094;</div>
    <div class="arrow arrow-right" onclick="changeBackground(1)">&#10095;</div>
</div>

<!-- Additional content can be added here -->

</main>
</div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
<script>
    let currentIndex = 0;
    const backgrounds = [
        "{ % static 'images/hair-styling.jpg' % }",
        "{ % static 'images/manicure.jpg' % }",
        "{ % static 'images/hair-spa.jpg' % }",
        "{ % static 'images/all-type.jpg' % }"
    ];

```

```
function changeBackground(direction = 1) {  
    currentIndex = (currentIndex + direction + backgrounds.length) % backgrounds.length;  
    updateBackground();  
}  
  
function updateBackground() {  
    const scrollingBackground = document.querySelector('.scrolling-background');  
    scrollingBackground.style.backgroundImage = `url(${backgrounds[currentIndex]})`;  
}  
  
// Set initial background and start cycling  
window.onload = function() {  
    updateBackground();  
    setInterval(() => changeBackground(1), 5000); // Change background every 5 seconds  
};  
</script>  
</body>  
</html>
```

Userdashboard view functions:

```
from django.shortcuts import render, redirect  
from django.contrib import messages  
from .models import Client  
from .forms import ClientProfileUpdateForm  
from django.shortcuts import render, redirect  
from django.contrib import messages  
from django.db.models import Q  
from .models import Client, ServiceSubcategory, Service  
from .models import Service, Booking, Employee, Client  
from .forms import BookingForm  
from django.utils import timezone  
from django.core.exceptions import ValidationError
```

```
from .models import Booking, Feedback
from .forms import FeedbackForm

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def client_dashboard(request):
    user_id = request.session.get('user_id')
    user_type = request.session.get('user_type')

    if not user_id or user_type != 'client':
        messages.error(request, "You need to log in to access the dashboard.")
        return redirect('login')

    try:
        client = Client.objects.get(id=user_id)
    except Client.DoesNotExist:
        messages.error(request, "Client not found.")
        return redirect('login')

    context = {
        'client': client,
    }

    return render(request, 'client_dashboard.html', context)

def toggle_client_status(request, client_id):
    if request.method == 'POST':
        client = Client.objects.get(id=client_id)
        client.status = not client.status # Toggle the status
        client.save()

        messages.success(request, f"Client {client.first_name}'s status updated to {'Active' if client.status else 'Inactive'}.")
        return redirect('manage_client')

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def client_update(request):
```

```
user_id = request.session.get('user_id')
if not user_id:
    messages.error(request, "You want to login to access dashboard.")
    return redirect('login')

client = Client.objects.get(id=user_id)

if request.method == 'POST':
    form = ClientProfileUpdateForm(request.POST, instance=client)
    if form.is_valid():
        form.save()
        messages.success(request, 'Profile updated successfully!')
        return redirect('client_dashboard') # Redirect to the client dashboard
    else:
        form = ClientProfileUpdateForm(instance=client)

return render(request, 'client_update.html', {'form': form, 'client': client})

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def client_profile(request):
    # Assuming you store the user ID in the session
    user_id = request.session.get('user_id')
    if not user_id:
        messages.error(request, "You want to login to access dashboard.")
        return redirect('login')

    # Fetch the client object based on the stored user ID
    client = Client.objects.get(id=user_id)

    # Pass the client object to the template for display
    return render(request, 'client_profile.html', {'client': client})

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def client_services(request):
```

```
# Assuming you store the user ID in the session, or use the request's user object
user_id = request.session.get('user_id') # Adjust this if needed based on how you store the session
if not user_id:
    messages.error(request, "You want to login to access dashboard.")
    return redirect('login')
client = Client.objects.get(id=user_id)

# Fetch all services from the database
services = Service.objects.all()

# Pass the client and services to the template
context = {
    'client': client, # Client data for the profile name and other info
    'services': services, # Services data to list available services
}

return render(request, 'client_services.html', context)

def hair_care_services(request):
    user_id = request.session.get('user_id')
    if not user_id:
        messages.error(request, "You need to log in to access the dashboard.")
        return redirect('login')

    try:
        client = Client.objects.get(id=user_id)
    except Client.DoesNotExist:
        messages.error(request, "Client profile not found.")
        return redirect('login')

    # Fetch all service subcategories where category name is 'Hair Care'
    hair_care_subcategories = ServiceSubcategory.objects.filter(category__name='Hair Care')

    # Handle search
```

```
query = request.GET.get('query', '')
if query:
    services = Service.objects.filter(
        Q(service_name__icontains=query) |
        Q(description__icontains=query) |
        Q(subcategory__name__icontains=query) |
        Q(subcategory__category__name__icontains=query),
        subcategory__category__name='Hair Care' # Ensure we're only searching within hair care
    services
    ).distinct()
else:
    services = None

context = {
    'hair_care_subcategories': hair_care_subcategories,
    'client': client,
    'services': services,
    'query': query,
}

return render(request, 'hair_care_services.html', context)

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def services_in_subcategory(request, subcategory_id):
    user_id = request.session.get('user_id') # Adjust this if needed based on how you store the session
    if not user_id:
        messages.error(request, "You want to login to access dashboard.")
        return redirect('login')
    client = Client.objects.get(id=user_id)

    # Fetch the subcategory and its related services
    subcategory = get_object_or_404(ServiceSubcategory, id=subcategory_id)
    services = Service.objects.filter(subcategory=subcategory)
```

```
context = {
    'subcategory': subcategory,
    'services': services,
    'client': client,
}

return render(request, 'services_in_subcategory.html', context)

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def facial_services(request):
    user_id = request.session.get('user_id')
    if not user_id:
        messages.error(request, "You need to log in to access the dashboard.")
        return redirect('login')

    try:
        client = Client.objects.get(id=user_id)
    except Client.DoesNotExist:
        messages.error(request, "Client profile not found.")
        return redirect('login')

    # Fetch all service subcategories where category id is 2 (assuming 2 is for Facial)
    facial_service_subcategories = ServiceSubcategory.objects.filter(category_id=2)

    # Handle search
    query = request.GET.get('query', '')
    if query:
        services = Service.objects.filter(
            Q(service_name__icontains=query) |
            Q(description__icontains=query) |
            Q(subcategory__name__icontains=query) |
            Q(subcategory__category__name__icontains=query),
            subcategory__category_id=2 # Ensure we're only searching within facial services
        ).distinct()
```

```
    else:
        services = None

    context = {
        'facial_service_subcategories': facial_service_subcategories,
        'client': client,
        'services': services,
        'query': query,
    }

    return render(request, 'facial_services.html', context)

def hair_cut_services(request):
    return render(request, 'hair_cut_services.html')

def all_type_skin(request):
    return render(request, 'all_type_skin.html')

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def mani_pedi_services(request):
    user_id = request.session.get('user_id')
    if not user_id:
        messages.error(request, "You need to log in to access the dashboard.")
        return redirect('login')

    try:
        client = Client.objects.get(id=user_id)
    except Client.DoesNotExist:
        messages.error(request, "Client profile not found.")
        return redirect('login')

    mani_pedi_service_subcategories = ServiceSubcategory.objects.filter(category_id=3)

    # Handle search
```



```
query = request.GET.get('query', '')
if query:
    services = Service.objects.filter(
        Q(service_name__icontains=query) |
        Q(description__icontains=query) |
        Q(subcategory__name__icontains=query) |
        Q(subcategory__category__name__icontains=query),
        subcategory__category_id=3 # Ensure we're only searching within mani & pedi services
    ).distinct()
else:
    services = None

context = {
    'mani_pedi_service_subcategories': mani_pedi_service_subcategories,
    'client': client,
    'services': services,
    'query': query,
}

return render(request, 'mani-pedi-services.html', context)

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def waxing_services(request):
    user_id = request.session.get('user_id')
    if not user_id:
        messages.error(request, "You need to log in to access the dashboard.")
        return redirect('login')

    try:
        client = Client.objects.get(id=user_id)
    except Client.DoesNotExist:
        messages.error(request, "Client profile not found.")
        return redirect('login')
```

```
waxing_service_subcategories = ServiceSubcategory.objects.filter(category_id=4)
```

```
# Handle search
```

```
query = request.GET.get('query', '')
```

```
if query:
```

```
    services = Service.objects.filter(
```

```
        Q(service_name__icontains=query) |
```

```
        Q(description__icontains=query) |
```

```
        Q(subcategory__name__icontains=query) |
```

```
        Q(subcategory__category__name__icontains=query),
```

```
        subcategory__category_id=4 # Ensure we're only searching within waxing services
```

```
    ).distinct()
```

```
else:
```

```
    services = None
```

```
context = {
```

```
    'waxing_service_subcategories': waxing_service_subcategories,
```

```
    'client': client,
```

```
    'services': services,
```

```
    'query': query,
```

```
}
```

```
return render(request, 'waxing-services.html', context)
```

```
def booking_service(request, service_id):
```

```
    service = get_object_or_404(Service, id=service_id)
```

```
    user_id = request.session.get('user_id')
```

```
    client = get_object_or_404(Client, id=user_id)
```

```
    specialized_employees = Employee.objects.filter(
```

```
        specializations=service.subcategory.category.specialization,
```

```
        approved=True,
```

```
        status=True
```

```
    ).distinct()
```

```
# Check if the client has already booked this service
existing_client_booking = Booking.objects.filter(
    client=client,
    service=service,
    status__in=['Pending', 'Confirmed']
).exists()

if request.method == 'POST':
    form = BookingForm(request.POST, specialized_employees=specialized_employees)
    if form.is_valid():
        booking = form.save(commit=False)
        booking.client = client
        booking.service = service

        # Check if the selected time is valid (not in the past)
        now = timezone.now()

        selected_datetime =
        timezone.make_aware(timezone.datetime.combine(booking.booking_date,
        booking.booking_time))
        if selected_datetime <= now:
            form.add_error('booking_time', 'Please select a future time.')
            messages.error(request, "Please select a future time.")
        else:
            try:
                booking.full_clean()
            except ValidationError as e:
                form.add_error(None, e)
            else:
                # Check if the selected time slot is available for the chosen employee
                existing_bookings = Booking.objects.filter(
                    staff=booking.staff,
                    booking_date=booking.booking_date,
                    booking_time=booking.booking_time,
```

```
        status__in=['Pending', 'Confirmed']
    )

    if existing_bookings.exists():
        messages.error(request, "This time slot is already booked for the selected employee.
Please choose another time or employee.")
    else:
        # Set the total cost based on the service rate
        booking.total_cost = service.rate # Assuming the service has a rate field

    if existing_client_booking:
        # If rebooking is confirmed
        if request.POST.get('confirm_rebooking') == 'yes':
            booking.save()
            messages.success(request, "Your booking has been confirmed!")
            return redirect('billing', booking_id=booking.id) # Redirect to billing
        else:
            # Show rebooking confirmation
            context = {
                'service': service,
                'form': form,
                'existing_booking': existing_client_booking,
                'show_rebooking_confirmation': True
            }
            return render(request, 'booking_service.html', context)
    else:
        booking.save()
        messages.success(request, "Your booking has been confirmed!")
        return redirect('billing', booking_id=booking.id) # Redirect to billing

    else:
        messages.error(request, "Please correct the errors below.")

    else:
        form = BookingForm(specialized_employees=specialized_employees)
```

```
context = {
    'service': service,
    'form': form,
    'existing_booking': existing_client_booking,
    'current_time': timezone.now(), # Pass current time to the template
}

return render(request, 'booking_service.html', context)

# The booking_confirmation view remains unchanged
def booking_confirmation(request, booking_id):
    user_id = request.session.get('user_id')
    client = get_object_or_404(Client, id=user_id)
    booking = get_object_or_404(Booking, id=booking_id, client=client)
    return render(request, 'booking_confirmation.html', {'booking': booking})

def client_current_bookings(request):
    user_id = request.session.get('user_id')
    client = get_object_or_404(Client, id=user_id)

    # Get only pending and confirmed bookings for the client
    current_bookings = Booking.objects.filter(
        client=client,
        status__in=['Pending', 'Confirmed'],
        booking_date__gte=timezone.now().date() # Only future and today's bookings
    ).order_by('booking_date', 'booking_time')

    context = {
        'client': client,
        'bookings': current_bookings,
        'today': timezone.now().date(),
    }

    return render(request, 'client_bookings.html', context)
```

```
def cancel_booking(request, booking_id):  
    # Get the user_id from the session  
    user_id = request.session.get('user_id')  
  
    # Get the client object or return 404 if not found  
    client = get_object_or_404(Client, id=user_id)  
  
    # Get the booking object or return 404 if not found  
    booking = get_object_or_404(Booking, id=booking_id, client=client)  
  
    # Check if the booking can be cancelled  
    if booking.status in ['Pending', 'Confirmed'] and booking.booking_date >= timezone.now().date():  
        # Update the booking status to 'Cancelled'  
        booking.status = 'Cancelled'  
        booking.save()  
  
        messages.success(request, f"Your booking for {booking.service.service_name} on  
{booking.booking_date} has been cancelled successfully.")  
    else:  
        messages.error(request, "This booking cannot be cancelled. It may be in the past or already  
completed/cancelled.")  
  
    # Redirect to the current bookings page  
    return redirect('client_dashboard') # Make sure this URL name exists in your urls.py  
  
def add_feedback(request, booking_id):  
    booking = get_object_or_404(Booking, id=booking_id)  
  
    if request.method == 'POST':  
        form = FeedbackForm(request.POST)  
        if form.is_valid():  
            feedback = form.save(commit=False)  
            feedback.booking = booking  
            feedback.save()
```

```
        messages.success(request, 'Feedback submitted successfully!')
        return redirect('client_bookings')
    else:
        form = FeedbackForm()

    return render(request, 'add_feedback.html', {'form': form, 'booking': booking})

def view_feedback(request, booking_id):
    booking = get_object_or_404(Booking, id=booking_id)
    feedback = get_object_or_404(Feedback, booking=booking)
    return render(request, 'view_feedback.html', {'feedback': feedback})
```

9.2 Screen Shots

Login Page

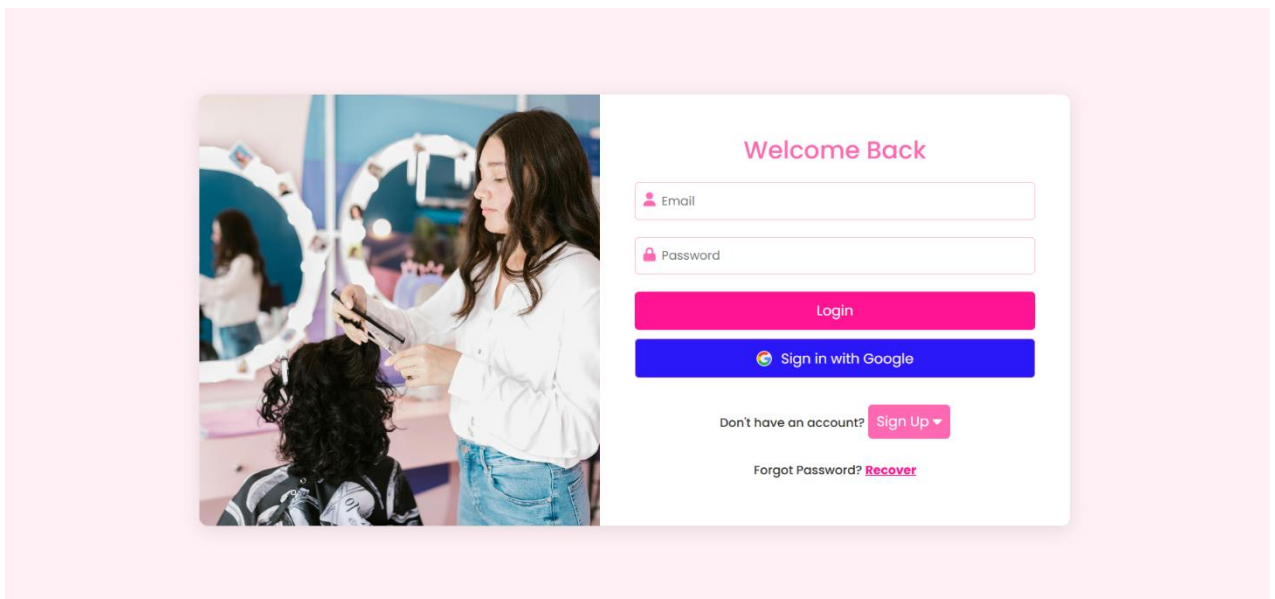
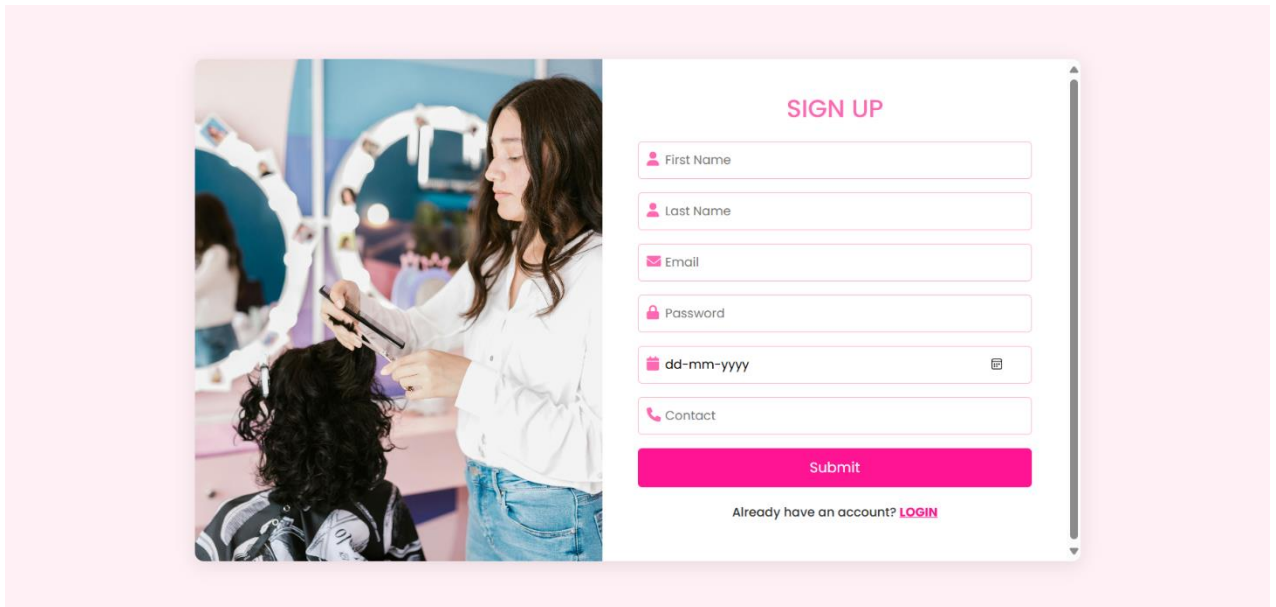


Fig: 9.2.1

Registration Page



The registration page features a light pink background. On the left, there is a vertical image of a woman in a white shirt styling the hair of a person with dark curly hair. On the right, a white card titled "SIGN UP" contains a form with the following fields: First Name, Last Name, Email, Password, a date field labeled "dd-mm-yyyy", and Contact. A pink "Submit" button is at the bottom of the form, followed by the text "Already have an account? [LOGIN](#)".

Fig: 9.2.2

User Dashboard

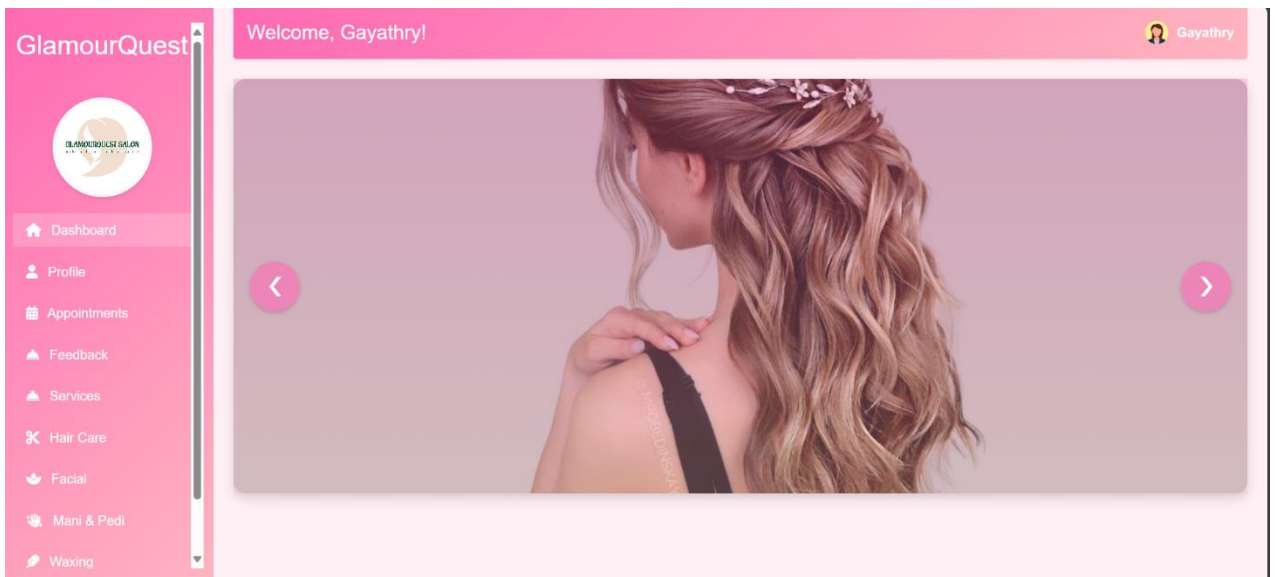
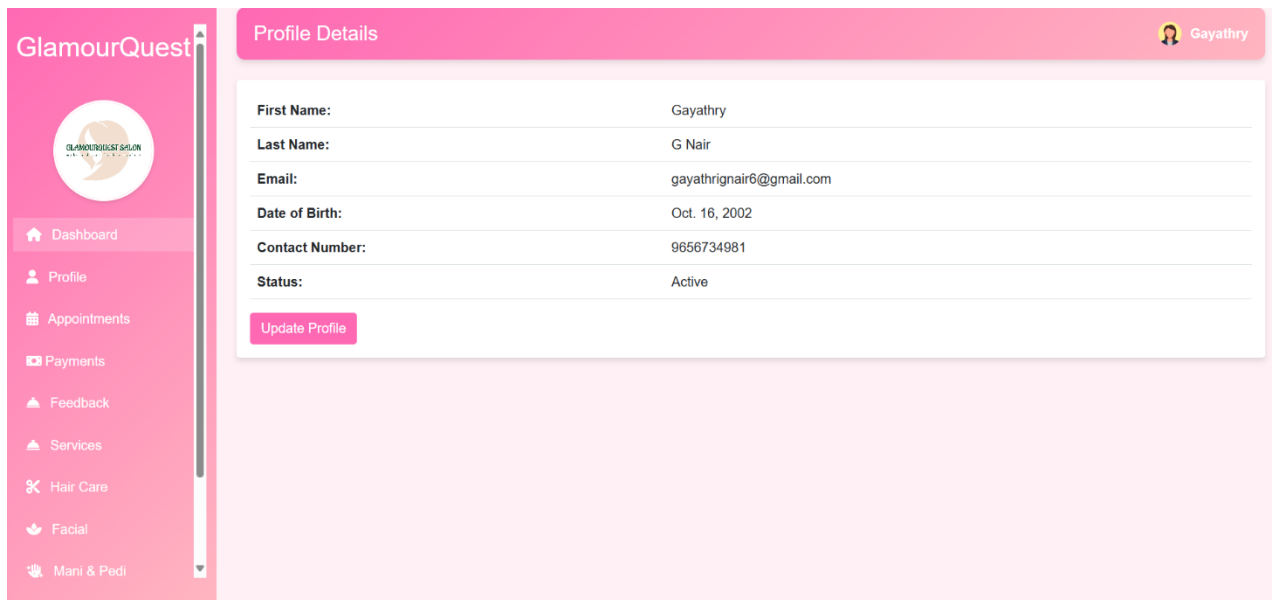


Fig: 9.2.3

User Profile



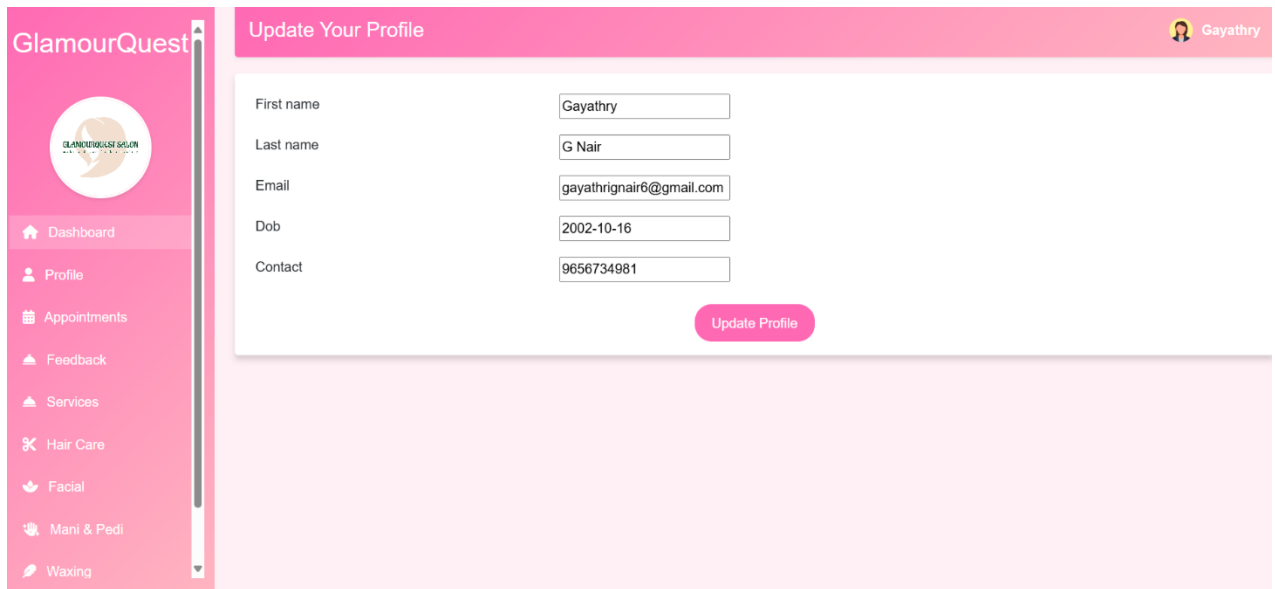
The screenshot shows the 'User Profile' page of the GlamourQuest application. On the left is a pink sidebar with the 'GlamourQuest' logo and a list of navigation items: Dashboard, Profile, Appointments, Payments, Feedback, Services, Hair Care, Facial, and Mani & Pedi. The main content area has a pink header 'Profile Details' with a user icon and the name 'Gayathry'. Below this is a white box containing a table of profile information. At the bottom of this box is a pink 'Update Profile' button.

First Name:	Gayathry
Last Name:	G Nair
Email:	gayathrignair6@gmail.com
Date of Birth:	Oct. 16, 2002
Contact Number:	9656734981
Status:	Active

Update Profile

Fig: 9.2.4

Update Profile



The screenshot shows the 'Update Your Profile' page of the GlamourQuest application. It features the same pink sidebar as the previous page. The main content area has a pink header 'Update Your Profile' with a user icon and the name 'Gayathry'. Below this is a white box with input fields for updating profile information. At the bottom right of this box is a pink 'Update Profile' button.

First name	<input type="text" value="Gayathry"/>
Last name	<input type="text" value="G Nair"/>
Email	<input type="text" value="gayathrignair6@gmail.com"/>
Dob	<input type="text" value="2002-10-16"/>
Contact	<input type="text" value="9656734981"/>

Update Profile

Fig: 9.2.5

Hair Care Services

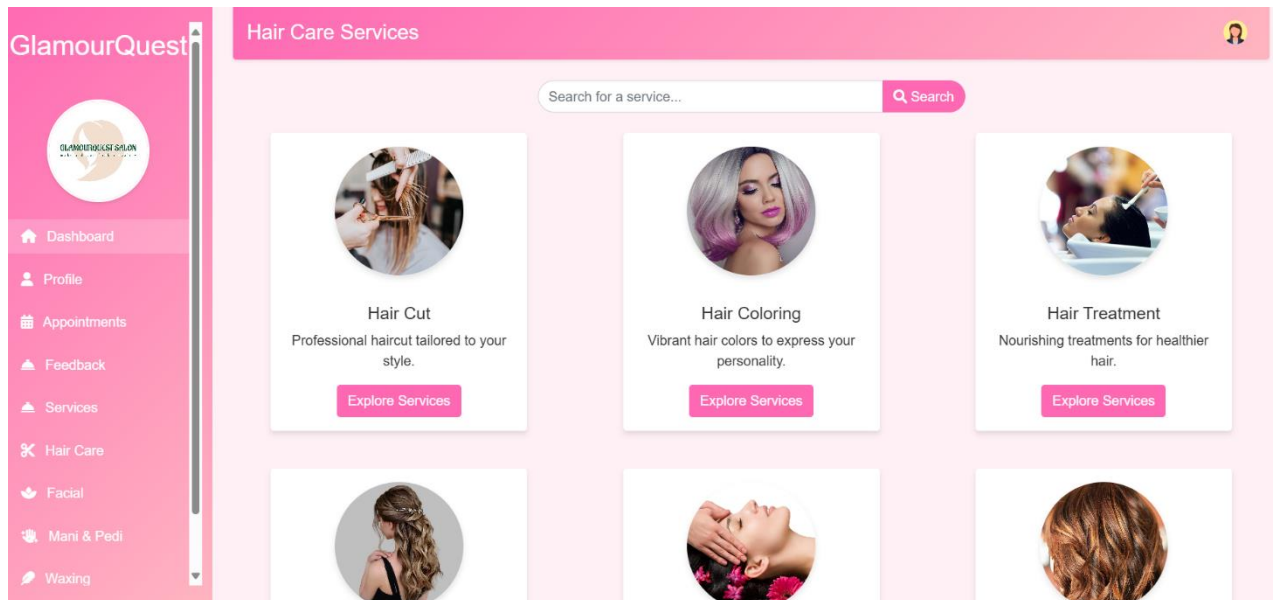


Fig: 9.2.6

Hair Cut Services

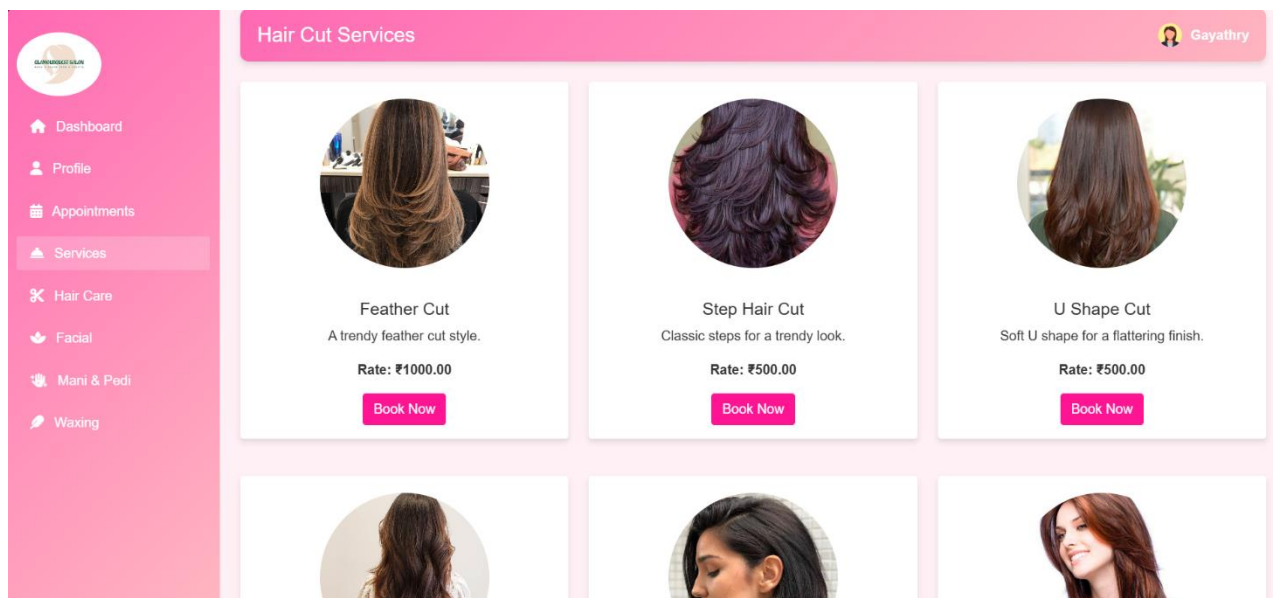


Fig: 9.2.7

Book Services

Book Service

Book Service: Feather Cut

Rate: ₹1000.00

Booking date:

Booking time:

Staff:

Additional notes:

Confirm Booking

Fig: 9.2.8

Client Current Appointments

Your Booking

Gayathry

Your Appoinments

Service	Date	Time	Status	Action
U Shape Cut	Nov. 7, 2024	8 a.m.	Pending	<button>Cancel</button>

Fig: 9.2.9

Feedback

GlamourQuest



[Dashboard](#)
[Profile](#)
[Appointments](#)
[Feedback](#)
[Services](#)
[Hair Care](#)
[Facial](#)
[Mani & Pedi](#)
[Waxing](#)

Add Feedback

Add Feedback for Feather Cut

Rating


Comment

Submit Feedback

Fig: 9.2.10

Payment Status

GlamourQuest



[Dashboard](#)
[Profile](#)
[Appointments](#)
[Payments](#)
[Feedback](#)
[Services](#)
[Hair Care](#)
[Facial](#)
[Mani & Pedi](#)

Payments

Service	Date	Time	Amount (₹)	Status	Action
Feather Cut	Nov. 6, 2024	11:57 a.m.	1000.00	Paid	Pay Now
Feather Cut	Nov. 6, 2024	12:09 p.m.	1000.00	Pending	Pay Now

Back to Dashboard

Fig: 9.2.11

Razarpay Payment

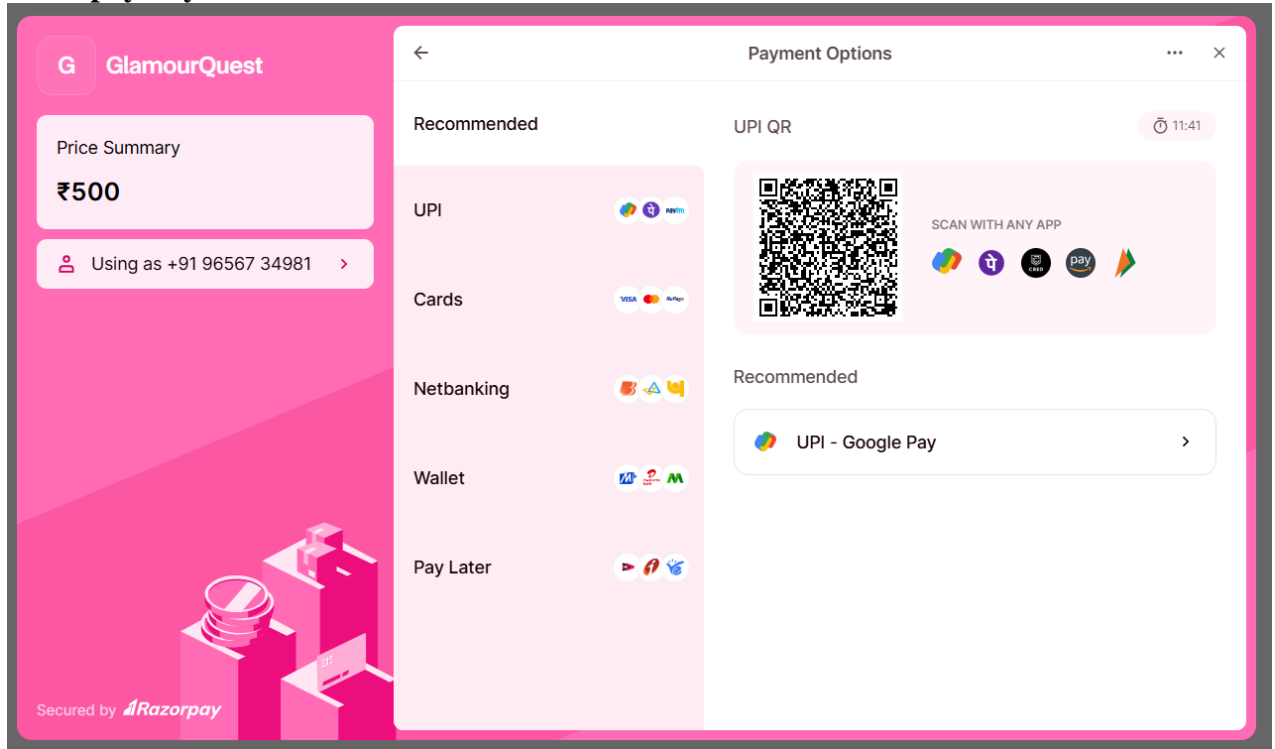


Fig: 9.2.12

