Beauty Service Booking System

Final Report

|  |  |
| --- | --- |
| Group 3 | |
| Group members: | Samira Bechi  Michael Maduakolam  Nhan Nguyen  Abel Thomas  Quan Vu |
| Professor | Mr. Emre Yilmaz |

-Houston, Dec 2024-

**Beauty Service Booking System**

1. **Introduction**

The **Service Booking System** is an advanced solution tailored to enhance the management of appointments and services within a business setting. Designed with efficiency and user-friendliness in mind, the system streamlines operations for both customers and service providers, while offering comprehensive administrative control to business managers.

This system serves as a centralized platform that allows customers to effortlessly book appointments for a variety of services, such as spa treatments, consultations, or classes, based on their preferences. It eliminates the need for manual scheduling, offering a seamless booking experience. Meanwhile, staff members can manage their schedules in real time, updating their availability and monitoring their assigned bookings. Administrators hold the reins of the system, with capabilities to oversee all operations, including user management, service configuration, and the generation of insightful reports.

The **Service Booking System** is structured to cater to three core user roles:

* **Admin**: Admins oversee the entire system, managing staff, services, and customer records. They ensure smooth operations by configuring the system's settings, resolving conflicts in schedules, and analyzing reports to enhance service efficiency.
* **Staff/Service Providers**: Staff members use the platform to view their schedules, manage their availability, and update the status of services delivered. They play a crucial role in maintaining customer satisfaction by ensuring prompt and efficient service delivery.
* **Customers**: Customers are at the heart of the system. They can browse available services, book appointments, modify or cancel bookings, and view their booking history. The intuitive design prioritizes their convenience and satisfaction.

**Core Features**

1. **Booking Management**: A robust booking system that allows customers to reserve time slots for services while preventing conflicts.
2. **Role-Based Access**: Differentiated functionalities for Admins, Staff, and Customers ensure the system remains secure and organized.
3. **Service Management**: Admins can add, modify, or delete services offered by the business, ensuring a dynamic and customizable service catalog.
4. **Staff Availability and Scheduling**: Staff can update their availability, allowing customers to book only at convenient times while maintaining operational efficiency.
5. **Administrative Control**: Admins manage user roles, monitor operations, and generate reports to analyze trends and optimize the business.
6. **Notifications and Reminders**: Notifications keep users informed about upcoming bookings or changes to their schedules, enhancing reliability.

**Why This System?**

Modern businesses often struggle with managing appointments efficiently, leading to errors, missed bookings, and dissatisfied customers. This system was conceived to address these challenges by leveraging technology to automate and simplify the process. By uniting customers, staff, and admins on a single platform, the **Service Booking System** bridges the gap between business operations and customer satisfaction.

**Glossary**

* **Booking**: The act of reserving a specific time slot for a service provided by the business.
* **Service**: An offering available for booking (e.g., spa treatment, medical consultation, personal training).
* **Admin**: A user responsible for overseeing the system, managing staff and services, and analyzing reports.
* **Customer**: A user who interacts with the system to browse, book, or manage appointments.
* **Staff/Service Provider**: Employees who manage their availability, view schedules, and update booking statuses.

1. **Functional Requirements**
2. The system must allow admins to manage user accounts (staff and customers) and oversee all system operations.
3. Customers must be able to register for an account, log in, and view available services and time slots.
4. Customers can book an appointment by selecting a service, choosing a time slot, and confirming their booking.
5. The system should allow customers to cancel or reschedule their appointments.
6. Admins must be able to add new services, set pricing, and assign staff members to specific services.
7. The system must allow staff to view and manage their schedules, including confirming or canceling appointments.
8. Staff must be able to update the status of a booking (e.g., completed, canceled).
9. Admins must be able to generate reports on the number of bookings, staff performance, and service usage.
10. The system must send automated email notifications for appointment confirmations, cancellations, and payment receipts.
11. Admins should be able to create and manage staff schedules.
12. **Use Cases**
13. Login (All Roles):
    1. Brief Description: All users (admins, staff, customers) log in to access the system.
    2. Precondition: The user must have valid credentials.
    3. Postcondition: The user is logged into the system and can perform actions based on their role.
       1. A diagram of a service

          Description automatically generated
14. Register New User (Admin):
    1. Brief Description: Admins can register new customers or staff members in the system.
    2. Precondition: The user must have admin access.
    3. Postcondition: The new user is added to the system.
       1. A diagram of a service

          Description automatically generated
15. Book Service (Customer):
    1. Brief Description: Customers select a service, choose an available date and time, and confirm the booking.
    2. Precondition: The customer must be logged in and select an available service and time.
    3. Postcondition: The booking is confirmed, and an email confirmation is sent to the customer.
       1. A diagram of service

          Description automatically generated
16. Cancel Booking (Admin/Customer):
    1. Brief Description: Customers can cancel a previously booked appointment.
    2. Precondition: The customer must have a valid booking.
    3. Postcondition: The booking is canceled, and a cancellation confirmation is sent to the customer.
       1. A diagram of service boarding system

          Description automatically generated
17. Select Treatment (Customer):
    1. Brief Description: Customers select the type of service (treatment) they want to book. Admin can do in case the customer asking them to do.
    2. Precondition: The customer must be logged in and the service must be available.
    3. Postcondition: The service is selected, and the customer can proceed to book the appointment.
       1. A diagram of service

          Description automatically generated
18. Manage Schedule (Admin/Staff):
    1. Brief Description: Staff can view and update their daily schedule by marking appointments as completed or canceled.
    2. Precondition: Staff must be logged in.
    3. Postcondition: The staff schedule is updated, and appointment statuses are reflected in the system.

A diagram of service schedule

Description automatically generated

1. View Customer Details (Staff/Admin):
   1. Brief Description: Staff and admins can view details of the customers they are serving.
   2. Precondition: The user must have access rights (staff or admin).
   3. A screenshot of a service

      Description automatically generatedPostcondition: Customer details are displayed.
2. Add/Remove Service (Admin):
   1. Brief Description: Admins can add new services to the system, set prices, and assign staff members to the services.
   2. Precondition: The user must have admin access.
   3. Postcondition: The service is added or removed from the system.

A diagram of service

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Figure 1: Full use case Diagram

Sequence of Operations

1. Customer registers for an account and logs into the system.
2. Customer selects a treatment, chooses a date and time, and confirms the booking.
3. The system sends a booking confirmation to the customer.
4. Customer proceeds to make a payment for the booking.
5. Staff views their updated schedule and marks completed appointments.
6. Admin manages staff schedules, adds or removes services, and generates reports on system performance.

**A black background with white dots

Description automatically generatedIV. Class diagram**

Figure 1: Class Diagram

A screenshot of a computer screen

Description automatically generated

**Figure 1: Class Diagram with All Attributes and Methods**

**V. State chart Diagram**

A blue and black lines

Description automatically generated with medium confidence

Figure 3: Project State chart diagram

**VI. Use case Realization**

1. User Log In

Scenario: Users want to log in to their account in the Beauty Salon System.

Pre-condition: The user must have an existing account (Admin, Staff, or Customer).

Steps:

1. The user opens the Beauty Salon System app.
2. The user selects the "Login" option.
3. The system displays the login interface, prompting the user to enter their username and password.
4. The user inputs their username and password in the respective fields.
5. The system sends the entered credentials to the Login Service for validation.
6. The Login Service verifies the credentials.
   * If the credentials are valid, the system logs the user into their account and redirects them to the appropriate interface based on their role (Admin, Staff, or Customer).
   * If the credentials are invalid, the system displays an error message informing the user to re-enter valid login information.
7. The user is directed to their respective dashboard (Admin, Staff, or Customer).

Post-condition: The user successfully logs in and is taken to their appropriate dashboard, or they receive an error message prompting them to retry.

A screenshot of a computer screen

Description automatically generated

2. Register new user

Scenario: Admin wants to register a new user (could be another admin, staff, or customer) in the Beauty Salon System.

Pre-condition: The admin must be logged in and have appropriate permissions to register new users.

Steps:

1. The admin opens the Register New Account page from the system interface.
2. The system prompts the admin to enter the new user’s information (name, username, password, role, etc.).
3. The admin inputs the required details (Name, Username, Password, Role) and submits the information.
4. The system verifies that the inputted information is valid (e.g., ensuring the username is not already taken and passwords meet criteria).
5. The system includes the following actions based on the entered information:
   * Insert User Info: The system stores the new user information in the database.
   * Update User Info: If the admin modifies the entered information before submission, the system updates it accordingly.
   * Delete User: If the admin cancels the operation, the system discards the entered information.
   * Search User: The system checks if a user with the same username already exists in the database.
6. Once all the information is successfully validated and inserted, the system confirms the successful registration of the new user.
7. The system returns to the admin interface, allowing the admin to either view the newly added user or proceed with other actions.

A screenshot of a computer screen

Description automatically generatedPost-condition: The new user account is successfully registered and stored in the system.

3. Booking Service

Scenario: A customer wants to book a treatment at the beauty salon.

Pre-condition:

* The customer must be logged in to the system.
* Services and availability schedules must be updated in the system.

Steps:

1. The customer logs into the system and navigates to the Book Service section.
2. The system displays a list of available treatments (services) offered by the salon.
3. The customer selects a treatment from the list.
4. The system prompts the customer to select an available date and time for the chosen treatment.
5. The customer selects a preferred date and time from the available options.
6. The system verifies that the selected time slot is available.
7. If the date and time are available, the system proceeds to Confirm Booking.
8. The customer reviews the details of the booking (treatment, date, time, and price) and confirms the booking.
9. The system completes the booking and sends a confirmation message to the customer.
10. The system updates the schedule and service availability.

Post-condition: The service is successfully booked, and the system sends confirmation details to the customer.

A screenshot of a computer game

Description automatically generated

4. Cancel Booking

Scenario: The customer or admin wants to cancel an existing service booking in the Beauty Salon System.

Pre-condition:

* The user (either customer or admin) must be logged into the system.
* The booking must exist and be in a state that allows cancellation (e.g., it hasn't been serviced yet).

Steps:

1. The customer or admin logs into the system and navigates to the Cancel Booking section.
2. The system displays a list of existing bookings for the customer (if customer) or for any user (if admin).
3. The customer or admin selects the booking they wish to cancel.
4. The system prompts the user to confirm the cancellation.
5. The user confirms the cancellation.
6. The system processes the cancellation, updating the booking status to "Canceled."
7. The system sends a cancellation notification to the customer (if initiated by the admin) or to the staff (if initiated by the customer).
8. The system updates the appointment schedule and service availability.
9. The system displays a confirmation message to the user about the successful cancellation.

A black and white screen with white text

Description automatically generatedPost-condition: The booking is successfully canceled, the system updates the schedule, and a notification is sent to the appropriate parties.

5. Manage Schedule

Scenario: The admin or staff wants to manage the service schedule, including checking availability, adding new appointments, or adjusting their working hours.

Pre-condition:

* The user (either admin or staff) must be logged into the system.
* Admin has access to manage the schedules of all staff.
* Staff members can manage their own availability or working hours.

Steps:

1. The Admin or Staff logs into the system and navigates to the Manage Schedule section.
2. The system displays the staff member’s current schedule (if logged in as a staff member) or all schedules (if logged in as an admin).
3. The user can choose to:
   * View their schedule
   * Add a new time slot (for availability)
   * Remove an existing time slot or appointment
4. If the user adds or updates their schedule, the system prompts the user for the date, time, and availability.
5. The user inputs the relevant details.
6. The system verifies that the selected time slots do not overlap with existing bookings.
7. If the time slots are valid, the system updates the schedule in the database.
8. The system notifies the relevant staff members (if managed by the admin) or confirms the changes to the user.
9. The system updates the service availability accordingly.

A group of white rectangular objects

Description automatically generatedPost-condition: The schedule is updated, and availability is reflected in the system for future bookings.

6. Manage Service

Scenario: The admin manages the services offered by the salon, such as adding, editing, or removing services. Customers and staff can only view the available services but cannot modify them.

Pre-condition:

* The admin must be logged in to the system.
* Services must already be defined in the system, and customers and staff must have access to view them.

Steps:

1. The Admin logs into the system and navigates to the Manage Service section.
2. The system displays the list of current services.
3. The admin can choose to:
   * Add a new service
   * Edit an existing service
   * Remove a service
4. If the admin adds or edits a service, the system prompts the admin to input the service name, description, duration, and price.
5. The admin inputs the service details and submits the information.
6. The system verifies the details and updates the service list in the database.
7. The system displays a confirmation of the changes to the admin.
8. Customers and staff can view the updated services via their respective interfaces.

Post-condition: The service list is updated, and the changes are reflected in the system for customers and staff to view.

A screenshot of a computer

Description automatically generated

7. Edit user information

Scenario:

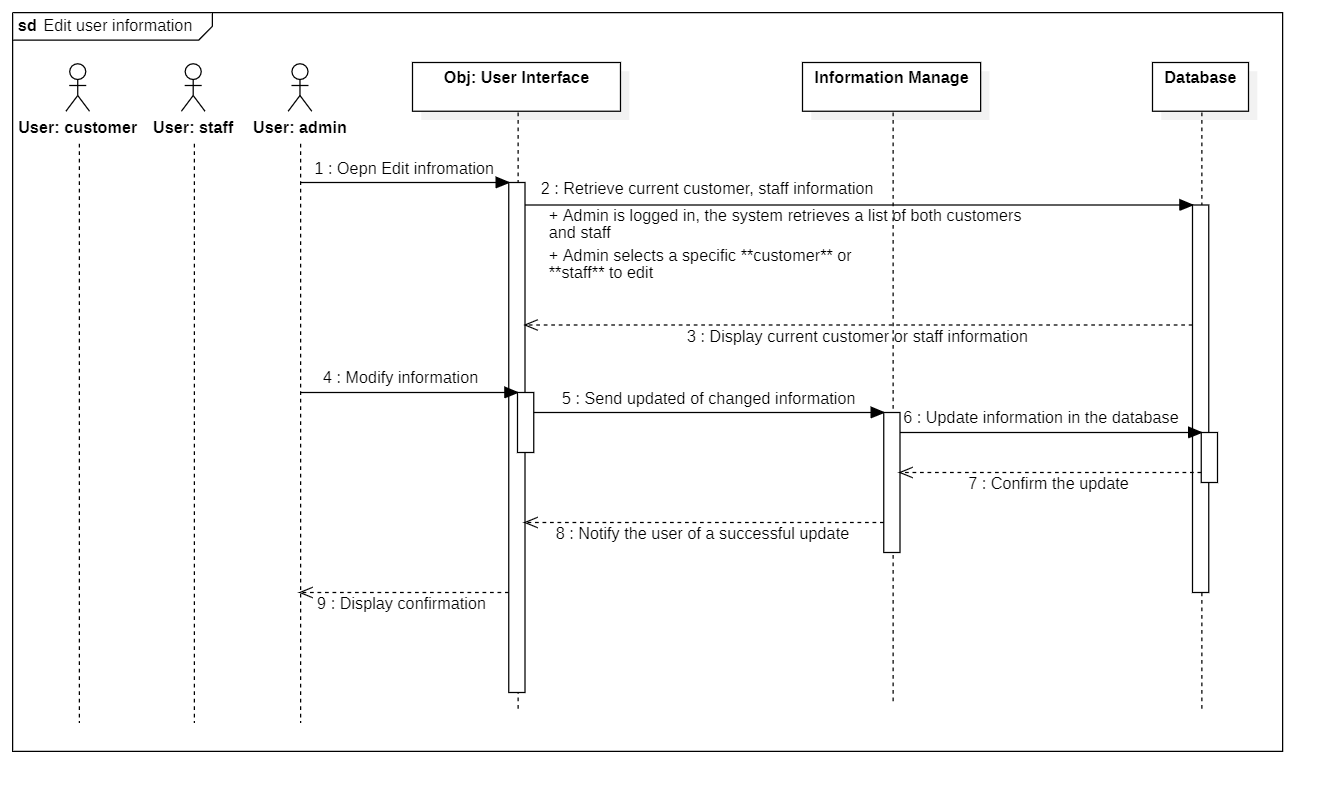
The Admin can update the information of both staff members and customers, while staff and customers can edit their own respective information (e.g., name, contact details, etc.).

Pre-condition:

* The user (either admin, staff, or customer) must be logged into the system.
* The staff or customer must already have an account in the system.

Steps:

1. The Admin, Staff, or Customer logs into the system and navigates to the Edit Information section.
2. If the admin is logged in, the system allows the admin to choose either customer or staff information to edit.
3. The system displays the current information of the selected customer, staff, or logged-in user.
4. The user selects the fields they wish to edit (e.g., name, phone number, email, etc.).
5. The user modifies the information and submits the changes.
6. The system verifies the changes and updates the customer or staff information in the database.
7. The system displays a confirmation message that the information has been successfully updated.

Post-condition: The customer's or staff'

**VII. Detailed Design**

**7.1 Admin Booking Management**

**Class Name**: AdminBookingManage  
**Purpose**: Manages the booking-related functionalities for admin users.

* **Methods**:
  + view\_bookings(): Retrieves and displays all bookings in the system.
  + cancel\_booking(booking\_id): Cancels a booking by its unique ID.

**7.2 Admin Interface**

**Class Name**: AdminInterface  
**Purpose**: Handles navigation and core functionalities for the admin user interface.

* **Methods**:
  + navigate\_to\_services(): Navigates to the service management interface.
  + navigate\_to\_staff(): Navigates to the staff management interface.
  + navigate\_to\_bookings(): Navigates to the booking management interface.
  + logout(): Logs out the admin.

**7.3 Admin Service Management**

**Class Name**: AdminServiceManage  
**Purpose**: Provides tools for managing services in the system.

* **Methods**:
  + add\_service(service\_data): Adds a new service.
  + update\_service(service\_name, new\_data): Updates details of an existing service.
  + delete\_service(service\_name): Deletes a service.

**7.4 Admin Staff Management**

**Class Name**: AdminStaffManage  
**Purpose**: Handles staff-related functionalities for admin users.

* **Methods**:
  + add\_staff(staff\_data): Adds a new staff member.
  + edit\_staff(staff\_id, updated\_data): Updates an existing staff member’s information.
  + delete\_staff(staff\_id): Removes a staff member by ID.

**7.5 Application File**

**Class Name**: App  
**Purpose**: The entry point for initializing the system and launching the application.

* **Methods**:
  + initialize\_system(): Ensures the database is ready for use.
  + launch\_login(): Starts the login interface.

**7.6 Booking Backend**

**Class Name**: BookingBackend  
**Purpose**: Manages the backend logic for customer bookings.

* **Methods**:
  + save\_booking(customer\_name, service, date, time, staff): Saves a new booking.
  + get\_customer\_bookings(customer\_name): Retrieves all bookings for a customer.
  + cancel\_booking(booking\_id): Cancels a booking.

**7.7 Account Creation**

**Class Name**: CreateAccount  
**Purpose**: Handles account creation for new users.

* **Methods**:
  + create\_user(username, password, role): Creates a new user account.

**7.8 Customer Interface**

**Class Name**: CustomerInterface  
**Purpose**: Manages functionalities available to customer users.

* **Methods**:
  + book\_service(): Allows customers to book a service.
  + view\_booking\_history(): Displays the customer’s booking history.
  + logout(): Logs out the customer.

**7.9 Customer Service Interface**

**Class Name**: CustomerServiceInterface  
**Purpose**: Displays and allows selection of services for customers.

* **Methods**:
  + display\_services(services\_list): Shows available services.
  + select\_service(service\_name): Allows selection of a service.

**7.10 Data Management**

**Class Name**: DataManager  
**Purpose**: Handles database-related operations for the system.

* **Methods**:
  + initialize\_data\_file(): Ensures the database file exists and is initialized.
  + read\_all\_rows(sheet\_name): Reads rows from the specified sheet.
  + append\_row(sheet\_name, row\_data): Appends data to a sheet.

**7.11 Login Backend**

**Class Name**: LoginBackend  
**Purpose**: Manages authentication-related operations.

* **Methods**:
  + authenticate\_user(username, password): Verifies user credentials.

**7.12 Login Interface**

**Class Name**: LoginInterface  
**Purpose**: Provides the login interface and navigation to account creation.

* **Methods**:
  + login(): Authenticates a user and grants access.
  + navigate\_to\_create\_account(): Redirects to the account creation interface.

**7.13 Service Backend**

**Class Name**: ServiceBackend  
**Purpose**: Handles backend logic for managing services.

* **Methods**:
  + add\_service(service\_data): Adds a new service.
  + get\_services(): Retrieves all available services.

**7.14 Staff Backend**

**Class Name**: StaffBackend  
**Purpose**: Manages backend operations for staff members.

* **Methods**:
  + add\_staff(staff\_data): Adds a new staff member.
  + update\_availability(staff\_id, availability\_data): Updates a staff member’s availability.

**7.15 Staff Interface**

**Class Name**: StaffInterface  
**Purpose**: Provides staff with tools to manage availability and bookings.

* **Methods**:
  + view\_schedule(): Displays a staff member’s schedule.
  + update\_availability(): Allows staff to update their availability.
  + logout(): Logs out the staff member.

**VIII. Implementation**

In this section, we discuss the details and experiences of implementing the system.

**8.1 Selection of Programming Language and IDE**

* Programming Language: Python
  + Python was chosen due to its simplicity, readability, and extensive library support for GUI (Tkinter) and data handling (OpenPyXL).
  + It is cross-platform and ensures the portability of the system across different operating systems (Windows, macOS, and Linux).
  + Its dynamic typing and large community support make development faster and more efficient.
* Integrated Development Environment: PyCharm
  + PyCharm was selected for its comprehensive debugging tools, code suggestions, and integration with Git.
  + The project structure tools helped maintain a modular approach, making the system easier to develop and manage.

**8.2 Programming Practices**

* Good Practices Applied:
  + Consistent and meaningful variable names were used for improved readability and maintainability of the code.
  + Modularization of the system into separate files for roles (admin, staff, customer) and features (booking, services, login) ensured clarity and scalability.
  + Inline comments and prologue comments were added to explain complex logic and provide descriptions for each class and method.
  + Python's PEP8 style guide was followed for uniform formatting and adherence to industry standards.
* Practices Avoided:
  + Deeply nested or hard-to-read logic was avoided to simplify debugging.
  + Hardcoding values where variables or configuration files could be used.

**8.3 Reuse and Portability**

* Code Reuse:
  + Backend logic (e.g., data\_manager.py, login\_backend.py) can be reused in future projects requiring user authentication or Excel data management.
  + GUI modules (e.g., staff\_interface.py, customer\_interface.py) are reusable for similar service-based systems or booking applications.
* Portability:
  + The system only requires Python and its dependencies, ensuring portability across Windows, macOS, and Linux.
  + The requirements.txt file enables easy installation of dependencies on new machines.
  + A structured folder system ensures the system can be packaged and distributed easily.

**8.4 Version Control System**

* System Used: Git
  + Git was selected for its robustness and wide adoption for version control.
  + The team utilized Git features through both the PyCharm IDE and the command-line interface (CLI).
* Advantages:
  + Collaboration among team members was seamless, allowing simultaneous development of different modules (e.g., admin, staff).
  + The ability to branch and merge helped manage features and bug fixes efficiently.
  + Git's history and diff features helped identify and resolve bugs quickly.
* Disadvantages:
  + Occasionally, merge conflicts occurred, but they were resolved through team communication.
* Most Useful Features:
  + Branching and pull requests enabled collaborative development and code review.
  + Stashing changes provided flexibility in switching between tasks.
* Suggestions for Improvement:
  + Templates for commit messages to ensure consistency.
  + Enhanced visualization tools for branch history.

**IX. Test Cases and Results**

**Unit Testing**

Unit testing focuses on testing individual components of the system. Below is a summary of test cases for unit testing:

| Test Case ID | Test Description | Input | Expected Output | Observed Output | Status |
| --- | --- | --- | --- | --- | --- |
| TC1 | Verify user login functionality | Username: "admin", Password: "password123" | Successful login for the admin role | As expected | Pass |
| TC2 | Add a new service | Service Name: "Facial", Price: 100.0 | Service added to the database | As expected | Pass |
| TC3 | Cancel an existing booking | Booking ID: "BK-1234" | Booking status updated to "Cancelled" | As expected | Pass |
| TC4 | Add availability for staff | Staff ID: "S123", Date: "2024-12-05", Time Slot: "10:00-12:00" | Availability added successfully | As expected | Pass |
| TC5 | Invalid login | Username: "user", Password: "wrongpass" | Login failure with an error message | As expected | Pass |

**Integration Testing**

Integration testing verifies that different components of the system interact as expected.

| Test Case ID | Test Description | Input | Expected Output | Observed Output | Status |
| --- | --- | --- | --- | --- | --- |
| IT1 | Login and navigate to the Admin Interface | Login as admin | Successful navigation to admin dashboard | As expected | Pass |
| IT2 | Admin adds staff and assigns availability | Staff Name: "John", Date: "2024-12-05" | Staff added, availability displayed | As expected | Pass |
| IT3 | Customer books a service | Service: "Haircut", Date: "2024-12-05" | Booking created and displayed in admin and customer dashboards | As expected | Pass |

**Product Testing**

Product testing ensures that the entire application works as a whole.

| Test Case ID | Test Description | Input | Expected Output | Observed Output | Status |
| --- | --- | --- | --- | --- | --- |
| PT1 | Simulate end-to-end booking process | Admin, staff, and customer workflows | Booking made, assigned to staff, and visible in all dashboards | As expected | Pass |
| PT2 | Test customer interface only | Customer login and booking management | Customers can view booking history and book services | As expected | Pass |

**Test Summary**

* **Total Test Cases: 10**
* **Passed Test Cases: 10**
* **Failed Test Cases: 0**
* **Coverage Achieved: Full statement and branch coverage.**

**X. User Documentation**

**Installation Instructions**

1. **Download the Application:**
   * Download the application files as a .zip archive or clone from the repository.
   * Ensure the Python environment is set up with necessary dependencies.
2. **Install Dependencies:**
   * Run the following command to install dependencies:

bash

Copy code

pip install -r requirements.txt

1. **Run the Application:**
   * Navigate to the project directory and run:

bash

Copy code

python app.py

**User Guide**

**Admin Role**

1. Login:
   * Use the admin credentials to log in.
2. Manage Staff:
   * Navigate to the "Staff Management" tab to add, edit, or remove staff.
3. Manage Services:
   * Use the "Service Management" tab to add or update services.
4. Manage Bookings:
   * View all bookings, assign staff, or cancel bookings.
5. Logout:
   * Click the "Logout" button to exit the system.

**Staff Role**

1. Login:
   * Enter staff credentials to log in.
2. Manage Availability:
   * Add or update availability using the calendar interface.
3. View Schedule:
   * Check assigned bookings and manage personal availability.
4. Logout:
   * Use the "Logout" button to return to the login screen.

**Customer Role**

1. Login:
   * Log in using customer credentials.
2. View Services:
   * Browse services and select the ones to book.
3. Book Service:
   * Use the booking form to select a date, time, and optional staff.
4. View Booking History:
   * Access past and current bookings in the booking history tab.
5. Logout:
   * Log out to return to the login screen.

**Screenshots**

1. **Login Interface:**

A screenshot of a login screen

Description automatically generated

1. **Admin Dashboard:**

A screenshot of a computer

Description automatically generated

1. **Staff Interface:**

A screenshot of a computer

Description automatically generated

1. **A screenshot of a computer

   Description automatically generatedCustomer Interface:**

**FAQs**

1. What if I forget my password?
   * Use the admin support email to reset your password.
2. Can I run this on any operating system?
   * Yes, the application is platform-independent as long as Python is installed.
3. How do I report a bug?
   * Submit an issue through the project's GitHub page or contact support.