

Department of Electrical and Computer Engineering
North South University



Fall, 2025

Cse327 Project Proposal Report

Section: 6

Fall, 2025

Project Name:

Home Service Booking Platform

Submitted To: **AKM Iqtidar Newaz [IQN]**

Submitted By:

Group Member's Name	ID
Rafsat wairan	2221602642
Md Kaif Afran khan	2222134642
Umme Umama	2222547042
Kazi Masrafi Munna	2221517042



Home Service Booking Platform

Project Proposal & System Description (Flask + Mongo + Design Patterns)

1. Introduction

The **Home Service Booking Platform** is a web-based application designed to connect customers with trusted local service providers (cleaners, electricians, plumbers, painters, and movers) through a seamless, easy-to-navigate interface.

The project's core purpose is to **digitalize the process of finding and booking home services**, replacing the outdated word-of-mouth and manual scheduling methods with a **secure, responsive, and automated system**.

This system is divided into two parts:

1. **User Application (Frontend + Flask Views):** Where customers can browse services, book appointments, and manage their profiles.
2. **Admin Dashboard (Backend Interface):** Where administrators can manage business data, monitor bookings, and analyze user trends.

To ensure **reusability, scalability, and maintainability**, the backend architecture will be designed following **Software Design Patterns**, mainly **MVC (Model-View-Controller)**, **Factory Pattern**, **Observer Pattern**, and **Singleton Pattern**.

These will structure the system into logical, independent layers that simplify development and future upgrades.

2. Project Objectives

1. To create a **web platform** that allows users to search and book various home services by location, category, or business name.
2. To provide a **secure login and registration system** with role separation (Customer vs. Admin).
3. To integrate a **dynamic scheduling system** that prevents overlapping bookings for the same user and time slot.

4. To implement **CRUD operations** for the admin to manage businesses, users, and bookings efficiently.
 5. To design the backend using **Flask + MongoDB** following **standard design patterns** for better scalability and code organization.
 6. To provide an intuitive and modern **frontend interface** using HTML, CSS, Bootstrap, and JavaScript.
-

3. End Users

1. Customer (Primary User)

- Can create an account, log in, and manage their profile (photo, address, phone number).
- Can browse categories (Cleaning, Plumbing, Painting, etc.), search by area, and book available time slots.
- Can view “Booked” and “Completed” services under “My Bookings.”

2. Admin

- Has a separate login dashboard (different Flask route and frontend interface).
 - Can add, update, or delete businesses.
 - Can view all user accounts, see analytics, and monitor bookings.
 - Uses the same database but interacts via an admin-only interface.
-

4. Functional Requirements

4.1 User Functions

1. **Register/Login** – User authentication via Flask routes with password hashing (using bcrypt or werkzeug).
2. **Profile Management** – Update name, address (street/house, city, district), and profile picture.
3. **Browse and Search** – View categories and filter businesses by type or area.
4. **Business View Page** – See detailed profile of service provider: image, description, gallery, achievements, availability.
5. **Booking System** – Book a service with date/time slot (non-overlapping for same user).
6. **View Bookings** – See active (“Booked”) and past (“Completed”) services.
7. **Notification System** – Toast or message displayed upon successful booking.

4.2 Admin Functions

1. **Admin Login** – Secure route for admin authentication.
2. **Business Management** – Create, edit, or remove business listings.
3. **User Management** – View all registered users with their details.
4. **Analytics** – Simple visual representation of total users, active bookings, and category performance.
5. **Booking Overview** – See all bookings across the system.

5. Non-Functional Requirements

Category	Description
Performance	The system should load pages in under 3 seconds and handle multiple simultaneous booking requests.
Security	Passwords stored as hashes; session-based login; role-based route protection; MongoDB field validation.
Scalability	Modular Flask blueprints and reusable pattern-based classes make adding new categories or user roles simple.
Usability	Frontend will use Bootstrap grid and responsive layout; booking flow remains simple and clear.
Reliability	MongoDB replication ensures data integrity; error handling in Flask prevents crashes.
Maintainability	Follows MVC and other design patterns to separate business logic from routing and database layers.
Compatibility	Works on all major browsers; Flask backend communicates over REST API.

6. Design Patterns and Architecture

a. MVC (Model-View-Controller) Pattern

- **Model:** Manages MongoDB schemas and data access (User, Business, Booking).
- **View:** HTML templates (Jinja2) rendered by Flask for different pages (Home, Services, Bookings).
- **Controller:** Flask route handlers that process user actions and communicate between models and views.

Example:

- `BookingController` handles booking logic and calls `BookingModel` to store data.
 - `BusinessController` fetches all businesses from `BusinessModel` and sends to the Jinja2 view.
-

b. Factory Pattern

Used for dynamically creating objects like `Business`, `Booking`, or `User` without exposing instantiation logic.

Example:

A `BusinessFactory` can generate different business objects (`CleaningBusiness`, `PlumbingBusiness`) depending on the category type.

c. Singleton Pattern

Ensures only one MongoDB connection instance is used throughout the Flask app. This prevents redundant connections and improves performance.

d. Observer Pattern

Used for notifications: when a booking is created, a “notification observer” automatically triggers a confirmation message for the user.

7. System Architecture

Frontend (Client Layer)

- Built with **HTML**, **CSS**, **Bootstrap**, and **JavaScript (Vanilla)**.
- Communicates with Flask backend through routes (`/login`, `/book`, `/services`, `/admin`).
- Responsive design ensures compatibility with desktop and mobile browsers.

Backend (Server Layer)

- **Flask** framework manages routes, authentication, and business logic.
- Modular structure using Blueprints:
 - `auth/` – Login and signup logic
 - `user/` – User dashboard and bookings
 - `admin/` – Admin dashboard routes
 - `business/` – Business management and service listing
 - `booking/` – Booking management and validation

Database (MongoDB)

Collections:

- `users` → `user_id`, `name`, `email`, `password_hashed`, `address`, `profile_pic`
- `businesses` → `business_id`, `name`, `category`, `location`, `description`, `images`
- `bookings` → `booking_id`, `user_id`, `business_id`, `date`, `time_slot`, `status`
- `notifications` → `notification_id`, `user_id`, `message`, `timestamp`, `is_read`

8. Expected Outcome

This project will deliver a **fully functional home service booking web application** with clean separation of roles, secure access, and maintainable architecture using **Python Flask + MongoDB**.

Customers will be able to find and book services easily, while admins will maintain control and analytics.

By applying key **software design patterns**, the system will remain robust, extendable, and easy to scale — providing a real-world example of applying software engineering best practices in a practical, service-oriented system.

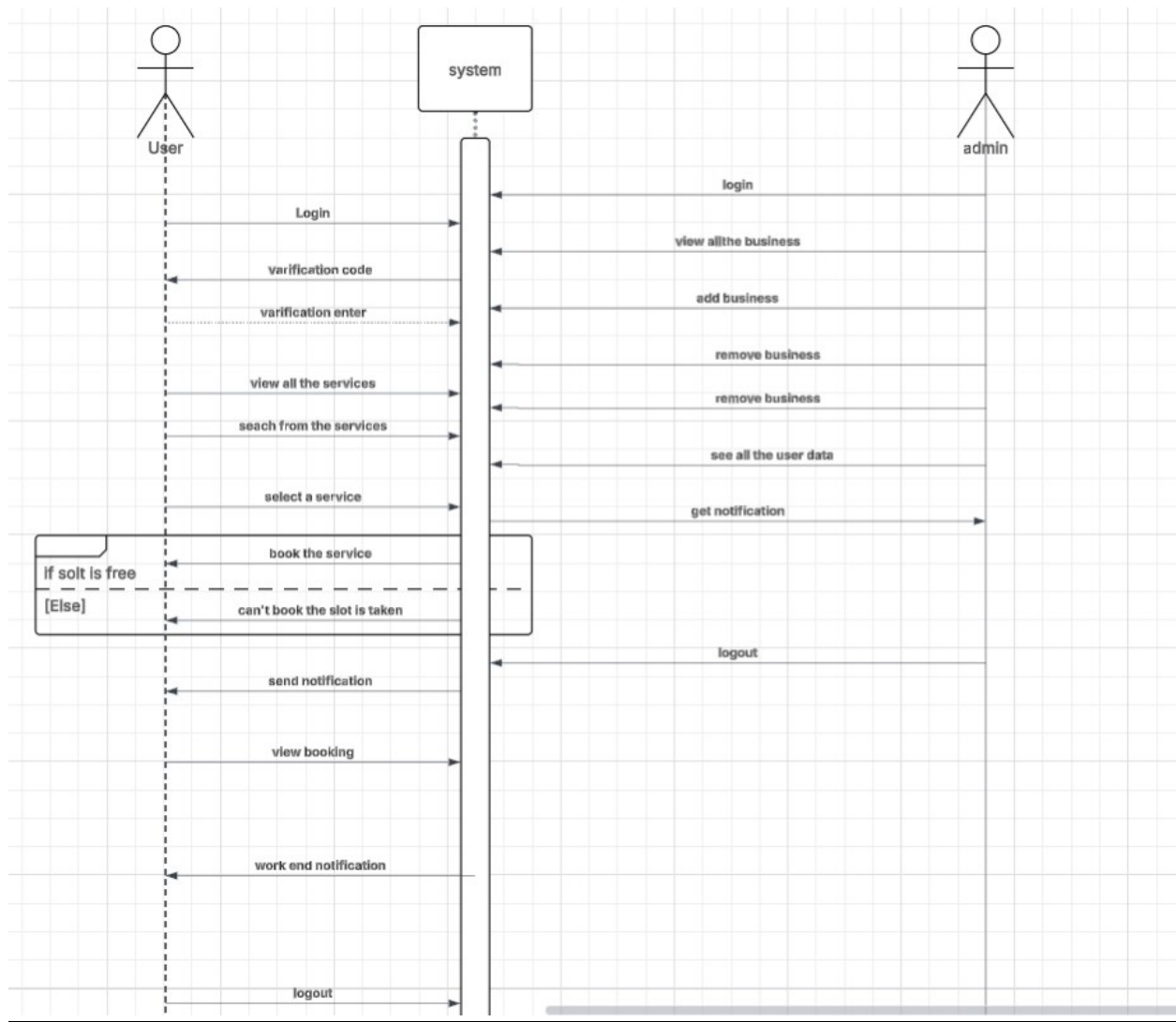
GitHub Repository Link:

https://github.com/wairan/service-provider_project/tree/main

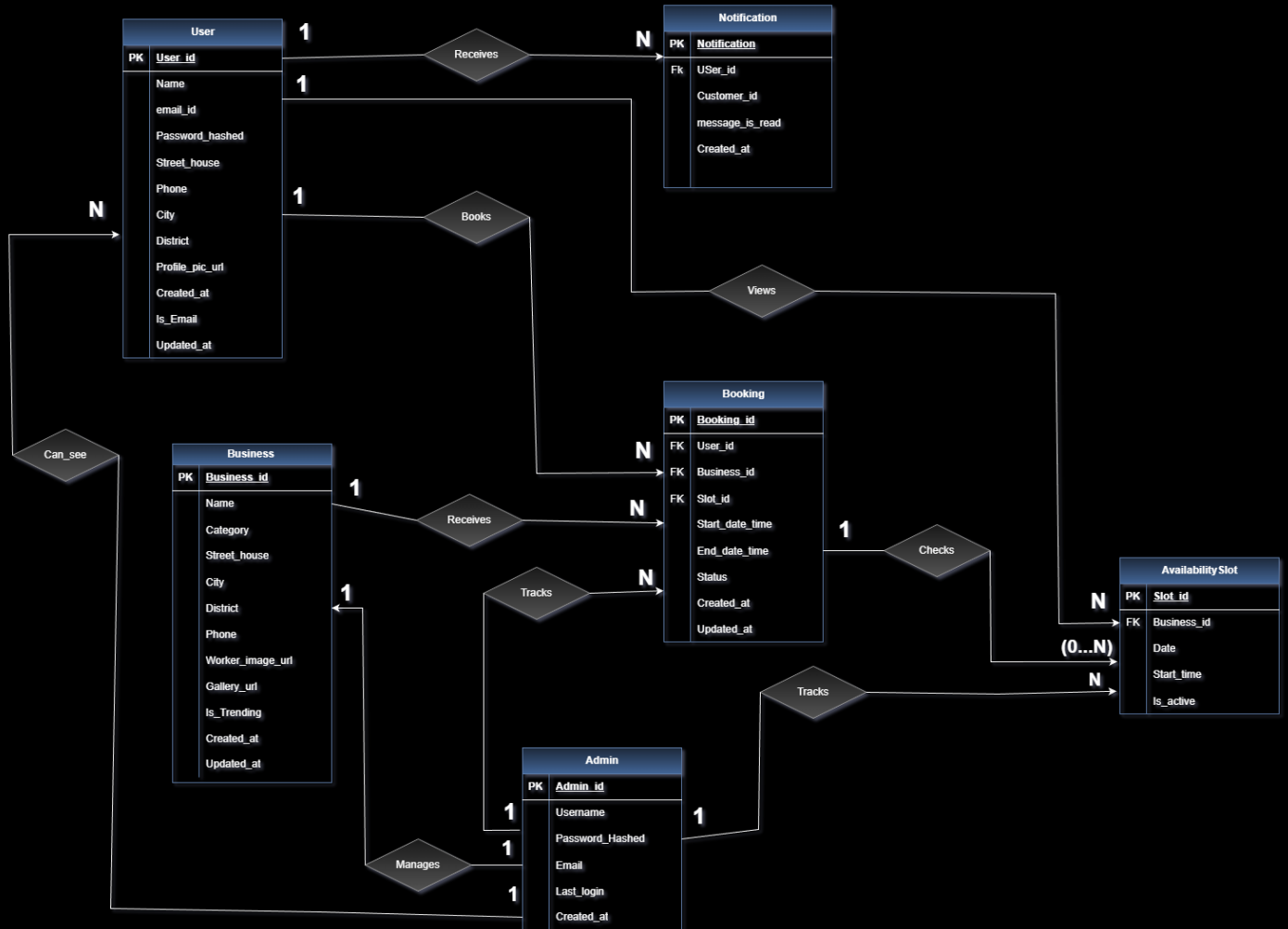
Class Diagram:



Sequence diagram:



Entity relationship diagram (ER):



Use Case Diagram:

