Tribhuvan University

Faculty of Humanities and Social Sciences

A PROJECT REPORT

On

"Room Cancellation Prediction using Random Forest Classification"

Submitted to

Department of Computer Application

National College of Computer Studies

In partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted By:

Yuwa Shrestha

6-2-551-82-2022

Under the Supervision of

Yuba Raj Devkota

Tribhuvan University

Faculty of Humanities and Social Sciences

National College of Computer Studies

Supervisor's Recommendation

I hereby recommend that this project prepared under my supervision by Yuwa Shrestha entitled "HamroGhar" in partial fulfillment of the requirements for the degree of Bachelor of Computer Application is recommended for the final evaluation.

Signature of the Supervisor

Yuba Raj Devkota
Faculty Member
Department of Computer Application
National College of Computer Studies
Paknajol, Kathmandu

Tribhuvan University

Faculty of Humanities and Social Sciences

National College of Computer Studies

LETTER OF APPROVAL

This is to certify that this project prepared by Yuwa Shrestha entitled "HamroGhar" in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

SIGNATURE of Supervisor	**SIGNATURE of HOD/ Coordinator**
Yuba Raj Devkota Faculty Member National College of Computer Studies Paknajol, Kathmandu	Rajan Poudel Academic Coordinator National College of Computer Studies Paknajol, Kathmandu
SIGNATURE of Internal Examiner	**SIGNATURE of External Examiner**

ABSTRACT

A comprehensive hotel booking system, HamroGhar is made to offer a customized and user-focused lodging experience. The platform gives users the ability to view room availability, book rooms that suit their individual preferences, and view their entire booking history. The flexible payment deposit option is a significant innovation that benefits users. The system includes a Random Forest Classifier-powered predictive analytics engine for administrators. To precisely predict cancellations, this model examines important booking characteristics such as lead time, check-in and check-out dates, guest count, and other variables. Proactive revenue management, better overbooking tactics, and less loss from last-minute cancellations are made possible by this.

ACKNOWLEDGEMENT

This Report entitled “HamroGhar” is prepared in the partial fulfillment of the requirements of Bachelor in Computer Application. It is the result of cooperation and support of our supervisor and teacher.

I would like to give thanks to the Mr. Sudil Maharjan for providing this wonderful opportunity to provide a guideline for making this project. I would also like to express my sincere gratitude to my supervisor, Mr. Yuba Raj Devkota, for providing suggestions for preparing this report.

Yuwa Shrestha

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CSS	-	Cascading Style Sheets
HTML	-	Hypertext Markup Language
GIF	-	Graphics Interchange Format
JPG	-	Joint Photographic Experts Group
JSON	-	JavaScript Object Notation
OTA	-	Online Travel Agency
PMS	-	Property Management System
PNG	-	Portable Network Graphics
RFC	-	Random Forest Classifier
UI	-	User Interface
XGBOOST	-	eXtreme Gradient Boosting

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Introduction

HamroGhar is a hotel booking application that helps users easily reserve accommodations that fit their needs. It has a simple and user-friendly interface, making it easy for users to browse available rooms, book their stays, and manage their reservations. The system was developed using the iterative model, employing Python for machine learning processes and using Flask and sqlite3 for the frontend and backend. The platform includes important features like flexible payment options and tracking booking history. A notable aspect of HamroGhar is its cancellation prediction system. It uses machine learning algorithms, especially the Random Forest Classifier, to predict possible booking cancellations. This feature helps hotel administrators maximize revenue and keep occupancy rates at optimal levels.

1.2 Problem Statement

In the pre-digital days, it was a must to go to the hotel in person or call the front desk to ask for the room availability and to reserve a room, which was a laborious and unproductive process. The upside was that there were no instant confirmations, and the travelers had to put up with waiting for the replies, often losing the opportunity of preferred lodging in case of urgency. The situation changed a bit with the launch of online booking sites, but still the issue of last-minute cancellations remained as hotels experienced unpredictable booking patterns which could turn into huge revenue loss and operational challenges for them.

1.3 Objectives

The main objectives of this system are:

- To develop a hotel booking system that allow users to book rooms with flexible payment options.
- To implement a cancellation prediction system that will determine the likelihood of booking cancellations.

1.4 Scope and Limitations

Scope

- Room Discovery and Booking: Allows users to view available rooms in detail and make reservation as per their preference.
- Cancellation prediction: Allows admin to get room cancellation risk assessment using random forest machine learning algorithm
- Booking Management: Allows users to view and cancel their current and past bookings.

Limitations

- Desktop focused design: This system is mainly focused for desktop users.
- No real time room availability updates: This system may not reflect instant room availability changes during manual refresh.

1.5 Development Methodology

The iterative model is used for the development of the system. The specification of this project will be modified as per the guidelines, and there might be the addition of new features, so it is most reasonable to use the iterative model for the development.

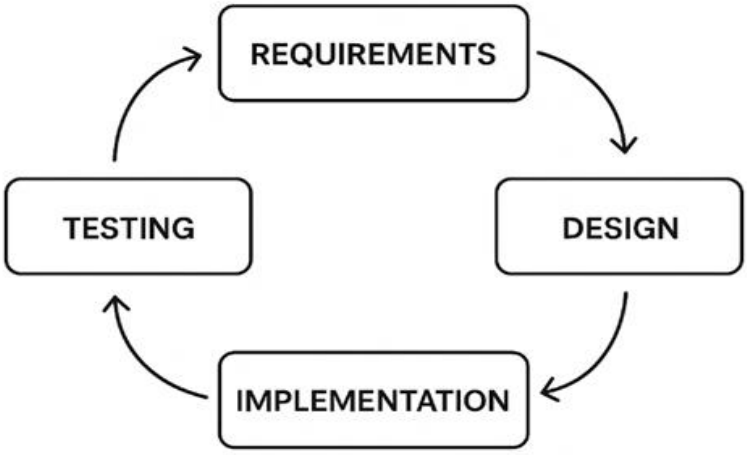


Figure 1: Iterative Model for HamroGhar

The Iterative model is capable to handle changes or modifications in requirements later on without affecting the quality of performance. The different phases of the iterative model can be revisited as necessary.

1.6 Report Organization

Introduction

The first chapter reveals the idea behind this project. It outlines the difficulties that have been around and how its goal can effectively address. Additionally, it shares the project's extent and constraints together with the development process.

Background study and literature review

This chapter will concentrate on the fundamentals of how this project will be constructed. It outlines the examination of various platforms and their mechanisms as well as past literatures reviewed.

System analysis and design

This chapter studies about the requirements gathering, feasibility study and designing of the project. It contains activities, use case, functional analysis and requirement gathering.

Implementation and testing

This chapter is gathered to provide details about the project implementation, software and tools used and the testing.

Conclusion and future recommendation

This chapter includes the possible outcome of this project, conclusion and future recommendations

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

Booking a hotel room involves selecting accommodations based on preferences like room type, budget, and dates, which can be challenging due to availability constraints and complex reservation processes. HamroGhar addresses this through a practical booking system that incorporates flexible deposit options including online and offline bookings to accommodate different user needs. The platform integrates secure online payment processing, allowing users to complete transactions safely. The system's standout feature is a machine learning-based cancellation prediction algorithm that assesses booking risks using Random Forest classification, analyzing factors like lead time and deposit type. Developed following the iterative model, the application uses HTML and CSS for frontend interface, Flask for backend services, SQLite for data storage, and Python's pandas for building predictive model. These elements combine to create a reliable hotel booking platform with financial flexibility and risk management capabilities.

2.2 Study of Existing system

Many hotel booking systems already exist, such as Booking.com, Agoda, MakeMyTrip, and traditional Property Management Systems (PMS) like Opera and Fidelio. However, these platforms have fixed structures and do not meet the specific needs of individual hotels. Aggregators like Booking.com and Agoda focus on listing many properties and charge commission fees, which limits control for hotels. Traditional PMS tools like Opera offer many features but are expensive and lack modern machine-learning analytics. Systems like OYO restrict branding freedom, while Airbnb is mainly designed for home-style rentals and not professional hotels. The system being studied aims to address these issues by providing integrated payment handling, machine-learning-based cancellation prediction, and full operational control, all without commission fees. It helps hotels manage bookings efficiently, understand customer patterns, and reduce revenue loss from cancellations.

2.3 Literature Review

Predicting hotel booking cancellations has become an active research and applied area within hospitality analytics because cancellations materially affect revenue management, staffing, and inventory decisions. Early applied studies showed that cancellations are predictable to a useful degree when historical booking records and behavioral indicators are available; Antonio et al.'s Hotel Booking Demand datasets provide a commonly used public benchmark for many of these studies and have enabled reproducible comparisons across approaches. [1]

Research consistently identifies a core set of predictors that capture booking characteristics, customer commitment, temporal demand patterns, and historical behavior. Typical predictors include lead time (days between booking and arrival), deposit type (no deposit / refundable / non-refundable), market segment (OTA, direct, corporate), number of adults/children, length of stay, previous cancellations or previous bookings, special requests, and booking channel/agent. Studies using the Hotel Booking Demand data and other PMS (property management system) extracts report that deposit type and previous cancellation history are frequently among the top features by importance, because they proxy the customer's financial commitment and reliability. [2]

Empirical analyses reveal robust, often non-linear relationships between key features and cancellation probability. Lead time is one of the most reliably predictive variables: cancellation risk tends to rise with increasing lead time, with multiple studies reporting accelerated risk beyond a threshold in the 20–40-day range (industry analyses and academic papers both observe this pattern). This likely reflects heterogeneity in booking types, leisure travelers who book further ahead may be more likely to change plans than business travelers booking closer to arrival. Seasonality and group size also modulate deposit effects: for example, "no deposit" bookings have substantially higher cancellation probabilities in some seasons or for larger group bookings. [3]

For tabular hotel booking data, tree-based ensembles (Random Forest, XGBoost/Gradient Boosting Machines) repeatedly achieve strong empirical performance and are standard baselines in the literature. Random Forests are frequently favored for operational systems because they handle mixed categorical and numerical variables without heavy preprocessing, are robust to noisy or missing values, reduce variance via bagging, and provide interpretable feature-importance measures that are easily communicated to hotel

managers. Recent comparative studies and applied papers show that while sophisticated boosting (e.g., XGBoost) can offer incremental gains, Random Forests deliver a strong balance of accuracy, stability, and interpretability for many hotel datasets, which is why many practitioners adopt them as a pragmatic first choice. [4]

Given the dataset type (tabular booking records), the need for interpretability, and common data-quality issues in hotel PMS exports, Random Forests are a defensible and well-supported baseline for HamroGhar. They provide robust predictive performance, straightforward feature-importance outputs, and are easy to integrate into operational pipelines. Nonetheless, the project should implement best practices from the literature: stratified/temporal validation, imbalance correction, and explain ability for operational activities. [5] Also, Random Forest does better than linear regression for prediction task. Linear regression makes the assumption of linearity. This assumption makes the model easy to interpret but is often not flexible enough for prediction. Random decision forests easily adapt to nonlinearities found in the data and therefore tend to predict better than linear regression. More specifically, ensemble learning algorithms like random forests are well suited for medium to large datasets. [6]

Random Forests are fast to build and even faster to predict. They don’t require any cross-validation or fully parallelizable. Random Forest algorithms are often more accurate than a single classifier. It can handle the data without preprocessing, which means data doesn´t need be rescaled or transformed. However, as a widely used algorithm, it is worthy of additional study on improving classification accuracy. [7]

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

3.1.1 Requirement Analysis

The process of exploring the needs and expectations of a new product is known as requirements analysis.

i. Functional Requirements

The system will be able to handle rooms booked by various users and allow them to perform different actions such as viewing available rooms, making bookings and canceling reservations.

Figure 2: User Case Diagram for HamroGhar

1. Admin:

- Login and logout: The admin can login to the system and logout.
- View Users: The admin can view the users registered in the system.
- Add Rooms: The admin can add rooms to the system.
- Room Cancellation Prediction: The system includes a predictive model that helps admin to estimate the likelihood of room cancellations based on user behavior and booking details.
- View Bookings: The admin can view the rooms booked by the user.
- Delete Rooms: The admin can delete rooms in the system.
- Manage Room: The admin will be able to manage room categories, meal plans, rooms and market segments.

2. User:

- Login and Logout: The user can login to the system and logout.
- Register: The user can register into the system.
- View Available Rooms: The users can view the rooms available in the system.
- Book Rooms: The user can book rooms as per their comfort.
- Cancel Booking: The user can cancel the room that they booked.
- Update User Information: The user can update their information as per their requirement and needs.
- View Previous Booking History: The user can view their past booking history.

ii. Non-Functional Requirements

The non-functional requirements of the system include the following:

- Responsiveness: The system loads and responds to user actions within 3 seconds under normal operating conditions.
- User-Friendly: The user interface is simple, intuitive, and user-friendly, enabling users to easily navigate and perform actions like booking, canceling, or viewing room.
- Availability: The system is available 24/7, that means it allow users to make or manage bookings anytime.

3.1.2 Feasibility Study

This project is fully feasible to be developed because the following feasibilities can be achieved easily.

Technical Feasibility

The system's technological feasibility arises from the easily available requirements for its development. There is hardware and software available to support the system's development and implementation. This system meets the technical feasibility as it will be using existing technologies like Flask, SQLite, HTML, CSS and JavaScript.

Operational Feasibility

The system is operationally feasible because It’s simple to use if users have a computer network or internet access. No operative manpower is necessary. Hence, the system is viable.

Economic Feasibility

Because all of the necessary tools and resources are available for free, the system is both economical and effective. The system is therefore financially viable.

3.1.3 Data Modelling

The following ER diagram represents the overall structure of HamroGhar. It shows the overall interaction of user interacting with the system. The system consists of six entities (customers, bookings, meal_plans, room_types, rooms and market segments). The customer entity contains user data such as customer_id, name, email, password, etc. and other entities like rooms, room_types, meal_plans and market_segments contain valuable room details which allow the flexibility in changing individual elements of the room services. The Booking entity mainly holds booking data. It consists of various attributes and 3 main composite attributes. They are booking_data, arrival_info and stay_info. Booking_data contains attributes that are related to booking, whereas stay_info is related to number of guests, total weekend nights and total week nights. The arrival_year is related to the arrival dates.

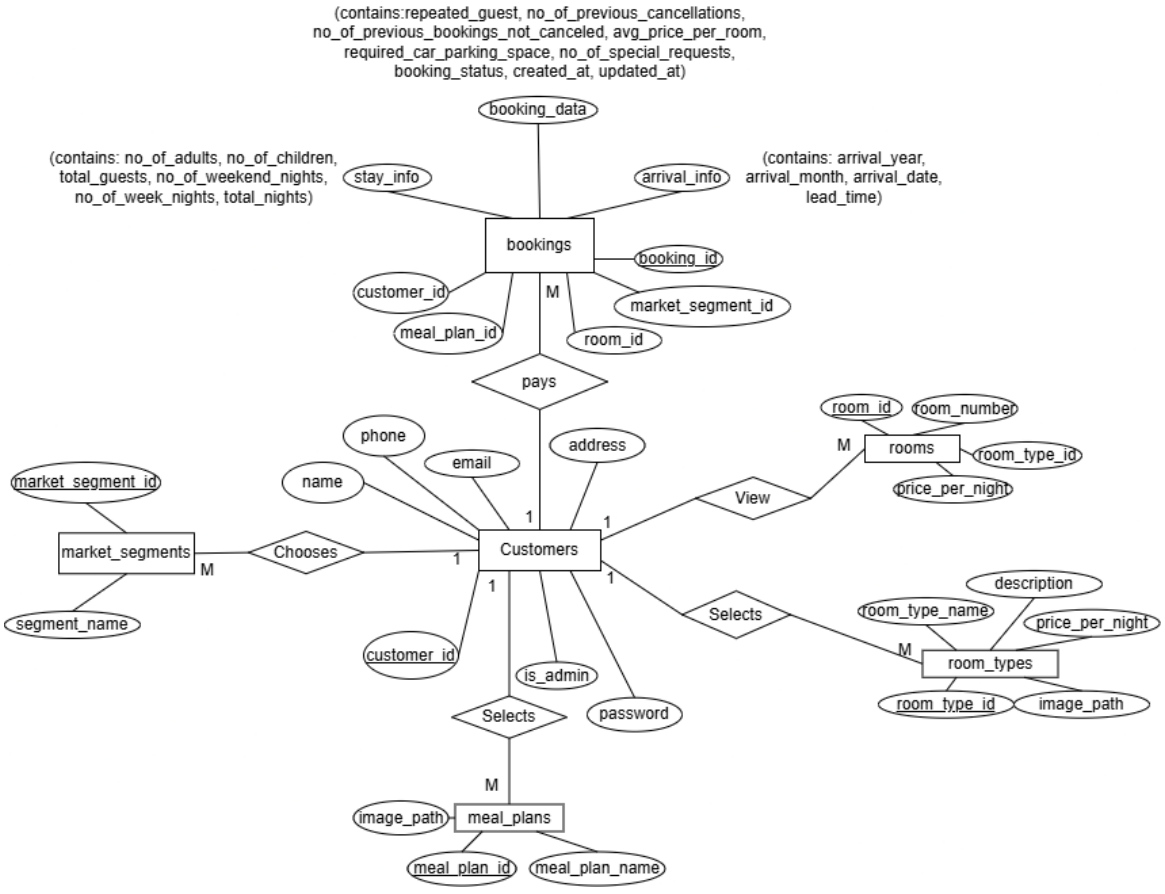


Figure 3: Er Diagram of HamroGhar

3.1.4 Process Modelling

Level 0 DFD

The diagram shows how the Hotel Booking System interacts with both the User and the Admin. The User initiates room bookings and receives information such as available rooms and their own booked rooms. The admin manages the system by creating rooms, room categories, meal plans, and market segments. In return, the system provides the admin with details like booked rooms, total users, and cancellation prediction results.

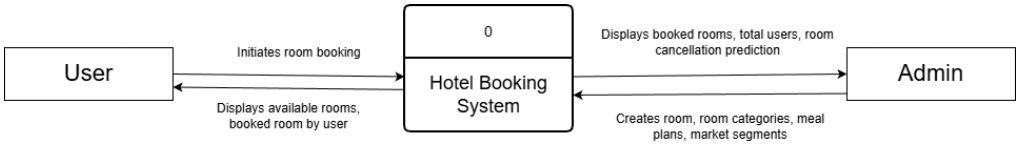


Figure 4:Level 0 DFD of HamroGhar

Level 1 DFD

The level 1 DFD describes the processes in the system in-depth. There are different system processes such as user management system, that deals with user authentication, Payment Management System, that deals with payment and booking, etc.

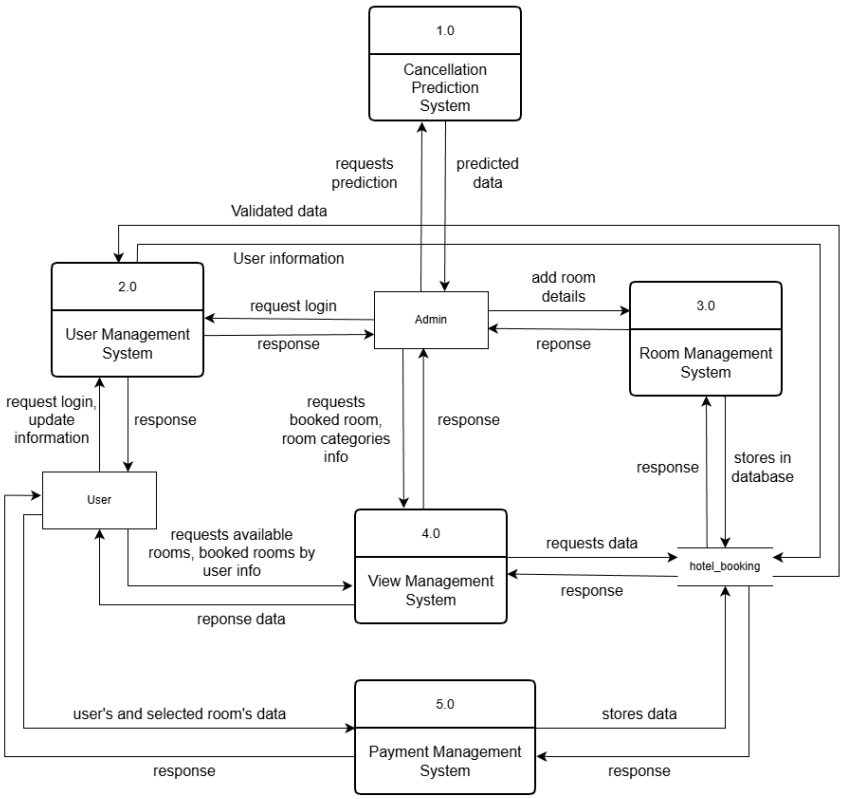


Figure 5: Level 1 DFD of HamroGhar

3.2 System Design

3.2.1 Architectural Design

For HamroGhar, client server architectural design model is used. Client server architecture is the most suitable architectural design because it cleanly separates responsibilities and ensures smooth, scalable interactions between users, admins, and the central system. In this setup, the client is responsible only for the things people interact with, like pages, buttons, forms, and displaying results. All the complicated work happens on the server side. The server safely stores the data, handles room bookings, manages room details, processes payments, and even runs the machine-learning model for cancellation prediction. This way, the client stays simple and fast, while the server does the heavy lifting behind the scenes.

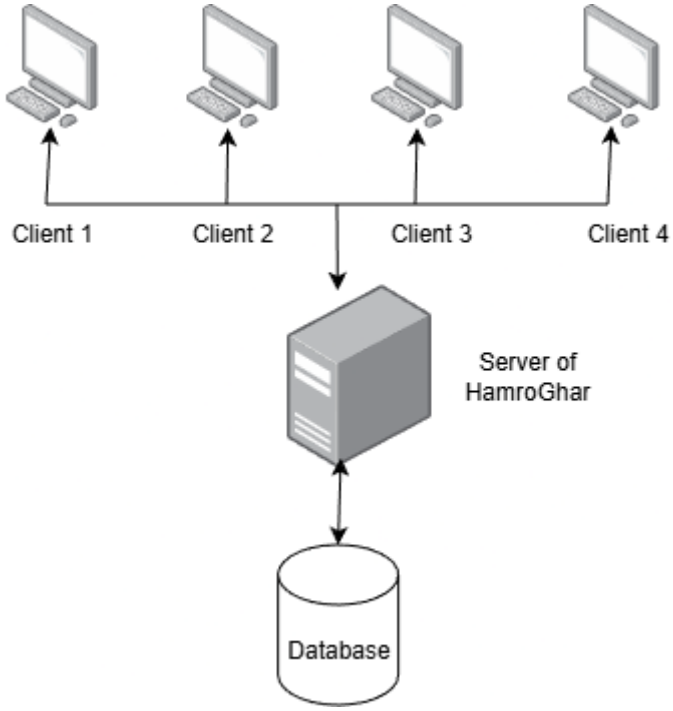


Figure 6: Architectural Design

3.2.2 Database Schema Design

The following diagram represents the database schema design of HamroGhar. It describes the overall structure of the tables of the system. Booking_id, Customer_id, room_id, meal_plan_id, market_segment_id and room_type_id are the major primary keys.

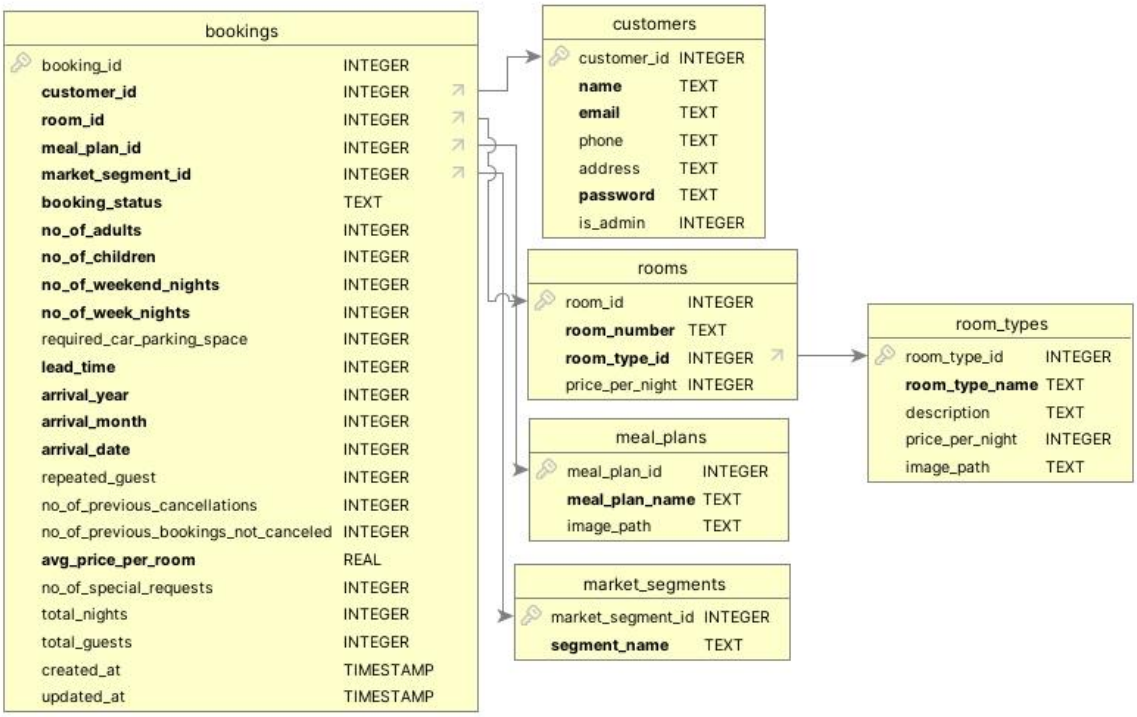


Figure 7: Database Schema for HamroGhar

3.2.3 Interface Design

The interface design is used to show how the overall design of the system is done.

Landing Page

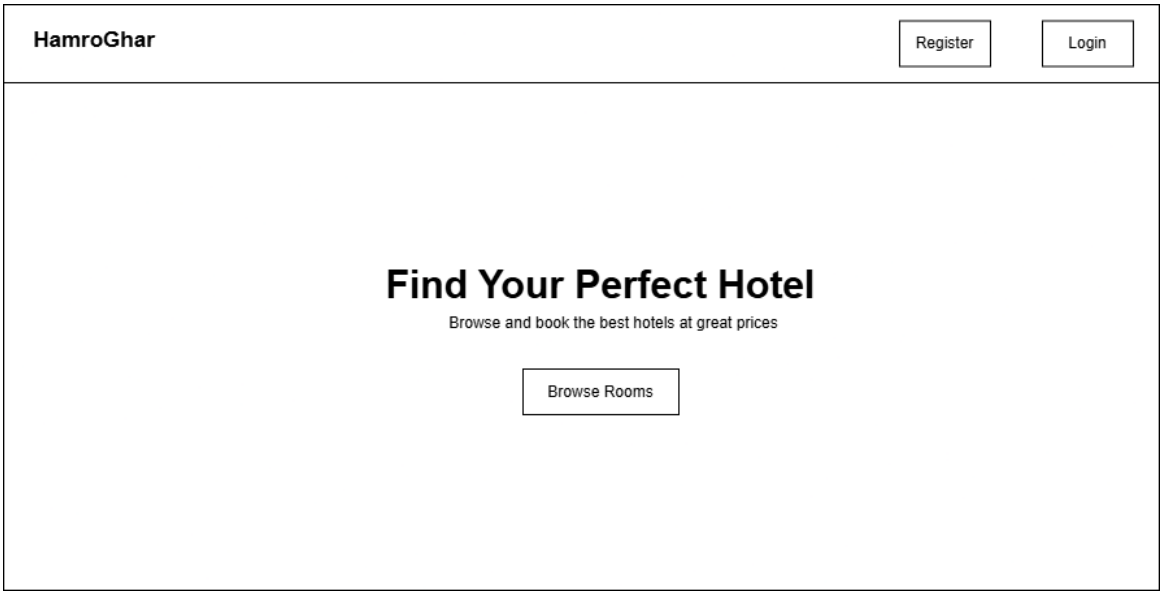


Figure 8: UI for Landing Page

Login Page

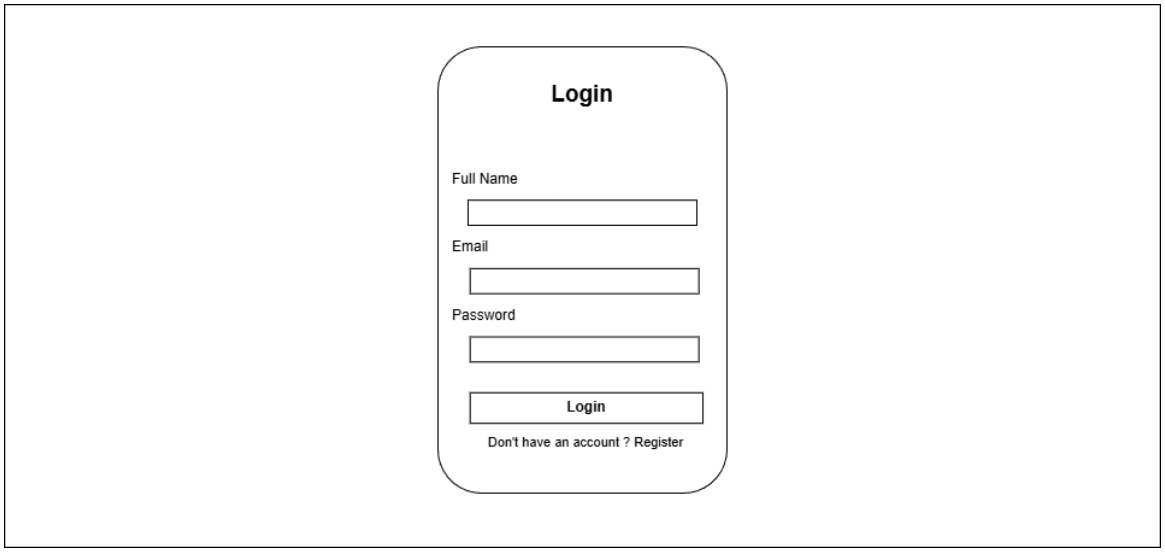


Figure 9:UI for login Page

User Dashboard

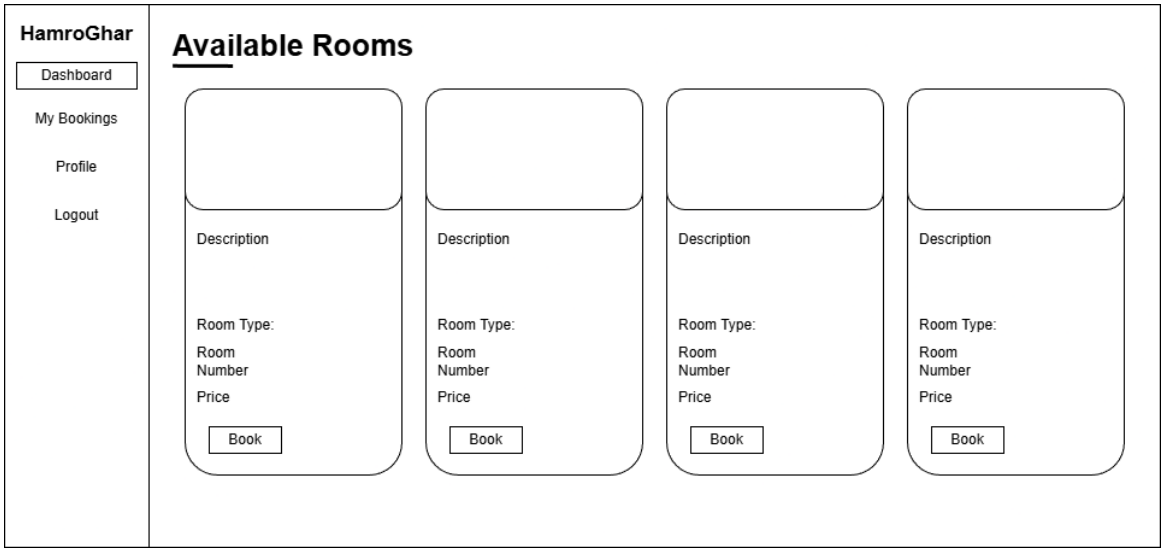


Figure 10: UI for User Dashboard

3.2.4 Physical DFD

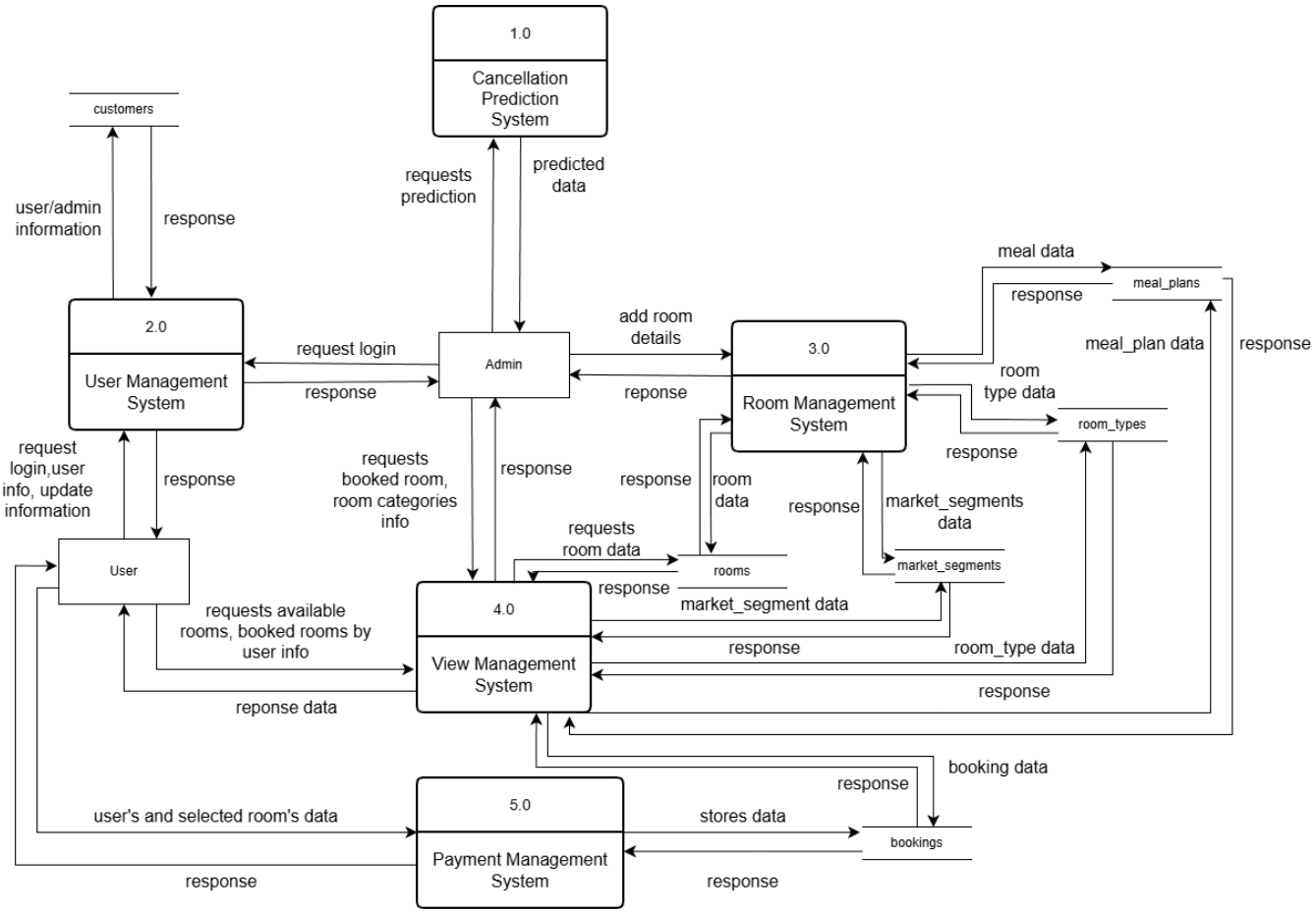


Figure 11: Physical DFD of HamroGhar

3.3 Algorithm Details

a. Random Forest Classifier

In this project Random Forest Classifier is used which is a powerful supervised machine learning algorithm, in HamroGhar hotel booking system to predict the probability of booking cancellations. In order to increase accuracy and avoid overfitting, Random Forest builds numerous decision trees during training and combines their predictions. Because it can handle various data types (numerical and categorical), is resistant to noise, and provides trustworthy probability scores, it is perfect for my project.

Algorithm Overview

- **Type:** Supervised Machine Learning

- **Description:** The ML model predicts the cancellation probability of a booking by analyzing historical patterns and real-time booking inputs. It uses a Random Forest ensemble to classify risk and output a probability score helping administrators make informed decision about booking management.

How Random Forest Works

In this project, Random Forest Classifier is used, which is a powerful supervised machine learning algorithm, in the hotel booking system to predict the probability of booking cancellations. In order to increase accuracy and avoid overfitting, Random Forest builds numerous decision trees during training and combines their predictions. Because it can handle various data types (numerical and categorical), is resistant to noise, and provides trustworthy probability scores, it is perfect for this project. Each tree is built using:

- **Bootstrap Aggregating (Bagging):** Each tree is trained on a random sample of the training data (with replacement)
- **Feature Randomness:** At each split in a tree, a random subset of features is considered
- **Majority Voting:** Final prediction is based on averaging probabilities across all trees

Important Features Used in Prediction:

The model uses 19 features extracted from booking data.

1. Temporal Features:

- Lead time (days between booking date and arrival date)
- Arrival year, month, and date
- Number of weekend nights
- Number of week nights
- Total nights (computed)

2. Guest Information:

- Number of adults
- Number of children
- Total guests (computed)

- Repeated guest flag (0 or 1)

3. Historical Behavior:

- Number of previous cancellations
- Number of previous bookings not canceled

4. Booking Characteristics:

- Average price per room
- Number of special requests
- Required car parking space (0 or 1)

5. Encoded Categorical Features:

- Type of meal plan (encoded: 0-3)
- Room type reserved (encoded: 0-6)
- Market segment type (encoded: 0-4)

Usage in Project:

When an admin views the bookings dashboard, the system automatically

- Fetches all booking records from the database
- Extracts and prepares the 19 features for each booking
- Applies encoding to categorical features
- Creates a pandas dataframe with features in correct order
- Passes the feature vector to the Random Forest model
- Receives probability scores from the model
- Classifies bookings into risk levels

Mathematical Equation:

1. **Cancellation Probability Calculation:**

   The random forest aggregates predictions from all trees using ensemble averaging:

   Cancellation Probability: $P(\mathrm{y}=1|\mathrm{X})=(1/\mathrm{T})\sum(\mathrm{t}=1)^{\mathrm{T}}\mathrm{P}_{\mathrm{t}}(\mathrm{y}=1|\mathrm{X})$

   Where:

- T → number of trees in Random Forest
- Pt(y=1|X) → probability from tree t predicts cancellation (class 1) given feature vector X
- X → feature vector (scaled & encoded)

2. **Risk Classification:**

Based on the calculated probability, bookings are classified into three risk levels.

$$\text{RiskLevel} = \text{cases}\begin{pmatrix} \text{HIGH if P} > 0.7; \\ \text{MEDIUM if } 0.4 < \text{P} \leq 0.7; \\ \text{LOW if P} < 0.4 \end{pmatrix}$$

- High: more than 70% probability of cancellation
- Medium: around 40-70% probability of cancellation
- Low: less than 40% probability of cancellation

3. **Recommended Action:**

The system generates recommendations based on multiple factors:

Recommendation = arg max (Risk(P) + LeadTimeFactor + SpecialRequestFactor+ HistoricalBehavior)

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Implementation

4.1.1 Tools Used

Various frontend, backend, and designing tools have been utilized in the development of the HamroGhar Hotel Room Booking System to ensure a responsive, secure, and well-documented application. The detailed usage of each tool is described below:

- Frontend Tools
    - CSS: Used to style and design the user interface across all web pages.
    - JavaScript: Used to perform client-side validations.
- Backend Tools
    - Flask: Used as the core backend engine for routing, logic, and security.
    - SQLite: Used to keep record of data in database.
- Version Control Tools
    - Git & GitHub – for project collaboration and version management.
- Designing Tools
    - Draw.io: Used to design various diagrams for implementation.

4.1.2 Implementation details of modules

Module 1: User Authentication & Session Management

Description: This module handles user registration, login, logout, and session management using Flask's session mechanism. It implements secure password storage with PBKDF2-SHA256 hashing via Werkzeug's generate_password_hash () and check_password_hash () functions. The system supports both regular users and administrators with role-based access control. Session data includes user_id, is_admin status, and name attributes, persisting across requests to maintain authenticated state.

Usage: This module is used in the /login, /register, and /logout routes to authenticate users, verify credentials using password hashing, and manage user sessions throughout the application.

Module 2: Database Models & Data Storage

Description: This module defines the complete relational database schema using SQLite with explicit table creation SQL statements. It includes six interconnected tables: customers (user accounts), room_types (room categories), rooms (individual room instances), meal_plans (dining options), market_segments (booking sources), and bookings (reservation records). The schema enforces data integrity through UNIQUE constraints (email, room_number), CHECK constraints (booking_status values), and referential integrity via foreign key relationships with ON DELETE CASCADE.

Usage: Used across all application routes for data persistence. When users register, their information is stored in the customers table. When admins create rooms, data goes into room_types and rooms tables. When bookings are made, a comprehensive record is created in the bookings table linking to customers, rooms, meal_plans, and market_segments.

Module 3: File Upload & Image Management

Description: This module handles secure file uploads for room images and meal plan images. It validates file extensions (png, jpg, jpeg, gif), sanitizes filenames using Werkzeug's secure_filename () to prevent directory traversal attacks, and saves files to designated upload directories (static/uploads/rooms and static/uploads/menu_plans). The module includes the allowed_file () helper function for validation and automatically creates necessary directories if they don't exist. Uploaded file paths are stored in the database for later retrieval and display.

Usage: Used in /admin/room_types and /admin/manage_meal_plans routes when administrators upload images for room types or meal plans. The saved filenames are referenced in HTML templates to display images on the landing page, room details pages, and admin management interfaces.

Module 4: Room Inventory Management

Description: This module manages the hotel's room inventory including room types and individual room instances. It handles creation of room categories (Standard, Deluxe, Executive, etc.) with descriptions, prices, and images, as well as individual room assignments with room numbers and pricing. The module includes validation to prevent duplicate room numbers and ensures price consistency between room types and individual rooms. It also provides comprehensive room listing with join queries to display room details along with their type information.

Usage: Used in /admin/room_types and /admin/rooms routes for administrators to add, view, and manage room inventory. Also used in /user_dashboard and / (landing page) routes to display available rooms to users with images, descriptions, and pricing information for browsing and selection.

Module 5: Booking Creation & Reservation System

Description: This module implements the core booking functionality with comprehensive date validation and availability checking. It processes booking requests by calculating stay duration from check-in/check-out dates, verifying room availability through the is_room_available () function that checks for date overlaps with existing "Not_Canceled" bookings, and computing derived features like total nights and total guests. The module handles both online payments (via Khalti integration) and offline "pay at reception" bookings. It automatically calculates user booking history statistics (repeated guest status, previous cancellations, successful bookings) for prediction features.

Usage: Used in the /book_room/<int: room_id> route where users submit booking forms. The system validates dates, checks availability, processes payments (if online), and creates booking records in the database with all required features for both operational use and cancellation prediction.

Module 6: Booking Cancellation & Status Management

Description: This module manages booking cancellations and status tracking. It allows users to cancel their own future bookings while preventing cancellation of bookings they don't own. The module updates the booking_status field from "Not_Canceled" to "Canceled" and records the update timestamp. It includes authorization checks to ensure users can only cancel their own bookings and provides appropriate feedback messages. Cancelled bookings remain in the database for historical analysis and prediction feature calculation.

Usage: Used in the /cancel_booking/<int: booking_id> POST route where users request to cancel reservations. Also integrated into the /my_bookings route to display cancellation options alongside booking details, and in availability checking to exclude cancelled bookings from room availability calculations.

Module 7: Booking Analytics & Admin Dashboard

Description: This module provides administrators with a comprehensive overview of system metrics and recent activity. It calculates key performance indicators including total bookings count, available rooms count, total meal plans, and total registered users. It displays the five most recent bookings with detailed information (customer name, room details, meal plan, segment, status) and lists all room types for quick reference. The dashboard serves as the central monitoring interface for hotel operations.

Usage: Used in the /admin_dashboard route, accessible only to administrators. The module queries multiple database tables to compile statistics and recent activity, presenting them in an organized dashboard view that helps admins monitor hotel performance at a glance.

Module 8: Machine Learning Cancellation Risk Prediction

Description: This module implements a machine learning-based cancellation prediction system using a pre-trained Random Forest model. It loads the model, encoders, and feature columns from serialized pickle files. The module maps database values (room types, meal plans, market segments) to the model's expected categorical encodings using mapping dictionaries (MEAL_MAP, ROOM_MAP, SEGMENT_MAP) and the map_and_encode () function. It processes booking features into the model's expected format, generates cancellation probability scores (0-1), and classifies risk levels (High/Medium/Low) based on probability thresholds.

Usage: Used in the /admin/bookings route to display cancellation predictions alongside each booking in the admin view, and in the /admin/booking_features/<int: booking_id> route to show detailed feature encodings for specific bookings. The prediction helps administrators identify high-risk bookings for proactive management.

Implementation of Machine Learning Model

- **Load model and encoders**

  This section is mainly focused on loading pre-trained machine learning components at application startup. Due to Machine learning models expecting features in the exact same order as training data, it was trained with features in this specific sequence.

```python
rf_model = None
encoders = {}
feature_cols = [
    'no_of_adults', 'no_of_children', 'no_of_weekend_nights', 'no_of_week_nights',
    'required_car_parking_space', 'lead_time', 'arrival_year', 'arrival_month', 'arrival_date',
    'repeated_guest', 'no_of_previous_cancellations', 'no_of_previous_bookings_not_canceled',
    'avg_price_per_room', 'no_of_special_requests', 'type_of_meal_plan_encoded',
    'room_type_reserved_encoded', 'market_segment_type_encoded', 'total_nights', 'total_guests'
]

if os.path.exists(RF_MODEL_PATH):
    try:
        with open(RF_MODEL_PATH, "rb") as f:
            rf_model = pickle.load(f)
    except Exception as e:
        print("Could not load RF model:", e)

if os.path.exists(ENCODERS_PATH):
    try:
        with open(ENCODERS_PATH, "rb") as f:
            encoders = pickle.load(f)
    except Exception as e:
        print("Could not load encoders:", e)

if os.path.exists(FEATURE_COLS_PATH):
    try:
        with open(FEATURE_COLS_PATH, "rb") as f:
            feature_cols = pickle.load(f)
    except Exception as e:
        print("Could not load feature columns:", e)
```

- **Model mapping**

  This section uses dictionaries to serve as a translation layer between the system's database values and the machine learning model's expected category names.

```python
MEAL_MAP = {
    "Mixed": "Meal Plan 1",
    "Veg": "Meal Plan 2",
    "Non Veg": "Meal Plan 3",
    "No Meal": "Not Selected",
}
ROOM_MAP = {
    "Standard": "Room_Type 1",
    "Deluxe": "Room_Type 2",
    "Executive": "Room_Type 3",
    "Family Suite": "Room_Type 4",
    "Presidential Suite": "Room_Type 5",
    "Single": "Room_Type 6",
    "Double": "Room_Type 7",
}
SEGMENT_MAP = {
    "Online": "Online",
    "Offline": "Offline",
    "Corporate": "Corporate",
    "Airline Guest": "Aviation",
    "Complementary": "Complementary",
}
```

- **Mapping and encoding Process**

  This section converts the database values into an encoded integer. It handles none/ null values which prevents error when meal plan, room type or segment is missing. It also checks if the exact database value exists as a key in the mapping dictionary. It also handles case-insensitive matching, which converts both database value and dictionary keys to lowercase for comparison. When unexpected values appear in the database, it provides a reasonable fallback. It uses the default category provided by the caller.

```
def map_and_encode(db_value, mapping_dict, encoder, default_model_cat=None):
    if db_value is None:
        return -1
    model_cat = mapping_dict.get(db_value)
    if model_cat is None:
        db_lower = str(db_value).strip().lower()
        for k, v in mapping_dict.items():
            if isinstance(k, str) and k.strip().lower() == db_lower:
                model_cat = v
                break
    if model_cat is None:
        model_cat = default_model_cat
    try:
        if model_cat is not None and encoder is not None and hasattr(encoder, "classes_"):
            if model_cat in encoder.classes_:
                return int(encoder.transform([model_cat])[0])
            mc_lower = str(model_cat).lower()
            for c in encoder.classes_:
                if mc_lower in str(c).lower() or str(c).lower() in mc_lower:
                    return int(encoder.transform([c])[0])
    except Exception:
        pass
    return -1
```

- **Prediction Execution**

This section predicts the likelihood of each hotel booking being cancelled. First, it loads the correct encoders for meal plans, room types, and market segments so that text values can be converted into numbers the model understands. For every booking, it takes human-readable values like meal plan name, room type, and segment, maps them to standard categories, and then encodes them safely, even handling missing or unknown values using defaults.

```
meal_encoder = encoders.get("type_of_meal_plan") or encoders.get("type_of_meal_plan_encoded")
room_encoder = encoders.get("room_type_reserved") or encoders.get("room_type_reserved_encoded")
seg_encoder = encoders.get("market_segment_type") or encoders.get("market_segment_type_encoded")

for b in bookings:
    meal_enc = map_and_encode(
        b["meal_plan_name"],
        MEAL_MAP,
        meal_encoder,
        default_model_cat="Not Selected"
    )
    room_enc = map_and_encode(
        b["room_type_name"],
        ROOM_MAP,
        room_encoder,
        default_model_cat="Room_Type 1"
    )
    seg_enc = map_and_encode(
        b["segment_name"],
        SEGMENT_MAP,
        seg_encoder,
        default_model_cat="Offline"
    )
```

Next, it builds a structured data row (df_input) containing all required booking features, such as number of guests, stay duration, lead time, price, and special requests. If some derived values like total nights or total guests are missing, the code calculates them automatically. The input data is then aligned exactly with the feature order the model was trained on, ensuring accurate predictions. The Random Forest model calculates the probability that the booking will be canceled. Based on this probability, the code assigns a clear prediction label (likely to cancel or not) and a risk level (low, medium, or high). Finally, these predictions are attached to each booking and sent to the admin page, allowing administrators to see both booking details and cancellation risk in one place.

```
df_input = pd.DataFrame([{
    "no_of_adults": b["no_of_adults"],
    "no_of_children": b["no_of_children"],
    "no_of_weekend_nights": b["no_of_weekend_nights"],
    "no_of_week_nights": b["no_of_week_nights"],
    "required_car_parking_space": b["required_car_parking_space"],
    "lead_time": b["lead_time"],
    "arrival_year": b["arrival_year"],
    "arrival_month": b["arrival_month"],
    "arrival_date": b["arrival_date"],
    "repeated_guest": b["repeated_guest"],
    "no_of_previous_cancellations": b["no_of_previous_cancellations"],
    "no_of_previous_bookings_not_canceled": b["no_of_previous_bookings_not_canceled"],
    "avg_price_per_room": b["avg_price_per_room"],
    "no_of_special_requests": b["no_of_special_requests"],
    "type_of_meal_plan_encoded": meal_enc,
    "room_type_reserved_encoded": room_enc,
    "market_segment_type_encoded": seg_enc,
    "total_nights": b["total_nights"] if b["total_nights"] is not None else (b["no_of_weekend_nights"] + b["no_of_week_nights"]),
    "total_guests": b["total_guests"] if b["total_guests"] is not None else (b["no_of_adults"] + b["no_of_children"])
}])

df_input = df_input.reindex(columns=feature_cols, fill_value=0)
prob = rf_model.predict_proba(df_input)[0, 1] if rf_model is not None else 0.0
```

4.2 Testing

4.2.1 Test cases for Unit Testing

Table 4.1: Test case for unit testing of User login and logout

S. No	Action	Expected Outcome	Actual Outcome	Test Result
1	Enter correct name, email and password for user	User Home page	User Home page	Pass
2	Enter correct name, email or password for admin	Admin Home page	Admin Home page	Pass
3	Enter incorrect login credentials	Invalid credentials	Invalid credentials	Pass
4	Logging out	Returns to login page	Login page	Pass

Table 4.2: Test cases for unit testing of User registration

S. No	Action	Expected Outcome	Actual Outcome	Test Result
1	Enter invalid data type	Returns Error: invalid datatype	Returns error	Pass
2	Enter the correct data type	Redirects to Login page	login page	Pass

Table 4.3: Test cases for unit testing of User operations

S. No	Action	Expected Outcome	Actual Outcome	Test Result
1	Views Available Rooms	List of available rooms	List of available rooms	Pass
2	View Rooms Booked	Displays previous bookings	Previous booking datas	Pass
3	Books new room	Room booked	Room booked	Pass

Table 4.4: Test cases for unit testing of Admin operations

S. No	Action	Expected Outcome	Actual Outcome	Test Result
1	View Rooms	List of total rooms	List of total rooms	Pass
2	View Booked Rooms	List of data for booked room	List of data for booked room	Pass
3	Adds Room	Room added to the system	Room added to the system	Pass
4	Views the cancellation rate of a room	Shows the cancellation rate	Shows the cancellation rate	Pass
5	Deletes room	Room deleted from the system	Deletes room from the system	Pass

4.2.2 Test case for System Testing

Table 4.1: Test cases for system testing of Risk Cancellation Prediction

S. No	Action	Expected Outcome	Actual Outcome	Test Result
1	Low Risk Booking	Probability < 0.4, risk level: "Low"	Probability < 0.4, risk level: "Low"	Pass
2	Medium Risk Booking	Probability 0.4-0.7, risk level: "Medium"	Probability 0.4-0.7, risk level: "Medium"	Pass
3	High Risk Booking	Probability > 0.7, risk level: "High"	Probability > 0.7, risk level: "High"	Pass

Table 4.2: Test cases for system testing of Access Control

S. No	Action	Expected Outcome	Actual Outcome	Test Result
1	Logged in, access /my_bookings	User booking displayed	Page loads successfully, user's bookings displayed	Pass
2	Not logged in, access /admin/rooms	Redirects to /login page	Redirected to /login	Pass
3	User logs out, tries to access /user	Session cleared and redirects to login	Redirected to /login, session data cleared	Pass
4	Logged in, access /user	User info loads successfully	Logged in, access /user	Pass

CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATION

5.1 Conclusion

Hence the system successfully predicts hotel booking cancellations using machine learning. It helps admin manage resources efficiently and minimize revenue loss. On the plus side, users will be able to have flexible options for payment and able to book rooms as per their preferences.

5.2 Lesson Learnt

Through this project, I have enhanced my problem-solving abilities, learned how to select appropriate resources, and developed the skill to effectively identify and implement solutions to different challenges encountered during the development process

- **Problem solving skills**
  I developed strong problem-solving abilities by tackling various bugs and errors that occurred during system development. I learned the importance of selecting the right techniques and approaches to efficiently identify and resolve issues.
- **Writing skills**
  I improved my writing skills through the preparation of the project. Additionally, I gained experience in using different CASE tools to design Use Case Diagram, DFDs, Database Schema, etc.
- **Time management**
  Time management proved to be a vital lesson throughout this project. Following the project schedule helped me understand how to allocate time effectively for each task.

5.3 Future Recommendation

Further plans for my hotel room cancellation systems can include things like using deep learning models for better accuracy. Integrating a dynamic dashboard for both user and admin would also improve the overall design of the system. Finally, using multiple machine learning technology and their comparison can help the admin to take more precise decisions and provide better stats than the current system provides.

References

[1] I. &. o. Hermawan, "Predicting Hotel Booking Cancellations Using Machine Learning for Revenue Optimization".

[2] E. &. N. Saputro, "Exploratory Data Analysis and Booking Cancellation Prediction on Hotel Booking Demands Datasets".

[3] P. Biecek, "Hotel Booking: Explaining Model Predictions for Hotel Cancellations.," [Online]. Available: https://pbiecek.github.io/xai_stories/story-hotel-booking.html.

[4] N. d. A. A. &. N. L. Antonio. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340918315191.

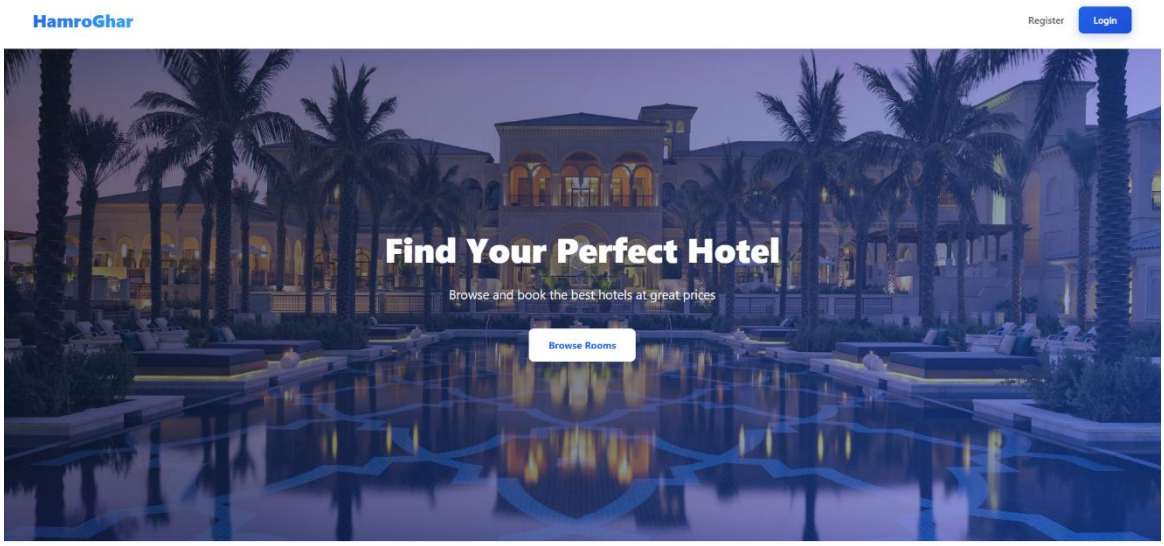
[5] A. i. E. Innovation, "Hotel Booking Cancellation and Machine Learning: A Comparative Study of Algorithms," [Online]. Available: https://www.ewadirect.com/journals/aei/article/view/20826.

[6] M. S. a. R. Y. Zou, "The random forest algorithm for statistical learning," vol. 20, no. 1, 2020.

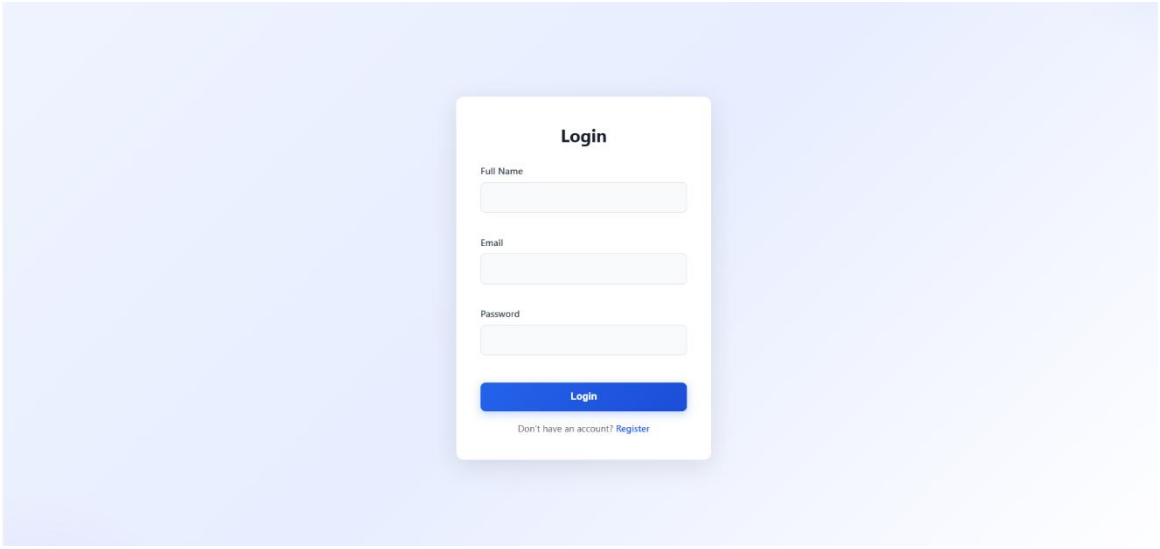
[7] N. M. A. a. A. M. Abdulazeez, "Machine Learning Classification Based on Radom Forest Algorithm: A Review," 2021.

Appendices

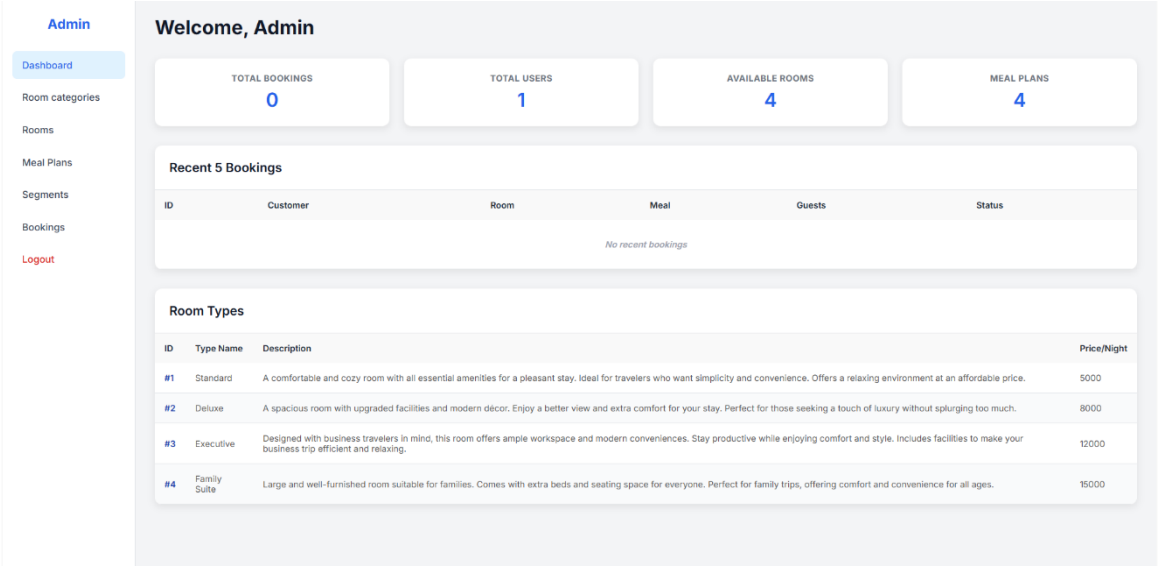
Landing Page



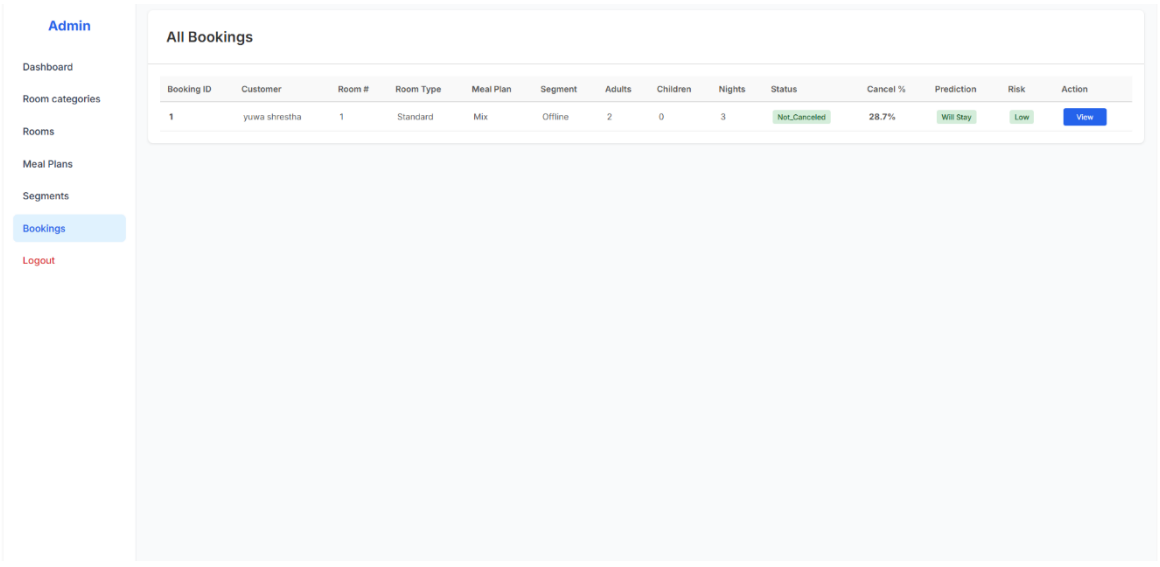
Login Page



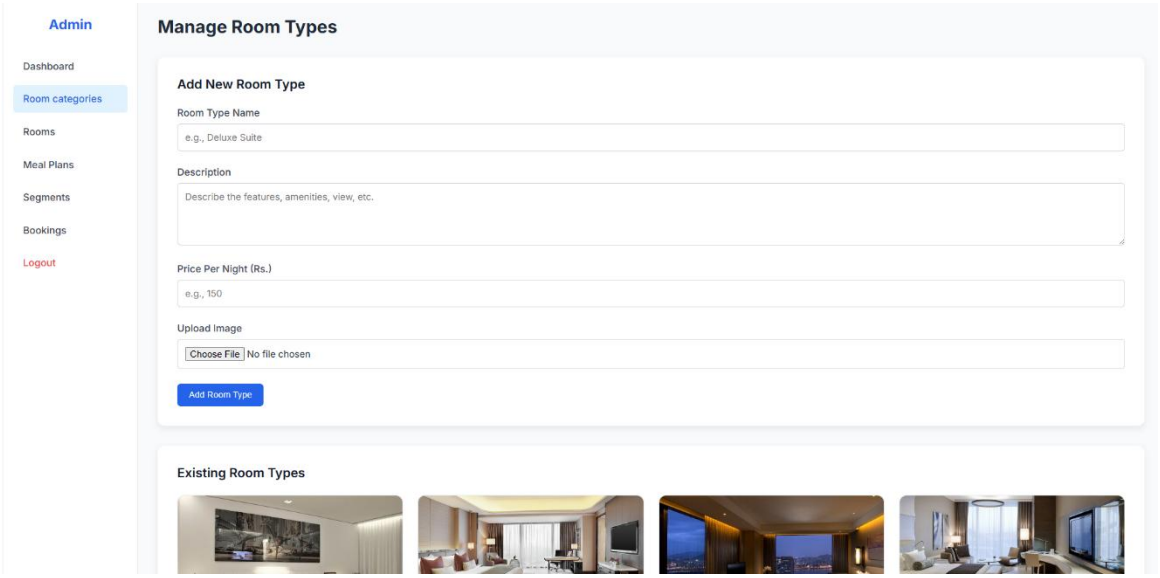
Admin Dashboard Page



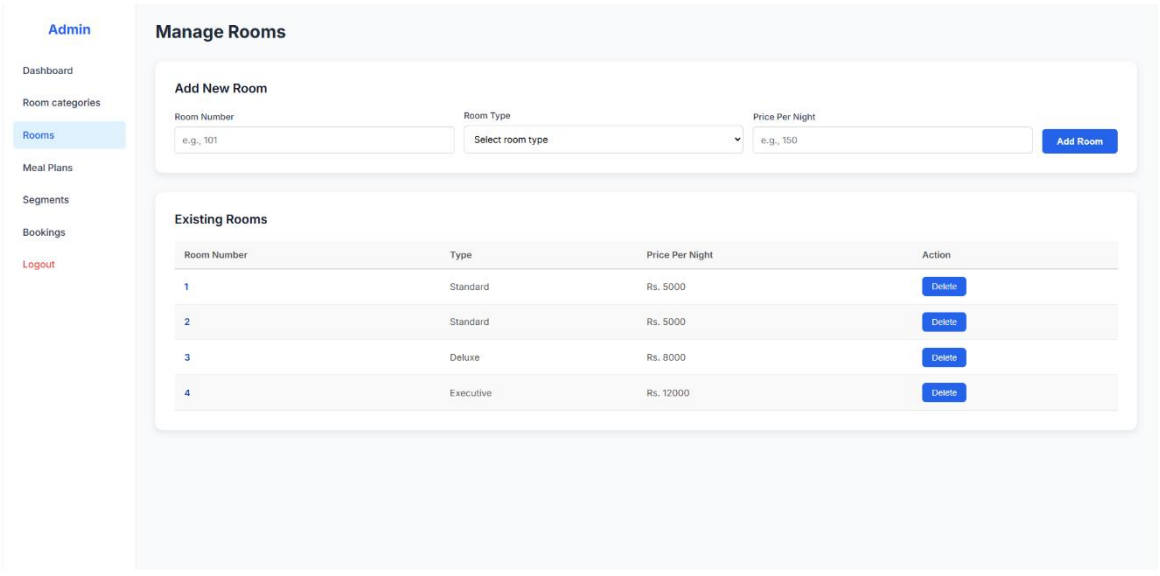
View Total Booking page



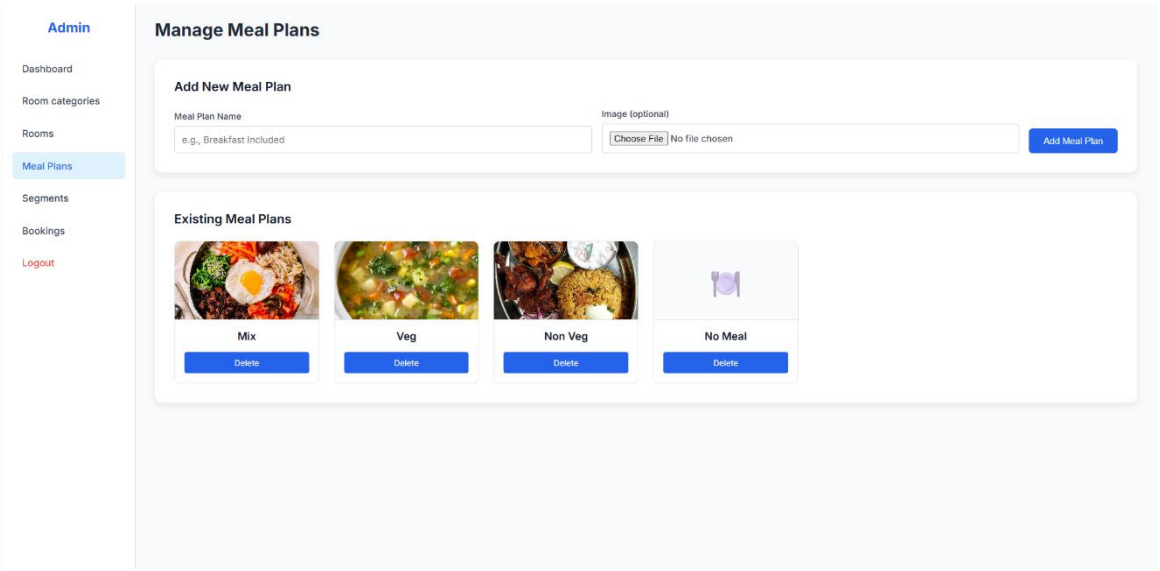
Add Room Categories page



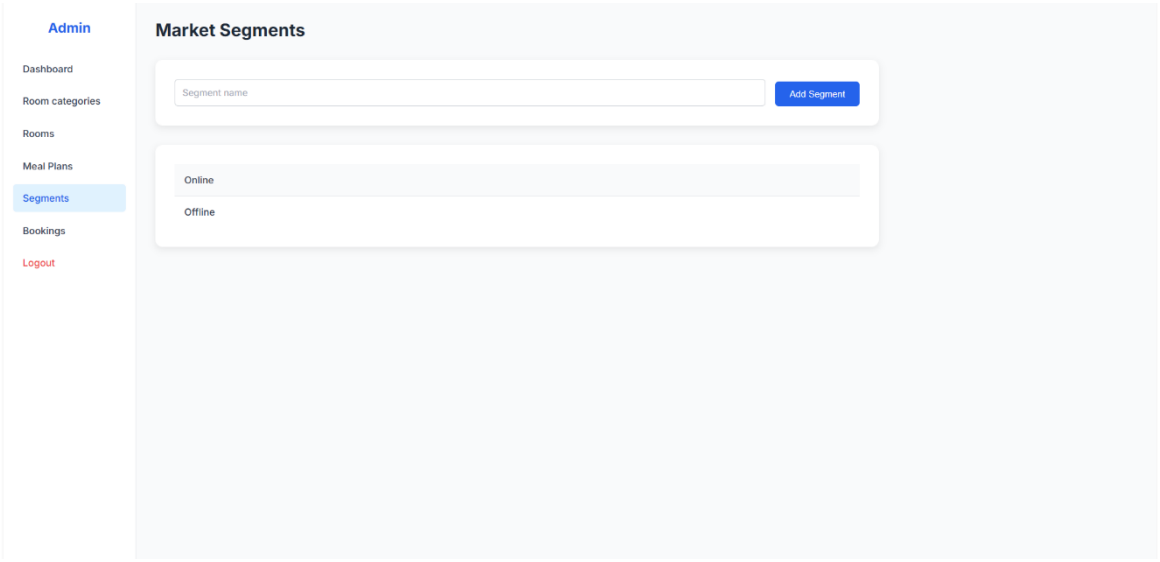
Add Room Page



Add Room meals Page

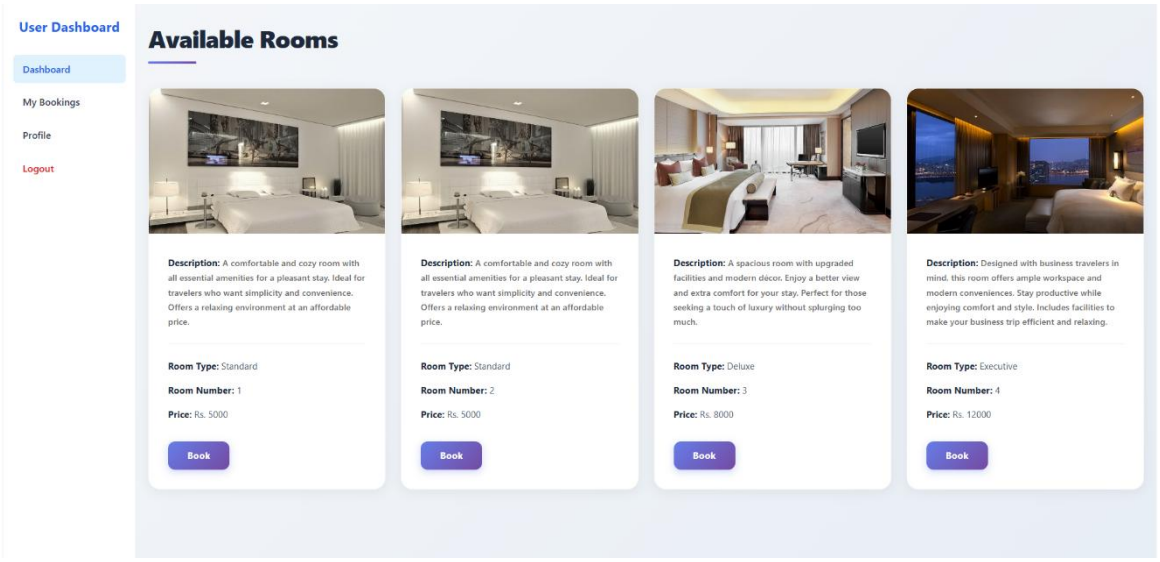


Add Room Segments Page

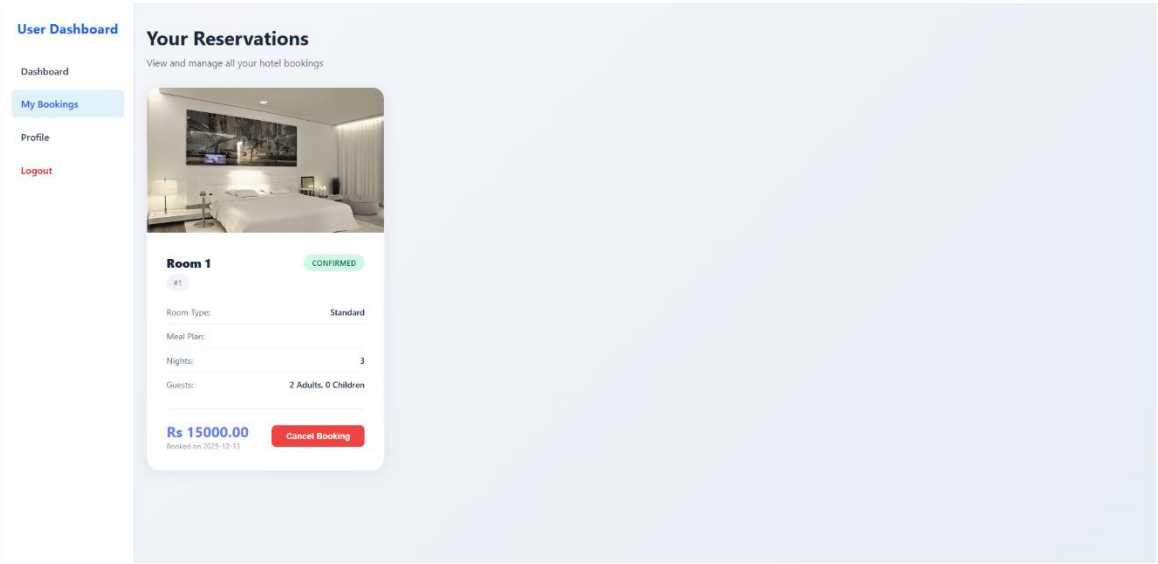


d

User Dashboard page



User's Booking History Page



Room’s detail Page