VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi-560018, Karnataka, India

A Mini Project Report On

"RAILWAY RESERVATION SYSTEM"

Submitted in partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering in Computer Science and Engineering

Submitted by

Aruna Mrutyunjay Badiger [1VE21CS020]
Dhanuja K M [1VE21CS040]
Bhuvaneshwari M[1VE22CS402]
Chaitra T R [1VE22CS403]

Under the Guidance of

Mrs. ARPITHA J
Assistant Professor
Department of CSE

SVCE BENGALURU
SRI VENKATESHWARA COLLEGE OF ENGINEERING
— Affiliated to VTU, Approved by AICTE, Recognised by UGC u/s 2(f) & 12(B)—

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESHWARA COLLEGE OF ENGINEERING
Vidyanagar, Bengaluru-562157

2023-2024

SRI VENKATESHWARA COLLEGE OF ENGINEERING

(Affiliated by A.I.C.T.E and approved by V.T.U)

CERTIFICATE

This is to certify that the Mini-Project entitled **"RAILWAY RESERVATION SYSTEM"** carried out by **Ms. AKANKSHA SINGH-1VE21CS072, Ms. MUSKAN-1VE21CS103 and Mr. NIRUV BHARDWAJ-1VE21CS118**, of VI Semester students of Sri Venkateshwara College of Engineering, in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum during the academic year 2023-2024. The Mini-Project report has been approved as it satisfies the academic requirements in respect of Full Stack Development (21CS62) work prescribed for the said Degree.

....	....
Signature of Course Teacher	**Signature of HOD**
Mrs. ARPITHA J	**Dr. HEMA M S**
Asst. Prof, Dept. of CSE	**Dept. of CSE**
SVCE, Bangalore	**SVCE, Bangalore**

Name of the Examiners:	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without complementing those who made it possible, whose guidance and encouragement made our efforts successful.

Our sincere thanks to highly esteemed institution **SRI VENKATESHWARA COLLEGE OF ENGINEERING** for grooming up us in to be software engineer.

We express our sincere gratitude to **Dr.. NAGESHWARA GUPTHA**, Principal, SVCE, Bengaluru for providing the required facility.

We are extremely thankful to **Dr. HEMA M S**, HOD of CSE, SVCE for providing support and encouragement.

We are grateful to **Mrs. ARPITHA J**, Professor, Dept. of CSE, SVCE who helped us to complete this project successfully by providing guidance, encouragement and valuable suggestion during entire period of the project. We thank all our computer science staff and others who helped directly or indirectly to meet our project work with grand success.

Finally, we are grateful to our parents and friends for their invaluable support guidance and encouragement.

TABLE OF CONTENTS

ABSTRACT

The Railway Reservation System using Django is a web-based application designed to facilitate the process of booking and managing train tickets. This system aims to streamline and automate the entire railway reservation process, providing a user-friendly interface for both passengers and railway administrators.

The application includes a secure user registration and login system, with different roles and permissions for passengers and administrators. Passengers can easily search for trains based on various criteria such as source, destination, and date, and view real-time train schedules. The booking process is designed to be user-friendly, allowing passengers to select seats and calculate fares. Electronic tickets (e-tickets) are generated upon successful booking.

Users can also view their booking history and have the option to cancel or modify their bookings. The system includes an administrative interface where administrators can manage train schedules, routes, and fares. They can also monitor and manage user bookings, and generate reports and analytics for operational insights.

For secure payment processing, the system integrates with various payment gateways, supporting multiple payment methods. The backend of the system is powered by Django, a robust and secure web framework, with the database managed by PostgreSQL. The frontend is built using HTML, CSS, JavaScript, and Bootstrap for a responsive and interactive user interface. REST APIs are used for integrating with external services.

The primary objectives of this system are to automate the railway ticket booking and management process, reduce manual efforts and errors, and provide an easy-to-use platform for passengers. Additionally, it aims to enhance the efficiency of railway operations through a centralized and automated system, ensuring the security of user data and transactions through robust authentication and encryption mechanisms.

In conclusion, the Railway Reservation System using Django is a comprehensive solution designed to modernize the traditional railway reservation process. By leveraging the capabilities of Django and modern web technologies, the system provides a seamless and efficient platform for both passengers and railway administrators.

INTRODUCTION

The Railway Reservation System using Django is a web-based application designed to streamline the process of booking and managing train tickets. It offers a user-friendly interface for passengers to search for trains, book tickets, and view booking history, while administrators can manage train schedules, routes, and fares. The system features secure user authentication, real-time train schedules, and electronic ticket generation. By leveraging Django for backend development and modern web technologies for the frontend, this system aims to automate railway operations, enhance user experience, and ensure data security through robust authentication and encryption mechanisms.

➢ Background

The traditional railway reservation systems have long been reliant on manual processes, which are often prone to errors, inefficiencies, and delays. Historically, passengers had to visit ticket counters physically or rely on third-party agents to book their train tickets, which often led to long waiting times and inconvenience. With the advent of the internet, many railway systems introduced online booking portals. However, these early systems were often plagued by usability issues, lack of real-time updates, and security vulnerabilities.

Recognizing the need for a more efficient, user-friendly, and secure solution, the Railway Reservation System using Django was developed. This system addresses the limitations of traditional reservation methods by providing a comprehensive platform that automates and streamlines the booking process. It offers real-time train schedules, a user-friendly booking interface, and secure payment integration.

By leveraging modern web technologies and the robust capabilities of Django, this Railway Reservation System aims to provide a seamless and reliable experience for both passengers and railway administrators, ultimately transforming the traditional approach to railway ticketing and management.

➢ Purpose of the Project

The purpose of the Railway Reservation System using Django is to create an efficient, automated, and user-friendly platform for booking and managing train tickets. This project aims to streamline the entire reservation process by providing real-time train schedules, easy seat selection, and secure payment options. It seeks to enhance user convenience by allowing passengers to book tickets online, view booking history, and manage reservations without the need for physical visits to counters.

Additionally, the project aims to improve operational efficiency for railway administrators by offering tools to manage train schedules, routes, and fares seamlessly. By integrating modern web technologies and robust security measures, the system ensures the protection of user data and transactions. Ultimately, the project intends to modernize the railway reservation process, reduce manual errors, and deliver a reliable and enhanced user experience for both passengers and administrators.

➢ Scope

The Railway Reservation System using Django encompasses a wide range of functionalities and modules aimed at transforming the traditional railway reservation process. The scope of this project includes the following key areas:

1. User Management

   - Secure registration and login for passengers and administrators.

   - Profile management for users to update personal information.

2. Train Schedule Management

   - Display and management of real-time train schedules.

   - Search functionality to find trains based on source, destination, and date.

3. Ticket Booking

   - Seat selection and dynamic fare calculation.

   - Generation of electronic tickets (e-tickets) upon successful booking.

4. Booking Management

   - View and manage booking history for users.

   - Options to cancel or modify existing bookings.

5. Administrative Function

- Monitoring and managing user bookings.

- Generation of operational reports and analytics for decision-making.

6. Payment Integration

- Secure integration with multiple payment gateways.

- Support for various payment methods to facilitate smooth transactions.

7. Security Features

- Implementation of robust authentication and authorization mechanisms.

- Encryption of sensitive user data to ensure privacy and security.

8. Scalability and Performance

- Designed to handle a large number of concurrent users and transactions.

- Efficient database management to support extensive data storage and retrieval.

OBJECTIVES

The primary objective of the Railway Reservation System using Django is to streamline and automate the railway ticket booking process, significantly reducing reliance on manual systems and minimizing errors. The system aims to provide a user-friendly platform for passengers, allowing them to easily search for trains, book tickets, and manage their reservations with convenience. Ensuring robust data security is another critical objective, with measures implemented to protect user data and transactions, thereby ensuring the privacy and safety of sensitive information.

Enhancing operational efficiency is a key goal, providing railway administrators with effective tools to manage train schedules, routes, and fares, thus optimizing workflow. The system offers real-time updates on train schedules and availability, aiding users in making informed booking decisions. It also integrates with multiple payment gateways, offering passengers a variety of secure payment methods for purchasing tickets.

Furthermore, the system enables users to view their booking history and modify or cancel their reservations as needed. Designed for scalability, it can handle a large number of concurrent users and transactions, ensuring reliable performance. Additionally, the system includes analytics and reporting tools to provide administrators with valuable operational insights for better decision-making. Promoting accessibility through a responsive design, the system ensures it is usable across various devices and browsers, catering to a diverse user base. By achieving these objectives, the Railway Reservation System using Django aims to modernize the traditional railway reservation process, enhancing both user satisfaction.

METHODOLOGY

The methodology involves designing a user-friendly interface using HTML, CSS, and JavaScript, developing the backend with Django for secure and scalable operations, and integrating a relational database (e.g., SQLite). Real-time train schedules, secure payment gateways, and administrative tools are implemented, followed by rigorous testing to ensure reliability and performance.

➢ Tools and Technologies

Frontend Development:

- HTML
- CSS
- JavaScript
- Bootstrap

Backend Development:

- Django (Python)

Database Management:

- PostgreSQL

Development Tools:

- Visual Studio Code

Payment Gateway Integration:

- PayPal

APIs:

- REST APIs

Testing:

- Django Testing Framework

Security:

- Django's built-in authentication and authorization mechanisms

➢ System Requirements

1.Operating System:

- Server OS:
- Linux (Ubuntu Server, CentOS, Red Hat Enterprise Linux) - preferred for its stability, security, and performance.
- Windows Server (for organizations preferring Windows environments).
- Client OS:
- Cross-platform compatibility (Windows 10/11, macOS, Linux distributions).

2.Database Management System (DBMS):

- PostgreSQL - preferred for its robustness, scalability, and support for complex queries.

3.Web Server:

- Apache HTTP Server or Nginx - for serving web pages and handling HTTP requests.

4.Programming Languages:

- Backend:
- Python - using frameworks like Django or Flask for handling server-side logic, APIs, and database interactions.
- Frontend:
- HTML, CSS, JavaScript - for building user interfaces.
- Frameworks/Libraries: React, Angular, or Vue.js for dynamic and responsive UI.

5.Python Environment:

- Version: Python 3.8 or higher.
- Dependencies: Package management with pip, virtual environments using venv or virtualenv.

➢ Development Environment

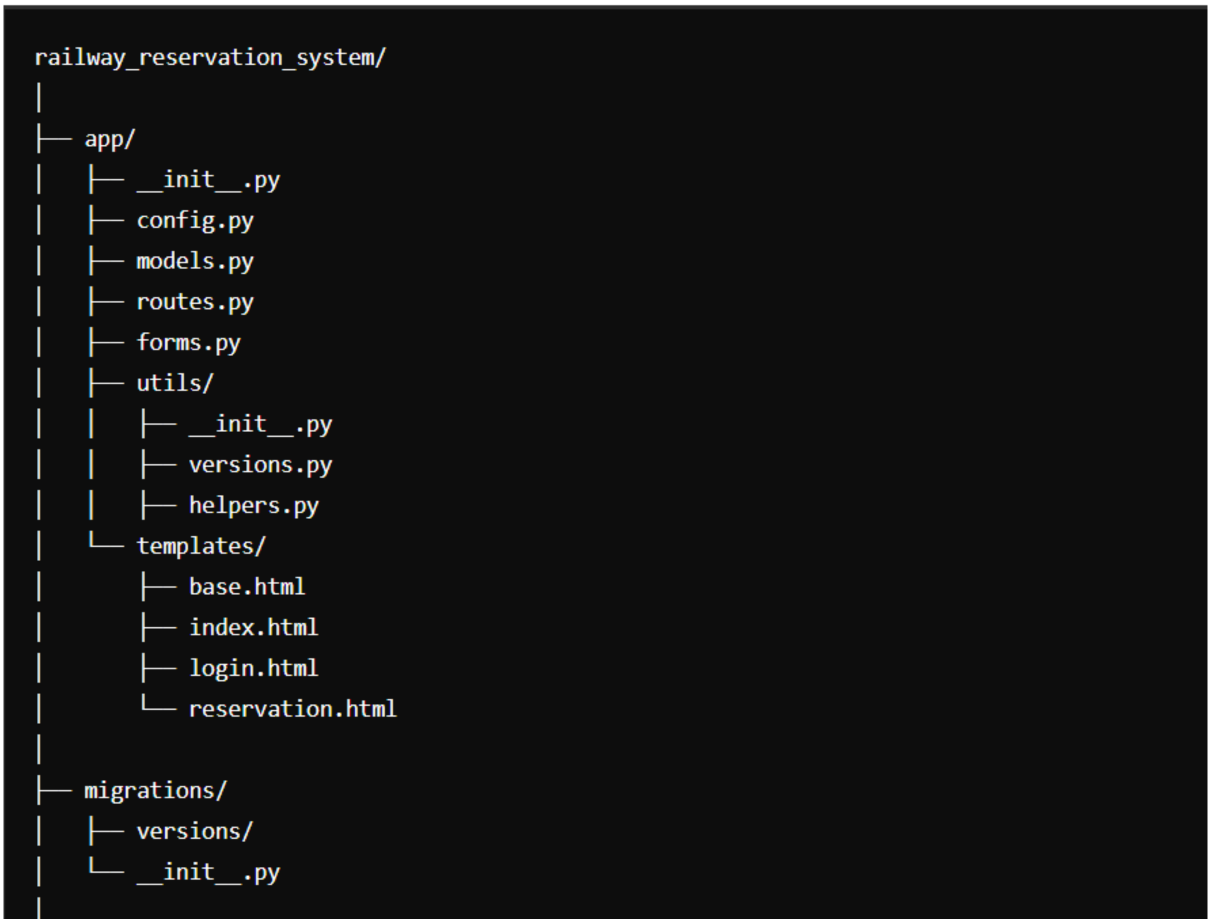
- Processor: Intel i5 or higher / AMD Ryzen 5 or higher.
- RAM: Minimum 8 GB, recommended 16 GB.
- Storage: 256 GB SSD (recommended) for faster read/write speeds.
- Display: Full HD (1920x1080) or higher resolution monitor.
- Network: Reliable internet connection.

IMPLEMENTATION

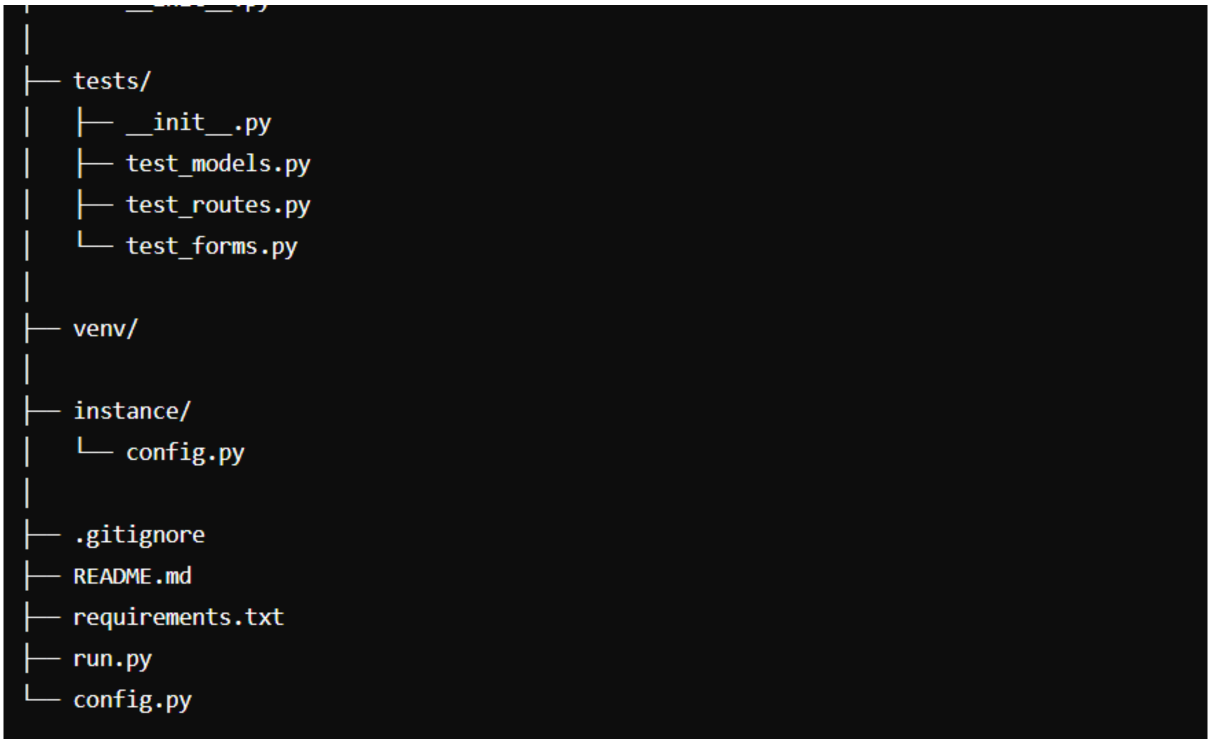
The implementation of the Django Railway Resesrvation System involves structuring the project, identifying and developing key components, and providing a detailed walkthrough of the development process. This section outlines how the project was built and how its various parts come together to create a functional railway resesrvation system.

➢ Project Structure

Creating a railway reservation system involves organizing your project structure in a way that promotes maintainability, scalability, and clarity. Below is a sample project structure for a railway reservation system using Python, Flask (a web framework), and a database (e.g., SQLite or PostgreSQL). This structure can be adjusted based on your specific requirements and technology stack.



```
railway_reservation_system/
|
├── app/
|   ├── __init__.py
|   ├── config.py
|   ├── models.py
|   ├── routes.py
|   ├── forms.py
|   ├── utils/
|   |   ├── __init__.py
|   |   ├── versions.py
|   |   ├── helpers.py
|   └── templates/
|       ├── base.html
|       ├── index.html
|       ├── login.html
|       └── reservation.html
|
├── migrations/
|   ├── versions/
|   └── __init__.py
|
```



```
|
├── tests/
|   ├── __init__.py
|   ├── test_models.py
|   ├── test_routes.py
|   └── test_forms.py
|
├── venv/
|
├── instance/
|   └── config.py
|
├── .gitignore
├── README.md
├── requirements.txt
├── run.py
└── config.py
```

- **Key Components:-**
- app/: This is the main application package.
- __init__.py: Initializes the Flask application and sets up configurations.
- config.py: Configuration settings for the application.
- models.py: Database models (e.g., User, Train, Reservation).
- routes.py: Defines the routes/endpoints for the application.
- forms.py: Defines forms for user input (e.g., login form, reservation form).

railway_reservation/settings.py:-

```
import os

from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'your-secret-key'

DEBUG = True

ALLOWED_HOSTS = []

INSTALLED_APPS = [

    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'reservations',
    'users',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'railway_reservation.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
WSGI_APPLICATION = 'railway_reservation.wsgi.application'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
STATIC_URL = '/static/'
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

reservations/models.py:

```
from django.db import models
from django.contrib.auth.models import User
class Train(models.Model):
    name = models.CharField(max_length=100)
    capacity = models.IntegerField()
    def __str__(self):
        return self.name
class Reservation(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    train = models.ForeignKey(Train, on_delete=models.CASCADE)
    date = models.DateField()
    def __str__(self):
        return f"{self.user.username} - {self.train.name} on {self.date}"
```

reservations/views.py:

```
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required
from .models import Train, Reservation
from .forms import ReservationForm
def index(request):
    trains = Train.objects.all()
    return render(request, 'reservations/index.html', {'trains': trains})
@login_required
def reserve(request):
    if request.method == 'POST':
        form = ReservationForm(request.POST)
        if form.is_valid():
            reservation = form.save(commit=False)
            reservation.user = request.user
            reservation.save()
            return redirect('index')
    else:
        form = ReservationForm()
    return render(request, 'reservations/reservation.html', {'form': form})
```

reservations/forms.py:

```
from django import forms
from .models import Reservation
class ReservationForm(forms.ModelForm):
    class Meta:
        model = Reservation
        fields = ['train', 'date']
```

railway_reservation/urls.py:

```
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('reservations.urls')),
    path('users/', include('users.urls')),]
```

Base.html

```
<!DOCTYPE html>
<html>
<head>
    <title>{% block title %}Railway Reservation System{% endblock %}</title>
</head>
<body>
    <header>
        <!-- Navigation bar -->
    </header>
    <main>
        {% block content %}
        {% endblock %}
    </main>
    <footer>
        <!-- Footer content -->
    </footer>
</body>
```

```
</html>
```

Index.html

```
{% extends 'base.html' %}
{% block content %}
<h1>Available Trains</h1>
<ul>
   {% for train in trains %}
   <li>{{ train.name }} (Capacity: {{ train.capacity }})</li>
   {% endfor %}
</ul>
{% endblock %}
```

Reservation.html

```
{% extends 'base.html' %}
{% block content %}
<h1>Make a Reservation</h1>
<form method="post">
   {% csrf_token %}
   {{ form.as_p }}
   <button type="submit">Reserve</button>
</form>
{% endblock %}
```

➢ Overview of Components

- Settings and Configuration: Manages project-wide settings.
- Models: Defines the database schema.
- Views: Handles the logic for rendering templates and processing user input.
- Forms: Manages form input and validation.
- Templates: HTML templates for rendering web pages.
- URLs: Routes user requests to the appropriate view.
- Static and Media Files: Serves static assets and user-uploaded files.

These components work together to create a functional, maintainable, and scalable railway reservation system using Django. Adjust and expand these components based on your specific requirements and the complexity of your project.

Here’s a detailed walkthrough for creating a railway reservation system using Django, covering the setup, models, views, forms, templates, URLs, and static files.

Step-by-Step Walkthrough

Step 1: Setting Up the Django Project

1. **Install Django**:

```
pip install Django
```

2. **Create Django Project:**

```
    django-admin startproject railway_reservation_system
cd railway_reservation_system
```

3. **Create Applications:**

```
        python manage.py startapp reservations
python manage.py startapp users
```

Step 2: Configuring the Project

1. **Update settings.py in railway_reservation_system:**

Add installed apps:

```
INSTALLED_APPS = [
   ...
   'reservations',
   'users',
]
```

Configure static and media files:

```
STATIC_URL = '/static/'
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'
```

1. Create urls.py in the reservations app:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('reserve/', views.reserve, name='reserve'),
]
```

2. **Update urls.py in railway_reservation_system:**

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('reservations.urls')),
    path('users/', include('users.urls')),
]
```

Step 4: Static and Media Files

1. **Add static files in static/:**
    - static/css/styles.css

```
body {
    font-family: Arial, sans-serif;
}
header {
    background: #f8f8f8;
    padding: 1em;
    text-align: center;
}
nav ul {
    list-style: none;
    padding: 0;
}
nav ul li {
    display: inline;
    margin-right: 10px;
}
```

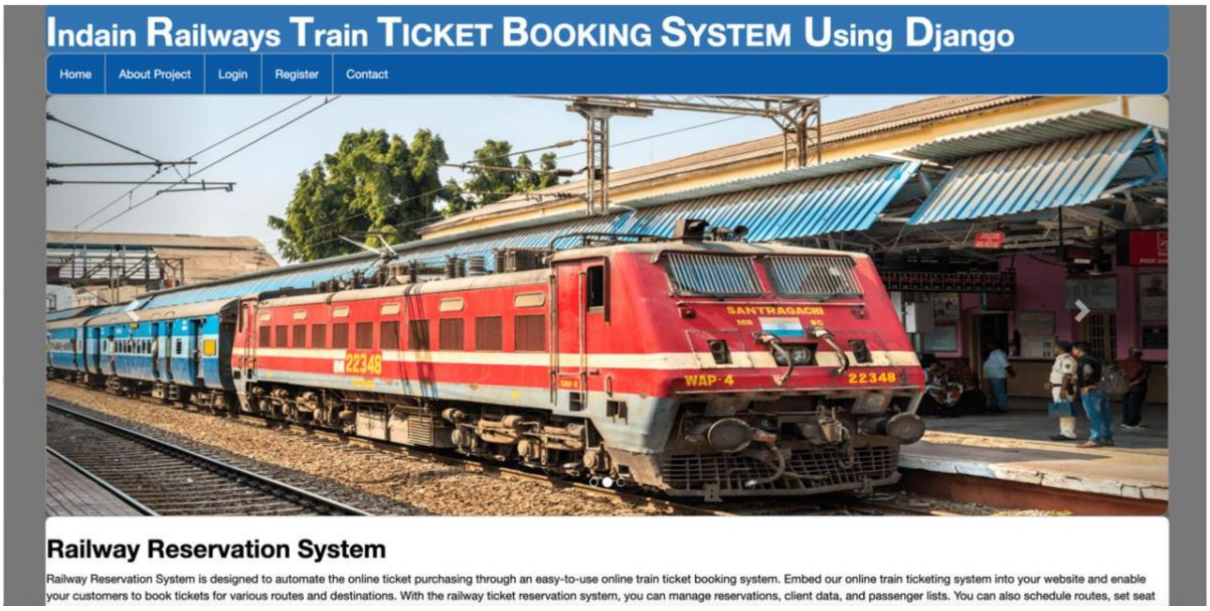
RESULTS

The results section highlights the features implemented in the Django Railway Reservation Website and the output of these features. This section demonstrates how the project objectives were achieved and the functionality provided by the platform.

➢ Features

The Django Railway Reservation Website includes a range of features designed to facilitate user interaction, content sharing, and communication:

User Authentication and Authorization: Secure user registration, login, and password management functionalities were implemented, ensuring that users can create accounts, log in securely, and manage their passwords. Django's built-in authentication system was leveraged for robust security.



Indain Railways Train TICKET BOOKING SYSTEM Using Django

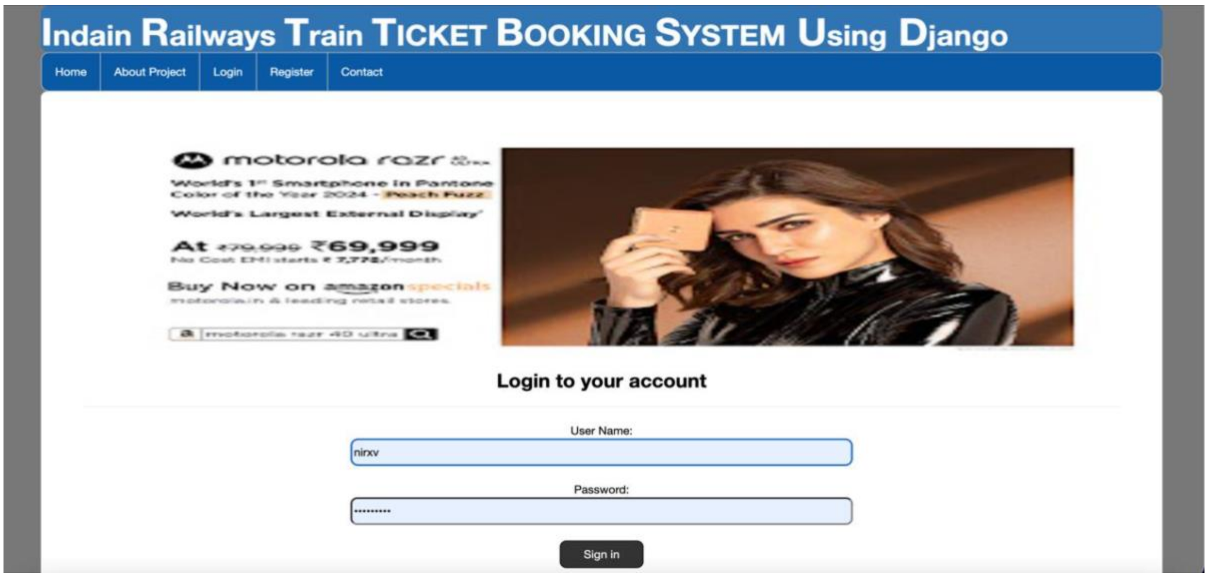
Railway Reservation System

Railway Reservation System is designed to automate the online ticket purchasing through an easy-to-use online train ticket booking system. Embed our online train ticketing system into your website and enable your customers to book tickets for various routes and destinations. With the railway ticket reservation system, you can manage reservations, client data, and passenger lists. You can also schedule routes, set seat

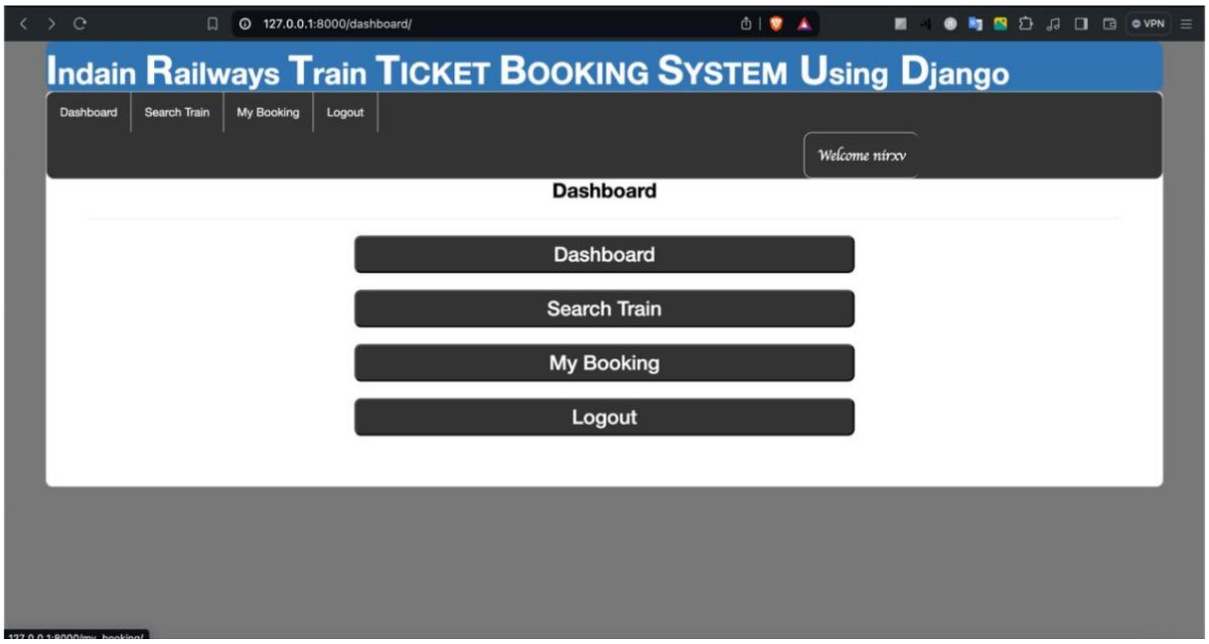
Login Page (login.html)

URL: /users/login/

Description: This page allows users to log in. The form includes fields for the username and password. Once logged in, users will be redirected to the home page.

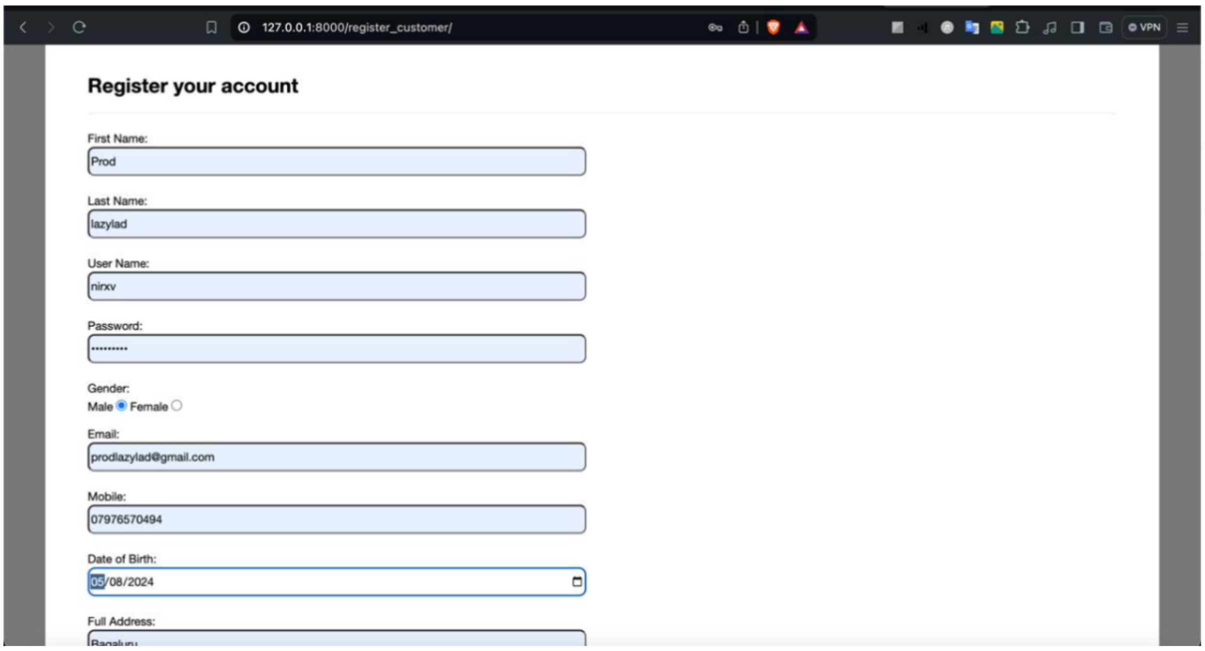


Dashboard:



Reservation-page:

A registration page allows new users to create an account on your website. This is particularly useful for a railway reservation system, as it enables users to manage their reservations, view booking history, and personalize their experience. Here's a step-by-step guide to adding a registration page to your Django project.



127.0.0.1:8000/register_customer/

Register your account

First Name: Prod

Last Name: lazylad

User Name: nirxv

Password: ••••••••••

Gender: Male [✓ ●] Female ○

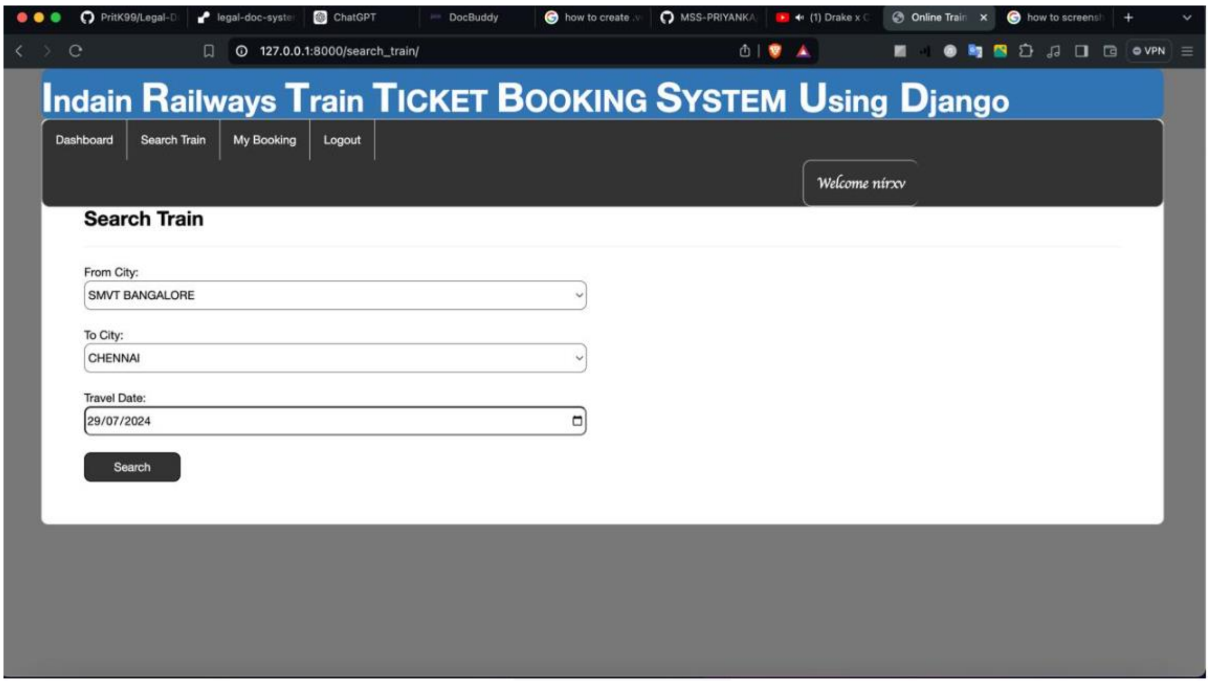
Email: prodlazylad@gmail.com

Mobile: 07976570494

Date of Birth: 05/08/2024

Full Address: Bagaluru

Search-Trains:



127.0.0.1:8000/search_train/

Indain Railways Train TICKET BOOKING SYSTEM Using Django

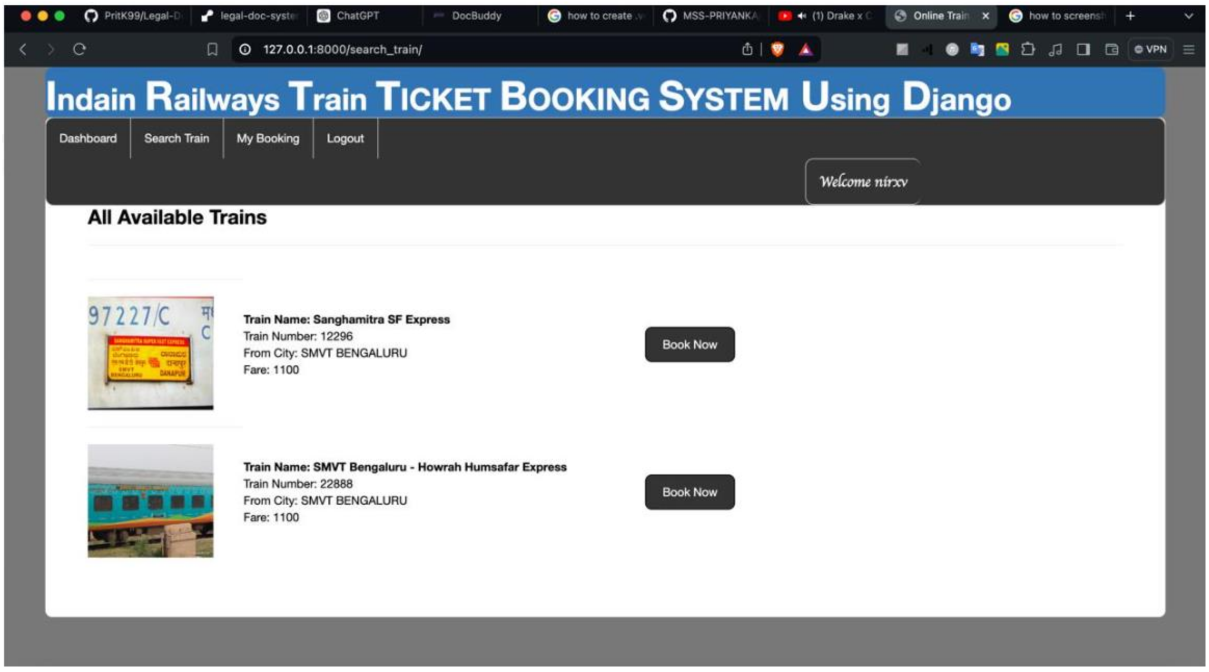
Search Train

From City: SMVT BANGALORE

To City: CHENNAI

Travel Date: 29/07/2024

All available trains :



Indain Railways Train Ticket Booking System Using Django

Dashboard | Search Train | My Booking | Logout

Welcome nirav

All Available Trains

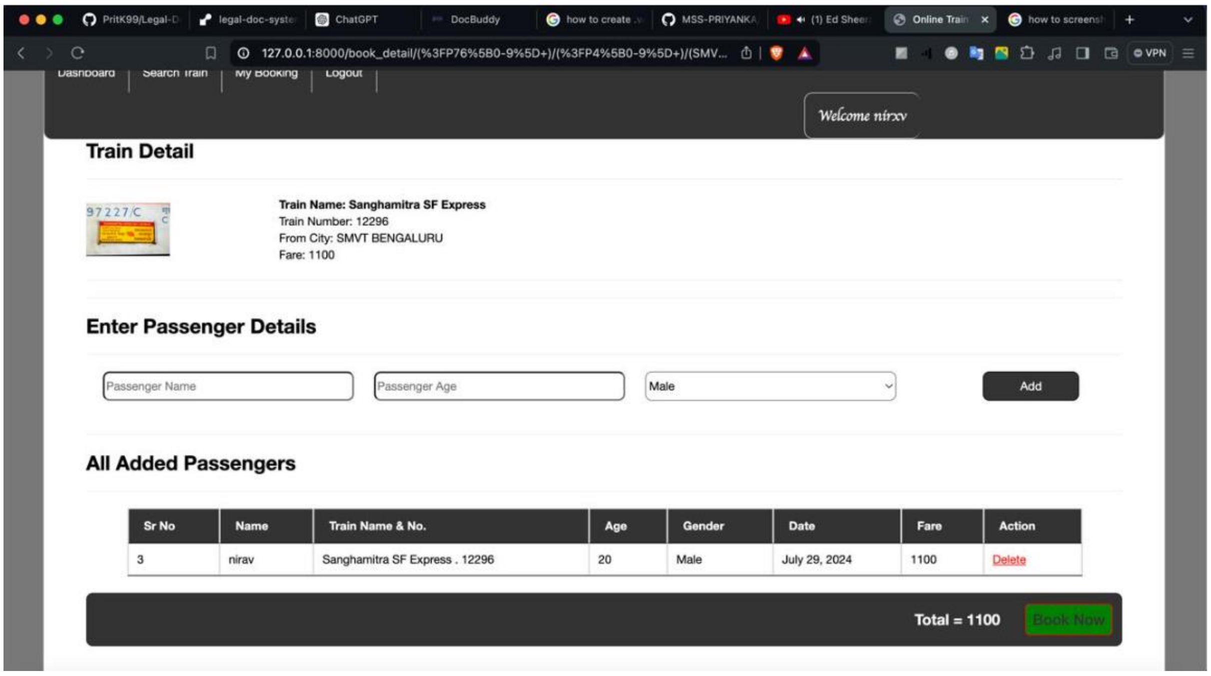
Train Name: Sanghamitra SF Express
Train Number: 12296
From City: SMVT BENGALURU
Fare: 1100

Book Now

Train Name: SMVT Bengaluru - Howrah Humsafar Express
Train Number: 22888
From City: SMVT BENGALURU
Fare: 1100

Book Now

Adding Passenger Details :



Train Detail

Train Name: Sanghamitra SF Express
Train Number: 12296
From City: SMVT BENGALURU
Fare: 1100

Enter Passenger Details

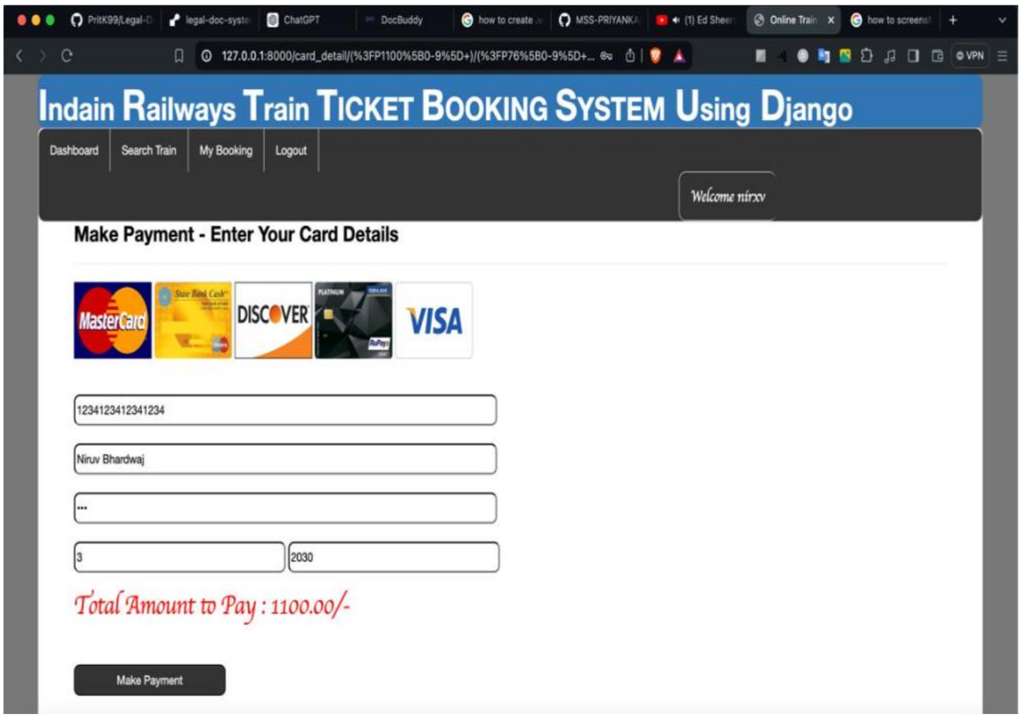
Passenger Name | Passenger Age | Male | Add

All Added Passengers

Sr No	Name	Train Name & No.	Age	Gender	Date	Fare	Action
3	nirav	Sanghamitra SF Express . 12296	20	Male	July 29, 2024	1100	Delete

Total = 1100 Book Now

Make Payment :


127.0.0.1:8000/card_detail/(%3FP1100%5B0-9%5D+)/(%3FP76%5B0-9%5D+...
Indain Railways Train TICKET BOOKING SYSTEM Using Django
Dashboard
Search Train
My Booking
Logout
Welcome nirxv
MasterCard
DISCOVER
VISA
Make Payment

Make Payment - Enter Your Card Details

1234123412341234

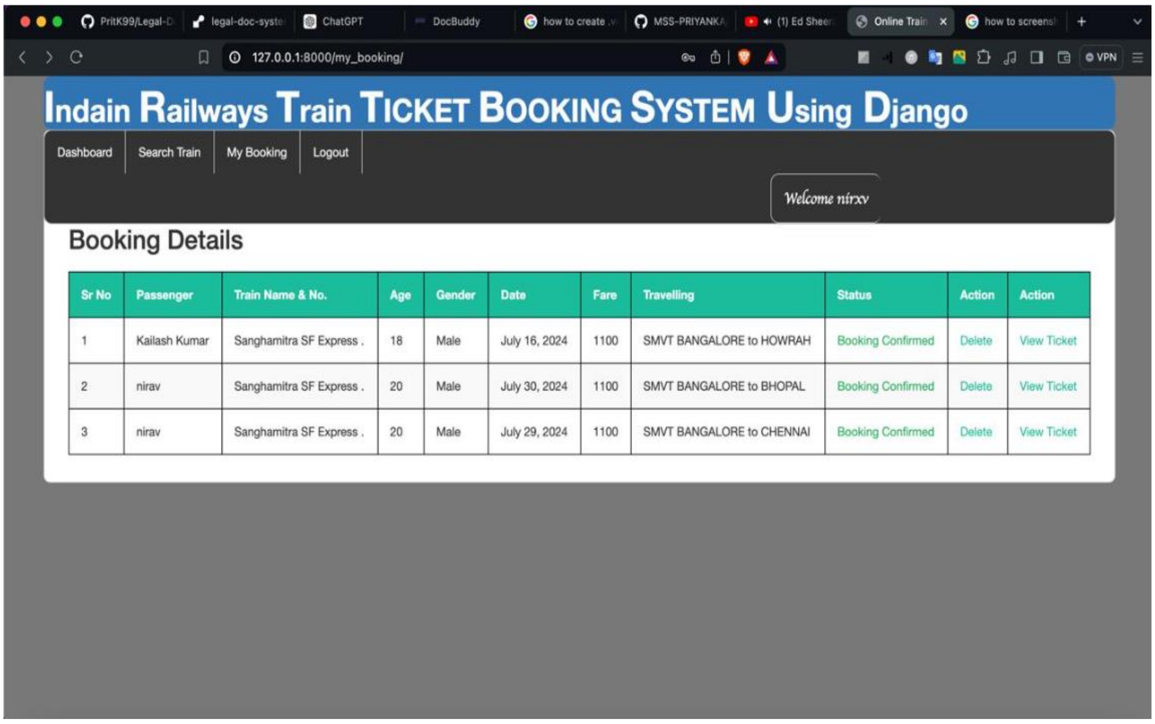
Niruv Bhardwaj

•••

3 | 2030

Total Amount to Pay : 1100.00/-

Booking details :


127.0.0.1:8000/my_booking/
Indain Railways Train TICKET BOOKING SYSTEM Using Django
Dashboard
Search Train
My Booking
Logout
Welcome nirxv

Booking Details

Sr No	Passenger	Train Name & No.	Age	Gender	Date	Fare	Travelling	Status	Action	Action
1	Kailash Kumar	Sanghamitra SF Express .	18	Male	July 16, 2024	1100	SMVT BANGALORE to HOWRAH	Booking Confirmed	Delete	View Ticket
2	nirav	Sanghamitra SF Express .	20	Male	July 30, 2024	1100	SMVT BANGALORE to BHOPAL	Booking Confirmed	Delete	View Ticket
3	nirav	Sanghamitra SF Express .	20	Male	July 29, 2024	1100	SMVT BANGALORE to CHENNAI	Booking Confirmed	Delete	View Ticket

CONCLUSION

➢ Summary

Creating a railway reservation system using Django involves several essential steps to ensure a robust and user-friendly application. The process begins with setting up the project and configuring necessary settings such as database connections, installed apps, and static and media file handling. These configurations form the foundation of the project, enabling all components to work together seamlessly.

Next, the core models are created. For this system, you would define models for `Train` and `Reservation`, representing the trains and reservations in your database. Django's built-in `User` model is used for managing user accounts. This step is crucial as it defines the structure and relationships of the data in your application.

Forms are then built to handle user input. A user registration form allows new users to create accounts, while a reservation form lets users book train tickets. These forms must include validation to ensure data integrity and provide a smooth user experience.

Views are created to handle the logic for rendering templates and processing user input. For instance, views for displaying available trains, handling reservation requests, and managing user registration are essential. These views interact with models and forms to fetch and process data, subsequently rendering the appropriate templates.

Templates are developed to display the data and forms to the users. A base template can be used to provide a consistent layout across different pages, while specific templates for the home page, registration, and reservation pages display relevant information and forms.

URLs are configured to route user requests to the appropriate views. This involves setting up URL patterns in the main project `urls.py` file and within individual app URLs, ensuring users can navigate the site easily.

Static files, such as CSS for styling and JavaScript for interactivity, enhance the user interface. Proper handling of these files ensures that the site is visually appealing and user-friendly.

FUTURE WORK

1. User Profile Management:

- Implement features that allow users to view and manage their reservations, update personal information, and view booking history through a user dashboard.

2. Search and Filtering:

- Add advanced search and filtering options to help users find trains based on criteria like departure time, destination, and train name, improving the ease of booking.

3. Payment Integration:

- Integrate an online payment gateway to facilitate secure and seamless transactions for reservations, including the ability to handle payments and provide confirmation receipts.

4. Notification System:

- Develop an automated notification system to send email or SMS alerts for reservation confirmations, cancellations, and reminders about upcoming trips.

5. Admin Features:

- Enhance the admin interface with tools for managing train schedules, capacities, and reservations, allowing for more efficient oversight and control of the system.

REFERENCES

1. Django Documentation: Django Software Foundation. (2023). Django Documentation. Retrieved from [https://docs.djangoproject.com] (https://docs.djangoproject.com)

2. Bootstrap Documentation: Bootstrap. (2023). Bootstrap Documentation. Retrieved from [https://getbootstrap.com/docs] (https://getbootstrap.com/docs)

3. PostgreSQL Documentation: PostgreSQL Global Development Group. (2023). PostgreSQL 12 Documentation. Retrieved from [https://www.postgresql.org/docs] (https://www.postgresql.org/docs)

4. Django REST Framework Documentation: Encode. (2023). Django REST Framework Documentation. Retrieved from https://www.django-rest-framework.org

5. Django Channels Documentation: Django Software Foundation. (2023). Django Channels Documentation. Retrieved from [https://channels.readthedocs.io] (https://channels.readthedocs.io)

6. Docker Documentation: Docker Inc. (2023). Docker Documentation. Retrieved from [https://docs.docker.com] (https://docs.docker.com)

7. Git Documentation: Software Freedom Conservancy. (2023). Git Documentation. Retrieved from [https://git-scm.com/doc](https://git- scm.com/doc)

8. Nginx Documentation: Nginx, Inc. (2023). Nginx Documentation. Retrieved from [https://nginx.org/en/docs] (https://nginx.org/en/docs)

9. Travis CI Documentation: Travis CI GmbH. (2023). Travis CI Documentation. Retrieved from [https://docs.travis-ci.com] (https://docs.travis-ci.com)

10. Python Documentation: Python Software Foundation. (2023). Python 3.6 Documentation. Retrieved from [https://docs.python.org/3.6] (https://docs.python.org/3.6)