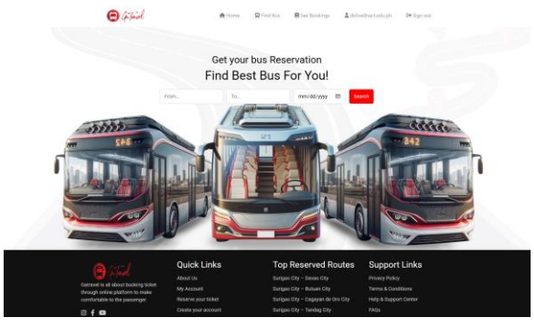
Implementing a Web-Based Employment Platform with Django: Bus Reservations System

Daryl D. Silva
BSCpE-3A



ABSTRACT

The Bus Reservation System is a comprehensive software solution designed to streamline the process of booking bus tickets, managing routes, and improving customer experience. This system enables users to search for available buses, view schedules, and reserve seats conveniently through an intuitive interface. It integrates real-time seat availability, secure payment processing, and automated ticket generation to reduce manual errors and enhance operational efficiency. For administrators, the system provides tools to manage bus routes, schedules, bookings, and customer information, ensuring seamless control over daily operations. By digitizing the traditional ticketing process, the Bus Reservation System reduces queues, minimizes paperwork, and offers a reliable, scalable platform adaptable to different transportation companies. The system aims to increase accessibility, improve user satisfaction, and support the growth of the public transportation sector through modern technology.

Key words: *The Bus Reservation System is an efficient platform that facilitates online ticket booking, route management, and real-time seat availability updates, while also incorporating secure payment integration to provide a seamless travel experience*.

I. INTRODUCTION

In today's fast-paced world, efficient and convenient transportation services are essential for both daily commuters and travelers. Traditional bus ticket booking methods, which often involve long queues and manual processes, can be time-consuming and prone to errors. To address these challenges, the Bus Reservation System has been developed as an automated solution that simplifies the process of booking bus tickets and managing routes.

The Bus Reservation System is a web-based application designed to provide users with an easy-to-use platform for searching available buses, selecting seats, and completing bookings online. It aims to improve customer experience by offering real-time updates on seat availability and schedules, eliminating the need for physical ticket counters.

For administrators, the system provides comprehensive tools to manage bus routes, schedules, and bookings efficiently, ensuring smooth operations and reducing manual workload. By integrating secure payment gateways, the system supports safe and convenient transactions.

II. SYSTEM OVERVIEW

The Bus Reservation System is a web-based application designed to simplify and automate the process of booking bus tickets. Developed using the Django framework and backed by a relational database, the system provides an integrated platform for passengers to search, reserve, and manage their bus travel plans online, while also offering bus operators tools to manage routes, schedules, and bookings efficiently.

The system's architecture is based on Django's Model-View-Template (MVT) pattern, which separates the data (Model), user interface (Template), and application logic (View). This structure ensures a clean, maintainable, and scalable codebase. The system communicates with a relational database (such as PostgreSQL or SQLite), which stores critical information including user details, bus routes, schedules, seat availability, and transaction records.

User Module:

End-users or passengers interact with the system through an intuitive web interface where they can register and log in securely. Once logged in, users can search for available bus routes by selecting departure points, destinations, and travel dates. The system then displays available buses with details such as departure time, arrival time, fare, and remaining seats. Users can select their preferred seats, make reservations, and

Implementing a Web-Based Employment Platform with Django: Bus Reservations System

Daryl D. Silva
BSCpE-3A

receive booking confirmation notifications. They also have access to a personal dashboard where they can view, modify, or cancel their bookings.

Administrator Module:

Bus operators or system administrators have a dedicated backend interface to manage the bus operations. They can add, edit, or remove bus routes and schedules, specify bus capacities, and monitor real-time booking statuses. Administrators can also generate reports on bookings, revenue, and user activity to aid operational planning. The admin interface is secured with role-based access control to ensure only authorized personnel can perform sensitive operations.

Database Management:

The database is the backbone of the system, responsible for maintaining data consistency and integrity. It handles complex transactions such as seat allocations, ensuring no overbooking occurs. The use of Django's ORM abstracts direct database queries, promoting efficient and secure data handling. The database schema includes tables for users, buses, routes, schedules, bookings, and payments (if integrated).

Key Features:

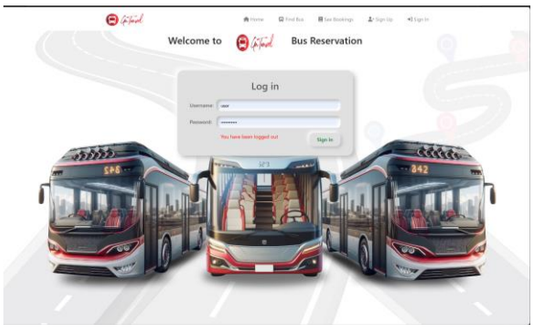
- Real-time seat availability updates prevent double booking.
- Secure user authentication and authorization.
- Responsive web design for seamless access across devices.
- Search and filter options for quick route discovery.
- Administrative control panel for comprehensive management.
- Automated email notifications for booking confirmations.
- Extensible architecture for future integration with payment gateways and third-party services.

Deployment and Scalability:

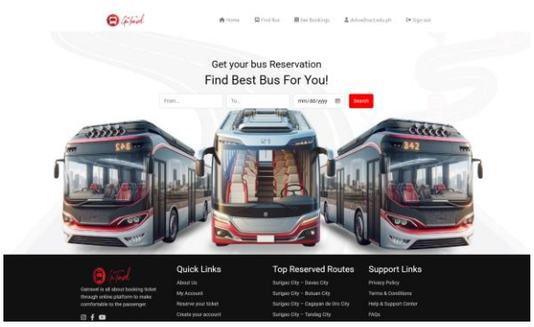
The system is deployable on various platforms, including cloud servers, enabling scalability to handle increased user traffic. Its modular design allows easy enhancement and maintenance, supporting business growth and evolving user needs.

INTERFACE OVERVIEW:

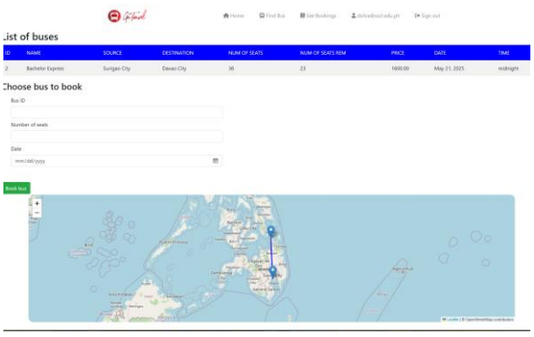
- **Log in**



- **Home**



- **Find Routes**

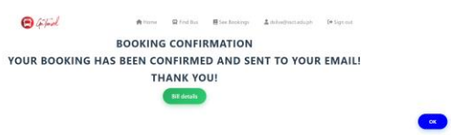


- **Able to Cancel Book**

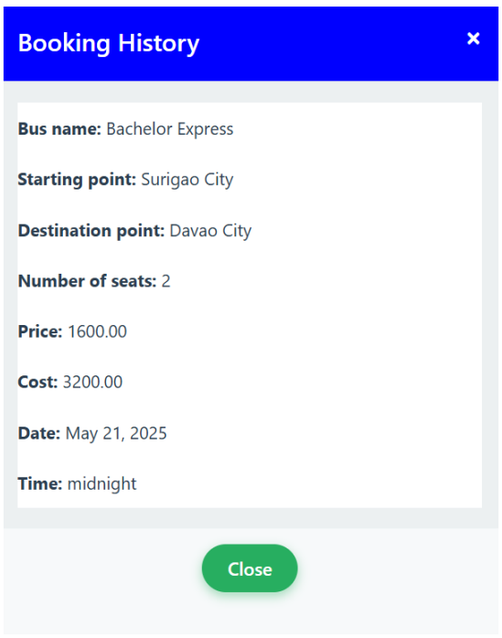
- **Book Confirmation**

Implementing a Web-Based Employment Platform with Django: Bus Reservations System

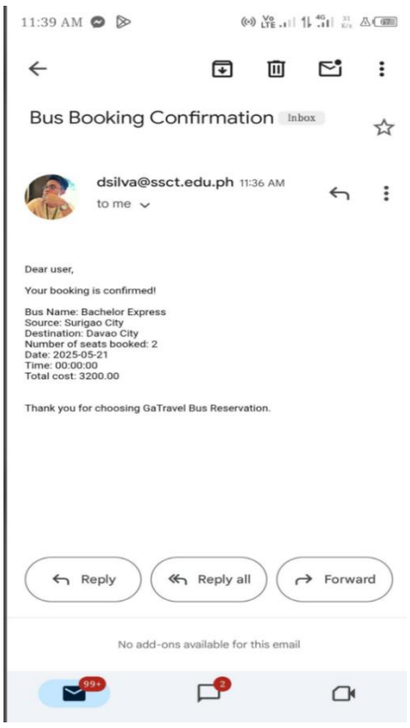
Daryl D. Silva
BSCpE-3A



- **Book History**

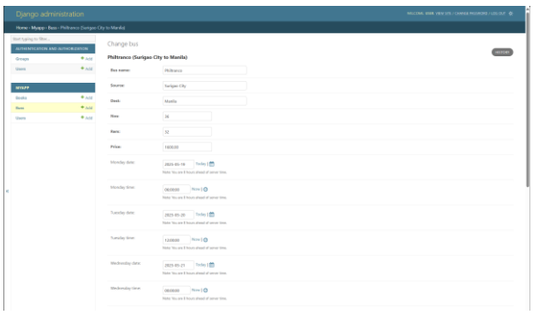


- **After Confirmation it will send via Email**

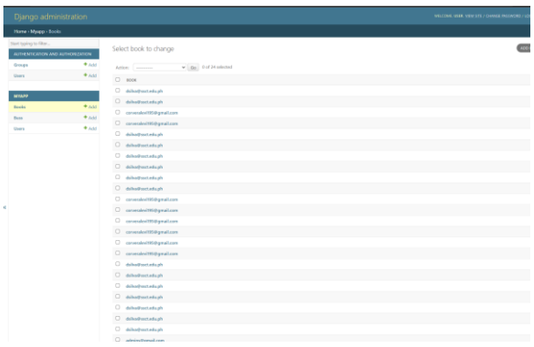


- **Admin**

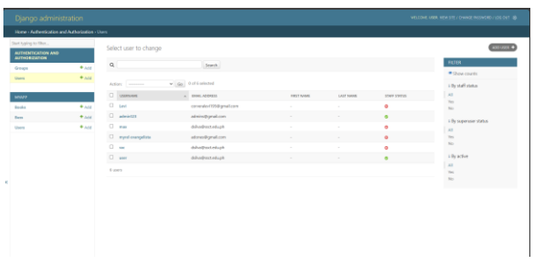
- **Admin Add Bus**



- **Admin See all who Booked**



- **Admin See all User**



III. FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Functional Requirements

Functional requirements define the specific behavior or functions of the Bus Reservation System. These are the core features the system must offer to ensure correct operation and user interaction.

1. **User Registration and Login**

- Users must be able to create an account with a valid email and password.
- The system should allow users to log in securely and manage their sessions.

2. **Search Bus Routes**

Implementing a Web-Based Employment Platform with Django: Bus Reservations System

Daryl D. Silva
BSCpE-3A

- Users can search for buses by selecting departure and destination locations along with the travel date.
- The system should display available buses that match the search criteria.

3. Bus Seat Booking

- Users should be able to view seat availability for a selected bus.
- Users can select one or more seats and complete the reservation process.

4. Booking Management

- Users can view, update, or cancel their bookings from their dashboard.
- Booking history should be stored and displayed for future reference.

5. Bus and Route Management (Admin)

- Administrators can add, update, or delete bus details, routes, schedules, and fares.
- The admin panel should allow monitoring of all bookings and users.

6. Seat Allocation Logic

- The system must allocate seats based on real-time availability and prevent double bookings.

7. Notifications

- Upon successful booking, users should receive confirmation via email or the website interface.

8. earch Filter Options

- Users should be able to filter search results by departure time, fare, or seat availability.

Non-Functional Requirements

Non-functional requirements define the overall system attributes such as performance, security, reliability, and usability.

1. Performance

- The system should respond to user actions (like search or booking) within 2-3 seconds under normal load conditions.

2. Scalability

- The system must support a growing number of users and bookings without performance degradation.
- It should be easy to scale the database and server infrastructure.

3. Security

- User data and login credentials must be securely stored using encryption and secure authentication protocols (e.g., Django's built-in user authentication).
- Only authenticated users can access booking features, and only administrators can access the admin panel.

4. Availability

- The system should be available 24/7, with minimal downtime for maintenance.
- Scheduled maintenance should not affect core booking functions.

5. Usability

- The interface must be intuitive, user-friendly, and accessible on desktop and mobile devices.
- Clear instructions and prompts should guide users through the booking process.

6. Maintainability

Implementing a Web-Based Employment Platform with Django: Bus Reservations System

Daryl D. Silva
BSCpE-3A

- The codebase should be modular and follow Django best practices to ensure ease of maintenance and future enhancements.

7. Data Integrity

- All transactions must ensure consistency—especially in seat booking and cancellation—to prevent data anomalies.

8. Compatibility

- The system should be compatible with major web browsers (Chrome, Firefox, Edge, Safari).
- It should support responsive design for proper display on smartphones and tablets.

IV. Models Source Code

```
# Create your models here.

from django.db import models



# Create your models here.


from django.db import models


from django.db import models


from django.db import models


class Bus(models.Model):
    bus_name = models.CharField(max_length=30)
    source = models.CharField(max_length=30)
    dest = models.CharField(max_length=30)
    nos = models.DecimalField(decimal_places=0, max_digits=2)
    rem = models.DecimalField(decimal_places=0, max_digits=2)
    price = models.DecimalField(decimal_places=2, max_digits=6)


    # Weekly schedule - for each day, date and time of the bus
    monday_date = models.DateField(null=True, blank=True)
    monday_time = models.TimeField(null=True, blank=True)


    tuesday_date = models.DateField(null=True, blank=True)
    tuesday_time = models.TimeField(null=True, blank=True)


    wednesday_date = models.DateField(null=True, blank=True)
    wednesday_time = models.TimeField(null=True, blank=True)


    thursday_date = models.DateField(null=True, blank=True)
    thursday_time = models.TimeField(null=True, blank=True)


    friday_date = models.DateField(null=True, blank=True)
    friday_time = models.TimeField(null=True, blank=True)


    saturday_date = models.DateField(null=True, blank=True)
```

Implementing a Web-Based Employment Platform with Django: Bus Reservations System

Daryl D. Silva
BSCpE-3A

```
    saturday_time =
models.TimeField(null=True, blank=True)


    sunday_date =
models.DateField(null=True, blank=True)
    sunday_time =
models.TimeField(null=True, blank=True)


    def __str__(self):
        return f"{self.bus_name} ({self.source}
to {self.dest})"




class User(models.Model):
    user_id =
models.AutoField(primary_key=True)
    email = models.EmailField()
    name =
models.CharField(max_length=30)
    password =
models.CharField(max_length=30)


    def __str__(self):
        return self.email



class Book(models.Model):
    BOOKED = 'B'
    CANCELLED = 'C'


    TICKET_STATUSES = ((BOOKED,
'Booked'),
                       (CANCELLED, 'Cancelled'),)
    email = models.EmailField()
    name =
models.CharField(max_length=30)
    userid
=models.DecimalField(decimal_places=0,
max_digits=2)


busid=models.DecimalField(decimal_places
=0, max_digits=2)
    bus_name =
models.CharField(max_length=30)
    source =
models.CharField(max_length=30)
    dest = models.CharField(max_length=30)
    nos =
models.DecimalField(decimal_places=0,
max_digits=2)
    price =
models.DecimalField(decimal_places=2,
max_digits=6)
    date = models.DateField()
    time = models.TimeField()
    status =
models.CharField(choices=TICKET_STAT
USES, default=BOOKED, max_length=2)


    def __str__(self):
        return self.email
```