

# **BUS RESERVATION MANAGEMENT SYSTEM**

*A Project Report submitted to  
Bishop Heber College (Autonomous), Tiruchirappalli  
affiliated to Bharathidasan University, Tiruchirappalli – 620024  
in partial fulfilment of the requirements for the award of the degree of*

## **Bachelor of Vocation in Information Technology**

**By**  
**M. SANKARANARAYANAN**  
**(Register No.: 215915143)**

*Under the guidance of*  
**Dr. M. ARRIVUKANNAMMA, MCA., M. Tech., Ph.D.,**



**Department of Information Technology**  
**Bishop Heber College (Autonomous)**  
*(Nationally Accredited with 'A<sup>++</sup>' Grade by NAAC with a CGPA of 3.69 out of 4)*  
*(Recognized by UGC as "College of Excellence")*  
**Tiruchirappalli – 620 017**

**APRIL 2024**



**Department of Information Technology,  
Bishop Heber College (Autonomous)  
Tiruchirappalli – 620 017, Tamil Nadu, India  
Phone No.: 0431 – 277 0136.**

---

## **CERTIFICATE**

This Viva-Voce examination for the candidate **M. SANKARANARAYANAN**  
(Reg. No.: 215915143) was held on\_\_\_\_\_

**Signature of the HOD**

**Examiners:**

**1.**

**2.**

**Dr. M. ARRIVUKANNAMMA, MCA., M. Tech., Ph.D.,**  
Assistant Professor,  
Department of Information Technology,  
Bishop Heber College (Autonomous),  
Tiruchirappalli – 620 017.

---



**Date:**

## **CERTIFICATE**

This is to certify that the project work entitled **“BUS RESERVATION MANAGEMENT SYSTEM”** is a Bonafide work done under my supervision by **M. SANKARANARAYANAN (Reg. No. 215915143)** and submitted to Bishop Heber College (Autonomous), Tiruchirappalli – 620017 in partial fulfilment of the requirements for the award of the degree of Bachelor of Vocation in Information Technology during the odd semester of the academic year (2023 – 2024).

**Signature of the Guide**

## DECLARATION

I hereby declare that the work presented in this project work report is the original work done by me under the guidance of **Dr. M. ARRIVUKANNAMMA, MCA., M. Tech., Ph.D.**, Assistant Professors, Department of Information, Bishop Heber College (Autonomous), Tiruchirapalli-620 017 and has not been included in any other project work submitted for any other degree.

Name of the Candidate : M. SANKARANARAYANAN

Register Number : 215915143

Semester : SIXTH

Academic Year : 2023 – 2024

Course Code : U21ITPJ6

**Signature of the Candidate**

## **ACKNOWLEDGEMENTS**

First of all, I would thank **ALMIGHTY GOD** for granting abundant grace, good health and knowledge to do this Project.

I express my sincere gratitude to **J. PRINCY MERLIN, M.Sc., M.Phil., Ph.D., PGDCI., B.Ed.**, Principal of Bishop Heber College (Autonomous), Tiruchirappalli for his blessings.

I am highly indebted to thank **Dr. J. JOHN RAYBIN JOSE, M.SC., M.Phil., M.Phil., MCA., PGDCSA., Ph.D., SET.**, Associate Professor and Head of Department, of Information Technology, Bishop Heber College (Autonomous), Trichy for granting me permission to carry out his project.

I am highly indebted to **Dr. M. ARRIVUKANNAMMA, MCA., M. Tech., Ph.D.**, Assistant Professor, Department of Information Technology, Bishop Heber College (Autonomous), Trichy for providing his support and guidance during this project work.

I thank all the faculty members of the department of Information Technology for their support and imparting us with the essential knowledge to do this project work successfully.

I record my deep sense of gratitude to my beloved parents and my friends for their encouragement and moral support extended during the period of my project.

**M.SANKARANARAYANAN**

## **ABSTRACT**

The Bus Reservation Management System, developed in Python, is a comprehensive solution designed to automate the traditional bus ticket booking process. This system provides a seamless interface for both passengers and bus operators, facilitating easy reservation of bus seats. Moreover, the system includes a secure payment gateway for ticket transactions, ensuring the safety and privacy of user data. It also provides an admin panel for bus operators to manage bus routes, schedules, and track bookings effectively.

The system incorporates essential features such as bus scheduling, ticket reservation, and cancellation. It provides distinct modules for administrators and users. Administrators can manage bus routes, schedules, and track bookings effectively, while users can easily book, cancel, and view their reservations.

In conclusion, the Bus Reservation Management System project in Python is a significant step towards modernizing the public transportation sector, making it more accessible, user-centric, and efficient. It stands as a testament to the potential of Python in developing practical and impactful solutions. This project not only provides a convenient platform for bus reservation but also serves as a model for similar systems in other sectors.

# CONTENTS

S. No.	CHAPTER	PAGE No.
<b>1</b>	<b>INTRODUCTION</b>	1
<b>2</b>	<b>SYSTEM STUDY</b>	2
	2.1 Project Description	2
	2.1.1. Existing system	3
	2.1.2. Proposed system	4
<b>3</b>	<b>SYSTEM DESIGN</b>	5
	3.1 Logical design	5
	3.2 Program Design	6
<b>4</b>	<b>SYSTEM DEVELOPMENT</b>	11
	4.1 Program Development	11
<b>5</b>	<b>SYSTEM TESTING</b>	29
	5.1 Unit Testing	29
	5.2 Integration Testing	30
	5.3 Validation testing	31
<b>6</b>	<b>SYSTEM IMPLEMENTATION</b>	32
	6.1 Software Deployment	32
	6.2 Software Demonstration	32
<b>7</b>	<b>CONCLUSION</b>	39
	<b>BIBLIOGRAPHY</b>	40

# **1. INTRODUCTION**

Bus Reservation Management System (BRMS) is an innovative solution designed to automate and streamline the operations of bus services. This system is a game-changer in the public transportation sector, offering a seamless and efficient platform for managing bus reservations.

The BRMS provides real-time information about bus schedules, seat availability, and ticket prices, enabling passengers to make informed decisions. It offers a user-friendly interface for booking tickets, selecting seats, making online payments, and even cancelling tickets if necessary. This level of convenience significantly enhances the overall customer experience.

Moreover, the BRMS plays a crucial role in promoting sustainable public transportation. By making bus travel more accessible and convenient, it encourages more people to opt for public transport, thereby reducing the reliance on private vehicles and contributing to environmental sustainability.

In conclusion, the implementation of a Bus Reservation Management System not only benefits passengers and operators but also aligns with broader societal goals. Its impact extends beyond the realm of transportation, marking a significant step towards a more connected and sustainable future.



## **2. SYSTEM STUDY**

A Bus Reservation Management System (BRMS) is a digital solution that enhances the efficiency of bus services. It provides real-time information, facilitates ticket booking, and optimizes operations. This system improves customer experience, aids operators in service management, and promotes sustainable public transportation. Overall, BRMS is a significant advancement towards a connected and sustainable future.

### **2.1 PROJECT DESCRIPTION**

The he Bus Reservation Management System project is a digital platform for managing bus services. It streamlines ticket booking, tracks buses in real-time, and provides data-driven insights for operators. This project aims to enhance customer satisfaction, improve operational efficiency, and contribute to sustainable urban mobility.

#### **2.1.1 EXISTING SYSTEM**

The Bus Reservation Management System primarily relies on manual processes, including ticket booking and tracking. It lacks real-time updates, leading to inefficiencies and customer dissatisfaction. The system's limited data analysis capabilities hinder strategic decision-making for operators, impacting overall service quality and sustainability.

#### **2.2.2 PROPOSED SYSTEM**

The proposed Bus Reservation Management System aims to digitize operations, offering online ticket booking and real-time tracking. Enhanced data analytics will enable strategic decision-making for operators. The system will improve customer satisfaction through efficient service, real-time updates, and a user-friendly interface, leading to a sustainable and high-quality service.

## 2.2 REQUIREMENT ANALYSIS

Requirements analysis involves frequent communication with system users to determine specific feature expectations, the resolution of conflict or ambiguity in requirements as demanded by the various users or groups of users, the avoidance of feature creep, and the documentation of all aspects of the project development process from start to finish. Energy should be directed towards ensuring that the final system or product conforms to client needs rather than attempting to adjust user expectations to fit the requirements. Requirements analysis is a team effort that demands a combination of hardware, software, and human factors engineering expertise as well as skills in dealing with people.

### 2.2.1. HARDWARE REQUIREMENT

The hardware specification of the laptop computer system available for developing the project is given below:

<b>Processor</b>	<b>: Intel Core I3</b>
<b>Hard disk</b>	<b>: 500GB</b>
<b>RAM</b>	<b>:4GB</b>
<b>Keyboard</b>	<b>: Standard QWERTY Keyboard</b>
<b>Mouse</b>	<b>: Standard mouse with 2 buttons</b>

### 2.2.2. SOFTWARE REQUIREMENTS

A software requirement specification (SRS) is a complete description of the behaviour of the system to be developed. It includes a set of use cases that describe all the interaction the users will have with the software.

<b>➤ Front End</b>	<b>: HTML AND CSS</b>
<b>➤ Back End</b>	<b>: MYSQL</b>

## **WINDOWS**

An operating system is software that Communicate with the hardware and allows other to run. It comprises the system software, and other utilities. The Operating system used as platform to develop this project in windows 8.1 Pro. It is a multi-user operating system.

## **PYCHARM**

PyCharm is an integrated development environment (IDE) used for programming in Python. It is developed by JetBrains, the company behind other popular IDEs like IntelliJ IDEA and PhpStorm PyCharm is available in two editions: the Community Edition and the Professional Edition. The Community Edition of PyCharm is a free and open-source IDE that provides basic features for Python development, such as code highlighting, autocompletion, debugging, and support for version control systems. The Professional Edition is a paid version of PyCharm that provides advanced features such as remote development, web development, database management, scientific tools, and more.

## **MYSQL**

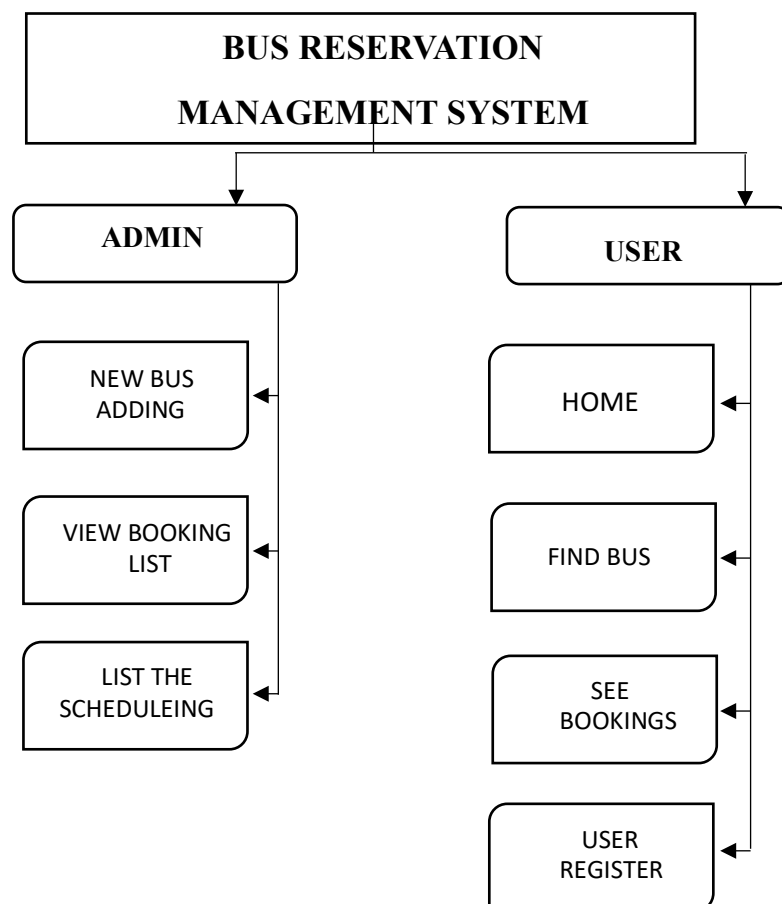
MySQL, a dynamic duo in web development, work together to create powerful and interactive web applications, A server-side scripting language, provides the logic and functionality necessary to build dynamic web pages. Enables developers to generate dynamic content, process user input, and interact with databases, while MySQL efficiently manages data storage, retrieval, and manipulation.

### 3. SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

#### 3.1. LOGICAL DESIGN

Logical design is an abstract concept in computer programming by which programmers arrange data in a series of logical relationships known as attributes or entities. An entity refers to a chunk of information, whereas an attribute defines the unique properties of an entity.



**Figure 3.1 Logical Design for “Bus Booking Management System”**

## **3.2 PROGRAM DESIGN**

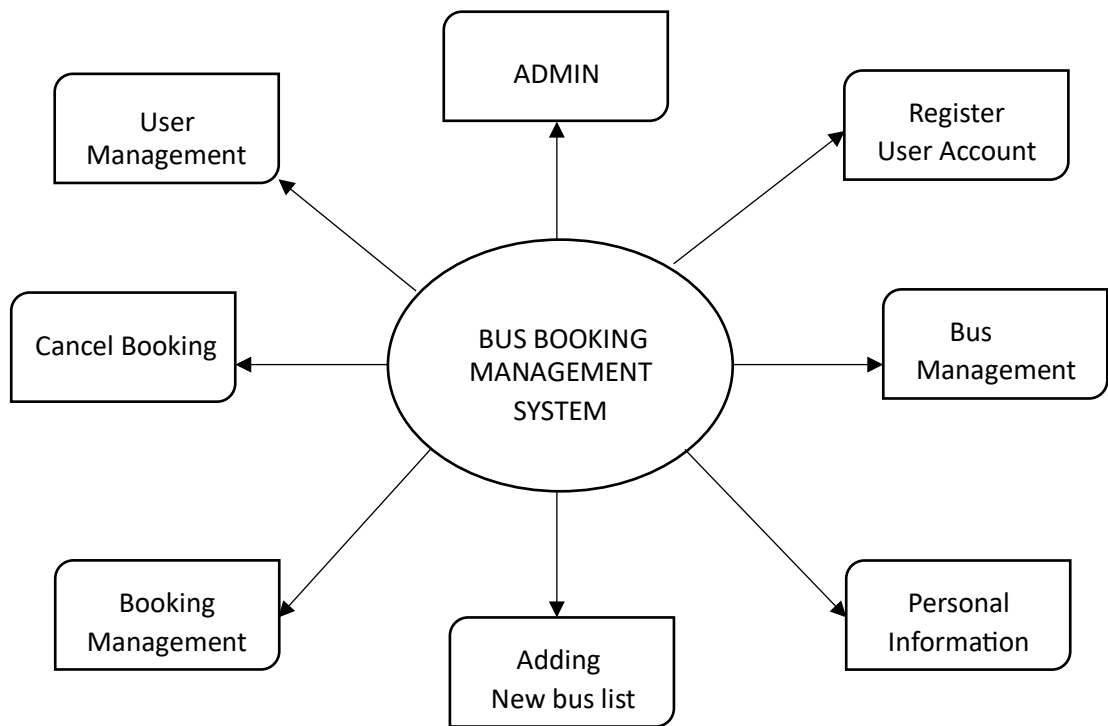
Program design is the activity of progressing from a specification of some required program to a description of the program itself. Most phase models of the software life cycle recognize program design as one of the phases. The input to this phase is a specification of what the program is required to do. During the design phase, decisions are made as to how the program will meet these requirements, and the output of the phase is a description of the program in some form that provides a "subtitle" (basis) for subsequent implementation.

## **3.3 DATABASE DESIGN**

Database design is the process of producing a detailed data model of a database. This data model contains all the logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity. A good database design is important for ensuring consistent data, the elimination of data redundancy, efficient query execution, and highperformance applications. Taking the time to design a database saves time and frustration during development, and a well-designed database ensures ease of access and retrieval of information.

### **3.3.1 DATAFLOW DIAGRAM**

A data flow diagram (DFD) provides a visual representation of the flow of information (i.e., data) within a system. By drawing a data flow diagram, you can tell which information is provided by and delivered to someone who takes part in system processes, which information is needed in order to complete the processes, and which information needs to be stored and accessed.



**Figure 3.3 Dataflow Diagram for “Stock Management System”**

### 3.4 TABLE DESIGN

One of the most common data structures is a database table. A database table consists of rows and columns. A database table is also called a two-dimensional array. An array is like a list of values, and each value is identified by a specific index.

#### 3.4.1 User Table

Attributes Name	Data Type
User Name	Varchar
Password	Integer
E-mail	Varchar

### 3.4.2 Admin Table

Attributes Name	Data Type
User Name	Varchar
Password	Integer

### 3.4.3 Bus Adding Table

Attributes Name	Data Type
Bus Name	Varchar
Source	Varchar
Destination	Varchar
Date & Time	Integer
Number of Seats	Integer
Price	Integer

### 3.4.4 Booking Details Table

Attributes Name	Data Type
Bus ID	Integer
Number of Your Seat	Integer

## 3.5 PROGRAM DESIGN

Program design consists of the steps a programmer should take before they start coding the programme in a specific language. Most phase models of the software life cycle recognise programme design as one of the phases. The input for this phase is a specification of what the programme is required to do. During the phase, design decisions are made as to how the programme will meet these requirements, and the output of the phase is a description of the programme in some form that provides a suitable basis for subsequent implementation.

### **3.5.1 MODULE DESCRIPTION**

Module is a software component or part of a programme that allows a programmer to focus on one area of functionality.

### **MODULES**

The Bus Booking Management Booking System contains four modules. The details of these modules are as follows:

#### **❖ USER REGISTRATION**

In this Module There is registration form available where new user can create their account by providing required information to the system. The registration form details are like name, email, password. These details are stored in the database. And then can getting to the username and password in the system.

#### **❖ ADMIN**

In this Module admin can be login to the account using the user's name and password. Then admin have responsibilities to maintain all information. Authorized user only can access this module.

#### **❖ INVENTORY MANAGEMENT**

In this Modules, this module is the core component of the system and includes functionalities such as adding new bus , updating existing bus information.

#### **❖ PAYMENT INTERGRATION**

In this Module Retailer can securely enter their credit or debit card information to authorize payment for booking. Payment gateways facilitate encryption and processing of card transactions, ensuring data security and compliance with industry standards.



## 4. SYSTEM DEVELOPMENT

System development is the process of code construction. It includes in the internal development of customized systems, the creation of database systems, or the acquisition of third party developed software.

### 4.1. PROGRAM DEVELOPMENT

#### BASE.HTML

```
<!doctype html>
<html lang="en">
<head>
<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
<title>Bus Reservation System in Django Framework</title>

<!-- jQuery first, then Bootstrap JS -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"
integrity="sha512-
894YE6QWD5I59HgZOGReFYm4dnWc1Qt5NtvYSaNcOP+u1T9qYdvdiZh0PPSiiqn/+/3e7J
o4EaG7TubfWGUrMQ==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
ka7Sk0GlIn4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
crossorigin="anonymous"></script>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-gradient bg-primary">
<div class="container">
<a class="navbar-brand" href="#" style="color: white;">Sankaran Liner Transit
Cporation</a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavAltMarkup"
```

```

aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNavAltMarkup">
<div class="navbar-nav" >
<a class="nav-item nav-link active" href="{% url 'home' %}" style="color: white;">Home</a>
<a class="nav-item nav-link" href="{% url 'findbus' %}" style="color: white;">Find Bus</a>
<a class="nav-item nav-link" href="{% url 'seebookings' %}" style="color: white;">See
Bookings</a>
{% if request.user.is_active %}
<a class="nav-item nav-link" href="{% url 'signout' %}" style="color: white;">Logout</a>
{% else %}
<a class="nav-item nav-link" href="{% url 'signup' %}" style="color:
white;">Registration</a>
{% endif %}

</div>
</div>
</div>
</nav>
{% block content %}
{% endblock %}

</body>
</html>

```

## SIGIN.HTML

```

{% extends 'myapp/base.html' %} {% block content %}
<div class="container py-3">
<div class="card bg-info bg-gradient">
<div class="card-body">
<h2 class="my-3 text-center text-light">Welcome to Sankaran Liner Transit Coporation Bus
Reservation</h2>
</div>
</div>
<div class="row justify-content-center mt-4">
<div class="col-lg-5 col-md-7 col-sm-10 col-xs-12">
<div class="card rounded-0">
<div class="card-header">
<div class="card-title h3 fw-bolde mb-0 text-center">Login</div>
</div>
<div class="card-body">
<div class="container-fluid">
<form action="{% url 'signin' %}" method="post" id="signin-form">
{% csrf_token %}
<div class="mb-3">
<label for="username" class="control-label">Username:</label>

```

```

<input name='name' class="form-control rounded-0" type="text" id="username" required>
</div>
<div class="mb-3">
<label for="example-email-input" class="control-label">Password:</label>
<input name='password' class="form-control rounded-0" type="password" required>
</div>

</form>
<p>{{error}}</p>
</div>
</div>
<div class="card-footer text-center">
<button class="btn btn-sm rounded-0 btn-primary btn-block w-100" form="signin-
form">Login</button>
</div>
</div>
</div>
</div>

{% endblock %}

```

## BOOKING.HTML

```
{% extends 'myapp/base.html' %} {% block content %}
<div class="container my-5">
  {% for message in messages %} {% if message.tags %}
  <div class="alert alert-{{ message.tags }}">{{ message }}</div>
  {% endif %} {% endfor %}
  <div class="card rounded-0">
    <div class="card-header">
    <div class="card-title h3 mb-0">List of Bookings</div>
    </div>
    <div class="card-body">
    <div class="container-fluid">
    <table class="table table-bordered table-striped">
    <thead>
    <tr class="bg-gradient bg-primary text-light">
    <th class="p-1 text-center">BOOKING ID</th>
    <th class="p-1 text-center">USER NAME</th>
    <th class="p-1 text-center">BUS NAME</th>
    <th class="p-1 text-center">SOURCE</th>
    <th class="p-1 text-center">DESTINATION</th>
    <th class="p-1 text-center">NUM OF SEAT</th>
    <th class="p-1 text-center">PRICE</th>
    <th class="p-1 text-center">DATE</th>
    <th class="p-1 text-center">TIME</th>
    <th class="p-1 text-center">STATUS</th>
    </tr>
    </thead>

    {% for row in book_list %}
    <tr>
    <td class="px-2 py-1 align-middle">{{ row.id }}</td>
    <td class="px-2 py-1 align-middle">{{ row.name }}</td>
    <td class="px-2 py-1 align-middle">{{ row.bus_name }}</td>
    <td class="px-2 py-1 align-middle">{{ row.source }}</td>
    <td class="px-2 py-1 align-middle">{{ row.dest }}</td>
    <td class="px-2 py-1 align-middle">{{ row.nos }}</td>
    <td class="px-2 py-1 align-middle">{{ row.price }}</td>
    <td class="px-2 py-1 align-middle">{{ row.date }}</td>
    <td class="px-2 py-1 align-middle">{{ row.time }}</td>
    <td class="px-2 py-1 align-middle">{{ row.status }}</td>
    </tr>
    {% endfor %}
    </table>
  </div>
</div>
<div class="row mt-3 justify-content-center">
<div class="col-lg-5 col-md-7 col-sm-10 col-xs-12">
<div class="card rounded-0">
<div class="card-header">
```

```

<div class="card-title h3 mb-0">Cancell Booking Form</div>
</div>
<div class="card-body">
<div class="container-fluid">
<form action="{% url 'cancellings' %}" method="post" id="cancel-form">
{% csrf_token %}
<div class="mb-3">
<label for="bus_id" class="control-label">Booking ID</label>
<input name='bus_id' class="form-control" type="number" id="bus_id">
</div>
{% if error %}
<div class="alert alert-danger">
{{error}}
</div>
{% endif %}

</form>
</div>
</div>
<div class="card-footer text-center">
<button class="btn btn-sm rounded-0 btn-danger w-100" form="cancel-form">Cancel
Booking</button>
</div>
</div>
</div>
</div>
</div>

{% endblock %}

```

## MANAGE.PY

```

from django import forms

from django.contrib.auth import (
    authenticate,
    get_user_model

)

```

```

User = get_user_model()

```

```

class UserLoginForm(forms.Form):

```

```

username = forms.CharField()
password = forms.CharField(widget=forms.PasswordInput)

def clean(self, *args, **kwargs):
    username = self.cleaned_data.get('username')
    password = self.cleaned_data.get('password')

    if username and password:
        user = authenticate(username=username, password=password)
        if not user:
            raise forms.ValidationError('This user does not exist')
        if not user.check_password(password):
            raise forms.ValidationError('Incorrect password')
        if not user.is_active:
            raise forms.ValidationError('This user is not active')
        return super(UserLoginForm, self).clean(*args, **kwargs)

class UserRegisterForm(forms.ModelForm):
    email = forms.EmailField(label='Email address')
    email2 = forms.EmailField(label='Confirm Email')
    password = forms.CharField(widget=forms.PasswordInput)

class Meta:
    model = User
    fields = [
        'username',
        'email',
        'email2',
        'password'
    ]

```

```

def clean(self, *args, **kwargs):
    email = self.cleaned_data.get('email')
    email2 = self.cleaned_data.get('email2')
    if email != email2:
        raise forms.ValidationError("Emails must match")
    email_qs = User.objects.filter(email=email)
    if email_qs.exists():
        raise forms.ValidationError(
            "This email has already been registered")
    return super(UserRegisterForm, self).clean(*args, **kwargs)

# Create your models here.
from django.db import models

# Create your models here.

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)
    date = models.DateField()
    time = models.TimeField()

    class Meta:
        verbose_name_plural = "List of Busses"

    def __str__(self):
        return self.bus_name

```

```

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)

```

```

class Meta:
    verbose_name_plural = "List of Users"

```

```

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_STATUSES, default=BOOKED, max_length=2)


class Meta:
    verbose_name_plural = "List of Books"

    def __str__(self):
        return self.email

    from datetime import datetime

    from django.contrib import messages
    from django.shortcuts import render
    from decimal import Decimal


# Create your views here.

    from django.shortcuts import render, redirect
    from django.http import HttpResponse, HttpResponseRedirect
    from .models import User, Bus, Book
    from django.contrib.auth import authenticate, login, logout
    from django.contrib.auth.models import User
    from .forms import UserLoginForm, UserRegisterForm
    from django.contrib.auth.decorators import login_required
    from decimal import Decimal


    def home(request):
        if request.user.is_authenticated:
            return render(request, 'myapp/home.html')
        else:
            return render(request, 'myapp/signin.html')


    @login_required(login_url='signin')
    def findbus(request):

```

```

context = {}

if request.method == 'POST':
    source_r = request.POST.get('source')
    dest_r = request.POST.get('destination')
    date_r = request.POST.get('date')
    date_r = datetime.strptime(date_r, "%Y-%m-%d").date()
    year = date_r.strftime("%Y")
    month = date_r.strftime("%m")
    day = date_r.strftime("%d")

    bus_list = Bus.objects.filter(source=source_r, dest=dest_r, date__year=year,
    date__month=month, date__day=day)

    if bus_list:
        return render(request, 'myapp/list.html', locals())
    else:
        context['data'] = request.POST
        context["error"] = "No available Bus Schedule for entered Route and Date"
        return render(request, 'myapp/findbus.html', context)
    else:
        return render(request, 'myapp/findbus.html')

```

```

@login_required(login_url='signin')
def bookings(request):
    context = {}

    if request.method == 'POST':
        id_r = request.POST.get('bus_id')
        seats_r = int(request.POST.get('no_seats'))
        bus = Bus.objects.get(id=id_r)

        if bus:
            if bus.rem > int(seats_r):
                name_r = bus.bus_name
                cost = int(seats_r) * bus.price
                source_r = bus.source

```

```

dest_r = bus.dest
nos_r = Decimal(bus.nos)
price_r = bus.price
date_r = bus.date
time_r = bus.time
username_r = request.user.username
email_r = request.user.email
userid_r = request.user.id
rem_r = bus.rem - seats_r
Bus.objects.filter(id=id_r).update(rem=rem_r)

book = Book.objects.create(name=username_r, email=email_r, userid=userid_r,
bus_name=name_r,
source=source_r, busid=id_r,
dest=dest_r, price=price_r, nos=seats_r, date=date_r, time=time_r,
status='BOOKED')
print('-----book id-----', book.id)
# book.save()

return render(request, 'myapp/bookings.html', locals())
else:
context["error"] = "Sorry select fewer number of seats"
return render(request, 'myapp/findbus.html', context)

else:
return render(request, 'myapp/findbus.html')

@login_required(login_url='signin')
def cancellings(request):
context = {}
if request.method == 'POST':
id_r = request.POST.get('bus_id')
#seats_r = int(request.POST.get('no_seats'))

```

```

try:
book = Book.objects.get(id=id_r)
bus = Bus.objects.get(id=book.busid)
rem_r = bus.rem + book.nos
Bus.objects.filter(id=book.busid).update(rem=rem_r)
#nos_r = book.nos - seats_r
Book.objects.filter(id=id_r).update(status='CANCELLED')
Book.objects.filter(id=id_r).update(nos=0)
messages.success(request, "Booked Bus has been cancelled successfully.")
return redirect(seebookings)
except Book.DoesNotExist:
context["error"] = "Sorry You have not booked that bus"
return render(request, 'myapp/error.html', context)
else:
return render(request, 'myapp/findbus.html')

```

```

@login_required(login_url='signin')
def seebookings(request,new={}):
context = {}
id_r = request.user.id
book_list = Book.objects.filter(userid=id_r)
if book_list:
return render(request, 'myapp/booklist.html', locals())
else:
context["error"] = "Sorry no buses booked"
return render(request, 'myapp/findbus.html', context)

```

```

def signup(request):
context = {}
if request.method == 'POST':

```

```

name_r = request.POST.get('name')
email_r = request.POST.get('email')
password_r = request.POST.get('password')
user = User.objects.create_user(name_r, email_r, password_r, )
if user:
    login(request, user)
    return render(request, 'myapp/thank.html')
else:
    context["error"] = "Provide valid credentials"
    return render(request, 'myapp/signup.html', context)
else:
    return render(request, 'myapp/signup.html', context)

```

```

def signin(request):
    context = {}
    if request.method == 'POST':
        name_r = request.POST.get('name')
        password_r = request.POST.get('password')
        user = authenticate(request, username=name_r, password=password_r)
        if user:
            login(request, user)
            # username = request.session['username']
            context["user"] = name_r
            context["id"] = request.user.id
            return render(request, 'myapp/success.html', context)
            # return HttpResponseRedirect('success')
        else:
            context["error"] = "Provide valid credentials"
            return render(request, 'myapp/signin.html', context)
        else:
            context["error"] = "You are not logged in"

```

```
return render(request, 'myapp/signin.html', context)
```

```
def signout(request):
```

```
    context = {}
```

```
    logout(request)
```

```
    context['error'] = "You have been logged out"
```

```
    return render(request, 'myapp/signin.html', context)
```

```
def success(request):
```

```
    context = {}
```

```
    context['user'] = request.user
```

```
    return render(request, 'myapp/success.html', context)
```

```
"""
```

Django settings for myproject project.

Generated by 'django-admin startproject' using Django 2.1.2.

For more information on this file, see

<https://docs.djangoproject.com/en/2.1/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/2.1/ref/settings/>

```
"""
```

```
import os
```

```
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
```

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

```

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'betsff(e3$v-n@gc53*afg^r36tiz$4#lujpxub1!(@zkxfl9'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp',
]

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 = 'myproject.urls'
```

```
TEMPLATES = [
```

```
{
```

```
'BACKEND': 'django.template.backends.django.DjangoTemplates',
```

```
'DIRS': [],
```

```
'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 = 'myproject.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases
```

```
DATABASES = {
```

```
'default': {
```

```
'ENGINE': 'django.db.backends.sqlite3',
```

```
'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
```

```
}
```



```

}

# Password validation
# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators

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',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/2.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

```

```
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/2.1/howto/static-files/
```

```
STATIC_URL = '/static/'
```

```
# Generated by Django 2.1.2 on 2018-12-17 11:38
```

```
from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
    initial = True
```

```
    dependencies = [
```

```
]
```

```
    operations = [
```

```
        migrations.CreateModel(
```

```
            name='Book',
```

```
            fields=[
```

```
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
```

```
                ('email', models.EmailField(max_length=254)),
```

```
                ('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=[('B', 'Booked'), ('C', 'Cancelled')], default='B',
max_length=2)),
],
),
migrations.CreateModel(
name='Bus',
fields=[
('id', models.AutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
('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)),
('date', models.DateField()),
('time', models.TimeField()),
],
),
migrations.CreateModel(
name='User',
fields=[
('user_id', models.AutoField(primary_key=True, serialize=False)),
('email', models.EmailField(max_length=254)),
('name', models.CharField(max_length=30)),
('password', models.CharField(max_length=30))
]

```

## 5. SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing is carried out by specialist's testers or independent testers. The system testing should investigate both functional and non-functional requirements of the testing.

### 5.1. PROJECT TESTING

The project testing is used to test the whole project by using the different of testing techniques. Testing is a series of different tests that whose primary purpose is fully exercise the computer-based system.

#### 5.1.1 UNIT TESTING:

Unit testing emphasizes the verification effort on the smallest unit of software design i.e.; a software component or module. Unit testing is a dynamic method for verification, where program is actually compiled and executed. Unit testing is performed parallel with the coding phase. The functionality of the modules was also tested as separate units. Each of the three modules was tested as separate units, in each module all the functionalities were tested in isolation.

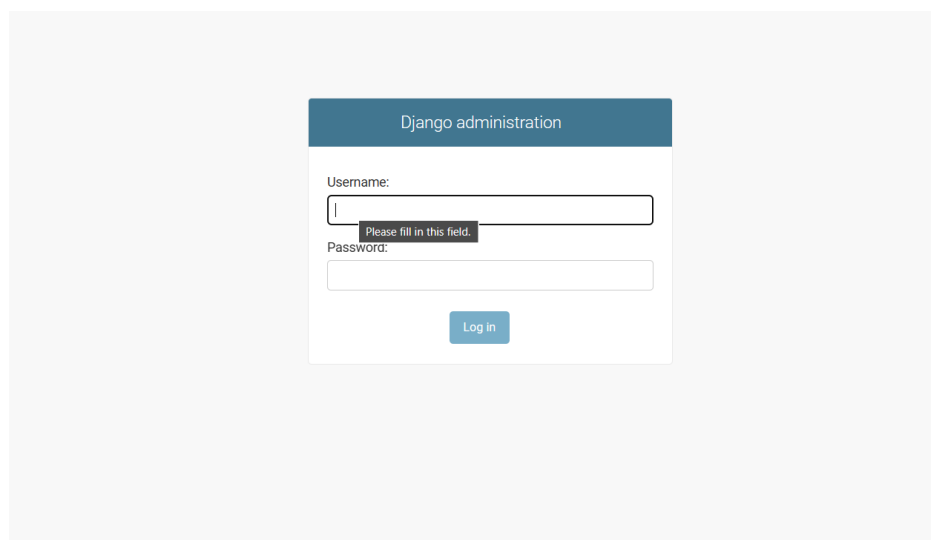
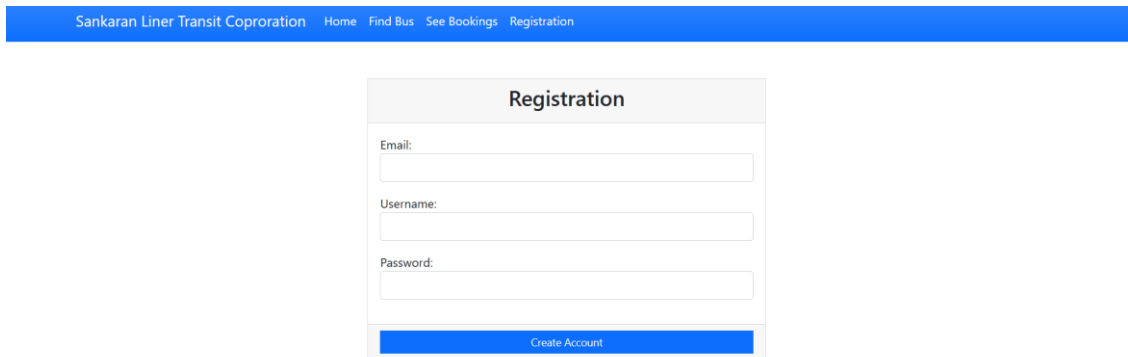


Figure 5.1 Unit Testing on Admin Page

## 5.2 INTEGRATION TESTING:

Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed system.

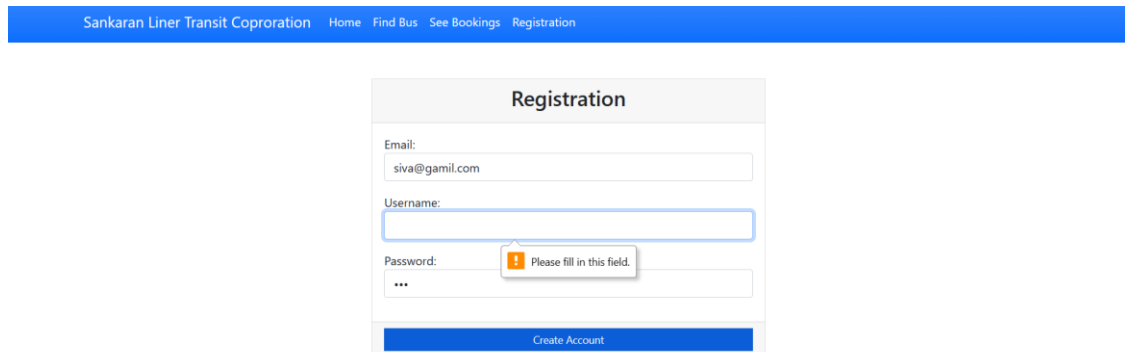


The image shows a web application interface for a bus transit corporation. At the top, there is a blue navigation bar with the text "Sankaran Liner Transit Coproration" and links for "Home", "Find Bus", "See Bookings", and "Registration". Below the navigation bar, there is a registration form. The form has a title "Registration" and three input fields: "Email:", "Username:", and "Password:". Each input field is a white box with a light gray border. At the bottom of the form, there is a blue button with the text "Create Account".

**Figure 5.2 Integration Testing on Register Page**

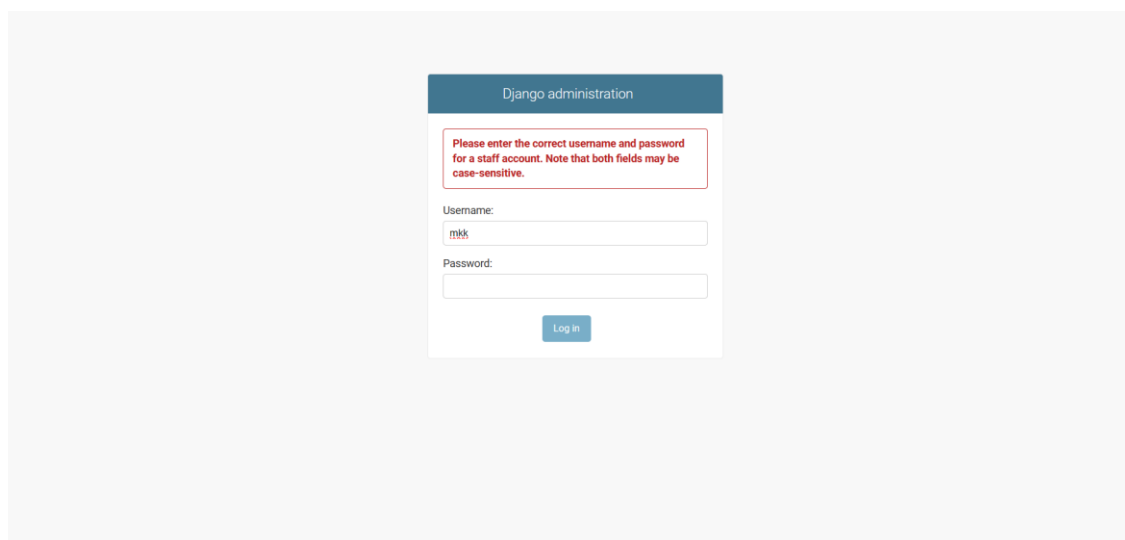
## 5.3 VALIDATION TESTING:

This model of testing is extremely important especially if you want to be one of the best software testers. The software verification and validation testing are the process after the validation testing stage is secondary to verification testing.



The screenshot shows a web application header with the text "Sankaran Liner Transit Coproration" and navigation links: "Home", "Find Bus", "See Bookings", and "Registration". Below the header is a "Registration" form. The form contains three input fields: "Email:" with the value "siva@gamil.com", "Username:" which is empty and highlighted with a blue border, and "Password:" with masked characters "..." and a tooltip that says "Please fill in this field." Below the fields is a blue button labeled "Create Account".

**Figure 5.3 Validation Testing for Register Page**



The screenshot shows a "Django administration" login page. At the top is a dark blue header with the text "Django administration". Below the header is a red-bordered box containing the message: "Please enter the correct username and password for a staff account. Note that both fields may be case-sensitive." Below this message are two input fields: "Username:" with the value "mkk" and "Password:" which is empty. At the bottom of the form is a blue button labeled "Log in".

**Figure 5.4 Validation Testing for Admin Page**

## **6. SYSTEM IMPLEMENTATION**

System implementation is the stage of the project when the theoretical design is turned into a working system. Thus, it can be considered the most critical stage in achieving a successful new system and in giving the user confidence that the new system will work and be effective.

### **6.1 SOFTWARE DEPLOYMENT**

Software deployment is the process of making the application work on a target device, whether it be a test server, a production environment, or a user's computer. Software deployment should only take place after thorough testing has ensured that all flaws and bugs have been identified and fixed. It is usually a planned initiative that consists of different steps or stages that occur in the production of operational software.

### **6.2 SOFTWARE DEMONSTRATION**

Demo software is a trial version of a software program which allows people to use it for free while they decide whether or not to buy it. For example, a program may not allow people to save files, meaning that people can use the program to see how it feels, but they cannot save the work they produce.

## 6.1 Home Page

❖ This Module, Manage the Home Page

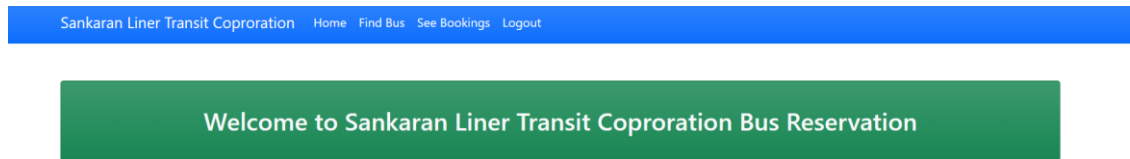


Figure 6.1 Home Page

## 6.2 Admin Login Page

❖ This Module, Manage the Admin Login Page

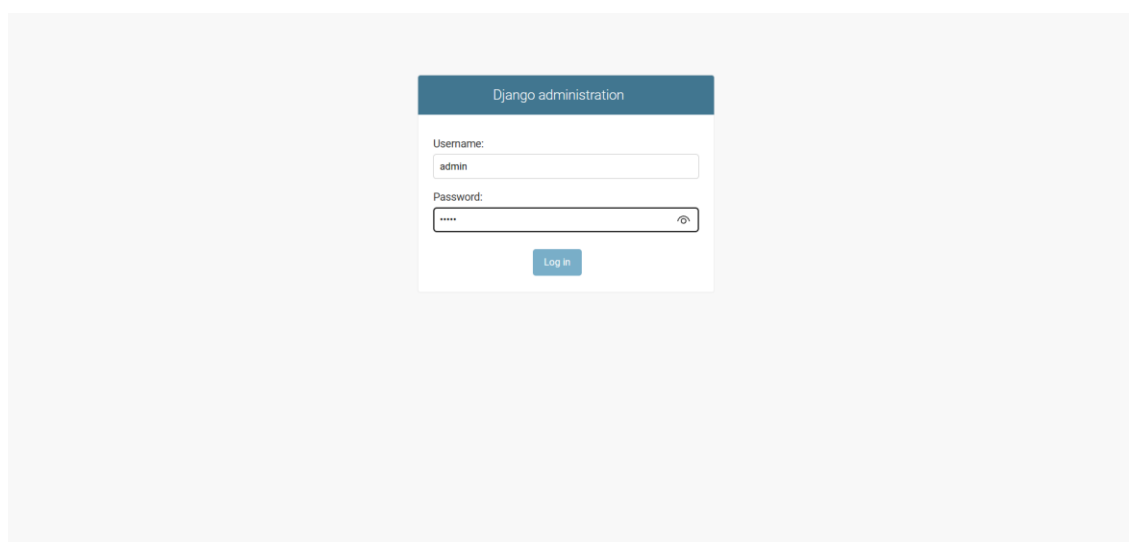
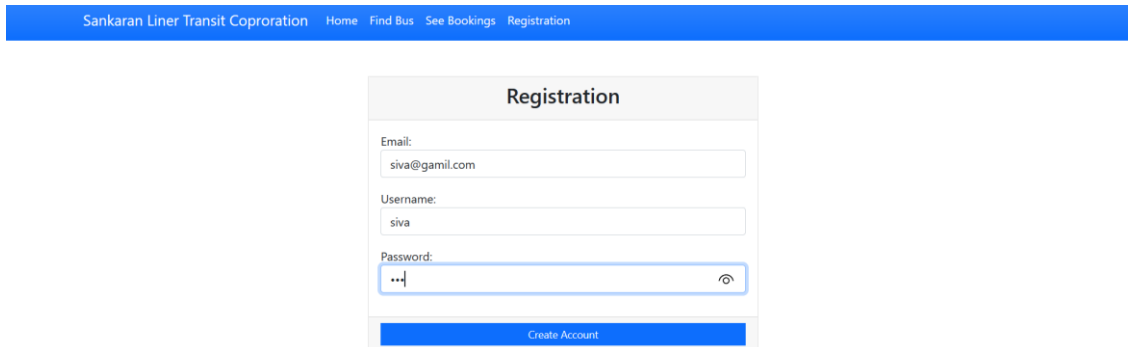


Figure 6.2 Admin Login Page



## 6.3 User Registration Page

❖ This Module, Manage the New User Register Page

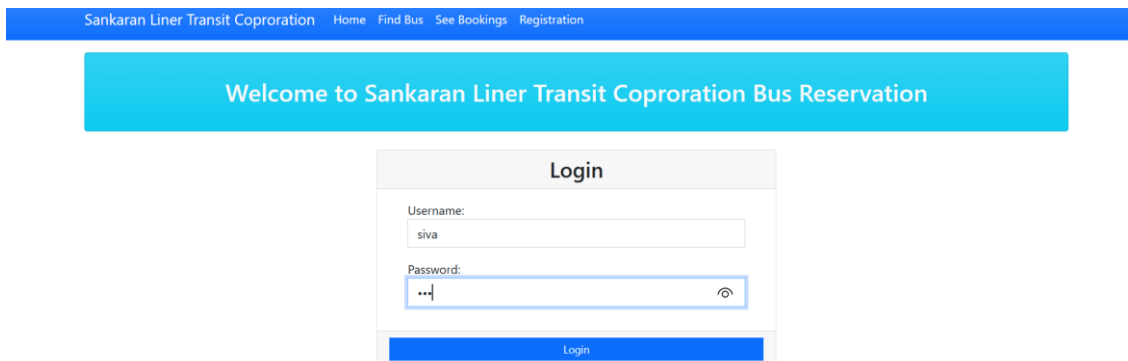


The screenshot shows the 'Registration' page of the Sankaran Liner Transit Corporation. At the top is a blue navigation bar with the text 'Sankaran Liner Transit Coproration' and links for 'Home', 'Find Bus', 'See Bookings', and 'Registration'. Below the navigation bar is a white registration form with a grey header 'Registration'. The form contains three input fields: 'Email:' with the value 'siva@gamil.com', 'Username:' with the value 'siva', and 'Password:' with masked characters '\*\*\*'. A blue 'Create Account' button is at the bottom of the form.

Figure 6.3 User Registration Page

## 6.4 User Login Page

❖ This Module, Manage the User Login Page



The screenshot shows the 'Login' page of the Sankaran Liner Transit Corporation. At the top is a blue navigation bar with the text 'Sankaran Liner Transit Coproration' and links for 'Home', 'Find Bus', 'See Bookings', and 'Registration'. Below the navigation bar is a large cyan banner with the text 'Welcome to Sankaran Liner Transit Coproration Bus Reservation'. Below the banner is a white login form with a grey header 'Login'. The form contains two input fields: 'Username:' with the value 'siva' and 'Password:' with masked characters '\*\*\*'. A blue 'Login' button is at the bottom of the form.

Figure 6.4 User Login Page

## 6.6 User Information Page

❖ This Module, Manage the Admin, User Information Page

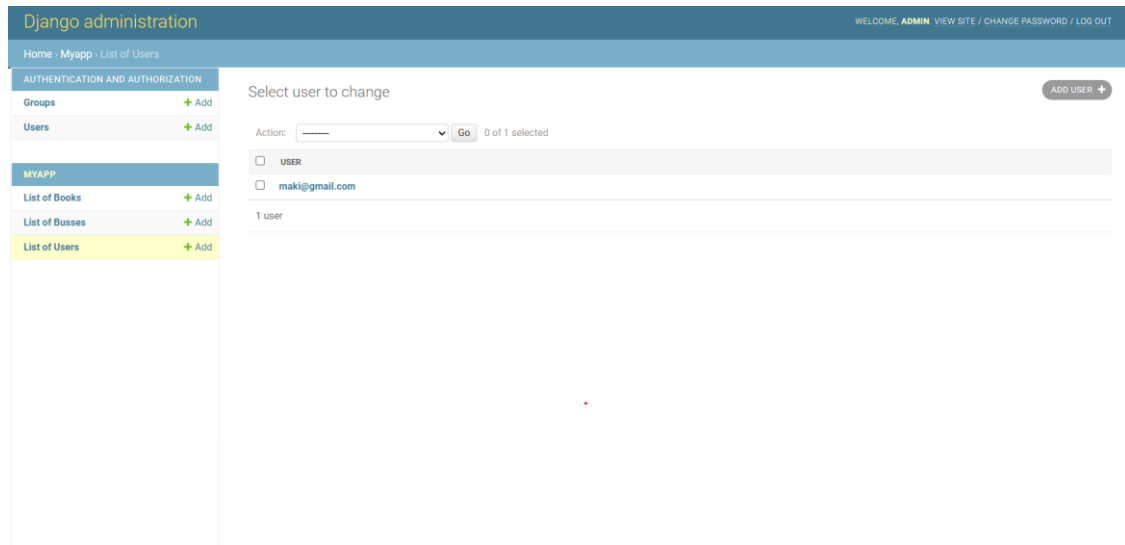


Figure 6.6 User Information Page

## 6.7 Bus Information Page

❖ This Module, Manage the Admin, Bus Information Page

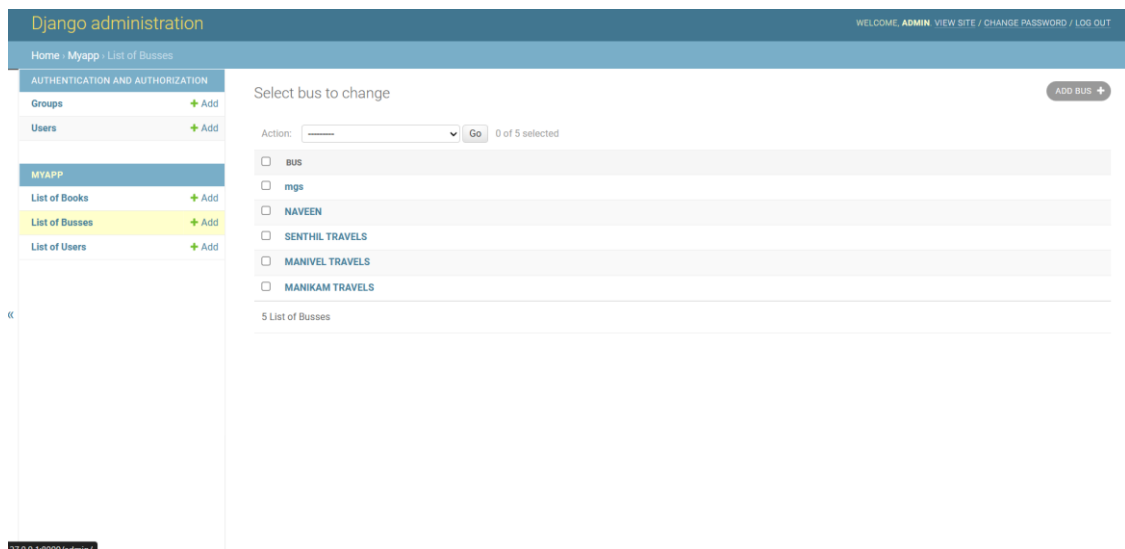


Figure 6.7 Bus Information Page

## 6.8 New Bus Registration Page

❖ This Module, Manage the Admin , New Bus Registration Page

The screenshot shows the Django administration interface for adding a new bus. The left sidebar contains navigation links for 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'MYAPP' (List of Books, List of Busses, List of Users). The main content area is titled 'Add bus' and contains a form with the following fields: 'Bus name' (text), 'Source' (text), 'Dest' (text), 'Nos' (text), 'Rem' (text), 'Price' (text), 'Date' (calendar icon, text '2024-03-21', and text 'Today'), and 'Time' (text '05:40:21', 'Now', and a clock icon). Below the form are three buttons: 'Save and add another', 'Save and continue editing', and 'SAVE'. The top of the page shows 'Django administration' and a user welcome message.

Figure 6.8 New Bus Register Page

## 6.9 Update Bus Information Page

❖ This Module, Manage the Admin, Update Bus Information Page

The screenshot shows the Django administration interface for updating an existing bus. The left sidebar is identical to the previous page. The main content area is titled 'Change bus' and contains a form with the following fields: 'Bus name' (text), 'Source' (text), 'Dest' (text), 'Nos' (text), 'Rem' (text), 'Price' (text), 'Date' (calendar icon, text '2024-03-08', and text 'Today'), and 'Time' (text '04:09:13', 'Now', and a clock icon). Below the form are four buttons: 'Delete', 'Save and add another', 'Save and continue editing', and 'SAVE'. The top of the page shows 'Django administration' and a user welcome message.

Figure 6.9 Update Bus Information Page

## 6.10 Product Search Page

❖ This Module, Manage the User, Bus Search Page

Sankaran Liner Transit Coproration   Home   Find Bus   See Bookings   Logout

### Find Bus

From  
tpj

Destination  
mm

Date  
21-03-2024

Find

Figure 6.10 Bus Search Page

## 6.11 Scheduled Page

❖ This Module, Manage the User, Scheduled Page

Sankaran Liner Transit Coproration   Home   Find Bus   See Bookings   Logout

### List of Scheduled Busses

ID	NAME	SOURCE	DESTINATION	NUM OF SEAT	NUM OF SEATS REM	PRICE	DATE	TIME
5	mgs	tpj	mm	50	25	250.00	March 21, 2024	5:42 a.m.

### Booking Form

#### Choose bus to book

Enter Bus ID  
5

Number of your Seat  
2

Book Now

Figure 6.11 Schedled Page

## 6.12 Payment Page

❖ This Module, Manage the User, Payment Page

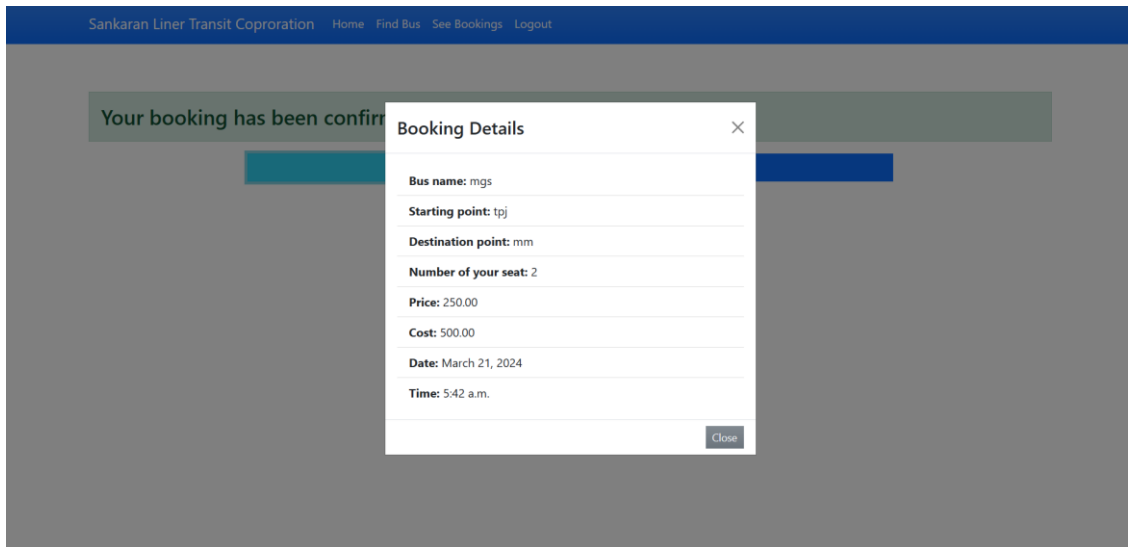


Figure 6.12 Payment Page

## **7. CONCLUSION**

The Bus Booking Management System project has successfully demonstrated the potential of digitalization in the travel industry. This system, implemented on a robust platform, has effectively managed bus scheduling and bookings, providing an integrated end-to-end solution from searching bus routes to booking them. Effective stock management is crucial in balancing the delicate equation of ensuring sufficient stock levels to meet customer demand while minimizing excess inventory that ties up valuable resources. A robust Stock Management System contributes to achieving this balance by providing features like real-time tracking, automated alerts, forecasting capabilities, and detailed reporting mechanisms.

The system has addressed the limitations of the previous manual system, such as the need for physical presence for ticket booking and lack of real-time seat availability information. By transitioning to a web-based application connected to a comprehensive database, it has enabled users to reserve tickets from any part of the world at any time.

Moreover, the system has given paramount importance to security and usability, ensuring a user-friendly and secure environment for transactions. It has also incorporated functionalities like route search, bus selection, seat selection, pick-up point selection, payment gateway integration, and ticket cancellation.

## **BIBLIOGRAPHY**

### **BOOK REFERENCES:**

- Van Rossum, Guido, and Fred L. Drake. The python language reference manual. Network Theory Ltd., 2011.
- Van Rossum, Guido, and Fred L. Drake. The python language reference manual. Network Theory Ltd., 2011.
- Dierbach, Charles. Introduction to Computer Science using Python: A Computational Problem-Solving Focus. Wiley Publishing, 2012.
- James, Mike. Programmer's Python: Everything is an Object Something Completely Different. I/O Press, 2018.
- Reges, Stuart, Marty Stepp, and Allison Obourn. Building Python Programs. Pearson, 2018.

#### **WEB REFERENCES:**

- <https://docs.python.org/3/tutorial/>
- <https://www.w3schools.com/python/>
- <https://www.tutorialspoint.com/python/index.htm>
- <https://www.programiz.com/python-programming>
- <https://www.learnpython.org/>