A

Project Report On

**Online Bus Reservation System** Submitted in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF ENGINEERING**

# IN

**COMPUTER SCIENCE AND ENGINEERING**

BY

E.Srinivas

(1602-19-733-307)

M Aarthi

(1602-19-733-309)

Under the guidance of Smt. B. Syamala

ASSISTANT PROFESSOR

Sri. I. Navakanth ASSISTANT PROFESSOR



Department of Computer Science & Engineering Vasavi College of Engineering.

(Affiliated to Osmania University) Ibrahimbagh, Hyderabad.

# TABLE OF CONTENTS

|  |  |
| --- | --- |
| **CONTENT** | **PAGE NO.** |
| 1. Acknowledgement | 03 |
| 2. Abstract | 04-05 |
| 3. SRS | 06 |
| 4. Project Description | 07 |
| 5. Implementation of Code | 08-10 |
| 6. Output Screenshots | 28-32 |
| 7. Technologies Used | 33 |
| 8. Conclusion | 35 |
| 9. References | 35 |

1. **ACKNOWLEDGEMENT**

We would like to express our heartfelt gratitude to Smt. B. Syamala, our project guide, for her valuable guidance and constant support, along with her capable instruction and persistent encouragement.

We are grateful to our Head of Department, Dr. T. Adilakshmi, for her steady support and the provision of every resource required for the completion of this project.

We would like to take this opportunity to thank our Principal, Dr. S. V. RAMANA, as well as the management of the institute, for having designed an excellent learning atmosphere.

# ABSTRACT

## OBJECTIVE:

* + 1. The main objective of this project is to provide the better work efficiency, security, accuracy, reliability, feasibility.
    2. The main objective of the proposed system” Bus Reservation System” is to eliminate the

manual reservation system.

## WORKING:

1. Online Bus Reservation System is a web based application that allows visitors to check bus ticket availability, buy bus ticket and pay the bus ticket online and cancel booking
2. This system is established for everyone user after gaining access from the administrator.
3. Bus Reservation System project in Python Django is a simple system developed in Django,SQLlite,Python which is designed to automate the purchase of online tickets through an easy online bus booking system.
4. You can manage/book reservations, client info, and passenger lists with the bus ticket reservation system’s Django admin, and book tickets easily via the Bus reservation Website.5) Django is a Python-based free and open-source web framework that follows the model–template–views architectural pattern.

**Features:**

* + **Admin module**

**User** – For the user, The admin can add, update, and delete user information and also the admin can change the password of the user.

**Manage Booking –** For the booking, The admin can add, update, and delete booking information.

**Bus –** For the bus, The admin can add, update, and delete booking information.

**User module**

1. Log in – For the login, the user will login first before he/she can use the system.
2. Sign up – For the sign up, the user will sign up first before he/she can use to login in the system 3.Homepage – For the homepage, you will be able to all the basic access in the whole system

Such as home, find bus, see booking, and login

4.Search Bus – For the search bus, the user will be able to search the available bus. 5.View Booking – For the view booking, the user will be able to view the booking details. 6.Cancel Booking – For the cancel booking, the user will be able to cancel there booking.

## Installation Steps

Install Python 3.7

Install Django Framework 2.0

Run the project: python manage.py runserver.

## .SCOPE:

This application can be easily implemented under various situations. We can add new features as and when we require. Reusability is possible as and when require in this application. There is flexibility in all the modules.

## Extensibility:

This software is extendable in ways that its original developers may not expect. The following principles enhances extensibility like hide data structure, avoid traversing multiple links or methods, avoid case statements on object type and distinguish public and private operations.

## Reusability:

Reusability is possible as and when require in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort over several designs. Reducing the amount of code also simplifies understanding, which increases the likelihood that the code is correct. We follow up both types of reusability: Sharing of newly written code within a project and reuse of previously written code on new projects.

## Understandability:

A method is understandable if someone other than the creator of the method can understand the code(as well as the creator after a time lapse). We use the method, which small and coherent helps to accomplish this.

## Cost- effectiveness:

Its cost is under the budget and make within given time period. It is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy the entire requirement. Scope of this document is to put down the requirements, clearly identifying the information needed by the user, the source of the information and outputs expected from the system.

# Software Requirement Specifications

A Software Requirements Specification (SRS) is a complete set of information about the system on which the developed project will be running. It includes all the hardware as well as the recommended system requirement for running the software is also mentioned in detail separately. The aim of this document is to gather and analyse and give in-depth insight of the complete software requirement of the Quiz Application

* 1. Hardware Required: • Processor: Intel i3 or Above
     + RAM: 2GB or above
     + Hard Disk: 1 GB or above
     + Input Devices: Keyboard, Mouse
     + Output Devices: Monitor
  2. Software Required: • Operating System: Linux, Ubuntu, Mac, Windows 10
     + Frontend: HTML, CSS, Bootstrap
     + Backend: Python
     + Framework: Django
     + Web browser
     + IDE: PyCharm, VS Code

# Project Database and Tables

Database is critical for all businesses. A good database does not allow any form of anomalies and stores only relevant information in an ordered manner. If a database has anomalies, it is affecting the efficiency and data integrity. For example, delete anomaly arise upon the deletion of a row which also forces other useful data to be lost. As such, the tables need to be normalized. This fulfils the last objective of ensuring data are accurate and retrieved correctly.

Database files are the key source of information into the system. It is the process of designing database files, which are the key source of information to the system. The files should be properly designed and planned for collection, accumulation, editing and retrieving the required information.

The organization of data in database aims to achieve three major objectives:-

* Data integration
* Data integrity
* Data independency.

# IMPLEMENTATION OF CODE

## Online RESERVATION FOLDER

### SETTINGS.PY:

import os

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath( file ))) SECRET\_KEY = 'betsff(e3$v-n@gc53\*afg^r36tiz$4#lujpjxub1!(@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

### URLS.PY:

from django.urls import path from . import views urlpatterns = [

path('', views.home, name="home"), path('findbus', views.findbus, name="findbus"),

path('bookings', views.bookings, name="bookings"), path('cancellings', views.cancellings, name="cancellings"), path('seebookings', views.seebookings, name="seebookings"), path('signup', views.signup, name="signup"),

path('signin', views.signin, name="signin"), path('success', views.success, name="success"), path('signout', views.signout, name="signout"),

### VIEWS.PY:

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')

bus\_list = Bus.objects.filter(source=source\_r, dest=dest\_r, date=date\_r) if bus\_list:

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

context["error"] = "Sorry no buses availiable"

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)

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)

WSGI.PY:

import os

from django.core.wsgi import get\_wsgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'myproject.settings') application = get\_wsgi\_application()

### Models.py

# 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()

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)

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)

def str (self): return self.email

### Admin.py

from django.contrib import admin

from .models import Bus, User, Book

# Register your models here.

admin.site.register(Bus) admin.site.register(User) admin.site.register(Book)

#### form.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)

Views.py

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')

bus\_list = Bus.objects.filter(source=source\_r, dest=dest\_r, date=date\_r) if bus\_list:

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

context["error"] = "Sorry no buses availiable"

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)

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)

#### manage.py

import os import sys

if name == ' main ': os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'myproject.settings') try:

from django.core.management import execute\_from\_command\_line except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and " "available on your PYTHONPATH environment variable? Did you " "forget to activate a virtual environment?"

) from exc execute\_from\_command\_line(sys.argv)

## HTM FILES:

### Quiz 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://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"

crossorigin="anonymous">

<title>Bus Reservation System</title>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-success">

<div class="container">

<a class="navbar-brand" href="#" style="color: white;">Hyrawath Transit Incorporated</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 <span class="sr- only">(current)</span></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;">Sign out</a>

{% else %}

<a class="nav-item nav-link " href="{% url 'signup' %}" style="color: white;">Sign Up</a>

{% endif %}

</div>

</div>

</div>

</nav>

{% block content %}

{% endblock %}

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"

integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"

integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></script>

</body>

</html>

#### Bookings .html

{% extends 'myapp/base.html' %}

{% block content %}

<h2>Booking Confirmation</h2>

<form action="{% url 'home' %}" method="post">

{% csrf\_token %}

<h2>Your booking has been confirmed!</h2>

<h2>Thank you!</h2>

<h3>Bill details</h3>

<!-- Button to Open the Modal -->

<button type="button" class="btn btn-success" data-toggle="modal" data-target="#myModal"> Bill details

</button>

<!-- The Modal -->

<div class="modal" id="myModal">

<div class="modal-dialog">

<div class="modal-content">

<!-- Modal Header -->

<div class="modal-header">

<h4 class="modal-title">Modal Heading</h4>

<button type="button" class="close" data-dismiss="modal">&times;</button>

</div>

<!-- Modal body -->

<div class="modal-body">

<ul class="list-group list-group-flush">

<li class="list-group-item"><b>Bus name:</b> {{book.bus\_name}}</li>

<li class="list-group-item"><b>Starting point:</b> {{book.source}}</li>

<li class="list-group-item"><b>Destination point:</b> {{book.dest}}</li>

<li class="list-group-item"><b>Number of seats:</b> {{book.nos}}</li>

<li class="list-group-item"><b>Price:</b> {{book.price}}</li>

<li class="list-group-item"><b>Cost:</b> {{cost}}</li>

<li class="list-group-item"><b>Date:</b> {{book.date}}</li>

<li class="list-group-item"><b>Time:</b> {{book.time}}</li>

</ul>

</div>

<!-- Modal footer -->

<div class="modal-footer">

<button type="button" class="btn btn-success" data-dismiss="modal">Close</button>

</div>

</div>

</div>

</div>

<div class="pull-right">

<button type="submit" class="btn btn-primary float-right">OK</button>

</div>

</form>

{% endblock %}

#### Bookinglist.html

{% extends 'myapp/base.html' %}

{% block content %}

<h3>{{msg}}</h3>

<h2>List of buses</h2>

<table class="table table-striped">

<thead style="background-color: blue;color: white;">

<td>BOOKING ID</td>

<td>USER NAME</td>

<td>BUS NAME</td>

<td>SOURCE</td>

<td>DESTINATION</td>

<td>NUM OF SEATS</td>

<td>PRICE</td>

<td>DATE</td>

<td>TIME</td>

<td>STATUS</td>

</thead>

{% for row in book\_list %}

<tr>

<td>{{row.id}}</td>

<td>{{row.name}}</td>

<td>{{row.bus\_name}}</td>

<td>{{row.source}}</td>

<td>{{row.dest}}</td>

<td>{{row.nos}}</td>

<td>{{row.price}}</td>

<td>{{row.date}}</td>

<td>{{row.time}}</td>

<td>{{row.status}}</td>

</tr>

{% endfor %}

</table>

<form action="{% url 'cancellings' %}" method="post">

<h3>Choose bus to book</h3>

{% csrf\_token %}

<div class="col-auto">

<label for="example-email-input" class="col-2 col-form-label">Bus ID</label>

<div class="col-5">

<input name='bus\_id' class="form-control" type="number" id="example-email-input">

</div>

</div>

<br>

<br>

<div class="pull-right">

<button type="submit" class="btn btn-danger float-left">Cancel bus</button>

</div>

{{error}}

</form>

{% endblock %}

#### Error.html

{% extends 'myapp/base.html' %}

{% block content %}

<div class="container">

<div class="row">

<div class="col-sm-6 mx-auto" style="margin-top: 70px">

<h1>{{error}}</h1>

</div>

</div>

</div>

{% endblock %}

#### Find bus.html

{% extends 'myapp/base.html' %}

{% block content %}

<div class="container">

<div class="row">

<div class="col-sm-6 mx-auto" style="margin-top: 70px">

<h2>Find bus</h2>

<form action="{% url 'findbus' %}" method="post">

{% csrf\_token %}

<div class="form-group row">

<label for="example-email-input" class="col-2 col-form-label">From</label>

<div class="col-10">

<input name='source' class="form-control" type="text" id="example-email-input">

</div>

</div>

<div class="form-group row">

<label for="example-email-input" class="col-2 col-form-label">Destination</label>

<div class="col-10">

<input name='destination' class="form-control" type="text">

</div>

</div>

<div class="form-group row">

<label for="example-email-input" class="col-2 col-form-label" >Date</label>

<div class="col-10">

<input name='date' class="form-control" type="date">

</div>

</div>

<div class="pull-right">

<button type="submit" class="btn btn-success float-right">Find bus</button>

</div>

{{error}}

</form>

</div>

</div>

</div>

#### Home.html

{% extends 'myapp/base.html' %}

{% block content %}

<center><h1>Welcome to Bus Reservation System</h1></center>

{% endblock %}

#### List.html

{% extends 'myapp/base.html' %}

{% block content %}

<h3>{{msg}}</h3>

<h2>List of buses</h2>

<table class="table table-striped" >

<thead style="background-color: blue;color: white;">

<td>ID</td>

<td>NAME</td>

<td>SOURCE</td>

<td>DESTINATION</td>

<td>NUM OF SEATS</td>

<td>NUM OF SEATS REM</td>

<td>PRICE</td>

<td>DATE</td>

<td>TIME</td>

</thead>

{% for row in bus\_list %}

<tr>

<td>{{row.id }}</td>

<td>{{row.bus\_name}}</td>

<td>{{row.source}}</td>

<td>{{row.dest}}</td>

<td>{{row.nos}}</td>

<td>{{row.rem}}</td>

<td>{{row.price}}</td>

<td>{{row.date}}</td>

<td>{{row.time}}</td>

</tr>

{% endfor %}

</table>

<form action="{% url 'bookings' %}" method="post">

<h3>Choose bus to book</h3>

{% csrf\_token %}

<div class="col-auto">

<label for="example-email-input" class="col-2 col-form-label">Bus ID</label>

<div class="col-5">

<input name='bus\_id' class="form-control" type="number" id="example-email-input">

</div>

</div>

<div class="col=auto">

<label for="example-email-input" class="col-2 col-form-label">Number of seats</label>

<div class="col-5">

<input name='no\_seats' class="form-control" type="number">

</div>

</div>

<br>

<br>

<div class="pull-right">

<button type="submit" class="btn btn-success float-left">Book bus</button>

</div>

{{error}}

</form>

{% endblock %}

#### Signin.html

{% extends 'myapp/base.html' %}

{% block content %}

<div class="container">

<center><h2 style="margin-top: 70px; background-color: blue;color: white">Welcome to Hyrawath Transit Bus Reservation</h2></center>

<div class="row">

<div class="col-sm-6 mx-auto" style="margin-top: 70px">

<h2>Log in</h2>

<form action="{% url 'signin' %}" method="post">

{% csrf\_token %}

<div class="form-group row">

<label for="example-email-input" class="col-2 col-form-label">Username:</label>

<div class="col-10">

<input name='name' class="form-control" type="text" id="example-email-input">

</div>

</div>

<div class="form-group row">

<label for="example-email-input" class="col-2 col-form-label">Password:</label>

<div class="col-10">

<input name='password' class="form-control" type="password">

</div>

</div>

<div class="pull-right">

<button type="Submit" class="btn btn-success float-right">Sign in</button>

</div>

</form>

<p>{{error}}</p>

</div>

</div>

</div>

{% endblock %}

#### Signup.html

{% extends 'myapp/base.html' %}

{% block content %}

<div class="container">

<div class="row">

<div class="col-sm-6 mx-auto" style="margin-top: 70px">

<form action="{% url 'signup' %}" method="post">

<h2>Sign up!</h2>

{% csrf\_token %}

<div class="form-group row">

<label for="example-email-input" class="col-2 col-form-label">Email:</label>

<div class="col-10">

<input name='email' class="form-control" type="text" id="example-email-input" required>

</div>

</div>

<div class="form-group row">

<label for="example-email-input" class="col-2 col-form-label">Username:</label>

<div class="col-10">

<input name='name' class="form-control" type="text" required>

</div>

</div>

<div class="form-group row">

<label for="example-email-input" class="col-2 col-form-label">Password:</label>

<div class="col-10">

<input name='password' class="form-control" type="text" required>

</div>

</div>

<div class="pull-right">

<button type="submit" class="btn btn-success float-right">Sign up</button>

</div>

</form>

</div>

</div>

</div>

{% endblock %}

#### Success.html

{% extends 'myapp/base.html' %}

{% block content %}

<div class="container">

<div class="row">

<div class="col-sm-6 mx-auto" style="margin-top: 70px">

<form action="{% url 'signout' %}" method="post">

{% csrf\_token %}

{{user}} successful login! {{id}}

<div class="pull-right">

<button type="Submit" class="btn btn-success float-right">Sign out</button>

</div>

</form>

</div>

</div>

</div>

{% endblock %}

#### Thank.html

{% extends 'myapp/base.html' %}

{% block content %}

<div class="container">

<div class="row">

<div class="col-sm-6 mx-auto" style="margin-top: 70px">

<h1>Thank you for signing up!</h1>

</div>

</div>

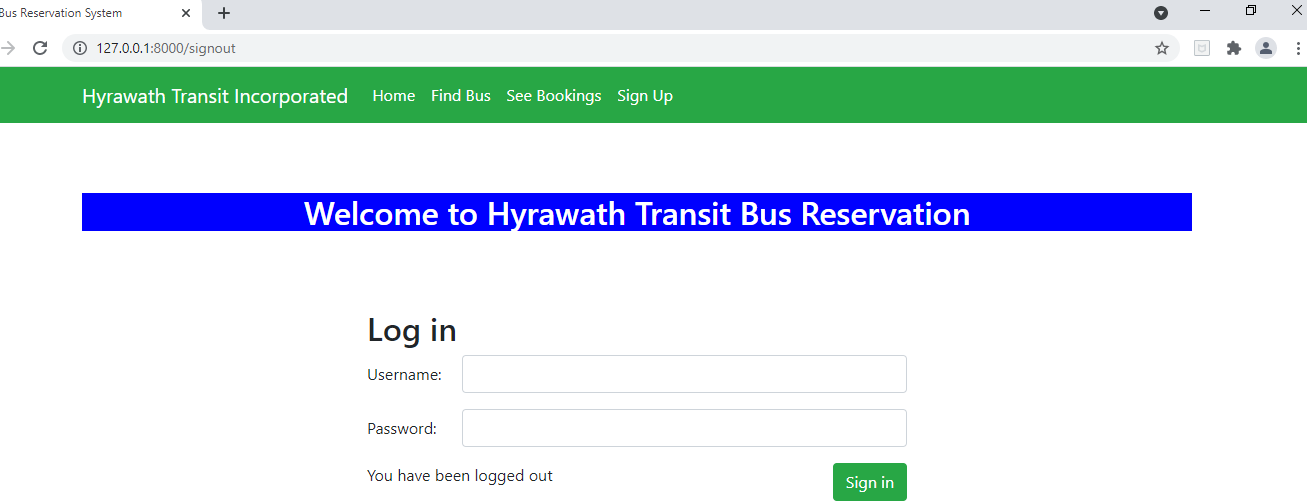
</div>

{% endblock %}

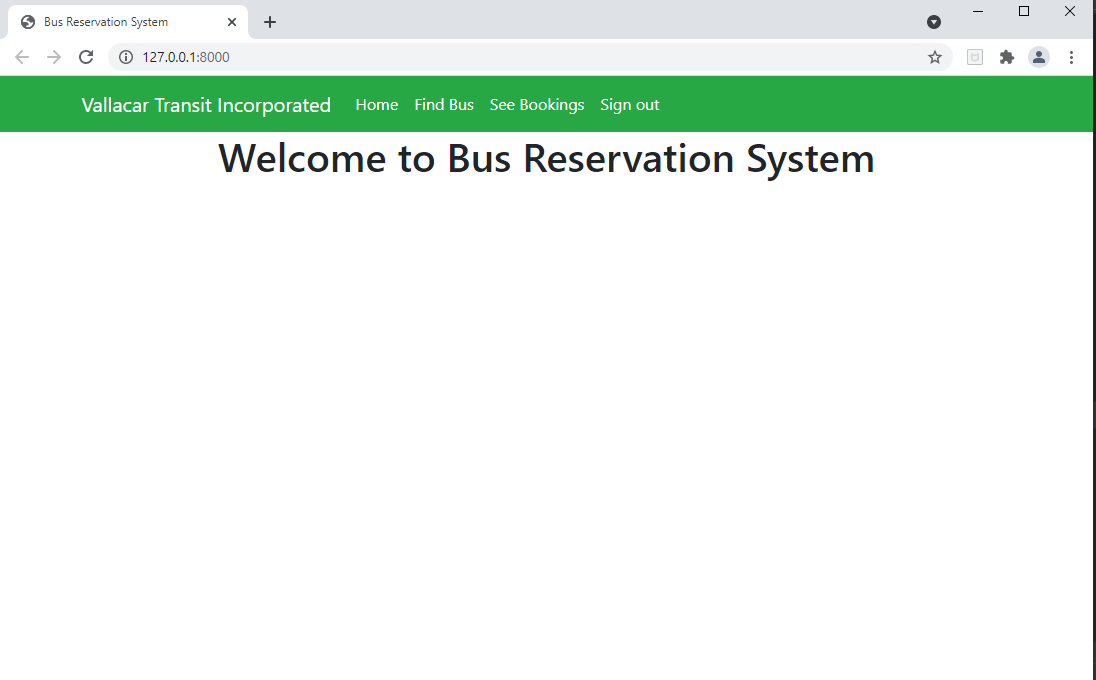
**OUTPUT SCREENSHOTS**

# USER MODULE

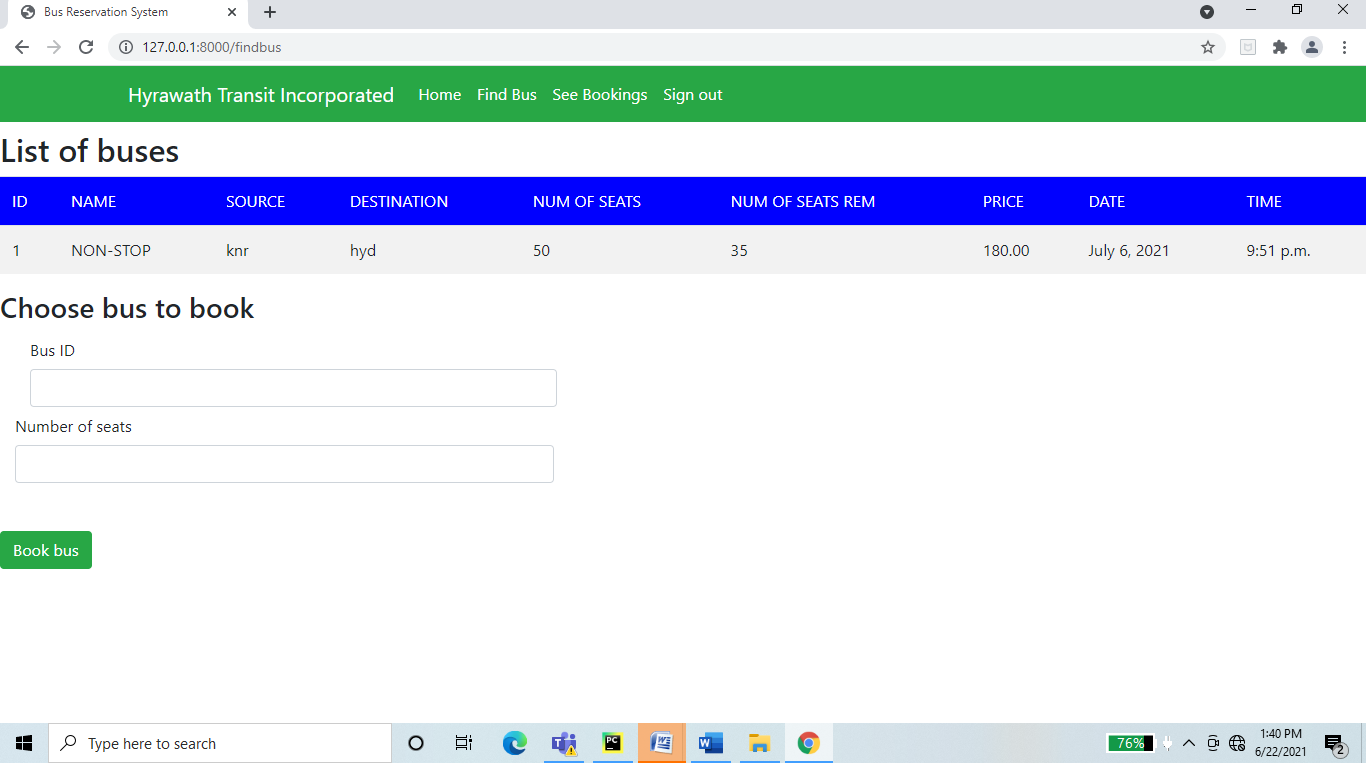
1. **USER Login Page**



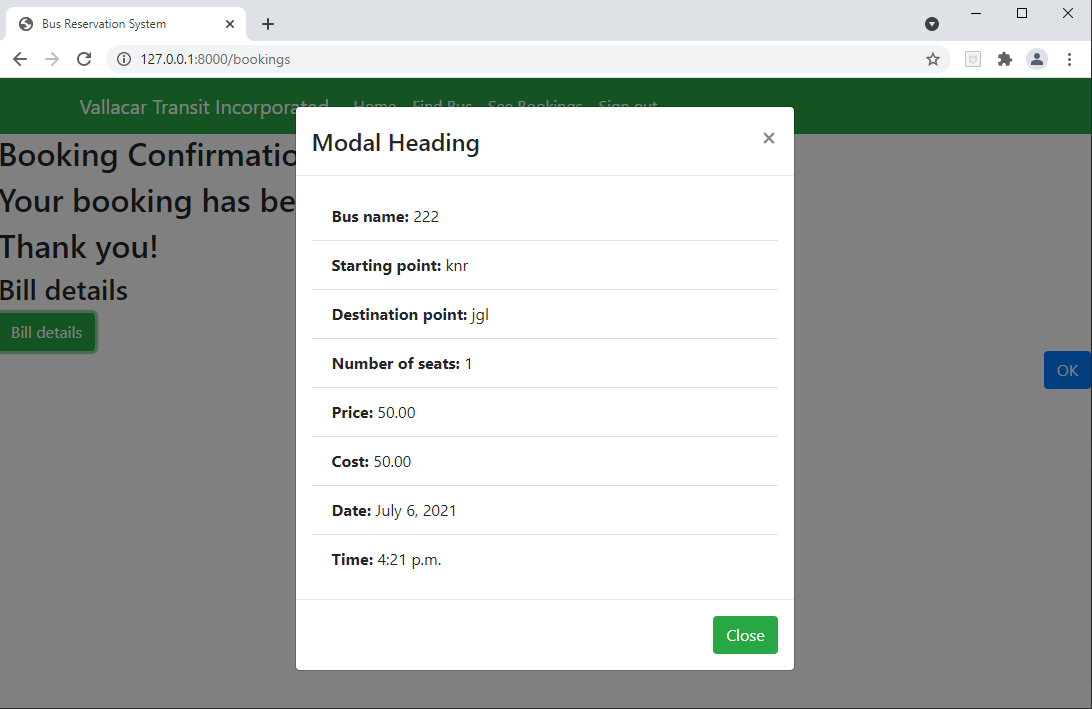
## Dashboard



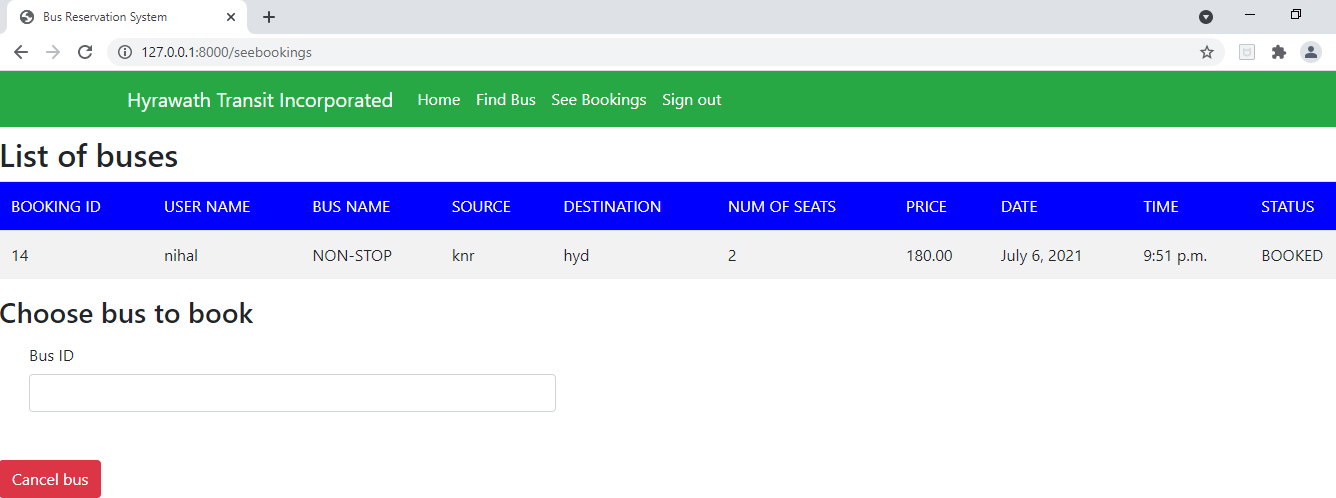
1. **It shows list of buses available along the root.**



## After selecting select book bus button then billing receipt shown.



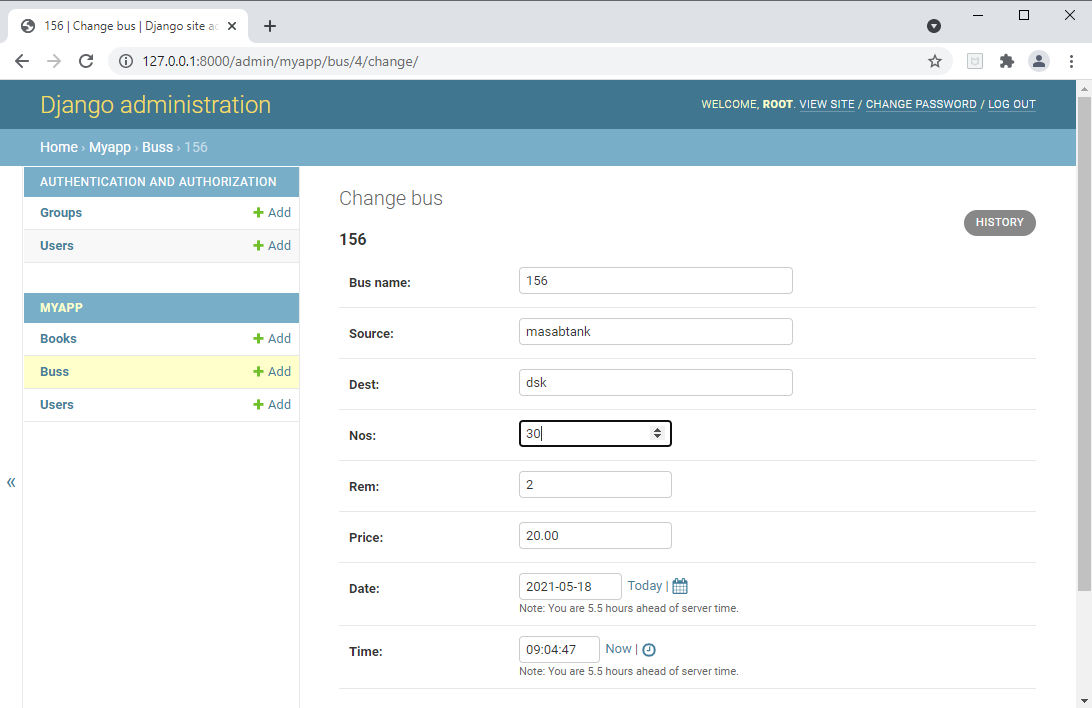
1. **After selecting see booking the buses booked by the user is displayed. user can can also cancel bus.**



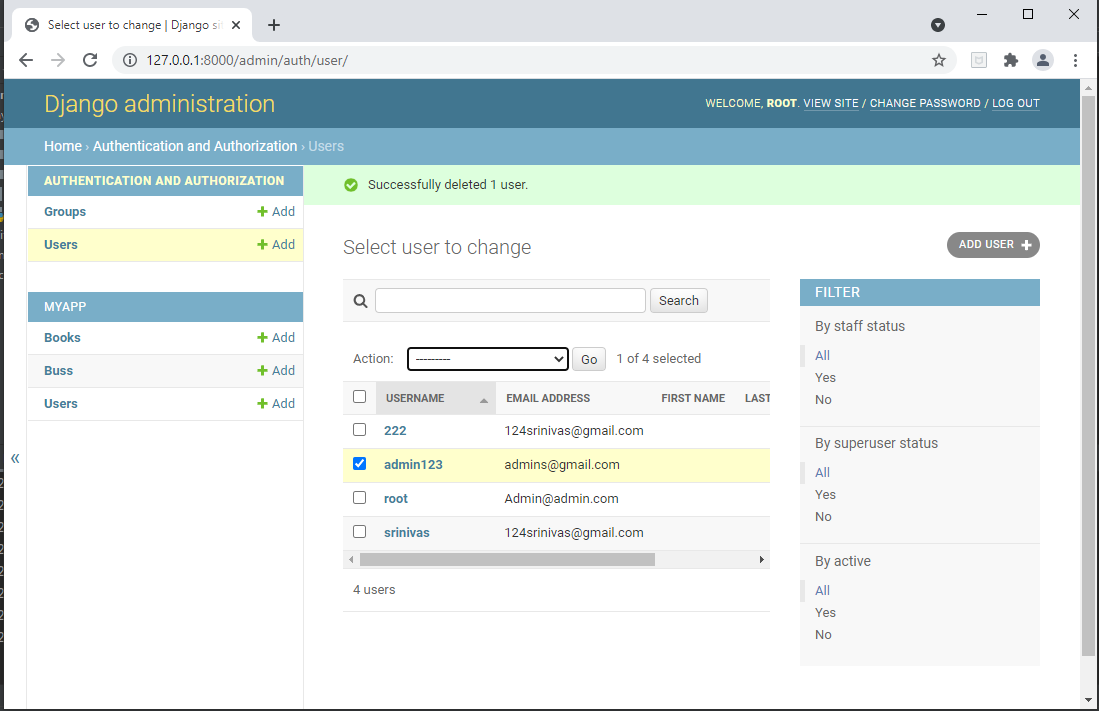
# ADMIN MODULE

## ADMIN LOGIN:

1. **ADMIN CAN BUSES:**



## ADMIN CAN CHANGE USERS:



TECHNOLOGIES USED

## Tools

* + HTML
  + CSS
  + Bootstrap
  + Django
  + SQLite

1. **HTML**

Every webpage you look at is written in a language called HTML. You can think of HTML as the skeleton that gives every webpage structure. In this course, we will use HTML to add paragraphs, headings, images, and links to a webpage. In the editor to the right, there is a tab called test.html. This is the file we will type our HTML into. Like any language, it has its own special syntax. A browser's job is to transform the code in test.html into a recognizable webpage! It knows how to lay out the page by following the HTML syntax.

### CSS

ICascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .CSS file, and reduce complexity and repetition in the structural content.

### Bootstrap

Bootstrap is a potent front-end framework used to create modern websites and web apps. It's open- source and free to use yet features numerous HTML and CSS templates for UI interface elements such as forms typography, forms, buttons, navigation, and other interface components. Bootstrap also supports JavaScript extensions. Responsive design makes it possible for a web page or app to detect the visitor’s screen size and orientation and automatically adapt the display; accordingly, the mobile first approach

assumes that smartphones, tablets and task-specific Mobile apps are employees' primary tools for getting work done and addresses the requirements of those technologies in design. Bootstrap includes user interface components, layouts, and JS tools along with the framework for implementation. The software is available precompiled or as source code.

### Django

Django (named after the Django Reinhardt) is a high-level python-based free and open-source web framework that follows the model-view-template (MVT) architectural pattern. It is slightly different from the MVC pattern as it maintains its own conventions, so, the controller is handled by the framework itself. The template is a presentation layer. It is an HTML file mixed with Django Template Language (DTL). The developer provides the model, the view, and the template then maps it to a URL, and finally, Django serves it to the user.

Django can be broken into many components:

Models.py file: This file defines your data model by extending your single line of code into full database tables and add a pre-built administration section to manage content.

Urls.py file: It uses a regular expression to capture URL patterns for processing. Views.py file: It is the main part of Django. The actual processing happens in view.

When a visitor lands on Django page, first Django checks the URLs pattern you have created and used the information to retrieve the view. After that view processes the request, querying your database if necessary, and passes the requested information to a template.

### SQLite

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others.

SQLite is a compact library. With all features enabled, the library size can be less than 600KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used, SQLite can be faster than direct filesystem I/O.

# CONCLUSION

By using this application we can book bus and cancel the bus.it is easy handle the booklist belongs to user available.so the user can easy to handle it.And information of every user is provided security with password. Now adays all transactions going in online. So users require more features in every website so it helps in bringing more features.

# REFERENCES

* https://docs.djangoproject.com, Django Documentation
* https://getbootstrap.com, Bootstrap
* https://developer.mozilla.org/en-US/docs/Web/HTML, HTML Documentation
* https://[www.sqlite.org/docs.html,](http://www.sqlite.org/docs.html) SQLite Documentation