



Train Ticketing System

Informatics Practices

**Chandrasekhar &
Aaditya Kumar
Muktavarapu**

Investigatory Project

—

CHIREC International School

—

12 A

Certificate

Acknowledgement

We, Chandrasekhar and Aaditya Kumar Muktavarapu, of class 12 A of CHIREC International School would first like to thank the Central Board of Secondary Education (CBSE) for giving us this opportunity to create a software project 'Train Ticketing System - 2020'.

Next, we would like to express our special gratitude to our teacher Ms. Naseema as well as our Principal Ms. Iffat Ibrahim who gave us the golden opportunity to work on this project which encouraged us to expand our knowledge and learn many new things.

Lastly, we would like to thank our parents and friends who have helped us finish this project through their undying support and encouragement.

INDEX

S No	Title	Page No
1	Abstract	5
2	Hardware Requirements	6
3	Software Requirements	7
4	Project Code and Screens 1. Project Directory 2. Trains-settings.py 3. Login/Signup Screen 4. Home Page 5. Log Out 6. Reset Password Page 7. Booking Page 8. Booking History Page 9. Graph Page	12
5	MySQL Tables	52
6	Future Extensions	55
7	Conclusion	56
8	References	57

Abstract

The Train Ticketing System is an online webpage created using Django framework. It is a user-friendly site that is used to book train tickets.

The aim of this project is to be able to book train tickets by entering the source and destination along with the travel date, being able to view all the tickets a user has booked and cancelling any tickets by using databases and thus reduce any errors which comes from manually booking tickets.

Users can:

1. Create accounts by entering the necessary details in the registration page if the account does not exist.
2. Change passwords of already existing account.
3. Book tickets by entering source, destination, travel date and number of tickets in the booking page.
4. View booking history of the user through the history page.
5. Cancel booked tickets at the latest till before the departure date.

The application also ensures efficient security and non-redundancy of data and has effective data handling mechanisms.

Hardware Requirements

- Windows 7 or 10

OR

Mac OS X 10.11 or higher, 64-bit

OR

Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu)

- x86 64-bit CPU (Intel / AMD architecture)
- 4 GB RAM
- 5 GB free disk space
- Cores: Single (Dual/Quad Core is recommended)
- RAM: 4 GB (6 GB recommended)
- Display Resolution: 1280×1024 is recommended, 1024×768 is minimum.

Software Requirements

Mysql

The name is a combination of “My”, the name of co-founder Michael Widenius’ daughter, and “SQL”, the abbreviation for Structured Query Language and is pronounced as either “My S-Q-L” or “My Sequel”.

It is an open-source relational database management system. It is based on the structured query language (SQL), which is used for adding, removing, and modifying information in the database. Standard SQL commands, such as ADD, DROP, INSERT and UPDATE can be used with MySQL.

MySQL can be used for a variety of applications, but is most commonly found on Web servers. Many database-driven websites that use MySQL also use a Web scripting language like PHP to access information from the database. Because both MySQL and PHP are open source (meaning they are free to download and use), the PHP/MySQL combination has become a popular choice for database-driven websites.



Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for web development(server-side), software development, mathematics, system scripting. It can be used on a server to create web applications, can be used alongside software to create workflows, connect to database systems, and also read and modify files. It can be used to handle big data and perform complex mathematics for rapid prototyping, or for production-ready software development.

Python was designed for readability, and has some similarities to the English language with influence from mathematics. It uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.



Django Framework

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up to date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is built with Django!

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and clickjacking.

Scalable

Django uses a component-based “shared-nothing” architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no

unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

Django was initially developed between 2003 and 2005 by a web team who were responsible for creating and maintaining newspaper websites. After creating a number of sites, the team began to factor out and reuse lots of common code and design patterns. This common code evolved into a generic web development framework, which was open-sourced as the "Django" project in July 2005.

Django has continued to grow and improve, from its first milestone release (1.0) in September 2008 through to the recently-released version 3.1 (2020). Each release has added new functionality and bug fixes, ranging from support for new types of databases, template engines, and caching, through to the addition of "generic" view functions and classes (which reduce the amount of code that developers have to write for a number of programming tasks).

Django is now a thriving, collaborative open-source project, with many thousands of users and contributors. While it does still have some features that reflect its origin, Django has evolved into a versatile framework that is capable of developing any type of website.



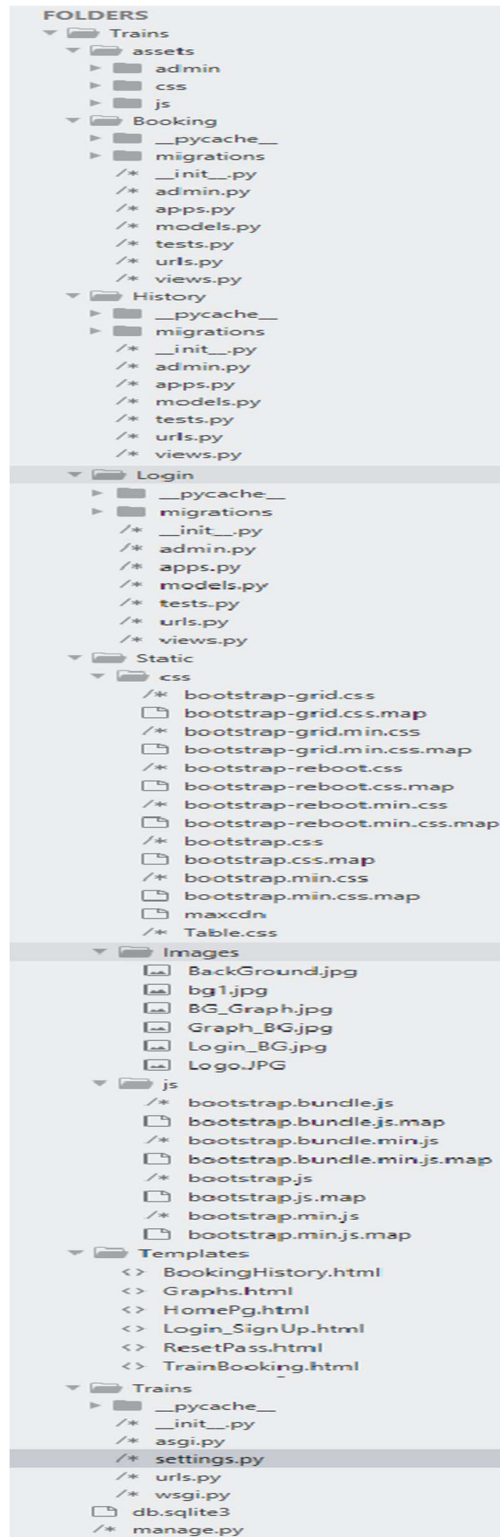
Bootstrap

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 buttons and forms. It can be used with Django. Bootstrap also supports JavaScript extensions.



Project Code And Screens

Project Directory



Trains-settings.py

```
import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(_file_).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'e2n!ewu(sw(&8m80-ts@-17oo%lzeu-9-nclga=_vwanv#^p@l'

# 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',
    'Login',
    'Booking',
    'History',
]

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 = 'Trains.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'Templates')],
    },
]
```

```

'APP_DIRS': True,
'OPTIONS': {
    'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
},
]

```

```
WSGI_APPLICATION = 'Trains.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases
```

```

DATABASES = {
    #default: {
        # 'ENGINE': 'django.db.backends.sqlite3',
        # 'NAME': BASE_DIR / 'db.sqlite3',
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'ProjTrain',
        'USER': 'root',
        'PASSWORD': 'tumba',
        'HOST': 'localhost', # Or an IP Address that your DB is hosted on
        'PORT': '3306',
    }
}

```

```
# Password validation
```

```
# https://docs.djangoproject.com/en/3.1/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/3.1/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/3.1/howto/static-files/
```

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

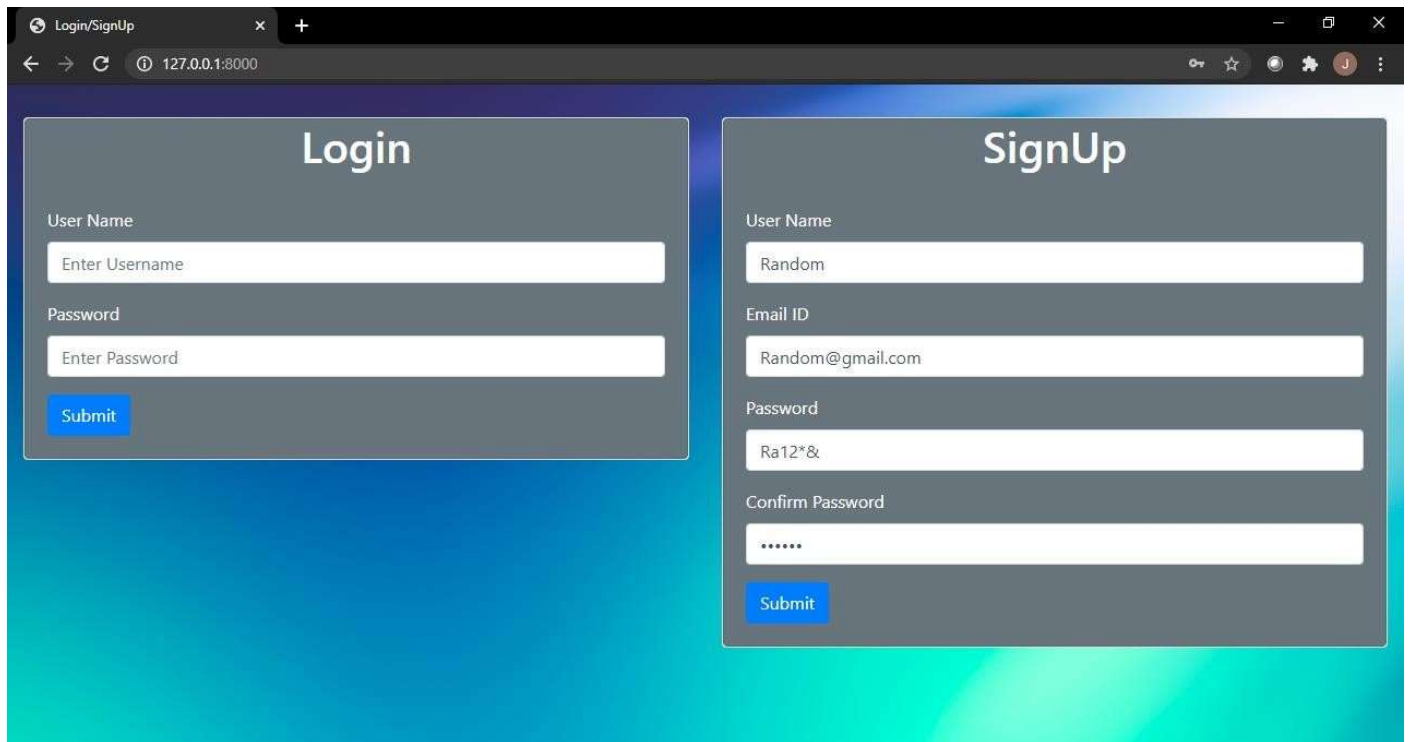
```
STATICFILES_DIRS=[  
    os.path.join(BASE_DIR, 'Static')
```

```
]
```

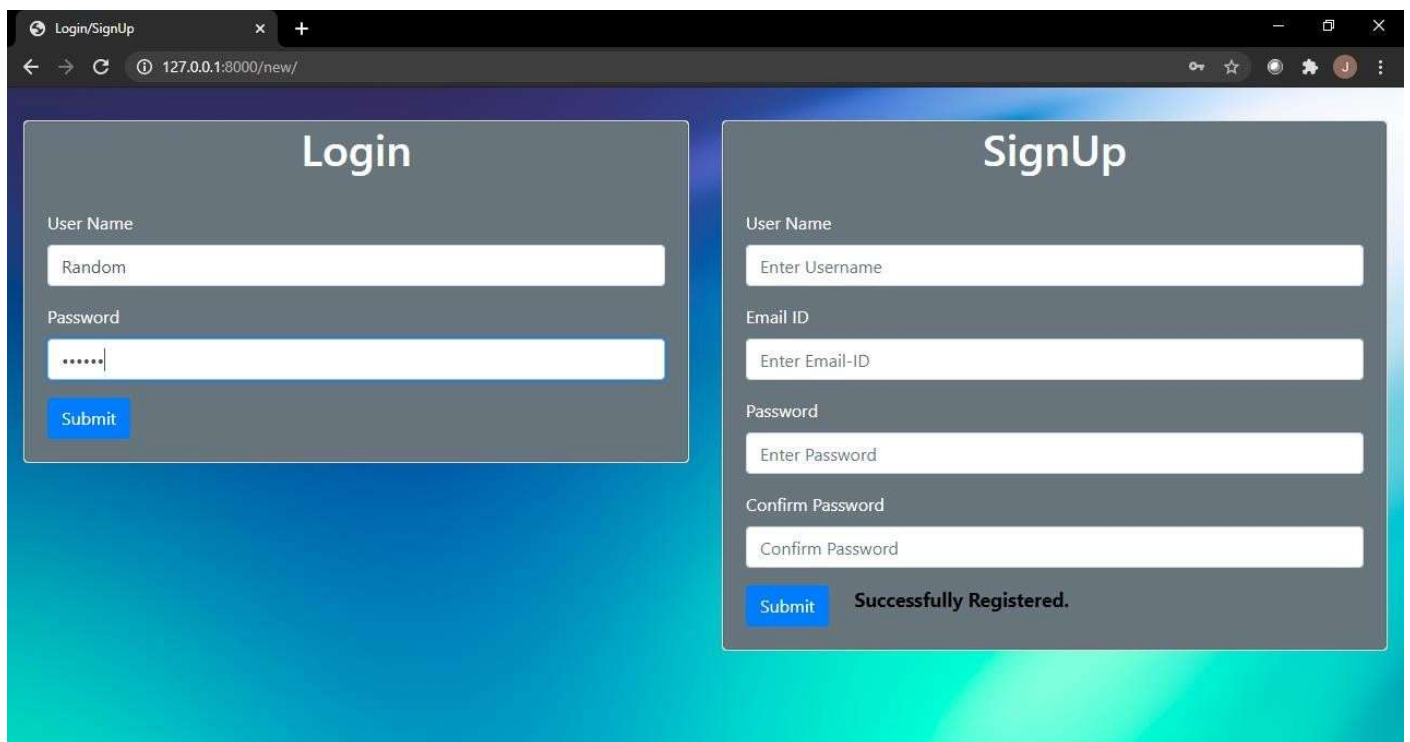
```
STATIC_ROOT=os.path.join(BASE_DIR, 'assets')
```

Login/Signup Screen

This is the first page where users can register and login to their accounts.



A screenshot of a web browser displaying a 'Login/Signup' page. The browser's address bar shows '127.0.0.1:8000'. The page has a blue gradient background. On the left, the 'Login' form has two input fields: 'User Name' with the placeholder 'Enter Username' and 'Password' with the placeholder 'Enter Password'. A blue 'Submit' button is at the bottom. On the right, the 'SignUp' form has four input fields: 'User Name' with the placeholder 'Enter Username' (containing the text 'Random'), 'Email ID' with the placeholder 'Enter Email-ID' (containing 'Random@gmail.com'), 'Password' with the placeholder 'Enter Password' (containing 'Ra12*&'), and 'Confirm Password' with the placeholder 'Confirm Password' (containing six dots). A blue 'Submit' button is at the bottom.



A screenshot of the same 'Login/Signup' page, but the browser's address bar now shows '127.0.0.1:8000/new/'. The 'Login' form on the left now has 'Random' in the 'User Name' field and six dots in the 'Password' field. The 'SignUp' form on the right now has 'Enter Username' in the 'User Name' field, 'Enter Email-ID' in the 'Email ID' field, 'Enter Password' in the 'Password' field, and 'Confirm Password' in the 'Confirm Password' field. A blue 'Submit' button is at the bottom of the 'SignUp' form, and the text 'Successfully Registered.' is displayed next to it.

Login_SignUp.html

```
{% block content %}
{% load static %}
<!doctype html>
<html lang="en">

<head>
  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
  <title>Login/SignUp</title>
</style>
  body {
    background-color: #eee;
    background-position: all;
    background-repeat: no-repeat;
    background-size: cover;
  }
</style>
</head>
<body background = "{% static 'Images/Login_BG.jpg' %}">
<body>

  <div class="container-fluid">
    <div class="row">
      <div class="col-md-6" style="padding-top: 30px">
        <div class="card">
          <div class="card text-white bg-secondary">
            <h1 class="card-title text-center">Login</h1>
            <div class="card-body">
              <form action="/login/" method="POST">
                {% csrf_token %}
                <div class="form-group">
                  <label>User Name</label>
                  <input type="text" class="form-control" name="UsLog" placeholder="Enter Username">
                </div>
                <div class="form-group">
                  <label for="exampleInputPassword1">Password</label>
                  <input type="password" class="form-control" name="PwLog" placeholder="Enter Password">
                </div>
                <div class="row">
                  <div class="col-md-2">
                    <button type="submit" class="btn btn-primary">Submit</button></div>
                  <div class="col-md-10" style="color: black;font-size: 110%;font-weight: bold">{{invalid}}</div>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
      <div class="col-md-6" style="padding-top: 30px">
        <div class="card">
          <form action="/new/" method="POST">
```


Login-models.py

```
from django.db import models
from django.forms import ModelForm, Textarea
class Users_Login(models.Model):
    # these are mysql col names
    user_name=models.CharField(max_length=100,unique=True)
    email=models.CharField(max_length=100,unique=True)
    pwd=models.CharField(max_length=100)

class Active_Users(models.Model):
    Uname_Act=models.CharField(max_length=100,unique=True)
    Status=models.CharField(max_length=100)
```

Login-views.py

```
from django.shortcuts import render
from django.http import HttpResponse
import urllib, base64
import matplotlib.pyplot as plt
from io import BytesIO
import pandas as pd
from .models import Users_Login,Active_Users
# Create your views here.
def LoginPg(request):
    if Active_Users.objects.filter(Status="Active").exists():
        Stat=Active_Users.objects.get(Status="Active")
        Stat.Status="Inactive"
        Stat.save()
        return render(request,'Login_SignUp.html')
    else:
        return render(request,'Login_SignUp.html')

def Login(request):
    LogUname=request.POST.get("UsLog")
    LogPwd=request.POST.get("PwLog")
    if Users_Login.objects.filter(user_name=LogUname,pwd=LogPwd).exists():
        names=Active_Users.objects.get(Uname_Act=LogUname)
        names.Status="Active"
        names.save()
        return render(request,"HomePg.html")
    else:
        return render(request,"Login_SignUp.html",{'invalid':'Incorrect Username/Password.'})

def SignUp(request):
    RegUser=request.POST.get("UsReg")
    RegPwd=request.POST.get("PwReg")
    RegEmail=request.POST.get("Email")
    RegPwdConfirm=request.POST.get("ConPwReg")
    if Users_Login.objects.filter(user_name=RegUser).exists():
        return render(request,"Login_SignUp.html",{'Message':'User already exists.'})
    elif RegUser==None or RegPwd==None or RegEmail==None or RegPwdConfirm==None:
```

```

        return render(request, "Login_SignUp.html", {'Message': 'All column details not filled.'})
    elif RegPwd != RegPwdConfirm:
        return render(request, "Login_SignUp.html", {'Message': 'Password and Confirm Password does not
match.'})
    elif len(RegUser) <= 0:
        return render(request, "Login_SignUp.html", {'Message': 'Enter Username.'})
    else:
        if RegPwd == RegPwdConfirm:
            Register = Users_Login(user_name=RegUser, email=RegEmail, pwd=RegPwd)
            Active = Active_Users(Uname_Act=RegUser, Status="Inactive")
            Register.save()
            Active.save()
            return render(request, "Login_SignUp.html", {'Message': 'Successfully Registered.'})

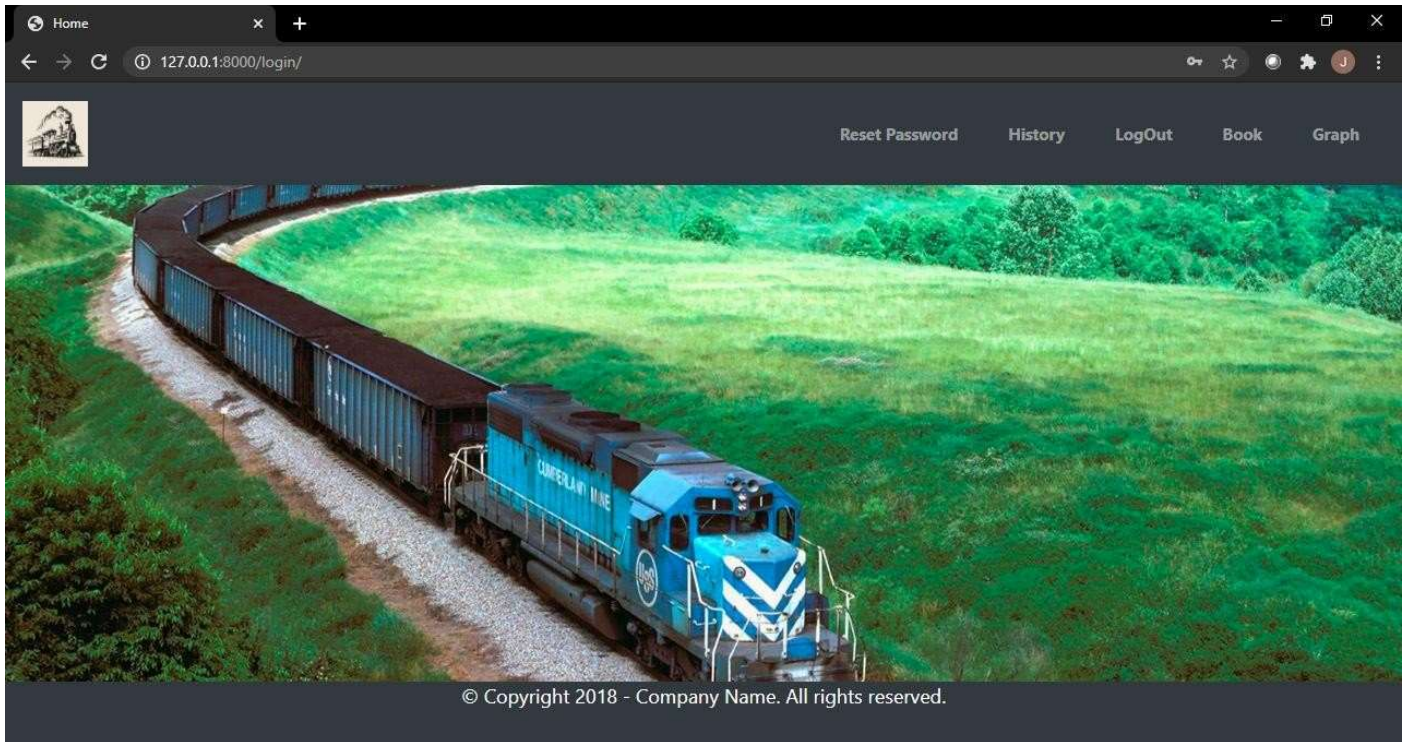
def Reset_Password(request):
    return render(request, "ResetPass.html")

def Reset_Done(request):
    Pwd_Old = request.POST.get("OldPwd")
    Pwd_New = request.POST.get("NewPwd")
    Pwd_New_Confirm = request.POST.get("ConNewPwd")
    if Users_Login.objects.filter(pwd=Pwd_Old).exists():
        if Pwd_New == Pwd_New_Confirm:
            Reset_Pwd = Users_Login.objects.get(pwd=Pwd_Old)
            Reset_Pwd.pwd = Pwd_New
            Reset_Pwd.save()
            return render(request, "Login_SignUp.html")
        else:
            return render(request, "ResetPass.html", {"error": "New Password and Confirm Password does not
match"})
    else:
        return render(request, "ResetPass.html", {"error": "Old Password does not exist"})

```

Home Page

This is the home page where the user is taken to after logging in. It contains a Navbar with the options Reset Password, History, LogOut, Book and Graph along with the train logo.



HomePg.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Home</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href="{% static 'css/bootstrap.css' %}">
</style>
<body>
  <body {
    background-color: #eee;
    background-position: all;
    background-repeat: no-repeat;
    background-size: cover;
    background-position: 0% 75%;
  }
```

```

label.radio {
  cursor: pointer
}

label.radio input {
  position: absolute;
  top: 0;
  left: 0;
  visibility: hidden;
  pointer-events: none
}

label.radio span {
  padding: 4px 0px;
  border: 1px solid red;
  display: inline-block;
  color: red;
  width: 100px;
  text-align: center;
  border-radius: 3px;
  margin-top: 7px;
  text-transform: uppercase
}

label.radio input:checked+span {
  border-color: red;
  background-color: red;
  color: #fff
}

.ans {
  margin-left: 36px !important
}

.btn:focus {
  outline: 0 !important;
  box-shadow: none !important
}

.btn:active {
  outline: 0 !important;
  box-shadow: none !important
}
</style>
<style type="text/css">
.banner {
  background-color:#DC143C;

  width: 100%;

  height: 257px;

```

```

}
.s_box{
  -webkit-box-shadow: 4px 4px 4px rgb(49 144 140 / 40%);
  background: #fff;
  display: table;
  border-radius: 3px;
  width:100%;

  display:inline-flex;
}
.padding10{padding:10px;}
.green{color:green;}
.m-container{
  width: 100%;
  background: #fff;
  border: 1px solid #dfe2e2;
  border-bottom-color: rgb(223, 226, 226);
  border-bottom-style: solid;
  border-bottom-width: 1px;
  border-bottom: 2px solid #dfe2e2;
  margin: 20px;
  border-radius: 3px;
  display: inline-block;
  padding: 1% 2%;
}
ul li {
  margin: 0;
  list-style: none;
  display: inline;
  padding: 15px;
  font-size: 15px;
  color: #676666;
  line-height: 32px;
  border-bottom: 1px #dfe2e2;
}
</style>

</head>
<body background = "{% static 'Images/bg1.jpg' %}">
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" href="#">
    
  </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbar1" aria-controls="navbar1"
aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbar1">
    <ul class="navbar-nav ml-auto">
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Reset' %}">Reset Password</a>
      </li>

```

```

<li class="nav-item" style="color:white;font-weight: bolder">
  <a class="nav-link" href="{% url 'History' %}">History</a>
</li>
<li class="nav-item" style="color:white;font-weight: bolder">
  <a class="nav-link" href="{% url 'Start' %}">LogOut</a>
</li>
<li class="nav-item" style="color:white;font-weight: bolder">
  <a class="nav-link" href="{% url 'Book-Train' %}">Book</a>
</li>
<li class="nav-item" style="color:white;font-weight: bolder">
  <a class="nav-link" href="{% url 'Graph' %}">Graph</a>
</li>
</ul>
</div>
</nav>

<footer class="fixed-bottom bg-dark" style="height: 60px">
  <div class="container">
    <p class="text-white" style="text-align: center;font-size: 18px">&copy; Copyright 2018 - Company Name. All rights
reserved.</p>
  </div>
</footer>
</div>
</body>
</html>

```

Trains-urls.py

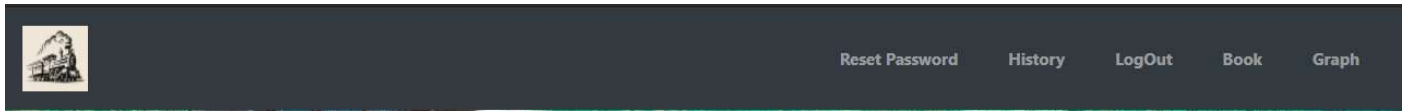
```

from django.contrib import admin
from django.urls import path,include
urlpatterns = [
    path("",include('Login.urls')),
    path('Book/',include('Booking.urls')),
    path('Hist/',include('History.urls')),
    path('admin/', admin.site.urls),
]

```


Log Out

On clicking LogOut in the home page the user is logged out and taken back to the login/sign up page.



HomePg.html

```
<li class="nav-item" style="color:white;font-weight: bolder">
  <a class="nav-link" href="{% url 'Start' %}">LogOut</a>
</li>
```

Login-urls.py

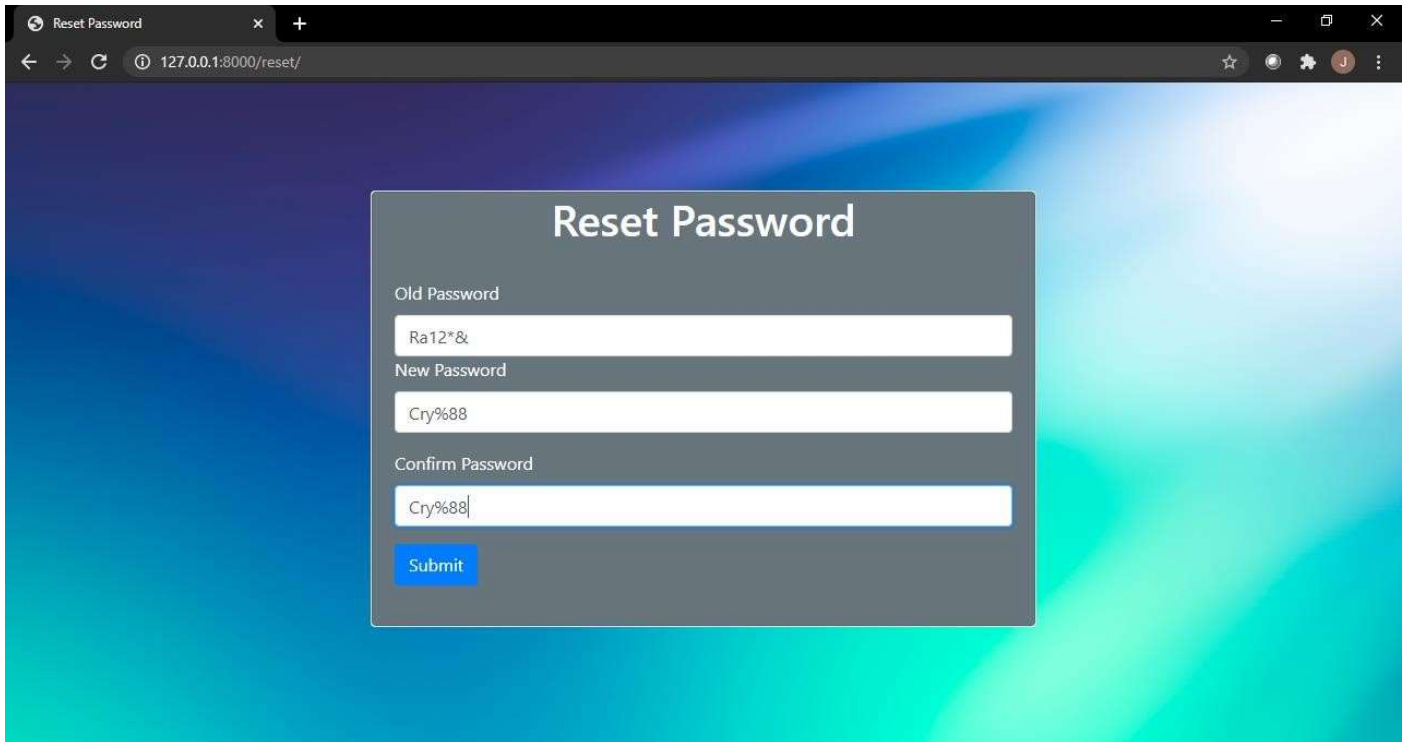
```
path("",views.LoginPg,name="Start"),
```

Login-views.py

```
def LoginPg(request):
    if Active_Users.objects.filter(Status="Active").exists():
        Stat=Active_Users.objects.get(Status="Active")
        Stat.Status="Inactive"
        Stat.save()
        return render(request,'Login_SignUp.html')
    else:
        return render(request,'Login_SignUp.html')
```

Reset Password Page

On clicking Reset Password in the home page the user is taken to the reset password screen and after successfully resetting their password, the user is taken back to the login/sign up page.



HomePg.html

```
<li class="nav-item" style="color:white;font-weight: bolder">
  <a class="nav-link" href="/reset/">Reset Password</a>
</li>
```

ResetPass.html

```
{% block content %}
{% load static %}
<!doctype html>
<html lang="en">

<head>
  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
  <title>Reset Password</title>

</head>
<body background = "{% static 'Images/Login_BG.jpg' %}" style="background-size: 100%">
<body>
```

```

<div class="container-fluid">
  <div class="row justify-content-center">
    <div class="col-md-6" style="padding-top: 30px">
      <div class="card">
        <div class="card text-white bg-secondary">
          <h1 class="card-title text-center">Reset Password</h1>

          <!--</div>
          <div class="col-md-6" border>-->
            <div class="card-body">
              <form action="/reset_done/" method="POST">
                {% csrf_token %}
                <div class="form-group">
                  <label>Old Password</label>
                  <input type="text" class="form-control" name="OldPwd">

                  <div class="form-group">
                    <label for="exampleInputPassword1">New Password</label>
                    <input type="text" class="form-control" name="NewPwd">
                  </div>
                  <div class="form-group">
                    <label for="exampleInputPassword1">Confirm Password</label>
                    <input type="text" class="form-control" name="ConNewPwd">
                  </div>
                  <div class="row">
                    <div class="col-md-6">
                      <button type="submit" class="btn btn-primary">Submit</button>
                    <div class="col-md-6">{{ error }}</div>
                  </div></div>
                </div></div></div></form></div></div>
              </form>
            </fieldset></fieldset>
          </body>

          {% endblock %}

```

Login-urls.py

```

path('reset/', views.Reset_Password),
path('reset_done/', views.Reset_Done),

```

Login-views.py

```

def Reset_Password(request):
    return render(request, "ResetPass.html")

def Reset_Done(request):
    Pwd_Old=request.POST.get("OldPwd")
    Pwd_New=request.POST.get("NewPwd")
    Pwd_New_Confirm=request.POST.get("ConNewPwd")
    if Users_Login.objects.filter(pwd=Pwd_Old).exists():

```

```
    if Pwd_New==Pwd_New_Confirm:
        Reset_Pwd=Users_Login.objects.get(pwd=Pwd_Old)
        Reset_Pwd.pwd=Pwd_New
        Reset_Pwd.save()
        return render(request,"Login_SignUp.html")
    else:
        return render(request,"ResetPass.html",{"error":"New Password and Confirm Password does not
match"})
    else:
        return render(request,"ResetPass.html",{"error":"Old Password does not exist"})
```

Booking Page

On clicking Book in the home page the user is taken to the booking page where they can book tickets.

After entering the details(“From”,”to”,”Date”), the train details are displayed and after inputting the number of tickets and clicking “Add”, the ticket is booked.

TrainBooking.html

```
{% load static %}
<!doctype html>
<html lang="en">
<head>
  <title>Book Tickets</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href="{% static 'css/bootstrap.css' %}">
<style>
  body {
    background-color: #eee;
    background-position: all;
    background-repeat: no-repeat;
    background-size: cover;
  }

  label.radio {
    cursor: pointer
  }

  label.radio input {
    position: absolute;
    top: 0;
    left: 0;
    visibility: hidden;
    pointer-events: none
  }

  label.radio span {
    padding: 4px 0px;
    border: 1px solid red;
    display: inline-block;
    color: red;
    width: 100px;
    text-align: center;
    border-radius: 3px;
    margin-top: 7px;
    text-transform: uppercase
  }

  label.radio input:checked+span {
    border-color: red;
    background-color: red;
    color: #fff
  }

  .ans {
```

```

    margin-left: 36px !important
}

.btn:focus {
    outline: 0 !important;
    box-shadow: none !important
}

.btn:active {
    outline: 0 !important;
    box-shadow: none !important
}
</style>
<style type="text/css">
    .form-control[readonly], .form-control:focus,.form-control:disabled{
        border-color: white;
        -webkit-box-shadow: none;
        box-shadow: none;
        background-color: white;
    }
    .banner {
        background-color:#DC143C;
        width: 100%;

        height: 257px;
    }
    .s_box {
        -webkit-box-shadow: 4px 4px 4px rgb(49 144 140 / 40%);
        background: #fff;
        display: table;
        border-radius: 3px;
        width:100%;

display:inline-flex;
    }
    .padding10{padding:10px;}
    .green{color:green;}
    .m-container{
        width: 100%;
        background: #fff;
        border: 1px solid #dfe2e2;
        border-bottom-color: rgb(223, 226, 226);
        border-bottom-style: solid;
        border-bottom-width: 1px;
        border-bottom: 2px solid #dfe2e2;
        margin: 20px;
        border-radius: 3px;
        display: inline-block;
        padding: 1% 2%;
    }
    ul li {

```

```

margin: 0;
list-style: none;
display: inline;
padding: 15px;
font-size: 15px;
color: #676666;
line-height: 32px;
border-bottom: 1px #dfe2e2;
}
</style>
</head>
<body background = "{% static 'Images/BackGround.jpg' %}" style="background-size: 100%">
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" href="#">
    
  </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbar1" aria-controls="navbar1"
aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbar1">
    <ul class="navbar-nav ml-auto">
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Reset' %}">Reset Password</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'History' %}">History</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Start' %}">LogOut</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'HomePg' %}">Home</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Graph' %}">Graph</a>
      </li>
    </ul>
  </div>
</nav>

<div class="container-fluid" style="background-color: rgba(76, 175, 80, 0.3)">
  <div class="padding10 row" style="background-color: rgba(76, 175, 80, 0.3)">
    <div class="col-md-12 banner" style="background-color: rgba(51, 204, 255,0.6)">
      <h1 align=center style="color:white;padding-bottom: 20px">Book Train Tickets</h1>
      <form action="{% url 'Book-Train' %}" method="POST">
        {% csrf_token %}
      <div class="s_box">
        <div class="col-md-3 border-right" style="padding:8px;background-color: rgba(0,200,0,0.5)">
          <label for="" class="black" style="font-weight: bolder">FROM</label>
          <select name="source" class="form-control">

```



```

        <option>HYD</option>
        <option>DEL</option>

    </select>
</div>
<div class="col-md-3 border-right" style="padding:8px;background-color: rgba(0,200,0,0.5)">
    <label for="" class="black" style="font-weight: bolder">TO</label>
    <select name="destination" class="form-control" id="">

        <option>HYD</option>
        <option>DEL</option>

    </select>
</div>
<div class="col-md-3 border-right" style="padding:8px;background-color: rgba(0,200,0,0.5)">
    <label for="" class="black" style="font-weight: bolder">DATE</label>
    <input type="date" class="form-control" name="date">
</div>
<div class="col-md-3" style="padding:35px;background-color: rgba(0,200,0,0.5)">
    <input id="" style="background-color: rgba(0,0,245,0.7);color:white;font-weight: bolder"
value="SEARCH TRAIN" style="-webkit-appearance: none;" type="submit" class="form-control">
</div>
</div>
</form>
</div>
</div>
</div>
<div class="container-fluid">
    <div class="row" style="padding-bottom:60px;">
        <div class="m-container">
            <div class="row" style="margin-bottom:10px">
                <div class="col-md-2" style="font-weight: bolder;font-size:120%">
                    <span class="label">Train Name</span>
                </div>
                <div class="col-md-2" style="font-weight: bolder;font-size:120%">
                    <span class="label">Price</span>
                </div>
                <div class="col-md-2" style="font-weight: bolder;font-size:120%">
                    <span class="label">Seats</span>
                </div>
                <div class="col-md-2" style="font-weight: bolder;font-size:120%">
                    <span class="label">Time</span>
                </div>
                <div class="col-md-3" style="font-weight: bolder;font-size:120%">
                    <span class="label">Tickets</span>
                </div>
                <div class="col-md-1" style="font-weight: bolder;font-size:120%">
                    <span class="label">Add</span>
                </div>
            </div>
            <div class="row" style="margin-bottom:10px;padding-bottom: 30px">
                <div class="col-md-12" style="font-size: 105%;text-align: center">

```

```

        {{RES}}</div>
        {% for i in Train_Details %}
        <form name="trainInfo" action="Booking_Train/" method="POST">
        {% csrf_token %}
        <div class="row" style="margin-bottom:10px;padding-bottom: 30px">
        <div class="col-md-2">
            <input style="font-size:110%" type="text" class="form-control" name="TrName" value={{i.Train_Name}}
readonly>
        </div>
        <div class="col-md-2">
            <input style="font-size:110%" type="text" class="form-control" name="TrPrice" value={{i.Price}}
readonly>
        </div>
        <div class="col-md-2">
            <input style="font-size:110%" type="text" class="form-control" name="TrSeats" value={{i.noOfSeats}}
readonly>
        </div>
        <div class="col-md-2">
            <input style="font-size:110%" type="text" class="form-control" name="TrTime" value={{i.time}}
readonly>
        </div>
        <div class="col-md-2">
            {{ form.as_p }}
            <input type="number" name="TrTickets">
        </div>
        <div class="col-md-2">
            <button type="submit" class="btn btn-primary float-right">Add
            </button>
        </div>
        </div>
        </form>
        {% endfor %}
    </div>
</div>
</div>

<footer class="fixed-bottom bg-dark" style="height: 60px">
    <div class="container">
        <p class="text-white" style="text-align: center;font-size: 18px">&copy; Copyright 2018 - Company Name. All rights
reserved.</p>
    </div>
</footer>

</body>

```

Booking-urls.py

```

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

```

```

path("",views.Search_Train,name="Book-Train"),
path('Home/',views.Home),
path('Booking_Train/',views.Booking_Train),
path('Booking_Train/Home/',views.Home),

```

```

]

```

Booking-models.py

```

from django.db import models
from django.forms import ModelForm,Textarea

class Train_Info(models.Model):
    # these are mysql col names
    Train_Name=models.CharField(max_length=100)
    TFrom=models.CharField(max_length=100)
    Tto=models.CharField(max_length=100)
    noOfSeats=models.IntegerField()
    Price=models.IntegerField()
    time=models.TimeField(blank=True)
    days=models.CharField(max_length=200)
    num=models.IntegerField()

class Train_Running_Info(models.Model):
    # these are mysql col names
    Train_Name=models.CharField(max_length=100)
    TFrom=models.CharField(max_length=100)
    Tto=models.CharField(max_length=100)
    noOfSeats=models.IntegerField()
    date=models.DateField(blank=True)

```

Booking-views.py

```

import datetime
from django.shortcuts import render
from django.http import HttpResponse
from .models import Train_Info,Train_Running_Info
from Login.models import Active_Users
from History.models import Book_Train

# Create your views here.
def Search_Train(request):
    From=request.POST.get("source")
    To=request.POST.get("destination")
    Date=request.POST.get("date")
    print(Date,type(Date))
    Cur_Date=datetime.datetime.today().strftime("%Y-%m-%d")
    print(Cur_Date,type(Cur_Date))
    if Date!=None:
        if Date>Cur_Date:
            if Train_Info.objects.filter(TFrom=From,Tto=To).exists():
                Day=datetime.datetime.strptime(Date,"%Y-%m-%d").strftime("%A")

```

```

        Train_Data=Train_Info.objects.filter(TFrom=From,Tto=To,days_contains=Day)
        print(Day,type(Day))
        if Train_Running_Info.objects.filter(Train_Name="TEMP").exists():
            Train_Running_Info.objects.get(Train_Name="TEMP").delete()

    Running=Train_Running_Info(TFrom=From,Tto=To,date=Date,Train_Name="TEMP",noOfSeats=10)
        Running.save()
    else:

    Running=Train_Running_Info(TFrom=From,Tto=To,date=Date,Train_Name="TEMP",noOfSeats=10)
        Running.save()
        return render(request,"TrainBooking.html",{"Train_Details":Train_Data})
    else:
        return render(request,"TrainBooking.html",{"RES":"Search Results Appear Here."})

    else:
        return render(request,"TrainBooking.html",{"RES":"Enter Correct Date"})

    else:
        return render(request,"TrainBooking.html",{"RES":"Search Results Appear Here."})

def Booking_Train(request):
    Ticket=request.POST.get("TrTickets")
    Name=request.POST.get("TrName")
    Price=request.POST.get("TrPrice")
    Seats=request.POST.get("TrSeats")
    Time=request.POST.get("TrTime")
    UserName=Active_Users.objects.get(Status="Active")
    Total_Price=int(Ticket)*int(Price)
    Trains=Train_Info.objects.get(Train_Name=Name)
    OldTrain=Train_Running_Info.objects.get(Train_Name="TEMP")
    Remaining_Tickets=int(Trains.noOfSeats)-int(Ticket)
    if Train_Running_Info.objects.filter(Train_Name=Name,date=OldTrain.date).exists():
        Train_Data=Train_Running_Info.objects.get(Train_Name=Name)
        if Train_Running_Info.objects.filter(date=Train_Data.date).exists():
            Other_Remaining_Tickets=int(Train_Data.noOfSeats)-int(Ticket)
            print(Train_Data.noOfSeats,Other_Remaining_Tickets)
            if Other_Remaining_Tickets>=0:
                Train_Data.noOfSeats=Other_Remaining_Tickets
                Train_Running_Info.objects.filter(Train_Name="TEMP").delete()
                Train_Data.save()
            if
Book_Train.objects.filter(Train_names=Name,User=UserName.Uname_Act,Date=Train_Data.date).exists():

    book=Book_Train.objects.get(Train_names=Name,User=UserName.Uname_Act,Date=Train_Data.date)
        book.Tickets=book.Tickets+int(Ticket)
        book.Price=book.Price+Total_Price
        book.save()
        return render(request,"TrainBooking.html",{"RES":"Tickets Booked!"})
    else:

    book=Book_Train(Train_names=Name,Price=Total_Price,Tickets=int(Ticket),User=UserName.Uname_Act,Time=Time,Date=Train_Data.date)

        book.save()
        return render(request,"TrainBooking.html",{"RES":"Tickets Booked!"})

```

```

        else:
            Train_Running_Info.objects.filter(Train_Name="TEMP").delete()
            return render(request,"TrainBooking.html",{"RES":"Tickets Not Available."})
    else:
        running=Train_Running_Info.objects.get(Train_Name="TEMP")
        running.Train_Name=Name
        running.noOfSeats=Remaining_Tickets
        running.save()
        print(Name>Total_Price,Ticket,Time,type(Name),type>Total_Price),type(Ticket),type(Time))

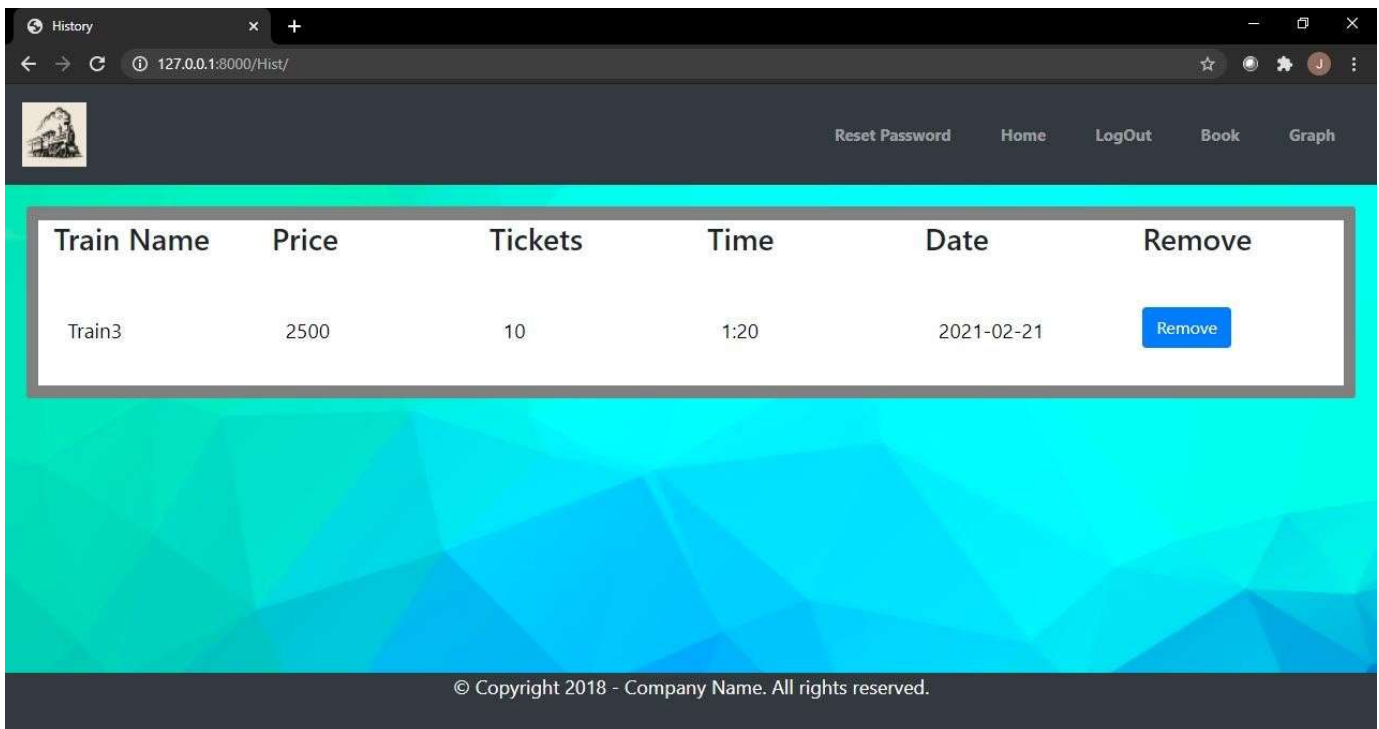
        book=Book_Train(Train_names=Name,Price=Total_Price,Tickets=int(Ticket),User=UserName.Uname_Act,Time=Time,Date=running.date)
        book.save()
        return render(request,"TrainBooking.html",{"RES":"Tickets Booked!"})

def Home(request):
    if Train_Running_Info.objects.filter(Train_Name="TEMP").exists():
        Train_Running_Info.objects.get(Train_Name="TEMP").delete()
        return render(request,"HomePg.html")
    else:
        return render(request,"HomePg.html")

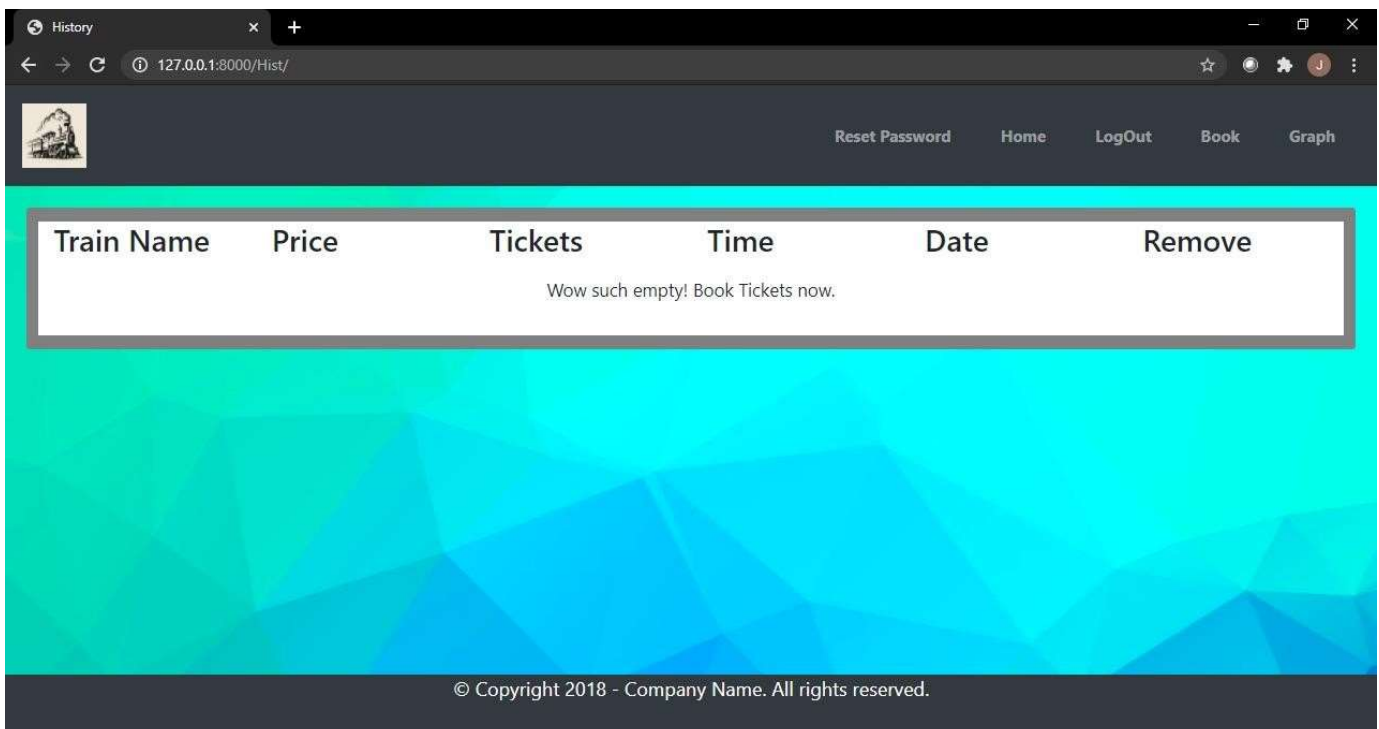
```

Booking History Page

On clicking History in the home page, the user is taken to the booking history page where they cancel any previous bookings.



After clicking "Remove" the corresponding record is removed and an updated history page is displayed.



BookingHistory.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <title>History</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href="{% static 'css/bootstrap.css' %}">
<style>
  body {
    background-color: #eee;
    background-position: all;
    background-repeat: no-repeat;
    background-size: cover;
  }

  label.radio {
    cursor: pointer
  }

  label.radio input {
    position: absolute;
    top: 0;
    left: 0;
    visibility: hidden;
    pointer-events: none
  }

  label.radio span {
    padding: 4px 0px;
    border: 1px solid red;
    display: inline-block;
    color: red;
    width: 100px;
    text-align: center;
    border-radius: 3px;
    margin-top: 7px;
    text-transform: uppercase
  }

  label.radio input:checked+span {
    border-color: red;
    background-color: red;
    color: #fff
  }
```

```

.ans {
  margin-left: 36px !important
}

.btn:focus {
  outline: 0 !important;
  box-shadow: none !important
}

.btn:active {
  outline: 0 !important;
  box-shadow: none !important
}
</style>
<style type="text/css">
  .form-control[readonly], .form-control:focus,.form-control:disabled{
    border-color: white;
    -webkit-box-shadow: none;
    box-shadow: none;
    background-color: white;
  }
  .banner {
    background-color:#DC143C;
    width: 100%;

    height: 257px;
  }
  .s_box{
    -webkit-box-shadow: 4px 4px 4px rgb(49 144 140 / 40%);
    background: #fff;
    display: table;
    border-radius: 3px;
    width:100%;
    display:inline-flex;
  }
  .m-container{
    width: 100%;
    background: #fff;
    margin: 20px;
    border-radius: 3px;
    display: inline-block;
    padding: 1% 2%;
  }
  ul li {
    margin: 0;
    list-style: none;
    display: inline;
    padding: 15px;
    font-size: 15px;
    color: #676666;
    line-height: 32px;

```



```

border-bottom: 1px #dfe2e2;
}
</style>
</head>
<body background = "{% static 'Images/BackGround.jpg' %}" style="background-size: 100%">
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" href="#">
    
  </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbar1" aria-controls="navbar1"
aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbar1">
    <ul class="navbar-nav ml-auto">
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Reset' %}">Reset Password</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'HomePg' %}">Home</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Start' %}">LogOut</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Book-Train' %}">Book</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Graph' %}">Graph</a>
      </li>
    </ul>
  </div>
</nav>
<div class="container-fluid">
<div class="row" style="padding-bottom:60px;">
  <div class="m-container" style="background-color: gray">
    <div class="row" style="padding-bottom:10px;background: white">
      <div class="col-md-2">
        <h3><span class="label">Train Name</span></h3>
      </div>
      <div class="col-md-2">
        <h3><span class="label">Price</span></h3>
      </div>
      <div class="col-md-2">
        <h3><span class="label">Tickets</span></h3>
      </div>
      <div class="col-md-2">
        <h3><span class="label">Time</span></h3>
      </div>
      <div class="col-md-2">
        <h3><span class="label">Date</span></h3>
      </div>
    </div>
  </div>
</div>

```

```

        <div class="col-md-2">
            <h3><span class="label">Remove</span></h3>
        </div>
    </div>
    <div class="row" style="padding-bottom:30px;background: white">
        <div class="col-md-12" style="font-size: 105%;text-align: center;">
            {{Empty}}</div></div>
            {% for i in Booked %}
                <form name="trainInfo" action="DelHis/" method="POST">
                    {% csrf_token %}
                    <div class="row" style="padding-bottom:30px;background: white">
                        <div class="col-md-2">
                            <input style="background: white;color: black;font-size:120%" type="text" value={{i.Train_names}} class="form-control" name="TNAME" readonly>
                        </div>
                        <div class="col-md-2">
                            <input style="background: white;color: black;font-size:120%" type="text" value={{i.Price}} class="form-control" name="PRICE" readonly>
                        </div>
                        <div class="col-md-2">
                            <input style="background: white;color: black;font-size:120%" type="text" value={{i.Tickets}} class="form-control" name="TICKETS" readonly>
                        </div>
                        <div class="col-md-2">
                            <input style="background: white;color: black;font-size:120%" type="text" value={{i.Time}} class="form-control" name="TIME" readonly>
                        </div>
                        <div class="col-md-2">
                            <input style="background: white;color: black;font-size:120%" type="text" value={{i.Date}} class="form-control" name="DATE" readonly>
                        </div>
                        <div class="col-md-1">
                            <button type="submit" class="btn btn-primary">Remove</button>
                        </div>
                        <div class="col-md-1">
                            <input style="background: white;color: black;font-size:120%" type="hidden" value={{i.id}} class="form-control" name="ID" readonly>
                        </div>
                    </form>
                </div>

            {% endfor %}

    <footer class="fixed-bottom bg-dark" style="height: 60px">
        <div class="container">
            <p class="text-white" style="text-align: center;font-size: 18px">&copy; Copyright 2018 - Company Name. All rights reserved.</p>
        </div>
    </footer>

</body>
</html>

```

History-urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path("", views.Show_History, name="History"),
    path('Home/', views.Home, name="HomePg"),
    path('DelHis/', views.Del_History),
    path('DelHis/Home/', views.Home),
    path('Graph/', views.Show_Graph, name="Graph"),
    path('Graph/Home/', views.Home),
]
```

History-models.py

```
from django.db import models
from django.forms import ModelForm, Textarea

class Book_Train(models.Model):
    Train_names=models.CharField(max_length=100)
    Price=models.IntegerField()
    Tickets=models.IntegerField()
    User=models.CharField(max_length=100)
    Time=models.CharField(max_length=20)
    Date=models.CharField(max_length=50)
```

History-views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from .models import Book_Train
from Login.models import Active_Users
from Booking.models import Train_Running_Info, Train_Info
import urllib, base64, datetime
from io import BytesIO
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create your views here.
def Show_History(request):
    Stat=Active_Users.objects.get(Status="Active")
    Booking=Book_Train.objects.filter(User=Stat.Username_Act)
    print(len(Booking))
    if len(Booking)==0:
        return render(request, "BookingHistory.html", {"Empty": "Wow such empty!\n Book Tickets now."})
    else:
        return render(request, "BookingHistory.html", {"Booked": Booking})

def Home(request):
    return render(request, "HomePg.html")
```

```

def Del_History(request):
    ID=request.POST.get("ID")
    DEL=Book_Train.objects.get(id=ID)
    CurDate=datetime.datetime.today().strftime('%Y-%m-%d')
    print(CurDate,type(CurDate))
    print(DEL.Date,type(DEL.Date))
    if DEL.Date>CurDate:
        Change=Train_Running_Info.objects.get(Train_Name=DEL.Train_names,date=DEL.Date)
        Change.noOfSeats=int(Change.noOfSeats)+int(DEL.Tickets)
        Change.save()
        DEL.delete()
        Stat=Active_Users.objects.get(Status="Active")
        Booking=Book_Train.objects.filter(User=Stat.Uname_Act)
        return render(request,"BookingHistory.html",{"Booked":Booking})
    else:
        Stat=Active_Users.objects.get(Status="Active")
        Booking=Book_Train.objects.filter(User=Stat.Uname_Act)
        return render(request,"BookingHistory.html",{"Booked":Booking})

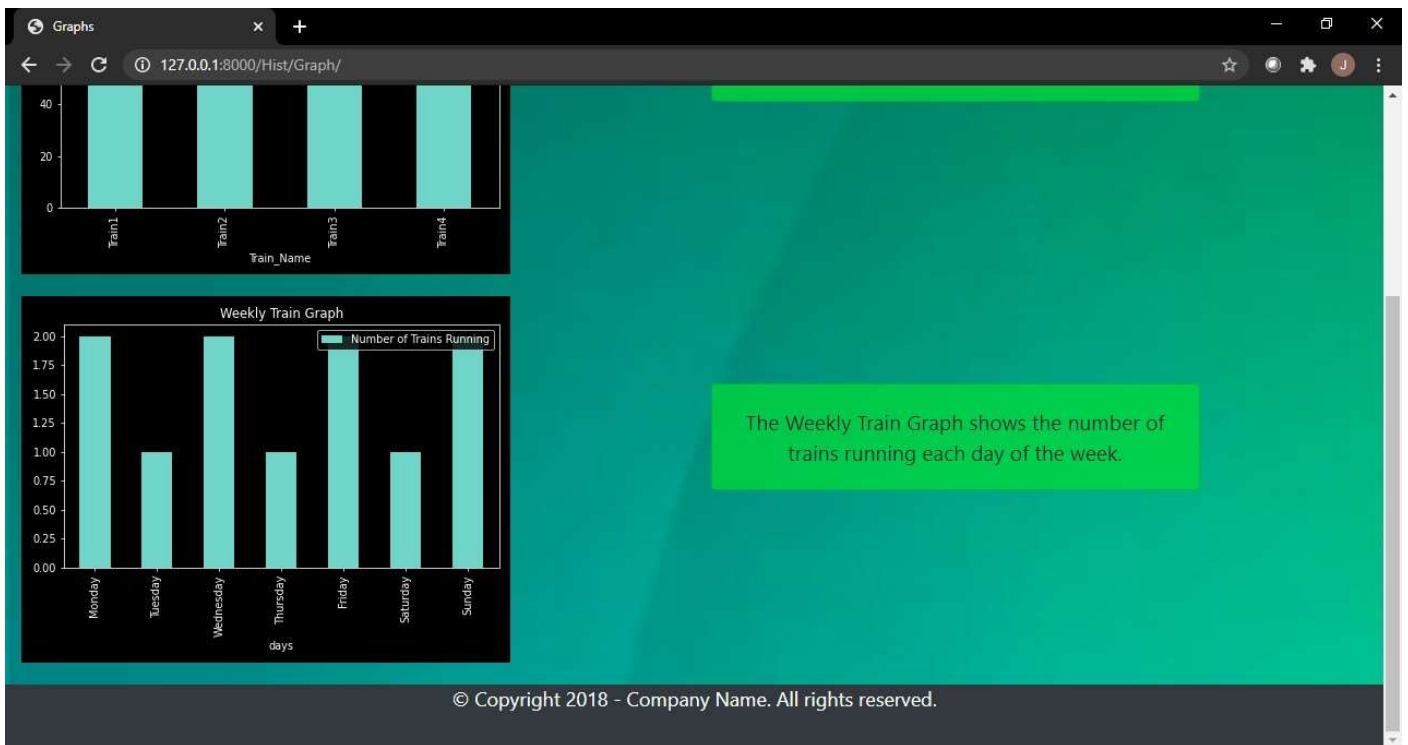
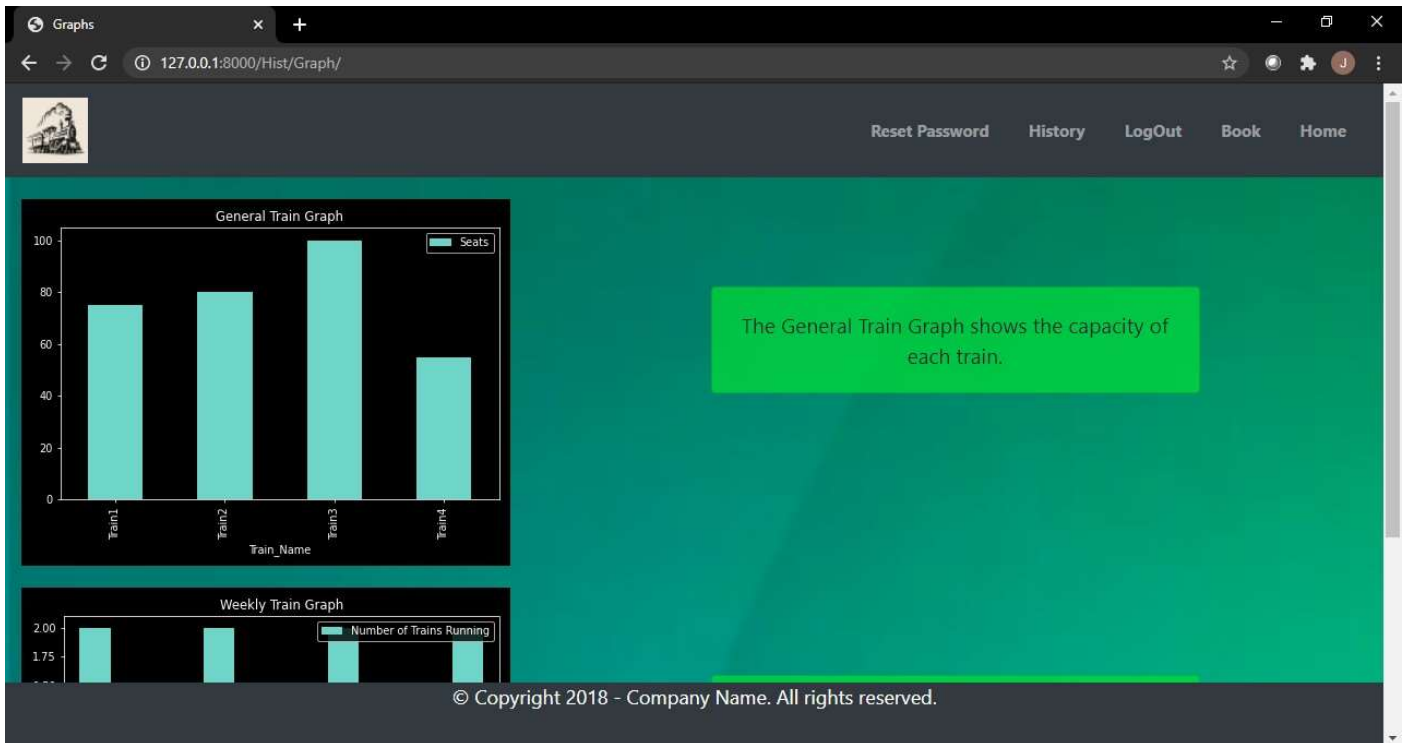
def Show_Graph(request):
    plt.style.use('dark_background')
    lst=["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"]
    df2=pd.DataFrame(columns=['num','days'])
    for i in lst:
        print(i,type(i))
        df=pd.DataFrame(Train_Info.objects.filter(days_contains=i).values())
        df['days']=i
        aggregation_functions = {'num': 'sum', 'days': 'first'}
        df_0 = df.groupby(df['days']).aggregate(aggregation_functions)
        print(df_0,type(df_0))
        df2=df2.append(df_0)
        print(df2)
    Uname=Active_Users.objects.get(Status="Active")
    df1 = pd.DataFrame(Train_Info.objects.all().values())
    print(df1)
    df1.plot.bar(x="Train_Name",y="noOfSeats",title="General Train Graph",label="Seats")
    fig1=plt.gcf()
    plt.tight_layout()
    buf1 = BytesIO()
    fig1.savefig(buf1, format='png', dpi=70) # dpi sets the size of the fig
    buf1.seek(0)
    image_base64_1 = base64.b64encode(buf1.read())
    uri1 = 'data:image_1/png;base64,' + urllib.parse.quote(image_base64_1)
    ""df2 = pd.DataFrame(Book_Train.objects.filter(User=Uname.Uname_Act).values("Train_names").distinct())
    print(df2)""
    df2.plot.bar(x="days",y="num",title="Weekly Train Graph",label="Number of Trains Running")
    fig2=plt.gcf()
    plt.tight_layout()
    buf2 = BytesIO()
    fig2.savefig(buf2, format='png', dpi=70) # dpi sets the size of the fig
    buf2.seek(0)
    image_base64_2 = base64.b64encode(buf2.read())
    uri2 = 'data:image_2/png;base64,' + urllib.parse.quote(image_base64_2)

```

```
print(uri1,uri2)
return render(request,"Graphs.html",{"graphlib1":uri1,"graphlib2":uri2})
```

Graph Page

On clicking Graph in the home page, the user is taken to the graph page which has two graphs. One displays how many tickets each train has while the other displays the number of trains running each day of the week.



Graphs.html

```
{% load static %}
<!doctype html>
<html lang="en">
<head>
  <title>Graphs</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href="{% static 'css/bootstrap.css' %}">
<style>
  body {
    background-color: #eee;
    background-position: all;
    background-repeat: no-repeat;
    background-size: 400%;
  }

  label.radio {
    cursor: pointer
  }

  label.radio input {
    position: absolute;
    top: 0;
    left: 0;
    visibility: hidden;
    pointer-events: none
  }

  label.radio span {
    padding: 4px 0px;
    border: 1px solid red;
    display: inline-block;
    color: red;
    width: 100px;
    text-align: center;
    border-radius: 3px;
    margin-top: 7px;
    text-transform: uppercase
  }

  label.radio input:checked+span {
    border-color: red;
    background-color: red;
    color: #fff
  }

  .ans {
```

```

margin-left: 36px !important
}

.btn:focus {
outline: 0 !important;
box-shadow: none !important
}

.btn:active {
outline: 0 !important;
box-shadow: none !important
}
</style>
<style type="text/css">
.form-control[readonly], .form-control:focus,.form-control:disabled{
border-color: white;
-webkit-box-shadow: none;
box-shadow: none;
background-color: white;
}
.banner {
background-color:#DC143C;
width: 100%;

height: 257px;
}
.s_box {
-webkit-box-shadow: 4px 4px 4px rgb(49 144 140 / 40%);
background: #fff;
display: table;
border-radius: 3px;
width:100%;

display:inline-flex;
}
.padding10{padding:10px;}
.green{color:green;}
.m-container{
width: 100%;
background: #fff;
border: 1px solid #dfe2e2;
border-bottom-color: rgb(223, 226, 226);
border-bottom-style: solid;
border-bottom-width: 1px;
border-bottom: 2px solid #dfe2e2;
margin: 20px;
border-radius: 3px;
display: inline-block;
padding: 1% 2%;
}
ul li {

```



```

margin: 0;
list-style: none;
display: inline;
padding: 10px;
font-size: 15px;
color: #676666;
line-height: 32px;
border-bottom: 1px #dfe2e2;
}
</style>
</head>
<body background = "{% static 'Images/BG_Graph.jpg' %}" %">
<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" href="#">
    
  </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbar1" aria-controls="navbar1"
aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbar1">
    <ul class="navbar-nav ml-auto">
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Reset' %}">Reset Password</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'History' %}">History</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Start' %}">LogOut</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'Book-Train' %}">Book</a>
      </li>
      <li class="nav-item" style="color:white;font-weight: bolder">
        <a class="nav-link" href="{% url 'HomePg' %}">Home</a>
      </li>
    </ul>
  </div>
</nav>
<div class="container-fluid">
<div class="row" style="padding-top: 20px;padding-bottom: 10px">
  <div class="col-md-6">
    <img src={{graphlib1}}>
  </div>
  <div class="col-md-6" style="padding-top: 80px;padding-bottom: 25px">
    <div class="card text-center" style="width: 28rem;background-color: rgba(0,220,10,0.7)">
  <div class="card-body">
    <p class="card-text" style="font-size: 120%">
      The General Train Graph shows the capacity of each train.
    </p>
  </div>
  </div>
</div>

```

```

</div>
</div>
</div>
</div>
<div class="row" style="padding-top: 10px;padding-bottom: 80px">
  <div class="col-md-6">
    <img src={{graphlib2}}>
  </div>
  <div class="col-md-6" style="padding-top: 80px;padding-bottom: 25px">
    <div class="card text-center" style="width: 28rem;background-color: rgba(0,220,10,0.7)">
      <div class="card-body">
        <p class="card-text" style="font-size: 120%">
          The Weekly Train Graph shows the number of trains running each day of the week.
        </p>
      </div>
    </div>
  </div>
  </div>
  </div>
  </div>
<footer class="fixed-bottom bg-dark" style="height: 60px">
  <div class="container">
    <p class="text-white" style="text-align: center;font-size: 18px">&copy; Copyright 2018 - Company Name. All rights
reserved.</p>
  </div>
</footer>
</body>
</html>

```

History-urls.py

```

path('Graph/',views.Show_Graph,name="Graph"),
path('Graph/Home/',views.Home),

```

History-views.py

```

def Home(request):
    return render(request,"HomePg.html")

def Show_Graph(request):
    plt.style.use('dark_background')
    lst=["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"]
    df2=pd.DataFrame(columns=['num','days'])
    for i in lst:
        print(i,type(i))
        df=pd.DataFrame(Train_Info.objects.filter(days_contains=i).values())
        df['days']=i
        aggregation_functions = {'num': 'sum', 'days': 'first'}
        df_0 = df.groupby(df['days']).aggregate(aggregation_functions)
        print(df_0,type(df_0))
        df2=df2.append(df_0)
        print(df2)

```

```

Uname=Active_Users.objects.get(Status="Active")
df1 = pd.DataFrame(Train_Info.objects.all().values())
print(df1)
df1.plot.bar(x="Train_Name",y="noOfSeats",title="General Train Graph",label="Seats")
fig1=plt.gcf()
plt.tight_layout()
buf1 = BytesIO()
fig1.savefig(buf1, format='png', dpi=70) # dpi sets the size of the fig
buf1.seek(0)
image_base64_1 = base64.b64encode(buf1.read())
uri1 = 'data:image_1/png;base64,' + urllib.parse.quote(image_base64_1)
""df2 = pd.DataFrame(Book_Train.objects.filter(User=Uname.Uname_Act).values("Train_names").distinct())
print(df2)""
df2.plot.bar(x="days",y="num",title="Weekly Train Graph",label="Number of Trains Running")
fig2=plt.gcf()
plt.tight_layout()
buf2 = BytesIO()
fig2.savefig(buf2, format='png', dpi=70) # dpi sets the size of the fig
buf2.seek(0)
image_base64_2 = base64.b64encode(buf2.read())
uri2 = 'data:image_2/png;base64,' + urllib.parse.quote(image_base64_2)
print(uri1,uri2)
return render(request,"Graphs.html",{"graphlib1":uri1,"graphlib2":uri2})

```

MySQL Tables

All Tables:

```
mysql> show tables;
+-----+
| Tables_in_projtrain |
+-----+
| auth_group           |
| auth_group_permissions |
| auth_permission      |
| auth_user            |
| auth_user_groups     |
| auth_user_user_permissions |
| booking_train_info   |
| booking_train_running_info |
| django_admin_log     |
| django_content_type  |
| django_migrations    |
| django_session       |
| history_book_train   |
| login_active_users   |
| login_users_login    |
+-----+
15 rows in set (0.00 sec)
```

Login_Users_Login:

```
mysql> select * from Login_Users_Login;
+----+-----+-----+-----+
| id | user_name | email                | pwd |
+----+-----+-----+-----+
| 1  | Chandu   | Chandu@gmail.com    | 123 |
| 2  | Abec     | Abec@gmail.com      | NEW |
| 3  | Adithya  | Adithya@gmail.com   | 98765 |
| 4  | Random   | Random@gmail.com    | Cry%88 |
+----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Login_Active_Users:

```
mysql> select * from Login_Active_Users;
```

id	Uname_Act	Status
1	Chandu	Inactive
2	Abec	Inactive
3	Adithya	Inactive
4	Random	Active

```
4 rows in set (0.00 sec)
```

Booking_Train_Info:

```
mysql> select * from Booking_Train_Info;
```

id	Train_Name	TFrom	Tto	noOfSeats	Price	time	days	num
1	Train1	HYD	DEL	75	350	23:40:00.000000	Monday,Tuesday,Wednesday	1
2	Train2	HYD	DEL	80	300	12:35:00.000000	Thursday,Friday	1
3	Train3	HYD	DEL	100	250	01:20:00.000000	Saturday,Sunday	1
4	Train4	HYD	DEL	55	500	04:00:00.000000	Monday,Wednesday,Friday,Sunday	1

```
4 rows in set (0.00 sec)
```

History_Book_Train:

```
mysql> select * from History_Book_Train;
```

id	Train_names	Price	Tickets	User	Date	Time
10	Train2	1500	5	Chandu	2021-01-22	12:35
12	Train3	3000	12	Chandu	2021-01-17	1:20
13	Train3	1750	7	Chandu	2021-01-30	1:20
20	Train3	2500	10	Random	2021-02-21	1:20

```
4 rows in set (0.00 sec)
```

Booking_Train_Running_Info:

```
mysql> select * from Booking_Train_Running_Info;
```

id	Train_Name	TFrom	Tto	noOfSeats	date
9	Train3	HYD	DEL	61	2021-01-17
24	Train4	HYD	DEL	38	2021-01-25
60	Train1	HYD	DEL	71	2021-01-25
61	Train2	HYD	DEL	75	2021-01-22
73	Train3	HYD	DEL	93	2021-01-30
93	Train3	HYD	DEL	90	2021-02-21

```
6 rows in set (0.00 sec)
```

Future Extensions

1. More train stations to be added.
2. Handling operations of trains with multiple stops from start to end point.
3. Adding and working with non-weekly trains.
4. Being able to edit already booked tickets instead of deleting them and booking again.
5. Admin login to add and delete train data.

Conclusion

The Train Ticketing System was built using Python with Django Framework, MySQL, HTML and Bootstrap. It was a great learning experience for us. We got the opportunity to use Django framework which led us to learn lots of new things. We made use of various python modules and got to practice making html webpages. There were many difficulties faced and this was an ideal way for us to learn to be patient and figure out how to solve the problems. This project taught us to stay committed and to not give up.

References

1. www.stackoverflow.com
2. en.wikipedia.org
3. <https://getbootstrap.com/>
4. www.mysql.org
5. <https://www.python.org/>
6. CBSE Grade 11 and 12 Informatics Practices Textbooks
7. <https://www.geeksforgeeks.org/>
8. <https://matplotlib.org/>
9. <https://docs.djangoproject.com/>
10. <https://www.edureka.co/>
11. <https://stackabuse.com/>
12. <https://docs.python.org/>