AMRITA VIDYALAYAM

TRICHY

Managed by: MATA AMRITANANDAMAYI MATH

DEPARTMENT OF COMPUTER SCIENCE

PROJECT FILE 2021-2022

Topic: **Student Marks Application**

AMRITA VIDYALAYAM

TRICHY

Managed by: MATA AMRITANANDAMAYI MATH

Certificate

This is to certify that Master **Arjun G., Ashwin Balaji S., Aravind Chokalingham M.** of Class **XII** Section **B** Register No: **20666445, 20666446, 20666444** have completed their investigatory project in the subject of **Computer Science** as required according to the Central Board of Secondary Education for the academic session **2021-2022**.

Date: **Teacher in charge**

Examiner's Signature **Principal**

ACKNOWLEDGEMENT

I would like to convey my thanks to my **Computer Science** Teacher **V. Nithya** Amrita Vidyalayam, Trichy for her immense help and guidance in the completion of my project.

I would also like to convey my thanks to our Principal **Mrs. Usha Raghavan** and our school management for providing the necessary materials.

I would like to extend my gratitude to everyone who helped me to complete this project.

Name of the Students: **Arjun G., Ashwin Balaji S., Aravind Chokalingham M.**

Register Nos: **20666445, 20666446, 20666444**

Table of Contents

Aim

We aim to create a web application where teachers can enter the raw marks of the students and that will be stored in a cloud database and accessible by the teacher at any time. Additional information such as Rank, Percentage, Average, Grade, etc. can also be calculated (soon). Teachers can access it whenever needed, wherever needed just with an Internet connection in a laptop or a desktop.

This project aims to provide this useful, elegant-looking web application that could be used by teachers intuitively and simply.

Introduction

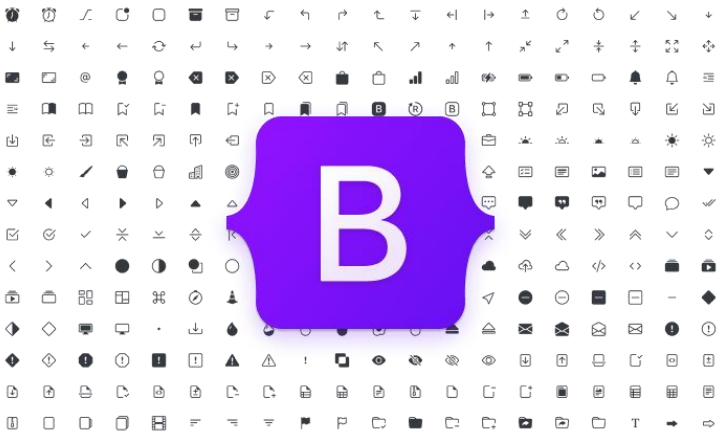
Here we are going to explain the basic structure of our project i.e., how it works, what it uses to function, what are the components used, etc.

Firstly, a website that you can see on the internet consists of two parts. One is the *frontend* and the other is the *backend***.** Frontend refers to the graphical user interface of a website. In this development, we create the view that a user would see when they visit our site. Frontend development is done using:

a) **HTML** – Hyper Text Markup Language – The backbone, the raw text content of the website
b) **CSS** – Cascading Style Sheets – The one that makes everything look beautiful
c) **JavaScript** – The one which makes the site responsive to user interactions

F-1: Frontend development

But here is the catch! CSS and JavaScript are whole other languages. Learning them enough to implement in the way we want in our project is an extensive and time-consuming task that would be impractical to do in a tightly scheduled 12th Grade life. It is indeed better to learn them and use them directly since that gives a much higher control over the GUI as well as it provides a ton of creative implementations. But that is not the most optimal and effective path for us. Instead, we found an alternative, highly efficient path to manage our frontend so that we can work more on the backend in Python and MySQL. And that is the use of *Bootstrap*.



F-2: Bootstrap

Bootstrap is a very popular, free and open-source CSS framework. Framework is a tool that provides ready-made components or solutions that are customized in order to speed up development. So, Bootstrap gave us a much easier way to design our frontend without going beyond the scope of our resources. But still, it had a learning curve and took considerable amount of time to develop the web pages that you are going to see in this project. In the end, it was a very efficient and manageable way to create good looking frontend.

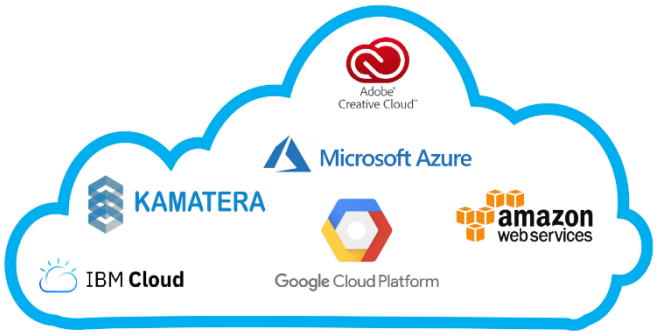
Second is the *backend* our website. This is the part of the website that the users will not see. Here is where all the magic happens. The backend is the place where we decide what content to show to the user as well as many more tasks. It communicates with our database server, collects the info from that database and sends only the necessary info to our frontend which the user would see. There are variety of languages and utilities that works together to make the backend function as intended. In our project, we are going to utilize a massively popular backend web framework called as **Django** (pronounced as *jang-goh*).

F-3: Frontend and Backend

F-4: Django

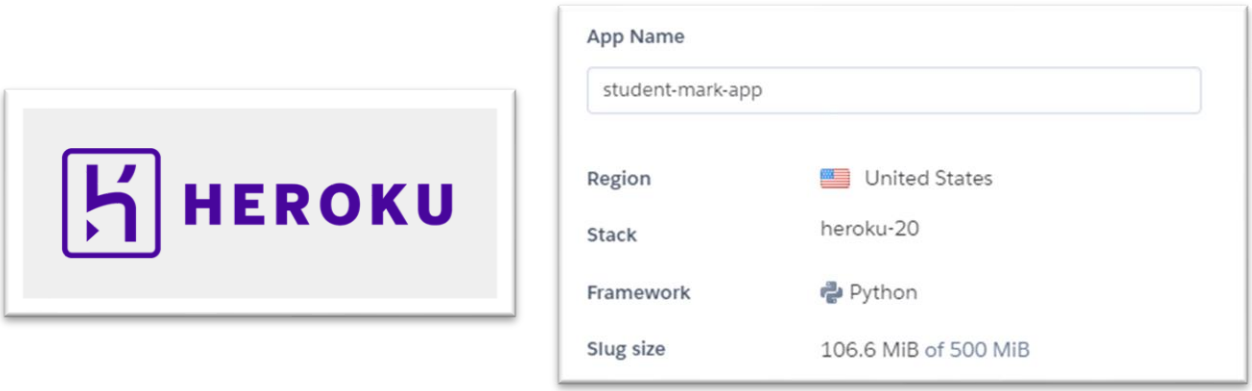
Django is a Python based popular free and open-source web framework. Django enables us to execute all the necessary features that we want in our web application. It is organized very well, and the official documentation and YouTube tutorials provided the necessary help to understand Django's structure and functions. Django also required the use of something called "Jinja" template engine. Template language is the text we use in the source HTML in place of actual data. Instead of fixed values, we place variables or loops using template language in source and let these variables take values according to the backend instructions. For e.g., if an e-commerce website has different product cards, instead of typing each card, one card is written in template language, and it gets the data from backend in for loop and populates the entire page with all the values present in the database.

Now our frontend and backend are taken care of. The next part is database. We are using a remote MySQL database for storing all our student and teacher data. This means that the database is not in our computer but instead it is in a server somewhere on the world connected through Internet. This allows us to access the database from anywhere on the planet if we have Internet connection. As of now, we are using a developmental server for storage. In the future if the project proves to be economically feasible, we will rent our own server instances from one of the cloud providers such as Amazon Web Services, Microsoft Azure, Google Cloud, etc.



F-5: Cloud Services

Now, if we run the Django project that we created in our computer, it will create a website for us, but it will be hosted locally. That means only those who are connected to my computer directly can access that website. No one else could. And therefore, we need to use a web server to store and run our web applications. The very same cloud providers that we saw before also offer services to host web apps. But as expected they do cost some money and we wanted to keep the project free of cost. Thus, we found a free online cloud hosting solution known as **Heroku**. They offer free cloud server for running apps up to *500 Mb* of size. That means, we can host and run our app for free as long as we don't cross the *500 Mb* mark. And sure, we didn't. This means that we have an URL from the Heroku server that can be used anywhere on this world to access our website effortlessly. You only need an Internet connection.



F-6: Heroku

With these said, let us now look at the next section where we have listed all the resources that we used to make this project possible.

Resources

The policy that we prefer is to use as much free and open-source software and libraries as possibly can because we want the project to be cost effective. We also want to have a wide community to rely on in case we face issues or bugs related to the software we use.

The following are the resources that we utilized to develop this project:

I. **Visual Studio Code** – Amazing and Free source code editor
II. **Python** – Our favourite language
III. **Django** – Module that made this possible
IV. **GitHub** – Source code controller

We use GitHub to store all our source code on the internet so that we can access them from different locations. It is also very much convenient and productive to use since it enables us to save progress stepwise and go back to old versions in case anything happens catastrophically.

A. Python Modules

```
asgiref==3.4.1
cycler==0.11.0
dj-database-url==0.5.0
Django==4.0.1
django-heroku==0.3.1
fonttools==4.28.5
gunicorn==20.1.0
install==1.3.5
kiwisolver==1.3.2
matplotlib==3.5.1
mysql-connector==2.2.9
mysqlclient==2.1.0
numpy==1.22.1
packaging==21.3
Pillow==9.0.0
```

```
psycopg2==2.9.3
pyparsing==3.0.7
python-dateutil==2.8.2
python-dotenv==0.19.2
six==1.16.0
sqlparse==0.4.2
tzdata==2021.5
whitenoise==5.3.0
```

Some of these modules are present for making the cloud app work. Some other notable functions performed by them are:

- `Django==4.0.1` – Our backend web framework
- `mysql-connector==2.2.9` – Connector for our database so that Django can retrieve and send info to it
- `Pillow==9.0.0` – It is a Python image processing library
- `python-dotenv==0.19.2`– Used to store confidential data in .env file and prevent it from being pushed to our GitHub repo.

V. Hardware

We used normal computer hardware that was available with us and in school. No additional hardware was required to develop this project.

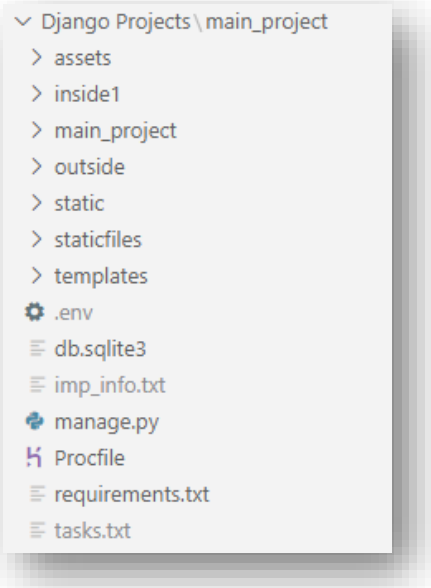
VI. Software Helpers

We used YouTube to learn the basics of Django (Refer *Bibliography*). After that we mostly relied on the official documentation of Django and Google Search for building our project. Open forums such as *Stack Overflow* helped us tremendously to find solutions for our bugs and deployment problems.

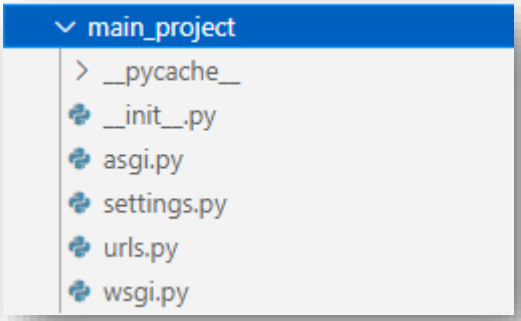
Coding

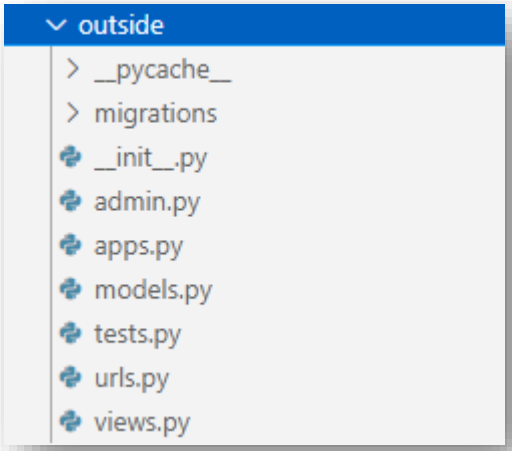
In this section, we showcase all our important pieces of code involved in this project. Some of them have some explanation beneath them for more clarity and understanding.

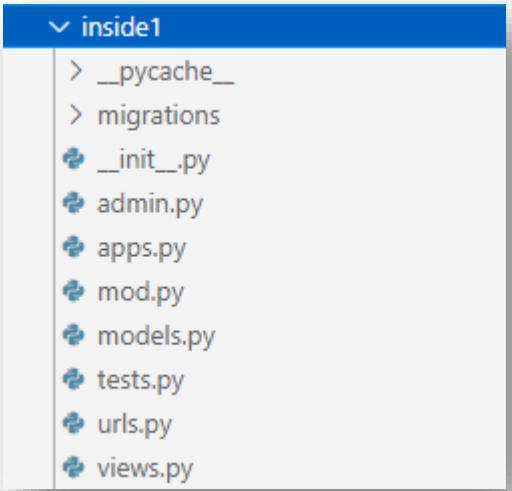
I. Overall Project Structure

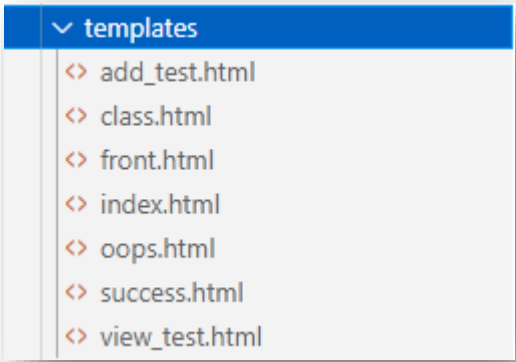


II. Important Sub-Folders in "`Django Projects\main_project`"









III. manage.py - '\Django Projects\main_project\manage.py'

```
#!/usr/bin/env python
"""Django's command-line utility for administrative
tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'main_project.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)


if __name__ == '__main__':
    main()
```

As you can see, using this code only, we would execute all our command line statements to run and manage the server app. Example,

```
(cs_project) D:\CS Project\Code\Django Projects\main_project>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 24, 2022 - 16:43:58
Django version 4.0.1, using settings 'main_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

The *(cs_project)* that you see in front of the address in CMD refers to the Python *Virtual Environment* we are in. Virtual Environment in Python is basically a separate place other than the default Python space where we can install our required modules alone. These modules which are installed are only inside our environment, thus the normal default Python user of this computer would not be able to access or import these modules.

IV. settings.py - `'\Django Projects\main_project\main_project\settings.py'`

```
"""
Django settings for main_project project.


For more information on this file, see
https://docs.djangoproject.com/en/4.0/topics/settings/
"""

from pathlib import Path
from dotenv import load_dotenv
import os
import django_heroku

load_dotenv()

# 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/4.0/howto/deployment/che
cklist/

# SECURITY WARNING: keep the secret key used in production
secret!
# POINT 1 IN EXPLAINATION
SECRET_KEY = str(os.getenv('SECRET_KEY'))

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

ALLOWED_HOSTS = ['student-mark-app.herokuapp.com',
'localhost', '127.0.0.1']


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'inside1.apps.Inside1Config'
]

MIDDLEWARE = [
```

```
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware'
,
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddlewa
re',
    'django.contrib.messages.middleware.MessageMiddleware'
,
    'django.middleware.clickjacking.XFrameOptionsMiddlewar
e',
]

ROOT_URLCONF = 'main_project.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.reques
t',
                'django.contrib.auth.context_processors.au
th',
                'django.contrib.messages.context_processor
s.messages',
            ],
        },
    },
```

```
]

WSGI_APPLICATION = 'main_project.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases
# POINT 2 IN EXPLAINATION

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'sepl3iQWc0',
        'USER': 'sepl3iQWc0',
        'PASSWORD': 'WwVwAgjSgd',
        'HOST': 'remotemysql.com',
        'PORT': '3306'
    }
}


# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
```

```
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthVali
dator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordVal
idator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordVa
lidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-
files/
```

```
STATIC_URL = 'static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
STATIC_ROOT = os.path.join(BASE_DIR, 'assets')

# Default primary key field type
#
https://docs.djangoproject.com/en/4.0/ref/settings/#defaul
t-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

# Activate Django-Heroku.
django_heroku.settings(locals())
```

Refer the code near the comment `# POINT 1 IN EXPLAINATION`

Here the secret key is obtained from the .env file present in our main_project folder. Since .env file is mentioned in .gitignore, it would not be pushed to our repository.

Refer the code near the comment `# POINT 2 IN EXPLAINATION`

This is our details of the remote database that we are using for free for developmental purposes only and not for actual use when the application goes into production.

V. urls.py - `'\Django Projects\main_project\main_project\urls.py'`

```
"""main_project URL Configuration

The `urlpatterns` list routes URLs to views. For more
information please see:
    https://docs.djangoproject.com/en/4.0/topics/http/urls
/
"""
from django.contrib import admin
```

```
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('outside.urls')),
    path('app1/', include('inside1.urls'))
]
```

In this page, we declare the main URL references for our entire web application. The first path 'admin/' is default and provides the URL for the admin page. The home directory or '/' URL is handled by 'outside' app i.e. 'outside.urls'. Similarly, all address starting with 'app1/' is taken care by 'inside1' app. All the further sub-addresses can be found in the respective app's urls.py code.

VI. urls.py - `'\Django Projects\main_project\outside\urls.py'`

Now, we are going to see some important files of the 'outside' app. An app in Django project refers to a group of python programs, that enables some specific functionalities. Here, the app 'outside' is used by us to show the user the front-login page and handle their login requests. We also have planned to add two sections (pages) in the future, one for 'News and Updates' and another for 'About Us'.

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

urlpatterns = [
    path('', views.home, name='Home'),
    path('login', views.login, name='Login')
]
```

Previously we saw that the 'outside' app was linked to '' URL. The sub-addresses from that point onwards are given in this code. This '' or '/' will call *home* function of views.py and '/login' will call the *login* function of views.py.

VII. views.py - '\Django Projects\main_project\outside\views.py'

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth.models import User, auth

def home(request):
    if request.user.is_authenticated:
        return redirect('app1/front')
    else:
        return render(request, 'index.html')

def login(request):
    if request.method == 'POST':
        uname = request.POST['username']
        pswd = request.POST['pswd']

        user = auth.authenticate(username=uname,
password=pswd)

        if user != None:
            auth.login(request, user)
            return redirect('app1/front')
        else:
            messages.info(request, 'Wrong Password or
Username!!')
            return redirect('/')
    else:
        return render(request, 'index.html')
```

In the first function (*home*), we can see that it is redirecting the user to 'app1/front' if the user is found to be authenticated. 'app1/front' is the front page of the main teacher UI. That is where the teachers can see all their classrooms and

start adding or viewing marks. So, in case the teacher is already logged in, i.e., authenticated, then we can directly go to the front page instead of logging in once again from the home page. (Also, once you login, the login details are stored as cookies thus you can go to front page in a device many times without logging in again) If you haven't logged in, the function renders i.e., shows you the home page's HTML which is 'index.html'.

Note: All the HTML files are stored in templates folder. Render method has access to the templates folder in order to fetch the file.

The next function *login* first checks if the request is in POST method. This is because, we have set the login form on the index.html to send a POST request when the user clicks Submit button. Thus, if the method is POST, we first fetch the entered Username and Password and send it to our auth app which is built into Django. If the auth returns a non-empty user, we direct them to our teacher front page. Or else, we send message to the 'messages' object that the credentials are wrong which in turn will be showed in the HTML page. If the request method was not POST, then we simply render the home page and display it using render.

VIII. index.html - '\Django Projects\main_project\templates\index.html'

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <title>Student Marks Application</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/cs
s/bootstrap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/
bootstrap.bundle.min.js"></script>
```

```
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4
.1.1/animate.min.css"/>
    <link rel="stylesheet" href="{% static 'css/main.css'
%}">
</head>

<body>
    <div class="d-flex mt-0 p-3 bg-secondary text-white
justify-content-center animate__animated animate__fadeIn
animate__slow">
        <div class="display-6"><strong>Student Marks
Application</strong></div>
    </div>

    <nav class='mt-0 navbar sticky-top navbar-expand-sm
bg-dark navbar-dark animate__animated animate__fadeInUp'>
        <div class='container-fluid'>
            <a class="navbar-brand" href="/">
                <img src="{% static 'imgs/logo.png' %}"
alt="Logo" style="width:40px;">
            </a>
            <ul class='navbar-nav'>
                <li class="nav-item"><a class="nav-link
active" href="#">Home</a></li>
                <li class="nav-item"><a class="nav-link"
href="#">News and Updates</a></li>
                <li class="nav-item"><a class="nav-link"
href="#">About Us</a></li>
            </ul>
        </div>
    </nav>
    <div class='container-fluid mt-0 p-0 animate__animated
animate__fadeInUp animate__delay-1s'>
```

```
        <div class="carousel slide" data-bs-
ride="carousel">
            <div class="carousel-inner">
                <div class="carousel-item active">
                    <div class="carousel-caption display-
3"><strong>LOGIN</strong></div>
                    <img src="{% static 'imgs/home1.jpg'
%}" alt="Login" class="d-block w-100">
                </div>
            </div>
        </div>
    </div>

    {% if messages %}
    <div class='my-3 mx-5 alert alert-danger alert-
dismissible animate__animated animate__fadeInUp
animate__delay-2s'>
        <button type="button" class="btn-close" data-bs-
dismiss="alert"></button>
        {% for m in messages %}
        <strong>Sorry!</strong> Username and Password did
not match, Try again!
        {% endfor %}
    </div>
    {% endif %}

    <div class='container border border-2 border-primary
mt-3 rounded-3 shadow-lg bg-light p-4 animate__animated
animate__fadeInUp animate__delay-2s'>
        <form action="login" method='post'>
            {% csrf_token %}
            <div class="form-floating mb-3">
                <input type="text" class="form-control"
placeholder="Enter Username" name="username">
```

```
                <label for="username">Username</label>
            </div>
            <div class="form-floating mb-3">
              <input type="password" class="form-control"
placeholder="Enter Password" name="pswd">
              <label for="password">Password</label>
            </div>
            <button type="submit" class="btn btn-
primary">Submit</button>
        </form>
    </div>

    <nav class="navbar mt-5 navbar-expand-sm bg-dark
navbar-dark animate__animated animate__fadeIn">
        <div class="container-fluid">
          <span class="navbar-text mx-auto">A Creation By
Arjun And His Team</span>
        </div>
    </nav>
</body>
</html>
```

This is the index.html i.e., the home page of our web application. The jinja template language can be seen here in some places such as:

- {% for m in messages %}
- {% if messages %}
- {{ m }}
- {% endfor %}
- etc…

This is where the template language comes into use. They allow us to put dynamic things in the HTML page whose data are provided by Django in our views.py module. In this case, messages object is directly given to the page from Django instead of manual addition in views.py. It is a functionality of Django.

You can also notice the use of classes for most of the HTML elements. This is all bootstrap classes. They allow us to beautify the website without extensive CSS coding. Instead, we modify these classes to get the appearance that we want on our site.

IX. models.py - '\Django Projects\main_project\inside1\models.py'

Now we are entering the most important part of this project – the core. We are now looking at the important files of 'inside1' app. This app is right now responsible for all the internal functionalities of the web application such as displaying classes, adding marks, displaying marks etc.

```
# from tkinter import CASCADE
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from django.db import models
from django.contrib.auth.models import User

class Teacher(models.Model):
    user = models.OneToOneField(User,
on_delete=models.CASCADE)
    perm = models.CharField(max_length=30)
    subs = models.CharField(max_length=10)

class Student(models.Model):
    rollno = models.CharField(max_length=5,
primary_key=True)
    name = models.CharField(max_length=30)
    initial = models.CharField(max_length=5)
    address = models.TextField()
    phone1 = models.CharField(max_length=12)
    phone2 = models.CharField(max_length=12)
    std = models.CharField(max_length=2)
    sec = models.CharField(max_length=1)
    bg = models.CharField(max_length=3)
```

```
class Subject(models.Model):
    sub_code = models.CharField(max_length=2,
primary_key=True)
    sub_name = models.CharField(max_length=30)

class Test(models.Model):
    test_id = models.AutoField(primary_key=True)
    test_name = models.CharField(max_length=40)
    max_marks = models.CharField(max_length=50)
    test_subs = models.CharField(max_length=30)
    test_class = models.CharField(max_length=2)

class Marks(models.Model):
    student = models.ForeignKey(Student,
on_delete=models.CASCADE)
    test = models.ForeignKey(Test,
on_delete=models.CASCADE)
    sub = models.ForeignKey(Subject,
on_delete=models.CASCADE)
    marks = models.CharField(max_length=4)
```

As you might notice on the first line, Tkinter is commented out. Why so? It is because, Tkinter is not supported in Heroku server which is going to run our app. So instead, we had to do some alterations to make the app function inside Heroku.

In this file, we have we created our database models. These are in the form of Classes in Python. Each class here corresponds to a table in our database with the same name. The attributes that we have given to these classes are going to be the model reference fields of each table.

For e.g., refer class Student. This corresponds to the table created in the database. The fields of the table are as given: rollno, name, initial, address, std,

sec, etc. Each of their value type is also mentioned in the code. Also, rollno is set as the Primary Key for this table.

For convenience purpose, the class Teacher is linked with the User object. Thus, it can be considered as an extension of User object. Another thing to note is that, in the class Marks, the following are Foreign Keys and are referring to Primary Keys of different tables: *student*, *test* and *sub*.

The actual tables of these models are shown in the upcoming section below.

X. urls.py - '\Django Projects\main_project\inside1\urls.py'

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

urlpatterns = [
    path('front', views.front, name='Front'),
    path('logout', views.logout, name='Logout'),
    path('front/<str:code>', views.class_view,
name='Class'),
    path('front/<str:code>/add/<str:test_id>',
views.add_test, name='Add Test'),
    path('front/<str:code>/add/<str:test_id>/submit',
views.submit_test, name='Submit Test'),
    path('front/<str:code>/view/<str:test_id>',
views.view_test, name='View Test')
]
```

Same functionality as the previous urls.py. Only difference is that these are now sub-addresses of the path 'app1/'.

XI. mod.py - '\Django Projects\main_project\inside1\mod.py'

```
from django.contrib.auth.models import User

def classes(request):
    u = User.objects.get(username=request.user.username)
```

```
    perms = u.teacher.perm
    l = perms.split('-')
    grade = {'0':'9', '1':'10', '2':'11', '3':'12'}
    sec = {'0':'A', '1':'B', '2':'C', '3':'D', '4':'E'}
    classes = []
    for i in l:
        c = (grade[i[0]], sec[i[1]], i)
        classes.append(c)
    return classes

def code_to_class(code):
    grade = {'0':'9', '1':'10', '2':'11', '3':'12'}
    sec = {'0':'A', '1':'B', '2':'C', '3':'D', '4':'E'}
    c = (grade[code[0]], sec[code[1]])
    return c

def class_alone(request):
    u = User.objects.get(username=request.user.username)
    perms = u.teacher.perm
    l = perms.split('-')
    grade = {'0':'9', '1':'10', '2':'11', '3':'12'}
    classes = []
    for i in l:
        c = grade[i[0]]
        if c not in classes:
            classes.append(c)
    return classes
```

This is a module for some general functions that we would be using in views.py of this app. These functions are used to figure out the teacher's assigned classes and sections from the code value which is stored. For e.g., we could convert perm = '31-32' as Class 12-B and Class 12-C respectively.

XII. views.py - '\Django Projects\main_project\inside1\views.py'

```
import http
from http.client import HTTPResponse
import re
from django.http import HttpResponse
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, auth
from inside1.mod import classes, code_to_class,
class_alone
from inside1.models import Test, Marks, Student, Subject

class Class():
    std : str
    sec : str
    code : str

# This function provides the front page for the teacher
def front(request):
    if request.user.is_authenticated:
        l = classes(request)
        classes_class = []
        for i in l:
            c = Class()
            c.std, c.sec, c.code = i
            classes_class.append(c)
        return render(request, 'front.html',
{'classes':classes_class})
    else:
        return render(request, 'oops.html')

# This shows the class' tests details for each class
def class_view(request, code):
```

```
    if request.user.is_authenticated and (code in
User.objects.get(username=request.user.username).teacher.p
erm):
        std, sec = code_to_class(code)
        tests = Test.objects.all()
        teacher_subs =
User.objects.get(username=request.user.username).teacher.s
ubs
        teacher_subs = teacher_subs.split('-')
        tests1 = []
        for i in teacher_subs:
            for j in tests:
                if int(i) in eval(j.test_subs) and
(j.test_class == std):
                    tests1.append(j)
        pending_tests = []
        comp_tests = []
        for i in tests1:
            marks = Marks.objects.all().filter(test=i)
            req_marks = []
            for m in marks:
                if m.student.std == std and m.student.sec
== sec:
                    req_marks.append(m)
            if not req_marks:
                pending_tests.append(i)
            else:
                comp_tests.append(i)
        return render(request, 'class.html', {'std':std,
'sec':sec, 'code':code, 'pending':pending_tests,
'comp':comp_tests})
    else:
        return render(request, 'oops.html')
```

```
# This function provides the adding marks page
def add_test(request, code, test_id):
    if request.user.is_authenticated and (code in
User.objects.get(username=request.user.username).teacher.p
erm):
        test = Test.objects.get(test_id = test_id)
        students =
Student.objects.all().filter(std=code_to_class(code)[0],
sec=code_to_class(code)[1])
        return render(request, 'add_test.html',
{'test':test, 'students':students, 'code':code})
    else:
        return render(request, 'oops.html')

# This function shows a confirmation when marks are added
def submit_test(request, code, test_id):
    if request.user.is_authenticated and (code in
User.objects.get(username=request.user.username).teacher.p
erm):
        test = Test.objects.get(test_id = test_id)
        students =
Student.objects.all().filter(std=code_to_class(code)[0],
sec=code_to_class(code)[1])
        c = 0
        for s in students:
            m = Marks()
            m.student = s
            m.test = test
            m.sub =
Subject.objects.get(sub_code=User.objects.get(username=req
uest.user.username).teacher.subs)
            m.marks = request.POST[s.name]
            m.save()
            c += 1
```

```
        return render(request, 'success.html', {'c':c,
'code':code})
    else:
        return render(request, 'oops.html')

# This function shows the previously entered details
def view_test(request, code, test_id):
    if request.user.is_authenticated and (code in
User.objects.get(username=request.user.username).teacher.p
erm):
        std, sec = code_to_class(code)
        test = Test.objects.get(test_id=test_id)
        marks = Marks.objects.all()
        req_marks = []
        students = Student.objects.all().filter(std=std,
sec=sec)
        sub =
Subject.objects.all().filter(sub_code=User.objects.get(use
rname=request.user.username).teacher.subs)
        for m in marks:
            if (m.student in students) and (m.sub in sub):
                req_marks.append(m)
        return render(request, 'view_test.html',
{'test':test, 'marks':req_marks, 'code':code})

# This function logs out an active user from the app
def logout(request):
    auth.logout(request)
    return redirect('/')
```

Like the previous views.py of ‘outside’ app, here also the views.py decides what to do when a user request some specific function. Refer the code above here for all the things that the application is providing the user wherever required.

Note: When a user tries to go into an ‘inside1’ app URL without logging in, we have created if clauses in each of these functions to show them ‘oops.html’ in those times. This is because we don’t want someone without authorisation to view even the empty pages with no data.

XIII. front.html - `'\Django Projects\main_project\templates\front.html’`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>App Home</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/cs
s/bootstrap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/
bootstrap.bundle.min.js"></script>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4
.1.1/animate.min.css"/>
</head>
<body>

    <nav class='mt-0 navbar sticky-top navbar-expand-sm
bg-dark navbar-dark animate__animated animate__fadeInUp'>
        <div class='container-fluid'>
            <div class="navbar-brand">
                Hi, {{ user.first_name }}!
            </div>
            <ul class='navbar-nav'>
```

```
                <li class="nav-item"><strong><a
class="nav-link active"
href="/app1/front">Front</a></strong></li>
                <li class="nav-item"><strong><a
class="nav-link"
href="/app1/logout">Logout</a></strong></li>
            </ul>
        </div>
    </nav>

    <div class='display-5 mt-4 ms-3 animate__animated
animate__fadeInUp animate__delay-1s'>Your Classes: </div>
    <div class='animate__animated animate__fadeInUp
animate__delay-2s'>
        {% for c in classes %}
        <div class='container border border-3 rounded-3 m-
4 w-25'
            <div class="card col">
                <div class="card-body">
                <h4 class="card-title">Class {{c.std}}-
{{c.sec}}</h4>
                <p class="card-text">This is Class
{{c.std}}-{{c.sec}}!</p>
                <a href="front/{{c.code}}" class="btn btn-
primary">Edit Marks</a>
                </div>
            </div>
        {% endfor %}
    </div>
</body>
</html>
```

This is the HTML source of front page. All the elements are self-explanatory.

XIV. class.html - '\Django Projects\main_project\templates\class.html'

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Class {{ std }}-{{ sec }}</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/cs
s/bootstrap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/
bootstrap.bundle.min.js"></script>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4
.1.1/animate.min.css"/>
</head>
<body>
    <nav class='mt-0 navbar sticky-top navbar-expand-sm
bg-dark navbar-dark'>
        <div class='container-fluid'>
            <div class="navbar-brand">
                <strong>Class {{ std }}-{{ sec }}</strong>
            </div>
            <ul class='navbar-nav'>
                <li class="nav-item"><strong><a
class="nav-link"
href="/app1/front">Front</a></strong></li>
                <li class="nav-item"><strong><a
class="nav-link"
href="/app1/logout">Logout</a></strong></li>
```

```
            </ul>
        </div>
    </nav>
    <div class='container-fluid p-0 mt-4'>
        <div class="d-grid mx-2">
            <button type='button' class='btn btn-primary
btn-lg btn-block' data-bs-toggle="collapse" data-bs-
target="#c1"><strong>Pending Test(s) to be added
▼</strong></button>
        </div>

        <div id="c1" class="collapse row border border-2
rounded-3 p-4 mx-2">

            {% if not pending %}

                <div class="alert alert-success">
                    <strong>Hurray!</strong> You are up to
date! No tests to be added as of now!!
                </div>

            {% else %}

                {% for t in pending %}
                <div class="list-group col">
                    <a href="{{code}}/add/{{t.test_id}}"
class="list-group-item list-group-item-action list-group-
item-info"><b>{{t.test_name}}</b></a>
                </div>
                {% endfor %}

            {% endif %}

        </div>
```

```
    </div>

    <div class='container-fluid mt-4 p-0'>
        <div class="d-grid mt-4 mx-2">
            <button type='button' class='btn btn-primary
btn-lg btn-block' data-bs-toggle="collapse" data-bs-
target="#c2"><strong>Added Tests ▼</strong></button>
        </div>

        <div id="c2" class="collapse row border border-2
rounded-3 p-4 mx-2">

            {% if not comp %}

                <div class="alert alert-warning">
                    <strong>None!</strong> No tests added
yet!
                </div>

            {% else %}

                {% for t in comp %}
                <div class="list-group col">
                    <a href="{{code}}/view/{{t.test_id}}"
class="list-group-item list-group-item-action list-group-
item-info"><b>{{t.test_name}}</b></a>
                </div>
                {% endfor %}

            {% endif %}
        </div>
    </div>

</body>
```

```
</html>
```

This is the HTML source of class view page. All the elements are self-explanatory.

Note: The tabs and indentation in HTML does not mean anything mostly. They are present here for readability and understanding purposes in VS Code or any other editor on a desktop.

XV. add_test.html - '\Django Projects\main_project\templates\add_test.html'

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Add</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/cs
s/bootstrap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/
bootstrap.bundle.min.js"></script>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4
.1.1/animate.min.css"/>
</head>
<body>
    <nav class='mt-0 navbar sticky-top navbar-expand-sm
bg-dark navbar-dark'>
        <div class='container-fluid'>
            <div class="navbar-brand">
                <strong>{{test.test_name}}</strong>
```

```
            </div>
            <ul class='navbar-nav'>
                <li class="nav-item"><strong><a
class="nav-link"
href="/app1/front/{{code}}">Class</a></strong></li>
                <li class="nav-item"><strong><a
class="nav-link"
href="/app1/front">Front</a></strong></li>
                <li class="nav-item"><strong><a
class="nav-link"
href="/app1/logout">Logout</a></strong></li>
            </ul>
        </div>
    </nav>

    <div class='container-fluid p-4'>
        <form action='{{test.test_id}}/submit'
method="post" class='was-validated'>
            {% csrf_token %}

            {% for s in students %}
            <div class="row mt-3">
                <div class="col container bg-secondary
rounded-3 px-2 pt-3 text-white">
                  <h4>{{s.name}}</h4>
                </div>
                <div class="col">
                  <input type="text" class="form-control"
placeholder="Marks" maxlength="4" name="{{s.name}}"
required>
                  <div class="invalid-feedback">Please
fill this.</div>
                  <div class="valid-feedback">Done.</div>
                </div>
```

```
            </div>
            {% endfor %}

            <button type="submit" class="btn btn-primary
mt-4 ms-0">Submit</button>
        </form>
    </div>
</body>
</html>
```

This is the HTML source of adding marks to test page. All the elements are self-explanatory.

XVI. success.html - `'\Django Projects\main_project\templates\success.html'`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Success</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/cs
s/bootstrap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/
bootstrap.bundle.min.js"></script>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4
.1.1/animate.min.css"/>
</head>

<body>
```

```
    <div class="mt-4 mx-4 p-5 bg-success text-white
rounded animate__animated animate__fadeInUp">
        <h1>Added {{c}} students marks!</h1>
        <h4>Go back to class page <a
href='/app1/front/{{code}}'
style='color:white'>here</a></h4>
    </div>
</body>
</html>
```

This is the HTML source of submission page. All the elements are self-explanatory.

XVII. view_test.html - '\Django Projects\main_project\templates\view_test.html'

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>View</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/cs
s/bootstrap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/
bootstrap.bundle.min.js"></script>
    <link rel="stylesheet"
href="https://unpkg.com/bootstrap-
table@1.19.1/dist/bootstrap-table.min.css">
    <script
src="https://cdn.jsdelivr.net/npm/jquery/dist/jquery.min.j
s"></script>
```

```
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/um
d/popper.min.js"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js
/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+
B07jRM" crossorigin="anonymous"></script>
    <script src="https://unpkg.com/bootstrap-
table@1.19.1/dist/bootstrap-table.min.js"></script>

</head>
<body>
    <nav class='mt-0 navbar sticky-top navbar-expand-sm
bg-dark navbar-dark'>
        <div class='container-fluid'>
            <div class="navbar-brand">
                <strong>{{test.test_name}}</strong>
            </div>
            <ul class='navbar-nav'>
                <li class="nav-item"><strong><a
class="nav-link"
href="/app1/front/{{code}}">Class</a></strong></li>
                <li class="nav-item"><strong><a
class="nav-link"
href="/app1/front">Front</a></strong></li>
                <li class="nav-item"><strong><a
class="nav-link"
href="/app1/logout">Logout</a></strong></li>
            </ul>
        </div>
    </nav>

    <div class="container mt-3">
```

```
        <table class='table table-striped table-bordered
table-hover '>
          <thead>
            <tr>
              <th>Name</th>
              <th>Roll No.</th>
              <th>Subject</th>
              <th>Marks</th>
            </tr>
          </thead>
          <tbody>
            {% for m in marks %}
            <tr>
              <td>{{m.student.name}}</td>
              <td>{{m.student.rollno}}</td>
              <td>{{m.sub.sub_name}}</td>
              <td>{{m.marks}}</td>
            </tr>
            {% endfor %}
          </tbody>
        </table>
      </div>

</body>
</html>
```

This is the HTML source of test viewing page. All the elements are self-explanatory.

XVIII.oops.html - `'\Django Projects\main_project\templates\oops.html'`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Oops</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/cs
s/bootstrap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/
bootstrap.bundle.min.js"></script>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4
.1.1/animate.min.css"/>
</head>
    <div class="mt-4 mx-4 p-5 bg-primary text-white
rounded animate__animated animate__fadeInUp">
        <h1>Oops! You don't have access to this page!</h1>
        <h4>Here is the <a href='/'
style='color:white'>home page</a></h4>
    </div>
```

This is the HTML source of Oops page. All the elements are self-explanatory.

Remote Database

In this section, we are going to show you the contents of the remote database that stores all our necessary information. This database is accessed by us using *phpMyAdmin*.

I. Overall Database

Table	Action	Rows	Type	Collation	Size	Overhead
auth_group	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_unicode_ci	32 KiB	-
auth_group_permissions	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_unicode_ci	48 KiB	-
auth_permission	Browse Structure Search Insert Empty Drop	44	InnoDB	utf8_unicode_ci	32 KiB	-
auth_user	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8_unicode_ci	32 KiB	-
auth_user_groups	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_unicode_ci	48 KiB	-
auth_user_user_permissions	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_unicode_ci	48 KiB	-
django_admin_log	Browse Structure Search Insert Empty Drop	23	InnoDB	utf8_unicode_ci	48 KiB	-
django_content_type	Browse Structure Search Insert Empty Drop	11	InnoDB	utf8_unicode_ci	32 KiB	-
django_migrations	Browse Structure Search Insert Empty Drop	23	InnoDB	utf8_unicode_ci	16 KiB	-
django_session	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_unicode_ci	32 KiB	-
inside1_marks	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8_unicode_ci	64 KiB	-
inside1_student	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8_unicode_ci	16 KiB	-
inside1_subject	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8_unicode_ci	16 KiB	-
inside1_teacher	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_unicode_ci	32 KiB	-
inside1_test	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8_unicode_ci	16 KiB	-
15 tables	Sum	133	InnoDB	utf8_unicode_ci	512 KiB	0 B

II. Tables' structure

Server: localhost » Database: sepl3iQWc0 » Table: inside1_teacher

Browse | Structure | SQL | Search | Insert | Export | Import | Operations

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	*None*		AUTO_INCREMENT	Change Drop More
2	perm	varchar(30)	utf8_unicode_ci		No	*None*			Change Drop More
3	user_id	int(11)			No	*None*			Change Drop More
4	subs	varchar(10)	utf8_unicode_ci		No	*None*			Change Drop More

Server: localhost » Database: sepl3iQWc0 » Table: auth_user

Browse | Structure | SQL | Search | Insert | Export | Import | Operations

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	password	varchar(128)	utf8_unicode_ci		No	*None*			Change Drop More
3	last_login	datetime(6)			Yes	*NULL*			Change Drop More
4	is_superuser	tinyint(1)			No	*None*			Change Drop More
5	username	varchar(150)	utf8_unicode_ci		No	*None*			Change Drop More
6	first_name	varchar(150)	utf8_unicode_ci		No	*None*			Change Drop More
7	last_name	varchar(150)	utf8_unicode_ci		No	*None*			Change Drop More
8	email	varchar(254)	utf8_unicode_ci		No	*None*			Change Drop More
9	is_staff	tinyint(1)			No	*None*			Change Drop More
10	is_active	tinyint(1)			No	*None*			Change Drop More
11	date_joined	datetime(6)			No	*None*			Change Drop More

Server: localhost » Database: sepl3iQWc0 » Table: inside1_student

Browse | Structure | SQL | Search | Insert | Export | Import | Operations

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	rollno	varchar(5)	utf8_unicode_ci		No	None			Change Drop More
2	name	varchar(30)	utf8_unicode_ci		No	None			Change Drop More
3	initial	varchar(5)	utf8_unicode_ci		No	None			Change Drop More
4	address	longtext	utf8_unicode_ci		No	None			Change Drop More
5	phone1	varchar(12)	utf8_unicode_ci		No	None			Change Drop More
6	phone2	varchar(12)	utf8_unicode_ci		No	None			Change Drop More
7	std	varchar(2)	utf8_unicode_ci		No	None			Change Drop More
8	sec	varchar(1)	utf8_unicode_ci		No	None			Change Drop More
9	bg	varchar(3)	utf8_unicode_ci		No	None			Change Drop More

Server: localhost » Database: sepl3iQWc0 » Table: inside1_subject

Browse | Structure | SQL | Search | Insert | Export | Import | Operations

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	sub_code	varchar(2)	utf8_unicode_ci		No	None			Change Drop More
2	sub_name	varchar(30)	utf8_unicode_ci		No	None			Change Drop More

Server: localhost » Database: sepl3iQWc0 » Table: inside1_test

Browse | Structure | SQL | Search | Insert | Export | Import | Operations

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	test_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	test_name	varchar(40)	utf8_unicode_ci		No	None			Change Drop More
3	max_marks	varchar(50)	utf8_unicode_ci		No	None			Change Drop More
4	test_subs	varchar(30)	utf8_unicode_ci		No	None			Change Drop More
5	test_class	varchar(2)	utf8_unicode_ci		No	None			Change Drop More

Server: localhost » Database: sepl3iQWc0 » Table: inside1_marks

Browse | Structure | SQL | Search | Insert | Export | Import | Operations

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	*None*		AUTO_INCREMENT	Change Drop More
2	marks	varchar(4)	utf8_unicode_ci		No	*None*			Change Drop More
3	student_id	varchar(5)	utf8_unicode_ci		No	*None*			Change Drop More
4	sub_id	varchar(2)	utf8_unicode_ci		No	*None*			Change Drop More
5	test_id	int(11)			No	*None*			Change Drop More

III. auth_user table

	id	password	last_lo	is_superu	usern	first_n	last_name	email	is_staff	is_active	date_joined
Edit Copy Delete	1	pbkdf2_sha	2022-(	1	admi			arjun@arjun.com	1	1	2022-01-19 05:07:42.559641
Edit Copy Delete	2	pbkdf2_sha	2022-(	0	nithy	Nithya			0	1	2022-01-19 15:57:55.000000

IV. inside1_teacher table

	id	perm	user_id	subs
Edit Copy Delete	1	31-32	2	9

V. inside1_student table

	rollno	name	initial	address	phone1	phone2	std	sec	bg
Edit Copy Delete	12B01	Arun	F	47, Whites Road, Royapeetah, Chennai, Tamil Nadu,...	04428514337	None	12	B	O+
Edit Copy Delete	12B02	Bala	A	Cenotaph Road, T Nagar Chennai, Tamil Nadu, 60001...	04424338412	None	12	B	O+
Edit Copy Delete	12B03	Kiran	D	130, Kodambakkam Hih Road, Nungambakkam Chennai, ...	04428231897	None	12	B	A1+
Edit Copy Delete	12C01	David	F	EARTH	123456789	None	12	C	O+
Edit Copy Delete	12C02	Dabid	D	Planet Earth	1234	None	12	C	A1+

VI. inside1_subject table

	sub_code	sub_name
Edit Copy Delete	0	English
Edit Copy Delete	10	KTPI
Edit Copy Delete	11	Accountancy
Edit Copy Delete	12	Economics
Edit Copy Delete	13	Business Studies
Edit Copy Delete	5	Physics
Edit Copy Delete	6	Chemistry
Edit Copy Delete	7	Maths-Standard
Edit Copy Delete	8	Maths-Applied
Edit Copy Delete	9	Computer Science

VII. inside1_test table

	test_id	test_name	max_marks	test_subs	test_class
Edit Copy Delete	1	Pre-term Test 1	0:40, 5:35, 6:35, 7:40, 9:35	[0,5,6,7,9]	12
Edit Copy Delete	2	Pre term test 2	9:45	[9, 0]	12

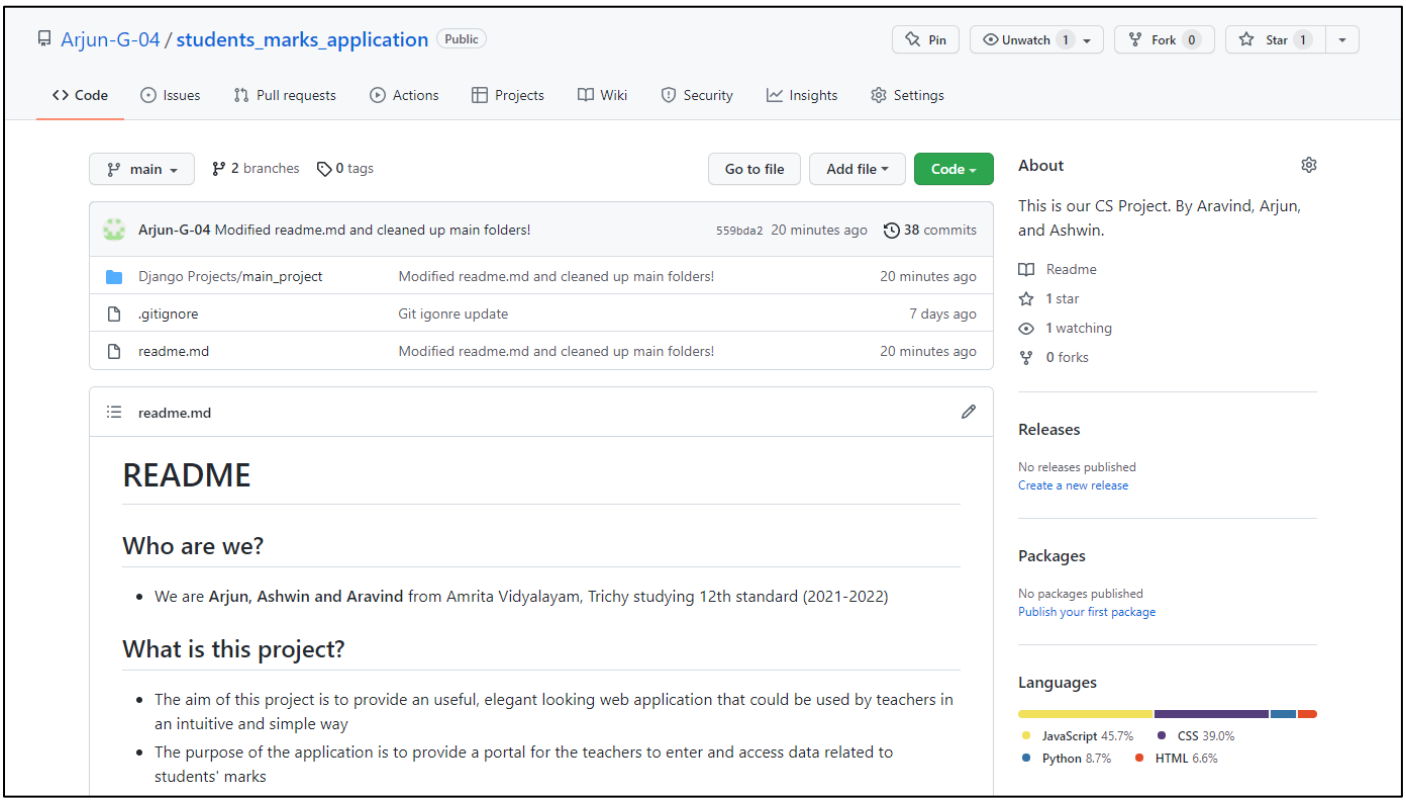
VIII. inside1_marks table

	id	marks	student_id	sub_id	test_id
Edit Copy Delete	2	12	12B01	9	1
Edit Copy Delete	3	12	12B02	9	1
Edit Copy Delete	4	12	12B03	9	1
Edit Copy Delete	5	42	12C01	9	1
Edit Copy Delete	6	30	12C02	9	1
Edit Copy Delete	7	67	12B01	9	2
Edit Copy Delete	8	45	12B02	9	2
Edit Copy Delete	9	36	12B03	9	2

Results

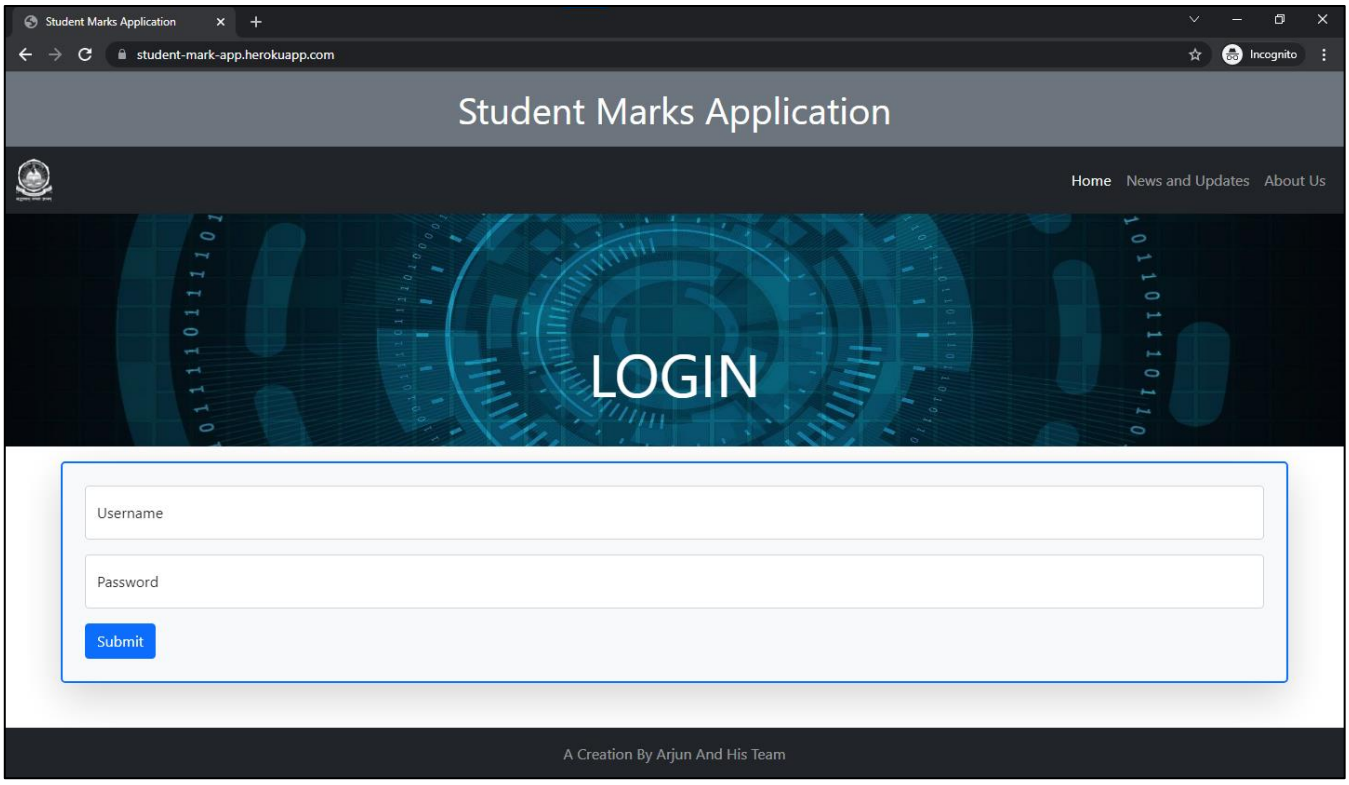
In this section, we are going to look the outcome of this project.

I. GitHub Repo - *https://github.com/Arjun-G-04/students_marks_application*

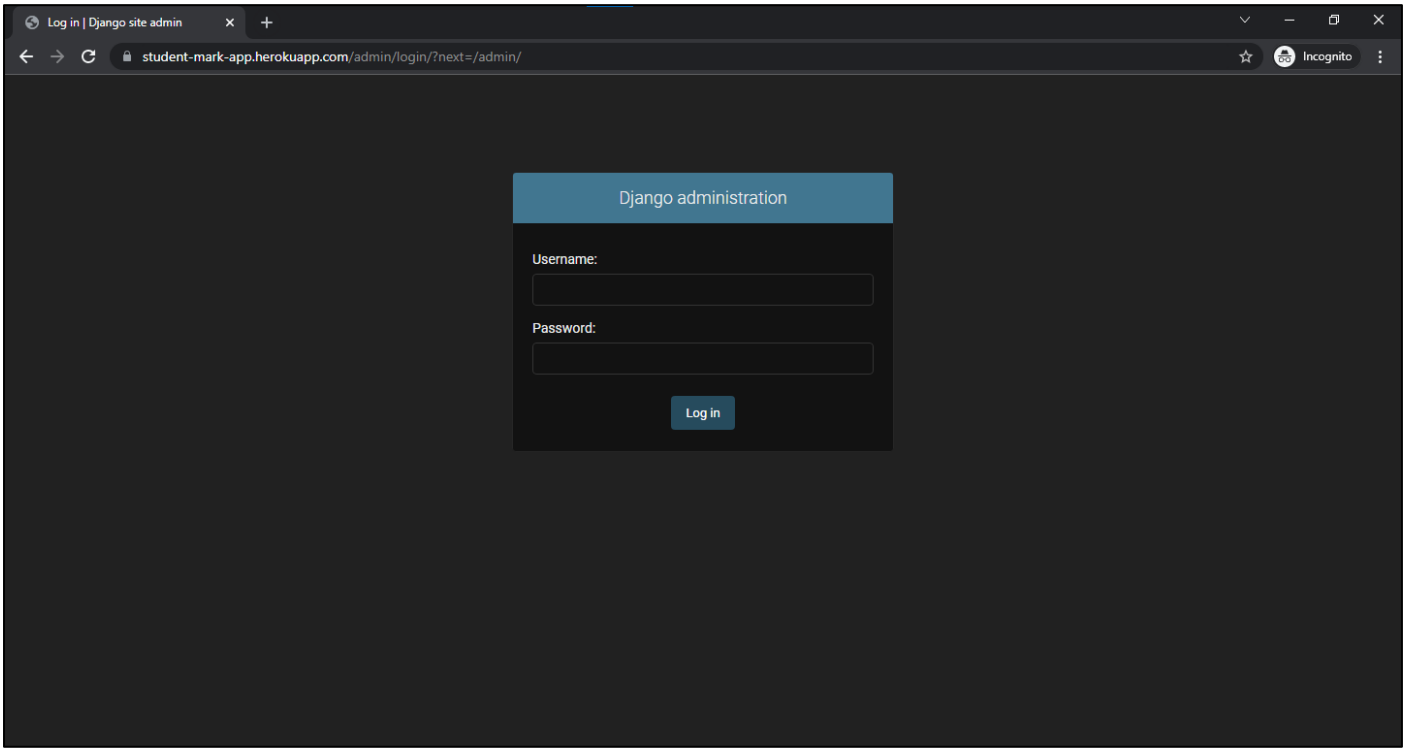


II. Main Website - *https://student-mark-app.herokuapp.com*

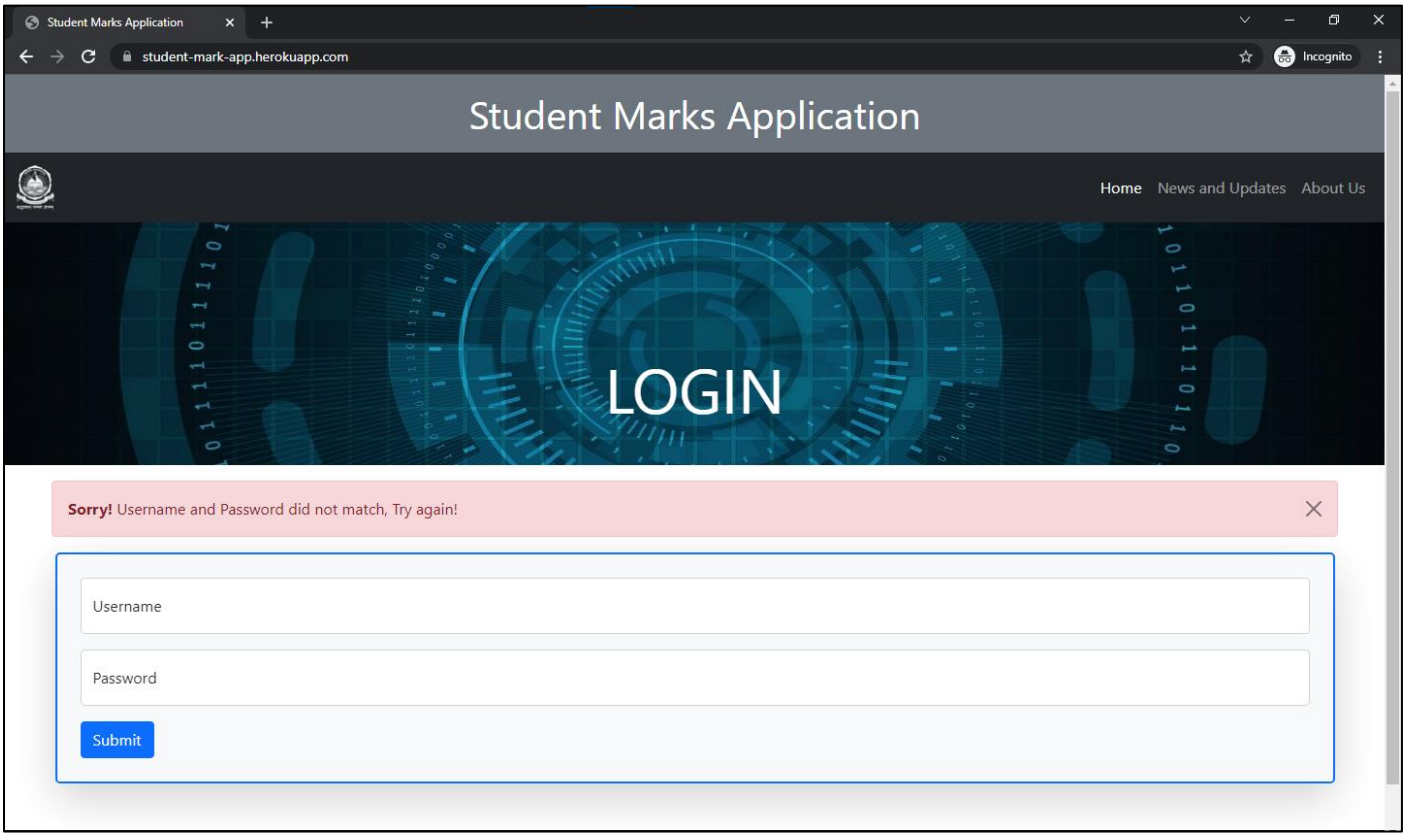
a. Home page - *https://student-mark-app.herokuapp.com/*



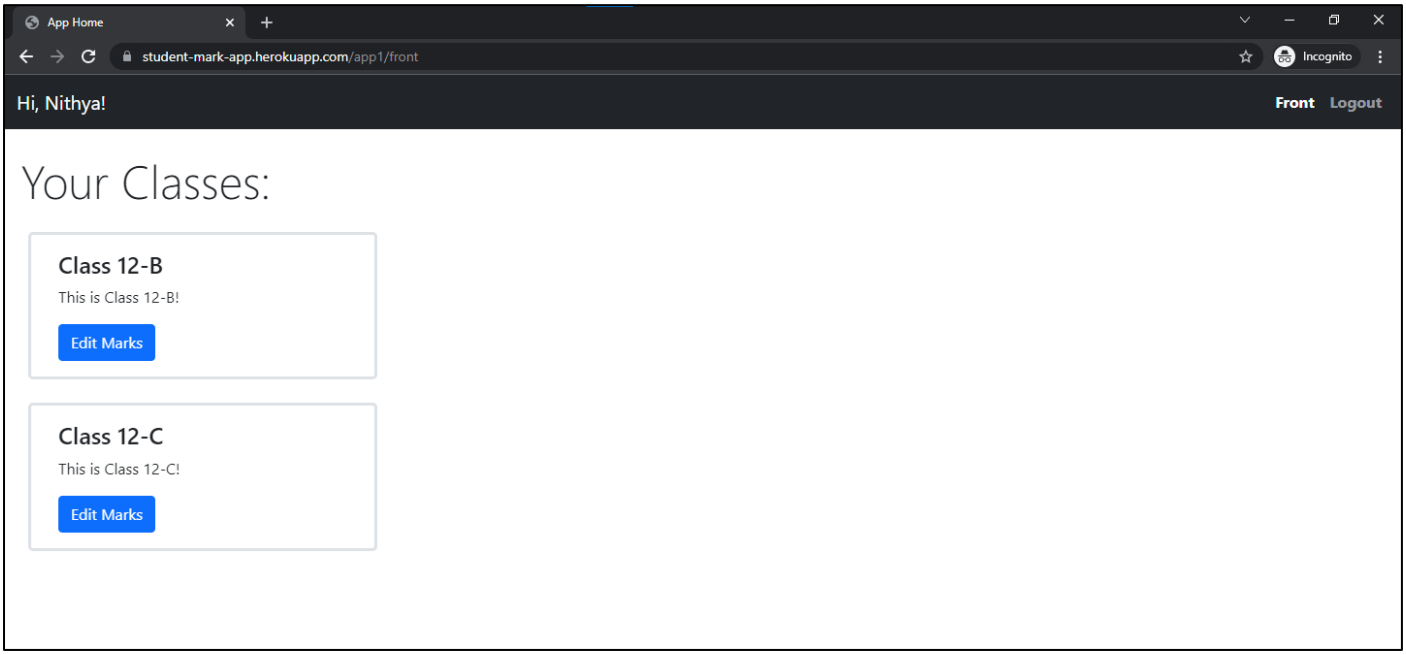
b. Admin login page - */admin/login/?next=/admin/*



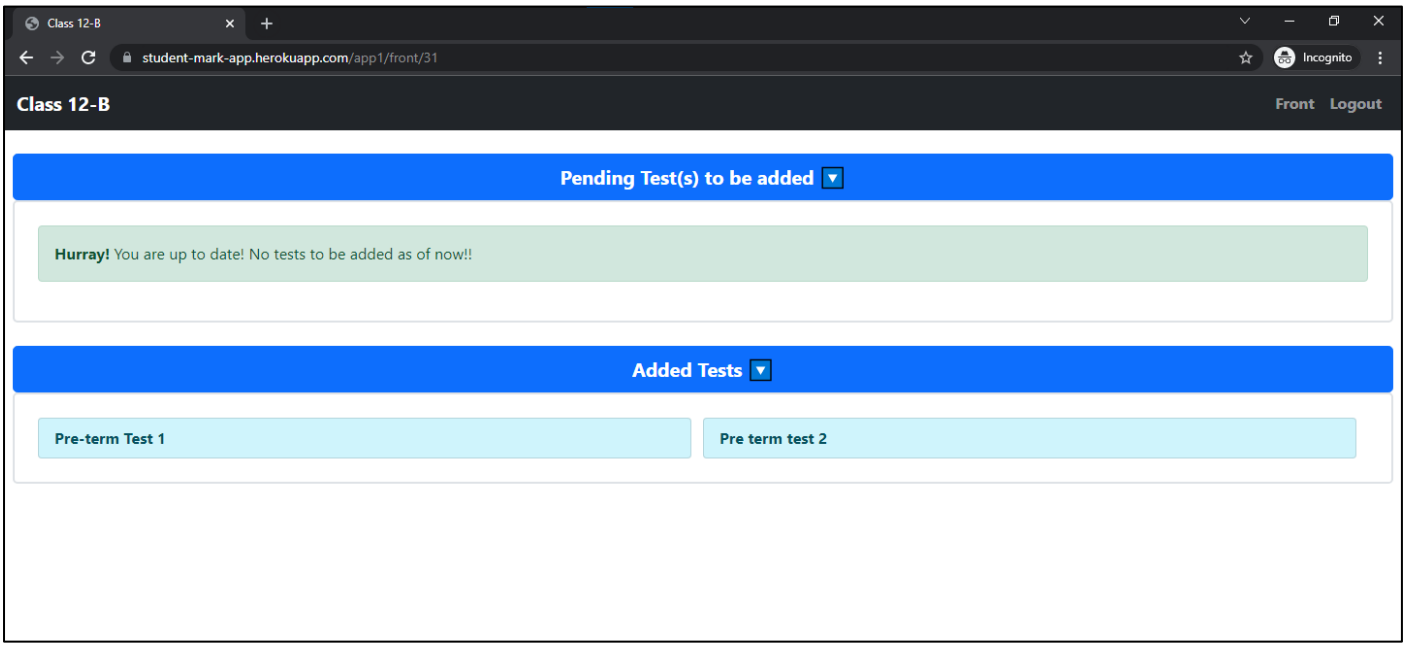
c. Login incorrect page - *https://student-mark-app.herokuapp.com/*



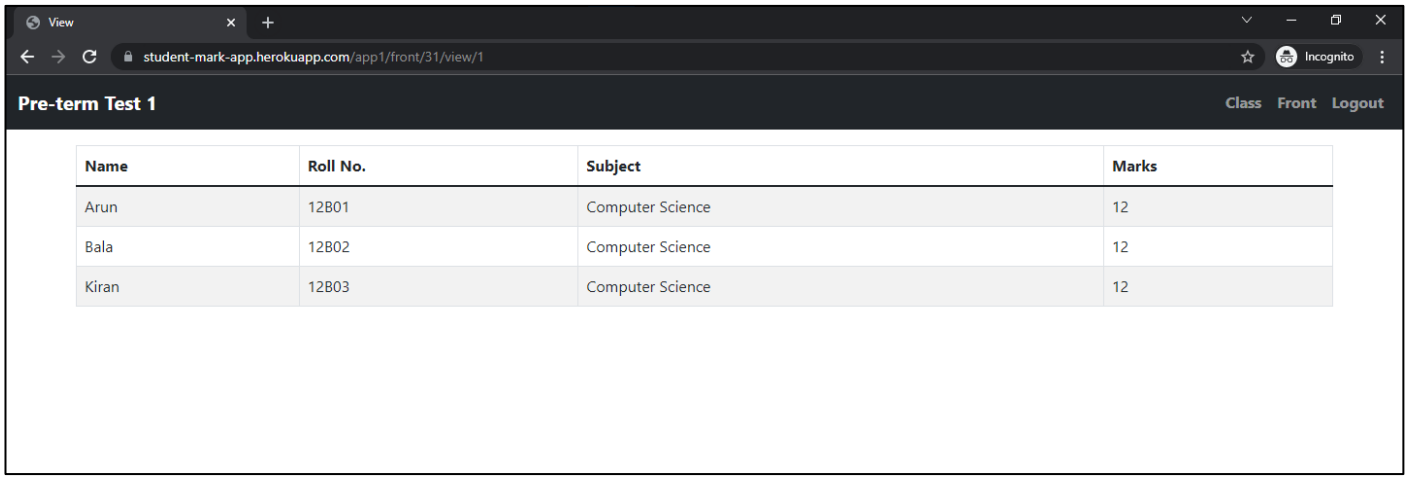
d. Front Page - *<u>/app1/front</u>*



e. Class Page (12-B) - *<u>/app1/front/31</u>*

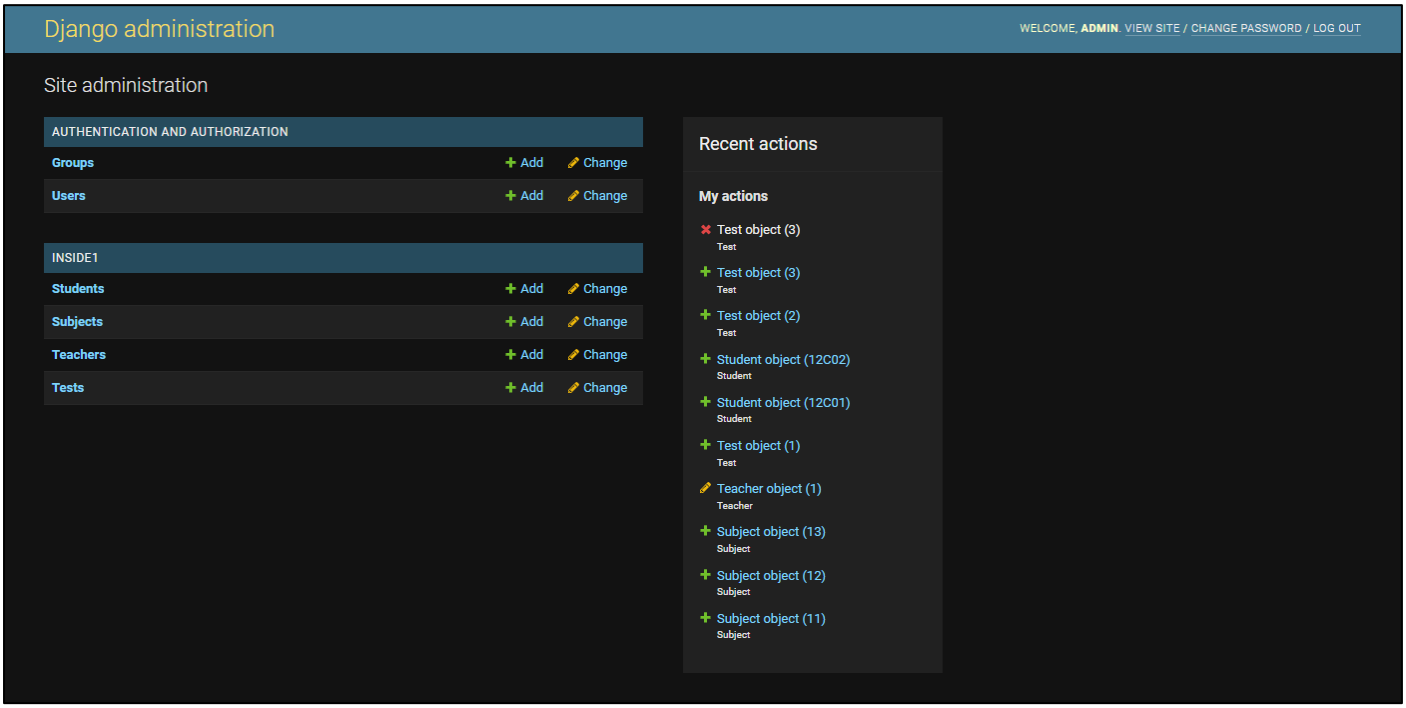


f. Added Test Page (Pre-term Test 1) - *<u>/app1/front/31/view/1</u>*

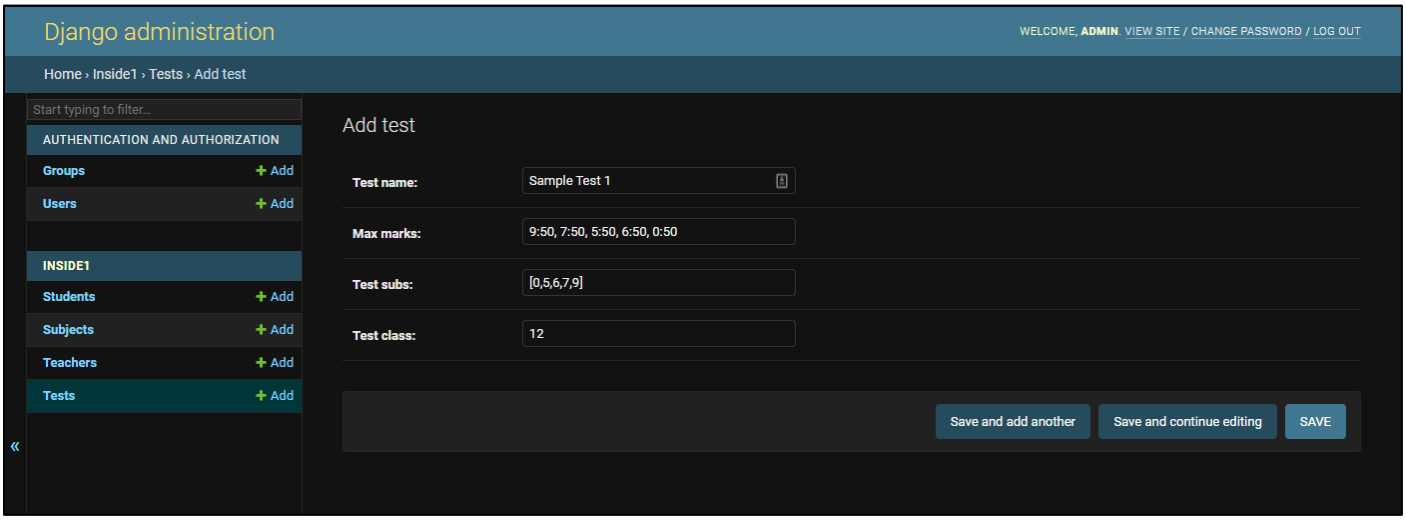


Name	Roll No.	Subject	Marks
Arun	12B01	Computer Science	12
Bala	12B02	Computer Science	12
Kiran	12B03	Computer Science	12

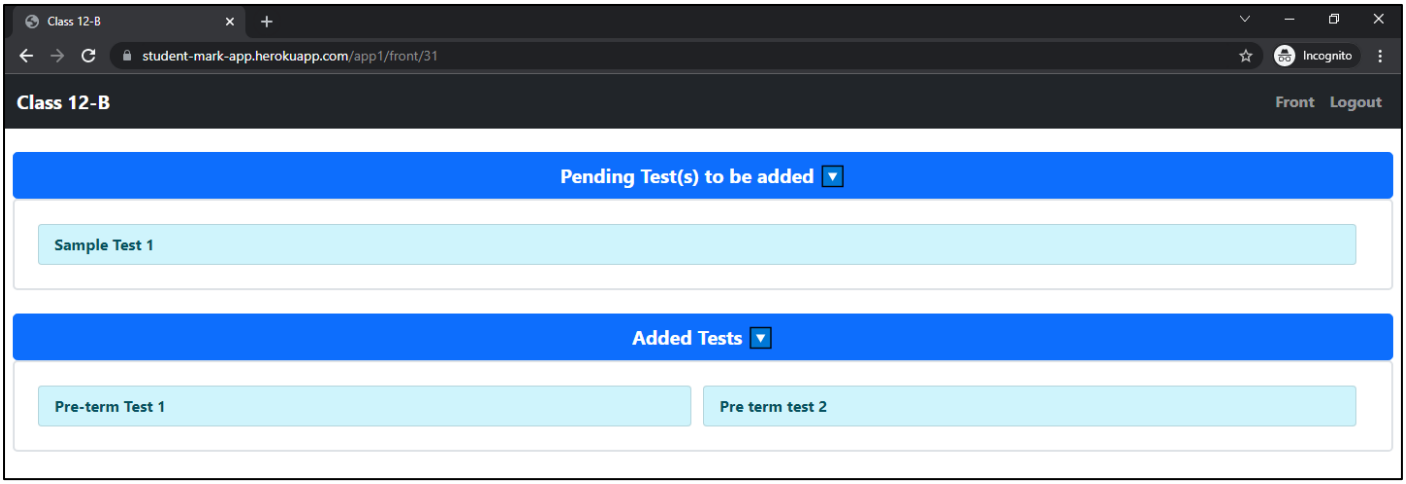
g. Admin home page - */admin*



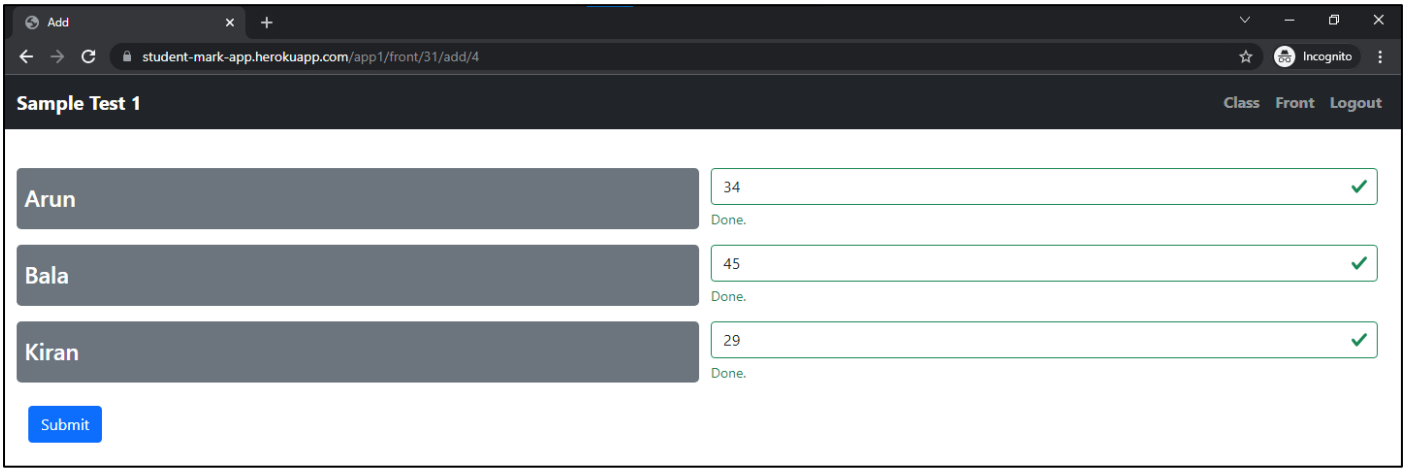
h. Test addition in Admin page - */admin/inside1/test/add/*



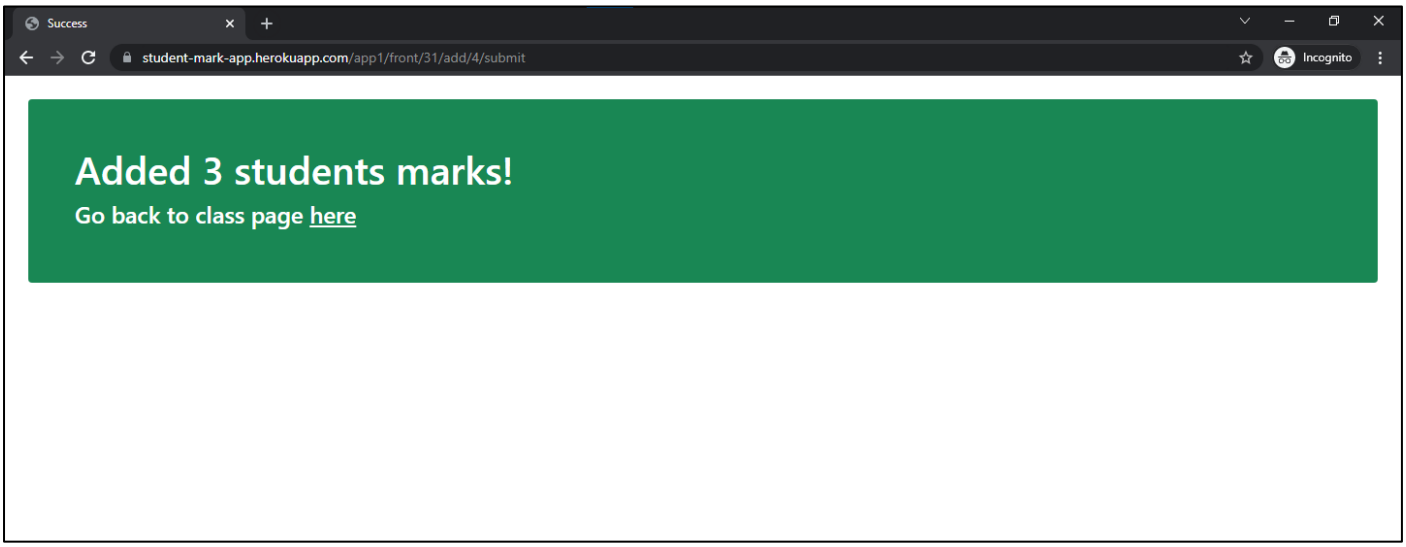
i. New pending test in Class page - */app1/front/31*



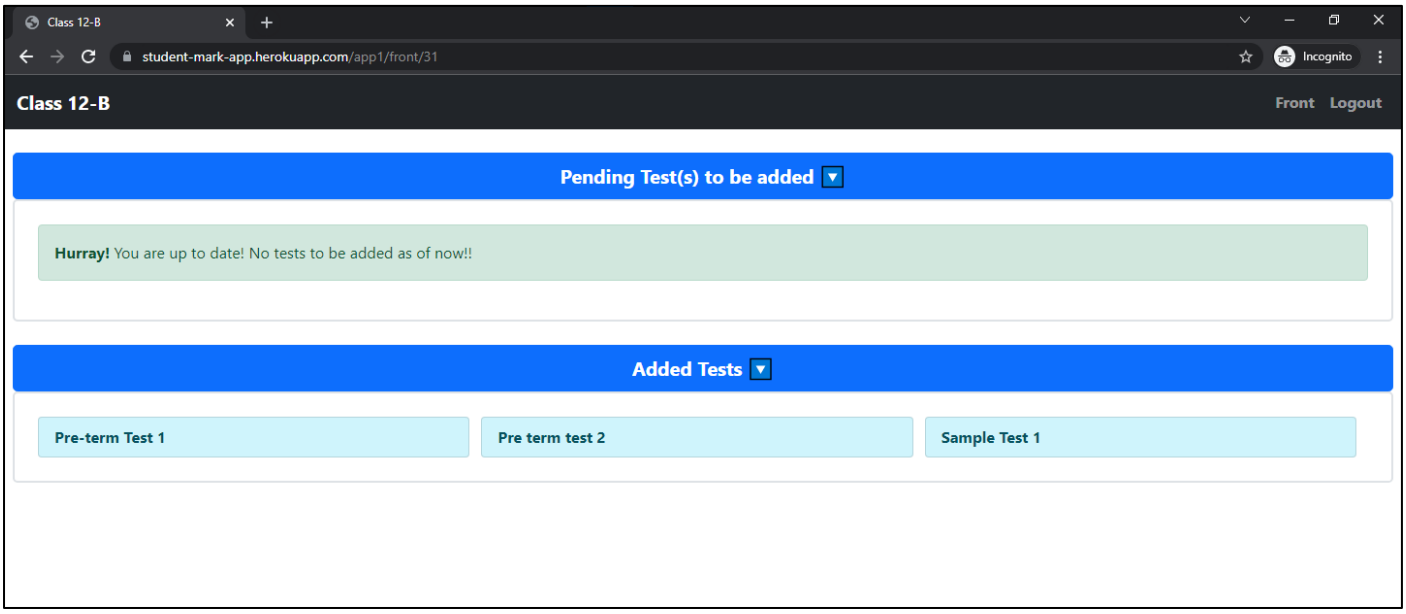
j. Adding marks to test - *<u>/app1/front/31/add/4</u>*



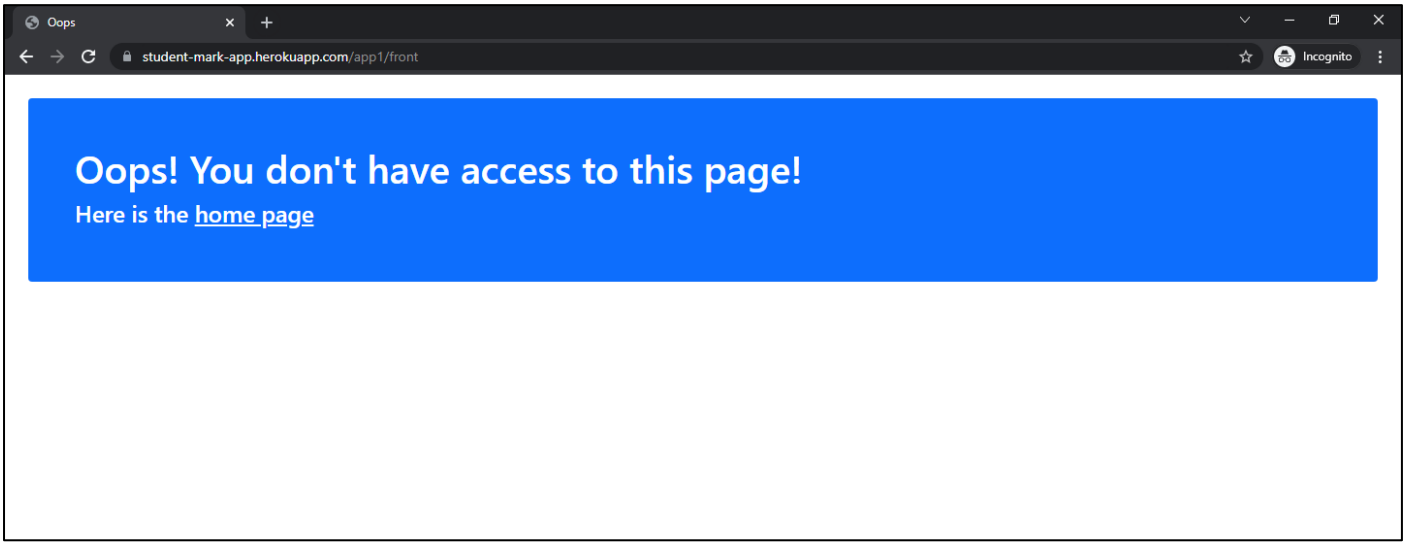
k. Submission Page - *<u>/app1/front/31/add/4/submit</u>*



l. Updated Class Page (12-B) - *<u>/app1/front/31</u>*



m. Oops page - */app1/front/31*



Bibliography

- **YouTube Videos List** - https://bit.ly/3Fd7gk8
- **Bootstrap 5 Tutorial** - https://bit.ly/3tYHA8w
- **Stackoverflow** - https://bit.ly/3o0AyMH
- **Computer Science With Python Textbook For Class 12** - by Sumita Arora