Student online application documentation.

Instruments and methods API Integration Testing:

- Microsoft Visual Studio – Integrated Development Environment (IDE) for application development.
- MySQL – Relational database for storing user information.
- Windows 10 Operating System (Recommended) – Since python commands differ between and for easy integration and compatibility with other machines.
- Python programming language and Django framework (personal preference) for backend programming instead of node.js.
- Html and CSS (personal preference) for frontend programming instead of react.

Note that this documentation focuses only on testing components of the system while being developed to meet the final requirements.

Python commands used:

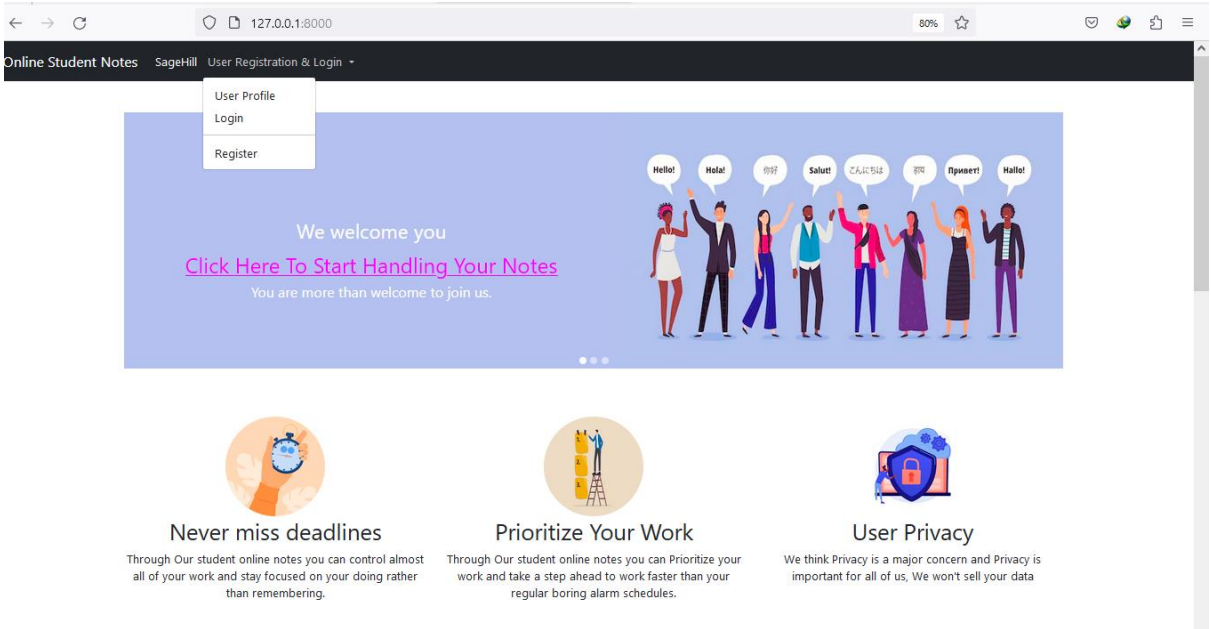
These commands work on python 3.11.2 and pip 22.3.1 and might not work on older or newer version of pip

1. Virtual environment creation – (py -m venv naruto) where naruto is the name of the virtual environment.
2. Virtual environment activation – (naruto\Scripts\activate.bat)
3. Requirements installation – (py -m pip install -r requirements.txt)
4. Running Migrations – (py -m manage migrate)
5. Creating a super user – (py -m manage createsuperuser)
6. Running the server – (py -m manage runserver)

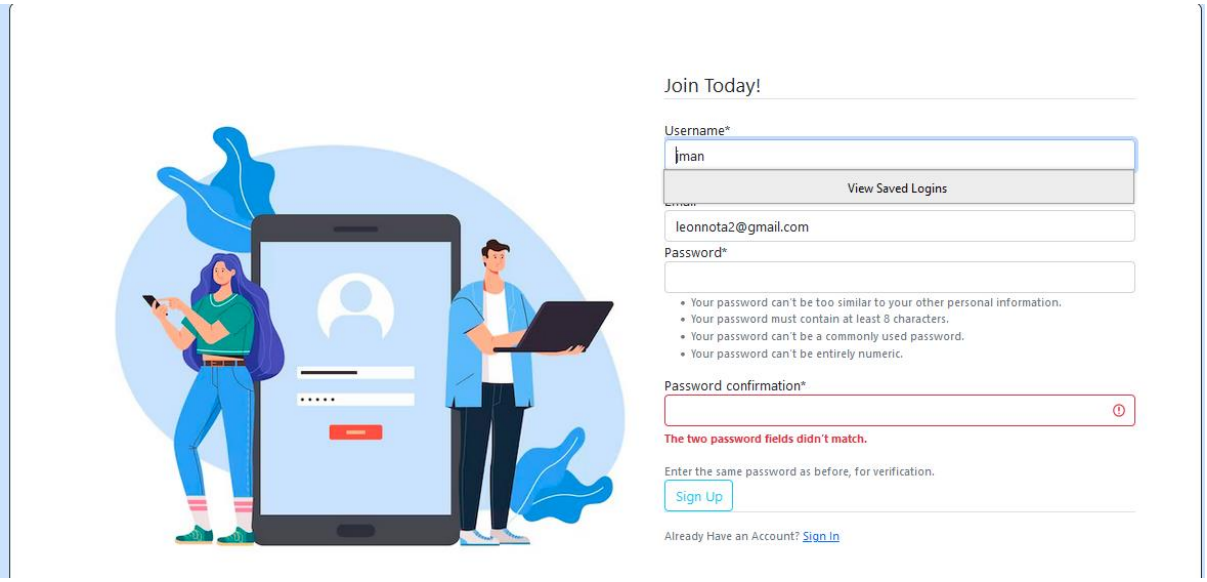
All the codes are case sensitive and within the README file there has been provided an extra code for a different version of pip running on Windows Operating System as well.

Front-end Unit Testing:

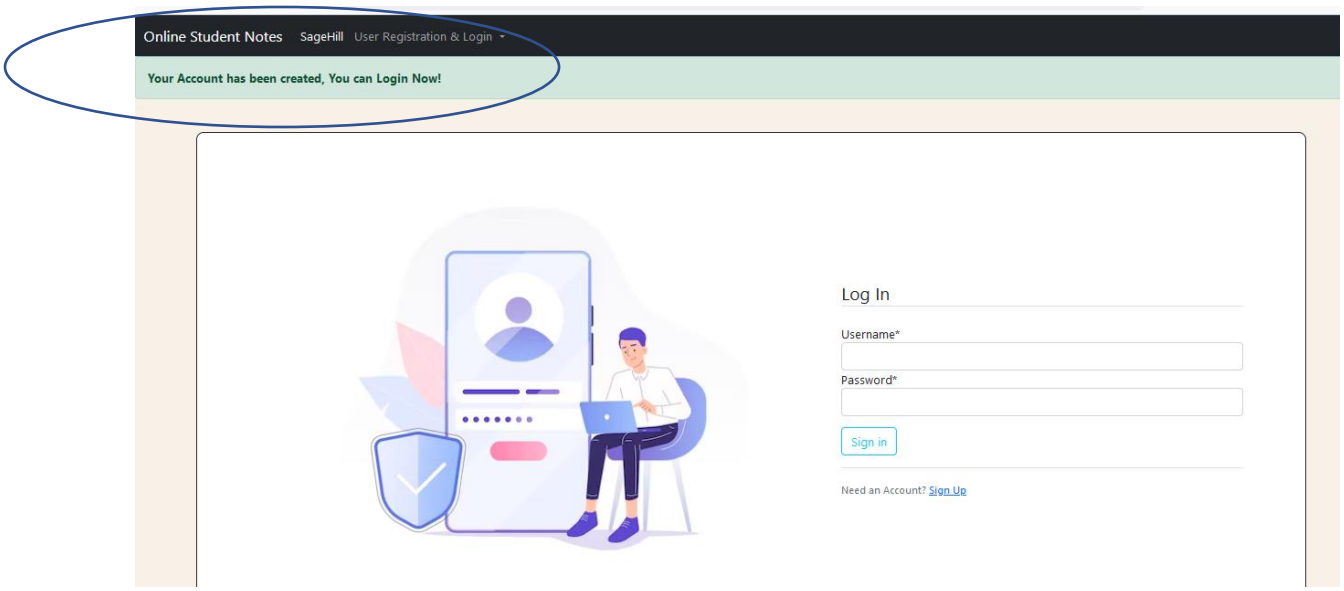
At this phase there was use of modular programming where each component was built and tested for its proper functionality starting with user registration as shown by the image below. It is important to note that the link on the dashboard slider may only direct you to your notes if you are registered otherwise it will first direct you to the log in page.



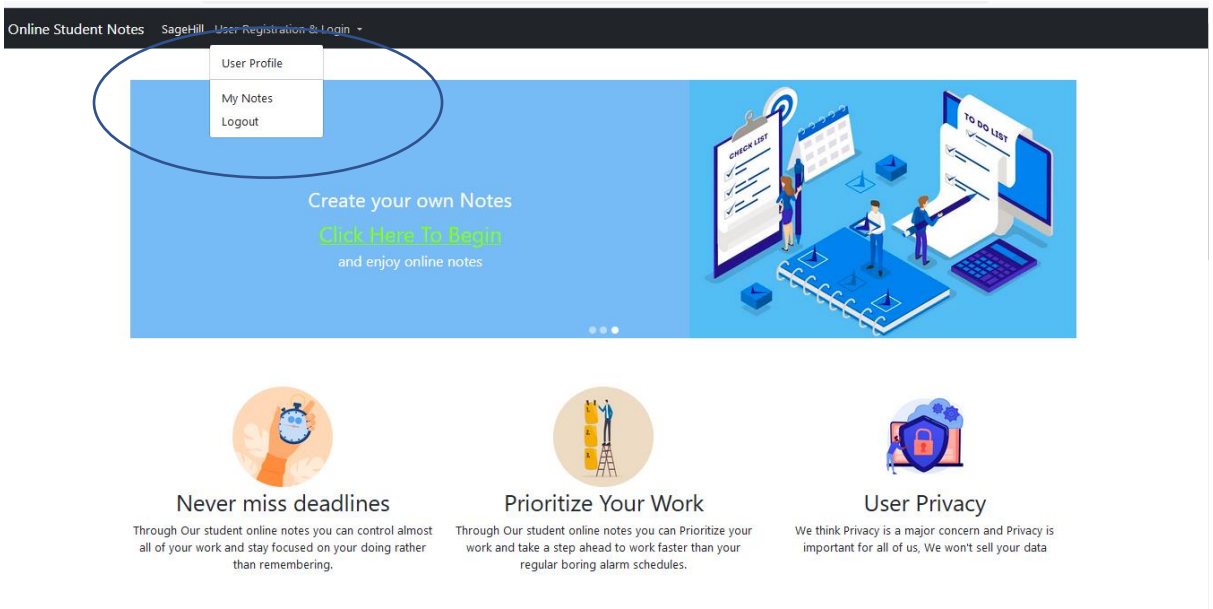
Firstly the user opens the server dashboard then move on to account registration which is shown below and if the user fails to validate their password by following proper guidelines of creating a strong password then account creation fails. Same goes for inserting a non existent email address on the email text field



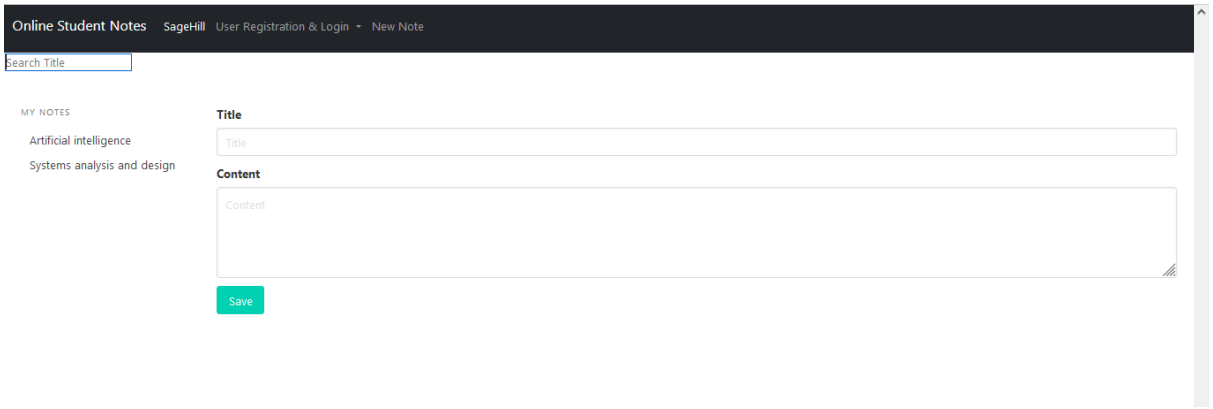
When proper procedures are followed a successful prompt message will be shown above the web page so as to notify the user that they can now log in and will be automatically directed to the Log in page as shown below.



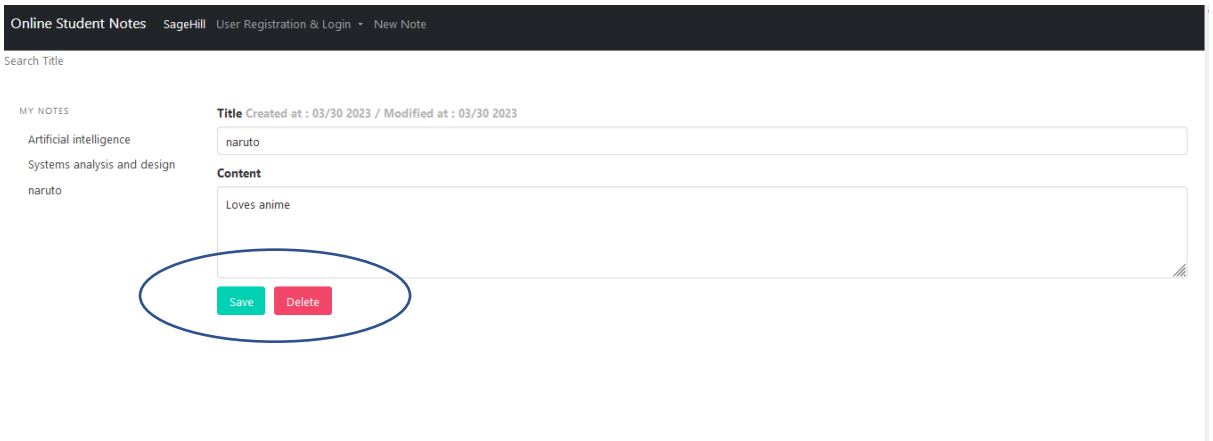
When a user logs in, s/he will be directed to a page that seems similar to the startup page however its completely different as now the drop box for User registration and login will allow the user to view notes, profile and logout as shown below and its different from the one shown on the drop box of the main dashboard on fig 1



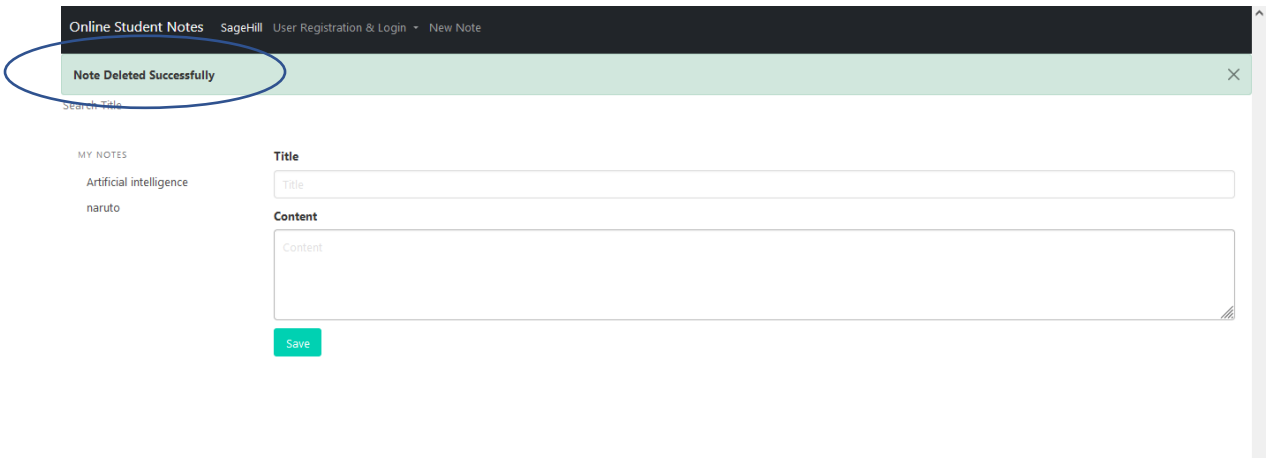
Below shows *my notes* section where a user creates, edits and delete unwanted notes.



After a user creates a not it will be saved under my notes section in the left sidebar. Also a prompt *New Note* notification will appear on the task bar as well as the delete button after creation of a Note.



A user can open an already saved note and edit it, if they delete it they will be notified on the task bar of the success in deleting the unwanted note as shown below



Backend unit testing:

Below shows the code of unit testing the views.py module while the program was being developed. Restful API was used so as to meet the system requirements. All the necessary python functions that handle http requests are listed here.

```python
from home.models import Contact
from django.shortcuts import render, HttpResponse
from datetime import datetime
from django.contrib import messages
# Create your views here.
def index(request):
    return render(request,"index.html")
    # return HttpResponse("Hello Django App")

def contact(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        email = request.POST.get('email')
        contact = request.POST.get('contact')
        desc = request.POST.get('desc')
        contact = Contact(name=name, email=email, contact=contact, desc=desc, date=datetime.today())
        contact.save()
        messages.success(request, 'Your Form has been Submitted')
    return render(request,"contact.html")
```

Defined below are the settings.py which govern the overall performance of the software and where all Django installed and wanted apps are listed for the software to properly function.

```python
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'ardyy-btlw)qfsy*f!p_3!6v3nv-2b(fta##_l3yqhza@6!pj5'

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

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'users.apps.UsersConfig',
    'home.apps.HomeConfig',
    'crispy_forms',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
```

Models.py is a crucial import concept of Django framework that defines the SQL database used in this web application

```python
from django.db import models

# Create your models here.
class Contact(models.Model):
    name = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    contact = models.CharField(max_length=10)
    desc = models.TextField()
    date = models.DateField()

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

API Integration Testing:

When integrating the the system we start by install prerequisite needs or requirements as shown in the visual studio terminal

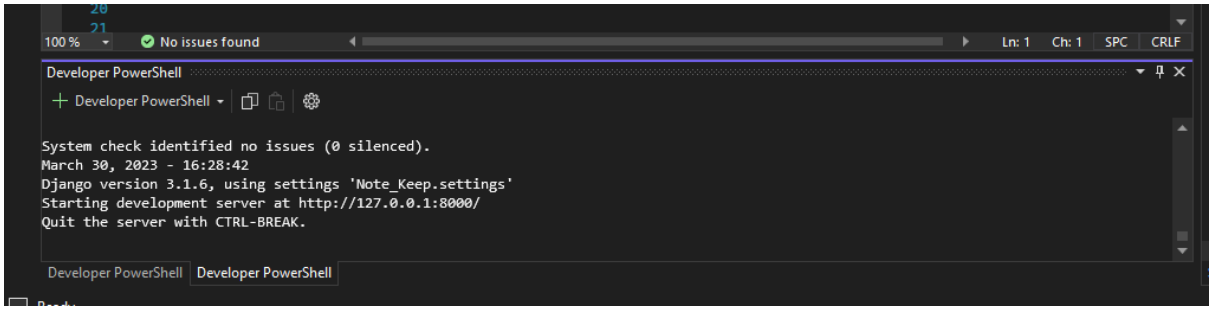
```
ango==3.1.6->-r requirements.txt (line 3)) (0.3.1)
Requirement already satisfied: six>=1.5 in c:\users\naruto lee\appdata\local\programs\python\python311\lib\site-packages (from python-da
teutil==2.8.1->-r requirements.txt (line 9)) (1.16.0)

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: C:\Users\NARUTO LEE\AppData\Local\Programs\Python\Python311\python.exe -m pip install --upgrade pip
PS C:\Users\NARUTO LEE\Desktop\Student-Online-Notes-app-django-main>
```

After successful installation we then run our migrations, if we successfully migrations to propagate the changes made to models we can move run our server

```
[notice] To update, run: C:\Users\NARUTO LEE\AppData\Local\Programs\Python\Python311\python.exe -m pip install --upgrade pip
PS C:\Users\NARUTO LEE\Desktop\Student-Online-Notes-app-django-main> py -m manage migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.
PS C:\Users\NARUTO LEE\Desktop\Student-Online-Notes-app-django-main>
```

Running the server successfully means successfully integration testing.



Installation requirements

- Download and Install Microsoft Visual Studio community from visualstudio.microsoft.com
- Download and Install Python 3.11.2 from python.org
- Open Microsoft Visual Studio and open the folder Student online notes
- Right click on the folder in visual studio and open in terminal
- Now start by creating a virtual environment and activate
  e.g., py -m venv env or virtualenv env (creation of virtualenvironment)
  env\Scripts\activate.bat(activation of virtual environment)
- Install requirements.txt
  e.g., py -m pip install -r rqeuirements.txt
  or pip install -r requirements.txt
- Create super user(optional)
  e.g., py -m manage createsuperuser
- Run migration
  e.g., py -m manage migrate
- Then lastly, we runserver
  e.g., py -m manage runserver

For further details contact the developer…