**MZUMBE UNIVERSITY**



**FACULTY OF SCIENCE AND TECHNOLOGY (FST)**

**DEPARTMENT OF COMPUTING SCIENCE STUDIES (CSS)**

**INSTRUCTOR NAME**: DR. CHURI

**PROGRAMME NAME:** BSC. ITS

**COURSE NAME:** ADVANCED DATABASE MANAGEMENT SYSTEM

**COURSE CODE:** CSS 321

**ACADEMIC YEAR:** 2023/2024

**SUBMISSION DATE:** 30/04/2024

**TASK:** GROUP ASSIGNMENT

|  |  |  |
| --- | --- | --- |
| **S/N** | **NAMES** | **REGISTRATION NUMBER** |
| 1 | PASCHAL P. MATIKU | 14322075/T.21 |
| 2 | DAVINA MATHEW | 14322054/T.21 |
| 3 | ANTHONY ANSELIM MALYA | 14322073/T.21 |
| 4 | KARIM NOMANO | 14322032/T.21 |
| 5 | ESTER MBONEA MNDAMBI | 14322003/T.21 |

**GROUP PARTICIPANTS**

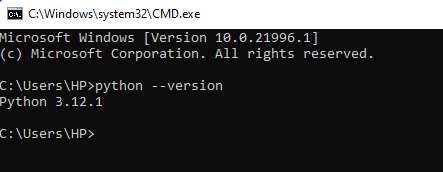
**1.THE OVERVIEW:**

The Django application is a web framework written in Python, designed to simplify the process of building web applications. Its primary purpose is to enable developers to create robust, scalable, and maintainable web applications quickly and efficiently. Django follows the "don't repeat yourself" (DRY) principle, emphasizing code reusability and reducing redundancy. It provides a clean and pragmatic design, comprehensive documentation, and a thriving community, making it an excellent choice for developing a wide range of web applications, from simple blogs to complex enterprise-level systems. With built-in features like an object-relational mapper (ORM), a powerful template engine, and a secure authentication system, Django accelerates the development process and empowers developers to focus on building unique and innovative web experiences.

**2.NSTALLATION:**

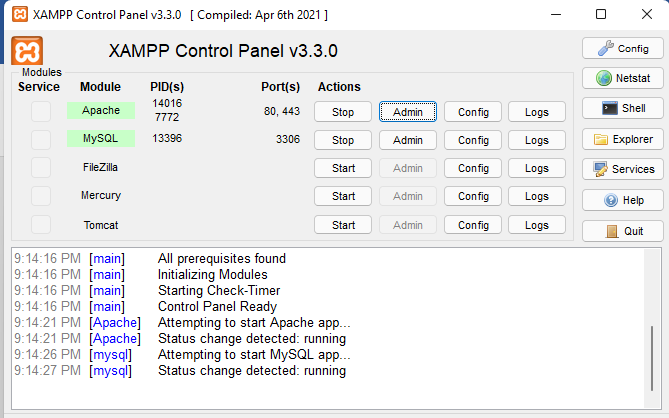
**1.Installing Python**

To set up Python for our project, we first need to ensure that Python is installed on our system. We downloaded the latest version of Python from the official Python website and installed in our device. During the installation process, we'll have the option to add Python to our system's PATH environment variable, which allows us to run Python commands from any directory in the command line. Once Python is installed, we verified the installation by opening a command prompt and running the command “python –version”, which should display the installed Python version. With Python successfully installed, we're ready to proceed with setting up our Django project and developing our web application.



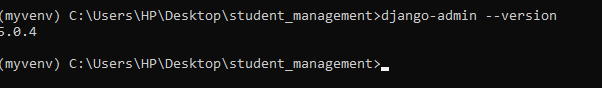
**2.Installing Xampp**

To install XAMPP for our project, we'll first need to download the XAMPP installer from the official website. Once downloaded, we run the installer and follow the on-screen instructions to install XAMPP on our system. During the installation process, we'll be prompted to choose which components to install; ensure that Apache, MySQL, and PHP are selected, as they are required for running Django projects. After completing the installation, we launch the XAMPP Control Panel and start the Apache and MySQL services. This will start the Apache web server and MySQL database server, allowing us to host our Django project locally. Finally, we navigate to the htdocs directory within the XAMPP installation directory and create a new folder for our Django project. We can then copy our Django project files into this folder and access our project via the localhost URL in our web browser. XAMPP provides a convenient way to set up a local development environment for Django projects, allowing us to develop and test our applications without needing to deploy them to a remote server.



**3.Install Django**

To install Django, we first needed to ensure that we had Python installed to our computer. Then, we used “pip”, the Python package manager, to install Django within a virtual environment. We used to open our command line interface and execute the command pip install Django. This downloaded and installed the latest version of Django and its dependencies. Once the installation completed, we verified it by running “django-admin –version” in the command line, which displayed the installed Django version. With Django installed, we were ready to start creating your web application.





**Creation of a Django Project**

We create a Django project named "student", then navigated to the directory where we wanted to create the project using the command line interface. Then, we executed the command “django-admin startproject student”. This command created a new directory named "student" containing the necessary files and directories for a Django project. Inside the "student" directory, we found a manage.py file, which was used to interact with your Django project, as well as a subdirectory named "student", containing settings.py, urls.py, and other configuration files. This newly created directory structure formed the foundation of our Django project named "student", ready for further development and customization.

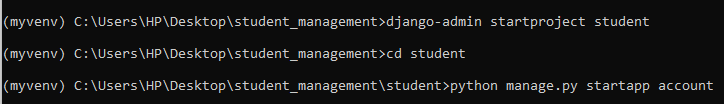


**Set Up Database:**

To set up the database for our Django project, we needed to configure the database settings in the project's settings.py file. Django supports various database engines such as SQLite, MySQL, PostgreSQL, and Oracle. We specified the database engine as MYSQL, database name as Student Management System, Once the database settings were configured, Django automatically handled creating and managing database tables based on our models. This ensures that your Django project interacts seamlessly with the specified database, allowing you to store and retrieve data efficiently

**Creation Django App**

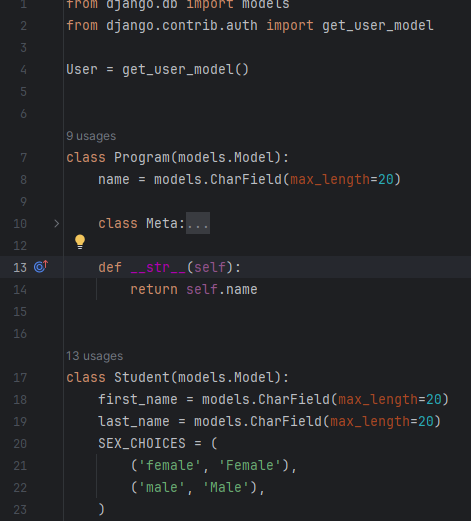
To create a Django app within our project, we use the command python manage.py startapp account. This command generates a directory with the “account” app name, containing the necessary files and directories for the app's structure. Inside the app directory, we'll find files such as models.py for defining data models, views.py for handling requests and generating responses, urls.py for routing URLs to views, and templates directory for HTML templates. Additionally, there may be other files like admin.py for registering models with the Django admin interface and apps.py for configuring the app. Once the app is created, we can integrate it into our project by adding its name to the INSTALLED\_APPS list in the project's settings.py file. This allows Django to recognize and utilize the functionality provided by our app within the project.



**3.IMPLEMENTATION:**

In our Django project, each component plays a crucial role in handling different aspects of the application. Models, views, URLs, and templates work together to create a functional web application.

**Models:** In our Django project, models define the structure of the data and how it is stored in the database. We define models as Python classes in the models.py file of our app. Each model class represents a database table, with attributes mapping to table columns. We use Django's ORM to interact with the database, allowing us to perform database operations using Python code without writing SQL queries directly.

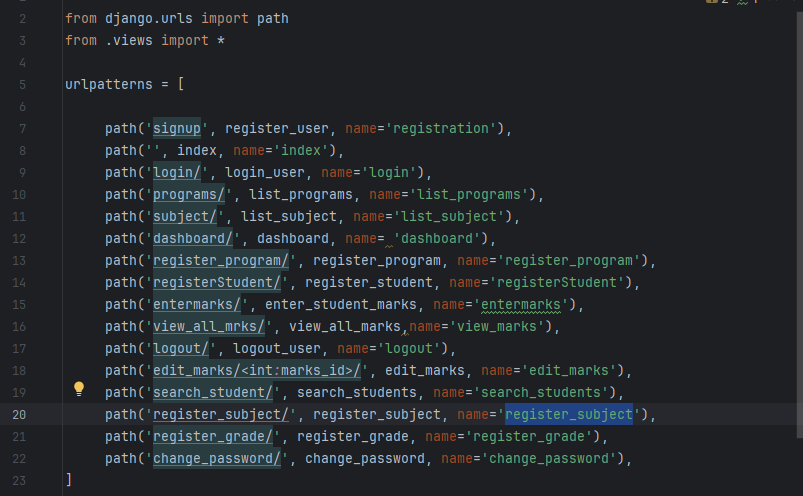






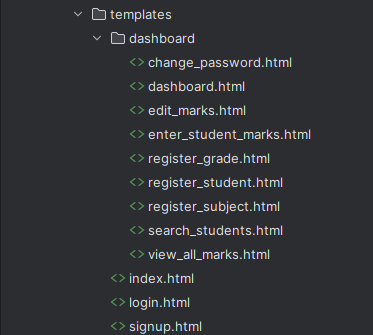
**Views:** Views are Python functions or classes that handle incoming requests and return appropriate responses. We define views in the views.py file of our app. Views interacts with models to retrieve or modify data as needed and then render templates to generate HTML responses. We use Django's built-in view decorators or class-based views to define view logic, such as handling form submissions, displaying data, or processing user input.

**URLs**: URLs in our Django project map incoming requests to corresponding views. We define URL patterns in the urls.py file of our app or project. Each URL pattern consists of a URL pattern string and a reference to the corresponding view function or class. We use regular expressions or path converters to define dynamic URL patterns that capture URL parameters and pass them to view functions as arguments.



**Templates:** Templates are HTML files containing static content and dynamic placeholders for data. We define templates in the templates directory of our app. Templates use Django's template language to insert dynamic data retrieved from views or models. We can use template inheritance and include tags to reuse common template components across multiple pages and maintain a consistent layout and design throughout our application.

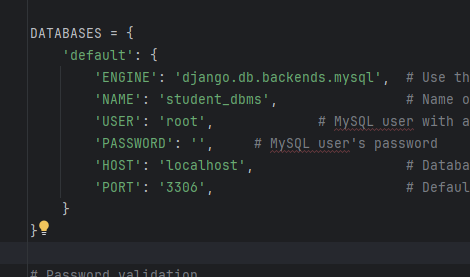
By understanding how each component of our Django application works together, we can effectively develop and maintain a functional and user-friendly web application.



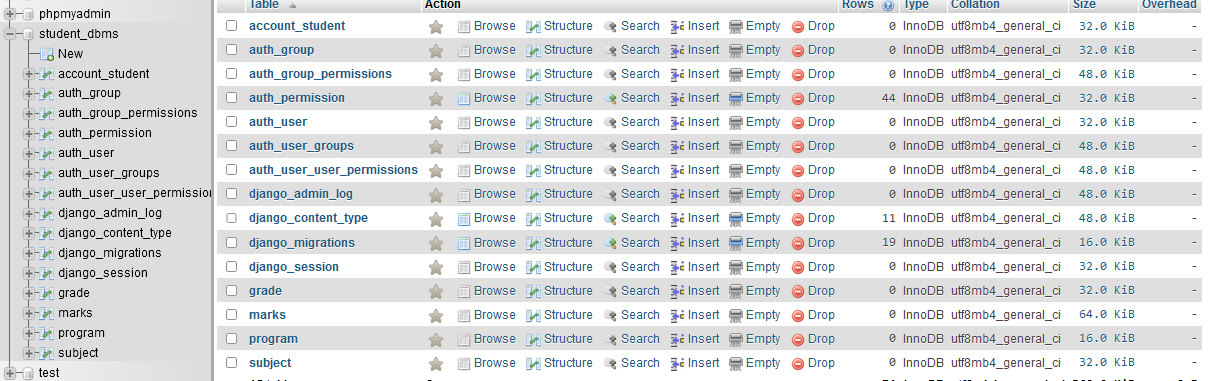
**4.CONFIGURATION INSTRUCTION**

Configuration instructions refer to specific steps or settings that need to be applied to set up, adjust, or customize a system or application according to specific requirements

For configuring the database connection and deploying the application in our "student" project, we first need to update the database settings in the settings.py file of our Django project. We navigate to the project's settings.py file and locate the DATABASES dictionary. Within this dictionary, we specify the database engine as MYSQL, database name as, user, password, host, and port for our MySQL database. We ensure that the database credentials match those of our local MySQL server provided by XAMPP. Once the database settings are configured, we save the changes to the settings.py file. Next, we deploy our Django application by starting the Apache and MySQL services in the XAMPP Control Panel. This action initiates the Apache web server and MySQL database server, allowing us to host our Django project locally. Finally, we navigate to the htdocs directory within the XAMPP installation directory and create a new folder for our Django project. We copy our Django project files into this folder and access our project via the localhost URL in our web browser. With these configuration instructions, we ensure that our Django project is connected to the MySQL database and deployed on our local server for development and testing.

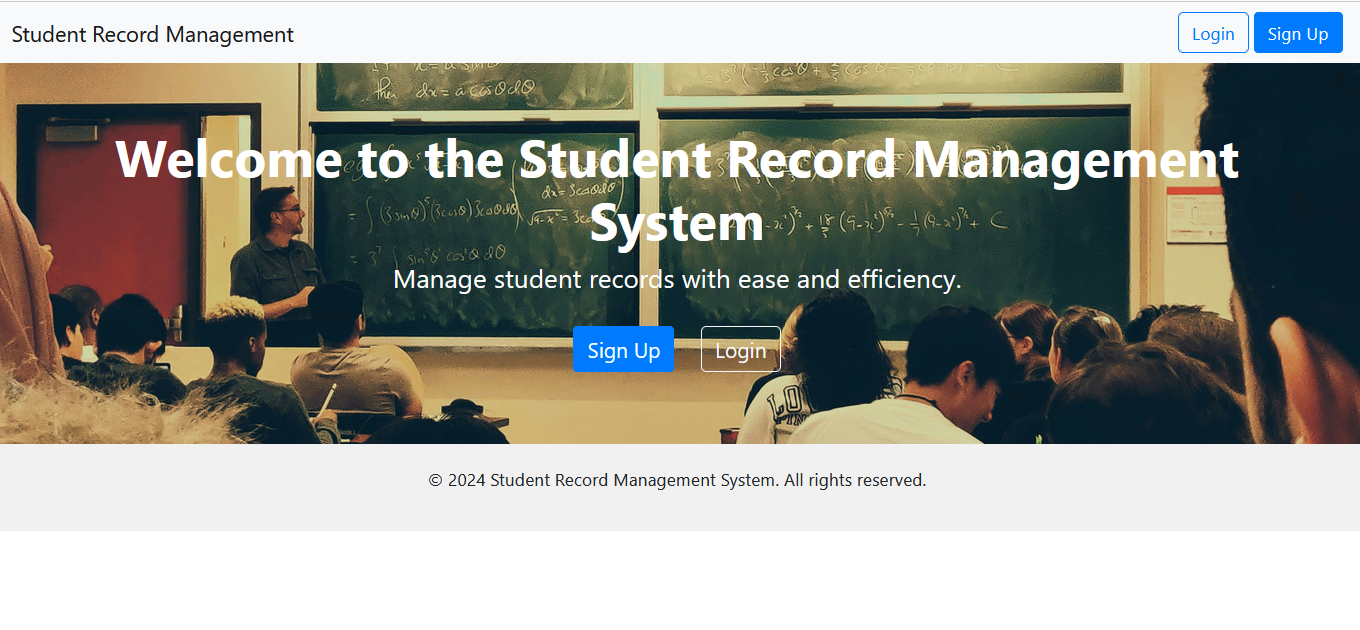


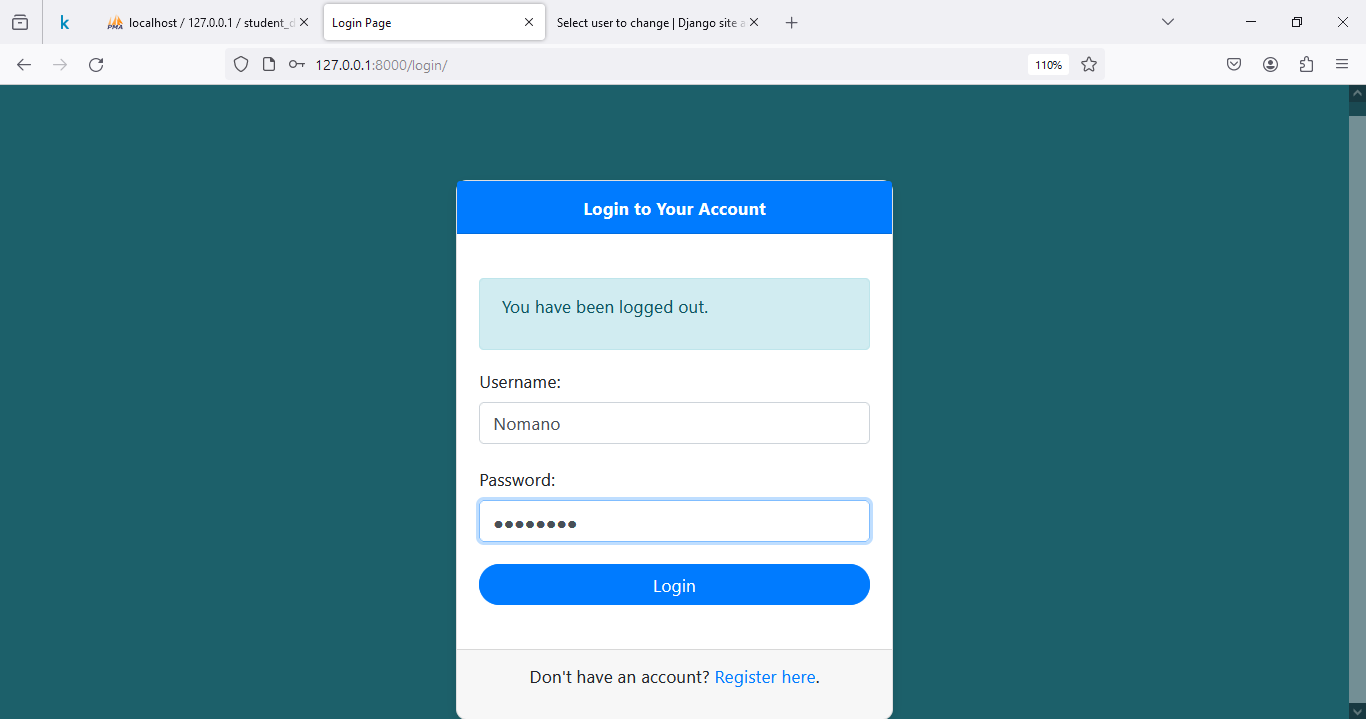
By following configuration instructions above, users can customize and optimize the functionality and behavior of the system to align with their unique use cases or preferences.

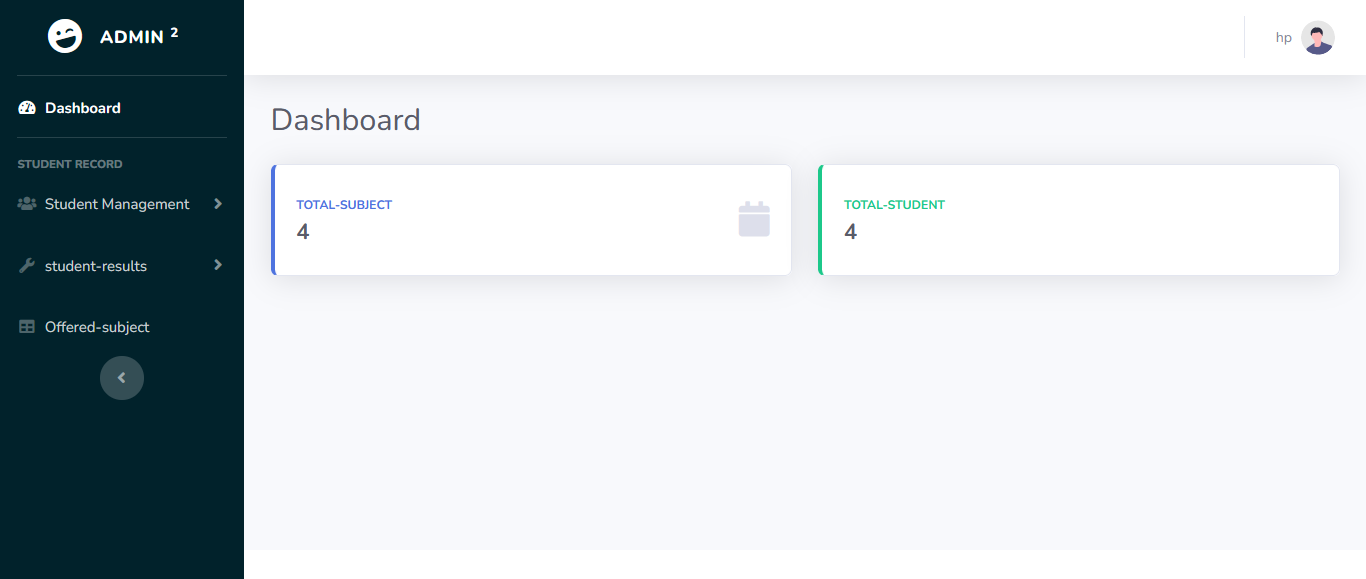


**5.TESTING**

On testing the functionality of our system, we login to navigate to the dashboard and starting processing the system functionalities that involves recording, searching and updating the students records.







To test the functionality of our project, particularly for tasks searching for students

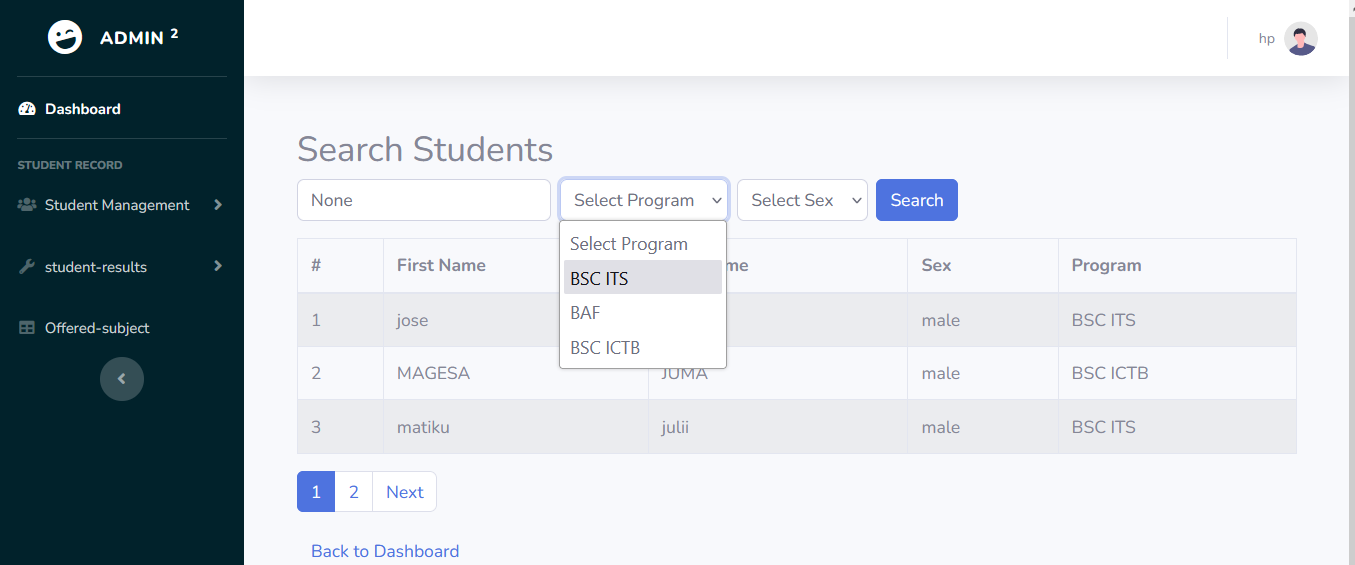
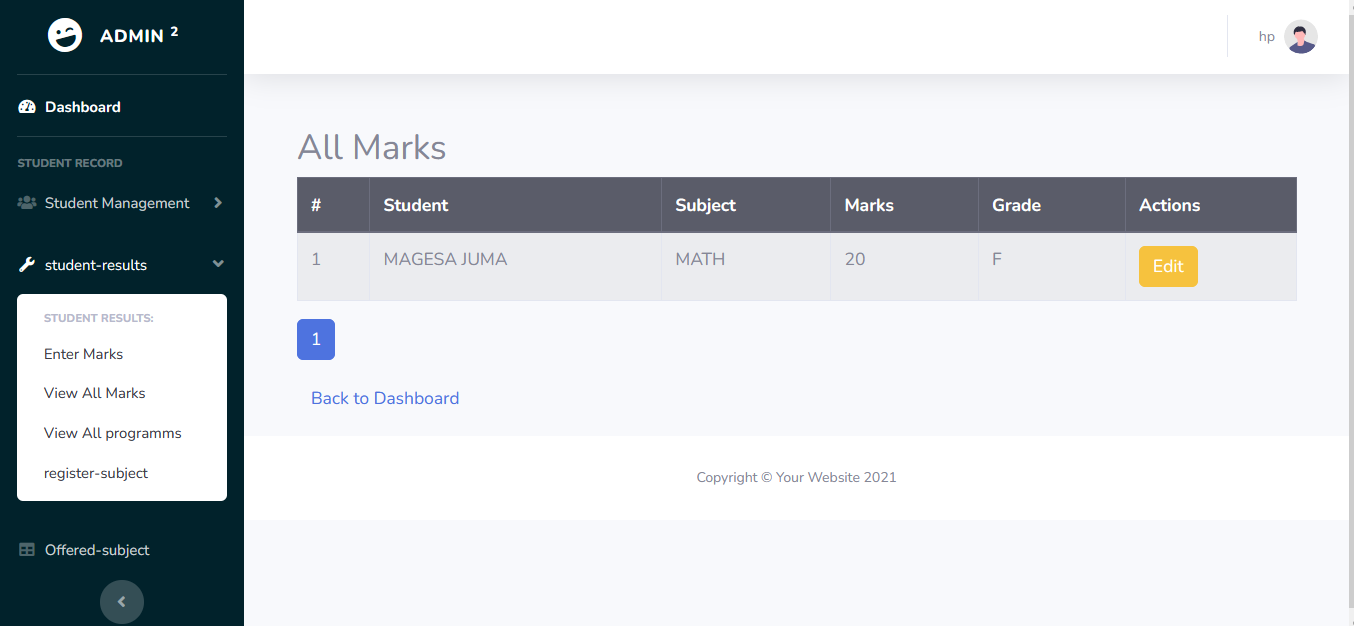
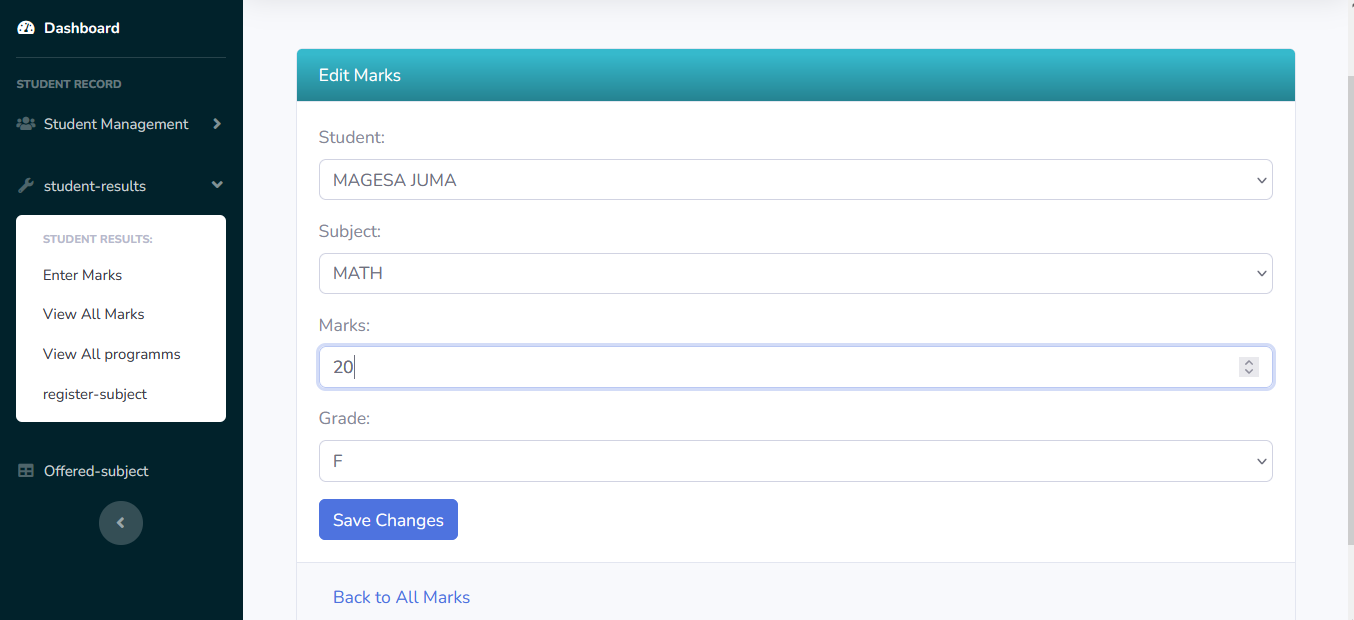


Fig. searching student by using its program, sex or name

To updating their marks, we employ a systematic approach.





Firstly, we ensure that the views responsible for handling these functionalities are correctly implemented, with appropriate logic to process incoming requests and generate the expected responses. These test cases simulate various scenarios, including searching for students with different criteria and updating their marks with valid and invalid inputs.

We validate the expected outcomes against the actual results returned by the views, ensuring that the application behaves as intended under different conditions. Additionally, we perform integration tests to ensure that the views, models, and templates work seamlessly together.

By rigorously testing the application's functionalities, we can identify and address any bugs or issues early in the development process, ensuring the reliability and correctness of our Django project.

**6.ADDITIONAL FUNCTIONALITY**

There are several additional features and enhancements that could be added to the project in the future to extend its functionality and improve the user experience. Some potential ideas include register students, subjects, grade and program.

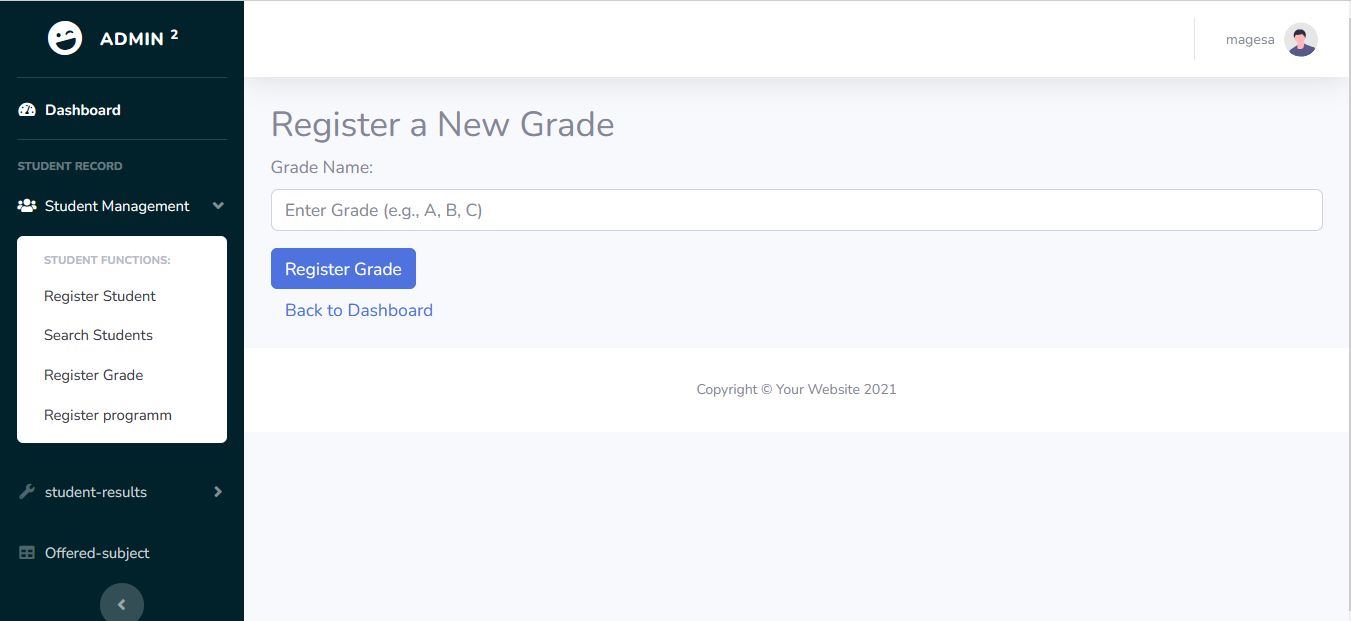


Fig. registration of new grade

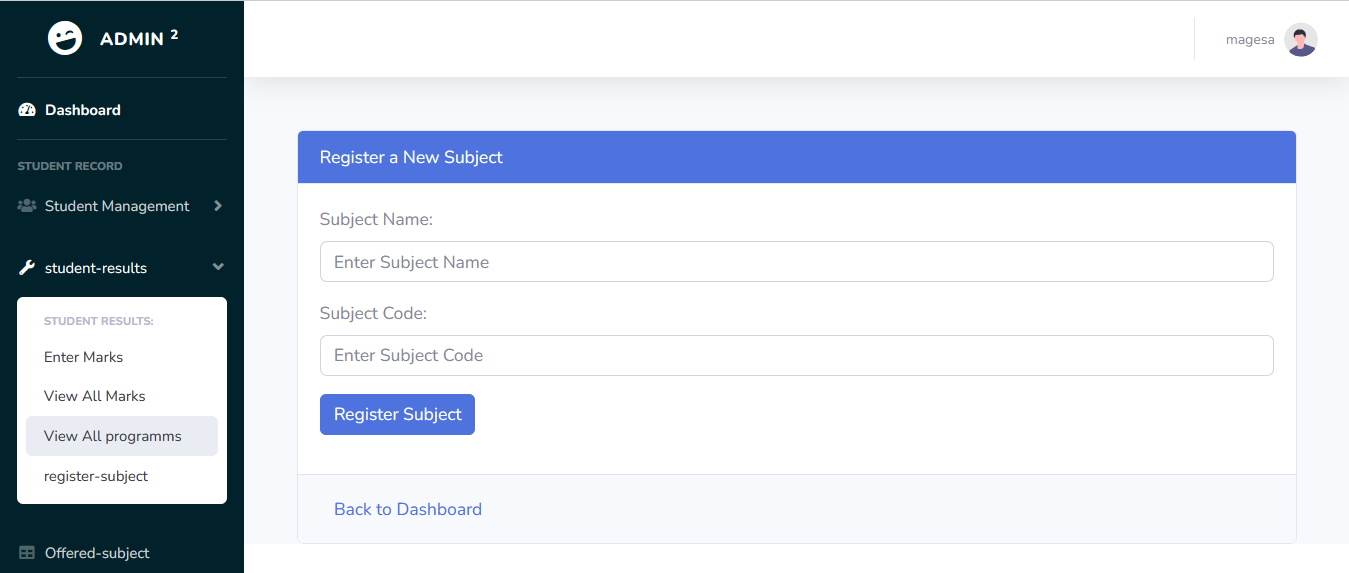


Fig. registration of new subject

**7.CONCLUTION**

Django offers a robust framework for web development, embodying the DRY principle and fostering a supportive community. By ensuring seamless setup with Python and XAMPP, our project operates smoothly, with configuration instructions facilitating database connectivity. Rigorous testing ensures functionality, while future enhancements promise expanded capabilities, underscoring Django's potential for impactful web experiences.