

Міністерство освіти і науки України  
Національний технічний університет  
«Дніпровська політехніка»

Інститут електроенергетики  
(інститут)

Факультет інформаційних технологій  
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем  
(повна назва)

**ПОЯСНЮВАЛЬНА ЗАПИСКА**  
**кваліфікаційної роботи ступеня**  
**бакалавра**

(назва освітньо-кваліфікаційного рівня)

студента *Удоха Сундая Тома Річарда*  
(ПІБ)

академічної групи *122-17-3*  
(шифр)

спеціальності *122 Комп'ютерні науки*  
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*  
(назва освітньої програми)

на тему: *Розробка веб-додатку для обліку студентів на базі  
фреймворку Django*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	<i>Доц. Сироткіна О.І.</i>			
<b>розділів:</b>				
спеціальний	<i>Доц. Сироткіна О.І.</i>			
економічний	<i>Доц. Касьяненко Л.В.</i>			
<b>Рецензент</b>				
<b>Нормоконтролер</b>	<i>доц. Гуліна І.Г.</i>			

Дніпро  
2021

Міністерство освіти і науки України  
НТУ «Дніпровська політехніка»

**ЗАТВЕРДЖЕНО:**

завідувач кафедри  
програмного забезпечення комп'ютерних систем

(повна назва)

І.М. Удовик

(підпис)

(прізвище, ініціали)

«        »        2021 року

**ЗАВДАННЯ**

**на кваліфікаційну роботу**  
**бакалавра**

(назва освітньо-кваліфікаційного рівня)

студента 122-17-3 Удоха Сундая Тома Річарда  
(група) (прізвище та ініціали)

тема кваліфікаційної роботи Розробка веб-додатку для обліку студентів  
на базі фреймворку Django

затверджена наказом ректора НТУ «ДП» від 07.06.2021 № 317-с

Розділ	Зміст виконання	Термін виконання
<i>Спеціальний</i>	<i>На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми</i>	<i>13.05.2021 р.</i>
<i>Економічний</i>	<i>Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки</i>	<i>27.05.2021 р.</i>

Завдання видав \_\_\_\_\_ Доц. Сироткіна О.І.  
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання \_\_\_\_\_ Удох Сундай Том Річард  
(підпис) (прізвище, ініціали)

Дата видачі завдання: 14.01.2021 р.

Термін подання кваліфікаційної роботи до ЕК: 11.06.2021 р.

## РЕФЕРАТ

Пояснювальна записка: 94с., 53 рис., 1 табл., 3 дод., 22 джерела.

Об'єкт розробки: розробка веб-додатку для системи обліку студентів на основі Django Framework.

Мета кваліфікаційної роботи: розробка веб-додатку, що надає можливість працювати з інформацією про студентів онлайн замість фізичного ведення даних. Веб-ресурс надає зручну форму для створення нових клієнтів.

У вступі аналізується поточний стан питання, з'ясовується проблема, мета кваліфікаційної роботи та сфера її застосування, обґрунтовується актуальність теми.

У першому розділі проводиться дослідження предметної галузі та існуючих рішень, визначається актуальність завдання та мета розробки, розробляється постановка завдання.

У другому розділі вибирається платформа для розробки, здійснюється розробка програми та її розробка, дається опис алгоритму та структури функціонування системи, визначаються вхідні та вихідні дані, характеристики структури параметрів наводяться технічні засоби, описується робота програми.

В економічному розділі визначається складність розробленого програмного продукту, підраховується вартість робіт зі створення додатку та обчислюється час на його створення.

Практичне значення роботи полягає у створенні веб-сервісу, який забезпечує можливість обліку даних студентів та можливість їх графічного відображення на деяких діаграмах.

Актуальність інформаційної системи визначається високим попитом на такі системи через перехід на онлайн-курс та відсутність вільного та зручного рішення, яке може поєднати в собі багато функцій корисних для установ, для яких необхідно легко та зручно вести облік даних студентів.

Список ключових слів: СХЕМА, КЛІЄНТ, СЕРВЕР, ІНФОРМАЦІЙНА СИСТЕМА, КОМП'ЮТЕР, ВЕБ-ДОДАТОК, ФОРМИ, БАЗА ДАНИХ, ШАБЛони, ПЕРЕГЛЯДИ, DJANGO.

## **ABSTRACT**

Explanatory note: 94pp., 53 figs., 1 tab., 3 apps, 22 sources.

Object of development: development of a web application for students' management system based on Django Framework.

Purpose of the qualification thesis: development of a web app favouring an easier method for dealing with students' information than physical kept records. Developers will provide a convenient form for creating their clients.

The introduction analyses the current state of the problem, clarifies the problem, the purpose of the qualification work and the scope of its application, and substantiates the relevance of the topic.

In the first section the research of the subject area and existing decisions is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed.

In the second section the platform for development is chosen, the program design and its development are carried out, the description of algorithm and structure of functioning of system is given, input and output data are defined, characteristics of structure of parameters of technical means are given, work of the program is described.

The economic section determines the complexity of the developed software product, calculates the cost of work to create an application and calculates the time to create it.

The practical significance lies in the creation of a web service that provides the ability to manage student data, and the ability to graphically display the data in some charts.

The relevance of the information system is determined by the high demand for such systems due to the switch to the online course and the lack of a free and convenient solution, which we have combined many features useful for institutions who want to manage with ease students' data.

Keywords: CHART, CLIENT, SERVER, INFORMATION SYSTEM, COMPUTER, WEB APP, FORMS, DATABASE, TEMPLATES, VIEWS, DJANGO.

## CONTENTS

PEΦEPAT .....	3
ABSTRACT .....	4
LIST OF ACRONYMS.....	7
INTRODUCTION.....	8
SECTION 1 .....	9
ANALYSIS OF THE SUBJECT AREA AND PROBLEM STAMENT.....	9
1.1 General Information about the Subject Area .....	9
1.2 Development Objectives and Area of Use .....	12
1.3 Grounds for the Project Development.....	13
1.4 Problem Statement .....	14
1.5 Software Requirements .....	15
1.5.1. Functional Requirements.....	15
1.5.2. Information Security Requirements .....	16
1.5.3. Hardware Environment Requirements.....	16
1.5.4. Compatibility Requirements .....	16
SECTION 2 .....	18
DESIGN AND DEVELOPMENT OF THE STUDENT MANAGEMENT.....	18
2.1 Functional purpose of the system.....	18
2.2 Description of applied mathematical methods.....	19
2.3 Description of used technologies and programming languages .....	19
2.4 Description of the system structure and algorithms of its functioning .....	26
2.4.1 Relationship between different parts of the system structure. ....	39
2.4.2 Database of the Student Management System.....	40

2.4.3 Django architecture function view .....	41
2.5 Justification and organization of input and output data of the program .....	42
2.6 Description of the developed system .....	43
2.6.1 Description of the developed system .....	43
2.6.2 The software used.....	43
2.6.3 Calling and downloading the program.....	43
2.6.4. Description of the user interface. ....	43
РОЗДІЛ 3 .....	59
ЕКОНОМІЧНИЙ РОЗДІЛ .....	59
3.1. Розрахунок трудомісткості та вартості розробки програмного продукту	59
3.2. Розрахунок витрат на створення програми .....	62
CONCLUSIONS .....	65
REFERENCES .....	67
Appendix A. Source code.....	69
Додаток Б. Відгук керівника економічного розділу .....	93
Appendix C. List of files on the disc .....	94

## **LIST OF ACRONYMS**

API – Application Programming;

OS – Operating System;

UI – User Interface;

HTTP – Hypertext Transfer Protocol;

ERP – Enterprise Resource Planning;

SMS – Student Management System;

SIS – Student Information System;

HOD – Head of Department;

ASGI – Asynchronous Server Gateway interface;

VS Code – Visual Studio Code;

## INTRODUCTION

The objectives of this qualification thesis and the object of its activities are directly related to the direction of training and corresponds to the generalized topics of qualification work and a list of these production functions, typical tasks, skills and competencies that must have bachelors in 122 "Computer Science".

The theme of the qualification thesis is the development of a web application for students' management system based on Django Framework.

The current record management system in some university is already defined as an older generation management system and it already cannot satisfy the wanted user.

As a result, a system called Student Management System is being developed as an upgrade version of the old system in order to replace the manual system to solve the problem it faced while using the old system or the manual system. The Student Management System (SMS) is a software application for education that uses to manage student information and data. It is also known as, Student Information System (SIS), Student Information Management System (SIMS) or School Management System (SMS).

The qualification thesis involves the development of database Management with the use of Python Django, HTML, CSS, JavaScript and Bootstrap.

This design of this system is web type, so that user can also directly operate the system by connecting to the internet. The users of this system are divided into three groups, which are administrators (admin), teachers (staff) and students.

This software is relevant because it fit the time in which we currently are with the sudden adoption of the online course due to current pandemic.



# SECTION 1

## ANALYSIS OF THE SUBJECT AREA AND PROBLEM STATEMENT

### 1.1 General Information about the Subject Area

Student management system is an environment where all the method of the scholar within the institution is managed. It is done through the automated computerized method. Conventionally this technique is completed using papers, file and binders.

This system saves the time of the scholar and of the administrator. It includes process like registration of the student's details, assigning the department supported their course and maintenance of the record.

This system reduces the value and workforce required for this job. because the system is online the knowledge is globally present to everyone. This makes the system easy to handle and feasible for locating the omission with updating at an equivalent time.

As for the prevailing system, they use to take care of their record manually which makes it susceptible to security. If filed a question to look or update during a manual system, it'll take tons of your time to process the query and make a report which may be a tedious job.

As the system utilized in the institute is outdated because it requires paper, files and therefore the binders, which can require the human workforce to take care of them. to urge registered within the institute, a student during this system one should come to the university.

Get the forms from the counter while standing within the queue which consumes tons of the student's time also as of management team. because the number of the scholar increases within the institute manually managing the strength becomes a busy job for the administrator.

This computerized system stores all the info within the database, which makes it easy to fetch and update whenever needed. This technology offers us advantages and disadvantages.

## **Advantages and Disadvantages of SMS.**

Advantages for schools:

– At the level of the school structures:

Setting up such an application makes it easier to manage student files, and it facilitates a better communication between the various departments of the university. There is less documentation, which avoids errors and waste of paper. There is also a better communication between the administration and the teaching body and a follow-up of the seriousness of the teachers.

– At the level of teachers:

It makes the job easier because they have the flexibility to work wherever they want. It is easier to follow the progress of the graduates through their grades, to transmit the lessons to the graduates in case of absence and to inform them. It is an excellent means of following the seriousness and implication of the student in their work according to the subjects.

– At the student level;

It provides real-time access to information about university and assignments or courses disclosed by professors. It allows you to follow their progress in the various subjects and to have information on their grades, absence and sanction taken against them. There is the possibility to communicate directly and have a follow-up with the Head of Department.

Disadvantages for school:

– At the level of the school structures:

A follow-up of the professors and their work which can be frowned upon. The web application needs a maintenance agency to monitor the site in the event of a problem. It can happen that they would be too much message from students and teachers which could end up being difficult to manage.

– At the Level of Teacher:

There is going to be a double work because of the obligation to postpone the notes after correction on paper. The Permanent follow-up from the university management can also not be accepted by the teacher.

– At the Student level:

There is too much information at the same time, lack of rest on the part of teachers who are often too busy and the traceability of its school data.

Presently at the market, we have a lot of commercialized student managing system, which are efficient and have been successful in simplifying the works of institutions when it comes to student management. The list is very long, however an analysis will be made on four of the most influential system.

### **Comparisons of top Student Management System**

MasterSoft - MasterSoft SIS works as a centralized database whereby all the information related to students is stored. It helps the institutions to manage every single detail about students in an organized and cost-effective way. Besides, multiple users can access at the same time. The undoubted advantage of the system is that it has a best work environment and Seniors are supportive. The software is an online website, which is different from others that are desktop. One of the inconvenient is that the system lack of technology in SQL not using tools. It is not free.

Fakera - Fakera acts as a student ERP system, which provides a complete 360-degree solution to student management. From taking care of student's admission process to settling of fees online and maintaining attendance record, it does it all. It is a very user friendly and free for universities. Fekara's pro version is totally free for non-profits; the basic version is free for all educational institutes.

PowerSchool - PowerSchool is changing the way a school runs. With the help of technology, this school management system attempts to unify back office with the classroom to put everyone on the same team. It is a great system that helps realize students' full potential. The advantage is that we can connect to an API. The only

problem is that it is a Desktop application; in a case of less memory storage, you cannot install the application to use it.

Edu-Orbit - Edu-Orbit is an online student management system that follows a modular approach. For starters, you can choose as many numbers of modules you want. Besides, it allows teachers to create their teaching and evaluation plan and generate a performance report on the same. The advantage is that the application is both an online software and a Desktop app for the Windows system. Probably they are developing an IOS Desktop app. You can use API, there is an option for customization, and they have a mobile support. The inconvenience with Edu-Orbit could be also be called non-intuitive interface and no display of graphics.

Table 1.1

**Advantages and disadvantages of programs**

Opportunities	MasterSoft	Fakera	PowerSchool	Edu-Orbit
Online website	YES	YES	NO	YES
Desktop Platforms	NO	NO	YES	YES
API	NO	NO	YES	YES
GRAPHICS	NO	YES	NO	NO
CUSTOMIZATION	YES	YES	YES	YES
MOBILE SUPPORT	YES	YES	YES	YES
FREE	NO	YES	NO	NO

From the table of capabilities of each of the programs (Table 1.1), you can do conclusions about what opportunities the development system should provide, and which ones shortcomings it must get rid of.

**1.2 Development Objectives and Area of Use**

The full name of the developed system for qualification work:

“Development of a web application for Student Management System based on Django Framework “.

### Basic Terminology and Keywords:

A web application (web app) is an application program that is stored on a remote server and delivered over the internet through a browser interface. Web services are web apps by definition and many, although not all, websites contain Web apps. According to Web.AppStorm editor Jarel Remick, any website component that performs some function for the user qualifies as a Web app.

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it 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's free and open source.

A Framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software.

The developed web app should be used in the field of managing student records by universities, institutions and any group interested of managing records such as attendees, marks and so on and so forth in other to have real data saved in the system.

The aim of the project is to develop an easier method for dealing with students' information than physically kept records at a free course and with better graphic design.

### **1.3 Grounds for the Project Development**

According to the educational program, according to the curriculum and schedules of the educational process, at the end of the study the student performs qualification work.

The topic of work is agreed with the project manager, the graduating department, and approved by order of the rector.

Thus, the grounds for development (performance of qualifying work) are:

- Educational program of specialty 122 "Computer Science"
- Curriculum and schedule of the educational process;

- Order of the Rector of Dnipro University of Technology № 317-c from 07.06.2021;
- Tasks for qualification thesis on “Development of a web application for Students’ Management System based on Django Framework.”

## **1.4 Problem Statement**

The aim of the task is to develop a web application for managing student records. Since in this category we are dealing with 3 User Interface, we are to notate that every user has different attribute likewise they have different Dashboard.

### **Admin:**

- See Overall Summary Charts of Students Performance, Staffs Performances, Courses, Subjects, Leave, etc.
- Manage Staffs (Add, Update and Delete).
- Manage Students (Add, Update and Delete).
- Manage Course (Add, Update and Delete).
- Manage Subjects (Add, Update and Delete).
- Manage Sessions (Add, Update and Delete).
- View Student Attendance.
- Review and Reply Student/Staff Feedback.
- Review (Approve/Reject) Student/Staff Leave.

### **Staff/Teachers:**

- See the Overall Summary Charts related to their students, their subjects, leave status, etc.
- Take/Update Students Attendance.
- Add/Update Result.
- Apply for Leave.
- Send Feedback to HOD.

### **Students:**

- See the Overall Summary Charts related to their attendance, their subjects, leave status, etc.
- View Attendance.
- View Result.
- Apply for Leave.
- Send Feedback to HOD.

## **1.5 Software Requirements**

### **1.5.1. Functional Requirements**

The developed application must meet the following functional characteristics:

- Every user have their own dashboard accordingly;
- The server must process the information and give a “response” to each HTTP request;
- In case of errors the web service must process errors and display the correct error messages;
- It should be possible to add information about the Staff/Teacher or Student details such as name, surname, sessions, course etc.;
- The opportunity to create a relationship between the admin and the teacher and student;
- The ability to create different user accounts must be implemented and authorization and authentication must be implemented;
- The possibility to see the data in a convenient graphic form should be realized
- The Creation of the new record for the new student;
- Deletion of the record which already exist in the system based on the requirement of the institute;
- Update in the record which is present in the system as per the need;
- Generate the report on the attendance of the student as per his/her record;
- Admin’s handle of the department, this function eases the process of management;

### **1.5.2. Information Security Requirements**

To ensure a reliable operation of the system, it is important to implement:

- Protection against unauthorized access to the web service.
- Control of source information, exclusion of cases of sending important/secret information from the student or teacher.
- Handling of exceptional situations and output of messages in case of errors.

### **1.5.3. Hardware Environment Requirements**

For the normal operation of the web app, it is necessary that the computer, which the web-oriented subsystem will operate, meet the following requirements:

- I3 core processor with a clock speed of at least 2.4GHz;
- RAM of at least 8GB; -120 GB SSD;

The above technical characteristics are recommended, i.e. in the presence of technical means not lower than those specified, the developed software product will function in accordance with the requirements for reliability, data processing speed and security, set by the customer.

### **1.5.4. Compatibility Requirements**

For the normal operation of the program, it is necessary that the software of the computer on which the web-oriented system will operate, meet the following requirements:

- OS: Windows (7, 8, 10), Mac OS, Linux;
- Browsers: Chrome, Safari, Mozilla Firefox and others;
- Framework for automation of project collection: Django Framework
- MYSQL database (Or Django database SQL)
- Recommended internet connection speed: from 3 Mbps.



The main programming languages that were used: Python, JavaScript, Bootstrap, HTML and SQLite query language.

The entire application will be developed in Visual Studio Code.

## **SECTION 2**

### **DESIGN AND DEVELOPMENT OF THE STUDENT MANAGEMENT**

#### **2.1 Functional purpose of the system**

The Student Management System, serve to provide a 'simple to use' User Interface. The aim of this project is to develop an easier method for dealing with students' information than physically kept records. I believe this project will minimize the amount of effort and paper work and thereby save time.

In application, this innovation will facilitate the work of data collection and records such as personal details, academic performance, attendances, etc. According to a recent statistics, we find out that at least four billion people have access to the internet; it is safe to say, that the internet is a part of our daily lives. Moreover, with the recent outbreak of the Covid-19 pandemic, there was a strong shift to online platforms because of the lockdown restrictions that meant more and more people interacted on the internet, further highlighting the fault of the old system in collecting, accessing and storing data. For instance, the use of books and papers to collect student's marks and other information. All the above remarks show that there is a need for Student Information System software for management of all educational data.

There are numerous branches of organization for the preservation of students' data in any institution in the world. Every one of these offices keep and give different records concerning student information. The information about the students has to be updated on a regular basis. The relevant information to be collected would be student details like name, age, performance, attendance, marks, subjects, and so forth. Every one of the modules in universities are related and their maintenance is done manually. The information between subjects of study and activities can be related, hence, the need to cluster and automate it.

An example of the system in use would be, a student looking to apply for a leave of absence in advance, the HOD seeing the application request for authentication of the reason from the students and the teacher before approving. This proves as a form of

accountability and avoidance of unnecessary lies, which may bolster average student's attendance.

## **2.2 Description of applied mathematical methods**

Mathematical methods are not used in the system developed.

## **2.3 Description of used technologies and programming languages**

This web service has been developed using the following technologies and programming languages:

- Python;
- Django Framework;
- Bootstrap;
- HTML;
- CSS;
- JavaScript;
- SQLite database.

### **Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It is a high-level built language in data structures combined with dynamic typing and binding. This make it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together. Python is a simple easy to learn syntax that emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program does not catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables. It allows evaluation of arbitrary expressions. It permits setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python differs from many languages because it is one of the few that is easy to learn and easy to understand. Many consider Ruby a great place to start, like Python, yet the latter has a four-year head start. This means that it has a big foothold in the enterprise world, and it is much more popular with C developers. This is because it is easy to crossover between the two languages. Both Ruby and Python share a significant amount of growth in the job market, so choosing either language would be beneficial in terms of a career. PHP is also used often though the application of the language is different. Ultimately, it comes down to what type of content you will be developing, as each language has its niche.

- Cross platform:

Python is a cross-platform language: a Python program written on a Macintosh computer will run on a Linux system and vice versa. Python programs can run on a Windows computer, as long as the Windows machine has the Python interpreter installed, (most other operating systems come with Python pre-installed).

- Memory management:

Having a good understanding of how chunks of memory are allocated, re-used, and de-allocated for python objects enables you to write code that is more efficient and

solve many issues related to extra memory that your program pulls. Python memory management plays a major role to make it much more popular and adaptable.

## **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 (see website security[15] for more details of such attacks).

- 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.

- 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.

## **Bootstrap**

Bootstrap is a free toolkit for building site and web application interfaces. Its capabilities are focused solely on front-end development. Bootstrap is a very popular project, as evidenced, for example, by the fact that it ranks (as of early March 2018) the second largest star on GitHub.

The main advantage of using Bootstrap is less, a dynamic style language that greatly enhances CSS. With it, developers can create variables, nested columns, manage colors, etc. Also Less is very easy to use. Simply paste the code into the pages. Basic Bootstrap tools:

- **Grids:**

Predefined column sizes that can be used immediately, such as the 90px column width refers to the `.span2` class that we can use in the CSS document description.

- **Templates:**

Fixed or rubber document template.

- **Typography:**

Font descriptions, defining some classes for such fonts like code, quotes, etc.

- **Media:**

It presents image and video management capabilities.

- **Tables:**

It a means of designing tables, before adding functionality to enable sorting.

- **Forms:**

It classes for designing not only forms but also some events, what is happening to them.

- **Navigation:**

It is layout classes for tabs, pages, menus, and panel's tools.

- **Alert:**

Its design dialog boxes, tooltips, and pop-ups.

Examples of sites created using the Bootstrap framework:

netflix.com - Netflix, the world's largest online provider of series and films;

Gitlab.com - Gitlab, one of the world's largest code repositories;

Toyota.com - Toyota Motor, the world's largest car company.

## **HTML**

HTML can be called the main language of the World Wide Web. Most web pages hosted on the web are written in any variation of HTML. With it, developers determine how multimedia, text, or hyperlinks will be displayed among other content in the browser.

From the elements that link to your document (hypertext) to the elements that make those documents interactive (such as forms), these are all part of HTML. W3C or the World Wide Web Consortium developed the HTML standard in 1997. In HTML, tags are used to define text structure; tags and elements are highlighted using `<and>` characters. Some examples of the tags mentioned above are headings, tables, paragraphs, etc.

In turn, browsers are responsible for rendering the content of a page using these tags. HTML was not the only standard for web development. In the early days of internet development, all content and style tags were present in one giant, cumbersome (and quite complex) language. Subsequently, the W3C came to a decision about the need to share content and page style. This led to the creation of style sheets or CSS. Currently, tags used to define text style (such as font) are unwanted and almost unused, with style sheets coming in, and only content definition tags (such as h1) still form the core of HTML.

There have been many updates to HTML over time, and it is currently the latest version of HTML5. It is, of course, primarily a markup language, but it has received many features unlike HTML and has eliminated some of the severe restrictions that were present in XHTML. Although HTML5 is updated almost daily, there are no new numbered releases. The main difference between HTML and HTML5 is that neither audio nor video is an integral part of HTML, while both can be regarded as integral parts of HTML5.



## **Cascading CSS Style Sheets**

It is a technology for describing the appearance of a document written in a markup language. It is mainly used as a HTML and XHTML web page design tool, but can be used with any type of XML document, including SVG and XUL.

## **JavaScript**

JavaScript is a multi-paradigm programming language. Supports object-oriented, imperative and functional styles. JavaScript is commonly used as a built-in language for programming access to application objects. The most widely used is in Browsers as a scripting language to make web pages interactive.

Main architectural features: dynamic typification, weak typification, automatic memory management, prototype programming, functions as objects first class. Script do not need special training or compilation to run.

## **SQLite database**

Django in its 'out-of-the-box' state is set up to communicate with SQLite -- a lightweight relational database included with the Python distribution. Therefore by default, Django automatically creates a SQLite database for your project. In addition to SQLite, Django supports (i.e. included in Django itself) four other popular relational databases that include PostgreSQL, MySQL, Maria DB and Oracle. Django supports connectivity to other relational databases that include SAP (Sybase) SQL Anywhere, IBM DB2 and Firebird, as well as the ADO (ActiveX Data Objects) and ODBC (Open Database Connectivity) interfaces, the last two of which are standard for connecting to Microsoft SQL Server and the latter is supported by most relational database brands. The Django configuration to connect to a database is done inside the `setting.py` file (Fig.2.) of a Django project in the database variable (Fig.2.11).

## 2.4 Description of the system structure and algorithms of its functioning

The structure of the Student Management System web app is interconnected files, where each file performs its corresponding function. For the convenience of work with the project, and its further expansion, files are located on the folders corresponding to them where each folder (fig. 2.1) contains files on the maintenance with their respective content.

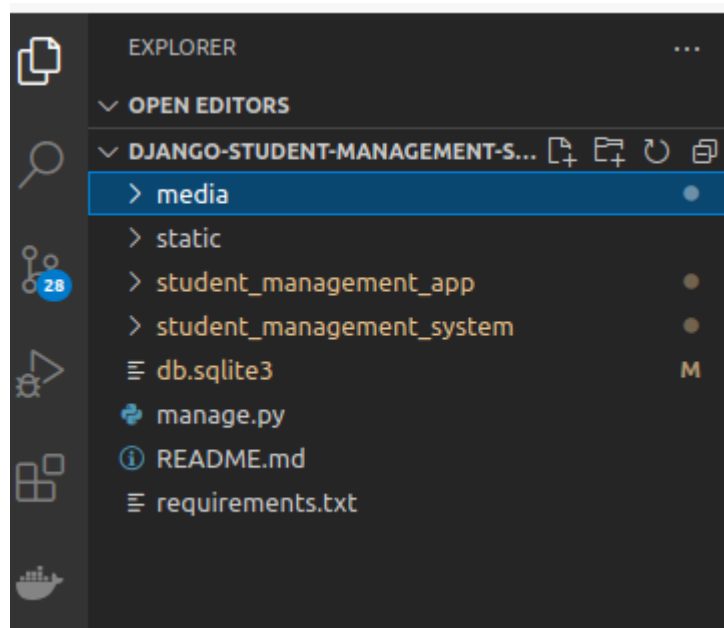


Fig.2.1. The structure of the folders of the SMS

The media folder (Fig. 2.2) contains uploaded files by users on the system. The uploaded files are JPG files. Most of the uploaded files in the media folder will be student's profile pictures.

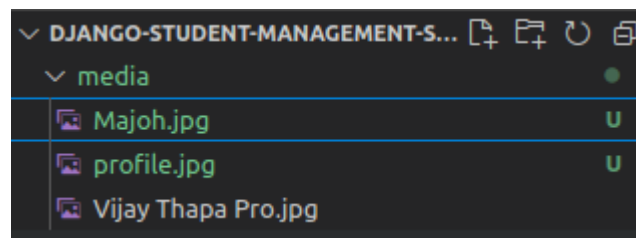


Fig.2.2. Contents of the media folder

Media files should not be trusted like static files.

The static folder (Fig.2.3) contains files like CSS, JavaScript, and fonts. These files are the core piece of any modern web application. Django provides tremendous flexibility around *how* these files are used, however this often leads to confusion for newcomers. For local development, the Django web server will serve static files and minimal configuration is required.

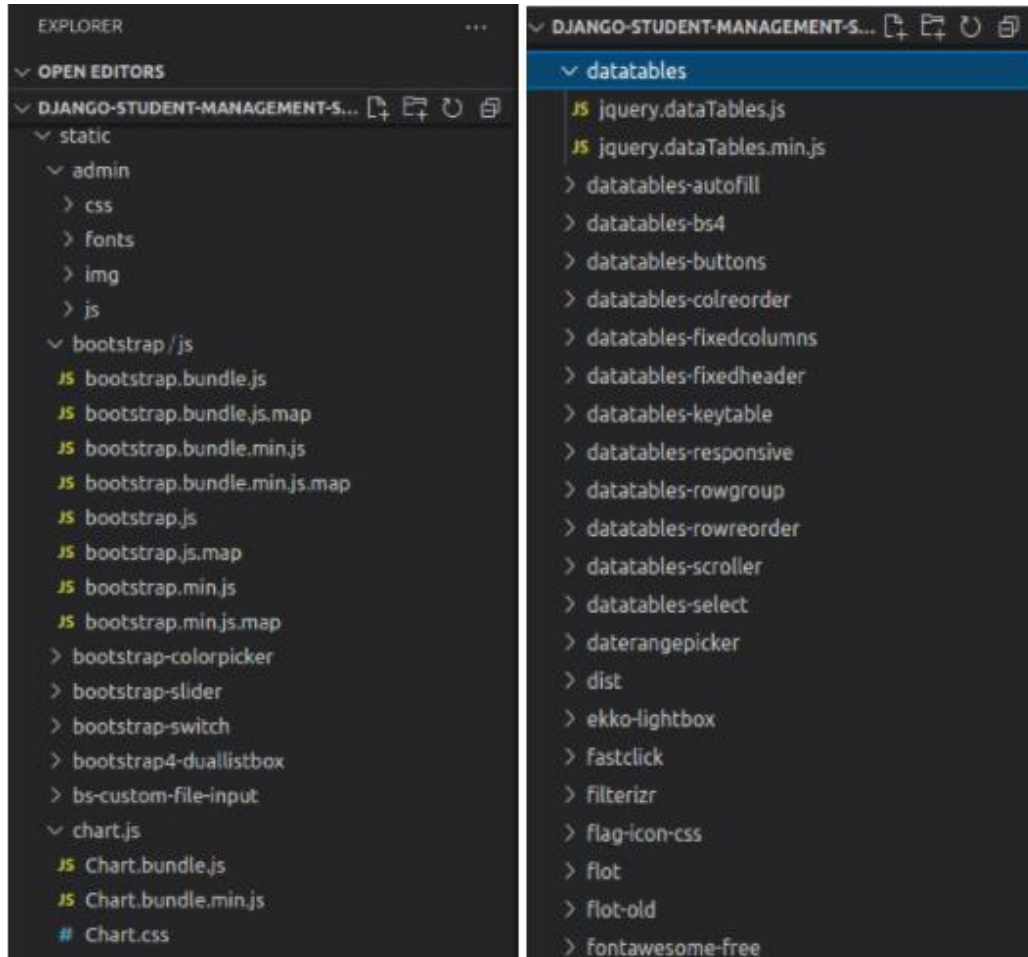


Fig.2.3. Contents of the static folder.

The Student\_management\_app folder (Fig. 2.4) is the app structure on the SMS web app. This folder contains python files that are the center of the development of the web app.

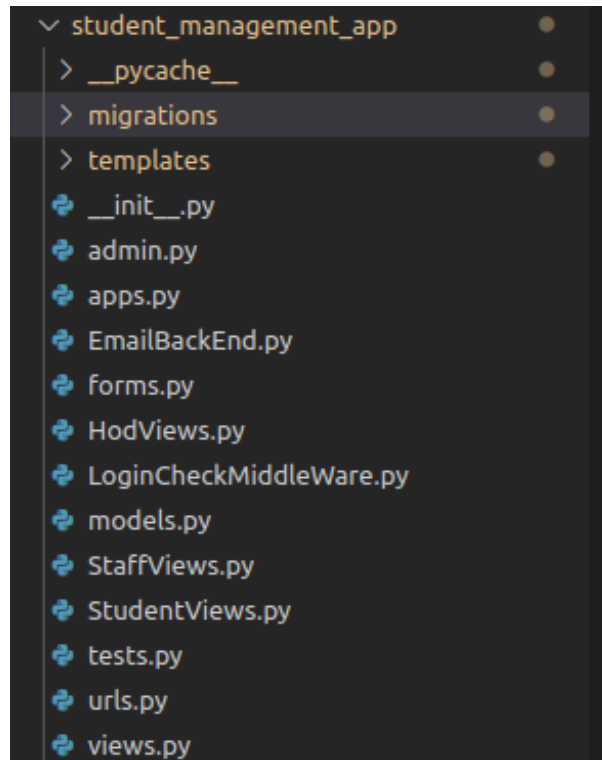


Fig.2.4. Contents of the student\_management\_app folder.

- `_init_.py`

This file (Fig.2.5) has the same functionality just as in the `_init_.py` file in the Django project structure. It remains empty and is present just to indicate that the specific app directory is a package.

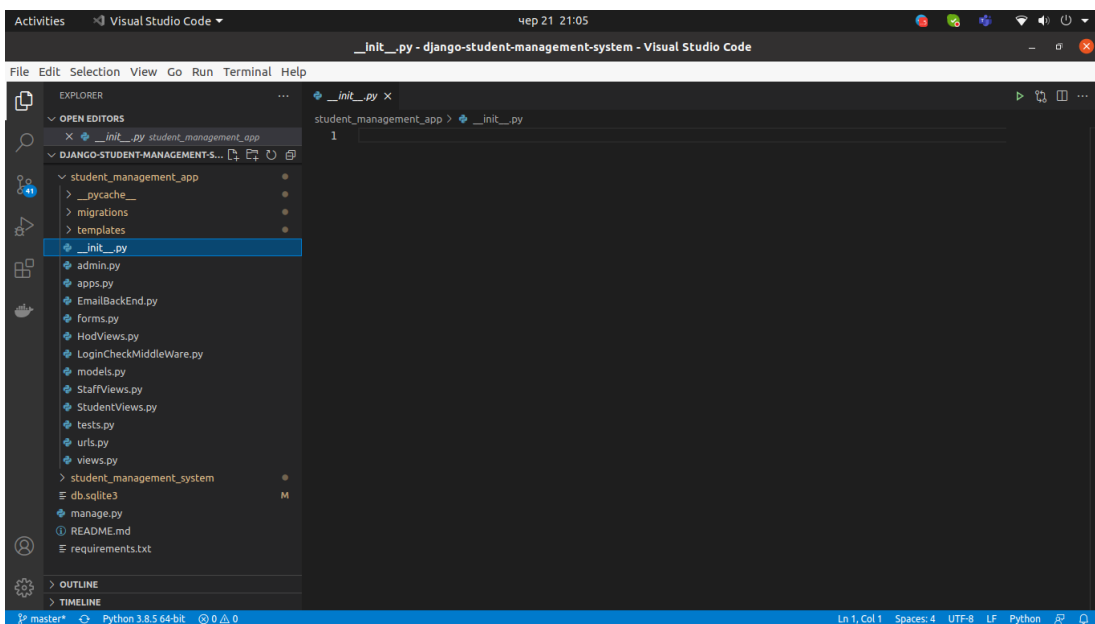
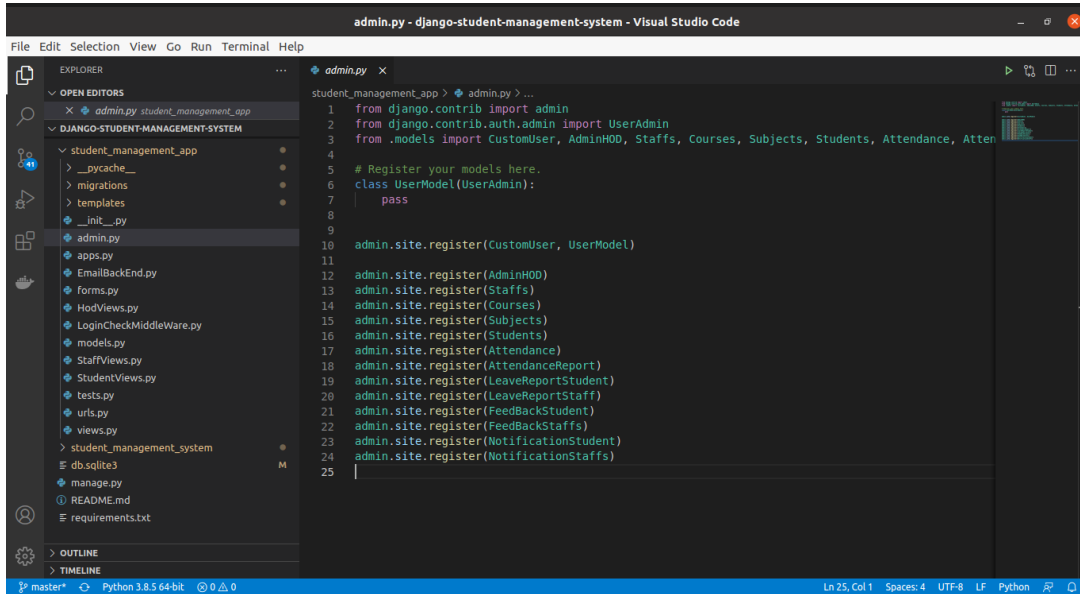


Fig.2.5. Contents of `_init_.py` file.

- Admin.py

As the name suggests, this file (Fig.2.6) is used for registering the models into the Django administration. The models that are present have a superuser/admin who can control the information that is being stored.

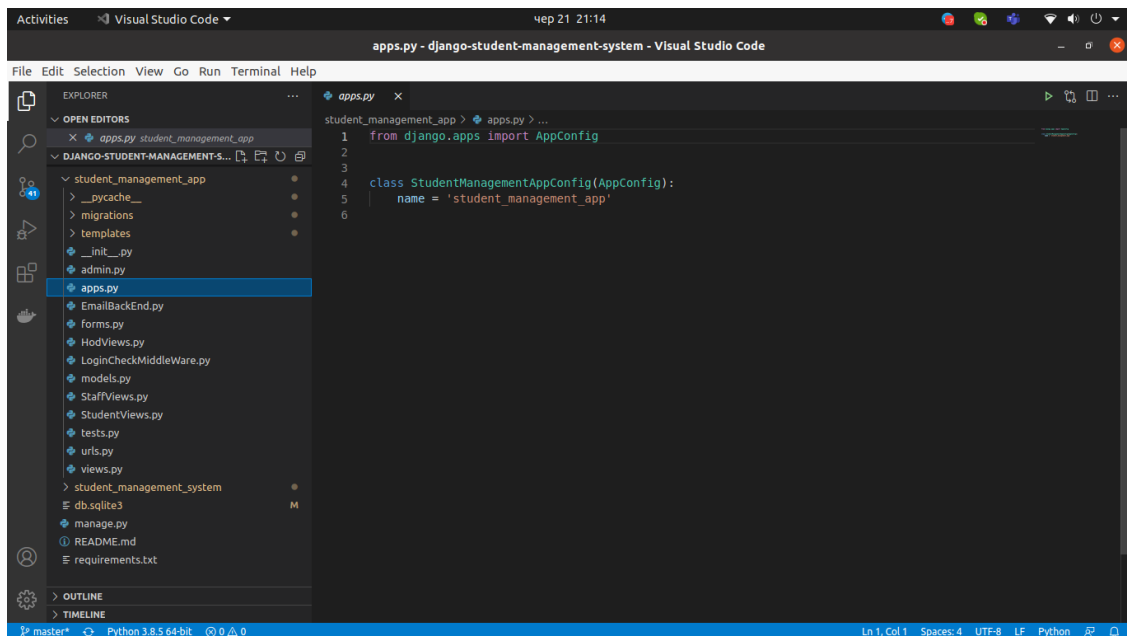


```
1 from django.contrib import admin
2 from django.contrib.auth.admin import UserAdmin
3 from models import CustomUser, AdminHOD, Staffs, Courses, Subjects, Students, Attendance, Atten
4
5 # Register your models here.
6 class UserModel(UserAdmin):
7     pass
8
9
10 admin.site.register(CustomUser, UserModel)
11
12 admin.site.register(AdminHOD)
13 admin.site.register(Staffs)
14 admin.site.register(Courses)
15 admin.site.register(Subjects)
16 admin.site.register(Students)
17 admin.site.register(Attendance)
18 admin.site.register(AttendanceReport)
19 admin.site.register(LeaveReportStudent)
20 admin.site.register(LeaveReportStaff)
21 admin.site.register(FeedbackStudent)
22 admin.site.register(FeedbackStaffs)
23 admin.site.register(NotificationStudent)
24 admin.site.register(NotificationStaffs)
25
```

Fig.2.6. Contents of the admin.py file.

- apps.py

This file (Fig.2.7) deals with the application configuration of the apps. The default configuration is sufficient in most of the cases.



```
1 from django.apps import AppConfig
2
3
4 class StudentManagementAppConfig(AppConfig):
5     name = 'student_management_app'
6
```

Fig.2.7. Contents of the apps.py file.

- EmailBackend.py

This file (Fig.2.8) is responsible for sending and receiving emails.

```

1 from django.contrib.auth import get_user_model
2 from django.contrib.auth.backends import ModelBackend
3
4
5
6
7 class EmailBackend(ModelBackend):
8     def authenticate(self, username=None, password=None, **kwargs):
9         UserModel = get_user_model()
10        try:
11            user = UserModel.objects.get(email=username)
12        except UserModel.DoesNotExist:
13            return None
14        else:
15            if user.check_password(password):
16                return user
17        return None

```

Fig.2.8. Contents of the apps.py file.

- AdminHODviews.py

This file (Fig.2.9) is responsible for the view of the HOD(Admin) board.

```

1 from django.shortcuts import render, redirect
2 from django.http import HttpResponseRedirect, JsonResponse
3 from django.contrib import messages
4 from django.core.files.storage import FileSystemStorage #To upload Profile Picture
5 from django.urls import reverse
6 from django.views.decorators.csrf import csrf_exempt
7 from django.core import serializers
8 import json
9
10 from student_management_app.models import CustomUser, Staffs, Courses, Subjects, Students, Sess
11 from .forms import AddStudentForm, EditStudentForm
12
13
14 def admin_home(request):
15     all_student_count = Students.objects.all().count()
16     subject_count = Subjects.objects.all().count()
17     course_count = Courses.objects.all().count()
18     staff_count = Staffs.objects.all().count()
19
20     # Total Subjects and students in Each Course
21     course_all = Courses.objects.all()
22     course_name_list = []
23     subject_count_list = []
24     student_count_list_in_course = []
25
26
27     for course in course_all:
28         subjects = Subjects.objects.filter(course_id=course.id).count()
29         students = Students.objects.filter(course_id=course.id).count()
30         course_name_list.append(course.course_name)
31         subject_count_list.append(subjects)
32         student_count_list_in_course.append(students)

```

Fig.2.9. Contents of the AdminHODviews.py file.

- LoginCheckMiddleWare.py

This file (Fig.2.10) contains code that check and verify login process.

```

1 from django.utils.deprecation import MiddlewareMixin
2 from django.shortcuts import render, redirect
3 from django.urls import reverse
4
5 (class) MiddlewareMixin
6 class LoginCheckMiddleware(MiddlewareMixin):
7
8     def process_view(self, request, view_func, view_args, view_kwargs):
9         modulename = view_func.__module__
10        # print(modulename)
11        user = request.user
12
13        #Check whether the user is logged in or not
14        if user.is_authenticated:
15            if user.user_type == "1":
16                if modulename == "student_management_app.HodViews":
17                    pass
18                elif modulename == "student_management_app.views" or modulename == "django.views":
19                    pass
20                else:
21                    return redirect("admin_home")
22
23            elif user.user_type == "2":
24                if modulename == "student_management_app.StaffViews":
25                    pass
26                elif modulename == "student_management_app.views" or modulename == "django.views":
27                    pass
28                else:
29                    return redirect("staff_home")
30
31            elif user.user_type == "3":
32                if modulename == "student_management_app.StudentViews":
33                    pass
34                else:
35                    return redirect("student_home")

```

Fig.2.10. Contents of the AdminHODviews.py file.

- Models.py

This file (Fig.2.11) contains the models of our web applications (usually as classes). Models are basically the blueprints of the database we are using and hence contain the information regarding attributes and the fields etc of the database.

```

1 from django.contrib.auth.models import AbstractUser
2 from django.db import models
3 from django.db.models.signals import post_save
4 from django.dispatch import receiver
5
6 class SessionYearModel(models.Model):
7     id = models.AutoField(primary_key=True)
8     session_start_year = models.DateField()
9     session_end_year = models.DateField()
10    objects = models.Manager()
11
12 # Overriding the Default Django Auth User and adding One More Field (user_type)
13 class CustomUser(AbstractUser):
14    user_type_data = ((1, "HOD"), (2, "Staff"), (3, "Student"))
15    user_type = models.CharField(default=1, choices=user_type_data, max_length=10)
16
17 class AdminHOD(models.Model):
18    id = models.AutoField(primary_key=True)
19    admin = models.OneToOneField(CustomUser, on_delete = models.CASCADE)
20    created_at = models.DateTimeField(auto_now_add=True)
21    updated_at = models.DateTimeField(auto_now=True)
22    objects = models.Manager()
23
24 class Staffs(models.Model):
25    id = models.AutoField(primary_key=True)

```

Fig.2.10. Contents of the models.py file.

- Staffsviews.py

This file (Fig.2.12) is responsible for the view of the staff board.

```

1 from django.shortcuts import render, redirect
2 from django.http import HttpResponse, HttpResponseRedirect, JsonResponse
3 from django.contrib import messages
4 from django.core.files.storage import FileSystemStorage #To upload Profile Picture
5 from django.urls import reverse
6 from django.views.decorators.csrf import csrf_exempt
7 from django.core import serializers
8 import json
9
10
11 from student_management_app.models import CustomUser, Staffs, Courses, Subjects, Students, Sessi
12
13
14 def staff_home(request):
15     # Fetching All Students under Staff
16
17     subjects = Subjects.objects.filter(staff_id=request.user.id)
18     course_id_list = []
19     for subject in subjects:
20         course = Courses.objects.get(id=subject.course_id.id)
21         course_id_list.append(course.id)
22
23     final_course = []
24     # Removing Duplicate Course Id
25     for course_id in course_id_list:
26         if course_id not in final_course:
27             final_course.append(course_id)
28
29     students_count = Students.objects.filter(course_id__in=final_course).count()
30     subject_count = subjects.count()
31
32     # Fetch All Attendance Count

```

Fig.2.11. Contents of the models.py file.

- Studentsviews.py

This file (Fig.2.12) is responsible for the view of student board.

```

1 from django.shortcuts import render, redirect
2 from django.http import HttpResponse, HttpResponseRedirect
3 from django.contrib import messages
4 from django.core.files.storage import FileSystemStorage #To upload Profile Picture
5 from django.urls import reverse
6 import datetime # To Parse input DateTime into Python Date Time Object
7
8 from student_management_app.models import CustomUser, Staffs, Courses, Subjects, Students, Attendance, Atte
9
10
11 def student_home(request):
12     student_obj = Students.objects.get(admin=request.user.id)
13     total_attendance = AttendanceReport.objects.filter(student_id=student_obj).count()
14     attendance_present = AttendanceReport.objects.filter(student_id=student_obj, status=True).count()
15     attendance_absent = AttendanceReport.objects.filter(student_id=student_obj, status=False).count()
16
17     course_obj = Courses.objects.get(id=student_obj.course_id.id)
18     total_subjects = Subjects.objects.filter(course_id=course_obj).count()
19
20     subject_name = []
21     data_present = []
22     data_absent = []
23     subject_data = Subjects.objects.filter(course_id=student_obj.course_id)
24     for subject in subject_data:
25         attendance = Attendance.objects.filter(subject_id=subject.id)
26         attendance_present_count = AttendanceReport.objects.filter(attendance_id__in=attendance, status=True)
27         attendance_absent_count = AttendanceReport.objects.filter(attendance_id__in=attendance, status=False)
28         subject_name.append(subject.subject_name)
29         data_present.append(attendance_present_count)
30         data_absent.append(attendance_absent_count)
31
32     context = {

```

Fig.2.12. Contents of the models.py file.



- Tests.py

This This file contains the code that contains different test cases for the application. It is used to test the working of the application. We will not be working on this file.

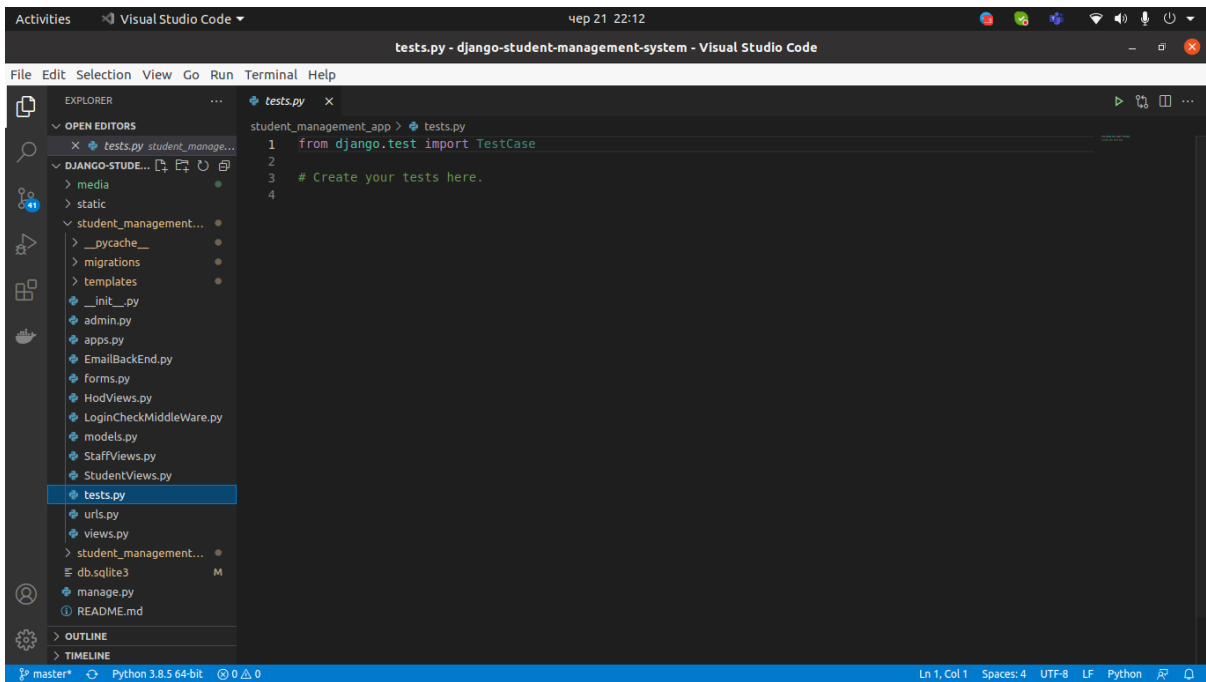


Fig.2.12. Contents of the tests.py file.

- Urls.py

This file (Fig.2.13) handles all the URLs of our web application.

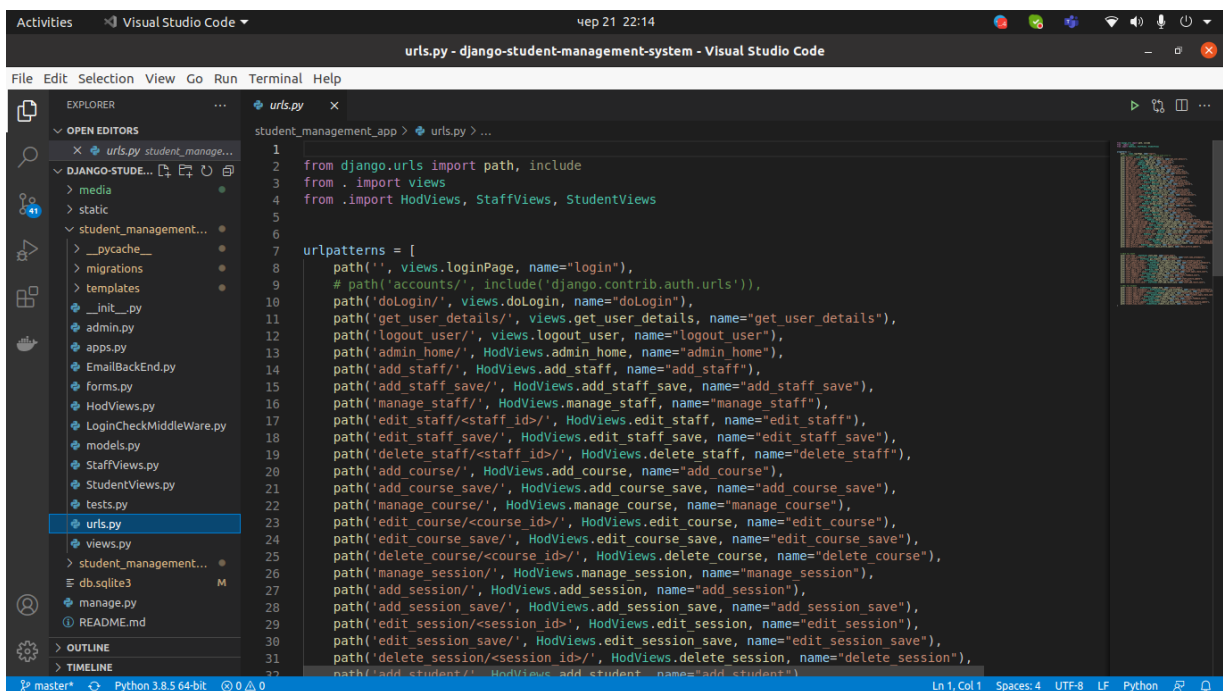


Fig.2.13. Contents of the tests.py file.

- Views.py

This file (Fig.2.14) contains all the views (usually as classes). Views.py can be considered as a file that interacts with the client. Views are a user interface for what we see when we render a Django Web application.

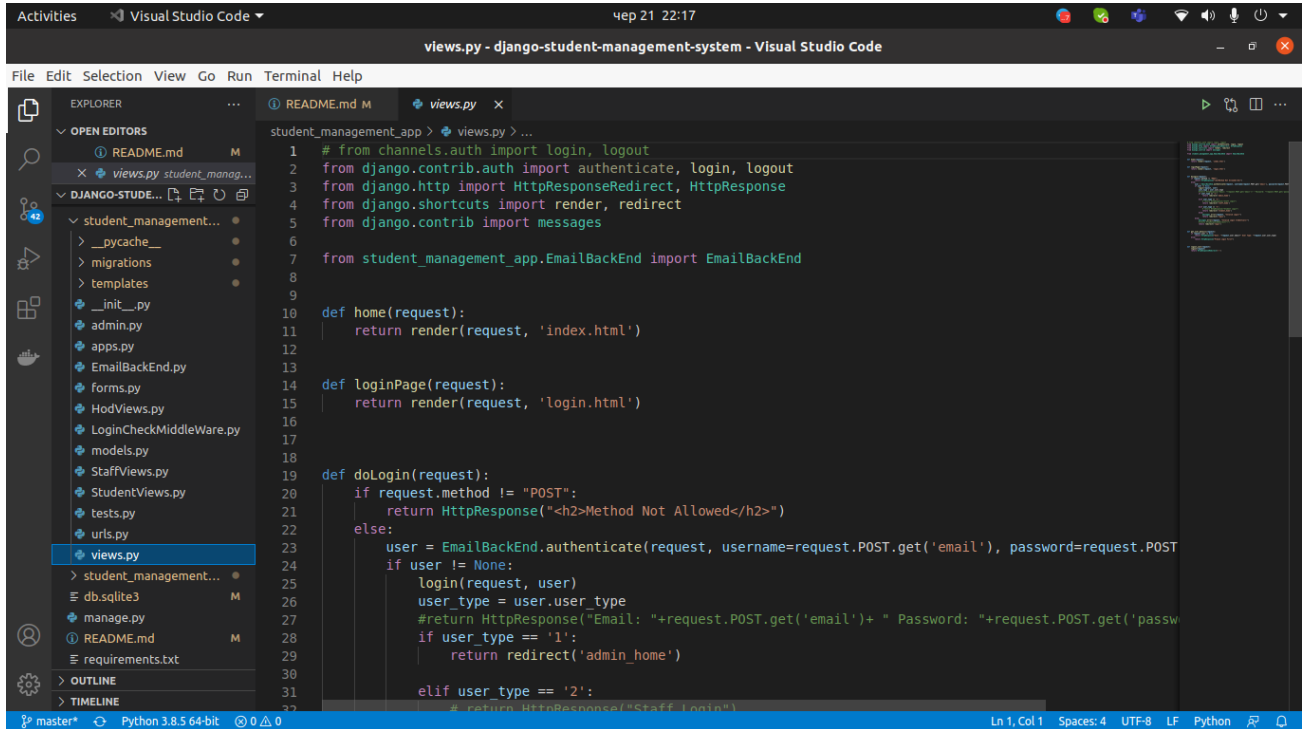


Fig.2.14. Contents of the views.py file.

The Student\_management\_system folder (Fig. 2.15) is also composed of python files. Generally the setup is the default project structure.

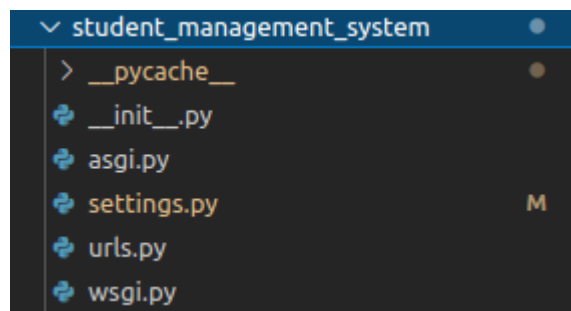


Fig.2.15. Contents of the views.py file.

- Asgi.py

In the newer versions of Django, you will also find a file named as asgi.py (Fig.2.16) apart from wsgi.py. ASGI can be considered as a successor interface to the

WSGI. ASGI also has the work similar to WSGI but this is better than the previous one as it gives better freedom in Django development. That is why WSGI is now being increasingly replaced by ASGI.

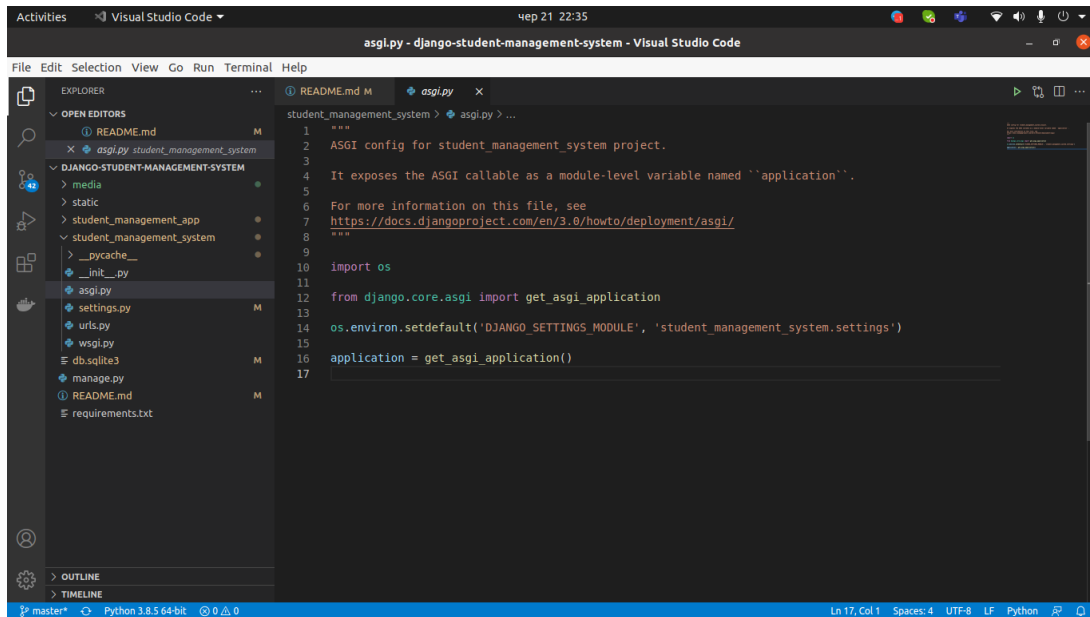


Fig.2.16. Contents of the asgi.py file

- Settings.py

This file (Fig.2.17) is present for adding all the applications and the middleware application present. Also, it has information about templates and databases. Overall, this is the main file of our Django web application.

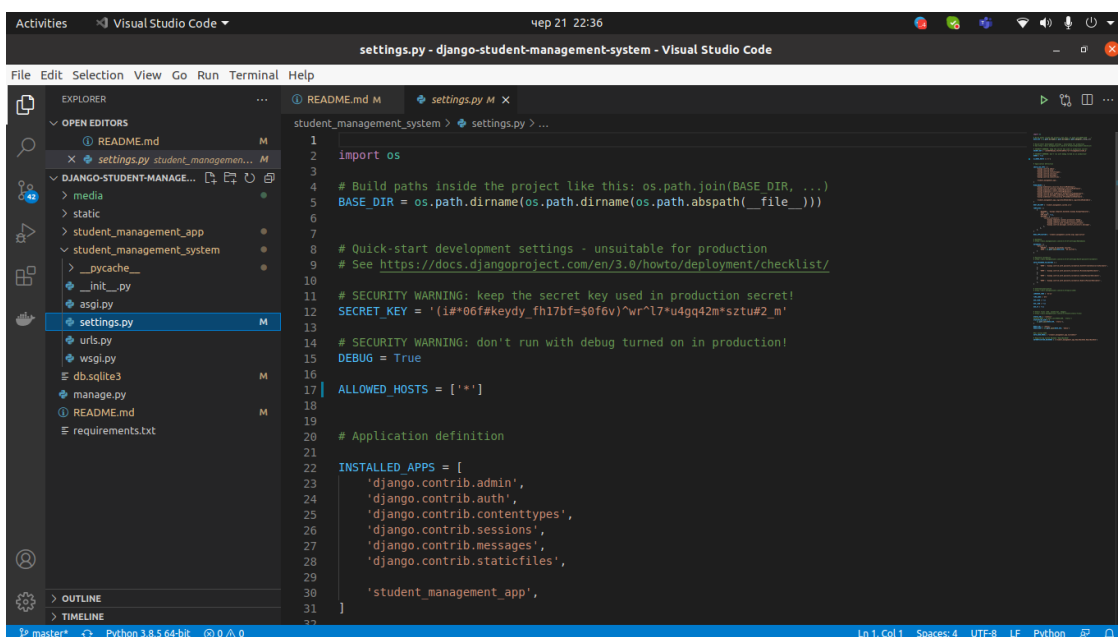
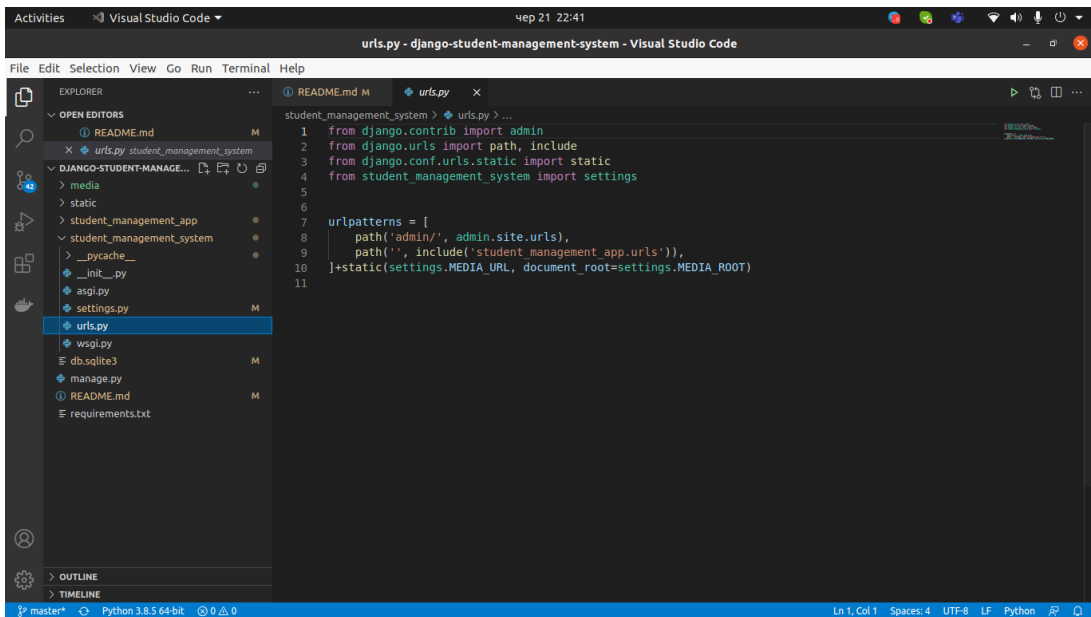


Fig.2.17. Contents of the settings.py file.

- urls.py

This file (Fig.2.18) handles all the URLs of the web application. This file has the lists of all the endpoints that we will have for our website.

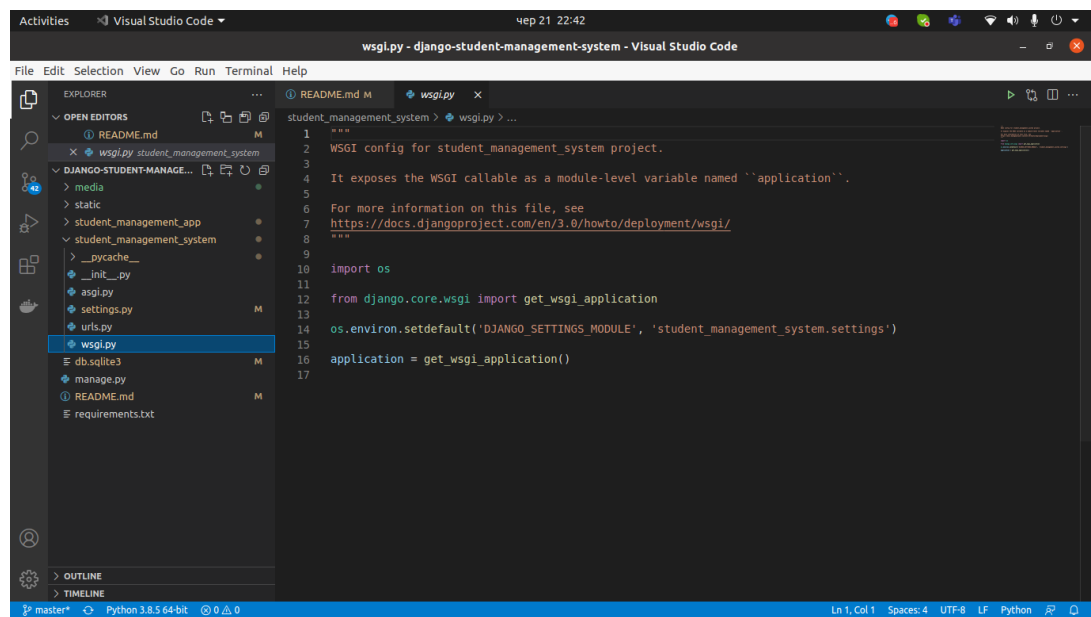


```
student_management_system > urls.py > ...
1 from django.contrib import admin
2 from django.urls import path, include
3 from django.conf.urls.static import static
4 from student_management_system import settings
5
6
7 urlpatterns = [
8     path('admin/', admin.site.urls),
9     path('', include('student_management_app.urls')),
10 ]+static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
11
```

Fig.2.18. Contents of the settings.py file.

- Wsgi.py

This file (Fig.2.19) mainly concerns the WSGI server and is used for deploying our applications onto servers like Apache etc. However if you intend to deploy the project on Apache, the code would like what is described in Fig.2.19.



```
student_management_system > wsgi.py > ...
1 """
2 WSGI config for student_management_system project.
3
4 It exposes the WSGI callable as a module-level variable named ``application``.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/3.0/howto/deployment/wsgi/
8 """
9
10 import os
11
12 from django.core.wsgi import get_wsgi_application
13
14 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'student_management_system.settings')
15
16 application = get_wsgi_application()
17
```

Fig.2.19. Contents of the Wsgi.py file.

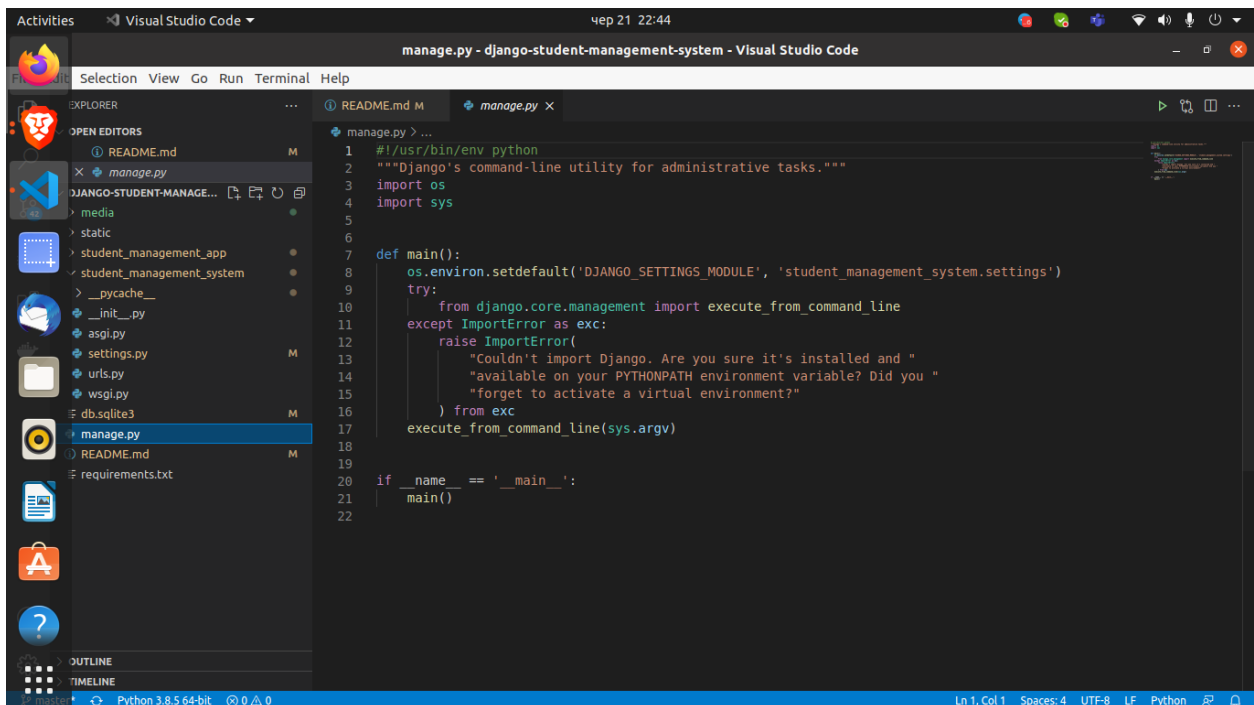
- manage.py

This file (Fig.2.20) is used as a command-line utility and for deploying, debugging, or running the web application. This file contains code for ‘runserver”, or “makemigrations” or “migrations”, etc. that we use in the terminal.

**Runserver:** this is the command to run the server for our web application.

**Migration:** this is used for applying the changes done to our models into the database. That is if we make any changes to our database then we use migrate command. This is used the first time we create a database.

**Makemigration:** this is done to apply new migrations that have been carried out due to the changes in the database.



```
1 #!/usr/bin/env python
2 """Django's command-line utility for administrative tasks."""
3 import os
4 import sys
5
6
7 def main():
8     os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'student_management_system.settings')
9     try:
10         from django.core.management import execute_from_command_line
11     except ImportError as exc:
12         raise ImportError(
13             "Couldn't import Django. Are you sure it's installed and "
14             "available on your PYTHONPATH environment variable? Did you "
15             "forget to activate a virtual environment?"
16         ) from exc
17     execute_from_command_line(sys.argv)
18
19
20 if __name__ == '__main__':
21     main()
22
```

Fig.2.20. Contents of the manage.py file.

There is also an important folder that we skipped in the listing of the structure of the student\_management\_app (Fig.2.4). I am referring to the folder called templates (Fig2.21). This is the folder where all the HTML files of the project are stored.

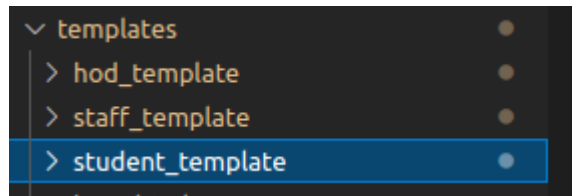


Fig.2.21. Contents of the templates folder.

The templates folder is sorted in three groups such as:

- Hod\_template

This template (Fig2.22) contains all the HTML files for the HOD (Admin) board.

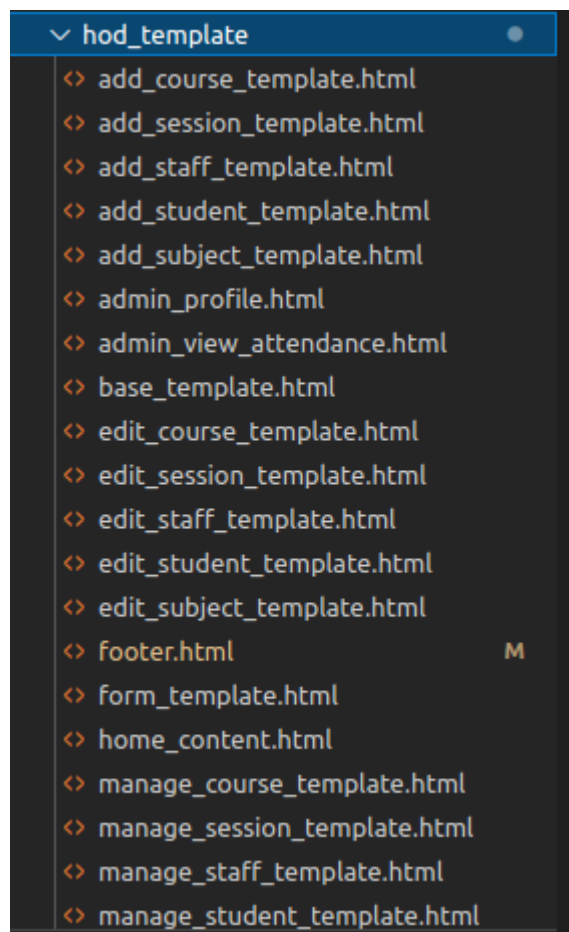


Fig.2.22. Contents of the HTML files of hod\_template's folder.

- Staff\_template

This template (Fig2.23) contains all the HTML files for the staff board.

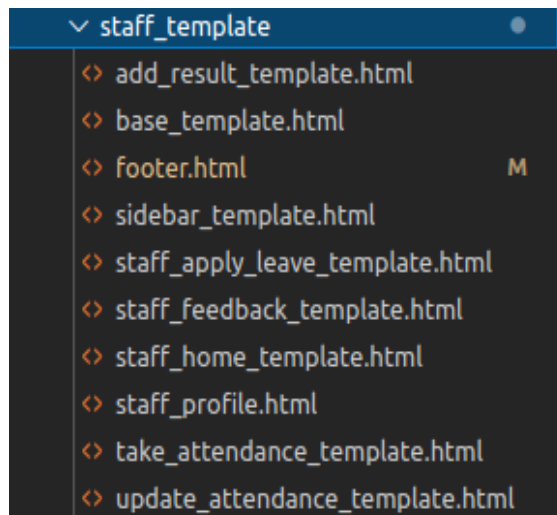


Fig.2.23. Contents of the HTML files of staff\_template's folder.

- Student\_template

This template (Fig2.24) contains all the HTML files for the student board.

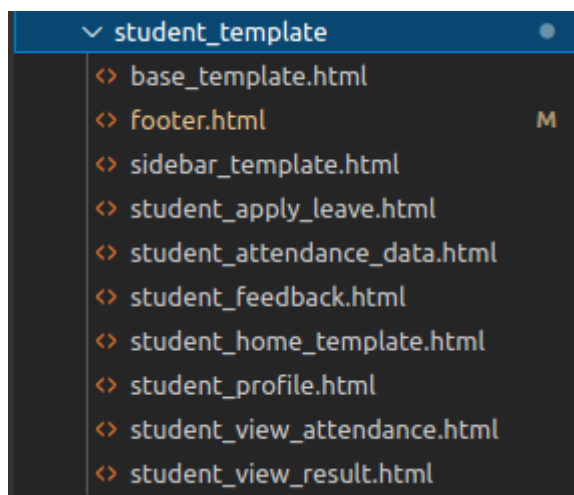


Fig.2.24. Contents of the HTML files of student\_template's folder

### 2.4.1 Relationship between different parts of the system structure.

In a traditional data-driven website, an internet application waits for HTTP requests from the online browser (or other client). When an invitation is received, the appliance works out what is needed supported the URL and possibly information in POST data or GET data. Counting on what is required it is going to then read or write information from a database or perform other tasks required to

satisfy the request. The appliance will then return a response to the online browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.

Django web applications typically group the code that handles each of those steps into separate files (Fig.2.25):

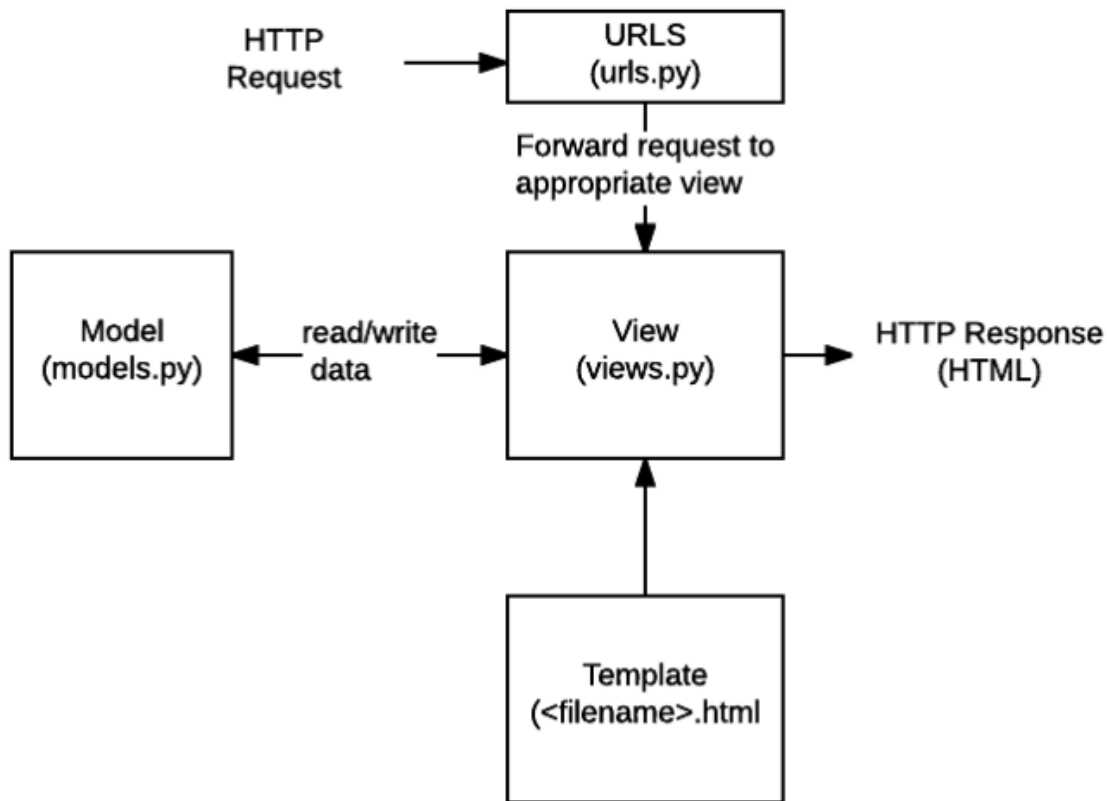


Fig.2.25 Relationship between different parts of the system structure.

### 2.4.2 Database of the Student Management System.

The structure of the database (Fig. 2.26) is as follows:



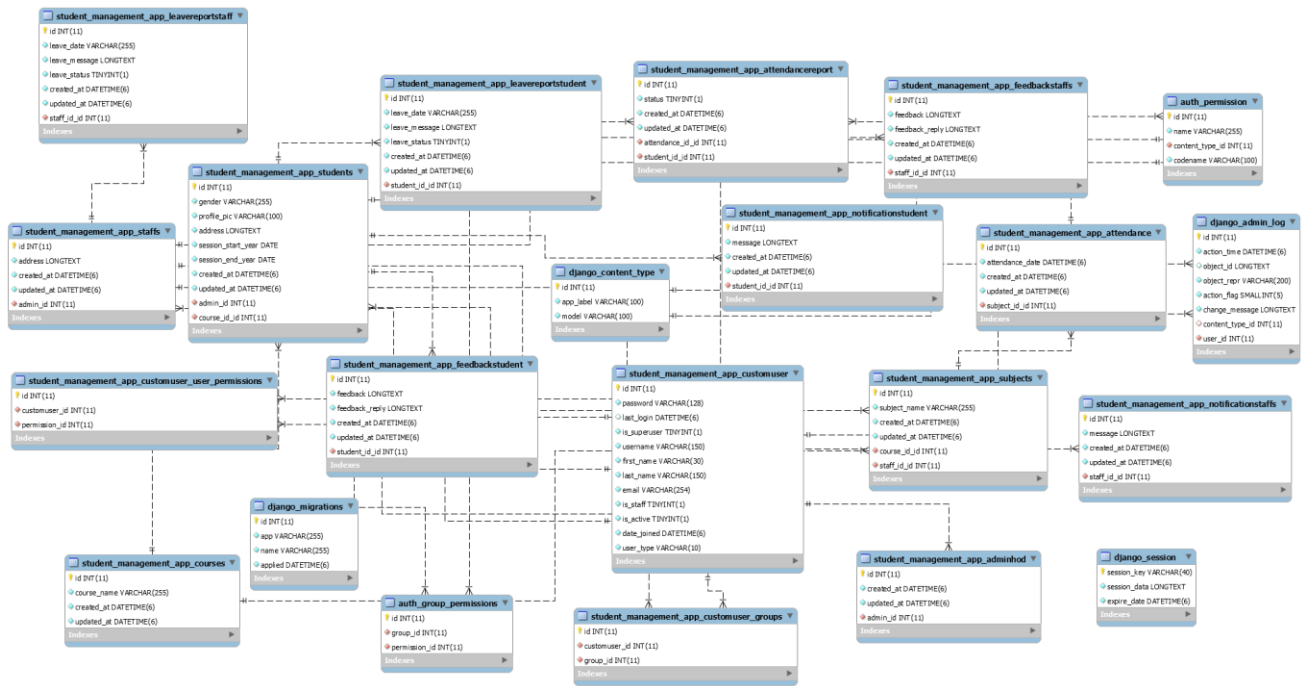


Fig.2.26 Database design.

### 2.4.3 Django architecture function view

The view retrieves data from the database via the model, formats it, bundles it up in an HTTP response object and sends it to the client (browser).

The structure of the architecture (Fig. 2.27) is as follows:

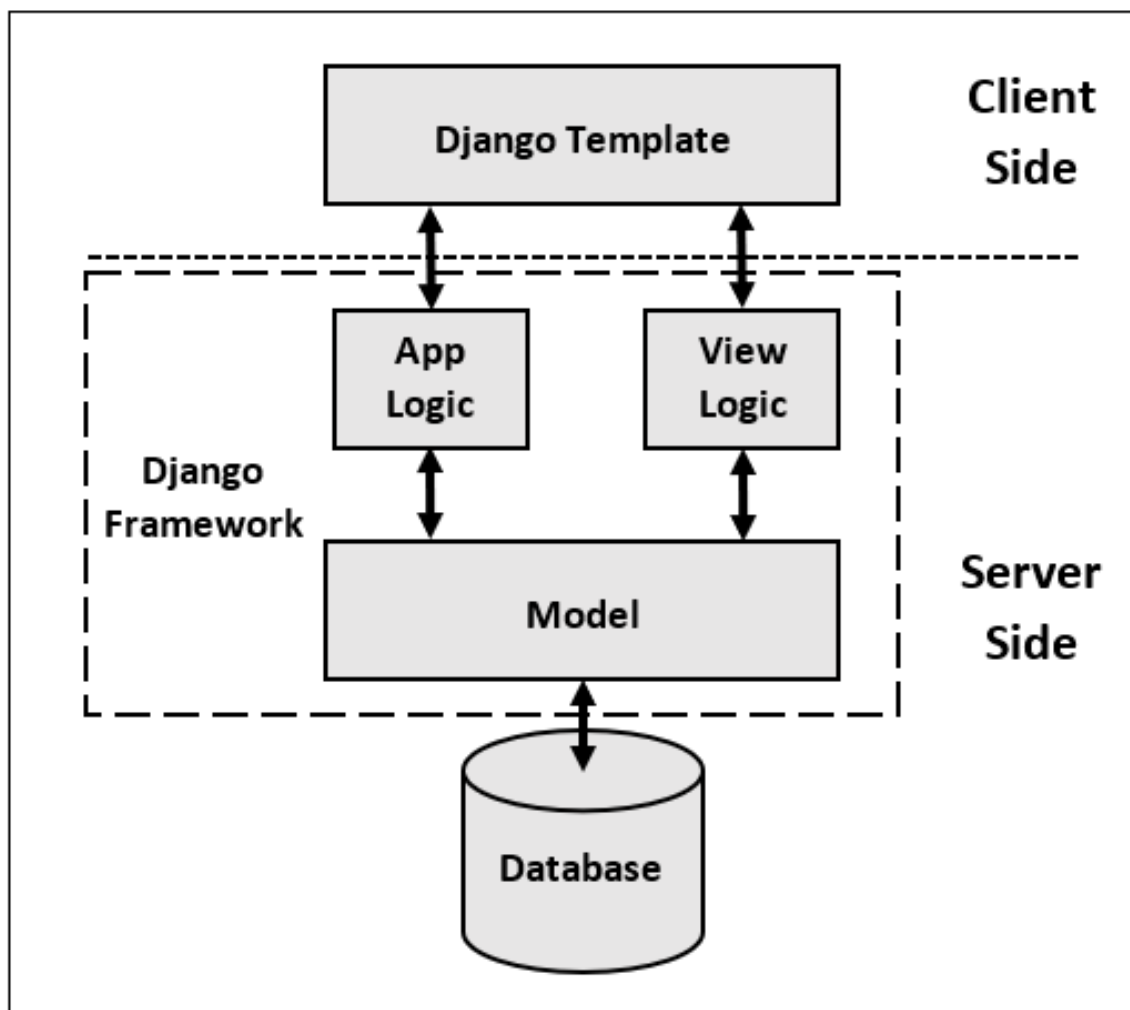


Fig.2.27 View of Django's architecture.

## 2.5 Justification and organization of input and output data of the program

The input for the web app system is plain text, UTF-8 encoded. The data is entered manually by the user and are transmitted to the server in its pure form, or using the Django forms format, which is a simple format for exchanging tribute, the data is transmitted in this way easy to handle and use on both the client side and the side server.

A form is a collection of elements inside `<form>...</form>` that allow a visitor to do things like enter text, select options, manipulate objects or controls, and so on, and then send that information back to the server.

The advantage of this format is also the possibility of always check input and output data without seeking help from third parties programs because the data is transmitted in plain text, and easily readable by both computer and human.

## **2.6 Description of the developed system**

### **2.6.1 Description of the developed system**

For the development of the web app, an HP ELITEBOOK - laptop, with the characteristic of Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz 2.70 GHz, 8GB RAM, running under Ubuntu 20.04.2 LTS was used.

### **2.6.2 The software used**

The following were used during the development of the qualification work:

- Text editor: VSCode.
- Local server: Ubuntu Terminal.
- Web browser: Brave Browser.
- Environment: Python environment.
- GitHub: Clone of the admin template.

### **2.6.3 Calling and downloading the program.**

To start working with the SMS, you need to download the system to a web server and go to its internet address. The system can also be used locally for development, or testing, using any local server.

### **2.6.4. Description of the user interface.**

The SMS has three User Interfaces:

- **The admin side**

The admin side (Fig.2.28), can see overall summary charts of students performance, staffs performances, courses, subjects, leave, etc.

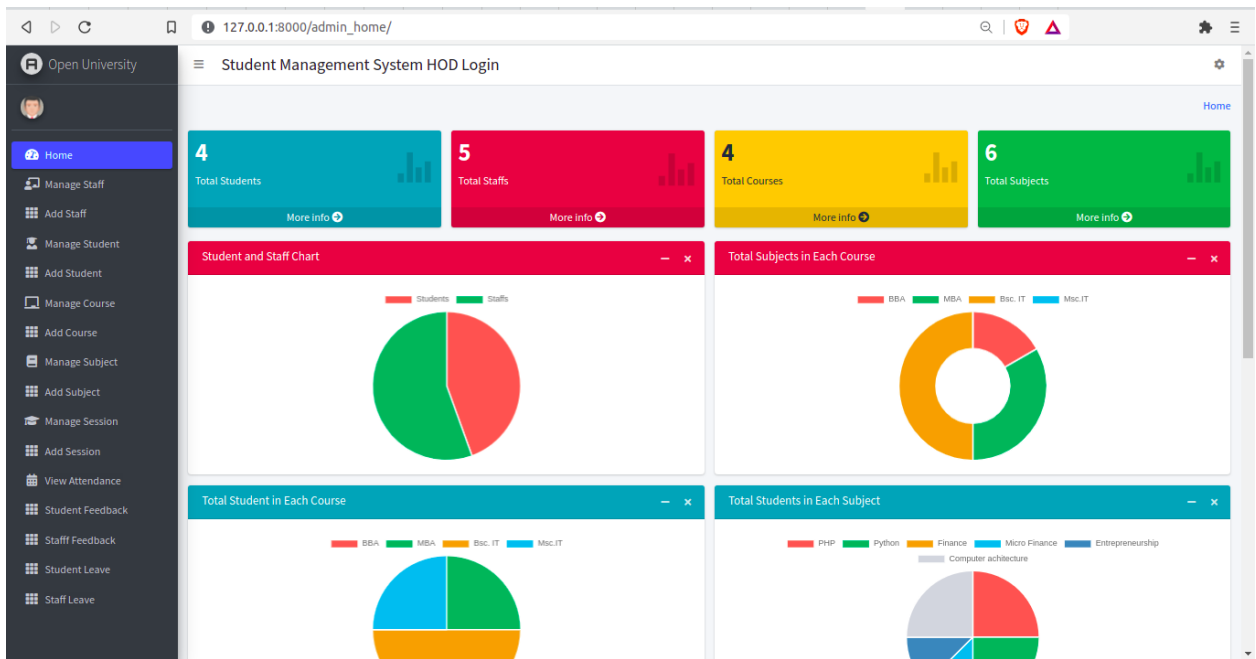


Fig.2.28 UI of the admin side

- **The staff/teacher side**

Staff/Teacher side (Fig2.29), can see the overall summary charts related to their students, their subjects, leave status, etc.

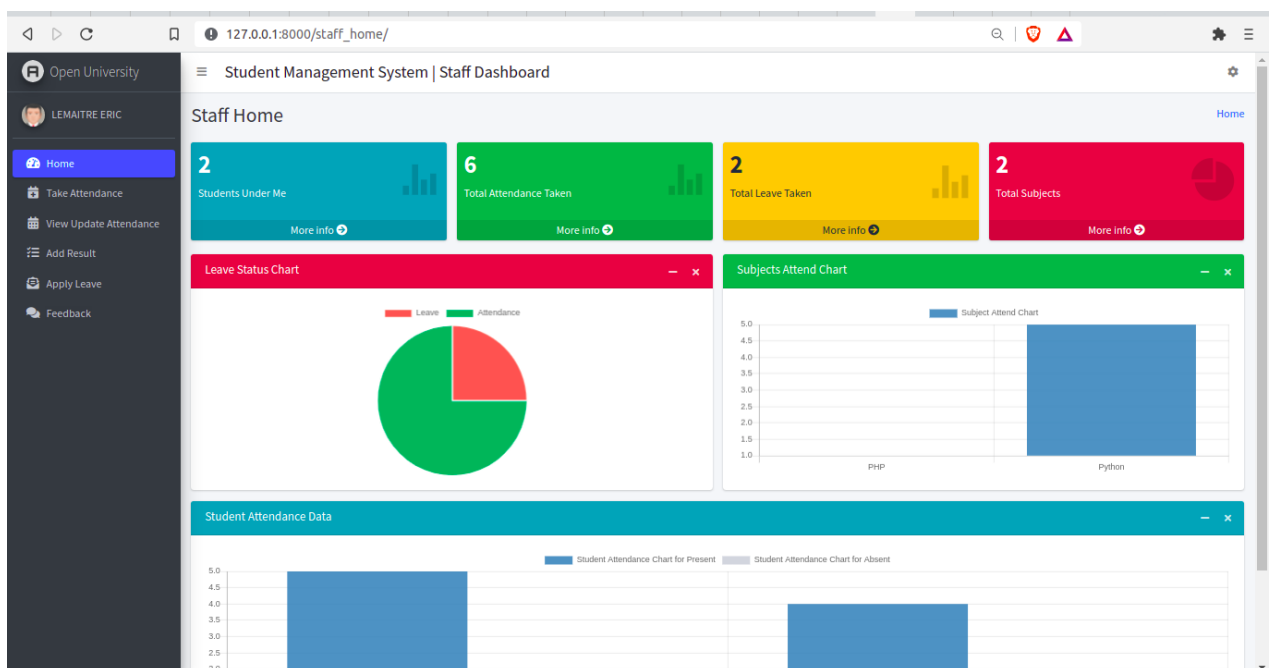


Fig.2.29 UI of the staff/teacher side

- **The student side**

The Student side (Fig2.30), can see the overall summary charts related to their attendance, their subjects, leave status, etc.

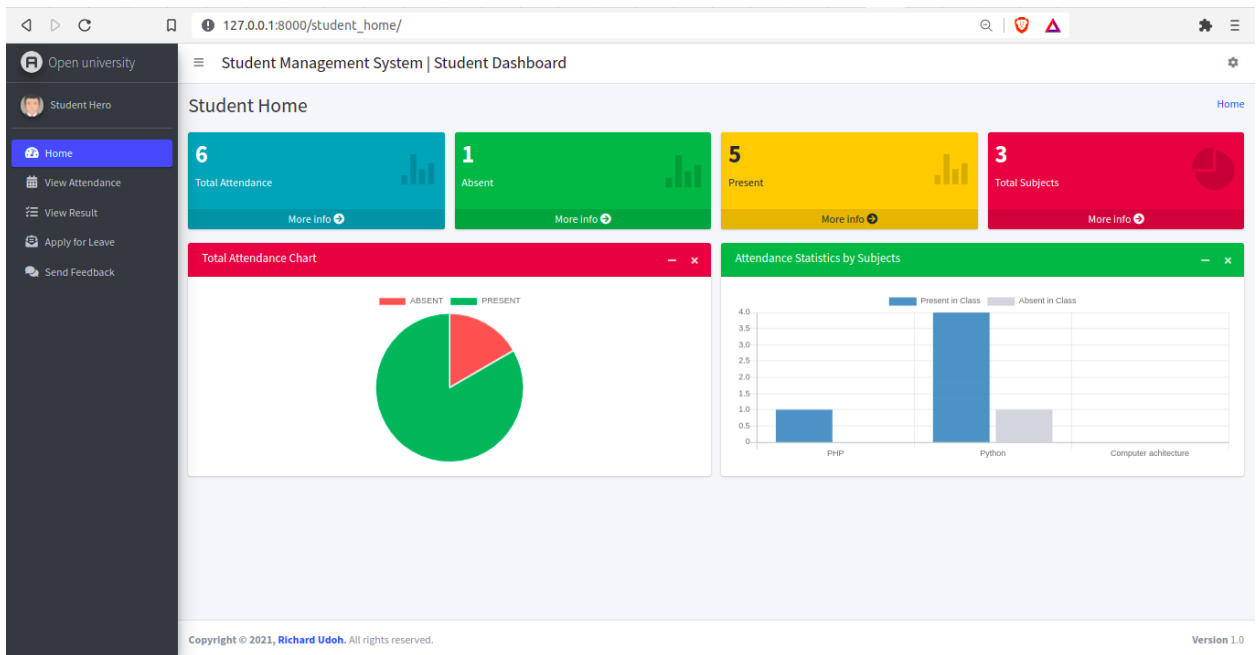


Fig.2.30 UI of the student side

The charts on the UI are included in the free Admin template theme (Fig 2.31) that we cloned from GitHub: <https://github.com/ColorlibHQ/AdminLTE>.

The template was edited to get this final design with the accurate data displayed on the chart.

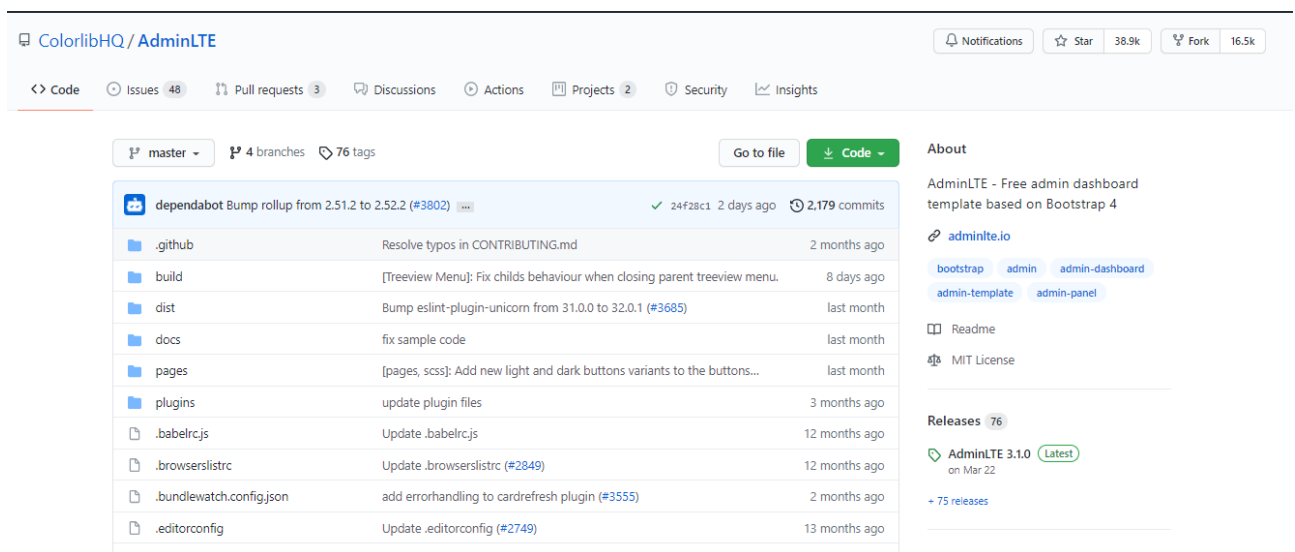


Fig 2.31 GitHub page of the free admin template theme

Like every other web app, there is a need for the customer or user to log in before having access to all the features of its system. The same process is applicable with the SMS. However, in this project, the admin or the HOD is the superuser meaning to say that from the database, he has already signed up, and through him, users like students or staff/teacher can be added in from his dashboard.

The block of authorization (Fig. 2.32) of users is under the categories of goods and performs the function of entering the personal account of the user.

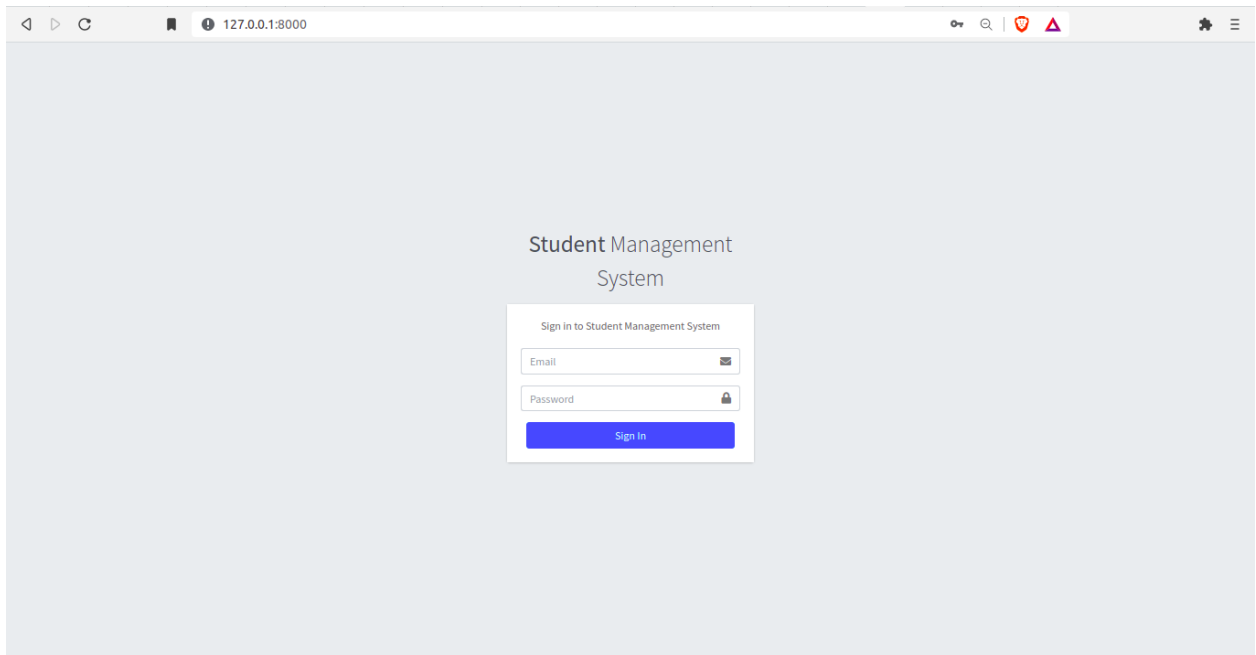


Fig.2.32 Login page

Adding students (Fig2.33) and staffs (Fig.2.34) are of necessity in this project.

The screenshot shows the 'Add Student' form within the 'Student Management System HOD Login' interface. The left sidebar contains a navigation menu with options like Home, Manage Staff, Add Staff, Manage Student, Manage Course, Add Course, Manage Subject, Add Subject, Manage Session, Add Session, View Attendance, Student Feedback, Staff Feedback, Student Leave, and Staff Leave. The 'Add Student' option is highlighted. The main content area features a blue header 'Add Student' and a form with the following fields: Email, Password, First Name, Last Name, Username, Address, Course (a dropdown menu with 'BBA' selected), Gender (a dropdown menu with 'Male' selected), and Session Year.

Fig2.33 Student form from admin board

The screenshot shows the 'Add Staff' form within the 'Student Management System HOD Login' interface. The browser address bar shows '127.0.0.1:8000/add\_staff/'. The left sidebar is identical to the previous screenshot, with 'Add Staff' highlighted. The main content area features a blue header 'Add Staff' and a form with the following fields: Email address (with a placeholder 'Enter email'), Username, Password, First Name, Last Name, and Address. A blue 'Add Staff' button is located at the bottom of the form. The footer contains the text 'Copyright © 2021, Richard Udoh. All rights reserved.' and 'Version 1.0'.

Fig2.33 Staff form from admin board

Furthermore, the admin have the ability to add also courses (Fig 2.34), subjects (Fig 2.35) and sessions (Fig 2.36).

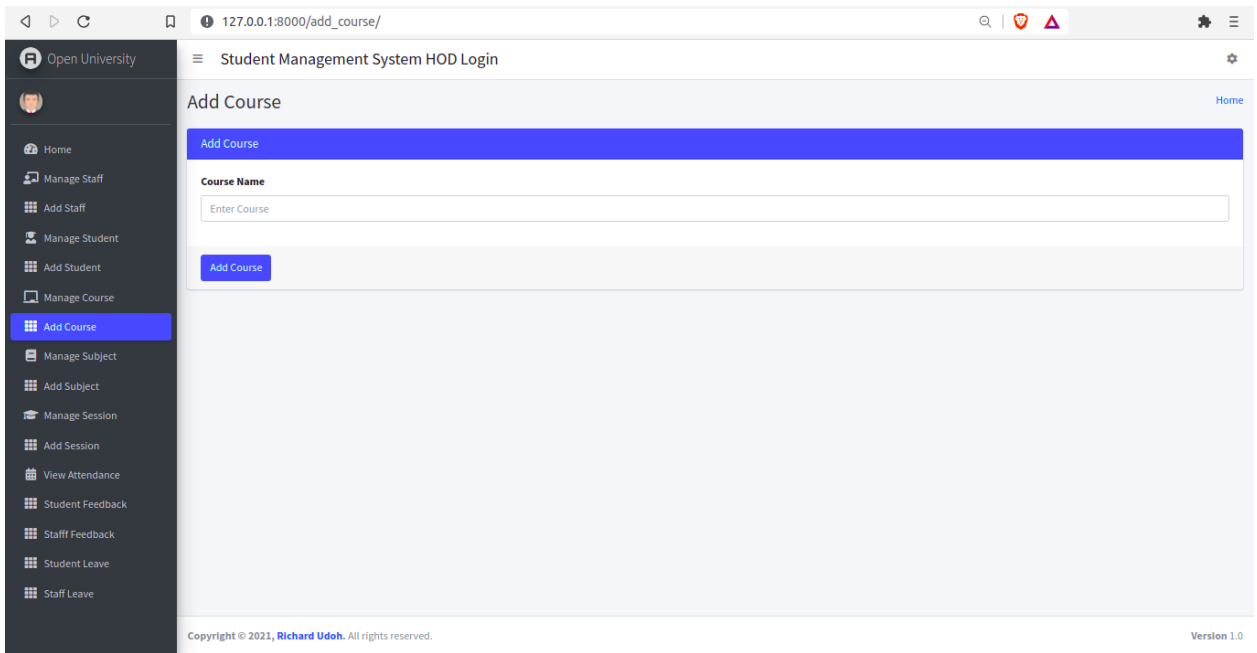


Fig 2.34 Courses page from admin board



Fig.2.35 Add Subjects page from admin board



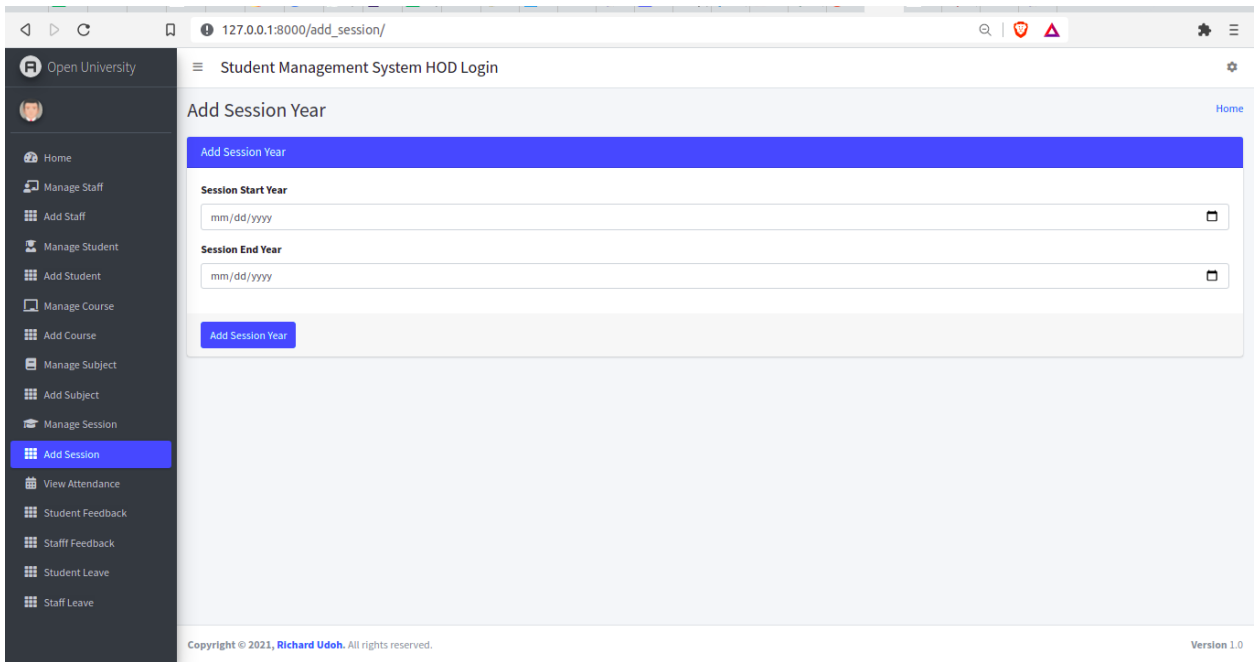


Fig.2.36. Add session page from admin board

The Admin manages the courses, staffs, subjects, sessions and students pages. (Respectively Fig.2.37, Fig.2.38, Fig2.39, Fig.2.40, Fig.2.41)

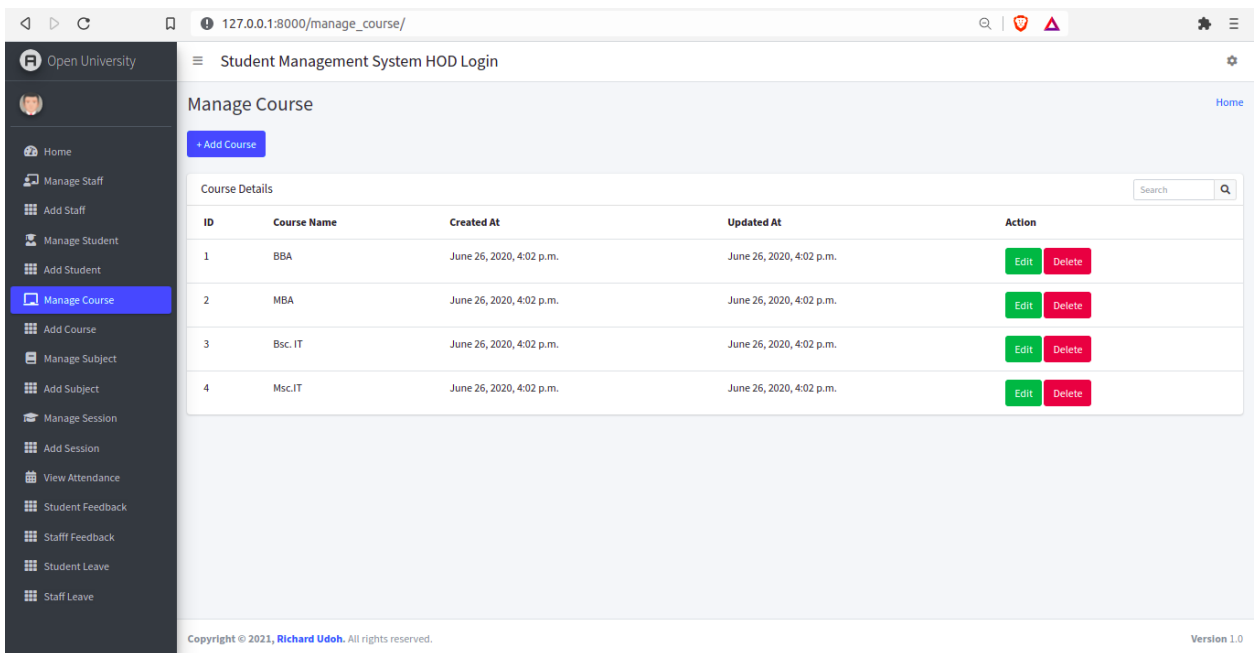


Fig.2.37 Manage course page from admin board

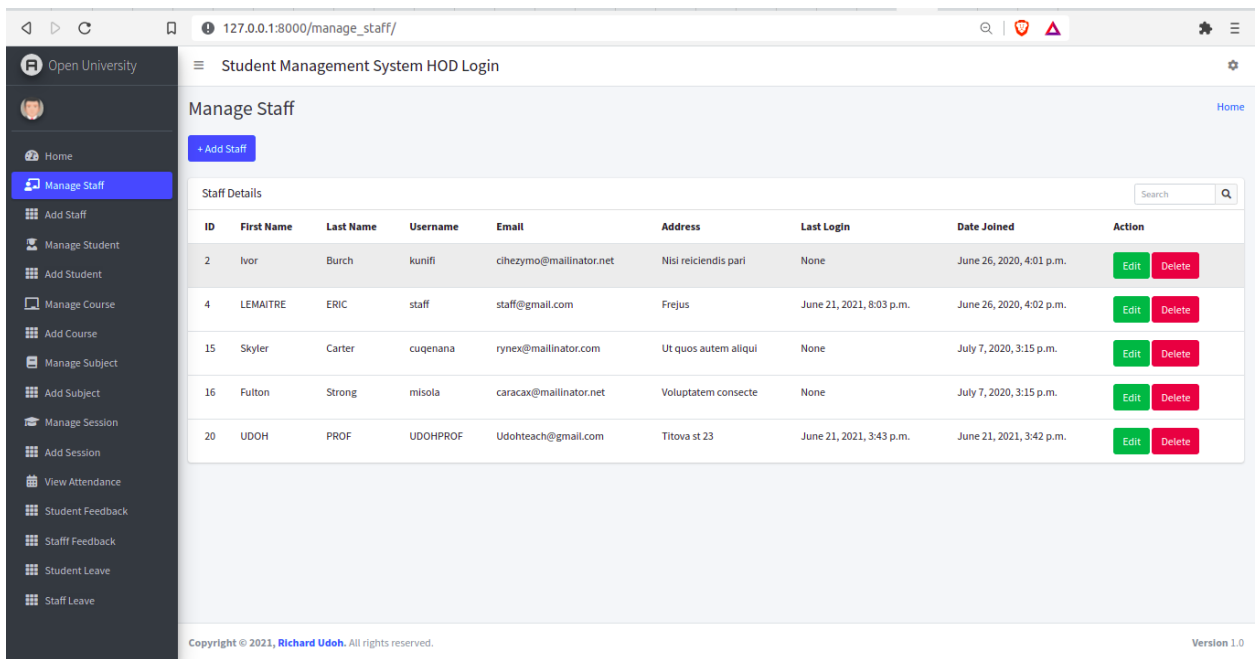


Fig.2.38 Manage staff page from admin board

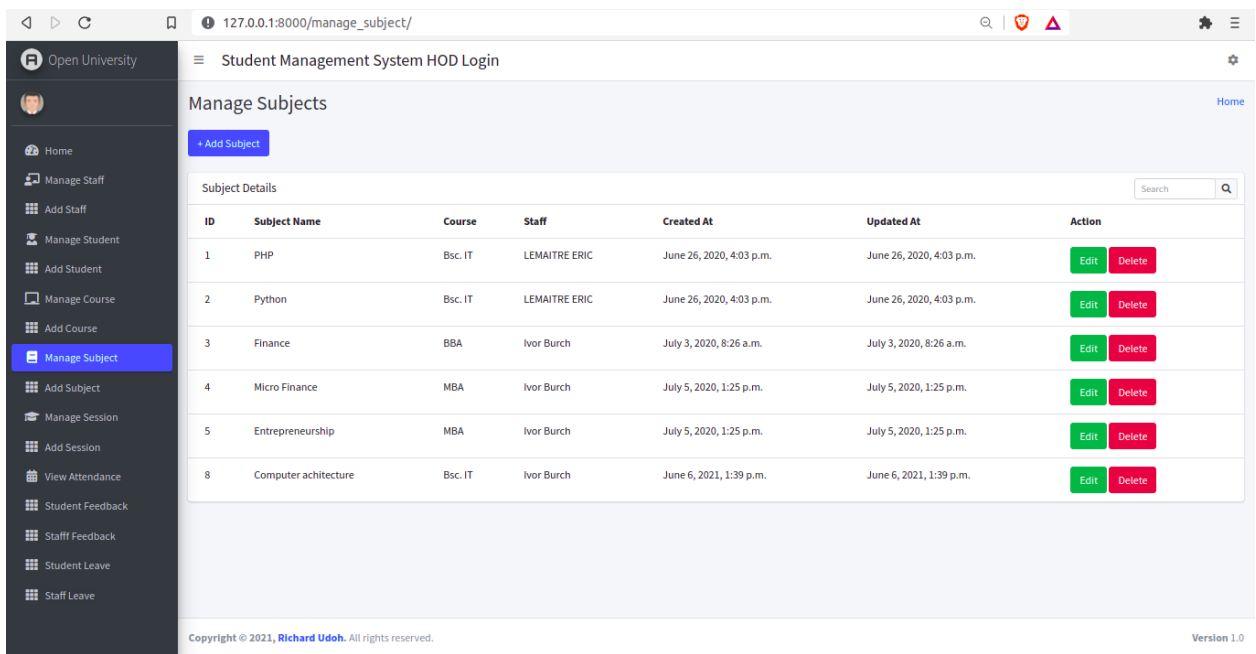


Fig.2.39 Manage subjects page from admin board

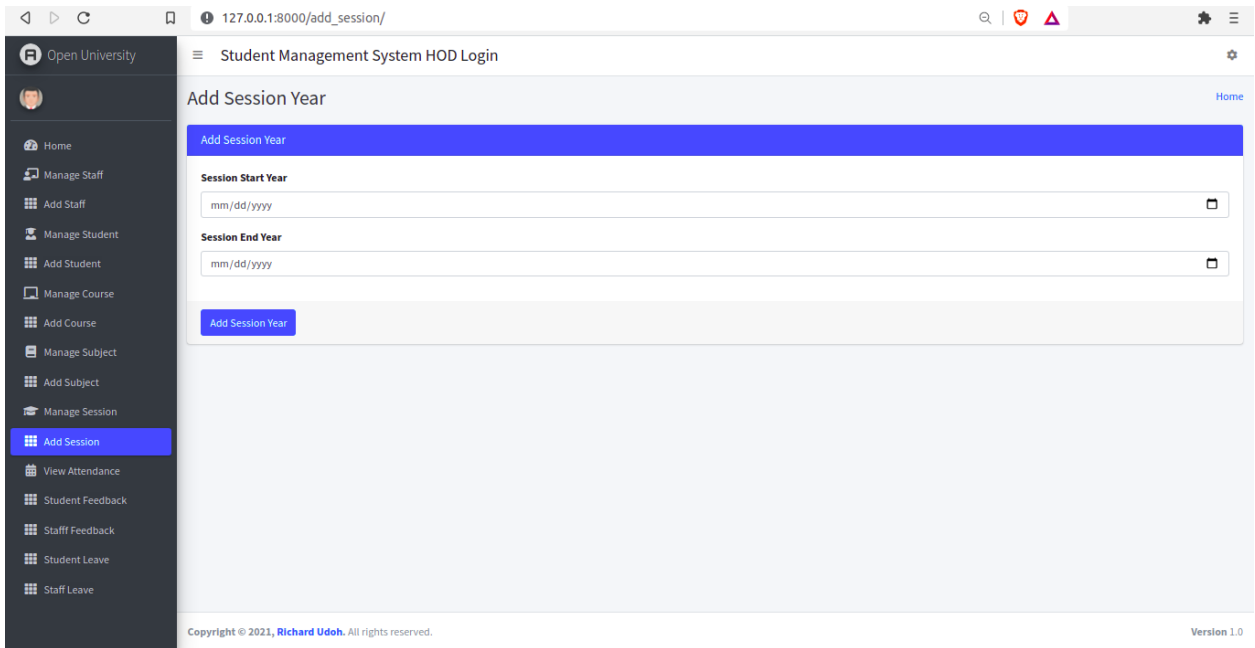


Fig.2.40 Manage session page from admin board

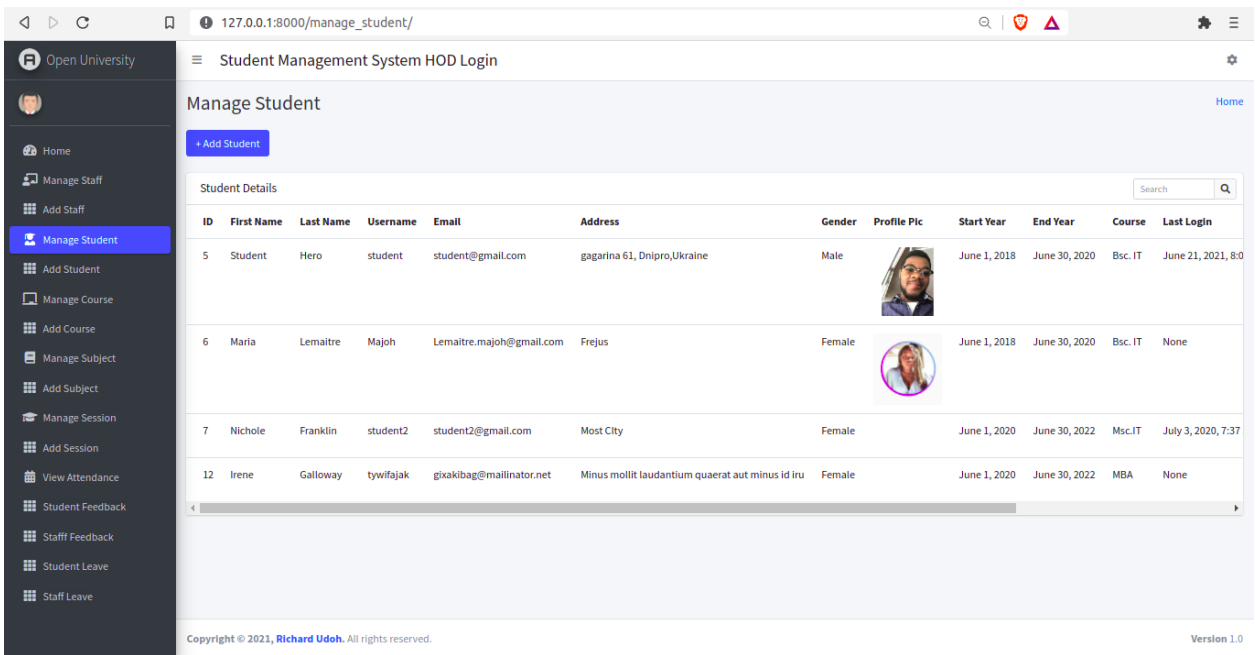


Fig.2.41 Manage student page from admin board

The admin receive and can reply to feedbacks from students (Fig.2.42) and staffs (Fig.43)

Student Management System HOD Login

### Student Feedback

Student ID	Student Name	Student Session	Message	Sended On	Reply
1	Student Hero	June 1, 2018 - June 30, 2020	Please bring some guest lecturers.	July 1, 2020, 6:04 a.m.	We'll try our best.
2	Nichole Franklin	June 1, 2020 - June 30, 2022	Test Feedback from New Student	July 1, 2020, 9:01 a.m.	Tested Successfully. Thanks!
3	Student Hero	June 1, 2018 - June 30, 2020	Online courses should be interactive. And lectures should be recorded and published, so that the students who have missed the live zoom class can take the classes later.	July 8, 2020, 10:53 a.m.	<a href="#">Reply</a>
4	Student Hero	June 1, 2018 - June 30, 2020	Hello Head of department, could please tell teacher Syrotkina to come 2 min before the lessons	June 6, 2021, 1:53 p.m.	Ok, no problem i will speak to her and from today, she will be able to be online 2 min before the lessons
5	Student Hero	June 1, 2018 - June 30, 2020	Je n'ai pas aime le cours d'aujourd'hui, le professeur etait trop rapide, c'etait comme s'il s'enfouait du fait qu'on ne participais pas a son cours.	June 9, 2021, 5:07 p.m.	POuvez vous me donnez le nom du prof comme ca je sais comment gerer avec lui.

Copyright © 2021, Richard Udoh. All rights reserved. Version 1.0

Fig.2.42. Student Feedback page from admin board

Student Management System HOD Login

### Staff Feedback

Staff ID	Staff Name	Message	Sended On	Reply
1	LEMAITRE ERIC	Adding Smart board	June 27, 2020, 8:26 a.m.	We'll consider it in new Session. Thanks!
2	LEMAITRE ERIC	Test Feedback 2	July 1, 2020, 10:35 a.m.	Test Reply.
3	LEMAITRE ERIC	This is Third Feedback.	July 1, 2020, 10:39 a.m.	Okay, I'll Reply here.
4	LEMAITRE ERIC	We need a better and secure Learning Management System. It'll help to deliver educational contents easily and also helps to monitor the progress of students.	July 8, 2020, 10:52 a.m.	<a href="#">Reply</a>

Copyright © 2021, Richard Udoh. All rights reserved. Version 1.0

Fig.2.43. Staff Feedback page from admin board

In addition to the role of management, the admin can view attendances (Fig.2.44), approve or refuse leave from student (Fig.2.45) and staff (Fig.2.46).

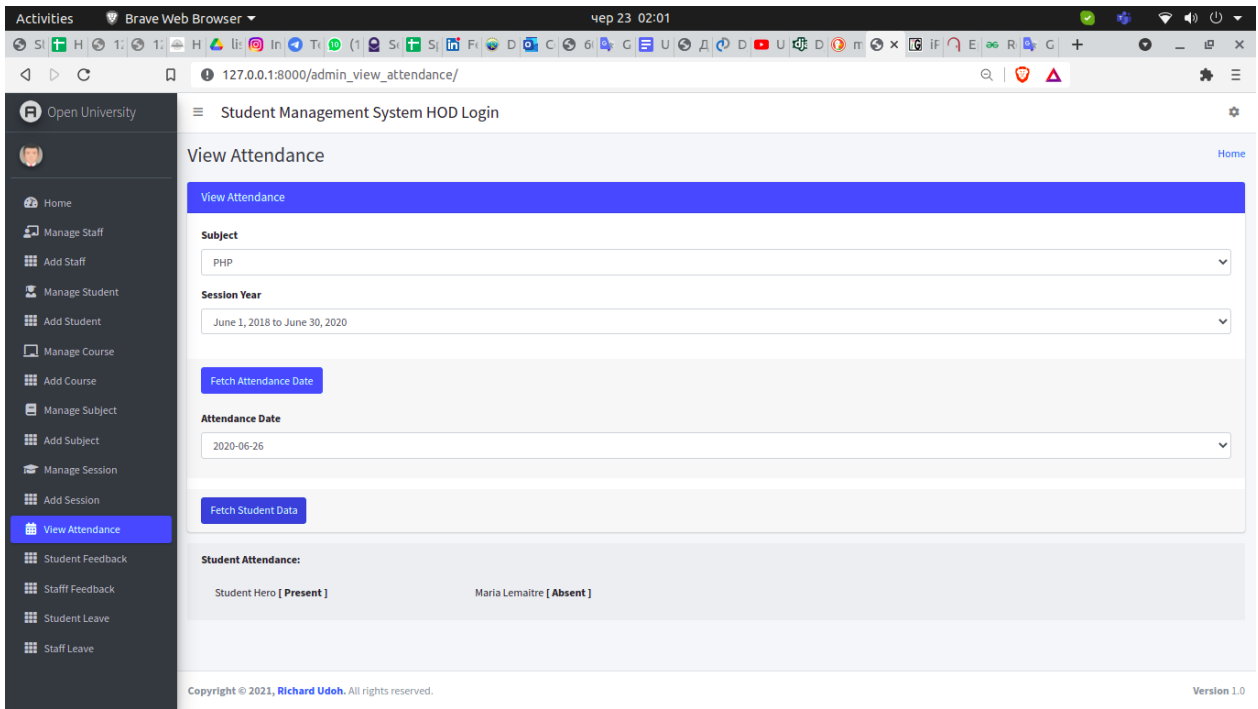


Fig.2.43 View attendance page from admin board

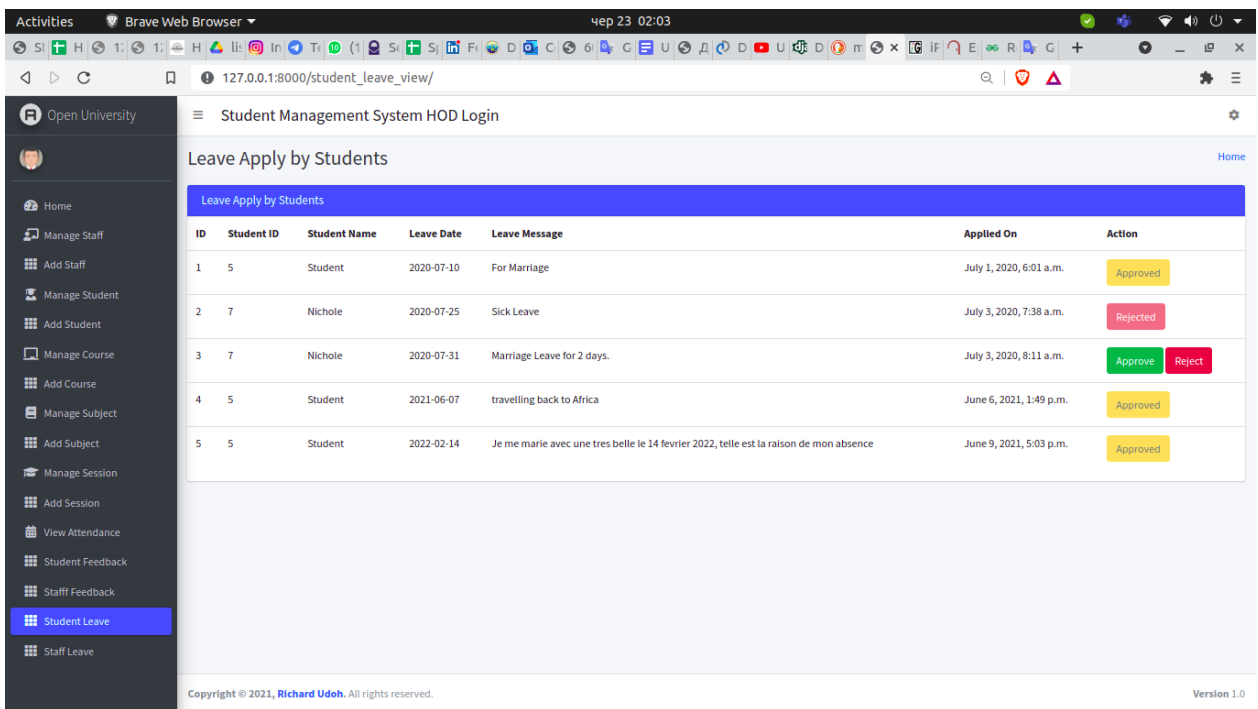


Fig.2.45 Application of leave by students from the admin board

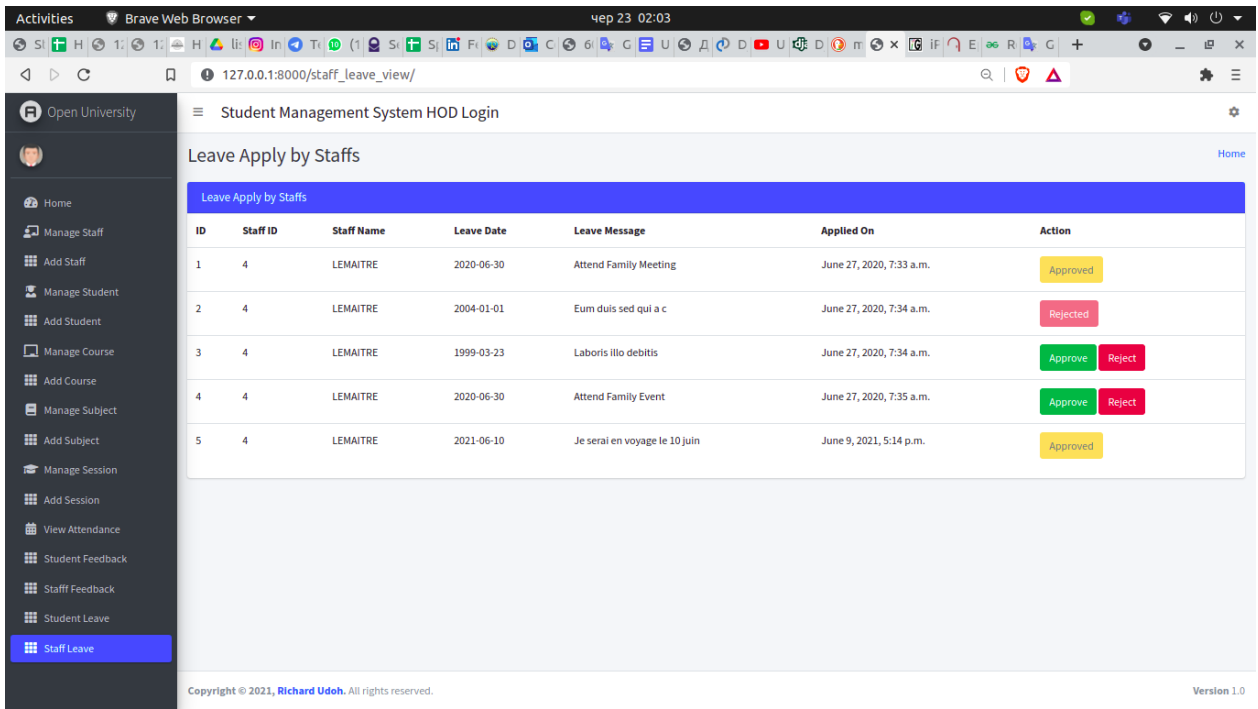


Fig.2.45 Leave apply by staffs from the admin board

The Students can see their results (Fig.2.46), apply for leave (Fig.2.47) and send a feedback message to the HOD (Fig.2.48).

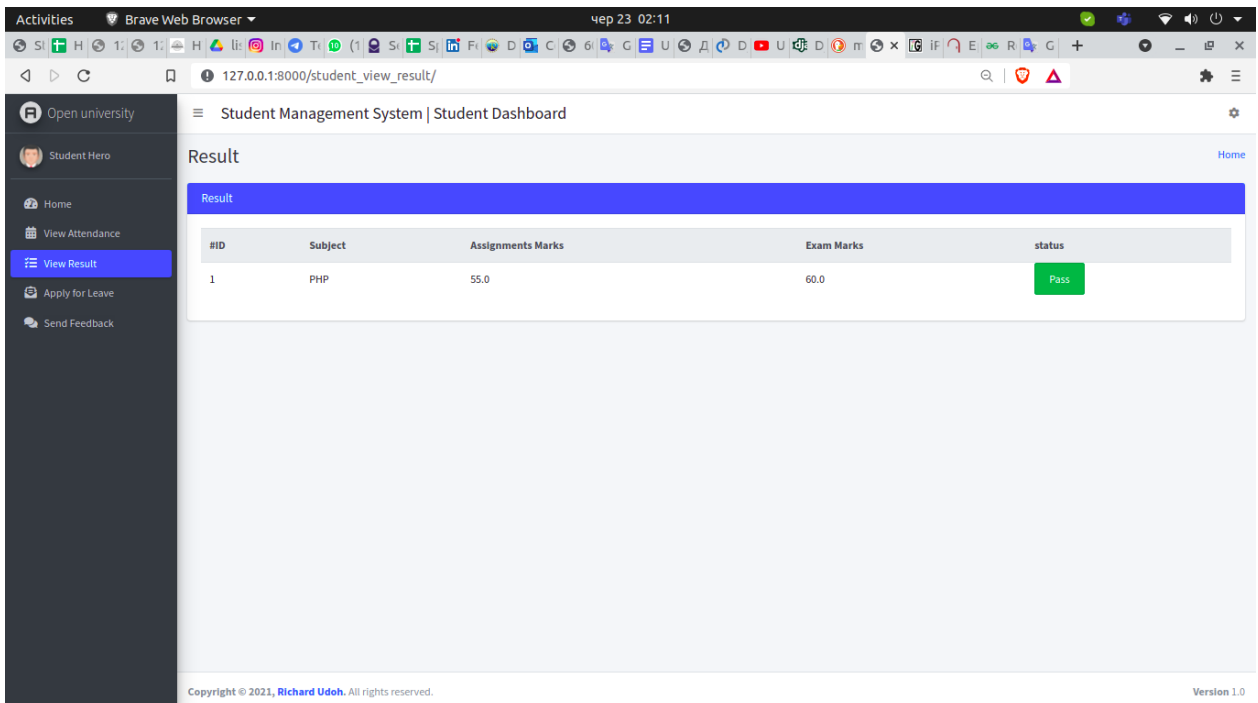


Fig.2.46 Result of Exams from student board

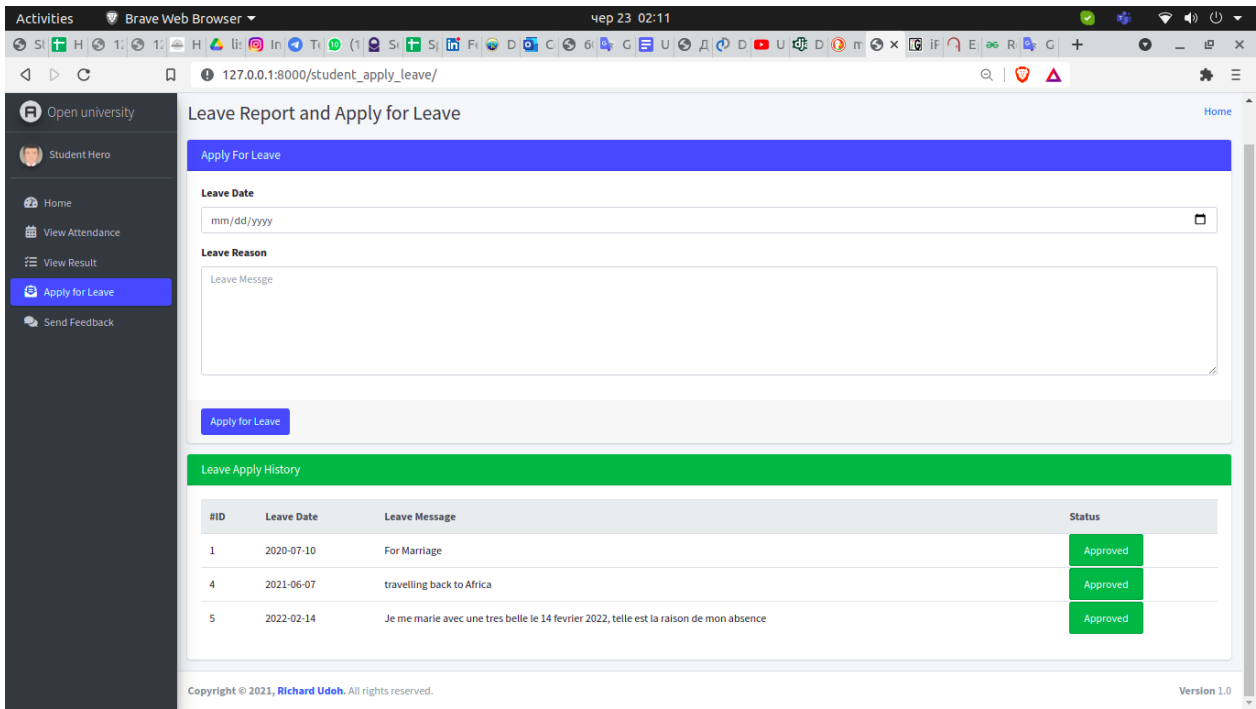


Fig.2.47. Leave Report and leave application from student board

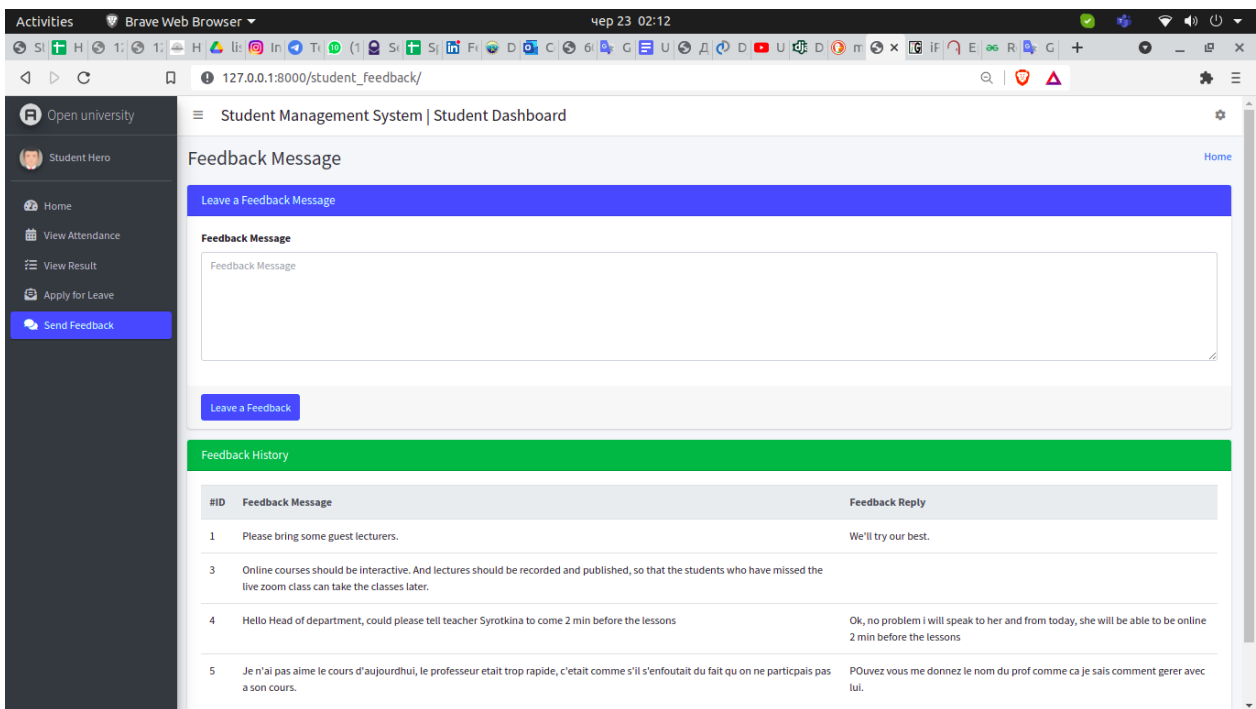


Fig.2.48. Feedback message from student board

The Staffs can View and update attendances (Fig2.49), add students results (Fig.2.50), apply for leave (Fig.2.51) and send a feedback message to the HOD (Fig.2.52).

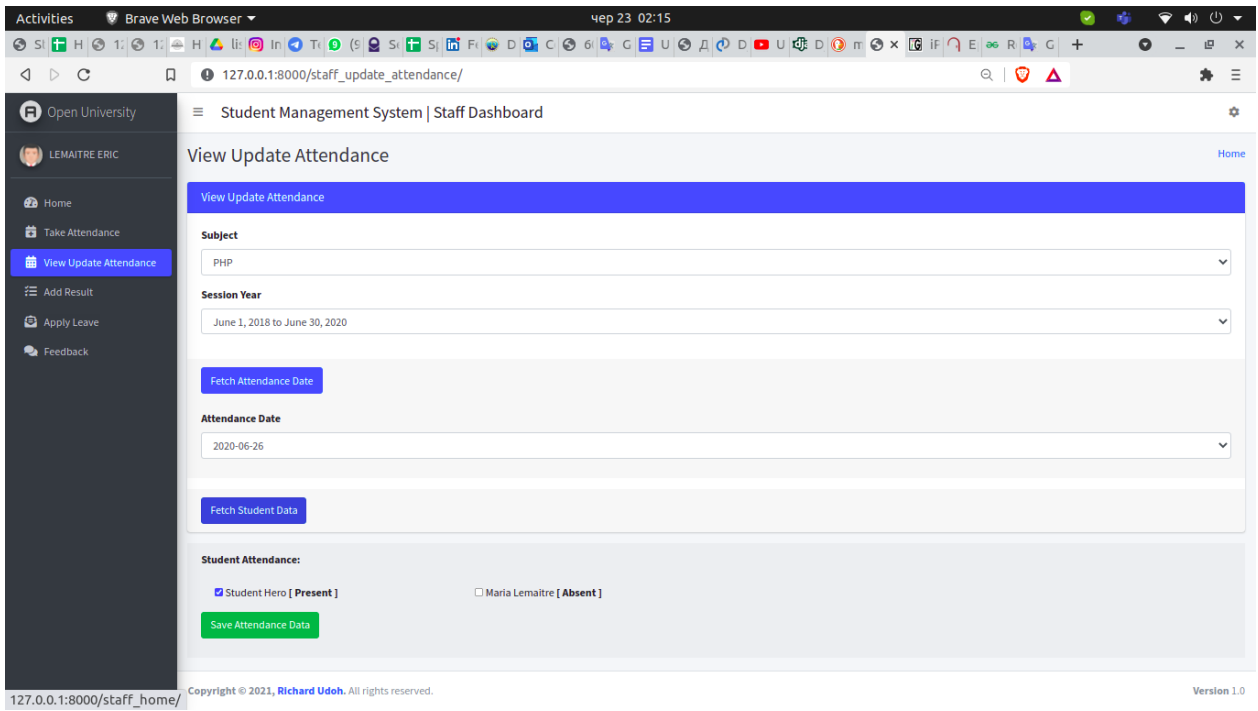


Fig.2.49 View update attendance from staff board

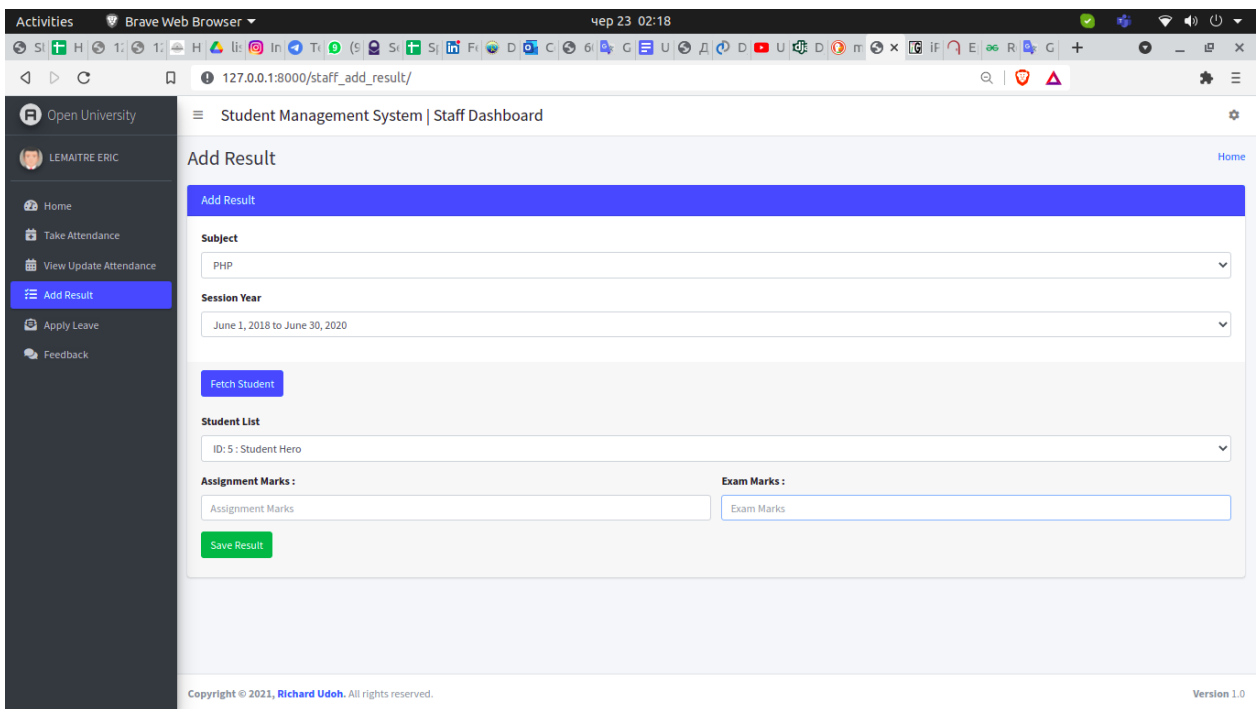


Fig.2.50 Adding Result from Staff Board



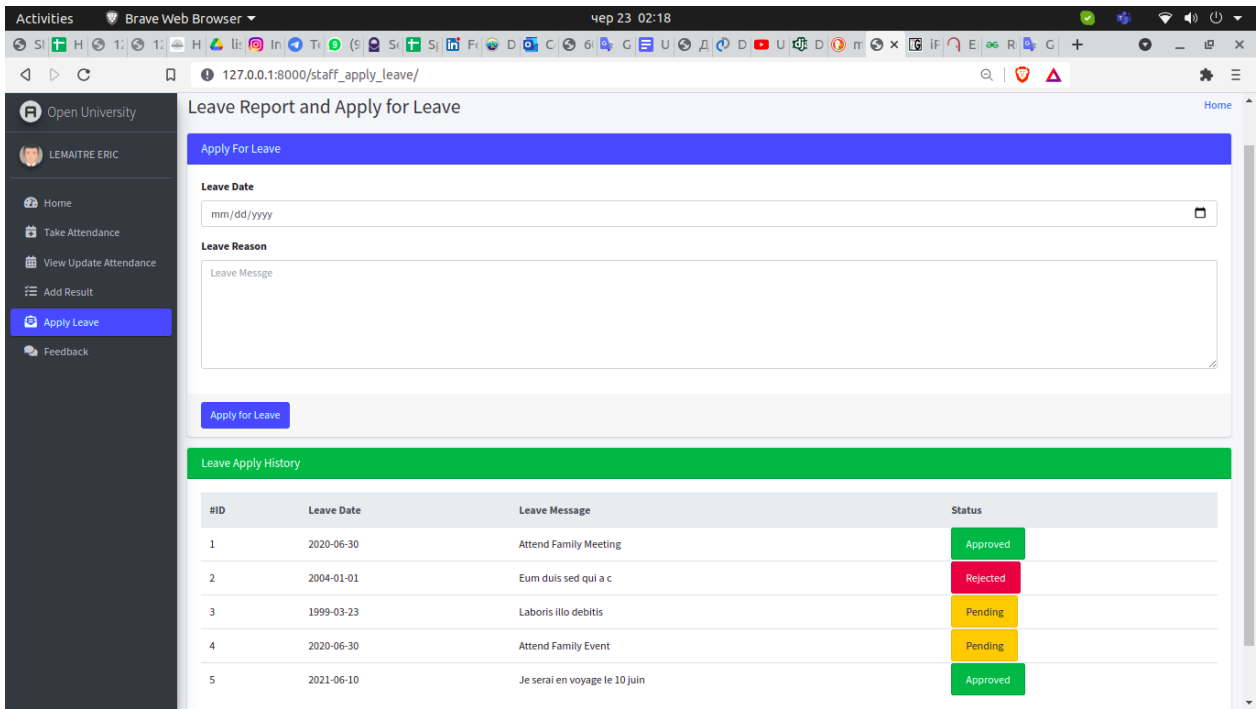


Fig.2.51 Leave Report and leave application from staff board.

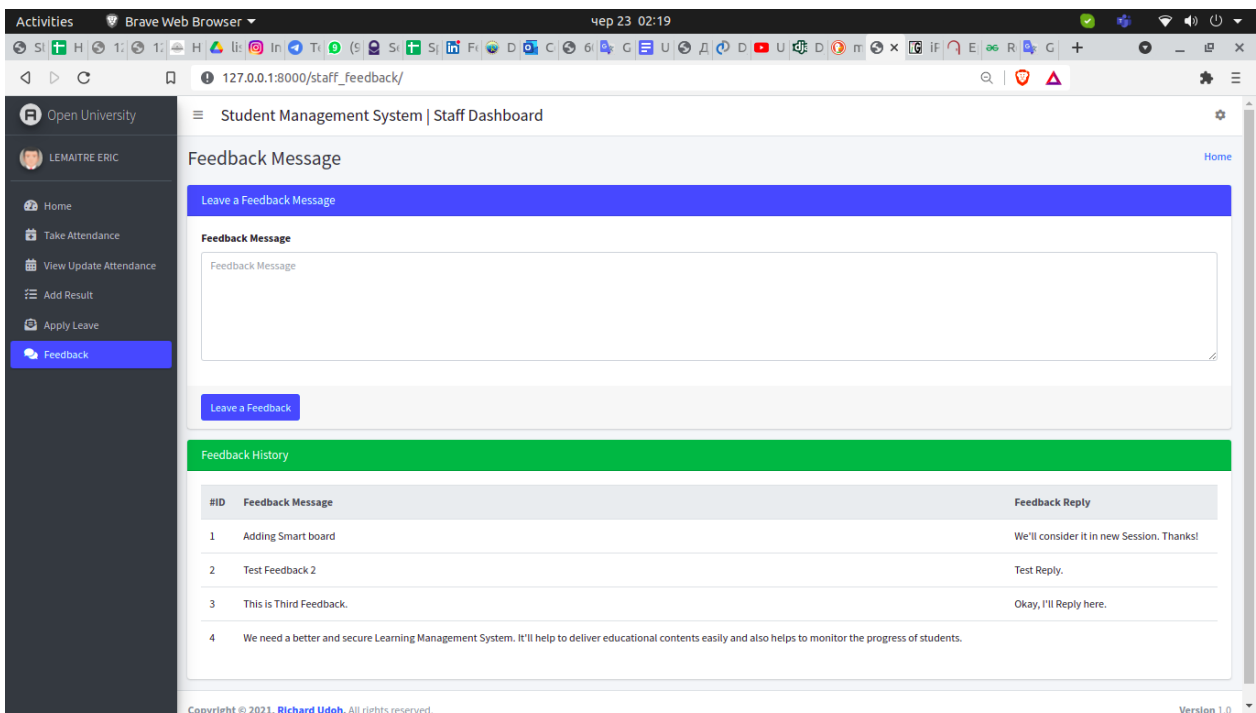
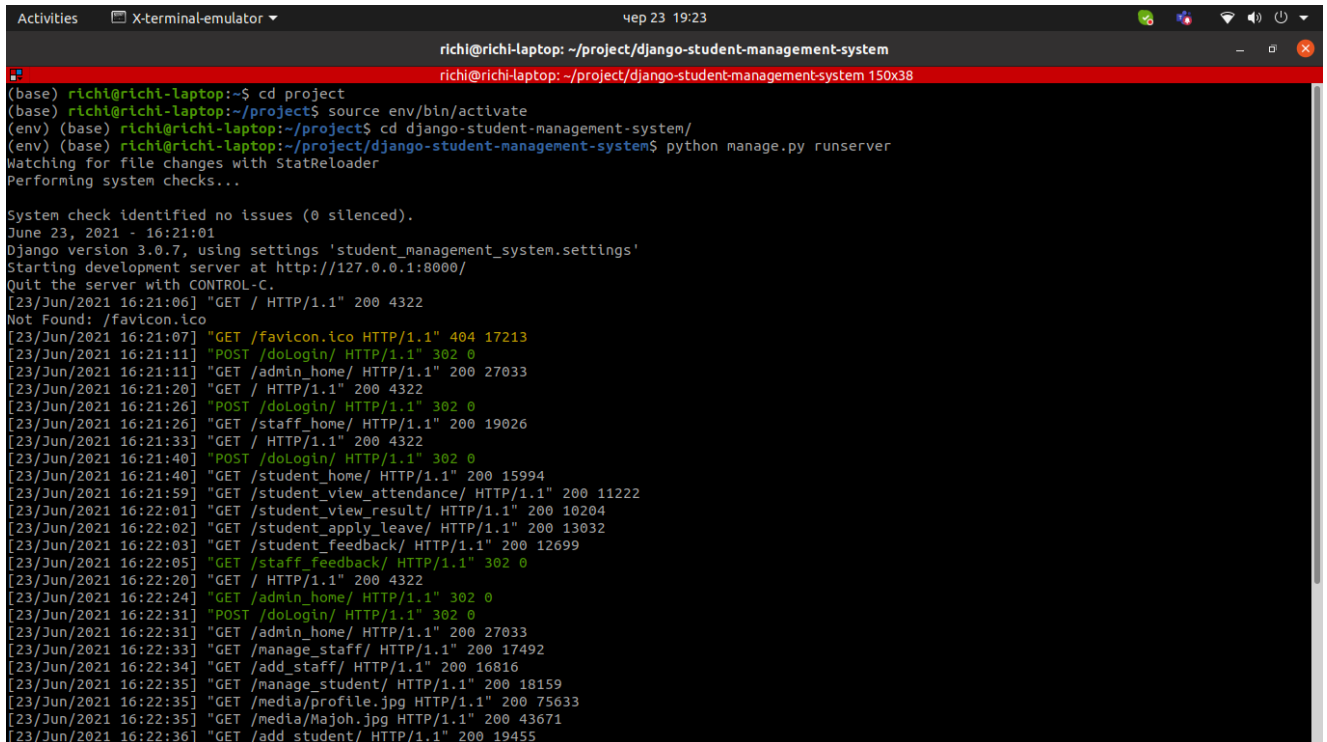


Fig.2.52. Feedback message from staff board

It is important to mention that the whole project is running on a local server. Therefore the Ubuntu terminal (Fig.2.53) helped the installation of the virtual

environment, which provides the necessary tools for developing web applications with Django.



```
richi@richi-laptop: ~/project/django-student-management-system
richi@richi-laptop: ~/project/django-student-management-system 150x38
(base) richi@richi-laptop:~$ cd project
(base) richi@richi-laptop:~/project$ source env/bin/activate
(env) (base) richi@richi-laptop:~/project$ cd django-student-management-system/
(env) (base) richi@richi-laptop:~/project/django-student-management-system$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 23, 2021 - 16:21:01
Django version 3.0.7, using settings 'student_management_system.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[23/Jun/2021 16:21:06] "GET / HTTP/1.1" 200 4322
Not Found: /favicon.ico
[23/Jun/2021 16:21:07] "GET /favicon.ico HTTP/1.1" 404 17213
[23/Jun/2021 16:21:11] "POST /doLogin/ HTTP/1.1" 302 0
[23/Jun/2021 16:21:11] "GET /admin_home/ HTTP/1.1" 200 27033
[23/Jun/2021 16:21:20] "GET / HTTP/1.1" 200 4322
[23/Jun/2021 16:21:26] "POST /doLogin/ HTTP/1.1" 302 0
[23/Jun/2021 16:21:26] "GET /staff_home/ HTTP/1.1" 200 19026
[23/Jun/2021 16:21:33] "GET / HTTP/1.1" 200 4322
[23/Jun/2021 16:21:40] "POST /doLogin/ HTTP/1.1" 302 0
[23/Jun/2021 16:21:40] "GET /student_home/ HTTP/1.1" 200 15994
[23/Jun/2021 16:21:59] "GET /student_view_attendance/ HTTP/1.1" 200 11222
[23/Jun/2021 16:22:01] "GET /student_view_result/ HTTP/1.1" 200 10204
[23/Jun/2021 16:22:02] "GET /student_apply_leave/ HTTP/1.1" 200 13032
[23/Jun/2021 16:22:03] "GET /student_feedback/ HTTP/1.1" 200 12699
[23/Jun/2021 16:22:05] "GET /staff_feedback/ HTTP/1.1" 302 0
[23/Jun/2021 16:22:20] "GET / HTTP/1.1" 200 4322
[23/Jun/2021 16:22:24] "GET /admin_home/ HTTP/1.1" 302 0
[23/Jun/2021 16:22:31] "POST /doLogin/ HTTP/1.1" 302 0
[23/Jun/2021 16:22:31] "GET /admin_home/ HTTP/1.1" 200 27033
[23/Jun/2021 16:22:33] "GET /manage_staff/ HTTP/1.1" 200 17492
[23/Jun/2021 16:22:34] "GET /add_staff/ HTTP/1.1" 200 16816
[23/Jun/2021 16:22:35] "GET /manage_student/ HTTP/1.1" 200 18159
[23/Jun/2021 16:22:35] "GET /media/profile.jpg HTTP/1.1" 200 75633
[23/Jun/2021 16:22:35] "GET /media/Majoh.jpg HTTP/1.1" 200 43671
[23/Jun/2021 16:22:36] "GET /add_student/ HTTP/1.1" 200 19455
```

Fig.2.53. Ubuntu terminal

## РОЗДІЛ 3

### ЕКОНОМІЧНИЙ РОЗДІЛ

#### 3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

- 1) передбачуване число операторів програми – 2058;
- 2) коефіцієнт складності програми – 1,2;
- 3) коефіцієнт корекції програми в ході її розробки – 0,06;
- 4) годинна заробітна плата програміста – 125 грн/год;
- 5) коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
- 6) коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
- 7) вартість машино-години ЕОМ – 14 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{oml} + t_\partial, \text{ людино-годин,} \quad (3.1)$$

де  $t_o$  – витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_u$  – витрати праці на дослідження алгоритму рішення задачі;

$t_a$  – витрати праці на розробку блок-схеми алгоритму;

$t_n$  – витрати праці на програмування по готовій блок-схемі;

$t_{oml}$  – витрати праці на налагодження програми на ЕОМ;

$t_d$  – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \text{ де} \quad (3.2)$$

$q$  – передбачуване число операторів;

$C$  – коефіцієнт складності програми;

$p$  – коефіцієнт корекції програми в ході її розробки.

$$Q = 2058 \cdot 1,3 \cdot (1 + 0,06) = 2835,92;$$

Витрати праці на вивчення опису задачі  $t_u$  визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75 \dots 85) \cdot K}, \text{ людино-годин,} \quad (3.3)$$

де  $B$  – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

$K$  – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$$t_u = \frac{2835,92 \cdot 1,2}{80 \cdot 1,2} = 35,44, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot K}; \quad (3.4)$$

$$t_a = \frac{2835,92}{24 \cdot 1,2} = 98,46, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) \cdot K}; \quad (3.5)$$

$$t_n = \frac{2835,92}{25 \cdot 1,2} = 94,53, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5) \cdot K}; \quad (3.6)$$

$$t_{отл} = \frac{2835,92}{5 \cdot 1,2} = 472,65, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання:

$$t_{отл}^k = 1,2 \cdot t_{отл}; \quad (3.7)$$

$$t_{отл}^k = 1,2 \cdot 472,65 = 567,18, \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_{\partial} = t_{\partial p} + t_{\partial o}; \quad (3.8)$$

де  $t_{\partial p}$  – трудомісткість підготовки матеріалів і рукопису

$$t_{\partial p} = \frac{q}{(15...20) \cdot K}; \quad (3.9)$$

$$t_{\partial p} = \frac{2835,92}{17 \cdot 1,2} = 139,01, \text{ людино-годин.}$$

$t_{\partial o}$  – трудомісткість редагування, печатки й оформлення документації

$$t_{\partial o} = 0,75 \cdot t_{\partial p}; \quad (3.10)$$

$$t_{\partial o} = 0,75 \cdot 139,01 = 104,25, \text{ людино-годин.}$$

$$t_{\partial} = 139,01 + 104,25 = 243,26, \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 35,44 + 98,46 + 94,53 + 427,65 + 243,26 = 949,34, \text{ людино-годин.}$$

У результаті розрахував, що в загальній складності необхідно 949,34 людино-годин для розробки даного програмного забезпечення.

### **3.2. Розрахунок витрат на створення програми**

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{по} = Z_{зп} + Z_{мв}, \text{ грн,} \quad (3.11)$$

де  $Z_{ЗП}$  – заробітна плата виконавців, яка визначається за формулою:

$$Z_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.12)$$

де  $t$  – загальна трудомісткість, людино-годин;

$C_{ПР}$  – середня годинна заробітна плата програміста, грн/година

$$Z_{ЗП} = 949,34 \cdot 125 = 118667,5 \text{ грн.}$$

$Z_{МВ}$  – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$Z_{МВ} = t_{омл} \cdot C_{М}, \text{ грн,} \quad (3.13)$$

де  $t_{омл}$  – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$  – вартість машино-години ЕОМ, грн/год.

$$Z_{МВ} = 949,34 \cdot 14 = 13290,75 \text{ грн.}$$

$$K_{ПО} = 118667,5 + 13290,75 = 131958,25 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \quad (3.14)$$

де  $B_k$  – число виконавців;

$F_p$  – місячний фонд робочого часу (при 40 годинному робочому тижні  $F_p=176$  годин).

$$T = \frac{949,34}{1 \cdot 176} = 5,3 \text{ міс.}$$

**Висновки.** На розробку даного програмного забезпечення піде 949,34 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 5,3 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 131958,25 грн.



## CONCLUSIONS

In this qualification thesis, a web app was developed to manage student's data different that the archaic manner.

This software is designed to reduce the amount of paper work. It valorizes the importance of storing students' data or records online than keeping it physically on papers that can be burned or misplaced. The Student Management System is of a great necessity in our modern days.

As a result of web app development, the following features were implemented: admin registration, student registration, staff registration in the system, authorization and authentication. The three different group of users have in common the ability to add / edit / receive / delete/ and send. The Admin manages the courses, subjects, sessions, staffs and students. He approves requests and can receive feedbacks from the staffs and students. The teacher marks attendances, add results of any exams, send feedbacks and can apply for a leave. The student can view his attendance; choose courses, subjects and sessions. The student can send feedbacks and apply for a leave.

During the implementation of this qualification thesis the following tasks were performed:

- the subject area of the problem is analyzed;
- a comparison was made with the capabilities of existing similar services;
- the rational structure of the database is chosen;
- the program code of the web app is written;
- the user interface on different board with display of graphics and charts in real time.
- Further development of the project is analyzed.

The web application is implemented in Python programming language using the Django Framework. The database used is SQLite database integrated in Django. The

client part was developed using the JavaScript, bootstrap, HTML and CSS programming language using the Django way to implement the code. Databases and files are stored locally on the laptop used to write the whole project.

Also in the qualification thesis, an economic study was made to determine the complexity of the developed software product. The study shows that nine hundred forty-nine point thirty four hours (949.34 person-hours) were dedicated to the realization of the project. The calculation shows that the cost of work to create a program like the SMS is about one hundred thirty-one thousand nine hundred fifty-eight point two five hryvnia (131958,25 UAH). The calculation indicates that the time to create it is five point 3 months (5.3 months).

## REFERENCES

1. Mark Lutz, Learning Python 5<sup>TH</sup> Edition, July 2013.
2. Nigele Gorge, Django Framework: Build a website with Django 3, June 12, 2020.
3. Jogn Dean, Web programming with HTML5, CSS, and JavaScript. O'Reilly Media, 2019. 678 c.
4. Tal Ater, Building Progressive Web Apps: Bringing the Power of Native to the Browser, 2017.
5. James Bennett, Practical Django Projects, July 01, 2009.
6. Bryan Sullivan, Web Application Security, A Beginner's Guide 1<sup>st</sup> Edition.
7. Learn JavaScript Quickly by Code Quickly, November 10, 2020.
8. Jon Duckett, Web Design with HTML, CSS, JavaScript and jQuery Set 1st Edition.
9. Harry Percival, Architecture Patterns with Python: Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven Micro services, Mar 31, 2020.
10. Arun Ravindran, Django Design Patterns and Best Practices, Mar 31, 2015.
11. Harold Kerzner, Project Management: A Systems Approach to Planning, Scheduling, and Controlling 12th Edition.
12. John Rezig. JavaScript for professionals. -M.: "Williams", 2016, 242 s.
13. Alessandro Del Sole, Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux 1st ed. Edition, November 30, 2018.
14. Philip Kirkbride, Basic Linux Terminal Tips and Tricks: Learn to Work Quickly on the Command Line 1st ed. Edition, August 5, 2020.
15. Andrew Hoffman, Web Application Security: Exploitation and Countermeasures for Modern Web Applications, Mar 24, 2020.
16. Jay A. Kreibich, Using SQLite: Small. Fast. Reliable. Choose Any Three, Jul 6, 2012
17. David Alastair Hayden, The Forbidden Library (Storm Phase) (Volume 3), May 14, 2014

18. Full Stack Python Security: Cryptography, TLS, and attack resistance

By Dennis Byrne, Aug 24, 2011.

19. Learn Server-side Django

URL: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django>

20. Explanation of Django structure

URL: <https://djangobook.com/mdj2-django-structure/>

21. Using Django authentication system.

URL: <https://docs.djangoproject.com/en/3.2/topics/auth/default/>

22. Best Student Management system.

URL: <https://www.softwaresuggest.com/student-management-system>

## SOURCE CODE

**base.html**

```

{% load static % }
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Student Management System | Dashboard</title>
  <!-- Tell the browser to be responsive to screen width -->
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <!-- Font Awesome -->
  <link rel="stylesheet" href="{% static 'fontawesome-free/css/all.min.css' %}">
  <!-- Ionicons -->
  <link rel="stylesheet" href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
  <!-- Tempusdominus Bbootstrap 4 -->
  <link rel="stylesheet" href="{% static 'tempusdominus-bootstrap-4/css/tempusdominus-bootstrap-4.min.css'
%}">
  <!-- iCheck -->
  <link rel="stylesheet" href="{% static 'icheck-bootstrap/icheck-bootstrap.min.css' %}">
  <!-- JQVMap -->
  <link rel="stylesheet" href="{% static 'jqvmap/jqvmap.min.css' %}">
  <!-- Theme style -->
  <link rel="stylesheet" href="{% static 'dist/css/adminlte.min.css' %}">
  <!-- overlayScrollbars -->
  <link rel="stylesheet" href="{% static 'overlayScrollbars/css/OverlayScrollbars.min.css' %}">
  <!-- Daterange picker -->
  <link rel="stylesheet" href="{% static 'daterangepicker/daterangepicker.css' %}">
  <!-- summernote -->
  <link rel="stylesheet" href="{% static 'summernote/summernote-bs4.css' %}">
  <!-- Google Font: Source Sans Pro -->
  <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700" rel="stylesheet">

</head>

```

{% block content %}

{% endblock %}

<!-- jQuery -->

<script src="{% static "jquery/jquery.min.js" %}"></script>

<!-- jQuery UI 1.11.4 -->

<script src="{% static "jquery-ui/jquery-ui.min.js" %}"></script>

<!-- Resolve conflict in jQuery UI tooltip with Bootstrap tooltip -->

<script>

\$.widget.bridge('uibutton', \$.ui.button)

</script>

<!-- Bootstrap 4 -->

<script src="{% static "bootstrap/js/bootstrap.bundle.min.js" %}"></script>

<!-- ChartJS -->

<script src="{% static "chart.js/Chart.min.js" %}"></script>

<!-- Sparkline -->

<script src="{% static "sparklines/sparkline.js" %}"></script>

<!-- JQVMap -->

<script src="{% static "jqvmap/jquery.vmap.min.js" %}"></script>

<script src="{% static "jqvmap/maps/jquery.vmap.usa.js" %}"></script>

<!-- jQuery Knob Chart -->

<script src="{% static "jquery-knob/jquery.knob.min.js" %}"></script>

<!-- daterangepicker -->

<script src="{% static "moment/moment.min.js" %}"></script>

<script src="{% static "daterangepicker/daterangepicker.js" %}"></script>

<!-- Tempusdominus Bootstrap 4 -->

<script src="{% static "tempusdominus-bootstrap-4/js/tempusdominus-bootstrap-4.min.js" %}"></script>

<!-- Summernote -->

<script src="{% static "summernote/summernote-bs4.min.js" %}"></script>

<!-- overlayScrollbars -->

<script src="{% static "overlayScrollbars/js/jquery.overlayScrollbars.min.js" %}"></script>

<!-- AdminLTE App -->

<script src="{% static 'dist/js/adminlte.js' %}"></script>

<!-- AdminLTE dashboard demo (This is only for demo purposes) -->

<script src="{% static 'dist/js/pages/dashboard.js' %}"></script>

<!-- AdminLTE for demo purposes -->

<script src="{% static 'dist/js/demo.js' %}"></script>

```
</body>
```

```
</html>
```

## Login.html

```
{% extends 'base.html' %}
```

```
{% load static %}
```

```
{% block content %}
```

```
    <body class="hold-transition login-page">
```

```
<div class="login-box">
```

```
    <div class="login-logo">
```

```
        <a href="{% url 'login' %}"><b>Student </b>Management System</a>
```

```
    </div>
```

```
    <!-- /.login-logo -->
```

```
    <div class="card">
```

```
        <div class="card-body login-card-body">
```

```
            <p class="login-box-msg">Sign in to Student Management System</p>
```

```
        <form action="{% url 'doLogin' %}" method="POST">
```

```
            {% csrf_token %}
```

```
            <div class="input-group mb-3">
```

```
                <input type="email" class="form-control" placeholder="Email" name="email">
```

```
                <div class="input-group-append">
```

```
                    <div class="input-group-text">
```

```
                        <span class="fas fa-envelope"></span>
```

```
                    </div>
```

```
                </div>
```

```
            </div>
```

```
            <div class="input-group mb-3">
```

```
                <input type="password" class="form-control" placeholder="Password" name="password">
```

```
                <div class="input-group-append">
```

```
                    <div class="input-group-text">
```

```
                        <span class="fas fa-lock"></span>
```

```
                    </div>
```

```
                </div>
```

```
            </div>
```

```
        <div class="col-12">
```

```
            <button type="submit" class="btn btn-primary btn-block">Sign In</button>
```

```
        </div>
```

```

{% comment %} <div class="col-12 text-center">
  <a href="{% url 'password_reset' %}">Reset Password</a>
</div> {% endcomment %}

{% comment %} Display Messages {% endcomment %}
{% if messages %}
<div class="col-12">
  {% for message in messages %}
    {% if message.tags == "error" %}
      <div class="alert alert-danger alert-dismissible fade show" role="alert" style="margin-top: 10px;">
        {{ message }}
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
    {% endif %}
  {% endfor %}
</div>
{% endif %}

<!-- /.col -->
</div>
</form>

</div>
</div>
<!-- /.login-box -->

{% endblock %}

```

## base\_template.html

```

{% load static %}

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Student Management System | Dashboard</title>
  <!-- Tell the browser to be responsive to screen width -->

```



```

<meta name="viewport" content="width=device-width, initial-scale=1">
<!-- Font Awesome -->
<link rel="stylesheet" href="{% static 'fontawesome-free/css/all.min.css' %}">
<!-- Ionicons -->
<link rel="stylesheet" href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
<!-- Tempusdominus Bbootstrap 4 -->
<link rel="stylesheet" href="{% static 'tempusdominus-bootstrap-4/css/tempusdominus-bootstrap-4.min.css'
%}">
<!-- iCheck -->
<link rel="stylesheet" href="{% static 'icheck-bootstrap/icheck-bootstrap.min.css' %}">
<!-- JQVMap -->
<link rel="stylesheet" href="{% static 'jqvmap/jqvmap.min.css' %}">
<!-- Theme style -->
<link rel="stylesheet" href="{% static 'dist/css/adminlte.min.css' %}">
<!-- overlayScrollbars -->
<link rel="stylesheet" href="{% static 'overlayScrollbars/css/OverlayScrollbars.min.css' %}">
<!-- Daterange picker -->
<link rel="stylesheet" href="{% static 'daterangepicker/daterangepicker.css' %}">
<!-- summernote -->
<link rel="stylesheet" href="{% static 'summernote/summernote-bs4.css' %}">
<!-- Google Font: Source Sans Pro -->
<link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700" rel="stylesheet">

{% comment %} For Custom CSS {% endcomment %}

{% block custom_css %}
{% endblock custom_css %}

</head>

<body class="hold-transition sidebar-mini layout-fixed">
<div class="wrapper">

<!-- Navbar -->
<nav class="main-header navbar navbar-expand navbar-white navbar-light">
<!-- Left navbar links -->
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link" data-widget="pushmenu" href="#" role="button"><i class="fas fa-bars"></i></a>
</li>
</ul>

```

```

<h4 style="margin-left: 10px; margin-top: 5px;">Student Management System HOD Login</h4>

<ul class="navbar-nav ml-auto">

<!-- Notifications Dropdown Menu -->
<li class="nav-item dropdown">
  <a class="nav-link" data-toggle="dropdown" href="#" aria-expanded="false">
    <i class="fas fa-cog"></i>
  </a>
  <div class="dropdown-menu dropdown-menu-lg dropdown-menu-right" style="left: inherit; right: 0px;">

    <a href="{% url 'admin_profile' %}" class="dropdown-item">
      <i class="fas fa-user-edit mr-2"></i> Update Profile
    </a>

    <div class="dropdown-divider"></div>
    <a href="{% url 'logout_user' %}" class="dropdown-item">
      <i class="fas fa-power-off mr-2"></i> Logout
    </a>
    <div class="dropdown-divider"></div>
  </div>
</li>
</ul>

</nav>
<!-- /.navbar -->

<!-- Main Sidebar Container -->
{% include 'hod_template/sidebar_template.html' with user=user id=id %}

<!-- Content Wrapper. Contains page content -->
<div class="content-wrapper">

<!-- Content Header (Page header) -->
<div class="content-header">
  <div class="container-fluid">
    <div class="row mb-2">
      <div class="col-sm-6">
        <h1 class="m-0 text-dark">
          {% block page_title %}

```

```

        {% endblock page_title %}
    </h1>
</div><!-- /.col -->
<div class="col-sm-6">
    <ol class="breadcrumb float-sm-right">
        <li class="breadcrumb-item"><a href="{% url 'admin_home' %}">Home</a></li>
    </ol>
</div><!-- /.col -->
</div><!-- /.row -->
</div><!-- /.container-fluid -->
</div>
<!-- /.content-header -->

<!-- Main content -->
{% block main_content %}
{% endblock main_content %}
<!-- /.content -->
</div>
<!-- /.content-wrapper -->
{% include 'hod_template/footer.html' %}

</div>
<!-- ./wrapper -->

<!-- jQuery -->
<script src="{% static 'jquery/jquery.min.js' %}"></script>
<!-- jQuery UI 1.11.4 -->
<script src="{% static 'jquery-ui/jquery-ui.min.js' %}"></script>
<!-- Resolve conflict in jQuery UI tooltip with Bootstrap tooltip -->
<script>
$.widget.bridge('uibutton', $.ui.button)
</script>
<!-- Bootstrap 4 -->
<script src="{% static 'bootstrap/js/bootstrap.bundle.min.js' %}"></script>
<!-- ChartJS -->
<script src="{% static 'chart.js/Chart.min.js' %}"></script>
<!-- Sparkline -->
<script src="{% static 'sparklines/sparkline.js' %}"></script>
<!-- JQVMap -->

```

```

<script src="{% static "jqvmap/jquery.vmap.min.js" %}"></script>
<script src="{% static "jqvmap/maps/jquery.vmap.usa.js" %}"></script>
<!-- jQuery Knob Chart -->
<script src="{% static "jquery-knob/jquery.knob.min.js" %}"></script>
<!-- daterangepicker -->
<script src="{% static "moment/moment.min.js" %}"></script>
<script src="{% static "daterangepicker/daterangepicker.js" %}"></script>
<!-- Tempusdominus Bootstrap 4 -->
<script src="{% static "tempusdominus-bootstrap-4/js/tempusdominus-bootstrap-4.min.js" %}"></script>
<!-- Summernote -->
<script src="{% static "summernote/summernote-bs4.min.js" %}"></script>
<!-- overlayScrollbars -->
<script src="{% static "overlayScrollbars/js/jquery.overlayScrollbars.min.js" %}"></script>
<!-- AdminLTE App -->
<script src="{% static 'dist/js/adminlte.js' %}"></script>
<!-- AdminLTE dashboard demo (This is only for demo purposes) -->
<script src="{% static 'dist/js/pages/dashboard.js' %}"></script>
<!-- AdminLTE for demo purposes -->
<script src="{% static 'dist/js/demo.js' %}"></script>

{% comment %} For Custom JS {% endcomment %}
{% block custom_js %}
{% endblock custom_js %}

</body>
</html>

```

## add\_course\_template.html

```

{% block main_content %}

{% load static %}

<section class="content">
  <div class="container-fluid">

    <div class="row">
      <div class="col-md-12">
        <!-- general form elements -->
        <div class="card card-primary">
          <div class="card-header">
            <h3 class="card-title">Add Course</h3>
          </div>

```

```

<!-- /.card-header -->
<!-- form start -->
<form role="form" method="POST" action="{% url 'add_course_save' %}">
    {% csrf_token %}

    {% comment %} Display Messages {% endcomment %}
    {% if messages %}
    <div class="form-group">
    <div class="col-12">
        {% for message in messages %}
        {% if message.tags == "error" %}
            <div class="alert alert-danger alert-dismissible fade show" role="alert" style="margin-top:
10px;">

                {{ message }}
                <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
        {% elif message.tags == "success" %}
            <div class="alert alert-success alert-dismissible fade show" role="alert" style="margin-
top: 10px;">

                {{ message }}
                <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
        {% endif %}
        {% endfor %}
    </div>
    </div>
    {% endif %}

    <div class="card-body">
        <div class="form-group">
            <label>Course Name </label>
            <input type="text" class="form-control" name="course" placeholder="Enter Course">
        </div>

    </div>

```

```

        <!-- /.card-body -->

        <div class="card-footer">
            <button type="submit" class="btn btn-primary">Add Course</button>
        </div>
    </form>
</div>
<!-- /.card -->

</div>
</div>

</div><!-- /.container-fluid -->
</section>

{% endblock main_content %}

```

## add\_session\_template.html

```

{% extends 'hod_template/base_template.html' %}

{% block page_title %}
    Add Course
{% endblock page_title %}
{% extends 'hod_template/base_template.html' %}

{% block page_title %}
    Add Session Year
{% endblock page_title %}

{% block main_content %}

{% load static %}

<section class="content">
    <div class="container-fluid">

        <div class="row">
            <div class="col-md-12">
                <!-- general form elements -->
                <div class="card card-primary">
                    <div class="card-header">

```

```

    <h3 class="card-title">Add Session Year</h3>
</div>
<!-- /.card-header -->
<!-- form start -->
<form role="form" method="POST" action="{% url 'add_session_save' %}">
    {% csrf_token %}

    {% comment %} Display Messages {% endcomment %}
    {% if messages %}
    <div class="form-group">
    <div class="col-12">
        {% for message in messages %}
        {% if message.tags == "error" %}
            <div class="alert alert-danger alert-dismissible fade show" role="alert" style="margin-top:
10px;">
                {{ message }}
                <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
        {% elif message.tags == "success" %}
            <div class="alert alert-success alert-dismissible fade show" role="alert" style="margin-
top: 10px;">
                {{ message }}
                <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
        {% endif %}
        {% endfor %}
    </div>
    </div>
    {% endif %}

    <div class="card-body">
    <div class="form-group">
        <label>Session Start Year </label>
        <input type="date" class="form-control" name="session_start_year">
    </div>

```

```

        <div class="form-group">
            <label>Session End Year </label>
            <input type="date" class="form-control" name="session_end_year">
        </div>

    </div>
    <!-- /.card-body -->

    <div class="card-footer">
        <button type="submit" class="btn btn-primary">Add Session Year</button>
    </div>
</form>
</div>
<!-- /.card -->

</div>
</div>

</div><!-- /.container-fluid -->
</section>

```

```
{% endblock main_content %}
```

## add\_staff\_template.html

```
{% extends 'hod_template/base_template.html' %}
```

```
{% block page_title %}
```

```
    Add Staff
```

```
{% endblock page_title %}
```

```
{% block main_content %}
```

```
{% load static %}
```

```
<section class="content">
```

```
    <div class="container-fluid">
```

```
        <div class="row">
```

```
            <div class="col-md-12">
```

```
                <!-- general form elements -->
```



```

<div class="card card-primary">
<div class="card-header">
  <h3 class="card-title">Add Staff</h3>
</div>
<!-- /.card-header -->
<!-- form start -->
<form role="form" method="POST" action="{% url 'add_staff_save' %}">
  {% csrf_token %}

  {% comment %} Display Messages {% endcomment %}
  {% if messages %}
  <div class="form-group">
  <div class="col-12">
    {% for message in messages %}
    {% if message.tags == "error" %}
      <div class="alert alert-danger alert-dismissible fade show" role="alert" style="margin-top:
10px;">
        {{ message }}
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
    {% elif message.tags == "success" %}
      <div class="alert alert-success alert-dismissible fade show" role="alert" style="margin-
top: 10px;">
        {{ message }}
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
    {% endif %}
    {% endfor %}
  </div>
  </div>
  {% endif %}

  <div class="card-body">
    <div class="form-group">
      <label>Email address</label>

```

```

        <input type="email" class="form-control" name="email" placeholder="Enter email"
id="id_email">
    </div>

    <div class="form-group">
        <label>Username</label>
        <input type="text" class="form-control" name="username" placeholder="Username"
id="id_username">
    </div>

    <div class="form-group">
        <label>Password</label>
        <input type="password" class="form-control" name="password" placeholder="Password">
    </div>

    <div class="form-group">
        <label>First Name</label>
        <input type="text" class="form-control" name="first_name" placeholder="First Name">
    </div>

    <div class="form-group">
        <label>Last Name</label>
        <input type="text" class="form-control" name="last_name" placeholder="Last Name">
    </div>

    <div class="form-group">
        <label>Address</label>
        <textarea class="form-control" name="address" placeholder="Address"></textarea>
    </div>

</div>
<!-- /.card-body -->

<div class="card-footer">
    <button type="submit" class="btn btn-primary">Add Staff</button>
</div>
</form>
</div>

```

```

        <!-- /.card -->

    </div>
</div>

</div><!-- /.container-fluid -->
</section>

{% endblock main_content %}

{% block custom_js %}
{% comment %} Checking if email and username already exists or not usin Ajax {% endcomment %}

<script>
    $(document).ready(function(){
        // keyup event will be triggered when user leaves keyboard
        $("#id_email").keyup(function(){
            var email = $(this).val();

            if(email!=""){
                $.ajax({
                    url : '{% url 'check_email_exist' %}',
                    type : 'POST',
                    data : {email:email}
                })
                .done(function(response){
                    //console.log(response);

                    if(response == "True"){
                        $(".email_error").remove();
                        $("<span class='email_error' style='color: red; padding: 5px; font-weight: bold;'> Email Not
Available. </span>").insertAfter("#id_email")
                    }
                    else{
                        $(".email_error").remove();
                        $("<span class='email_error' style='color: green; padding: 5px; font-weight: bold;'> Email
Available. </span>").insertAfter("#id_email")
                    }
                })

                .fail(function(){

```

```

        console.log("Failed");
    })
}
else{
    $(".email_error").remove();
}

})

$("#id_username").keyup(function(){
    var username = $(this).val();

    if(username!=""){
        $.ajax({
            url : '{% url 'check_username_exist' %}',
            type : 'POST',
            data : {username:username}
        })
        .done(function(response){
            //console.log(response);

            if(response == "True"){
                $(".username_error").remove();
                $(".username_error").insertAfter("#id_username")
            }
            else{
                $(".username_error").remove();
                $(".username_error").insertAfter("#id_username")
            }
        })

        .fail(function(){
            console.log("Failed");
        })
    }
    else{
        $(".username_error").remove();
    }
}

```

```
    })
  })
</script>
```

```
{% endblock custom_js % }
```

## add\_student\_template.html

```
{% extends 'hod_template/base_template.html' % }
```

```
{% block page_title % }
```

```
    Add Student
```

```
{% endblock page_title % }
```

```
{% block main_content % }
```

```
{% load static % }
```

```
<section class="content">
```

```
    <div class="container-fluid">
```

```
        <div class="row">
```

```
            <div class="col-md-12">
```

```
                <!-- general form elements -->
```

```
                <div class="card card-primary">
```

```
                    <div class="card-header">
```

```
                        <h3 class="card-title">Add Student</h3>
```

```
                    </div>
```

```
                <!-- /.card-header -->
```

```
                <!-- form start -->
```

```
                {% url 'add_student_save' as action_path % }
```

```
                {% include 'hod_template/form_template.html' with messages=messages form=form
action_path=action_path button_text="Add Student" % }
```

```
            </div>
```

```
        <!-- /.card -->
```

```
    </div>
```

```
</div>
```

```
</div><!-- /.container-fluid -->
```

```
</section>
```

```

{% endblock main_content %}

{% block custom_js %}
{% comment %} Checking if email and username already exists or not usin Ajax {% endcomment %}

<script>
$(document).ready(function(){
    // keyup event will be triggered when user leaves keyboard
    $("#id_email").keyup(function(){
        var email = $(this).val();

        if(email!=""){
            $.ajax({
                url : '{% url 'check_email_exist' %}',
                type : 'POST',
                data : {email:email}
            })
            .done(function(response){
                //console.log(response);

                if(response == "True"){
                    $(".email_error").remove();
                    $("<span class='email_error' style='color: red; padding: 5px; font-weight: bold;'> Email Not
Available. </span>").insertAfter("#id_email")
                }
                else{
                    $(".email_error").remove();
                    $("<span class='email_error' style='color: green; padding: 5px; font-weight: bold;'> Email
Available. </span>").insertAfter("#id_email")
                }
            })

            .fail(function(){
                console.log("Failed");
            })
        }
        else{
            $(".email_error").remove();
        }
    }
}
}

```

```

    })

    $("#id_username").keyup(function(){
        var username = $(this).val();

        if(username!=""){
            $.ajax({
                url : '{% url 'check_username_exist' %}',
                type : 'POST',
                data : {username:username}
            })
            .done(function(response){
                //console.log(response);

                if(response == "True"){
                    $(".username_error").remove();
                    $("<span class='username_error' style='color: red; padding: 5px; font-weight: bold;'> Username
Not Available. </span>").insertAfter("#id_username")
                }
                else{
                    $(".username_error").remove();
                    $("<span class='username_error' style='color: green; padding: 5px; font-weight: bold;'> Username
Available. </span>").insertAfter("#id_username")
                }
            })
        }

        .fail(function(){
            console.log("Failed");
        })
    }
    else{
        $(".username_error").remove();
    }
}
})
})
</script>

```

```
{% endblock custom_js %}
```

## add\_subject\_template.html

```
{% extends 'hod_template/base_template.html' %}

{% block page_title %}
    Add Subject
{% endblock page_title %}

{% block main_content %}

{% load static %}

<section class="content">
    <div class="container-fluid">

        <div class="row">
            <div class="col-md-12">
                <!-- general form elements -->
                <div class="card card-primary">
                    <div class="card-header">
                        <h3 class="card-title">Add Subject</h3>
                    </div>
                    <!-- /.card-header -->
                    <!-- form start -->
                    <form role="form" method="POST" action="{% url 'add_subject_save' %}">
                        {% csrf_token %}

                        {% comment %} Display Messages {% endcomment %}
                        {% if messages %}
                            <div class="form-group">
                                <div class="col-12">
                                    {% for message in messages %}
                                        {% if message.tags == "error" %}
                                            <div class="alert alert-danger alert-dismissible fade show" role="alert" style="margin-top:
10px;">
                                                {{ message }}
                                                <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                                                    <span aria-hidden="true">&times;</span>
                                                </button>
                                            </div>
                                        {% endif %}
                                    {% endfor %}
                                </div>
                            </div>
                        {% endif %}
                    </form>
                </div>
            </div>
        </div>
    </div>
</section>
```



```

    </div>
    {% elif message.tags == "success" %}
    <div class="alert alert-success alert-dismissible fade show" role="alert" style="margin-
top: 10px;">

    {{ message }}
    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
    </button>
    </div>
    {% endif %}
    {% endfor %}
</div>
</div>
{% endif %}

```

```

<div class="card-body">
  <div class="form-group">
    <label>Subject Name </label>
    <input type="text" class="form-control" name="subject" placeholder="Enter Subject">
  </div>

```

```

<div class="form-group">
  <label>Course </label>
  <select class="form-control" name="course">
    {% for course in courses %}
    <option value="{{ course.id }}">{{ course.course_name }}</option>
    {% endfor %}
  </select>
</div>

```

```

<div class="form-group">
  <label>Staff </label>
  <select class="form-control" name="staff">
    {% for staff in staffs %}
    <option value="{{ staff.id }}">{{ staff.first_name }} {{ staff.last_name }}</option>
    {% endfor %}
  </select>
</div>

```

```

</div>

```

```

        <!-- /.card-body -->

        <div class="card-footer">
        <button type="submit" class="btn btn-primary">Add Subject</button>
        </div>
    </form>
</div>
<!-- /.card -->

</div>
</div>

</div><!-- /.container-fluid -->
</section>

{% endblock main_content %}

```

## admin\_profile.html

```

{% extends 'hod_template/base_template.html' %}

{% block page_title %}
    Update Profile
{% endblock page_title %}

{% block main_content %}

{% load static %}

<section class="content">
    <div class="container-fluid">

        <div class="row">
            <div class="col-md-12">
                <!-- general form elements -->
                <div class="card card-primary">
                    <div class="card-header">
                        <h3 class="card-title">Update Profile</h3>
                    </div>
                    <!-- /.card-header -->
                    <!-- form start -->
                    <form role="form" method="POST" action="{% url 'admin_profile_update' %}">

```

```
{% csrf_token %}
```

```
{% comment %} Display Messages {% endcomment %}
```

```
{% if messages %}
```

```
<div class="form-group">
```

```
<div class="col-12">
```

```
{% for message in messages %}
```

```
{% if message.tags == "error" %}
```

```
<div class="alert alert-danger alert-dismissible fade show" role="alert" style="margin-top:
```

```
10px;">
```

```
{{ message }}
```

```
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
```

```
<span aria-hidden="true">&times;</span>
```

```
</button>
```

```
</div>
```

```
{% elif message.tags == "success" %}
```

```
<div class="alert alert-success alert-dismissible fade show" role="alert" style="margin-
```

```
top: 10px;">
```

```
{{ message }}
```

```
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
```

```
<span aria-hidden="true">&times;</span>
```

```
</button>
```

```
</div>
```

```
{% endif %}
```

```
{% endfor %}
```

```
</div>
```

```
</div>
```

```
{% endif %}
```

```
<div class="card-body">
```

```
<div class="form-group">
```

```
<label>Username </label>
```

```
<input type="text" class="form-control" name="username" value="{{ user.username }}"
```

```
disabled="disabled">
```

```
</div>
```

```
<div class="form-group">
```

```
<label>Email </label>
```

```

        <input type="text" class="form-control" name="email" value="{{ user.email }}"
disabled="disabled">
    </div>

    <div class="form-group">
        <label>First Name </label>
        <input type="text" class="form-control" name="first_name" value="{{ user.first_name }}">
    </div>

    <div class="form-group">
        <label>Last Name </label>
        <input type="text" class="form-control" name="last_name" value="{{ user.last_name }}">
    </div>

    <div class="form-group">
        <label>Password </label>
        <input type="text" class="form-control" name="password" placeholder="Fill only if you want
to change Password.">
    </div>

</div>
<!-- /.card-body -->

<div class="card-footer">
    <button type="submit" class="btn btn-primary">Update Profile</button>
</div>
</form>
</div>
<!-- /.card -->

</div>
</div>

</div><!-- /.container-fluid -->
</section>

{% endblock main_content %}

```

Весь інший код можна знайти в файлах проекту.

**ВІДГУК**

**керівника економічного розділу  
на кваліфікаційну роботу бакалавра**

**на тему:**

**«Розробка веб-додатку для обліку студентів на базі фреймворку Django »  
студента групи 122-17-3 Удоха Сундая Тома Річарда**

**Керівник економічного розділу  
доцент каф. ПЕП та ПУ, к.е.н**

**Л. В. Касьяненко**

## LIST OF FILES ON THE DISC

Ім'я файлу	Опис
Пояснювальні документи	
RichardUdoh_Diploma2021.docx	Пояснювальна записка кваліфікаційної роботи. Документ Word.
RichardUdoh_Diploma2021.pdf	Пояснювальна записка кваліфікаційної роботи в форматі PDF.
Програма	
RichardUdoh_Diploma2021.zip	Архів. Містить коди програми.
Презентація	
RichardUdoh_Diploma2021.pptx	Презентація кваліфікаційної роботи.