SMART ACADEMIC SCHEDULER

Project Report Submitted By

ABHISHEK SCARIYA M B

Reg. No.: AJC20MCA-2001

In Partial fulfillment for the Award of the Degree Of

MASTER OF COMPUTER APPLICATIONS (2 Year) (MCA)
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2020-2022

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "**SMART ACADEMIC SCHEDULER"** is the bonafide work of **ABHISHEK SCARIYA M B (Reg.No:AJC20MCA-2001)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

Ms. Paulin Paul	**Ms. Nimmy Francis**
Internal Guide	**Coordinator**
Rev. Fr. Dr. Rubin Thottupurathu Jose	**External Examiner**
Head of the Department	

DECLARATION

I hereby declare that the project report **"SMART ACADEMIC SCHEDULER"** is a bonafide work done at Amal Jyothi College of Engineering, towards the partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

Date: 21-07-2022	**ABHISHEK SCARIYA M B**
KANJIRAPPALLY	**Reg. No: AJC20MCA-2001**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish toexpress my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev. Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinators **Rev. Fr. Dr. Rubin Thottupurathu Jose** and **Ms. Nimmy Francis** for their valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also like to express sincere gratitude to my guide**, Ms**. **Paulin Paul** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication,and encouragement shown towards the project. I convey my hearty thanks to my family forthe moral support, suggestions, and encouragement to make this venture a success.

ABHISHEK SCARIYA M B

ABSTRACT

The smart academic scheduler aims to help students and teachers stay organized and up-to-date at all times. A person may apply for a course from the college website. These applications are collected and a rank list is prepared by an admission in-charge. A selected number of students are invited to the college for interview and further admission process at the discretion of the admission-in charge. After the admission procedure, the data of these students are automatically sent to the corresponding head of department. The HOD adds the admitted students to the department, and thus batches of students are formed. The HOD allocates subjects and batches to teachers.

This system features an automated timetable generation algorithm which generates a timetable with no clashes in the timings. The system uses genetic algorithm, which is a machine learning algorithm that falls under the umbrella of evolutionary algorithms. This timetable can be viewed by the teachers and students. Furthermore, it also features an attendance management system for student attendance. The attendance for individual hours gets updated according to the timetable.

CONTENT

List of Abbreviation

IDE	-	Integrated Development Environment
HTML	-	Hyper Text Markup Language.
CSS	-	Cascading Style Sheet
SQL	-	Structured Query Language
HOD	-	Head of Department
UML	-	Unified Modeling Language

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The smart academic scheduler aims to help students and teachers stay organized and up-to-date at all times. A person may apply for a course from the college website. These applications are collected and classified as stage-1 applications by an admission in-charge. A confirmation mail is sent to these applicants regarding their application. Confirmed applicants are considered for rank list preparation. A selected number of students are invited to the college for interview and further admission process at the discretion of the admission-in charge. After the admission procedure, the data of these students are automatically sent to the corresponding head of department. The HOD adds the admitted students to the department, and thus batches of students are formed. The HOD allocates subjects and batches to teachers.

This system features an automated timetable generation algorithm which generates a timetable with no clashes in the timings. It uses genetic algorithm to achieve this. Genetic algorithm is a machine learning algorithm which falls under the umbrella of evolutionary algorithms. The generated timetable can be viewed by teachers and students. Furthermore, it also features an attendance management system for student attendance. The attendance for individual hours gets updated based on the timetable.

1.2 PROJECT SPECIFICATION

The proposed system is a college website in which an applicant can apply for a course. The admission process is automated.

The system includes 5 modules. They are:

1. Admin

Admin must have a login into this system. He has the overall control of the system. Admin canadd or update department, course. Admin can add teachers and admission-in-charge user into the system. He can also set the HOD for each department and change as required.

2. Admission in charge

Admission in charge can view the new course applications under stage-1 applications. They can send a confirmation mail to the applicants. Confirmed applicants are considered for preparing the rank list. Selected number of students based on the rank list are invited for interview. Interview will be held offline. After the offline admission process, the admission-in-charge can admit applicants to the college. These admitted students' details are sent to the HOD of the corresponding department.

3. Head of Department (HOD)

HOD can view the details of the students who have been admitted to the courses under their department. This user creates new batches and adds the newly admitted students to these batches. They are responsible for setting class teacher for the new batches. They also add new subjects and assigns teachers to these subjects. These subjects are also linked to courses. HOD also manages class hours and class rooms. Based on these details, a timetable is created. The timetable for a single batch can be generated perfectly.

4. Teacher

Teacher can manage their assigned batch. They can edit student details of their respective batch. They can view their timetable. They can add study materials for the designated subjects. They can also schedule assignments for these subjects and evaluate students' submissions. They can mark attendance for students of their batch subject-wise based on timetable and date. They can use these details to publish students' internals.

5. Student

Student can login and download study materials. They can upload assignments scheduled by their teachers. They can view timetable. They can view their attendance and internal marks.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System analysis is the process of acquiring and analyzing data, identifying issues, and using the data to suggest system changes. The system users and system developers must communicate extensively during this problem-solving process. Any system development process should start with a system analysis or research. The system is meticulously examined and assessed. The system analyst assumes the role of an interrogator and delves deeply into how the current system functions. The input to the system is recognized, and the system is seen as a whole. The different procedures may be linked to the outputs from the organizations. Understanding the issue, identifying the pertinent and important variables, evaluating and synthesizing the many elements, and selecting the best or, at the very least, most acceptable course of action are all part of system analysis.

The process must be thoroughly studied using a variety of methodologies, including surveys and interviews. To reach a conclusion, the information gathered by various sources must be carefully examined. Understanding how the system works is the conclusion. The current system is also known as the existing system. Now, the current system is carefully examined, and issue areas are found. The designer now acts as a problem-solver and works to resolve the issues the business is having. Proposals are made in place of the solutions. The suggestion is then analytically compared against the current system, and the best one is chosen. The user is given the opportunity to approve or reject the suggestion. On user request, the proposal is assessed and appropriate revisions are implemented. As soon as the user is content with the suggestion, this loop breaks.

The process of acquiring and analysing data in order to use it for future system studies is known as preliminary study. Preliminary study is a problem-solving activity that necessitates close coordination between system users and developers. It conducts a number of feasibility studies. These investigations provide an approximate estimate of the system activities, which may be used to determine the tactics to be used for an efficient system research and analysis.

2.2 EXISTING SYSTEM

Existing system is not a fully automated system. Students may register online. However the admission process is manually done which is time consuming. Timetable needs to be generated manually. Attendance is also done manually in most systems.

It is necessary to modify the existing system in order to include additional information and make the system efficient, flexible and secure.

2.3 DRAWBACKS OF EXISTING SYSTEM

- No proper online management of admission process
- Lot of human effort is needed to create timetable
- It is difficult to maintain physical attendance records and tally them
- It is difficult to send mail or call the applicants manually.
- More manual work is to be done to add students to database

2.4 PROPOSED SYSTEM

The proposed system is defined to meet all the disadvantages of the existing system. The admission process gets automated to a great extent. The applicants send their application online. Their details are received by the admission in charge who can view their details, prepare a rank list based on their previous academic record and send invitation to eligible candidates for further process. After the offline interview and document verification, the admission in charge sends admitted students' details to the HOD of the corresponding department. HOD adds these students to a batch after after which the students can get their login credentials. Here a lot of the database operations are automated as it is not manually entered by any user of the system. Also preparing rank list and sending invites are just a matter of a few button clicks.

Timetable is generated automatically without any clashes in regards to timing. For individual hours can be updated by the class teacher. Also assignments can be done online. Study materials can also be made available for the students which makes it easily accessible.

2.5 ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources, and the system will work in almost all configurations. It has got following features:

- **Admission process: -**

Admission process is held offline in most academic institutions. This makes it necessary for manual data entry after admissions and invites other mundane tasks. In our system, the admission process has been automated to a great extent. Applications are collected and goes through various phases before the actual interview. This helps filter eligible candidates more easily and stores all data for future use.

- **Timetable generation: -**

The proposed system generates timetable for the institution. It uses genetic algorithm to achieve this. This solves many problems for teachers. Drafting a timetable manually is a very mundane and arduous task for the teachers. This system can help overcome this.

- **Better service: -**

The product will avoid the burden of hard copy storage. We can also conserve the time and human resources for doing the same task. The data can be maintained for longer period with noloss of data.

- **Attendance system: -**

The attendance marking is done by the class teachers for each hour. This takes into consideration subject-wise attendance of a student by accessing the generated timetable. This is stored in the database and can be accessed for calculating internal marks of students.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

To determine if the project will ultimately achieve the goals of the organization given the labor, time, and effort expended on it, a feasibility study is carried out. The developer can forecast the project's usefulness and potential future thanks to a feasibility study. The premise for a feasibility study is the system proposal's viability, which includes the impact on the organization, ability to meet user requests, and effective use of resources. As a result, a feasibility evaluation is frequently performed before a new application is approved for development. The paper outlines the project's viability and contains a number of factors that were carefully taken into account throughout this project's feasibility assessment, including its technical, economic, and operational viabilities. The following are its features: -

3.1.1 Economical Feasibility

Cost and benefit analyses are required to support the emerging system. criteria to make sure that focus is placed on the project that will yield the best results the earliest. The price that would be involved in developing a new system is one of the variables.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

The suggested system is being created as part of a project; it has no associated costs. Additionally, all of the resources are easily accessible, demonstrating that the system may be developed affordably.

The project's costs were broken down into three categories: system costs, development costs, and hosting costs. The project was developed at a cheap cost, according to calculations. Because it was created with open source software, this is feasible

3.1.2 Technical Feasibility

The system has to be assessed first from a technical standpoint. An overview design of the system's requirements in terms of input, output, programmes, and processes must serve as the foundation for the assessment of this viability. The inquiry must next advise the kind of equipment, necessary procedure for constructing the system, and means of operating the system once it has been developed after having identified an outline system.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be created in such a way that the required performance and functionality are met within the limitations. Therefore, this project only has a few limitations. The system was created using HTML, CSS, and JS for the front end and a MYSQL server for the back end; it is theoretically viable to complete the project. The system was created using Django for the web scripting and a MYSQL server for the back end; it is theoretically viable to complete the project. The System used was also of good performance of Processor Intel i3 core; RAM 4GB and, Hard disk 1TB

3.1.3 Behavioral Feasibility

The proposed system includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed.All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	- Intel core i3
RAM	- 4 GB
Hard disk	- 1 TB

3.2.2 Software Specification

Front End	- HTML, CSS,
Backend	- Django, MYSQL
Client on PC	- Windows 7 and above.
Technologies used	- JS, HTML5, Django, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO

Quick prototyping and rational, elegant design are encouraged by the high-level Python web framework Django. It was developed by skilled programmers and takes care of a lot of the hassles related to Web development, freeing you up to focus on creating your app without having to reinvent the wheel. It is open source and cost nothing.

A standard approach for quickly and easily building websites is provided by a collection of elements known as a web framework. The core goal of Django is to make complex database-driven webpages easier to construct. A number of well-known websites, such as PBS, Instagram, Disqus, the Washington Times, Bitbucket, and Mozilla, employ Django. Python-based Django is a free and open source framework for building web applications. A web framework is a group of components that facilitates the quick and easy creation of websites.

Every time you create a website, a similar set of components is needed: a way to handle user authentication (signing up, signing in, and signing out), a management panel for your website, forms, a way to submit files, etc.

For your advantage, it has long been known that web designers encounter comparable challenges while building a new website. As a result, they worked together to develop frameworks, including Django, that provide you access to pre-made components.

Frameworks are available to help you avoid starting from scratch and to cut down on some of the costs associated with developing a new website.

Why do you need a framework?

We must examine the servers in more detail if we want to comprehend what Django is truly used for. The server must first be informed that you want a web page sent to you.

Think of a mailbox (port) that is watched for letters arriving (requests). A web server performs this. After reading the mail, the web server replies with a website. However, you need substance when you wish to convey something. Django is a tool that aids with content creation.

When a request is sent to a web server, Django receives it and attempts to determine what is being asked for. It begins by taking a web page address and attempting to determine what to do. The URL resolver in Django does this task. The term URL resolver makes sense given that a website address is referred to as a URL, or uniform resource locator. It attempts to match the URL using a collection of patterns, which is not very intelligent. Django runs a top-to-bottom pattern check and forwards requests to corresponding functions if anything matches (which is called view).

Consider a mailman carrying a letter. As he moves down the street, he compares every home number to the one on the letter. He places the letter there if it corresponds. The URL resolver operates in this manner.

All the fascinating activities take place in the view function, where we may explore through a database to find specific information. Perhaps the user requested a modification to the data? like a letter requesting that my job description be changed. The view can determine if you are authorized to do that before updating the job description and returning with the word "Done." The answer is then produced by the view and sent by Django to the user's web browser.

You don't need to be familiar with all the technical details just now; the explanation above has been somewhat simplified. It suffices to have an overall concept.

So rather than getting too deep into the specifics, let's get started with Django and pick up all the essentials as we go!

Python-based Django is a free and open source web application framework. Simply said, a framework is a group of modules that facilitate development. They are grouped together and enable you to build websites or apps using an existing source rather than starting from scratch.

This is how even straightforward websites created by a single person may nevertheless incorporate cutting-edge features like login support, administration and management panels, contact forms, comment boxes, websites created by seasoned developers, support for file uploads, and more. In other words, you would have to design these components yourself if you were building a website from scratch. Instead, by utilizing a framework, these elements are already constructed; all that is required is for correct configuration to fit your site.

The Django, "a high-level Python Web framework that supports quick development and straightforward, practical design. It was created by skilled programmers, taking care of a lot of the hassles associated with Web development so you can concentrate on creating your app instead of having to build the wheel. It is open source and cost nothing."
You may use the extensive library of modules provided by Django in your own applications. The main reason that template frameworks exist is to save developers a tonne of wasted time and hassles, and Django is no exception.

The fact that Django was developed with front-end developers in mind may also be of interest to you. "The template language used by Django is intended for front-end developers and designers who are accustomed to dealing with HTML. However, it is also adaptable and incredibly extendable, enabling programmers to improve the template language as necessary."

3.1.1 MySQL

The most well-known Open Source SQL database management system, MySQL, was developed, distributed, and supported by Oracle Corporation. On the MySQL website, you may find the most latest information about the MySQL programme..

- **MySQL is a database management system.**

  Databases are organized volumes of data. A simple grocery list, a photo gallery, or the vast amount of data in a business network are just a few examples of what it could be. To add, retrieve, and process data stored in a database, a database management system is needed, such as MySQL Server. Database management systems are essential to computing because computers are effective at processing enormous volumes of data, whether they are utilized as standalone programmes or as a part of other applications.

- **MySQL databases are relational.**

  Instead of combining all the data into one huge warehouse, a relational database keeps it in individual tables. Physical files that are designed for speed make up the database structures. The logical model, which consists of objects like databases, tables, views, rows, and columns, provides a flexible programming environment. The relationships between various data fields, such as one-to-one, one-to-many, unique, obligatory or optional, and "pointers" across other tables, could be governed by rules you write. The database ensures that your application never encounters inconsistent, duplicate, orphan, out-of-date, or missing data by abiding by these principles. MySQL stands for "Structured Query Language" with the SQL prefix.

  The most widely used standard language for communicating with databases is SQL. Depending on your programming environment, you could manually enter SQL (for example, to create reports), utilise a language-specific API that hides the SQL syntax, or incorporate SQL statements into the code of other programming languages. The ANSI/ISO SQL Standard defines SQL. Since its inception in 1986, the SQL standard has seen a lot of modifications. In this text, the 1992 standard is referred to as "SQL92," the 1999 standard as "SQL," and the most recent version of the standard as "SQL: 2003." Using the prefix SQL, "the SQL standard" refers to the SQL Standard as at any time.

- **MySQL software is Open Source.**

  Anyone may use and change the programme because it is open source. The MySQL software is available for free download and usage online by anybody. You are free to examine the source code and modify it as necessary. The GPL (GNU General Public License) is used by the MySQL software to specify what you are allowed to do

  and are not allowed to do with the programme in certain circumstances. You can purchase a commercially licenced version from us if the GPL makes you uncomfortable or if you need to integrate MySQL code into a for-profit application. For further details, see the MySQL Licensing Overview.

- **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**

  If that's what you want, you should give it a try. On a desktop or laptop, MySQL Server may run without a hitch and require little to no maintenance, along with your other apps, web servers, and other software. If you dedicate an entire machine to MySQL, you can alter the settings to make full use of the RAM, CPU, and I/O capabilities.

- **MySQL Server works in client/server or embedded systems**

  The MySQL Database Software is a client/server system that includes a multi-threaded SQL server, a number of unique client applications and libraries, administration tools, and a wide range of application programming interfaces (APIs). In addition, we provide MySQL Server as a multi-threaded integrated library that you can incorporate into your software to produce a standalone offering that is more compact, speedy, and easy to use.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Any engineering system or product development process begins with design. A creative process is design. The secret to an efficient system is a decent design. Design is the process of thoroughly outlining a process or system so that it can be physically executed by using a variety of methodologies and concepts. It can be defined as the process of using a variety of techniques and ideas to precisely specify a part, a process, or a system in enough detail to allow for its physical embodiment. Regardless of the development paradigm utilized, software design serves as the technical foundation of the software engineering process. The architectural information needed to construct a system or product is generated by the system design. This software underwent the best design phase imaginable, fine-tuning all efficiency, performance, and accuracy levels, as with any methodical approach. A user-oriented document becomes a document for programmers or database specialists during the design phase. The two stages of system design development are logical design and physical design.

4.2 UML DIAGRAM

A standard language called UML is used to describe, illustrate, build, and record the artefacts of a software system. The Object Management Group (OMG), which was in charge of developing UML, received a draught of the UML 1.0 definition in January 1997.

UML stands for Unified Modeling Language. UML is different from other well-known programming languages like C++, Java, COBOL, etc. Software designs are created using a visual language referred to as UML. UML is a general-purpose visual modelling language used for the specification, design, construction, and documentation of software systems. Although representing software systems is the most popular use of UML, it is not the only one. Simulating non-software systems is another use for it. The Object Management Group (OMG) was responsible for developing UML, and a draught of the UML 1.0 definition was presented to the OMG in January 1997.

For instance, the manufacturing facility's process flow, etc. Although UML is not a programming language, tools may be used to produce code using UML diagrams in a variety of languages. The analysis and design of objects-oriented systems are directly related to UML. UML has been standardised to the point that it is now an OMG standard. A comprehensive UML diagram that depicts a system is made up of all the parts and relationships.

The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete. UML includes the following nine diagrams.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

An illustration of the interactions between system components is a use case diagram. A use case is a method for locating, defining, and organising system needs. The word "system" in this context refers to a thing that is being built or operated, such as a website for the selling of products and services by mail order. Use case diagrams are used in UML (Unified Modeling Language), a standard language for modelling actual items and systems.

Planning general requirements, validating hardware designs, testing and debugging software products while they are still in development, creating online help resources, and finishing customer support-focused tasks are a few examples of system objectives. For instance, customer support, item ordering, catalogue updating, and payment processing are examples of use cases in a setting of product sales. A use case diagram consists of four components.

- The border, which isolates the system of interest from its surroundings.
- The actors, who are often system participants identified by the roles they play.
- The actors inside and around the system play the roles specified by the use cases.
- The connections and interactions between the actors and use cases.

Use case diagrams are created to depict a system's functional needs. To create an effective use case diagram after identifying the aforementioned things, we must adhere to the following rules

- The name of a use case is very important. It is important to choose a name that makes it obvious what tasks are being carried out.
- Assign appropriate names to the actors.
- Show dependencies and relationships in the diagram with clarity.
- As the diagram's main purpose is to establish the needs, refrain from attempting to include all possible relationships.
- When required, jot down important information to help you remember it.

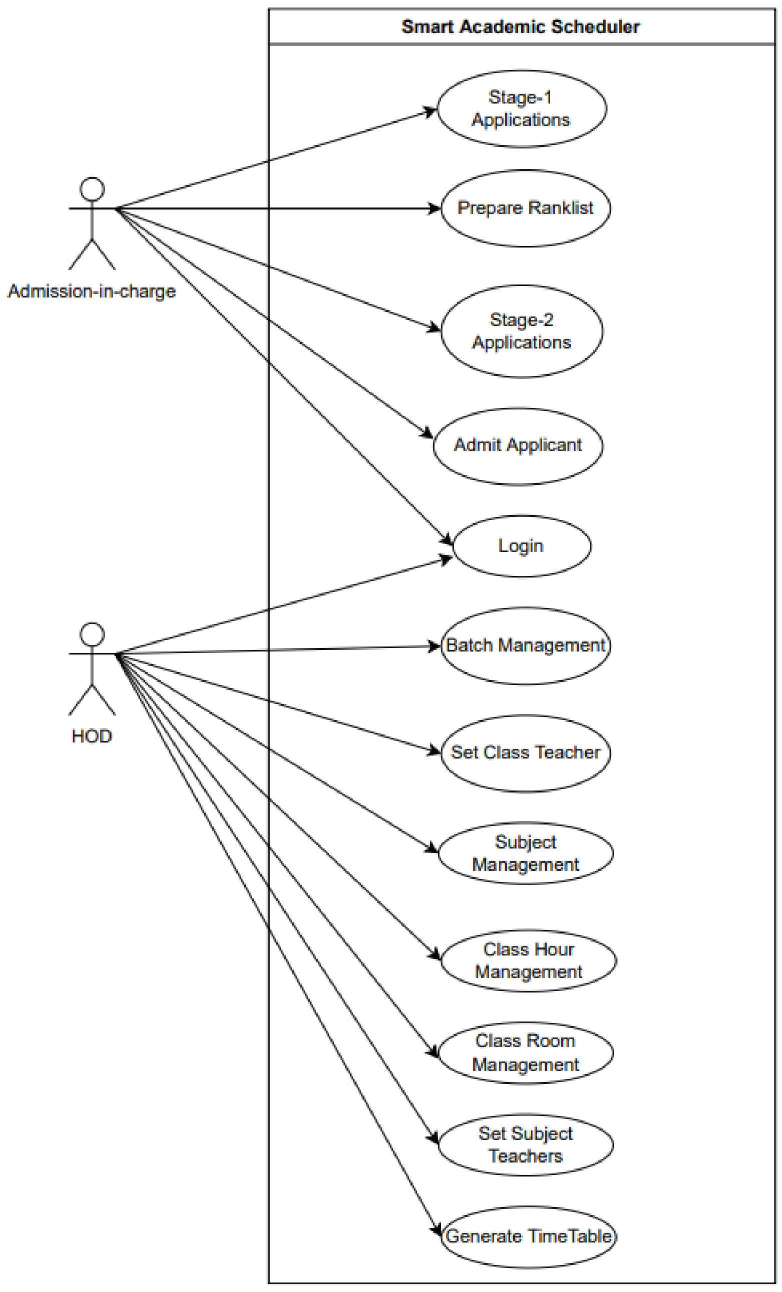


Fig 1: Use case diagram for HOD and admission-in-charge in Smart Academic Scheduler

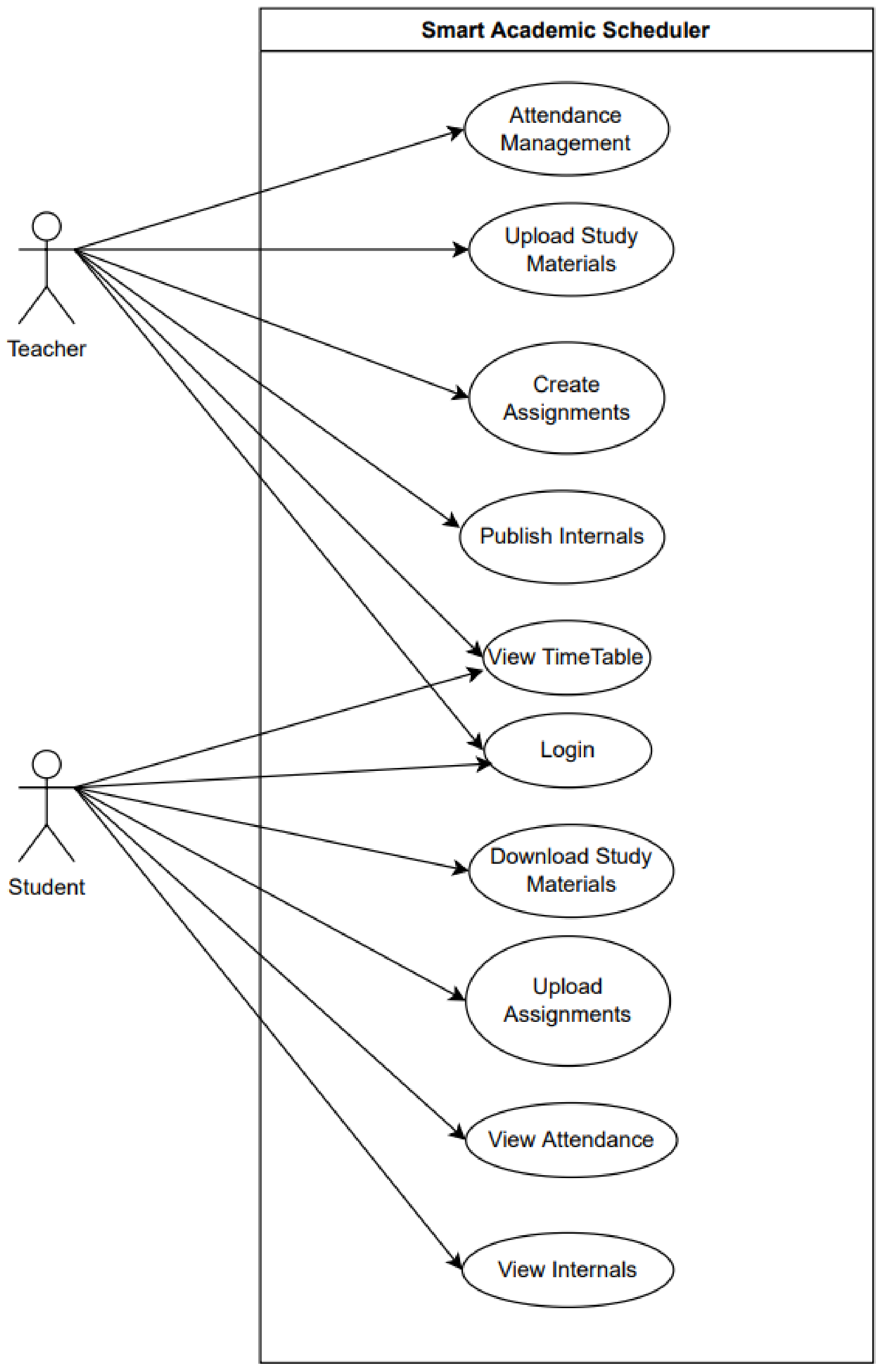


Fig 2(continuation): Use case diagram for student and teacher in Smart Academic Scheduler

4.2.2 SEQUENCE DIAGRAM

A sequence diagram essentially depicts the sequential order in which different items interact with one another. Event diagrams and event scenarios are other names for sequence diagrams. Sequence diagrams display the actions performed by a system's parts in time order. These diagrams are widely used by businesspeople and software engineers to document and explain the requirements for new and existing systems.

Sequence Diagram Notations –

i. **Actors –** An actor in a UML diagram symbolizes a specific kind of role in which it interacts with the system's elements. An actor is never within the scope of the system that we want to describe using the UML diagram. For a range of roles, including those of human users and other external subjects, we use actors. An actor is shown using the stick person notation in a UML diagram. A sequence diagram could contain multiple actors.

ii. **Lifelines –** A lifeline is a named component that displays a particular participant in a sequence diagram. A lifeline essentially stands in for each incident in a sequence diagram. In a sequence diagram, the lifeline components are at the top.

iii. **Messages –** Using messages, communication between objects is demonstrated. The messages are shown on the lifeline in chronological sequence. Arrows are how messages are represented. A sequence diagram's main components are lifelines and messages.

Messages can be broadly classified into the following categories:

- Synchronous messages
- Asynchronous Messages
- Create message
- Delete Message
- Self-Message
- Reply Message

- Found Message
- Lost Message

iv. **Guards –** We use guards in the UML to model scenarios. We employ them when we need to restrict communication under the pretence that a criterion has been met. Software engineers rely on guards to let them know when a system or particular technique has restrictions.

Uses of sequence diagrams –

- Employed to model and illustrate the reasoning behind a complex function, process, or procedure.
- They are also employed to display information about UML use case diagrams.
- Employed to comprehend the precise operation of present or upcoming systems.
- Visualize the flow of tasks and messages within a system's objects or components.

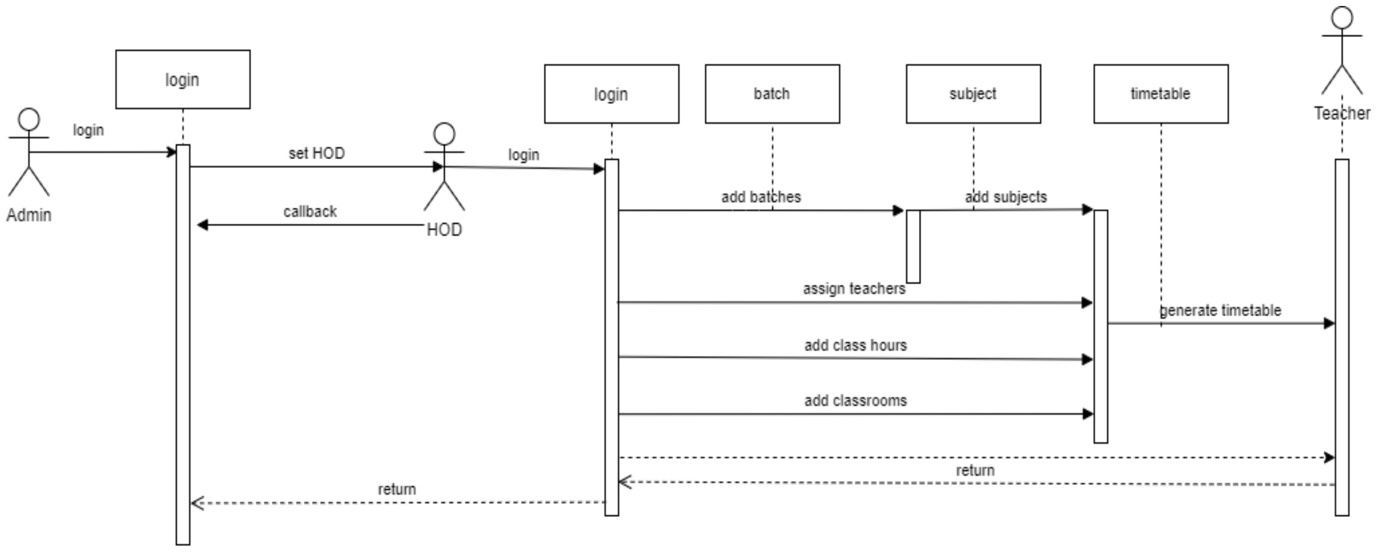


Fig 3: Sequence diagram for Smart Academic Scheduler

4.2.3 ACTIVITY DIAGRAM

The activity diagram describes the system's dynamic properties. A flowchart that illustrates how one action leads to another is an activity diagram. You could consider the activity to be a system's operation. The control flow starts when one procedure leads to another. This flow may be concurrent, parallel, or branching. Activity diagrams show a variety of flow control techniques that use different parts like forks, affix, etc.

Activity diagrams are used to visualize a system's dynamic nature as well as to develop the executable system using forward and reverse engineering techniques. The only thing missing from the activity diagram is the message section.

HOD

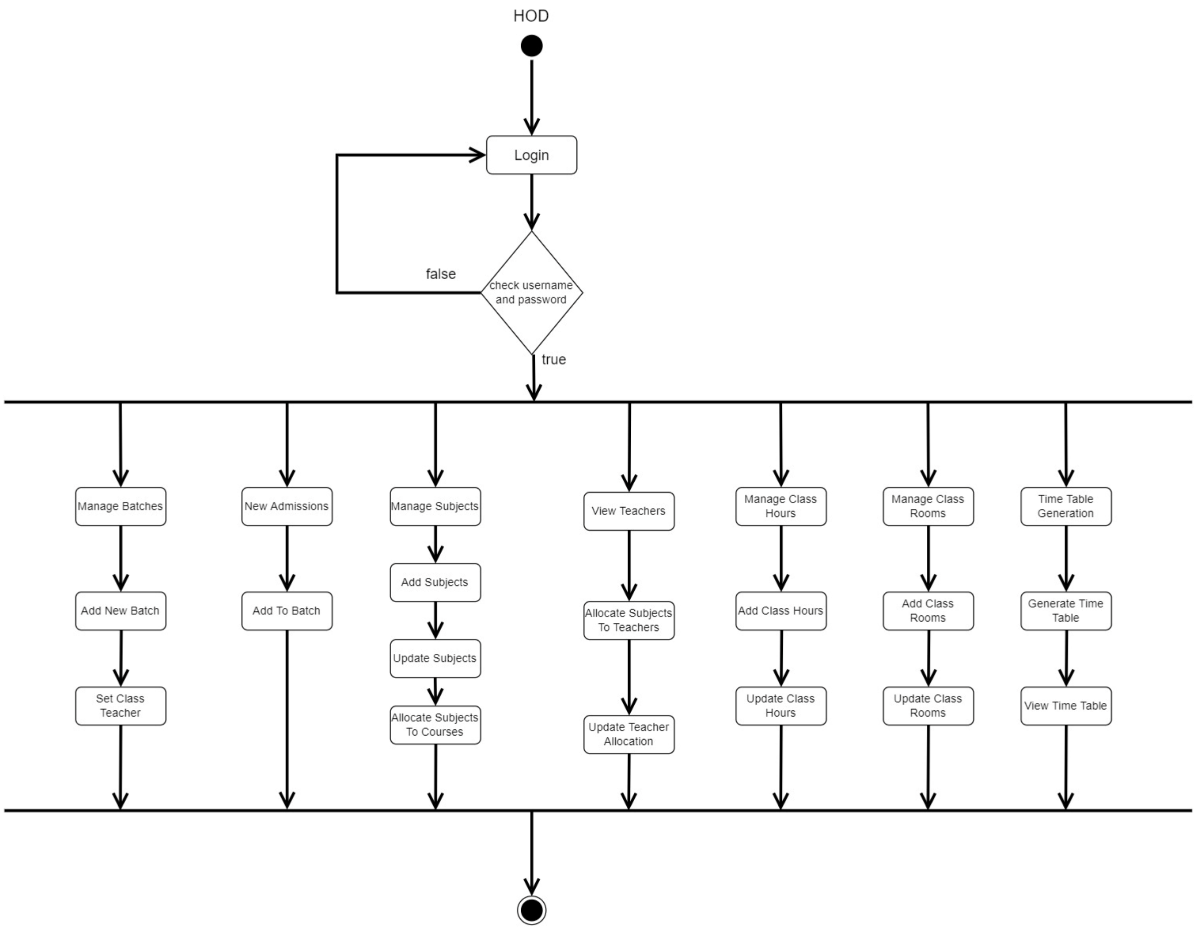


Fig 4: Activity diagram for HOD in Smart Academic Scheduler

ADMIN

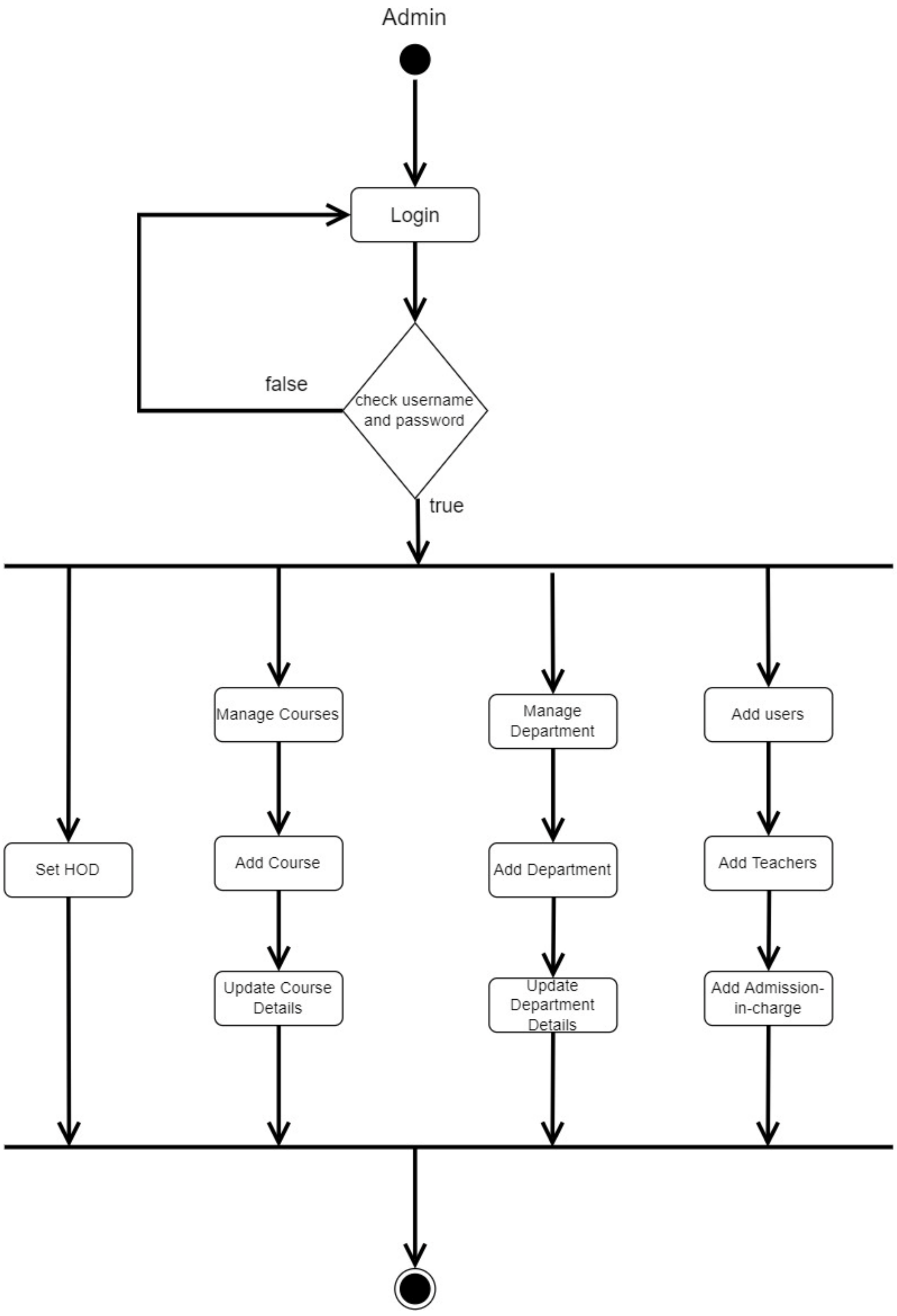


Fig 5(continuation): Activity diagram for admin in Smart Academic Scheduler

ADMISSION-IN-CHARGE

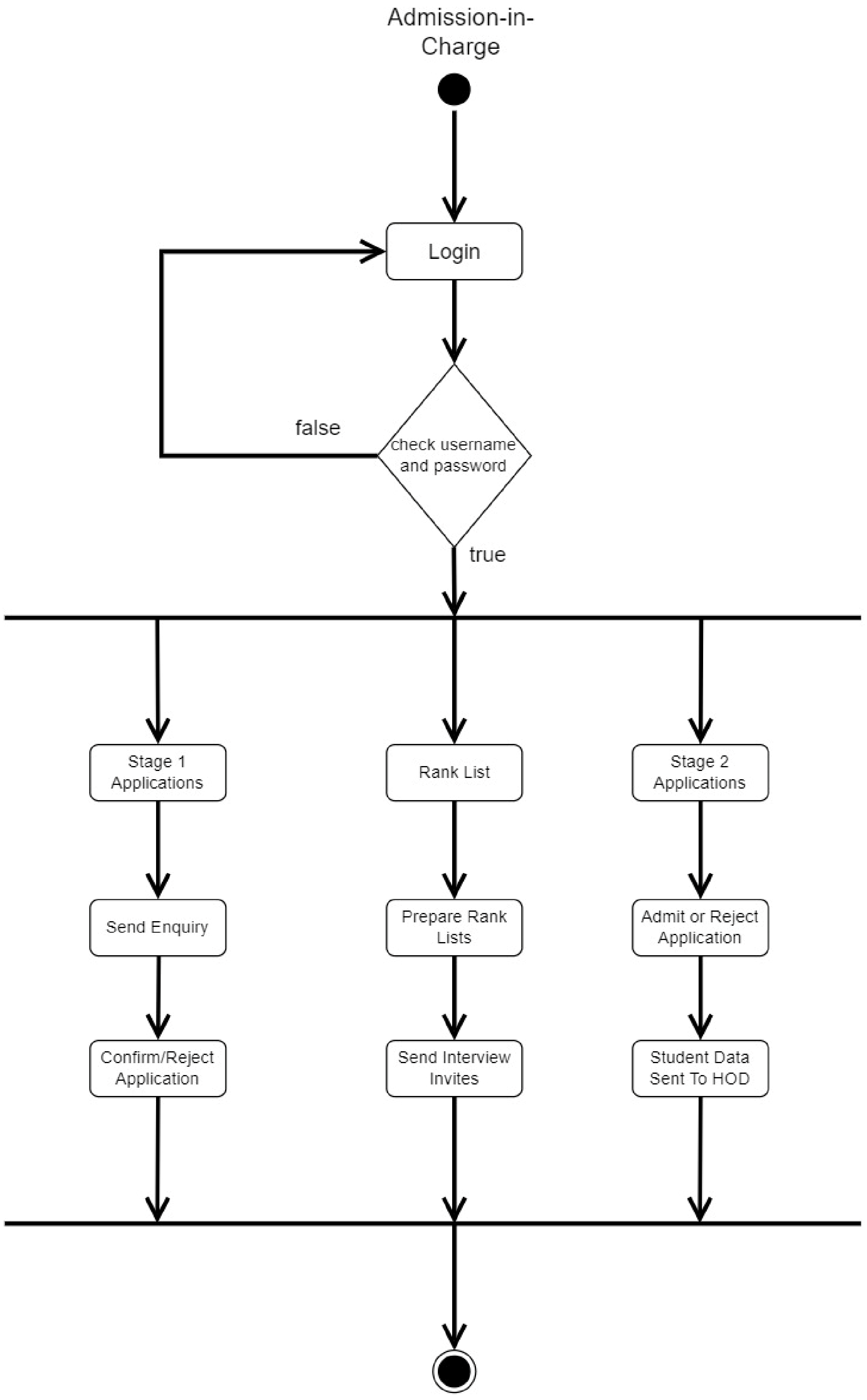


Fig 6(continuation): Activity diagram for admission-in-charge in Smart Academic Scheduler

4.2.4 CLASS DIAGRAM

Class diagrams are a type of static diagram. It represents the static view of the application. Class diagrams are useful for visualising, describing, and documenting numerous system components as well as for writing executable code for software applications. A class diagram describes the constraints imposed on the system along with the properties and operations of a class. Class diagrams are frequently used in the modelling of object-oriented systems since they are the only UML diagrams that can be directly mapped with object-oriented languages. A class diagram shows a collection of classes, interfaces, affiliations, collaborations, and limits. It is also known as a structural diagram.

The class diagram's goal may be summed up as –

• Analysis and design of an application's static view.

• Describe the duties of a system.

• Foundation for deployment and component diagrams.

• Forward and reverse engineering.

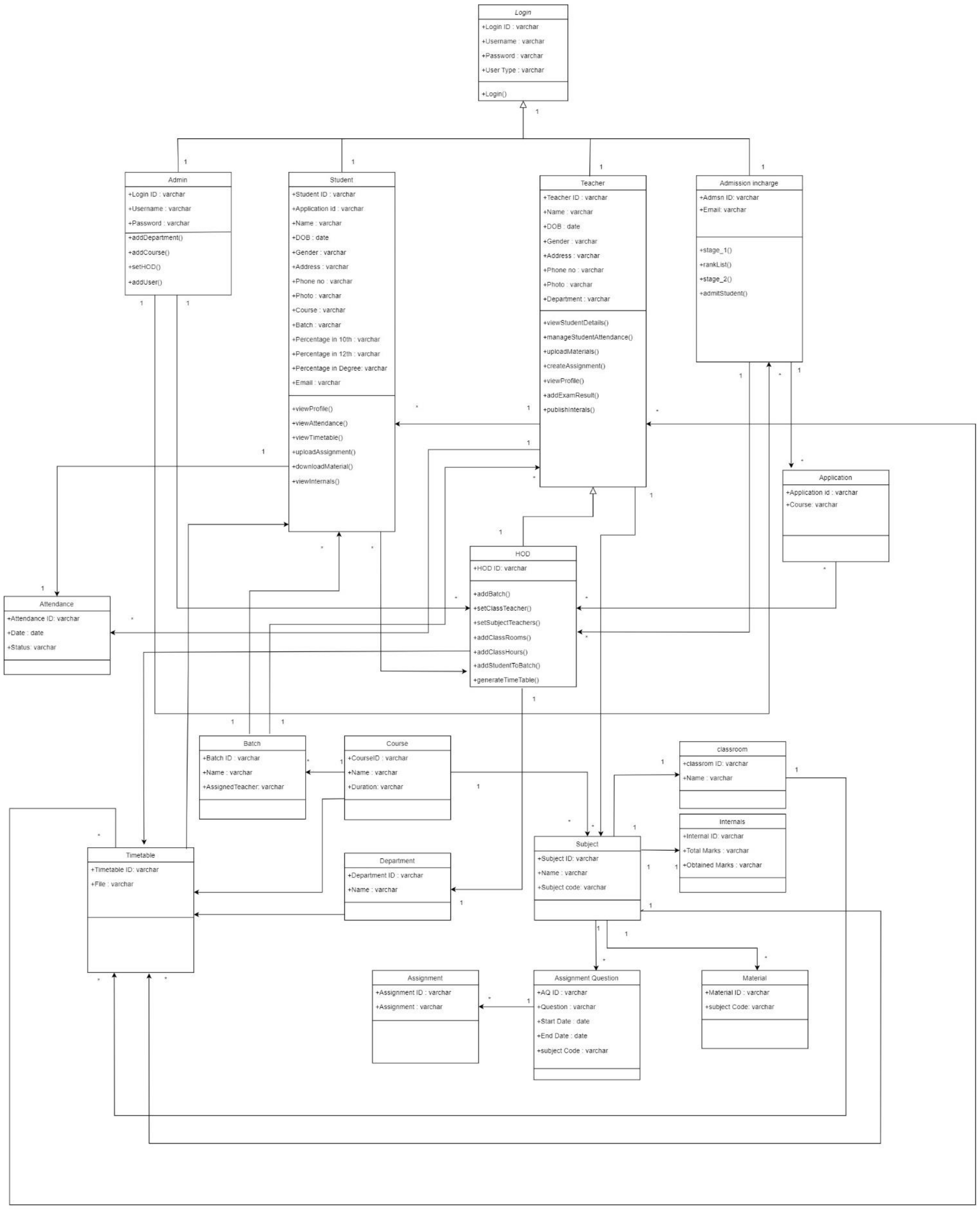


Fig 7: Class diagram for Smart Academic Scheduler

4.2.5 COMPONENT DIAGRAM

Component diagrams are a particular kind of UML diagram. Additionally, the purpose is different from the previously given diagrams. It describes the components involved in producing that functionality, but it does not define the functioning of the system as a whole. Component diagrams are used to represent the real physical components of a system from that point of view.
These components include, among others, files, libraries, and packages. Component diagrams can also be thought of as a static implementation view of a system. The components' configuration is shown in a static implementation at a certain point in time. A single component diagram is unable to depict the entire system, thus a set of diagrams are utilized instead. The purpose of the component diagram can be summarized as –

- Visualize the components of a system.
- Construct executable by employing forward and reverse engineering.
- Describe how the elements are arranged and connected.

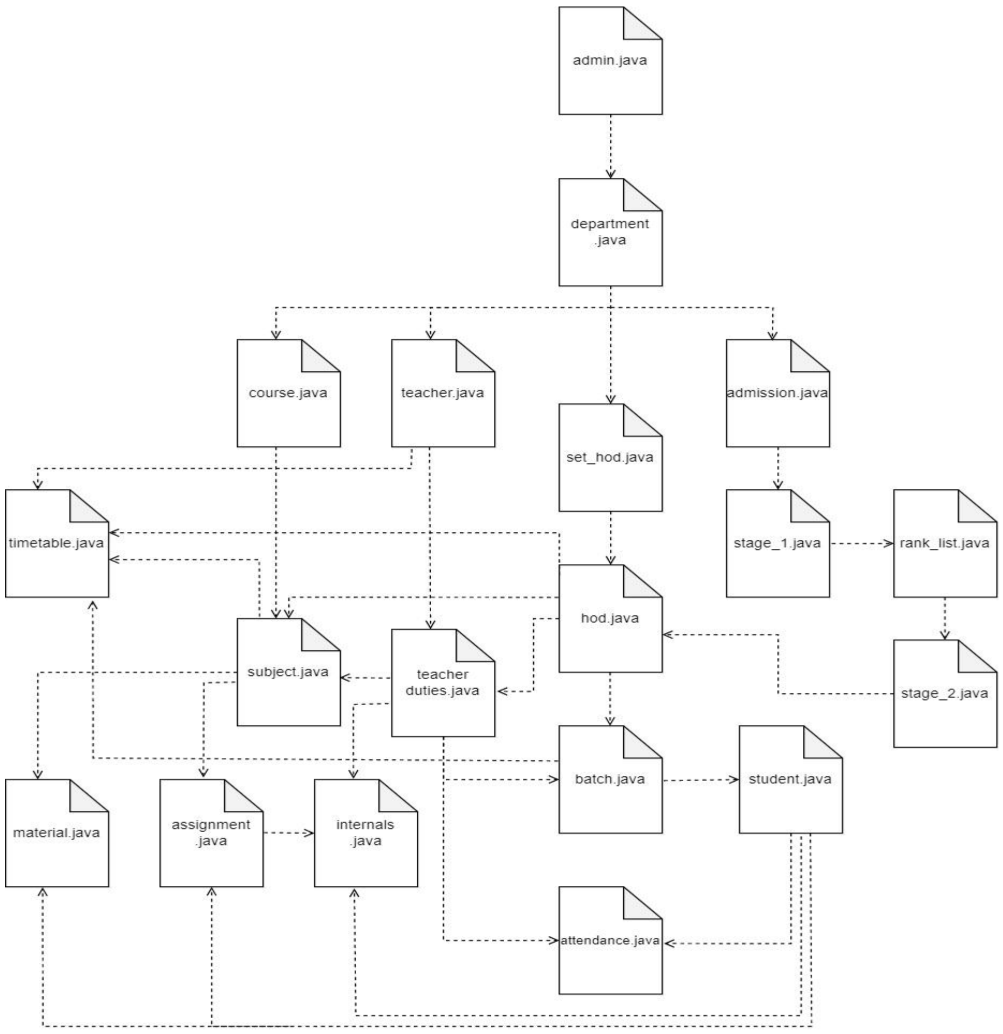


Fig 8: Component diagram for Smart Academic Scheduler

4.2.6 OBJECT DIAGRAM

Since class diagrams are the source of object diagrams, class diagrams are a prerequisite for object diagrams. An instance of a class diagram is represented by an object diagram. Class and object diagrams both use the same fundamental ideas. The static view of a system is likewise represented by object diagrams, but this static view represents a momentary snapshot of the system. To represent a group of items and their connections as an instance, object diagrams are employed. The purpose of the object diagram can be summarized as –

• Forward and reverse engineering.

• Object relationships of a system

• Static representation of a dialogue.

• Recognize object behaviour and its relationship from a practical standpoint.

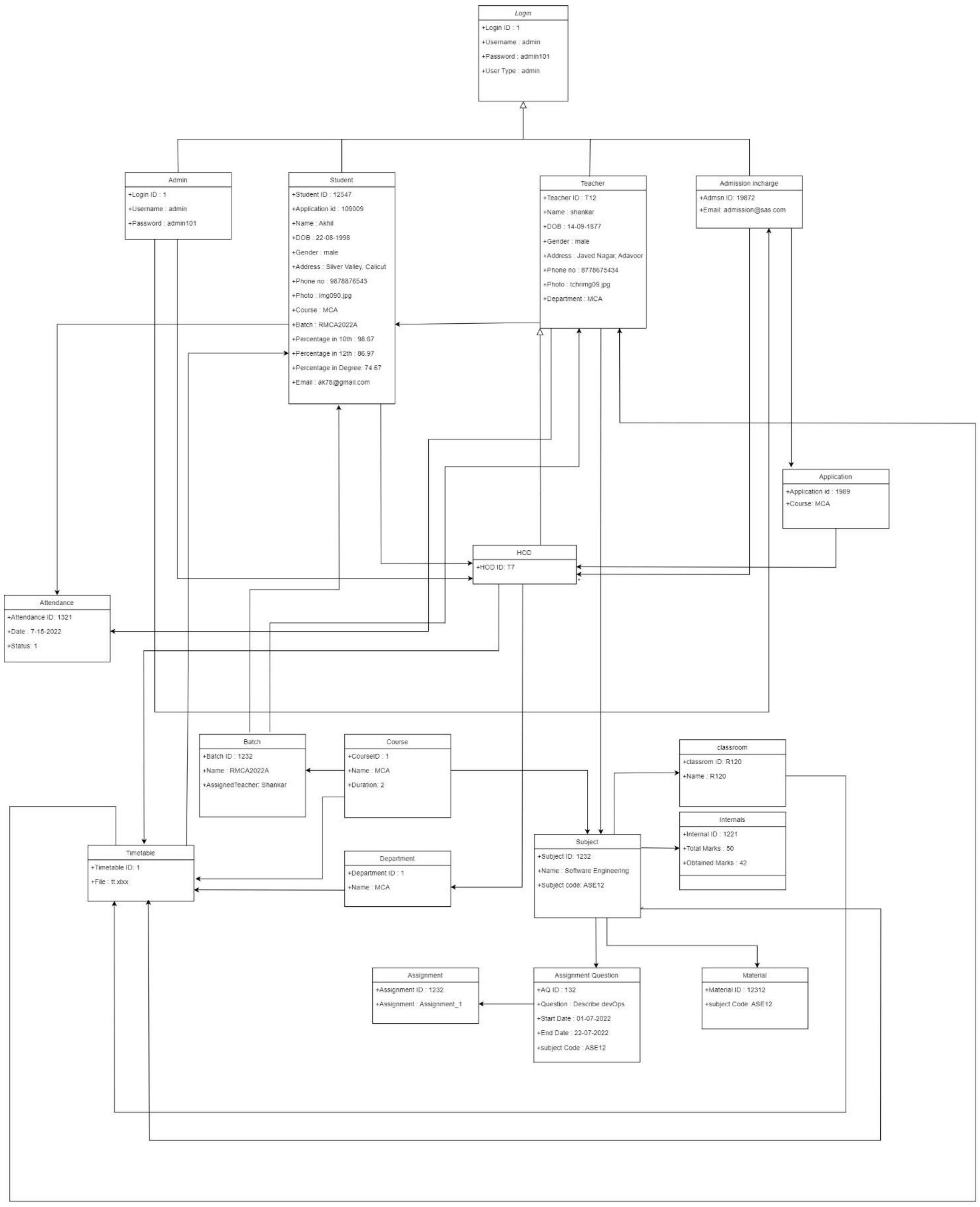


Fig 9: Object diagram for Smart Academic Scheduler

4.2.7 DEPLOYMENT DIAGRAM

The structure of the physical parts of a system, where the software components are installed, is depicted using deployment diagrams. The static deployment perspective of a system is described using deployment diagrams. Nodes and their connections are the main components of deployment diagrams. It determines the software deployment strategy on the hardware. It connects the design-created software architecture to the actual system architecture, where the programme will run as a node. Communication channels are used to demonstrate the link because there are several nodes involved.

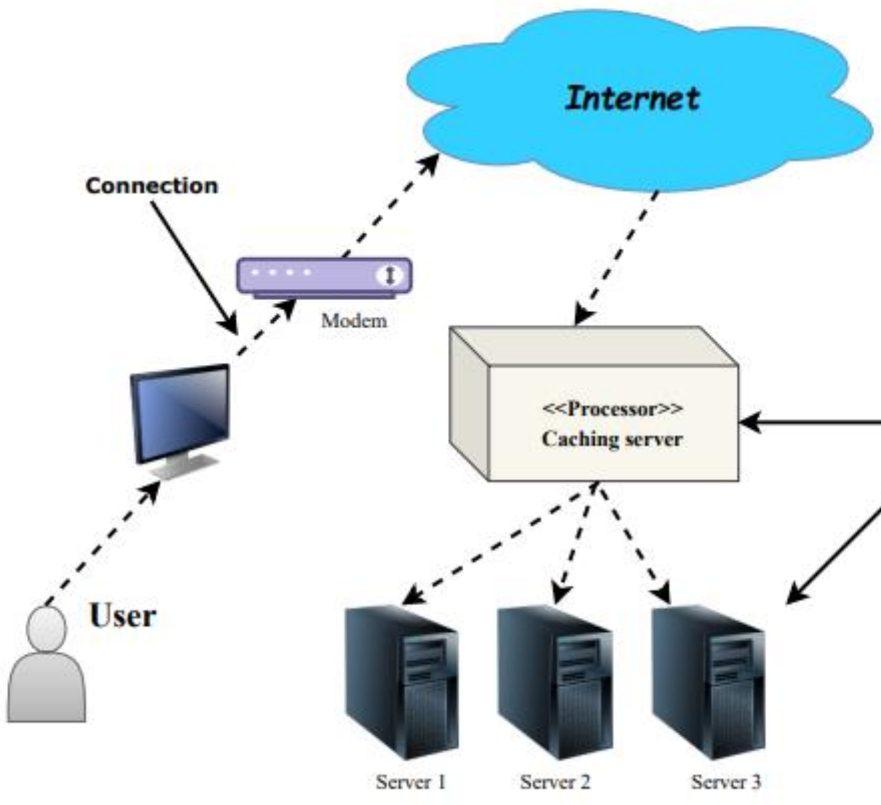


Fig 10: Deployment diagram for Smart Academic Scheduler

4.2.8 STATE CHART DIAGRAM

It describes various system components' statuses. The states are unique to a particular system item or component. A state machine is described in a Statechart diagram. State machine can be described as a device that distinguishes between an object's various states, and these events either internal or external control states. They specify several object states over its lifespan, and events alter these states. Statecharts are helpful for simulating the responsive systems. A system that reacts to internal or external events is known as a reactive system. The transition from one state to another is depicted in a statechart. According to definitions, states are conditions in which objects exist and undergo changes.

The main goal of a statechart diagram is to model an object's lifespan from creation to destruction. For both forward and backward engineering of a system, statechart diagrams are employed. But modelling the reactive system is the fundamental objective. Following are the main purposes of using Statechart diagrams –

- To simulate a system's dynamic component.
- To simulate a reactive system's lifetime.
- To describe different states of an object during its life time.
- Create a state machine to represent an object's states.

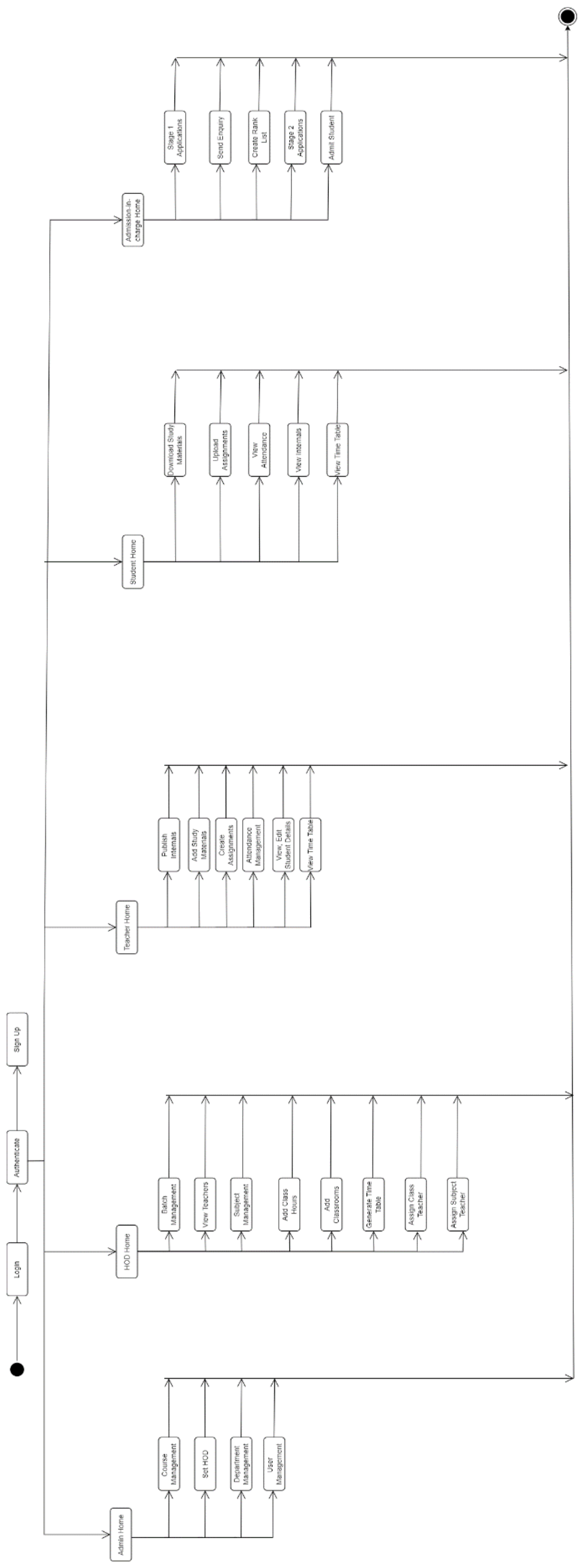
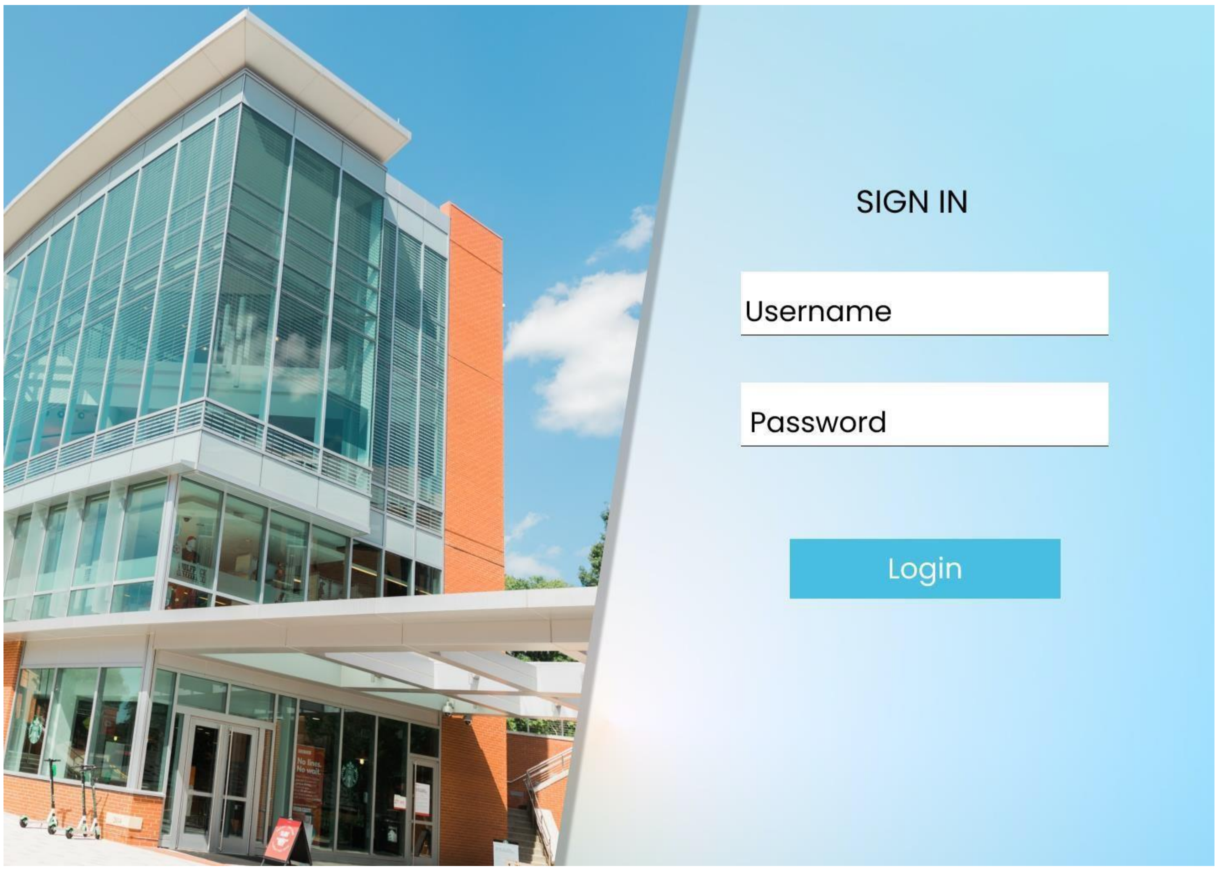


Fig 11: State chart diagram for Smart Academic Scheduler

4.3 USER INTERFACE DESIGN

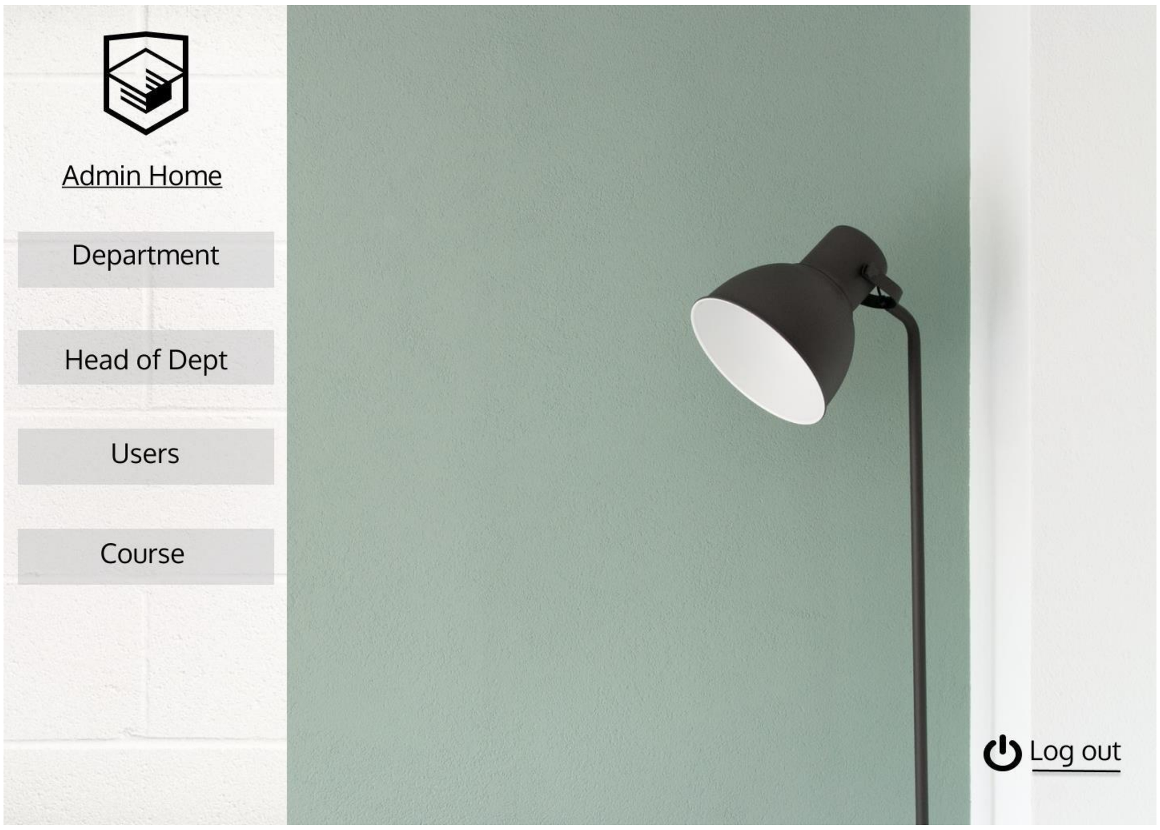
4.3.1-INPUT DESIGN

Form Name : User Login

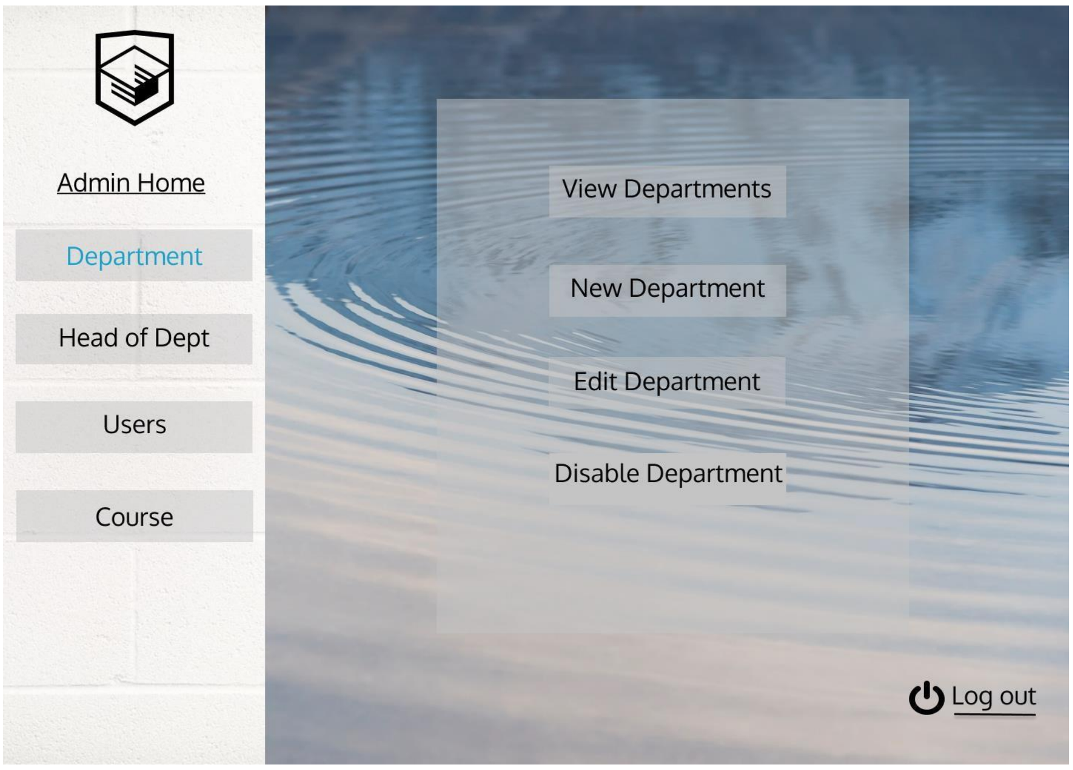


4.3.2 OUTPUT DESIGN

Form Name : Admin Homepage



Form Name : Admin – Department Management



4.4 DATABASE DESIGN

A database is an organised system that has the ability to store information and gives users quick, efficient access to that information. The main objective of every database is its data, which must be secured.

The database design process is divided into two steps. In the first stage, the user demands are ascertained, and a database is made to as clearly as possible satisfy these requirements. This procedure, referred to as information level design, is done separately from every DBMS.

In the second stage, the information level design for the specific DBMS that will be used to build the system in question is transformed into a design. The stage of physical level design is where the properties of the specific DBMS that will be used are determined. The database's data arrangement aims to accomplish the following two main goals.

- Data Integrity
- Data independence

4.4.1 Relational Database Management System (RDBMS)

In a relational paradigm, the database is represented as a collection of relationships. Each relation is represented as a file of records or a table of values. In the formal language of the relational model, a row is referred to as a tuple, a column heading is referred to as an attribute, and the table is referred to as a relation. The tables in a relational database are separated into groups and given distinctive names. In a story, a row represents a group of related values.

Relations, Domains & Attributes

A relation is a table. Tuples are the units of a table's rows. An ordered group of n items is a tuple. Attributes are referred to as columns. Every table in the database has relationships already established between them. This guarantees the integrity of both referential and entity relationships. A group of atomic values make up a domain D. Choosing a data type from which the domain's data values are derived is a typical way to define a domain. To make it easier to understand the values of the domain, it is also helpful to give it a name.

Every value in a relationship is atomic, meaning it cannot be broken down.

Relationships

- Key is used to create table connections. Primary Key and Foreign Key are the two principal keys that are most crucial. Referential integrity as well as entity integrity These are the basics of establishing relationships.
- Entity Integrity enforces that no Primary Key can have null values.
- No Primary Key may contain null values, according to Referential Integrity.
- Referential Integrity: A Primary Key value in the same domain must correspond to each unique Foreign Key value. Super Key and Candidate Keys are additional keys.

4.4.2 Normalization

The simplest possible grouping of data is used to put them together so that future modifications may be made with little influence on the data structures. The formal process of normalising data structures in a way that reduces duplication and fosters integrity. Using the normalisation approach, superfluous fields are removed and a huge table is divided into several smaller ones. Anomalies in insertion, deletion, and updating are also prevented by using it. Keys and relationships are two notions used in the standard style of data modelling. A row in a table is uniquely identified by a key. Primary keys and foreign keys are the two different kinds of keys. A primary key is an element, or set of components, in a database that serves as a means of distinguishing between records from the same table. A column in a database known as a foreign key is used to uniquely identify records from other tables. Up to the third normal form, all tables have been normalised.

It means placing things in their natural form, as the name suggests. By using normalisation, the application developer aims to establish a coherent arrangement of the data into appropriate tables and columns, where names may be quickly connected to the data by the user. By removing recurring groups from data, normalisation prevents data redundancy, which places a heavy demand on the computer's resources. These include:

- ✓ Normalize the data.
- ✓ Choose proper names for the tables and columns.
- ✓ Choose the proper name for the data.

First Normal Form

Each attribute's domain must only include atomic values, and each attribute's value in a tuple must be a single value from that domain, according to the First Normal Form. Using relations as attribute values within tuples or relations within relations is prohibited by 1NF, in other words. Under 1NF, only single atomic or indivisible values are allowed for attribute values. First Normal Form must be used to enter the data. Data can be divided into tables of a similar type in each table to achieve this. Each table is given either a Primary Key or a Foreign Key depending on the requirements of the project. In this process, new relations are created for each nested relation or non-atomic attribute.

Second Normal Form

When the primary key includes many characteristics, no non-key attribute should be functionally dependent on a component of the primary key for relations. This entails creating a new relation for each partial key and dissecting it into its dependent properties. Keep your database's original primary key and any characteristics that are wholly dependent on it. Data that only depends on a small amount of the key can be removed with the help of this approach. A connection is said to be in second normal form if and only if the main key of the connection satisfies all conditions for first normal form and every aspect of the connection's non-primary key totally depends on the main key alone.

Third Normal Form

According to the Third Normal Form, a relation should not have a non-key attribute that is functionally determined by another non-key property or by a group of non-key characteristics. In other words, the main key shouldn't be transitively dependent. The deconstructed non-key attributes are then arranged in relation to the non-key attributes that they functionally determine. This method is used to get rid of anything that isn't completely dependant on the primary key. A relationship is only regarded as being in third normal form if it is in second normal form and should not be dependent on any non-key attributes of any other relationships.

TABLE DESIGN

Table No	: 01
Table Name	: application
Primary Key	: id
Primary Key	: course_id
Table Description	: Stores applications from candidates

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Application id
2	name	Varchar	Not Null	Applicant’s name
3	dob	Date	Not Null	Date of birth
4	gender	Varchar	Not Null	Applicant’s gender
5	address	Varchar	Not Null	Applicant’s address
6	phone	Varchar	Not Null	Applicant’s phone number
7	email	Varchar	Not Null	Applicant’s email id
8	photo	Varchar	Not Null	Applicant’s photo
9	course_id	Int	Not Null	Course applied
10	stage	Varchar	Not Null	Application stage
11	score	Varchar	Not Null	Score for rank-list preparation

Table No	: 02
Table Name	: assignment
Primary Key	: id
Primary Key	: teacher_id
Table Description	: Stores assignment details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Assignment id
2	subject_number	Varchar	Not Null	Subject
3	title	Date	Not Null	Assignment title
4	problem	Varchar	Not Null	Assignment question
5	teacher_id	Int	Not Null	Teacher who made the assignment

6	fromtime	Varchar	Not Null	Assignment start time
7	totime	Varchar	Not Null	Assignment end time
8	status	Varchar	Not Null	Active/inactive status

Table No : 03
Table Name : attendancepercent
Primary Key : id
Primary Key : student_id
Table Description : Stores calculated attendance

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Attendancepercent id
2	student_id	Int	Not Null	Student admission no
3	fromdate	Date	Not Null	Start date
4	todate	Date	Not Null	End date
5	attendancevalue	Varchar	Not Null	Calculated attendance string

Table No : 04
Table Name : attendance
Primary Key : id
Table Description : stores attendance marked by teacher

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Attendance id
2	student_id	Int	Not Null	Student admission no
3	date	Date	Not Null	Date
4	att_str	Varchar	Not Null	Attendance string
5	day	Varchar	Not Null	Day of week

Table No : 05
Table Name : attstring
Primary Key : id
Table Description : stores attendance string for calculation purposes

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Attstring id
2	batch_id	Int	Not Null	Batch
3	day	Varchar	Not Null	Day of week
4	def_string	Varchar	Not Null	Attendance string for calculation purposes

Table No : 06
Table Name : batch
Primary Key : batch_id
Foreign Key : course_id,meeting_time_id,room_id,subject_id,teacher_id
Table Description : stores batch details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	batch_id	Int	Primary Key	Batch id
2	num_class_in_week	Int	Not Null	Number of classes a week
3	class_teacher	Int	Not Null	In-charge teacher
4	semester	Int	Not Null	Semester number
5	status	Varchar	Not Null	Active/Inactive status
6	course_id	Int	Foreign Key	Course
7	meeting_time_id	Int	Foreign Key	For timetable generator
8	room_id	Int	Foreign Key	For timetable generator
9	subject_id	Varchar	Foreign Key	For timetable generator
10	teacher_id	Int	Foreign Key	For timetable generator

Table No	: 07
Table Name	: course
Primary Key	: id
Primary Key	: dept_id
Table Description	: stores course details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Course id
2	course_name	Varchar	Not Null	Course name
3	duration	Varchar	Not Null	Course duration
4	dept_id	Int	Not Null	Department
5	status	Varchar	Not Null	Active/Inactive status
6	fee	Varchar	Not Null	Course fee

Table No	: 08
Table Name	: course_subjects
Primary Key	: id
Foreign Key	: course_id, subject_id
Table Description	: stores subjects for course

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Course_subject id
2	course_id	Int	Foreign Key	Course
3	subject_id	Int	Foreign Key	Subject

Table No	: 09
Table Name	: department
Primary Key	: id
Table Description	: stores department details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Department id
2	name	Varchar	Not Null	Department name
3	descrip	Varchar	Not Null	Department description
4	status	Varchar	Not Null	Active/Inactive status

Table No	: 10
Table Name	: internals
Primary Key	: id
Foreign Key	: student_id,subject_number
Table Description	: stores internal marks of students

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Internals id
2	student_id	Int	Not Null	Student admission no
3	subject_number	Varchar	Not Null	Subject
4	assignment_one	Varchar	Not Null	Marks for assignment one
5	assignment_two	Varchar	Not Null	Marks for assignment two
6	series_one	Varchar	Not Null	Marks for series one
7	series_two	Varchar	Not Null	Marks for series two
8	attendance_mark	Varchar	Not Null	Attendance marks

Table No	: 11
Table Name	: login
Primary Key	: id
Foreign Key	: utype_id,
Table Description	: stores login details of users

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Login id
2	username	Varchar	Not Null	Username
3	password	Varchar	Not Null	Password
4	utype_id	Int	Not Null	User type
5	status	Varchar	Not Null	Active/Inactive status

Table No : 12
Table Name : meetingtime
Primary Key : pid
Table Description : stores class hours

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	pid	Int	Primary Key	Meeting id
2	time	Varchar	Not Null	Time slot
3	day	Varchar	Not Null	Day of week

Table No : 13
Table Name : parent
Primary Key : id
Foreign Key : app_id,
Table Description : stores students’ parents’ details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Parent id
2	fname	Varchar	Not Null	Father’s name
3	mname	Varchar	Not Null	Mother’s name
4	fmail	Varchar	Not Null	Father’s email id
5	mmail	Varchar	Not Null	Mother’s email id
6	fjob	Varchar	Foreign Key	Father’s occupation
7	mjob	Varchar	Foreign Key	Mother’s occupation
8	fphone	Varchar	Foreign Key	Father’s phone number
9	mphone	Varchar	Foreign Key	Mother’s phone number
10	app_id	Int	Foreign Key	Application id

Table No : 14
Table Name : record
Primary Key : id
Foreign Key : app_id
Table Description : stores academic records of students

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Record id
2	tenth	Varchar	Not Null	Percentage in 10th std
3	twelfth	Varchar	Not Null	Percentage in 12th std
4	ug	Varchar	Not Null	Percentage in UG degree
5	certificatetenth	Varchar	Not Null	10th std certificate
6	certificatetwelfth	Varchar	Not Null	12th std certificate
7	certificateug	Varchar	Not Null	UG degree certificate
8	App_id	Int	Not Null	Application id

Table No : 15
Table Name : room
Primary Key : id
Table Description : stores classroom details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Classroom id
2	r_number	Varchar	Not Null	Classroom number
3	Seating_capacity	Int	Not Null	Seating capacity
4	status	Varchar	Not Null	Active/Inactive status

Table No : 16
Table Name : student
Primary Key : id
Foreign Key : app_id, batch_id, login_id
Table Description : stores student details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Student admission number
2	app_id	Varchar	Not Null	Application id
3	batch_id	Int	Not Null	Batch
4	login_id	Int	Not Null	Login id

Table No : 17
Table Name : studymaterial
Primary Key : id
Foreign Key : subject_number, teacher_id
Table Description : stores studymaterial details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Study material id
2	subject_number	Varchar	Not Null	Subject code
3	material	Varchar	Not Null	Study material file
4	title	Varchar	Not Null	Title of study material
5	teacher_id	Int	Not Null	Teacher who uploaded the study material

Table No : 18
Table Name : subject
Primary Key : subject_number
Foreign Key : dept_id
Table Description : stores subject details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	subject_number	Varchar	Primary Key	Subject code

2	subject_name	Varchar	Not Null	Subject name
3	max_numb_students	Varchar	Not Null	For timetable generation
4	subject_type	Varchar	Not Null	Lab or theory
5	dept_id	Int	Not Null	Department

Table No : 19
Table Name : subject_teachers
Primary Key : id
Foreign Key : subject_id, teacher_id
Table Description : stores subject-teacher allocation

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Subject teacher id
2	subject_id	Varchar	Not Null	Subject code
3	teacher_id	Int	Not Null	Teacher id

Table No : 20
Table Name : submission
Primary Key : id
Foreign Key : assignment_id, student_id
Table Description : stores assignment submissions

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Submission id
2	assignment_id	Int	Not Null	Assignment id
3	answer	Varchar	Not Null	Answer file
4	marks	Varchar	Not Null	Marks rewarded
5	student_id	Int	Not Null	Student admission no

Table No : 21
Table Name : teacher
Primary Key : id
Foreign Key : dept_id, login_id
Table Description : stores teacher details

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Teacher id
2	name	Varchar	Not Null	Teacher’s name
3	dob	Date	Not Null	Teacher’s date of birth
4	gender	Varchar	Not Null	Teacher’s gender
5	address	Varchar	Not Null	Teacher’s address
6	phone	Varchar	Not Null	Teacher’s phone no
7	email	Varchar	Not Null	Teacher’s email id
8	dept_id	Int	Not Null	Department
9	photo	Varchar	Not Null	Teacher’s photo
10	qualification	Varchar	Not Null	Highest qualification
11	login_id	Int	Not Null	Login id
12	uid	Varchar	Not Null, Unique	Unique teacher id

Table No : 22
Table Name : utype
Primary Key : id
Table Description : stores user types for login purposes

SLNO	FIELD NAME	TYPE	CONSTRAINS	DESCRIPTION
1	id	Int	Primary Key	Utype id
2	name	Varchar	Not Null	User type

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is the process of closely monitoring the way software is used to see if it functions as intended. Software testing is usually used in conjunction with the words validation and verification. A product, including software, is validated by being examined or evaluated to see if it conforms with all pertinent requirements. Software testing, one sort of verification, uses techniques including reviews, analyses, inspections, and walkthroughs as well. Validation is the process of ensuring that what has been specified matches what the user actually wants.

Other procedures that are typically connected to software testing include static analysis and dynamic analysis. Without actually running the code, static analysis examines the software's source code to look for errors and gather statistics. Dynamic analysis looks at the behavior of software while it is in use to provide information like execution traces, timing profiles, and test coverage specifics.

The activities that make up testing can be planned out in advance and carried out methodically. Individual modules are tested first, followed by the integration of the entire computer-based system. Testing is necessary to achieve the system's testing objectives, thus nothing is complete without it. As testing objectives, a number of rules can be employed.. They are:

Executing a programme during testing is a procedure used to look for errors.

- A excellent test case is one that has a strong chance of revealing a mistake that hasn't
- been found yet.
- A test that finds a mistake that hasn't been noticed is successful.
- For correctness
- For implementation efficiency
- For computational complexity

Testing for correctness is used to ensure that a programme performs precisely as it was intended to. This is considerably harder than it would initially seem, especially for big projects.

5.2 TEST PLAN

The procedures that must be followed in order to fulfil various testing methodologies are suggested in a test plan. The test plan specifies the activities that must be completed. Software developers are responsible for creating a computer programme, as well as any related documentation and data structures. It is always the software developers' job to test each of the program's individual parts to ensure that it serves the intended function. There is an impartial test group in order to address the problems of letting the developer evaluate what they have created (ITG). The precise objectives of testing should be stated in terms of numbers. Consequently, information on the mean time to failure, the cost, and the test plan should be included. The levels of testing include:

- ❖ Unit testing
- ❖ Integration Testing
- ❖ Data validation Testing
- ❖ Output Testing

5.2.1 Unit Testing

The smallest unit of software design, the software component or module, is the focus of unit testing, which focuses verification efforts on it. For the purpose of verifying critical control pathways and identifying flaws inside the module's boundary, the component level design description is used as a guide. the specified untested area for unit testing and the test complexity level. Unit testing focuses on the white box, and multiple components may be tested at once. The modular interface is examined to ensure that data enters and departs the software unit under test in a proper manner. The local data structure is reviewed to ensure that data temporarily stored maintains its integrity during each step of an algorithm's execution

Data flow testing via a module interface must be completed before beginning any further tests. If data cannot enter and exit the system properly, then none of the other tests matter. Examining execution pathways selectively is one of the unit test's key responsibilities. In order to cleanly reroute or stop work when an error does occur, it is necessary to anticipate problem conditions during good design. Boundary testing is the last stage of the testing process for units. Boundary problems are a typical occurrence in software.

When testing individual modules of the Sell-Soft System, each was treated as a separate entity and put through a range of test inputs. Certain bugs were corrected in the modules' internal logic.

5.2.2 Integration Testing

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a programme structure that has been determined by design using unit tested components. The software as a whole is tested. Correction is challenging since the size of the overall programme makes it hard to isolate the reasons. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All of the modules were merged once unit testing was completed in the system to check for any interface inconsistencies. Additionally, variations in programme architectures were eliminated, and a singular programme structure emerged.

5.2.3 Validation Testing or System Testing

This marks the conclusion of the testing procedure. This required comprehensive testing of the entire system, including all forms, codes, modules, and class modules. This kind of testing is frequently referred to as "Black Box" testing and "System tests."

The primary focus of the black box testing approach is on the program's functional requirements. A software engineer may, for instance, use Black Box testing to generate sets of input conditions that would exhaustively test each programme requirement.

5.2.4 Output Testing or User Acceptance Testing

User approval of the system under consideration is tested; in this case, it must meet the needs of the company. When development, the programme should stay in touch with the user and perspective system to make any necessary modifications. This done with respect to the following points:

- ➢ Input Screen Designs,
- ➢ Output Screen Designs,

The aforementioned testing is carried out using a variety of test data. The preparation of test data is essential to the system testing process. The system under investigation is then put to the test using the prepared test data. When testing the system, test data issues are found again and fixed using the testing procedures described above. The fixes are also logged for use in the future.

5.2.5 Automation testing

Software and other computer goods are tested automatically to make sure they abide by tight guidelines. In essence, it's a test to ensure that the hardware or software performs exactly as intended. It checks for errors, flaws, and any other problems that might occur throughout the creation of the product. Any time of day can be used to do automation testing. It looks at the software using scripted sequences. It then summarises what was discovered, and this data can be compared to results from earlier test runs. Advantages of Automated Testing

• Detailed reporting capabilities - Automation testing uses well-crafted test cases for various scenarios. These scripted sequences can be incredibly in-depth, and provide detailed reports that simply wouldn't be possible when done by a human.

• Improved bug detection - One of the main reasons to test a product is to detect bugs and other defects. Automation testing makes this process an easier one. It's also able to analyze a wider test coverage than humans may be able to.

• Simplifies testing - Testing is a routine part of the operations of most SaaS and tech companies. Making it as simple as possible is key. Using automation is extremely beneficial. When automating test tools, the test scripts can be reused

• Speeds up the testing process - Machines and automated technology work faster than humans. Along with improved accuracy, this is why we use them. In turn, this shortens your software development cycles.

• Reduces human intervention - Tests can be run at any time of day, even overnight, without needing humans to oversee it. Plus, when it's conducted automatically, this can also reduce the risk of human error.

5.2.6 Selenium testing

Selenium is an open-source tool that automates web browsers. It provides a single interface that lets you write test scripts in programming languages like Ruby, Java, NodeJS, PHP, Perl, Python, and C#, among others. The Selenium testing tool is used to automate tests across browsers for web applications. It's used to ensure high-quality web applications — whether they are responsive, progressive, or regular. Selenium is an open-source tool.

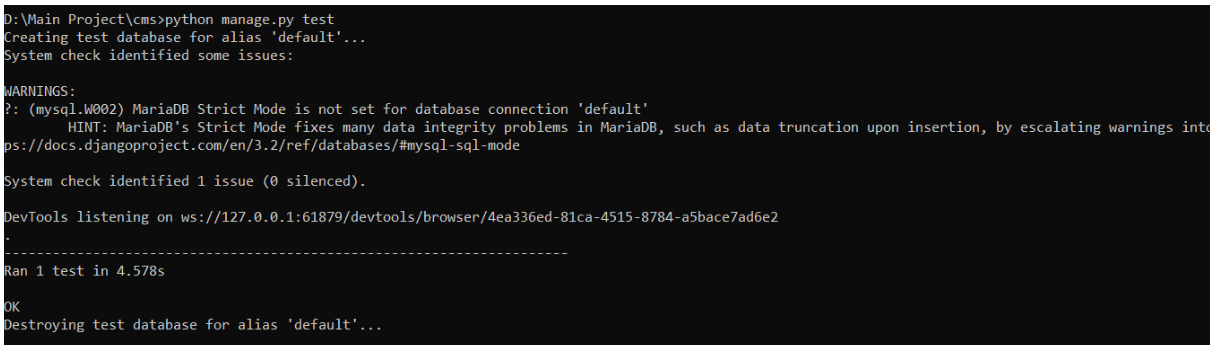
Test Case ID: Fun_1	**Test Designed By: Abhishek Scariya M B**
Test Priority (Low/Medium/High): High	**Test Designed Date:** 18-07-2022
Module Name: Login Screen	**Test Executed By: Ms. Paulin Paul**
Test Title: Verify login with valid username and password	**Test Execution Date:** 18-07-2022
Description: Test the Login Page	

Pre-Condition: User has valid username and password

Test Step	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
Navigation to LoginPage		Login Page should be displayed	Logi n page displ ayed	Pass
Provide ValidUser Name	User Name: admin@sas.in	User should be able to Login	User Logged in and navigated to the dashboard with records	Pass
Provide Valid Password	Password: admin			
Click on Sign In button				
Provide Invalid user name or Password	Username: admin@sas.in Password: admmn	User should not be able to Login	Message for enter valid username or password displayed	Pass
Provide Null Username Id or Password	Email Id: null Password: null			
Click on Sign In button				

Post-Condition: User is validated with database and successfully login into account. TheAccount session details are logged in database

```
from django.test import TestCase

from django.test import LiveServerTestCase

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

class PlayerFormTest(LiveServerTestCase):


 def testform(self):

          selenium = webdriver.Chrome()

          selenium.get('http://127.0.0.1:8000/login')

          player_name = selenium.find_element('name','username')

          player_ps = selenium.find_element('name','password')


  submit= selenium.find_element('name','submit')

  player_name.send_keys('abzrkz@gmail.com')

  player_ps.send_keys('abzrkz')

  submit.send_keys(Keys.RETURN)
```



```
D:\Main Project\cms>python manage.py test
Creating test database for alias 'default'...
System check identified some issues:

WARNINGS:
?: (mysql.W002) MariaDB Strict Mode is not set for database connection 'default'
        HINT: MariaDB's Strict Mode fixes many data integrity problems in MariaDB, such as data truncation upon insertion, by escalating warnings into
ps://docs.djangoproject.com/en/3.2/ref/databases/#mysql-sql-mode

System check identified 1 issue (0 silenced).

DevTools listening on ws://127.0.0.1:61879/devtools/browser/4ea336ed-81ca-4515-8784-a5bace7ad6e2
.
----------------------------------------------------------------------
Ran 1 test in 4.578s

OK
Destroying test database for alias 'default'...
```

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

The conceptual design is developed into a usable system during the project's execution phase. Since it assures users that the system will function as planned and be dependable and accurate, it can be viewed as the most crucial stage in developing a successful new system. Its primary concerns are training and user documentation. Conversion typically occurs either during or following user training. Implementation, which essentially means putting a new system design into operation, is the process of making a recently revised system design operational.

The user department currently handles the heaviest burden, endures the most disturbance, and has the greatest impact on the system as it stands. If it were implemented, there might be chaos and conflict.

The entire process of moving from the old system to the new one is referred to as implementation. The new system could replace a current human or automated system, be completely different, or be improved upon. A reliable system that meets organisational needs must be appropriately implemented. System implementation describes the process of putting the created system into use. This includes every procedure needed to change from the old to the new system. The system can only be placed into operation after comprehensive testing and if it is established that it is operating in compliance with the standards. Employees of the system evaluate the system's feasibility.The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

6.2 IMPLEMENTATION PROCEDURES

Software implementation refers to the complete installation of the package in its intended environment, as well as to the system's functionality and fulfilment of its intended applications. The software development project is frequently commissioned by someone who will not be using it. People have early reservations about the programme, but it's important to watch out for the following to prevent resistance from growing:

- The active user has to understand the advantages of utilising the new system.

- Their faith in the software is increased.
- The user is given the appropriate instruction so that he feels comfortable using the programme.

Before examining the system, the user must be aware that the server software has to be running on the server in order to access the results. The real procedure won't happen if the server object is not active and functioning on the server.

6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer based system, it is essential for the people who will be involved to be confident of theirrole in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

6.2.2 Training on the Application Software

The user will need to receive the essential basic training on computer awareness after which the new application software will need to be taught to them. This will explain the fundamental principles of how to use the new system, including how the screens work, what kind of help is displayed on them, what kinds of errors are made while entering data, how each entry is validated, and how to change the date that was entered. Then, while imparting the program's training on the application, it should cover the knowledge required by the particular user or group to operate the system or a certain component of the system. It's possible that this training will vary depending on the user group and the degree of hierarchy.

6.2.3 System Maintenance

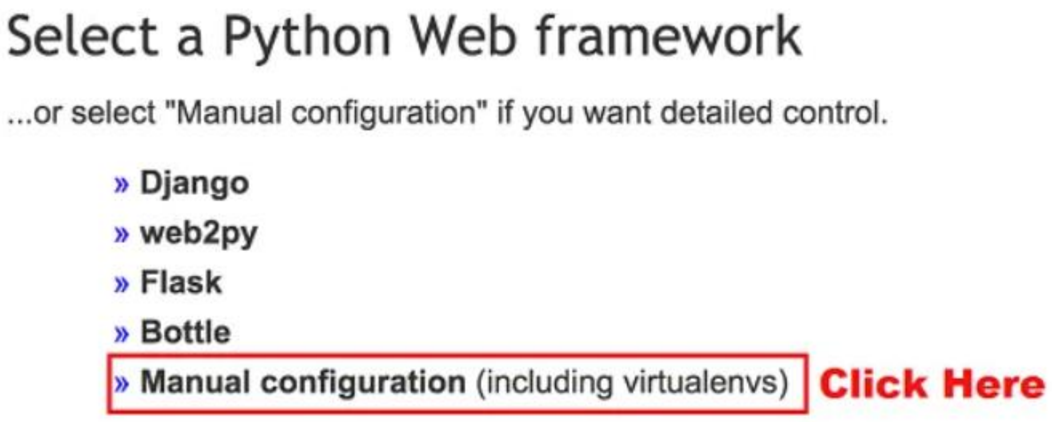
The mystery of system development is maintenance. When a software product is in the maintenance stage of its lifecycle, it is actively working. A system should be properly maintained after it has been effectively installed. An essential part of the software development life cycle is system maintenance. In order for a system to be flexible to changes in the system environment, maintenance is required. Of course, software maintenance involves much more than just "Finding Mistakes".

6.2.4 Hosting

We are using PythonAnywhere to Host the Web app. PythonAnywhere provide free and easy hosting of python projects up to 500 MB of project size. The database is Sqllite3 which is already in the Web app. PythonAnywhere provide a free domain with our registered username

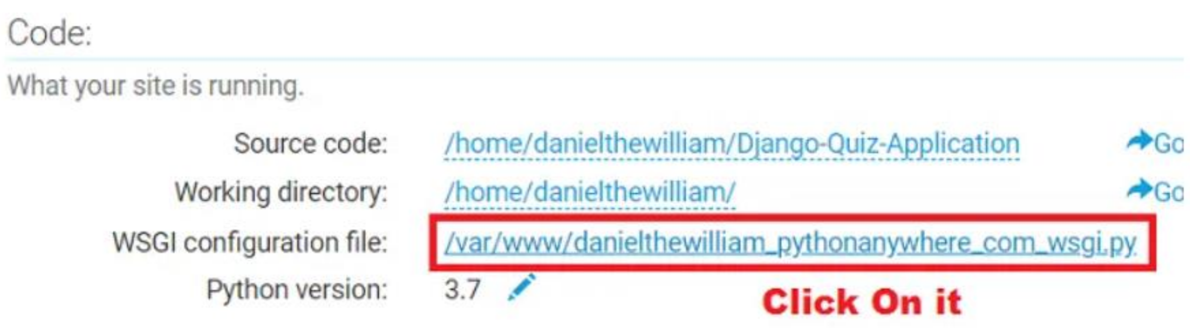
1. Sign up to pythonanywhere.com
2. As our code is ready on GitHub, we will clone it using Bash Console. Create a Bash Console by clicking on the Console Tab. You will see the terminal interface where you need to type git clone command.
3. Create VirtualEnv and Install Django and Dependencie, In your Bash console, create a virtualenv, named after your project name, and choose the version of Python you want to use

   mkvirtualenv --python=/usr/bin/python3.7 mysite-virtualenv
4. Once the Virtual Env is activated. Install Django inside it.
5. Create a Web Application with Manual Config


Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

6. Once your web app is created, scroll down and you need to add some paths. Maker sure you add folders' names accordingly to your project names.
   * Code – /home/myusername/mysite/
   * Virtualenv – /home/username/.virtualenvs./mysite-virtualenv
7. Edit WSGI File to Point our Django Project to Server.


Code:
What your site is running.

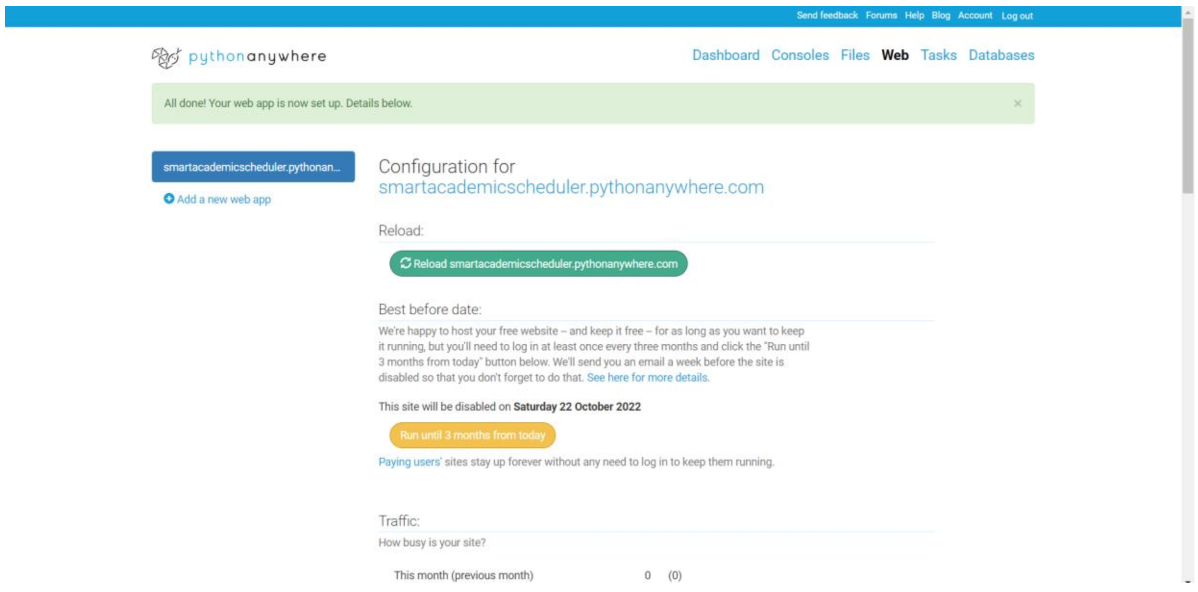
Source code: /home/danielthewilliam/Django-Quiz-Application
Working directory: /home/danielthewilliam/
WSGI configuration file: /var/www/danielthewilliam_pythonanywhere_com_wsgi.py
Python version: 3.7

Click on the WSGI configuration file link. It will take us to the Hosted Django Project WSGI File and add the following code.

```
import os
import sys

# assuming your Django settings file is at
# '/home/myusername/mysite/mysite/settings.py'
path = '/home/myusername/mysite'
if path not in sys.path:
    sys.path.insert(0, path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'

from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
```

That’s it. Now the final step is to go to Web Tab and reload the web app and click on the link of your web app

The URL of the hosted website is: https://smartacademicscheduler.pythonanywhere.com/


Send feedback Forums Help Blog Account Log out
pythonanywhere
Dashboard Consoles Files Web Tasks Databases
smartacademicscheduler.pythonan...
Add a new web app
Reload smartacademicscheduler.pythonanywhere.com
Run until 3 months from today

All done! Your web app is now set up. Details below.

Configuration for
smartacademicscheduler.pythonanywhere.com

Reload:

Best before date:

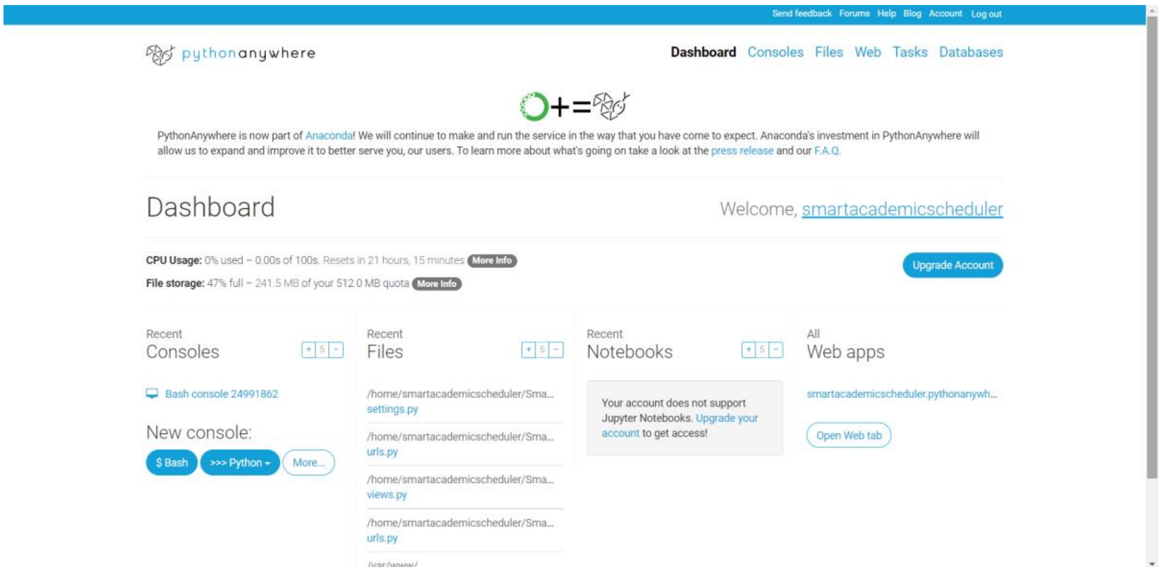
We're happy to host your free website – and keep it free – for as long as you want to keep it running, but you'll need to log in at least once every three months and click the "Run until 3 months from today" button below. We'll send you an email a week before the site is disabled so that you don't forget to do that. See here for more details.

This site will be disabled on **Saturday 22 October 2022**

Paying users' sites stay up forever without any need to log in to keep them running.

Traffic:
How busy is your site?

This month (previous month) 0 (0)


Send feedback Forums Help Blog Account Log out
pythonanywhere
Dashboard Consoles Files Web Tasks Databases

PythonAnywhere is now part of Anaconda! We will continue to make and run the service in the way that you have come to expect. Anaconda's investment in PythonAnywhere will allow us to expand and improve it to better serve you, our users. To learn more about what's going on take a look at the press release and our F.A.Q.

Dashboard

Welcome, smartacademicscheduler

CPU Usage: 0% used – 0.00s of 100s. Resets in 21 hours, 15 minutes More Info

File storage: 47% full – 241.5 MB of your 512.0 MB quota More Info

/home/smartacademicscheduler/Sma... settings.py

/home/smartacademicscheduler/Sma... urls.py

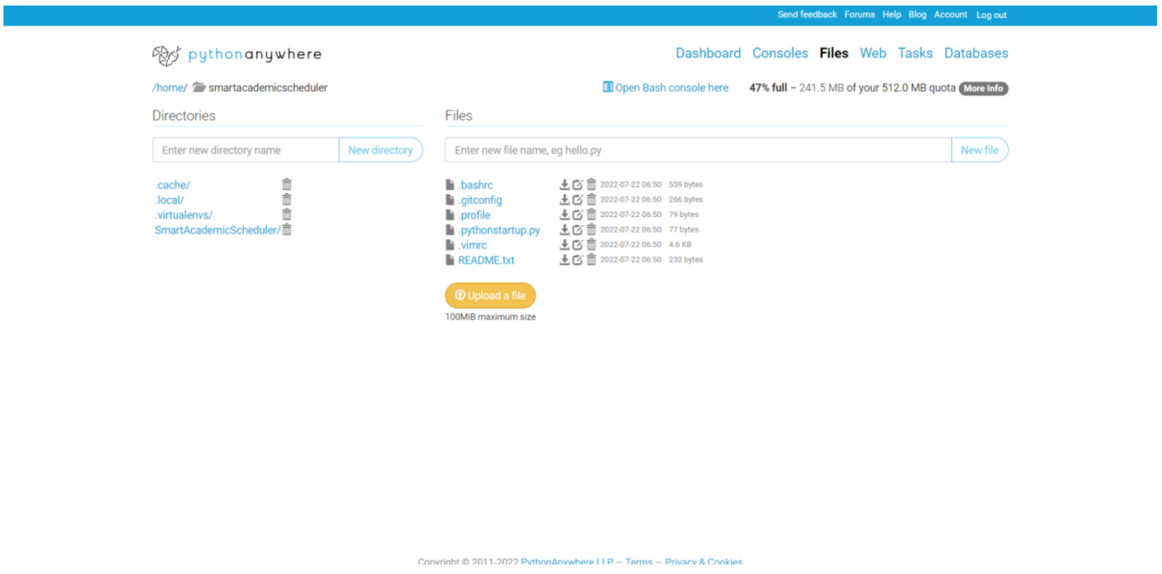
/home/smartacademicscheduler/Sma... views.py

/home/smartacademicscheduler/Sma... urls.py

/var/www/

Your account does not support Jupyter Notebooks. Upgrade your account to get access!

All Web apps
smartacademicscheduler.pythonanywh...
Open Web tab


Send feedback Forums Help Blog Account Log out
pythonanywhere
Dashboard Consoles Files Web Tasks Databases

/home/ smartacademicscheduler

Open Bash console here

47% full – 241.5 MB of your 512.0 MB quota More Info

Directories
Enter new directory name
New directory

.cache/
.local/
.virtualenvs/
SmartAcademicScheduler/

Files
Enter new file name, eg hello.py
New file

.bashrc	2022-07-22 06:50	559 bytes
.gitconfig	2022-07-22 06:50	266 bytes
.profile	2022-07-22 06:50	79 bytes
.pythonstartup.py	2022-07-22 06:50	77 bytes
.vimrc	2022-07-22 06:50	4.6 KB
README.txt	2022-07-22 06:50	232 bytes

Upload a file
100MiB maximum size

Copyright © 2011-2022 PythonAnywhere LLP — Terms — Privacy & Cookies

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The current system is old fashioned and most of the processes are done offline and manually. The proposed system introduces automation of a major part of the admission process. It also expands beyond and assist in day to day class activities of students and teachers. This system also helps to reduce the burden on the staff to create a working timetable. It cleverly employs genetic algorithm to generate a timetable. It also manages the attendance of student. Subject-wise attendance of each student can be accessed through it.

7.2 FUTURE SCOPE

The system can be improved and optimized to create more efficient timetables. It can be developed further to consider many more input factors like teacher workload. This can introduce the "Smart Academic Scheduler" as a product which many institutions can rely on to schedule their academic activities.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”, 2009.
- Roger S Pressman, “*Software Engineering*”, 1994.
- PankajJalote, “So*ftware engineering*: a precise approach”, 2006.
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

- www.w3schools.com
- https://docs.djangoproject.com/en/4.0/
- https://stackoverflow.com/
- www.agilemodeling.com/artifacts/useCaseDiagram.html
- https://realpython.com/tutorials/django/

CHAPTER 9

APPENDIX

9.1 Sample Code

Login.html

```
{%load static%}
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<!-- Main CSS-->
<link rel="stylesheet" type="text/css" href="{% static 'css/main.css' %}">

<link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<title>Login</title>
</head>
<body>
<header class="header-area header-sticky">
<div class="container">
<div class="row">
<div class="col-12">
<nav class="main-nav">
<!-- ***** Logo Start ***** -->
<a href="index.html"
class="logo">EDU EXPERT
</a>

</nav>
</div>
</div>
</div>
</header>
<section class="material-half-bg">
<div class="cover"></div>
</section>
<section class="login-content">
<div class="logo">
<h1>Smart Academic Scheduler</h1>
</div>
{% for message in messages %}

<div class="alert alert-dismissible alert-danger">
<a class="close" href="#" data-dismiss="alert">×</a>
{{ message }}
</div>

{% endfor %}
<div class="login-box">

<form class="login-form" action="/dashboard/" method="post">
```

```
{% csrf_token %}
<h3 class="login-head"><i class="fa fa-lg fa-fw fa-user"></i>SIGN IN</h3>
<div class="form-group">
<label class="control-label">USERNAME</label>
<input class="form-control" type="text" placeholder="Email" name="username"
autofocusautocomplete="off" required>
</div>
<div class="form-group">
<label class="control-label">PASSWORD</label>
<input class="form-control" type="password" placeholder="Password"
name="password"autocomplete="off" required>
</div>
<div class="form-group">
<div class="utility">
<!-- <div class="animated-checkbox">
<label>
<input type="checkbox"><span class="label-text">Stay Signed in</span>
</label>
</div> -->
<p class="semibold-text mb-2"><a href="#" data-toggle="flip">Forgot Password ?</a></p>
<p class="semibold-text mb-2"><a href="/">Home</a></p>
</div>
</div>
<div class="form-group btn-container">
<button class="btn btn-primary btn-block"><i class="fa fa-sign-in fa-lg fa-fw"></i>SIGN IN</button>
</div>
</form>
<form class="forget-form" action="/dashboard/" method="post">
<h3 class="login-head"><i class="fa fa-lg fa-fw fa-lock"></i>Forgot Password ?</h3>
<div class="form-group">
<label class="control-label">EMAIL</label>
<input class="form-control" type="text" placeholder="Email" required>
</div>
<div class="form-group btn-container">
<button class="btn btn-primary btn-block"><i class="fa fa-unlock fa-lg fa-fw"></i>RESET</button>
</div>
<div class="form-group mt-3">
<p class="semibold-text mb-0"><a href="#" data-toggle="flip"><i class="fa fa-angle-left fa-fw"></i>
Backto Login</a></p>
</div>
</form>
</div>
</section>
<!-- Essential javascripts for application to work-->
<script src="{% static 'js/jquery-3.3.1.min.js' %}"></script>
<script src="{% static 'js/popper.min.js' %}"></script>
<script src="{% static 'js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/main.js' %}"></script>
<!-- The javascript plugin to display page loading on top-->
<script src="{% static 'js/plugins/pace.min.js' %}"></script>
<script type="text/javascript">
// Login Page Flipbox control
$('.login-content [data-toggle="flip"]').click(function() {
$('.login-
box').toggleClass('flipped');return
false;

    });
</script>
</body>
</html>
```

Models.py

```
from django.db import models

#--------------------SCHEDULER---------------------------

import random as rnd
import datetime
from django.db import models
import math
from django.core.validators import MinValueValidator, MaxValueValidator
from django.contrib.auth.models import AbstractUser
from django.db.models.signals import post_save, post_delete
from datetime import timedelta, date


time_slots = (
    ('9:30 - 10:30', '9:30 - 10:30'),
    ('10:30 - 11:30', '10:30 - 11:30'),
    ('11:30 - 12:30', '11:30 - 12:30'),
    ('12:30 - 1:30', '12:30 - 1:30'),
    ('2:30 - 3:30', '2:30 - 3:30'),
    ('3:30 - 4:30', '3:30 - 4:30'),
    ('4:30 - 5:30', '4:30 - 5:30'),
)
DAYS_OF_WEEK = (
    ('Monday', 'Monday'),
    ('Tuesday', 'Tuesday'),
    ('Wednesday', 'Wednesday'),
    ('Thursday', 'Thursday'),
    ('Friday', 'Friday'),
    ('Saturday', 'Saturday'),
)

POPULATION_SIZE = 9
NUMB_OF_ELITE_SCHEDULES = 1
TOURNAMENT_SELECTION_SIZE = 3
MUTATION_RATE = 0.1

#--------------------SCHEDULER---------------------------

# Create your models here.

class login(models.Model):
    username=models.CharField(max_length=30)
    password=models.CharField(max_length=20)
    utype_id=models.BigIntegerField()
    status=models.CharField(max_length=5,default=1)

    class Meta:
        db_table = "login"

class utype(models.Model):
    name=models.CharField(max_length=15)

    class Meta:
        db_table = "utype"

class department(models.Model):
    name=models.CharField(max_length=50)
    descrip=models.CharField(max_length=300)
    status=models.CharField(max_length=5,default=1)

    class Meta:
        db_table = "department"



class teacher(models.Model):
    name=models.CharField(max_length=50)
    dob=models.DateField(null=True)
    gender=models.CharField(max_length=15)
    address=models.CharField(max_length=200)
    email=models.CharField(max_length=30)
    phone=models.CharField(max_length=20)
    dept_id=models.BigIntegerField(default=0)
    photo=models.CharField(max_length=100)
    qualification=models.CharField(max_length=30)
    login_id=models.BigIntegerField()

    #Timetable generator
    uid = models.CharField(max_length=10,default="None")

    def __str__(self):
        return f'{self.uid} {self.name}'
    class Meta:
        db_table = "teacher"

class application(models.Model):
    name=models.CharField(max_length=50)
    dob=models.DateField()
    gender=models.CharField(max_length=15)
    address=models.CharField(max_length=200)
    phone=models.CharField(max_length=20)
    email=models.CharField(max_length=30)
    photo=models.CharField(max_length=100)
    course_id=models.BigIntegerField(default=0)
    stage=models.CharField(max_length=15,default='1')
    score=models.DecimalField(max_digits=4, decimal_places=2,default=0)
    class Meta:
        db_table = "application"

class parent(models.Model):
    fname=models.CharField(max_length=50)
    mname=models.CharField(max_length=50)
    fmail=models.CharField(max_length=30)
    mmail=models.CharField(max_length=30)
    fjob=models.CharField(max_length=30)
    mjob=models.CharField(max_length=30)
    fphone=models.CharField(max_length=20)
    mphone=models.CharField(max_length=20)
    app_id=models.BigIntegerField(default=0)
    class Meta:
        db_table = "parent"

class record(models.Model):
    tenth=models.DecimalField(max_digits=4, decimal_places=2)
    twelfth=models.DecimalField(max_digits=4, decimal_places=2)
    ug=models.DecimalField(max_digits=4, decimal_places=2,null=True)
    certificatetenth=models.CharField(max_length=100)
    certificatetwelfth=models.CharField(max_length=100)
    certificateug=models.CharField(max_length=100)
    app_id=models.BigIntegerField(default=0)
    class Meta:
        db_table = "record"

class student(models.Model):
    app_id=models.BigIntegerField(default=0)
    batch_id=models.CharField(max_length=100,default=0)
    class Meta:
        db_table = "student"



class Room(models.Model):
    r_number = models.CharField(max_length=10)
    seating_capacity = models.IntegerField(default=0)

    #additional fields for project
    status=models.CharField(max_length=5,default=1)
    def __str__(self):
        return self.r_number
    class Meta:
        db_table = "room"

class MeetingTime(models.Model):
    pid = models.BigIntegerField(primary_key=True)
    time = models.CharField(max_length=50, choices=time_slots, default='11:30 - 12:30')
    day = models.CharField(max_length=15, choices=DAYS_OF_WEEK)

    def __str__(self):
        return f'{self.pid} {self.day} {self.time}'
    class Meta:
        db_table = "meetingtime"

class subject(models.Model):
    subject_number = models.CharField(max_length=10, primary_key=True) #subject code
    subject_name = models.CharField(max_length=40)
    max_numb_students = models.CharField(max_length=65)
    teachers = models.ManyToManyField(teacher)
    dept_id=models.BigIntegerField(default=0)
    #additional fields for project
    subject_type=models.CharField(max_length=10,default="theory")


    def __str__(self):
        return f'{self.subject_number} {self.subject_name}'
    class Meta:
        db_table = "subject"

class course(models.Model):
    #-----------------------------------------
    course_name=models.CharField(max_length=50)
    subjects = models.ManyToManyField(subject)
    #-----------------------------------------
    duration=models.CharField(max_length=5)
    dept_id=models.BigIntegerField()
    status=models.CharField(max_length=5,default=1)
    fee=models.CharField(max_length=7,default=25000)
    class Meta:
        db_table = "course"
    @property
    def get_subjects(self):
        return self.subjects

    def __str__(self):
        return self.course_name
    class Meta:
        db_table = "course"

class batch(models.Model):
    batch_id = models.CharField(max_length=25, primary_key=True)
    course = models.ForeignKey(course, on_delete=models.CASCADE,default=1)
    num_class_in_week = models.IntegerField(default=0)
    subject = models.ForeignKey(subject, on_delete=models.CASCADE, blank=True, null=True)
    meeting_time = models.ForeignKey(MeetingTime, on_delete=models.CASCADE, blank=True,
                null=True)
    room = models.ForeignKey(Room,on_delete=models.CASCADE, blank=True, null=True)
    teacher = models.ForeignKey(teacher, on_delete=models.CASCADE, blank=True, null=True)

    class_teacher=models.BigIntegerField(default=0)
    semester=models.BigIntegerField(default=0)
    status= models.CharField(max_length=25,default="batched")

    def set_room(self, room):
        batch = batch.objects.get(pk = self.batch_id)
        batch.room = room
        batch.save()

    def set_meetingTime(self, meetingTime):
        batch = batch.objects.get(pk = self.batch_id)
        batch.meeting_time = meetingTime
        batch.save()

    def set_teacher(self, teacher):
        batch = batch.objects.get(pk=self.batch_id)
        batch.teacher = teacher
        batch.save()
    class Meta:
        db_table = "batch"


class attendence(models.Model):
    student_id=models.CharField(max_length=10)
    date=models.DateField()
    day=models.CharField(max_length=50)
    att_str=models.CharField(max_length=200)
    class Meta:
        db_table = "attendence"

class attstring(models.Model):
    batch_id = models.CharField(max_length=25)
    day=models.CharField(max_length=50)
    def_string=models.CharField(max_length=200)
    class Meta:
        db_table = "attstring"
```

views.py

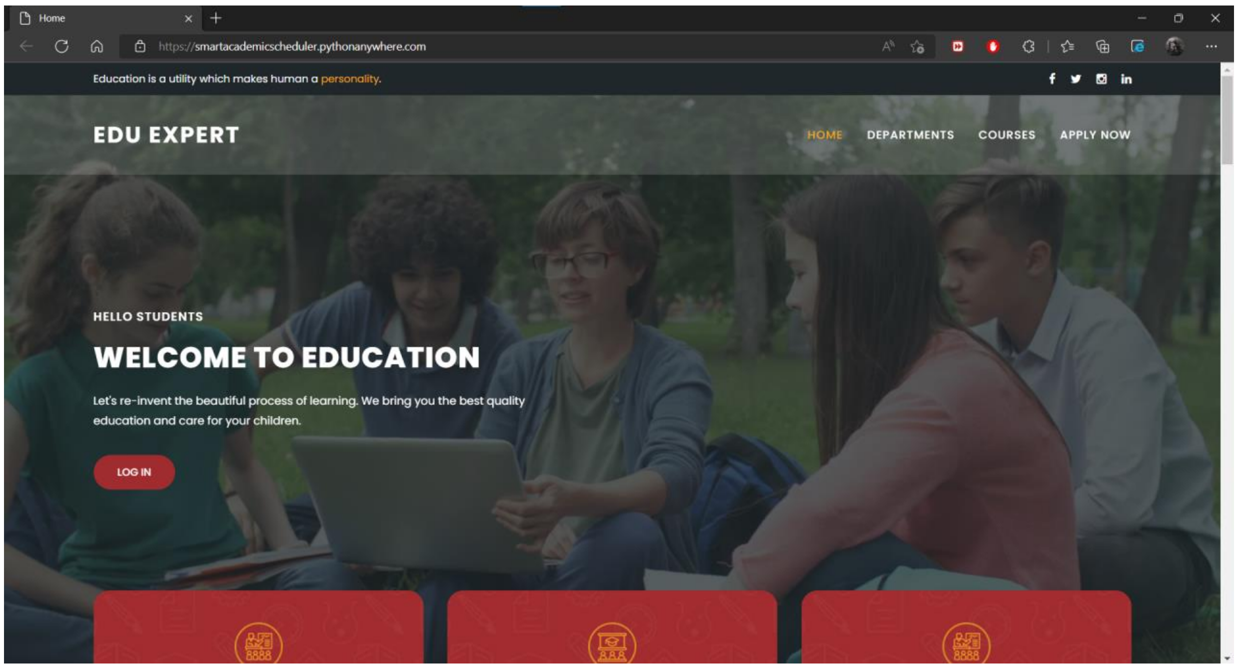
```
from django.shortcuts import render, redirect
from django.http import HttpResponse, HttpResponseRedirect
from app.models import login , utype, department, course, teacher, application, parent,
recordfrom django.contrib import messages
from django.core.mail import send_mail
from django.core.files.storage import
FileSystemStoragefrom django.db.models import Q
import cms.settings


from django.contrib.auth.models import User

def user_login(request):
    dat=login.objects.all()
    dat2=utype.objects.all()
    flag=0
    un=request.POST.get("username")
    pw=request.POST.get("password")
    for d in dat:
        if d.username==un and d.password==pw  and d.status=='1':
            request.session['id']=d.id
            flag =1
            id = request.session['id']
            data=login.objects.get(id=id)
            for e in dat2:
                if d.utype_id==e.id:
                    request.session['utype']=d.utype_id
                    if e.name=='admin':
                        return redirect('/dash/')
                    elif e.name=='teacher':
                        dat3=teacher.objects.get(login_id=d.id)
                        if dat3.name=='no data':
                            return render(request, 'teacherregister.html')
                        else:
                            return redirect('/dash2/')
                    elif e.name=='hod':
                        return redirect('/dash3/')
                    elif e.name=='admission':
                       return redirect('/dash4/')
                    else:
                        return render(request, 'login.html')
    if flag==0:
        messages.error(request,'Username or Password is incorrect!')
        return redirect('/login/')
```

```
def test_form(request):
    return render(request, 'form-samples-copy.html')
def dash(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    return render(request, 'dashboard.html')

def dash2(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    return render(request, 'teacher.html')

def dash3(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    return render(request, 'hod.html')

def dash4(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    return render(request, 'incharge.html')

def dashboardref(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    uid=request.session.get('utype')
    print(uid)
    if (uid == 1):
        data=application.objects.all()
        return render(request, 'dashboard.html')
    elif (uid == 2):
        return render(request, 'teacher.html')
    elif (uid == 3):
        return render(request, 'hod.html')
    else:
        return render(request, 'incharge.html')

def deptadd(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    return render(request, 'deptadd.html')

def deptaddval(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    name = request.POST['name']
    descrip = request.POST['descrip']
    data=department.objects.all()
    for i in data:
        if i.name == name:
            messages.warning(request, 'Department already exists... Insertion failed!')
            return redirect('/deptadd/')
    dept=department.objects.create(name=name,descrip=descrip)
    dept.save()
    messages.success(request, 'Department added successfully...!')
    return redirect('/deptadd/')

def deptview(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    data=department.objects.all()
    dept={
        "dept_no":data
    }
    return render(request,"deptview.html",dept)

def depteditview(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    data=department.objects.all()
    dept={
        "dept_no":data
    }
    return render(request,"depteditview.html",dept)

def deptedit(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    data=department.objects.all()
    id=request.POST.get("id")
    for d in data:
        if int(id)==d.id:
          dept={"d":d}
          return render(request,"deptedit.html",dept)

def deptupdate(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    id=request.POST.get("id")
    data=department.objects.get(pk=id)
    data2=course.objects.filter(dept_id=id)
    data.name=request.POST.get("name")
    data.status=request.POST.get("status")
    data.descrip=request.POST.get("descrip")
    if data.status=='0':
        for c in data2:
            c.status=data.status
            c.save()
    data.save()

    messages.success(request, 'Department updated successfully...!')
    return redirect("/depteditview/")

def logout(request):
    if request.session.is_empty():
        return HttpResponseRedirect('/login')
    request.session.flush()
    return HttpResponseRedirect('/login')

def courseadd(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    data=department.objects.all()
    dept={
        "dept_no":data
    }
    return render(request,"courseadd.html",dept)

def courseaddval(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    name = request.POST['name']
    duration = request.POST['duration']
    dept_id = request.POST['dept_id']
    fee = request.POST['fee']
    data2 = course.objects.all()
    for i in data2:
        if i.course_name == name:
            messages.warning(request, 'Course already exists..! Insertion failed!')
            return redirect('/courseadd/')

    c=course.objects.create(course_name=name,duration=duration,dept_id=dept_id,fee=fee)
    c.save()
    messages.success(request, 'Course added successfully...!')
    return redirect('/courseadd/')

def courseview(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    data1=course.objects.all()
    data2=department.objects.all()
    data={
        "course":data1,
        "dept":data2
    }


    return render(request,"courseview.html",data)

def courseeditview(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    data1=course.objects.all()
    data2=department.objects.all()
    data={
        "course":data1,
        "dept":data2
    }


    return render(request,"courseeditview.html",data)

def courseedit(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    data=course.objects.all()
    data2=department.objects.all()
    cid=request.POST.get("cid")
    for c in data:
        if int(cid)==c.id:
          dat={"c":c,"d":data2}
          return render(request,"courseedit.html",dat)

def courseupdate(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    id=request.POST.get("id")
    data=course.objects.get(pk=id)
    data.name=request.POST.get("name")
    data.duration=request.POST.get("duration")
    data.fee = request.POST['fee']
    data.dept_id=request.POST.get("dept")
    data.status=request.POST.get("status")
    data.save()
    messages.success(request, 'Course updated successfully...!')
    return redirect("/courseeditview/")
```

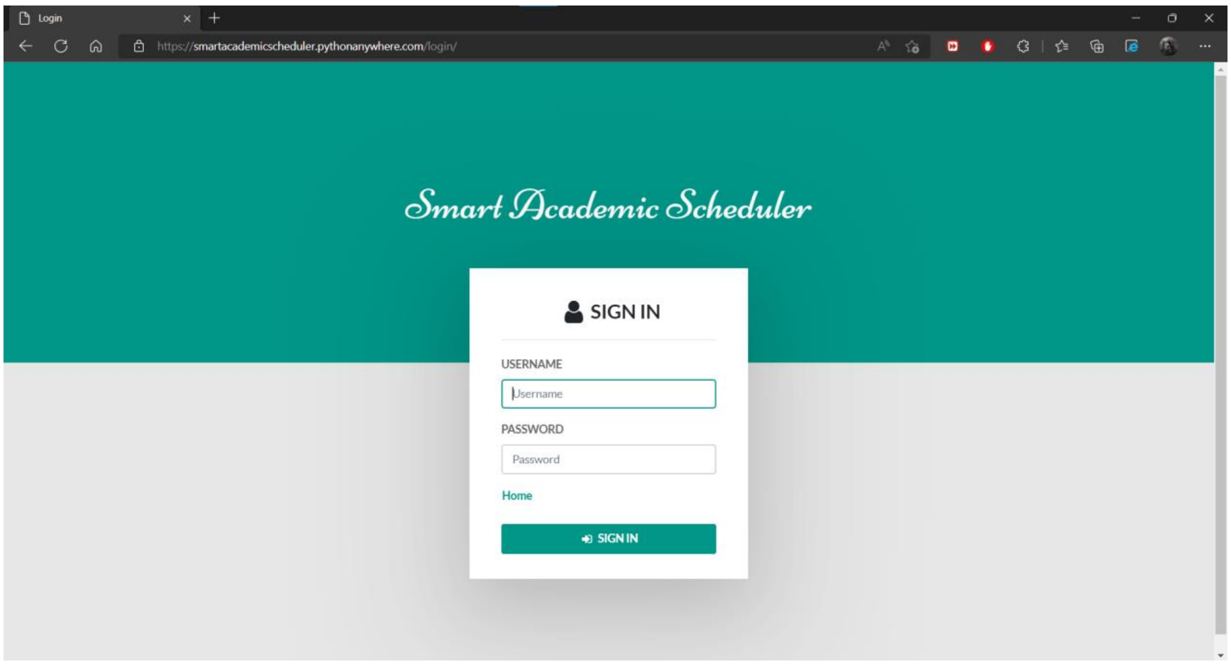
```
def useradd(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    return render(request, 'useradd.html')

def teacheradd(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    data=department.objects.all()
    password = User.objects.make_random_password(length=14,
                    allowed_chars="abcdefghjkmnpqrstuvwxyz01234567889")
    dept={
        "dept_no":data,
        "password":password
    }
    return render(request, 'teacheradd.html',dept)

def admissionadd(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    password = User.objects.make_random_password(length=14,
                    allowed_chars="abcdefghjkmnpqrstuvwxyz01234567889")
    dat={
        "password":password
    }
    return render(request, 'admissionadd.html',dat)

def admissiongen(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    email = request.POST['email']
    password = request.POST['password']
    mailto = request.POST['mailto']
    data=login.objects.all()
    for d in data:
        if d.username == email:
            messages.warning(request, 'User already exists...!')
            return redirect('/useradd/')
    l=login.objects.create(username=email,password=password,utype_id=4,status='1')
    subject = 'Login credentials'
    message = f'Welcome to Smart Academic Scheduler. Here are your login credentials to manage
                student admission.\nUsername: {email}\nPassword: {password}'
    email_from = cms.settings.EMAIL_HOST_USER
    recipient_list = [mailto]
    send_mail( subject, message, email_from, recipient_list )
    l.save()
    login_id=l.id
    messages.success(request, 'Admission in-charge added successfully...!')
    return redirect('/useradd/')

def teachergen(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    email = request.POST['email']
    password = request.POST['password']
    dept_id = request.POST['dept_id']
    uid = request.POST['uid']
    data=login.objects.all()
    for d in data:
        if d.username == email:
            messages.warning(request, 'User already exists...!')
            return redirect('/useradd/')
    l=login.objects.create(username=email,password=password,utype_id=2,status='1')
    l.save()
    login_id=l.id
    t=teacher.objects.create(name='no data',phone='no data',dob=None,gender='no data',address='no
        data',email=email,dept_id=dept_id,qualification='no data',login_id=login_id,uid=uid)
    t.save()
    subject = 'Login credentials'
    message = f'Welcome to Smart Academic Scheduler. Here are your login credentials.\nUsername:
                {email}\nPassword: {password}'
    email_from = cms.settings.EMAIL_HOST_USER
    recipient_list = [email]
    send_mail( subject, message, email_from, recipient_list )
    messages.success(request, 'Teacher added successfully...!')
    return redirect('/useradd/')

def userview(request):
    if request.session.is_empty():
        messages.error(request,'Session has expired, please login to continue!')
        return HttpResponseRedirect('/login')
    data1=login.objects.filter(~Q(utype_id='1'))
    data2=utype.objects.all()
    data={
        "login":data1,
        "utype":data2
    }
    return render(request, 'userview.html',data)
```

9.2 Screen Shots

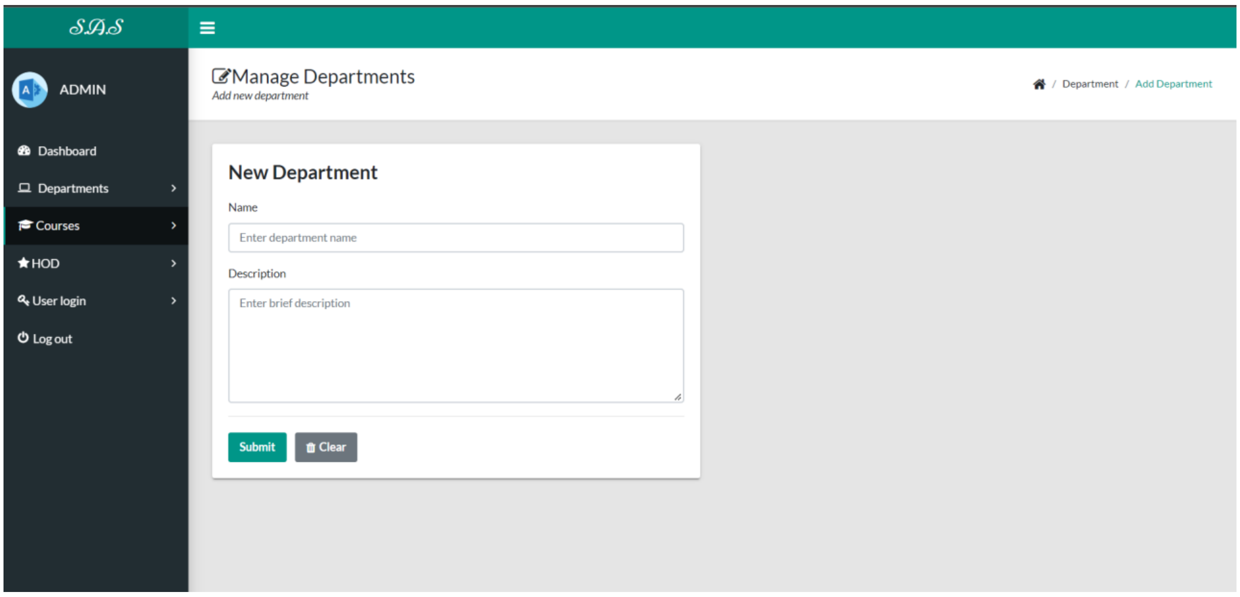
Home page


Home
https://smartacademicscheduler.pythonanywhere.com
Education is a utility which makes human a personality.
EDU EXPERT
HOME
DEPARTMENTS
COURSES
APPLY NOW
HELLO STUDENTS
WELCOME TO EDUCATION
Let's re-invent the beautiful process of learning. We bring you the best quality education and care for your children.
LOG IN

Login page



Admin – Add Department



Manage Departments

Add new department

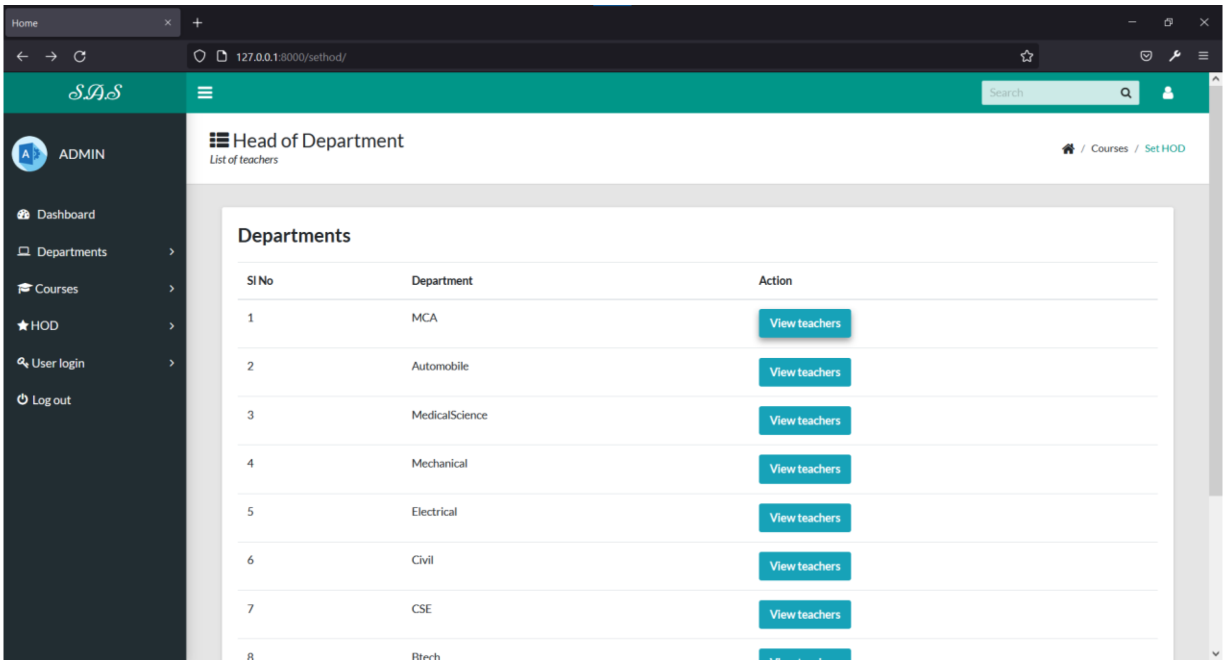
Department / Add Department

New Department

Name: Enter department name

Description: Enter brief description

Set HOD



127.0.0.1:8000/sethod/

Head of Department

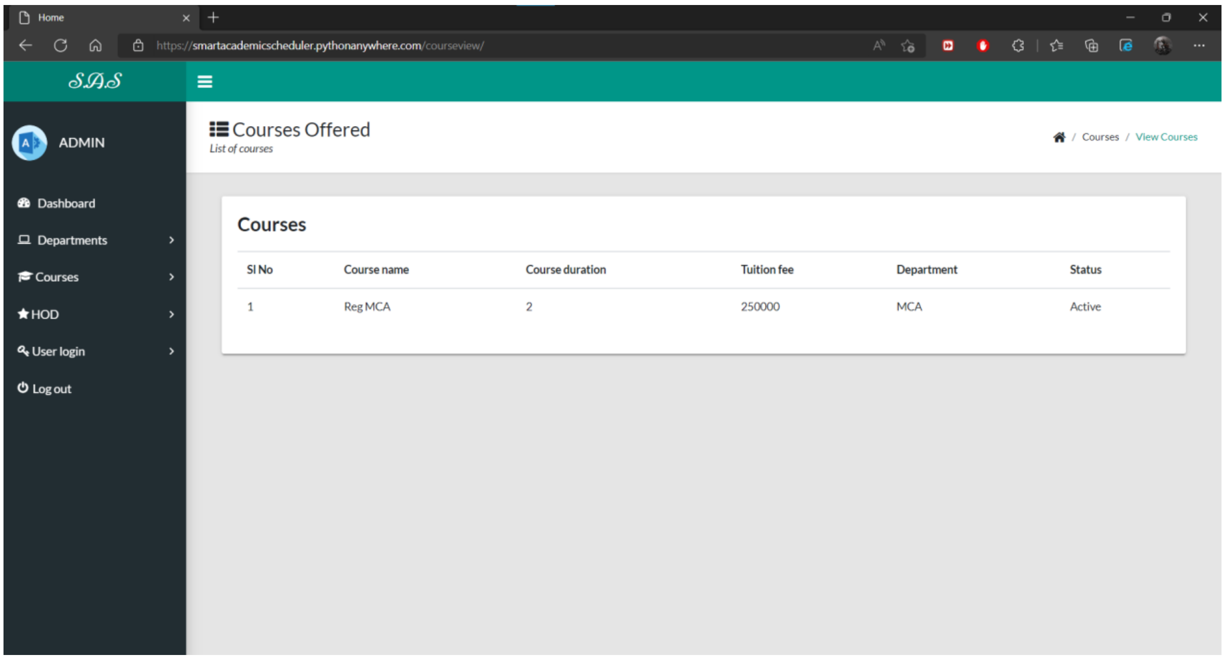
List of teachers

Courses / Set HOD

Departments

Sl No	Department	Action
1	MCA	View teachers
2	Automobile	View teachers
3	MedicalScience	View teachers
4	Mechanical	View teachers
5	Electrical	View teachers
6	Civil	View teachers
7	CSE	View teachers
8	Btech	

View Courses


Home
https://smartacademicscheduler.pythonanywhere.com/courseview/
SAS
ADMIN
Dashboard
Departments
Courses
HOD
User login
Log out
Home / Courses / View Courses

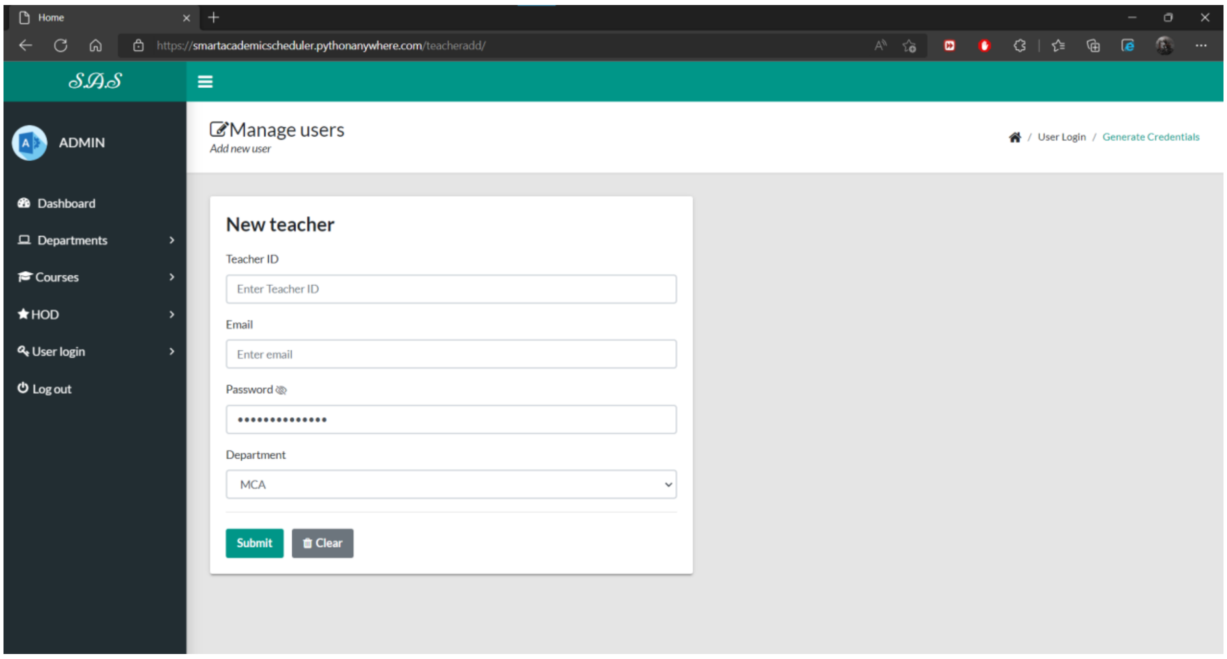
Courses Offered

List of courses

Courses

Sl No	Course name	Course duration	Tuition fee	Department	Status
1	Reg MCA	2	250000	MCA	Active

Add Teachers


Home
https://smartacademicscheduler.pythonanywhere.com/teacheradd/
SAS
ADMIN
Dashboard
Departments
Courses
HOD
User login
Log out
Home / User Login / Generate Credentials
Submit
Clear

Manage users

Add new user

New teacher

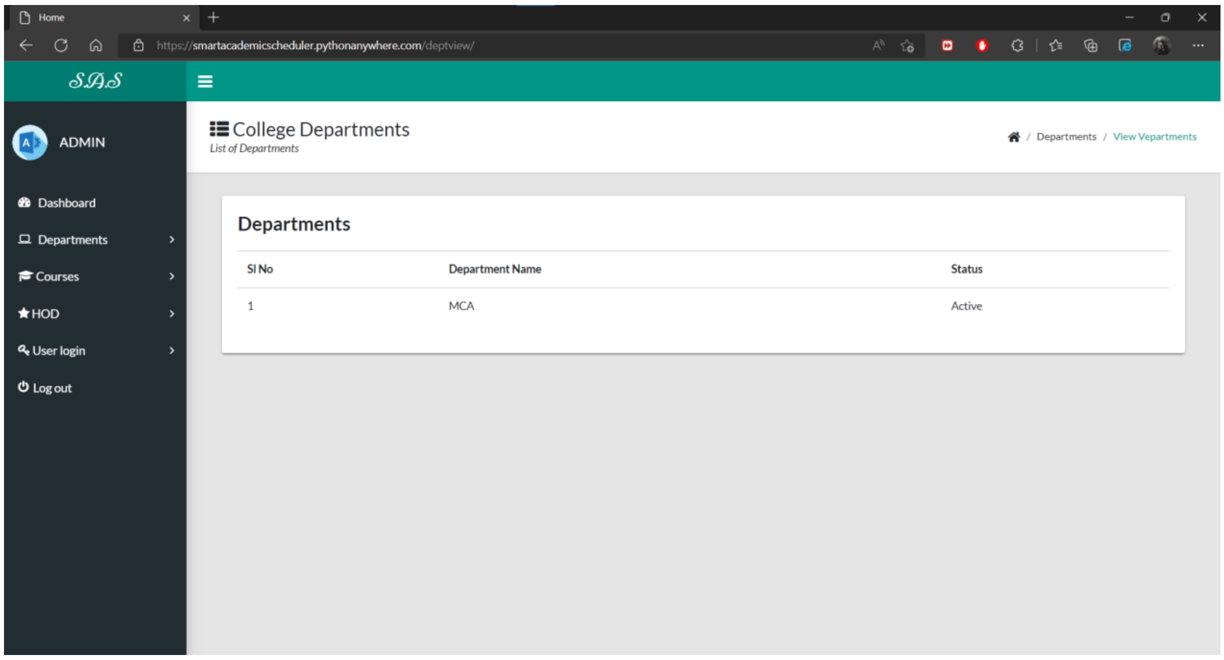
Teacher ID: Enter Teacher ID

Email: Enter email

Password: ••••••••••••••

Department: MCA

View Departments


Home
https://smartacademicscheduler.pythonanywhere.com/deptview/
SAS
ADMIN
Dashboard
Departments
Courses
HOD
User login
Log out
Departments / View Vepartments

College Departments

List of Departments

Departments

Sl No	Department Name	Status
1	MCA	Active