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 analysing 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 recognised, and the system is seen as a whole. The different procedures may be linked to the outputs from the organisations. Understanding the issue, identifying the pertinent and important variables, evaluating and synthesising 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

A feasibility study is conducted to determine if the project will, upon completion, fulfil the objectives of the organisation in relation to the labour, effort, and time invested in it. A feasibility study enables the developer to predict the project's usefulness and potential future. A system proposal's workability, which includes the influence on the organisation, capacity to satisfy user demands, and efficient use of resources, is the basis for a feasibility study. As a result, before a new application is accepted for development, it often undergoes a feasibility assessment.

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,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

A high-level Python web framework called Django promotes quick prototyping and logical, elegant design. It was created by seasoned programmers and handles a lot of the pain associated with Web development, freeing you up to concentrate on building your app without having to invent the wheel. It is open source and free.

A group of components known as a web framework offers a standardised method for quickly and simply creating websites. Django's main objective is to make it simpler to create intricate database-driven websites. Django is used by some well-known websites, including PBS, Instagram, Disqus, the Washington Times, Bitbucket, and Mozilla.

Django (/ˈdʒæŋgoʊ/ jang-goh) is a free and open source web application framework, written in Python. A web framework is a collection of elements that makes it quicker and simpler to create websites.

A similar set of elements is required whenever you establish a website: a method for handling user authentication (signing up, signing in, and signing out), a management panel for your website, forms, a method for uploading files, etc.

For your benefit, other people long ago realised that web developers have similar issues when creating a new website. As a result, they collaborated to create frameworks, one of which is Django, that provide you with ready-made components to utilise. Frameworks exist to help you avoid having to create the wheel from scratch and to reduce some of the overhead involved in creating a new website.

<u>Why do you need a framework?</u>

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 urlresolver in Django does this task. The term urlresolver 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 authorised 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 utilising 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."

3.3.2 MySQL

Oracle Corporation created, distributed, and provided support for MySQL, the most well-known Open Source SQL database management system. The most recent details regarding MySQL software are available on the MySQL website.

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

A systematic collection of data is called a database. It might be anything, such as a straightforward grocery list, a photo gallery, or the enormous quantity of data in a business network. A database management system, such as MySQL Server, is required to add, retrieve, and process data saved in a data base. Database management systems, whether used as stand-alone programmes or as a component of other applications, are vital to computing since computers are efficient at processing vast volumes of data.

- **MySQL databases are relational.**

Instead of placing all the data in one huge warehouse, a relational database keeps the data in individual tables. Physical files that are designed for speed include the database structures. The logical model provides a versatile programming environment with objects like databases, tables, views, rows, and columns. One-to-one, one-to-many, unique, compulsory or optional, and "pointers" across distinct tables are a few examples of the rules you might build up to regulate the connections between various data fields. The database upholds these regulations, ensuring that your application never encounters inconsistent, duplicate, orphan, out-of-date, or missing data thanks to a well-designed database. MySQL stands for "Structured Query Language" with the SQL prefix.

The most used standard language for accessing databases is SQL. Depending on your programming environment, you could explicitly input SQL (for example, to produce reports), incorporate SQL statements into other languages' code, or utilise a language-specific API that obscures the SQL syntax. By way of the ANSI/ISO SQL Standard, SQL is defined. Since its inception in 1986, the SQL standard has undergone multiple revisions. In this document, "SQL92" refers to the 1992 standard, "SQL: 1999" to the 1999 standard, and "SQL: 2003" to the most recent version of the standard. The SQL Standard as it exists at any one moment is referred to as "the SQL standard."

- **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.**

   You ought to give it a shot if that is what you're after. In addition to your other apps, web servers, and other software, MySQL Server may function smoothly on a desktop or laptop while requiring little to no maintenance. You may modify the settings to utilise all the RAM, CPU power, and I/O capacity if you dedicate an entire computer to MySQL.

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

   The MySQL Database Software is a client/server system made up of a multi-threaded SQL server that supports several backends, a number of distinct client applications and libraries, administrative tools, and a broad variety of application programming interfaces (APIs). Additionally, we provide MySQL Server as an integrated multi-threaded library that you can connect into your programme to create a standalone offering that is smaller, quicker, and simpler to operate.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Any engineering system or product's development process begins with design. A creative process is design. The secret to an efficient system is a decent design. The act of using different methodologies and concepts to specify a process or a system in enough detail to allow for its physical implementation is referred to as "design." One way to describe it is as the process of using different methodologies and concepts to specify a device, a process, or a system in enough detail to allow for its physical realization. Regardless of the development paradigm that is employed, software design forms the technical core of the software engineering process. The architectural detail needed to construct a system or product is developed through the system design. This programme has also through the best possible design phase, fine tuning all efficiency, performance, and accuracy levels, as in the case of any systematic technique. A user-oriented document is converted into a document for programmers or database staff throughout the design process. The two stages of system design development are logical design and physical design.

4.2 UML DIAGRAM

A common language known as UML is used to describe, visualise, build, and record the software system artefacts. 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.

Unified Modeling Language is known as UML. Compared to other popular programming languages like C++, Java, COBOL, etc., UML is unique. A visual language called UML is used to create software designs. A general-purpose visual modelling language for software system visualisation, specification, construction, and documentation is what UML is known as. UML is not just used to represent software systems, despite the fact that this is its most common application. It is also used to simulate systems that are not software-based.

is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) andUML 1.0 specification draft was proposed 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

A use case diagram is a visual representation of the interactions between system components. A approach for identifying, outlining, and organising system requirements is called a use case. The word "system" here refers to a thing that is being created or run, like a website for mail-order goods sales and services. UML (Unified Modeling Language), a standard language for the modelling of real-world objects and systems, uses use case diagrams.

The planning of general requirements, the validation of a hardware design, the testing and debugging of a software product in development, the creation of an online help reference, or the completion of a job focused on customer support are all examples of system objectives. For instance, use cases in a product sales context can involve customer service, item ordering, catalogue updating, and payment processing. There are four elements in a use case diagram.

- 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

- A use case's naming is highly significant. The name should be selected in a way that makes it clear what functions are being performed.
- Give the actors names that fit them.
- Clearly depict links and dependencies in the diagram.
- As the primary function of the diagram is to establish the needs, avoid attempting to incorporate all kinds of relationships.
- When necessary, take notes to help you remember some crucial details.
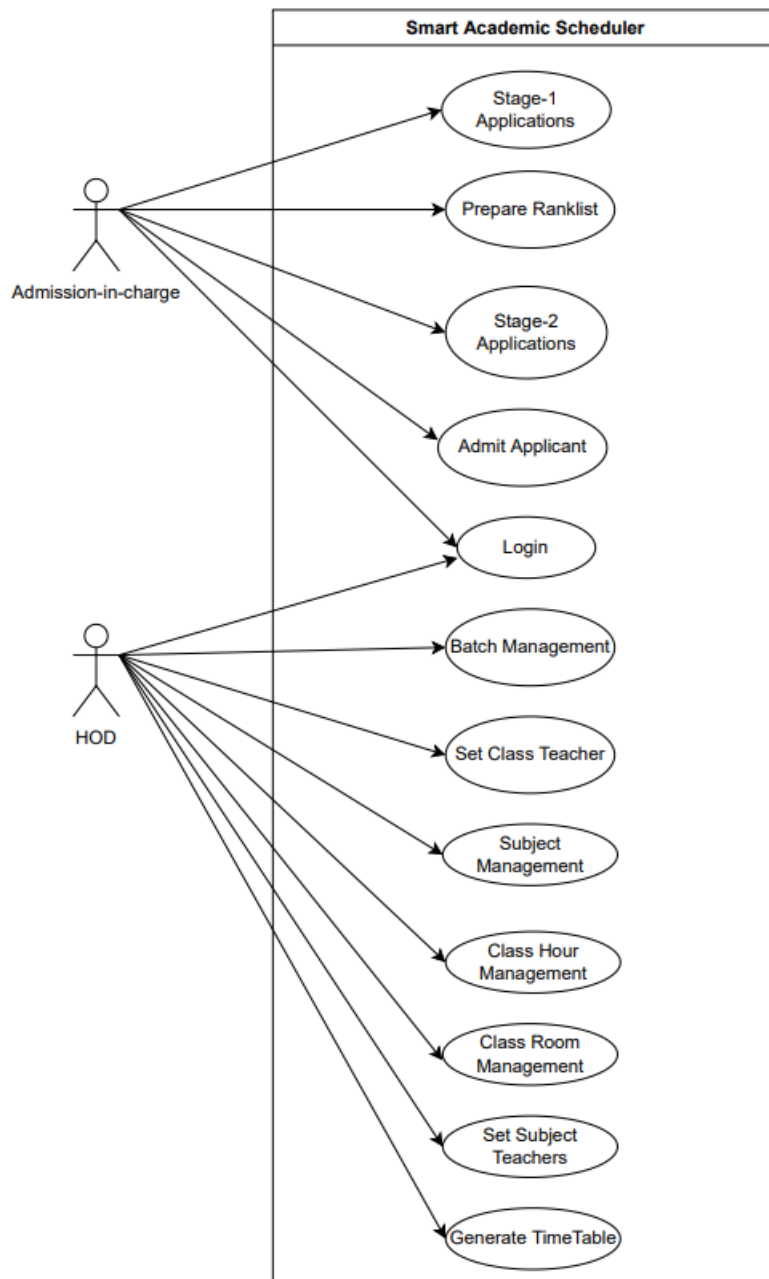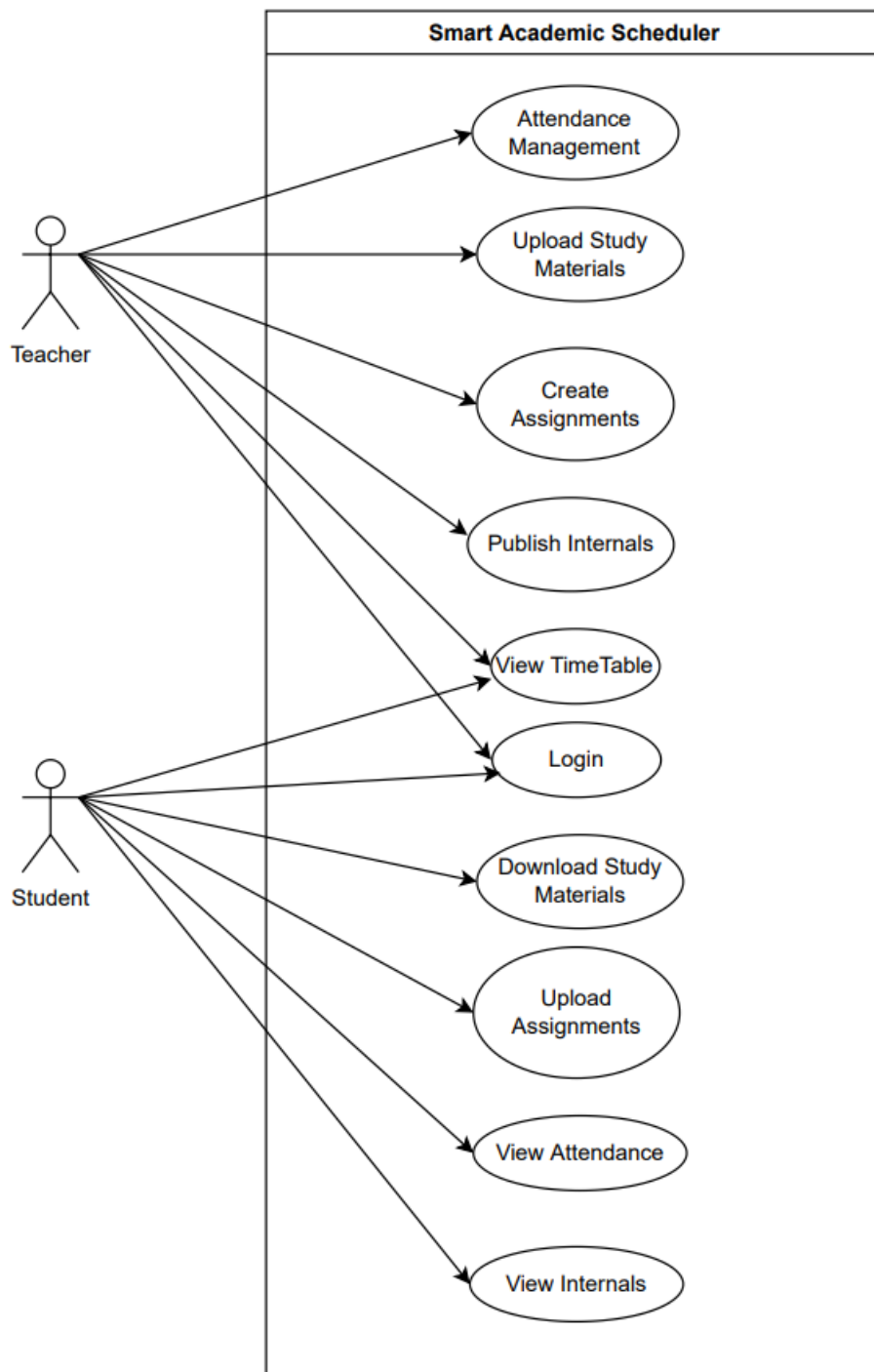
4.2.2 SEQUENCE DIAGRAM

A sequence diagram essentially shows how objects interact with one another sequentially, or the order in which these interactions occur. A sequence diagram can also be referred to as event diagrams or event scenarios. Sequence diagrams show the actions taken by the components of a system in chronological sequence. Businesspeople and software engineers frequently use these diagrams to record and comprehend the requirements for new and current systems.

Sequence Diagram Notations –

i.   **Actors –** In a UML diagram, an actor represents a particular kind of role in which it communicates with the system's objects. An actor is always beyond the purview of the system that we want to use the UML diagram to represent. We employ actors to portray a variety of roles, including those of human users and other outside subjects. In a UML diagram, an actor is represented using a stick person notation. In a sequence diagram, there might be several actors.

ii. **Lifelines –** A named piece that shows a specific participant in a sequence diagram is called a lifeline. In essence, a lifeline represents each incident in a sequence diagram. The lifeline components in a sequence diagram 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 – In the UML, we utilise guards to model circumstances. When we need to limit the flow of communications under the guise of a condition being satisfied, we utilise them. Software engineers rely on guards to inform them of the limitations imposed by a system or specific procedure.

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.
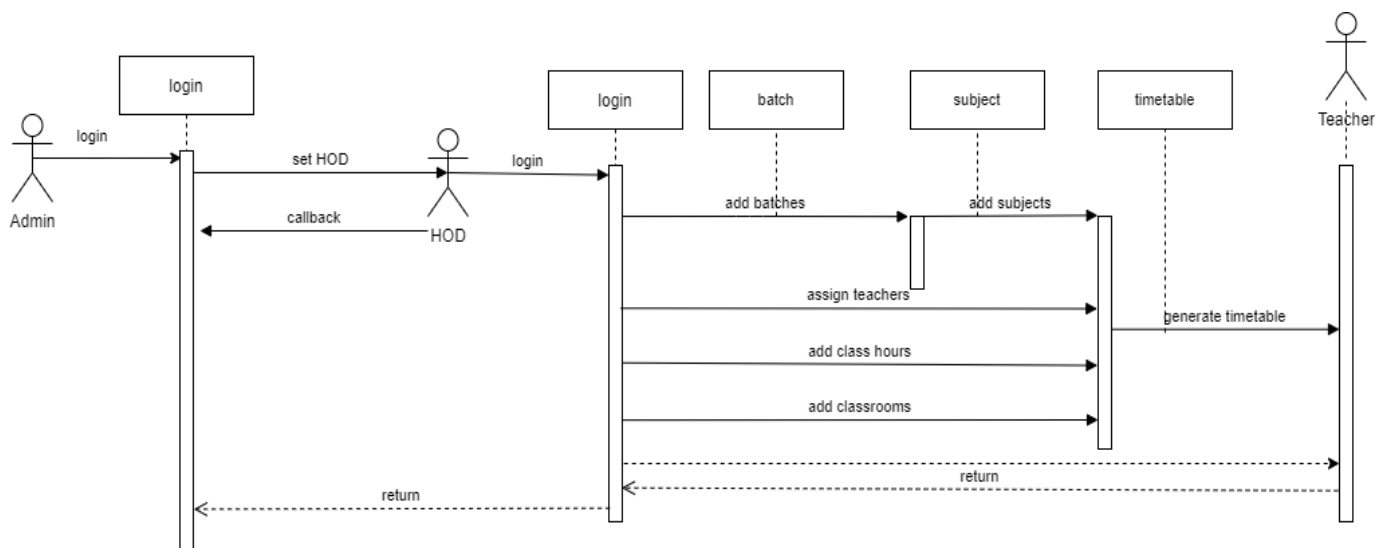
4.2.3 ACTIVITY DIAGRAM

The dynamic features of the system are described in the activity diagram. An activity diagram is essentially a flowchart that shows how one action leads to another. The activity may be thought of as a system's operation. The control flow originates in from one procedure to the next. This flow may be parallel, contemporaneous, or branched. Activity diagrams depict several flow control mechanisms employing various components like forks, affix, etc..

Activity diagrams are used to build the executable system utilising forward and reverse engineering approaches, as well as to visualise the dynamic nature of a system. The message portion is the only item the activity diagram is lacking. No message flow from one activity to another is shown. Occasionally, an activity diagram is used in place of a flowchart. The diagrams are not flowcharts, despite their appearance.

HOD
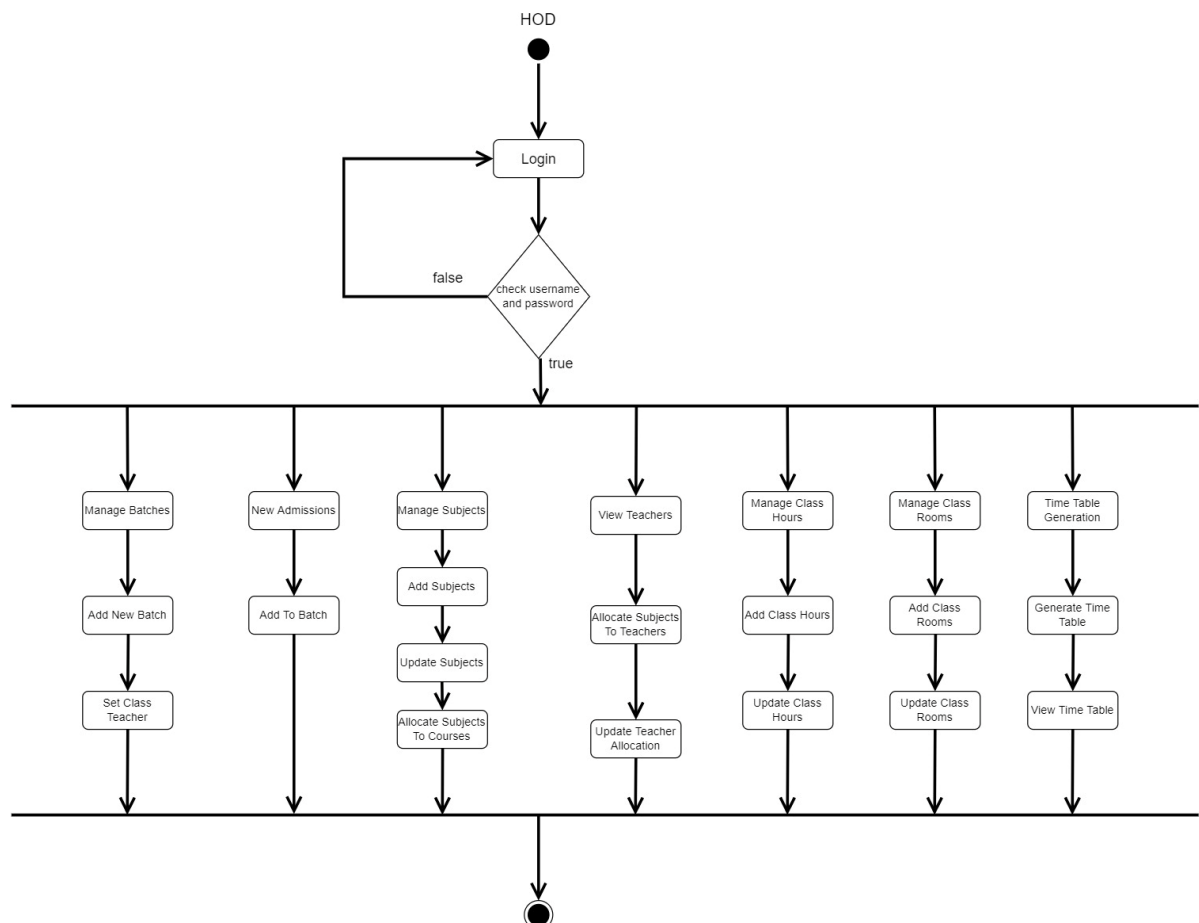
ADMIN
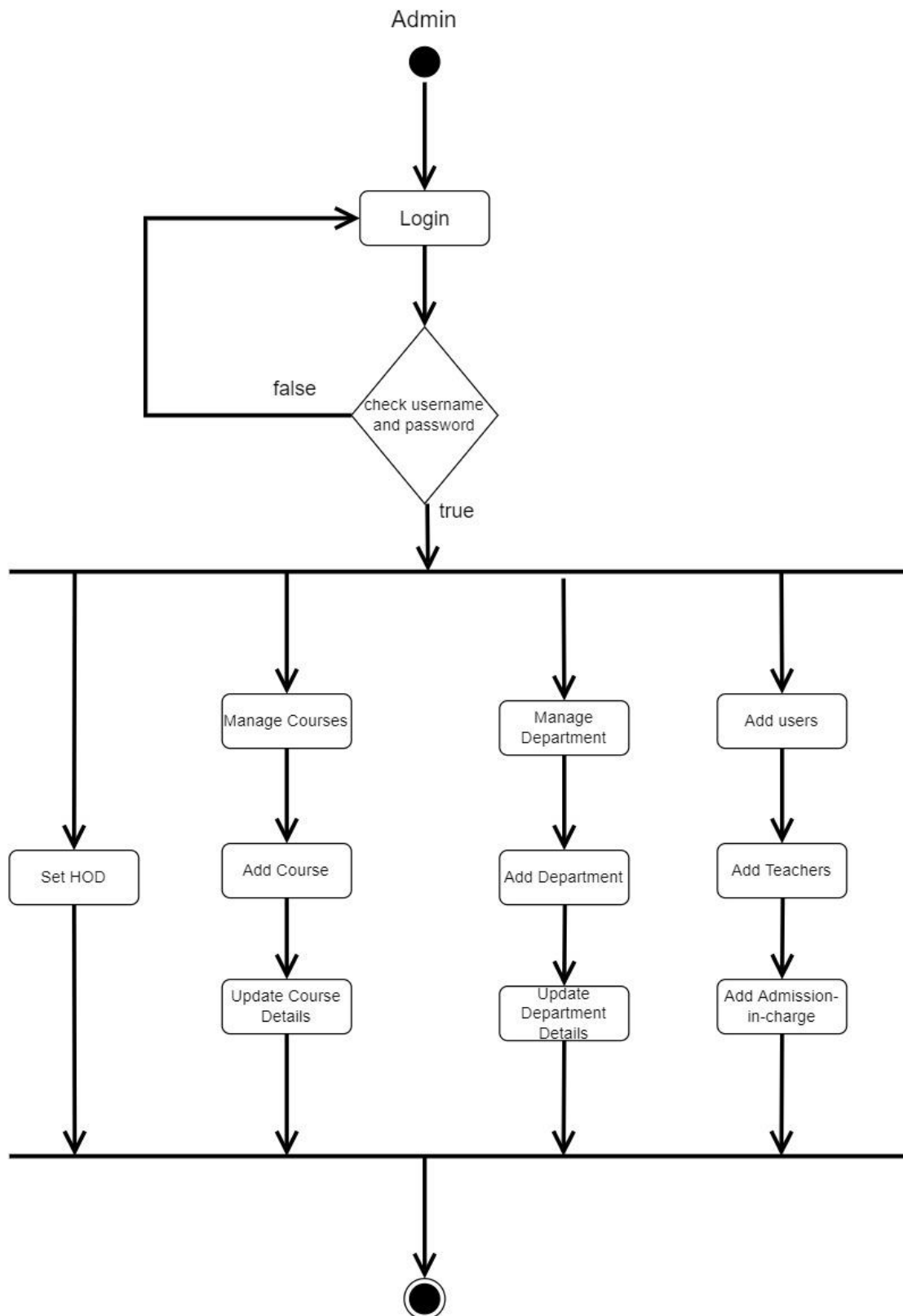
ADMISSION-IN-CHARGE
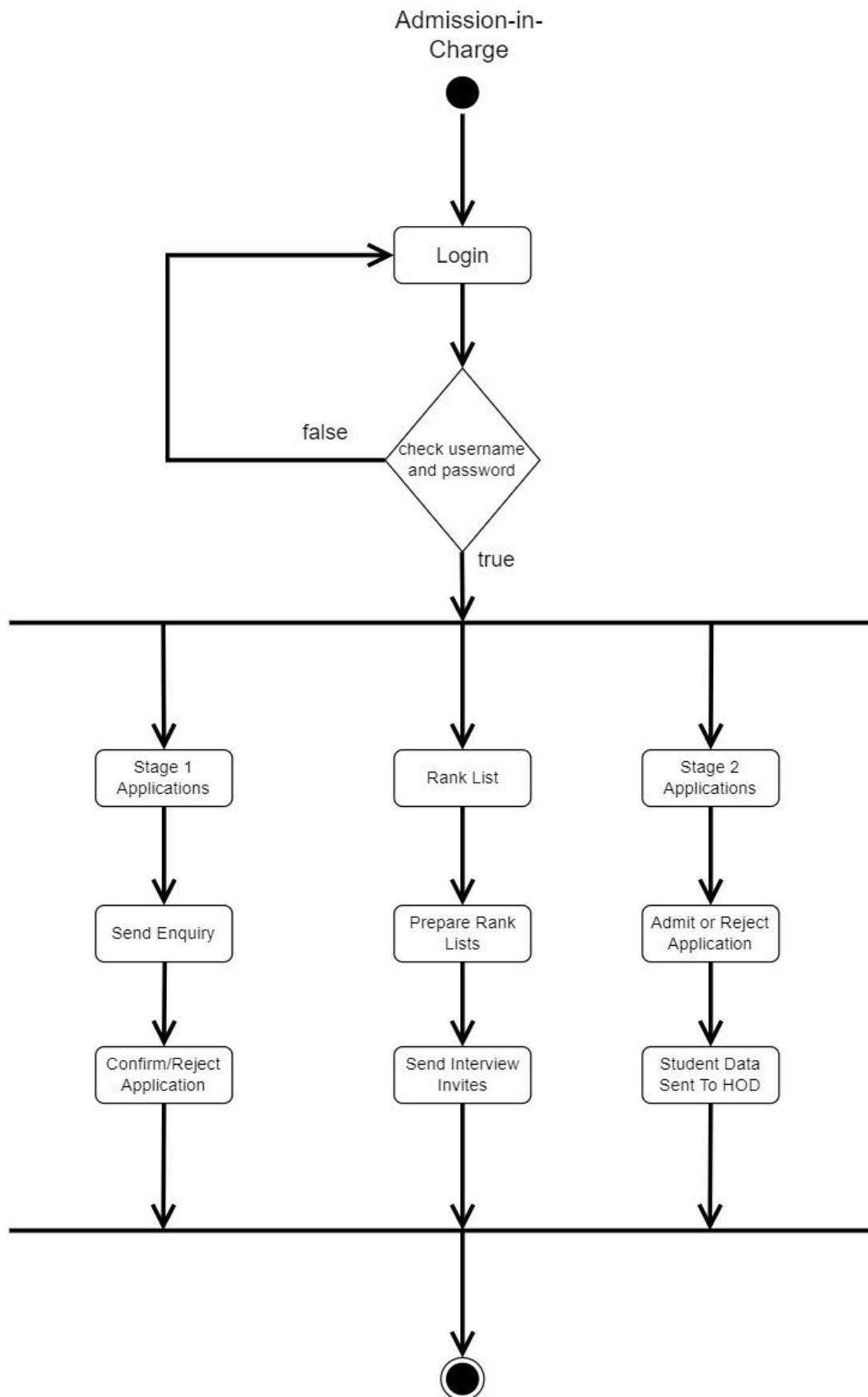
4.2.4 CLASS DIAGRAM

Static diagrams include class diagrams. It represents the application's static view. Class diagrams are used to create executable code for software applications as well as for visualising, explaining, and documenting many elements of systems. The characteristics and functions of a class are described in a class diagram, along with the restrictions placed on the system. Because they are the only UML diagrams that can be directly mapped with object-oriented languages, class diagrams are often employed in the modelling of object-oriented systems. A collection of classes, interfaces, affiliations, collaborations, and restrictions are displayed in a class diagram. Another name for it is 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.
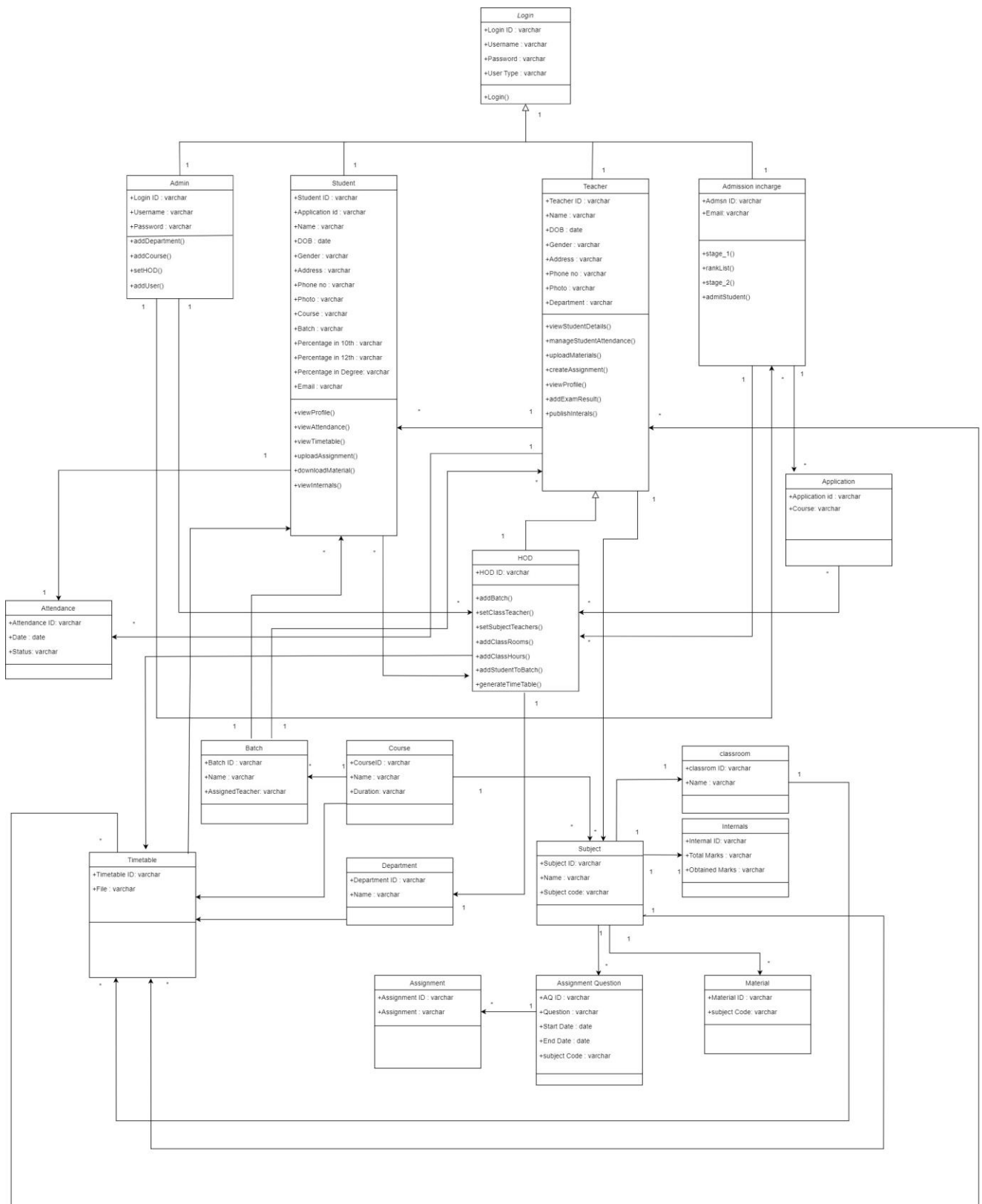
4.2.5 COMPONENT DIAGRAM

A specific type of UML diagram is a component diagram. In addition, the goal is distinct from the previous diagrams mentioned. Although it does not define the system's functionality, it does describe the parts that go into creating that functionality. Therefore, from that perspective, component diagrams are utilised to represent the actual physical parts of a system. These parts include files, libraries, and packages, among others. Another way to think of component diagrams is as a static implementation perspective of a system. Static implementation depicts how the components are arranged at a specific time. A collection of diagrams is used to illustrate the full system because a single component diagram is unable to do so.

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.
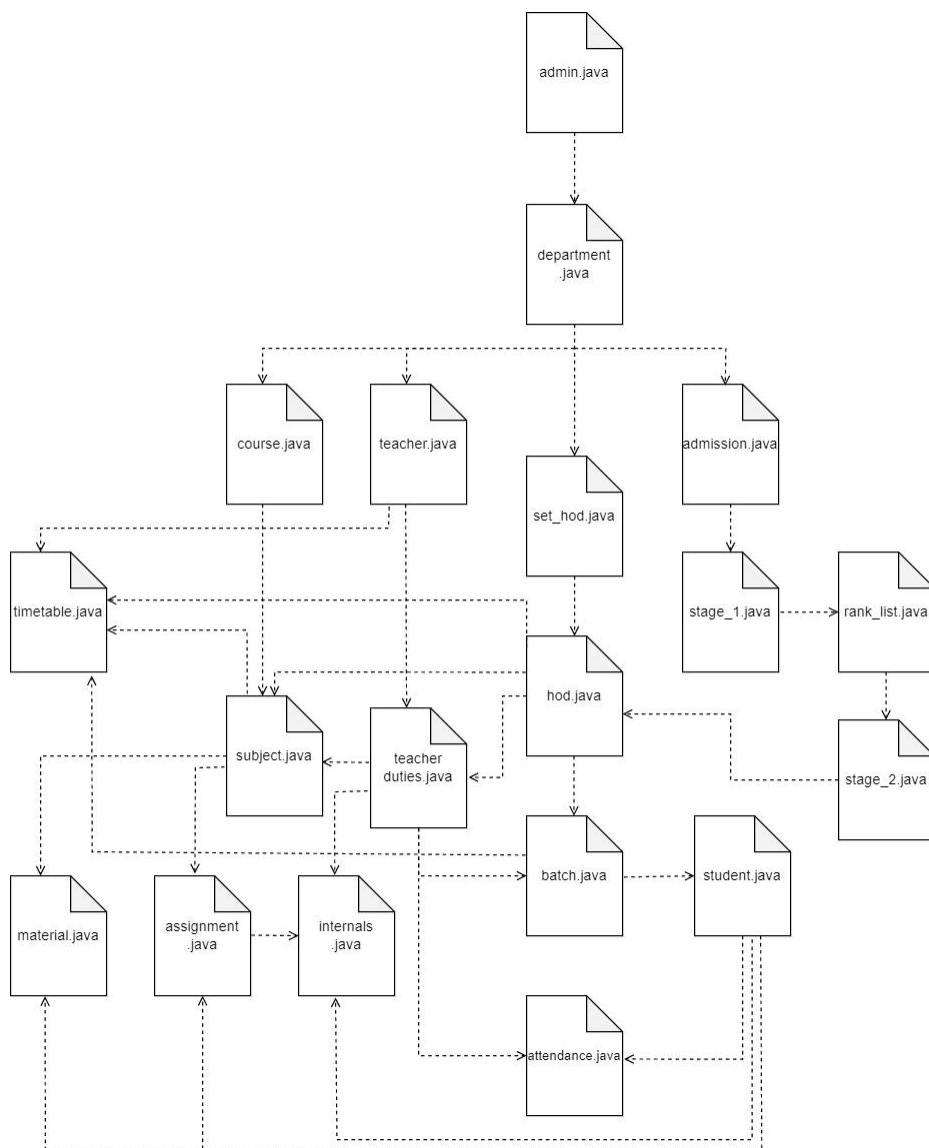
4.2.6 OBJECT DIAGRAM

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.

Login
+Login ID : 1
+Username : admin
+Password : admin101
+User Type : admin

Admin
+Login ID : 1
+Username : admin
+Password : admin101

Student
+Student ID : 12547
+Application id : 109009
+Name : Akhil
+DOB : 22-08-1998
+Gender : male
+Address : Silver Valley, Calicut
+Phone no : 9678876543
+Photo : img090.jpg
+Course : MCA
+Batch : RMCA2022A
+Percentage in 10th : 98.67
+Percentage in 12th : 86.97
+Percentage in Degree: 74.67
+Email : ak76@gmail.com

Teacher
+Teacher ID : T12
+Name : shankar
+DOB : 14-09-1877
+Gender : male
+Address : Javed Nagar, Adavoor
+Phone no : 8778675434
+Photo : tchrimg09.jpg
+Department : MCA

Admission incharge
+Admsn ID : 19872
+Email: admission@sas.com

Application
+Application id : 1969
+Course: MCA

HOD
+HOD ID: T7

Attendance
+Attendance ID: 1321
+Date : 7-15-2022
+Status: 1

Batch
+Batch ID : 1232
+Name : RMCA2022A
+AssignedTeacher: Shankar

Course
+CourseID : 1
+Name : MCA
+Duration: 2

classroom
+classrom ID: R120
+Name : R120

Internals
+Internal ID : 1221
+Total Marks : 50
+Obtained Marks : 42

Timetable
+Timetable ID : 1
+File : tt.xlxx

Department
+Department ID : 1
+Name : MCA

Subject
+Subject ID: 1232
+Name : Software Engineering
+Subject code: ASE12

Assignment
+Assignment ID : 1232
+Assignment : Assignment_1

Assignment Question
+AQ ID : 132
+Question : Describe devOps
+Start Date : 01-07-2022
+End Date : 22-07-2022
+subject Code : ASE12

Material
+Material ID : 12312
+subject Code: ASE12

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.
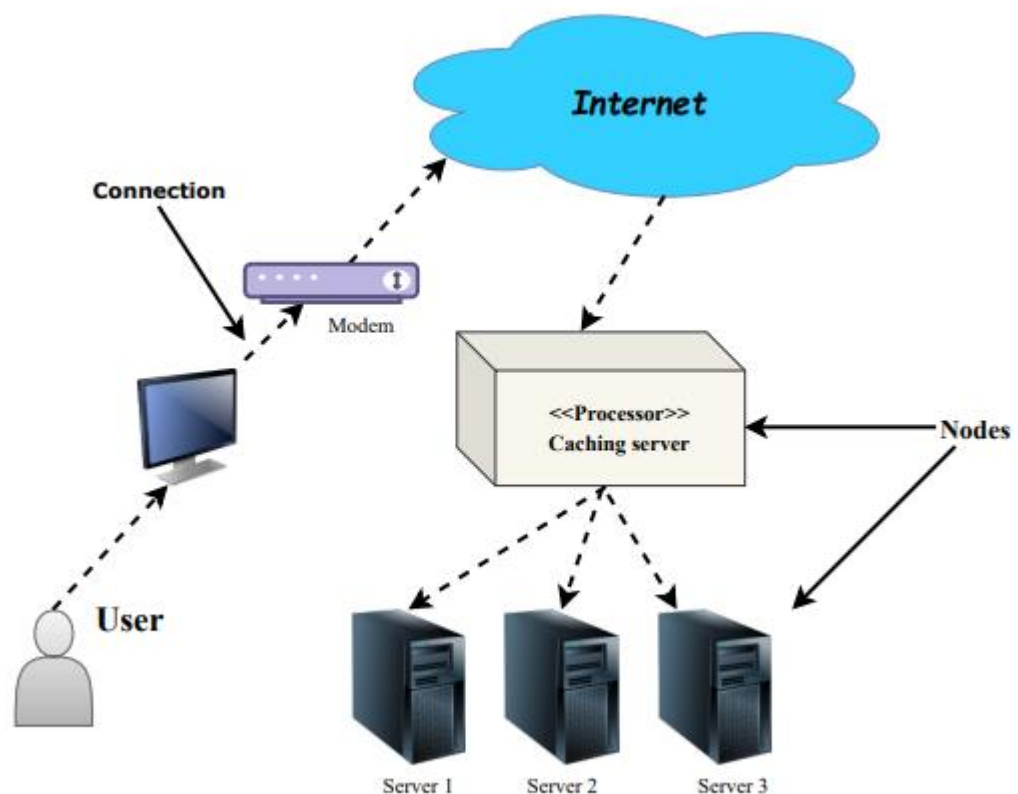
4.2.8 STATE DIAGRAM

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 –
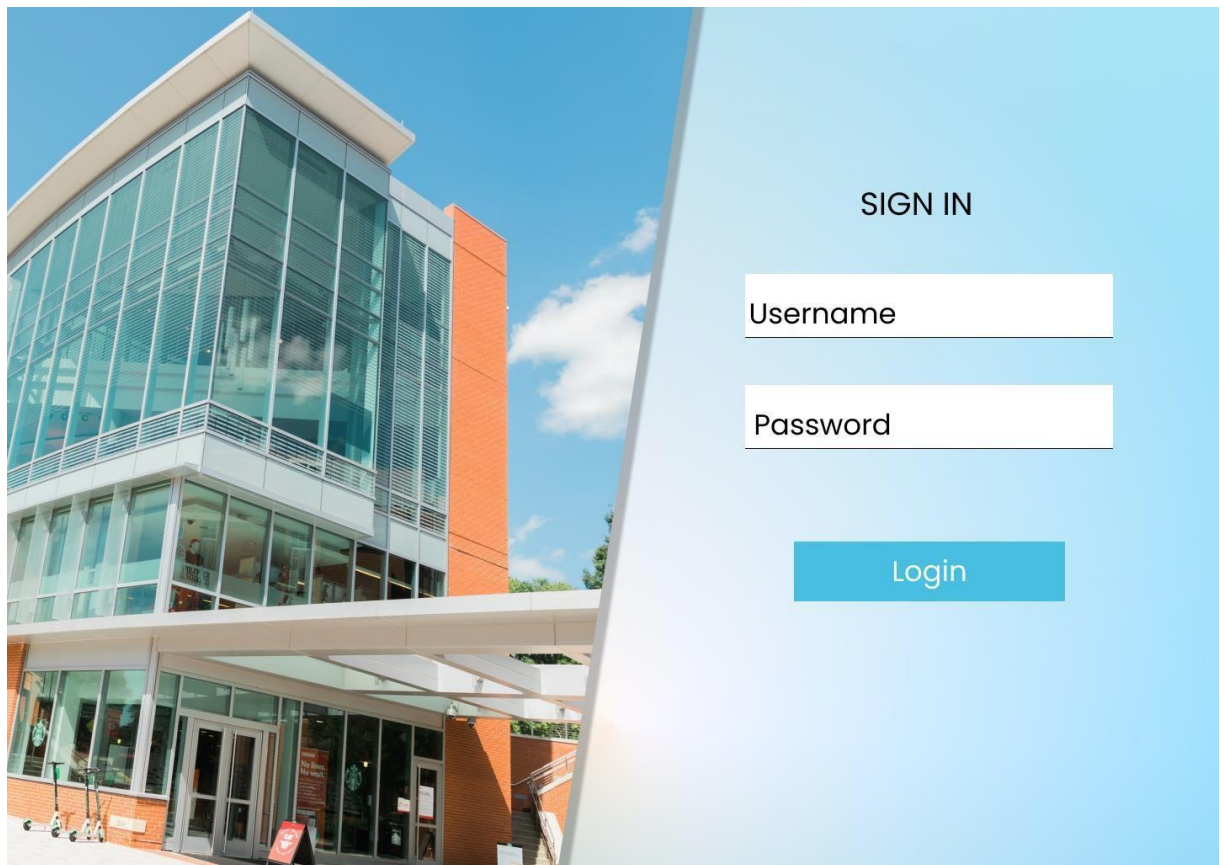
• To simulate a system's dynamic component.

• 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.
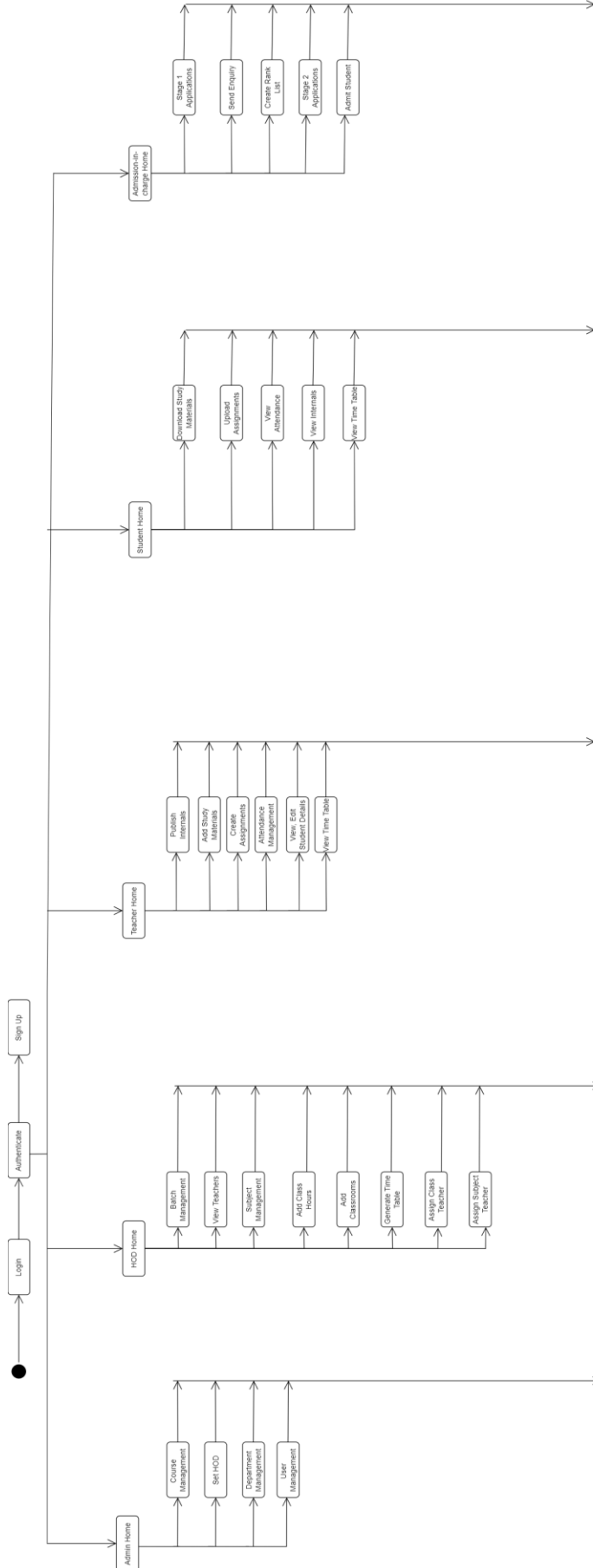
4.3 USER INTERFACE DESIGN

4.3.1-INPUT DESIGN
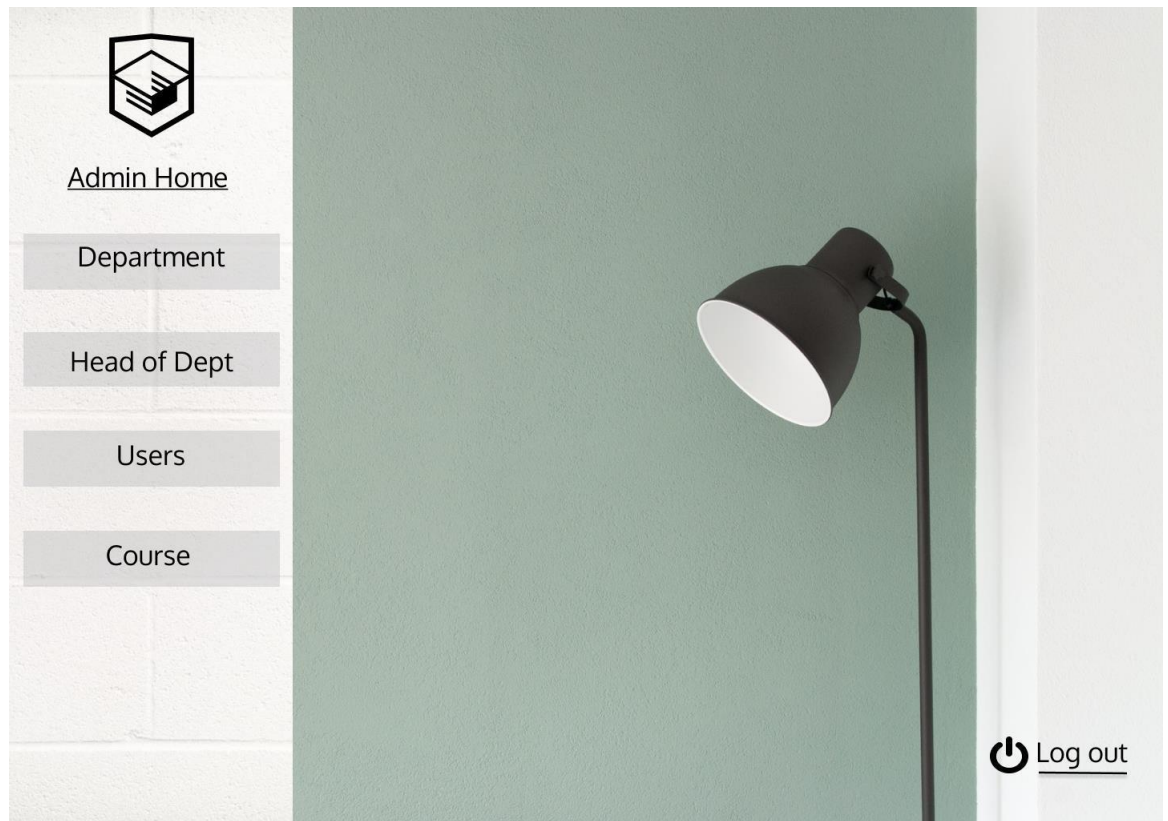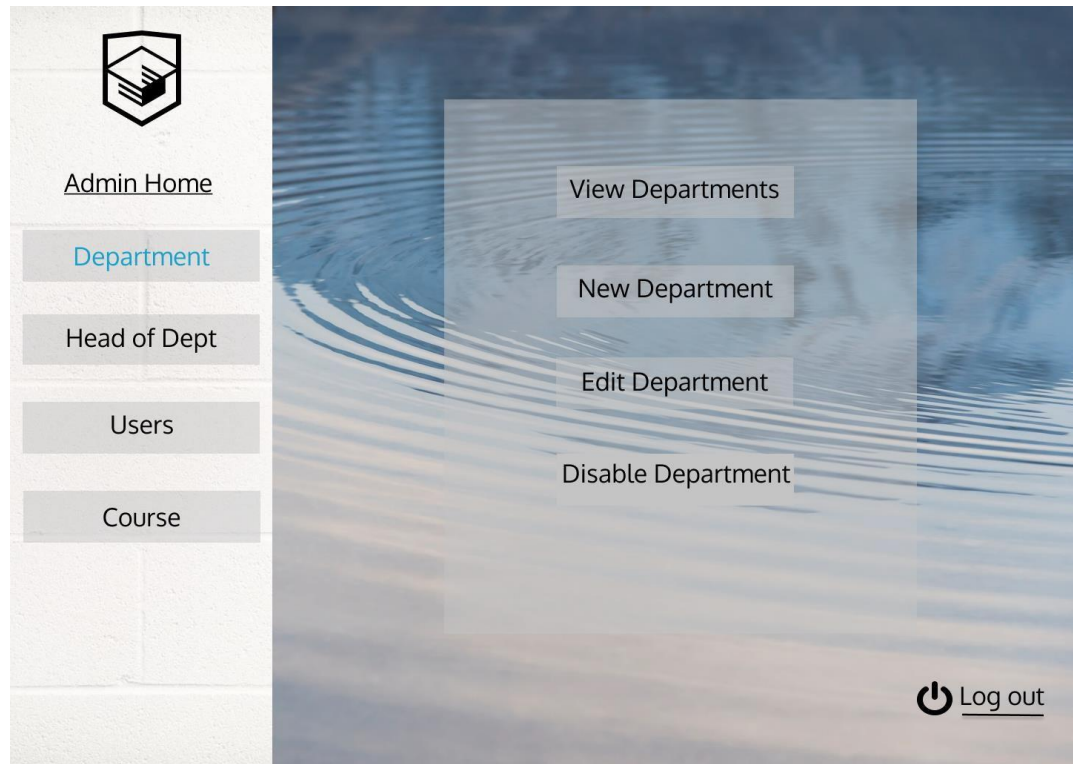
Form Name   : User Login

4.3.2 OUTPUT DESIGN

4.4 DATABASE DESIGN

A database is a structured system with the capacity to store information and allows users to access stored information quickly and effectively. Any database's primary goal is its data, which need protection.

There are two stages to the database design process. The user needs are obtained in the first phase, and a database is created to as clearly as possible satisfy these criteria. This process, known as information level design, is carried out independently of all DBMSs.

The design for the specific DBMS that will be used to construct the system in issue is converted from an information level design to a design in the second stage. Physical Level Design is the stage where the characteristics of the particular DBMS that will be utilised are discussed. Parallel to the system design is a database design. 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)

The database is represented as a set of relationships in a relational paradigm. Each relation appears as a table of values or a file of records. 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 in the formal language of the relational model. In a relational database, the tables are divided into categories and given unique names. A row in a tale corresponds to a collection of linked 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

According to the 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. In other words, 1NF forbids using relationships as attribute values within tuples or relations within relations. Single atomic or indivisible values are the only attribute values that are permitted under 1NF. The data must first be entered into First Normal Form. This may be accomplished by separating data into tables of a similar type in each table. Depending on the needs of the project, a Primary Key or Foreign Key is assigned to each table. For each nested relation or non-atomic property, additional relations are formed in this process. This got rid of data groupings that were repeated. If a relation solely meets the constraints that include the main key, it is said to be in first normal form.

Second Normal Form

No non-key attribute should be functionally dependent on a portion of the main key for relations when the primary key has several attributes, according to Second Normal Form. This involves breaking down each partial key into its dependent characteristics and setting up a new relation for each one. Keep the original primary key and any properties that are entirely dependent on it in your database. This procedure aids in removing data that depends only on a small portion of the key. If and only if a connection meets all the requirements for first normal form for the main key and every one of its non-primary key qualities completely depends on its primary key alone, then that relationship is said to be in second normal form.

Third Normal Form

Relation should not have a non-key attribute that is functionally determined by another non-key property or by a collection of non-key attributes, according to the Third Normal Form. The main key should not be transitively dependent, in other words. The non-key qualities that functionally determine other non-key attributes are decomposed in this way put up in relation. This procedure is used to eliminate everything not wholly dependent on the Primary Key. A relationship is only considered to be in third normal form if it is in second normal form, and it should also not depend on any other relationships' non-key properties.

TABLE DESIGN

Login

Primary key : **Login_id**

Foreign key **Utype_id** references table **User_type**

Name	Type(size)	Constraints	Description
Login_id	Varchar2(10)	Primary key	Login ID
Email	Varchar2(20)	Foreign key	Email of selected applicant
Password	Varchar2(10)	NOT NULL	Secret password
Utype_id	Varchar2(10)	Foreign key	User type id

User_type

Primary key : **Utype_id**

Name	Type(size)	Constraints	Description
Utype_id	Varchar2(10)	Primary key	User type ID
Name	Varchar2(20)	NOT NULL	User type

Department

Primary key : **D_id**

Name	Type(size)	Constraints	Description
D_id	Varchar2(10)	Primary key	Department ID
Name	Varchar2(20)	NOT NULL	Department name
Status	Varchar2(10)	NOT NULL	Active/inactive status

Teacher

Primary key : **Teacher_id**

Foreign key **Login_id** references table **Login**

Foreign key **Dept_id** references table **Department**

Name	Type(size)	Constraints	Description
Teacher_id	Varchar2(10)	Primary key	Teacher ID
Name	Varchar2(30)	NOT NULL	Teacher's name
DOB	Date	NOT NULL	Date of Birth
Gender	Varchar2(10)	NOT NULL	Teacher's gender
Address	Varchar2(50)	NOT NULL	Teacher's address
Email	Varchar2(20)	NOT NULL, UNIQUE	Teacher's email address
Phone	Varchar2(20)	NOT NULL	Teacher's phone number
Dept_id	Varchar2(10)	Foreign key	Department ID
Qualification	Varchar2(10)	NOT NULL	Highest Qualification
Login_id	Varchar2(10)	Foreign key	Login ID

Course

Primary key : **Course_id**

Foreign key **Dept_id** references table **Department**

Name	Type(size)	Constraints	Description
Course_id	Varchar2(10)	Primary key	Course ID
Name	Varchar2(20)	NOT NULL	Course Name
Duration	Varchar2(5)	NOT NULL	Course Duration
Dept_id	Varchar2(10)	Foreign key	Department ID
Status	Varchar2(10)	NOT NULL	Active/inactive status

Application

Primary key : **Application_no**

Foreign key **Course_id** references table **Course**

Name	Type(size)	Constraints	Description
Application_no	Varchar2(10)	Primary key	Application number
Name	Varchar2(30)	NOT NULL	Name of applicant
DOB	Date	NOT NULL	Date of birth
Gender	Varchar2(10)	NOT NULL	Gender of applicant
Address	Varchar2(50)	NOT NULL	Address of applicant
Phone_no	Varchar2(12)	NOT NULL	Phone number
Email	Varchar2(20)	NOT NULL, UNIQUE	Email of applicant
Photo	Varchar2(20)	NOT NULL	Photo path
Course_id	Varchar2(10)	Foreign key	Course ID student applied for
Stage	Varchar2(10)	NOT NULL	Application phase

Student

Primary key : **Admission_no**

Foreign key **Application_no** references table **Application**

Foreign key **Batch_no** references table **Batch**

Name	Type(size)	Constraints	Description
Admission_no	Varchar2(10)	Primary key	Admission number
Application_no	Varchar2(10)	Foreign key	Application number
Batch_id	Varchar2(10)	Foreign key	Batch ID

Parent

Primary key : **Parent_id**

Foreign key **Application_no** references table **Application**

Name	Type(size)	Constraints	Description
Parent_id	Varchar2(10)	Primary key	Parent ID
Father_name	Varchar2(30)	NOT NULL	Name of father
Mother_name	Varchar2(30)	NOT NULL	Name of mother
Father_occupation	Varchar2(20)	NOT NULL	Father's occupation
Mother_occupation	Varchar2(20)	NOT NULL	Mother's occupation
Father_mob_no	Varchar2(12)	NOT NULL	Father's phone number
Mother_mob_no	Varchar2(12)	NOT NULL	Mother's phone number
Father_email_id	Varchar2(20)	NOT NULL	Father's email id
Mother_email_id	Varchar2(20)	NOT NULL	Mother's email id
Application_no	Varchar2(10)	NOT NULL	Foreign key

Record

Primary key : **Record_id**

Foreign key **Application_no** references table **Application**

Name	Type(size)	Constraints	Description
Record_id	Varchar2(10)	Primary key	Academic ID
Application_no	Varchar2(10)	Foreign key	Application number
Percentage_10	Varchar2(10)	NOT NULL	Percentage in 10th
Percentage_12	Varchar2(10)	NOT NULL	Percentage in 12th
Percentage_diploma	Varchar2(10)	NOT NULL	Percentage in Diploma
Percentage_UG	Varchar2(10)	NOT NULL	Percentage in UG
Certificate_10	Varchar2(30)	NOT NULL	10th std certificate
Certificate_12	Varchar2(30)	NOT NULL	12th std certificate
Certificate_diploma	Varchar2(30)	NOT NULL	Diploma certificate
Certificate_UG	Varchar2(30)	NOT NULL	UG marklist

Batch

Primary key : **Batch_id**

Foreign key **Course_id** references table **Course**

Foreign key **Teacher_id** references table **Teacher**

Name	Type(size)	Constraints	Description
Batch_id	Varchar2(10)	Primary key	Batch ID
Name	Varchar2(20)	NOT NULL	Batch Name
Duration	Varchar2(5)	NOT NULL	Course Duration
Semester	Varchar2(5)	DEFAULT=0	Semester
Course_id	Varchar2(10)	Foreign key	Course ID
Teacher_id	Varchar2(10)	Foreign key	Class teacher ID

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do.This is much more difficult than it may at first appear, especially for large programs.

5.2 TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

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

5.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code where removed and ensured that all modules are working, and gives the expected result.

5.2.2 Integration Testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

5.2.3 Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

5.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time ofdeveloping and makingchanges whenever required. This done with respect to the following points:

➤ Input Screen Designs,

➤ Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using thattest data. While testing the system by which test data errors are again uncovered and corrected byusing above testing steps and corrections are also noted for future use.

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening anew system design into operation, which is the process of converting a new revised system designinto an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it cancreate chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing andchangeover.

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

Implementation of software refers to the final installation of the package in its real environment,to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software

development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that:

- ☐ The active user must be aware of the benefits of using the new system. ☐ Their confidence in the software is built up.
- ☐ Proper guidance is imparted to the user so that he is comfortable in using theapplication.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

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

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correctthe date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. Thistraining may be different across different user groups and across different levels of hierarchy

6.2.3 System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycleis the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION AND FUTURE SCOPE

The current system is old fashioned and most of the processes are done offline and manually. The proposed mini system introduces automation of a major part of the admission process. It also has the scope to expand beyond and assist in day to day class activities of students and teachers. In future, looking forward to the development of the main system, the main functionality is developing the module for automatic timetable generation.