

CLASSROOM MANAGEMENT SYSTEM

Project Report Submitted By

DIVYA REJI

Reg No: AJC19MCA008

In Partial fulfillment for the Award of the Degree Of

**MASTER OF COMPUTER APPLICATIONS
(REGULAR MCA)**

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



**AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPALLY**

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

2019-22

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPALLY



CERTIFICATE

This is to certify that the Project report, “**CLASSROOM MANAGEMENT SYSTEM**” is the bonafide work of **DIVYA REJI (Reg No: AJC19MCA008)** in partial fulfillment of the requirements for the award of the Degree of Regular Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2019-22.

Ms. Pauline Paul
Internal Guide

Rev. Fr. Dr. Rubin Thottupurathu Jose
Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

External Examiner

DECLARATION

I hereby declare that the project report “**CLASSROOM MANAGEMENT SYSTEM**” is a bonafide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Regular Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

Date:

KANJIRAPPALLY

DIVYA REJI

Reg No: AJC19MCA008

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 to express 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 coordinator **Rev. Fr. Dr. Rubin Thottupurathu Jose** for his 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. Pauline 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 for the moral support, suggestions, and encouragement to make this venture a success.

DIVYA REJI

ABSTRACT

The Classroom Management Project is a software application which is effective in record-keeping for an educational enterprise. This website keeps the data in a centralized way which is available to all the users at the same time. It manages data in database efficiently. The project is used to manage students, staff and admin HOD himself. It also handles their daily work. It consists of profile, courses, subjects, leaves taken by the students and staff both. It also aids in improving the system efficiency by receiving feedbacks and notifications from both students and staff.

CONTENT

Sl. No	Topic	Page No
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	3
2.1	INTRODUCTION	4
2.2	EXISTING SYSTEM	4
2.3	DRAWBACKS OF EXISTING SYSTEM	5
2.4	PROPOSED SYSTEM	5
2.5	ADVANTAGES OF PROPOSED SYSTEM	5
3	REQUIREMENT ANALYSIS	7
3.1	FEASIBILITY STUDY	8
3.1.1	ECONOMICAL FEASIBILITY	8
3.1.2	TECHNICAL FEASIBILITY	9
3.1.3	BEHAVIORAL FEASIBILITY	9
3.2	SYSTEM SPECIFICATION	10
3.2.1	HARDWARE SPECIFICATION	10
3.2.2	SOFTWARE SPECIFICATION	10
3.3	SOFTWARE DESCRIPTION	10
3.3.1	DJANGO	10
3.3.2	MYSQL	11
4	SYSTEM DESIGN	13
4.1	INTRODUCTION	14
4.2	UML DIAGRAM	14
4.2.1	USE CASE DIAGRAM	15
4.2.2	SEQUENCE DIAGRAM	18
4.2.3	COLLABORATION DIAGRAM	19
4.2.4	STATE CHART DIAGRAM	20
4.2.5	ACTIVITY DIAGRAM	21
4.2.6	CLASS DIAGRAM	21
4.2.7	OBJECT DIAGRAM	22
4.2.8	COMPONENT DIAGRAM	23

4.2.9	DEPLOYMENT DIAGRAM	23
4.5	USER INTERFACE DESIGN	24
4.6	DATABASE DESIGN	29
5	SYSTEM TESTING	37
5.1	INTRODUCTION	39
5.2	TEST PLAN	39
5.2.1	UNIT TESTING	39
5.2.2	INTEGRATION TESTING	40
5.2.3	VALIDATION TESTING	40
5.2.4	USER ACCEPTANCE TASTING	40
6	IMPLEMENTATION	45
6.1	INTRODUCTION	46
6.2	IMPLEMENTATION PROCEDURE	47
6.2.1	USER TRAINING	49
6.2.2	TRAINING ON APPLICATION SOFTWARE	48
6.2.3	SYSTEM MAINTENANCE	49
7	CONCLUSION & FUTURESCOPE	49
7.1	CONCLUSION	50
8	BIBLIOGRAPHY	51
9	APPENDIX	53
9.1	SAMPLE CODE	55
9.2	SCREEN SHOTS	57

List of Abbreviation

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

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The Classroom Management Project is a software application which is effective in record-keeping for an educational enterprise. This website keeps the data in a centralized way which is available to all the users at the same time. It manages data in database efficiently. The project is used to manage students, staff and admin HOD himself. It also handles their daily work. It consists of profile, courses, subjects, leaves taken by the students and staff both. It also aids in improving the system by receiving feedbacks and notifications from both students and staff.

1.2 PROJECT SPECIFICATION

The proposed system is a website in which users like admin, student and teacher can access the website. Also, that the student and teachers can mark and view attendance to facilitate attendance taking process. The system includes 3 modules. They are:

1. ADMIN MODULE

Admin must have a login into this system. He has the overall control of the system. Admin can create, read, update and remove students and staff details. Along with that, they can also add the courses, subjects and manage feedbacks, leaves and manage notifications. Admin can view all the registered users and manage all his data.

2. STUDENT MODULE

Through the help of the login system, here the user can view details of their courses or subject. Also, they can make changes to one's profile, apply for leaves, give feedbacks, mark attendance and pass notifications.

3. STAFF MODULE

Here the teacher can see all the features of the system. One can view and make changes in their own profile, apply for leaves, take student attendance, view it. One can also receive feedbacks.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minute's detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

2.2 EXISTING SYSTEM

Existing system is not a fully automated system. With such huge amount of data meant to be stored which is changing on an everyday basis, it is hard to be managed by the admin and the institution itself. Such a way of record keeping traditionally is

not at all wise approach. The proposed system rectifies the drawbacks of the present system. 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 system
- Human effort is needed.
- More manual hours need to generate required reports.

2.4 PROPOSED SYSTEM

The proposed system is defined to meets all the disadvantages of the existing system. It is necessary to have a system that is more user friendly and user attractive for growth of children and the institution at the end of the day; on such consideration the system is proposed. In our proposed system there is admin who can view all the customers. It allows users to access and use the system and mark their attendance. Users of this proposed system are admin, students and teachers. This software application avoids more manual hours that need to spend in record keeping. This application keeps the data in a centralized way which is available to all the users simultaneously. It is very easy to manage historical data in database. No specific training is required for the distributors to use this application. They can easily use the tool that decreases manual hours spending for normal things and hence increases the performance. It is very easy to record the information of work flow of the institution in the databases. Such a system helps in the accreditation process of the institution in the long run.

2.5 ADVANTAGES OF PROPOSED SYSTEM

The system is exceptionally straightforward in design and to actualize. The system requires exceptionally low system assets, and the system will work in nearly all setups. It has got following features:

- **Better security: -**

For data to remain secure measures must be taken to prevent unauthorized access. Security means that data are protected from various forms of destruction. The system security problem can be divided into four related issues: security, integrity, privacy and confidentiality. Username and password requirement to sign in ensures security. It will also provide data

security as we are using the secured databases for maintaining the documents.

- **Ensure data accuracy: -**

The proposed system disposes of the manual mistakes whereas entering the subtle elements of the clients amid the registration.

- **Better service: -**

The item will dodge the burden of hard copy storage. We will moreover moderate the time and human assets for doing the same task. The information can be kept up for longer period with no misfortune of information.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features: -

3.1.1 Economical Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

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 proposed system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

The cost of project was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open-source software.

3.1.2 Technical Feasibility

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

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 developed such that the necessary functions and performance are achieved within the constraints. Through the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using Django in front end and MySQL in server in back end, the project is technically feasible for development. 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, AJAX, DJANGO
Backend	- MySQL
Client on PC	- Windows 7 and above.
Technologies used	- JavaScript, HTML5

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO

Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement. By using Django, we can build web applications in very less time. Django is designed in such a manner that it handles much of configure things automatically, so we can focus on application development only.

Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly. Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly. Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project. Django includes various helping task modules and libraries which can be used to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds etc.

3.3.2 MySQL

MySQL, the most popular Open-Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. The MySQL Web site provides the latest information about MySQL software.

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

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database; you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- ***MySQL databases are relational.***

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well- designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data. The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax. SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL92” refers to the standard released in 1992,

“SQL: 1999” refers to the standard released in 1999, and “SQL: 2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

- ***MySQL software is Open Source.***

OpenSource means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy

a commercially licensed version from us. See the MySQL Licensing Overview for more information.

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

If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available.

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

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). We also provide MySQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user-oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. UML stands for **Unified Modeling Language**. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general-purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. 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.

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

4.2.1 USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram:

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use not 'es whenever required to clarify some important points.

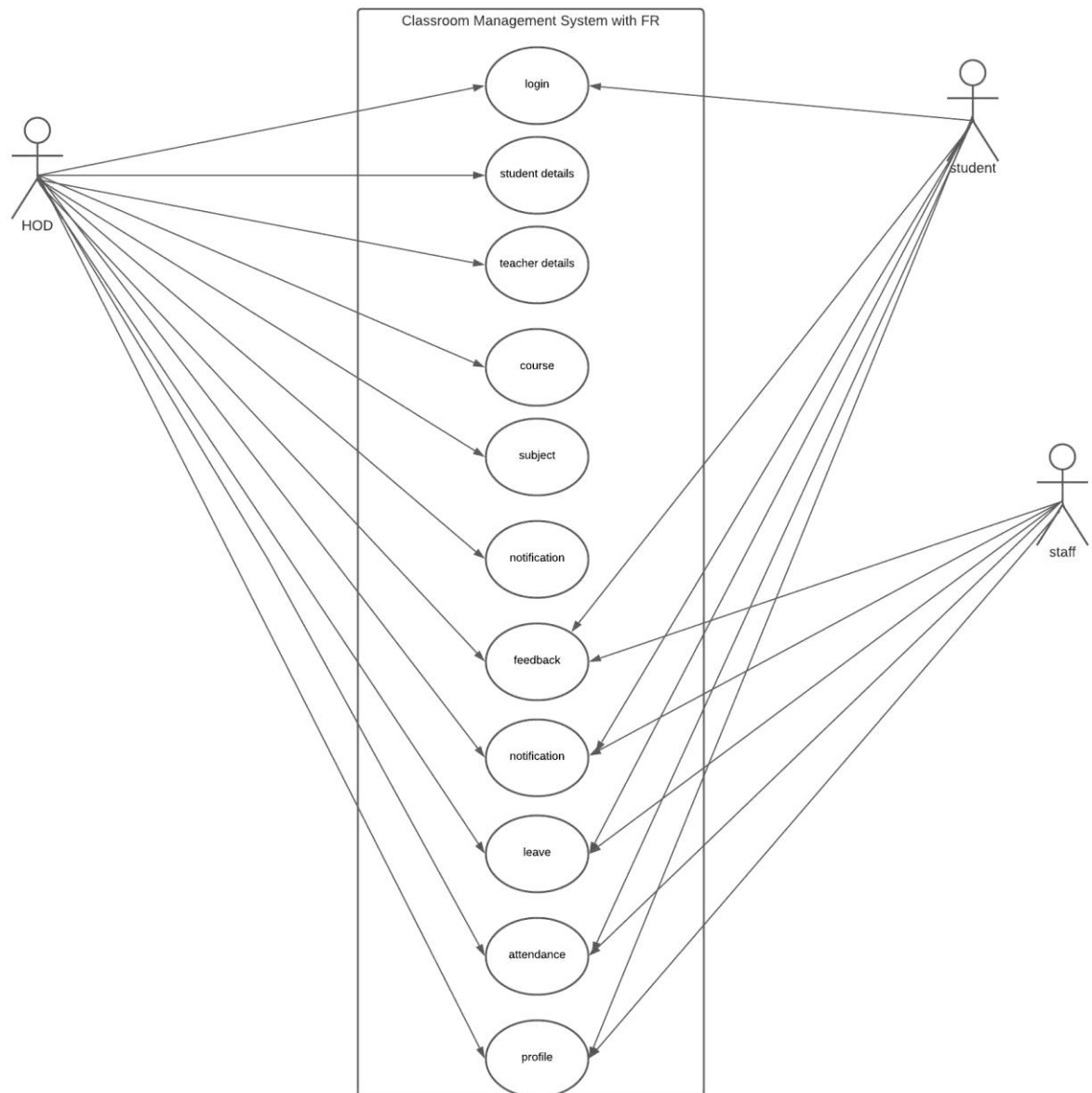


Figure 1: Use Case Diagram

4.2.2 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

Sequence Diagram Notations –

- i. **Actors** – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.
- ii. **Lifelines** – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically, each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram
- iii. **Messages** – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

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 – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Uses of sequence diagrams –

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.

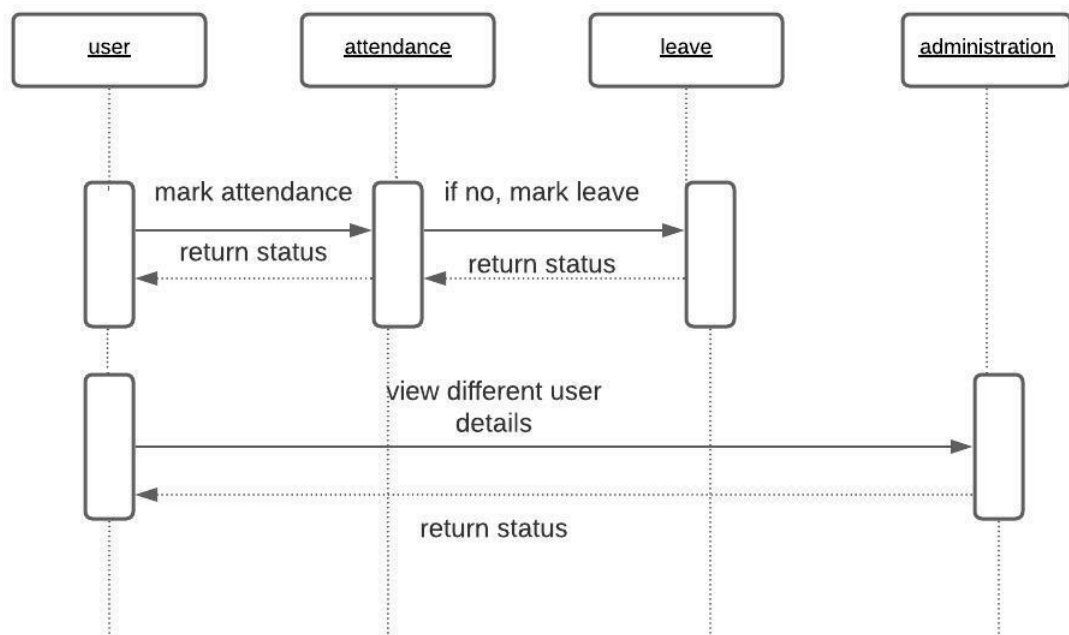


Figure 2: Sequence Diagram

4.2.3 COLLABORATION DIAGRAM

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of

several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system

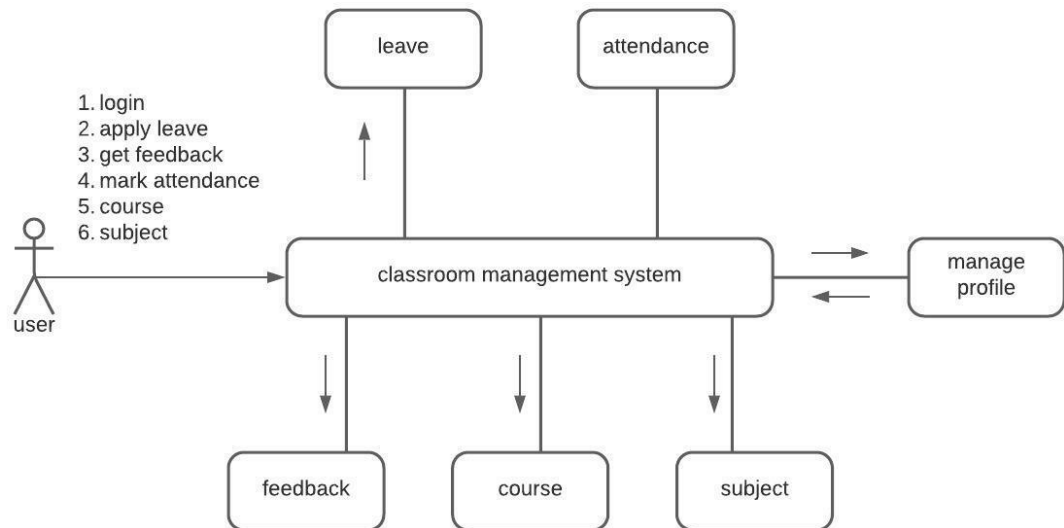


Figure 3: Collaboration Diagram

4.2.4 STATE-CHART DIAGRAM

It describes different states of a component in a system. The states are specific to a component/object of a system. A State chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events. State chart diagram describes the flow of control from one state to another state. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination. State chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system. Following are the main purposes of using State chart diagrams –

- To model the dynamic aspect of a system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object

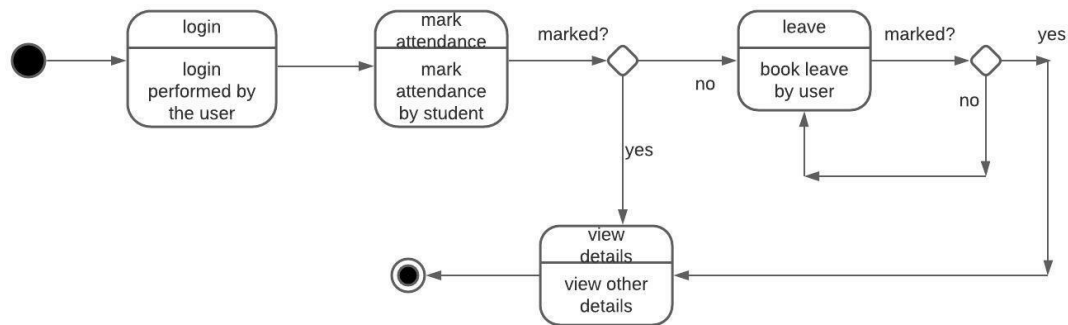


Figure 4: State-Chart Diagram

4.2.5 ACTIVITY DIAGRAM

Activity Diagram describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart.

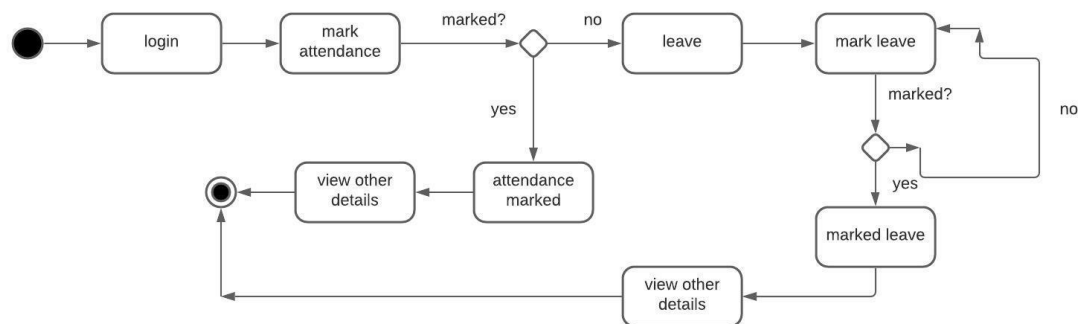


Figure 5: Activity Diagram

4.2.6 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of

classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram. The purpose of the class diagram can be summarized as –

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.

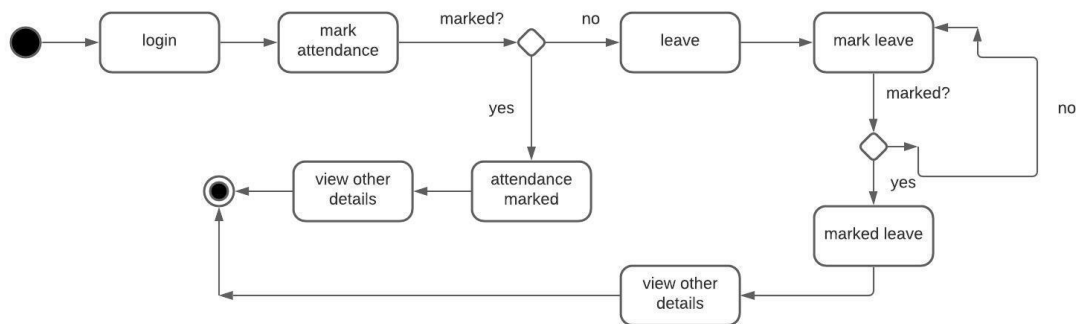


Figure 6:Class Diagram

4.2.7 OBJECT DIAGRAM

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.

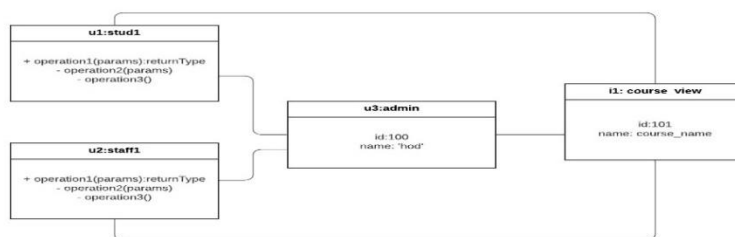


Figure 7:Object Diagram

4.2.8 COMPONENT DIAGRAM

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. Thus, from that point of view, component diagrams are used to visualize the physical components in a system. components are libraries, packages, files, etc. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment. A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

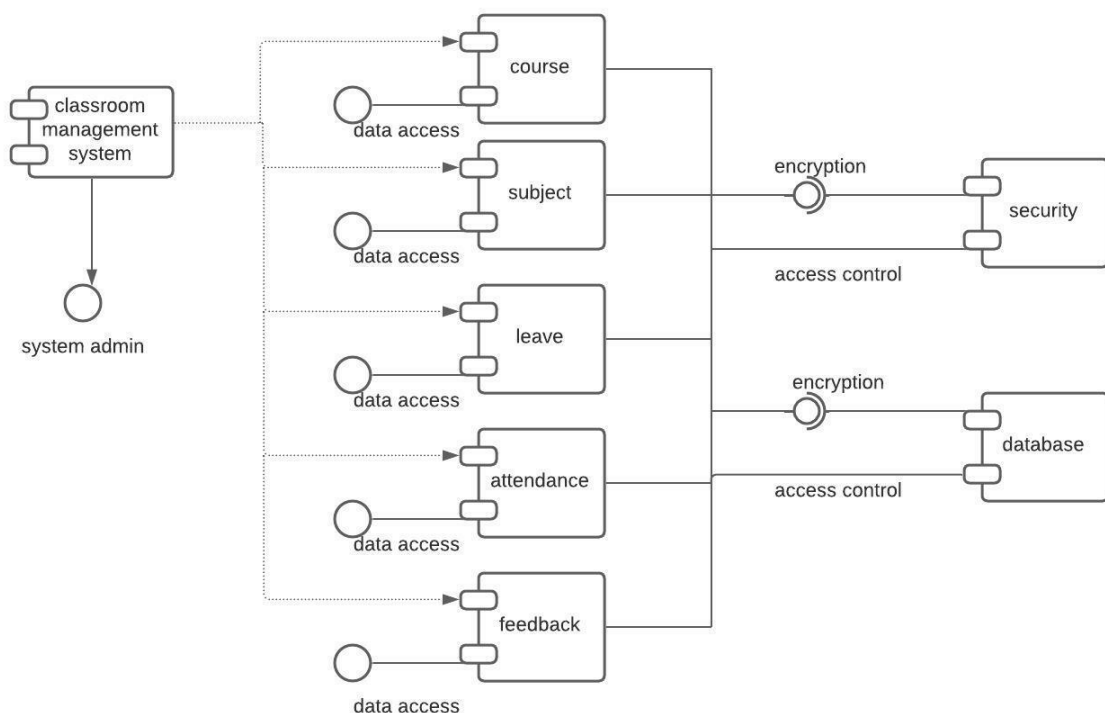


Figure 8: Component Diagram

4.2.9 DEPLOYMENT DIAGRAM

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. It consists of nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

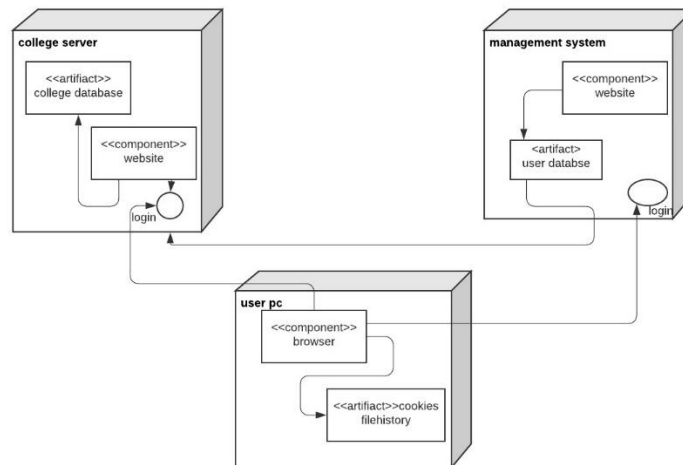


Figure 9:Deployment Diagram

4.3 FORM DESIGN

4.3.1-LOGIN FORM

Form Name : User Login

ONLINE CLASSROOM MANAGEMENT
SYSTEM

LOGIN

New? Register here

4.3.3-INDEX FORM

Form Name : Student Index Page

The screenshot shows the 'Student Index Page' within the 'Online Classroom Management System'. The page has a grey header with the system name and a green 'LOGOUT' button. The main content area has a red background. On the left, under 'Your Courses', there is a large red box labeled 'Subjects'. On the right, under 'Notifications', there is a grey box labeled 'Notifications list'.

4.4 OUTPUT DESIGN

User Login:

The screenshot shows the 'Student Management System' login page. The title 'Student Management System' is centered at the top. Below it, the text 'Log in to Student Management System' is displayed. The login form includes fields for 'Email' and 'Password', each with an icon (envelope and lock respectively). Below these fields is a reCAPTCHA widget with the text 'I'm not a robot' and a checkbox. A blue 'Log In' button is positioned below the reCAPTCHA. At the bottom of the form, there are links for 'Reset Password', 'Admin Signup', 'Staff Signup', and 'Student Signup'.

Admin Home Page:**Admin Add Staff Page:**

The Admin Add Staff Page displays a form for adding a new staff member. The form includes the following fields:


- Email address:** Enter email
- Password:** Enter password
- First Name:** Enter first name
- Last Name:** Enter last name
- Username:** Enter username
- Address:** Enter address

At the bottom of the form, there is a blue button labeled "Add Staff".

The sidebar contains the following navigation options: Home, Add Staff, Manage Staff, Add Student, Manage Students, Add Course, Manage Course, Add Subject, Manage Subject, Manage Session Year, Student Feedback, Staff Feedback, View Attendance, and Student Leave.

Copyright © 2021-2022 Divya Reji. All rights reserved. Version 1

Admin Staff Leave Page:

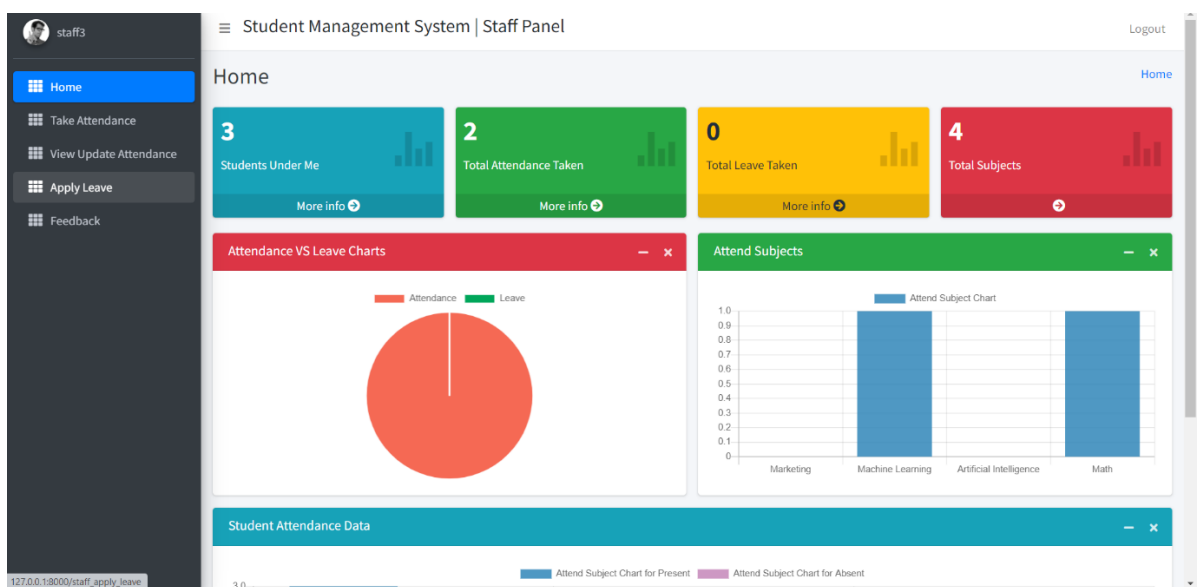
 admin3

- Home
- Add Staff
- Manage Staff
- Add Student
- Manage Students
- Add Course
- Manage Course
- Add Subject
- Manage Subject
- Manage Session Year
- Student Feedback
- Staff Feedback
- View Attendance
- Student Leave

Student Management System HOD Login
 Logout

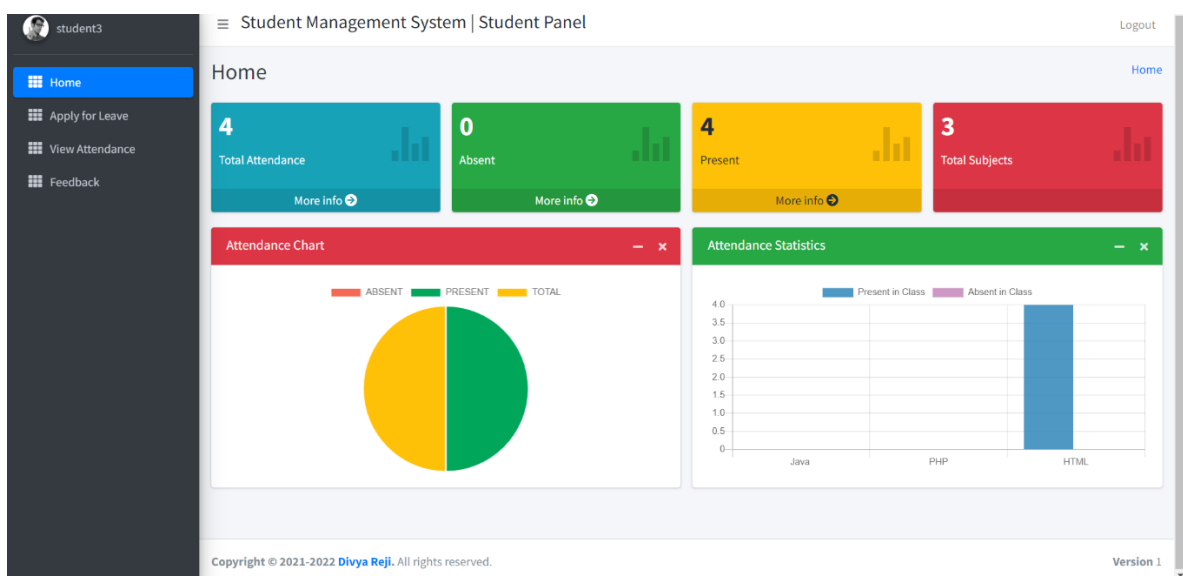
Staff Apply For Leave

ID	Staff ID	Staff Name	Leave Date	Leave Message	Apply On	Action
1	2	Staff One	2022-04-06	Viral Fever	April 5, 2022, 4:34 a.m.	Approved
2	2	Staff One	2022-04-12	Brother's Wedding	April 5, 2022, 6:17 a.m.	Approved
3	2	Staff One	2022-04-14	Government Exam today	April 5, 2022, 7:28 a.m.	Approved
4	12	Sample Email	2022-04-06	sick	April 6, 2022, 9:08 a.m.	Disapproved
5	3	Staff first Two last	2022-05-09	viral fever	May 7, 2022, 5:45 a.m.	Approved
6	3	Staff first Two last	2022-05-11	sister's wedding	May 7, 2022, 5:47 a.m.	Disapproved
7	4	Staff Three	2022-05-11	outing	May 7, 2022, 6:05 a.m.	Disapproved
8	5	Staff Four	2022-05-12	son's school function	May 7, 2022, 6:06 a.m.	Approved

Staff Home Page:

Staff Mark Attendance Page:

The screenshot shows the 'View Update Attendance' page for a staff member. The left sidebar contains navigation links: Home, Take Attendance, View Update Attendance (highlighted), Apply Leave, and Feedback. The main content area has a header 'Student Management System | Staff Panel' and a 'Logout' link. Below the header, there's a 'View Update Attendance' section with a blue bar. It includes a 'Subject' dropdown menu (Machine Learning), a 'Session Year' dropdown menu (Jan. 1, 2022 TO Jan. 1, 2025), a 'Fetch Attendance' button, an 'Attendance Date' dropdown menu (2022-05-06), a 'Fetch Student Data' button, and a 'Student Attendance' section with a checkbox for 'Student Two [Present]' and a 'Save Attendance Data' button. A version number '1.27.0.0 1-60202/staff_feedback' is visible in the bottom left corner.

Student Home Page:

Student View Attendance Page:

student3 Student Management System | Student Panel Logout

View Attendance

View Attendance

Subject: HTML

Start Date: 30-03-2022 End Date: 09-05-2022

Fetch Attendance

Copyright © 2021-2022 Divya Reji. All rights reserved. Version 1

student3 Student Management System | Student Panel Logout

Attendance Data

View Attendance

Date : May 6, 2022 [Status : Present]

Date : May 6, 2022 [Status : Present]

Date : May 5, 2022 [Status : Present]

Date : May 5, 2022 [Status : Present]

Copyright © 2021-2022 Divya Reji. All rights reserved. Version 1

4.3 DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected. The database design is a two-level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS. In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called

Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

- Data Integrity
- Data independence

Relational Database Management System (RDBMS)

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a table represents a set of related values.

Relations, Domains & Attributes:

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values. Every value in a relation is atomic, that is not decomposable.

Relationships

- Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
- Entity Integrity enforces that no Primary Key can have null values.
- Referential Integrity enforces that no Primary Key can have null values.
- Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other keys are Super Key and Candidate Keys.

Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a

technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form.

As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups at data and thereby avoids data redundancy which proves to be a great burden on the computer resources. These include:

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

First Normal Form

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words, 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be done by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

Second Normal Form

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data

that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attribute of the relation is fully dependent on its primary key alone.

Third Normal Form

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on another non- key attribute.

4.4 TABLE DESIGN:

1. Staff Table – ‘staff’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Staff_id	varchar	Not null	50	Holds the id of the user.
2	name	varchar	Not null	50	Holds the name of the user.
3	email	varchar	Not null	50	Holds the email of the user
4	password	vaarchar	Not null	50	Holds the password of the user.
5	Created_at	varchar	Not null	10	Holds the creation date
6	Updated_at	varchar	Not null	10	Holds the updation date
7	address	varchar	Not null	100	Holds the address of the staff

2. Student Table – ‘student’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Student_id	varchar	Not null	50	Holds the id of the student
2	name	varchar	Not null	50	Holds the name.
3.	email	varchar	Not null	20	Holds the email.
4.	gender	varchar	Not null	10	Holds the gender.
5	Profile_pic	image	Not null	10	Hold the picture of the student
6	Password	varchar	Not null	10	Holds the password
7	Created_At	Varchar	Not null	10	Creation date
8	Updated_at	varchar	Not null	10	Updation date
9	address	varchar	Not null	50	Address of the student

3. Admin HOD Table – ‘admin_hod’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Hod_id	varchar	Not null	50	Holds the id of the hod
2	name	varchar	Not null	50	Holds the name.
3.	email	varchar	Not null	20	Holds the email.
4	Password	varchar	Not null	10	Holds the password
5	Created_At	Varchar	Not null	10	Creation date
6	Updated_at	varchar	Not null	10	Updation date

4. Course Table – ‘course’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Course_id	varchar	Not null	50	Holds the id of the course
2	Course_name	varchar	Not null	50	Holds the name.
3	Created_At	Varchar	Not null	10	Creation date
4	Updated_at	varchar	Not null	10	Updation date

5. Subject Table – ‘subject’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Subject_id	varchar	Not null	50	Holds the id of the subject
2	Subject_name	varchar	Not null	50	Holds the subject name.
3.	Course_id	varchar	Not null	20	Holds the course name.
4.	Staff_id	varchar	Not null	10	Holds the staff id.
5	Created_At	Varchar	Not null	10	Creation date
6	Updated_at	varchar	Not null	10	Updation date

6. Notification Staff Table – ‘notification staff’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Notification_id	varchar	Not null	50	Holds the id of the notification.
2	Staff_id	varchar	Not null	50	Holds the staff_id.
3.	message	varchar	Not null	100	Holds the message.
7	Created_At	Varchar	Not null	10	Creation date
8	Updated_at	varchar	Not null	10	Updation date

7. Leave Table – ‘leave’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Leave_id	varchar	Not null	50	Holds the id of the leave.
2	Student_staff_id	varchar	Not null	50	Holds the id.
3.	Is_staff	varchar	Not null	20	Holds the if staff or not.
4.	status	varchar	Not null	10	Holds the status.
5	Leave_Date	varchar	Not null		Hold the date of leave.
6	Created_At	Varchar	Not null	10	Creation date
7	Updated_at	varchar	Not null	10	Updation date

8. Notification Student Table – ‘notification_student’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Notification_student_id	varchar	Not null	50	Holds the id of the notification.
2	Student_id	varchar	Not null	50	Holds the student id.
3.	message	varchar	Not null	100	Holds the message.
4	Created_At	Varchar	Not null	10	Creation date
5	Updated_at	varchar	Not null	10	Updation date

9. Feedback Table – ‘feedback’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Feedback_id	varchar	Not null	50	Holds the id of the feedback.
2	Student_staff_id	varchar	Not null	50	Holds the id.
3.	Is_staff	varchar	Not null	20	Holds the if staff or not.
4.	message	varchar	Not null	100	Holds the message.
5	Created_At	Varchar	Not null	10	Creation date
6	Updated_at	varchar	Not null	10	Updation date

10. Attendance Table – ‘attendance’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	Attendance_id	varchar	Not null	50	Holds the attendance_id of the student.
2	Subject_id	varchar	Not null	50	Holds the name.
3.	Date_time	varchar	Not null	20	Holds the email.
4	Created_At	Varchar	Not null	10	Creation date
5	Updated_at	varchar	Not null	10	Updation date

11. Attendance Report Table – ‘attendance_report’:

SNO.	Fieldname	Data type	Constraints	Size	Description
1	id	varchar	Not null	50	Holds the id of the student
2	Student_id	varchar	Not null	50	Holds student id.
3.	Attendance_id	varchar	Not null	20	Holds attendance id.
4.	status	varchar	Not null	10	Holds the status of the attendance.
5	Created_At	Varchar	Not null	10	Creation date
6	Updated_at	varchar	Not null	10	Updation date

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 term 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 is 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 were 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 of developing and making changes 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 that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

SELENIUM

Selenium is a portable software testing framework for web applications. The tests can be written as HTML tables or coded in a number of popular programming languages. They can be run directly in most modern web browsers. Selenium can be deployed on Windows, Linux and Macintosh.

Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing.

Selenium Software is not just a single tool but a suite of software, each piece catering to different Selenium QA testing needs of an organization. Here is the list of tools

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- WebDriver
- Selenium Grid

Test Case-1

Project Name: Classroom Management System					
Login Test Case					
Test Case ID: Fun_1			Test Designed By: Divya Reji		
Test Priority(Low/Medium/High):High			Test Designed Date: 24-05-2022		
Module Name: Login Page			Test Executed By: Ms. Pauline Paul		
Test Title : Verify login with valid email and password			Test Execution Date: 24-05-2022		
Description: Test the Login Page					
Pre-Condition: User has valid email id and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be	Login page displayed	Pass

			displayed		
2	Provide Valid Email Id	User Name: admin@gmail.com	User should be able to Login	User Logged in and navigated to Sub admin Dashboard with records	Pass
3	Provide Valid Password	Password: admin123			
4	Click on Log In button				
5	Provide Invalid Email Id or password	Email Id: superuser1@gmail. Com Password: 1234	User should not be able to Login	Message for enter valid email id or password displayed	Pass
6	Provide Null Email Id or Password	Email Id: null Password: null			
7	Click on Sign In button				
Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database					

CODE:

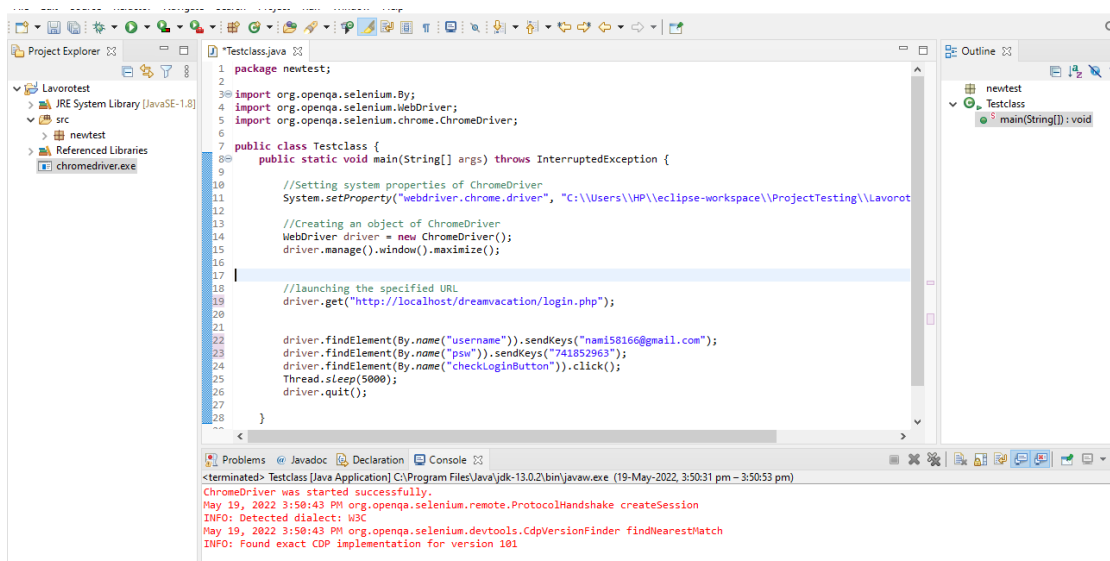
```

Package newestest;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class Testclass {
    public static void main(String[] args) throws InterruptedException {

        //Setting system properties of ChromeDriver
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\eclipse-workspace\\ProjectTesting\\Lavorotest\\chromedriver.exe");

        //Creating an object of ChromeDriver
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        //launching the specified URL
        driver.get("http://127.0.0.1:8000/");
        driver.findElement(By.name("email")).sendKeys("admin@gmail.com");
        driver.findElement(By.name("password")).sendKeys("admin123");
        driver.findElement(By.name("Button")).click();
        Thread.sleep(5000);
        driver.quit();
    }
}

```

OUTPUT SCREENSHOT:**Test Case-2****CODE:**

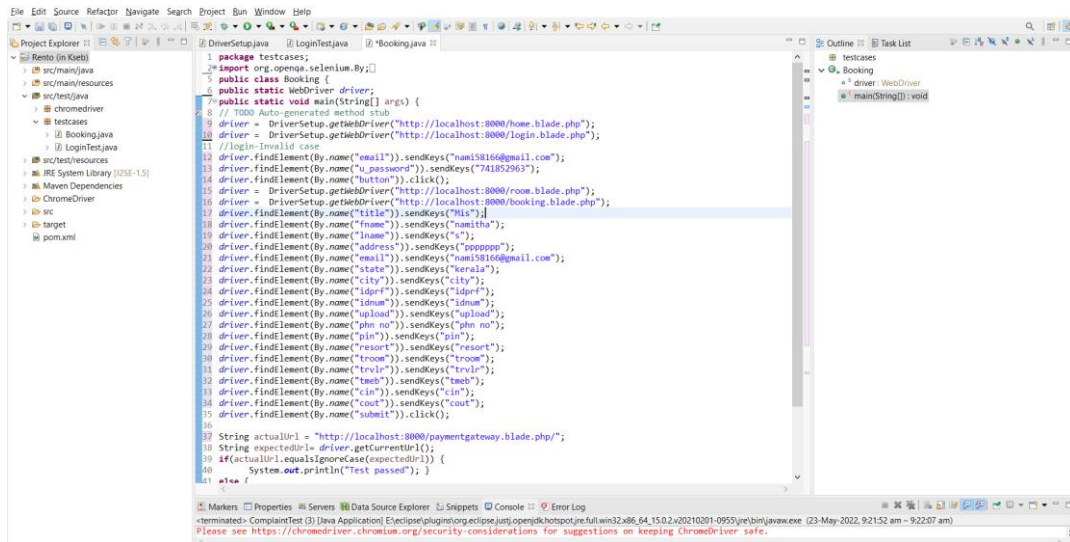
```

package testcases;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import chromedriver.DriverSetup;
public class Booking {
    public static WebDriver driver;
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        driver = DriverSetup.getWebDriver("http://localhost:8000/login_page.html");
        //login-Invalid case
        driver.findElement(By.name("email")).sendKeys("anuj@gmail.com");
        driver.findElement(By.name("u_password")).sendKeys("anuj");
        driver.findElement(By.name("button")).click();

        driver = DriverSetup.getWebDriver("http://localhost:8000/add_staff_template.html");
        driver.findElement(By.name("email")).sendKeys("anuj@gmail.com");
        driver.findElement(By.name("fname")).sendKeys("anuj");
        driver.findElement(By.name("lname")).sendKeys("sen");
        driver.findElement(By.name("username")).sendKeys("anujsen");
        driver.findElement(By.name("address")).sendKeys("calcutta");
        driver.findElement(By.name("submit")).click();

        String actualUrl = "http://localhost:8000/paymentgateway.blade.php/";
        String expectedUrl= driver.getCurrentUrl();
        if(actualUrl.equalsIgnoreCase(expectedUrl)) {
            System.out.println("Test passed"); }
        else {
            System.out.println("Test failed"); }
        driver.quit();
    }
}

```

OUTPUT SCREENSHOT:

```
1 package testcases;
2 import org.openqa.selenium.By;
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.WebElement;
5 public class Booking {
6     public static WebDriver driver;
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         driver = DriverSetup.getWebDriver("http://localhost:8000/home.blade.php");
10        driver = DriverSetup.getWebDriver("http://localhost:8000/login.blade.php");
11        //login-Invalid case
12        driver.findElement(By.name("email")).sendKeys("nam158166@gmail.com");
13        driver.findElement(By.name("password")).sendKeys("741852963");
14        driver.findElement(By.name("button")).click();
15        driver = DriverSetup.getWebDriver("http://localhost:8000/room.blade.php");
16        driver = DriverSetup.getWebDriver("http://localhost:8000/booking.blade.php");
17        driver.findElement(By.name("title")).sendKeys("HLS");
18        driver.findElement(By.name("fname")).sendKeys("namitha");
19        driver.findElement(By.name("lname")).sendKeys("s");
20        driver.findElement(By.name("address")).sendKeys("toppppp");
21        driver.findElement(By.name("email")).sendKeys("nam158166@gmail.com");
22        driver.findElement(By.name("state")).sendKeys("kerala");
23        driver.findElement(By.name("city")).sendKeys("city");
24        driver.findElement(By.name("idprf")).sendKeys("idprf");
25        driver.findElement(By.name("idnum")).sendKeys("idnum");
26        driver.findElement(By.name("upload")).sendKeys("upload");
27        driver.findElement(By.name("phn no")).sendKeys("phn no");
28        driver.findElement(By.name("pin")).sendKeys("pin");
29        driver.findElement(By.name("resort")).sendKeys("resort");
30        driver.findElement(By.name("troom")).sendKeys("troom");
31        driver.findElement(By.name("trvlr")).sendKeys("trvlr");
32        driver.findElement(By.name("taeb")).sendKeys("taeb");
33        driver.findElement(By.name("cin")).sendKeys("cin");
34        driver.findElement(By.name("cout")).sendKeys("cout");
35        driver.findElement(By.name("submit")).click();
36
37        String actualUrl = "http://localhost:8000/paymentgateway.blade.php/";
38        String expectedUrl = driver.getCurrentUrl();
39        if(actualUrl.equalsIgnoreCase(expectedUrl)) {
40            System.out.println("Test passed");
41        }
42    }
43 }
```

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 a new system design into operation, which is the process of converting a new revised system design into 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 can create 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 and changeover.

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 the application.

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 their role 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 correct the data 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. This training 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 cycle is 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 & FUTURE SCOPE

The current system working technology is working fine but there is no usage of commonly used technologies like facial recognition etc.

The future enhancements to this project could be:

- implementation of attending classes online
- taking attendance via face-recognition through webcam via real-time clicks of the users or video recordings to avoid proxies.
- Also, in-between teaching, questions can be raised amidst to ensure student's vigilant factor in online classes.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”, 2009.
- Roger S Pressman, “*Software Engineering*”, 1994.
- PankajJalote, “*Software engineering: a precise approach*”, 2006.
- James lee and Brent ware Addison, “Open source web development with LAMP”, 2003
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

- www.w3schools.com
- <https://www.djangoproject.com/>
- <http://homepages.dcc.ufmg.br/~rodolfo/es-1-03/IEEE-Std-830-1998.pdf>
- www.agilemodeling.com/artifacts/useCaseDiagram.html

CHAPTER 9

APPENDIX

9.1 Sample Code:

HodViews.py:

```
import json
from django.urls import reverse
from datetime import datetime
from email import message
from django.http import HttpResponseRedirect, JsonResponse
from django.shortcuts import render
from django.contrib import messages
from django.core.files.storage import FileSystemStorage
from authentication.forms import AddStudentForm, EditStudentForm
from authentication.models import Attendance, AttendanceReport, Courses, CustomUser,
FeedBackStaffs, FeedBackStudent, LeaveReportStaff, LeaveReportStudent, SessionYearModel, Staffs,
Students, Subjects
from django.views.decorators.csrf import csrf_exempt

def admin_home(request):
    student_count=Students.objects.all().count()
    staff_count=Staffs.objects.all().count()
    subject_count=Subjects.objects.all().count()
    course_count=Courses.objects.all().count()

    course_all=Courses.objects.all()
    course_name_list=[]
    subject_count_list=[]
    student_count_list_in_course=[]
    for course in course_all:
        subjects=Subjects.objects.filter(course_id=course.id).count()
        students=Students.objects.filter(course_id=course.id).count()
        course_name_list.append(course.course_name)
        subject_count_list.append(subjects)
        student_count_list_in_course.append(students)

    subject_all=Subjects.objects.all()
    subject_list=[]
    student_count_list_in_subject=[]
    for subject in subject_all:
        course=Courses.objects.get(id=subject.course_id.id)
        student_count=Students.objects.filter(course_id=course.id).count()
        subject_list.append(subject.subject_name)
        student_count_list_in_subject.append(student_count)

    staffs=Staffs.objects.all()
    attendance_present_list_staff=[]
    attendance_absent_list_staff=[]
    staff_name_list=[]
    for staff in staffs:
        subject_ids=Subjects.objects.filter(staff_id=staff.admin.id)
        attendance=Attendance.objects.filter(subject_id__in=subject_ids).count()
        leaves=LeaveReportStaff.objects.filter(staff_id=staff.id,leave_status=1).count()
        attendance_present_list_staff.append(attendance)
        attendance_absent_list_staff.append(leaves)
```

```

staff_name_list.append(staff.admin.username)

students_all=Students.objects.all()
attendance_present_list_student=[]
attendance_absent_list_student=[]
student_name_list=[]
for student in students_all:
attendance=AttendanceReport.objects.filter(student_id=student.id,status=True).count()
absent=AttendanceReport.objects.filter(student_id=student.id,status=False).count()
leaves=LeaveReportStudent.objects.filter(student_id=student.id,leave_status=1).count()
attendance_present_list_student.append(attendance)
attendance_absent_list_student.append(leaves+absent)
student_name_list.append(student.admin.username)

return
    render(request,"hod_template/home_content.html",{ "student_count":student_count,"staff_count":staff
    _count,"subject_count":subject_count,"course_count":course_count,"course_name_list":course_name
    _list,"subject_count_list":subject_count_list,"student_count_list_in_course":student_count_list_in_co
    urse,"student_count_list_in_subject":student_count_list_in_subject,"subject_list":subject_list,"staff_n
    ame_list":staff_name_list,"attendance_present_list_staff":attendance_present_list_staff,"attendance_a
    bsent_list_staff":attendance_absent_list_staff,"student_name_list":student_name_list,"attendance_pre
    sent_list_student":attendance_present_list_student,"attendance_absent_list_student":attendance_absen
    t_list_student})

def add_staff(request):
return render(request,"hod_template/add_staff_template.html")

def add_staff_save(request):
if request.method!='POST':
return HttpResponse("Method not allowed!")
else:
first_name = request.POST.get("first_name")
last_name = request.POST.get("last_name")
username = request.POST.get("username")
email = request.POST.get("email")
password = request.POST.get("password")
address = request.POST.get("address")

try:
user
    CustomUser.objects.create_user(username=username,password=password,email=email,last_name=la
    st_name,first_name=first_name,user_type=2)
user.staffs.address = address
user.save()
messages.success(request,"Successfully Added Staff!")
return HttpResponseRedirect(reverse("add_staff"))
except:
messages.error(request,"Failed To Add Staff")
return HttpResponseRedirect(reverse("add_staff"))

def add_course(request):
return render(request,"hod_template/add_course_template.html")

def add_course_save(request):
if request.method!='POST':

```

```
return HttpResponseRedirect("Method Not Allowed!")
else:
    course = request.POST.get("course")
    try:
        course_model = Courses(course_name=course)
        course_model.save()
        messages.success(request, "Successfully Added Course!")
        return HttpResponseRedirect(reverse("add_course"))
    except:
        messages.error(request, "Failed To Add Course")
        return HttpResponseRedirect(reverse("add_course"))

def add_student(request):
    form = AddStudentForm()
    return render(request, "hod_template/add_student_template.html", {"form": form})

def add_student_save(request):
    if request.method != 'POST':
        return HttpResponseRedirect("Method not allowed!")
    else:
        form = AddStudentForm(request.POST, request.FILES)
        if form.is_valid():
            first_name = form.cleaned_data["first_name"]
            last_name = form.cleaned_data["last_name"]
            username = form.cleaned_data["username"]
            email = form.cleaned_data["email"]
            password = form.cleaned_data["password"]
            address = form.cleaned_data["address"]
            session_year_id = form.cleaned_data["session_year_id"]
            course_id = form.cleaned_data["course"]
            sex = form.cleaned_data["sex"]

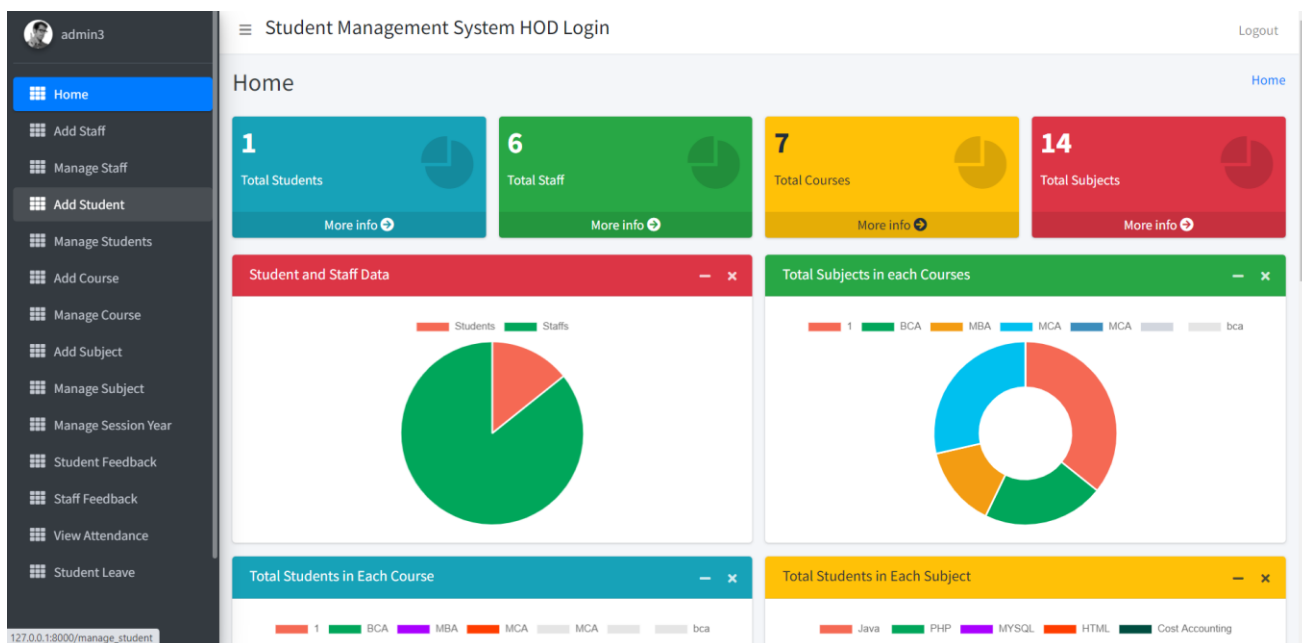
            profile_pic = request.FILES['profile_pic']
            fs = FileSystemStorage()
            filename = fs.save(profile_pic.name, profile_pic)
            profile_pic_url = fs.url(filename)
            try:
                user = CustomUser.objects.create_user(username=username, password=password, email=email, last_name=last_name, first_name=first_name, user_type=3)
                user.students.address = address
                course_obj = Courses.objects.get(id=course_id)
                user.students.course_id = course_obj
                session_year = SessionYearModel.objects.get(id=session_year_id)
                user.students.session_year_id = session_year
                user.students.profile_pic = profile_pic_url
                user.students.gender = sex
                user.save()
                messages.success(request, "Successfully Added Student!")
                return HttpResponseRedirect(reverse("add_student"))
            except:
                messages.error(request, "Failed To Add Student")
                return HttpResponseRedirect(reverse("add_student"))
            else:
                form = AddStudentForm(request.POST)
```



```
return render(request,"hod_template/add_student_template.html",{"form": form})
```

9.2 Screen Shots:

Admin HOD Home Page:



Staff Home Page:

