

ASURE
Tution Center
Student Management System

Mini Project Report

Submitted by

Ivan Joseph Regi

Reg. No.: AJC22MCA-2050

In Partial Fulfillment for the Award of the Degree of

**MASTER OF COMPUTER APPLICATIONS
(MCA TWO YEAR)**

[Accredited by NBA]

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



**AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY**

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

2022-2024

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “Asure” is the bona fide work of **IVAN JOSEPH REGI** (**Regno: AJC22MCA-2050**) in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Ankitha Philip

Internal Guide

Ms. Meera Rose Mathew

Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report “**ASURE**” is a bona fide work of done at Amal Jyothi College of Engineering, towards the partial fulfillment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: 08-12-2023

IVAN JOSEPH REGI

KANJIRAPPALLY

Reg: AJC22MCA-2050

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 **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Ankitha Philip** 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.

IVAN JOSEPH REGI

ABSTRACT

The Tutition Centre Student Management System is a web-based application developed on the Django framework, with the primary goal of improving the management of student-related information in educational institutions. This comprehensive system is designed to streamline administrative processes and enhance communication among administrators, staff, students, and parents. The user-friendly interface accommodates different roles, including Admin (Office) and Students each with specific functionalities and permissions.

The Admin(Office) module empowers administrators to add and delete students. Additionally, administrators can manage subjects and approve/deny leave applications submitted by staff and students, and gather valuable feedback from students. The system facilitates the dissemination of important events, deadlines, and announcements through notifications.

Students, through their dedicated module, can access attendance records, leave requests, and academic notifications. They have visibility into their academic progression, notifications, and the ability to provide feedback and make payments through the system. Furthermore, students can review information about their assigned staff.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	4
2.1	INTRODUCTION	5
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	6
2.4	PROPOSED SYSTEM	6
2.5	ADVANTAGES OF PROPOSED SYSTEM	7
3	REQUIREMENT ANALYSIS	8
3.1	FEASIBILITY STUDY	9
3.1.1	TECHNICAL FEASIBILITY	9
3.1.2	BEHAIORAL FEASIBILITY	9
3.1.3	OPERATIONAL FEASIBILITY	10
3.1.4	ECONOMIC FEASIBILITY	10
3.1.5	FEASIBILITY STUDY QUESTIONNAIRE	10
3.1.6	QUESTIONNAIRE TO COLLECT DETAILS	12
3.2	SYSTEM SPECIFICATION	14
3.2.1	HARDWARE SPECIFICATION	14
3.2.2	SOFTWARE SPECIFICATION	14
3.3	SOFTWARE DESCRIPTION	14
3.3.1	DJANJO	14
3.3.2	SQLITE	15
4	SYSTEM DESIGN	16
4.1	INTRODUCTION	17
4.2	UML DIAGRAM	17
4.2.1	USE CASE DIAGRAM	18
4.2.2	SEQUENCE DIAGRAM	19
4.2.3	STATE CHART DIAGRAM	21
4.2.5	CLASS DIAGRAM	22
4.2.6	OBJECT DIAGRAM	23

4.2.7	COMPONENT DIAGRAM	24
4.2.8	DEPLOYMENT DIAGRAM	25
4.3	USER INTERFACE DESIGN USING FIGMA	26
4.4	DATABASE DESIGN	28
5	SYSTEM TESTING	33
5.1	INTRODUCTION	34
5.2	TEST PLAN	34
5.2.1	UNIT TESTING	35
5.2.2	INTEGRATION TESTING	35
5.2.3	VALIDATION TESTING	36
5.2.4	USER ACCEPTANCE TESTING	36
5.2.5	AUTOMATION TESTING	36
5.2.6	SELENIUM TESTING	37
6	IMPLEMENTATION	47
6.1	INTRODUCTION	48
6.2	IMPLEMENTATION PROCEDURE	48
6.2.1	USER TRAINING	49
6.2.2	TRAINING ON APPLICATION SOFTWARE	49
6.2.3	SYSTEM MAINTENANCE	50
7	CONCLUSION & FUTURE SCOPE	51
7.1	CONCLUSION	52
7.2	FUTURE SCOPE	52
8	BIBLIOGRAPHY	54
9	APPENDIX	56
9.1	SAMPLE CODE	57
9.2	SCREEN SHOTS	67

List of Abbreviation

HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
Django	Python web framework for building web applications
JS	JavaScript
SQLite	Structured Query Language Database Management System
Ajax	Asynchronous JavaScript and XML
IDE	Integrated Development Environment
UML	Unified Modeling Language
PK	Primary Key
FK	Foreign Key
PY	Python
1NF	First Normal Form
2NF	Second Normal Form
3NF	Third Normal Form
MVT	Model-View-Template
MVC	Model-View-Controller
ORM	Object-Relational Mapping
RDBMS	Relational Database Management System

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The Tuition Centre Student Management System, a web-based application developed on the Django framework, addresses the complexities of managing student information in educational institutions. This project prioritizes a user-friendly interface for both administrators and students, featuring two core modules: Admin and Student. Administrators can efficiently add and view student details, with a notable Leave Management system enabling the review and approval/denial of student leave applications. On the student side, the system provides access to personal profiles, empowering students to manage their information and submit leave applications related to academics. The technological stack, incorporating HTML, CSS, JS for the frontend and Django for the backend, ensures a seamless and efficient user experience. By streamlining administrative processes, enhancing transparency, and fostering student engagement, the Tuition Centre Student Management System aims to revolutionize data management in educational institutions, contributing to a more organized and efficient learning environment.

1.2 PROJECT SPECIFICATION

The Tuition Centre Student Management System is a web application designed using the Django framework to simplify the management of student information in educational settings. It features an intuitive interface for both administrators and students. Administrators can easily add or view student details and manage leave applications, while students can access their profiles for information updates and submit academic leave requests. The system employs HTML, CSS, and JS for an engaging frontend and Django for a robust backend. By focusing on efficiency, transparency, and accessibility, the project aims to improve administrative processes, enhance communication, and empower students to actively participate in managing their academic profiles, ultimately contributing to a more organized and efficient learning environment. The system emphasizes security through role-based access controls and employs scalable design principles for future expansion. Overall, it offers a comprehensive solution for tuition centers to enhance organizational efficiency and communication within the educational community.

Technologies Used

Frontend: HTML, CSS, JavaScript for a responsive and interactive user interface.

Backend: Django for robust server-side development.

Database: SQLite for secure data storage and retrieval.

User Authentication: Secure mechanisms to control access to sensitive information.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The Tution Centre Student Management Systems (TC-SMS) is a web-based application designed to streamline and enhance the management of student-related information within tution centers. This System aims to provide a Comprehensive and user-friendly solution that empowers administrators, Staff, Students and Parents to efficiently manage and access critical educational data.

2.2 EXISTING SYSTEM

The existing system for managing student information in tuition centers typically relies on manual and paper-based processes. Administrators manually collect, store, and update student details using physical documents, which can lead to inefficiencies, data redundancies, and an increased likelihood of errors. Additionally, the process of handling leave applications often involves paperwork and face-to-face communication, making it time-consuming and less transparent. Students may face challenges accessing and updating their information, as the existing system lacks a centralized and user-friendly platform for them to manage their profiles and submit leave requests. Overall, the reliance on traditional methods in the existing system can hinder the smooth and streamlined management of student-related data in tuition centers.

2.2.1 NATURAL SYSTEM STUDIED

The natural system studied refers to the current state of managing student information in tuition centers, where administrators and students interact within the existing educational environment. In this system, student data is often collected manually, stored in physical records, and updated using traditional paperwork methods. The natural system also includes the process of handling student leave applications, typically involving face-to-face communication and manual approval processes. Students may have limited access to their academic profiles, with minimal autonomy in updating their information. The natural system reflects the organic, unstructured way in which student data is currently managed, often lacking the efficiency and transparency that a digitalized system could provide. The study of this natural system serves as the foundation for identifying areas of improvement and informing the design of a more streamlined and effective Tution Centre Student Management System.

2.2.2 DESIGNED SYSTEM STUDIED

The designed system studies involve envisioning and planning for the Tuition Centre Student Management System, an improved and digitalized solution to enhance the management of student information in educational institutions. In the designed system, administrators will benefit from a user-friendly interface to efficiently add, view, and manage student details. The implementation of a digitalized Leave Management system will enable a more transparent and streamlined process for reviewing and approving/denying student leave applications. Students will have increased autonomy with easy access to their profiles, allowing them to actively manage and update their academic information. The system will utilize HTML, CSS, JS for an engaging frontend, and Django for a robust backend, ensuring efficiency and scalability. The designed system aims to revolutionize data management, improving overall administrative processes, communication, and student engagement within tuition centers. The study of this designed system provides a roadmap for the development and implementation of a more effective and modern solution.

2.3 DRAWBACKS OF EXISTING SYSTEM

The drawbacks of the existing system for managing student information in tuition centers are notable. Firstly, the reliance on manual and paper-based processes leads to inefficiencies, increased likelihood of errors, and data redundancies. The lack of a centralized digital platform makes it challenging for administrators to maintain an organized and up-to-date database of student details. The traditional approach to handling leave applications through paperwork and face-to-face communication is time-consuming and lacks transparency. Students face limitations in accessing and updating their information, as the existing system offers minimal autonomy and is not user-friendly. These drawbacks collectively contribute to a system that is less efficient, prone to errors, and lacks the modern features necessary for effective student data management. The transition to a digitalized Tuition Centre Student Management System addresses these limitations by providing a more streamlined and technologically advanced solution.

2.4 PROPOSED SYSTEM

The proposed Tuition Centre Student Management System is designed to replace the existing manual processes with a modern, digitalized solution. This system introduces a user-friendly

interface accessible to both administrators and students, allowing for efficient data management. Key features include a centralized database for accurate student information, a digitalized leave management system for streamlined approval processes, and increased autonomy for students in managing their academic profiles. By utilizing HTML, CSS, JS for the frontend and Django for the backend, the proposed system aims to improve overall efficiency, transparency, and communication within tuition centers, providing a more organized and user-centric solution.

2.5 ADVANTAGES OF PROPOSED SYSTEM

The proposed Tuition Centre Student Management System offers several advantages over the existing manual system. Firstly, it introduces a digitalized and centralized database, reducing the reliance on manual paperwork and minimizing the risk of errors. The user-friendly interface enhances accessibility for both administrators and students, fostering efficient data management. The implementation of a digital leave management system streamlines the application and approval process, saving time and increasing transparency. Students benefit from increased autonomy, with easy access to their profiles for information management. The technological stack, including HTML, CSS, JS, and Django, ensures a robust and scalable system capable of accommodating future enhancements. Overall, the proposed system improves efficiency, transparency, and communication within tuition centers, contributing to a more organized and streamlined educational environment.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility is the degree to which a project can be carried out successfully. A feasibility study is conducted to assess the solution's viability, which establishes whether it is viable and implementable in the program. The feasibility study considers details like the availability of resources, software development costs, the advantages of the software to the business once it is built, and the costs associated with maintaining it. The outcome of the feasibility study should be a report recommending whether the requirements engineering, and system development process should be continued. A system is of no real value to a corporation if it does not serve its goals. Even though this may seem obvious, many organizations create systems that do not support their goals, either because they lack a clear statement of these goals, because they fail to specify the system's business requirements, or because other organizational or political factors have an impact on the procurement of the system.

3.1.1 Technical Feasibility

The technical feasibility of the Tuition Centre Student Management System is highly promising, leveraging the Django framework to ensure robustness and scalability. Django's object-relational mapping simplifies database interactions, providing a structured and efficient approach to manage student information. The web-based architecture facilitates accessibility from various devices, promoting flexibility for both administrators and students. The integration of a user-friendly interface ensures that individuals with varying technical proficiencies can navigate the system seamlessly. Additionally, the system's adaptability to accommodate future technological advancements and scalability to handle increasing data loads make it a technically viable solution for educational institutions.

3.1.2 Behavioral Feasibility

Behavioral feasibility is a critical aspect, considering the system's impact on user behavior and acceptance. The Tuition Centre Student Management System's user-friendly interface is designed with the end-users in mind, ensuring that administrators, staff, and students find the system intuitive and easy to use. The introduction of features such as notifications and feedback mechanisms aligns with user expectations, fostering positive interactions. Training sessions and user guides are planned to facilitate a smooth transition, addressing any potential resistance to change. Overall, the behavioral feasibility is optimistic, anticipating a positive

response from users due to the system's user-centric design and functionality.

3.1.3 Operational Feasibility

Operationally, the Tuition Centre Student Management System brings substantial improvements to the current manual processes prevalent in educational institutions. The system streamlines administrative tasks such as student enrollment, attendance tracking, and leave management, reducing the likelihood of errors and enhancing overall efficiency. By providing a centralized platform for communication and data management, the system promotes collaboration among administrators, staff, students, and parents. The operational feasibility is evident in the system's ability to simplify day-to-day tasks, ultimately saving time and resources for educational institutions.

3.1.4 Economic Feasibility

The economic feasibility of the Tuition Centre Student Management System is evident in its potential to generate long-term cost savings for educational institutions. By automating manual processes, the system reduces the need for extensive paperwork and manual data entry, leading to operational cost efficiencies. The centralized communication platform minimizes the reliance on traditional communication channels, further decreasing administrative costs. While there is an initial investment in development and implementation, the long-term benefits in terms of time savings, reduced errors, and improved resource allocation make the system economically viable for educational institutions aiming to enhance their management processes. In conclusion, the Tuition Centre Student Management System demonstrates strong feasibility across technical, operational, behavioral, and economic dimensions, positioning itself as a valuable solution for educational institutions seeking to modernize their administrative processes and improve overall efficiency.

3.1.5 FEASIBILITY STUDY AND QUESTIONNAIRE

1. Projects Overview

The platform aims to create a web-based application built on Django framework, designed to enhance the management of student-related information within educational Institutions. This system aims to provide a comprehensive solution for many tuition centers to efficiently manage student data, academic records, attendance and communication between students and Admin.

2. To what extend the system is proposed for?

The proposed system appears to be a comprehensive solution design to address various aspects of managing student related information within educational institution like tuition-centers. The extend to which the system is proposed for includes: The system covers core functionalities expected in such a system, including managing student records, academic information, attendance, communication and payments. It provide communication channel between staff, parents, students and admin allowing for efficient and transparent communication. Features like the chatbot facilitate real-time communication. The system covers core functionalities handles essential data management tasks such as adding, editing and deleting student, staff and parent records. It also allow for the management of subjects and payments, which are vital for any tuition centre. For Staff and parents, the system offers academic tracking capabilities, including attendance records and academic progression. This ensures that stack holders can monitor a student's educational journey effectively. The System include features for feedback and review students and parents can provide feedback, and there's even a mention of using Machine Learning for reviews, suggesting an advanced Feedback System. Efficient payment management is vital, and the system allows for handling of payments, which is especially important for tuition centres. The system integrates various technologies such as chatbots for natural language processing and machine learning for reviews, indicating an openness to utilizing modern technology for enhanced fuctionality. The system is proposed for comprehensive and covers a wide range of fuctionalities, making it suitable for managing various aspects of student information and communication within tuition centers.

3. Specify the viewers/Public which is to be involved in System?

The platform is designed to serve several viewers or users who interacted with the system. Here's a specification of viewers and public who are expect to be involved.

Users (Students):

Students can be involved in various aspects related to their education and interaction with the system.

Administrator:

Managing the entire system. They have access to all administrative features and are responsible for system.

4. Modules included in your system

- Leave

5. Who owns the system

The tuition centre initiated the development of system and fund the projects, they may retain ownership of system. In this scenario, tuition centre would have control over the system's use, modifications, and access.

6. Users Identified

- Admin
- Students

7. Related firm/Industry/Organization:

The Platform is designed for educational institutions, specially tuition centres. This system is intended to stream line and enhance the management of student-related information, communication and administrative tasks within tuition centres

3.1.6 QUESTIONNAIRE TO COLLECT DETAILS ABOUT THE PROJECT:

1. As an educational professional, how do you currently manage student information and administrative tasks in your institution?

The current method involves manual record-keeping and communication, which can be time-consuming and prone to errors. We are looking to streamline these processes through the Tuition Centre Student Management System.

2. In your experience, what challenges do administrators face when managing student-related information and communication within educational institutions?

Challenges often include the time-intensive nature of administrative tasks, difficulties in tracking student progress efficiently, and ensuring effective communication among administrators, staff, students, and parents.

3. How crucial do you think a user-friendly interface is in an educational management system, especially for individuals with different roles such as administrators and students?

A user-friendly interface is essential for easy navigation and efficient utilization of the system. It ensures that individuals with various roles can quickly access the features relevant to them, enhancing overall usability.

4. What features do you believe are most important for administrators in an educational management system, particularly in terms of student information and communication?

Key features include the ability to manage student records, handle leave applications, and facilitate communication through notifications. Additionally, features that enable event management and feedback collection are highly valuable.

5. From a staff perspective, how do you envision the Tuition Centre Student Management System improving daily administrative tasks and responsibilities?

The system is expected to streamline tasks such as student enrollment, leave management, and communication. This should result in time savings and increased accuracy in managing administrative responsibilities.

6. How do you see the system contributing to better communication between administrators, staff, students, and parents within an educational institution?

The system's notification features and centralized communication platform are expected to enhance communication by providing a reliable and efficient means to disseminate important information and announcements.

7. In your opinion, how important is it for students to have easy access to their academic information, attendance records, and the ability to submit leave requests through an online system?

Providing students with easy access to academic information and streamlined processes like leave requests is crucial for their convenience and engagement in their academic journey.

8. How can a system like this contribute to improving the overall educational experience for students?

The system can contribute by providing students with a centralized platform for accessing information, submitting requests, and engaging in communication. This enhances their overall experience by reducing administrative hassles and promoting transparency.

9. What security and privacy considerations do you think are essential for an educational management system, especially when handling sensitive student information?

Robust security measures, such as encryption and access controls, are critical to safeguard sensitive student information. Ensuring compliance with data protection regulations is also essential to maintain privacy.

10. From your professional perspective, how valuable is student feedback, and how do you think the Tuition Centre Student Management System can facilitate the collection and utilization of such feedback?

Student feedback is invaluable for continuous improvement. The system's feedback collection feature can provide administrators with insights into areas that need improvement, allowing for enhancements that align with the needs and preferences of the students.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	Intel core i5
RAM	8GB
Hard disk	512 SSD

3.2.2 Software Specification

Front End	HTML, CSS, Bootstrap, Javascript
Back End	Django Framework, Python
Database	SQLite
Client on PC	Windows 7 and above
Technologies used	Django, HTML5, Bootstrap, JS, AJAX, J Query

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO FRAMEWORK

Django, an open-source web framework, is chosen for its efficiency and versatility. Built on Python, it follows the Model-View-Controller architecture, emphasizing simplicity and flexibility. Features like Object-Relational Mapping, streamlined URL routing, and a user-friendly template engine for

contribute to its appeal. The built-in admin interface simplifies data management, and security measures enhance application integrity. Django scales effortlessly and supports REST API development. With an active community and extensive documentation, Django is a powerful toolkit for crafting sophisticated and scalable web applications.

3.3.2 SQLite

SQLite, a lightweight and serverless relational database management system, is chosen for its simplicity, portability, and efficiency. Operating as a self-contained, single-file database, SQLite requires minimal setup and eliminates the need for a separate server process. It supports ACID transactions, ensuring data integrity and reliability. Its cross-platform compatibility and dynamic typing feature make it versatile for various applications, including the Azure. SQLite is an ideal solution for projects requiring a lightweight, self-contained, and easily deployable databasesystem. Noteworthy is SQLite's cross-platform compatibility, making it versatile across different operating systems and environments. Its wide adoption is evident in applications ranging from mobile apps to web browsers, where it is utilized for local data storage. SQLite stands out as a go-to solution for projects requiring a lightweight, self-contained, and easily deployable database system, embodying simplicity without compromising on functionality and reliability.

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 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. 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. 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. They enhance communication by providing a visual representation that is easily understood by technical and non-technical stakeholders alike. They aid in analysis and design by capturing system requirements, relationships, and interactions. UML diagrams also facilitate software development by serving as a blueprint for developers to implement and test systems.

UML includes the following nine diagrams.

- Use case diagram

- Sequence diagram
- State chart diagram
- Activity diagram
- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

4.2.1 USE CASE DIAGRAM

A use case diagram serves as a visual representation illustrating the interactions and relationships among the various elements within a system. This graphical tool is particularly useful in system analysis, employing the use case methodology to identify, clarify, and organize system requirements. In the realm of system development or operation, such as in the context of a mailorder product sales and service website, use case diagrams find their application. These diagrams are part of the Unified Modeling Language (UML), a standardized notation for modeling real-world objects and systems. System objectives, ranging from overall planning requirements to validating hardware designs or testing software products, are effectively addressed through the use of use case diagrams. In a product sales environment, for instance, use cases may encompass activities like item ordering, catalog updating, payment processing, and customer relations. A comprehensive use case diagram consists of four key components: the boundary, delineating the system's scope; actors, representing individuals based on their roles in the system; use cases, depicting specific roles played by actors within and around the system; and the relationships connecting actors and use cases, capturing the dynamic interactions within the system

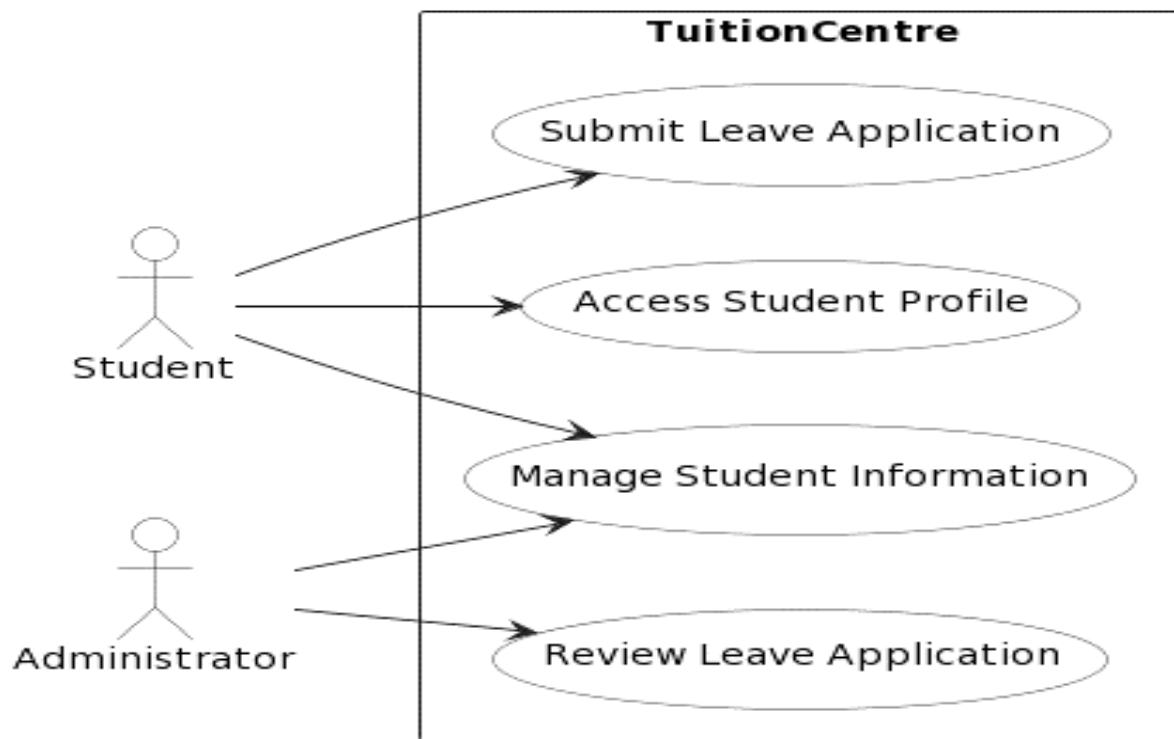


Figure 1: Use Case diagram

4.2.2 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram that illustrates the chronological order of interactions among various components within a system. Serving as a valuable tool in software engineering, sequence diagrams showcase how different entities communicate by exchanging a series of messages, often referred to as event scenarios or event scenario diagrams. These diagrams aid in understanding and describing the requirements of both new and existing systems, offering a visual representation of object control relationships and facilitating the identification of systemic issues.

Sequence Diagram Notations:

- Actors:** Represented by stick figures in UML, actors are roles that interact with the system and its objects. Actors can be external entities or human users, playing various roles within the modeled system.
- Lifelines:** Lifelines are vertical dashed lines that portray the lifespan of an object participating in the interaction. Each lifeline corresponds to an individual participant, labeled with their name, and shows the timeline of events from activation to deactivation.
- Messages:** Messages are fundamental components of sequence diagrams, denoting interactions and communication between system objects or components. They can be synchronous or asynchronous and include create, delete, self, reply, found, and lost messages.
- Guards:** Guards

are used to model conditions that restrict the flow of messages based on certain criteria. They are crucial for conveying constraints or limitations associated with a system or a specific process. Uses of Sequence Diagram: Modeling Complex Logic: Sequence diagrams are employed to model and visualize the logic of intricate functions, operations, or procedures within a system. Detailing UML Use Cases: They provide a detailed view of UML use case diagrams, enhancing comprehension of system functionalities. Understanding System Functionality: Sequence diagrams aid in comprehending the detailed functionality of current or prospective systems. Visualizing Message Flow: They offer a visual representation of how messages and tasks move between objects or components in a system, enhancing overall system understanding and communication

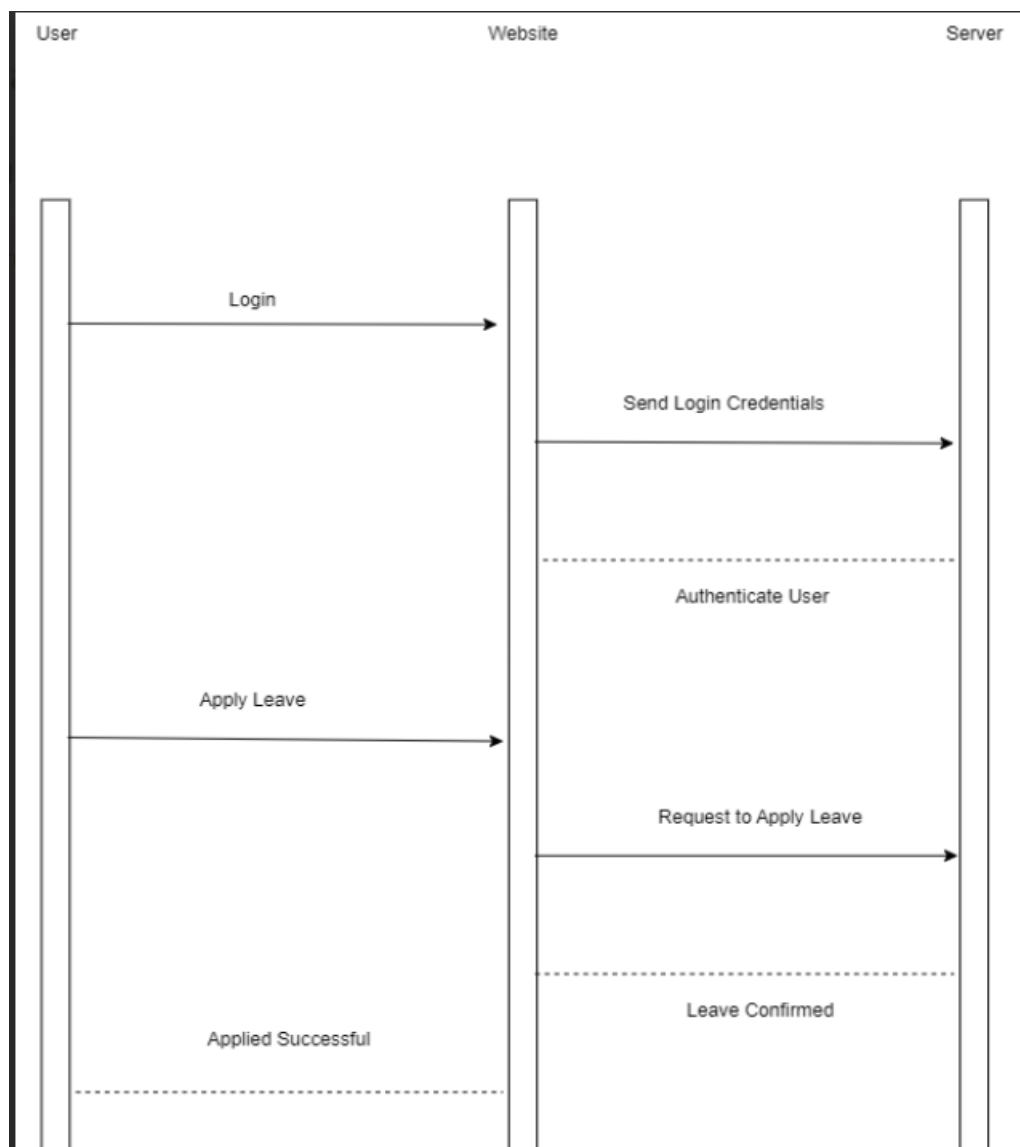


Figure 2: Sequence diagram

4.2.3 STATE CHART DIAGRAM

A state diagram, often constructed using the Unified Modeling Language (UML), serves as a visual depiction of the various states an object can inhabit and the transitions between these states. Alternatively known as a state machine diagram or state chart diagram, it falls under the category of behavioral diagrams in UML, providing insights into the system or object's behavior over time. Key elements in a State Chart Diagram include: Initial State: Marked by a solid black circle, this state signifies the system or object's starting point. State: Represented by a rectangle with rounded corners, this element characterizes the current condition of the system or object at a specific moment. Transition: Indicated by an arrow, this element illustrates the movement of the system or object from one state to another. Event and Action: Events act as triggers causing transitions, while actions denote the behavior or effect resulting from a transition. Signal: A message or trigger triggered by an event, leading to a transition within the state diagram. Final State: Concluding the State Chart Diagram, the Final State is depicted by a solid black circle with a dot inside, signifying the completion of the system or object's behavior.

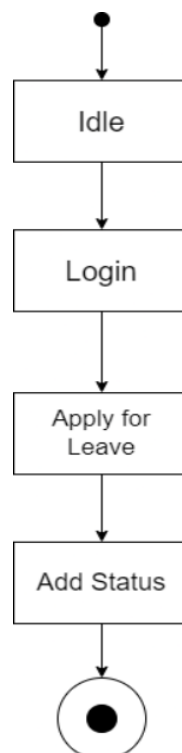


Figure 3: State Chart diagram

4.2.5 CLASS DIAGRAM

A class diagram, a fundamental static diagram in Unified Modeling Language (UML), provides a static view of an application's structure. Crucial for visualizing, describing, and documenting system components, it serves as a blueprint for software implementation in object-oriented systems. This structural diagram depicts classes, interfaces, associations, and constraints, offering insights into the properties and operations of classes. Class diagrams play a unique role by being directly convertible into object-oriented languages, aiding in the design phase and executable code generation. Widely used in software development, they help identify, organize, and understand classes, their attributes, and behaviors, fostering clarity in system architecture. Key elements include associations, inheritance relationships, and aggregations, enabling effective communication of class interactions and dependencies. Overall, class diagrams are indispensable for system design, modeling, and documentation, facilitating a comprehensive understanding of the static structure and responsibilities of different classes within the system.

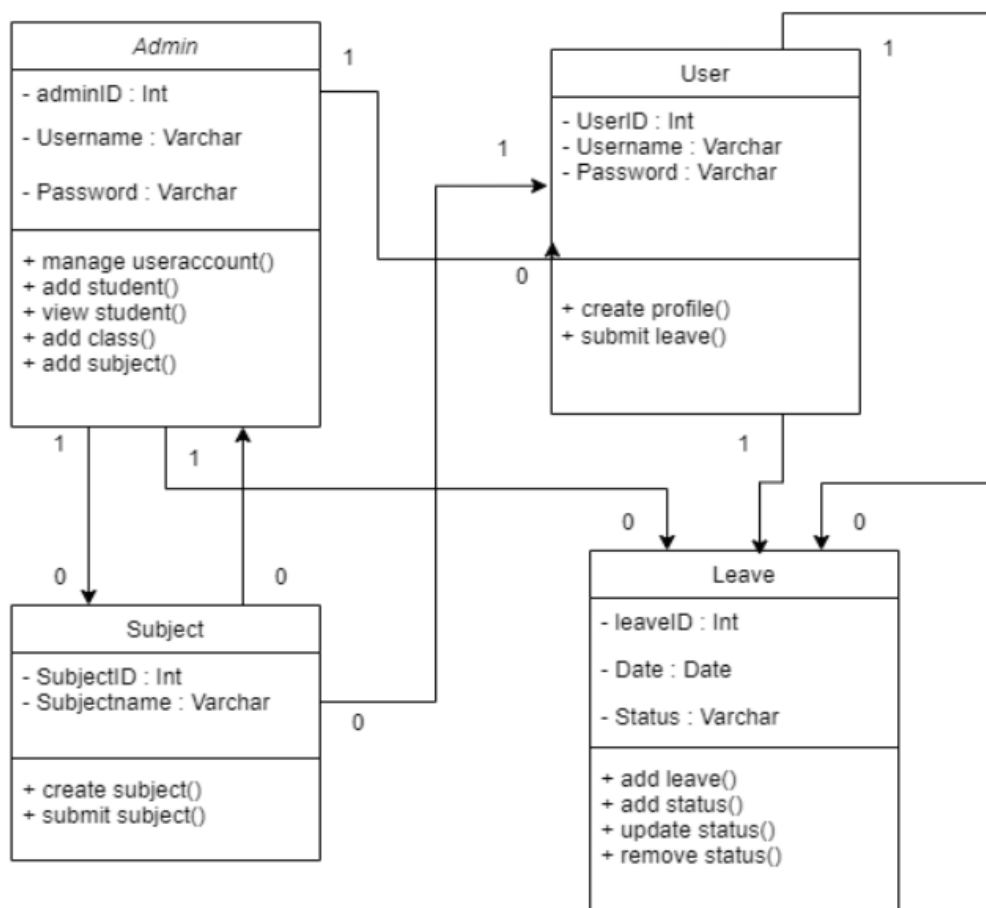


Figure 5: Class diagram

4.2.6 OBJECT DIAGRAM

Object diagrams and class diagrams are intimately linked in object-oriented modeling, with object diagrams serving as instances of class diagrams, capturing a snapshot of the system at a specific moment. Both employ similar concepts and notation to depict system structures. While class diagrams focus on modeling the overall system structure, including classes, attributes, and methods, object diagrams showcase a group of objects and their relationships at a particular instant. As a type of structural diagram in UML, an object diagram illustrates instances of classes and their connections. Key components include objects, representing specific entities with rectangles displaying their names; classes, serving as blueprints with three compartments for name, attributes, and methods; links denoting relationships; attributes detailing object properties; values specifying attribute instances; operations for object actions; and multiplicity indicating associations between classes. Object diagrams aid in visualizing relationships, attributes, and behaviors at a specific moment, facilitating a deeper understanding of system behavior and the identification of potential issues or inefficiencies.

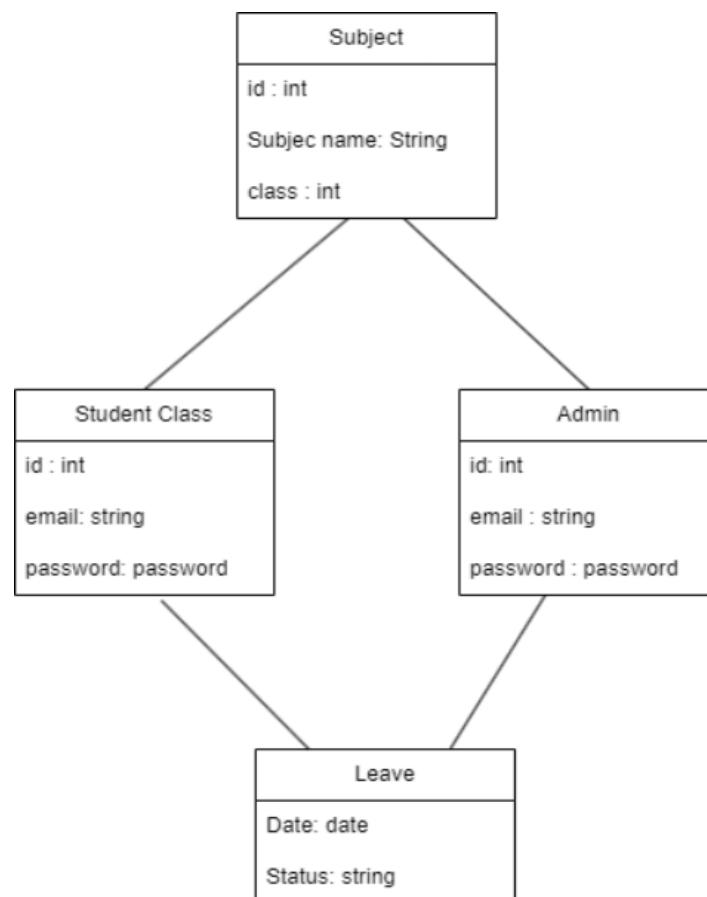


Figure 6: Object diagram

4.2.7 COMPONENT DIAGRAM

A component diagram in UML serves as a powerful tool for illustrating the interconnections among various components, showcasing how they collaboratively form larger components or intricate software systems. This diagram is particularly effective in depicting the structure of complex systems comprising multiple components. The key elements of a component diagram include:

- Component:** A self-contained unit of functionality within a system, encapsulating modular features and offering interfaces for interaction with other components. Represented as a rectangle with the component's name inside.
- Interface:** A contract specifying methods for interaction between a component and its environment or other components, depicted as a circle with the interface name inside.
- Port:** A point of interaction between a component and its environment or other components, visualized as a small square on the component's boundary.
- Connector:** A link facilitating communication or data exchange between two components, represented by a line with optional adornments and labels.
- Dependency:** A relationship indicating that one component relies on another for implementation or functionality, portrayed as a dashed line with an arrow pointing from the dependent to the independent component.
- Association:** A connection or link between two components, representing a relationship and displayed as a line with optional directionality, multiplicity, and role names.
- Provided/Required Interface:** Indicated by lollipops and half-circles, a provided interface is what a component offers to others, while a required interface is what a component needs from others.

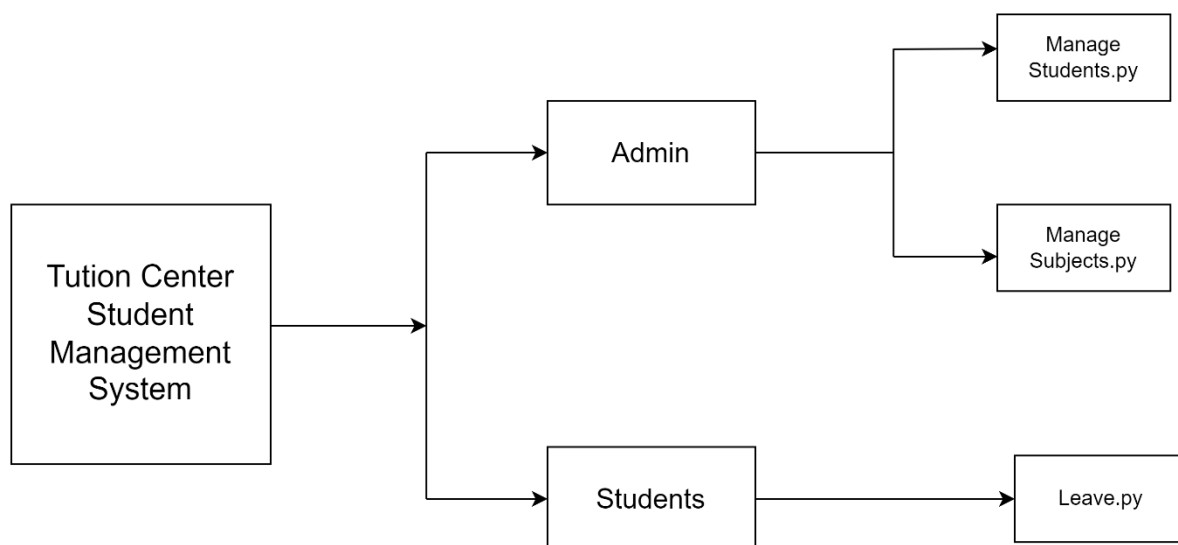


Figure 7: Component diagram

4.2.8 DEPLOYMENT DIAGRAM

A deployment diagram, a type of UML diagram, provides a static view of a system's physical architecture, focusing on the hardware used to deploy software. It maps software architecture to the physical system, showcasing how software components will be executed on nodes. Emphasizing the hardware topology rather than logical components, deployment diagrams involve nodes and their relationships, employing communication paths to illustrate node connections. Key components include: Node: A physical or virtual machine where a component or artifact is deployed, depicted as a box with the node's name inside. Component: A software element performing a specific function, represented by a rectangle with the component's name. Artifact: A physical data piece used or produced by a component, shown as a rectangle with the artifact's name. Deployment Specification: Describes how a component or artifact is deployed on a node, encompassing details like location, version, and configuration parameters. Association: Represents a deployment dependency between a node and a component or artifact, visualized as a line with optional directionality, multiplicity, and role names. Communication Path: Illustrates connections between nodes, like network connections or communication channels, displayed as a line with optional labels and adornments

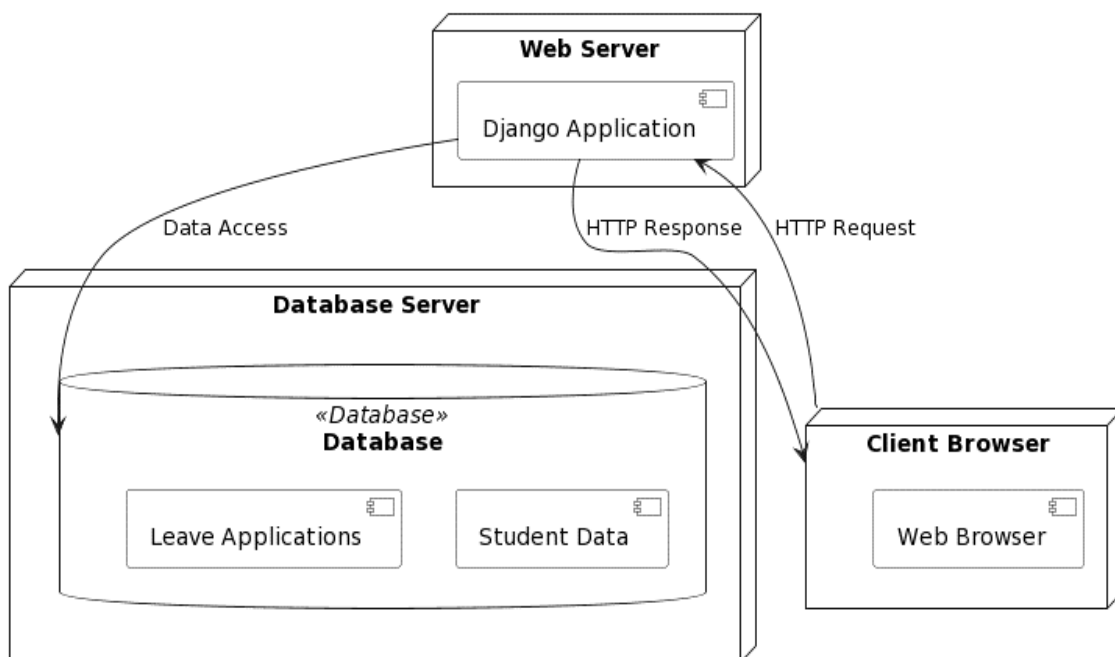
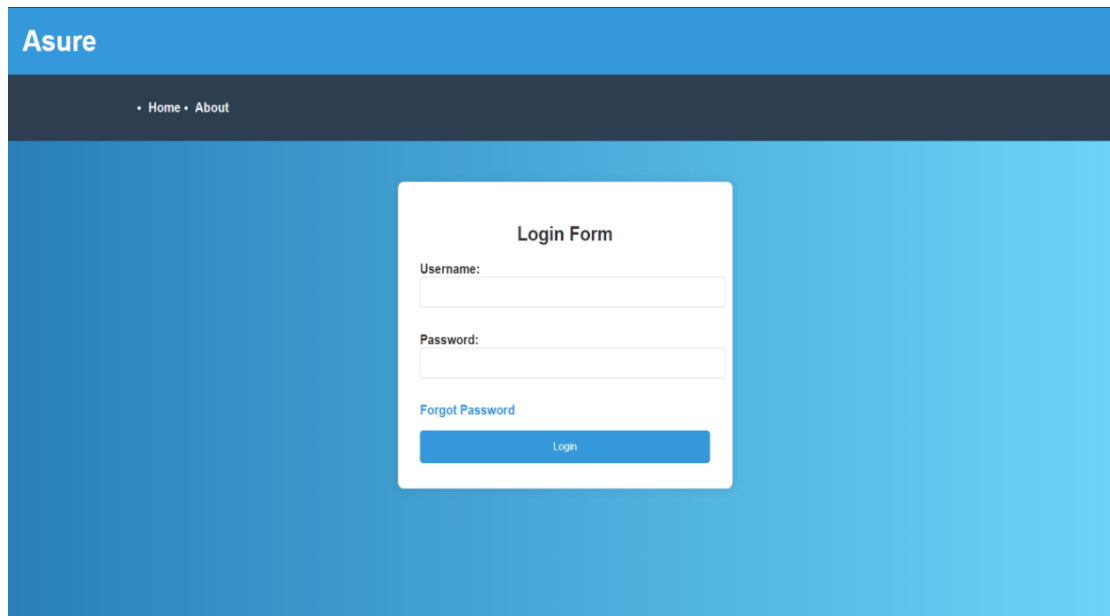


Figure 8: Deployment diagram

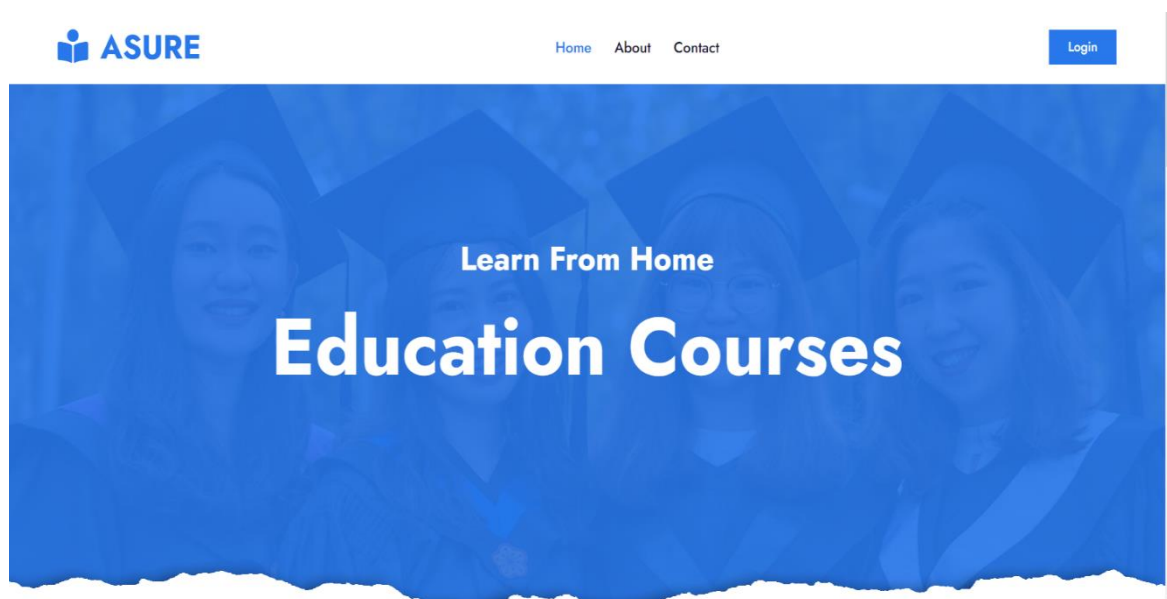
4.3 USER INTERFACE DESIGN USING FIGMA

Login page



The login page features a blue header with the 'Asure' logo. Below the header is a dark navigation bar with links for 'Home' and 'About'. The main content area has a light blue gradient background. A white login form is centered, containing fields for 'Username' and 'Password', a 'Forgot Password' link, and a blue 'Login' button.

Home Page



Add Student Page

[Back](#)

Student Details

Email:

First Name:

Last Name:

Address:

Course:

Gender:

Phone No:

Add Subject

Add a New Subject

Subject Name:

4.4 DATABASE DESIGN

A database is a structured repository of information organized to facilitate easy access, management, and updates. Ensuring the security of data is a fundamental objective in any database system. The database design process involves two key stages. In the initial stage, user requirements are gathered to create a database that aligns with those requirements as precisely as possible. This phase, known as information-level design, is conducted independently of any Database Management System (DBMS). In the subsequent stage, the design transitions from an information-level design to a specific DBMS design tailored for constructing the system. This phase, termed physical-level design, takes into account the characteristics of the chosen DBMS. In conjunction with system design, there exists the parallel discipline of database design, aiming to achieve two primary goals: data integrity, ensuring accuracy and consistency, and data independence, allowing modifications without affecting the applications using the data. The comprehensive database design process is pivotal for creating robust, efficient, and secure data management systems.

4.4.1 RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

A Relational Database Management System (RDBMS) is a critical software application for organizing and managing data in a structured manner. It stores data in tables with rows and columns, where each row represents a record, and each column is a specific attribute or field. RDBMS systems like MySQL, Oracle, or Microsoft SQL Server ensure data integrity, enforce relationships between tables, and allow for efficient data retrieval and manipulation through Structured Query Language (SQL). These systems are widely used in various applications, such as e-commerce websites, financial systems, and inventory management, due to their robustness, scalability, and ability to handle complex data structures, making them essential for modern data-driven environments.

A Relational Database Management System (RDBMS) is a cornerstone of modern data management. It structures data into tables, where each row represents a unique entry, and each column corresponds to a specific attribute, ensuring a logical and organized storage system. RDBMS systems employ complex algorithms for data retrieval and manipulation, guaranteeing data consistency and adherence to predefined relationships between tables. Popular RDBMS software, including PostgreSQL, MySQL, and Microsoft SQL Server, provides robust features for transactions, data security, and scalability. These systems are integral to a myriad of applications, from healthcare records and customer databases to inventory control and financial platforms, from

supporting the efficient storage, retrieval, and analysis of vast datasets, making them essential tools in today's data-driven world.

4.4.2 NORMALIZATION

Normalization is a database design technique used in relational database management systems (RDBMS) to eliminate data redundancy and improve data integrity. It involves organizing data in a way that reduces data duplication and ensures that relationships between tables are defined and maintained. The primary goals of normalization are:

Minimizing Data Redundancy: By breaking down data into separate tables and ensuring that each piece of data is stored only once, normalization helps reduce the chances of data inconsistencies or errors.

Ensuring Data Integrity: Normalization enforces the rules of referential integrity, which means that relationships between tables are well-defined and maintained. This ensures that data remains accurate and consistent. Normalization typically involves dividing a database into multiple related tables and using primary keys and foreign keys to establish relationships between these tables. There are several normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF), each with specific rules and requirements. The level of normalization achieved depends on the specific needs of the database and the trade-off between data redundancy and query performance. By applying normalization principles, designers can create efficient and reliable database structures that support data integrity and ease data maintenance and manipulation. There are several normal forms (NF) that define specific rules and requirements for achieving progressively higher levels of normalization. Here are the most common normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF):

First Normal Form (1NF):

- Each table has a primary key.
- All columns contain atomic (indivisible) values.
- There are no repeating groups or arrays in columns.

Second Normal Form (2NF):

- The table is in 1NF.
- All non-key attributes must be dependent on the entire primary key, not just part of it.

Third Normal Form (3NF):

- The table is in 2NF.
- There is no transitive dependency, meaning that non-key attributes are not dependent on other non-key attributes.

Boyce-Codd Normal Form (BCNF):

- A stricter version of 3NF.
- It enforces that every non-trivial functional dependency involves a super key.

Fourth Normal Form (4NF):

- Addresses multi-valued dependencies.
- Eliminates any multi-valued dependencies within the data.

Fifth Normal Form (5NF):

- Also known as Project-Join Normal Form (PJ/NF).
- Handles cases where data can be derived by joining multiple tables.

4.4.3 SANITIZATION

In Python Django, sanitization refers to the process of cleaning and validating data to ensure that it is safe and free from malicious content before it is used or stored in a database. Sanitization is a crucial security practice to prevent various forms of attacks, such as cross-site scripting (XSS) and SQL injection, which can compromise the security and integrity of a web application.

4.4.4 INDEXING

Indexing in Django is a fundamental database optimization technique. It involves creating data structures, or indexes, on specific fields to expedite data retrieval. These indexes act as signposts that enable the database to swiftly locate and fetch relevant data, especially in tables with substantial amounts of information. Django offers automatic index creation for primary keys, unique fields, and foreign keys. Additionally, developers can define custom indexes for fields frequently used in filtering, sorting, or searching. The choice of proper indexing plays a pivotal role in improving query performance, resulting in more responsive and scalable web applications. It's essential to consider application – specific query patterns and create indexes accordingly, as well as to understand the capabilities and limitations of the chosen database backend. Effective indexing is a cornerstone of efficient database operations in Django, contributing to enhanced application performance.

4.5 TABLE DESIGN

Certainly! Below are the tables based on the provided Django models:

USERS TABLE

1. Table Name: **tbl_Users**

Primary key: **User_Id**

No	Field Name	Data Type	Key Constraints	Description of the Field
1	User_Id	Int	Primary Key	Unique identifier for a user
2	Username	Varchar (255)	unique	User's username
3	Password	Varchar (255)	NOT NULL	User's password
4	is_user	BooleanField	NOT NULL	Role as User
5	is_superuser	BooleanField	NOT NULL	Role as Admin

STUDENT TABLE

2. Table Name: **tbl_Studentprofile**

Primary Key: **Student_Id**

Foreign Key: **User_Id** references table **tbl_Users**

No	Field Name	Data Type	Key Constraints	Description of the Field
1	Student_Id	Int	Primary Key	Unique identifier for a student
2	User_Id	Int	Foreign Key	Foreign key linking to User_Id
3	First Name	Varchar (255)	NULL	First name of the student
4	Last Name	Varchar (255)	NULL	Last name of the student
5	Phone Number	Varchar (255)	NULL	Phone number of the student
7	Gender	Varchar (255)	NULL	Gender of the student

8	Email	Email	DEFAULT 'default@email.com'	Email address of the student
---	-------	-------	-----------------------------	------------------------------

SUBJECT TABLE

4. Table Name: tbl_Subject

Primary key: **Subject_Id**

No	Field Name	Data Type	Key Constraints	Description of the Field
1	Subject_Id	Int	Primary Key	Unique identifier for a resource
2	Name	Varchar (255)	Not Null	Name of the resource

CLASS TABLE

5. Table Name: tbl_Class

Primary key: **Class_Id**

Foreign Key: **Student_Id** references table **tbl_Studentprofile**

No	Field Name	Data Type	Key Constraints	Description of the Field
1	Class_Id	Int	Primary Key	Unique identifier for a resource
2	Name	Varchar (255)	Not Null	Name of the resource
3	Student_Id	Int	Foreign Key	Foreign key linking to Student_Id

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is a way to check if the computer program works like it's supposed to. We use testing to make sure the software does what it is supposed to do. Validation means checking or testing things like software to make sure they meet the requirements and standards they are supposed to follow. Software testing is a way to check if a program works well. It goes along with other methods like checking and walking through the program. Validation means making sure that what the user wanted is what they got. There are several rules that can serve as testing objectives. It is the responsibility of software developer to test each of the program separate it,

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 test works well and follows its goals, it can find mistakes in the software. The test showed that it complete its system test and the following methods in the way and it ensures main part in the computer program is working like it's supposed to and is doing well. There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

5.2 TEST PLAN

A test plan suggests several required steps that need be taken to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer program, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfills the purpose for which it was intended. To solve the inherent issues with allowing the builder evaluate what they have developed, there is an independent test group (ITG). Testing's precise goals should be laid forth in quantifiable language. 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 on the basics of levels are cover within the program and of following methods in the way and it ensures main part in the computer program is working like all should be stated within the test. The levels of testing include:

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

5.2.1 Unit Testing

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a guide when testing crucial control paths to find faults inside the module's perimeter. The level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. 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. Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling paths must be put up to cleanly reroute or halt work when an error does arise. The final step of unit testing is boundary testing. Software frequently fails at its limits.

In the Sell-Soft System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had some issues, which were fixed. Each module is tested and run separately after coding. All unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome.

5.2.2 Integration Testing

Integration testing is a systematic methodology that involves the concurrent creation of the program structure and the execution of tests to uncover interface issues. The primary goal is to construct a program structure based on the design using unit-tested components, followed by comprehensive testing of the entire program. Correcting errors in integration testing can be complex due to the overall program's size, making it challenging to pinpoint the root causes of errors. The process of fixing errors may result in new issues, creating a potentially endless cycle. After completing unit testing for all system modules, integration testing is initiated to detect any

interfaces. Discrepancies in program structures are addressed, leading to the development of a cohesive and integrated program structure

5.2.3 Validation Testing or System Testing

The testing process comes to an end here. This involved testing the entire system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include system tests and black box testing. The functional requirements of the software are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer can create sets of input conditions that will fully test every program requirement. The following sorts of problems are targeted by black box testing: erroneous or missing functions, interface errors, data structure or external data access errors, performance errors, 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 major 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.

5.2.5 Automation Testing

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. Essentially, it's a test to double-check that the equipment or software does exactly what it was designed to do. It tests for bugs, defects, and any other issues that can arise with product development. Although some types of testing, such as regression or functional testing can be done manually, there are greater benefits of doing it automatically. Automation testing can be run at any time of the day. It uses scripted sequences to examine the software. It then reports on what has been found, and this information can be compared with earlier

test runs. Automation developers generally write in the following programming languages: C#, JavaScript, and Ruby.

5.2.6 Selenium Testing

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors. In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD). It allows for the creation of executable specifications in a human-readable format called Gherkin. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the requirements and helps ensure that the developed software meets the intended business goals. Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format.

Test Case 1

Code

```
from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
```

```

        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/login'
    def tearDown(self):
        self.driver.quit()

    def test_01_login_page(self):

        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(1)
        username=driver.find_element(By.CSS_SELECTOR,"input#username-
input[name='username']")
        username.send_keys("Ivan1")
        password=driver.find_element(By.CSS_SELECTOR,"input[type='password']#password-
input[name='password']")
        password.send_keys("12346")
        time.sleep(2)
        submit=driver.find_element(By.CSS_SELECTOR,"button[type='submit']")
        submit.click()
        time.sleep(2)

```

Screenshot

```

PS E:\Project Daily Update\Backup Folders\New folder\New folder\New folder\30-11-23\22-11-2023\1.MiniProject\TCSMS> python manage.py test AppOne
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:59504/devtools/browser/6c510dcd-c657-44fb-a09c-8a21764626cd
.
-----
Ran 1 test in 16.541s

OK
Destroying test database for alias 'default'...
PS E:\Project Daily Update\Backup Folders\New folder\New folder\New folder\30-11-23\22-11-2023\1.MiniProject\TCSMS>

```

Test Report

Project Name: Readify					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Ivan Joseph Regi		
Test Priority (Low/Medium/High): High			Test Designed Date: 06-12-2023		
Module Name: Login Screen			Test Executed By: Ms. Ankitha Philip		
Test Title: User Login			Test Execution Date: 06-12-2023		
Description: Verify login with valid email and password					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)

1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Email	Username: Ivan1	User should be able to login	User is logged in and navigated to corresponding Home Page	Pass
3	Provide Valid Password	Password: 12346			
4	Click on Login button				
Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database					

Test Case 2:

Code

```

from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/login'

    def tearDown(self):
        self.driver.quit()

    def test_01_login_page(self):

        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(1)
        username=driver.find_element(By.CSS_SELECTOR,"input#username-
input[name='username']")
        username.send_keys("admin")
        password=driver.find_element(By.CSS_SELECTOR,"input[type='password']#password-
input[name='password']")
        password.send_keys("admin")

```

```
time.sleep(2)
submit=driver.find_element(By.CSS_SELECTOR,"button[type='submit']")
submit.click()
time.sleep(2)
redirect=driver.find_element(By.CSS_SELECTOR,"a.nav-
link[name='add_student'][href='/update-students/']")
redirect.click()
time.sleep(2)
AddStudent=driver.find_element(By.CSS_SELECTOR,"a[href='/myhome']")
AddStudent.click()
time.sleep(2)
email=driver.find_element(By.CSS_SELECTOR,"input[type='email']#email[name='email
']][required]")
email.send_keys("raju@gmail.com")

time.sleep(2)
fname=driver.find_element(By.CSS_SELECTOR,"input[type='text'][name='first_name']["
required]")
fname.send_keys("Raju")
time.sleep(2)
lname=driver.find_element(By.CSS_SELECTOR,"input[type='text'][name='last_name']["
equired]")
lname.send_keys("joseph")
time.sleep(2)
address=driver.find_element(By.CSS_SELECTOR,"textarea[name='address'][rows='4']["
equired]")
address.send_keys("Kurishmootil(H), Pala")
time.sleep(2)
pno=driver.find_element(By.CSS_SELECTOR,"input[type='text'][name='phone_no']["r
equired]")
pno.send_keys("9876543210")
time.sleep(2)
course=driver.find_element(By.CSS_SELECTOR,"select[name='course']["required]")
course.click()
course1=driver.find_element(By.CSS_SELECTOR,"select[name='course']["required]")
course1.click()
time.sleep(2)
gender=driver.find_element(By.CSS_SELECTOR,"select[name='gender']["required]")
gender.click()
gender1=driver.find_element(By.CSS_SELECTOR,"select[name='gender']
option[value='male']")
gender1.click()
time.sleep(2)
submit=driver.find_element(By.CSS_SELECTOR,"button[type='submit']")
submit.click()
time.sleep(2)
```


Screenshot

```

PS E:\Project Daily Update\Backup Folders\New folder\New folder\New folder\30-11-23\22-11-2023\1.MiniProject\TCSMS> python manage.py test AppOne
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:59992/devtools/browser/653ea29a-15fd-40cf-9f40-89cc56e43b63
.
-----
Ran 1 test in 37.926s

OK
Destroying test database for alias 'default'...
PS E:\Project Daily Update\Backup Folders\New folder\New folder\New folder\30-11-23\22-11-2023\1.MiniProject\TCSMS>

```

Test report

Project Name: Readify					
User Enquiry					
Test Case ID: Test_2			Test Designed By: Ivan Joseph Regi		
Test Priority (Low/Medium/High): High			Test Designed Date: 06-12-2023		
Module Name: Add Student			Test Executed By: Ms. Ankitha Philip		
Test Title: Add Student			Test Execution Date: 06-12-2023		
Description: Admin adding Students					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Email	Username: admin	Admin should be able to login	Admin is logged in and navigated to corresponding Home Page	Pass
3	Provide Valid Password	Password: admin			
4	Click on Login button				
5	Click on the Add student			Add Student is displayed	Pass
6	Input new email	Email: Raju@gmail.com		Email is updated	Pass
7	Input new First name	First Name: Raju		First name updated	Pass

8	Input new Last name	Last Name: Joseph		Last name updated	Pass
9	Input new Address	Address: Kurishmootil(H) Pala		Address is updated	Pass
10	Input new Class	Class: 7		Class updated	Pass
11	Input new Gender	Gender: Male		Gender updated	Pass
12	Input new Phone no:	Phone no: 8086596811		Phone no. updated	Pass
Post-Condition: Admin is validated with database and successfully login into account.					

Test Case 3:

Code

```

from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/login'

    def tearDown(self):
        self.driver.quit()

    def test_01_login_page(self):

        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(1)
        username=driver.find_element(By.CSS_SELECTOR,"input#username-
input[name='username']")
        username.send_keys("admin")

```

```

password=driver.find_element(By.CSS_SELECTOR,"input[type='password']#password-
input[name='password']")
password.send_keys("admin")
time.sleep(2)
submit=driver.find_element(By.CSS_SELECTOR,"button[type='submit']")
submit.click()
time.sleep(2)
redirect=driver.find_element(By.CSS_SELECTOR,"a.nav-link[href='/add_subject/']")
redirect.click()
time.sleep(2)
sname=driver.find_element(By.CSS_SELECTOR,"input[type='text']#subject-
name[name='subject_name'][required]")
sname.send_keys("Hindi2")
submit=driver.find_element(By.CSS_SELECTOR,"button[type='submit']")
submit.click()
time.sleep(2)

```

Screenshot

```

PS E:\Project Daily Update\Backup Folders\New folder\New folder\New folder\30-11-23\22-11-2023\1.MiniProject\TCSMS> python manage.py test AppOne
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:59504/devtools/browser/6c510dcd-c657-44fb-a09c-8a21764626cd
.
-----
Ran 1 test in 16.541s

OK
Destroying test database for alias 'default'...
PS E:\Project Daily Update\Backup Folders\New folder\New folder\New folder\30-11-23\22-11-2023\1.MiniProject\TCSMS>

```

Test report

Project Name: Readify					
Update Profile					
Test Case ID: Test_3			Test Designed By: Ivan Joseph Regi		
Test Priority (Low/Medium/High): High			Test Designed Date: 06-12-2023		
Module Name: Add Subject			Test Executed By: Ms. Ankitha Philip		
Test Title: Add Subject			Test Execution Date: 06-12-2023		
Description: Admin adding Subjects					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Email	Username: admin	User should be	User is logged in and	Pass

3	Provide Valid Password	Password: admin	able to login	navigated to corresponding Home Page	
4	Click on Login button				
5	Click on Add Subject			Subject Page is displayed	Pass
6	Input Subject Field	Subject: Mathematics		Subject is entered	Pass
7	Click on Add Subject			Subject is added	Pass
Post-Condition: Admin Add Subject successfully.					

Test Case 4:

Code

```

from datetime import datetime
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/login'

    def tearDown(self):
        self.driver.quit()

    def test_01_login_page(self):

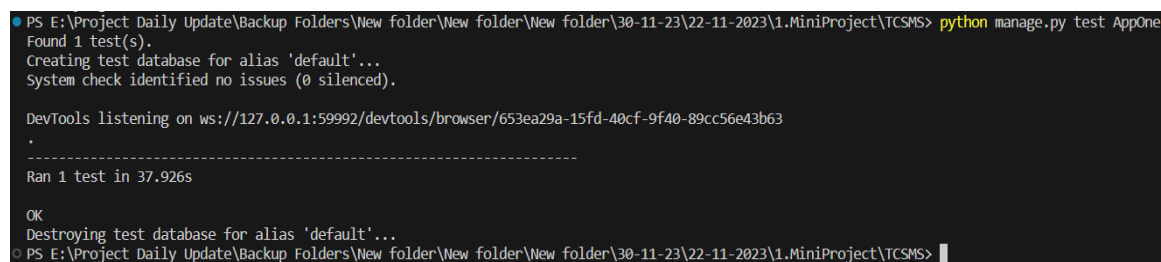
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(1)
        username=driver.find_element(By.CSS_SELECTOR,"input#username-
input[name='username']")
        username.send_keys("admin")

```

```
password=driver.find_element(By.CSS_SELECTOR,"input[type='password']"#password-input[name='password']")
```

```
password.send_keys("admin")
time.sleep(2)
submit=driver.find_element(By.CSS_SELECTOR,"button[type='submit']")
submit.click()
time.sleep(2)
redirect=driver.find_element(By.CSS_SELECTOR,
"a.nav-link[href='/add_class/']")
redirect.click()
time.sleep(2)
Addclass=driver.find_element(By.CSS_SELECTOR,"input[type='text']"[name='class_name']"[required]")
Addclass.send_keys("8")
time.sleep(2)
submit=driver.find_element(By.CSS_SELECTOR,"button[type='submit']")
submit.click()
time.sleep(2)
```

Screenshot



```
PS E:\Project Daily Update\Backup Folders\New folder\New folder\New folder\30-11-23\22-11-2023\1.MiniProject\TCSMS> python manage.py test AppOne
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:59992/devtools/browser/653ea29a-15fd-40cf-9f40-89cc56e43b63
.
-----
Ran 1 test in 37.926s

OK
Destroying test database for alias 'default'...
PS E:\Project Daily Update\Backup Folders\New folder\New folder\New folder\30-11-23\22-11-2023\1.MiniProject\TCSMS>
```

Test report

Project Name: Readify					
Update Profile					
Test Case ID: Test_3			Test Designed By: Ivan Joseph Regi		
Test Priority (Low/Medium/High): High			Test Designed Date: 06-12-2023		
Module Name: Add Class			Test Executed By: Ms. Ankitha Philip		
Test Title: Add Class			Test Execution Date: 06-12-2023		
Description: Admin adding Class					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)

1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Email	Username: admin	User should be able to login	User is logged in and navigated to corresponding Home Page	Pass
3	Provide Valid Password	Password: admin			
4	Click on Login button				
5	Click on Add Class			Class Page is displayed	Pass
6	Input Class Field	Class: Eight		Class is entered	Pass
7	Click on Add Class			Class is added	Pass
Post-Condition: Admin Add Class successfully.					

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 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

The implementation procedures for this project involve a systematic and phased approach to bring the online crime reporting portal to fruition. This complex system, designed to cater to the needs of various stakeholders, requires careful planning and execution of errors while entering the data. To commence the implementation, the project team will initiate a detailed requirements analysis phase. This stage involves comprehensive discussions with law enforcement officials, control room staff, prison wardens, and potential end-users to determine their specific needs and amount

expectations. The results of this analysis will inform the development of a detailed system specification and design and take place to convert from existing system to the new system design. The development phase will follow, involving the creation of the portal's architecture, databases, and user interfaces. It's during this stage that the functionalities outlined in the project's scope, including user management, crime reporting, and communication features, will be integrated into the system. The portal will be designed with a user-friendly interface to ensure ease of use for all categories of users. Simultaneously, a dedicated team will work on the incorporation of advanced technologies, such as machine learning capabilities for crime trend analysis and predictive features. These technologies will be implemented carefully to ensure the portal's robustness and security. Once the development phase is complete, rigorous testing will be conducted to identify and rectify any bugs or issues. The portal will undergo extensive quality assurance and user acceptance testing to ensure that it meets all the necessary requirements and is free of vulnerabilities. Deployment of the system will be carried out in a controlled manner, ensuring minimal disruption to the existing processes, and allowing for gradual adaptation by the involved stakeholders. Comprehensive training sessions will be conducted to familiarize law enforcement personnel, control room staff, prison wardens, and registered users with the portal's functionality and features. Post-deployment, the project team will continue to provide support and maintenance, addressing any issues that may arise and making necessary improvements as the system matures. Regular updates and security measures will be implemented to safeguard user data and maintain the portal's efficiency.

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 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. 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

In conclusion, the Tuition Centre Student Management System stands as a comprehensive solution, leveraging the robust capabilities of the Django framework to address the intricate challenges in managing student information within educational institutions. By providing a user-friendly interface for administrators and students alike, the system facilitates efficient handling of student accounts and streamlines processes, particularly in the critical area of managing leave requests. The administrative module empowers office administrators by allowing them to seamlessly add, view, and manage student details, ensuring a centralized hub for student information. On the student side, the system empowers students to take control of their academic journey by providing easy access to their profile information. The functionality for submitting leave applications related to academics further enhances the student experience, offering a convenient platform for communication and request submission. With its frontend crafted using HTML, CSS, and JS, the system ensures an intuitive and visually appealing user interface. Coupled with the robust backend powered by Django, it achieves a harmonious balance between aesthetics and functionality. This not only enhances the overall efficiency of tuition centers but also fosters improved organization and accessibility in the intricate realm of student data management. In essence, the Tuition Centre Student Management System emerges as a testament to the synergy between technology and education, providing a seamless, effective, and user-centric platform that caters to the unique needs of educational institutions, administrators, and students alike.

7.2 FUTURE SCOPE

The Tuition Centre Student Management System, built on Django, presents a promising future with avenues for expansion and enrichment. To enhance data-driven decision-making, the integration of advanced analytics for predictive insights into student performance and trend analysis could be explored. Additionally, a robust communication framework, featuring real-time messaging and notifications, would foster seamless interaction among administrators, teachers, and students. Furthering the project's reach, the inclusion of e-learning modules and mobile application development would empower users with accessible and dynamic educational resources, supporting a comprehensive and flexible learning experience. Looking ahead, financial management modules and attendance tracking systems could streamline administrative tasks, ensuring efficient fee collection and attendance monitoring. The system's security can be fortified

through measures like two-factor authentication and regular security audits. Customizable dashboards for users, integration with external platforms, and a feedback loop for continuous improvement are integral to ensuring the Tuition Centre Student Management System stays adaptive, user-friendly, and aligned with evolving educational needs. As the educational landscape continues to transform, this project remains poised for growth and innovation in meeting the ever-changing demands of educational technology.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Harvard Business Review - Unlocking Value through Questions.
- AlsoAsked - People Also Ask keyword research tool: A tool to find questions people also ask related to a specific topic.
- Google Support - How to search on Google: Tips on effective Google searches, including finding quick answers.
- Qualtrics - Multiple Choice Question: Information about the multiple-choice question type for surveys.
- MyGreatLearning - 180+ SQL Interview Questions and Answers in 2023: A compilation of SQL interview questions for job seekers.

WEBSITES:

- <https://www.chat.openai.com/>
- <https://www.djangoproject.com/>
- <https://www.getbootstrap.com/>
- <https://www.greeksforgreek.org/>

CHAPTER 9

APPENDIX

9.1 Sample Code

Index.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Asure - Tution Centre - Student management System</title>
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    <meta content="Free HTML Templates" name="keywords">
    <meta content="Free HTML Templates" name="description">
    <!-- Favicon -->
    <link href="{% static 'img/favicon.ico' %}" rel="icon">
    <!-- Google Web Fonts -->
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
        href="https://fonts.googleapis.com/css2?family=Jost:wght@500;600;700&family=Open+Sans:wght@400;600&display=swap" rel="stylesheet">
    <!-- Font Awesome -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css"
        rel="stylesheet">
    <!-- Libraries Stylesheet -->
    <link href="{% static 'lib/owlcarousel/assets/owl.carousel.min.css' %}" rel="stylesheet">
    <!-- Customized Bootstrap Stylesheet -->
    <link href="{% static 'css/style.css' %}" rel="stylesheet">
</head>
<body>
    <!-- Topbar Start -->
    <div class="container-fluid bg-dark">
        <div class="row py-2 px-lg-5">
            <div class="col-lg-6 text-center text-lg-left mb-2 mb-lg-0">
                <div class="d-inline-flex align-items-center text-white">
```

```

        <small><i class="fa fa-phone-alt mr-2"></i>+9876543210</small>
        <small class="px-3">|</small>
        <small><i class="fa fa-envelope mr-2"></i>Asure@gmail.com</small>
    </div>
</div>
<div class="col-lg-6 text-center text-lg-right">
    <div class="d-inline-flex align-items-center">
        <a class="text-white px-2" href="">
            <i class="fab fa-facebook-f"></i>
        </a>
        <a class="text-white px-2" href="">
            <i class="fab fa-twitter"></i>
        </a>
        <a class="text-white px-2" href="">
            <i class="fab fa-linkedin-in"></i>
        </a>
        <a class="text-white px-2" href="">
            <i class="fab fa-instagram"></i>
        </a>
        <a class="text-white pl-2" href="">
            <i class="fab fa-youtube"></i>
        </a>
    </div>
</div>
</div>
<div>
</div>
<!-- Topbar End -->
<!-- Navbar Start -->
<div class="container-fluid p-0">
    <nav class="navbar navbar-expand-lg bg-white navbar-light py-3 py-lg-0 px-lg-5">
        <a href="{ % url 'home' % }" class="navbar-brand ml-lg-3">
            <h1 class="m-0 text-uppercase text-primary"><i class="fa fa-book-reader mr-3"></i>ASURE</h1>

```

```

</a>
<button type="button" class="navbar-toggler" data-toggle="collapse" data-
target="#navbarCollapse">
    <span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse justify-content-between px-lg-3"
id="navbarCollapse">
    <div class="navbar-nav mx-auto py-0">
        <a href="{ % url 'home' % }" class="nav-item nav-link active">Home</a>
        <a href="{ % url 'about' % }" class="nav-item nav-link">About</a>

        <a href="{ % url 'contact' % }" class="nav-item nav-link">Contact</a>
    </div>
    <a href="{ % url 'login' % }" class="btn btn-primary py-2 px-4 d-none d-lg-
block">Login</a>
    { % comment % } <a href="{ % url 'slogin' % }" class="btn btn-primary py-2 px-4
d-none d-lg-block">StudentLogin</a> { % endcomment % }
    </div>
</nav>
</div>
<!-- Navbar End -->

<!-- Header Start -->
<div class="jumbotron jumbotron-fluid position-relative overlay-bottom" style="margin-
bottom: 90px;">
    <div class="container text-center my-5 py-5">
        <h1 class="text-white mt-4 mb-4">Learn From Home</h1>
        <h1 class="text-white display-1 mb-5">Education Courses</h1>
        <div class="mx-auto mb-5" style="width: 100%; max-width: 600px;">
            <div class="input-group">
                </div>
            </div>
        </div>
    </div>

```

```

    </div>
</div>
<!-- Header End -->

<!-- About Start -->
<div class="container-fluid py-5">
    <div class="container py-5">
        <div class="row">
            <div class="col-lg-5 mb-5 mb-lg-0" style="min-height: 500px;">
                <div class="position-relative h-100">
                    
                </div>
            </div>

            <div class="col-lg-7">
                <div class="section-title position-relative mb-4">
                    <h2 class="d-inline-block position-relative text-secondary text-uppercase pb-2">About Us</h2>
                    <h1 class="display-4">First Choice For Online Education Anywhere</h1>
                </div>
                <p>For a student's success in their personal and academic life, learning plays a
pivotal role. At present, when the digital age is expanding. It is helping the students and
parents in deeply shaping their future ways. Asure motivates and guides the student by being
a constant learning partner in the fast changing world. So that they can face their hurdles
with confidence and preparation.S</p>

            </div>
        </div>
    </div>
</div>
<!-- About End -->

```

```

<!-- Feature Start -->
<div class="container-fluid bg-image" style="margin: 90px 0;">
  <div class="container">
    <div class="row">
      <div class="col-lg-7 my-5 pt-5 pb-lg-5">
        <div class="section-title position-relative mb-4">
          <h6 class="d-inline-block position-relative text-secondary text-uppercase pb-2">Why Choose Us?</h6>
          <h1 class="display-4">Why You Should Start Learning with Us?</h1>
        </div>
        <p class="mb-4 pb-2">Learning is the process of gaining new skills, knowledge, understanding, and values. This is something people can do by themselves, although it's generally made easier with education: the process of helping someone or a group of others to learn.

```

With educational support, learning can happen more efficiently. Education is also how we collect and share all the skills and knowledge we learn individually. Benefitting from education instead of having to build new skills and knowledge by ourselves from scratch is part of what it means to live in a society instead of in isolation.

Learning and education impart more than just knowledge and skills. They also transmit the values, attitudes, and behaviours we have decided to share. </p>

Learning is the process of gaining new skills, knowledge, understanding, and values. This is something people can do by themselves, although it's generally made easier with education: the process of helping someone or a group of others to learn.

With educational support, learning can happen more efficiently. Education is also how we collect and share all the skills and knowledge we learn individually. Benefitting from education instead of having to build new skills and knowledge by ourselves from scratch is part of what it means to live in a society instead of in isolation.

Learning and education impart more than just knowledge and skills. They also transmit the values, attitudes, and behaviours we have decided to share.

```

</div>
<div class="col-lg-5" style="min-height: 500px;">

```

```
<div class="position-relative h-100">
    
</div>
</div>
</div>
</div>
</div>

<!-- Feature Start -->

<!-- Back to Top -->
<a href="#" class="btn btn-lg btn-primary rounded-0 btn-lg-square back-to-top"><i class=
"fa fa-angle-double-up"></i></a>

<!-- JavaScript Libraries -->
<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.bundle.min.js"></script>
<script src="lib/easing/easing.min.js"></script>
<script src="lib/waypoints/waypoints.min.js"></script>
<script src="lib/counterup/counterup.min.js"></script>
<script src="lib/owlcarousel/owl.carousel.min.js"></script>

<!-- Template Javascript -->
<script src="js/main.js"></script>
</body>
</html>
```

View_student_details.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Asure - Tuition Centre - Student Management System</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      font-family: 'Open Sans', sans-serif;
      background-color: #f5f5f5;
    }
    .container {
      max-width: 1000px;
      margin: 20px auto;
      background-color: #fff;
      padding: 40px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    .student-table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }
    .student-table th,
    .student-table td {
      border: 1px solid #ddd;
      padding: 12px;
```

```
        text-align: left;
    }
    .student-table th {
        background-color: #f2f2f2;
    }
    .profile-pic {
        max-width: 50px;
        max-height: 50px;
        border-radius: 50%;
        margin-right: 10px;
    }
    h2 {
        color: #333;
    }
    .no-image {
        color: #888;
    }
    a {
        display: inline-block;
        padding: 10px 20px;
        margin: 10px;
        text-decoration: none;
        color: #333;
        background-color: #fff;
        border: 2px solid #333;
        border-radius: 5px;
        transition: background-color 0.3s, color 0.3s;
    }
    a:hover {
        background-color: #333;
        color: #fff;
    }
}
```

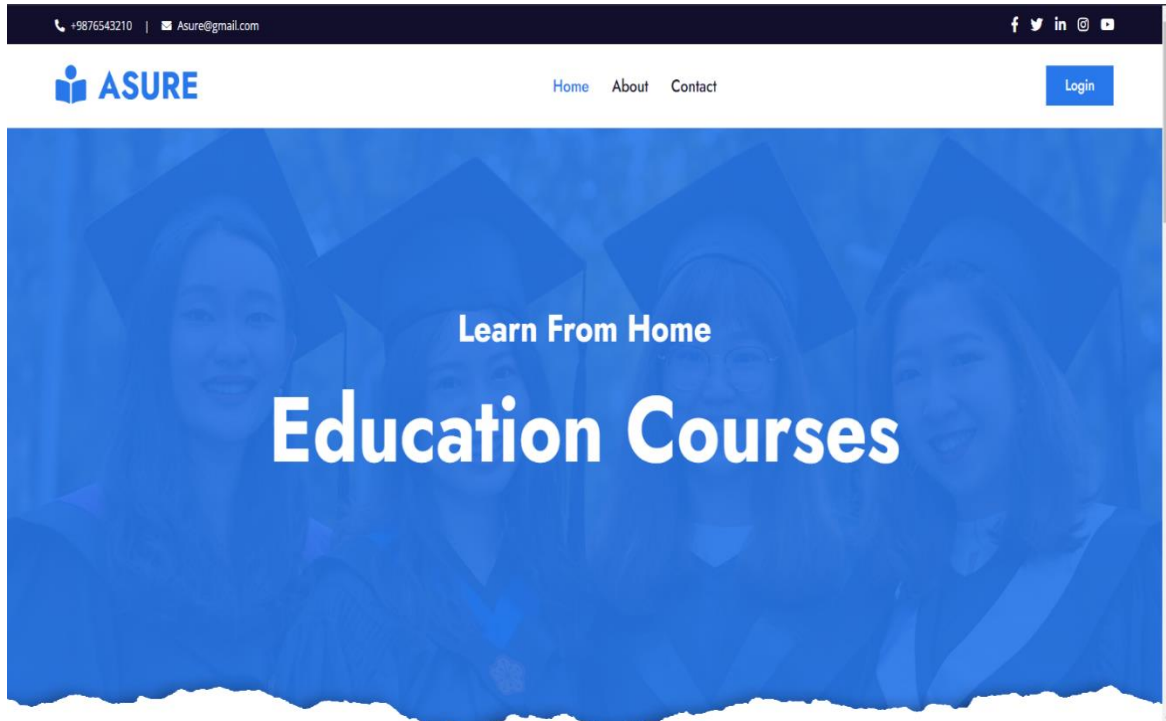


```
button {
    position: absolute;
    top: 10px;
    left: 10px;
    padding: 10px;
    color: black;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
button a {
    text-decoration: none;
    color: black;
}
/* Add additional styles as needed */
</style>
</head>
<body>
<button><a href="{ % url 'generate_user' % }">Back</a></button>
<div class="container">
<h2>Student Details</h2>
<table class="student-table">
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Address</th>
<th>Course</th>
<th>Gender</th>
<th>Phone No</th>
<th>Guardian's Name</th>
<th>Guardian's Phone No</th>
<th>Session Year</th>
```

```
<th>Profile Pic</th>
<!-- Add other fields as needed -->
</tr>
</thead>
<tbody>
  {% for student in students %}
    <tr>
      <td>{{ student.first_name }}</td>
      <td>{{ student.last_name }}</td>
      <td>{{ student.address }}</td>
      <td>{{ student.course }}</td>
      <td>{{ student.gender }}</td>
      <td>{{ student.phone_no }}</td>
      <td>{{ student.guardian_name }}</td>
      <td>{{ student.guardian_phone_no }}</td>
      <td>{{ student.session_year }}</td>
      <td>
        {% if student.profile_pic %}
          
        {% else %}
          <span class="no-image">No Image</span>
        {% endif %}
      </td>
      <!-- Add other fields as needed -->
    </tr>
  {% endfor %}
</tbody>
</table>
</div>
</body>
</html>
```

9.2 SCREEN SHOTS

Home





Login

The screenshot shows the login page of the Asure website. It has a blue header with the 'Asure' logo. Below the header is a dark blue navigation bar with a breadcrumb trail '• Home • About'. The main content area has a light blue background. In the center, there is a white login form box. The form is titled 'Login Form' and contains two input fields: 'Username:' and 'Password:'. Below the password field is a blue link that says 'Forgot Password'. At the bottom of the form is a blue button labeled 'Login'.

View Students

Back

First Name	Last Name	Address	Course	Gender	Phone No	Guardian's Name	Guardian's Phone No	Session Year	Profile Pic
Ivan	JR	Ooramattom Koodalloor	2	male	08086596811	Denny	8281647265	12-03-2021	
Ivan	JR	Ooramattom Koodalloor	3	male	08086596811	Regi	8281647265	29-01-23	

View Class

Back

Class 1

Class 2

Class 3

Class 4

Class 5

