

Project Report
on
ATTENDANCE MANAGEMENT SYSTEM
at
U. V. Patel College of Engineering



Project Guide:

Prof. Nishi P Patwa

Prepared By:

Ms. Suhada Khan (22172012038)

Mr. Trushar Panchal (22172012020)

Mr. Jaydeep Saxena (22172012036)

B.Tech Semester VI
(Computer Engineering – Institute of Technology)

January-June 2024

Submitted to,
Department of Computer Engineering – Institute of Technology
U.V. Patel College of Engineering
Ganpat University, Ganpat Vidyanagar, Mehsana - 384012

U.V. PATEL COLLEGE OF ENGINEERING



08/06/2024

CERTIFICATE

TO WHOM SO EVER IT MAY CONCERN

This is to certify that **Ms. SUHADA KHAN** student of **B.Tech. Semester VI (Computer Engineering – Institute of Technology)** has completed her full semester Capstone Project-II titled “**ATTENDANCE MANAGEMENT SYSTEM**” satisfactorily in partial fulfillment of the requirement of Bachelor of Technology degree of Computer Engineering at Ganpat University, Ganpat Vidyanagar, Mehsana in the year 2023-2024.

Project Guide

Sign

Prof. Nishi P Patwa

Dr. Paresh M. Solanki
Head, Computer Engineering

U.V. PATEL COLLEGE OF ENGINEERING



08/06/2024

CERTIFICATE

TO WHOM SO EVER IT MAY CONCERN

This is to certify that **Mr. TRUSHAR PANCHAL** student of **B.Tech. Semester VI (Computer Engineering - Institute of Technology)** has completed his full semester Capstone Project-II titled “**ATTENDANCE MANAGEMENT SYSTEM**” satisfactorily in partial fulfillment of the requirement of Bachelor of Technology degree of Information Technology at Ganpat University, Ganpat Vidyanagar, Mehsana in the year 2023-2024.

Project Guide

Sign

Prof. Nishi P Patwa

Dr. Paresh M. Solanki
Head, Computer Engineering

U.V. PATEL COLLEGE OF ENGINEERING



08/06/2024

CERTIFICATE

TO WHOM SO EVER IT MAY CONCERN

This is to certify that **Mr. JAYDEEP SAXENA** student of **B.Tech. Semester VI (Computer Engineering - Institute of Technology)** has completed his full semester Capstone Project-II work titled “**ATTENDANCE MANAGEMENT SYSTEM**” satisfactorily in partial fulfillment of the requirement of Bachelor of Technology degree of Computer Engineering- Artificial Intelligence at Ganpat University, Ganpat Vidyanagar, Mehsana in the year 2023-2024.

Project Guide

Sign

Prof. Nishi P Patwa

Dr. Paresh M. Solanki
Head, Computer Engineering

ACKNOWLEDGEMENT

We wish to express my sincere gratitude to the U. V. Patel College of Engineering, Ganpat University, GanpatVidhya Nagar. By providing a way to innovate our mind in such a way that helping me to build a project by bringing the old technology's inner strength, long lastingness', reliability and new technology's performance, security, speed to gather to make a **Attendance Management System** by using the concept of Python and SQLite

We are sincerely very thankful to Prof. Nishi P Patwa , Faculty at Ganpat University. By providing very valuable time and giving guidance and opportunities to work together as a team, and to do a project in the Web DevelopmentField. I Am very thankful to him for encouraging me to innovate my project to the highest peak.

We are very thankful to all the competitive students who are also very helpful to work together not as a competitor but as a family member joyfully.

Regards,
Ms. Suhada Khan
Mr. Trushar Panchal
Mr. Jaydeep Saxena

ABSTRACT

This project proposes a web-based **Attendance Management System** (AMS) for a specific department in an educational institution. Built with **Python, Qt5, and SQLite**, the system uses a secure login to provide eight user interfaces for different roles. To ensure data integrity, faculty removal automatically assigns students to another faculty for the same course. The department head can create users, manage faculty and students, and assign courses. Faculty record attendance for their students, while students can view their overall and individual course attendance. All data is stored securely in a **SQLite database**. This customizable system offers a flexible solution for department-level attendance management.

Table Of Contents

1 Introduction	1
1.1 Project Overview	1
1.2 Background.....	1
1.3 Purpose	1
1.3.1 Problem Statement.....	1
1.3.2 Project Aim	1
1.3.3 Project Objectives.....	1
1.4 Project Scope	1
1.5 Impact, Significance, and Contribution	2
 2 Literature Review	 3
2.1 Efficiency Improvement.....	3
2.2 Accuracy Enhancement	3
2.3 Enhanced Data Accessibility	3
2.4 Student Engagement.....	3
2.5 Administrative Benefits.....	3
2.6 Integration with Learning Management Systems	3
2.7 Data Security Concerns	3
2.8 User Acceptance	3
2.9 Mobile Accessibility	4
2.10 Future Trends.....	4
2.11 Integration with Biometric Authentication.....	4
2.12 Gamification Elements.....	4
2.13 Predictive Analytics.....	4
2.14 Blockchain Technology.....	4
2.15 Personalized Notifications.....	4
2.16 Wearable Technology Integration.....	4
 3 Software Requirement Specification (SRS).....	 5
3.1 Functional Requirements.....	5
3.1.1 Functional Requirements of module admin.....	5
3.1.1.1 Signup for admin.....	5
3.1.1.2 Signin for admin.....	5
3.1.1.3 User Management.....	5
3.1.1.3.1 Manage Faculty	5
3.1.1.3.2 Manage Students	5
3.1.1.4 Course Management.....	5
3.1.1.5 Course-Faculty Association Management.....	5

3.1.1.6	Student-Course Enrollment Management	5
3.1.1.7	Report Generation	5
3.1.2	Functional Requirements of module student	5
3.1.3	Functional Requirements of module faculty	6
3.2	Non-functional Requirements	6
3.2.1	Reliability	6
3.2.2	Security	6
3.2.3	Efficiency	6
3.2.4	Usability	6
3.2.5	Economic	6
4	Software and Hardware Requirements	7
4.1	Hardware Requirements	7
4.2	Software Requirements	7
4.2.1	Technology Required	7
4.2.1.1	Python 3	7
4.2.1.2	PyQt5	8
4.2.1.3	SQLite	8
5	Process Model	9
6	Feasibility Analysis	12
6.1	Technical Feasibility	12
6.2	Time Schedule Feasibility	12
6.3	Operational Feasibility	12
6.4	Implementation Feasibility	12
6.5	Economical Feasibility	13
7	Project Plan	14
7.1	Project Goal	14
7.2	Target Users	14
7.3	Development Methodology	14
7.4	Project Deliverables	14
7.5	Project Management Tools	14
7.6	Project Phases and Activities	14
8	System Design	17
8.1	ER Diagram	17
8.2	Use Case Diagram	18
8.3	Activity Diagram	19
8.3.1	Admin – Activity Diagram	20

8.3.2 Student – Activity Diagram	21
8.3.3 Faculty – Activity Diagram	22
8.4 Component Diagram	23
8.5 State Chart Diagram	24
8.6 Class Diagram	25
8.7 Sequence Diagram.....	26
8.8 Data Flow Diagram	27
8.8.1 DFD – 0 Level / Context Diagram	27
8.8.2 DFD – 1 Level	28
8.8.3 DFD – 2 Level	29
8.8.3.1 DFD – 2 Level – Admin.....	30
8.8.3.2 DFD – 2 Level – Student	30
8.8.3.3 DFD - 2 Level – Faculty	31
8.9 Data Dictionary	32
9 Implementation Details	34
9.1 Flowchart of Implementation	34
9.2 Actual Program Code	34
9.2.1 Main.py.....	34
9.2.2 Database.py.....	37
9.2.3 Info.py	40
9.2.4 Mainmenugui.py.....	41
10 Testing	45
11 User Manual.....	47
11.1 Used Software	47
11.2 Screenshots	49
12 Conclusion and Future Work.....	55
13 Annexure	56
13.1 Glossary of Terms and Conditions	56
13.2 References	56
13.3 About Tools and Technology.....	56
13.4 About College (UVPCE).....	56

List of Figures

Figure 1 ER Diagram.....	19
Figure 2 Use Case Diagram.....	20
Figure 3 Activity Diagram.....	21
Figure 4 Admin Activity Diagram.....	22
Figure 5 Student Activity Diagram	23
Figure 6 Faculty-Activity Diagram	24
Figure 7 Component Diagram.....	25
Figure 8 Statechart Diagram.....	26
Figure 9 Class Diagram	27
Figure 10 Sequence Diagram	28
Figure 11 DFD Diagram Level-0	29
Figure 12 DFD Diagram Level-1	30
Figure 13 DFD Diagram Level-2	31
Figure 14 DFD Level-2 Admin	32
Figure 15 DFD Level-2 Student.....	32
Figure 16 DFD Level-2 Faculty	33
Figure 17 Flowchart of Implementation.....	36
Figure 18 Login Page	50
Figure 19 Manage Students and Teachers	51
Figure 20 Add Teachers/Faculty.....	51
Figure 21 Add Students	52
Figure 22 Delete Teachers/Faculty	52
Figure 23 Delete Students	53
Figure 24 Attendance Sheet.....	55
Figure 25 Attendance Report.....	55
Figure 26 Information about Database	56

List Of Tables

Table 1 Course Table	34
Table 2 Teacher Table	34
Table 3 Student Table	34
Table 4 Attendance Table	34
Table 5 Studies Table.....	35

1. Introduction

1.1 Project Overview

This project report details the development of a web-based Attendance Management System (AMS) for a specific department within an educational institution. This AMS aims to simplify and streamline the process of recording and managing student attendance.

1.2 Background

Traditionally, attendance tracking in educational institutions relies on manual methods, which are often time-consuming, prone to errors, and inefficient. This project proposes a solution to address these shortcomings by introducing an automated AMS.

1.3 Purpose

1.3.1 Problem Statement

Manual attendance tracking suffers from several drawbacks:

- **Time Consumption:** Manually recording attendance takes valuable time away from both faculty and students.
- **Error Prone:** Manual entries are susceptible to human error, leading to inaccurate data.
- **Inefficiency:** Data retrieval and analysis of manual attendance records can be cumbersome and inefficient.

1.3.2 Project Aim

This project aims to develop a user-friendly and efficient web-based AMS to address the limitations of manual attendance tracking.

1.3.3 Project Objectives

The specific objectives of this project are:

- To design and develop a secure web-based application for attendance management.
- To provide functionalities for user roles such as department head, faculty, and students.
- To enable faculty to efficiently record student attendance for assigned courses.
- To allow students to view their overall and date-wise attendance for their chosen subjects.
- To store attendance data securely within a SQLite database for easy retrieval and analysis.

1.4 Project Scope

This AMS is designed for a specific department within an educational institution. The scope encompasses user roles like department head, faculty, and students belonging to that department. While future enhancements could expand functionalities, the initial focus is on core attendance management within this defined scope.

1.5 Impact, Significance, and Contributions

This AMS offers significant benefits by:

- **Enhancing Efficiency:** Automating attendance recording saves valuable time for faculty and students.
- **Improving Accuracy:** Eliminating manual entries reduces errors and ensures data accuracy.
- **Facilitating Data Analysis:** Stored electronic data allows for easy retrieval and analysis to gain insights into attendance patterns.
- **User-Friendly Interface:** The web-based interface provides a convenient and accessible platform for all users.

This project contributes to the field of educational technology by providing a practical solution for efficient and reliable attendance management.

2.Literature Review

Attendance Management Systems (AMS) have become essential tools for educational institutions worldwide, offering a multitude of advantages. This review explores key findings from relevant literature to highlight the significance of AMS and inform the development of this project.

2.1 Efficiency Improvement

- Zhang et al. (2018) emphasize that AMS significantly improves efficiency by automating attendance tracking processes. This reduces manual effort and time consumption for faculty and administrators.

2.2 Accuracy Enhancement

- Anwar et al. (2020) highlight that automated attendance systems tend to be more accurate than traditional methods. This minimizes errors and discrepancies in record-keeping, ensuring data integrity.

2.3 Enhanced Data Accessibility

- Khan et al. (2019) discusses the benefits of digital attendance systems, which provide real-time access to attendance data. This allows stakeholders to effectively monitor and analyze attendance trends, enabling informed decision-making.

2.4 Student Engagement

- Research by Chen et al. (2017) suggests that the implementation of attendance management systems can positively influence student engagement and participation in classes. This can potentially lead to improved learning outcomes.

2.5 Administrative Benefits

- Dharma lingam & Vadivel (2021) explore the administrative benefits of AMS. Streamlining attendance management tasks allows educational institutions to allocate resources more efficiently, potentially leading to cost savings.

2.6 Integration with Learning Management Systems

- Alvi et al. (2020) investigate the integration of AMS with existing learning management systems. This enhances the functionality and usability of both systems, creating a more seamless experience for users.

2.7 Data Security Concerns

- While digital attendance systems offer numerous benefits, Shrestha et al. (2021) raise concerns regarding data security and privacy. Robust security measures are crucial to protect sensitive student information.

2.8 User Acceptance

- Dwivedi et al. (2019) emphasize the importance of user acceptance and satisfaction for the successful implementation of attendance management systems. User-centric design principles are essential to ensure user adoption and satisfaction.

2.9 Mobile Accessibility

- Suthar et al. (2018) discuss the value of mobile-friendly interfaces for attendance management systems. This enhances accessibility, allowing users to manage attendance remotely, improving flexibility and convenience.

2.10 Future Trends

- Rai et al. (2022) explore emerging technologies that are expected to revolutionize attendance management systems. Biometric authentication and artificial intelligence hold promise for offering advanced features and functionalities in the future.

2.11 Integration with Biometric Authentication

- Building on the discussion of future trends, Gupta and Sharma (2023) delve into the integration of biometric authentication with AMS. Biometric data such as fingerprints or facial recognition could further enhance security and streamline the attendance tracking process.

2.12 Gamification Elements

- Recent studies by Lee and Park (2024) suggest incorporating gamification elements into AMS to incentivize and reward students for consistent attendance. Gamified features could include badges, leaderboards, or virtual rewards, fostering a more engaging and enjoyable experience.

2.13 Predictive Analytics

- Investigating the potential of predictive analytics in attendance management, Patel and Patel (2023) propose utilizing machine learning algorithms to forecast attendance patterns. By analyzing past data and contextual factors, institutions can proactively identify students at risk of absenteeism and intervene early to support their academic progress.

2.14 Blockchain Technology

- Advancing discussions on data security, Kumar and Singh (2023) explore the application of blockchain technology in AMS. By decentralizing attendance records and implementing immutable ledgers, blockchain can offer unprecedented levels of data integrity and transparency, addressing concerns related to tampering and unauthorized access.

2.15 Personalized Notifications

- To enhance communication between students and educators, Li et al. (2024) advocate for personalized notification systems within AMS. Utilizing student preferences and behavioral data, these systems can deliver timely reminders and alerts about upcoming classes or attendance requirements, promoting accountability and engagement.

2.16 Wearable Technology Integration

- Building on the concept of mobile accessibility, Park and Kim (2023) propose integrating AMS with wearable technology such as smartwatches or RFID-enabled accessories. This integration would enable seamless attendance tracking without the need for manual input, further streamlining the process for both students and instructors.

3. Software Requirement Specifications (SRS)

3.1 Functional requirements

The functional requirements define the specific functionalities the system must provide. These requirements are categorized by user role (Admin, Student, Faculty).

3.1.1 Functional requirements of module Admin

3.1.1.1 SignUp for Admin: The system allows the department head to create a new admin account with appropriate credentials.

3.1.1.2 SignIn for Admin: The system enables the admin to log in using their registered credentials.

3.1.1.3 User Management: The admin can:

3.1.1.3.1 Manage Faculty:

- Add new faculty members to the system, assigning them to specific courses.
- Edit or delete existing faculty information.
- View a list of all registered faculty members.

3.1.1.3.2 Manage Students:

- Add new students to the system.
- Edit or delete existing student information.
- View a list of all registered students.

3.1.1.4 Course Management: The admin can:

- Add new courses offered by the department.
- Edit or delete existing courses.
- View a list of all offered courses.

3.1.1.5 Course-Faculty Association Management: The admin can assign faculty members to teach specific courses.

3.1.1.6 Student-Course Enrollment Management: The admin can enroll students in courses offered by the department. This may involve functionalities like allowing students to choose their courses from a list of available options.

3.1.1.7 Report Generation: The admin can generate reports on student attendance for specific courses, time periods, or individual students.

3.1.2 Functional requirements of module Student

- **SignUp for Student:** The system allows students to register for an account using their unique identification information.
- **SignIn for Student:** The system enables students to log in using their registered credentials.
- **View Profile:** Students can view their profile information, including their name, ID, enrolled courses, and contact details (optional).
- **View Attendance:** Students can view their overall and date-wise attendance for their enrolled courses. This may include functionalities like displaying attendance percentages, absence details, and lecture dates.

3.1.3 Functional requirements of module Faculty

- **SignIn for Faculty:** The system enables faculty members to log in using their registered credentials.
- **View Assigned Courses:** Faculty can view a list of courses they are assigned to teach.
- **Take Attendance:** Faculty can record student attendance for their assigned courses. This may involve functionalities like marking students present or absent, potentially with the option to specify reasons for absences.
- **View Attendance Report (Optional):** Faculty may be able to view attendance reports for their courses, providing insights into overall attendance patterns within their classes.

3.2 Non-Functional Requirements

Non-functional requirements define the overall characteristics of the system. These encompass:

3.2.1 Reliability: The system should be reliable and function consistently without unexpected crashes or downtime.

3.2.2 Security: The system must ensure data security, including user authentication, authorization, and encryption of sensitive information like student IDs and attendance records.

3.2.3 Efficiency: The system should be efficient in terms of performance and resource utilization. Attendance recording and data retrieval should be quick and responsive.

3.2.4 Usability: The user interface should be user-friendly and intuitive for all user roles (admin, student, faculty). It should be well-documented and easy to navigate.

3.2.5 Economic: The system should be cost-effective to develop, implement, and maintain. Utilizing open-source technologies like Python, QT5, and SQLite contributes to this goal.

4. Software and Hardware Requirements

This section outlines the hardware, software, process model, and prototype model for the Attendance Management System (AMS).

4.1 Hardware Requirements

The AMS has the following minimum hardware requirements:

- Processor: Intel Core i3 or equivalent
- RAM: 4GB or higher
- Storage: 100MB free disk space
- Display: 1280x800 resolution or higher

4.2 Software Requirements

The AMS requires the following software:

- Operating System: Windows 7 or later, macOS, or Linux
- Python 3.x (latest version recommended)
- QT5 framework (version compatible with Python 3.x)
- SQLite database engine

4.2.1 Technology Required

The technology required are the following as Python3, PyQt5, and SQLite:

4.2.1.1 Python 3

- **General-purpose, high-level programming language:** Python is known for its readability, due to its clear syntax that emphasizes code clarity. This makes it a popular choice for beginners and experienced programmers alike.
- **Interpreted language:** Python code doesn't need to be compiled into machine code beforehand. Instead, an interpreter reads the code line by line and executes it directly. This allows for faster development cycles and easier debugging.
- **Extensive standard library:** Python comes with a robust standard library that provides built-in modules for various tasks, including file handling, networking, web development, scientific computing, and more. This reduces the need for external libraries in many cases.
- **Cross-platform:** Python code can run on different operating systems like Windows, macOS, Linux, and more, without major modifications.
- **Object-oriented and functional programming:** Python supports both object-oriented and functional programming paradigms, offering flexibility in how you approach problem-solving.
- **Large and active community:** Python boasts a vast and supportive community that provides extensive documentation, tutorials, and libraries. This makes it easier to find help and learn new things.
- **Wide range of applications:** Python is used in diverse areas, including web development (Django, Flask), data science (NumPy, Pandas), machine learning (Scikit-learn, TensorFlow), scripting, automation, and more.

4.2.1.2 PyQt5

- **Cross-platform GUI toolkit:** PyQt5 is a powerful library that allows you to create graphical user interfaces (GUIs) for desktop applications using Python. It's a Python binding for the Qt framework, which is a mature and widely used GUI toolkit.
- **Native look and feel:** PyQt5 applications seamlessly integrate with the native look and feel of the operating system on which they run, providing a familiar user experience.
- **Rich set of widgets:** PyQt5 offers a comprehensive collection of UI components, including buttons, text boxes, layouts, menus, tables, and more. This makes it easy to build complex and interactive GUIs.
- **Event-driven programming:** PyQt5 uses an event-driven approach, where the application responds to user interactions and other events from the operating system. This enables a dynamic and responsive user experience.
- **Integration with other PyQt modules:** PyQt5 complements other PyQt modules that extend its functionality. These modules provide support for specific platforms (e.g., PyQt for Android, PyQt for iOS) or specialized tasks (e.g., PyQtChart for data visualization).
- **Popular for GUI development:** PyQt5 is a popular choice for building desktop applications with Python due to its ease of use, flexibility, and extensive capabilities.

4.2.1.3 SQLite

- **Lightweight, self-contained, embedded database engine:** SQLite is a relational database management system (RDBMS) that stores data in a single file. This makes it easy to distribute, embed in applications, and use without the need for a separate database server.
- **Simple and efficient:** SQLite prioritizes simplicity and efficiency. It's a good choice for applications that need a local database solution without complex administration or setup.
- **ACID compliance:** SQLite adheres to the ACID properties of transactions (Atomicity, Consistency, Isolation, Durability), ensuring data integrity.
- **SQL support:** SQLite supports the industry-standard SQL (Structured Query Language) for data manipulation. This allows you to query, insert, update, and delete data using familiar SQL syntax.
- **Zero-configuration:** SQLite doesn't require any configuration or installation, making it a quick and convenient option for getting started with databases.
- **Limited scalability:** While SQLite is ideal for smaller datasets and embedded applications, its scalability is limited compared to robust database servers for very large datasets.
- **Wide range of use cases:** SQLite is often used in desktop applications, mobile applications, web applications (backend storage), data analysis, and as a lightweight solution for various tasks.

5. Process Model

An attendance management system in Python, the Agile process model would be most suitable. These approaches accommodate changing requirements, provide flexibility, and allow for iterative development and continuous feedback. Given the nature of software projects, where requirements can evolve based on user feedback, Agile methodologies help ensure that the final product meets user needs and expectations effectively.

The development of the AMS was conducted using the Agile process model. Agile is an iterative and incremental approach that promotes flexibility, customer collaboration, and continuous improvement.

The meaning of Agile is swift or versatile. The "Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations or parts and do not involve long-term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration, and the scope of each iteration are clearly defined in advance.

Each iteration is considered a short time "frame" in the Agile process model, typically lasting from one to four weeks. The division of the entire project into smaller parts helps to minimize project risk and reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle, including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

•Phases of the Agile Model

The Agile model comprises several phases, as follows:

1. Requirements Gathering

In this phase, the requirements are defined. Business opportunities are explained, and the time and effort needed to build the project are planned. Based on this information, the technical and economic feasibility of the project is evaluated.

2. Design the Requirements

Once the project is identified, stakeholders work together to define the requirements. Tools such as user flow diagrams or high-level UML diagrams are used to illustrate the functionality of new features and how they integrate with the existing system.

3. Construction/Iteration

After defining the requirements, the team begins the construction phase. Designers and developers start working on the project, aiming to deploy a working product. The product undergoes various stages of improvement, starting with simple, minimal functionality.

4. Testing/Quality Assurance

In this phase, the Quality Assurance team examines the product's performance and looks for bugs to ensure the product meets the required standards and performs as expected.

5. Deployment

The team releases the product into the user's work environment for actual use.

6. Feedback

After the product is released, feedback is gathered from the users. The team then works on this feedback to make necessary improvements.

- Agile Testing Methods
 - Scrum

Scrum is an agile development process focused on managing tasks in team-based development conditions. It includes three roles:

1. **Scrum Master:** Sets up the team, arranges meetings, and removes obstacles.
2. **Product Owner:** Creates the product backlog, prioritizes tasks, and is responsible for functionality distribution.
3. **Scrum Team:** Manages and organizes the work to complete each sprint.

➤ eXtreme Programming (XP)

This methodology is used when customers frequently change demands or are unsure about the system's performance. It emphasizes continuous feedback and flexible responses to changing requirements.

➤ Crystal

Crystal methodology includes three concepts:

1. **Chartering:** Involves creating the development team, performing feasibility analysis, and developing plans.
2. **Cyclic Delivery:** Includes updating the release plan and delivering an integrated product to users.
3. **Wrap-Up:** Performs deployment and post-deployment tasks according to the user environment.

➤ Dynamic Software Development Method (DSDM)

- DSDM is a rapid application development strategy that emphasizes user involvement and team decision-making authority. Key techniques include Time Boxing, MoSCoW Rules, and Prototyping. It consists of seven stages: Pre-project, Feasibility Study, Business Study, Functional Model Iteration, Design and Build Iteration, Implementation, and Post-project.

➤ Feature-Driven Development (FDD)

- FDD focuses on "Designing and Building" features, describing small steps of work to be achieved per function.

➤ Lean Software Development

- Lean methodology follows the principle of "just in time production," aiming to increase software development speed and reduce costs. Its phases include Eliminating Waste, Amplifying Learning, Deferring Commitment, Early Delivery, Empowering the Team, Building Integrity, and Optimizing the Whole.

• When to Use the Agile Model

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When the customer is ready to have regular meetings with the software team.
- When the project size is small.

- Advantages of Agile Method
 - Frequent delivery of working software.
 - Face-to-face communication with clients.
 - Efficient design that fulfills business requirements.
 - Acceptance of changes at any time.
 - Reduced total development time.

The Agile process model's iterative and incremental approach makes it highly suitable for developing an Attendance Management System. By breaking down the project into smaller, manageable parts and continuously integrating user feedback, the AMS can be efficiently developed to meet the needs of the educational institution while minimizing risks and ensuring high-quality delivery.

6. Feasibility Analysis

6.1 Technical Feasibility

- **Programming Language and Framework:** Python is a widely used, well-supported language with extensive libraries for web development (like QT5). SQLite is a lightweight and embedded database management system suitable for this project's scale. These technologies are technically feasible for developing the AMS.
- **Complexity:** The system involves user management, authentication, database interactions, and interface development. While the functionalities are defined, a thorough analysis of user roles and data management needs is crucial to assess the overall technical complexity.
- **Hardware Requirements:** The provided hardware requirements (Intel Core i3, 4GB RAM) are moderate and readily available on most computers. This makes the system accessible to a wide range of users.

6.2 Time Schedule Feasibility

- **Development Effort:** The system involves several interfaces with user interactions, database management, and security considerations. Depending on the complexity of functionalities and the experience of the developer(s), development time could range from several weeks to a few months.
- **Project Scope:** A well-defined project scope outlining features and functionalities is essential for creating a realistic timeline. Prioritizing core functionalities initially and adding advanced features later through iterative development can help manage time constraints.

6.3 Operational Feasibility

- **User Training:** The system requires user training for department heads, faculty, and students to understand their roles and effectively utilize the AMS. User guides and training sessions can address this need.
- **System Administration:** The system may require ongoing administration tasks such as user management, data backups, and potential software updates. Defining clear administrative procedures will be crucial.
- **Integration with Existing Systems:** If the AMS needs to integrate with existing student information systems or other departmental software, technical feasibility and compatibility need to be assessed.

6.4 Implementation Feasibility

- **Technical Skills:** Development requires skills in Python programming, QT5 framework, SQLite database management, and potentially web development concepts. Having developers with these skills or allocating time for skill development is necessary.
- **Deployment Environment:** The system needs a deployment environment (server or personal computer) with Python, QT5, and SQLite installed. Additionally, security considerations for user access and data storage need to be addressed.
- **Testing and Maintenance:** Thorough testing of functionalities, security, and performance is crucial before deployment. Ongoing maintenance needs to be factored in for bug fixes, updates, and potential future enhancements.

6.5 Economic Feasibility

- **Development Costs:** Development costs depend on factors like developer salaries, tools used, and potential infrastructure needs. Utilizing open-source technologies like Python, QT5, and SQLite helps reduce costs significantly.
- **Deployment and Maintenance Costs:** Costs associated with deployment environment (server rental or dedicated computer) and ongoing maintenance need to be considered. The economic feasibility depends on the budget allocated for development, deployment, and maintenance.

7. Project Plan

This project plan outlines the development process for the Attendance Management System (AMS) based on the provided information and a timeframe from January 29th, 2024, to April 20th, 2024 (estimated 12 weeks).

7.1 Project Goal: Develop a web-based AMS using Python, QT5, and SQLite to manage student attendance for a specific department within an educational institution.

7.2 Target Users: Faculty and Students

7.3 Development Methodology: Iterative and Incremental Development with an Incremental Prototype Model.

7.4 Project Deliverables:

- Functional AMS with user interfaces for Admin Faculty, and Students
- User Documentation
- Source Code

7.5 Project Management Tools:

Developing the AMS requires a set of tools to manage tasks, collaborate effectively, and ensure project success. Here are some recommended options:

1.GitHub: Version Control and Collaboration

GitHub is a version control system that enables developers to manage code changes, collaborate effectively, and track project history. By hosting the project repository on GitHub, developers can push their code updates, create branches for different features, and leverage features like issue tracking for bug reporting and feature requests. This promotes efficient collaboration and version control for the codebase.

2.Google Workspace: Collaborative Documentation

Google Workspace offers a suite of tools like Docs, Sheets, and Drive that are perfect for project documentation and collaboration. Create and share documents outlining user stories, system requirements, and user manuals. Utilize Sheets for data management or task tracking. With Drive For centralized storage and collaborative editing, the team can work on documents simultaneously, ensuring everyone has access to the latest information.

3.Zoom: Virtual Meetings and Screen Sharing

Zoom is a valuable tool for conducting virtual meetings, especially for remote teams or geographically dispersed stakeholders. Schedule meetings for project updates, discussions, or addressing issues. Utilize video conferencing, screen sharing, and meeting recording features. This allows for effective communication and collaboration regardless of location.

7.6 Project Phases and Activities:

Phase 1: Requirements Gathering and Analysis (Week 1-2)

- **Activities:**
 - Conduct stakeholder meetings with the department head, faculty, and students to gather detailed requirements.
 - Define user roles, functionalities, and data needs.
 - Document user stories and system specifications (SRS).
 - Finalize project scope and prioritize functionalities for initial development.

- **Deliverables:**

- User Stories Document
- System Requirements Document (SRS)
- Project Scope Definition

Phase 2: Design and Development (Week 3-8)

- **Activities:**

- Design the system architecture, database schema, and user interfaces using diagrams and wireframes.
- Develop core functionalities for user login, user management, course management, and attendance recording.
- Implement the system using Python, QT5, and SQLite according to the design.
- Conduct unit testing of individual modules.

- **Deliverables:**

- System Architecture Diagrams
- Database Schema
- User Interface Mockups
- Functional Prototype (initial version)
- Unit Test Reports

Phase 3: Prototype Testing and Refinement (Week 9-10)

- **Activities:**

- Conduct user testing with the department head, faculty, and students to gather feedback on the initial prototype.
- Analyze user feedback and identify areas for improvement.
- Refine functionalities and user interfaces based on user feedback.
- Conduct integration testing to ensure modules work together seamlessly.

- **Deliverables:**

- User Testing Reports
- Refined Prototype with Enhanced Functionalities
- Integration Test Reports

Phase 4: Deployment and Training (Week 11)

- **Activities:**

- Deploy the AMS on a designated server or computer.
- Configure security settings for user access and data storage.
- Develop user documentation for each user role (Department Head, Faculty, Students).
- Conduct user training sessions for all users.

- **Deliverables:**

- Deployed AMS
- User Documentation (Department Head, Faculty, Student)

Phase 5: System Testing and Maintenance (Week 12)

- **Activities:**
 - Conduct system testing to ensure overall functionality, performance, and security.
 - Address any identified bugs or issues.
 - Conduct user acceptance testing (UAT) with stakeholders for final approval.
 - Create a maintenance plan for bug fixes, updates, and future enhancements.
- **Deliverables:**
 - System Testing Reports
 - User Acceptance Testing (UAT) Report
 - System Maintenance Plan

8. System Design

8.1 Entity Relationship Diagram

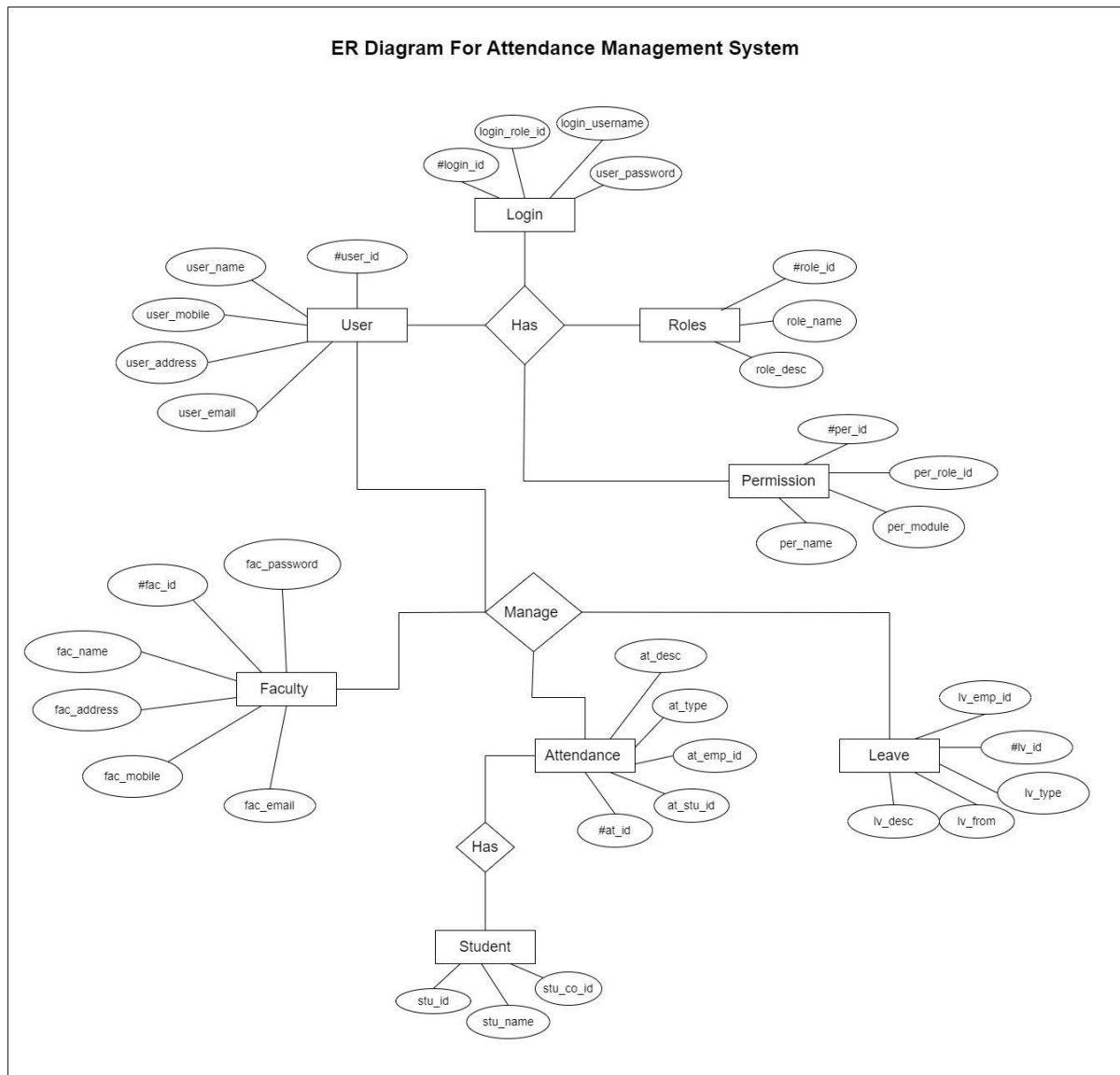


Figure 1 ER Diagram

- An Entity-Relationship (ER) diagram is a visual representation of the entities (things of interest) in a system or database, their attributes (properties or characteristics), and the relationships between them. It helps in understanding the structure of a database by illustrating how different entities relate to each other, aiding in database design and communication between stakeholders.

8.2 Use Case Diagram

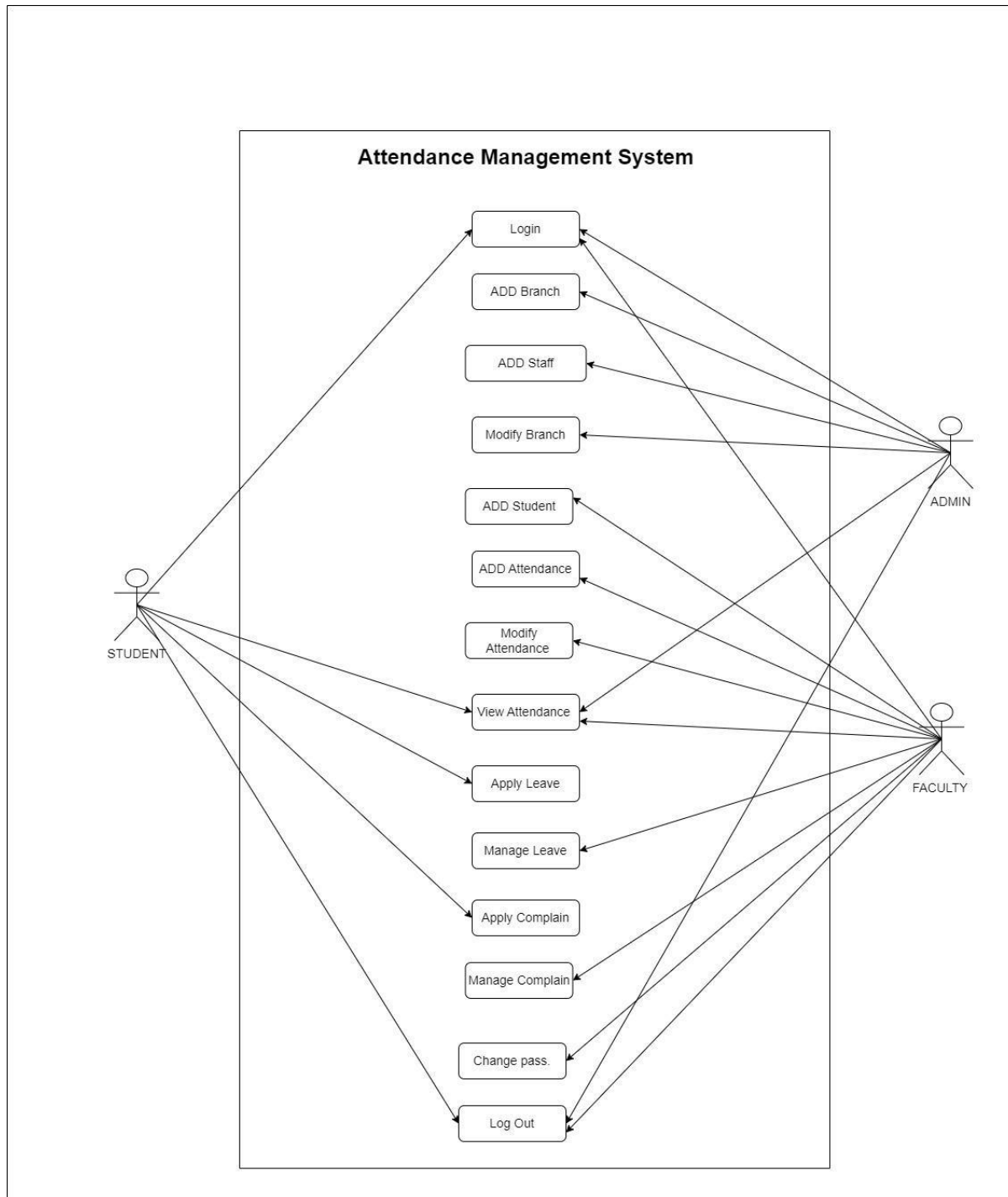


Figure 2 Use Case Diagram

- A Use Case Diagram is a visual representation of the interactions between actors (users or external systems) and a system, showcasing the different ways the system can be used to achieve specific goals. It provides a high-level view of system functionality, illustrating the various use cases (actions or tasks) that actors can perform and how they relate to one another.

8.3 Activity Diagram

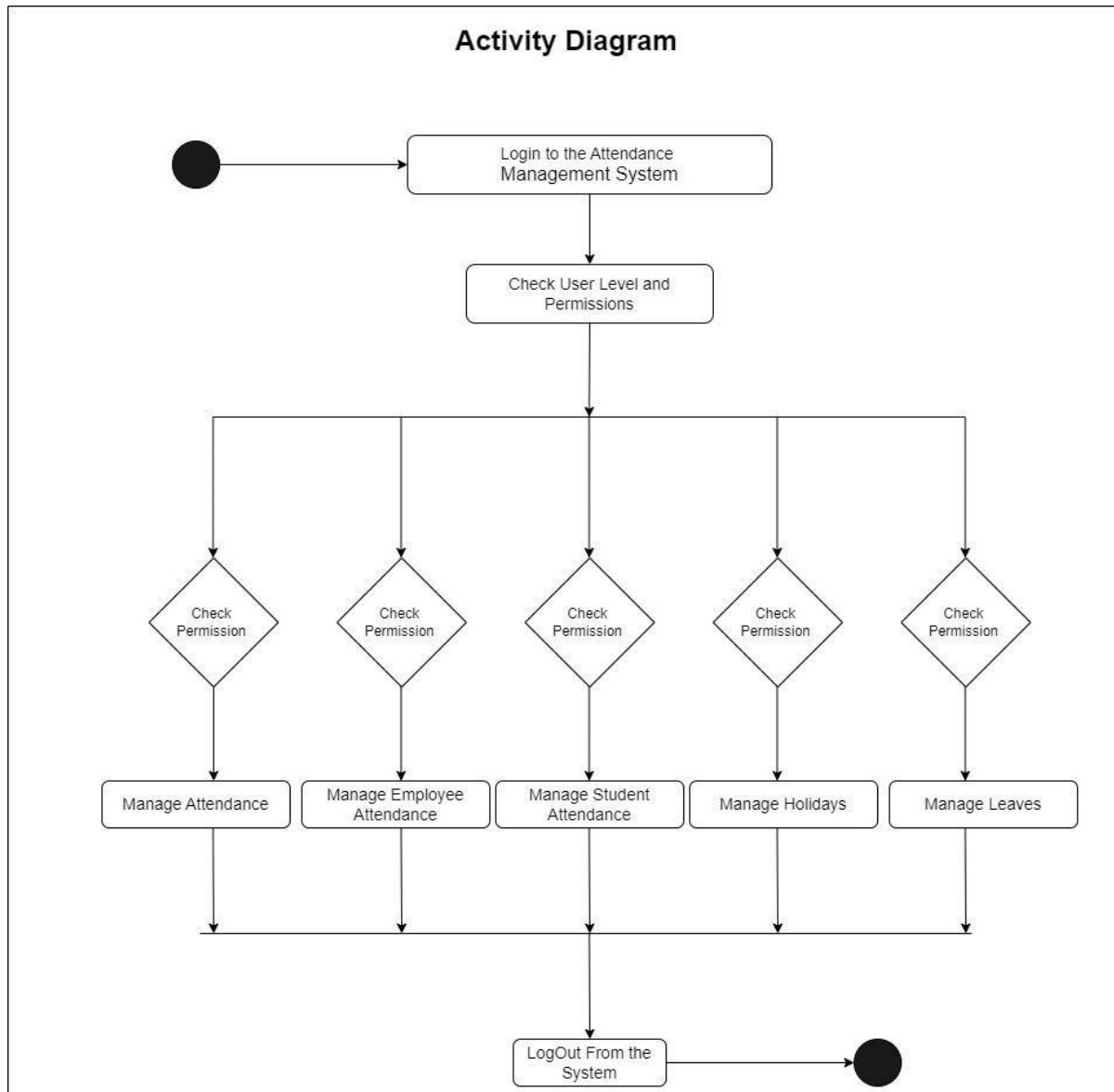


Figure 3 Activity Diagram

- Activity Diagram is a visual representation of the flow of activities within a system or process, showing the sequence of actions or steps that occur and the decisions made along the way. It is commonly used to model business processes, software workflows, or any complex system where understanding the flow of activities is important.

8.3.1 Admin- Activity Diagram

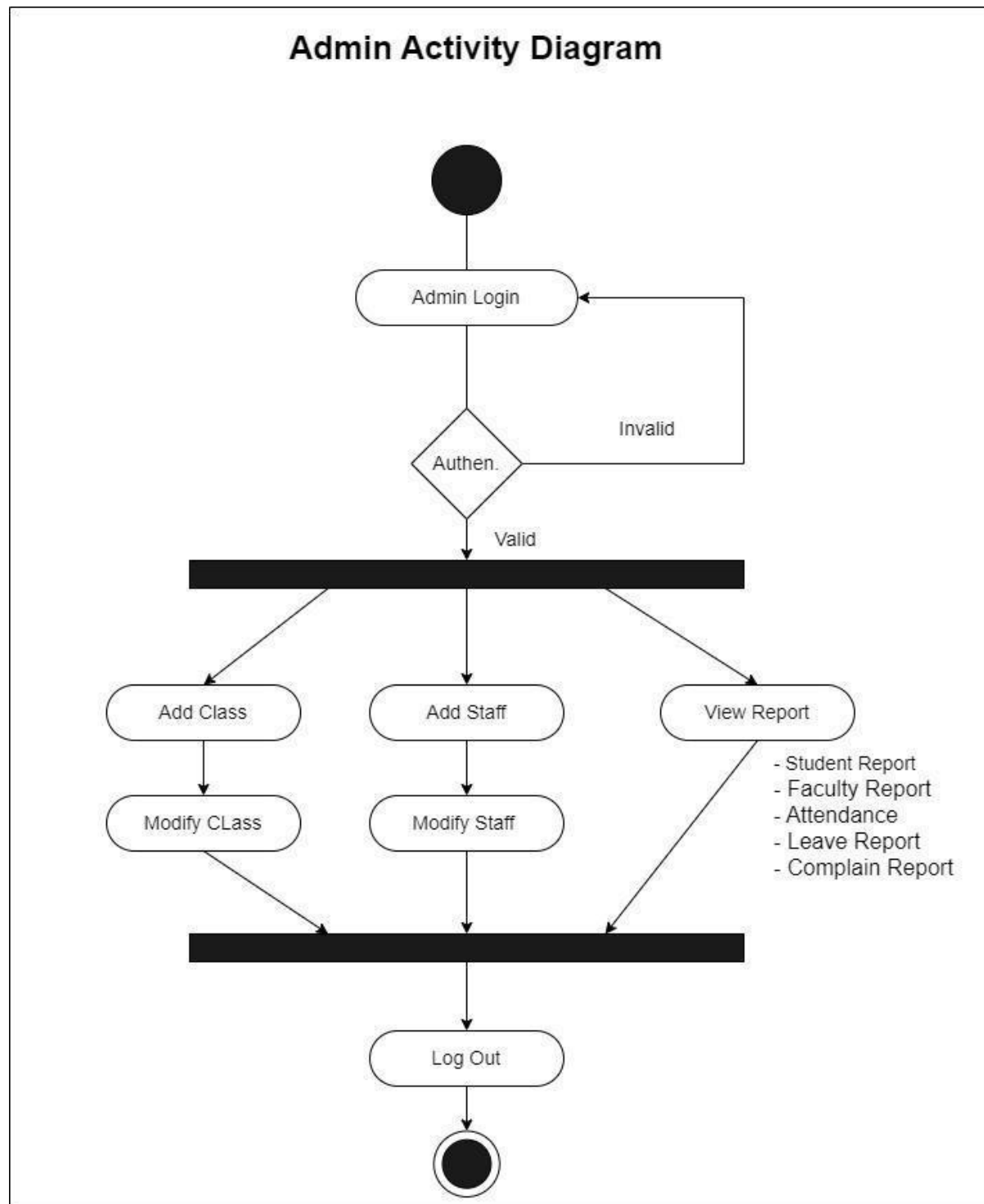


Figure 4 Admin Activity Diagram

8.3.2 Student Activity Diagram

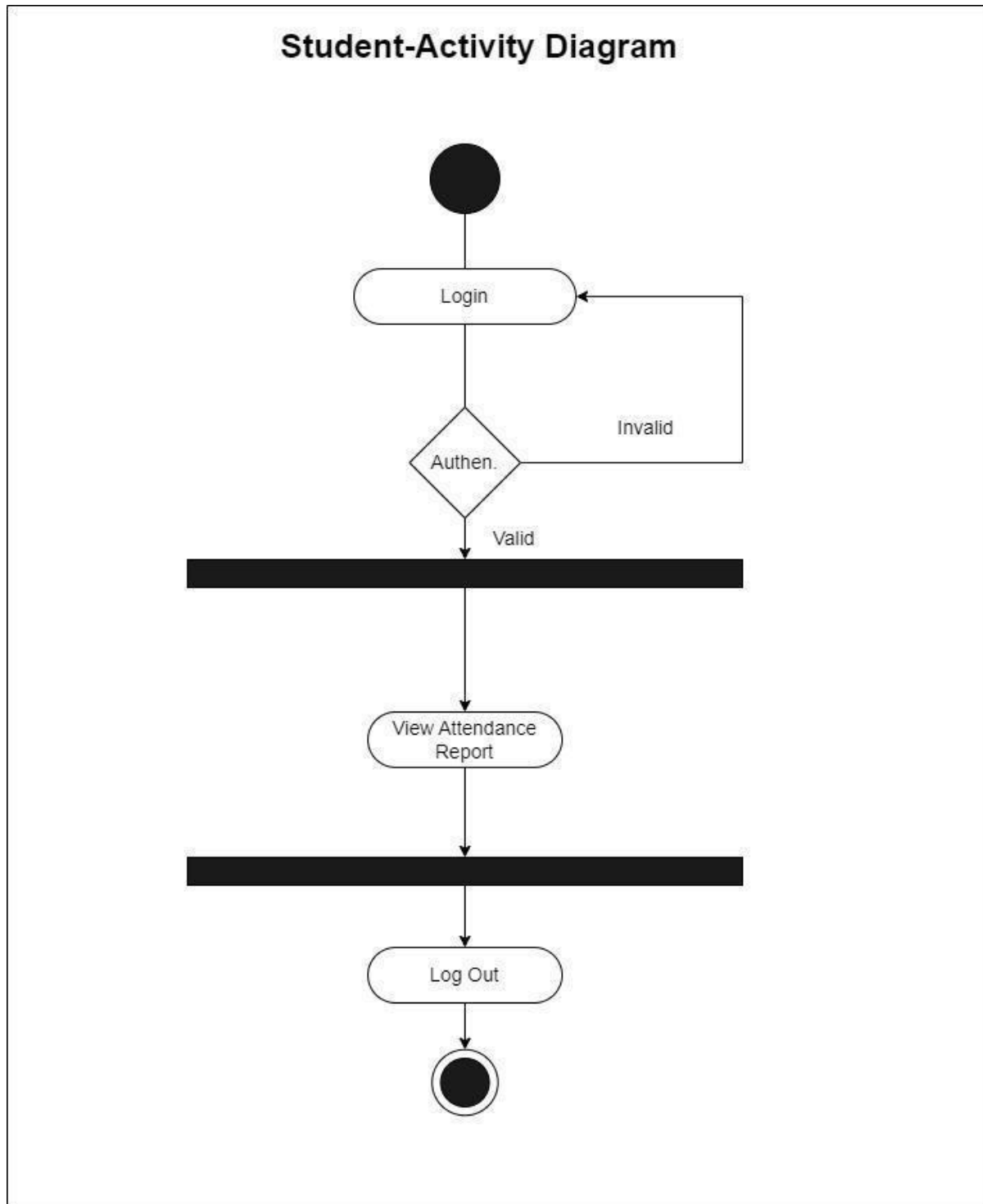


Figure 5 Student Activity Diagram

8.3.3 Faculty Activity Diagram

Faculty-Activity Diagram

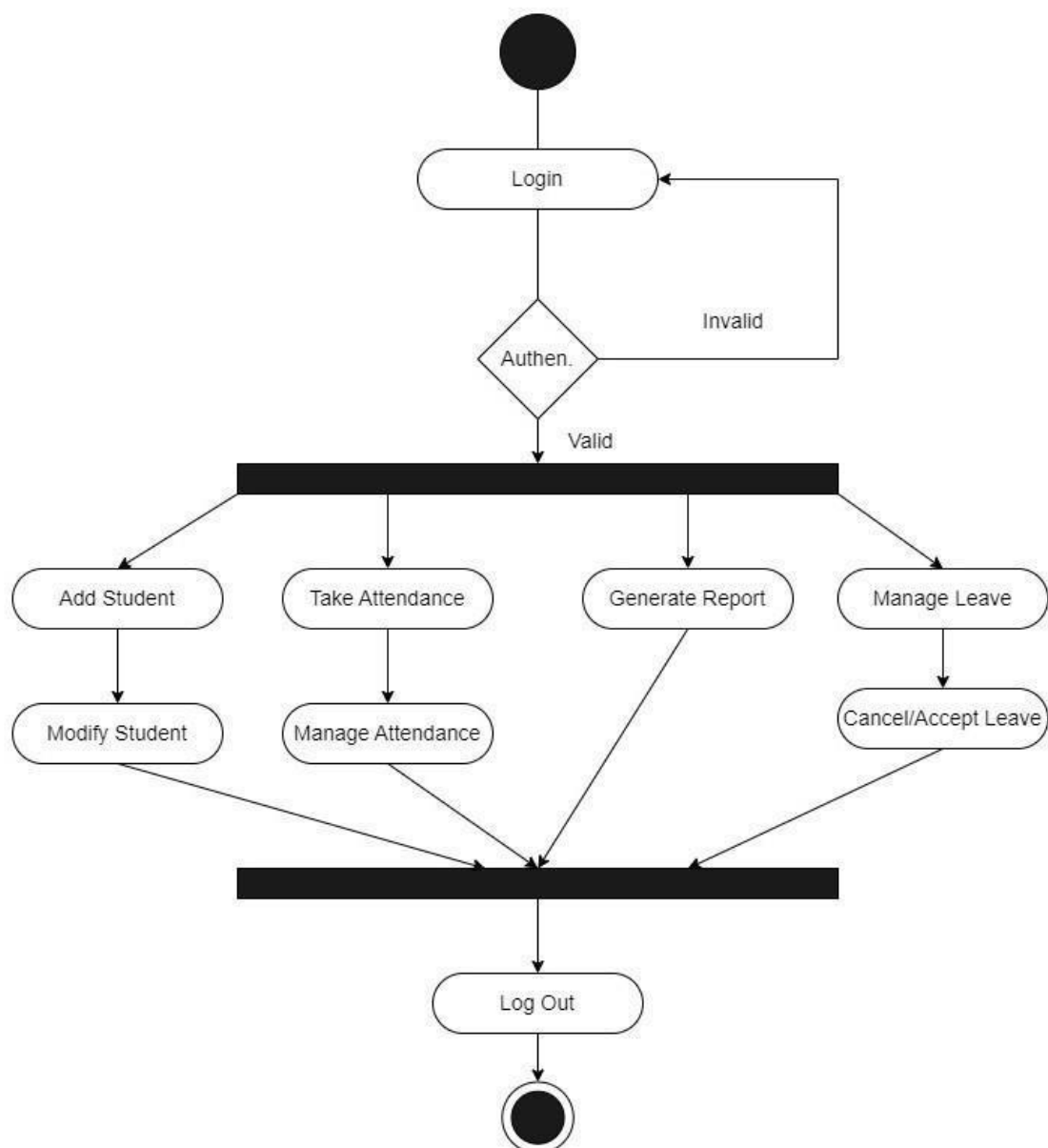


Figure 6 Faculty-Activity Diagram

8.4 Component Diagram

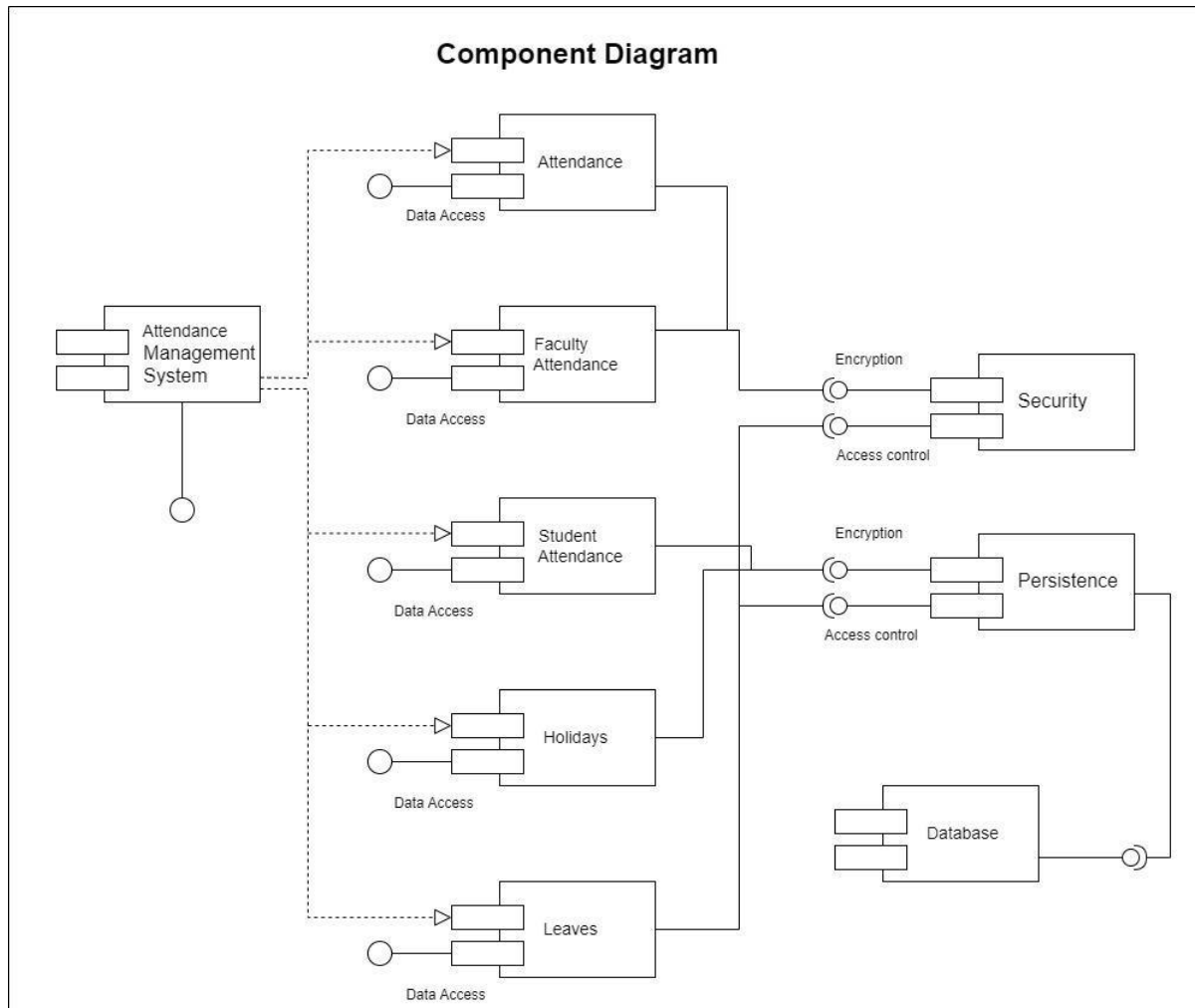


Figure 7 Component Diagram

- A component diagram is a visual representation that illustrates the structural organization of software components within a system or application. It depicts the high-level architecture of the system, showing how components are interconnected and interact with each other to achieve specific functionalities. Components represent modular units of software that encapsulate reusable and cohesive functionality, making it easier to manage and maintain complex systems.

8.5 State chart Diagram

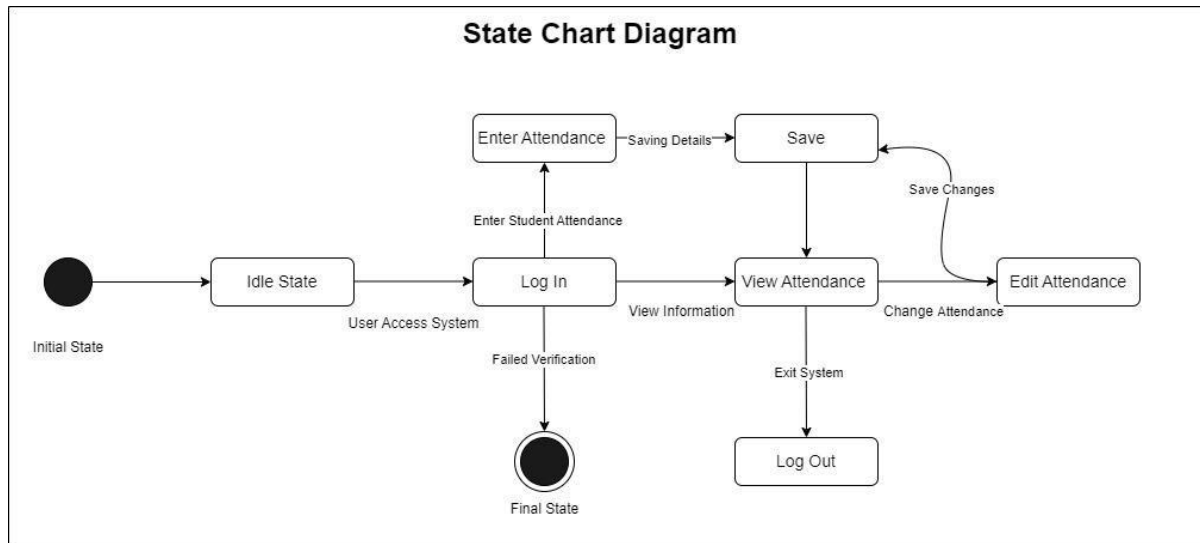


Figure 8 Statechart Diagram

- A state chart diagram, also known as a state machine diagram, is a visual representation that depicts the various states that an object or system can transition through in response to events. It illustrates the behavior of a system by showing the states, transitions between states, and the events that trigger those transitions.

8.6 Class Diagram

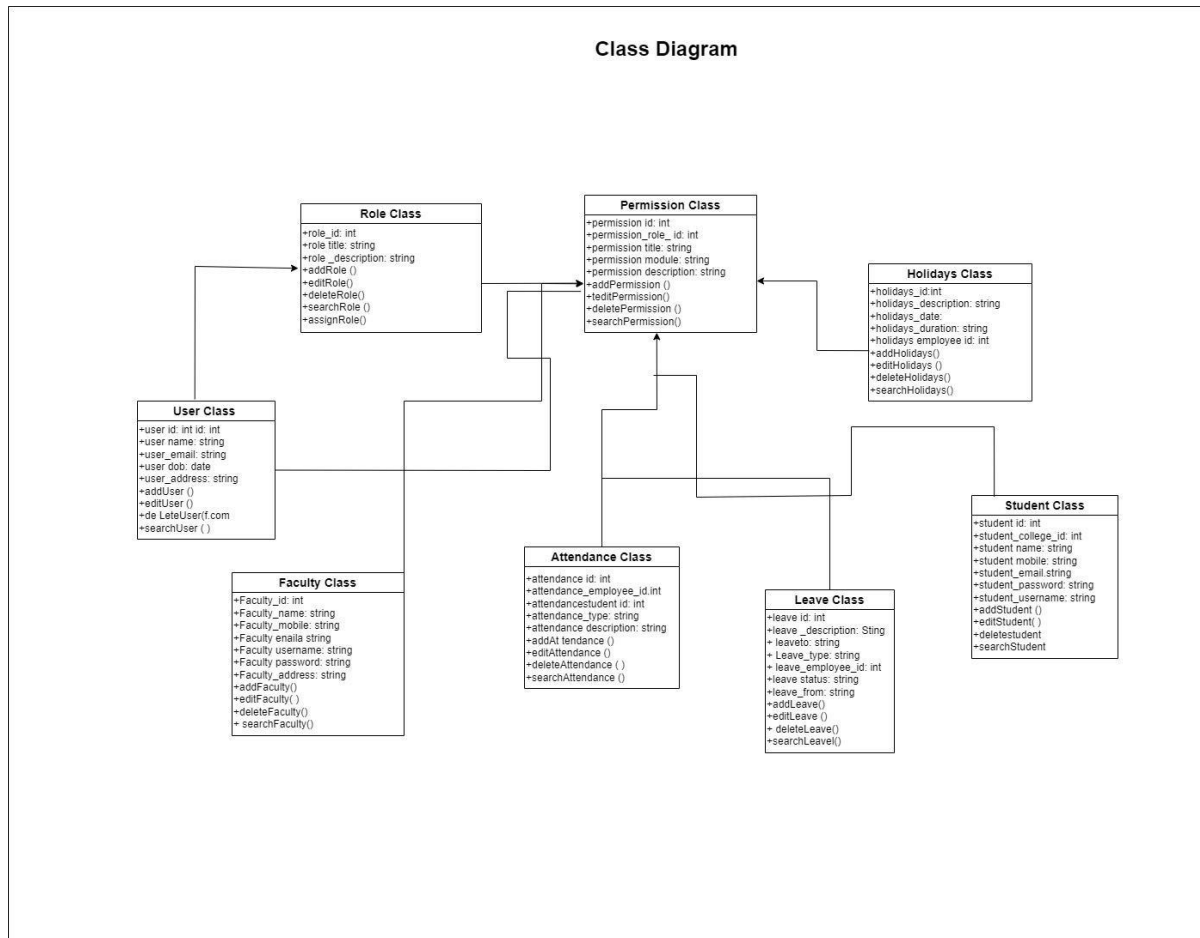


Figure 9 Class Diagram

- A class diagram is a type of static structure diagram in UML (Unified Modeling Language) that represents the structure of a system by showing the classes, their attributes, methods, and relationships. It provides a visual representation of the classes and their interactions in the system.

8.7 Sequence Diagram

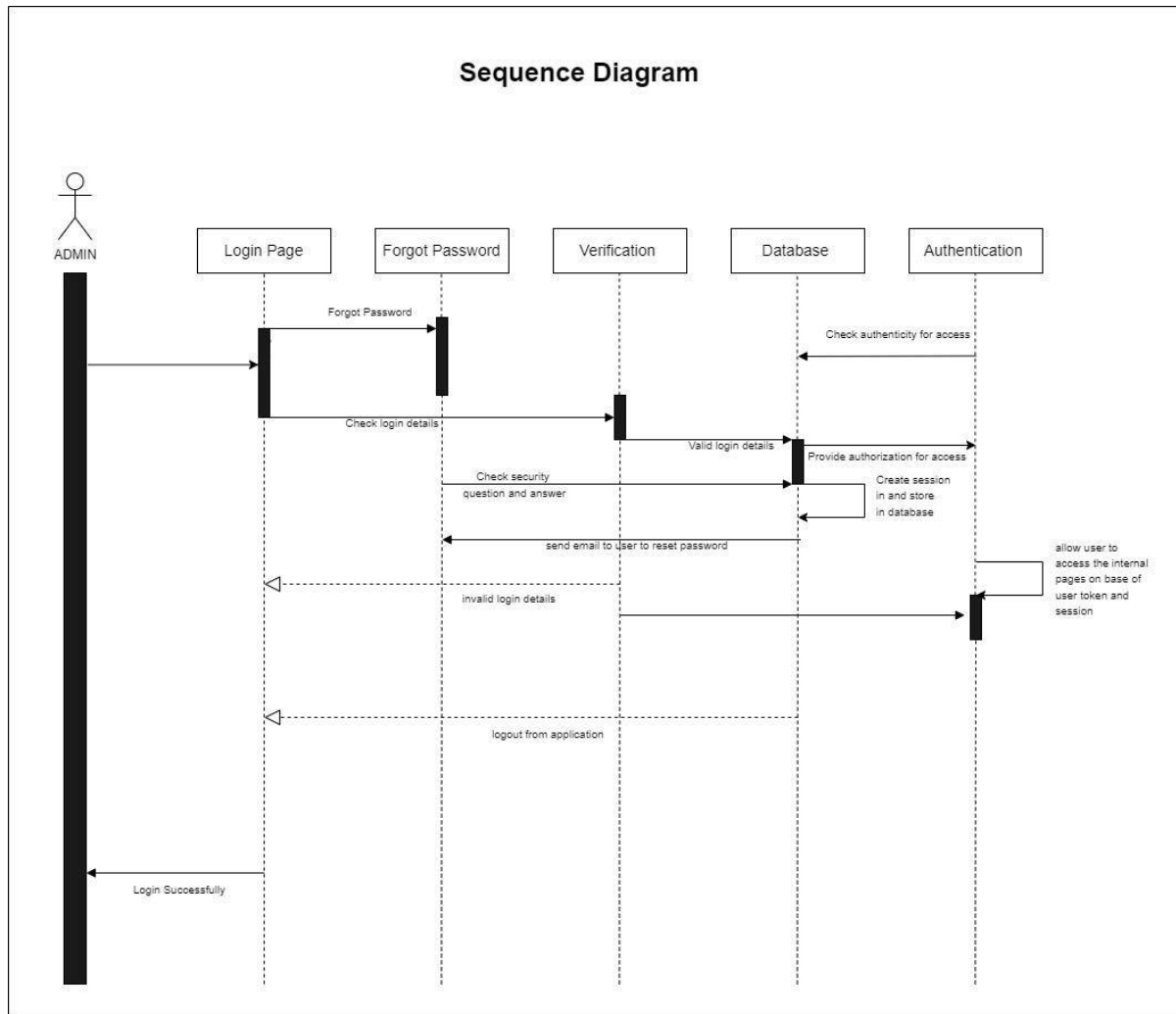


Figure 10 Sequence Diagram

- A sequence diagram is a type of interaction diagram that visualizes the flow of messages and interactions between objects or components in a system over time. It depicts the chronological order of events, showing how objects collaborate to accomplish a specific task or scenario. Sequence diagrams are particularly useful for understanding the dynamic behavior of systems, including the sequence of method calls, the timing of interactions, and the flow of control between different components.

8.8 Data Flow Diagram

8.8.1 DFD-0 level / Context Diagram

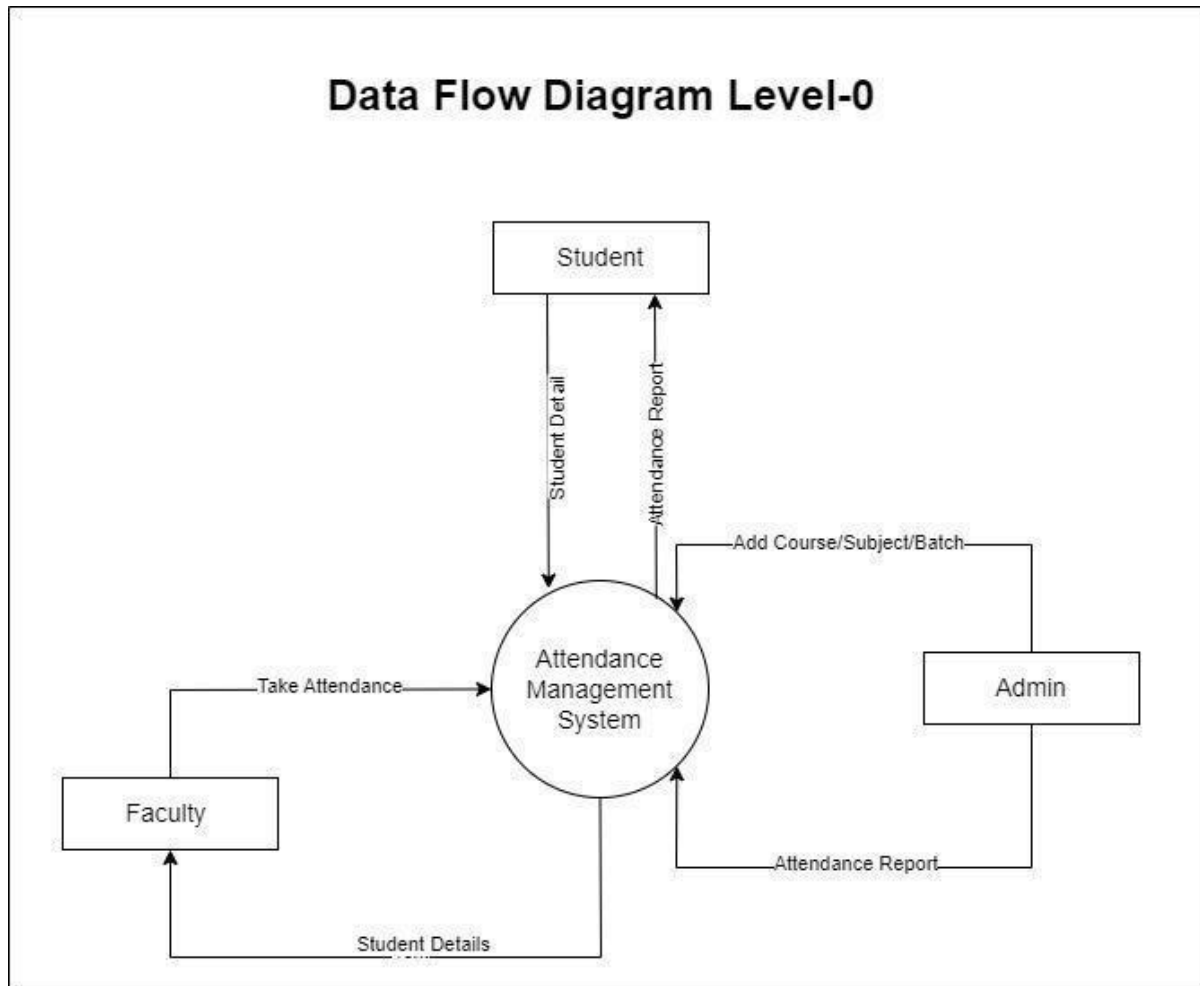


Figure 11 DFD Diagram Level-0

- A data flow diagram (DFD) is a visual representation of the flow of data within a system, illustrating how information is input, processed, stored, and output. It consists of processes, data stores, data flows, and external entities, which are connected to each other to depict the movement of data throughout the system. DFDs provide a high-level overview of system functionality, focusing on the interactions between different components rather than the internal workings of individual processes. They are valuable tools for understanding system requirements, identifying data sources and destinations, and designing efficient data processing systems.

8.8.2 DFD-1 level

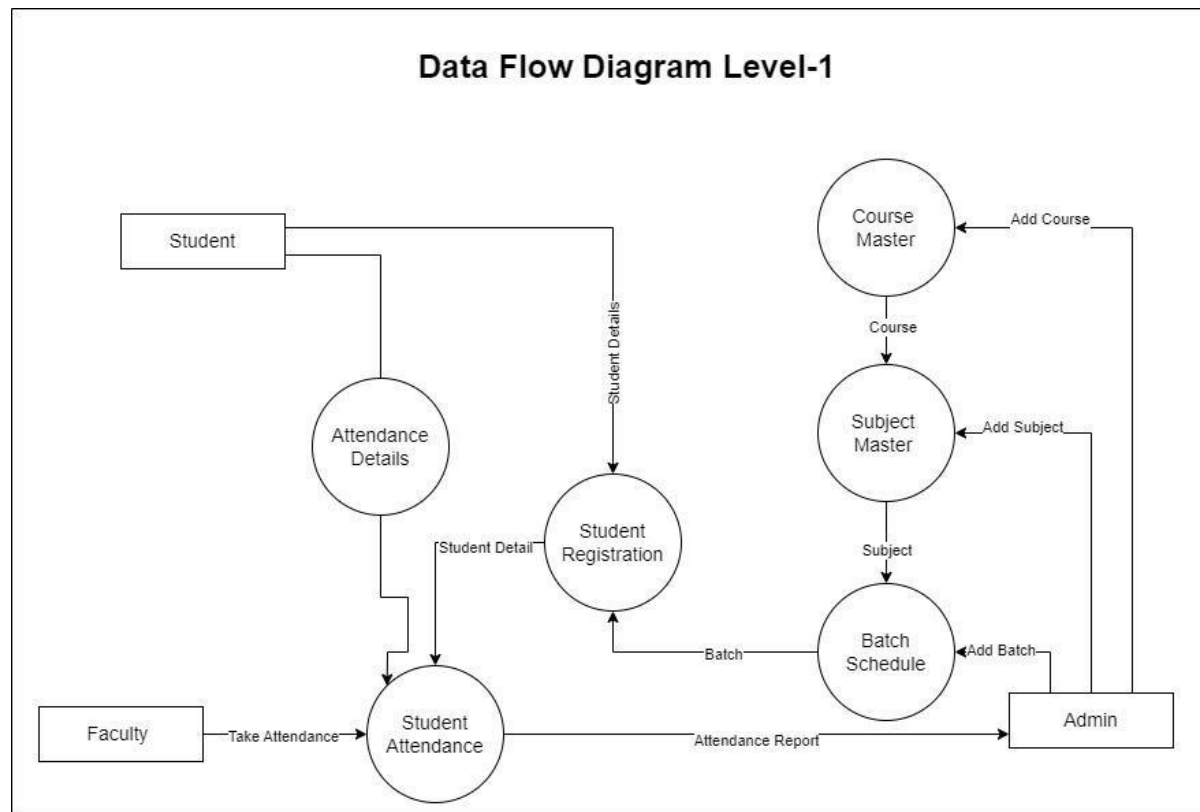


Figure 12 DFD Diagram Level-1

8.8.3 DFD-2 level

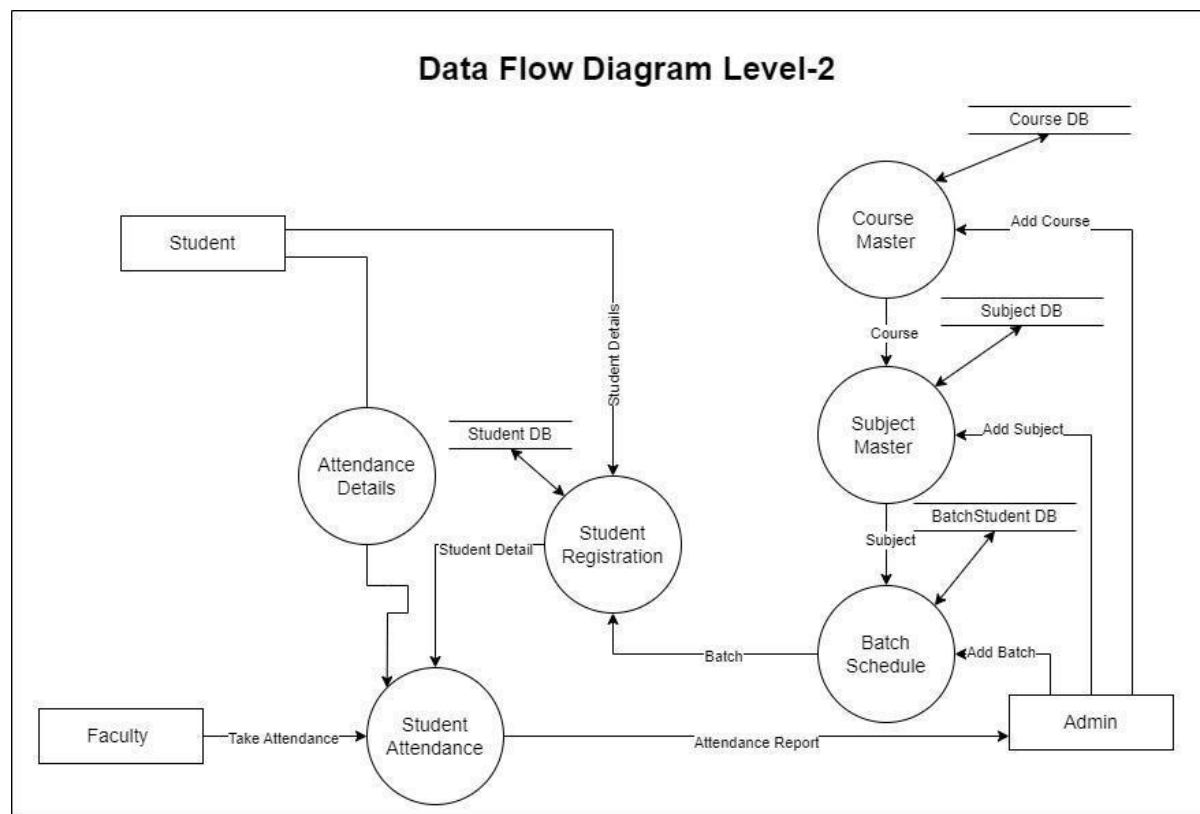


Figure 13 DFD Diagram Level-2

8.8.3.1 DFD-2 level -Admin

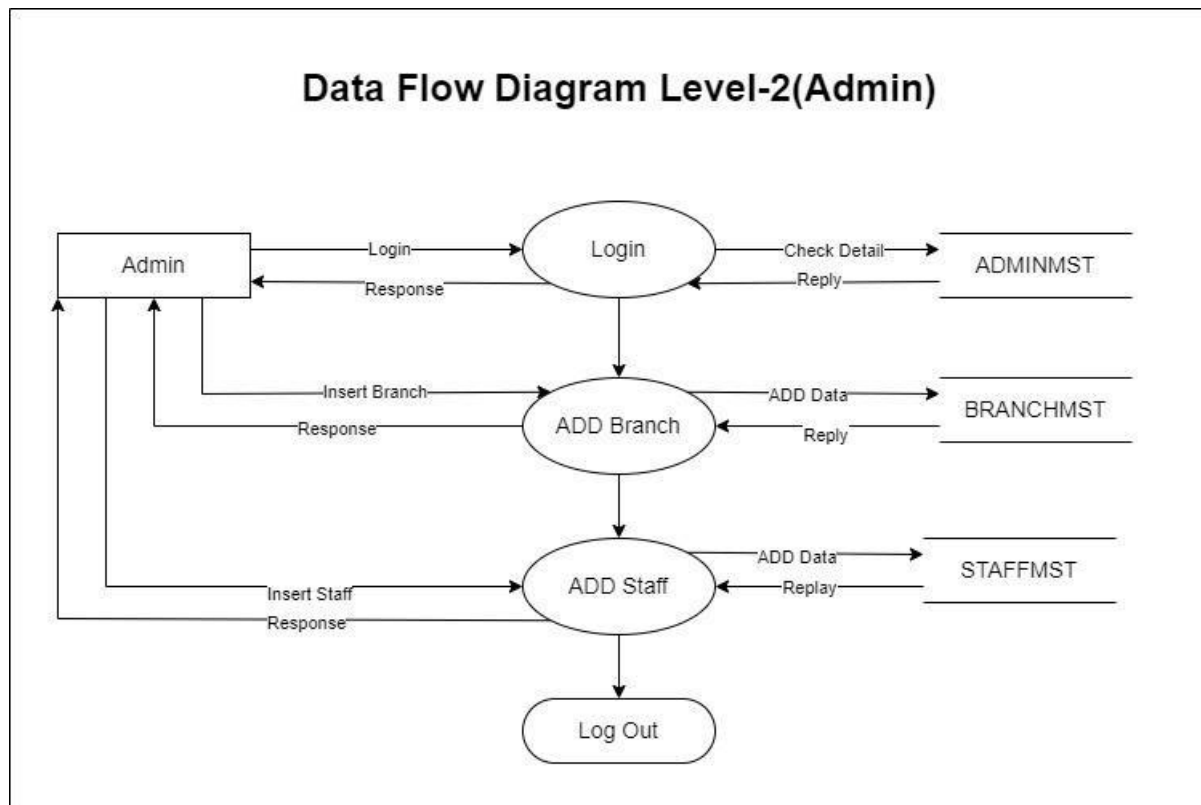


Figure 14 DFD Level-2 Admin

8.8.3.2 DFD-2 level -Student

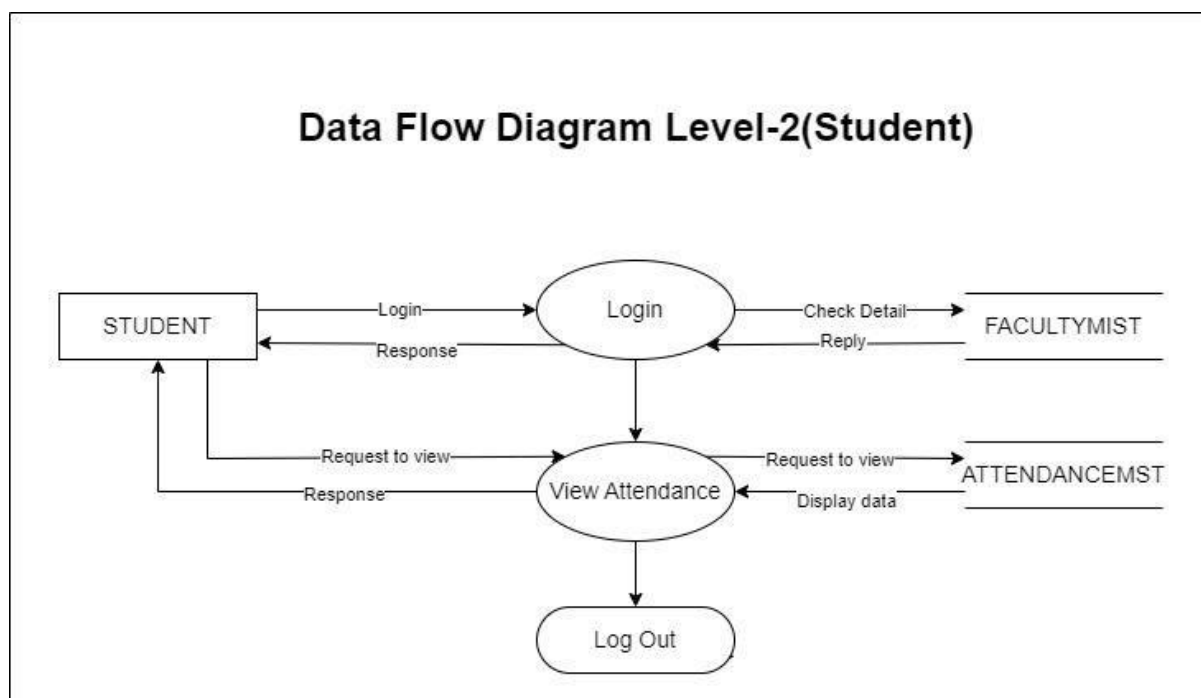


Figure 15 DFD Level-2 Student

8.8.3.3 DFD-2 level -Faculty

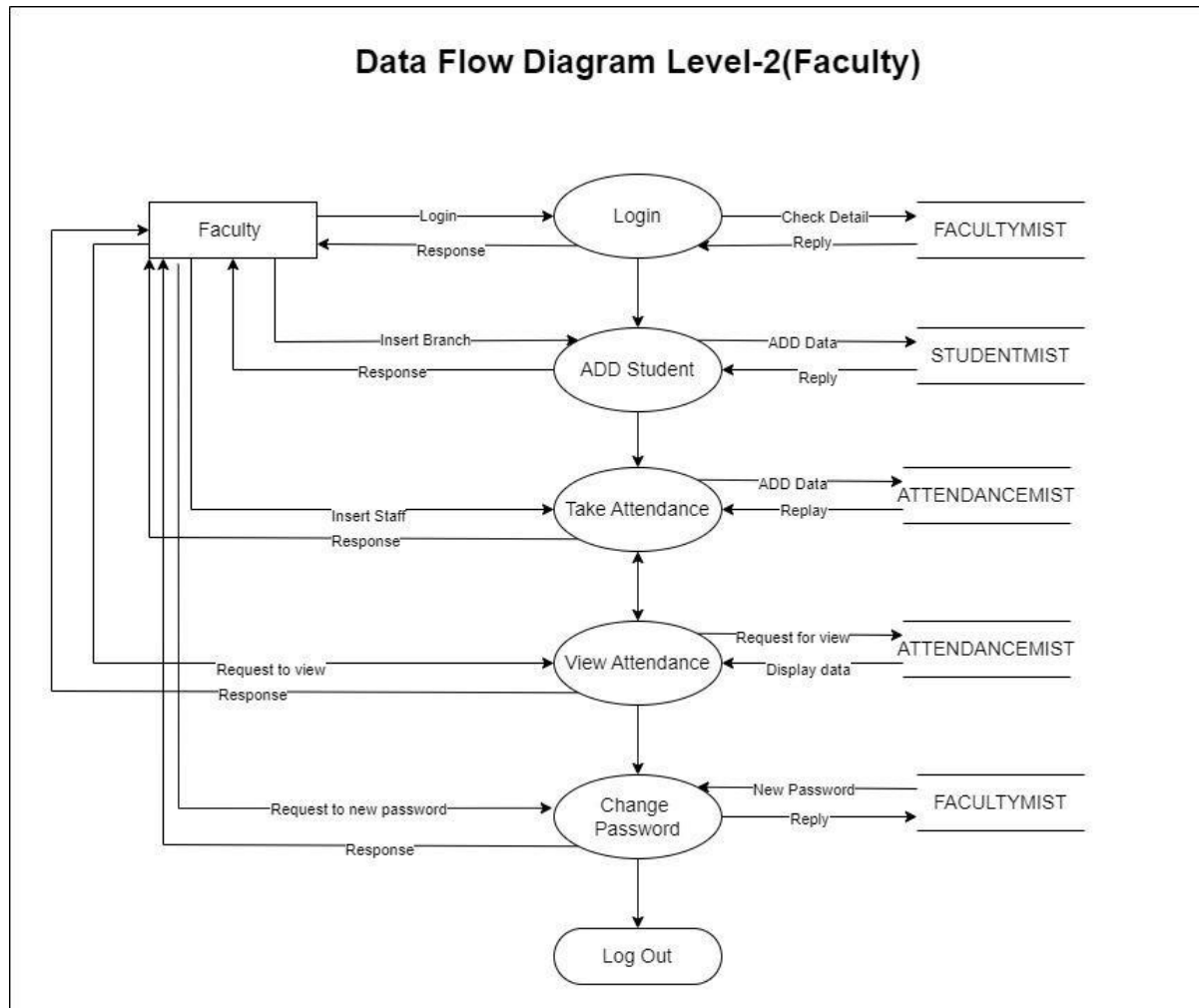


Figure 16 DFD Level-2 Faculty

8.9 Data Dictionary:

Course Table:

Table 1 Course Table

S.No	Field	Type	Size	Description
1	course_id	VARCHAR	20	Unique Course ID
2	course_name	VARCHAR	30	Name of the Course

Teacher Table:

Table 2 Teacher Table

S.No	Field	Type	Size	Description
1	faculty_id	VARCHAR	20	Unique faculty ID
2	ffname	VARCHAR	30	Teacher's first name
3	fllname	VARCHAR	30	Teacher's last name
4	password	VARCHAR	20	Password for the teacher
5	course_id	VARCHAR	10	Course ID taught by the faculty

Student Table:

Table 3 Student Table

S.No	Field	Type	Size	Description
1	roll_no	VARCHAR	30	Unique student ID
2	Sfname	VARCHAR	30	Student's first name
3	Sllname	VARCHAR	30	Student's last name
4	password	VARCHAR	30	Password for the student

Attendance Table:*Table 4 Attendance Table*

S.No	Field	Type	Size	Description
1	roll_no	VARCHAR	30	Student ID
2	date	DATE	-	Date of attendance
3	status	BOOLEAN	-	Attendance status (P: Present, A: Absent)
4	course_id	VARCHAR	30	Course ID
5	faculty_id	VARCHAR	30	Faculty ID

Studies Table:*Table 5 Studies Table*

S.No	Field	Type	Size	Description
1	course_id	VARCHAR	30	Course ID
2	roll_no	VARCHAR	30	Student ID
3	faculty_id	VARCHAR	30	Faculty ID

9. Implementation Details

9.1 Flowchart of Implementation:

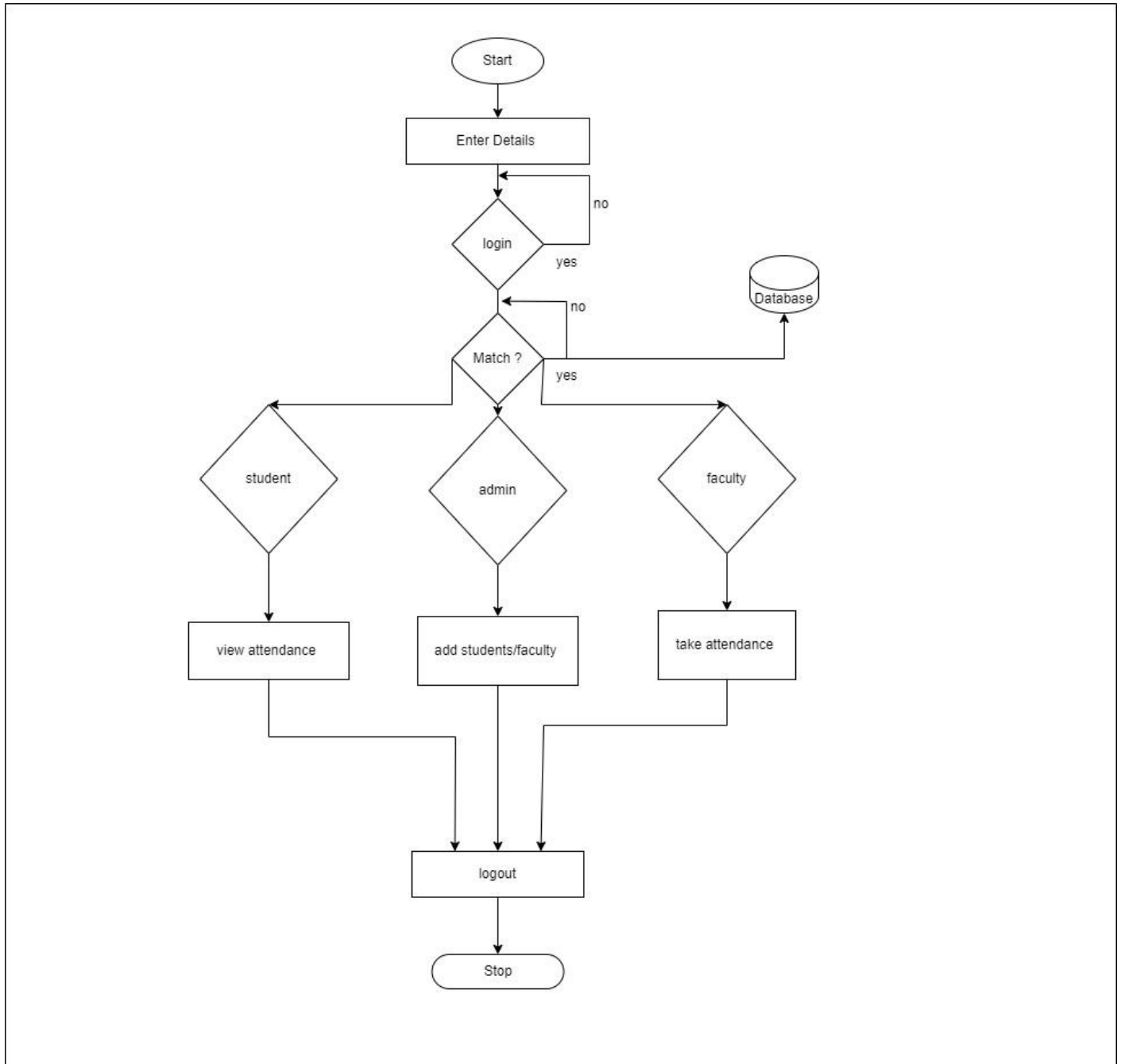


Figure 17 Flowchart of Implementation

9.2 Actual Program Code:

9.2.1 Main.py

```
from PyQt5.QtWidgets import *
from PyQt5 import QtCore
import student
import teacher
from mainmenugui import Ui_MainWindow
import info
import database
import head
import sqlite3
import sys

class MainWindow(QMainWindow, Ui_MainWindow):
    switch_window = QtCore.pyqtSignal()

    def __init__(self, *args, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)
        self.setupUi(self)
        self.defaultid="admin"
        self.defaultpass="admin"
        self.info.clicked.connect(self.openInfoPanel)
        self.head.clicked.connect(self.openHeadPanel)
        self.student.clicked.connect(self.openStudentPanel)
        self.teacher.clicked.connect(self.openTeacherPanel)

    def openInfoPanel(self):
        self.infoWindow=info.MainWindow()
        self.id.clear()
        self.password.clear()

    def openHeadPanel(self):
        idinput=self.id.toPlainText()
        passinput=self.password.toPlainText()
```

```
if idinput==self.defaultid and passinput==self.defaultpass:

    self.id.clear()

    self.password.clear()

    self.headWindow=head.MainWindow()

    self.errorDisplay.setText("")

elif idinput!=self.defaultid:

    self.errorDisplay.setText("No Such ID Found! Please Check Your ID Again!")

else :

    self.errorDisplay.setText("Incorrect Password! Please Check Your Password Again")

def openStudentPanel(self):

    idinput = self.id.toPlainText()

    passinput = self.password.toPlainText()

    conn = sqlite3.connect('attendance.db')

    conn.execute('PRAGMA foreign_keys=on;')

    cursor = conn.execute("SELECT * FROM STUDENT WHERE roll_no=?", (idinput,))

    result=cursor.fetchall()

    conn.close()

    if len(result)==0:

        self.errorDisplay.setText("No Such ID Found! Please Check Your ID Again!")

    elif result[0][3]!=passinput:

        self.errorDisplay.setText("Incorrect Password! Please Check Your Password Again")

    else:

        self.errorDisplay.setText("")

        self.id.clear()

        self.password.clear()

        self.studentWindow=student.MainWindow(idinput)

def openTeacherPanel(self):

    idinput = self.id.toPlainText()

    passinput = self.password.toPlainText()

    conn = sqlite3.connect('attendance.db')

    conn.execute('PRAGMA foreign_keys=on;')

    cursor = conn.execute("SELECT * FROM TEACHER WHERE faculty_id=?", (idinput,))
```

```

    result = cursor.fetchall()

    conn.close()

    if len(result) == 0:

        self.errorDisplay.setText("No Such ID Found! Please Check Your ID Again!")

    elif result[0][3] != passinput:

        self.errorDisplay.setText("Incorrect Password! Please Check Your Password Again")

    else:

        self.errorDisplay.setText("")

        self.id.clear()

        self.password.clear()

        self.teacherWindow = teacher.MainWindow(idinput)

database.Tables()

app = QApplication(sys.argv)

window = MainWindow()

window.show()

sys.exit(app.exec_())

```

9.2.2 Database.py

```

import sqlite3

class Tables:

    def __init__(self):

        conn = sqlite3.connect('attendance.db')

        conn.execute('PRAGMA foreign_keys=on;')

        conn.execute("""CREATE TABLE IF NOT EXISTS COURSE

            (course_id    varchar(30)  PRIMARY KEY,

             course_name  varchar(30)  NOT NULL);""")

        conn.execute("""CREATE TABLE IF NOT EXISTS TEACHER

            (faculty_id varchar(30)  PRIMARY KEY,

             ffname     varchar(30)  NOT NULL,

             flname     varchar(30)  NOT NULL,

             password   varchar(30)  NOT NULL,

             course_id  varchar(30)  NOT NULL,

             FOREIGN KEY (course_id) REFERENCES course (course_id) ON DELETE CASCADE);""")

```

```

conn.execute("CREATE TABLE IF NOT EXISTS STUDENT
    (roll_no  varchar(30)  PRIMARY KEY,
     sfname   varchar(30)  NOT NULL,
     slname   varchar(30)  NOT NULL,
     password varchar(30)  NOT NULL);")

conn.execute("CREATE TABLE IF NOT EXISTS ATTENDANCE
    (roll_no  varchar(30)  NOT NULL,
     date     DATE        NOT NULL,
     status   BOOLEAN     NOT NULL,
     course_id varchar(30) NOT NULL,
     faculty_id varchar(30) NOT NULL,
     FOREIGN KEY (roll_no) REFERENCES STUDENT (roll_no) ON DELETE CASCADE,
     FOREIGN KEY (course_id) REFERENCES course (course_id) ON DELETE CASCADE);")

conn.execute("CREATE TABLE IF NOT EXISTS STUDIES
    (course_id varchar(30) NOT NULL,
     roll_no   varchar(30) NOT NULL,
     faculty_id varchar(30) NOT NULL,
     FOREIGN KEY (course_id) REFERENCES course (course_id) ON DELETE CASCADE,
     FOREIGN KEY (roll_no) REFERENCES STUDENT (roll_no) ON DELETE CASCADE,
     FOREIGN KEY (faculty_id) REFERENCES TEACHER (faculty_id) ON DELETE CASCADE)
;")

cursor = conn.execute("SELECT COUNT(*) FROM COURSE")
cnt = cursor.fetchall()
if cnt[0][0] == 0:
    cursor.execute(
        "INSERT INTO COURSE VALUES ('2CE1','DSA')")
    cursor.execute(
        "INSERT INTO COURSE VALUES ('2CE2','OOP')")
    cursor.execute(
        "INSERT INTO COURSE VALUES ('2CE3','DBMS')")
    cursor.execute(
        "INSERT INTO COURSE VALUES ('2CE4','OS')")

```



```
cursor.execute(
    "INSERT INTO COURSE VALUES ('2CE5','CP2')")
cursor.execute(
    "INSERT INTO COURSE VALUES ('2CE6','TOC')")
cursor.execute(
    "INSERT INTO COURSE VALUES ('2CE7','AI')")
cursor.execute(
    "INSERT INTO COURSE VALUES ('2CE8','CC')")
cursor.execute(
    "INSERT INTO COURSE VALUES ('2CE9','SVT')")
cursor.execute(
    "INSERT INTO COURSE VALUES ('2CE10','ITIM')")
cursor.execute(
    "INSERT INTO COURSE VALUES ('2CE11','ASB2')")
cursor.execute(
    "INSERT INTO COURSE VALUES ('2CE12','UED')")
cursor = conn.execute("SELECT COUNT(*) FROM TEACHER")
cnt = cursor.fetchall()
if cnt[0][0] == 0:
    cursor.execute(
        "INSERT INTO TEACHER VALUES ('1', 'Nishi', 'Patwa', '2001', '2CE1')")
    cursor.execute(
        "INSERT INTO TEACHER VALUES ('2', 'Om', 'Prakash', '2002', '2CE2')")
    cursor.execute(
        "INSERT INTO TEACHER VALUES ('3', 'Megha', 'Patel', '2003', '2CE3')")
    cursor.execute(
        "INSERT INTO TEACHER VALUES ('4', 'Paresh', 'Solanki', '2004', '2CE4')")
    cursor.execute(
        "INSERT INTO TEACHER VALUES ('5', 'Rajul', 'Suthar', '2005', '2CE5')")
    cursor.execute(
        "INSERT INTO TEACHER VALUES ('6', 'Chirag', 'Patel', '2006', '2CE6')")
    cursor.execute(
```

```

"INSERT INTO TEACHER VALUES ('7', 'Ravi', 'Raval', '2007', '2CE7')")

cursor.execute(

"INSERT INTO TEACHER VALUES ('8', 'Devang', 'Pandeya', '2008', '2CE8')")

cursor.execute(

"INSERT INTO TEACHER VALUES ('9', 'Manan', 'Thakkar', '2009', '2CE9')")

cursor.execute(

"INSERT INTO TEACHER VALUES ('10', 'Bhavesha', 'Suthar', '2010', '2CE10')")

conn.commit()

conn.close()

```

9.2.3 info.py

```

from PyQt5.QtWidgets import *
from PyQt5 import QtCore
from infogui import Ui_MainWindow
import sqlite3

class MainWindow(QMainWindow, Ui_MainWindow):
    switch_window = QtCore.pyqtSignal()

    def __init__(self, *args, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)
        self.setupUi(self)
        self.infoDisplay()
        self.run()

    def infoDisplay(self):
        conn = sqlite3.connect('attendance.db')
        cursor = conn.execute("SELECT course_id, course_name FROM COURSE")
        stringOut = "{} {}".format("Course ID".ljust(20), "Course Name".ljust(20))
        for row in cursor:
            s = "{} {}".format(row[0].ljust(20), row[1].ljust(20))
            stringOut = stringOut + '\n' + s
        self.course_details.setText(stringOut)
        conn.close()
        conn = sqlite3.connect('attendance.db')
        cursor = conn.execute("SELECT fname, lname, course_id FROM TEACHER")

```

```
stringOut = "{} {} {}".format("First Name".ljust(20), "Last Name".ljust(20), "Course ID".ljust(20))
```

```
for row in cursor:
```

```
    s = "{} {} {}".format(row[0].ljust(20), row[1].ljust(20), row[2].ljust(20))
```

```
    stringOut = stringOut + '\n' + s
```

```
self.teacher_details.setText(stringOut)
```

```
conn.close()
```

```
def run(self):
```

```
    self.show()
```

9.2.3 mainmenugui.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
```

```
class Ui_MainWindow(object):
```

```
    def setupUi(self, MainWindow):
```

```
        MainWindow.setObjectName("MainWindow")
```

```
        MainWindow.resize(1280, 720)
```

```
        MainWindow.setStyleSheet("background-color: rgb(160,32,240);")
```

```
        MainWindow.setStyleSheet("")
```

```
        icon = QtGui.QIcon()
```

```
        icon.addPixmap(QtGui.QPixmap("icon.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
```

```
        MainWindow.setWindowIcon(icon)
```

```
        self.centralwidget = QtWidgets.QWidget(MainWindow)
```

```
        self.centralwidget.setObjectName("centralwidget")
```

```
        self.id = QtWidgets.QPlainTextEdit(self.centralwidget)
```

```
        self.id.setGeometry(QtCore.QRect(920, 440, 181, 31))
```

```
        self.id.viewport().setProperty("cursor", QtGui.QCursor(QtCore.Qt.IBeamCursor))
```

```
        self.id.setStyleSheet("background-color: rgb(255, 255, 255);")
```

```
        self.id.setObjectName("id")
```

```
        self.label = QtWidgets.QLabel(self.centralwidget)
```

```
        self.label.setGeometry(QtCore.QRect(790, 440, 111, 41))
```

```
        self.label.setAutoFillBackground(False)
```

```
        self.label.setStyleSheet("color: rgb(0, 85, 127);\n"
```

```
"font: 14pt \"Nirmala UI\";")
```

```
        self.label.setObjectName("label")
```

```
self.head = QtWidgets.QPushButton(self.centralwidget)

self.head.setGeometry(QtCore.QRect(630, 570, 171, 31))

self.head.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";\n"
"color: rgb(83, 83, 255);\n"
"background-color: rgb(255,255,255);")

self.head.setObjectName("head")

self.teacher = QtWidgets.QPushButton(self.centralwidget)

self.teacher.setGeometry(QtCore.QRect(840, 570, 171, 31))

self.teacher.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";\n"
"color: rgb(83, 83, 255);\n"
"background-color: rgb(255,255,255);")

self.teacher.setObjectName("teacher")

self.student = QtWidgets.QPushButton(self.centralwidget)

self.student.setGeometry(QtCore.QRect(1050, 570, 171, 31))

self.student.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";\n"
"color: rgb(83, 83, 255);\n"
"background-color: rgb(255,255,255);")

self.student.setObjectName("student")

self.password = QtWidgets.QPlainTextEdit(self.centralwidget)

self.password.setGeometry(QtCore.QRect(920, 480, 181, 31))

self.password.setMaximumSize(QtCore.QSize(16777215, 31))

self.password.viewport().setProperty("cursor", QtGui.QCursor(QtCore.Qt.IBeamCursor))

self.password.setStyleSheet("background-color: rgb(255, 255, 255);")

self.password.setObjectName("password")

self.errorDisplay = QtWidgets.QLabel(self.centralwidget)

self.errorDisplay.setGeometry(QtCore.QRect(630, 620, 181, 41))

self.errorDisplay.setStyleSheet("font: 75 14pt \"MS Shell Dlg 2\";\n"
"color:  rgb(217, 48, 37);\n"
"qproperty-alignment: AlignCenter;")

self.errorDisplay.setObjectName("errorDisplay")

self.info = QtWidgets.QPushButton(self.centralwidget)

self.info.setGeometry(QtCore.QRect(130, 640, 181, 41))
```

```

self.info.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";\n"
"font: 10pt \"MS Shell Dlg 2\";\n"
"color: rgb(255, 255, 255);\n"
"background-color: rgb(52, 168, 83);")

self.info.setObjectName("info")

self.label_3 = QtWidgets.QLabel(self.centralwidget)
self.label_3.setGeometry(QtCore.QRect(840, 530, 171, 41))
self.label_3.setStyleSheet("color: rgb(255, 255, 255);\n"
"font: 12pt \"Nirmala UI\";")

self.label_3.setObjectName("label_3")

self.label_2 = QtWidgets.QLabel(self.centralwidget)
self.label_2.setGeometry(QtCore.QRect(260, 480, 641, 28))
self.label_2.setStyleSheet("color: rgb(160,32,240);\n"
"font: 14pt \"Nirmala UI\";")

self.label_2.setObjectName("label_2")

# Add label_5 to the layout and stretch it to fill available space
self.label_5 = QtWidgets.QLabel(self.centralwidget)
self.label_5.setGeometry(QtCore.QRect(0, 0, 1250, 700)) # Cover full screen
self.label_5.setPixmap(QtGui.QPixmap("Attendance.png"))
self.label_5.setScaledContents(True)
self.label_5.setObjectName("label_5")

self.label_5.raise_()
self.id.raise_()
self.label.raise_()
self.head.raise_()
self.teacher.raise_()
self.student.raise_()
self.password.raise_()
self.errorDisplay.raise_()
self.info.raise_()
self.label_3.raise_()
self.label_2.raise_()

```

```

MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Login"))
    self.label.setText(_translate("MainWindow", "<html><head><body><p align=\"right\"><span style=\"font-weight:600; color:#ffffff;\">Enter ID</span></p></body></html>"))
    self.head.setText(_translate("MainWindow", "Head of Institute"))
    self.teacher.setText(_translate("MainWindow", "Teacher"))
    self.student.setText(_translate("MainWindow", "Student"))
    self.errorDisplay.setText(_translate("MainWindow", "<html><head><body><p align=\"center\"><span style=\"font-weight:600; color:#ffffff;\"><br/></span></p></body></html>"))
    self.info.setText(_translate("MainWindow", "Information Related to Database"))
    self.label_3.setText(_translate("MainWindow", "<html><head><body><p align=\"center\"><span style=\"font-size:14pt; font-weight:600;\">Login As</span></p></body></html>"))
    self.label_2.setText(_translate("MainWindow", "<html><head><body><p align=\"right\"><span style=\"font-weight:600; color:#ffffff;\">Enter Password</span></p></body></html>"))

```

- Language Choice: Python is a well-suited choice due to its:
- Readability and simplicity.
- Extensive libraries for GUI development (Qt5) and database interaction (SQLite).
- GUI Framework: Qt5 provides a robust and cross-platform foundation for building the user interfaces.
- Database Management:
- SQLite offers a lightweight and embedded database solution suitable for storing attendance data efficiently.
- Python's sqlite3 module simplifies database interactions.

10. Testing

The testing models used are the combination of Scrum, eXtreme Programming (XP), and Dynamic Software Development Method (DSDM) to provide a robust framework for developing the Attendance Management System. Scrum facilitated efficient team collaboration and task management, XP ensured high-quality code and adaptability to changing requirements, and DSDM prioritized user involvement and structured delivery. Together, these Agile testing methods contributed to the successful development and deployment of the AMS.

For the Attendance Management System (AMS) project, the Agile testing methods utilized were Scrum and eXtreme Programming (XP). Here are the detailed points explaining their use:

➤ Scrum Methodology

1. Scrum Roles and Responsibilities

- **Scrum Master:** Facilitated the Scrum process, arranged meetings, and removed obstacles to ensure smooth progress. The Scrum Master also guided the team in following Agile principles.
- **Product Owner:** Created and managed the product backlog, prioritizing tasks based on business value and user feedback. The Product Owner ensured that the most critical features were developed first.
- **Scrum Team:** Consisted of developers, designers, and testers who collaboratively managed their work and organized tasks to complete each sprint. The team self-organized to achieve sprint goals.

2. Sprint Planning and Execution

- **Sprint Planning:** At the beginning of each sprint, the team and Product Owner collaborated to select user stories from the product backlog to work on. These stories were broken down into tasks and estimated.
- **Sprint Duration:** Each sprint lasted two weeks, allowing the team to deliver incremental updates frequently. Short sprints helped in adapting to changes quickly.
- **Daily Stand-Ups:** Daily meetings were held to discuss progress, identify impediments, and plan the day's tasks. This ensured continuous communication and quick resolution of issues.

3. Sprint Review and Retrospective

- **Sprint Review:** At the end of each sprint, the team demonstrated the completed work to stakeholders, gathering feedback to refine future sprints. This iterative feedback loop helped in aligning the product with user expectations.
- **Sprint Retrospective:** The team reflected on the sprint to discuss what went well, what could be improved, and actionable items for improvement in the next sprint. This fostered continuous improvement.

4. Product Backlog and Burndown Chart

- **Product Backlog:** A dynamic list of user stories and tasks that needed to be completed. The backlog was constantly refined based on feedback and changing requirements.
- **Burndown Chart:** Used to visualize the progress of the sprint, showing the remaining work and helping the team stay on track.

➤ **eXtreme Programming (XP)**

1. Continuous Integration

- **Frequent Code Integration** Developers integrated code changes frequently, often multiple times a day. This practice minimized integration issues and ensured that the system was always in a deployable state.
- **Automated Testing:** Automated unit tests were run with each integration to catch defects early and ensure that new changes did not break existing functionality.

2. Test-Driven Development (TDD)

- **Writing Tests First:** Developers wrote automated test cases before writing the actual code. This practice ensured that the code met the specified requirements and improved code quality.
- **Refactoring:** Continuous refactoring was performed to improve code structure without changing its behavior, making the codebase more maintainable.

3. Pair Programming

- **Collaborative Coding:** Two developers worked together at a single workstation, one writing the code (driver) and the other reviewing it (observer). This practice enhanced code quality and knowledge sharing within the team.

4. User Stories and Customer Involvement

- **User Stories:** Requirements were captured as user stories, providing a clear and concise description of the desired functionality from the user's perspective.
- **Customer Involvement :** Frequent interactions with stakeholders and end-users were maintained to gather requirements, provide updates, and get feedback. This ensured that the product met user needs and expectations.

5. Simplicity and Feedback

- **Simplicity:** XP emphasized simplicity in design and implementation, focusing on delivering only what was necessary and avoiding over-eng

11. User manual

11.1 Used Software

➤ Overview

- Developer: Microsoft
- Type: Source-code editor
- License: Free and open source

➤ Key Features

- **Cross-Platform Support:** Works on Windows, macOS, and Linux for a consistent development experience.
- **IntelliSense:** Provides smart code completions, boosting productivity.
- **Built-in Git Integration:** Manages source control with Git commands, diffs, branching, and history.
- **Extensions Marketplace:** Offers thousands of extensions for enhanced functionality, including language support, debuggers, themes, and utilities.
- **Debugger:** Supports debugging for multiple programming languages, allowing for breakpoints, stepping through code, and variable inspection.
- **Customizability:** Highly customizable with settings, keybindings, snippets, and themes to tailor the editor to your needs.
- **Integrated Terminal:** Built-in terminal to run shell commands within the editor, supporting multiple terminals and various shells.
- **Code Navigation:** Features like Go to Definition, Peek Definition, and Find All References for easy navigation, especially in large projects.
- **Code Refactoring:** Improves code quality with tools for renaming variables, extracting methods, and moving files or classes.
- **Live Share:** Enables real-time collaborative coding by sharing your VS Code session with others, supporting co-editing, debugging, and sharing terminals.
- **Remote Development:** Work with remote servers, containers, or WSL (Windows Subsystem for Linux) seamlessly through extensions.

➤ **Benefits**

- **Free and Open-Source:** Cost-effective and widely supported by the community.
- **Lightweight and Fast:** Optimized for performance with a small footprint.
- **Wide Language Support:** Supports multiple languages out-of-the-box, with extensions for many more.
- **Active Community:** Benefits from regular updates and a large community contributing extensions.
- **Integration with Other Tools:** Integrates easily with build tools, linters, and task runners.

➤ **How to Use Effectively**

• **Setting Up**

- 1.Download and install from the official website.
- 2.Customize settings or modify settings.json.
- 3.Install extensions from the Extensions view.

• **Basic Usage**

- 1.Open your project directory using File > Open Folder.
- 2.Access the terminal using View > Terminal or Ctrl+.
- 3.Manage Git repositories using the Source Control view.

• **Advanced Usage**

- 1.Set breakpoints and use the Debug view for debugging.
- 2.Leverage Ctrl+P for opening files, F12 for going to definitions, and Shift+F12 for finding references for code navigation.
- 3.Create and use code snippets for repetitive tasks using File > Preferences > User Snippets.

• **Collaboration and Remote Development**

- 1.Install the Live Share extension for real-time collaboration.
- 2.Use the Remote Development extension pack to connect to remote environments

VS Code is a powerful and versatile code editor suitable for various development tasks. Its extensive features and vibrant extension ecosystem make it a top choice for developers seeking a customizable and efficient coding environment. Whether you're working on small scripts or large projects, VS Code offers the tools and integrations to enhance your productivity and workflow.

11.2 Screenshots

Login Page

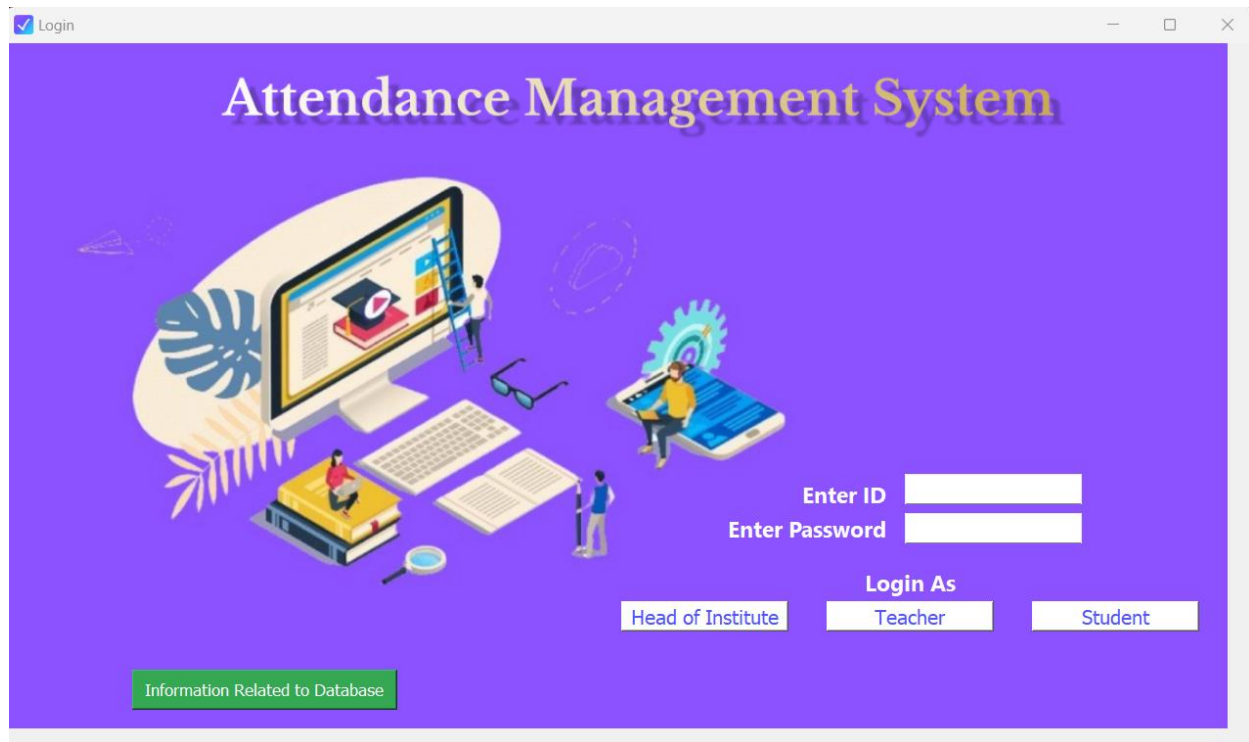


Figure 18 Login Page

Manage Students and Teachers

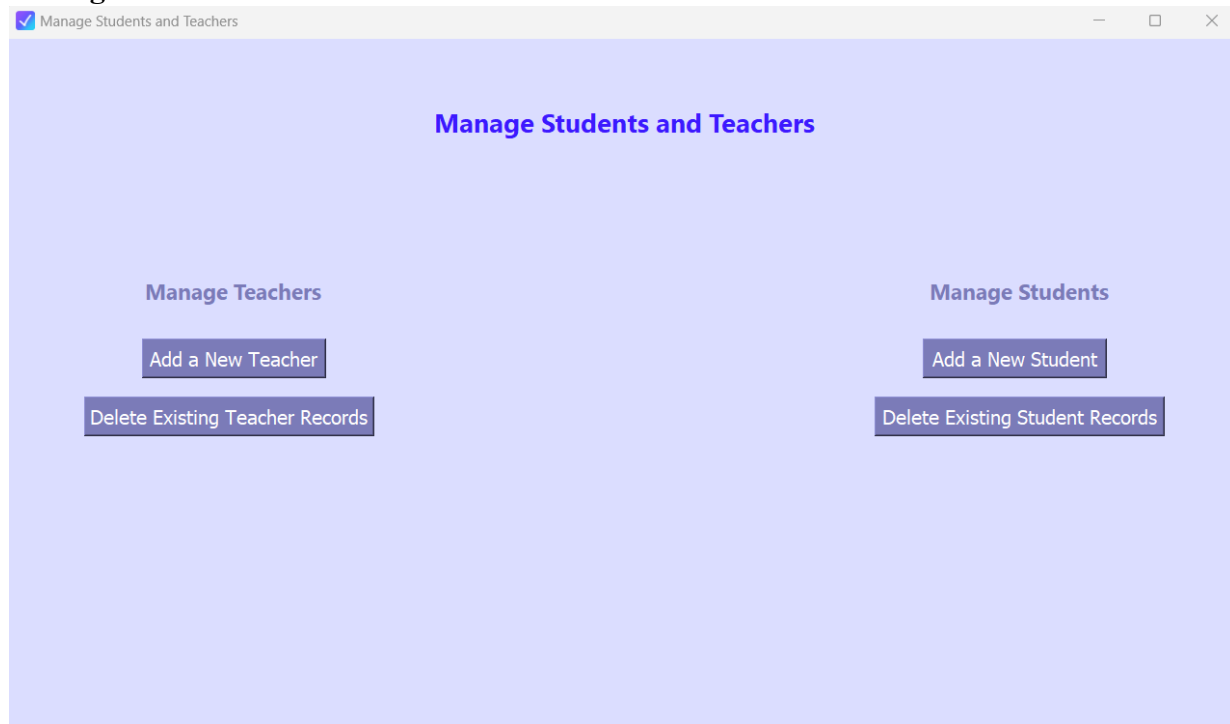
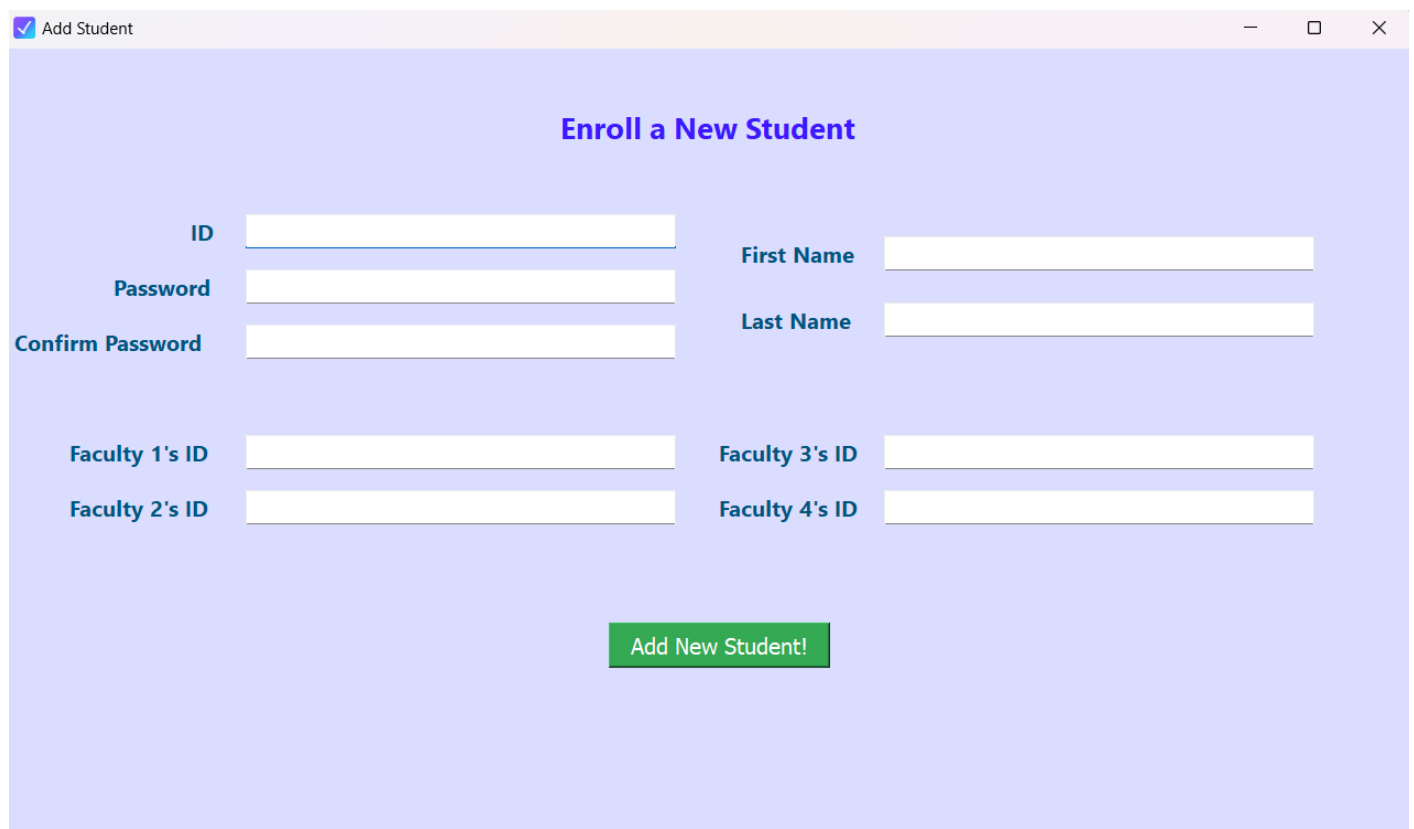


Figure 19 Manage Students and Teachers

Add Teacher / Faculty

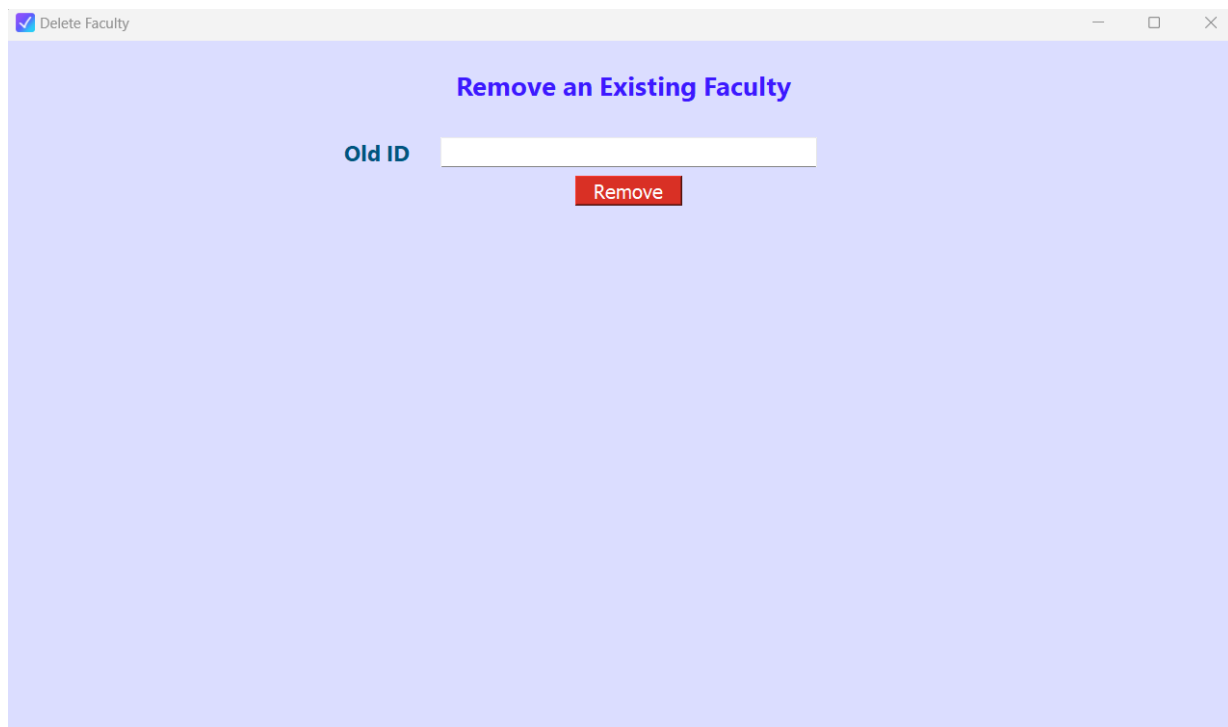
The screenshot shows a web browser window titled "Add Teacher". The main heading is "Assign a New Faculty" in blue. Below the heading, there are six input fields arranged in two columns. The left column contains "ID", "Password", and "Confirm Password". The right column contains "First Name", "Last Name", and "Course ID". All fields are empty. At the bottom center, there is a green button labeled "Add New Teacher!".

*Figure 20 Add Teachers/Faculty***Add Students**

The screenshot shows a web browser window titled "Add Student". The main heading is "Enroll a New Student" in blue. Below the heading, there are nine input fields arranged in three rows and two columns. The first three rows are for student information: "ID", "Password", "Confirm Password" on the left, and "First Name", "Last Name" on the right. The last two rows are for faculty assignments: "Faculty 1's ID", "Faculty 2's ID" on the left, and "Faculty 3's ID", "Faculty 4's ID" on the right. All fields are empty. At the bottom center, there is a green button labeled "Add New Student!".

Figure 21 Add Students

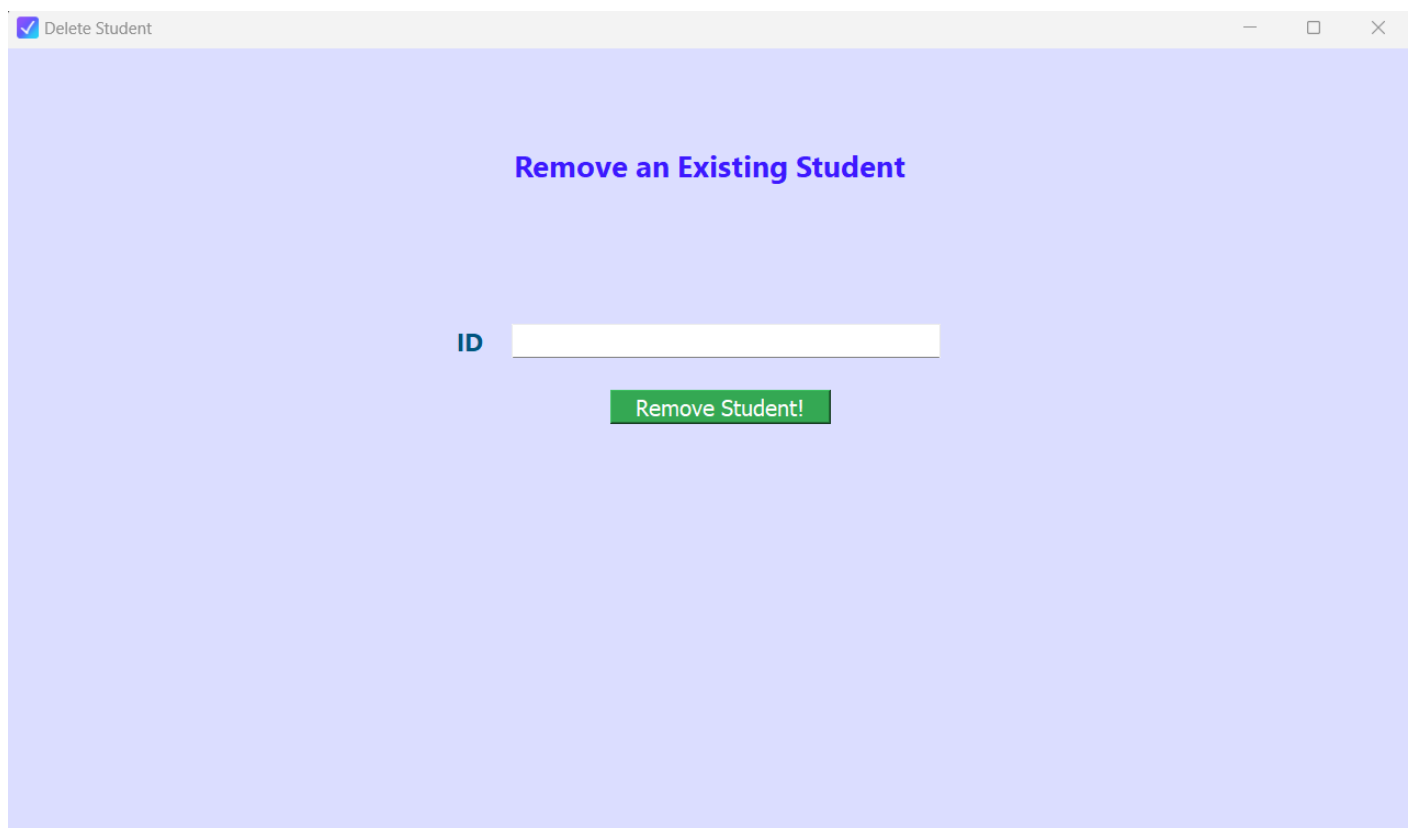
Delete Teachers / Faculty



The screenshot shows a web browser window titled "Delete Faculty". The page has a light blue background. At the top center, the text "Remove an Existing Faculty" is displayed in a bold, dark blue font. Below this text, on the left, is the label "Old ID" in a dark blue font. To the right of the label is a white rectangular input field. Directly beneath the input field is a red rectangular button with the word "Remove" written in white text.

Figure 22 Delete Teachers/Faculty

Delete Students



The screenshot shows a web browser window titled "Delete Student". The page has a light blue background. At the top center, the text "Remove an Existing Student" is displayed in a bold, dark blue font. Below this text, on the left, is the label "ID" in a dark blue font. To the right of the label is a white rectangular input field. Directly beneath the input field is a green rectangular button with the text "Remove Student!" written in white text.

Figure 23 Delete Students

Attendance Sheet

Attendance Sheet

Attendance Sheet of 2CE1 DSA

Start Taking Attendance

Attendance Sheet

Attendance Sheet of 2CE1 DSA

Student's ID

22172012038

Student's Name

Suhada Khan

Enter Status

☐ Absent ☒ Present

Next

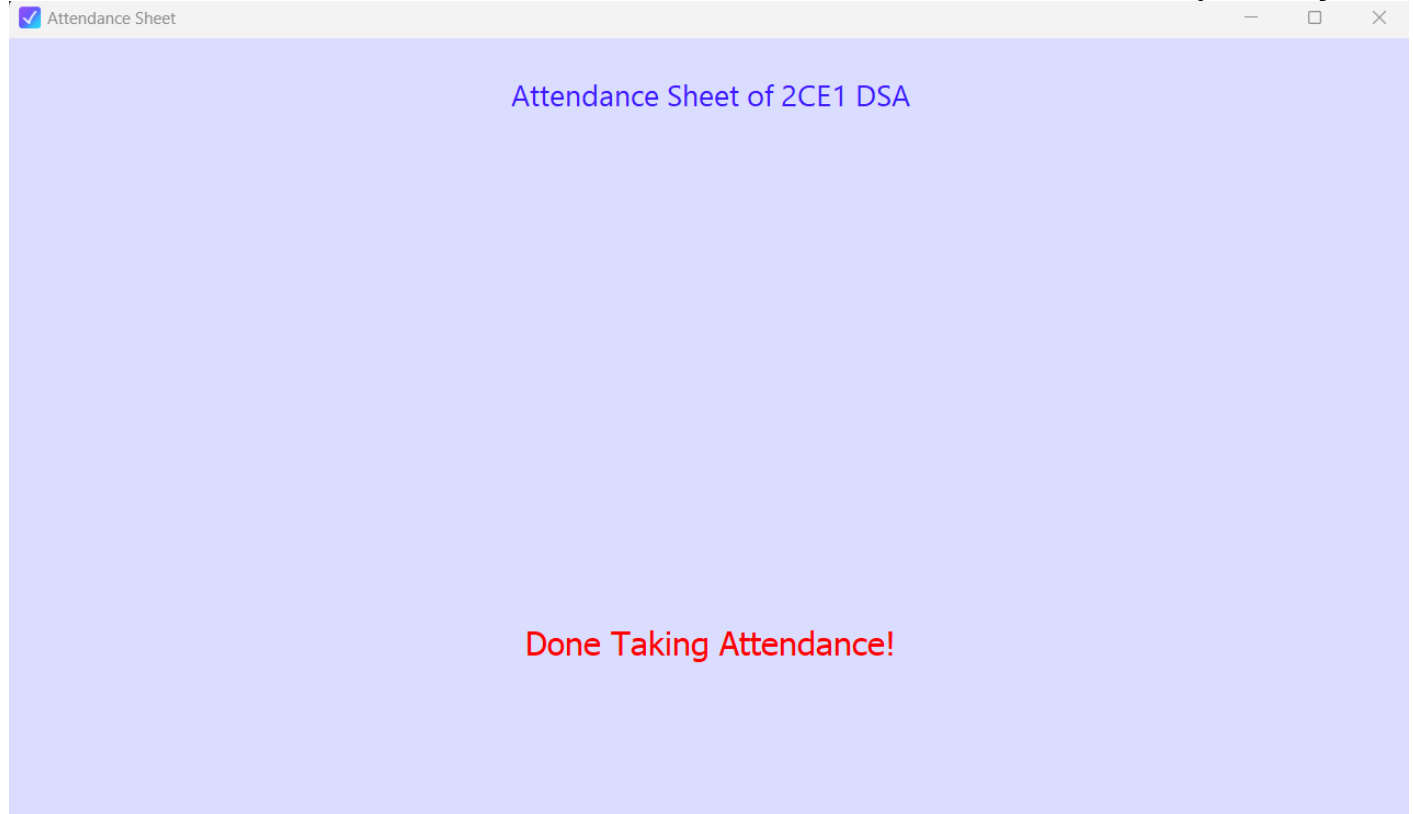


Figure 24 Attendance Sheet

Attendance Report

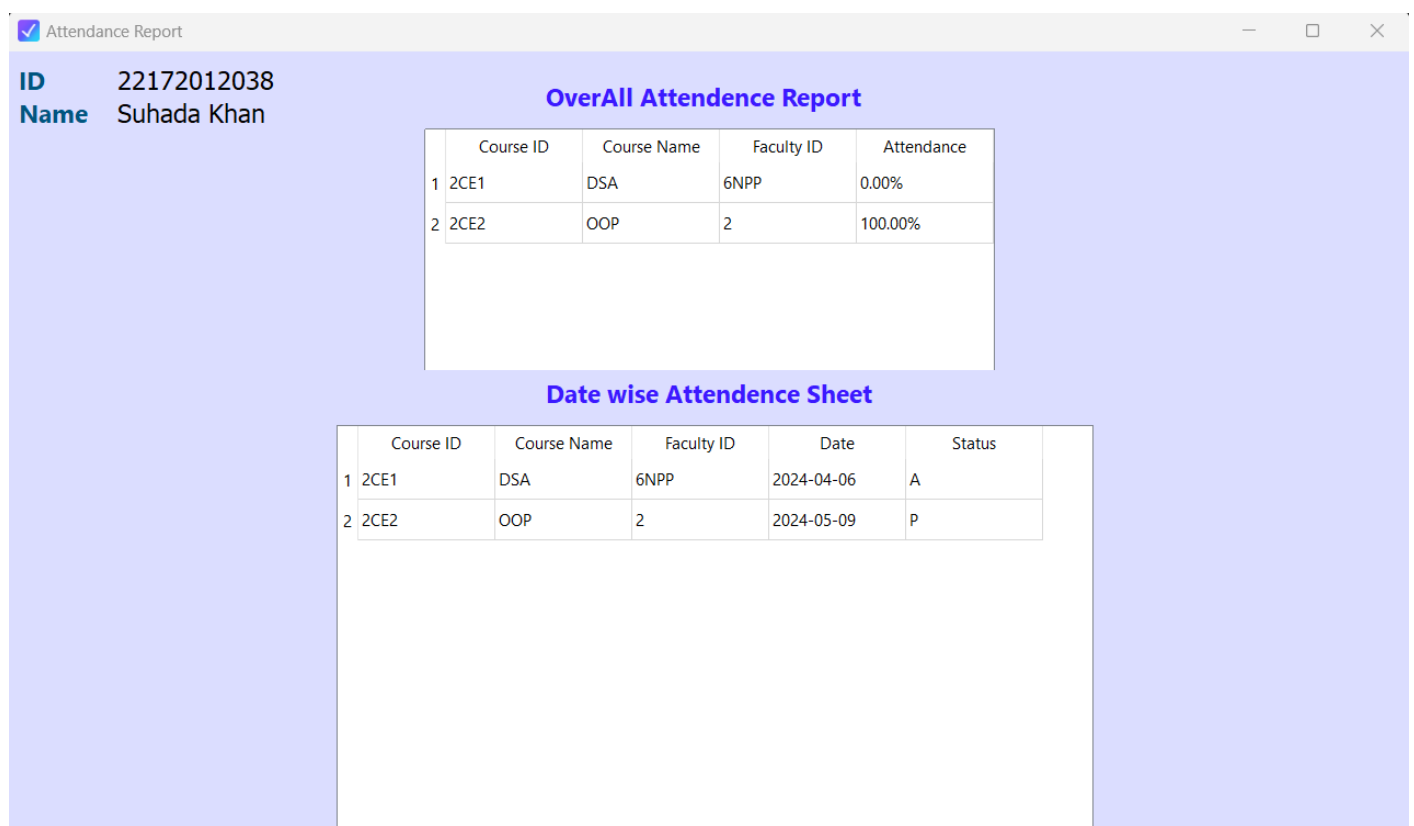


Figure 25 Attendance Report

Information about Database

Teacher Details		
First Name	Last Name	Course ID
Nishi	Patwa	2CE1
Om	Prakash	2CE2
Megha	Patel	2CE3
Paresh	Solanki	2CE4
Rajul	Suthar	2CE5
Chirag	Patel	2CE6
Ravi	Raval	2CE7
Devang	Pandeya	2CE8
Mannan	Thakkar	2CE9
Bhavesha	Suthar	2CE10

Course Details	
Course ID	Course Name
2CE1	DSA
2CE2	OOP
2CE3	DBMS
2CE4	OS
2CE5	TOC
2CE6	AI
2CE7	SVT
2CE8	ITIM
2CE9	CP2
2CE10	CC

Figure 26 Information about Database

12. Conclusion and Future Work

Conclusion:

The Attendance Management System (AMS) described utilizes Python, Qt5, and SQLite to streamline student attendance tracking within a department. It offers a user-friendly experience with distinct interfaces for admins, faculty, and students. The system efficiently stores data in an SQLite database.

Future Work:

- **Enhanced Security:** Implement secure password hashing (e.g., bcrypt) to protect user credentials. Validate user input to prevent SQL injection vulnerabilities.
- **Scalability Enhancements:** As the user base grows, consider optimizing database performance and GUI responsiveness.
- **Advanced Reporting:** Develop features for generating detailed attendance reports (e.g., individual student reports, class-wise reports, filtered reports by date or course).
- **Multi-Department Support:** Extend the system to manage student attendance across multiple departments, potentially integrating with a central user management system.
- **Mobile App Development:** Develop a mobile application for students and/or faculty to access attendance information on the go.
- **Face Recognition:** Future development of attendance management system will integrate advanced AI for facial recognition, enhancing accuracy and security, while also optimizing efficiency through real-time monitoring and analytics.
- **Additionally,** it supports the leave application features along with more functionalities and modules.

13. Annexure

13.1 Glossary of Terms and Abbreviations

- **AMS:** Attendance Management System
- **GUI:** Graphical User Interface
- **Qt5:** A cross-platform application framework
- **SQLite:** A lightweight, embedded database management system
- **ID:** Identification Number
- **Password:** Secret access code
- **Database:** A collection of structured data

13.2 References

Qt5 Documentation: <https://doc.qt.io/>

SQLite Documentation: <https://www.sqlite.org/docs.html>

Secure Password Hashing in Python (using bcrypt): <https://www.geeksforgeeks.org/hashing-passwords-in-python-with-bcrypt/>

Chat GPT: <https://chat.openai.com/>

Wikipedia: <https://www.wikipedia.org/>

Freeprojectz :<https://www.freeprojectz.com/>

HubSpot: <https://www.hubspot.com/products/crm>

JavatPoint: <https://www.javatpoint.com/>

TechTarget: <https://www.techtarget.com/>

1000 Projects: 1000 Projects

13.3 About Tools and Technology

Python: A general-purpose, high-level programming language known for its readability and extensive libraries.

Qt5: A powerful framework for developing cross-platform applications with rich graphical user interfaces.

SQLite: A self-contained, lightweight database engine well-suited for storing and managing structured data like the AMS's attendance records.

13.4 About College (UVPCE) :

Ganpat University-U. V. Patel College of Engineering (GUNI-UVPCE), established in 1997, aims to mold students into technically adept individuals. With a sprawling 25-acre campus, it offers diverse undergraduate, postgraduate, and Ph.D. programs. The institute prioritizes industry-relevant education, fostering ties with prominent companies for placements and research collaboration. Boasting superior infrastructure, including CoEs like Bosch-Rexroth and Energy Innovation Centre, it emphasizes practical learning. Named after industrialist Shri Ugarchandbhai Varanasi Bhai Patel, GUNI-UVPCE is approved by AICTE and offers a conducive environment for learning, research, and training, contributing to the holistic development of aspiring engineers.