ABC COLLEGE WEBSITE

A Project Report

Submitted in partial fulfillment of the Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

GYAN GUPTA (8433) and PRATHAMESH GAWADE (8430)

Under the esteemed guidance of

Prof. Omkar Sherkhane

DEPARTMENT OF INFORMATION TECHNOLOGY

MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE (AUTONOMOUS)

(*Affiliated to University of Mumbai)*

NEW PANVEL – 410206 , MAHARASHTRA

YEAR 2023-2024

MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS COMMERCE & SCIENCE

(AUTONOMOUS)

NEW PANVEL, MAHARASHTRA-410206

(Affiliated to University of Mumbai)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project is entitled **"ABC COLLEGE WEBSITE"** is Bonafede work of **Mr. GYAN GUPTA** bearing Seat No: **8433** submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Date: **Coordinator**

Internal guide **College Seal** **External Examiner**

ACKNOWLEDGEMENT

I, **Mr. GYAN GUPTA** student of **Pillai College Of Arts, Commerce & Science (Autonomous), New Panvel** would like to express my sincere gratitude towards our college's Information Technology Department.

I would like to thank Mrs. Deepika Sharma (Vice Principal) for granting me the opportunity to build a project for the college. Last but not least I thank our guide Prof. Omkar Sherkhane for his constant support during this project. The project would have not been completed without the dedication, creativity and the enthusiasm my family provided me.

Yours faithfully,

GYAN GUPTA

(Final Year Information Technology)

DECLARATION

I hereby declare that the project entitled, **"ABC COLLEGE WEBSITE"** done at **Pillai College Of Arts, Commerce & Science (Autonomous), New Panvel**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

Mr. GYAN GUPTA

TABLE OF CONTENTS

ABC COLLEGE WEBSITE

A Project Report

Submitted in partial fulfillment of the
Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

GYAN GUPTA(8433) and PRATHAMESH GAWADE (8430)

Under the esteemed guidance of

Prof. Omkar Sherkhane

DEPARTMENT OF INFORMATION TECHNOLOGY

MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(AUTONOMOUS)

(*Affiliated to University of Mumbai)*

NEW PANVEL – 410206 , MAHARASHTRA

YEAR 2022-2023

MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS COMMERCE & SCIENCE

(AUTONOMOUS)

NEW PANVEL, MAHARASHTRA-410206

(Affiliated to University of Mumbai)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project is entitled **"ABC COLLEGE WEBSITE"**, is bonafide work of **Mr. GYAN GUPTA** bearing Seat.No: **8433** submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Date: **Coordinator**

Internal guide **College Seal** **External Examiner**

ACKNOWLEDGEMENT

I, **Ms. GYAN GUPTA** student of **Pillai College Of Arts, Commerce & Science (Autonomous), New Panvel** would like to express my sincere gratitude towards our college's Information Technology Department.

I would like to thank Mrs. Deepika Sharma (Vice Principal) for granting me the opportunity to build a project for the college. Last but not least I thank our guide Prof. Omkar Sherkhane for his constant support during this project. The project would have not been completed without the dedication, creativity and the enthusiasm my family provided me.

Yours faithfully,

GYAN GUPTA

(Final Year Information Technology)

DECLARATION

I hereby declare that the project entitled, **"ABC COLLEGE WEBSITE"** done at **Pillai College Of Arts, Commerce & Science (Autonomous), New Panvel**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

Mr. GYAN GUPTA

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

This Project Named 'Event Manager' Basically it is a college website, in which I had to try to do something different to make it unique. So we know that a college website content like home page, career, admission, contact page, gallery of the college, but I had added some new and different things to look different. My website has 11 .aspx pages with an extra Master page, with some css pages and any other. The following are the web pages I have in my website.

Page Name	Description
carrer.aspx	Page for career-related information.
contact_us.aspx	Contact page for users to reach out.
createdtimetable.aspx	Displays a message after successfully creating a timetable.
Event.aspx	Allows participation in events.
event.css	CSS file for styling event-related elements.
faculty_dashboard.aspx	Page for faculty to create timetables viewed by students and teachers.
home.aspx	Homepage of the website.
login.aspx	Login page for user authentication.
logion.css	CSS file for styling the login page.
logout.aspx	Logout page.
logout.css	CSS file for styling the logout page.
site1.master	Master page containing header and footer for the website.
style.css	CSS file for styling the master page.
Student_dashboard.aspx	Page for students to view the timetable.
Teacher_dashboard.aspx	Page for teachers to view the timetable.
Upload_certificate.aspx	Page for students to upload certificates for courses done.
viewcertificates.aspx	Page for teachers/faculty to view the certificates uploaded by students.
upload_certificates.css	CSS file for styling the certificate upload page.
viewcertificates.css	CSS file for styling the view certificates page.

Figure 1: Names of web form pages

1.1) Background:

In today's educational landscape, colleges have embarked on a new initiative to enhance the credit skills of their students. These credit skills encompass certification courses obtained either externally or within the same institution. Each certificate, with a minimum duration of 30 hours, is eligible for the allocation of 2 credit points. However, a prevailing concern within this process has arisen with certain students engaging in the unethical practice of altering their certificates by merely changing their names before submission. This raises a significant challenge, as faculty members encounter difficulties in adequately verifying these certificates.

To address this pressing issue, I have devised a solution that can provide colleges with a reliable source of certification documentation. This approach will enable faculty members to conduct thorough and accurate verifications of certificates, eliminating any potential for fraudulent submissions.

Our college frequently organizes numerous events throughout the year, and to effectively promote these events and sell event passes, committee members are traditionally required to engage in marketing efforts, persuading fellow students to purchase the event passes.

To address this challenge, I have conceived an innovative solution: the implementation of an online pass sales system. By adopting this approach, committee members can streamline their promotional efforts. Students interested in attending these events can conveniently complete an online registration form and make online payments. Subsequently, a confirmation message will be promptly sent to their provided email address. This digital approach not only saves time but also enhances the efficiency of event pass sales, marketing, advertisement and registration processes.

I have introduced a new component to the website: a timetable page. This addition serves to benefit both college teaching staff and students significantly. The inclusion of this feature simplifies the process of class allocation for teachers. Moreover, students can conveniently access their timetables by logging in with their respective college email IDs. It is worth noting that staff and students possess distinct types of timetables tailored to their specific needs.

Firstly, it aims to streamline class allocation for college teaching staff, making the scheduling process more efficient and less prone to conflicts. Secondly, it seeks to provide students with improved accessibility to their timetables, allowing them to plan their academic activities more effectively. Lastly, by enabling direct login with their college email IDs, the feature enhances security and ensures that staff and students have access to timetables specifically tailored to their roles and requirements.

1.2) Objectives:

1. Certification Documentation Solution:

Objective: To ensure the integrity and accuracy of certification documentation while simplifying the verification process.

Rationale: To address the unethical practice of certificate alteration by students and to provide faculty members with a reliable source for verifying the authenticity of certificates.

2. Online Event Pass Sales System:

Objective: To streamline the promotion, sale, and registration processes for college events, saving time and enhancing efficiency.

Rationale: To eliminate the need for committee members to engage in traditional marketing efforts and to provide students with a convenient digital platform for event registration and payment.

3. Timetable Page :

Objective: To optimize the class allocation process for teaching staff and improve timetable accessibility for students, enhancing overall academic planning.

Rationale: To simplify the scheduling process, reduce conflicts, and ensure secure and convenient access to timetables for both staff and students.

In summary, the common objectives across these initiatives are to enhance operational efficiency, reduce fraudulent activities, and provide a more user-friendly and secure experience for both staff and students within the college environment. These improvements aim tou contribute to a more effective and streamlined educational experience.

1.3) Purpose:

- **Streamlined Information Access:**

To centralize access to essential information and resources related to the college, its programs, and services. This includes providing timetables, library resources, and career guidance.

- **Enhanced Communication:**

To facilitate effective communication between students, staff, and the institution. This includes features like contact information and an "about us" page.

- **Efficient Academic Planning:**

To help students and staff plan their academic activities efficiently by providing easy access to class schedules and timetables.

- **Credential Verification:**

To ensure the integrity of certification documentation and streamline the verification process for academic certificates.

- **Event Promotion and Management:**

To promote college events and streamline the process of selling event passes and managing event registrations.

- **Resource Sharing:**

To provide a platform for sharing and showcasing resources such as images and photos in a gallery format.

- **User Authentication and Styling:**

To ensure secure access to user-specific information through a login page while maintaining a consistent and visually appealing website design using CSS.

In essence, the purpose of this project is to create a comprehensive digital hub for the college community, catering to the academic, informational, and administrative needs of students, staff, and other stakeholders. It aims to enhance efficiency, communication, and user experience within the college environment.

1.4 Scope:

The scope of the entire project involves the development of a comprehensive digital platform for your college community, catering to the academic, informational, and administrative needs of students, staff, and other stakeholders.

1) Future Development Considerations:

- Continuous updates and maintenance to ensure security and compatibility with evolving web technologies and browsers.
- Periodic content updates and expansion of informational pages based on college developments and user feedback.
- Integration of additional features, such as discussion forums, news feeds, or online resource libraries.
- Enhanced user interactivity and engagement through features like surveys, feedback forms, and polls.
- Implementation of analytics tools to gather insights on user behaviour and preferences, allowing for data-driven improvements.
- Expansion of the gallery to support multimedia content and user-generated contributions.
- Integration with external systems or services for seamless data exchange, such as connecting with the college's academic systems for real-time timetable updates.
- Regular security audits and updates to protect user data and maintain the platform's integrity.
- Potential expansion to include a mobile app version of the platform for improved accessibility.
- Collaboration with stakeholders to gather ongoing feedback and identify areas for improvement.
- Consideration of accessibility features to ensure the platform is usable by individuals with disabilities.

This comprehensive scope for both the initial development and future enhancements ensures that the project remains adaptable and continues to meet the evolving needs of the college community. It reflects a commitment to creating a dynamic and valuable digital hub for students, staff, and other users.

1.4 Applicability:

The applicability of the project on which I am working currently is quite significant, as it addresses several common challenges and needs within an educational institution like a college.

2) **Credential Verification:**
    a) **Applicability:** This feature is crucial for ensuring the integrity and accuracy of certification documentation. It directly addresses the issue of students altering certificates.
    b) **Benefit:** It provides faculty members with a reliable and efficient way to verify certificates, which is a common requirement in academic and professional settings.

3) **Online Event Pass Sales System:**
    a) Applicability: This system streamlines event promotion, ticket sales, and registration processes.
    b) Benefit: It simplifies the process for event organizers and makes it convenient for students to register and attend college events. This is applicable for colleges that host various events throughout the year.

4) **Timetable Page:**
    a) Applicability: Timetables are a fundamental part of college operations, benefiting both teaching staff and students.
    b) Benefit: It optimizes class allocation, reducing scheduling conflicts for staff and providing students with a convenient way to access their schedules. This is applicable to all colleges with class schedules.

5) **Information Centralization:**
    a) **Applicability:** Centralizing essential college information and resources is applicable to all colleges, as it helps students and staff access important information easily.
    b) **Benefit:** It simplifies information retrieval, improving the overall efficiency of college operations.

6) **Enhanced Communication:**
    a) **Applicability:** Effective communication is essential in any educational institution.
    b) **Benefit:** Features like contact information and an "about us" page help facilitate communication between students, staff, and the institution. This is universally applicable.

7) **Resource Sharing:**
    a) **Applicability:** Sharing resources like images and photos is relevant for colleges looking to promote events and showcase their activities.
    b) **Benefit:** It enhances marketing and engagement efforts, making it applicable to colleges with active event and activity calendars.

8) **User Authentication and Styling:**
    a) **Applicability:** User authentication ensures secure access to personalized information.
    b) **Benefit:** It's essential for maintaining data security and providing a consistent, visually appealing user experience, applicable to all websites with user-specific content.

In summary, the applicability of your project extends to a wide range of colleges and educational institutions. The features and functionalities you've developed address common challenges and enhance the overall educational experience, making it relevant and valuable for various stakeholders within a college community.

CHAPTER 2: SYSTEM PLANNING

System planning is a crucial phase in the development of any project, including your college's educational initiative website. It involves several key steps, starting with a survey of technologies to determine the best tools and frameworks for your project. This is followed by fact-finding techniques to gather requirements and understand the needs of stakeholders. A feasibility study is then conducted to assess the practicality and potential success of the project. Finally, stakeholders are identified and their interests and expectations are considered throughout the planning process. These steps lay the foundation for a successful project by ensuring that the technology, requirements, and stakeholders are all aligned towards achieving the project's goals.

2.1) Survey of technologies:

A. Frontend Development:

- HTML/CSS/JavaScript: For building the user interface and adding interactivity.
- ASP.NET (C#): For server-side rendering and integrating backend logic.
- ASP.NET Labels: For displaying dynamic data from the backend.

B. Backend Development:

- ASP.NET (C#): Since you're using ASP.NET for the frontend, you can continue using C# for the backend logic.

C. Database:

- SQL Server (Local .mdf file): Using a local SQL Server database file (.mdf) for storing data.

D. Other Considerations:

- Git Repository: Using Git for version control and collaboration.
- Monitoring: Monitoring the application can be done through SQL Server Management Studio or similar tools to track database performance and queries.

2.2) Fact finding technique:

"In the planning phase of the project, several fact-finding techniques were considered for gathering information and requirements. These techniques would have included:

- **Interviews:** Conducting interviews with stakeholders such as students, faculty, and administrators to understand their needs and expectations for the website.
- **Questionnaires/Surveys:** Creating questionnaires or surveys to collect feedback and preferences from a larger group of stakeholders.
- **Observation:** Spending time observing how students, faculty, and staff interact with the current systems or processes.
- **Document Analysis:** Reviewing existing documents, reports, and data related to the college's educational initiatives.
- **Prototyping:** Creating prototypes or mockups of the website to gather feedback from stakeholders.
- **Focus Groups:** Organizing focus groups with stakeholders to discuss specific aspects of the website design or functionality.
- **Brainstorming:** Holding brainstorming sessions with a diverse group of stakeholders to generate ideas and gather input on potential features or improvements for the website.

These fact-finding techniques would have been instrumental in gathering comprehensive information and requirements for the project, ensuring that it meets the needs and expectations of Faculty and Teachers."

2.3) Feasibility Study:

The feasibility study aims to assess the viability of developing a college website as a final year project. The website is intended to enhance the educational experience of students and improve communication between faculty, students, and administration.

a. Project Description

The college website will include features such as course information, event announcements, student-faculty interaction forums, and online certificate submission. The objectives include improving information accessibility and promoting student engagement.

b. Technical Feasibility

1. The website will be developed using ASP.NET and SQL Server, technologies readily available and familiar to the development team.
2. The college has sufficient technical expertise to support the development and maintenance of the website.
3. The website can be easily integrated with existing college systems and infrastructure.

c. Economically Feasibility

1. The estimated cost for developing and maintaining the website is within the allocated budget.
2. The potential benefits, including improved communication and administrative efficiency, outweigh the costs.
3. The return on investment is expected to be positive over the project's lifespan.

d. Operational Feasibility

1. The website will complement existing college operations and can be easily adopted by staff and students.
2. Minimal disruption to day-to-day operations is expected during the implementation phase.
3. Training will be provided to ensure smooth adoption of the website.

e. Schedule Feasibility

1. The project timeline allows for the completion of key milestones and deliverables within the specified timeframe.
2. Potential risks, such as technical challenges or resource constraints, have been identified and mitigation strategies are in place.
3. Regular progress reviews will be conducted to ensure project deadlines are met.

f. Legal and Ethical Feasibility

1. The website will comply with all relevant legal and regulatory requirements, including data protection and privacy laws.
2. Ethical considerations, such as ensuring fair access to information and protecting user data, will be addressed through proper design and implementation.

g. Conclusion

1. Based on the findings of the feasibility study, it is recommended to proceed with the development of the college website. The project is deemed viable and is expected to provide significant benefits to the college and its stakeholders.

2.4) Stakeholders:

Stakeholders are individuals or groups who have an interest in the outcome of a project. For your college website project, stakeholders may include:

1) **College Administration:** They are interested in improving communication with students and faculty, enhancing the college's online presence, and streamlining administrative processes.
2) **B. Faculty Members:** They are interested in accessing and managing course information, posting announcements, and facilitating student interaction online.
3) **Students:** They are interested in accessing course materials, event information, and online forums for interaction with peers and faculty.
4) **IT Department:** They are interested in supporting the technical infrastructure required for the website, ensuring data security, and maintaining the website.
5) **Parents:** They may be interested in accessing information about the college and their child's academic progress.
6) **Prospective Students:** They are interested in accessing information about courses, admissions, and campus life.
7) **Community Members:** They may be interested in accessing information about community events hosted by the college and community outreach programs.

Understanding the needs and expectations of these stakeholders is crucial for the success of the project. Engaging with stakeholders throughout the development process can help ensure that the website meets their requirements and provides value to the college community.

CHAPTER 3:REQUIREMENT AND ANALYSIS

The Requirement and Analysis chapter focuses on defining the problem statement, specifying project requirements, planning and scheduling the development process, and identifying the software and hardware requirements. This chapter also includes conceptual models such as the ER Diagram, Use Case Diagram, Class Diagram, Sequence Diagram, Package Diagram, Deployment Diagram, and System Flowchart. These models provide a visual representation of the system's structure, behavior, and interactions, helping to clarify and organize the project's requirements and design. The chapter serves as a foundation for the development phase, ensuring that the project meets the needs of stakeholders and is implemented efficiently.

3.1 Problem Definition

The development of the college website project is driven by a comprehensive understanding of the challenges and issues faced by our educational institution. This section outlines the key problems and concerns that the website aims to address:

1. Certification Documentation Integrity:

Challenge: The prevailing issue of students engaging in unethical practices, such as altering certificates, presents a significant challenge.
Issue: The authenticity and accuracy of certification documentation are at risk due to these unethical practices.
Objective: To ensure the integrity and accuracy of certification documentation while simplifying the verification process for faculty members.

2. Inefficient Event Promotion and Pass Sales:

Challenge: Traditional event promotion methods rely heavily on committee members engaging in manual marketing efforts.
Issue: This approach is time-consuming, and event pass sales, marketing, and registration processes lack efficiency.
Objective: To streamline event promotion, ticket sales, and registration processes by introducing an online event pass sales system.

3. Timetable Complexity and Accessibility:

Challenge: Class allocation for teaching staff can be a complex and time-consuming process, leading to scheduling conflicts.
Issue: Students often struggle to access and navigate their class schedules efficiently.
Objective: To optimize the class allocation process for teaching staff and improve timetable accessibility for students, enhancing overall academic planning.

4. Dispersed Information and Communication:

Challenge: Essential college information and communication channels are dispersed across various platforms and channels.

Issue: This dispersion leads to difficulties in accessing crucial resources and effective communication between students, staff, and the institution.

Objective: To centralize access to essential information and resources related to the college, its programs, and services, while enhancing communication within the college community.

5. Credential Verification Complexity:

Challenge: Faculty members face difficulties in adequately verifying certification documents submitted by students.

Issue: The lack of a reliable source for certificate verification can lead to fraudulent submissions going undetected.

Objective: To provide faculty members with a reliable source for verifying the authenticity of certificates, simplifying the verification process and reducing the potential for fraudulent submissions.

By addressing these challenges and issues comprehensively, the college website project aims to enhance operational efficiency, reduce fraudulent activities, and provide a more user-friendly and secure experience for both staff and students within the college environment.

3.2 Planning and Scheduling:

Planning and scheduling are a complicated part of software development. Planning, for our purposes, can be thought of as determining all the small tasks that must be carried out in order to accomplish the goal. Planning also takes into account rules, known as constraints, which control when certain tasks can or cannot happen. Scheduling can be thought of as determining whether adequate resources are available to carry out the plan. Purpose of planning and scheduling is to finish work properly in the given time period.

For our project, planning was started from the month of June. In the planning process, we have discussed the project's purpose, reason of the existence, comparison with the existing work and the new thing to be developed in the beginning part. In the later section, after having a clear vision of the project, we started working on documentation. During the planning part, we have also discussed the software and hardware requirements and the languages required for coding.

After all the planning part is done, according to it scheduling of work was started. In this part, time and resources are distributed according to our requirements. Purpose of scheduling is that the work should be done without any obstacles. For example, overlapping two works at the same time or same resource shouldn't be used at two different places at the same time. To avoid this circumstance, planning and scheduling is done.

Planning and scheduling can be shown with the help of using Gantt Chart. It properly shows when the work starts, how much time is required and who is doing which work. It gives us the overview knowledge of time duration and working structure of the project. It even helps us to know if the work is going in the estimated time duration or not.

Ask Assignments and Responsibilities:

Project Manager (Myself): Overall project oversight, timeline management, and coordination.
Web Developer(s) (Myself): Responsible for developing and implementing website features.
Database Administrator (Myself): Manages the database structure and data **storage. UI/UX Designer (Myself):** Designs the user interface and ensures a user-friendly experience.
Quality Assurance Tester (Myself): Conducts testing and quality assurance procedures.

GANTT'S CHART

Sr. No.	Task	Month	June			July			August			September	
		Date	13	23	30	13	23	30	13	23	30	3	20
1	INTRODUCTION												
	Background	Planned											
		Actual											
	Objectives	Planned											
		Actual											
	Purpose, Scope and Applicability	Planned											
		Actual											
2	SURVEY OF TECHNOLOGY												
	Survey	Planned											
		Actual											
	Fact-Finding	Planned											
		Actual											
	Feasibility Study	Planned											
		Actual											
	Stake holder	Planned											
		Actual											
3	REQUIREMENT AND ANALYSIS												
	Problem Definition	Planned											
		Actual											
	Requirement Specification	Planned											
		Actual											
	Planning and Scheduling	Planned											
		Actual											
	Software and Hardware Requirement	Planned											
		Actual											
	Conceptual Models	Planned											
		Actual											
4	SYSTEM DESIGN												
	Basic Models	Planned											
		Actual											
	Data Design	Planned											
		Actual											
	Data Integrity and constraints	Planned											
		Actual											
	User Interface Design	Planned											
		Actual											
	Security Issues	Planned											

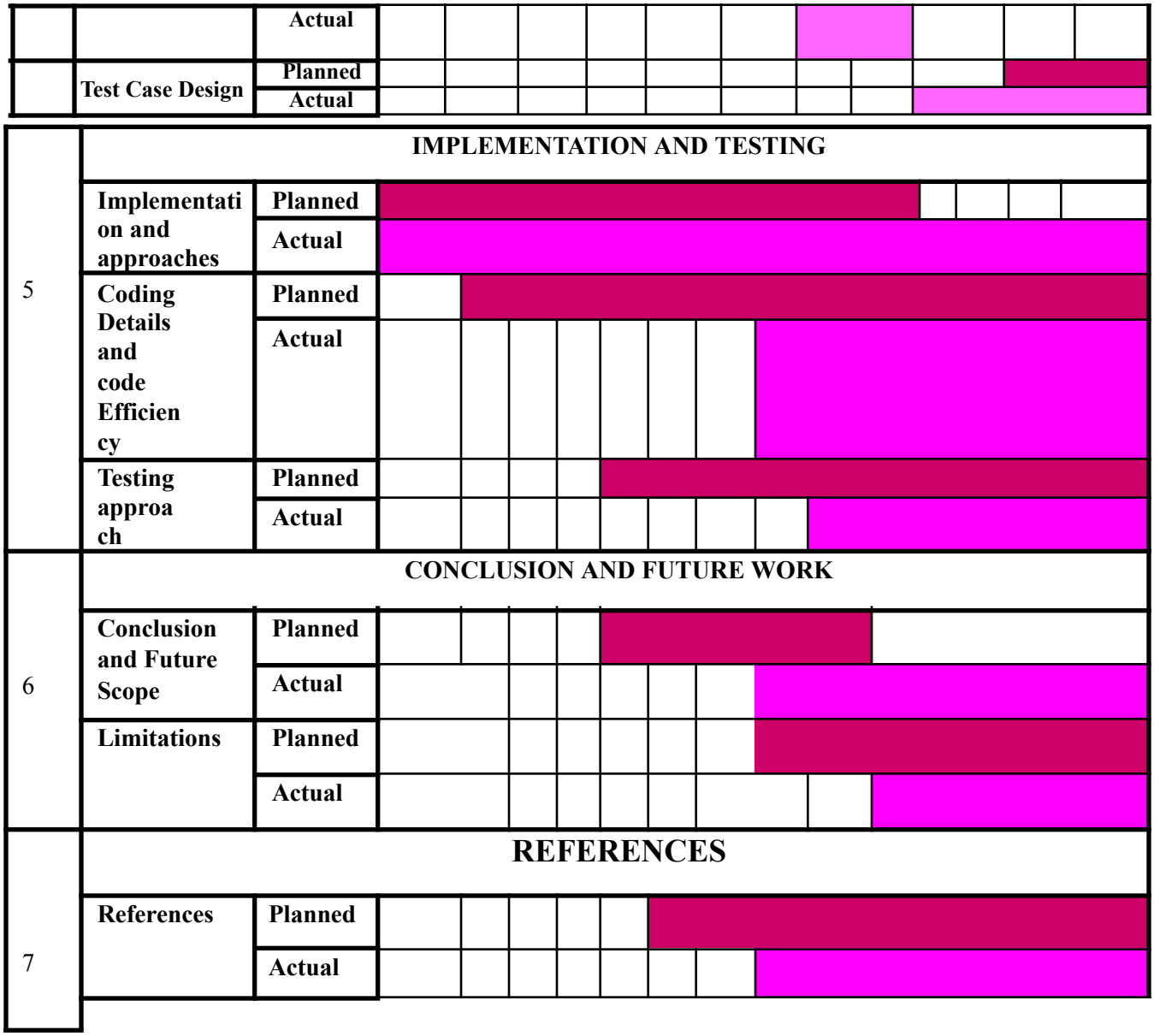


Figure 2: Gantt's Charts

❖ Resource Allocation:

Resource allocation is a critical aspect of project management, ensuring that the right resources are available at the right time to complete the college website project successfully. Here's an overview of resource allocation for the project:

Human Resources:

Project Team: Assign and allocate team members with relevant skills and expertise to specific tasks and phases.
Roles and Responsibilities: Clearly define the roles and responsibilities of each team member, including the project manager, web developers, database administrators, UI/UX designers, quality assurance testers, and IT support. But As this is my final year project so I'm Working alone on this project by fulfilling all the needs.

Time Management:

Project Timeline: Strictly adhere to the project timeline and milestones to efficiently allocate time resources.
Task Prioritization: Prioritize tasks based on their criticality and dependencies to optimize time allocation.

Infrastructure and Technology:

Hardware: Ensure that the necessary hardware resources, such as servers and development machines, are available and properly configured.
Software: Procure and allocate software licenses for development tools, database management, and other required software.

Training and Skill Enhancement:

Training Programs: Allocate time and resources for training team members in new technologies or tools that may be required during the project.
Skill Enhancement: Encourage continuous skill enhancement among team members to keep up with evolving technologies and best practices.

External Resources:

Third-Party Services: If necessary, allocate resources for third-party services or consultants with specific expertise, such as security audits or database optimization.

❖ Risk Management:

Effective risk management is crucial to identify and mitigate potential issues that may affect the success of the college website project. Here's an overview of risk management for the project:

Risk Identification:

Team Input: Encourage team members to identify and report potential risks related to their respective areas of expertise.
Stakeholder Input: For this project I'm including students, staff, and administrators, to identify any concerns or risks they foresee.

❖ Risk Mitigation:

Risk Response Plans: Develop response plans for high-priority risks, including risk avoidance, risk reduction, risk transfer, or risk acceptance.
Contingency Plans: Create contingency plans to address risks that may still materialize despite mitigation efforts.

❖ Monitoring and Reporting:

Regular Monitoring: Continuously monitor the project environment for emerging risks and changes in risk factors.
Reporting: Provide regular risk reports to stakeholders, highlighting the status of identified risks and the effectiveness of mitigation efforts.

❖ Adaptation and Learning:

Lessons Learned: After project completion, conduct a lessons-learned session to gather insights and feedback for future risk management improvements.
Adaptive Approach: Be flexible and ready to adapt risk mitigation strategies as the project progresses and new risks emerge.

3.3 Software & Hardware Requirements:

- **Web Development Tools:**

  Code Editor or Integrated Development Environment (IDE) for coding HTML, CSS, JavaScript, and backend code if applicable (e.g., Visual Studio Code, Visual Studio, Sublime Text).

- **Database Management System:**

  Database management system (MySQL) storing user data, certificates, and other relevant information.

- **Front-End Libraries and Frameworks:**

  Libraries and frameworks (Angular.Js) for building interactive and responsive user interfaces.

- **Payment Gateway Integration:**

  Payment gateway integration tools and APIs (e.g., Google pay, PhonePe, PayPal) For online payment processing.

- **Development and Testing Environments:**

  Local development server (e.g., Apache) for testing and a staging environment for pre-production testing.

- **Deployment Tools:**

  Deployment tools and platforms (e.g., AWS) for hosting and deploying the website.

- **Development Machines:**

  High-performance computers or laptops for web development with sufficient RAM, CPU, and storage capacity.

- **Hardware Requirement**
  - Server Side
    - PROCESSOR: -Server Environment Capable H/W

      RAM: -2GB RAM

      DISK SPACE: -As per the size of requirements DBMS

  - Client Side
    - PROCESSOR: -Desktop
      RAM: -2GB RAM
      DISK SPACE: -As per the size of requirements DBMS
- **Software Requirement**
  1. Server Side
     - Visual Studio Code
  2. Client Side

     - Operating System (Windows/MacOS/Linux)
       Web Browser (Google Chrome / Mozilla Firefox / Apple Safari / Microsoft Edge)

3.3) Conceptual Models:

A conceptual model is a representation of a system, made of the composition of concepts which are used to help people know, understand, or simulate a subject the model represents. It is also a set of concepts. Conceptual modeling is a crucial software development activity.

Conceptual models give you background details about the project work. It tells you about the resources, criteria, requirements, involvements of people, process, etc.

3.3.1) Entity Relationship Diagrams:

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity- Relationship diagram which is used to visually represent data objects. Since Chen wrote his paper the model has been extended and today it is commonly used for database design for the database designer, the utility of the ER model is:

A. It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.
B. It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.

In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.

Connectivity and Cardinality

The basic types of connectivity for relations are: one-to-one, one-to-many, and many-to-many.

C. A one-to-one (1:1) relationship is when at most one instance of an entity A is associated with one instance of entity B.

For example, "employees in the company are each assigned their own office. For each employee there exists a unique office and for each office there exists a unique employee.

D. A one-to-many (1:N) relationship is when for one instance of entity A, there are zero, one, or many instances of entity B, but for one instance of entity B, there is only one instance of entity A.

An example of a 1:N relationships is a department has many employees each employee is assigned to one department

E. A *many-to-many* (M:N) relationship, sometimes called non-specific, is when for one instance of entity A, there are zero, one, or many instances of entity B and for one instance of entity B there are zero, one, or many instances of entity A. The connectivity of a relationship describes the mapping of associated ER Notation

There is no standard for representing data objects in ER diagrams. Each modeling methodology uses its own notation. The original notation used by Chen is widely used in academic texts and journals but rarely seen in either CASE tools or publications by non-academics. Today, there are a number of notations used; among the more common are Bachman, crow's foot, and IDEFIX.

All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection. The notation used in this document is from Martin. The symbols used for the basic ER constructs are:

- Entities are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.
- Relationships are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs
- Attributes when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.
- Cardinality of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.

A. Student Faculty Relationship while Participating in event the ER diagram for this relationship will look like this:

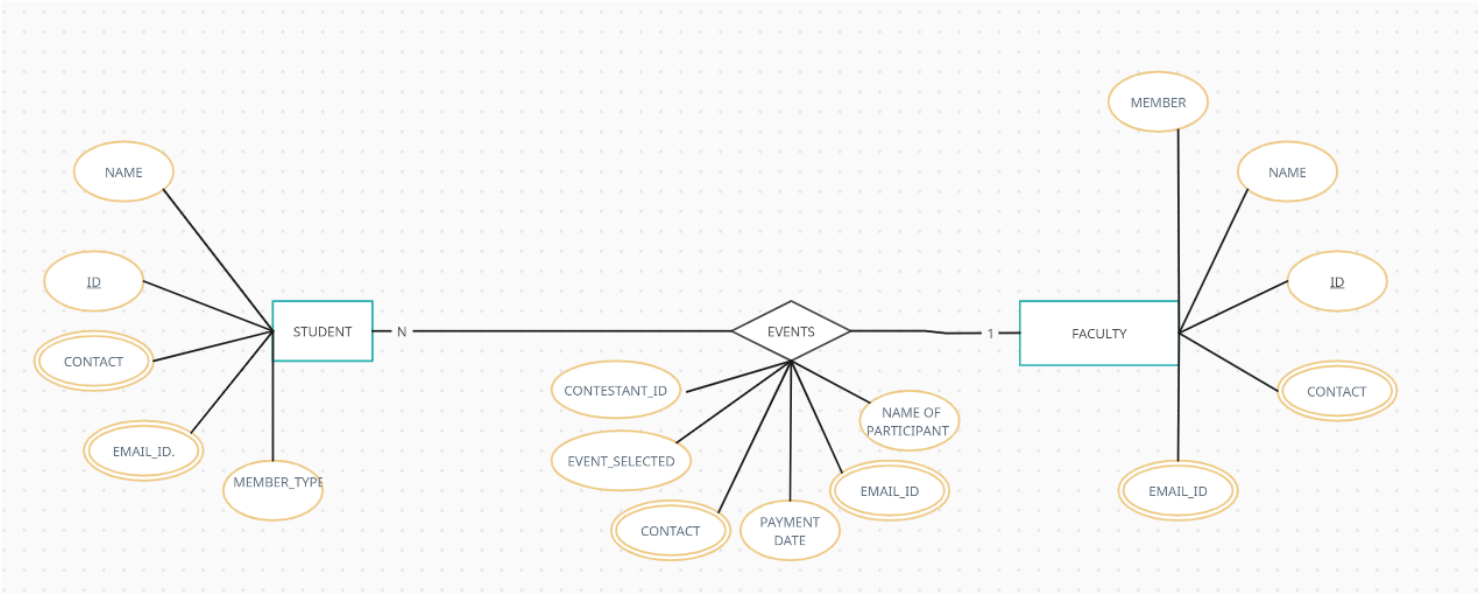


Figure 3: Student,Faculty Entity Relationship Diagram

This is an Entity-Relationship (ER) diagram for college management of events. It has three main entities: STUDENT, EVENTS, and FACULTY.

1. **STUDENT**: This entity has five attributes: NAME, ID, CONTACT, EMAIL_ID, MEMBER_TYPE.
2. **EVENTS**: This entity has six attributes: CONTESTANT_ID, EVENT_SELECTED, NAME OF PARTICIPANT, CONTACT, EMAIL_ID, PAYMENT DATE.
3. **FACULTY**: This entity has four attributes: MEMBER ID, NAME ID CONTACT EMAIL_ID.

The relationships between these entities are:

- **STUDENT and EVENTS**: They are connected with a relationship labeled as MEMBER_TYPE. This could mean that a student can participate in multiple events based on their member type.
- **EVENTS and FACULTY**: They are also linked together through another relationship labeled as NAME OF PARTICIPANT. This could mean that a faculty member is associated with each event participant.

This ER diagram is used to represent the logical structure of the database. Each entity represents a real-world object, and the relationships between them represent how these objects interact with each other. The attributes are the properties or characteristics of these entities.

B. Relationship between the Students, Certificate Uploading page and Teachers, Let's see the ER Diagram for these Entities:

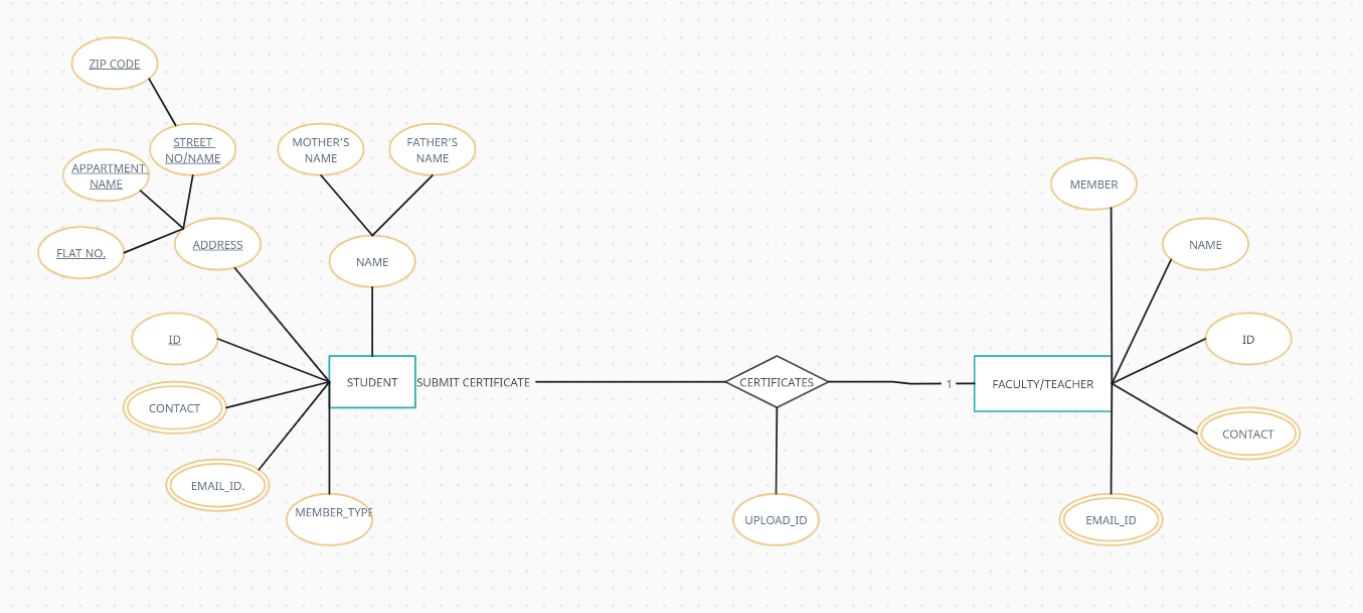


Figure 4: Entity Relationship between Student and Faculty

This is an ER diagram for a student and faculty. It has two main entities: STUDENT and FACULTY

1. **STUDENT**: This entity has attributes like ID, CONTACT, EMAIL_ID. It also has composite attributes like NAME (which further divides into MOTHER'S NAME and FATHER'S NAME) and ADDRESS (which breaks down into STREET NO./NAME, APARTMENT/HOUSE NAME FLAT NO., ZIP CODE).
2. **FACULTY/TEACHER**: This entity is associated with MEMBER having attributes of NAME,ID ,CONTACT ,EMAIL_ID

The relationships between these entities are:

- **STUDENT and CERTIFICATES**: They are connected with a relationship labeled as SUBMIT CERTIFICATE. This could mean that a student can submit multiple certificates.
- **FACULTY/TEACHER and CERTIFICATES**: They are also linked together through the same relationship labeled as SUBMIT CERTIFICATE. The CERTIFICATES entity has an attribute UPLOAD_ID which could be the unique identifier for each certificate uploaded.

C. Relationship between the Teacher, Time-Table page and the Students, Let's see the ER Diagram for these Entities:

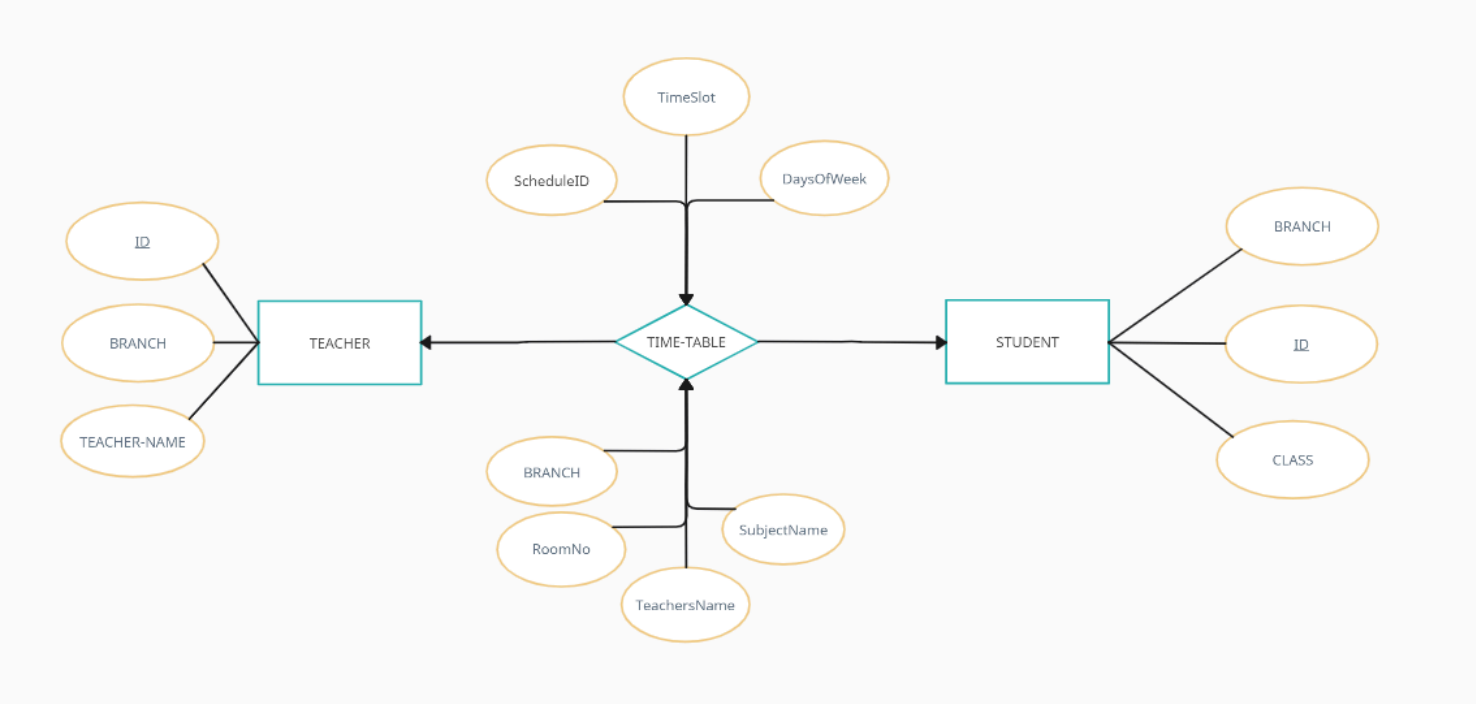


Figure 5: Entity Relationship between Teacher and Student

This is an Entity-Relationship (ER) diagram for the College database system. It has three main entities:

1. **TEACHER**: This entity has attributes like ID, BRANCH, and TEACHER-NAME.
2. **TIME-TABLE**: This entity is connected to both the TEACHER and STUDENT entities and contains attributes ScheduleID, TimeSlot, DaysOfWeek, BRANCH, RoomNo, SubjectName, and Teacher'sName.
3. **STUDENT**: This entity has attributes BRANCH, ID, and CLASS.

The relationships between these entities are:

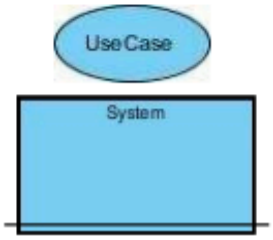
- **TEACHER and TIME-TABLE**: They are connected with a relationship. This could mean that a teacher can have multiple time-tables.
- **STUDENT and TIME-TABLE**: They are also linked together through a relationship. This could mean that a student can have multiple time-tables.

3.3.2) Use Case Diagram:

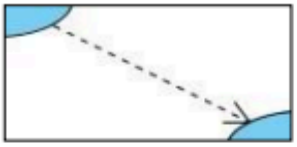
A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use case in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

Symbol used for use case are as follows

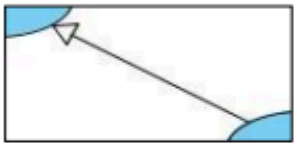
Represents users of system.
Represents a user goal.



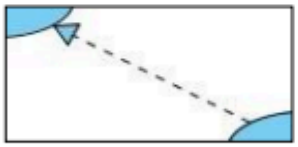
Use case are placed in the system.



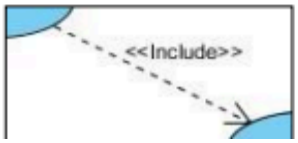
Represents the connection between Entity.



Represents a model element relies on another model element.

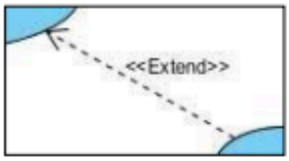


Represent inheritance relationship between model elements of same type.



Relationship between a specification and its implementation.

A use case contains the behavior defined in another use case.



Extend is intended to be used when there is some additional behavior that should be added

Use case diagrams model behavior within a system and helps the developers understand of what the user requires. The stick man represents what's called an actor. Use case diagram can be useful for getting an overall view of the system and clarifying that can do and more importantly what they can't do. Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

- The purpose is to show the interactions between the use case and actor.
- To represent the system requirements from user's perspective.
- An actor could be the end-user of the system or an external system.

A Use case is a description of set of sequence of actions graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship. It is an association between the use cases and actors. An actor represents a real-world object.

A. Student Use case diagram:

The Student actor interacts with the College Website. Each of these actions represents a use case that describes a specific functionality or task that the Student can perform within the system.

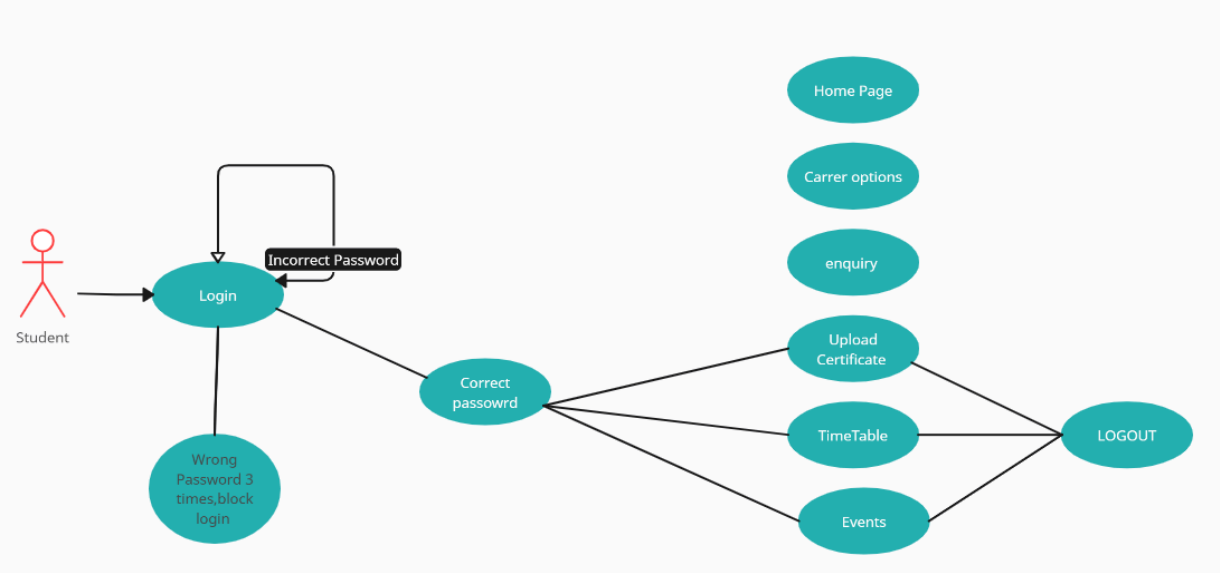


Figure 6: Student Use Case Diagram

This is a Use Case Diagram for a Student interacting with an online system:

1. **Student**: The primary actor in this system who interacts with the system to perform various tasks.
2. **Login**: The student attempts to login to the system. If the password is incorrect, they are notified. If the password is incorrect three times, their login is blocked. If the password is correct, they gain access to the system.
3. **System Interactions**: Once logged in, the student can perform several tasks:
    - **View Home Page**: The student can view the home page of the system.
    - **Explore Career Options**: The student can explore various career options available.
    - **Make Enquiries**: The student can make enquiries about various topics.
    - **Upload Certificates**: The student can upload their certificates to the system.
    - **Check Timetable**: The student can check their timetable.
    - **Participate in Events**: The student can participate in various events organized by the system.
    - **Logout**: The student can logout from the system.

B. Faculty Use Case Diagram:

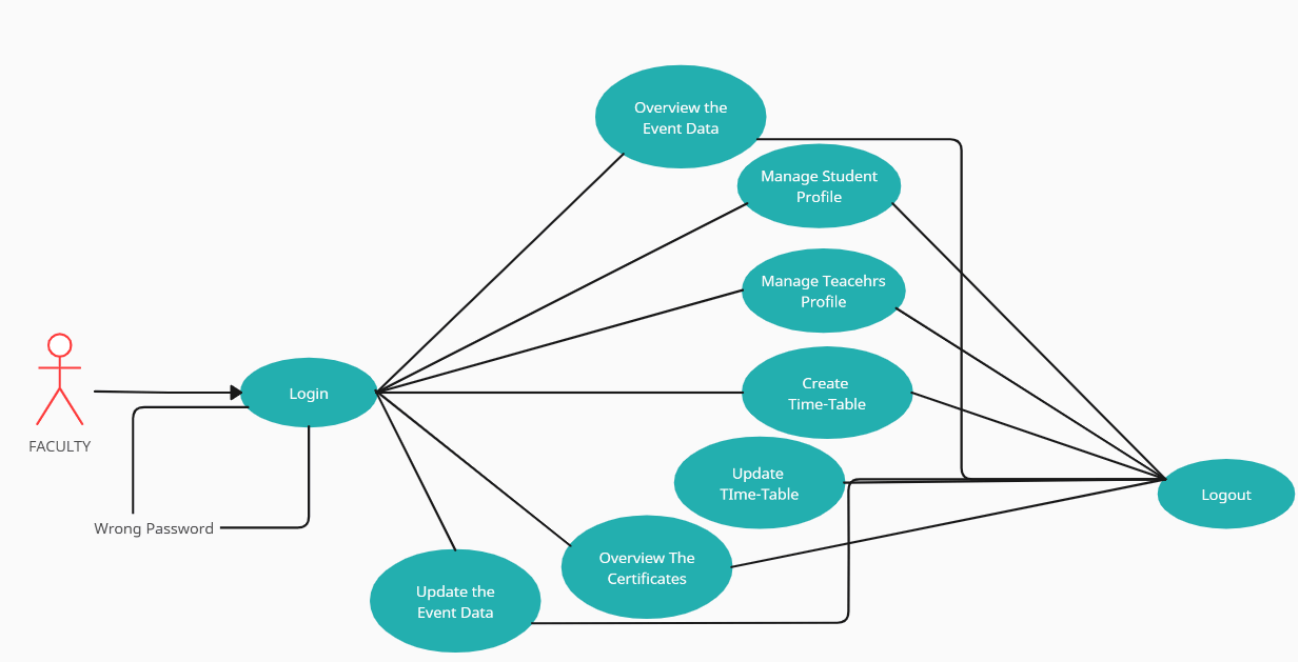


Figure 7: Faculty Use Case Diagram

This is a Use Case Diagram for a Faculty member interacting with an online system:

1. **Faculty**: The primary actor in this system who interacts with the system to perform various tasks.
2. **Login**: The faculty member attempts to login to the system. If the password is wrong, they are not allowed to proceed. Upon successful login, the faculty member gains access to the system.
3. **System Interactions**: Once logged in, the faculty member can perform several tasks:
    - **Overview the Event Data**: The faculty member can view the data related to various events.
    - **Manage Student Profile**: The faculty member can manage the profiles of the students.
    - **Manage Teachers Profile**: The faculty member can manage the profiles of other teachers.
    - **Create Time-Table**: The faculty member can create time-tables.
    - **Update Time Table**: The faculty member can update the time-tables.
    - **Overview The Certificates**: The faculty member can view the certificates uploaded by the students.
    - **Logout**: The faculty member can logout from the system.

C. Use case Diagram for Teacher:

The Use Case Diagram for a Teacher in the college website project illustrates the key interactions and functionalities available to teachers. Teachers can View Timetable to access their class schedule.

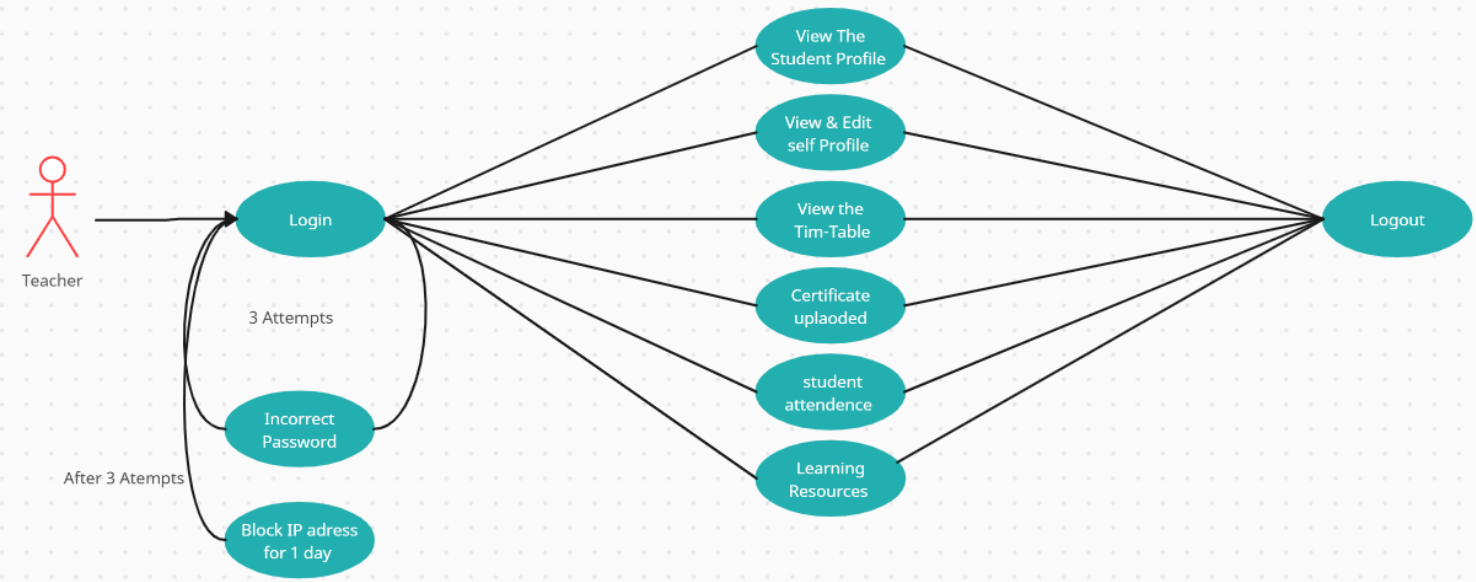


Figure 8: Teacher Use Case Diagram

This is a Use Case Diagram for a Faculty member interacting with an online system:

1. **Faculty**: The primary actor in this system who interacts with the system to perform various tasks.
2. **Login**: The faculty member attempts to login to the system. If the password is wrong, they are not allowed to proceed. Upon successful login, the faculty member gains access to the system.
3. **System Interactions**: Once logged in, the faculty member can perform several tasks:
    - **Overview the Event Data**: The faculty member can view the data related to various events.
    - **Manage Student Profile**: The faculty member can manage the profiles of the students.
    - **Manage Teachers Profile**: The faculty member can manage the profiles of other teachers.
    - **Create Time-Table**: The faculty member can create time-tables.
    - **Update Time Table**: The faculty member can update the time-tables.
    - **Overview The Certificates**: The faculty member can view the certificates uploaded by the students.
    - **Logout**: The faculty member can logout from the system.

D. Authorized committee member:

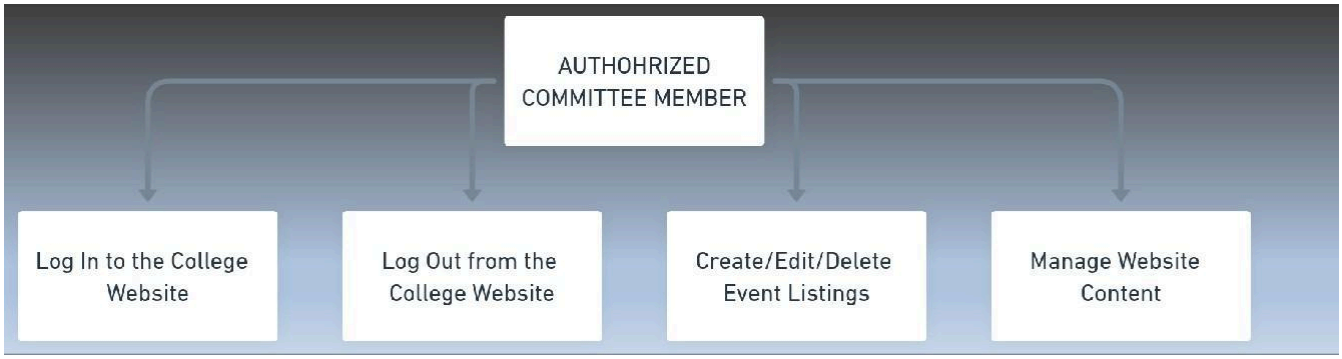


Figure 9: Authorized committee member

E. Other Actors (External Users):

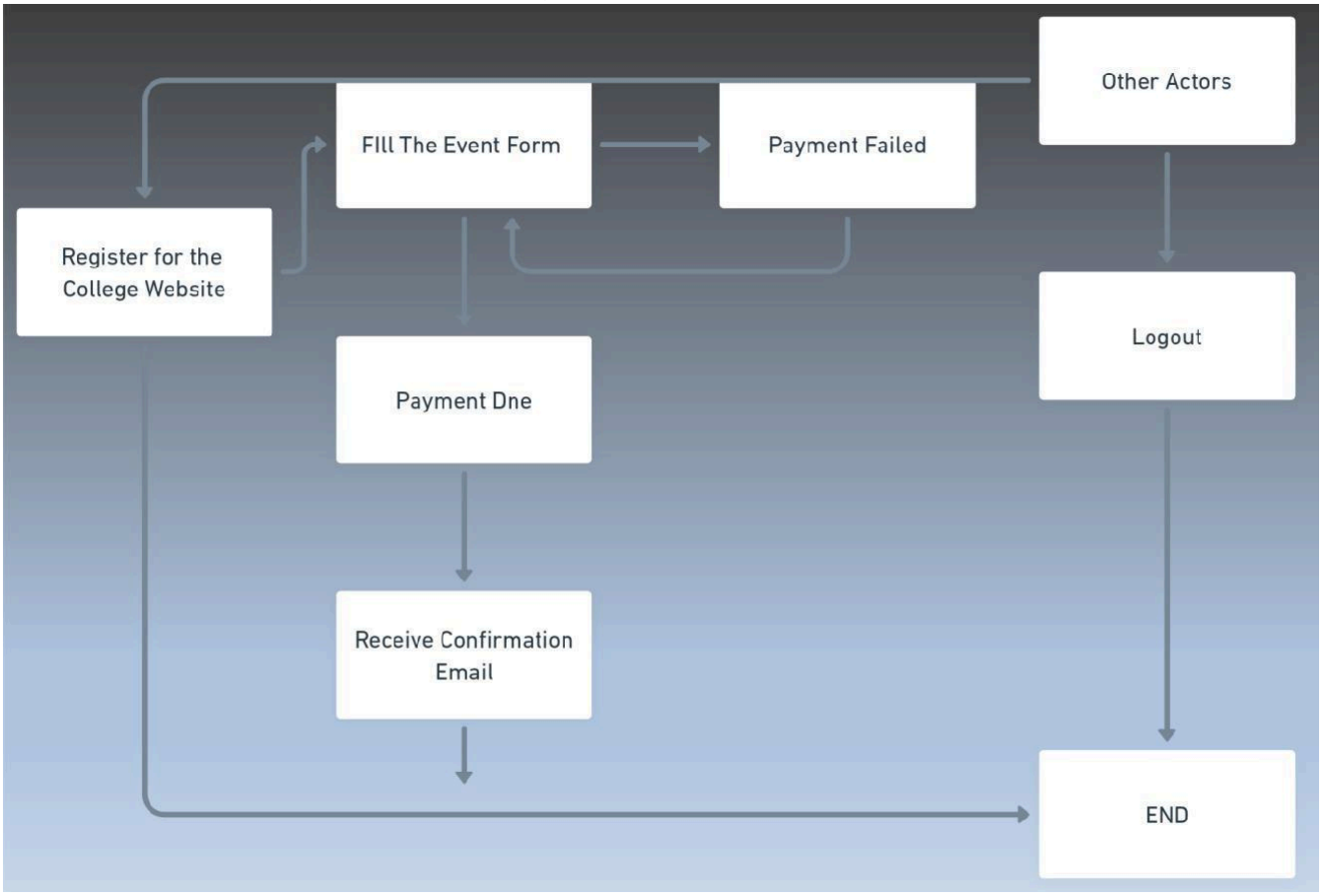


Figure 10: Other Actors Use Case Diagram

3.3.3) Class Diagram:

In software engineering, a class diagram in the (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

- The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom compartment contains the operations the class can execute. They are also left- aligned and the first letter is lowercase.

Basic Class Diagram Symbols and Notations Classes

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition (left- aligned, not bolded, and lowercase), and write operations into the third.

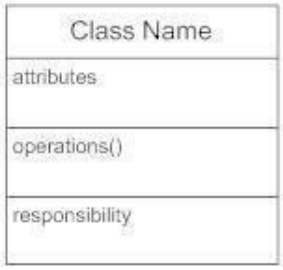


Figure 11: Class Diagram

Active Classes

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.

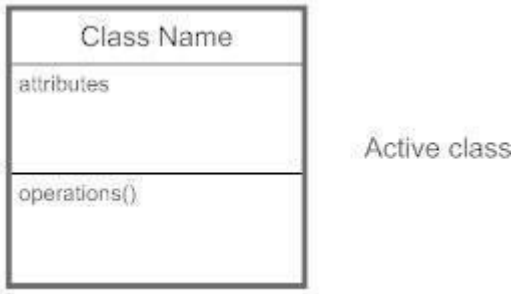


Figure 12: Active Classes

Figure 13: Class Visibility

Visibility

Use visibility markers to signify who can access the information contained within a class. Private visibility, denoted with a - sign, hides information from anything outside the class partition. Public visibility, denoted with a + sign, allows all other classes to view the marked information. Protected visibility, denoted with a # sign, allows child classes to access information they inherited from a parent class.

Associations

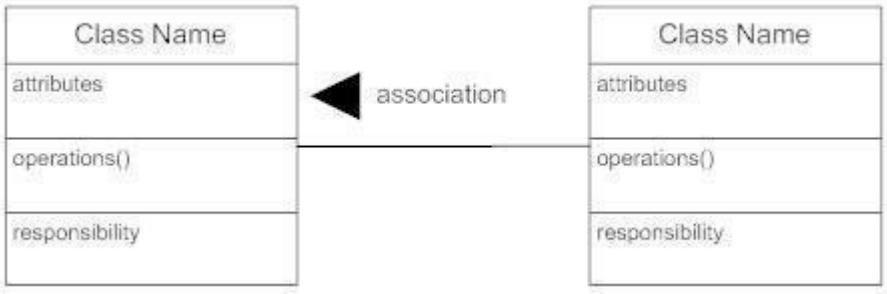
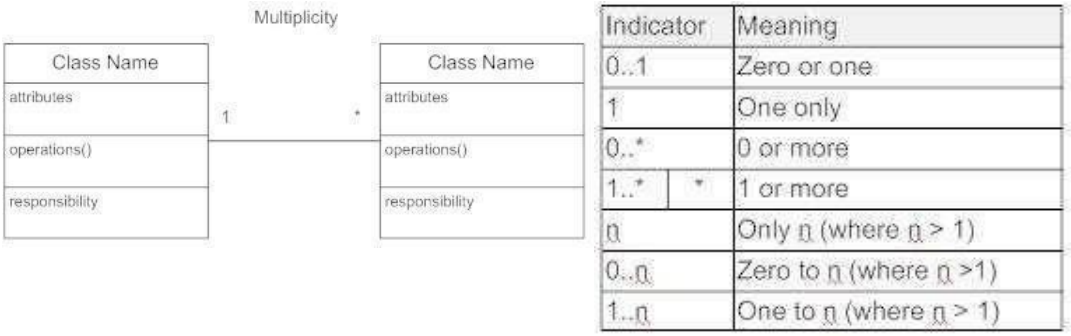


Figure 14: Class Associations

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.

Multiplicity (Cardinality)

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for just one company.



Indicator	Meaning
0..1	Zero or one
1	One only
0..*	0 or more
1..*	1 or more
n	Only n (where n > 1)
0..n	Zero to n (where n >1)
1..n	One to n (where n > 1)

Figure 15: Class Multiplicity(cardinality)

Constraint

Place constraints inside curly braces {}.

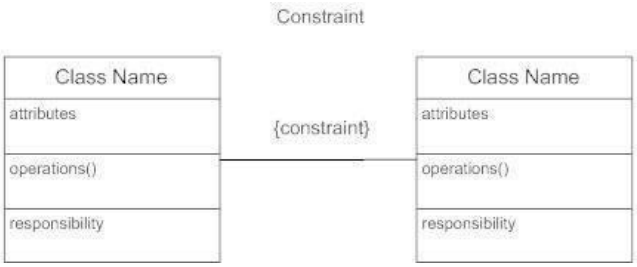


Figure 16: Class Constraints

Composition and Aggregation

Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate composition with a filled diamond. Use a hollow diamond to represent a simple aggregation relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond ends in both composition and aggregation relationships point toward the "whole" class (i.e., the aggregation).

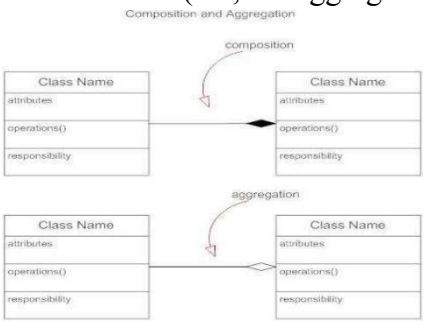


Figure 17: Class Composition and Aggregation

Generalization

Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So, the class Honda would have a generalization relationship with the class car.

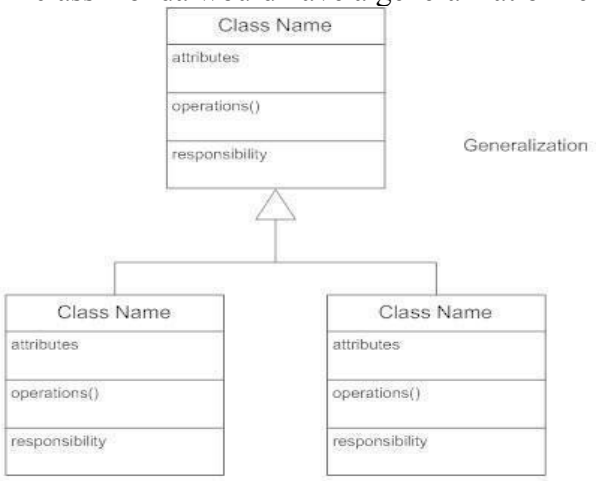
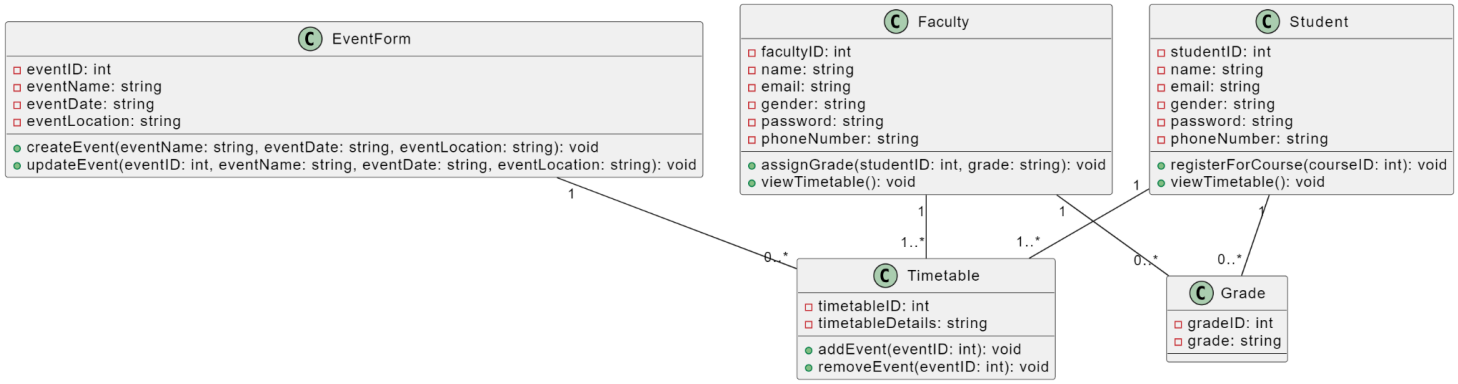


Figure 18 & 19: Class Generalization | Class Diagram



3.3.4) SEQUENCE DIAGRAM:

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Basic Sequence Diagram Notations Class

Roles or Participants

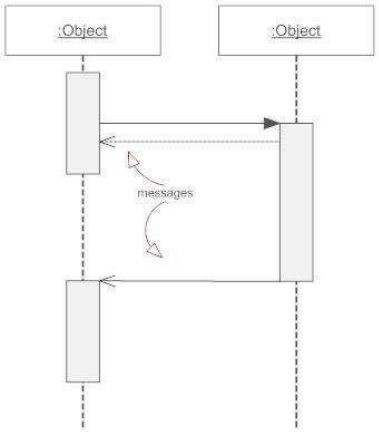
Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Activation or Execution Occurrence

Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.

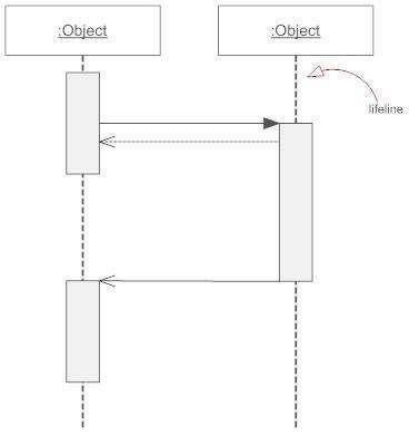
Messages

Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks. For message types, see below.



Lifelines

Lifelines are vertical dashed lines that indicate the object's presence over time.



Destroying Objects

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, you can place an X at the end of its lifeline to denote a destruction occurrence.

Loops

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [].

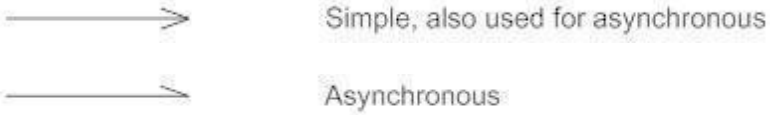
Types of Messages in Sequence Diagrams Synchronous

Message

A synchronous message requires a response before the interaction can continue. It's usually drawn using a line with a solid arrowhead pointing from one object to another.

Asynchronous Message

Asynchronous messages don't need a reply for interaction to continue. Like synchronous messages, they are drawn with an arrow connecting two lifelines; however, the arrowhead is usually open and there's no return message depicted.

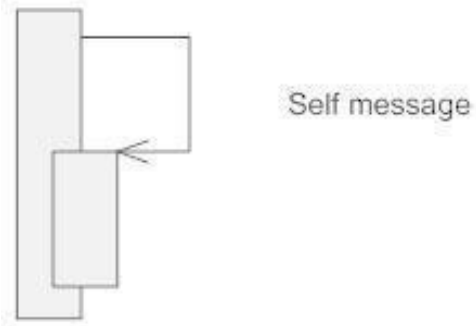


Reply or Return Message

A reply message is drawn with a dotted line and an open arrowhead pointing back to the original lifeline.

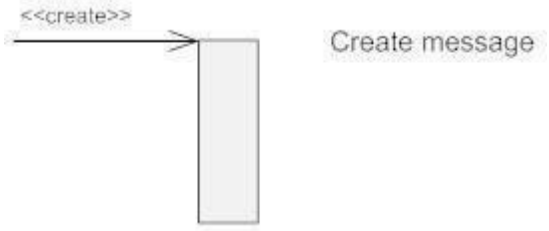
Self-Message

A message an object sends to itself, usually shown as a U-shaped arrow pointing back to itself



Create Message

This is a message that creates a new object. Similar to a return message, it's depicted with a dashed line and an open arrowhead that points to the rectangle representing the object created.



Delete Message

This is a message that destroys an object. It can be shown by an arrow with an x at the end.

Found Message

A message sent from an unknown recipient, shown by an arrow from an endpoint to a lifeline.

Lost Message

A message sent to an unknown recipient. It's shown by an arrow going from a lifeline to an endpoint, a filled circle or an x.

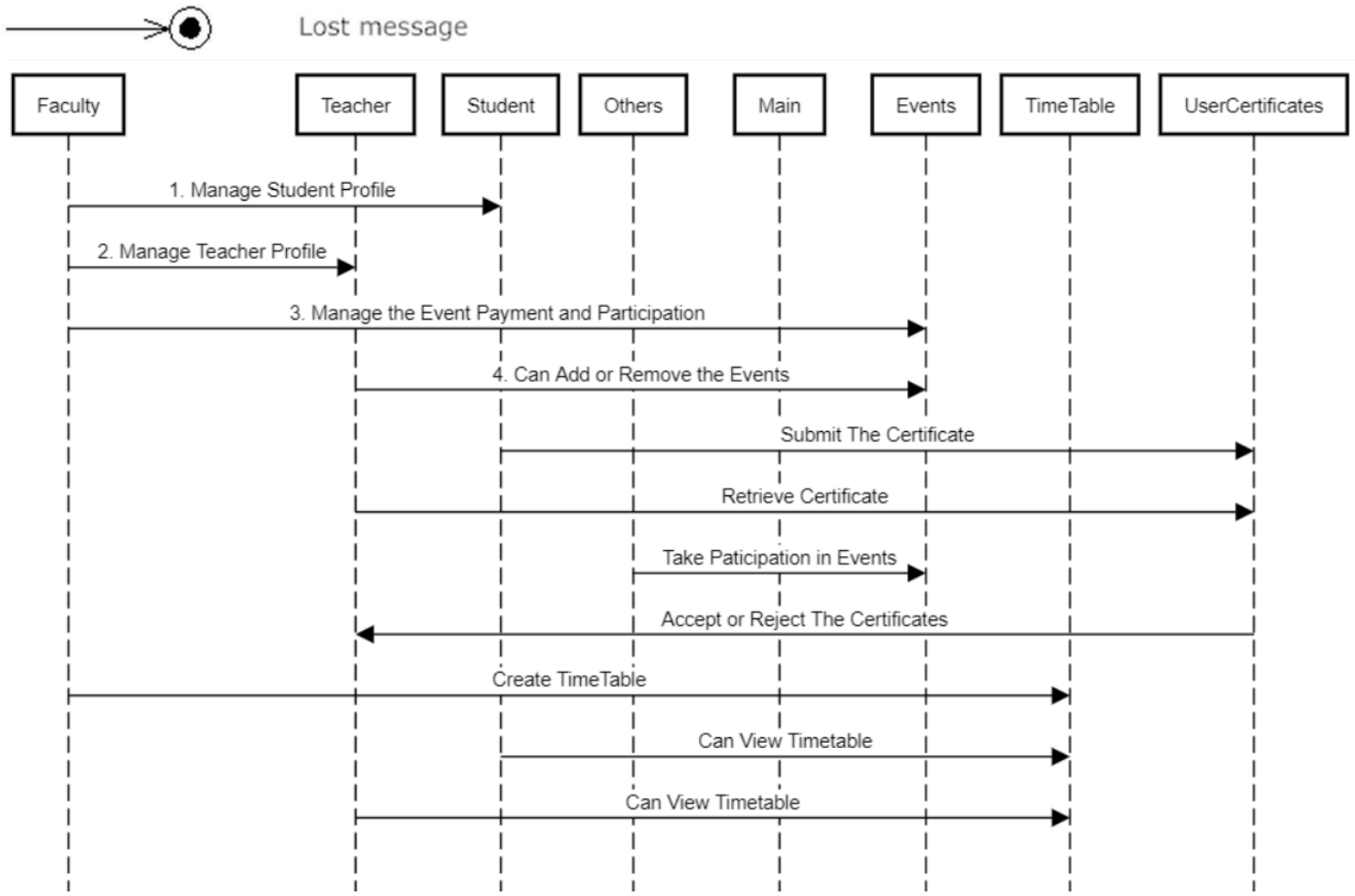


Figure 20: Sequence Diagram

3.3.5) PACKAGE DIAGRAM:

Package diagram is used to simplify complex class diagrams, you can group classes into packages.A package is a collection of logically relatedUMLelements.Thediagrambelow is a business model in which the classes are grouped into packages: Packages appear as rectangles with small tabs at the top.

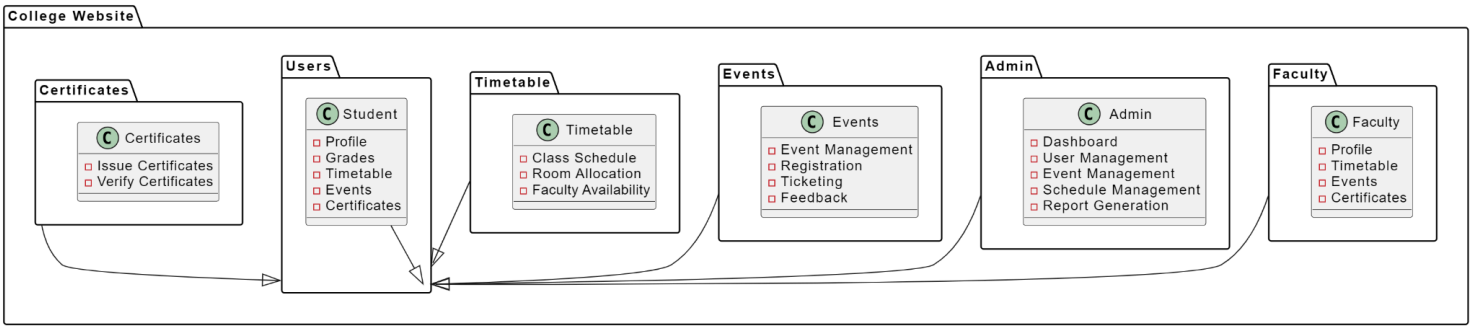


Figure 21: Package diagrams

3.3.6) Deployment Diagram:

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.

Deployment diagrams are typically used to visualize the physical hardware and software of a system.

Using it you can understand how the system will be physically deployed on the hardware.

Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system.

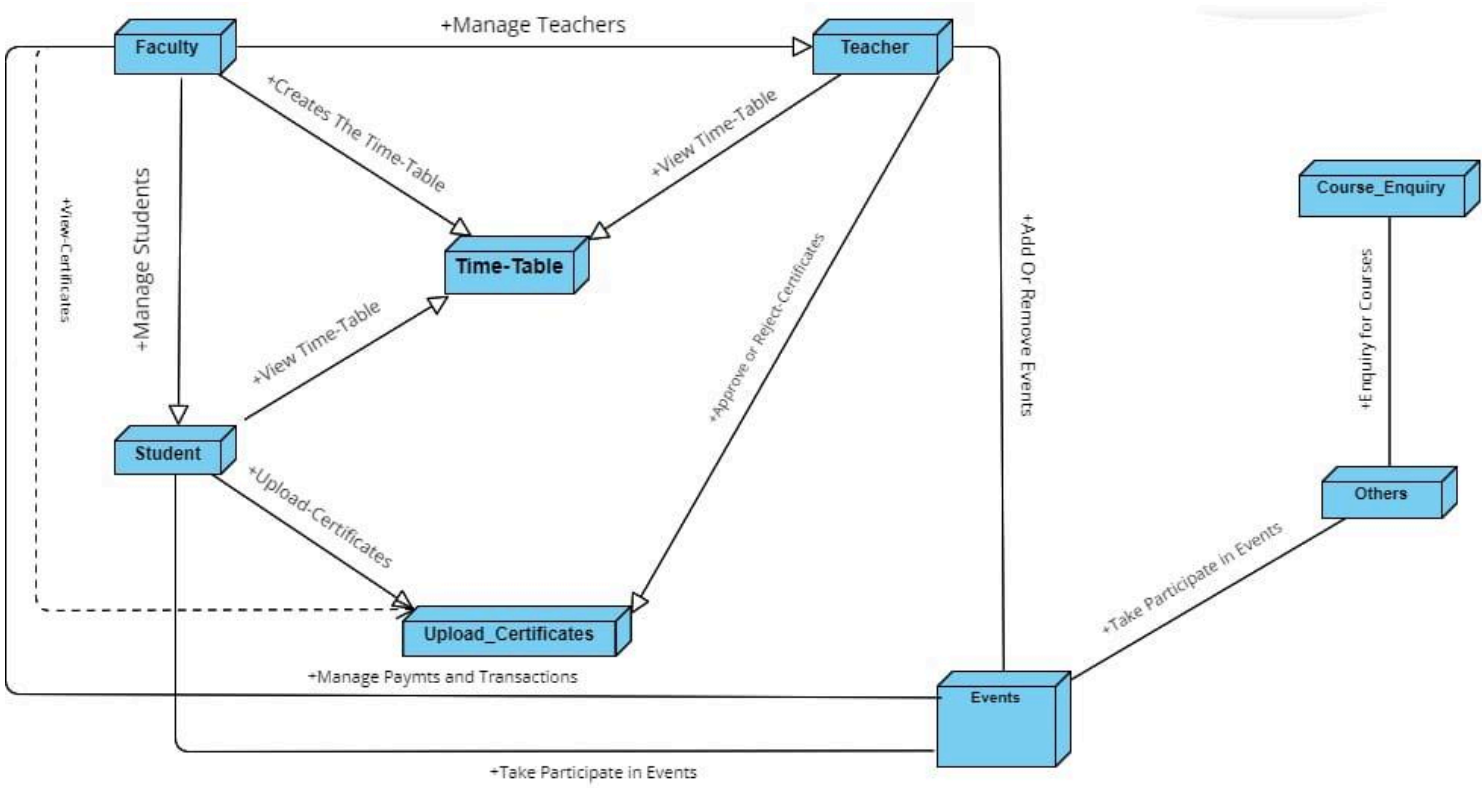


Figure 22: Deployment diagram

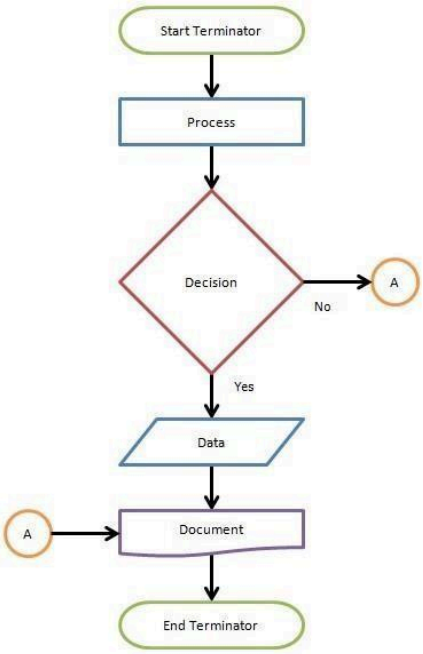
3.3.7) System Flowchart Diagram:

A flow chart is a graphical or symbolic representation of a process. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart
symbols are linked together with arrows showing the process flow direction.

Different flowchart symbols have different meanings. The most common flowchart symbols are:

- Terminator: An oval flow chart shape indicating the start or end of the process.
- Process: A rectangular flow chart shape indicating a normal process flow step.
- Decision: A diamond flowchart shape indicates a branch in the process flow.
- Connector: A small, labeled, circular flow chart shape used to indicate a jump in the process flow. (Shown as the circle with the letter "A", below.)
- Data: A parallelogram that indicates data input or output (I/O) for a process.
- Document: Used to indicate a document or report (see image in sample flow chart below).

A simple flow chart showing the symbols described above can be seen below:



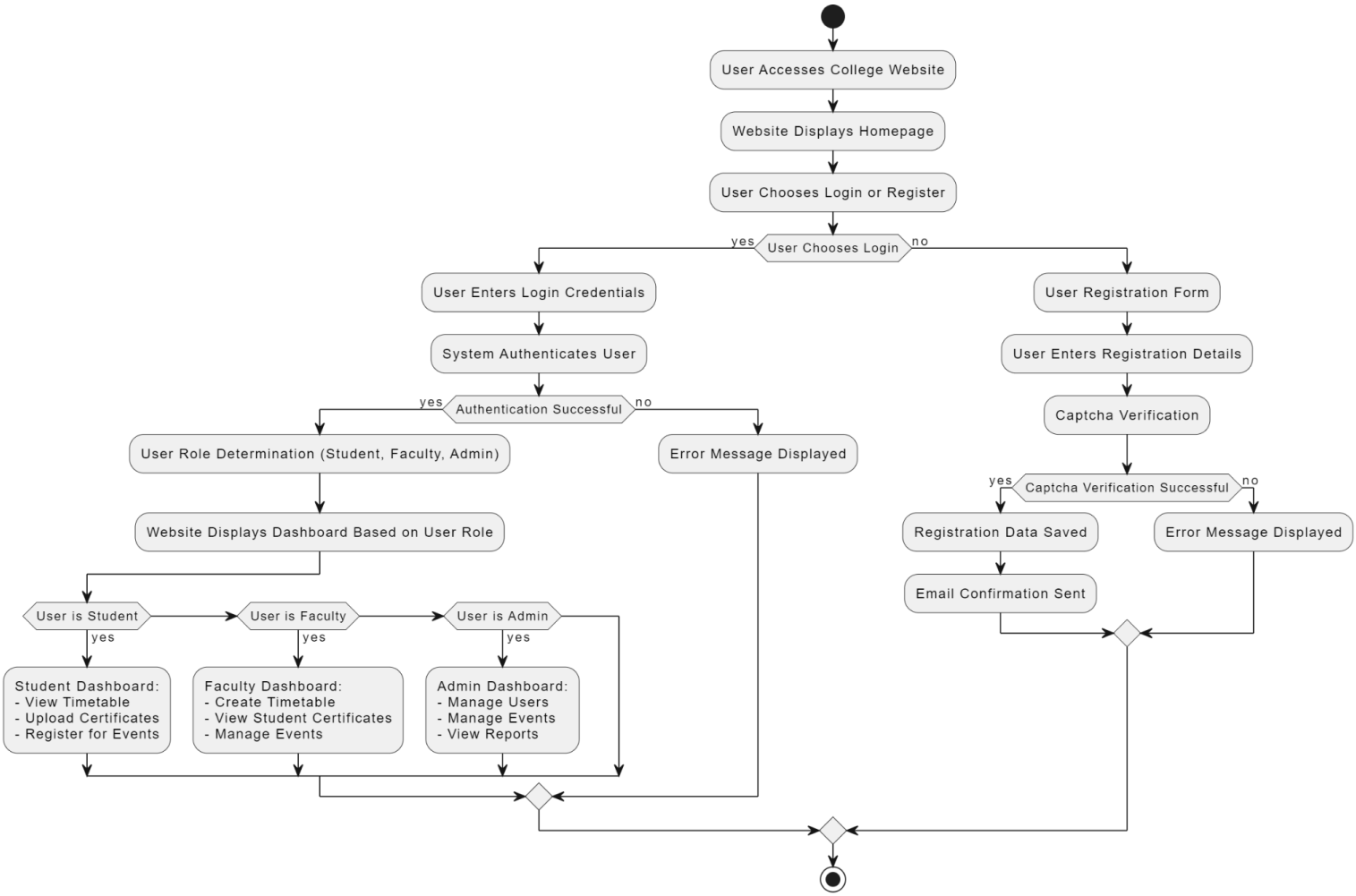


Figure 22: System Flowchart Diagram

The system flowchart outlines the sequence of interactions between users and the college website. It begins with users accessing the website and choosing to either log in with existing credentials or register as new users. If logging in, the system authenticates the user's credentials and determines their role (student, faculty, admin). Based on the role, the appropriate dashboard is displayed, allowing users to perform specific actions tailored to their role. If registration is chosen, users fill out the registration form, undergo captcha verification for security, and upon successful validation, their registration data is saved, and an email confirmation is sent. Any errors in authentication or registration trigger appropriate error messages for user feedback.

CHAPTER 4: SYSTEM DESIGN

System design is a critical phase in the development of any software application, including our college website system. This chapter focuses on designing the system architecture and features to meet the specified requirements. It encompasses various aspects, including defining basic modules, designing data structures, ensuring data integrity and constraints, creating user interfaces, addressing security concerns, and developing test cases. Each of these elements plays a crucial role in shaping the overall functionality, usability, and security of the system. This chapter aims to provide a comprehensive overview of the system design process and its importance in creating a robust and user-friendly college website system.

Objectives of System Design:

1. **Practicality:** We need a system that should be targeting the set of audiences(users) corresponding to which they are designing.
2. **Accuracy:** Above system design should be designed in such a way it fulfills nearly all requirements around which it is designed be it functional or nonfunctional requirements.
3. **Completeness:** System design should meet all user requirements
4. **Efficient:** The system design should be such that it should not overuse surpassing the cost of resources nor under use as it will by now we know will result in low throughput (output) and less response time(latency).
5. **Reliability:** The system designed should be in proximity to a failure-free environment for a certain period of time.
6. **Optimization:** Time and space are just likely what we do for code chunks for individual components to work in a system.
7. **Scalable(flexibility):** System design should be adaptable with time as per different user needs of customers which we know will keep on changing on time. The best example here is the well-known firm: Nokia. It is the most important aspect while designing systems and is the result of why 1 out of 100 startups succeed over the long run.

4.1) Basic Modules:

User Management Module:

- User Registration: Allows new users to create accounts by providing necessary information such as name, email, and password.
- User Login: Provides a secure mechanism for users to authenticate themselves and access their accounts.
- Profile Management: Enables users to update their profile information, change passwords, and manage account settings.
- Role Management: Manages different user roles (e.g., student, faculty, admin) and their permissions within the system.

Dashboard Module:

- Role-based Dashboards: Provides personalized dashboards based on the user's role, displaying relevant information and actions.
- Timetable Display: Shows the user's timetable, highlighting current classes and events.
- Event Information: Displays upcoming events and allows users to register for them.
- Certificate Upload: Allows students to upload certificates and faculty to view and verify them.

Timetable Management Module:

- Timetable Creation: Enables faculty to create and manage timetables by adding, editing, and deleting class schedules.
- Timetable Viewing: Allows students and faculty to view their respective timetables, including class timings and locations.

Event Management Module:

- Event Creation: Allows admins or authorized users to create new events, providing details such as event name, date, time, and venue.
- Event Registration: Enables users to register for events through the website, capturing necessary information for attendance tracking.
- Event Promotion: Provides features for promoting events, such as displaying event details on the website homepage and sending notifications to users.

Certificate Management Module:

- Certificate Upload: Allows students to upload certificates for completed courses, specifying details such as course name, completion date, and certificate file.
- Certificate Verification: Enables faculty or admins to view and verify uploaded certificates, ensuring their authenticity.

Reporting Module:

- Event Attendance Reports: Tracks attendance for various events and provides attendance reports for analysis.
- Certificate Verification Reports: Generates reports on certificate verification activities, showing verified and unverified certificates.

Notification Module:

- Email Notifications: Sends email notifications to users for important updates, such as event reminders, account notifications, and system announcements.
- In-App Notifications: Displays notifications within the website interface to alert users about new messages, event registrations, or other relevant information.

Settings Module:

- Email Preferences: Allows users to manage their email notification preferences, choosing which types of notifications to receive.
- Notification Settings: Enables users to customize their in-app notification settings, such as notification sounds and display options.

4.2) Data Design and Data integrity and Constraints:

Data Design encompasses the process of defining the structure and organization of data within a database system. It involves designing tables, specifying columns and their data types, and establishing relationships between tables to represent the logical structure of the data. The goal of data design is to create a database schema that efficiently stores and retrieves data while ensuring data integrity and consistency.

In a relational database, data is organized into tables, where each table represents a specific entity or concept. For example, in a college database, you might have tables for students, courses, and instructors. Each table consists of rows and columns, where rows represent individual records or instances of the entity, and columns represent attributes or properties of the entity.

Data Integrity Constraints are rules or conditions that enforce the accuracy and consistency of data stored in a database. These constraints are applied to ensure that data remains valid and meaningful throughout its lifecycle. Some common types of data integrity constraints include:

- Entity Integrity: Ensures that each row in a table is uniquely identified by a primary key, preventing duplicate records.
- Referential Integrity: Ensures that relationships between tables are maintained by enforcing foreign key constraints, ensuring that values in a foreign key column match values in a primary key column of another table.
- Domain Integrity: Defines the valid range of values for a column, preventing invalid data from being inserted into the database.
- User-Defined Integrity: Enforces additional business rules and constraints specific to the application, ensuring that data meets specific requirements.

By implementing these data integrity constraints, you can maintain the quality and consistency of data in the database, ensuring that it remains accurate and reliable for use in various applications and analytical tasks.

➢ **Main Table:**

ATTRIBUTES	DATATYPE	CONSTRAINT
ID	INT	PRIMARY KEY
Name	NVARCHAR(30)	NOTNULL
Address	NVARCHAR(255)	NOTNULL
Contact	INT	UNIQUE
EmailID	NVARCHAR (30)	UNIQUE
Password	NVARCHAR (20)	NOTNULL

➢ **EventList Table: -**

ATTRIBUTES	DATATYPE	CONSTRAINT
EventID	INT	PRIMARY KEY
EventNAME	NVARCHAR (30)	NOTNULL
EntryFees	INT	NOTNULL
FirstPrize	INT	NOTNULL
SecondPrize	INT	NOTNULL
ThirdPrize	INT	NOTNULL

➢ **Schedule Table:**

ATTRIBUTES	DATATYPE	CONSTRAINT
ScheduleID	INT	PRIMARY KEY
TimeSlot	NVARCHAR (30)	NOTNULL
DaysOfWeek	NVARCHAR (30)	NOTNULL
Branch	NVARCHAR (30)	Foreign Key(Reference Teachers(Branch)
TeacherName	NVARCHAR (50)	Foreign Key(Reference Teachers(TeachersName)
SubjectName	NVARCHAR (50)	Foreign Key(Reference Subjects(SubjectName)
RoomNo	NVARCHAR (50)	Foreign Key(Reference Rooms(RoomNo)

Subject Table:

ATTRIBUTES	DATATYPE	CONSTRAINT
SubjectID	INT	Primary Key
SubjectName	NVARCHAR (50)	NOTNULL

➢ **Room Table:**

ATTRIBUTES	DATATYPE	CONSTRAINT
RoomNo	NVARCHAR (10)	NOTNULL
Department	NVARCHAR (30)	NOTNULL
RoomID	INT	Primary Key

➢ **Teachers Table:**

ATTRIBUTES	DATATYPE	CONSTRAINT
TeacherID	INT	PRIMARY KEY
TeachersName	NVARCHAR (30)	NOTNULL
Subject	NVARCHAR (30)	NOTNULL
Branch	NVARCHAR (20)	NOTNULL

➢ **UserCertificates Table:**

ATTRIBUTES	DATATYPE	CONSTRAINT
ID	INT	PRIMARY KEY
Name	NVARCHAR (30)	Foreign Key(Reference Main(Name))
RollNo	NVARCHAR (30)	NOTNULL
Class	NVARCHAR (2)	NOTNULL
CertificateFile	VARBINARY(50)	NOTNULL
CertificateLink	NVARCHAR (50)	NOTNULL
Branch	NVARCHAR (50)	Foreign Key(Reference Teachers(Branch))
OrgName	VARCHAR(30)	NOTNULL
Course	VARCHAR(	NOTNULL
StartDate	DATE	Check:<=EndDate
EndDate	DATE	Check:<=StartDate

The database schema for the college website system includes tables for managing user information, event details, room allocations, schedules, subjects, teachers, and user certificates. The **Main** table stores user credentials, while the **EventList** table contains event information. **Room** details are managed in the Room table, and class **schedules** are stored in the Schedule table. **Subject** names are stored in the Subjects table, and teacher information is managed in the **Teachers** table. The **UserCertificates** table stores user certificate details. This schema efficiently organizes the data needed for the college website system.

4.3) User Interface and Design:

The user interface (UI) for the college website system is meticulously crafted to provide an exceptional user experience. It employs a responsive design approach, ensuring seamless functionality and aesthetic appeal across various devices, including desktops, laptops, tablets, and smartphones. This responsive design allows users to access the website from any device without compromising usability or visual appeal.

The UI features a clear and intuitive navigation system, enabling users to easily navigate through different sections of the website. A well-structured navigation menu provides quick access to essential pages such as home, courses, events, timetable, and certificates. This streamlined navigation enhances user engagement and encourages exploration of the website's content.

Consistency in layout and design elements is maintained throughout the website, offering users a cohesive and familiar browsing experience. Consistent use of colors, typography, and design patterns helps users quickly identify and understand the website's content and functionality.

Interactive elements such as buttons, forms, and dropdowns are strategically placed to facilitate user interaction. These elements are designed to be intuitive and user-friendly, enhancing the overall usability of the website. Users can easily navigate, search for information, and interact with various features without encountering any usability issues.

Accessibility is a key focus in the UI design, ensuring that the website is accessible to users with disabilities. The design complies with web accessibility standards, including providing alternative text for images, ensuring keyboard navigation, and using semantic HTML elements. These accessibility features make the website usable for all users, regardless of their abilities.

In conclusion, the user interface design of the college website system is carefully crafted to provide an engaging, intuitive, and accessible experience for all users. The design combines responsive layout, clear navigation, consistent design elements, thoughtful use of colors and typography, interactive features, and accessibility standards to deliver a user-friendly and visually appealing interface.

- **Client View:**

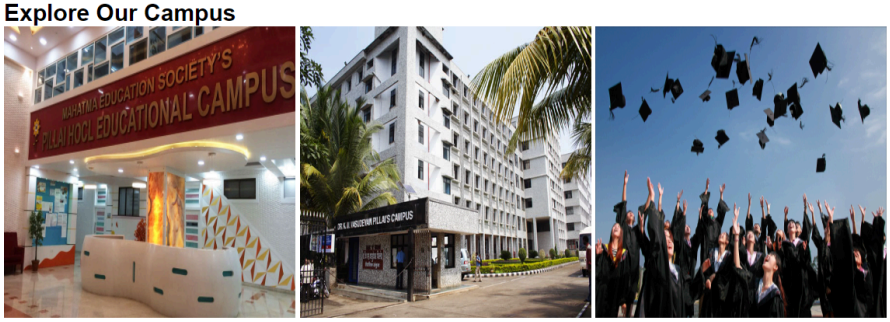
Welcome to ABC College! Today is Monday, February 19, 2024

ABC COLLEGE

EVENTS TIME TABLE LOGIN LOGOUT COURSE CERTIFICATE

Hello Student2 (student2.mes.ac.in)!

Explore Our Campus



KAVACH 2023

Proud to Share that ABC College of Engineering, New Panvel (Autonomous, NAAC A+) is selected as one of the Five Nodal Centres all over India to host the Grand Finale of KAVACH 2023, a 36 Hrs Non-Stop Cyber Security Hackathon, on 8th – 9th August, 2023 with teams participating from all over the nation at Dr. K. M. Vasudevan Campus, New Panvel. We are greatly honoured and thankful to AICTE, Ministry of Home Affairs, Ministry of Education and MIC for this opportunity

- **Apply for College:**

Why Choose ABC College?

- Quality Education
- Experienced Faculty
- Modern Facilities
- Research Opportunities

Apply Now!

Join our vibrant community and take the first step towards your future.

Apply Now

- **AboutUS:**

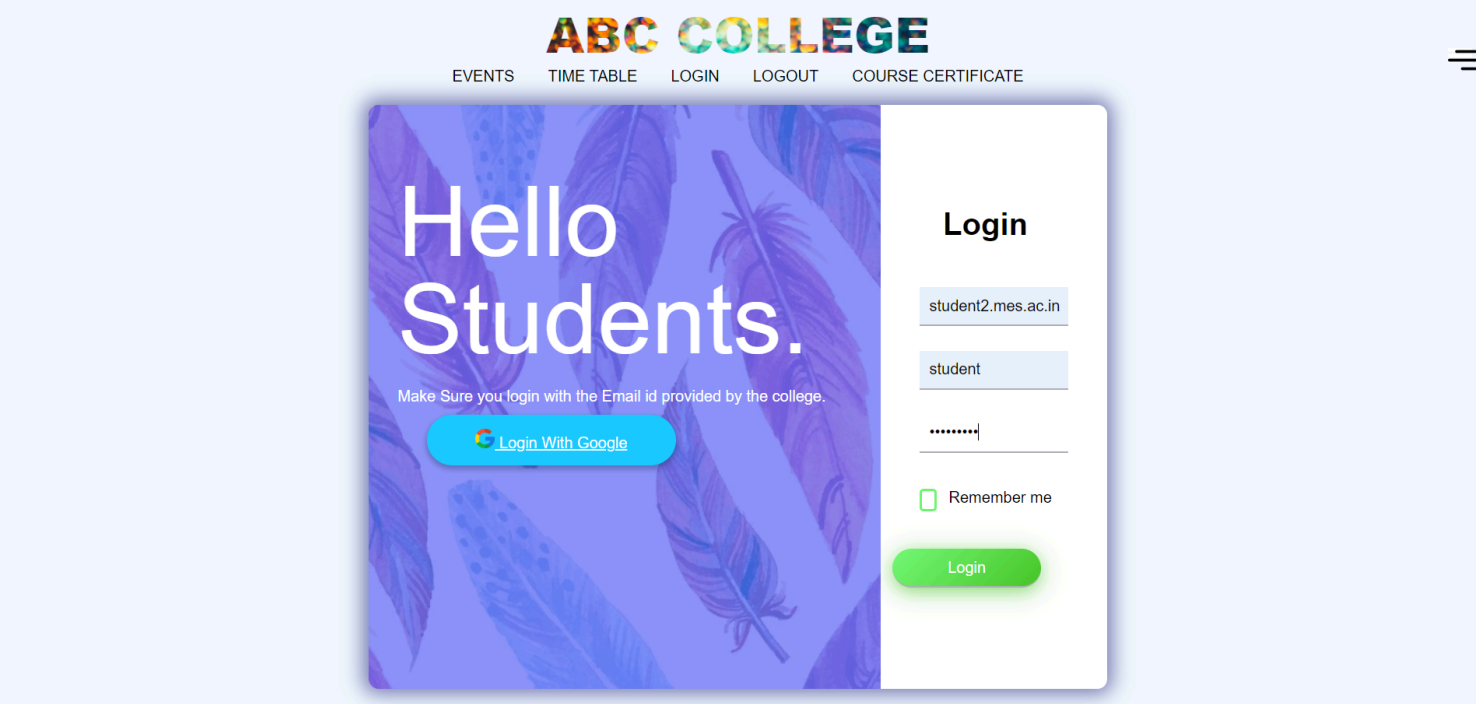
410206, Sector 15, New Panvel East, Panvel, Navi Mumbai, Maharashtra 410206

About Us
ABC College is dedicated to providing high-quality education...

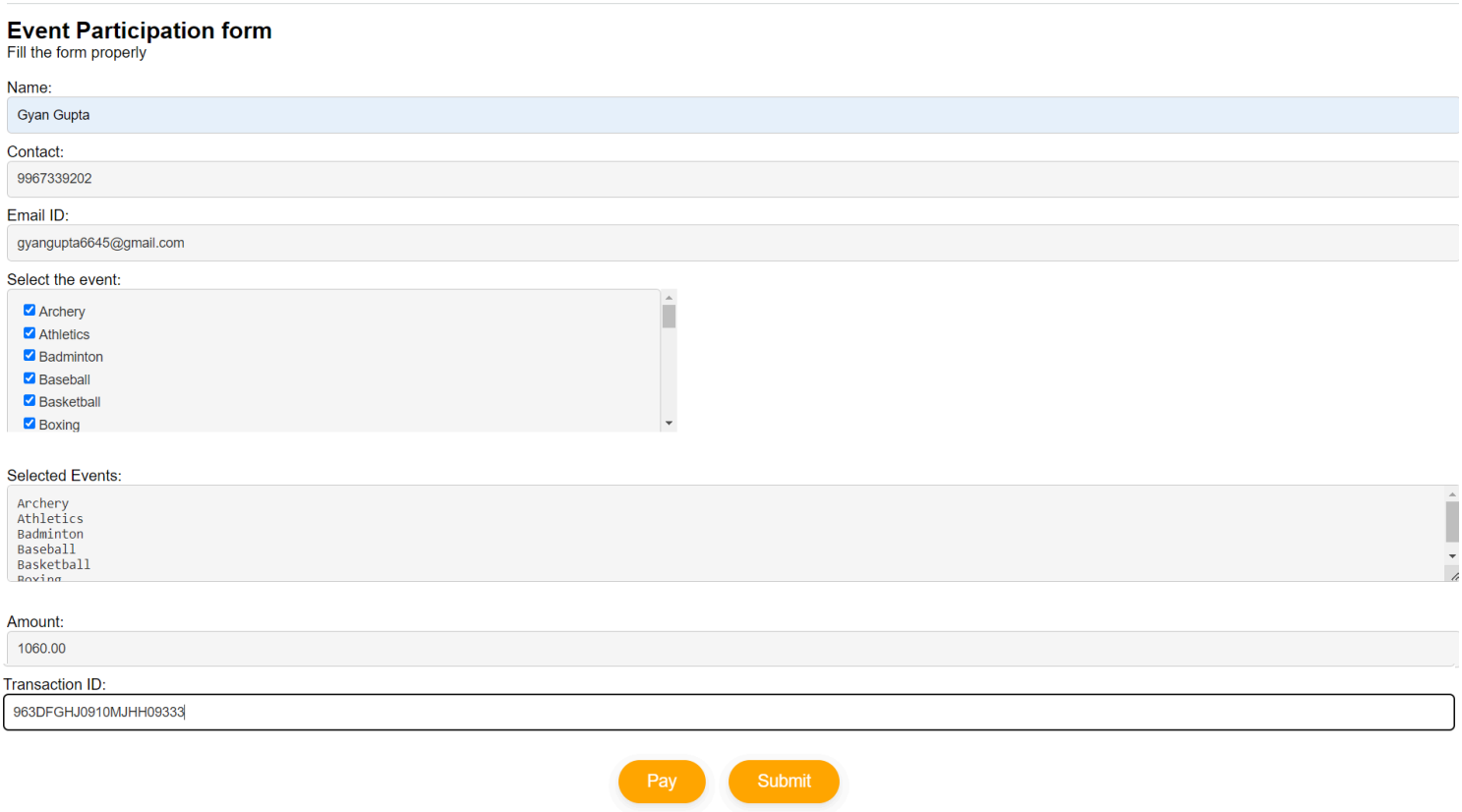
Contact Us
Phone:+91 912342424 Tel: 044 244 548
Whatsapp: +91 7843993477

Instagram Facebook Linkedin Gamil

- **LOGIN PAGE:**



- **Event Registration Form:**



Welcome to ABC College! Today is Monday, February 19, 2024

ABC COLLEGE

EVENTS TIME TABLE LOGIN LOGOUT COURSE CERTIFICATE

Hello Student2 (student2.mes.ac.in)!

Upload Certificate

Name:

Prathamesh Gawade

Roll No.:

8430

Branch:

Ty B.sc I.T

Class:

B

Certificate File:

Choose file 8433 BI PROJECT (1) (1).pdf

Link of the Certificate:

www.coursera.co.in

Name of the Organization:

Coursera

Course:

- Paid
- Free

Start Date:

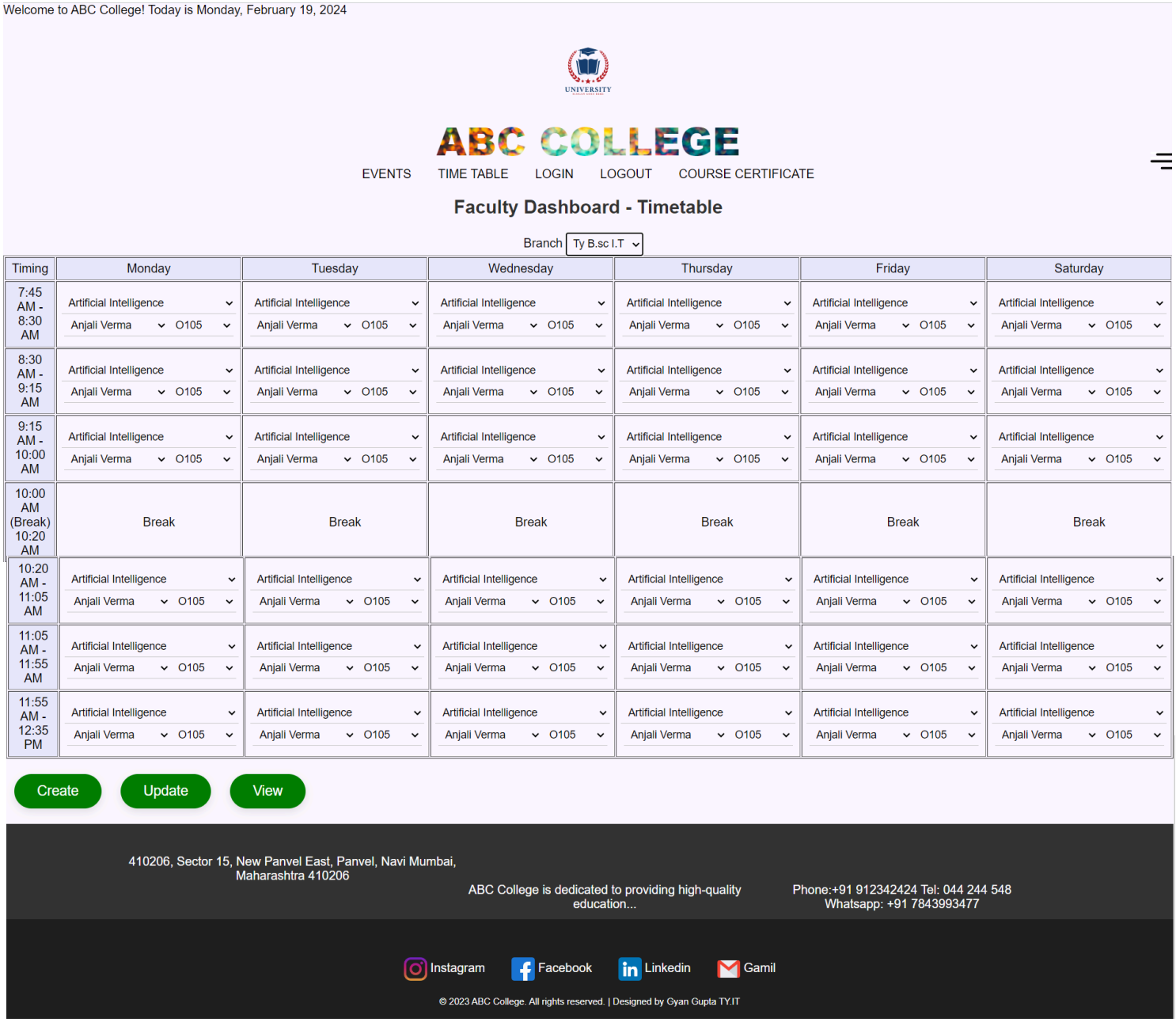
2019-09-01

End Date:

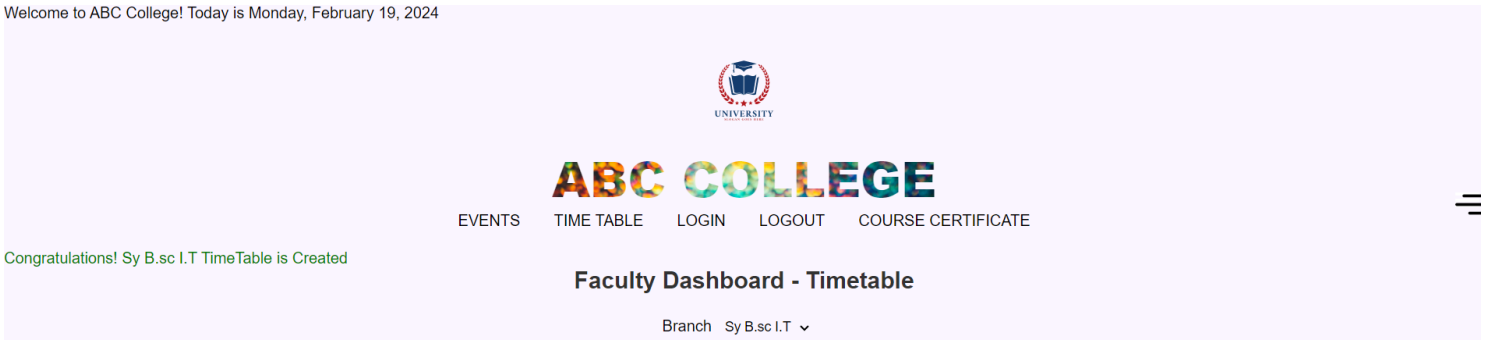
2019-12-31

Submit

- **Create Time-Table Page:**



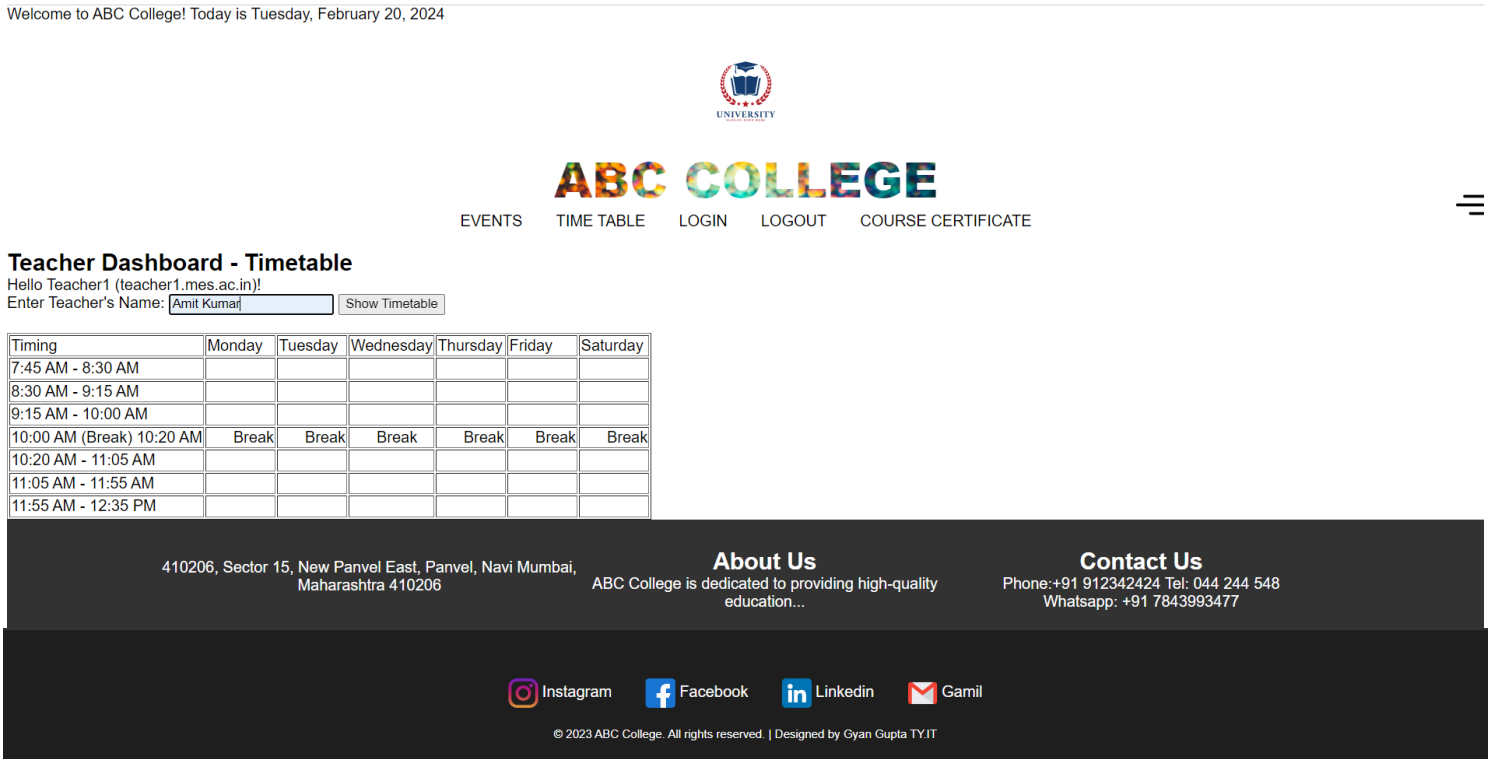
- **Timetable created Successfully:**



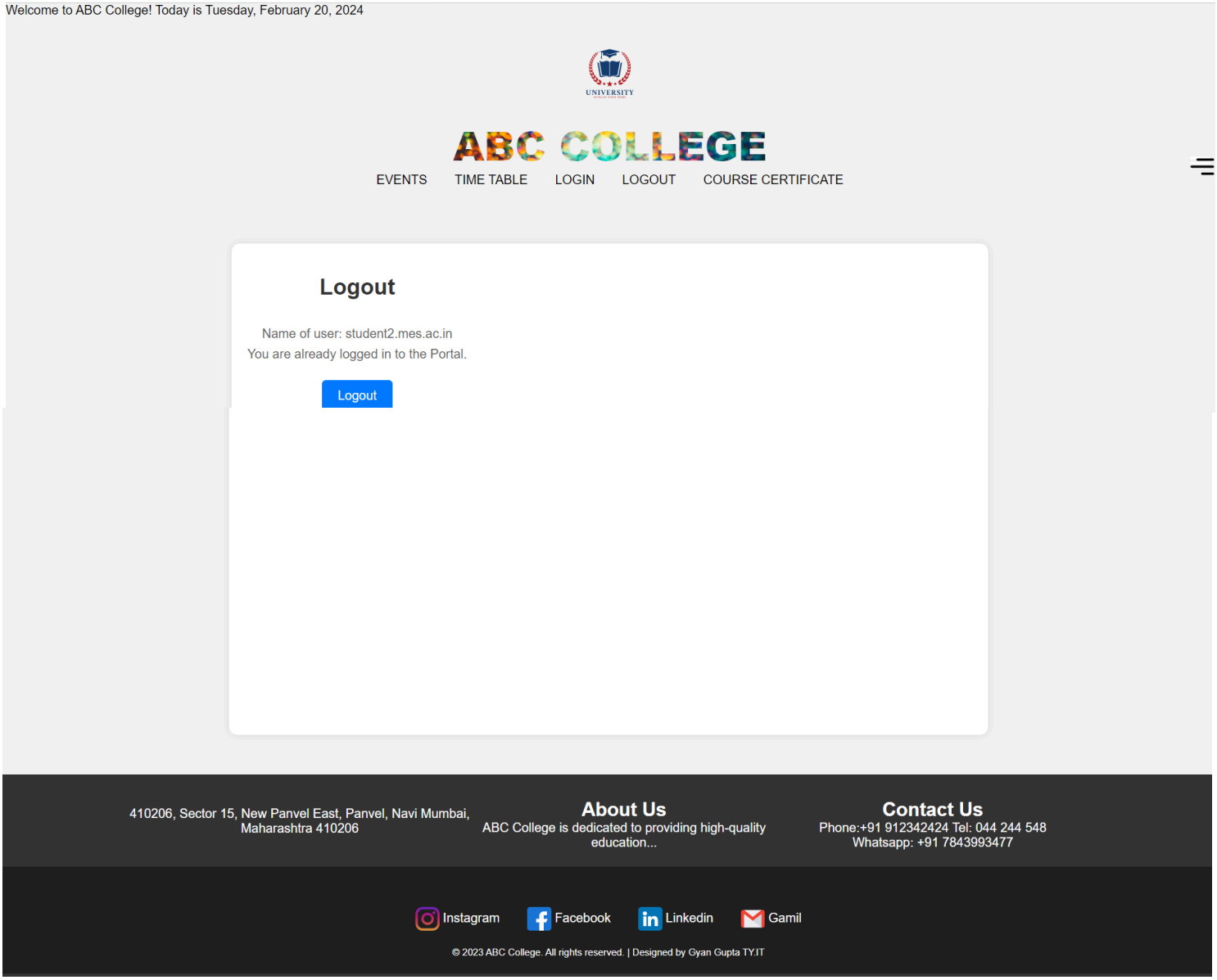
- **View Time-Table by Student:**

Timing	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
7:45 AM - 8:30 AM	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence
8:30 AM - 9:15 AM	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence
9:15 AM - 10:00 AM	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence
10:00 AM (Break) 10:20 AM	Subject: Teacher: Room:	Subject: Teacher: Room:	Subject: Teacher: Room:	Subject: Teacher: Room:	Subject: Teacher: Room:	Subject: Teacher: Room:
10:20 AM - 11:05 AM	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence
11:05 AM - 11:55 AM	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence
11:55 AM - 12:35 PM	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence	Subject: Artificial Intelligence Teacher: Artificial Intelligence Room: Artificial Intelligence

- **View Time-Table by Student:**



- **Logout Page For all students:**



4.4) Security Issues:

- **SQL INJECTIONS**

  SQL injection is a type of web application security vulnerability in which an attacker attempts to use application code to access or corrupt database content. If successful, this allows the attacker to create, read, update, alter, or delete data stored in the back-end database. SQL injection is one of the most prevalent types of web application security vulnerabilities.

- **CROSS SITE SCRIPTING (XSS)**

  Cross-site scripting (XSS) targets an application's users by injecting code, usually a client-side script such as JavaScript, into a web application's output. The concept of XSS is to manipulate client-side scripts of a web application to execute in the manner desired by the attacker. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface websites or redirect the user to malicious sites.

- **BROKEN AUTHENTICATION & SESSION MANAGEMENT**

  Broken authentication and session management encompass several security issues, all of them having to do with maintaining the identity of a user. If authentication credentials and session identifiers are not protected at all times, an attacker can hijack an active session and assume the identity of a user.

- **INSECURE DIRECT OBJECT REFERENCES**

  Insecure direct object reference is when a web application exposes a reference to an internal implementation object. Internal implementation objects include files, database records, directories and database keys. When an application exposes a reference to one of these objects in a URL, hackers can manipulate it to gain access to a user's personal data.

- **SECURITY MISCONFIGURATION**

  Security mis-configuration encompasses several types of vulnerabilities all centered on a lack of maintenance or a lack of attention to the web application configuration. A secure configuration must be defined and deployed for the application, frameworks, application server, web server, database server and platform. Security misconfiguration gives hackers access to private data or features and can result in a complete system compromise.

- **CROSS-SITE REQUEST FORGERY (CSRF)**

Cross-Site Request Forgery (CSRF) is a malicious attack where a user is tricked into performing an action he or she didn't intend to do. A third-party website will send a request to a web application that a user is already authenticated against (e.g., their bank). The attacker can then access functionality via the victim's already authenticated browser. Targets include web applications like social media, in browser email clients, online banking, and web interfaces for network devices.

Don't get caught with your guard down. Practice safe website security measures and always be ready to protect yourself, and your company's future, from an attack that you might never recover from. The best way to tell if your website or server is vulnerable is to conduct regular security audits.

- **Encryption Algorithms**

Encryption algorithms are commonly used in computer communications, including FTP transfers. Usually, they are used to provide secure transfers. If an algorithm is used in a transfer, the file is first translated into a seemingly meaningless cipher text and then transferred in this configuration; the receiving computer uses a key to translate the cipher into its original form. So, if the message or file is intercepted before it reaches the receiving computer it is in an unusable (or encrypted) form.

❖ **DES/3DES or TripleDES:**

This is an encryption algorithm called Data Encryption Standard that was first used by the U.S. Government in the late 70's. It is commonly used in ATM machines (to encrypt PINs) and is utilized in UNIX password encryption. Triple DES or 3DES has replaced the older versions as a more secure method of encryption, as it encrypts data three times and uses a different key for at least one of the versions.

❖ **Blowfish**

Blowfish is a symmetric block cipher that is unpatented and free to use. It was developed by Bruce Schneider and introduced in 1993.

- **AES**

  Advanced Encryption Standard or Rijndael; it uses the Rijndael block cipher approved by the National Institute of Standards and Technology (NIST). AES was originated by cryptographers Joan Daemen and Vincent Rijmen and replaced DES as the U.S. Government encryption technique in 2000.

- **Twofish**

  Twofish is a block cipher designed by Counterpane Labs. It was one of the five Advanced Encryption Standard (AES) finalists and is unpatented and open source.

- **IDEA**

  This encryption algorithm was used in Pretty Good Privacy (PGP) Version 2 and is an optional algorithm in OpenPGP. IDEA features 64 bit blocks with a 128-bit key.

- **MD5**

  MD5 was developed by Professor Ronald Riverst and was used to create digital signatures. It is a one- way hash function and intended for 32-bit machines. It replaced the MD4 algorithm.

- **SHA 1**

  SHA 1 is a hashing algorithm similar to MD5, yet SHA 1 may replace MD5 since it offers more security

- **HMAC**

  This is 3a hashing method similar to MD5 and SHA 1, sometimes referred to as HMAC MD5 and HMAC SHA1.

- **RSA Security**

  RC4: RC4 is a variable key size stream cipher based on the use of a random permutation. RC5:
  This is a parameterized algorithm with a variable block, key size and number of rounds.
  RC6: This an evolution of RC5, it is also a parameterized algorithm that has variable block, key and a number of rounds. This algorithm has integer multiplication and
  4-bit working registers

4.5) Test Case Design:

A test case is a defined format for software testing required to check if a particular application/software is working or not. A test case consists of a certain set of conditions that need to be checked to test an application or software i.e. in more simple terms when conditions are checked it checks if the resultant output meets with the expected output or not. A test case consists of various parameters such as ID, condition, steps, input, expected result, result, status, and remarks.

Parameters of a Test Case:

- Module Name: Subject or title that defines the functionality of the test.
- Test Case Id: A unique identifier assigned to every single condition in a test case.
- Tester Name: The name of the person who would be carrying out the test.
- Test scenario: The test scenario provides a brief description to the tester, as in providing a small overview to know about what needs to be performed and the small features, and components of the test.
- Test Case Description: The condition required to be checked for a given software. for eg. Check if only numbers validation is working or not for an age input box.
- Test Steps: Steps to be performed for the checking of the condition.
- Test Data: The inputs to be taken while checking for the conditions.
- Test Expected Result: The output which should be expected at the end of the test.
- Test parameters: Parameters assigned to a particular test case.
- Actual Result: The output that is displayed at the end.
- Environment Information: The environment in which the test is being performed, such as the operating system, security information, the software name, software version, etc.
- Status: The status of tests such as pass, fail, NA, etc.
- Comments: Remarks on the test regarding the test for the betterment of the software.

A. Login Page Test Case:

SR NO.	Form Name	Test Condition	Step or Procedure	Input Test Data	Expected Result	Actual Output	Pass/ Fail
1	Login	Wrong Email ID & MemberType with correct password	Enter incorrect email ID and member type with correct password, click login button	Incorrect email ID: faculty.mes.ac.in Incorrect member type: Student Correct password: Password7	Incorrect Email ID	Incorrect Email ID	Pass
3	Login	Wrong Member with correct email ID & password	Enter correct email ID and password with incorrect member type, click login button	Correct email ID: faculty1.mes.ac.in Correct password: Password7 Incorrect member type: student	Incorrect Member Type	Incorrect Member Type	Pass
4	Login	Wrong Password with correct Email ID & MemberType	Enter incorrect Password with correct Email ID and member type, click login button	Incorrect password: Password Correct email ID: faculty.mes.ac.in Correct member type: Faculty	Incorrect Password	Incorrect Password	Pass

B. Create Table Page:

SR NO.	Form Name	Test Condition	Step or Procedure	Input Test Data	Expected Result	Actual Output	Pass/ Fail
1	Create Table	Valid input	Enter valid table name and click on the create button	Table Name: Students	Table should be created successfully	Table created successfully	Pass
2	Create Table	Table name already exists	Enter a table name that already exists and click on the create button	Table Name: Students	Error message should be displayed	Error message displayed	Pass
3	Create Table	Table name with special characters	Enter a table name with special characters (e.g., !@#$%^&*) and click on the create button	Table Name: Students!@#$%^&*	Error message should be displayed	Error message displayed	Pass

4	Create Table	Table name with numbers	Enter a table name with numbers (e.g., Students123) and click on the create button	Table Name: Students123	Table should be created successfully	Table created successfully	Pass
5	Create Table	Table name with maximum length	Enter a table name with the maximum allowed length (e.g., 128 characters) and click on the create button	Table Name: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam euismod ultricies magna, eget finibus nibh.	Table should be created successfully	Table created successfully	Pass
6	Create Table	Table name with more than maximum length	Enter a table name with more than the maximum allowed length and click on the create button	Table Name: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam euismod ultricies magna, eget finibus nibh.	Error message should be displayed	Error message displayed	Pass
7	Create Table	Table name with leading/trailing whitespaces	Enter a table name with leading or trailing whitespaces and click on the create button	Table Name: Students	Error message should be displayed	Error message displayed	Pass
8	Create Table	Table name as empty string	Leave the table name field empty and click on the create button	Table Name:	Error message should be displayed	Error message displayed	Pass
9	Create Table	Table name with SQL injection	Enter a table name with SQL injection (e.g., Students'; DROP TABLE Students; --) and click on the create button	Table Name: Students'; DROP TABLE Students; --	Error message should be displayed	Error message displayed	Pass
10	Create Table	Table name with SQL keywords	Enter a table name with SQL keywords (e.g., SELECT, INSERT, UPDATE) and click on the create button	Table Name: SELECT	Error message should be displayed	Error message displayed	Pass

C. View Certificate Page:

SR NO.	Form Name	Test Condition	Step or Procedure	Input Test Data	Expected Result	Actual Output	Pass/ Fail
1	View Certificates	No certificates uploaded	Navigate to the View Certificates page	N/A	Display message indicating no certificates uploaded	Message displayed: No certificates uploaded	Pass
2	View Certificates	Certificates uploaded	Upload certificates for a user and navigate to the View Certificates page	Certificate file	Display uploaded certificates for the user	Uploaded certificates displayed	Pass
4	View Certificates	Valid user ID	Enter a valid user ID in the search field and click on the search button	Valid user ID	Display certificates uploaded by the user	Certificates displayed	Pass
5	View Certificates	Download certificate	Click on the download button next to a certificate	Certificate ID	Download the selected certificate	Certificate downloaded successfully	Pass
6	View Certificates	Delete certificate	Click on the delete button next to a certificate	Certificate ID	Delete the selected certificate	Certificate deleted successfully	Pass
7	View Certificates	Invalid certificate ID	Enter an invalid certificate ID in the delete confirmation popup and click on the delete button	Invalid certificate ID	Display message indicating invalid certificate ID	Message displayed: Invalid certificate ID	. Pass
9	View Certificates	Download all certificates	Click on the download all button	N/A	Download all certificates as a zip file	Certificates downloaded as a zip file	Pass

CHAPTER 5: System Coding, Implementation and Testing

In this pivotal chapter, we embark on the journey of translating our meticulously crafted system design into tangible code. We delve deep into the intricacies of coding, discussing the specific details of how our system is implemented. This includes the selection of appropriate technologies, programming languages, and frameworks that best suit our requirements. We also explore the architectural decisions that underpin the structure and organization of our codebase, ensuring that it is both scalable and maintainable.

Efficiency in coding is paramount to the success of our system. We devote considerable attention to coding practices that enhance efficiency, such as optimizing algorithms, managing memory usage effectively, and improving overall system performance. By adhering to these best practices, we strive to deliver a system that not only meets but exceeds the performance expectations of our users.

Testing is a cornerstone of our development process, and we adopt a comprehensive approach to ensure the robustness and reliability of our system. We discuss our testing strategy, which includes both unit testing to validate individual components and integrated testing to ensure that all components work harmoniously together. Through rigorous testing, we aim to identify and rectify any issues or bugs, ensuring that our system functions seamlessly in all scenarios.

In summary, this chapter is pivotal in the development lifecycle of our system. By focusing on coding, implementation, and testing, we are committed to delivering a high-quality product that fulfills the needs and expectations of our users and stakeholders.

5.1) Coding Details:

- **Languages and Frameworks:** The system is developed primarily using ASP.NET C# for the backend and JavaScript for the frontend. ASP.NET provides a robust framework for building web applications, offering features such as server-side logic, database connectivity, and security. JavaScript is used to enhance the user interface and provide interactive elements.
- **Database Management:** Microsoft SQL Server is utilized as the database management system (DBMS) for storing and managing the system's data. SQL queries are used to interact with the database, including retrieving, updating, and deleting data.
- **User Authentication:** The system implements user authentication using ASP.NET Identity, which provides features for managing user accounts, including registration, login, and password management. User passwords are securely hashed and stored in the database.
- **File Uploads:** The system allows users to upload files, such as certificates, which are stored in the file system or database, depending on the configuration. File uploads are validated to ensure they meet the system's requirements.
- **Error Handling:** Robust error handling mechanisms are implemented throughout the system to catch and handle exceptions gracefully. Error messages are logged for debugging purposes and to improve system reliability.
- **Code Structure:** The codebase is organized into logical modules and classes, following best practices for code readability and maintainability. Object-oriented programming principles are applied to ensure code reusability and scalability.
- **Testing:** Unit tests are written using NUnit and Moq to test individual components of the system. Integration tests are also conducted to ensure that different modules interact correctly and produce the expected outcomes.
- **Documentation:** Inline comments and documentation are provided throughout the codebase to explain the purpose and functionality of each component. This helps developers understand and maintain the codebase effectively.

Overall, the coding details demonstrate a meticulous approach to developing a robust and secure system that meets the requirements of its users. Through the use of industry-standard technologies and best practices, the system is designed to be scalable, maintainable, and reliable.

5.2) Coding Efficiency:

Client Side:

a. home.aspx.cs:

```
using System;
namespace try1
{
   public partial class Home : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         // Check if the user is logged in
         if (Session["IsLoggedIn"] != null && Convert.ToBoolean(Session["IsLoggedIn"]))
         {
            // Retrieve user details from the session
            string userEmail = Session["UserEmail"].ToString();
            string userName = Session["UserName"].ToString();

            // Display a personalized greeting
            Label2.Text = $"Hello {userName} ({userEmail})!";
         }

         // Code to be executed only on the initial page load
         // Set the text of a label based on the current date
      }
   }
}
```

b. login.aspx.cs:

```
using System;
using System.Data.SqlClient;
using System.Web;
using System.Web.Script.Serialization;
namespace try1
{
   public partial class login : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         HttpCookie authCookie = new HttpCookie("AuthenticationCookie");
         authCookie.Values["IsLoggedIn"] = "true";
         authCookie.Values["UserEmail"] = "Email"; // Set this to the actual user's name
         authCookie.Expires = DateTime.Now.AddMinutes(30); // Set the expiration time for the cookie
(adjust as needed)
         Response.Cookies.Add(authCookie);
      }
      protected void loginButton_Click(object sender, EventArgs e)
      {
         // Get user input
         string userEmail = Email.Text;
         string userPassword = Password.Text;

         // Create a connection to the database
         string connectionString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\gyang\\source\\repos\\try1\\try
```

```
1\\App_Data\\StudentDB.mdf;Integrated Security=True";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            try
            {
                connection.Open();
                // Prepare a SQL query to retrieve the user data associated with the given email
                string query = "SELECT Member, ID, Name, Address, Contact, Password FROM Main WHERE
EmailID = @Email";
                SqlCommand cmd = new SqlCommand(query, connection);
                cmd.Parameters.AddWithValue("@Email", userEmail);
                // Execute the query
                SqlDataReader reader = cmd.ExecuteReader();
                if (reader.Read())
                {
                    // Email exists, retrieve the user data
                    string storedPassword = reader["Password"].ToString();
                    // Verify the password
                    if (userPassword == storedPassword)
                    {
                        // Passwords match, user is authenticated
                        // Set session variables for user details
                        Session["IsLoggedIn"] = true;
                        Session["UserEmail"] = userEmail;
                        Session["UserName"] = reader["Name"].ToString();
                        Session["UserID"] = reader["ID"].ToString();
                        Session["UserAddress"] = reader["Address"].ToString();
```

```
            Session["UserContact"] = reader["Contact"].ToString();
            Session["MemberType"] = reader["Member"].ToString();
            // Redirect the user to a specific page based on the member type
            if (Session["MemberType"].ToString() == "student" ||
               Session["MemberType"].ToString() == "teacher" ||
               Session["MemberType"].ToString() == "faculty")
            {
               Response.Redirect("home.aspx");
            }}
         else
         {
            // Passwords do not match, show an error message
            Response.Write("Invalid email or password.");
         }
      }
      else
      {
         // Email does not exist, show an error message
         Response.Write("Please enter the valid email.");
      }
   }
   catch (Exception ex)
   {
      // Handle any exceptions (e.g., database connection error)
      Response.Write("An error occurred: " + ex.Message);
   }
}}}
```

c. logout.aspx.cs:

```
using System;
using System.Web.UI;
namespace try1
{
   public partial class logout : Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         // Redirect to login page if not logged in
         if (Session["userEmail"] == null)
         {
            Response.Redirect("login.aspx");
         }
      }
      protected void btnLogout_Click(object sender, EventArgs e)
      {
         // Clear the session
         Session.Abandon();
         // Redirect to the login page
         Response.Redirect("login.aspx");
      }
   }
}
```

d. event.aspx.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Web.UI.WebControls;
namespace try1
{
   public partial class _event : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         if (!IsPostBack)
         {
            DataTable eventDataTable = GetEventData(); // Corrected method name
            cblEvents.DataSource = eventDataTable;
            cblEvents.DataTextField = "EventName";
            cblEvents.DataValueField = "EntryFees";
            cblEvents.DataBind();
            foreach (ListItem item in cblEvents.Items)
            {
               item.Attributes["title"] = $"{item.Text}|{item.Value}";
            } }  }
      private DataTable GetEventData()
      {
  string connectionString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\gyang\\source\\repos\\try1
\\try1\\App_Data\\StudentDB.mdf;Integrated Security=True";
```

```
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            string query = "SELECT EventName, EntryFees FROM EventList"; // Replace
'YourEventTable' with your actual table name
            SqlDataAdapter adapter = new SqlDataAdapter(query, connection);
            DataTable EventList = new DataTable();
            adapter.Fill(EventList);
            return EventList;
        } }
    private void BindEventsToCheckBoxList()
    {
        // Replace the connection string with your actual database connection string
        string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\gyang\source\repos\try1\try1\A
pp_Data\StudentDB.mdf;Integrated Security=True";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            // Open the database connection
            connection.Open();
            // Define the SQL query to fetch data from the database
            string query = "SELECT EventName, " +
                    "CONCAT(EventName, ' - Entry Fees: ', EntryFees, ' | First Prize: ', FirstPrize, ' |
Second Prize: ', SecondPrize, ' | Third Prize: ', ThirdPrize) AS DisplayText " +
                    "FROM EventList";
            // Create a SqlCommand object to execute the query
            using (SqlCommand command = new SqlCommand(query, connection))
```

```
        {
            // Execute the query and get the data
            using (SqlDataReader reader = command.ExecuteReader())
            {
                // Bind the data to the CheckBoxList
                cblEvents.DataSource = reader;
                cblEvents.DataTextField = "DisplayText";
                cblEvents.DataValueField = "EventName";
                cblEvents.DataBind();
            } } } }
    protected void btnPayment_Click(object sender, EventArgs e)
    {
        // Your existing code for btnPayment_Click
    }
    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        String connectionSting = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\gyang\\source\\repos\\try1
\\try1\\App_Data\\StudentDB.mdf;Integrated Security=True";
        using (SqlConnection connection = new SqlConnection(connectionSting))
        {
            string Querry = "";
        } }
    protected void lbEvents_SelectedIndexChanged(object sender, EventArgs e)
    {
        // Your existing code for lbEvents_SelectedIndexChanged
    } } }
```

e. upload_certificate.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Web.UI.WebControls;
namespace try1
{
   public partial class upload_certificate : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         if (!IsPostBack)
         {
            // Call a method to populate the branch dropdown list
            PopulateBranchDropdown();
         }
         // Check if the user is logged in
         if (Session["IsLoggedIn"] == null || !(bool)Session["IsLoggedIn"])
         {
            // Redirect the user to the login page or display a message
            Response.Redirect("login.aspx"); // Assuming login page is named "login.aspx"
         }
         else
         {
            // Check if the logged-in member is a student
```

```
        if (Session["MemberType"] != null && Session["MemberType"].ToString() == "student")
        {
            // Retrieve user details from the session
            if (Session["UserEmail"] != null)
            {
                string userEmail = Session["UserEmail"].ToString();
                string userName = Session["UserName"].ToString();
                // Display a personalized greeting or perform any other actions
                Label2.Text = $"Hello {userName} ({userEmail})!";
            } }
        else
        {
            // Show a message indicating that only student members can upload certificates
            showPopup();
        } } }
private void PopulateBranchDropdown()
{
    // Retrieve branches from the database and populate the dropdown list
    List<string> branches = GetBranchesFromDatabase();
    ddlBranch.DataSource = branches;
    ddlBranch.DataBind();
}
private List<string> GetBranchesFromDatabase()
{
    List<string> branches = new List<string>();
    // Connection string for your database
    string connectionString = "Data
```

```
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\gyang\\source\\repos\\try1
\\try1\\App_Data\\StudentDB.mdf;Integrated Security=True";
        // SQL query to fetch branches
        string query = "SELECT DISTINCT Branch FROM Teachers"; // Assuming 'Branch' is a column
in the Teachers table
        // Create a SqlConnection and a SqlCommand
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
           using (SqlCommand command = new SqlCommand(query, connection))
           {
              try
              {
                 // Open the connection
                 connection.Open();
                 // Execute the query and read the results
                 SqlDataReader reader = command.ExecuteReader();
                 while (reader.Read())
                 {
                    // Add each branch name to the list
                    branches.Add(reader["Branch"].ToString());
                 } }
              catch (Exception ex)
              {
                 // Handle exceptions, if any
                 Console.WriteLine("An error occurred: " + ex.Message);
              } } }
        return branches;
```

```
    }
    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        // Check if the user is logged in and is a student member
        if (Session["IsLoggedIn"] != null && (bool)Session["IsLoggedIn"] && Session["MemberType"]
!= null && Session["MemberType"].ToString() == "student")
        {
            // Process form submission and store data in the database
            try
            {
                // Your database connection string
                string connectionString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\gyang\\source\\repos\\try1
\\try1\\App_Data\\StudentDB.mdf;Integrated Security=True";
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    // Your SQL insert query
                    string query = "INSERT INTO UserCertificates (Name, RollNo, Branch, Class,
CertificateFile, CertificateLink, OrgName, Course, StartDate, EndDate) " +
                                           "VALUES (@Name, @RollNo, @Branch, @Class,
CONVERT(varbinary(max), @CertificateFile), @CertificateLink, @OrgName, @Course, @StartDate,
@EndDate)";
                    using (SqlCommand command = new SqlCommand(query, connection))
                    {
                        // Add parameters
                        command.Parameters.AddWithValue("@Name", txtName.Text);
                        command.Parameters.AddWithValue("@RollNo", txtRollNo.Text);
```

```
                command.Parameters.AddWithValue("@Branch", ddlBranch.SelectedItem.Text);
                command.Parameters.AddWithValue("@Class", ddlClass.SelectedItem.Text);
                try
                {
                    // Save file to directory and get the file path
                    string fileName = Path.GetFileName(fileCertificate.FileName);
                    string filePath = "C://Users//gyang//source//repos//try1//student Certificate/" +
fileName;
                    fileCertificate.SaveAs(filePath);
                    command.Parameters.AddWithValue("@CertificateFile", filePath);
                }
                catch (Exception ex)
                {
                    // Handle the exception
                    lblMessage.Text = "An error occurred while saving the file: " + ex.Message;
                }
                command.Parameters.AddWithValue("@CertificateLink", txtCertificateLink.Text);
                command.Parameters.AddWithValue("@OrgName", txtOrgName.Text);
                command.Parameters.AddWithValue("@Course", rblCourse.SelectedItem.Value);
                command.Parameters.AddWithValue("@StartDate", txtStartDate.Text);
                command.Parameters.AddWithValue("@EndDate", txtEndDate.Text);
                // Open connection
                connection.Open();
                // Execute query
                int rowsAffected = command.ExecuteNonQuery();
                // Close connection
                connection.Close();
```

```
                if (rowsAffected > 0)
                {
                    // Display success message
                    lblMessage.Text = "Certificate uploaded successfully!";
                }
                else
                {
                    // Display error message
                    lblMessage.Text = "Failed to upload certificate!";
                } } } }
            catch (Exception ex)
            {
                // Handle exceptions
                lblMessage.Text = "An error occurred: " + ex.Message;
            } }
        else
        {
            // Show a message indicating that only student members can upload certificates
            showPopup();
        } }
    private void showPopup()
    {
        // You can use a modal popup library like Bootstrap modal to show a modal popup
        // For simplicity, I'll just display an alert here
        ClientScript.RegisterStartupScript(this.GetType(), "alert", "alert('You are not a student member.
Only student members can upload certificates.');", true);
    } } }
```

f. faculty_dashboard.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Linq;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml.Linq;
namespace try1
{
   public partial class faculty_dashboard : System.Web.UI.Page
   {
      protected HiddenField UpdateHiddenField;
      protected void Page_Load(object sender, EventArgs e)
      {
         if (!IsPostBack)
         {
            // Check if the user is logged in
            if (Session["IsLoggedIn"] != null && (bool)Session["IsLoggedIn"])
            {
               // User is logged in
               string userEmail = Session["UserEmail"].ToString();
               // Access other session variables as needed
            }
            else
```

```
            {
                // User is not logged in
                // Redirect to the login page or handle accordingly
                Response.Redirect("login.aspx");
            }
        }
        {
            // Populate dropdown list with branches from the database
            PopulateBranchDropdown();
            // Populate dropdown lists with subjects and teachers
            PopulateDropdowns();
            // Initialize the DropDownListArray
            // Initialize the arrays
            if (UpdateHiddenField == null)
            {
                UpdateHiddenField = new HiddenField();
                UpdateHiddenField.ID = "UpdateHiddenField";
                // Additional properties can be set as needed
            }
        }
    }
    protected void PopulateBranchDropdown()
    {
        string connectionString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\gyang\\source\\repos\\try1
\\try1\\App_Data\\StudentDB.mdf;Integrated Security=True";
        string branchQuery = "SELECT Branch FROM Teachers";
```

```
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            using (SqlCommand branchCmd = new SqlCommand(branchQuery, connection))
            {
                branchCmd.CommandType = CommandType.Text;
                BranchDropdown.DataSource = branchCmd.ExecuteReader();
            }
            BranchDropdown.DataTextField = "Branch";
            BranchDropdown.DataValueField = "Branch";
            BranchDropdown.DataBind();
            connection.Close();
        }
    }
    protected void PopulateDropdowns()
    {
        string connectionString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\gyang\\source\\repos\\try1
\\try1\\App_Data\\StudentDB.mdf;Integrated Security=True";
        string subjectQuery = "SELECT SubjectName FROM Subjects";
        string teacherQuery = "SELECT TeacherName FROM Teachers";
        string roomQuery = "SELECT RoomNo FROM Room";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            PopulateDropdownsForDay(connection, subjectQuery, teacherQuery, roomQuery,
SubjectDropDownList6, TeacherDropDownList7, RoomDropDownList78, SubjectDropDownList18,
```

TeacherDropDownList19, RoomDropDownList84, SubjectDropDownList30,
TeacherDropDownList31, RoomDropDownList90, SubjectDropDownList42,
TeacherDropDownList43, RoomDropDownList96, SubjectDropDownList54,
TeacherDropDownList55, RoomDropDownList102, SubjectDropDownList66,
TeacherDropDownList67, RoomDropDownList108);

PopulateDropdownsForDay(connection, subjectQuery, teacherQuery, roomQuery,
SubjectDropDownList8, TeacherDropDownList9, RoomDropDownList79, SubjectDropDownList20,
TeacherDropDownList21, RoomDropDownList85, SubjectDropDownList32,
TeacherDropDownList33, RoomDropDownList91, SubjectDropDownList44,
TeacherDropDownList45, RoomDropDownList97, SubjectDropDownList56,
TeacherDropDownList57, RoomDropDownList103, SubjectDropDownList68,
TeacherDropDownList69, RoomDropDownList109);

PopulateDropdownsForDay(connection, subjectQuery, teacherQuery, roomQuery,
SubjectDropDownList10, TeacherDropDownList11, RoomDropDownList80, SubjectDropDownList22,
TeacherDropDownList23, RoomDropDownList86, SubjectDropDownList34,
TeacherDropDownList35, RoomDropDownList92, SubjectDropDownList46,
TeacherDropDownList47, RoomDropDownList98, SubjectDropDownList58,
TeacherDropDownList59, RoomDropDownList104, SubjectDropDownList70,
TeacherDropDownList71, RoomDropDownList110);

PopulateDropdownsForDay(connection, subjectQuery, teacherQuery, roomQuery,
SubjectDropDownList12, TeacherDropDownList13, RoomDropDownList81, SubjectDropDownList24,
TeacherDropDownList25, RoomDropDownList87, SubjectDropDownList36,
TeacherDropDownList37, RoomDropDownList93, SubjectDropDownList48,
TeacherDropDownList49, RoomDropDownList99, SubjectDropDownList60,
TeacherDropDownList61, RoomDropDownList105, SubjectDropDownList72,
TeacherDropDownList73, RoomDropDownList111);

PopulateDropdownsForDay(connection, subjectQuery, teacherQuery, roomQuery,

```
SubjectDropDownList14, TeacherDropDownList15, RoomDropDownList82, SubjectDropDownList26,
TeacherDropDownList27, RoomDropDownList88, SubjectDropDownList38,
TeacherDropDownList39, RoomDropDownList94, SubjectDropDownList50,
TeacherDropDownList51, RoomDropDownList100, SubjectDropDownList62,
TeacherDropDownList63, RoomDropDownList106, SubjectDropDownList74,
TeacherDropDownList75, RoomDropDownList112);
            PopulateDropdownsForDay(connection, subjectQuery, teacherQuery, roomQuery,
SubjectDropDownList16, TeacherDropDownList17, RoomDropDownList83, SubjectDropDownList28,
TeacherDropDownList29, RoomDropDownList89, SubjectDropDownList40,
TeacherDropDownList41, RoomDropDownList95, SubjectDropDownList52,
TeacherDropDownList53, RoomDropDownList101, SubjectDropDownList64,
TeacherDropDownList65, RoomDropDownList107, SubjectDropDownList76,
TeacherDropDownList77, RoomDropDownList113);
            connection.Close();
        }
    }
    private DataTable FetchData(SqlConnection connection, string query)
    {
        DataTable dataTable = new DataTable();
        using (SqlCommand cmd = new SqlCommand(query, connection))
        {
            cmd.CommandType = CommandType.Text;
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(dataTable);
        }
        return dataTable;
    }
```

```
    private void ShowError(string errorMessage, Exception ex)
    {
        // Log the exception details (replace with your logging mechanism)
        // For example, you can use a logging library or write to a log file
        // LogException(ex);
       // Display the error message to the user
        ErrorLabel.Text = errorMessage;
    }
    // Method to populate subject and teacher dropdowns for a specific day
    // Method to populate subject and teacher dropdowns for a specific day
    protected void PopulateDropdownsForDay(SqlConnection connection, string subjectQuery,
string teacherQuery, string roomQuery, params DropDownList[] dropdowns)
    {
        DataTable Subjects = FetchData(connection, subjectQuery);
        DataTable Teachers = FetchData(connection, teacherQuery);
        DataTable Room = FetchData(connection, roomQuery);
       // Assuming you have 6 time slots for each day (adjust as needed)
        int numberOfTimeSlots = 6;
        for (int i = 0; i < numberOfTimeSlots; i++)
        {
            int subjectIndex = i * 3;
            int teacherIndex = subjectIndex + 1;
            int roomIndex = subjectIndex + 2;
            // Populate subject dropdown
            PopulateDropdown(dropdowns[subjectIndex], Subjects, "SubjectName", "SubjectName");
            // Populate teacher dropdown
            PopulateDropdown(dropdowns[teacherIndex], Teachers, "TeacherName", "TeacherName");
```

```
            // Populate room dropdown
            PopulateDropdown(dropdowns[roomIndex], Room, "RoomNo", "RoomNo");
        }
    }
    private void PopulateDropdown(DropDownList dropdown, DataTable data, string textField,
string valueField)
    {
        dropdown.DataSource = data;
        dropdown.DataTextField = textField;
        dropdown.DataValueField = valueField;
        dropdown.DataBind();
    }
    protected global::System.Web.UI.WebControls.Table MainTable;
    private string GetSelectedValueFromDropDown(DropDownList dropDownList)
    {
        if (dropDownList != null)
        {
            return dropDownList.SelectedValue;
        }
        return string.Empty;
    }
    protected void CreateButton_Click(object sender, EventArgs e)
    {
        // List of days to iterate over
        string[] daysOfWeek = { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
};
```

```
try
{
    // Iterate through each day of the week
    foreach (string dayOfWeek in daysOfWeek)
    {
        // Iterate through each row in the MainTable
        foreach (TableRow row in MainTable.Rows)
        {
            // Skip the header row
            if (row.Cells.Count == 0)
                continue;
            // Extract data from the row
            string timeSlot = row.Cells[0].Text;

            // Get the selected branch from the BranchDropdown control
            string branch = BranchDropdown.SelectedValue;
            Debug.WriteLine("Selected branch: " + branch); // Add this debug statement

            // Use the modified method to get the selected values
            string subjectName =
GetSelectedValueFromDropDown(GetDropDownFromCell(row.Cells[2]));
            string teacherName =
GetSelectedValueFromDropDown(GetDropDownFromCell(row.Cells[3]));
            string roomNo =
GetSelectedValueFromDropDown(GetDropDownFromCell(row.Cells[4]));
            // Insert data into the database
            InsertDataIntoDatabase(timeSlot, dayOfWeek, branch, subjectName, teacherName,
```

```
roomNo);
                }
            }
            // Display success message
            SuccessLabel.Text = $"Congratulations! {BranchDropdown.SelectedValue} TimeTable is
Created";
            SuccessLabel.ForeColor = System.Drawing.Color.Green;
            SuccessLabel.Visible = true;
        }
        catch (Exception ex)
        {
            // Display error message
            ErrorLabel.Text = errorMessage + " Error Details: " + ex.Message;
        }
    }
    private DropDownList GetDropDownFromCell(TableCell cell)
    {
        if (cell.Controls.Count > 0 && cell.Controls[0] is DropDownList)
        {
            return (DropDownList)cell.Controls[0];
        }
        else
        {
            // Handle the case where the control in the cell is not a DropDownList
            return null;
        }
    }
```

```
    private string GetSelectedValueFromDropDown(TableCell cell)
    {
        if (cell.Controls.Count > 0 && cell.Controls[0] is DropDownList)
        {
            DropDownList dropDownList = (DropDownList)cell.Controls[0];
            return dropDownList.SelectedValue;
        }
        return string.Empty; // or handle the case where the control is not found
    }
    private void InsertDataIntoDatabase(string timeSlot, string dayOfWeek, string branch, string subjectName, string teacherName, string roomNo)

    {
        string connectionString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\gyang\\source\\repos\\try1\\try1\\App_Data\\StudentDB.mdf;Integrated Security=True";
        string query = "INSERT INTO Schedule (TimeSlot, DaysOfWeek, Branch, SubjectName, TeacherName, RoomNo) VALUES (@TimeSlot, @DaysOfWeek, @Branch, @SubjectName, @TeacherName, @RoomNo)";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            using (SqlCommand command = new SqlCommand(query, connection))
            {

                command.Parameters.AddWithValue("@TimeSlot", timeSlot);
```

```
            command.Parameters.AddWithValue("@DaysOfWeek", dayOfWeek);
            command.Parameters.AddWithValue("@Branch", branch);
            command.Parameters.AddWithValue("@SubjectName", subjectName);
            command.Parameters.AddWithValue("@TeacherName", teacherName);
            command.Parameters.AddWithValue("@RoomNo", roomNo);
            connection.Open();
            command.ExecuteNonQuery();
            connection.Close();
        }
    }
}
private bool IsTimetableExists(string branch)
{
    string connectionString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\gyang\\source\\repos\\try1
\\try1\\App_Data\\StudentDB.mdf;Integrated Security=True";
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        string query = "SELECT COUNT(*) FROM Schedule WHERE Branch = @Branch";
        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Branch", branch);
            int timetableCount = (int)command.ExecuteScalar();
            return timetableCount > 0;
        }
    }
```

```
    }
    protected void UpdateButton_Click(object sender, EventArgs e)
    {
        string branchToUpdate = UpdateHiddenField.Value;
        try
        {
            // Check if the timetable for the selected branch already exists in the database
            if (IsTimetableExists(branchToUpdate))
            {
                // Display a confirmation pop-up
                string confirmScript = "if (confirm('The timetable for this branch already exists. Do you want to update it?')) { document.getElementById('" + UpdateHiddenField.ClientID + "').value = 'true'; }";
                ScriptManager.RegisterStartupScript(this, this.GetType(), "confirmUpdate", confirmScript, true);
                // Update the message label to inform the user
                UpdateMessageLabel.Text = "The timetable for this branch already exists. You can update it.";
            }
            else
            {
                // Update the message label to inform the user

                UpdateMessageLabel.Text = "No existing timetable found. You can create a new one.";
                // If the timetable doesn't exist, you can proceed with any other logic or display a message
            }
        }
```

```
        catch (Exception ex)
        {
            // Handle any exceptions
            ShowError("An error occurred while processing the update request.", ex);
        }
    }
    protected void ViewButton_Click(object sender, EventArgs e)
    {
        try
        {
            // Get the selected branch from the BranchDropdown control
            string branch = BranchDropdown.SelectedValue;

            // Redirect to ViewCertificates.aspx with the selected branch as a query parameter
            Response.Redirect("~/ViewCertificates.aspx?branch=" + Server.UrlEncode(branch));
        }
        catch (Exception ex)
        {
            // Handle any exceptions
            ShowError("An error occurred while processing the view request.", ex);
        }
    }
}
```

g. student_dashboard.aspx.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Reflection.Emit;
using System.Web.UI.WebControls;

namespace try1
{
   public partial class Student_dashboard : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {

         if (!IsPostBack)
         {
            PopulateBranchDropdown();
         }
         {
            // Check if the user is logged in
            if (Session["IsLoggedIn"] != null && Convert.ToBoolean(Session["IsLoggedIn"]))
            {
               // Retrieve user details from the session
               string userEmail = Session["UserEmail"].ToString();
               string userName = Session["UserName"].ToString();

               // Display a personalized greeting
```

```
            Label2.Text = $"Hello {userName} ({userEmail})!";
        }
        else
        {
            // Redirect the user to the login page if not logged in
            Response.Redirect("login.aspx");
        }

        // Code to be executed only on the initial page load
        // Set the text of a label based on the current date
    }
}


    private void PopulateBranchDropdown()
{
    using (SqlConnection connection = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\gyang\source\repos\try1\try1\A
pp_Data\StudentDB.mdf;Integrated Security=True"))
    {
        string query = "SELECT Branch FROM Teachers";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            connection.Open();

            using (SqlDataReader reader = command.ExecuteReader())
```

```
        {
            DropDownList1.Items.Clear();

            while (reader.Read())
            {
                string branchName = reader["Branch"].ToString();
                DropDownList1.Items.Add(branchName);
            }
        }
    }
  }
}

protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
  PopulateTimetableFromDatabase(DropDownList1.SelectedValue);
}

protected void SubmitButton_Click(object sender, EventArgs e)
{
  // Upon clicking the Submit button, repopulate the timetable based on the selected branch
  PopulateTimetableFromDatabase(DropDownList1.SelectedValue);
}

private void PopulateTimetableFromDatabase(string selectedBranch)
{
  using (SqlConnection connection = new SqlConnection(@"Data
```

```
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\gyang\source\repos\try1\try1\App_Data\StudentDB.mdf;Integrated Security=True"))
        {
            string query = "SELECT TimeSlot, DaysOfWeek, SubjectName, TeacherName, RoomNo FROM Schedule WHERE Branch = @Branch";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Branch", selectedBranch);

                connection.Open();

                SqlDataAdapter adapter = new SqlDataAdapter(command);
                DataTable dataTable = new DataTable();
                adapter.Fill(dataTable);

                // Clear existing data in the timetable table
                TimetableTable.Rows.Clear();

                // Add header row
                TableHeaderRow headerRow = new TableHeaderRow();
                headerRow.Cells.Add(new TableHeaderCell { Text = "Timing" });
                headerRow.Cells.Add(new TableHeaderCell { Text = "Monday" });
                headerRow.Cells.Add(new TableHeaderCell { Text = "Tuesday" });
                headerRow.Cells.Add(new TableHeaderCell { Text = "Wednesday" });
                headerRow.Cells.Add(new TableHeaderCell { Text = "Thursday" });
                headerRow.Cells.Add(new TableHeaderCell { Text = "Friday" });
```

```
            headerRow.Cells.Add(new TableHeaderCell { Text = "Saturday" });
            TimetableTable.Rows.Add(headerRow);


            // Define time slots
            string[] timeSlots = { "7:45 AM - 8:30 AM", "8:30 AM - 9:15 AM", "9:15 AM - 10:00 AM",
"10:00 AM (Break) 10:20 AM", "10:20 AM - 11:05 AM", "11:05 AM - 11:55 AM", "11:55 AM - 12:35 PM"
};

            // Loop through each time slot
            foreach (string timeSlot in timeSlots)
            {
               TableRow dataRow = new TableRow();
               dataRow.Cells.Add(new TableCell { Text = timeSlot });

               // Loop through each day of the week
               for (int i =1; i < 7; i++)
               {
                  string dayOfWeek = ((DayOfWeek)i).ToString();
                  string cellContent = GetCellContent(dataTable, selectedBranch, dayOfWeek,
timeSlot);
                  dataRow.Cells.Add(new TableCell { Text = cellContent });
               }

               // Add the row to the timetable table
               TimetableTable.Rows.Add(dataRow);
            }
```

```
            }
        }
    }

    private string GetCellContent(DataTable dataTable, string branch, string dayOfWeek, string
timeSlot)
    {
        DataRow[] rows = dataTable.Select($"DaysOfWeek = '{dayOfWeek}' AND TimeSlot =
'{timeSlot}'");

        if (rows.Length > 0)
        {
            string subject = rows[0]["SubjectName"].ToString();
            string teacher = rows[0]["TeacherName"].ToString();
            string room = rows[0]["RoomNo"].ToString();
            return $"Subject: {subject}<br/>Teacher: {teacher}<br/>Room: {room}";
        }
        else
        {
            return string.Empty;
        }
    }
  }
}
```

h. teacher_dashboard.aspx.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Reflection.Emit;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace try1
{
    public partial class teacher_dashboard : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                // Check if the user is logged in
                if (Session["IsLoggedIn"] != null && Convert.ToBoolean(Session["IsLoggedIn"]))
                {
                    // Retrieve user details from the session
                    string userEmail = Session["UserEmail"].ToString();
                    string userName = Session["UserName"].ToString();
                    Label2.Text = $"Hello {userName} ({userEmail})!";
                }
                else
                {
                    // Redirect the user to the login page if not logged in
```

```
            Response.Redirect("login.aspx");
        }
    }
}

protected void ShowTimetableButton_Click(object sender, EventArgs e)
{
    string teacherName = TeacherNameTextBox.Text.Trim();
    PopulateTimetableFromDatabase(teacherName);
}

private void PopulateTimetableFromDatabase(string teacherName)
{
    using (SqlConnection connection = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\gyang\source\repos\try1\try1\A
pp_Data\StudentDB.mdf;Integrated Security=True"))
    {
        string query = "SELECT TimeSlot, DaysOfWeek, SubjectName, RoomNo FROM Schedule
WHERE TeacherName = @TeacherName";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@TeacherName", teacherName);

            connection.Open();

            SqlDataAdapter adapter = new SqlDataAdapter(command);
```

```
        DataTable dataTable = new DataTable();
        adapter.Fill(dataTable);

        // Clear existing data in the timetable table
        ClearTimetable();

        // Loop through each row in the data table
        foreach (DataRow row in dataTable.Rows)
        {
            string timeSlot = row["TimeSlot"].ToString();
            string dayOfWeek = row["DaysOfWeek"].ToString();
            string subjectName = row["SubjectName"].ToString();
            string roomNo = row["RoomNo"].ToString();

            // Find the correct TableCell in the timetable table and update its text
            TableCell cell = FindTableCell(dayOfWeek, timeSlot);
            if (cell != null)
            {
                cell.Text = $"Subject: {subjectName}<br/>Room: {roomNo}";
            }
        }
    }
}

private TableCell FindTableCell(string dayOfWeek, string timeSlot)
{
```

```
        string controlId = $"_{dayOfWeek}_{timeSlot.Replace(" ", "_")}";
        Control control = FindControl(controlId);
        if (control is TableCell)
        {
            return (TableCell)control;
        }
        return null;
    }

    private void ClearTimetable()
    {
        // Loop through each row in the timetable table and clear the text of all cells except the first one
(timing)
        foreach (TableRow row in TimetableTable.Rows)
        {
            foreach (TableCell cell in row.Cells)
            {
                if (cell != row.Cells[0])
                {
                    cell.Text = "";
                }
            }
        }
    }
  }
}
```

5.3 Testing Approach:

- **Functional Testing:** Functional testing is performed to verify that the system meets the specified functional requirements. Test cases are designed based on user stories and use cases, covering all functionalities and features of the system. Functional tests ensure that users can perform tasks as intended and that the system behaves correctly under various scenarios.
- **Regression Testing:** Regression testing is carried out to ensure that new code changes or updates do not introduce unintended side effects or regressions in existing functionalities. Regression test suites are executed periodically, especially after making code modifications, to validate that the system's behavior remains consistent and stable.
- **Performance Testing:** Performance testing evaluates the system's responsiveness, scalability, and reliability under different load conditions. Tools such as Apache JMeter or Microsoft Visual Studio Load Test are used to simulate concurrent user activity and measure the system's performance metrics. Performance testing helps identify bottlenecks, optimize resource utilization, and ensure that the system can handle expected loads efficiently.
- **Security Testing:** Security testing is conducted to identify and mitigate potential vulnerabilities or security threats in the system. Various techniques, such as penetration testing, vulnerability scanning, and code review, are employed to assess the system's security posture. Security testing aims to safeguard sensitive data, prevent unauthorized access, and ensure compliance with security standards and regulations.

- **User Acceptance Testing (UAT):** UAT involves testing the system from an end-user perspective to validate that it meets the users' requirements and expectations. Real users or stakeholders participate in UAT sessions to perform tasks, provide feedback, and confirm that the system aligns with their needs. UAT ensures that the system is user-friendly, intuitive, and meets business objectives effectively.
- **Automated Testing:** Automated testing is utilized to streamline the testing process and improve efficiency. Automated test scripts are developed using frameworks like Selenium for web applications or Appium for mobile applications. Automated tests are executed regularly, enabling quick feedback on code changes and ensuring consistent test coverage across different functionalities.

A. **System Testing:** System testing is conducted to evaluate the system as a whole. It involves testing the entire system against the specified requirements to ensure that it meets the intended purpose. System testing for the system includes the following aspects:

- **Functionality Testing:** This involves testing the system's functionality to ensure that all features work according to the requirements.
- **Performance Testing:** Performance testing is conducted to assess the system's performance under various conditions, such as load, stress, and scalability.
- **Compatibility Testing:** Compatibility testing ensures that the system is compatible with different browsers, operating systems, and devices.
- **Usability Testing:** Usability testing evaluates the system's user interface and overall user experience to ensure it is intuitive and user-friendly.

5.3.1) Unit Testing:

Unit testing is a crucial part of the software development process, aiming to validate the individual components or units of code. In the context of your website project, unit testing would involve testing the various methods and functions that make up the functionality of your web application.

For example, in the faculty_dashboard.aspx.cs file, you might have methods for populating dropdown lists, fetching data from the database, handling user input, and so on. Each of these methods would be a unit that needs to be tested.

To perform unit testing, you would use a unit testing framework like NUnit or MSTest. You would write test cases for each method, covering different scenarios and edge cases. For instance, you might write a test case to check if the PopulateBranchDropdown method correctly populates the branch dropdown list from the database. Another test case might check if the InsertDataIntoDatabase method inserts data into the database correctly.

Unit testing helps ensure that each unit of code behaves as expected in isolation, facilitating easier debugging and maintenance. It also helps identify and fix issues early in the development process, leading to a more robust and reliable application.

5.3.2) Integrated Testing:

Integrated testing focuses on testing the interaction between different modules or components of the system. In the context of your website project, integrated testing would involve testing how different parts of your web application work together.

For example, you might have a form page (create_table.aspx) where users can input data, and a backend method (InsertDataIntoDatabase) that processes this data and inserts it into the database. Integrated testing would ensure that the form page correctly communicates with the backend method and that data is inserted into the database as expected.

To perform integrated testing, you would write test cases that simulate user interactions with the application. For instance, you might write a test case to simulate a user filling out the form and submitting it, then checking if the data is correctly inserted into the database.

Integrated testing helps ensure that different parts of your application integrate seamlessly and work together as intended. It helps identify issues related to communication between modules, data flow, and overall system behavior, ensuring that your web application functions correctly as a whole.

By adopting a comprehensive testing approach encompassing unit testing, integrated testing, functional testing, regression testing, performance testing, security testing, user acceptance testing, and automated testing, the system is thoroughly evaluated for quality, reliability, and compliance with requirements. This multifaceted testing strategy ensures that the system delivers a seamless user experience, performs optimally, and maintains high standards of performance and security.

CHAPTER 6: CONCLUSION & FUTURE WORK

Conclusion:

In conclusion, this website represents a significant advancement in academic management, offering a comprehensive solution to various challenges faced by students and faculty. By providing a centralized platform for event participation, timetable management, and certificate uploads, the website has greatly simplified these processes, making them more efficient and accessible.

Future Planning:

Looking ahead, there are several key areas for future enhancement. One important aspect is the implementation of a self-service profile management system for students. This will allow students to update their information, track their academic progress, and manage their schedules more effectively.

Another key area for improvement is the implementation of an online attendance system for teachers. This will streamline the process of taking attendance, reduce paperwork, and provide real-time data on student attendance patterns.

Additionally, the website could benefit from the integration of an online grade system. This would allow teachers to enter grades online, making it easier for students to track their progress and for parents to stay informed about their child's performance.

Finally, implementing an online fee payment system would further streamline administrative processes and make it easier for students to pay their fees on time.

Overall, these future enhancements will further improve the functionality and usability of the website, making it an invaluable tool for students, faculty, and administrators alike.

CHAPTER 7: REFERENCE

★ GUIDANCE FROM OMKAR SIR.

★ "Entity Relationship Diagram (ERD) Tutorial." Lucidchart, https://www.lucidchart.com/pages/er-diagrams

★ “Storing the data of Time-Table”: YouTube..

★ "Data Encryption and Decryption Using AES Algorithm in C#." C# Corner, https://www.c-sharpcorner.com/article/data-encryption-and-decryption-using-aes-algorithm-in-c-sharp/.

★ "Software Requirements Specification (SRS) Document." TutorialsPoint, https://www.tutorialspoint.com/sdlc/sdlc_software_requirements_specification.htm.

★ "Introduction to ASP.NET." Microsoft Docs, https://docs.microsoft.com/en-us/aspnet/.

★ "Introduction to Database Systems." W3Schools, https://www.w3schools.com/sql/.

★ "Introduction to Web Design." W3Schools, https://www.w3schools.com/html/.

★ "Introduction to Encryption." Tutorialspoint, https://www.tutorialspoint.com/cryptography/index.htm.

★ "Introduction to Decryption." Tutorialspoint, https://www.tutorialspoint.com/cryptography/index.htm.

!!…THANK YOU…!!