Project Title: Online Learning Management System (LMS)

TEAM ID : 6

MEMBERS NAME AND ID:

SUBHASH RADHAKRISHNAN - 1002169517

DURGASHREE HAKKINALU SOMASHEKARAIAH - 1002197918

VARSHITH KONDURU - 1002132051

SAIKIRAN REDDY PEDDAVOOTLA – 1002175284

Part 1: Project Initiation (15marks)

- **Project Proposal (5 marks): (Reference from Proposal Report)**

  The project involves the design, development, and implementation of an Online Learning Management System (LMS) similar to university course management platforms like Canvas. The LMS will provide a comprehensive solution for course management, student enrollment, assignment submission, grading, communication, and content sharing, with a focus on accessibility, security, and compliance with educational regulations.

  The **Online Learning Management System (LMS)** is a comprehensive digital platform designed to facilitate and streamline the administration, documentation, tracking, reporting, and delivery of educational courses and training programs. This project aims to create an LMS that mirrors the functionality of popular University Course Management Systems, such as Canvas and Blackboard, while introducing additional features that cater to the evolving needs of modern educational environments.

  The primary objective of this project is to develop a robust, user-friendly system that enhances the teaching and learning experience for both instructors and students. The LMS will support all core functions typically associated with course management systems, including course creation, student enrollment, assignment submission, grading, communication, assessments, and progress tracking. It will also incorporate advanced features such as mobile accessibility, integration with third-party content management systems (CMS), and compliance with global data privacy and security regulations.

  Instructors will be empowered with tools to create and manage courses, upload and organize multimedia content, assign and grade assignments, administer quizzes and exams, and track student progress with detailed reports. Students, on the other hand, will have access to a seamless learning experience, where they can enroll in courses, submit assignments, view grades, interact with peers and instructors, and monitor their own learning journey.

  The LMS will be designed with scalability in mind, allowing it to serve educational institutions of various sizes, from small training organizations to large universities. To ensure high performance and security, the system will leverage modern web technologies and cloud infrastructure, employing secure login systems, role-based access controls, encrypted communication channels, and regular data backups.

  Project scope:

  The scope of the **Online Learning Management System (LMS)** project outlines the boundaries, features, deliverables, and constraints involved in the successful development and implementation of the system. It ensures that all stakeholders have a clear understanding of the project's objectives and the work involved to meet them.

  In-Scope Features and Deliverables:

  1. **User Authentication and Role Management**:

     a. Secure, multi-tiered login system supporting different user roles (students, instructors, administrators).

     b. Role-based access control (RBAC) to restrict permissions and ensure security.

     c. Password management and account recovery features (e.g., password reset, multi- factor authentication).

2. **Course Creation and Management**:
    a. Tools for instructors to create, edit, and manage courses.
    b. Support for organizing content into modules, lessons, or units.
    c. Options to upload a variety of content types such as documents, videos, and presentations.
    d. Ability to set up prerequisites, learning paths, and schedules.
3. **Student Enrollment and Tracking**:
    a. Automated enrollment based on predefined criteria (e.g., registration, course prerequisites).
    b. Manual enrollment options for administrators and instructors.
    c. Tracking of student progress, course completion status, and attendance.
    d. Activity logging for both students and instructors for audit purposes.
4. **Assignment Submission and Grading**:
    a. Submission portals for assignments supporting multiple file formats (e.g., PDFs, Word documents, videos).
    b. Integrated grading system with customizable rubrics, feedback mechanisms, and an instructor-managed gradebook.
    c. Plagiarism detection integration for submitted assignments.
5. **Communication Tools**:
    a. In-platform messaging for direct communication between students and instructors.
    b. Class-wide announcements visible to all enrolled students.
    c. Discussion forums for collaborative learning and peer interactions.
    d. Real-time chat rooms for virtual office hours or group study sessions.
6. **Assessment and Quizzes**:
    a. Tools for creating quizzes, exams, and surveys.
    b. Support for multiple question types (e.g., multiple choice, true/false, essay).
    c. Auto-grading capabilities for objective questions (e.g., multiple choice).
    d. Secure exam administration with time limits and question randomization.
7. **Content Management System (CMS) Integration**:
    a. Integration with external CMS platforms such as Google Drive, YouTube, and

OneDrive.

b. Drag-and-drop file management for easy uploading, organizing, and sharing of course materials.

c. Multimedia support for images, audio files, and video streaming within course content.

8. **Reporting and Analytics**:

   a. Instructor and administrator dashboards to monitor course performance, student engagement, and outcomes.

   b. Exportable reports showing metrics like course completion rates, student grades, quiz performance, and overall engagement.

   c. Data visualization tools (e.g., charts and graphs) for tracking student progress over time.

9. **Mobile Accessibility**:

   a. Fully responsive web design ensuring access across all device types (desktops, tablets, smartphones).

   b. Dedicated mobile application for iOS and Android, allowing students and instructors to manage courses on-the-go.

   c. Offline access for certain features such as course materials and announcements, with data synchronization when back online.

10. **Security and Compliance**:

    a. Implementation of data encryption for secure communication and storage.

    b. Regular data backups and recovery plans to ensure system reliability.

    c. Compliance with data privacy regulations such as **FERPA** (Family Educational Rights and Privacy Act) for educational institutions and **GDPR** (General Data Protection Regulation) for global data protection.

    d. Adherence to **WCAG 2.1** accessibility standards to ensure that the LMS is usable by individuals with disabilities.

Out-of-Scope Features:

- Development of proprietary video conferencing tools (integration with third-party tools like Zoom, Microsoft Teams, etc. is in-scope).
- Features that cater to non-educational organizations or corporate training environments.
- Advanced artificial intelligence (AI)-driven personalization or adaptive learning algorithms.
- Non-English language support (unless specified as a future enhancement).
- Customizations for non-educational institutions (e.g., corporate training platforms).

Deliverables:

- A fully functional **Online Learning Management System** with the aforementioned in-scope features.
- Detailed documentation, including user guides for students, instructors, and administrators.
- Deployment of a mobile application on Android and iOS platforms.
- Integration of third-party tools for CMS, video conferencing, and plagiarism detection.
- Training and support for administrators and instructors on how to use the LMS efficiently.
- Regular updates and maintenance support for a specified period after deployment.

Project Constraints:

- **Time Constraint**: The project will be completed within a **16-week timeline**, with milestones scheduled at the end of each key phase (e.g., Planning, Design, Development, Testing, and Deployment).
- **Budget Constraint**: The project must remain within the allocated budget, with clear tracking of development, testing, and deployment costs.
- **Technical Constraints**: The LMS will be built using widely supported and reliable technologies (e.g., React, Node.js, MySQL) to ensure scalability and long-term support. Integration with existing platforms like Google Drive and third-party APIs will follow their specific constraints and usage limitations.

Assumptions:

- The users (students, instructors, and administrators) are familiar with basic online educational tools and systems.
- The infrastructure required for the LMS (servers, cloud storage, etc.) will be provided by the client or available through cloud service providers (e.g., AWS, Google Cloud).
- Adequate testing environments and tools will be available during the testing phase.
- All key stakeholders will be available to provide timely feedback during critical phases of the project, such as design review and user acceptance testing.

Exclusions:

- The system will not provide any form of hardware or physical classroom tools.
- Content creation (e.g., creating actual course materials) is the responsibility of the instructors and is not part of the system's automated processes.
- Legal compliance outside of the educational domain (e.g., corporate tax regulations, intellectual property not related to education) will not be addressed unless explicitly required by the client.

Objectives:

- **Enhance Educational Delivery:** Provide educators with a versatile platform to effectively deliver course content and engage with students.
- **Streamline Administrative Tasks:** Simplify processes such as enrollment, grading, and reporting, reducing administrative burdens.
- **Promote Student Engagement:** Offer interactive tools that foster collaboration, discussion, and active participation among learners.

- **Ensure Accessibility and Compliance:** Design the system to be accessible to all users, including those with disabilities, and comply with relevant legal and educational standards.

The **Online Learning Management System (LMS)** project scope covers all the necessary aspects for a fully functional platform that will meet the needs of modern educational institutions, ensuring a scalable, secure, and efficient system. The clear demarcation of in- scope and out-of-scope features will help prevent scope creep and ensure the project remains within the planned timeline and budget.

By developing this Online Learning Management System, we aim to create a dynamic and supportive educational environment that leverages technology to enhance learning experiences, making education more accessible, efficient, and engaging for both instructors and students.

PROJECT RESOURCES:

The successful development, deployment, and maintenance of the **Online Learning Management System (LMS)** will require a combination of technical, human, and financial resources. Below is a detailed breakdown of all the essential resources for the project.

1. Human Resources (Project Team)

The project will require a multi-disciplinary team with various skill sets to ensure all aspects of the LMS are successfully executed. Each team member will have specific roles and responsibilities:

- **Project Manager (PM)**:
    - Oversees the entire project lifecycle, manages timelines, resources, and risks.
    - Coordinates between different teams and stakeholders.
    - Ensures that the project stays within scope, budget, and schedule.
- **Lead Software Architect**:
    - Designs the system architecture, including front-end and back-end components.
    - Defines the technology stack and integration of third-party tools.
    - Ensures scalability, performance, and security are considered during the design phase.

- **Front-End Developers**:
    - They Develop the user interface (UI) for the LMS, ensuring a responsive and user-friendly experience.
    - They also implement the web-based front-end using technologies like **React**, **HTML5**, **CSS3**, and **JavaScript**.
    - Ensures the LMS is mobile-friendly and accessible.
- **Back-End Developers**:
    - They Develop the server-side logic and database interactions.
    - Build and manage APIs to connect the front-end with the database and third-party tools.
    - Can use technologies like **Node.js**, **Express**, and **MySQL** to handle data storage, user authentication, and other critical processes.
- **Mobile App Developers**:
    - They Develop and maintain native mobile applications for iOS and Android.
    - They also ensure that students and instructors can use the app for course management, submissions, and communication.
    - Can use technologies like **React Native** or **Flutter** for cross-platform development.
- **Database Administrator (DBA)**:
    - Designs and manages the database schema, ensuring efficient data storage and retrieval.
    - Handles backups, security, and database performance optimization.
    - Work with **MySQL** or another relational database to handle user data, courses, assignments, grades, etc.
- **QA/Test Engineers**:
    - Tests the LMS throughout the development cycle, ensuring functional correctness, usability, and security.
    - Performs automated and manual testing for both web and mobile platforms.
    - Conducts load testing, security testing, and user acceptance testing (UAT).
- **UI/UX Designers**:

- Design a clean, intuitive user interface with a focus on usability.
- Creates wireframes, mockups, and user flows.
- Ensures that the LMS follows accessibility standards such as **WCAG 2.1** and offers a smooth user experience.

- **Content Management Specialists**:
  - Ensures that the LMS integrates seamlessly with third-party CMS platforms like Google Drive and YouTube.
  - Assists instructors in organizing, uploading, and managing course materials.
- **Security Expert**:
  - Conducts security audits and ensure data privacy compliance (FERPA, GDPR).
  - Implements encryption protocols, secure APIs, and backup strategies.
  - Addresses potential vulnerabilities and prevent unauthorized access.
- **System Administrator**:
  - Manages the servers and cloud infrastructure needed for hosting the LMS.
  - Ensures uptime, performance monitoring, and issue resolution.
  - Manages domain registration, SSL certificates, and load balancing.
- **Technical Support Staff**:
  - Provides post-deployment support to instructors, students, and administrators.
  - Resolves technical issues and help with system navigation.

2. **Technology Resources**
   a. **Development Tools**:
      - **Visual Studio Code**, **Eclipse**, or similar Integrated Development Environments (IDEs) for coding.
        i. **Git** and **GitHub/GitLab** for version control and collaboration.
        ii. **JIRA** or **Trello** for task management and sprint planning.

   b. **Front-End Development**:
      i. **React.js** for building the web interface.

- **HTML5**, **CSS3**, **JavaScript** (ES6+), **Bootstrap** or **Material UI** for designing responsive interfaces.

ii. **React Native** or **Flutter** for mobile application development.

c. **Back-End Develo**

i. **Node.js** and **Express** for server-side development.

ii. **RESTful APIs** for communication between front-end and back-end.
- **OAuth2.0** or **JWT** (JSON Web Tokens) for secure user authentication and authorization.

d. **Database**:

i. **MySQL** or **PostgreSQL** for relational data storage.

ii. **MongoDB** or **Redis** for any NoSQL requirements or caching mechanisms.

iii. Database management tools like **phpMyAdmin** or **MySQL Workbench**.

e. **Cloud Infrastructure and Hosting**:

- **AWS** (Amazon Web Services), **Google Cloud**, or **Microsoft Azure** for hosting the LMS, ensuring scalability and performance.

i. Cloud-based services for storage, database management, and backups.
- **Load balancing** tools like AWS Elastic Load Balancer (ELB) for high availability.

f. **Mobile Development Tools**:

i. **Android Studio** for Android app development.

ii. **Xcode** for iOS app development.

iii. **React Native CLI** for cross-platform mobile development.

g. **Third-Party Integrations**:

i. **Google Drive**, **OneDrive**, and **Dropbox** for content management integration.

ii. **YouTube** API for embedding multimedia content into courses.

iii. **Zoom** or **Microsoft Teams** for video conferencing integration.

iv. **Turnitin** or **Grammarly** for plagiarism detection in assignments.

h. **Security Tools**:

i. **SSL Certificates** for encrypted communication.
   - **AWS IAM** (Identity and Access Management) for controlling access to AWS resources.

ii. **Firewall** configurations for server protection.

iii. **Backup** services and Disaster Recovery tools for data protection.

i. **Testing Tools**:

i. **Selenium** for automated testing of the web application.

ii. **Appium** for mobile app testing.

iii. **JMeter** for load and performance testing.

iv. **SonarQube** for static code analysis and security vulnerability detection.

3. Financial Resources (Budget)

The project will require funding for various activities, which can be categorized as follows:

a. Human Resource Costs:

- Salaries for developers, project managers, designers, testers, and support staff over the project's timeline.
- Consultants for specific roles such as security experts or UI/UX designers (if outsourced).

b. **Infrastructure and Technology Costs**:

- **Cloud Hosting Services** (AWS, Google Cloud, etc.) – Monthly fees for cloud infrastructure, database hosting, and content storage.
- **Development Tools** – Subscription costs for premium versions of tools like JIRA, GitHub, or testing frameworks.
- **Third-Party API Licensing** – Licensing fees for integrations like Turnitin (plagiarism detection) or Zoom (video conferencing).

c. **Hardware Costs**:

i. Servers and backup devices for storing data if hosted locally.

ii. Hardware for testing (multiple mobile devices for cross-platform testing).

d. Security and Compliance Costs:

i. Purchase of **SSL certificates** and other security infrastructure.

- Annual compliance audits for **GDPR**, **FERPA**, and **WCAG** accessibility standards.

- **Maintenance and Support Costs**:
  - Costs associated with ongoing support post-deployment (including system updates, bug fixes, and user support).

4. Time Resources (Project Timeline)

The project is divided into several phases, and each phase will require different resources:

- **Planning**: PM, Lead Architect, and team will finalize project goals, timeline, and resource allocation.
- **Design**: UI/UX Designers and Software Architects will focus on creating mockups and system architecture.
- **Development**: Front-end, back-end, mobile developers, and DBAs will be the primary resources.
- **Testing**: QA Engineers and Testers will validate the system's functionality and security.
- **Deployment**: System Admins and DevOps team will handle the deployment to the cloud.
- **Support**: Technical Support will handle end-user training and troubleshooting.

Other Resources

- **Training Resources**:
  - Instructional guides, manuals, and video tutorials for instructors, students, and administrators.
  - Training sessions for key personnel on how to use the LMS effectively.
- **Documentation**:
  - Comprehensive documentation for system architecture, API integrations, and database schemas.
  - User guides for various roles (instructors, students, administrators).
  - By allocating these resources efficiently, the **Online Learning Management System** project will have the necessary tools, skills, and infrastructure to ensure successful development, deployment, and support.

Desired Requirements/Constraints:

Functional Requirements:

1. **User Authentication**: Secure login and role-based access for students, instructors, and administrators.
2. **Course Creation and Management**: Instructors should be able to create, modify, and manage course content, including uploading materials and assignments.
3. **Assignment Submission and Grading**: Students should submit assignments online, and instructors should provide feedback and grades.

4. **Communication Tools**: A messaging system and discussion forum to facilitate instructor-student and peer-to-peer communication.
5. **Reporting and Analytics**: Instructors and administrators should have access to dashboards for performance monitoring and data-driven insights.
6. **Mobile Compatibility**: The system should support access from mobile devices.

Non-Functional Requirements:

1. **Scalability**: The system should be able to handle an increasing number of users as more courses and students are added.
2. **Security**: The platform must ensure data protection and privacy, including encrypted communication and secure storage of user information.
3. **Performance**: The system should load quickly, support high concurrency, and offer real-time performance for data updates and interactions.
4. **User Experience (UX)**: A user-friendly and intuitive interface for all types of users, with minimal training required.

Constraints:

1. **Budget**: The total estimated budget for the project is $89,000, including development, testing, and deployment.
2. **Time**: The project has a timeline of approximately 4 months, with an expected completion date in **December 2024**.
3. **Technology Stack**: The LMS will be built using **React** for the front-end, **Node.js** for the back-end, and **MongoDB** for the database.
4. **Maintenance**: Post-deployment maintenance will be limited to bi-annual updates due to budget constraints.

Software Development Life Cycle (SDLC) Model:

The **Agile SDLC model** is best suited for this project due to its flexibility and iterative approach, which allows continuous improvements through sprint cycles.

- Rationale for Choosing Agile:
    - **Iterative Development:** The LMS is complex with multiple modules that can be developed and tested in short iterations (sprints).
    - **Customer Feedback:** Agile allows for frequent feedback from stakeholders, ensuring the product meets end-user expectations.
    - **Risk Management:** Agile's iterative nature allows for continuous risk identification and mitigation.
    - **Scalability:** Agile supports gradual expansion of the system's features, allowing for future enhancements.

- For the Online Learning Management System (LMS) project, we are adopting the **Agile Software Development Lifecycle (SDLC) model**. The Agile methodology is the most suitable SDLC model for this project due to the following reasons:

Why Agile SDLC Model?

1. **Iterative Development:**
   - Agile follows an iterative approach, allowing for incremental feature development in small sprints. This fits perfectly with the LMS project, which has distinct features such as authentication, course management, assignment submission, grading systems, and communication tools that can be developed in phases. Each sprint focuses on completing a set of features, allowing the project to evolve with stakeholder feedback.
2. **Flexibility and Adaptability:**
   - The scope of the LMS may evolve as instructors, students, and other stakeholders provide input. Agile's flexibility allows changes to be accommodated within the scope without causing delays or disruption, as modifications can be implemented in subsequent sprints.
3. **User-Centric Development:**
   - Since the LMS will cater to multiple user types (students, instructors, administrators), Agile's user-centric approach is beneficial. Regular reviews and feedback from actual users after each sprint ensure that the system aligns with their needs and expectations.
4. **Continuous Testing and Integration:**
   - With Agile, testing is integrated into every sprint. This means that bugs or issues will be identified and fixed during the development process, reducing the risk of major issues at the end of the project. Each feature, such as the grading system or mobile accessibility, will be continuously tested and refined.
5. **Timely Deliverables and Early ROI:**
   - By releasing functional components after each sprint, stakeholders can start using and benefiting from key features like user authentication or course management even before the entire LMS is completed. This delivers early value to the users and keeps the project on track.

Sprint Backlog Details

Sprint 1: Set up Authentication System and User Roles:

- **Goal:** Develop and implement a secure authentication system and user role management.
- **Tasks:**

- Set up user authentication system (email/password, OAuth).
- Implement user roles (admin, student, instructor).
- User access control setup.

- **Duration:** 10 days.

Sprint 2: Develop Course Creation and Management Features:

- **Goal:** Enable instructors to create, manage, and organize courses.
- **Tasks:**
  - Develop course creation tools.
  - Implement content organization features (modules, units, lessons).
  - Develop course content upload and editing features.
- **Duration:** 15 days.

Sprint 3: Integrate Assignment Submission and Grading Systems:

- **Goal:** Provide a platform for students to submit assignments and for instructors to grade them.
- **Tasks:**
  - Develop assignment submission system.
  - Integrate multiple file format support (PDF, doc, etc.).
  - Build a grading system with rubrics and feedback features.
- **Duration:** 15 days.

Sprint 4: Implement Communication Tools and Content Management Integration:

- **Goal:** Provide tools for interaction between students and instructors and enable content sharing.
- **Tasks:**
  - Develop messaging and notification systems.
  - Implement discussion forums.
  - Integrate third-party content management systems (Google Drive, YouTube).
- **Duration:** 15 days.

Sprint 5: Implement Reporting, Analytics, and Mobile Accessibility:

- **Goal:** Provide data insights and ensure mobile accessibility.
- **Tasks:**

    - Develop dashboards for student performance and course analytics.
    - Build exportable reports for administrators.
    - Ensure responsive design for mobile devices and implement offline access.
- **Duration:** 10 days.

Sprint 6: Testing, Bug Fixing, and Final Deployment:

- **Goal:** Test all features, fix any bugs, and prepare the system for deployment.
- **Tasks:**
    - Conduct system-wide testing (unit, integration, and regression tests).
    - Address any discovered bugs.
    - Finalize deployment process and release the LMS to users.
- **Duration:** 15 days.

By using Agile with defined sprints, we ensure that each feature is developed, tested, and delivered in a structured yet flexible manner. The sprint backlog helps track and allocate tasks efficiently, ensuring project success while adapting to any changes in requirement

Cost and **time** estimation (10 marks): **(Reference from Proposal Report)**

For the Online Learning Management System (LMS) project, we'll break down costs into several categories, including labor, software tools, infrastructure, and miscellaneous costs. Following is a thorough estimation:

1. **Labor Costs**

Labor costs typically account for the largest portion of the project budget. We'll estimate based on the roles involved and the hours worked.

Role	**Hourly Rate (USD)**	**Estimated Hours**	**Total Cost (USD)**
Project Manager	$60	200	$12,000
UI/UX Designer	$50	150	$7,500
Front-End Developer	$55	300	$16,500
Back-End Developer	$60	350	$21,000
Database Developer	$55	250	$13,750
Security Specialist	$70	150	$10,500
QA/Testers	$45	200	$9,000
Technical Support/Trainer	$40	100	$4,000
Total Labor Cost			**$94,250**

2. **Software and Tools**

This includes licenses for any third-party software tools, cloud platforms, or integrations necessary for development, testing, and deployment.

Software/Tool	**Cost (USD)**
Design Tools (Figma, Sketch)	$1,000 (Annual Subscription)
Development Tools (Visual Studio Code, GitHub)	$500 (Miscellaneous Expenses)
Cloud Hosting (AWS, Azure)	$3,000 (for 6 months)
Database Licenses (MySQL, MongoDB)	$1,500
Testing Tools (Selenium, Postman)	$1,200
Total Software Cost	**$7,200**

3. Infrastructure Costs

Includes hosting, server setup, security tools, and other infrastructure requirements.

Infrastructure	**Cost (USD)**
Cloud Infrastructure (Servers, Storage)	$5,000 (for 6 months)
Security Implementation (SSL, Firewalls)	$2,500
Backup and Disaster Recovery Setup	$1,500
Total Infrastructure Cost	**$9,000**

4. Miscellaneous Costs

Additional costs for training, documentation, and any unexpected expenses.

Miscellaneous	**Cost (USD)**
User Training Materials	$1,000
Documentation and Manuals	$800
Contingency (10% of total costs)	$10,025
Total Miscellaneous Cost	**$11,825**

5. Total Project Cost Estimation

Now, summing up all categories:

Category	**Total Cost (USD)**
Labor	$94,250
Software and Tools	$7,200
Infrastructure	$9,000
Miscellaneous	$11,825
Total Project Cost	**$122,275**

The estimated total cost for the LMS project is **$122,275**. This covers labor, tools, infrastructure, and miscellaneous expenses, ensuring the project is delivered within budget and meets quality standards.

Below is a detailed **Function Point (FP) estimation** and the **Adjusted Function Point (AFP) calculation** for the LMS project, we'll follow a structured approach. These calculations are typically made as follows:

1. Function Point Estimation:-

The Function Point (FP) estimation involves identifying and classifying the functionalities into five major categories:

1. **External Inputs (EI)**: User inputs or data that is processed and stored in the system (e.g., course creation, assignment submission).
2. **External Outputs (EO)**: Data or reports generated and sent to the user (e.g., grade reports, progress tracking).
3. **External Inquiries (EQ)**: User-initiated requests for information (e.g., viewing course materials, student queries).
4. **Internal Logical Files (ILF)**: Data maintained by the system (e.g., user data, course information, assignments).
5. **External Interface Files (EIF)**: Interactions with external systems (e.g., integration with CMS, external databases).

Step 1: Classify and Weight Each Function

Function Type	Number of Functions (Estimated)	Complexity (Low, Medium, High)	Weighting Factor	Total Function Points
External Inputs (EI)	10	Medium	4	40
External Outputs (EO)	8	Medium	5	40
External Inquiries (EQ)	7	Low	3	21
Internal Logical Files (ILF)	6	High	10	60
External Interface Files (EIF)	4	Medium	7	28
Total Unadjusted Function Points (UFP)				**189**

Therefore, the **Unadjusted Function Points (UFP)** for the LMS project is **189**.

2. Calculate Complexity Adjustment Factor (CAF):-

To calculate the **Adjusted Function Points (AFP)**, we will first determine the **Complexity Adjustment Factor (CAF)**. This is done by assessing 14 General System Characteristics (GSCs) such as system reliability, data communication, and user friendliness. Each GSC is rated on a scale from 0 to 5 (0 = no influence, 5 = strong influence).

Here are some common GSCs and ratings for an LMS system:

General System Characteristic (GSC)	Rating (0-5)
Data communications	4
Distributed data processing	3
Performance	4
Heavily used configuration	3
Transaction rate	4
Online data entry	5
End-user efficiency	4
Online update	5
Complex processing	4
Reusability	3
Installation ease	2
Operational ease	3
Multiple sites	3
Facilitate change	4

Step 2: Calculating the Total Degree of Influence (DI)

The total **Degree of Influence (DI)** is the sum of all GSC ratings.

$$DI = 4 + 3 + 4 + 3 + 4 + 5 + 4 + 5 + 4 + 3 + 2 + 3 + 3 + 4 = 51$$

Step 3: Calculating the Complexity Adjustment Factor (CAF)

The **CAF** formula is:

CAF = 0.65 + (0.01×DI) CAF = 0.65 + (0.01×51)

= 0.65 + 0.51

CAF = 1.16

3. **Calculate Adjusted Function Points (AFP):-**

Finally, the **Adjusted Function Points (AFP)** are calculated as follows:

AFP=UFP×CAF AFP=189×1.16=219.24

Thus, the **Adjusted Function Points (AFP)** for the LMS project is approximately **219**.

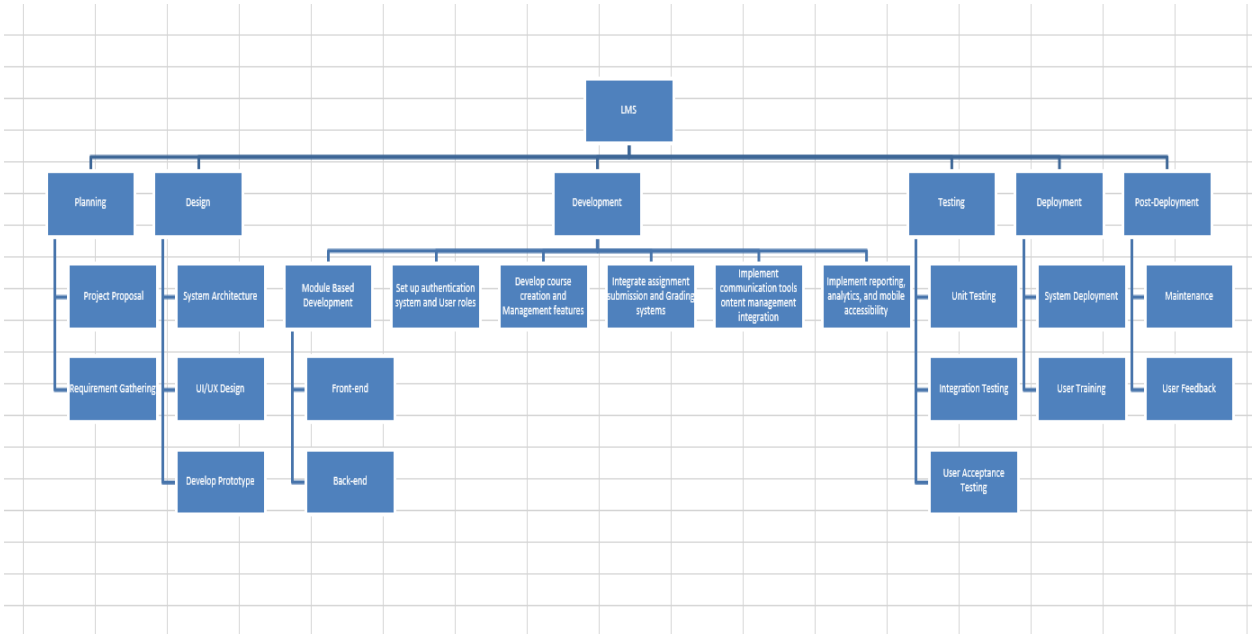
Summary

- **Unadjusted Function Points (UFP)**: 189
- **Degree of Influence (DI)**: 51
- **Complexity Adjustment Factor (CAF)**: 1.16
- **Adjusted Function Points (AFP)**: 219

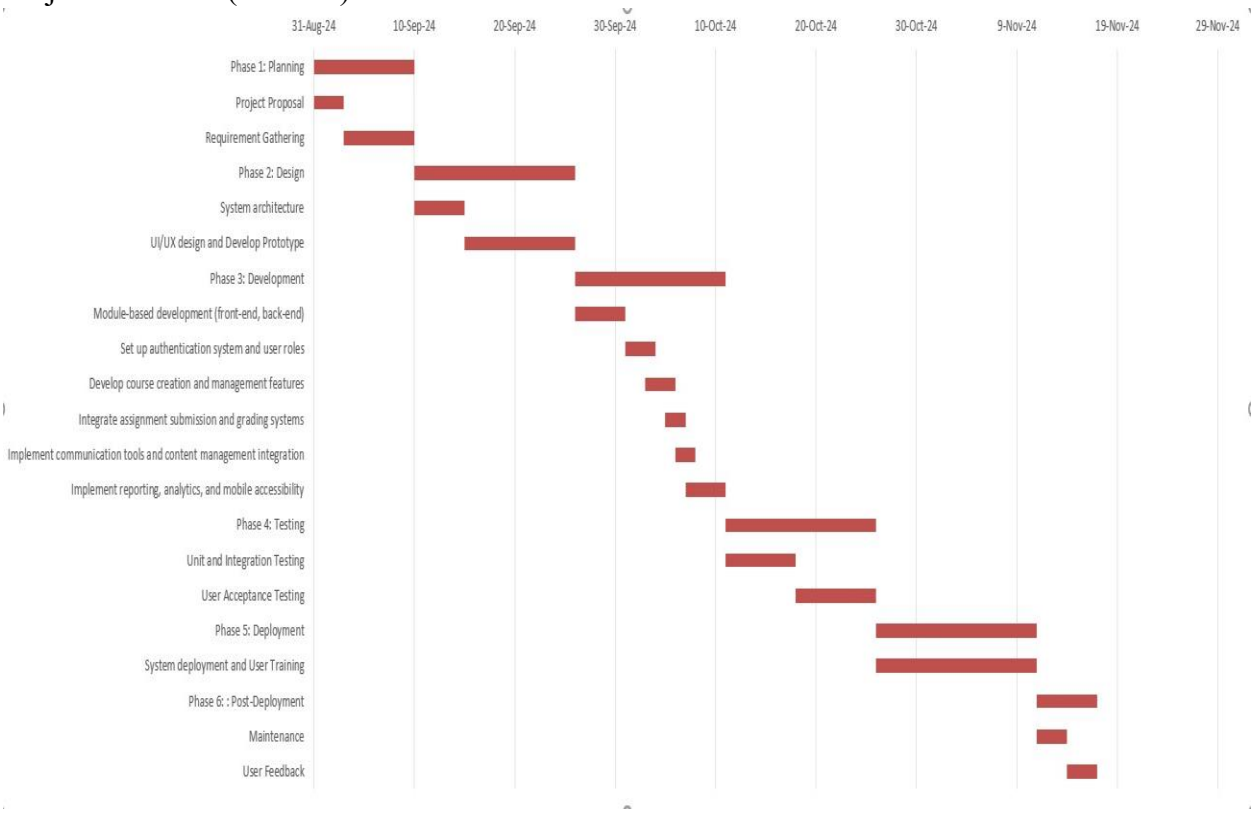
The **AFP** helps in refining the project cost, timeline, and resource estimations by considering the complexity and system characteristics of the LMS project.

Part 2: Project Planning (10 marks). (Reference from Proposal Report)

- Create a WBS for the project. Include at least three levels of decomposition. (5 marks)



- Project Schedule (5 marks): Gnatt Chart



Part 3: Risk Management (5 marks) .(Reference from Proposal Report)

1. **Risk Identification:**

The first step in risk management is identifying all potential risks that could affect the project. These risks are categorized into technical, project management, operational, and external risks.

1.1 Technical Risks:-

- **Risk 1: Security Vulnerabilities**
  - **Description:** The LMS stores sensitive information such as student data, course content, and grades. Any security breach could lead to data theft.
  - **Likelihood:** High
  - **Impact:** Critical
- **Risk 2: Integration Failures**
  - **Description:** The LMS needs to integrate with third-party tools (e.g., CMS, Google Drive). If integrations fail, system functionality will be compromised.
  - **Likelihood:** Medium
  - **Impact:** High
- **Risk 3: Poor System Performance**
  - **Description:** The LMS may encounter slow load times or crashes during peak usage.
  - **Likelihood:** Medium
  - **Impact:** High
- **Risk 4: Technology Obsolescence**
  - **Description:** Using outdated technologies can lead to compatibility and support issues in the long term.
  - **Likelihood:** Medium
  - **Impact:** Medium
- **Risk 5: Inadequate Testing**
  - **Description:** Insufficient time or effort spent on testing can lead to undetected bugs or performance issues in production.
  - **Likelihood:** Medium
  - **Impact:** High

1.2 Project Management Risks:-

- **Risk 6: Scope Creep**
  - **Description:** Additional features or modifications requested after the project starts may cause delays and

budget overruns.

- **Likelihood:** High
- **Impact:** High

- **Risk 7: Inaccurate Time Estimation**
  - **Description:** Incorrect timeline estimation can cause schedule overruns and affect project delivery.
  - **Likelihood:** Medium
  - **Impact:** High
- **Risk 8: Budget Overrun**
  - **Description:** Mismanagement of resources or unexpected expenses could lead to exceeding the project budget.
  - **Likelihood:** Medium
  - **Impact:** High

1.3 Organizational Risks:-

- **Risk 9: Resource Availability**
  - **Description:** Key personnel (developers, designers, etc.) might become unavailable during the project, leading to delays.
  - **Likelihood:** Medium
  - **Impact:** High

1.4 External Risks:-

- **Risk 10: Regulatory Compliance**
  - **Description:** The LMS must comply with educational data regulations (e.g., FERPA, GDPR). Non-compliance could result in legal penalties.
  - **Likelihood:** Medium
  - **Impact:** Critical
- **Risk 11: Vendor Lock-in**
  - **Description:** Reliance on a particular cloud service provider could result in high costs or limited flexibility.
  - **Likelihood:** Low
  - **Impact:** Medium
- **Risk 12: User Adoption Resistance**
  - **Description:** Instructors and students might resist adopting the new LMS, reducing its effectiveness.
  - **Likelihood:** Medium
  - **Impact:** Medium

2. Risk Assessment:

The risks are assessed based on two factors:

1. **Likelihood:** The probability of the risk occurring (Low, Medium, High).
2. **Impact:** The severity of the effect on the project if the risk occurs (Low, Medium, High, Critical).

The **Risk Matrix** can help categorize these risks into priorities:

Impact\Likelihood	**Low**	**Medium**	**High**	**Critical**
Low	Accept	Monitor	Manage	Manage
Medium	Monitor	Manage	Mitigate	Mitigate
High	Manage	Mitigate	Mitigate	Avoid
Critical	Manage	Mitigate	Avoid	Avoid

3. Risk Mitigation Strategies:

Once the risks have been identified and assessed, it's crucial to develop mitigation strategies.

3.1 Security Vulnerabilities:-

- **Mitigation Strategy:** Implement data encryption (SSL), two-factor authentication (2FA), regular security audits, and firewalls. Conduct vulnerability assessments and penetration testing.
- **Contingency Plan:** In case of a security breach, have an incident response team in place to investigate and patch vulnerabilities.

3.2 Integration Failures:-

- **Mitigation Strategy:** Perform rigorous testing of third-party APIs and have fallback options if integrations fail. Establish clear communication with vendors for integration support.
- **Contingency Plan:** Use alternative tools or APIs if primary integrations fail.

3.3 Poor System Performance:-

- **Mitigation Strategy:** Implement load balancing, scalable cloud infrastructure (AWS, Azure), and optimize database queries. Monitor performance metrics regularly.
- **Contingency Plan:** Increase server capacity during peak times or switch to a more robust cloud infrastructure.

3.4 Scope Creep:-

- **Mitigation Strategy:** Use strict project change control processes. Clearly define the project scope at the beginning and document any changes in the project log.
- **Contingency Plan:** Reassess timelines and reallocate resources if changes are approved.

3.5 Inadequate Testing:-

- **Mitigation Strategy:** Allocate sufficient time and resources for thorough testing. Use automated testing tools and have dedicated QA testers for manual testing.
- **Contingency Plan:** Extend the testing phase if critical bugs are found late in the process.

3.6 Inaccurate Time Estimation:-

- **Mitigation Strategy:** Use expert judgment and historical data to estimate timelines. Break down tasks into smaller work packages and monitor progress closely.
- **Contingency Plan:** Extend timelines for less critical features if necessary.

3.7 Budget Overrun:-

- **Mitigation Strategy:** Maintain a detailed budget tracking system and regularly monitor expenses. Set aside a contingency budget (10-15%) for unforeseen costs.
- **Contingency Plan:** Adjust scope or timelines to accommodate budget constraints.

3.8 Resource Availability:-

- **Mitigation Strategy:** Have backup resources (contractors or additional developers) ready to step in if required. Encourage knowledge sharing among team members to reduce dependency on specific individuals.

- **Contingency Plan:** Reassign tasks or extend deadlines if key personnel become unavailable.

3.9 Regulatory Compliance:-

- **Mitigation Strategy:** Ensure compliance with data protection regulations (FERPA, GDPR). Hire legal advisors to audit the system and ensure compliance.
- **Contingency Plan:** Implement immediate changes to the system if non-compliance issues are identified during development or post-launch.

3.10. Vendor Lock-in:-

- **Mitigation Strategy:** Use cloud providers that support multi-cloud deployment and avoid relying on proprietary APIs. Opt for open standards and tools.
- **Contingency Plan:** Develop strategies for migrating to another cloud provider if required.

3.11. User Adoption Resistance:-

- **Mitigation Strategy:** Provide thorough training and support materials for users. Run pilot programs to gather feedback and improve the system.
- **Contingency Plan:** Offer additional training sessions and encourage early adopters to promote the system to their peers.

4. Risk Monitoring:

To ensure ongoing risk management throughout the project lifecycle, regular risk reviews should be conducted. The following methods will be used:

- **Regular Risk Audits:** Review risks weekly during team meetings to track any changes in the

likelihood or impact.

- **Risk Log Updates:** Maintain a risk log that documents all identified risks, their status, and actions taken.
- **Risk Reporting:** Communicate high-priority risks to stakeholders regularly.

5. Risk Contingency Plans:

For critical risks e.g., security breaches, compliance failures, contingency plans will be developed to ensure the team can respond effectively if the risk materializes. These contingency plans should be tested during development to ensure they are effective in mitigating the impact.

6. Summary of Key Risks and Mitigation:

Here is a summary of the most critical risks and their mitigation strategies:

Risk	Likelihood	Impact	Mitigation Strategy
Security Vulnerabilities	High	Critical	Implement encryption, 2FA, and audits
Integration Failures	Medium	High	Test APIs, establish vendor support
Scope Creep	High	High	Implement strict change control
Budget Overrun	Medium	High	Track expenses, contingency budget
Inadequate Testing	Medium	High	Allocate sufficient testing resources
Regulatory Compliance	Medium	Critical	Ensure GDPR, FERPA compliance

This **Risk Management Plan** will ensure that the project team is aware of potential risks and has strategies in place to mitigate or avoid them, ensuring a smoother development process and successful project delivery.

Part 4: Project Quality Assurance (10 marks)

1. FUNCTIONALITY TESTING

1.1 User Authentication Scenarios

- **Account Creation Testing**
  - Verify user registration with valid information
  - Test email verification process
  - Validate password complexity requirements
  - Check duplicate account prevention
- **Login Process Testing**
  - Test login with valid credentials
  - Verify "Remember Me" functionality
  - Test password reset workflow
  - Validate multi-factor authentication (if implemented)

1.2 Course Management Scenarios

- **Course Creation**
  - Verify course setup with all required fields
  - Test multimedia content upload
  - Validate course scheduling features
  - Check course template functionality
- **Content Management**
  - Test different file type uploads
  - Verify content organization features
  - Validate content preview functionality
  - Test content version control

1.3 Assessment Features

- **Assignment Management**
  - Test assignment creation
  - Verify submission deadlines
  - Validate file submission types
  - Test grading system
- **Quiz/Test Features**
  - Verify various question types
  - Test time limit functionality
  - Validate automatic grading
  - Check result publication

2. USABILITY TESTING

2.1 Navigation Testing

- **User Interface Scenarios**
  - Verify intuitive menu structure
  - Test breadcrumb navigation
  - Check responsive design elements
  - Validate search functionality
- **Accessibility Scenarios**
  - Test screen reader compatibility
  - Verify keyboard navigation
  - Check color contrast ratios
  - Validate form field labels

2.2 Cross-Platform Testing

- **Device Compatibility**
  - Desktop browsers (Chrome, Firefox, Safari, Edge)
  - Mobile devices (iOS and Android)
  - Tablets and iPads
  - Different screen resolutions

3. RELIABILITY TESTING

3.1 Performance Under Load

- **Concurrent User Testing**
  - Test with 500 simultaneous users
  - Verify system behavior at peak loads
  - Test data processing speed
  - Monitor server response times

3.2 System Stability

- **Long-Duration Testing**
  - 24-hour continuous operation test
  - Memory leak detection
  - Database connection stability
  - Background task processing

4. SECURITY TESTING

4.1 Authentication Security

- **Access Control Testing**
  - Role-based access verification
  - Session management
  - Password security policies
  - Login attempt restrictions

4.2 Data Protection

- **Data Security Scenarios**
  - Encrypted data transmission
  - Secure file storage
  - Personal data protection
  - Backup and recovery

5. PERFORMANCE TESTING

5.1 Response Time

- **Speed Testing Scenarios**
  - Page load times (target: < 3 seconds)
  - File upload/download speeds
  - Search response times
  - API response times

5.2 Resource Utilization

- **System Resource Monitoring**
  - CPU usage tracking
  - Memory utilization
  - Network bandwidth usage
  - Database performance

6. QUALITY CONTROL PROCEDURES

6.1 Testing Methods

- **Automated Testing**
  - Unit testing for core functions
  - Integration testing for workflows
  - API testing for endpoints
  - Regression testing suite
- **Manual Testing**
  - Exploratory testing sessions
  - User acceptance testing
  - Bug verification
  - Feature validation

6.2 Quality Metrics

- **Performance Metrics**
  - Page load time < 3 seconds
  - API response time < 500ms
  - 99.9% uptime
  - Zero critical bugs in production
- **User Experience Metrics**
  - Task completion rate > 90%

- User satisfaction score > 4/5
- Support ticket resolution < 24 hours
- Feature adoption rate > 70%

Quality Planning:

- Testing Procedures
- Automated testing protocols
- Manual testing requirements
- Cross-platform testing standards
- Assessment Process
- Test documentation requirements
- Quality metrics tracking
- Reporting schedules
- Organizational Standards
- Security protocols and Performance benchmarks
- Accessibility requirements
- Device compatibility standards

7. CONTINUOUS IMPROVEMENT

7.1 Monitoring and Feedback

- **System Monitoring**
  - Real-time performance tracking
  - Error rate monitoring
  - User behavior analytics
  - Resource utilization tracking
- **User Feedback Collection**
  - In-app feedback forms
  - User surveys
  - Feature request tracking
  - Bug report analysis

7.2 Quality Assurance Schedule

- **Regular Testing Cycles**
  - Daily automated tests
  - Weekly security scans
  - Bi-weekly performance tests
  - Monthly accessibility audits

8. DOCUMENTATION AND REPORTING

8.1 Test Documentation

- **Required Documentation**
  - Test plans and cases

- Bug reports and tracking
- Performance test results
- Security audit reports

8.2 Quality Reports

- **Regular Reporting**
  - Daily test execution reports
  - Weekly quality metrics
  - Monthly performance trends
  - Quarterly quality assessments

9. RISK MITIGATION

9.1 Quality Risks

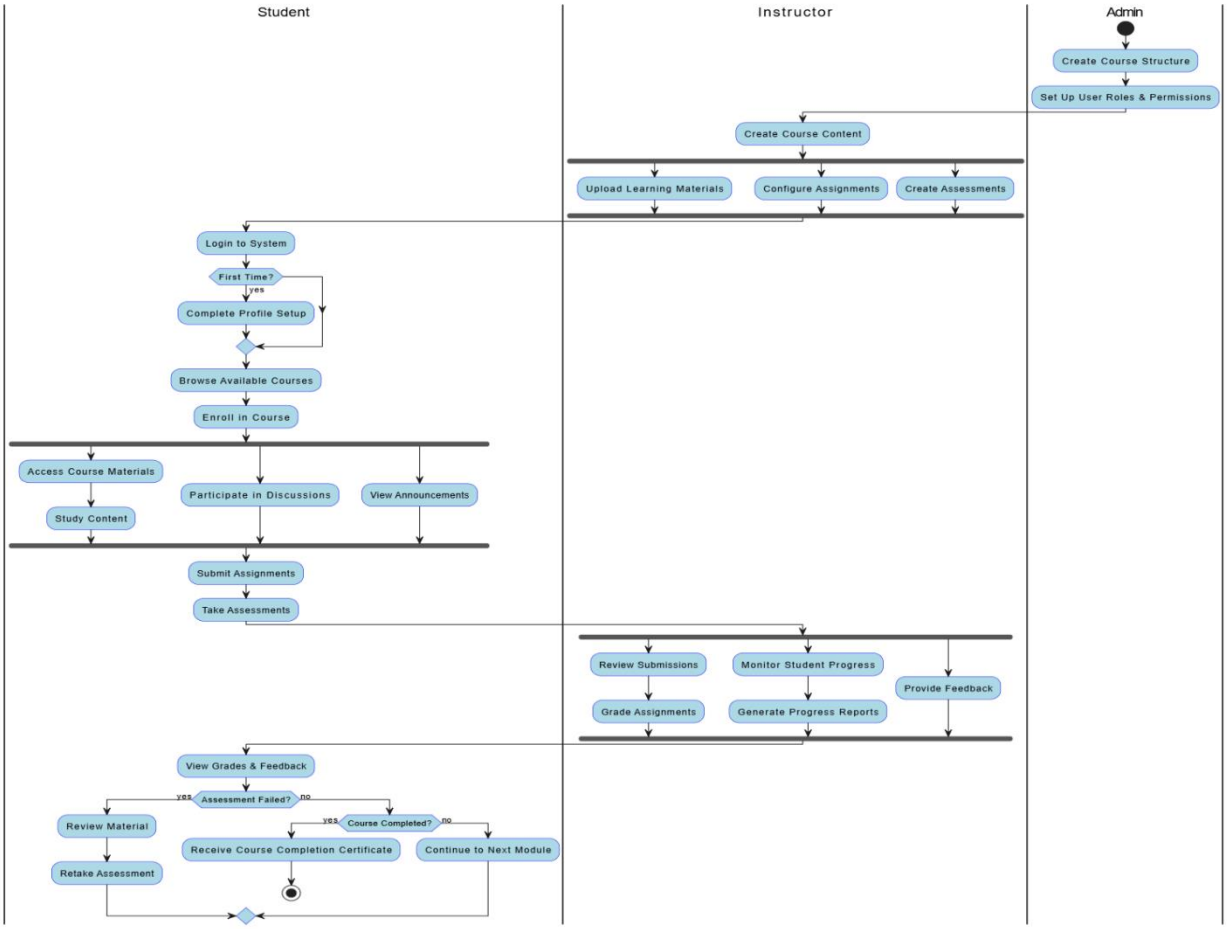
- **Risk Identification**
  - Performance degradation
  - Security vulnerabilities
  - Data integrity issues
  - User adoption challenges

9.2 Mitigation Strategies

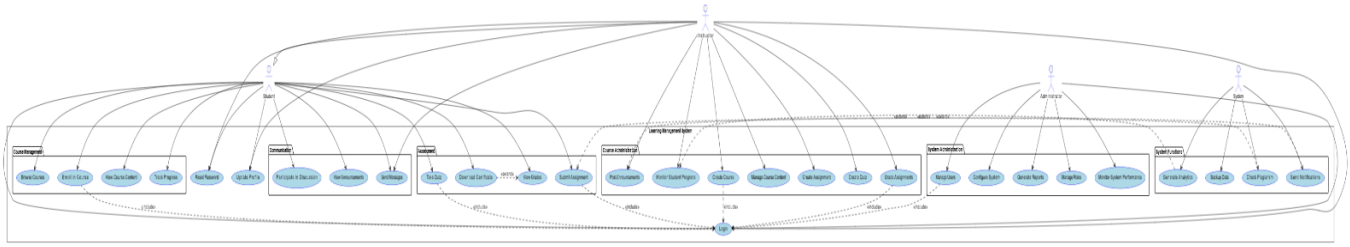
- **Prevention Measures**
  - Regular security updates
  - Performance optimization
  - Data backup procedures
  - User training programs

Part 4: Design (5 marks)

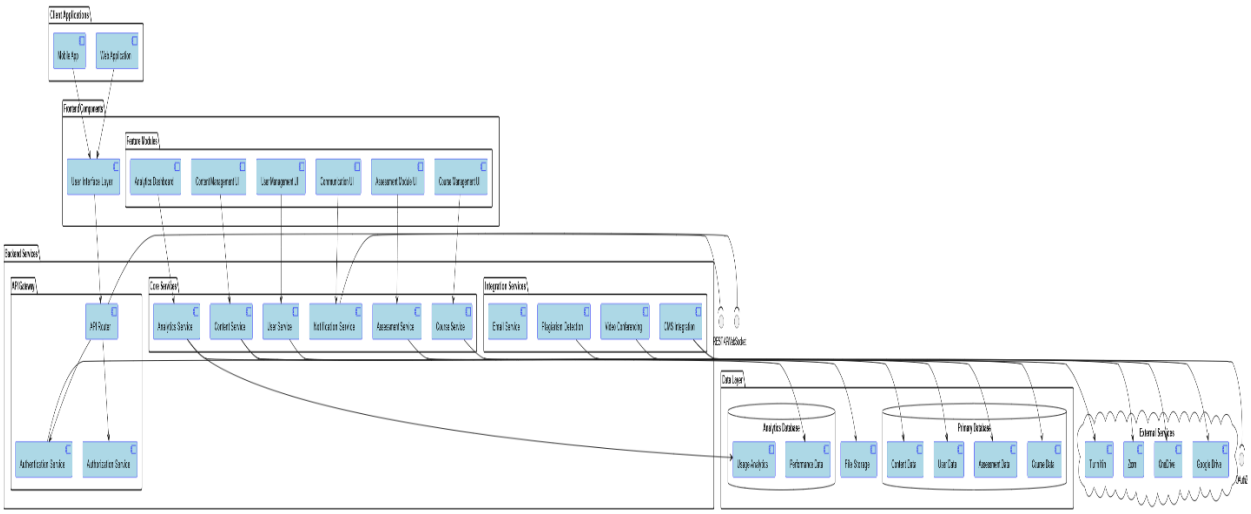
A. Activity Diagram:



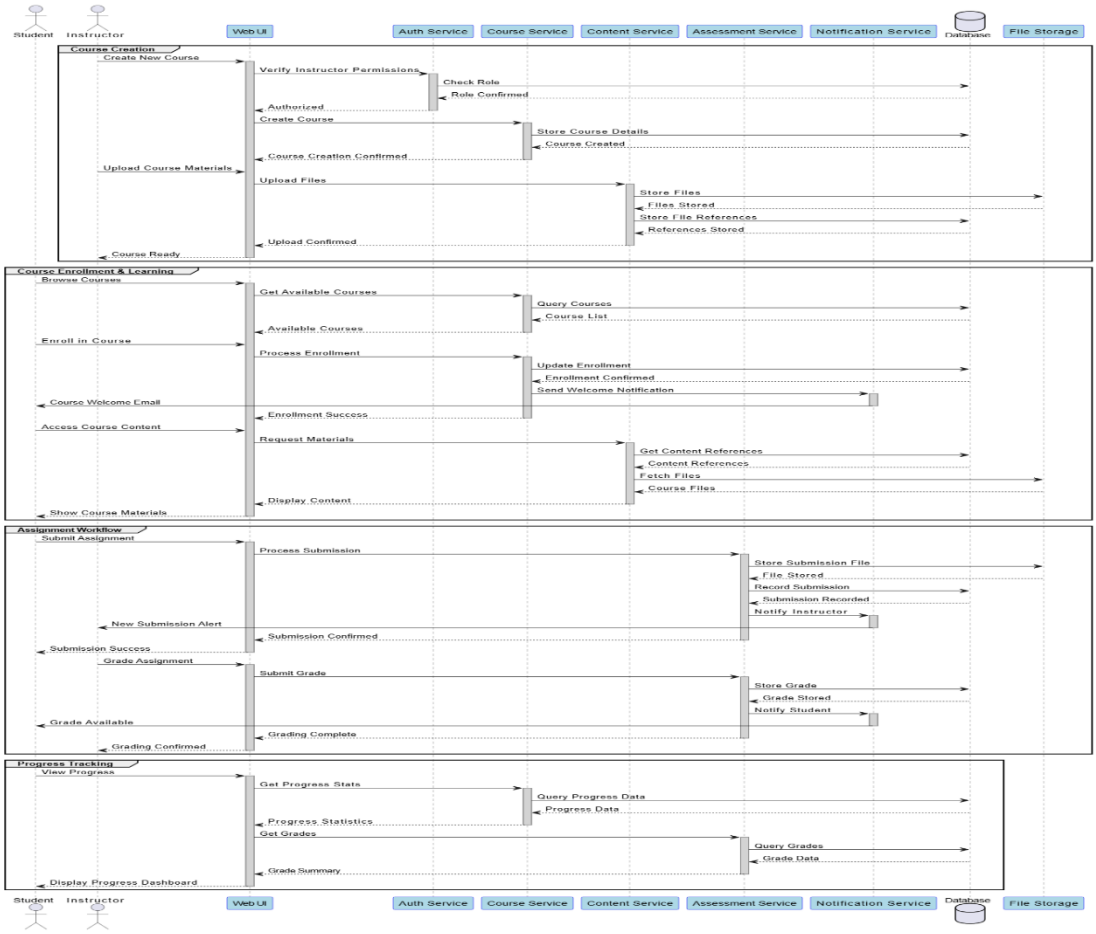
Use-case Diagram:



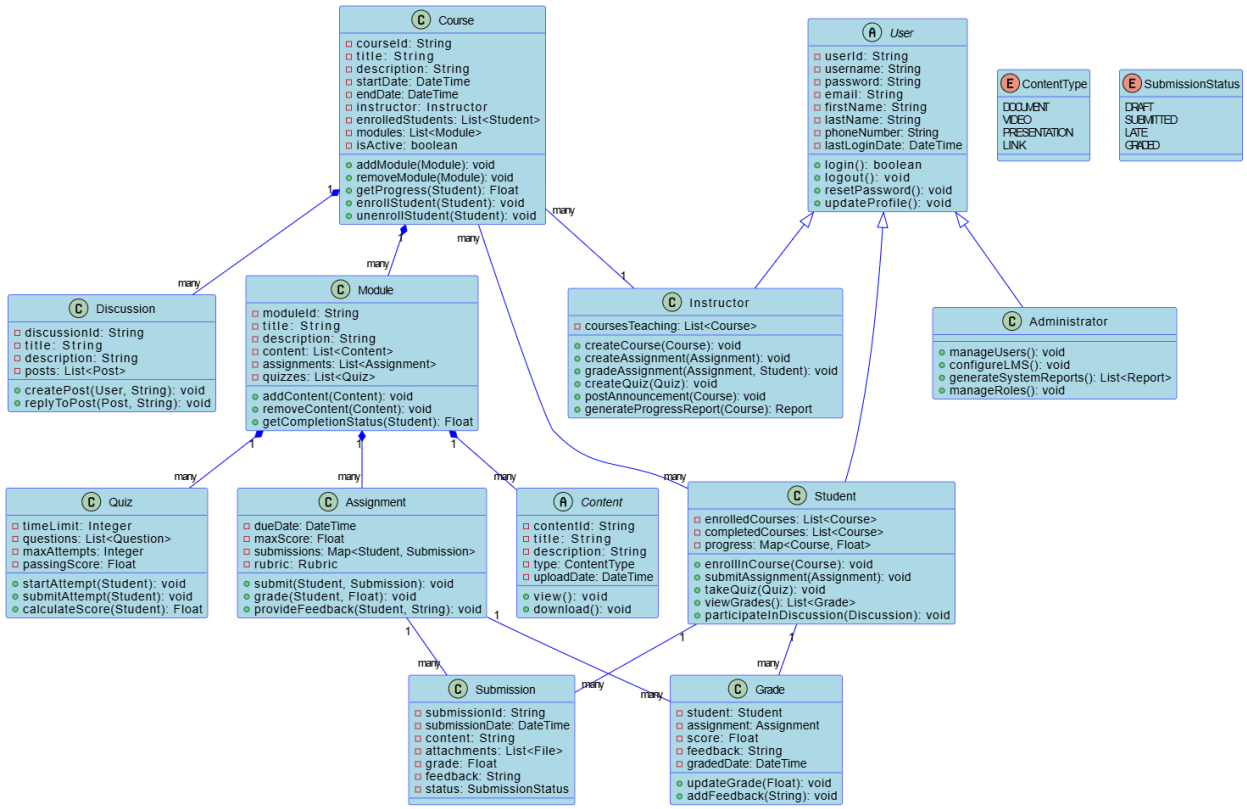
Component Diagram:



Sequential Diagram:



Class Diagram:



Part 5: Implementation (5 marks) (Reference from Progress Report)

1. React (Frontend)

- **Description**: React is a JavaScript library developed by Facebook for building user interfaces, particularly single-page applications. It allows developers to create reusable UI components and manage the application state effectively.
- **Use in Project**: React was used to build the user interface of the website, ensuring a responsive and dynamic user experience. It helped in building reusable components for different parts of the website, which reduced development time and enhanced maintainability.

2. Node.js (Backend)

- **Description**: Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser. It's built on Chrome's V8 JavaScript engine, making it fast and scalable for server-side applications.
- **Use in Project**: Node.js was used to handle the backend logic, server operations, and communicate with the frontend via APIs. It facilitated the creation of

RESTful services, which allowed seamless data exchange between the frontend and backend.

3. MySQL (Database)

- **Description**: MySQL is an open-source relational database management system. It's widely used for storing, managing, and querying data with structured schemas and supports SQL (Structured Query Language).
- **Use in Project**: MySQL was used to store and manage all the data related to users, courses, exams, and other entities of the website. It allowed structured data storage and efficient querying, which is essential for the dynamic nature of the web application.

4. XAMPP (Server)

- **Description**: XAMPP is a free and open-source cross-platform web server solution stack package that includes Apache HTTP Server, MariaDB (or MySQL), and interpreters for scripts written in PHP and Perl.
- **Use in Project**: XAMPP provided a local server environment to run the MySQL database and support backend development. It was essential for testing and deploying the application locally before moving it to a production environment.

5. Selenium (Automation Testing)

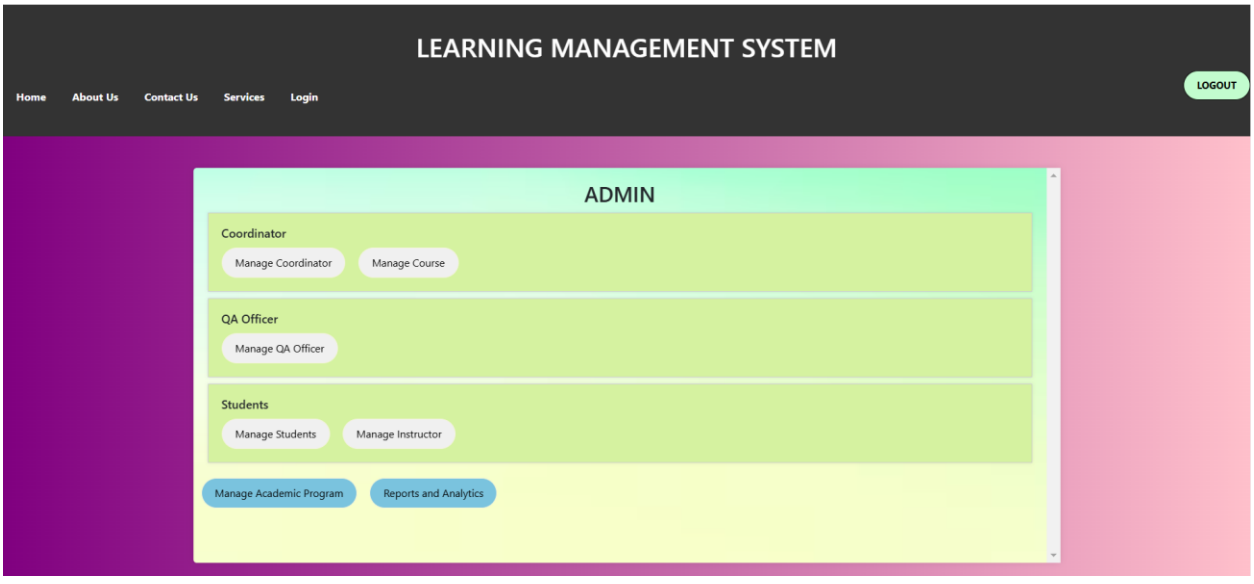
- **Description**: Selenium is an open-source tool for automating web browser interactions. It supports multiple programming languages and browsers, allowing developers to simulate user actions and test the functionality of their applications.
- **Use in Project**: Selenium was used for automation testing of the website, ensuring that critical functionalities worked as expected. It helped identify and resolve bugs in the UI, form submissions, and user interactions, enhancing the reliability and quality of the application.

6. **Bootstrap / Tailwind CSS**
   - **Description**: Bootstrap is a popular CSS framework for developing responsive and mobile-first websites, while Tailwind CSS is a utility-first CSS framework that allows more customization and styling flexibility.
   - **Use Case**: These frameworks help speed up frontend development by providing ready-made, customizable styles for layouts, components, and responsive design.

7. **Material-UI**
   - **Description**: Material-UI provides a set of React components that implement Google’s Material Design.

- **Use Case**: It's widely used in React projects to ensure a clean, consistent look with minimal custom CSS.

8. **Axios / Fetch API**
   - **Description**: Axios is a popular HTTP client for making API requests, and the Fetch API is a built-in JavaScript feature for handling requests and responses.
   - **Use Case**: Axios is widely used in React and other JavaScript applications to manage HTTP requests and handle API responses in a simpler and more configurable way than the native Fetch API.
9. **Express.js**
   - **Description**: Express is a fast, unopinionated, and minimalist web framework for Node.js. It's widely used for building APIs and web applications.
   - **Use Case**: Often used with Node.js, Express.js is perfect for handling routing, middleware, and APIs. It's highly flexible and customizable.
10. **Postman**
    - **Description**: Postman is a popular API testing tool that allows developers to send requests to an API and view responses.
    - **Use Case**: Used for testing and debugging APIs, as well as for verifying the backend responses during development.



UI ANALYSIS:

Reasons for choosing this Learning Management System (LMS) UI:

1. **Clear Navigation and Organization**:

- The UI is neatly organized into sections, making it easy for users to identify the different roles and functionalities available. For example, the roles of "Coordinator," "QA Officer," and "Students" each have dedicated sections with specific management options, such as "Manage Coordinator," "Manage Course," and "Manage Students."
- The navigation bar at the top allows quick access to key sections like "Home," "About Us," "Contact Us," "Services," and "Login," enhancing usability by giving users direct access to these main areas.

2. **Role-Based Access**:
   - This UI is tailored for an admin user, as indicated by the label "ADMIN" at the top of the main content area. By displaying only the relevant options for the admin, the interface avoids overwhelming users with unnecessary options, focusing instead on what an admin needs to manage the system effectively.
   - This role-based structure helps users quickly locate their respective tasks, whether they are coordinators, QA officers, or students. This segmentation also makes it easier to scale and add roles in the future without altering the core structure.
3. **Modern and Aesthetic Design**:
   - The color gradient background (from purple to pink) provides a visually appealing design that adds vibrancy and modernity to the interface. The pastel colors in the content sections (light green for role categories and blue for buttons) are subtle and soothing, which helps reduce eye strain and makes the interface look professional.
   - The use of rounded buttons and subtle borders gives the interface a contemporary feel, aligning with modern web design trends. This can increase user satisfaction and make the system more inviting.
4. **Functionally Grouped Action Buttons**:
   - The buttons for different management actions (e.g., "Manage Coordinator," "Manage Course," "Manage QA Officer") are grouped by role, which simplifies the user experience. This grouping allows users to quickly find actions related to each role, minimizing the time spent searching for options.
   - Additionally, the distinct color-coding of buttons within each section helps users distinguish between different types of actions. The "Manage Academic Program" and "Reports and Analytics" buttons are visually distinct in blue, possibly indicating that they are broader functions applicable across multiple roles.
5. **User-Friendly Layout with Effective Use of Space**:
   - The layout uses the screen space effectively without cluttering it, allowing users to focus on one section at a time. The minimalistic approach ensures that users are not overwhelmed with information, providing a clean and simple interface that improves usability.

- The logout button is prominently placed on the top right, making it easy for users to log out after completing their tasks, enhancing security and convenience. The main content area is centered and slightly elevated, drawing attention to the admin options and creating a clear focal point for the user.

Part 6: Testing (30 marks)

1. UNIT TESTING:

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Post Condition	Actual Result	Status
TC01	Launch Application	User is on the browser	Enter the URL to Launch the application	URL	Browsers lands the user to the website	User successfully access the website	Homepage Visible with content	Passed
TC02	Signup Functionality	User is on the login page and goes to signup page	Enter all the required details and click register	Username, Phone Number, Role, Email, Password	Account created, confirmation message shown	Account successfully created	Confirmation email sent and confirmation message visible	Passed
TC03	Login Functionality	User already has an account by signing up	Enter all the required details and click login	Username, Password	Successful login and user got redirected to the dashboard	User successfully logged in	Dashboard visible	Passed
TC04	Manage Coordinators (ADMIN)	User on Admin Dashboard	Click Manage Coordinator Button and Enter Name and Email of Coordinator	Name, Email	Displaying newly added coordinator immediately or vanishing the deleted coordinator immediately	New coordinator visible/ Old coordinator disappeared	New coordinator visible	Passed
TC05	Manage Course (ADMIN)	User on Admin Dashboard	Click Manage Course Button and Enter Name and Code of Course	Name, Code	Displaying newly added course immediately or vanishing the deleted course immediately	New course visible/ Old course disappeared	New course visible	Passed
TC06	Manage Students (ADMIN)	User on Admin Dashboard	Click Manage Students Button and Enter Name and Email of Student	Name, Email	Displaying newly added Student immediately or vanishing the deleted Student immediately	New Student visible/ Old Student disappeared	New Student visible	Passed
TC07	Create Exam (INSTRUCTOR)	User on Instructor Dashboard	Click Create Exam, Choose subject and enter questions, options and correct answer	Questions, 4 different options, answer for each	Confirmation about exam being created	New Exam created and visible to students	New Exam visible to students	Passed

TC08	**Create Assessment (INSTRUCTOR)**	**User on Instructor Dashboard**	**Click Create Assessment Button, choose subject and Enter description**	**Description about assignment**	**Confirmation about Assessment being created**	**New Assessment created and visible to students**	**New Assessment visible to students**	**Passed**
TC09	**Take Assessment (STUDENT)**	**User on Student Dashboard**	**Choose subject & Click Assessment Button**	**Selecting one option ss answer**	**Results being displayed in the end**	**Student successfully took the assessment and can see the score**	**Student got the score and his reports**	**Passed**
TC10	**Logout Functionality**	**User already has an account by signing up**	**User logs in and sees the dashboard and Click logout button on top right**		**Successful logout and user got redirected to the login page**	**User successfully logged out**	**Login page visible**	**Passed**

2. INTEGRATION TESTING:

Test Case ID	Components Involved	Test Scenario	Expected Result	Actual Result	Status
TC01	**Signup & Email Service**	**New Account created automatically calls emiljs API and sends confirmation email**	**Confirmation mail received successfully**	**Welcome Email received as expected to the user's mail address**	**Passed**
TC02	**User Role, & Dashboard**	**User login perfectly redirects to the appropriate dashboard**	**User lands in his dashboard based on his role**	**User landed to Appropriate Dashboard**	**Passed**
TC03	**Student Chat with Instructors**	**Message flow b/w student & instructor**	**Student & instructor both receive each other's messages**	**Student received instructor's messages and vice versa**	**Passed**
TC04	**Student Chat with Coordinators**	**Message flow b/w student & Coordinator**	**Student & Coordinator both receive each other's messages**	**Student received Coordinator's messages and vice versa**	**Passed**

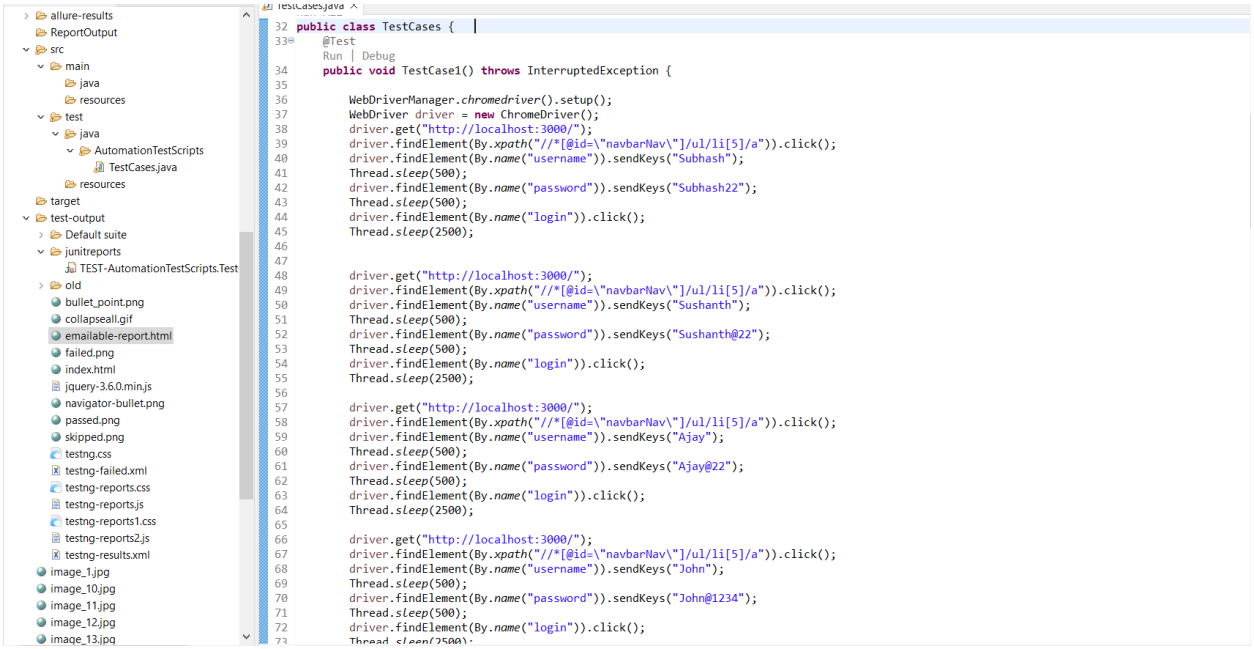
TC05	**Creating content for the courses**	**Adding new course content/ updating older one**	**New content gets created and shows up in the list**	**New course content displayed**	**Passed**

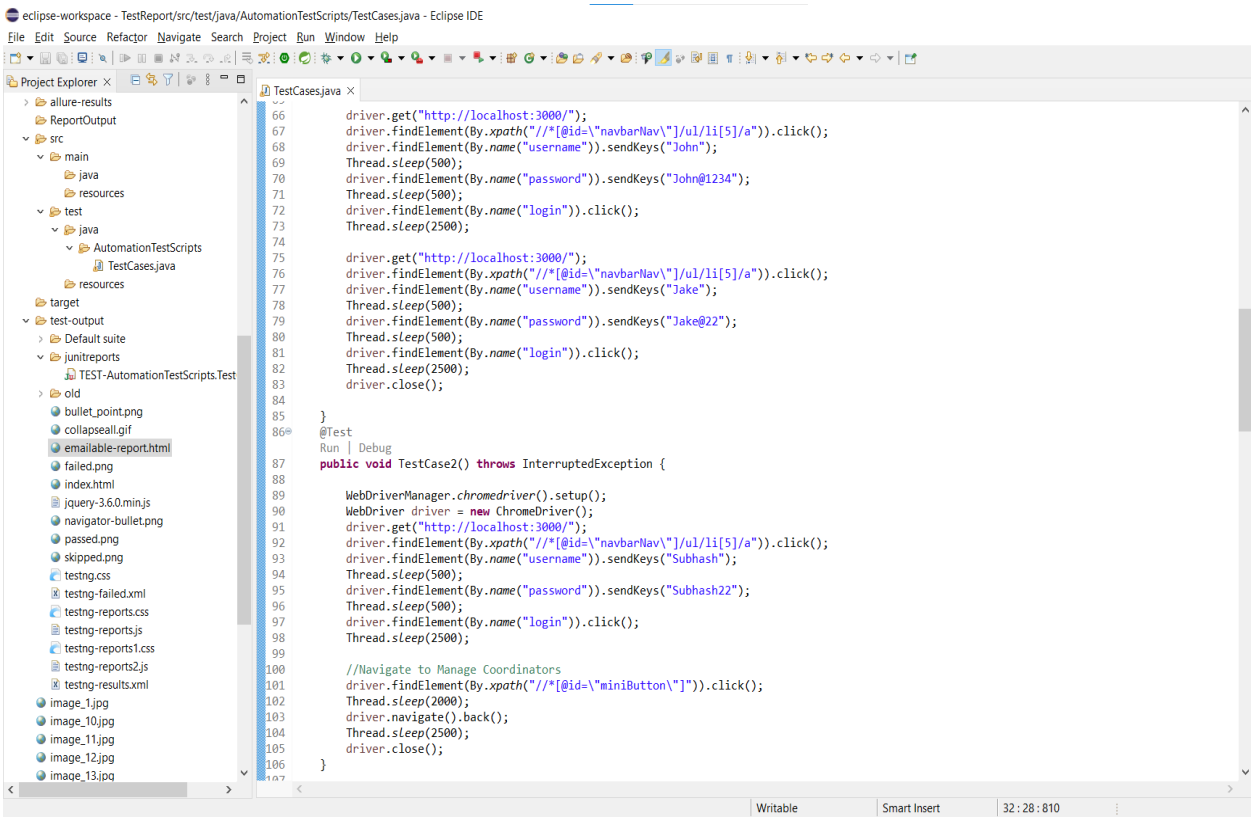
3. SYSTEM TESTING:

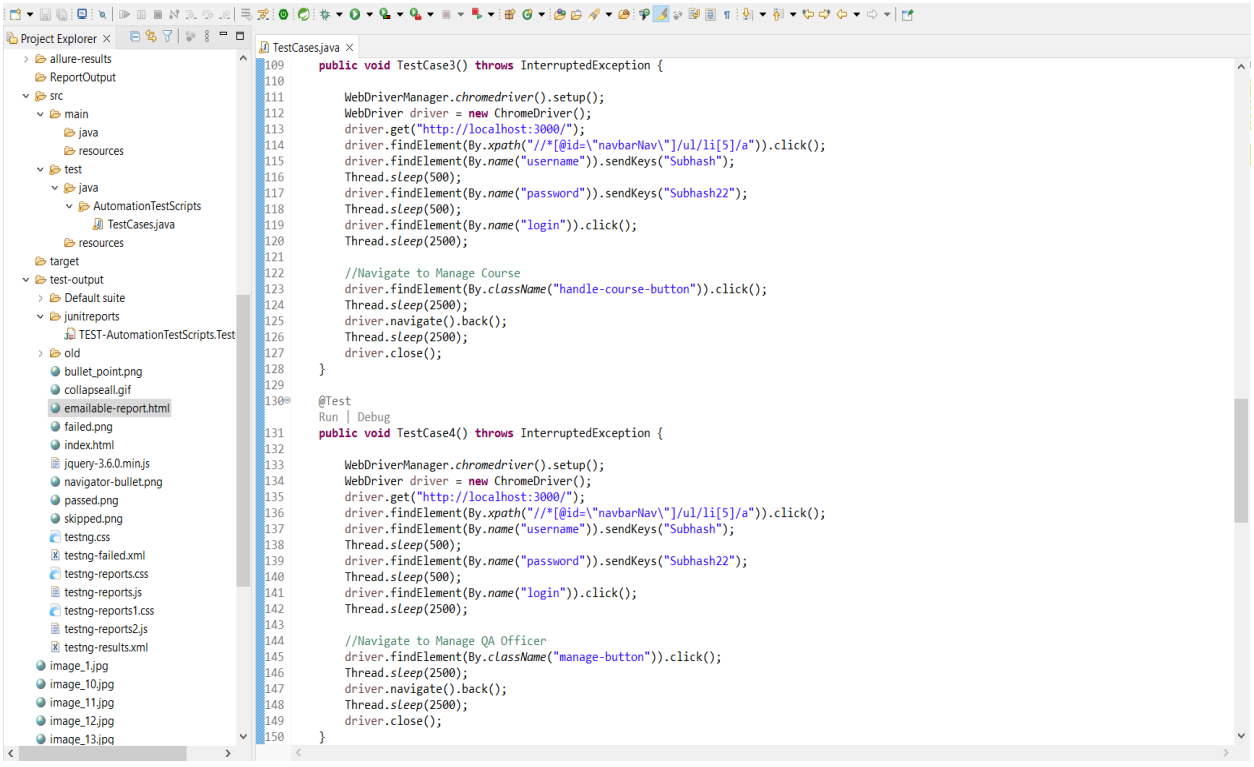
Functionality	Test Case Description	Expected Result	Results	Status
Page Rendering Speed	**Testing how fast and quick the website is responsive to the user**	**Pages render in acceptable time limits**	**Page response doesn't take much time. It loads quickly**	**Pass**
User Signup & Login	**Testing new user creation and login with the created user details**	**New user account is successfully created and logged in**	**Account successfully created and reflected in database**	**Pass**
Data Reflection	**Testing that the various data between various roles like messages, courses, reports are properly reflected upon change**	**Messages are sent and received. Course, content, exams are all being reflected**	**Data is updated whenever changed throughout the website**	**Pass**
Form Submission	**Test if all forms are submitted correctly and data is processed as expected.**	**Signup form is stored properly in the database upon successful**	**User registration successful**	**Pass**
Responsiveness	**Open the website on**	**No inappropriate**	**Webpages fits and**	**Pass**

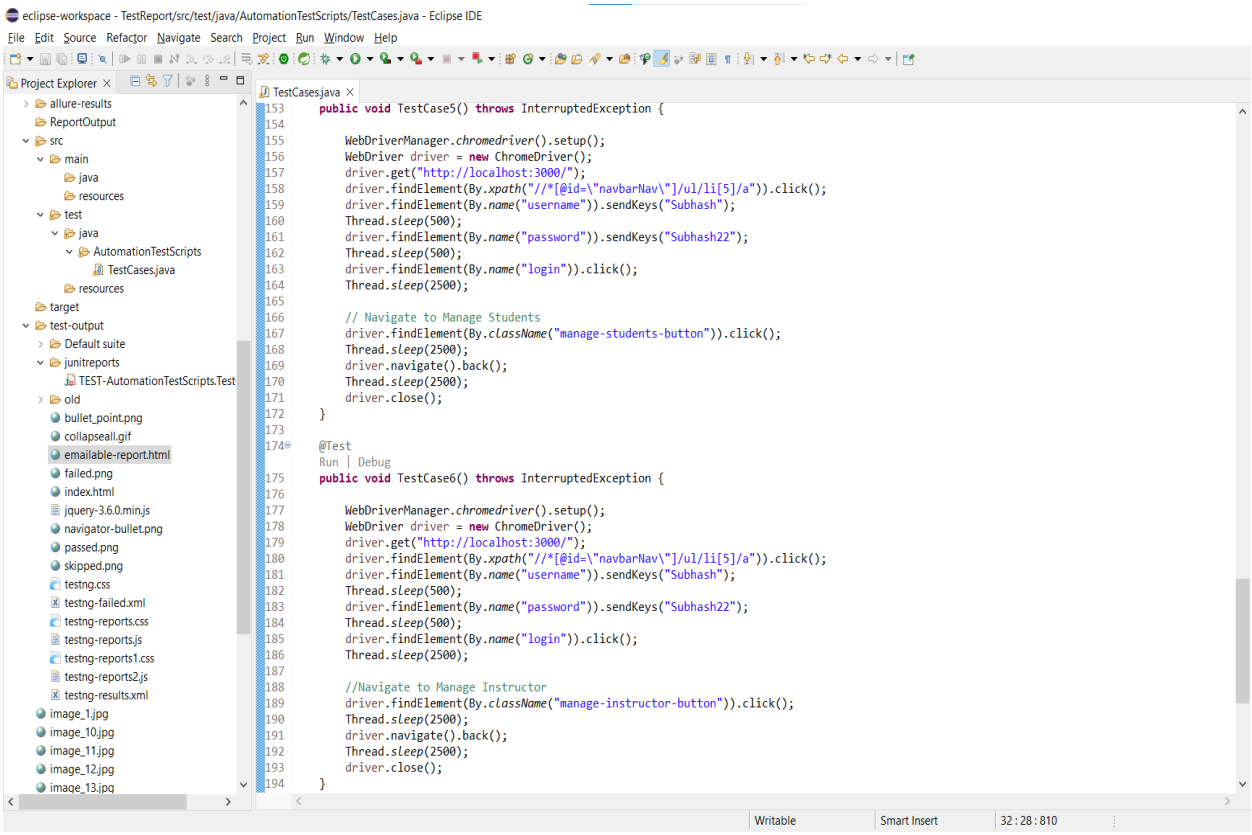
	multiple devices	**designs. It fits properly in different screens**	**works well in all devices**	

4. AUTOMATION TESTING:




eclipse-workspace - TestReport/src/test/java/AutomationTestScripts/TestCases.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
allure-results
ReportOutput
src
main
java
resources
test
java
AutomationTestScripts
TestCases.java
resources
target
test-output
Default suite
junitreports
TEST-AutomationTestScripts.Test
old
bullet_point.png
collapseall.gif
emailable-report.html
failed.png
index.html
jquery-3.6.0.min.js
navigator-bullet.png
passed.png
skipped.png
testng.css
testng-failed.xml
testng-reports.css
testng-reports.js
testng-reports1.css
testng-reports2.js
testng-results.xml
image_1.jpg
image_10.jpg
image_11.jpg
image_12.jpg
image_13.jpg
TestCases.java
driver.get("http://localhost:3000/");
driver.findElement(By.xpath("//*[@id=\"navbarNav\"]/ul/li[5]/a")).click();
driver.findElement(By.name("username")).sendKeys("John");
Thread.sleep(500);
driver.findElement(By.name("password")).sendKeys("John@1234");
Thread.sleep(500);
driver.findElement(By.name("login")).click();
Thread.sleep(2500);
driver.get("http://localhost:3000/");
driver.findElement(By.xpath("//*[@id=\"navbarNav\"]/ul/li[5]/a")).click();
driver.findElement(By.name("username")).sendKeys("Jake");
Thread.sleep(500);
driver.findElement(By.name("password")).sendKeys("Jake@22");
Thread.sleep(500);
driver.findElement(By.name("login")).click();
Thread.sleep(2500);
driver.close();
}
@Test
Run | Debug
public void TestCase2() throws InterruptedException {
WebDriverManager.chromedriver().setup();
WebDriver driver = new ChromeDriver();
driver.get("http://localhost:3000/");
driver.findElement(By.xpath("//*[@id=\"navbarNav\"]/ul/li[5]/a")).click();
driver.findElement(By.name("username")).sendKeys("Subhash");
Thread.sleep(500);
driver.findElement(By.name("password")).sendKeys("Subhash22");
Thread.sleep(500);
driver.findElement(By.name("login")).click();
Thread.sleep(2500);
//Navigate to Manage Coordinators
driver.findElement(By.xpath("//*[@id=\"miniButton\"]")).click();
Thread.sleep(2000);
driver.navigate().back();
Thread.sleep(2500);
driver.close();
}
Writable
Smart Insert
32 : 28 : 810


Project Explorer
allure-results
ReportOutput
src
main
java
resources
test
java
AutomationTestScripts
TestCases.java
resources
target
test-output
Default suite
junitreports
TEST-AutomationTestScripts.Test
old
bullet_point.png
collapseall.gif
emailable-report.html
failed.png
index.html
jquery-3.6.0.min.js
navigator-bullet.png
passed.png
skipped.png
testng.css
testng-failed.xml
testng-reports.css
testng-reports.js
testng-reports1.css
testng-reports2.js
testng-results.xml
image_1.jpg
image_10.jpg
image_11.jpg
image_12.jpg
image_13.jpg
TestCases.java
public void TestCase3() throws InterruptedException {
WebDriverManager.chromedriver().setup();
WebDriver driver = new ChromeDriver();
driver.get("http://localhost:3000/");
driver.findElement(By.xpath("//*[@id=\"navbarNav\"]/ul/li[5]/a")).click();
driver.findElement(By.name("username")).sendKeys("Subhash");
Thread.sleep(500);
driver.findElement(By.name("password")).sendKeys("Subhash22");
Thread.sleep(500);
driver.findElement(By.name("login")).click();
Thread.sleep(2500);
//Navigate to Manage Course
driver.findElement(By.className("handle-course-button")).click();
Thread.sleep(2500);
driver.navigate().back();
Thread.sleep(2500);
driver.close();
}
@Test
Run | Debug
public void TestCase4() throws InterruptedException {
WebDriverManager.chromedriver().setup();
WebDriver driver = new ChromeDriver();
driver.get("http://localhost:3000/");
driver.findElement(By.xpath("//*[@id=\"navbarNav\"]/ul/li[5]/a")).click();
driver.findElement(By.name("username")).sendKeys("Subhash");
Thread.sleep(500);
driver.findElement(By.name("password")).sendKeys("Subhash22");
Thread.sleep(500);
driver.findElement(By.name("login")).click();
Thread.sleep(2500);
//Navigate to Manage QA Officer
driver.findElement(By.className("manage-button")).click();
Thread.sleep(2500);
driver.navigate().back();
Thread.sleep(2500);
driver.close();
}


eclipse-workspace - TestReport/src/test/java/AutomationTestScripts/TestCases.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
allure-results
ReportOutput
src
main
java
resources
test
java
AutomationTestScripts
TestCases.java
resources
target
test-output
Default suite
junitreports
TEST-AutomationTestScripts.Test
old
bullet_point.png
collapseall.gif
emailable-report.html
failed.png
index.html
jquery-3.6.0.min.js
navigator-bullet.png
passed.png
skipped.png
testng.css
testng-failed.xml
testng-reports.css
testng-reports.js
testng-reports1.css
testng-reports2.js
testng-results.xml
image_1.jpg
image_10.jpg
image_11.jpg
image_12.jpg
image_13.jpg
TestCases.java
153 public void TestCase5() throws InterruptedException {
154
155 WebDriverManager.chromedriver().setup();
156 WebDriver driver = new ChromeDriver();
157 driver.get("http://localhost:3000/");
158 driver.findElement(By.xpath("//*[@id=\"navbarNav\"]/ul/li[5]/a")).click();
159 driver.findElement(By.name("username")).sendKeys("Subhash");
160 Thread.sleep(500);
161 driver.findElement(By.name("password")).sendKeys("Subhash22");
162 Thread.sleep(500);
163 driver.findElement(By.name("login")).click();
164 Thread.sleep(2500);
165
166 // Navigate to Manage Students
167 driver.findElement(By.className("manage-students-button")).click();
168 Thread.sleep(2500);
169 driver.navigate().back();
170 Thread.sleep(2500);
171 driver.close();
172 }
173
174 @Test
Run | Debug
175 public void TestCase6() throws InterruptedException {
176
177 WebDriverManager.chromedriver().setup();
178 WebDriver driver = new ChromeDriver();
179 driver.get("http://localhost:3000/");
180 driver.findElement(By.xpath("//*[@id=\"navbarNav\"]/ul/li[5]/a")).click();
181 driver.findElement(By.name("username")).sendKeys("Subhash");
182 Thread.sleep(500);
183 driver.findElement(By.name("password")).sendKeys("Subhash22");
184 Thread.sleep(500);
185 driver.findElement(By.name("login")).click();
186 Thread.sleep(2500);
187
188 //Navigate to Manage Instructor
189 driver.findElement(By.className("manage-instructor-button")).click();
190 Thread.sleep(2500);
191 driver.navigate().back();
192 Thread.sleep(2500);
193 driver.close();
194 }
Writable
Smart Insert
32 : 28 : 810

RESULTS:

Test	# Passed	# Skipped	# Retried	# Failed	Time (ms)	Included Groups	Excluded Groups
Default suite							
Default test	6	0	0	0	73,094		

Class	Method	Start	Time (ms)
Default suite			
Default test — passed			
AutomationTestScripts.TestCases	TestCase1	1731301529206	21384
	TestCase2	1731301550649	9998
	TestCase3	1731301560686	10446
	TestCase4	1731301571136	10300
	TestCase5	1731301581439	10393
	TestCase6	1731301591836	10431

```
PASSED: AutomationTestScripts.TestCases.TestCase2
PASSED: AutomationTestScripts.TestCases.TestCase5
PASSED: AutomationTestScripts.TestCases.TestCase1
PASSED: AutomationTestScripts.TestCases.TestCase3
PASSED: AutomationTestScripts.TestCases.TestCase4
PASSED: AutomationTestScripts.TestCases.TestCase6

===============================================
    Default test
    Tests run: 6, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 6, Passes: 6, Failures: 0, Skips: 0
===============================================
```

Part 7: Maintenance (10 marks):

1. **Enhanced User Profile and Dashboard Customization**

- **Description**: Allow users to customize their profile pages with additional fields (like bio, academic achievements, interests) and the ability to change dashboard layouts or themes.
- **Benefit**: This improvement will provide users with a more personalized experience, helping them showcase relevant details to others and improving engagement within the platform.
- **Implementation**: Use customizable components in React and expand the user schema in MySQL to store additional profile data.

2. Advanced Search and Filtering Options

- **Description**: Add more powerful search and filtering options across the site, allowing users to search by academic interests, institutions, location, or keywords in posts.
- **Benefit**: Enhanced search and filtering will allow users to find content and connections more efficiently, making the platform more useful for networking and academic collaboration.
- **Implementation**: Use a library like ElasticSearch or incorporate full-text search capabilities in MySQL for better performance with large datasets.

3. Messaging System Upgrade with Real-Time Chat

- **Description**: Improve the messaging system to support real-time chat functionality, similar to messaging apps like WhatsApp or Slack, with notifications, typing indicators, and message status.
- **Benefit**: Real-time messaging will facilitate instant communication, making the platform feel more dynamic and engaging, especially for academic discussions and networking.
- **Implementation**: Use WebSockets or a library like Socket.io for real-time functionality, coupled with Redux to manage state in the React frontend.

4. Integration with LinkedIn and Google Scholar

- **Description**: Allow users to connect their LinkedIn and Google Scholar accounts to display their professional network and academic publications directly on their profile.
- **Benefit**: This integration will add value by showcasing users' professional and academic achievements, making it easier to connect and collaborate with others.
- **Implementation**: Use OAuth 2.0 for LinkedIn and Google Scholar APIs to fetch and display user data securely with their consent.

5. Recommendation System for Connections and Content

- **Description**: Implement a recommendation engine that suggests connections (other users), relevant articles, or events based on user activity, interests, and academic background.

- **Benefit**: A recommendation system will enhance user engagement by suggesting relevant content and people, making the platform more interactive and user-friendly.
- **Implementation**: Build a recommendation engine using machine learning libraries like TensorFlow or Scikit-learn to analyze user behavior and make personalized recommendations.

6. Gamification Features

- **Description**: Add gamification elements like badges, achievements, and points for activities such as attending events, posting content, or connecting with new users.
- **Benefit**: Gamification encourages more interaction and active participation on the platform, as users get rewarded for their engagement.
- **Implementation**: Use a point system that tracks user actions and grants badges based on predefined milestones.

7. Admin Dashboard for Content Moderation and User Management

- **Description**: Create an admin dashboard that allows moderators to manage users, content, and reports, ensuring the platform stays safe and well-maintained.
- **Benefit**: This will improve the quality of the content on the platform, maintain community standards, and enhance user safety.
- **Implementation**: Develop a secure admin interface with role-based access control to allow admins to view and manage reports, ban users, or delete inappropriate content.

8. Mobile App Development

- **Description**: Develop a mobile app version of the platform to improve accessibility and engagement on mobile devices.
- **Benefit**: A mobile app would make the platform more accessible, especially for users who prefer interacting on-the-go, increasing usage and user retention.
- **Implementation**: Use React Native for cross-platform development or build separate native apps for iOS and Android. Integrate with the existing backend through REST APIs.

9. Advanced Analytics for User Insights

- **Description**: Provide detailed analytics for users, such as profile visits, content engagement, and network growth over time.
- **Benefit**: Analytics give users insights into their reach and engagement, motivating them to contribute more and grow their network.
- **Implementation**: Track user interactions with tools like Google Analytics or a custom solution, and display aggregated data on the user dashboard.

10. Event Management with RSVP and Notifications

- **Description**: Create an event management feature where users can organize, share, and RSVP to events such as academic conferences, seminars, or study groups.
- **Benefit**: Events help foster a sense of community and increase collaboration among users, making the platform a hub for academic networking.
- **Implementation**: Build event pages where users can RSVP, receive event notifications, and add events to their calendar.

11. Multi-Language Support

- **Description**: Add support for multiple languages to make the platform accessible to users from diverse linguistic backgrounds.
- **Benefit**: Multi-language support will expand the platform's reach globally, enhancing user experience for non-English-speaking users.
- **Implementation**: Use a library like react-i18next for translations on the frontend, and store translation data in the backend.

12. API for External Integrations

- **Description**: Create a public API that allows external applications to access platform data, such as user profiles, events, and academic resources.
- **Benefit**: A public API will enable integration with other academic and professional platforms, extending the functionality and usability of the platform.
- **Implementation**: Develop secure REST or GraphQL APIs and include documentation to make it easy for third-party developers to interact with the platform.

13. Content Moderation Using AI

- **Description**: Implement an AI-based content moderation system to detect and filter inappropriate content automatically.
- **Benefit**: AI moderation helps maintain a safe environment, reducing the need for manual moderation and improving user experience.
- **Implementation**: Use machine learning models for image and text moderation, like AWS Rekognition or Google Cloud Vision for image moderation, and NLP for text analysis.

14. Accessibility Improvements

- **Description**: Enhance accessibility by adding support for screen readers, keyboard navigation, and high-contrast modes for users with disabilities.
- **Benefit**: Accessibility improvements will ensure that the platform is usable by all, including users with visual, auditory, or motor impairments.

- **Implementation**: Follow Web Content Accessibility Guidelines (WCAG) and use tools like Axe for accessibility testing and improvements.

15. Single Sign-On (SSO)

- **Description**: Implement Single Sign-On integration to allow users to log in using their institutional or Google accounts.
- **Benefit**: SSO simplifies the login process, making it easier for users to access the platform while improving security.
- **Implementation**: Use SSO providers like Auth0, Firebase Authentication, or support OAuth 2.0 protocols for integration with third-party identity providers.

16. Offline Mode and Data Sync

- **Description**: Add offline capabilities to allow users to access certain features (like saved profiles and messages) without an internet connection, syncing data once they're back online.
- **Benefit**: Offline functionality improves usability for users with intermittent internet access, making the platform more robust and accessible.
- **Implementation**: Implement service workers and local storage for caching data, with a sync mechanism to update data when the user reconnects.

17. Integrate Video Conferencing for Virtual Meetings

- **Description**: Add a video conferencing feature to facilitate virtual meetings, study sessions, or academic consultations within the platform.
- **Benefit**: This provides users with a built-in, easy-to-access tool for real-time academic collaboration, reducing the need to switch to third-party tools.
- **Implementation**: Use third-party video conferencing APIs, such as Zoom or WebRTC, to add this functionality without needing a full custom video solution.

18. Progressive Web Application (PWA) Features

- **Description**: Transform the website into a Progressive Web Application (PWA) to provide a native app-like experience with offline capabilities, push notifications, and faster load times.
- **Benefit**: PWAs offer a mobile app-like experience on the web, improving performance and accessibility on mobile devices.
- **Implementation**: Use tools like Workbox for service worker management and cache strategies, along with manifest files to enable PWA features.

Part 8: Roles and Responsibilities: (5 marks)

1. Subhash Radhakrishnan: Frontend Developer

- **Responsibilities**:
    - Develop and implement the user interface of the website using React.
    - Collaborate with the designer (if applicable) to translate wireframes and designs into functional code.
    - Ensure responsive design across various devices and browsers.
    - Optimize front-end performance for a smooth user experience.
    - Work with the backend developer to integrate APIs and ensure data flows correctly.
    - Implement user interface enhancements like animations, transitions, and visual feedback.
    - Contribute to UX improvements, focusing on accessibility and usability.
- **Skills & Tools**:
    - Proficiency in HTML, CSS, JavaScript, and React.
    - Familiarity with UI frameworks (e.g., Bootstrap, Material UI) for quicker styling.
    - Knowledge of responsive and mobile-first design principles.
    - Experience with frontend build tools like Webpack and task runners like npm or Yarn.

2. Saikiranreddy peddavootla: Backend Developer

- **Responsibilities**:
    - Develop and manage the server-side logic of the application using Node.js.
    - Design and maintain RESTful APIs to interact with the frontend.
    - Ensure data security, authentication, and authorization through techniques like JWT or OAuth.
    - Optimize backend performance to handle concurrent user requests efficiently.
    - Handle data processing and business logic for features like search, recommendations, and messaging.
    - Implement data validation and error handling mechanisms.

- **Skills & Tools**:
    - Proficiency in Node.js, Express, and related frameworks.
    - Understanding of RESTful API design and best practices.
    - Experience with authentication/authorization libraries such Firebase Authentication.
    - Familiarity with database querying and optimization.

3. Durgashree Hakkinalu Somashekaraiah: Database Manager (Database & API Integrator)

- **Responsibilities**:
    - Design and maintain the database schema to store data for users, messages, profiles, and other entities.
    - Work closely with the backend developer to set up and integrate APIs for seamless data retrieval.
    - Optimize database queries to improve application speed and responsiveness.
    - Ensure data integrity, handle migrations, and implement backups and recovery processes.
    - Manage and monitor database performance, troubleshoot issues, and implement indexing as needed.
    - Maintain the security of the database by applying appropriate access control and encryption.
- **Skills & Tools**:
    - Proficiency in SQL, MySQL (or any other relational database), and database design principles.
    - Knowledge of database optimization techniques and indexing strategies.
    - Familiarity with MySQL Workbench, phpMyAdmin, or any database management tools.
    - Experience with database integration libraries

4. Varshith Konduru: Quality Assurance (QA) Engineer / Automation Tester

- **Responsibilities**:
    - Develop and execute test cases to verify the functionality, performance, and security of the application.

    - Perform various types of testing, including functional, integration, and user acceptance testing (UAT).
    - Create and maintain automated test scripts using Selenium to streamline the testing process.
    - Identify, document, and track bugs, and work closely with developers to resolve issues.
    - Conduct performance testing to ensure the application can handle expected user loads.
    - Test for cross-browser and device compatibility to ensure consistent user experience.
- **Skills & Tools**:
    - Proficiency in writing and executing manual and automated test cases.
    - Experience with Selenium and testing frameworks for JavaScript testing.
    - Familiarity with bug-tracking tools.
    - Knowledge of testing RESTful APIs using tools like Postman.

Additional Collaborative Responsibilities:

Each team member also contributed to:

- **Project Management**: Help manage the project timeline, set goals, and participate in daily stand-ups or regular meetings to ensure smooth progress. Use tools like Trello, Jira, or Asana to organize tasks and track progress.
- **Documentation**: Document code, API endpoints, database schema, and testing results for easy reference and handoff to new team members if needed.
- **Code Reviews**: Perform code reviews to ensure consistency, improve code quality, and learn from each other's work.
- **Deployment and Maintenance**: Collaborate on deploying the project, monitoring performance, and ensuring that the site is stable in a production environment.

Part 9: References (5 marks)

1. Framework and Library Documentation

- **React Documentation**: React. (n.d.). *A JavaScript library for building user interfaces*. Retrieved from https://reactjs.org/docs/getting-started.html

- **Node.js Documentation**: Node.js. (n.d.). *JavaScript runtime built on Chrome's V8 JavaScript engine*. Retrieved from https://nodejs.org/en/docs/
- **Express Documentation**: Express. (n.d.). *Fast, unopinionated, minimalist web framework for Node.js*. Retrieved from https://expressjs.com/
- **MySQL Documentation**: MySQL. (n.d.). *MySQL 8.0 Reference Manual*. Retrieved from https://dev.mysql.com/doc/

2. Books and Articles on Full-Stack Development

- Brown, E. (2019). *Web Development with Node and Express: Leveraging the JavaScript Stack*. O'Reilly Media.
- Banks, A. & Porcello, E. (2018). *Learning React: Functional Web Development with React and Redux*. O'Reilly Media.
- Johnson, E., & Schmitt, T. (2021). *Full-Stack Web Development with Node, Express, and MongoDB*. Packt Publishing.

3. Tools and Software Resources

- **XAMPP Documentation**: Apache Friends. (n.d.). *XAMPP Overview and Usage*. Retrieved from https://www.apachefriends.org/index.html
- **Visual Studio Code Documentation**: Microsoft. (n.d.). *Visual Studio Code User Guide*. Retrieved from https://code.visualstudio.com/docs
- **Lucidchart**: Lucid Software Inc. (n.d.). *Diagramming and Flowchart Creation Tool*. Retrieved from https://www.lucidchart.com/

4. Project Management and Workflow

- Scrum Alliance. (n.d.). *The Scrum Guide™*. Retrieved from https://scrumguides.org/
- Atlassian (n.d.). *Jira Software Documentation*. Retrieved from https://www.atlassian.com/software/jira/guides
- Trello. (n.d.). *Trello Guide: Manage Projects with Trello*. Retrieved from https://trello.com/en-US

5. Testing and QA

- **Selenium Documentation**: Selenium. (n.d.). *Selenium WebDriver*. Retrieved from https://www.selenium.dev/documentation/
- Postman. (n.d.). *API Development and Testing*. Retrieved from https://www.postman.com/

6. API and Database Design

- **REST API Design**: Richardson, L., & Amundsen, M. (2013). *RESTful Web APIs*. O'Reilly Media.
- **Database Design**: Hernandez, M. J. (2013). *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*. Addison-Wesley.