

Online Learning Management System

Team ID: 8

Members name and ID

Jaydeep Mojidra: 1002186827

Pranjal Kamboj : 1002157885

Samyak Jain : 1002166068

Dhruv Shah : 1002170901

Project Description:

We are developing a robust e-learning platform featuring secure user authentication, intuitive course management, and seamless student enrollment. Students can easily submit assignments, view grades, and participate in interactive modules. The platform also includes a dynamic discussion forum, a real-time notification system, and tools for document sharing. To ensure a well-rounded experience, users can provide feedback and access support when needed, creating an engaging and collaborative learning environment.

Project Plan: (hint: to include all parts, please check slide 10 in lecture 2)

Introduction:

- 1) Objectives:** The primary goal of this project is to create a user-friendly and efficient e-learning platform that supports seamless course management, student enrollment, and assignment tracking. This platform will provide students and instructors with a streamlined way to manage courses, submit and grade assignments, and engage in discussions, ensuring a productive and collaborative learning environment.
- 2) Features and Functionality:** Outline the specific features that the website will have, such as user authentication, course management, student enrollment, assignment submission, assignment grading, modules, discussion forum, notification system, document sharing, feedback and sharing.
- 3) Technical Requirements:** Outline the technical specifications for the project, including the programming languages, frameworks, and platforms utilized during the development process.
- 4) Design and User Experience:** Detail the design aesthetic and user experience for the website.

5) Technology which we going to use:

- Frontend: HTML, CSS, JavaScript
- Backend: NodeJS, ReactJS
- Database: MySQL
- IDE: Visual Studio Code

Standards, Guidelines and Procedures

Standards and Guidelines:

1. Security Standards:
 - Implement secure authentication (e.g., OAuth 2.0).
 - Use encryption for sensitive data and secure communication.
2. Development Standards:
 - Follow coding standards (e.g., consistent naming conventions, code commenting).
 - Use version control (e.g., Git) for collaborative development.
3. User Experience Guidelines:
 - Ensure a responsive and intuitive user interface.
 - Provide clear navigation and accessibility features.
4. Data Management Standards:
 - Maintain data integrity and validation.
 - Use a consistent database schema and naming conventions.
5. Testing Standards:
 - Conduct unit, integration, and user acceptance testing.
 - Automate testing where feasible to enhance reliability.

Procedures:

1. Development Procedure:
 - Follow Agile methodologies for iterative development and regular feedback.
 - Conduct code reviews to ensure quality and adherence to standards.
2. Testing Procedure:
 - Implement a testing strategy that includes automated and manual tests.
 - Document and track test results to identify issues early.
3. Deployment Procedure:
 - Use Continuous Integration/Continuous Deployment (CI/CD) practices for smooth releases.
 - Test in staging environments before deploying to production.
4. Maintenance Procedure:
 - Regularly update and review documentation and user manuals.
 - Monitor system performance and user feedback for ongoing improvements.

Process Model

We have chosen to use waterfall model for the project because it follows a structured, sequential approach, which works well when requirements like **user authentication** and **course management** are clearly defined from the start. It ensures that each phase (design, development, testing) is completed before moving on, with clear milestones and thorough documentation. This approach minimizes changes and is effective for projects with well-established features.

Project Cost Estimation

We've opted to employ the Basic Model of COCOMO, particularly the "Organic" variant, for our project. Our choice stems from various factors, such as our team's small size and shared comprehension of the project's goals. Furthermore, our team members bring relevant experience from tackling similar challenges in the past. Thus, the Organic model suits our project dynamics, aiding in precise estimation and efficient management of resources and timelines.

$$E = a(KLOC)^b$$

YOUR BASIC COCOMO RESULTS!!								
MODE	"A" variable	"B" variable	"C" variable	"D" variable	KLOC	EFFORT, (in person/months)	DURATION, (in months)	STAFFING, (recommended)
organic	2.4	1.05	2.5	0.38	5.5	14.374477442610152	6.883748480599694	2.0881758656815257


Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm,81). The final estimates are determined in the following manner:

effort = $a * KLOC^b$, in person/months, with KLOC = lines of code, (in the thousands), and:

duration = $c * effort^d$, finally:

staffing = $effort / duration$

For further reading, see Boehm, "Software Engineering Economics", (81)



The model uses a formula to calculate the development effort based on the estimated size.

$$Effort \text{ (in person-months)} = a * (Size \text{ in KLOC})^b$$

Where 'a' and 'b' are coefficients specific to the type of software project being estimated. These coefficients can be derived from the table.

Methods and Techniques

For the entire project, we will utilize **NodeJS** and **ReactJS** for a robust backend and dynamic frontend experience. **ReactJS** will handle the UI with **HTML, CSS, and JavaScript**, ensuring a responsive and interactive user experience for features like course management, discussion forums, and notifications. **NodeJS** will manage the backend, providing API services and handling business logic, while **MySQL** will serve as the relational database for secure storage of user data, course information, and assignment records. Development will be streamlined using **Visual Studio Code** as the IDE, allowing for efficient coding, debugging, and deployment. This stack ensures scalability, maintainability, and a smooth development process.

Quality Assurance

For quality assurance of the project encompassing the specified features, consider the following aspects:

1. **Test Planning:** Define a comprehensive test plan outlining objectives, scope, resources, schedule, and testing types.
2. **Functional Testing:** Verify that each feature (like user authentication, course management, etc.) operates as intended and meets the specified requirements.
3. **Usability Testing:** Assess the user interface and overall user experience to ensure it's intuitive and accessible for all users.
4. **Performance Testing:** Evaluate the system's responsiveness and stability under various load conditions, ensuring it can handle concurrent users and data transactions.
5. **Security Testing:** Conduct penetration testing and vulnerability assessments to identify and mitigate security risks, particularly in user authentication and document sharing.
6. **Integration Testing:** Ensure that all modules work seamlessly together, especially focusing on interactions like notifications and document sharing across features.
7. **Regression Testing:** Implement regular regression tests after updates to confirm that new changes do not adversely affect existing functionalities.
8. **User Acceptance Testing (UAT):** Engage end-users to validate that the system meets their needs and expectations before final deployment.
9. **Continuous Monitoring:** Post-deployment, monitor system performance and user feedback to identify any issues that may arise and address them promptly.

Project Resources

Human Resources: Our development team consists of four skilled members with expertise in both frontend and backend technologies, proficient in HTML, CSS, JavaScript, ReactJS and NodeJS.

Infrastructure Resources:

- **Development Environment:** We have all necessary hardware and software resources in place to efficiently develop and test the website.
- **Database Servers:** The project leverages MySQL for reliable and scalable data storage and management.

Software Resources:

- **Integrated Development Environment (IDE):** Visual Studio Code will be used for coding, debugging, and testing.
- **Version Control System:** Git will manage code versions, ensuring effective collaboration and change tracking.
- **Testing Tools:** We will employ Selenium and the pytest library to rigorously test the website's functionality and performance.

Financial Resources:

- Budget: Adequate financial resources have been allocated to ensure the project stays within budget constraints.

Resource Management:

- Personnel Collaboration: All team members are fully available for the project, ensuring seamless collaboration with no external commitments causing conflicts.
- Time Constraints: With well-aligned schedules, we will avoid delays and ensure on-time project delivery.

Budget and Schedule

We have listed the modules and Calculated the duration using the Pert Equation

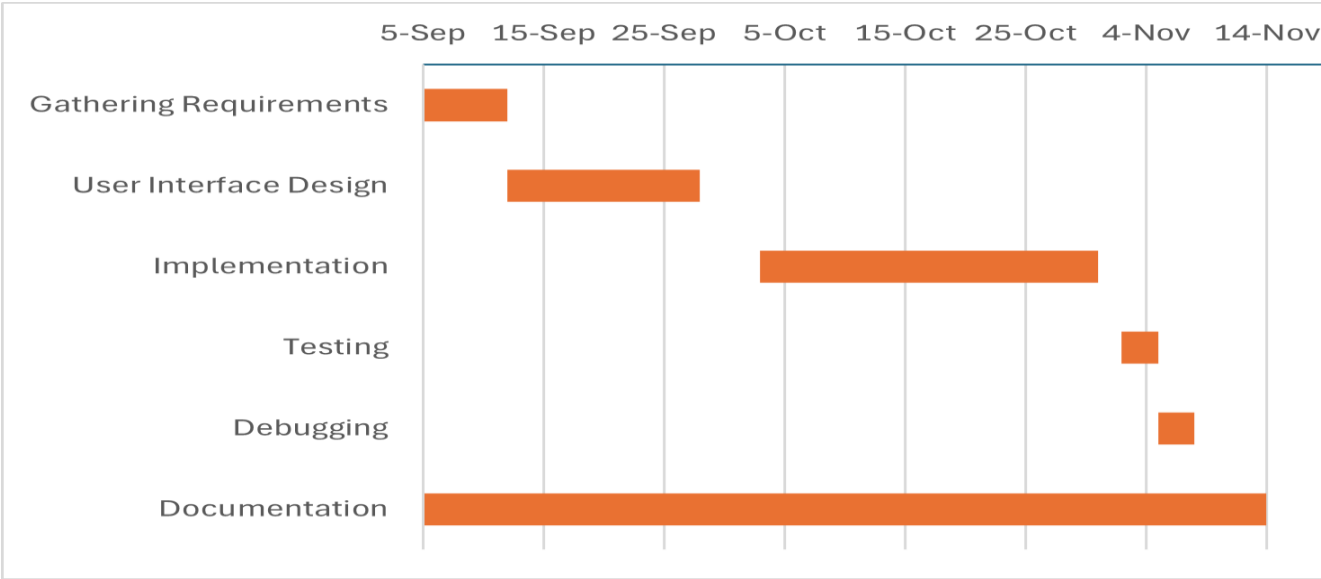
A	B	C	D	E
Task	Optimistic (O)	Most Likely	Pessimistic	Expected Time $(P + 4M + O)/6$
User Authentication , Course Management and Student Enrollment	2 Week	3 Week	4 Week	3 Week
Assignment Submission and Grading	1 Week	2 Week	3 Week	2 Week
Discussion Forum	1 Week	2 Week	3 Week	2 Week
Notification and Document Sharing	1 Week	2 Week	3 Week	2 Week
Completion	5 Week	9 Week	10 Week	9 Week

Which management tool will be used for your project?

We will be using Jira for the project because it supports task management, Agile workflows, and customizable processes, making it easy to track features like user authentication, course management, and assignment grading. It allows for clear organization with epics, stories, and sprints, while providing robust collaboration and communication tools. Jira's reporting, automation, and integration capabilities ensure smooth progress, and its scalability ensures that it can grow with your project as new features like notification systems and discussion forums are added.

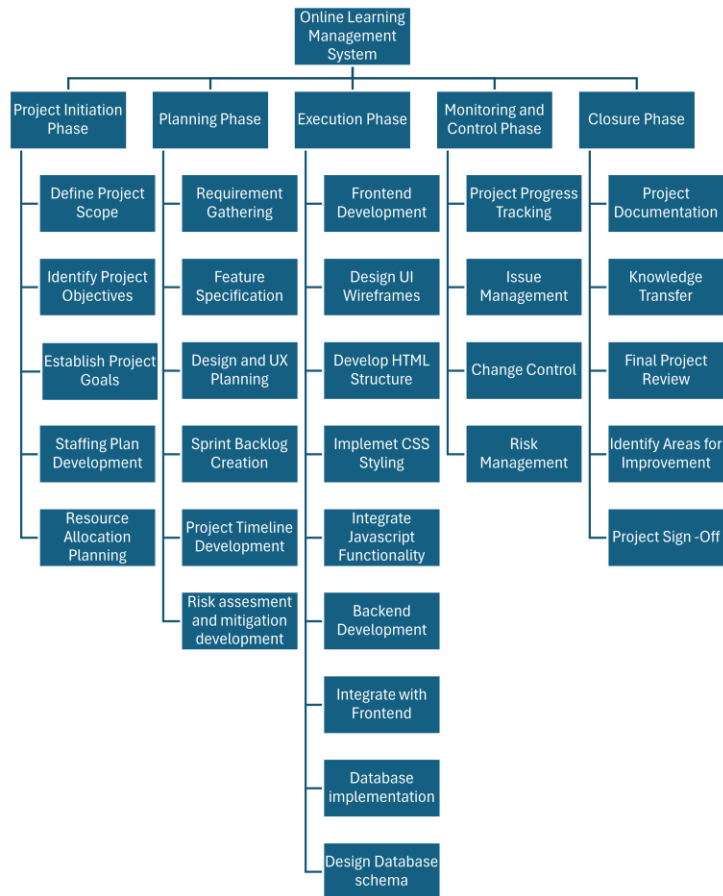
Project Timeline:

Gantt Chart



1	Task	Start Date	End Date	Duration
2	Gathering Requirements	5-Sep	11-Sep	7
3	User Interface Design	12-Sep	27-Sep	16
4	Implementation	3-Oct	30-Oct	28
5	Testing	2-Nov	4-Nov	3
6	Debugging	5-Nov	7-Nov	3
7	Documentation	5-Sep	14-Nov	70

Work Breakdown Structure



Risk Management:

- Risks to Data Security

Hazards: Unauthorized access to sensitive data (e.g., payment and student information) and potential data breaches.

Management: Implement data anonymization, strict access controls, regular security audits, and strong encryption protocols.

- Technical Problems

Hazards: Software bugs, browser or device incompatibilities, and system outages.

Management: Maintain a dedicated IT support team, regularly update systems, and thoroughly test before rolling out changes.

- User Adoption and Engagement

Hazards: Low user participation or resistance to adopting the LMS.

Management: Involve users in decision-making, provide comprehensive training and support, and continuously improve the user interface based on feedback.

- Compliance Risks

Hazards: Non-compliance with education-related regulations (e.g., FERPA, GDPR).

Management: Stay updated on relevant laws, establish policies to ensure compliance, and conduct routine reviews of procedures.

Establish project quality factors and Quality Assurance plan:

Quality Factors

1. **Functionality:** Ensure all features (e.g., user authentication, course management) work correctly.
2. **Reliability:** System must handle heavy usage without failure.
3. **Usability:** Easy and intuitive interface for users across devices.
4. **Performance:** Fast response times, even with large user bases.
5. **Security:** Protect user data and system access (e.g., encryption, secure authentication).
6. **Scalability:** Ability to grow without performance issues.
7. **Maintainability:** Modular, well-documented code for easy updates.
8. **Compatibility:** Works on various devices and browsers.

Quality Assurance (QA) Plan

1. **Testing:**
 - Unit, Integration, Functional, Performance, Security, and Usability testing.
2. **Tools:**
 - Use automated tools (e.g., Selenium) and version control (Git) for managing code and issues.
3. **Defect Tracking:**
 - Track and prioritize bugs using tools like Jira.
4. **Code Reviews:**
 - Peer reviews to catch issues early.
5. **Post-Release:**
 - Monitor performance and user feedback after deployment.

Each member's task (Who is going to do what?)

Student A: Jaydeep Mojidra (Frontend Development - HTML, CSS, JS)

Student B: Pranjal Kamboj (Backend Development - JS, Node.js, React.js)

Student C: Samyak Jain (Backend Development - Database MySQL, Node.js)

Student D: Dhruv Shah (Frontend Development - HTML, CSS , JS)