Digital Egypt Pioneers Initiative

React Front-End Development

Mentorship Connection Platform

Supervised by:

Eng: Basma Abdel Halim

Team members:

Mohamed Essam

Mostafa Gamal Eid

Abdelrhman Magdy

Nouran Said

Baher Mohamed

Acknowledgement

We extend our heartfelt appreciation to Eng. Basma Abdel Halim for her invaluable guidance and unwavering support throughout our journey in the Mentoring Platform Project. Her mentorship has been instrumental in shaping our experience, providing us with the knowledge, insights, and encouragement needed to excel.

Her technical expertise, constructive feedback, and commitment to fostering innovation have played a crucial role in the development and success of this initiative. She has consistently provided thoughtful advice and practical solutions, helping us overcome challenges and refine our approach.

We are truly grateful for the opportunities to learn, grow, and collaborate under her insightful leadership. Her dedication and passion have left a lasting impact on our journey, and we sincerely appreciate her efforts in guiding and inspiring us every step of the way.

Abstract

The growing demand for professional and personal guidance and mentorship has highlighted the urgent need for efficient, reliable, and user-friendly digital platforms to facilitate access to expertise. This project focuses on developing an e-platform for mentorship and guidance services, utilizing modern front-end technologies (such as React or Next.js) to deliver a seamless and responsive user experience. The platform is designed to streamline the connection process between mentors and mentees, ensuring ease of use for both individuals and educational or professional institutions. By integrating a dedicated back-end system, it enables secure profile management, effective appointment scheduling, organization of educational content, and a structured process for progress tracking. The system is tailored to provide a reliable and scalable solution that meets the specific needs of the human and professional development sector.

Through the integration of modern front-end technologies and a well-structured back-end architecture, this project aims to enhance the accessibility and availability of qualified mentors and experts. It contributes to the digital transformation of the knowledge exchange field, offering a practical and innovative approach to connecting individuals with the necessary mentorship resources for their growth and success.

Table of Contents

Project Planning & Management

1.1. Project Proposal Overview:

The project aims to develop a comprehensive digital mentoring application using modern web technologies like React js for the front-end, and a dedicated back-end system (such as Node.js with a database like MongoDB. This application will serve Mentors, Mentees, and Platform Administrators, offering a seamless experience for managing user profiles, scheduling one-on-one sessions, and facilitating efficient knowledge exchange.

Objectives:

- Develop a user-friendly platform where mentees can easily browse and securely book sessions with qualified mentors based on expertise.
- Implement a dedicated Mentor Dashboard for mentors to manage their profiles, set their availability schedules, and track mentees' progress and earnings.
- Enable secure online payment options for paid sessions, with robust tracking and recording of all transactions in the administrative panel.
- Ensure a scalable architecture to accommodate a growing user base and future expansions in features, such as the integration of group sessions and shared resource libraries.

Scope:

The system will have three main interfaces:

1. Mentee Portal:

- Browse and search for mentors using specific filters (expertise, rating, price).
- Book, schedule, and manage one-on-one sessions.
- View session history and track booking status.
- Provide ratings and feedback for completed mentorship sessions.

2. Mentor Dashboard:

- Manage profile details, set working hours, and update expertise areas.
- Accept, reject, or reschedule session requests.
- Monitor session calendars, earnings, and payment history.

3. Admin Dashboard:

- Manage all user accounts (mentors and mentees) and moderate platform content.
- Oversee transactions, payments, and platform fees.
- Generate reports on user engagement and platform performance.

1.2. Project Plan

Timeline:

The project is divided into four main phases to ensure structured and timely development:

- **Phase 1: Planning & Analysis:** (Weeks 1-4) Requirements Gathering, User Story Definition, and initial System Analysis.
- **Phase 2: Design & Architecture:** (Weeks 5-7) UI/UX Design, Database Schema Modeling, and System Architecture Definition.
- **Phase 3: Implementation & Integration:** (Weeks 8-12) Front-end and Back-end Development, and integration of core features like scheduling and profile management.
- **Phase 4: Testing & Deployment:** (Weeks 13-16) Quality Assurance (QA), comprehensive testing of all features (especially scheduling and payments), Bug Fixing, and Final Deployment.

Milestones:

- **UI/UX Design and Database Setup:** Completion of the design system and full database schema deployment.
- **Core Functionality Implementation:** Successful implementation of user authentication (Mentor/Mentee) and profile management features.
- **Scheduling and Payment Integration:** Successful testing of the secure online scheduling tool and payment processing operations.
- **Final Testing and Documentation:** Completion of all QA cycles, successful deployment to production, and submission of final project documentation.

Deliverables:

- A fully functional Mentoring Platform with Mentee Portal, Mentor Dashboard, and Admin interfaces.
- A secure, integrated scheduling and session tracking system.
- A comprehensive Admin dashboard for user management, content moderation, and financial oversight.
- Detailed Documentation & Presentation covering technical, architectural, and user aspects.

1.3. Task Assignment & Roles

Resource Allocation:

- **Development Team:** Project Lead, Front-end Developers, Back-end Developers, UI/UX Specialist.
- **Tech Stack:** React/Next.js, Node.js/Express, MongoDB/PostgreSQL, Tailwind CSS/Bootstrap, Integrated Scheduling API, Payment Gateway.

Responsibilities for team members:

Team Member	Role
[Mohamed Essam] (Project Lead / Full Stack)	Oversee project progress and ensure timely delivery. Define the core system architecture and database schema. Coordinate between development teams and manage deployment pipelines.
[Nouran Said] (Front-End Dev)	Design and implement the core Mentee Portal views (Browse Mentors, Session Booking flow). Ensure the platform is responsive and highly accessible for all users.
[Abdelrhman Magdy] (Front-End Dev)	Develop the Mentor Dashboard interface (Schedule Management, Profile Editing, Session History). Implement authentication and authorization views.
[Mostafa] (Back-End Dev)	Build and maintain the secure Back-end API endpoints for profile management, session booking, and data retrieval. Implement server-side logic and database queries.
[Baher] (Database & Security)	Design and implement the database structure (User, Session, Transaction schemas). Manage the integration of the secure payment gateway and ensure data integrity and security compliance.

1.4. Risk Assessment & Mitigation Plan

Delays in Development:

- Define clear milestones and conduct regular progress reviews (daily stand-ups, weekly summaries).
- Allocate buffer time (10-15% of total project time) in the project timeline for unforeseen challenges.

Integration Complexity (Scheduling & Payments):

- Prioritize development of integration components (scheduling and payment APIs) early in the process.
- Dedicate senior developers to handle external API integration to minimize technical risks.

Quality of Mentorship (Platform Credibility):

- Implement a rigorous mentor vetting process, including profile review and qualification verification.
- Establish a robust mentor/mentee rating and reporting system to maintain quality standards.

1.5. KPIs (Key Performance Indicators):

1. **System Uptime (%):** Ensures system availability and reliability. Target: **99.9% uptime.**
2. **Session Response Time (ms):** Measures how quickly the system responds to session booking requests and profile loading. Target: **< 500ms.**
3. **User Adoption Rate (%):** Percentage of new users (mentees) actively engaging with the platform (e.g., booking a session) within the first month.
4. **Mentor Conversion Rate (%):** Tracks the percentage of mentor applicants who complete the full registration and verification process.
5. **Customer Satisfaction Score (CSAT 1-5):** Based on feedback and reviews submitted by mentees after session completion.
6. **Session Completion Rate (%):** Tracks the percentage of booked sessions that are successfully completed without cancellation by either party.
7. **Scalability & Load Handling:** Performance of the system under peak usage conditions (e.g., 500 concurrent users searching and scheduling).

Literature Review

2.1. Feedback & Evaluation

This section analyzes the existing online mentoring platforms and educational resources, evaluating their strengths in connecting users, managing schedules, and facilitating payments. The review focuses on identifying current gaps in features, user experience, and scalability within the digital mentorship market to inform the design and requirements of this project.

2.2. Suggested Improvements

Enhanced Mentor/Mentee Matching System

Implement an advanced matching algorithm based on specific skills, professional goals, and preferred learning styles to improve the quality of mentor-mentee connections. This includes detailed profile fields and a search interface that goes beyond basic keywords.

Integrated Scheduling & Progress Tracking

Develop a robust, unified calendar system that automatically handles different time zones and provides pre-session reminders. Introduce a progress tracking feature that allows mentors and mentees to collaboratively log session outcomes, milestones achieved, and set goals for future sessions.

Flexible Session & Payment Models

Implement support for various session formats (e.g., individual, quick consultation calls, group webinars) to enhance flexibility. Introduce adaptive pricing models (per-session fee, subscription packages) and multi-currency support to accommodate a global user base.

Admin Dashboard & Quality Control Enhancements

Add visual analytics (charts & reports) tracking key metrics such as user engagement, session completion rates, and mentor activity. Enable platform administrators to conduct quality control audits and utilize detailed session feedback to ensure high mentor performance and platform credibility.

2.3. Final Grading Criteria

Requirements Gathering

3.1. Stakeholder Analysis

This section identifies all key parties invested in the Mentoring Platform Project, analyzing their specific interests and needs to inform system design.

- **Mentees (End-Users):** Need an easy-to-use platform to browse mentors, securely book and manage sessions, and receive clear communication/updates.
- **Mentors (End-Users):** Require a dedicated dashboard to manage their profiles, set flexible availability, approve/reject bookings, and monitor their earnings and session history.
- **Administrators (Platform Staff):** Need a comprehensive dashboard to manage all users, moderate content, track real-time activity, and oversee financial transactions and platform quality.
- **Sponsors/Partners (Funding Body):** Expect clear reports on progress, demonstration of platform impact, and compliance with project goals.

3.2. User Stories & Use Cases

This section outlines the functional needs of the system from the perspective of the different user roles.

User Stories:

- **As a Mentee, I want to filter mentors by expertise and rating, so that I can quickly find a qualified professional.**
- **As a Mentor, I want to sync my platform calendar with my external calendar (e.g., Google Calendar), so that my availability is always accurate.**
- **As an Administrator, I want to access real-time session booking and cancellation statistics, so that I can monitor platform traffic and detect anomalies.**
- **As a Mentee, I want to leave a detailed review and rating for my mentor, so that other mentees can make informed decisions.**

3.3. Functional Requirements

This section details the specific functions the system **must** perform.

- **F.R. 1.0 (User Authentication):** The system SHALL allow users to register, log in, and log out securely based on their role (Mentee, Mentor, Admin).
- **F.R. 2.0 (Search & Filter):** The system SHALL provide advanced search functionality for mentors based on skills, price, rating, and language.
- **F.R. 3.0 (Session Management):** The system SHALL enable Mentees to book available time slots and receive automated email/in-app notifications for confirmation, reminders, and cancellations.
- **F.R. 4.0 (Payment):** The system SHALL securely process payments for sessions using a recognized payment gateway (e.g., Stripe or similar) and record all transactions.
- **F.R. 5.0 (Admin Panel):** The system SHALL allow Administrators to view, approve, suspend, or delete any user account and manage platform settings.

3.4. Non-functional Requirements

This section details the quality attributes, constraints, and performance criteria.

- **N.R. 1.0 (Performance):** The Mentor Search results SHALL be displayed in less than **2 seconds** for all users.
- **N.R. 2.0 (Security):** The system SHALL protect all sensitive user data (passwords, payment information) using industry-standard encryption protocols (e.g., TLS/SSL, hashing).
- **N.R. 3.0 (Usability):** The platform SHALL offer a modern, responsive interface that is fully functional across all major desktop and mobile devices.
- **N.R. 4.0 (Scalability):** The system SHALL be built to handle up to **10,000 active users** and process **1,000 sessions per day** without degradation in performance.

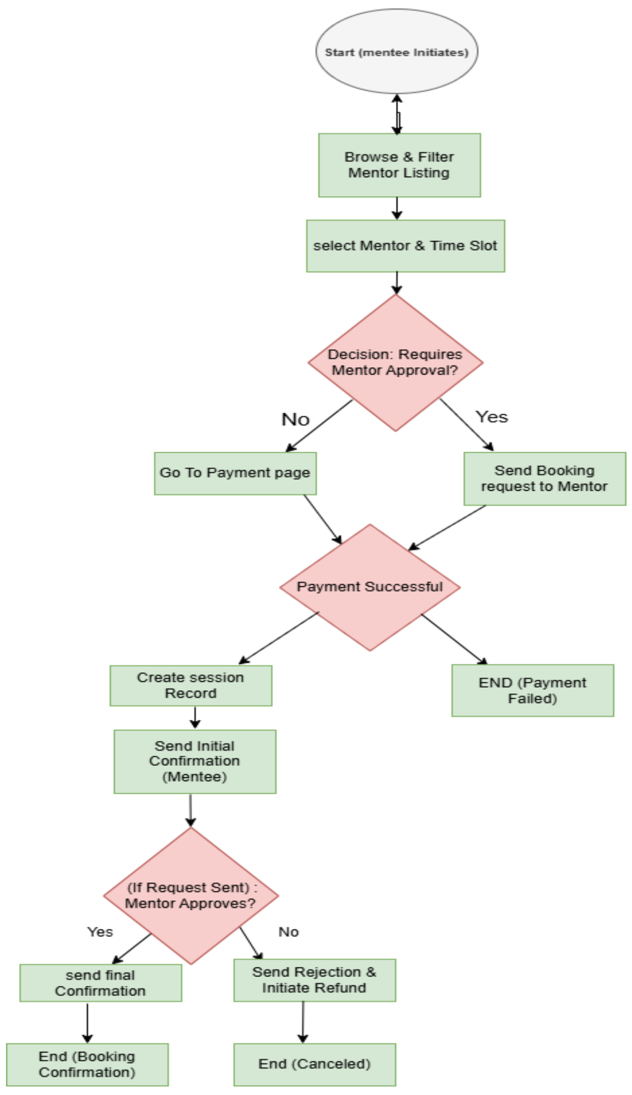
System Analysis & Design

4.1. Problem Statement & Objectives Problem

Statement:

Effective mentorship is often hindered by the challenges of connecting individuals with the right expertise due to scattered professional networks, inefficient manual scheduling, and the lack of a centralized system for quality assurance. The difficulty in tracking progress over multiple sessions and ensuring secure payment processing also limits the growth and reliability of independent online mentorship.

This project aims to build a scalable, secure, and user-friendly digital platform that streamlines mentor-mentee matching, automates scheduling and secure payment processing, and provides administrators with the necessary tools for effective quality control and user management.


Start (mentee Initiates)
Browse & Filter Mentor Listing
select Mentor & Time Slot
Decision: Requires Mentor Approval?
No
Yes
Go To Payment page
Send Booking request to Mentor
Payment Successful
Create session Record
END (Payment Failed)
Send Initial Confirmation (Mentee)
(If Request Sent) : Mentor Approves?
Yes
No
send final Confirmation
Send Rejection & Initiate Refund
End (Booking Confirmation)
End (Canceled)

4.2. Database Design & Data Modeling

This section defines the logical structure of the Mentoring Platform's database, ensuring data efficiency, security, and integrity. This is achieved by modeling the system's entities and their relationships using the Entity-Relationship Diagram (ERD).

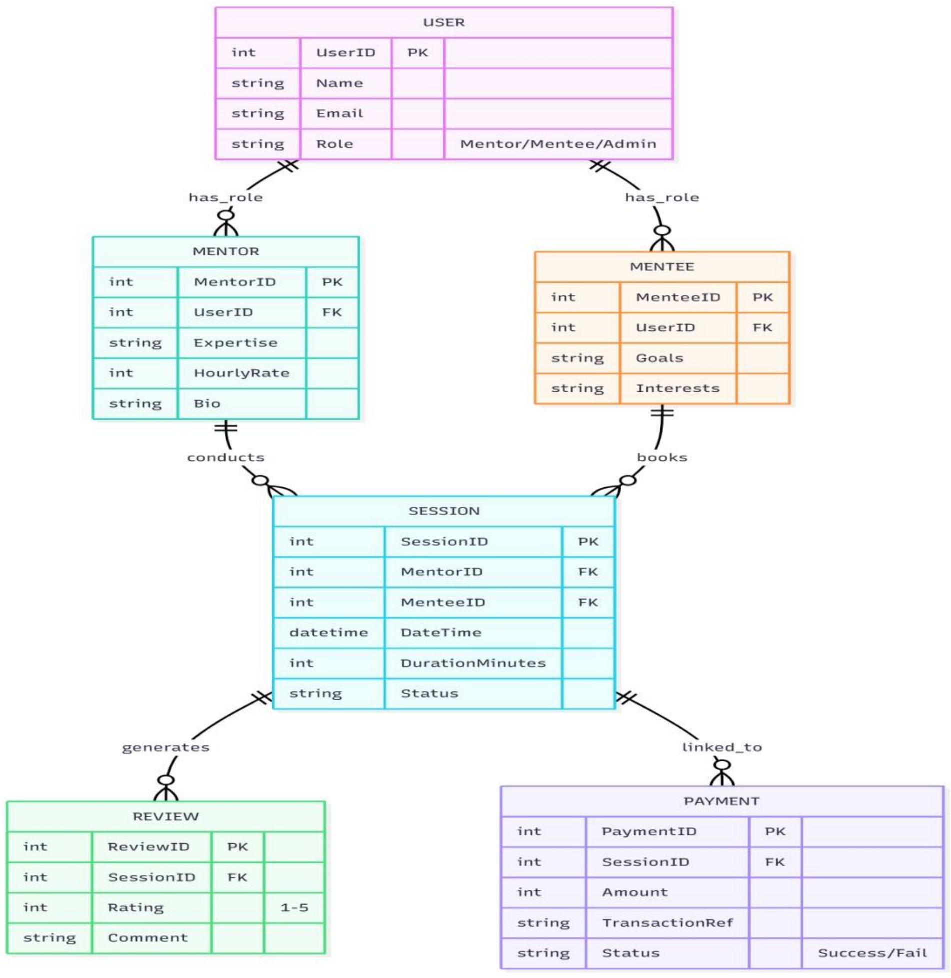
Core Entities for Mentoring Platform:

Entity	Description
User	Represents all users (Mentor, Mentee, Admin). Stores core data such as ID, Name, Email, and encrypted Password.
Mentor	Represents mentors (an extension of the User entity). Includes specialized information like Expertise, Hourly Rate, Bio, and overall Rating.
Mentee	Represents mentees (an extension of the User entity). Stores data such as learning goals and professional interests.
Session	Represents a booked mentorship session. Contains Session ID, Mentor ID, Mentee ID, Date and Time, Duration, and Status (Upcoming, Completed, Canceled).
Review	Represents the feedback submitted by a mentee for a completed session. Contains Review ID, Session ID, Rating score, and Comment text.
Payment	Represents the financial transaction details for a session. Contains Payment ID, Session ID, Amount, Payment Status (Success/Fail), and Transaction Reference.

Proposed Relationships:

- **User — Mentor / Mentee:** A one-to-one relationship (inheritance) where every user has a specific defined role.
- **Mentor — Session:** A one-to-many relationship, as one Mentor can conduct multiple Sessions.
- **Mentee — Session:** A one-to-many relationship, as one Mentee can book multiple Sessions.

- **Session — Review:** A one-to-one relationship, as each completed session is linked to one review.
- **Session — Payment:** A one-to-one relationship, as each session is tied to one payment record.



USER			
int	UserID	PK	
string	Name		
string	Email		
string	Role		Mentor/Mentee/Admin

MENTOR		
int	MentorID	PK
int	UserID	FK
string	Expertise	
int	HourlyRate	
string	Bio	

MENTEE		
int	MenteeID	PK
int	UserID	FK
string	Goals	
string	Interests	

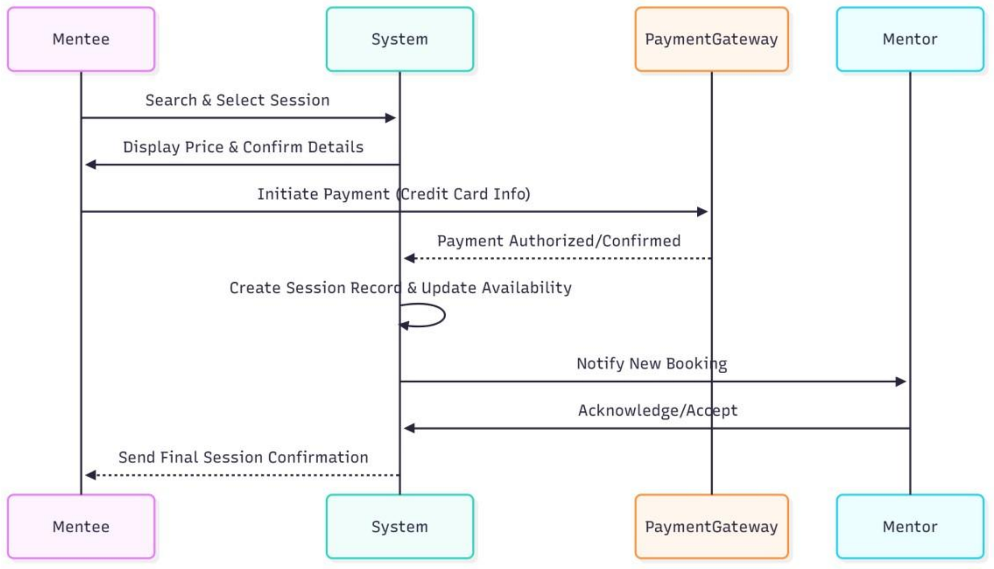
SESSION		
int	SessionID	PK
int	MentorID	FK
int	MenteeID	FK
datetime	DateTime	
int	DurationMinutes	
string	Status	

REVIEW			
int	ReviewID	PK	
int	SessionID	FK	
int	Rating		1-5
string	Comment		

PAYMENT			
int	PaymentID	PK	
int	SessionID	FK	
int	Amount		
string	TransactionRef		
string	Status		Success/Fail

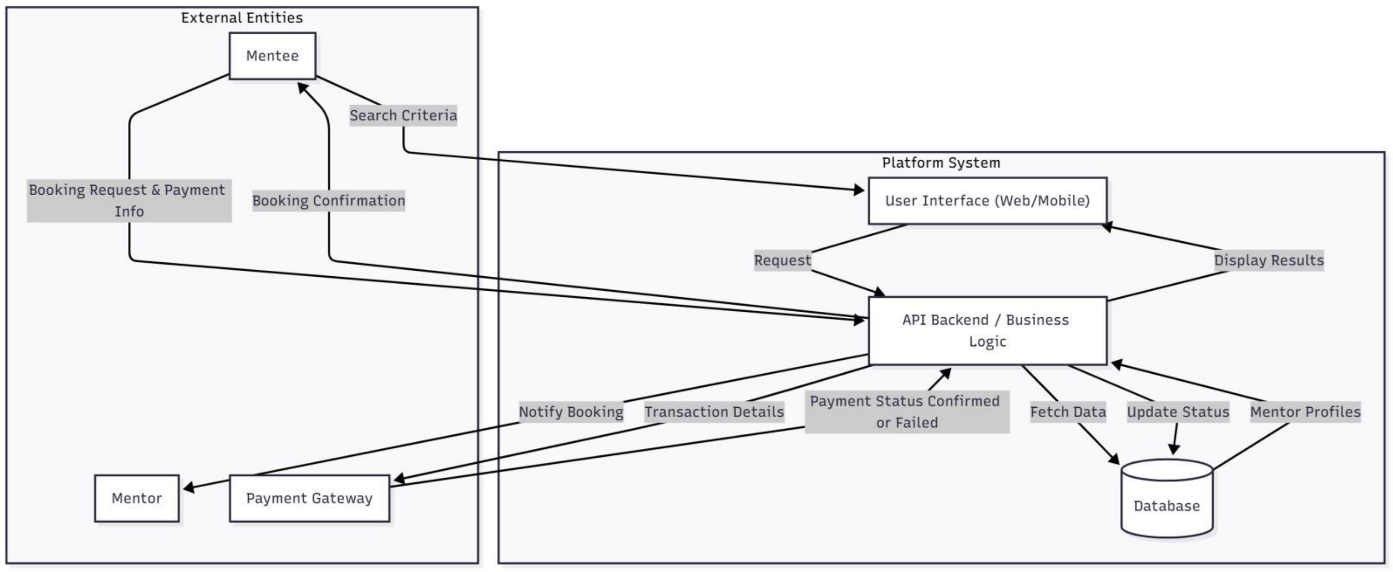
4.3. Data Flow & System Behavior

Sequence Diagram

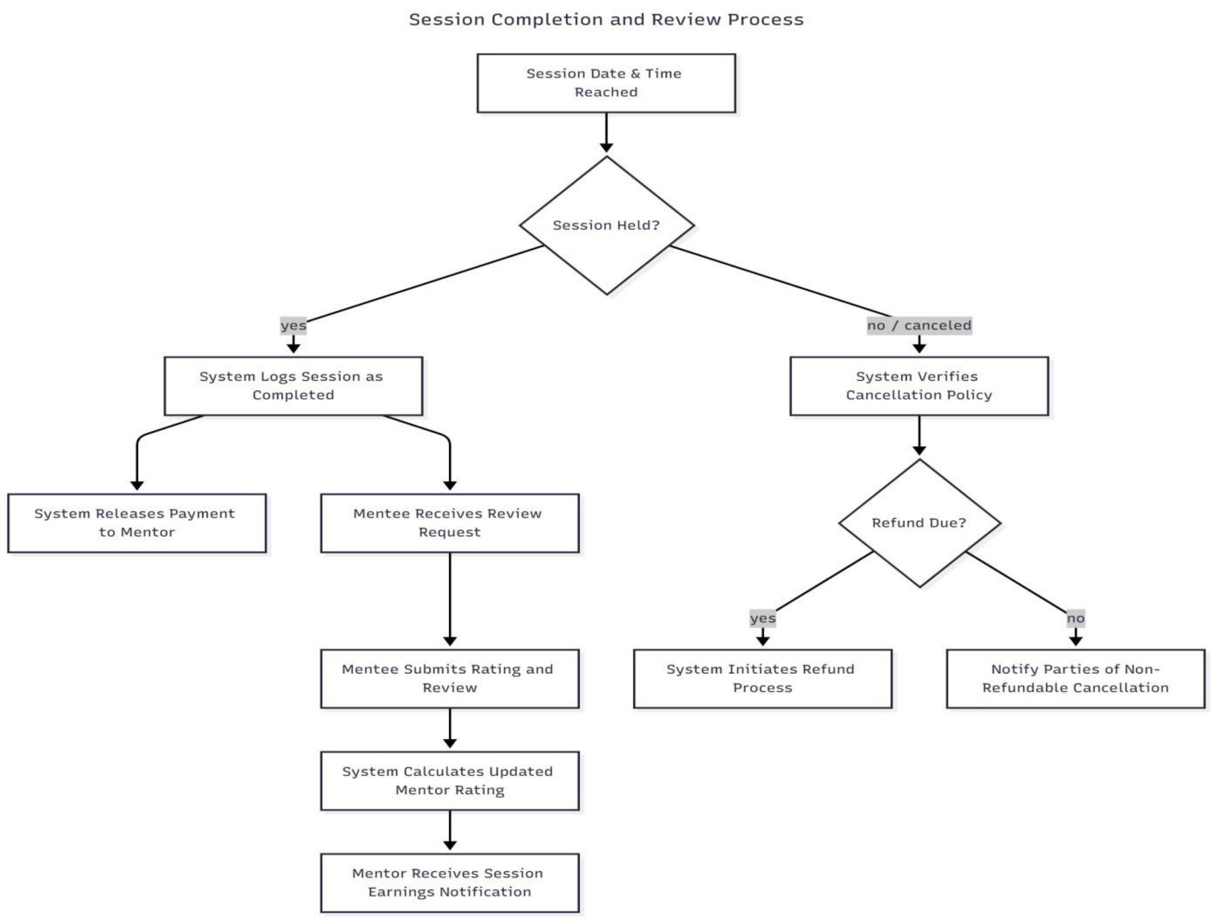


Mentee
System
PaymentGateway
Mentor
Search & Select Session
Display Price & Confirm Details
Initiate Payment (Credit Card Info)
Payment Authorized/Confirmed
Create Session Record & Update Availability
Notify New Booking
Acknowledge/Accept
Send Final Session Confirmation

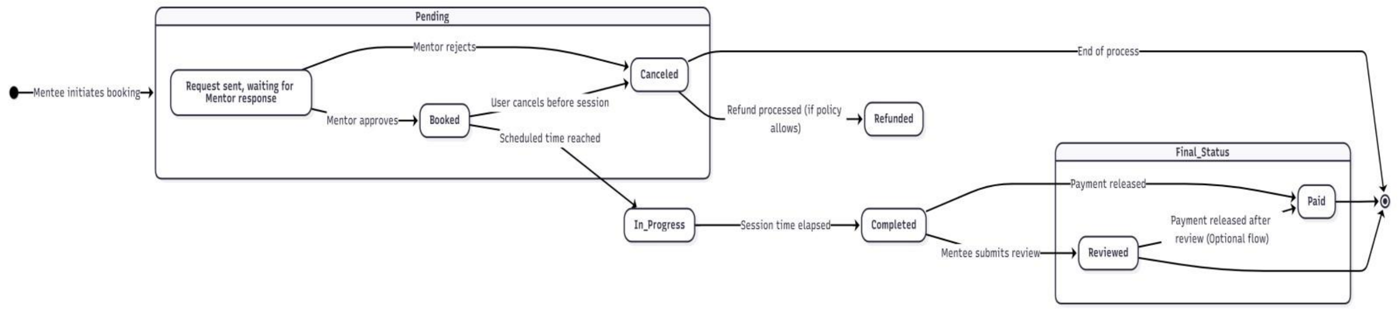
Data Flow Diagram



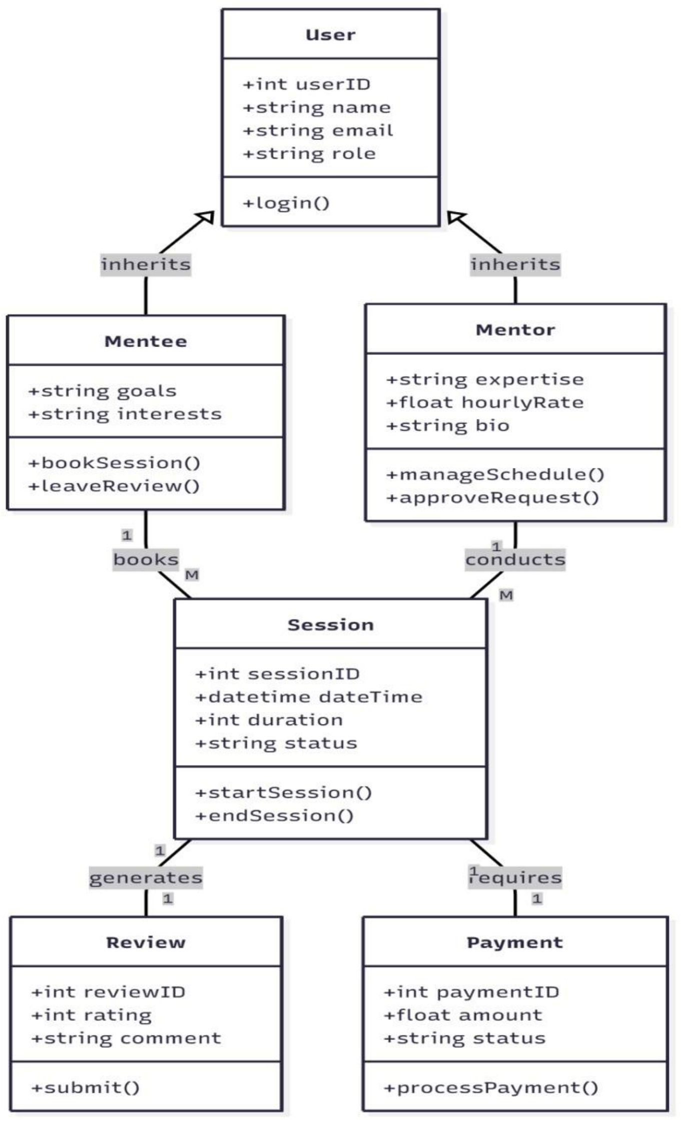
Activity Diagram:


Session Completion and Review Process
Session Date & Time Reached
Session Held?
yes
no / canceled
System Logs Session as Completed
System Verifies Cancellation Policy
System Releases Payment to Mentor
Mentee Receives Review Request
Refund Due?
yes
no
Mentee Submits Rating and Review
System Initiates Refund Process
Notify Parties of Non-Refundable Cancellation
System Calculates Updated Mentor Rating
Mentor Receives Session Earnings Notification

State Diagram:



Class Diagram:



4.4. UI/UX Design & Prototyping

The success of the Mentoring Platform relies heavily on providing an intuitive and seamless user experience for both Mentors and Mentees. This section outlines the design strategy and the prototyping phase of the project.

Design Philosophy and Strategy

The UI/UX design adheres to core principles focused on user-centricity, accessibility, and responsiveness across all devices. The primary goal is to create clear, goal-oriented navigation to streamline key actions, such as booking a session or updating a mentor's schedule.

Prototyping Deliverables

The visual design process involved translating initial low-fidelity wireframes into highfidelity prototypes.

- **App Wireframes:** The functional layouts and information hierarchy were defined through comprehensive wireframes. The detailed wireframes are available for review.
- **Our Idea Basis:** The design concepts and user flows were adapted and refined based on established systems and usability standards.

4.5. System Deployment & Integration

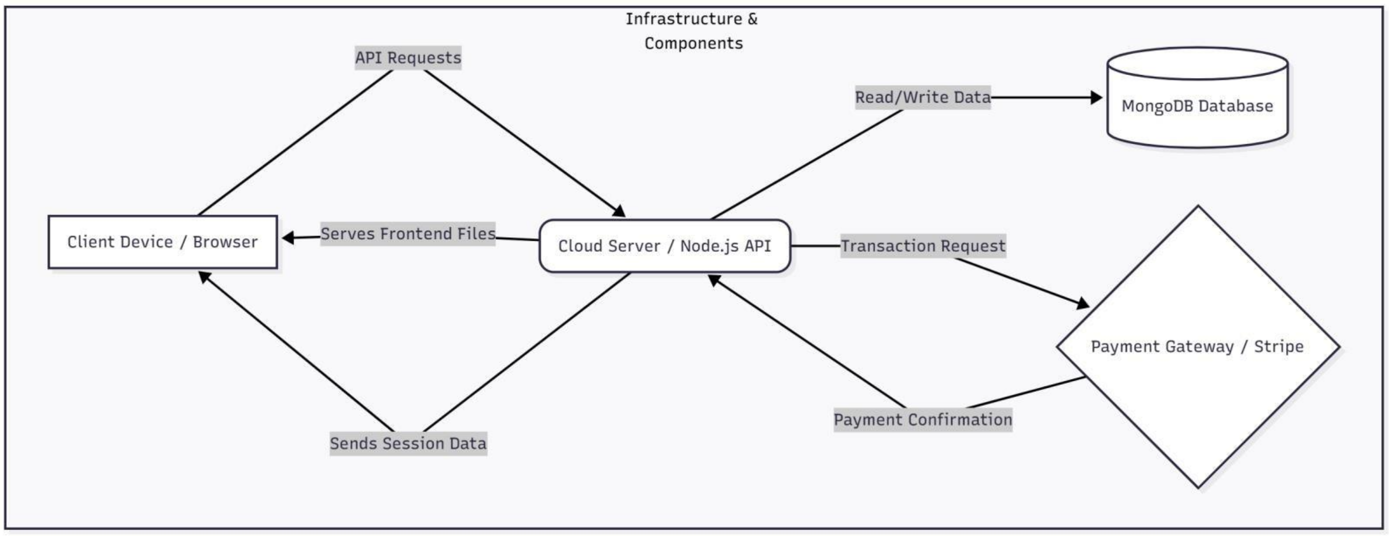
This section defines the infrastructure, architecture, and strategy used to deploy the Mentoring Platform and integrate it with essential third-party services.

Technology Stack

The platform is built on the following technologies:

- **Frontend:** Node.js / React-Bootstrap
- **Backend:** Node.js / Express.js
- **Database:** MongoDB (NoSQL)
- **Payment:** Stripe, Credit Card
- **Deployment:** Vercel **Deployment Diagram**

The **Client Device** hosts the React App, which runs in the user's browser and interacts with the backend through API calls. The **Cloud Server** houses the Node.js API and Express.js framework, handling business logic, authentication, and communication with the database. **MongoDB** serves as the primary database, storing user data, **session bookings**, **mentor profiles**, and payments. **Payment Gateways (Stripe & PayPal)** are integrated with the backend to handle secure payment processing. The **Admin Dashboard Server** provides an interface for administrators to manage users, **sessions**, and **mentor availability**.



5.1. Source Code

This section provides an overview of the platform's codebase structure and highlights key segments of code that represent the core business logic of the Mentoring Platform. The entire source code is maintained and managed in the project's GitHub repository (refer to Section 5.2).

Code Structure Overview

The codebase is organized into a modular structure to facilitate maintainability and scalability, following a clear separation of concerns (MVC architecture elements):

- **Client (Frontend):** Located in the `/client` directory. This component uses **React** and **React-Bootstrap** and is responsible for all UI/UX components (Mentor Search, Mentee Dashboard, etc.).
- **Server (Backend/API):** Located in the `/server` directory. This utilizes **Node.js** and **Express.js** and contains all API routes, controllers, services, and handles business logic and authentication.
- **Database Models:** Schemas are defined for the **MongoDB** database, representing the core entities: User, Mentor, Mentee, Session, and Payment.

Key Code Snippets

To illustrate the critical functionality of the system, specific code snippets are presented below, focusing on the platform's unique logic:

1. **Session Booking Controller:** Code demonstrating the logic for checking mentor availability, reserving a slot, and initiating the payment transaction.
2. **User Authentication Middleware:** Code snippet showing how user roles (Mentor, Mentee, Admin) are verified before granting access to specific API endpoints.

3. **MongoDB Schema (e.g., Session Model):** The code defining the structure of the `Session` data, including foreign key references to `Mentor` and `Mentee`.

5.2. Version Control & Collaboration

Effective collaboration and meticulous tracking of code changes were essential for the development team. The project utilizes industry-standard tools and practices to maintain code integrity and facilitate concurrent development.

Version Control System (VCS)

- **Tool:** Git (for local version control) and GitHub (for remote repository hosting).
- **Purpose:** The GitHub repository serves as the single source of truth for the entire codebase (Frontend, Backend, and Database Models). It ensures that all team members have access to the latest, stable version of the code and maintains a complete history of all changes.

Collaboration & Workflow

- **Branching Strategy:** The team adopted the **Feature Branching** model to manage concurrent development effectively: ○ **`main` Branch:** Holds only the production-ready, stable code. Direct commits are forbidden.
  ○ **`development` Branch:** The primary branch for integrating and rigorously testing all new features before deployment. ○ **Feature Branches:** Each new feature, bug fix, or task is developed in a dedicated, isolated branch (e.g., `feature/session-review`, `fix/payment-error`).
- **Code Review Process:** All code merged into the `development` branch requires a **Pull Request (PR)**. This PR must be reviewed and approved by at least one other team member before merging to enforce code quality and minimize integration bugs.
- **Issue Tracking:** Tasks and issues were managed using a project management tool (e.g., ClickUp, as suggested by the dashboard screenshot) to link code changes directly back to the assigned tasks and team roles.

5.3. Deployment & Execution

This section outlines the process and environment used to deploy the Mentoring Platform and execute the application for testing and production use. This process complements the architecture defined in Section 4.5.

Deployment Environment

- **Frontend Platform:** Vercel (used for static site hosting and fast distribution of the React application).
- **Backend Platform:** Cloud Server (e.g., AWS or DigitalOcean) used for hosting the robust Node.js/Express.js API.
- **Database:** MongoDB Atlas (A cloud-hosted database service ensuring scalability and reliability).

Execution Steps

1. **Backend Setup:** The Node.js application was configured with all necessary environment variables (Database connection URL, external API keys for Stripe/email). Dependencies were installed using `npm install`.
2. **Continuous Integration:** The CI/CD pipeline ensures that any approved merge into the `development` branch triggers an automatic build and update to the staging environment.
3. **Backend Deployment:** The server repository was linked to the cloud platform, which automatically executes the setup commands and starts the Node.js server instance.
4. **Frontend Deployment (Vercel):** The client repository was linked to Vercel. Vercel automatically detects the React framework, builds the application, and deploys it to a global CDN for optimal loading times.
5. **Testing Environment:** A dedicated staging environment was maintained throughout the project lifecycle for Quality Assurance (QA) testing before pushing features to the live production site.

6.1. Test Cases and Test Plan

Testing and Quality Assurance (QA) are critical phases aimed at verifying that the Mentoring Platform meets all functional and non-functional requirements and operates free of critical defects. The overall testing strategy includes Unit Testing, Integration Testing, and User Acceptance Testing (UAT).

Test Plan Overview

- **Objective:** To ensure all core functionalities (User Registration, Session Booking, Payment Processing, Review Submission) work as designed. The plan aims to maintain system stability and guarantee reliable performance (Target: 99.9% uptime) and quick response times (Target: $< 500ms$).

- **Methodology:** A combination of black-box testing (focused on functionality from a user's perspective) and white-box testing (focused on code logic and structure) was employed.
- **Testing Environment:** Testing was primarily conducted in a dedicated staging environment that mirrors the production setup.

Sample Test Cases

The following table provides a representative sample of test cases executed for the platform's core functionalities:

ID	Module	Test Case Description	Expected Result	Pass/Fail
TC-001	Login/Auth	Attempt login with valid Mentee credentials.	User successfully logs in and is redirected to the Mentee Dashboard.	
TC-002	Booking	Mentee attempts to book a session at a time marked 'Unavailable' by the Mentor.	The system displays an error message: "This slot is unavailable." Booking fails.	
TC-003	Payment	Mentee initiates booking and completes payment using Stripe's test card details.	Stripe processes the transaction, the Session Status changes to 'Booked,' and both parties receive confirmation emails.	
TC-004	Review	Mentee submits a review (4/5 stars) for a session logged as 'Completed.'	The review is successfully saved, and the Mentor's overall rating is updated instantly or in the next batch job.	
TC-005	Admin	Admin attempts to view a list of all Canceled Sessions.	A filtered list of Canceled Sessions, including reasons, is displayed correctly.	

6.2. Automated Testing

To ensure code stability and maintain the high quality standards required for continuous deployment, the platform implements a robust strategy for automated testing. This automation is vital for achieving the target system uptime and response time KPIs.

Objectives of Automation

- **Efficiency:** Automated tests dramatically reduce the time required for regression testing after every major code change, which is crucial given the CI/CD pipeline and frequent updates.
- **Reliability:** Ensure that core business logic, such as secure payment processing, scheduling, and user authentication, remains functional across all system updates.

Tools and Scope

The automation effort is segregated by the component being tested:

Test Type	Scope	Tool/Framework Used	Description
Unit Testing	Individual functions, classes, and components (Frontend/Backend).	**Jest / Mocha**	Ensures that isolated pieces of code perform their logic correctly (e.g., calculating the final session cost).
Integration Testing	API endpoints and database communication.	**Supertest** (for Node.js Backend)	Verifies that services interact correctly (e.g., checking if the API successfully creates a session record in MongoDB).
End-to-End (E2E) Testing	User workflows across the UI.	**Cypress / Selenium**	Simulates real user behavior, such as a Mentee successfully searching for a Mentor and completing the entire booking and payment process.

Integration with CI/CD

Automated tests are integrated directly into the CI pipeline (as defined in Section 5.2). Tests are configured to run automatically upon every Pull Request, preventing the merge of code that introduces regression errors, thereby enforcing quality assurance continuously.

6.3. Bugs

Bug tracking and management were handled as a critical part of the CI/CD pipeline and collaboration workflow (Section 5.2). This process was designed to meet the requirement of quickly detecting and resolving security incidents and vulnerabilities.

Bug Reporting and Tracking Process

1. **Identification:** Bugs were primarily identified during automated testing (Section 6.2) and manual QA testing, or reported directly by stakeholders during the User Acceptance Testing (UAT) phase.
2. **Reporting Tool:** All issues were logged using a project management tool (e.g., GitHub Issues or ClickUp), providing a central repository for bug tracking.
3. **Prioritization:** Bugs were categorized based on severity (Critical, High, Medium, Low) and assigned to the relevant team member (Frontend, Backend, or Full Stack) for resolution.
4. **Resolution:** The assigned developer created a dedicated feature branch for the fix, committed the changes, and submitted a Pull Request (PR) for code review (Section 5.2).
5. **Verification:** Once merged, the fix was automatically deployed to the staging environment for retesting and final verification by the QA team before deployment to production.

Sample Bug Log

The table below provides an example of a bug identified and resolved during the testing phase:

ID	Title	Priority	Status	Assigned To	Resolution
BUG-042	Stripe webhook failure during refund process.	High	Closed	Backend Team	Implemented secure webhook signature verification logic.
BUG-078	Mentor dashboard schedule not updating instantly after session cancellation.	Medium	Closed	Frontend Team	Implemented real-time state management using Context API to refresh schedule component.

7.1. User Manual

This section serves as a comprehensive guide for end-users (Mentees and Mentors), detailing how to effectively interact with and utilize the core functionalities of the Mentoring Platform.

I. Mentee Portal (Client Guide) 1. Access the Platform

Open a web browser and navigate the live application URL (`[https://mentoring-platform-topaz.vercel.app/]`).

2. Create an Account / Login

- **Sign Up:** If you are a new user, click on the "Sign Up" button, fill in your details (Name, Email, Password), and click "Register."
- **Login:** If you already have an account, click "Login," enter your credentials (Email, Password), and access your personalized dashboard.

3. Browse and Search for Mentors

- You can browse the available Mentors by category (expertise area) or search for specific skills using the search bar.
- Filters are available to refine this search based on criteria such as expertise, price range, or average rating.

4. Session Booking & Checkout

- Once you find a suitable Mentor, click on their profile to view their detailed bio and schedule.
- Select an available date and time slot, then click "Proceed to Checkout."
- Review your session details and select your payment method (Credit Card via Stripe, PayPal, or other options), confirm your billing details, and complete the payment process.

5. Session Confirmation & Review

- After successfully placing your booking and processing the payment, a confirmation message will appear with the session summary.
- A detailed confirmation will be automatically generated and sent to your registered email.
- After the session is marked as 'Completed' by the system, you will receive a prompt to submit a rating and a review for your Mentor.

II. Mentor Dashboard Guide 1. Profile Management

- Navigate to the "Profile Settings" section to update your expertise, bio, and hourly rate.
- Upload a professional profile picture to enhance your visibility.

2. Schedule Management

- Access the "Availability Calendar" to set your working hours and block out times when you are unavailable.
- Review the list of incoming booking requests and select to **Approve** or **Reject** them.

3. Earnings and Payouts

- View your "Earnings Report" to track revenue generated from completed sessions.
- Monitor pending payments and verify successful payouts processed by the platform.

7.2. Technical Documentation

This section provides a consolidated summary of the key architectural components, design decisions, and data modeling strategies implemented throughout the Mentoring Platform project.

I. Core Architecture and Components (Ref: 4.5 System Deployment)

The platform utilizes a modern, scalable architecture based on the MERN stack principles. The **Frontend** is built using Node.js and React-Bootstrap, designed to deliver a responsive user interface. The core logic resides in the **Backend API**, powered by Node.js and Express.js, which enables fast, asynchronous handling of concurrent requests and business logic execution. For **data storage**, MongoDB (NoSQL) is used, offering flexibility and scalability for managing diverse data types like user profiles and session details. **Deployment** is handled using Vercel for the Frontend and a Cloud Server for the Backend API, supporting a robust CI/CD pipeline.

II. API Endpoints Summary

The backend exposes a RESTful API structure to manage all application data and logic. Key functionalities are managed through specific endpoints:

- **User/Authentication:** This includes routes for user registration (`POST /api/auth/register`) and secure login (`POST /api/auth/login`) to manage access for both Mentors and Mentees.
- **Mentor Management:** Routes are dedicated to retrieving mentor profiles (`GET /api/mentors`), enabling search and filtering functionality, and allowing authenticated mentors to update their schedules (`PATCH /api/mentors/:id/schedule`).
- **Session Management:** Core functionality is handled by routes such as initiating a new session booking (`POST /api/sessions/book`), retrieving session lists, and managing status updates (e.g., approval/rejection) (`PATCH /api/sessions/:id/approve`).
- **Payment & Review:** Dedicated endpoints manage secure transaction processing (`POST /api/payments/charge`) and allow Mentees to submit ratings and feedback for completed sessions (`POST /api/reviews`).

III. Data Modeling Summary (Ref: 4.2 Database Design)

The project's logical structure is fully defined by the Entity-Relationship Diagram (ERD) and Class Diagram. The **Core Entities** modeled include User, Mentor, Mentee, Session, Payment, and Review. The design establishes clear **relationships** such as Mentor (one-to-many) Sessions and Session (one-to-one) Payment/Review to maintain data integrity. The data structure is implemented using MongoDB Schemas, as detailed in the source code models

7.3. Presentation

The final project presentation serves as a comprehensive overview of the Mentoring Platform's development lifecycle, key features, and successful outcomes. The presentation is designed to be concise yet thorough, highlighting the realization of the project's initial objectives.

Presentation Structure

The presentation will cover the following critical areas:

1. **Introduction & Problem Statement (Ref: 4.1):** Briefly introduce the team (Ref: Section 1.3) and define the problem the Mentoring Platform aims to solve (i.e., streamlining online purchasing, payment, and administration management).
2. **Solution & Design (Ref: 4.3, 4.4):** Showcase the application's UI/UX design and the core architectural diagrams (DFD, Deployment Diagram, Class Diagram) used for implementation.
3. **Technology Stack & Implementation (Ref: 4.5, 5.1):** Detail the choice of **React**, **Node.js/Express**, and **MongoDB** and the robust CI/CD deployment strategy used.
4. **Key Features & Demo:** Provide a live demonstration of the core functional requirements, including the secure session booking and payment flow.
5. **Testing & Quality (Ref: 6.1, 6.2):** Summarize the testing methodology and how quality assurance was achieved, referencing the target KPIs (e.g., 99.9% uptime, $< 500ms$ response time).

6. **Conclusion & Future Scope:** Summarize the project's success against the initial objectives and briefly discuss potential future features (e.g., advanced analytics, expanded payment options).

Final Conclusion

The successful deployment of the Mentoring Platform demonstrates the team's ability to transition from conceptual design to a fully functional, scalable, and secure system, fulfilling all requirements outlined in this project documentation.