Introduction to Software Engineering

Project Proposal

The student team is required to complete the **Project Proposal** documentation for the assigned course project, following the attached template.

Software Engineering Department

Faculty of Information and Technology

University of Science

Table of Contents

Project Proposal

Objectives

This document focus on the following topics:

- ✓ Completing the Project Proposal document with the following sections:
  - Preliminary Problem Statement
  - Proposed Solution
  - Development Plan
  - Human Resources & Costing Plan
- ✓ Understanding the Project Proposal document.

1 Member Contribution Assessment

ID	Name	Contribution (%)	Signature
23127086	Huỳnh Sĩ Luân	100%	Luân
23127148	Ân Tiến Nguyên An	100%	[illegible]
23127212	Nguyễn Quang Đăng Khoa	100%	[illegible]
23127280	Nguyễn Hiền Tuấn Anh	100%	[illegible]
23127442	Trầm Hữu Nhân	100%	[illegible]

2 Preliminary Problem Statement

The Online Project Management System is a platform designed to help individuals and organizations efficiently plan, assign, monitor, and collaborate on work. The system allows users to create and manage multiple projects, each consisting of various tasks that can be assigned to specific team members. Its primary goal is to help teams optimize their workflow, track progress in real time, and ensure projects are completed on schedule. Through intuitive interfaces and collaborative tools, the system enhances coordination, transparency, and productivity across all stages of project development.

- **User Management**

The online project management system allows users to create accounts, log in, and participate in multiple projects simultaneously. Each user account contains personal information such as name, email, password, profile image, and assigned role within the organization. Users can update their profiles, change passwords, and enable two-factor authentication (2FA) to enhance security. Additionally, users are granted different access permissions based on their role in each project, such as Administrator (Admin), Project Manager, or Team Member. Each role has different rights in performing actions like creating projects, editing tasks, and tracking progress. The system also supports notification management so users can receive the latest updates whenever a new task, comment, or status change occurs in any project.

- **Project Creation and Management**

Users can create new projects by entering detailed information such as name, description, start and end dates, and a list of project members. Once a project is created, the system automatically generates a Kanban Board with default columns like “To Do,” “In Progress,” and “Done.” The Project Manager can add or remove members, assign permissions for each participant, and track overall progress of the team. Projects can be edited, paused, or closed upon completion. The system also enables users to search for projects quickly by name, manager, or current status. This helps users manage multiple projects efficiently and collaborate more effectively across teams.

- **Task Management**

Within each project, members can create and manage tasks of various types. Each task includes details such as title, description, assignee, deadline, priority level, and current status. Users can attach files, images, or create sub-checklists within tasks for better organization. The system supports direct commenting inside each task, allowing members to @tag teammates for discussion and quick updates. Tasks are visually represented on the Kanban board and can be dragged and dropped between columns to change their status. Users can also filter and search tasks by name, due date, assignee, or priority level to ensure that every project milestone is properly tracked and managed.

- **Progress and Performance Tracking**

The system provides real-time progress tracking to help Project Managers and Team Members easily monitor ongoing activities. Each task displays a progress bar (%) that updates automatically based on its completion status or checklist items. Managers can view Gantt charts to observe the overall project timeline, milestones, and task dependencies. Additionally, the system supports time tracking for each member, recording working hours and providing individual performance analytics weekly or monthly. When a task nears or exceeds its deadline, the system will automatically send alerts to the responsible member to ensure timely completion and maintain project consistency.

- **Reporting and Notifications**

Another key feature of the system is its smart reporting and notification capability. Every action such as creating tasks, changing statuses, or adding comments is recorded in the Activity Log for transparency. Users receive real-time notifications via web and email whenever significant updates occur. The system also generates comprehensive reports summarizing project progress, completed tasks, ongoing tasks, and individual member performance. These reports can be exported as PDF or Excel files for presentation or archival purposes. With detailed reporting and instant notifications, users can maintain full control over project workflows and make timely management decisions.

- **Security and Subscription Plans**

To ensure data security, all system information is encrypted through HTTPS protocols and verified using JWT tokens. Users can activate two-factor authentication (2FA) to strengthen login protection. All project data, attachments, and activity logs are regularly backed up to prevent loss. The platform also provides flexible subscription plans: the Free plan targets individuals or small teams (limited to 3 projects), while the Premium plan supports organizations with unlimited projects, members, and advanced features such as Gantt charts, advanced reports, and performance analytics. Subscription upgrades are processed online via secure payment gateways like Momo, PayPal, or credit cards.

3 Proposed Solution

3.1 Software

3.1.1. Features

Actor	Demand	Request
Administrator	As an administrator, I want to view the list, create, lock and delete account users.	System user management
	As an admin, I want to define service packages (Free, Premium) and track the system's payment history.	Manage service packages and payments
	As an admin, I want to monitor the activity log of the entire system.	System monitoring
Project Manager	As PM, I want to be able to: - Create new projects (name, description, time) - Add/remove members and assign roles (Member, Guest) in the project	Create and manage projects
	As a PM, I want to keep an overview of progress through charts, performance statistics of team members	Track progress and productivity
	As PM, I want to create, edit, delete tasks, assign tasks to members, set deadlines and priorities.	Planning and assigning work
Member	As a Member, I want to be able to view and update my personal profile (avatar, password, etc.)	Personal information management
	As a Member, I want to be able to: - Display tasks assigned to members - Update task status (not done, in progress, done) - Create sub-checklists, attach files and comments to tasks	Carry out assigned work

	As a Member, I want to see working hours and submitted tasks.	Personal productivity tracking
Guest	As a Guest, I would like to see the project, and task details (read only)	Project tracking
System	The system automatically sends notifications when there are new updates (tasks, @tags, etc.), warnings about tasks that are close to or past the deadline.	Send notifications and alerts
	The system automatically records all changes in the Activity Log, automatically performs periodic backups, and validates all requests before execution.	Ensuring data security and integrity
	The system is securely integrated with online payment methods to process transactions.	Payment processing

3.1.2. Software Architecture

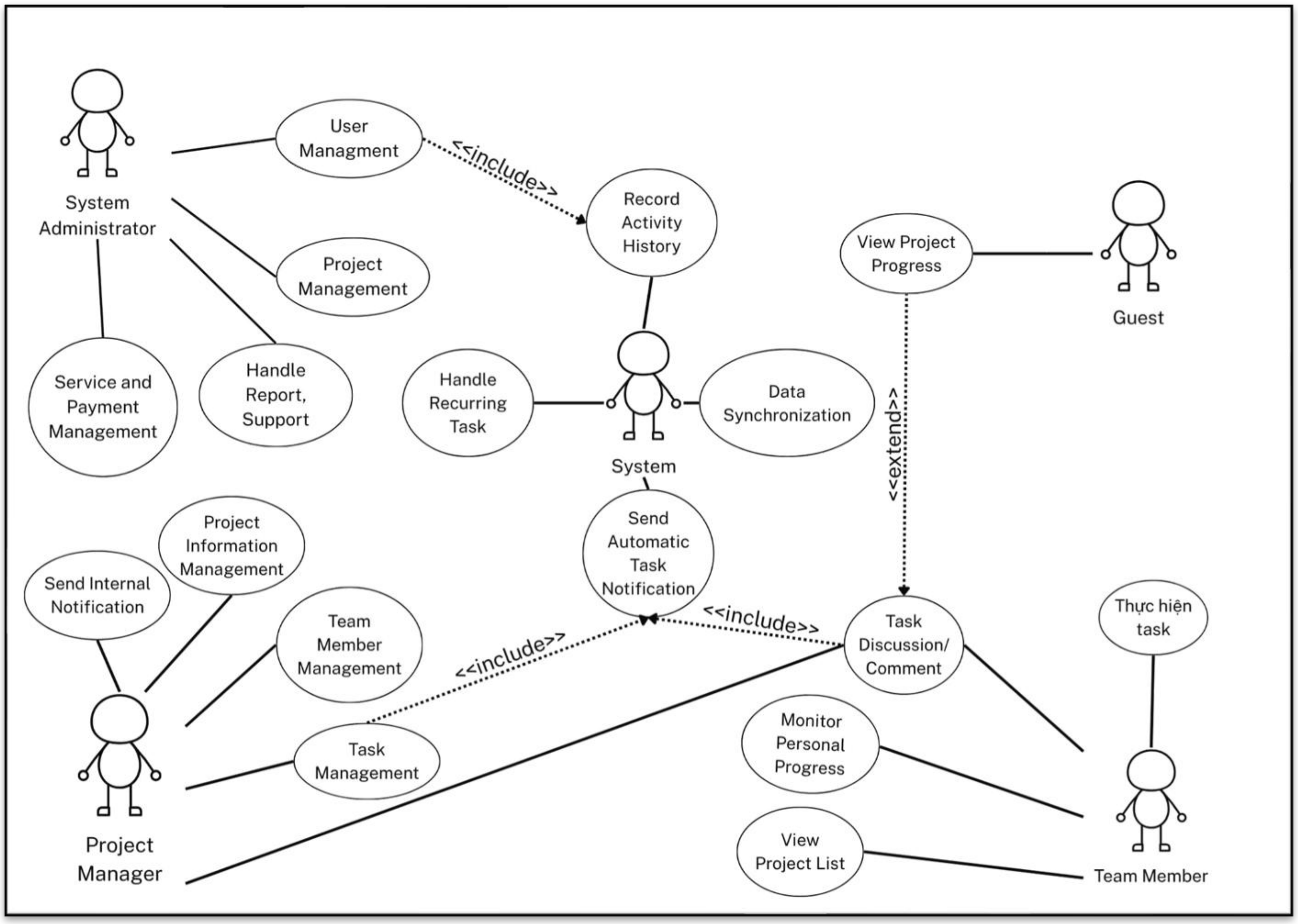
- **Architecture Pattern:** 3-Tier Architecture
  - **Presentation Layer** (Client): This layer is responsible for everything the end user sees and interacts with.
    - **Model:** Single Page Application (SPA).
    - **Details:** Delivering a flexible and instant user experience (UX) is at the core of a project management system. Key features include:
      - **Kanban:** Requires the ability to drag-and-drop Task cards.
      - **Gantt:** Requires dynamic display and interaction with timelines.
      - **Real-time nofications:** Update notifications and comments instantly. All of this requires the interface not to reload the entire page, which is what SPAs handle best.

  - **Business Logic Layer** (Server): This is the "brain" of the system, responsible for handling all logic, business rules and security, acting as a "gatekeeper" before accessing data.
    - **Model:** REST API (API Server).
    - **Details:** This layer will be built as a set of REST API endpoints. It will handle:
      - **Authentication & Authorization:** Manage login, registration, token authentication (JWT), and check user permissions (Admin, Project Manager, Member) on each resource.
      - **Logic:** Handle complex processes like creating projects, assigning tasks, calculating progress (%), and triggering notifications when changes occur.
      - **Integration:** Is the only place to handle third party services (if any) like email sending, payment gateway (for Premium plan)

    - **Security:** Ensures that the Client (browser) is never allowed to directly access the database, helping to protect data integrity.

- **Data Access Layer** (Database): This layer is solely responsible for storing and retrieving data efficiently.
  - **Model:** Document Database NoSQL.
  - **Details:** This layer includes the database server and connectivity logic from the Business layer. The reason for choosing MongoDB is because the project management data is semi-structured
    - A Project contains an array of Tasks.
    - A Task can contain an array of "Comments", an array of "Checklists", and attachments.
    - MongoDB's nested JSON/BSON document data structure allows for storing and querying these complex entities naturally and efficiently, with more flexibility than traditional relational (SQL) databases.

System Model



Technology Stack:

Layer	Technology	Reason
Front-end (Presentation)	**React + Vite**	**React** is the leading JavaScript library for building UIs, with a component-based model that is well suited for creating complex features like Kanban Boards. **Vite** provides a fast development experience (dev server) and project build speeds.

UI/UX	**Figma**	**Figma** is the industry-standard tool for interface design, prototyping, and collaboration between designers and developers.
Back-end (Business Logic)	**Nodejs, Express**	**Node.js**, combined with **Express**, is a powerful, flexible, and high-performance platform for building enterprise-grade APIs. Its module ecosystem (especially libraries like Passport.js and ORMs like Prisma or Sequelize) provides proven solutions for security requirements (JWT, 2FA) and database integration.
Database (Data Access)	**MongoDB**	**MongoDB** is a document-based NoSQL database that enables flexible storage and efficient querying of nested semi-structured data for projects and tasks.
Communication	**REST API & WebSockets**	**REST API:** Used for 90% of standard CRUD (Create, Read, Update, Delete) operations, which are "request and response" in nature (e.g., get project list, create new task). **WebSockets:** Used for Real-time features. When user A drags and drops a task, the server will push an event via WebSocket to user B and C immediately without B/C having to reload the page.

3.2 Hardware

- **Server-Side Environment:**
  - **Web Server** (Reverse Proxy): 1 Server (e.g. Apache - as per the binding/example in the Template). This server will handle HTTPS, load balancing if needed, and serve the SPA's static files.
  - **Application Server:** 1+ Server (Linux/Windows) to run Nodejs application. Requirements:
    - **RAM:** Needs enough RAM to handle multiple concurrent user connections and WebSocket connections.

        - **CPU:** Needs a multi-core CPU to handle business logic, especially heavy tasks like generating PDF/Excel Reports and Productivity Statistics.
    - **Database Server:** 1 dedicated server.
        - Requirements: Mongo Database is a large system, requiring a high-performance server with large RAM (for caching) and SSD hard drive (to increase I/O speed, fast task/comment data query).
    - **Storage & Backup Server:**
        - A file storage solution (e.g. Network Attached Storage - NAS or Cloud Storage) is required to store attachments.
        - A separate backup system is required to perform periodic backups of the MongoDB database and attachments.
- **Client-Side Environment**
    - **Device:** Computer (PC/Laptop) or mobile device (Smartphone/Tablet).
    - **Software:** Modern web browser (e.g. Google Chrome, Firefox, Safari, Edge) with full HTML5, CSS3, and JavaScript (ES6+) support to run SPA applications (React.js) and WebSockets support for real-time notifications.

4 Development Plan

4.1 Requirements Analysis

4.1.1 Objectives

The primary objective of this phase is to define, analyze, and finalize the complete scope of Jira. This process ensures an understanding of all project requirements among the student team and stakeholders, forming a stable baseline for all subsequent development and testing activities.

- **Specific Objectives**:
    - To formally catalogue all functional and non-functional requirements derived from the preliminary project documentation and stakeholder interviews.
    - To identify and specify all user roles (Administrator, Project Manager, Team Member, Guest) and the precise capabilities, permissions, and constraints (use cases) allocated to each.
    - To define the system's core data entities, attributes, and their relational schema (e.g., Users, Projects, Tasks, Comments, Attachments).
    - To establish and document all technical constraints and critical non-functional requirements, including security protocols (JWT, 2FA, HTTPS), performance benchmarks (e.g., real-time notification latency), data integrity policies, and service tier specifications (Free vs. Premium).
    - To produce a definitive requirements baseline document (the SRS) to formally govern the design, development, and validation phases.

4.1.2 Main Activities

- **Requirements Elicitation & Refinement:**
    - Conduct a precise review of the preliminary project documentation to extract all explicit and implicit requirements.
    - Conduct structured elicitation workshops (e.g., interviews, brainstorming sessions) with stakeholders to confirm workflows, identify ambiguities, resolve dependencies, and capture unstated needs.
- **Requirements Analysis & Modeling:**
    - Analyze and model the distinct permissions and data access needs for each defined user role and the system actor.

- Develop comprehensive Use Case diagrams, expanding upon the initial overview to model all primary, secondary, and error-handling interaction flows.
- Formulate preliminary data models, such as Entity-Relationship Diagrams (ERDs), to map the logical structure and relationships of all system entities.
- Model core business logic and workflows (e.g., using Activity Diagrams), including the task lifecycle (To Do -> In Progress -> Done) within the Kanban subsystem.

- **Requirements Specification:**
  - Compile all functional and non-functional requirements into a formal Software Requirements Specification (SRS).
  - Author clear, concise, and unambiguous user stories, complete with detailed acceptance criteria, for all key features.
- **Requirements Validation & Verification:**
  - Perform a formal peer review and stakeholder walk-through of the SRS document to ensure all requirements are correct, complete, consistent, unambiguous, and feasible.
  - Verify that every requirement is testable and possesses clear, objective acceptance criteria.

4.1.3 Phase Deliverables

The following deliverables will be formally produced upon completion of this phase:

- **Software Requirements Specification (SRS) Document:** A comprehensive document that will adhere to the following formal structure:
  - **1. Introduction:** (Defining Scope and Purpose)
  - **2. Positioning:** (Detailing the Problem Statement and Product Position)
  - **3. Stakeholder and User Descriptions:** (Stakeholder Summary, User Summary, User Environment, Alternatives & Competition)
  - **4. Product Features:** (A detailed breakdown of all features)
  - **5. Non-Functional Requirements:** (A formal list of all non-functional constraints)
  - **6. Phase Deliverables:** (A summary of the document itself)
- Building on the defined structure, the SRS will present a detailed catalogue of:
  - **Functional Requirements:** A complete catalogue of system capabilities, broken down by module:
    - *User Management:* Registration, login/logout, profile management, password reset, role-based access control (RBAC), and 2FA.
    - *Project & Task Management:* Project creation/archiving, member assignment, task creation with attributes (deadline, priority, assignee), checklists, and attachments.

    - *Collaboration & Tracking:* Kanban boards (drag-and-drop), Gantt charts, real-time notifications, commenting, and time tracking.
    - *Reporting & Administration:* PDF/Excel export, user support handling, and service tier management.
  - **Non-Functional Requirements:** Detailed specifications for:
    - *Security:* Data encryption (at-rest and in-transit), HTTPS, JWT token handling, 2FA, and input validation.
    - *Performance:* Page load times, API response latency, and real-time notification delay.
    - *Data Integrity:* Backup and recovery procedures.
    - *Usability:* Adherence to UI/UX guidelines, accessibility considerations.
    - *Scalability:* Definitions of service tier limitations (e.g., project/member counts).
  - **System Models:**
    - A complete set of Use Case Diagrams for all actors.
    - A high-level Data Dictionary and Entity-Relationship Diagram (ERD).

4.2 Software Design

4.2.1 Objectives

The primary objective of this phase is to transform the validated requirements defined in the Software Requirements Specification (SRS) into a comprehensive and technically sound design blueprint. This blueprint specifies how the system will be structured, organized, and implemented to fulfill all identified functional and non-functional requirements.

- **Specific objectives:**
  - To establish a robust, scalable, maintainable, and secure system architecture that satisfies all functional and non-functional requirements.
  - To formulate a normalized, efficient, and scalable database schema.
  - To produce detailed, low-level designs for all system components, modules, and their corresponding interfaces (APIs).
  - To create intuitive, accessible, and professional User Interface (UI) and User Experience (UX) designs that provide a seamless workflow for all specified features.

4.2.2 Main Activities

- **Architectural Design:**

- Finalize the core technology stack (e.g., front-end framework, back-end language, database management system), justifying choices based on performance, scalability, and team expertise.
- Define and document the high-level system architecture (e.g., 3-Tier, Microservices) and primary data flow patterns.
- Formulate the API strategy (e.g., RESTful API) and define the high-level contract, authentication mechanism, and error-handling standards for client-server communication.

- **Database Design:**
    - Translate the conceptual ERD from phase 4.1 into a detailed logical and physical database schema.
    - Define all data types, constraints (e.g., foreign keys, unique keys), indices, and relationships.
    - Apply normalization (e.g., to 3NF) to ensure data integrity and reduce redundancy.
- **Component & Module Design:**
    - Decompose the system into discrete, cohesive, and loosely coupled modules (e.g., User Authentication, Project Management, Notification Service, Payment Processing).
    - Define the specific responsibilities of each module and the precise interfaces (APIs) for inter-module communication.
    - Create sequence diagrams or activity diagrams for complex workflows (e.g., the payment integration flow).
- **UI/UX Design:**
    - Develop comprehensive **user flow diagrams** to map all critical user journeys (e.g., "new user registration" to "first task creation").
    - Create **low-fidelity wireframes** for all system screens to establish layout and information hierarchy.
    - Produce **high-fidelity mockups** that reflect final visual design elements, ensuring alignment with accessibility and usability standards.
    - Develop a **formal UI style guide**, defining typography, color palette, iconography, and reusable component patterns to maintain visual consistency throughout the system.
- **Security Design:**
    - Formulate the detailed, technical design for security implementation, including JWT token generation/validation/refresh logic, the 2FA workflow, and the Role-Based Access Control (RBAC) enforcement logic at the API and component level.

4.2.3 Phase Deliverables

- **Software Design Document (SDD):** A comprehensive document containing:
    - System Architecture Diagram (illustrating components, data flows, and technology stack).
    - Detailed Database Schema and Data Dictionary.
    - API Specification (e.g., OpenAPI/Swagger document) detailing all endpoints, request/response formats, data validation rules, and error status codes.
    - Component-level design specifications, including key sequence or interaction diagrams for complex processes.
- **UI/UX Artifacts:**
    - Complete set of low-fidelity wireframes and user flow diagrams.
    - High-fidelity mockups for all screens.
    - A clickable prototype (if feasible).
    - Formal UI Style Guide and Component Library.

4.3 Implementation

4.3.1 Objectives

The objective of this phase is to execute the construction of the software system. This involves translating the technical blueprints from the Software Design phase and the user stories from the backlog into a functional, compliant, and high-quality software product.

- **Specific objectives:**
- To establish a standardized, automated, and collaborative development environment.
- To construct the server-side infrastructure, database, and all defined RESTful API endpoints.
- To engineer the client-side user interface, implementing all state management and interactivity.
- To systematically integrate all discrete modules into a cohesive, functional application.
- To uphold code quality and maintainability through adherence to coding standards, comprehensive unit testing, and formal review processes.

4.3.2 Main Activities

- **Environment Setup:**
    - Configure the version control system (e.g., Git repository with a defined branching strategy).
    - Establish distinct, isolated environments (e.g., Development, Staging, Production).

    - Implement build automation, dependency management tools, and a Continuous Integration (CI) pipeline.
- **Back-End Development:**
    - Implement the physical database schema, data models, and any necessary seed data.
    - Construct the server-side business logic for all system features with Express - NodeJS.
    - Implement all RESTful API endpoints as defined in the SDD, ensuring robust data validation, error handling, and security.
    - Integrate security protocols (JWT, 2FA, RBAC) and third-party services (Momo, PayPal) securely.
- **Front-End Development:**
    - Translate the high-fidelity UI/UX mockups into responsive, component-based code (HTML5, CSS3, JavaScript, React, Vite, TailwindCSS).
    - Implement all client-side logic for interactivity (e.g., drag-and-drop for Kanban, Gantt chart rendering).
    - Implement asynchronous data fetching and state management (e.g., using a global state library or context) to interact with the back-end APIs.
- **Integration:**
    - Employ Continuous Integration (CI) practices to automatically build, test, and merge code changes, facilitating early detection of integration issues.
- **Code Quality & Review:**
    - Author comprehensive unit test cases for critical functions, modules, and business logic to ensure code correctness and facilitate safe refactoring.
    - Execute formal peer code reviews for all new code, enforcing coding standards, identifying defects, and improving maintainability.

4.3.3 Phase Deliverables

- **Source Code Repository:** A complete, version-controlled codebase with a full commit history and version tags for releases.
- **Runnable Application:** A functional software build deployed to a development or staging environment for testing.
- **Unit Test:** A collection of automated unit tests and their execution reports, demonstrating code coverage.
- **Technical Documentation:**
    - Developer-focused documentation, including in-code comments.
    - API usage guides (beyond the SDD).
    - Environment setup and build/run instructions.

- **Automated Build & Deployment Scripts:** (CI/CD pipeline configuration).

4.4 Testing

4.4.1 Objectives

The primary objective of this phase is to verify and validate the software system. Verification ensures the software is built *correctly* (as per the design), and validation ensures the *correct* software is built (as per the requirements). This phase is dedicated to identifying, documenting, and tracking defects to ensure the final product meets all quality standards.

- **Specific objectives:**
  - To ensure functional fidelity and correctness across all specified features in the SRS.
  - To verify all non-functional attributes, including security posture, usability, performance, and compatibility.
  - To manage the defect lifecycle from identification and triage to resolution and verification.
  - To provide a final, objective quality assessment and confirm the application's readiness for deployment.

4.4.2 Main Activities

- **Test Planning:**
  - Develop a comprehensive Test Plan detailing the scope, objectives, methodology, resources, and schedule for all testing activities.
  - Author detailed, step-by-step test cases, mapping each to a specific requirement or user story to ensure full traceability.
- **Testing Execution:**
  - **Integration Testing:** Validate the interfaces and data flow between integrated modules (e.g., verify that task creation correctly triggers the notification service and updates the database).
  - **Functional Testing:** Execute all system-level test cases to verify end-to-end functionality (e.g., user login, project creation, time tracking, report generation).
  - **Non-Functional Testing:**
    - **Security Testing:** Execute tests for security vulnerabilities, focusing on RBAC (e.g., confirm a 'Member' cannot access 'Project Manager' APIs), input validation (XSS, SQLi), and session management.
    - **Usability Testing:** Conduct sessions with representative users to evaluate the application's ease of use, intuitiveness, and adherence to UX principles.

    - **Compatibility Testing:** Verify application functionality and rendering across major web browsers (e.g., Chrome, Firefox, Safari) and device types (desktop, mobile).
    - **Performance Testing:** Assess system responsiveness, stability, and resource consumption under simulated load to validate performance benchmarks.
- **User Acceptance Testing (UAT):**
  - Provide a stable, pre-production (staging) build for stakeholders to perform final validation and formally "accept" the system as complete and fit for purpose.

4.4.3 Phase Deliverables

- **Test Plan Document:** The formal document outlining the comprehensive testing strategy.
- **Test Case Library:** The complete set of executable test cases with expected results, mapped to requirements.
- **Defect Log:** A detailed, prioritized log of all identified defects and their current status (e.g., open, in-progress, resolved, closed).
- **Test Summary Report:** A final report summarizing all testing activities, outcomes, key metrics (e.g., test coverage, defect density, pass/fail rates), and an assessment of overall software quality and readiness.

4.5 Deployment and Maintainance

4.5.1 Objectives

The objectives of this phase are to successfully transition the validated application from a testing environment into a live production environment for end-users, and to establish a framework for post-deployment support, monitoring, and lifecycle management.

- **Specific objectives:**
- To execute a controlled, predictable, and successful deployment to the production environment.
- To institute comprehensive, real-time monitoring to ensure system stability, performance, and operational health.
- To establish a formal process for user support, feedback collection, and post-release (corrective) defect resolution.
- To plan for the system's ongoing evolution, including future updates and enhancements.

4.5.2 Main Activities

- **Deployment Planning:**

  - Finalize the production hosting environment (e.g., cloud provider, server specifications) and perform cost analysis.
  - Prepare and secure the production environment (e.g., configure servers, databases, firewalls, security groups, and SSL certificates).
  - Develop a detailed deployment plan, checklist, and a formal rollback strategy in case of deployment failure.
- **Production Deployment:**
  - Implement the deployment strategy (e.g., blue-green, rolling update) to release the application to end-users with minimal to zero downtime.
  - Conduct post-deployment smoke testing on the live environment to verify core functionality and system health.
- **Monitoring:**
  - Deploy and configure application performance monitoring (APM) and centralized logging solutions to proactively track application health, performance metrics, and errors in real-time.
- **Maintenance:**
  - **Corrective Maintenance:** Triage, prioritize, and remediate production defects discovered by users or monitoring systems.
  - **Adaptive Maintenance:** Implement changes required to keep the application compatible with evolving environments (e.g., browser updates, third-party API changes, security patches).
  - **Perfective Maintenance:** Plan and implement future feature enhancements and optimizations based on user feedback and business goals (as part of a post-project lifecycle).
- **Support:**
  - Implement a formal support and incident reporting channel (as per the Administrator's "handle support requests" use case).
  - Develop and provide end-user documentation (e.g., User Manual, FAQ) to assist users and reduce support load.

4.5.3 Phase Deliverables

- **Live Application:** The deployed Jira, fully accessible to end-users.
- **Operations & Maintenance Manual:** A technical document for the operations team, detailing:
  - The application deployment and rollback procedures.
  - System monitoring and diagnostic procedures.
  - Routine maintenance tasks (e.g., backup and recovery, as per requirements).

- **User Support System:** A designated channel (e.g., support email, ticketing form) for users to submit incidents and feedback.
- **End-User Documentation:** A User Manual or Help Guide accessible to all users.

5 Human Resources & Costing Plan

Personnel Structure (5 People Team)

Role	Main Expertise	Quantity	Project Focus
Project Manager (PM)	Sprint Management, Communications, Final UAT.	1	Ensures daily sprints are on track, removes blockers, and performs final testing against requirements.
Product Analyst (PA)	Defining the core What (User Stories), Prioritization, Acceptance Criteria.	1	Provides quick answers on scope, maintains the backlog, and confirms features meet basic needs.
Core Full Stack Developer	System Setup, Database/Authentication implementation, Architecture.	1	Sets up the technical foundation, handles core user/project/task data models, and leads technical decisions.
Full Stack Developer (FE Focus)	Frontend Implementation, UI/UX (using standard library), Kanban UI.	1	Quickly builds the user interface and ensures basic data flow/interaction is smooth.
Full Stack Developer (BE Focus)	API Development, Business Logic (CRUD operations), Deployment.	1	Creates the necessary backend APIs for task/project creation, updates, and data retrieval.
Total		5 Members	

Personnel Costs

Role	Cost/Week	Estimated Total Cost (6 weeks)
Project Manager (PM)	600,000 VND	3,600,000 VND
Product Analyst (PA)	600,000 VND	3,600,000 VND
Core Full Stack Developer	1,000,000 VND	6,000,000 VND
Full Stack Developer (FE Focus)	800,000 VND	4,800,000 VND
Full Stack Developer (BE Focus)	800,000 VND	4,800,000 VND
Subtotal (Total Weekly Salary)	3,800,000 VND	22,800,000 VND

Operational and Contingency Costs

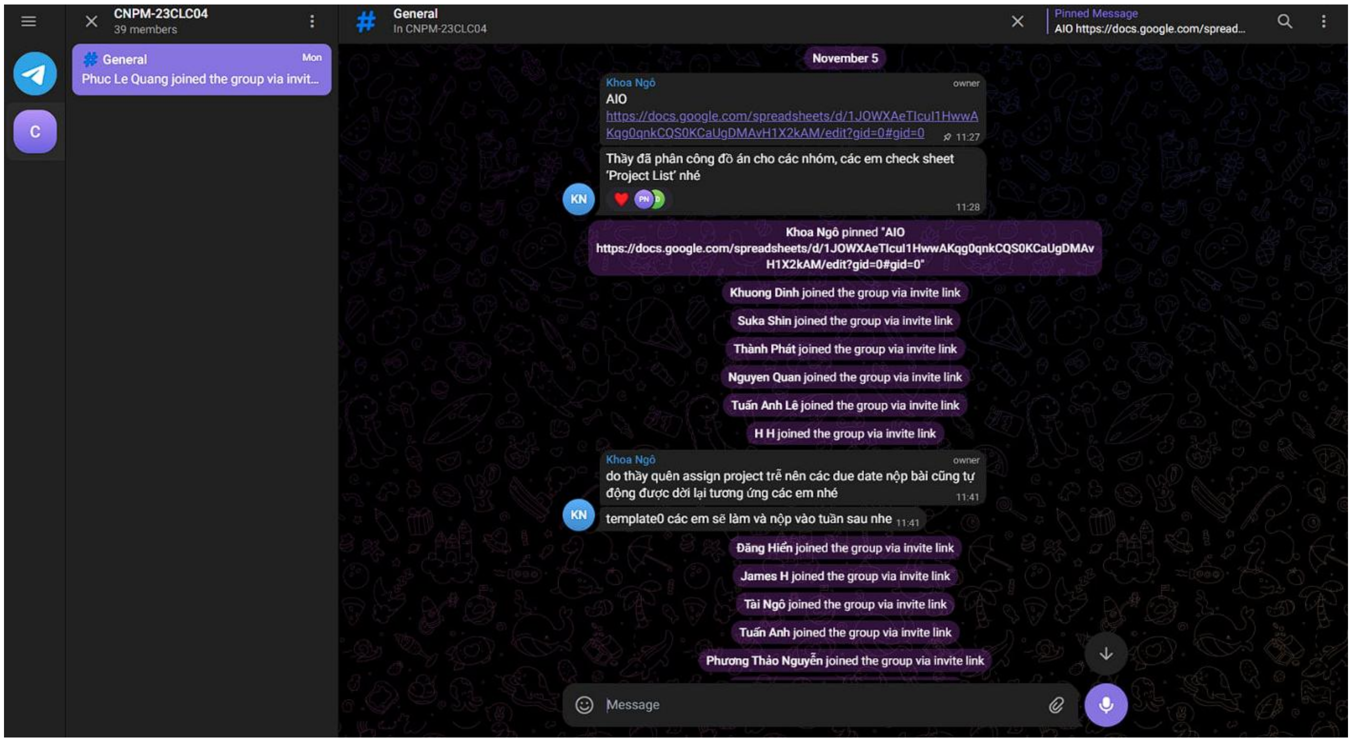
Item	Estimated Cost (6 weeks)	Description
Server/Cloud Hosting (MVP)	1,000,000 VND	Initial server setup and testing environment deployment.
Tools & Software	1,000,000 VND	Cost for necessary licenses (e.g., project management tools, repository fees).
Contingency Cost	2,800,000 VND (10% of Personnel Cost)	Reserved for unforeseen technical risks or minor scope adjustments.
Total Operational Costs (TOC)	4,800,000 VND	

Total Project Cost = Total Personnel Cost + Total Operational Costs = 22,800,000 + 4,800,000 = 27,600,000 VND.

6 Tools setup

Telegram

- This tool is used for communicating, reporting issues and problems during the software project development process with teaching assistants.
- Our group members have created Slack accounts and added two TA to the private channel – CNPM-23CLC04



Khoa Ngô owner
AIO
https://docs.google.com/spreadsheets/d/1JOWXAeTIcul1HwwAKqg0qnkCQS0KCaUgDMAvH1X2kAM/edit?gid=0#gid=0 11:27

Thầy đã phân công đồ án cho các nhóm, các em check sheet 'Project List' nhé 11:28

Khoa Ngô pinned "AIO https://docs.google.com/spreadsheets/d/1JOWXAeTIcul1HwwAKqg0qnkCQS0KCaUgDMAvH1X2kAM/edit?gid=0#gid=0"

Khuong Dinh joined the group via invite link

Suka Shin joined the group via invite link

Thành Phát joined the group via invite link

Nguyen Quan joined the group via invite link

Tuấn Anh Lê joined the group via invite link

H H joined the group via invite link

Khoa Ngô owner
do thầy quên assign project trễ nên các due date nộp bài cũng tự động được dời lại tương ứng các em nhé 11:41

template0 các em sẽ làm và nộp vào tuần sau nhe 11:41

Đăng Hiển joined the group via invite link

James H joined the group via invite link

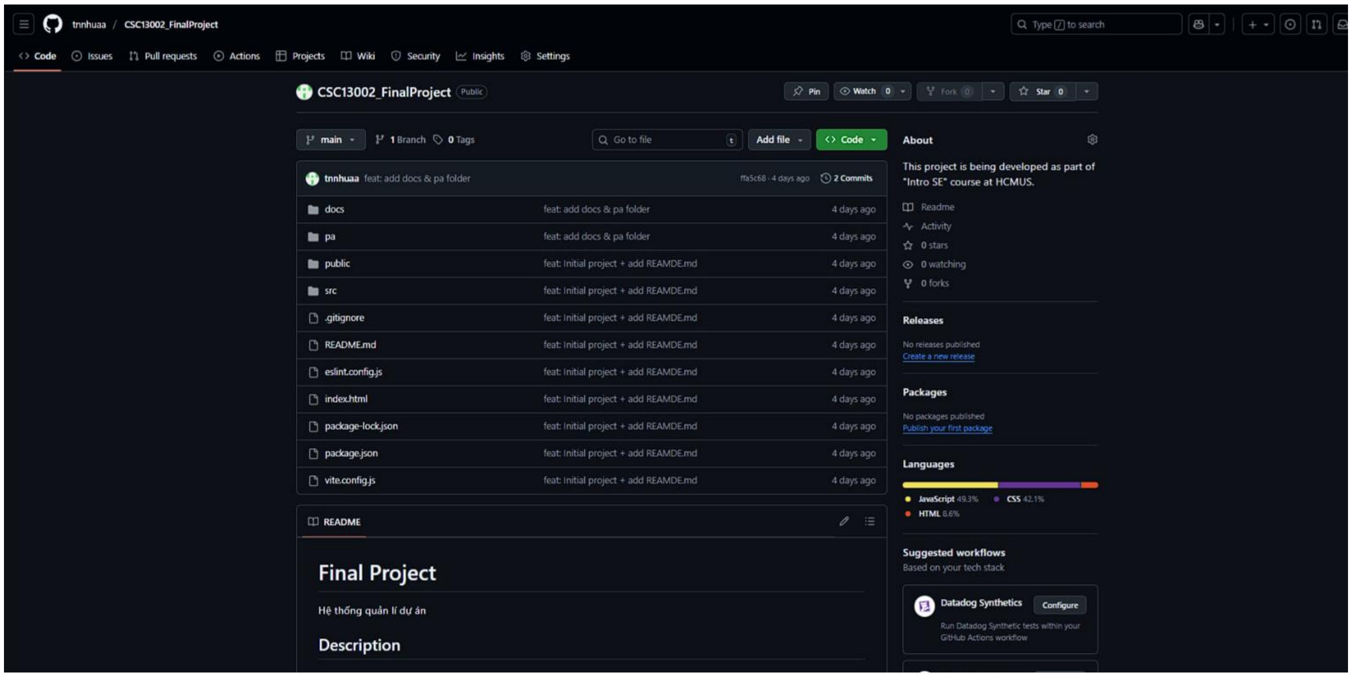
Tài Ngô joined the group via invite link

Tuấn Anh joined the group via invite link

Phương Thảo Nguyễn joined the group via invite link

Git/Github

- Git is used for managing source code versions of team projects. Team members are required to follow the coding guidelines when committing and performing Git operations (commit rule, creating branches, naming convention…)
- Github is a cloud repository for code storing, sharing and working together.



tnnhuaa / CSC13002_FinalProject

CSC13002_FinalProject Public

main · 1 Branch · 0 Tags

tnnhuaa feat: add docs & pa folder · ffa5c68 · 4 days ago · 2 Commits

docs	feat: add docs & pa folder	4 days ago
pa	feat: add docs & pa folder	4 days ago
public	feat: Initial project + add REAMDE.md	4 days ago
src	feat: Initial project + add REAMDE.md	4 days ago
.gitignore	feat: Initial project + add REAMDE.md	4 days ago
README.md	feat: Initial project + add REAMDE.md	4 days ago
eslint.config.js	feat: Initial project + add REAMDE.md	4 days ago
index.html	feat: Initial project + add REAMDE.md	4 days ago
package-lock.json	feat: Initial project + add REAMDE.md	4 days ago
package.json	feat: Initial project + add REAMDE.md	4 days ago
vite.config.js	feat: Initial project + add REAMDE.md	4 days ago

README

Final Project

Hệ thống quản lí dự án

Description

About

This project is being developed as part of "Intro SE" course at HCMUS.

Readme · Activity · 0 stars · 0 watching · 0 forks

Releases: No releases published. Create a new release

Packages: No packages published. Publish your first package

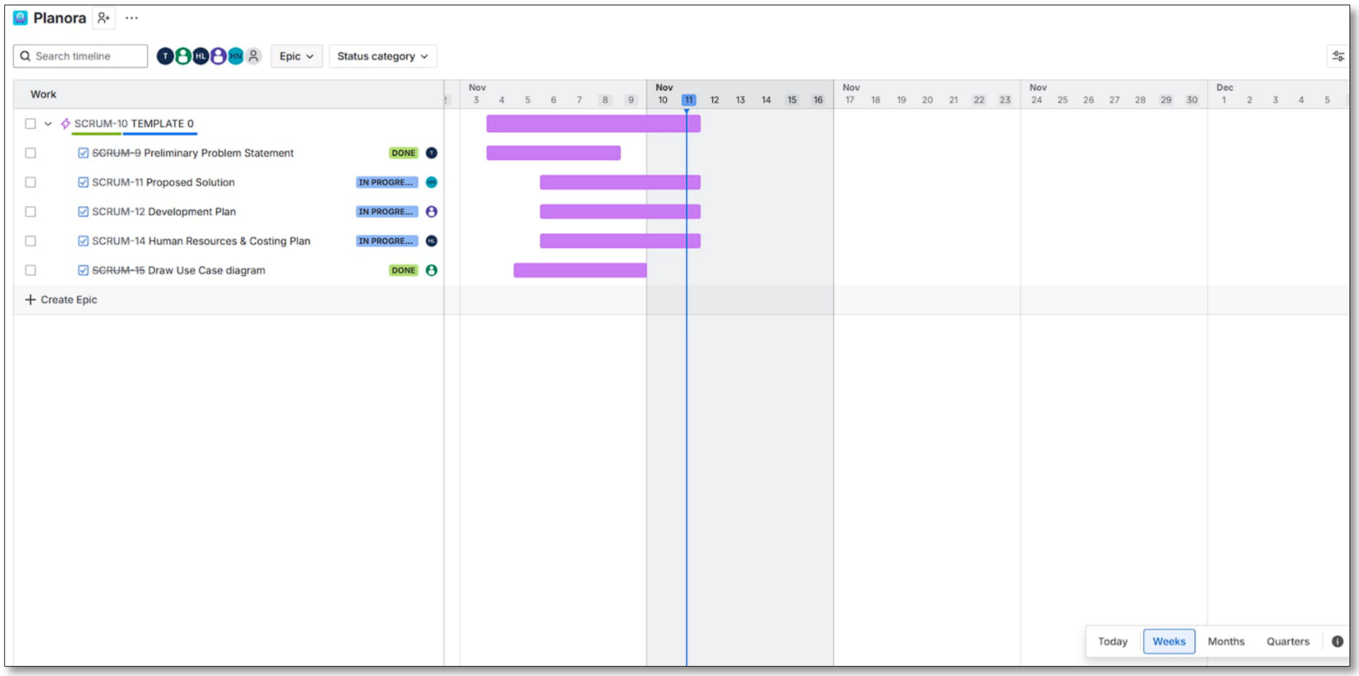
Languages: JavaScript 49.3% · CSS 42.1% · HTML 8.6%

Suggested workflows: Based on your tech stack

Datadog Synthetics — Run Datadog Synthetic tests within your GitHub Actions workflow

Jira

- This tool is used for project management, planning, tracking, releasing and supporting software



Planora

Work	Status
SCRUM-10 TEMPLATE 0	
~~SCRUM-9~~ Preliminary Problem Statement	DONE
SCRUM-11 Proposed Solution	IN PROGRE...
SCRUM-12 Development Plan	IN PROGRE...
SCRUM-14 Human Resources & Costing Plan	IN PROGRE...
~~SCRUM-15~~ Draw Use Case diagram	DONE