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
23127069	Nguyen Minh Khoi	100	[illegible]
23127359	Vo Tran Quoc Duy	100	[illegible]
23127367	Ha Thu Hoang	100	[illegible]
23127455	Truong Cong Thien Phu	100	[illegible]
23127511	Vo Minh Tuan	100	[illegible]

2 Preliminary Problem Statement

TUTORING CENTRE MANAGEMENT SYSTEM

Business Problem

The tutoring centre currently faces several challenges in managing its day-to-day operations. These challenges, typical of a growing educational establishment, likely include:

- **Manual Contract Management**: The process of creating, tracking, and updating student contracts, including fee payments, course durations, and prone to errors if overseen manually or through disparate systems.
- **Fragmented Academic Tracking**: Monitoring student attendance, daily performance (homework, class participation, etc.), and test results across various classes and teachers can be cumbersome. This makes it difficult to gain a holistic view of student progress and to generate comprehensive report cards efficiently.
- **Inefficient Staff Coordination and Monitoring**: Tracking the working hours and specific tasks accomplished by administrative and teaching staff lacks a formalised system. Managing teacher leave requests and ensuring substitute arrangements are also key operational concerns.
- **Lack of Centralised Class Information**: Accessing up-to-date information on all classes, both ongoing and completed, including student enrolment, schedules, and historical data, may not be straightforward, particularly for overall management.
- **Communication Bottlenecks**: Ensuring timely communication of contract details to teachers, and academic updates to administrative staff for report card generation, requires a smooth and reliable information flow.
- **Resource Management**: Tracking classroom supplies and the need for replenishment could be an ad-hoc process.

The absence of an integrated digital solution necessitates significant manual effort, increases the likelihood of administrative oversights, and limits the centre’s ability to leverage data for informed decision-making and proactive student support.

Operating Environment

- **Client-Side**: A modern web browser with support for HTML5 and contemporary web standards (e.g., Google Chrome, Firefox, etc.).
- **Server-Side**: Apache HTTP Server.
- **Database**: MySQL.

Design And Implementation Constraints

- **Programming Language**: Front-End: HTML, CSS, JavaScript (Framework: ReactJS). Back-End: Python
- **User Interface (UI)**: The UI must be "simple", intuitive, and user-friendly, catering to users with varying levels of technical proficiency.
- **Security**: Robust role-based access control is critical to ensure data confidentiality and integrity. Only authorised users should access specific modules and data.
- **Scalability**: The system should be designed to accommodate growth in the number of students, staff, and classes over time.
- **Data Integrity**: Mechanisms to ensure the accuracy, consistency, and reliability of all data, particularly student academic records and contract information.
- **Documentation**: Standard software development lifecycle documentation will be required, including user manuals.

3 Proposed Solution

3.1 Software

3.1.1. Features

Needs	Requirements
As a teacher or member of staff, I want to check my attendance online.	Attendance checks for teachers and staffs.
As a centre manager, I want my app to be beginner friendly.	Tutorial.
As a centre manager, I want to know how many students were enrolled in my centre.	List of students' enrolment.
As a teacher, I want to input my student's grades and give comments on quizzes/tests.	Student's grades and comments input.
As a member of staff, I want to use the teachers' input marks and comments to create report cards to distribute to parents.	Report card generator.
As a member of staff, I want to create contracts online, including course roadmaps, tuition fees, and so on.	Contract generator.
As a member of staff, I want to view class statistics, including number of students, class progress, course duration, and so on.	Class information.
As a teacher, I want to assign a student to a make-up class at the weekend whenever that student is absent from class.	Automatically add a student to a weekend class if marked absent on the application
As a teacher, I want to check my students' attendance online.	Students' attendance check.
As a teacher, I want the ability to report class issues (students' behavioural issues or technical issues) to members of staff.	Records of behavioural or technical issues for each class/classroom.
As a teacher, I want to be able to submit leave requests online.	Online leave request.

As a teacher, if I am taking a day off, I want to view other teachers' schedules to find out who is available to cover my classes that day.	Viewing colleagues' schedules.
As a manager, I want to issue accounts for my staff.	Account creation (only for headteacher).
As a member of staff, I want to create classes and add teachers and students to those classes.	Create classes and add students/teachers to classes.

3.1.2. Software Architecture

The Tutoring Centre Management System will adopt a **three-tier client-server architecture** to deliver the functionalities listed in section 3.1.1. This architecture provides a clear separation of concerns, enhancing maintainability and scalability.

1. **Presentation Tier (Client-Side):** This will be a **web-based interface** accessible via standard web browsers. It will be responsible for rendering the user interface (UI) tailored to different roles (Teacher, Staff, Centre Manager) and capturing user input. Technologies like HTML, CSS, and JavaScript (potentially with a front-end framework like React, Angular, or Vue.js) will be employed here to manage features such as online attendance checks, grade input, viewing class statistics, and submitting leave requests.
2. **Application Tier (Server-Side/Back-End):** This middle tier will host the core **business logic** of the application. It will process requests from the client-side, implement functionalities such as contract generation, report card creation, student enrolment management, user account creation, automated make-up class assignments, and manage access control. A server-side language and framework (e.g., Python with Django/Flask, Java with Spring, Node.js with Express) will be used. This tier will expose **RESTful APIs** for the front-end to consume.
3. **Data Tier (Database):** A robust **relational database** (e.g., PostgreSQL, MySQL) will be used to persistently store all application data. This includes student information, contracts, grades, attendance records, staff details, class schedules, user accounts, and reported issues. The back-end will interact with this tier to retrieve and store data

The application logic within the back-end and potentially the front-end may further be organised using a pattern such as **Model-View-Controller (MVC)** or a similar variant (e.g., MVVM, MVP) to promote modularity and separation of data, presentation, and control logic.

This architecture will support the various user interactions and data processing needs required by the system, from individual teachers checking attendance to administrators generating reports and managing contracts.

3.2 *Hardware*

In order for our application to function smoothly and reliably, the following hardware and equipment are required. These specifications cover both server-side infrastructure and client-side devices, ensuring that teachers, staff and centre managers can access the system without performance bottlenecks.

Component	Minimum Specification	Recommend Specification
Application Server	Quad-core 2.0 GHz CPU, 8 GB RAM, 256 GB SSD, 1 Gbps Ethernet	Hexa-core 2.5 GHz+ CPU, 16 GB+ RAM, 512 GB+ NVMe RAID, Dual 1 Gbps/10 Gbps Ethernet
Database Server	Quad-core 2.4 GHz CPU, 16 GB RAM, 512 GB SSD (RAID 1), 1 Gbps Ethernet	Hepta-core 2.8 GHz+ CPU, 32 GB+ RAM, 1 TB NVMe RAID 10, Dual 1 Gbps/10 Gbps Ethernet
Teacher Workstations	Dual-core 1.6 GHz CPU, 4 GB RAM, 128 GB HDD, 13″ (1366 × 768), 100 Mbps LAN	Quad-core 2.4 GHz+ CPU, 8 GB+ RAM, 256 GB+ SSD, 15″ (1920 × 1080), 1 Gbps LAN or Wi-Fi 5/6
Manager Workstations	Dual-core 1.6 GHz CPU, 4 GB RAM, 128 GB HDD, 15″ (1366 × 768), 100 Mbps LAN	Quad-core 2.4 GHz+ CPU, 8 GB+ RAM, 256 GB+ SSD, Dual 22″+ (1920 × 1080), 1 Gbps LAN or Wi-Fi 5/6
Network Switch	Unmanaged Gigabit Ethernet (8 port)	Managed Gigabit/10 Gbps Ethernet with VLAN support
Router / Gateway	Business-grade NAT/DHCP router	Enterprise-grade router with QoS, VPN, IDS/IPS
Wireless Access Point	Dual-band 802.11n	Mesh Wi-Fi 5/6 (802.11ac/ax)
Internet Link	50 Mbps symmetrical	100 Mbps – 500 Mbps symmetrical with optional 4G/5G backup
Printer	USB Network Laser Printer, 20 ppm	Network Laser Printer or Colour Laser Printer, 30 ppm+, duplex,

		Wi-Fi/Ethernet
Scanner (optional)	Basic flatbed USB scanner (10 ppm)	ADF Document Scanner (20 ppm)
UPS	600 VA	1000 VA – 1500 VA (providing 10 – 15 minutes runtime at full load)

4 Development Plan

4.1 *Requirements Analysis*

- <u>COLLECTING CLIENT'S REQUIREMENTS:</u> Our deliverables for this task are to have documentation identifying all **key stakeholders**, a defined **initial project scope**, and comprehensive notes from **requirements elicitation sessions** detailing client needs.
- <u>PROJECT ANALYSIS:</u> Our deliverable for this task is to have thoroughly written documentation detailing the project's **feasibility** (technical, economic, operational), a **preliminary risk assessment**, and, if applicable, mapping of **current ('as-is') business processes** that the software will impact.
- <u>FAMILIARISATION WITH GIT, FRONT-END & BACK-END TOOLS:</u> Our outcome for this task is **confirmation of tool installation and configuration** across the team and records of any **basic proficiency training** conducted, ensuring team readiness.
- <u>FAMILIARISATION WITH FIGMA:</u> Our outcome for this task is team members possessing **configured accounts** and a foundational understanding of using Figma for **prototyping basics**, potentially evidenced by completed trial exercises.

- FAMILIARISATION WITH JIRA: Our outcome for this task is a **configured project space within Jira** and clear team understanding of the **defined workflows** for task management.
- EXPLORATION OF TECHNOLOGY STACKS: Our deliverable for this task is a **comparative analysis report** evaluating different technology stacks, potentially including results from any **Proof-of-Concept (PoC) exercises** for critical components.
- USER STORY MAPPING: Our deliverable for this task is to have documented a **user story map**, clearly defined **user personas**, and a **prioritised list of user stories**.
- PRESENTATION OF THE OVERALL ARCHITECTURE: Our deliverables for this task are a **draft architecture document** and minutes or collated feedback from the **stakeholder Q&A session** regarding the proposed system architecture.
- DESCRIPTION OF HARDWARE AND EQUIPMENT REQUIREMENTS: Our deliverable for this task is documentation specifying **hardware and equipment needs** based on capacity planning, along with any **vendor research** conducted for potential procurement.
- FINALISING THE PROJECT PROPOSAL: Our deliverable for this task is a **finalised project proposal document**, which incorporates refined cost estimations, validated project timelines, and formally agreed terms & conditions.
- REQUIREMENTS VALIDATION & SIGN-OFF: Our deliverable for this task is a **formally signed-off requirements specification document**, confirming client agreement on the project's scope and objectives.

4.2 *Software Design*

- UI/UX DESIGN: Our deliverables for this task are a comprehensive UI/UX design package including the documented **information architecture**, iterative sets of **wireframes and interactive prototypes**, and a summary report from any **usability heuristics reviews** conducted.
- DATABASE DESIGN: Our deliverable for this task is detailed **database design documentation**, including conceptual, logical, and physical data models, **normalisation details**, and the proposed **indexing strategy**.

- SOFTWARE ARCHITECTURE DESIGN: Our deliverable for this task is a **detailed software architecture document**, featuring component diagrams, clear **API interface specifications**, and outlined **security design considerations**.
- ACCESSIBILITY DESIGN SPECIFICATIONS: Our deliverable for this task is a document outlining specific **accessibility design standards** to be and **design patterns** for key accessible components.
- SECURITY DESIGN REVIEW & THREAT MODELLING: Our deliverables for this task are a **security design review report** and a **threat model document** identifying potential system threats and outlining planned mitigation strategies at the design level.
- DATA FLOW DESIGN: Our deliverables for this task are **Data Flow Diagrams (DFDs)** and supporting documentation that clearly illustrate how data is inputted, processed, stored, and outputted across various system components.
- SELECTION OF TECHNOLOGY STACK: Our deliverable for this task is a finalised document specifying the **chosen technologies and their exact versions**, along with confirmation of necessary **licensing compliance reviews**.
- USABILITY TESTING OF PROTOTYPES: Our deliverable for this task is a **usability test report** based on user interactions with design prototypes, detailing observed issues, user feedback, and recommendations for design enhancements.
- REVIEW OF UI DESIGN WITH CLIENT: Our deliverables for this task are documented **client feedback on UI designs**, records of any design iterations undertaken, and notes on how **accessibility considerations** have been addressed.
- ALLOCATION OF TASKS: Our deliverable for this task is a **task allocation plan** or sprint backlog, clearly detailing assigned tasks, confirmed resource availability, and **mapped dependencies** between various tasks.
- DETAILED DESIGN DOCUMENTATION: Our deliverables for this task are comprehensive **technical specification documents** for system modules and initial **drafts of API documentation** for developer use.

4.3 Implementation

- ENVIRONMENT SETUP (DEVELOPMENT, STAGING): Our outcome for this task is fully operational **development and staging environments**, configured precisely according to project requirements.
- CODING STANDARDS ADHERENCE: Our deliverable for this task is an **agreed set of coding standards**, formally documented and clearly communicated to all members of the development team.
- DATABASE SCHEMA IMPLEMENTATION: Our deliverable for this task is an **implemented database schema** on the development and staging servers, precisely matching the approved database design, along with version-controlled creation scripts.
- BACK-END DEVELOPMENT: Our deliverables for this task are implemented and thoroughly unit-tested **back-end API endpoints**, fully coded **business logic** components, and functional **database integration** layers.
- FRONT-END DEVELOPMENT: Our deliverables for this task are developed and unit-tested **UI components**, a functional and robust **state management system**, and integrated **API consumption** logic within the user interface.
- THIRD-PARTY SERVICE INTEGRATION: Our deliverable for this task is functional and tested **integration points with specified third-party services** (e.g., payment gateways, mapping services), including robust API clients and comprehensive error handling.
- TEST DATA GENERATION: Our deliverable for this task is a set of scripts or mechanisms for **generating anonymised or synthetic test data** that realistically reflects diverse usage scenarios for thorough testing.
- INITIAL SECURITY IMPLEMENTATION: Our deliverable for this task is the implementation of foundational **authentication and authorisation mechanisms**, input validation routines, and other core security measures as defined in the design specifications
- UNIT TESTING: Our deliverable for this task is a comprehensive suite of **unit tests covering individual code modules**, with automated execution integrated into the development workflow to ensure ongoing code quality.

- PEER CODE REVIEWS: Our outcome for this task is established records or evidence (e.g., comments within the version control system, completed review checklists) demonstrating that **code reviews have been consistently conducted**.
- CONTINUOUS INTEGRATION (CI) SETUP: Our deliverable for this task is a functional **Continuous Integration (CI) pipeline** that automatically builds and tests code upon each significant change, facilitating early issue detection.
- BUILD AND DEPLOYMENT SCRIPTING (FOR DEV/STAGING): Our deliverable for this task is a set of **automated build and deployment scripts** for efficiently and consistently deploying the application to development and staging environments.
- INTEGRATION OF MODULES: Our outcome for this task is successfully **integrated front-end and back-end components**, with verified and robust data flow established and tested between them.

4.4 Testing

- TEST CASE DEVELOPMENT: Our deliverable for this task is a comprehensive suite of **documented test cases** that cover all specified functional, non-functional, and accessibility requirements.
- TEST ENVIRONMENT SETUP: Our outcome for this task is a **stable and correctly configured test environment** that accurately mirrors the intended production setup as closely as possible.
- SYSTEM TESTING (FUNCTIONAL, INTEGRATION): Our deliverables for this task are detailed execution logs and summary reports from the **functional and integration testing cycles**, documenting the outcomes and coverage.
- PERFORMANCE TESTING: Our deliverable for this task is a **performance test report** detailing the system's behaviour under various load conditions, including response times, stability, resource utilisation, and identifying any bottlenecks.
- SECURITY TESTING: Our deliverable for this task is a **security test report** that outlines any vulnerabilities found during penetration testing and vulnerability scanning, along with prioritised recommendations for their mitigation.
- ACCESSIBILITY TESTING: Our deliverable for this task is an **accessibility compliance report**, detailing results from testing against specified standards,

using a combination of automated tools and manual checks with assistive technologies.

- USABILITY TESTING (OF IMPLEMENTED SOFTWARE): Our deliverable for this task is a **usability test report** based on direct user interactions with the implemented software, identifying specific usability issues, task completion rates, and overall user satisfaction levels.
- DEFECT TRACKING & MANAGEMENT: Our deliverable for this task is an up-to-date **defect log** (e.g., within Jira) which clearly shows all identified bugs, their current status, priority, assigned developer, and eventual resolution.
- USER ACCEPTANCE TESTING (UAT) FACILITATION: Our deliverables for this task are documented **UAT scenarios** provided to users, records of all user feedback and issues raised, and a formal **UAT sign-off form** or report from the client.
- FAILOVER AND RECOVERY TESTING (IF APPLICABLE): Our deliverable for this task is a **failover and recovery test report**, documenting the system's ability to handle simulated failures and recover operations within predefined acceptable timeframes (RTO/RPO).
- REGRESSION TESTING: Our deliverables for this task are an established **regression test suite** (which may be partially or fully automated) and reports from its execution following code changes or bug fixes to ensure existing functionality remains intact.
- DOCUMENTATION REVIEW (USER & TECHNICAL): Our deliverable for this task is **reviewed user and technical documentation** with tracked changes, collated feedback, and identified corrections to ensure accuracy, completeness, and clarity.
- ALPHA/BETA TESTING (IF APPLICABLE): Our deliverables for this task are consolidated **feedback reports and defect logs from Alpha and/or Beta testing phases**, summarising user experiences and issues identified by an early-adopter audience.
- TEST REPORTING: Our deliverable for this task is one or more **overall test summary reports** indicating the scope of test coverage, pass/fail rates, defect density, and a clear list of any outstanding critical defects impacting release readiness.

4.5 Deployment and Maintenance

- USER TRAINING & MATERIAL PREPARATION: Our deliverables for this task are comprehensive **user training materials** (e.g., user manuals, tutorial videos, FAQs) and records verifying that **training sessions have been conducted** for designated end-users.
- PREPARATION FOR PRESENTATION: Our deliverables for this task are a polished **presentation slide deck** and well-defined **user demonstration scenarios** ready for the final client presentation and project handover.
- PRODUCTION ENVIRONMENT HARDENING: Our deliverable for this task is a completed **production environment hardening checklist** and documented confirmation that all specified security configurations and best practices have been applied.
- DEPLOYMENT TO PRODUCTION (OPTIONAL): OUR DELIVERABLES FOR THIS TASK ARE A DETAILED DEPLOYMENT PLAN, EXECUTION LOGS FROM THE PRODUCTION DEPLOYMENT PROCESS, AND THOROUGHLY DOCUMENTED POST-DEPLOYMENT VERIFICATION CHECKS.
- FINAL TESTING AND FINISHING TOUCHES: Our deliverables for this task are a completed **pre-deployment checklist**, confirmation of successful **data migration** (if applicable), and **finalised, approved user and technical documentation**.
- POST-LAUNCH SUPPORT & MONITORING: Our outcomes for this task are established **user support channels**, initial **system performance and availability monitoring reports** from the live environment, and a functioning mechanism for collecting **post-launch user feedback**.
- SERVICE LEVEL AGREEMENT (SLA) MONITORING SETUP: Our outcome for this task is configured **monitoring tools and dashboards** designed to track and report on adherence to defined Service Level Agreements (SLAs) for system uptime, performance, and support response times.
- MAINTENANCE: Our deliverable for this phase is an agreed **maintenance plan** that outlines procedures for ongoing support, application of updates and patches, regular security reviews, and continuous system health monitoring.

- <u>Project Closeout:</u> Our deliverables for this task are a **final project report** summarising outcomes against initial objectives and budget, a **lessons learnt document** capturing key insights for future projects, and a comprehensively **archived collection of all project artefacts**.

5 Human Resources & Costing Plan

ROLE	MEMBER	RESPONSIBILITY
FRONT-END	Vo Tran Quoc Duy Truong Cong Thien Phu	Developing and implementing the user interface (UI) and user experience (UX) based on design specifications. Writing client-side code using HTML, CSS, JavaScript, and relevant frameworks. Ensuring application responsiveness. Collaborating with UI/UX designers and integrating with back-end APIs. Optimising front-end performance and conducting unit tests for UI components. Implementing accessibility standards.
BACK-END	Nguyen Minh Khoi Vo Minh Tuan	Developing and maintaining server-side logic, databases, and APIs. Implementing business rules and ensuring system security and data protection. Managing server-side performance optimisation and scalability. Integrating with third-party services and ensuring seamless data exchange. Writing unit and integration tests for back-end components.

QA - DOCUMENT	Ha Thu Hoang	**QA:** Developing and executing comprehensive test plans, test cases, and test scripts to ensure software quality. Identifying, logging, tracking, and verifying defects. Performing various testing types (e.g., functional, integration, regression, usability) and creating detailed test reports. Collaborating with developers to resolve issues and ensuring the final product meets all requirements. **DOCUMENT:** Creating, managing, and maintaining all project-related documentation, including technical specifications, API documentation, user manuals, training materials, and test documentation. Ensuring all documents are clear, accurate, up-to-date, and readily accessible to the team and stakeholders.

6 Tools setup

In this assignment, we take advantage of the following tools to help our team collaborate more effectively and manage our project:

- Moodle is used to submit all Project Assignments at each stage.
- Discord is the main channel we use to communicate with our teacher and among teammates.
- Jira is used to manage tasks and track our project's progress.
- GitHub is where we store our source code.

Aside from GitHub, our team have finished setting up the other tools and begun using them to improve collaboration among teammates and keep the project on track.