

**International School**

**Scientific Research**

**Project Proposal**

Version 1.1

Date: June 25th, 2025

**Applying Large Language Models to optimize the software system analysis and design process**

**Submitted by**

Cuong, Doan Viet

Dat, Hoang Van Tien

Phap, Nguyen Van

Phong, Hoang Ba

Quynh, Ho Dang

**Approved by**

MSc Man, Duc Nguyen

Proposal Review Panel Representative:

Name Signature Date

Capstone Project 2- Mentor:

Name Signature Date

PROJECT INFORMATION

|  |  |  |  |
| --- | --- | --- | --- |
| Project acronym | ALMOSSAD | | |
| Project Title | Applying Large Language Models to optimize the software system analysis and design process. | | |
| Start Date | 1 July 2025 | End Date | 31 December 2025 |
| Lead Institution | International School, Duy Tan University | | |
| Project Mentor | MSc Man, Nguyen Duc | | |
| Scrum master / Project Leader & contact details | Phap, Nguyen Van  Email:mynamephap@gmail.com  Tel: 0979066067 | | |
| Partner Organization | Duy Tan University | | |
| Project Web URL |  | | |
| Team members | Name | Email | Tel |
|  | Cuong, Doan Viet | doancuong357@gmail.com | 0862496551 |
|  | Dat, Hoang Van Tien | hngvtdat010@gmail.com | 0961167003 |
|  | Phap, Nguyen Van | mynamephap@gmail.com | 0979066067 |
|  | Phong, Hoang Ba | hoangbaphongk20cva@gmail.com | 0388969066 |
|  | Quynh, Ho Dang | hodangquynh2k4@gmail.com | 0932342990 |

DOCUMENT APPROVALS

The following signatures are required for approval of this document.

|  |  |  |
| --- | --- | --- |
| Phap, Nguyen Van  Student ID: 28210250268  *Scrum Master* | Signature | Date |
| Cuong, Doan Viet  Student ID: 28217331026  *Team Member* | Signature | Date |
| Dat, Hoang Van Tien  Student ID: 28219005093  *Team Member* | Signature | Date |
| Phong, Hoang Ba  Student ID: 28217450789  *Team Member* | Signature | Date |
| Quynh, Ho Dang  Student ID: 28211103076  *Team Member* | Signature | Date |

REVISION HISTORY

| Version | Date | Comments | Author |
| --- | --- | --- | --- |
| 1.0 | June 25th, 2025 | Initial Release | C1SE.CDPPQ |
|  |  |  |  |

TABLE OF CONTENT

[1. Introduction 5](#_heading=h.bxt51ro29h29)

[1.1. Purpose Of Document 5](#_heading=h.ws3t12iyc0wd)

[1.2. Project Goal 5](#_heading=h.cucy9tqt2qx8)

[2. Problem Definition............................................................................................ 6](#_heading=h.r07qil3w16zt)

[2.1. Business need 6](#_heading=h.4ukyjjyux39z)

[2.2. Solution 8](#_heading=h.617yknxcls0a)

[3. Current Status of Art 8](#_heading=h.z3yp9nsh4rpq)

[4. Engineering Approach..................................................................................... 1](#_heading=h.5a935ss46tqh)2

[4.2. System context description 1](#_heading=h.vzpxirnvkyx5)2

[4.3. Technical Constraints 1](#_heading=h.8k0jcrt2m4i5)3

[5. Tasks and Deliverables................................................................................... 1](#_heading=h.igjtr2ofdx42)4

[6. Project Management........................................................................................ 1](#_heading=h.4ajcu6dddo4v)6

[6.1. Cost/Budget for Project 1](#_heading=h.n173e54205xu)6

[6.2. Tentative Schedule 1](#_heading=h.1l216jvonr9y)7

[6.2.1. Master Plan 1](#_heading=h.uvr2n51zpyld)7

[6.2.2. Scrum Process 1](#_heading=h.k2m7kbo3yq0n)8

[7. Project Constraints........................................................................................... 1](#_heading=h.4po1y9domz7u)9

[8. Conclusion....................................................................................................... 2](#_heading=h.t85zyq9agtr)1

[9. References....................................................................................................... 2](#_heading=h.sucuvke946pv)1

[10. Attachment.................................................................................................... 2](#_heading=h.enlpkt4e8ezb)2

**1. Introduction**

**1.1. Purpose of Document**

This document outlines the purpose and scope of our project: applying Large Language Models (LLMs) to optimize the software system analysis and design process. It details how LLMs can address current business needs and challenges in software development, particularly concerning efficiency, accuracy, and innovation during the early phases of the software development lifecycle.

Specifically, this document aims to:

* Identify key business needs and pain points in traditional software system analysis and design that LLMs can alleviate.
* Propose LLM-driven solutions for these identified challenges, demonstrating their potential to streamline and enhance existing workflows.
* Provide an overview of the proposed system architecture, illustrating how LLMs will integrate into and support the analysis and design activities.
* Summarize the necessary resources, projected schedule, proposed technical solutions, and estimated budget required for successful project execution.

**1.2. Project Goal**

The primary goal of this project is to significantly enhance the efficiency, accuracy, and comprehensiveness of the software system analysis and design process through the strategic application of Large Language Models (LLMs).

Specifically, this project aims to:

* Automate and accelerate the extraction and synthesis of requirements from various input formats (e.g., text documents, images, audio, direct input).
* Improve the consistency and completeness of generated analysis and design artifacts, such as user stories, use cases, and technical specifications, by leveraging LLMs' understanding and generation capabilities.
* Provide intelligent assistance to software analysts and designers, enabling them to identify potential risks, generate test cases, define system actors, and structure system architecture more effectively.
* Foster a more streamlined and collaborative workflow by providing integrated tools that leverage LLM-generated insights, ultimately reducing manual effort and potential for human error in the initial phases of software development.
* Deliver a proof-of-concept system demonstrating the tangible benefits of LLM integration in optimizing the crucial early stages of software project development.

**2. Problem Definition**

In today's fast-paced digital economy, the ability to build high-quality software quickly is more critical than ever for business success. Consequently, optimizing the system analysis and design process is a vital necessity.

However, many projects fail to meet their deadlines or quality expectations because their workflow is inefficient. Analysts (BAs, SAs) frequently face grueling manual processes, fragmented tools, and a lack of connectivity, all without an intelligent assistant to provide support. They become overloaded with repetitive tasks, leading to burnout, a higher propensity for errors, and a loss of creative motivation.

Furthermore, there are many experienced analysts and project managers who want to ensure projects are executed with consistency and quality, sharing the best practices they have cultivated. However, they lack a tool powerful enough to implement and standardize this workflow. They need a platform where they can apply advanced methodologies, improve the quality of project artifacts, and build a reliable, traceable bridge between business requirements and the final technical implementation.

**2.1. Business need**

The current landscape of software system analysis and design, particularly in complex projects, faces several significant challenges that impede efficiency, accuracy, and overall project success. Addressing these pain points is crucial for delivering high-quality software on time and within budget. The business needs driving the application of Large Language Models (LLMs) in this domain include:

* Inefficient Requirement Elicitation and Documentation:
  + Manual processing of diverse input formats (e.g., meeting minutes, informal discussions, existing documentation) is time-consuming and prone to human error and inconsistencies.
  + Difficulty in extracting and structuring clear, unambiguous requirements from large volumes of unstructured text.
  + Challenges in ensuring completeness and avoiding ambiguities in user stories, use cases, and functional specifications.
* Lack of Comprehensive Analysis and Design Automation:
  + The manual generation of design artifacts (e.g., test cases, architectural components, database schemas) from requirements is labor-intensive and requires specialized expertise for each artifact type.
  + Limited automated support for identifying potential risks, conflicts, or missing information early in the design phase.
  + Difficulty in maintaining traceability between initial requirements and final design specifications across a project's lifecycle.
* Suboptimal Knowledge Management and Collaboration:
  + Dispersed knowledge across various documents and team members makes it challenging to consolidate and leverage insights effectively during analysis and design.
  + Difficulty in facilitating real-time collaboration and consistent understanding among different stakeholders (business analysts, developers, testers) regarding system requirements and design choices.
* Time-to-Market Pressures:
  + The manual nature of analysis and design often leads to extended lead times, delaying the start of development and impacting time-to-market.
  + Rework due to late-stage requirement changes or design flaws is costly and could be mitigated by more robust early-stage analysis.

By leveraging LLMs, we aim to directly address these business needs by automating, enhancing, and streamlining critical aspects of the software system analysis and design process, ultimately leading to faster, more accurate, and more robust software solutions.

**2.2. Solution**

Our project proposes a solution that leverages Large Language Models (LLMs) to support and optimize the process of software system analysis and design. The system acts as an intelligent assistant capable of understanding business requirements provided by end users (such as clients or analysts), automatically suggesting design models (e.g., UML diagrams, business process flows), and identifying inconsistencies or logical gaps in system analysis.

Additionally, the system facilitates team collaboration during the design phase by providing automated suggestions and analyzing natural language from input documents. This helps reduce communication time and minimize errors when converting requirements into system models. Software engineers can interact directly with the LLM to verify the completeness, consistency, and correctness of specifications.

The solution aims to create an intelligent support environment where both novice and experienced software engineers can approach system design more easily, efficiently, and accurately through AI assistance.

**3. Current Status of Art**

The analysis and design phase of software systems is crucial for the success of a project, laying the foundation for development, testing, and deployment. Traditionally, this phase relies heavily on manual processes, human expertise, and specialized tools. However, the emergence of Artificial Intelligence (AI), particularly Large Language Models (LLMs), is rapidly transforming the landscape, opening new avenues for optimization and automation.

**3.1. Traditional Approaches to Software Analysis and Design**

Conventional software analysis and design methodologies employ a variety of tools and practices, each with its strengths and limitations:

* **Manual Documentation and Elicitation:** Business analysts and system architects spend significant time conducting interviews, workshops, and reviewing existing documentation to gather requirements. These requirements are then manually transcribed into various artifacts like Functional Specification Documents (FSDs), Software Requirement Specifications (SRSs), User Stories, and Use Cases. While offering flexibility and deep human understanding, this approach is labor-intensive, time-consuming, prone to inconsistencies, and highly dependent on the individual's experience.
* **Diagramming Tools:** Tools like Lucidchart, Microsoft Visio, and draw.io are widely used for creating Unified Modeling Language (UML) diagrams (e.g., Use Case Diagrams, Activity Diagrams, Sequence Diagrams), Data Flow Diagrams (DFDs), and architectural blueprints. These tools aid in visualizing system components and interactions but require manual input and interpretation from the user.
* **Requirements Management Systems (RMS):** Tools such as Jira, Azure DevOps, IBM Rational DOORS, and Jama Connect help manage, track, and trace requirements throughout the software development lifecycle. They offer version control and collaboration features but primarily serve as repositories, with limited inherent intelligence for generating or validating content.
* **Prototyping Tools:** Figma, Sketch, and Adobe XD enable the creation of interactive prototypes and mockups for UI/UX design. While excellent for validating user interfaces, they typically come into play after initial analysis and don't directly assist in the core requirement elicitation or backend design.

**Limitations of Traditional Approaches:** Despite their utility, traditional methods often suffer from:

* **Time Consumption:** Elicitation, analysis, and documentation are inherently slow processes.
* **Human Error and Bias:** Manual interpretation can lead to misinterpretations, omissions, or personal biases.
* **Inconsistency and Ambiguity:** Maintaining consistency across large and complex documentation sets is challenging.
* **Lack of Automation:** Limited automated support for generating design artifacts, detecting conflicts, or suggesting improvements.
* **Scalability Issues:** Difficulty in efficiently managing and processing vast amounts of unstructured information in large projects.

**3.2. Emerging Role of Artificial Intelligence in Software Engineering**

The field of AI has begun to permeate various aspects of the software development lifecycle (SDLC), moving beyond just code generation to impact earlier phases:

* **Code Generation and Autocompletion:** AI-powered tools like GitHub Copilot and Google's Codey assist developers by suggesting code snippets, completing lines, and even generating entire functions based on comments or existing code patterns.
* **Automated Testing and Debugging:** AI is being used to generate test cases, analyze logs, and identify potential bugs more efficiently than manual methods.
* **Project Management and Predictive Analytics:** AI algorithms are applied to predict project timelines, identify risks, and optimize resource allocation based on historical data.

**3.3. Large Language Models (LLMs) in Software Analysis and Design**

LLMs, such as OpenAI's GPT series, Google's Gemini, and other open-source models, represent a significant leap in AI capabilities, particularly in understanding, processing, and generating human-like text. Their potential to revolutionize software analysis and design is a burgeoning area of research and development:

* **Natural Language Understanding (NLU) for Requirements:** LLMs can parse and understand unstructured text inputs (e.g., meeting notes, user feedback, existing documents) to identify key entities, relationships, and implicit requirements.
* **Automated Artifact Generation:** Research and early tools are exploring the use of LLMs to automatically generate:
  + **User Stories and Use Cases:** From high-level descriptions or informal conversations.
  + **Test Cases:** Derived directly from functional requirements.
  + **API Specifications:** Based on system functionality descriptions.
  + **Basic Architecture Outlines:** Suggesting components and interactions.
* **Risk Identification and Conflict Detection:** LLMs can analyze requirements documents for inconsistencies, ambiguities, or potential conflicts, flagging them for human review. They can also identify implicit risks based on common project pitfalls learned from vast datasets.
* **Documentation and Summarization:** LLMs excel at summarizing lengthy technical documents and generating clear, concise project documentation, reducing manual effort.
* **Intelligent Assistant for Analysts:** LLMs can act as conversational assistants, answering queries about system requirements, suggesting design alternatives, or providing explanations.

**Current Gaps and Opportunities for this Project:** While promising, the direct application of LLMs to comprehensively optimize the **entire** software system analysis and design workflow, from diverse inputs to detailed, interlinked design artifacts, is still an evolving field. Many existing LLM applications focus on specific, isolated tasks (e.g., generating a single user story) rather than providing an integrated, end-to-end solution that connects various analysis and design components.

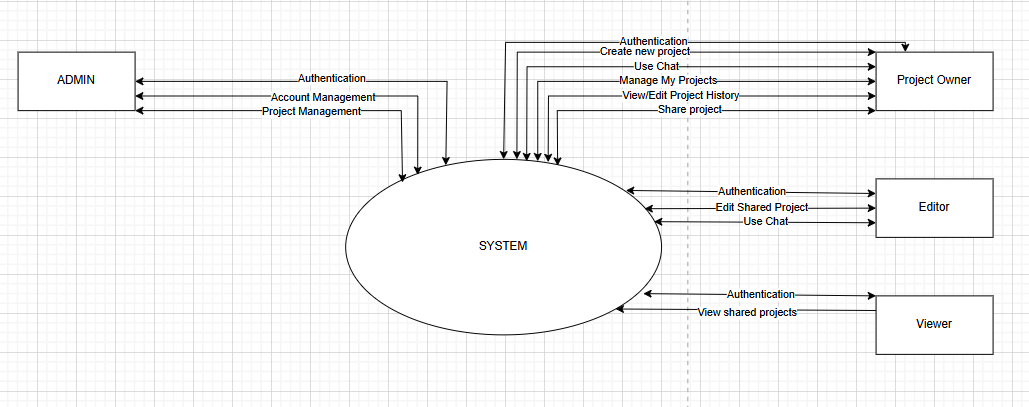
This project aims to bridge these gaps by developing a robust framework that leverages LLMs' capabilities to:

* **Integrate various input types:** Beyond just text, incorporating image and audio analysis for requirement gathering.
* **Generate a complete suite of interconnected artifacts:** Not just individual user stories or use cases, but also linking them to risks, test cases, actors, timelines, and preliminary architectural components.
* **Provide intelligent editing and refinement:** Allowing human experts to easily review, modify, and validate LLM-generated content through dedicated editors, ensuring accuracy and alignment with project goals.

By addressing these opportunities, this project seeks to push the boundaries of LLM application in software engineering, offering a more automated, efficient, and higher-quality approach to system analysis and design.

**4. Engineering Approach**

**4.1. System context diagram**



4.2. System context description

**Admin:**

* Admin can authenticate identity (login / logout).
* Admin can manage accounts in the system (Account Management).
* Admin can manage projects (Project Management).

**Project Owner:**

* Project Owner can authenticate identity (login / logout).
* Project Owner can create new projects.
* Project Owner can use chat functionality.
* Project Owner can manage their own projects.
* Project Owner can view/edit project history.
* Project Owner can share projects.

**Editor:**

* Editor can authenticate identity (login / logout).
* Editor can edit shared projects.
* Editor can use chat functionality.

**Viewer:**

* Viewer can authenticate identity (login / logout).
* Viewer can view shared projects.

**4.3. Technical Constraints**

Technical to develop:

* Framework backend: Nodejs (Javascript, Typescript), Flask (Python)
* Technology: HTML, CSS, Javascript
* Database: MongoDB.
* Version Control System: GitHub
* Team Management: Trello, Zalo, Google drive, Google doc
* Develop tools: Visual Studio Code, Postman

Environments:

* Internet Connection
* Operation System: Google Chrome, Microsoft Edge, CocCoc, Firefox

Other Constraints:

* Resource: 5 people.
* Budget: Limited.
* Time: The project must be completed within 05 months.

**5. Tasks and Deliverables**

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint | Task name | Description | Deliverables |
| **Sprint 0:** Initialization & Planning | Definition & Setup | - In-depth analysis of the architecture diagram to define the technology stack.  - Select and set up APIs for the LLM, OCR, and Speech-to-Text services.  - Set up the development environment, repository (Git), and CI/CD pipeline. | - Technology Stack Specification document.  - A detailed and prioritized Product Backlog.  - A ready development environment & API keys. |
| **Sprint 1:** Build Input Gateway & Pre-processor | Build the "Text Pre-processing Module | - Integrate an OCR library for character recognition from images.  - Integrate a Speech-to-Text service to transcribe audio.  - Build a module to extract content from document files. | A functional Pre-processing Module that can accept three input types (documents, images, audio) and return standardized, clean text content. |
| **Sprint 2:** Core Analysis & Initial Data Generation | Integrate the "LLM Brain" (First Pass) | - Develop initial prompts for the LLM to analyze text.  - Develop the logic for the "LLM Brain" to call and process results from the LLM API.  - Store the analysis results in a structured format (e.g., JSON, XML). | The "LLM Brain" can receive text and generate initial structured data.  - A quality assessment report of the LLM's analysis on a sample dataset. |
| **Sprint 3:** Build the Core Editors | Build the Interactive Editors (Core) | - Develop the user interface for:  + REQUIREMENT EDITOR  + USE CASE EDITOR  ACTOR EDITOR  + Connect these editors to the data generated by the LLM. | A functional application interface with three active editors (Requirement, Usecase, Actor). Users can view, edit, delete, and add objects. |
| **Sprint 4:** Expand Editor Suite & In-depth Analysis | Complete the Editor Tool Suite | - Develop advanced prompts to generate Test Cases and Risks.  - Build the user interface for:  + TEST CASE EDITOR  + RISK EDITOR  + TIMELINE EDITOR (basic version) | - An updated application with three new editors. The analysis flow from Requirement -> Use case -> Test case begins to form. |
| **Sprint 5:** Build Traceability Feature & Finalize | Build the Traceability Matrix | - Build the logic to create and store relationships between objects.&lt;br>- Develop the interface to display the traceability matrix.  - Complete the remaining editors (TEAM, GLOSSARY). | - A functional Traceability Matrix feature.  - All editors from the architecture are developed at a basic level.  - "Final Data (Live)" can now be generated. |
| **Sprint 6:** Optimization, Integration & Handover | Optimization & UAT | - Refine all prompts based on testing results.&lt;br>- Conduct User Acceptance Testing (UAT) with end-users (BAs, SAs, PMs) to gather feedback.&lt;br>- Optimize performance and fix outstanding bugs. | - UAT results report.&lt;br>- A stable, high-performance version of the application.&lt;br>- The final, optimized prompt set (final version). |
|  | Packaging & Documentation | - Write detailed user manuals for each editor and workflow.  - Package the product for deployment. | - **Final Product:**  - A complete application, ready for deployment.  - A full set of User Guide & Technical Documentation |

**6. Project Management**

**6.1. Cost/Budget for Project**

|  |  |  |
| --- | --- | --- |
| Full Name | Role | Salary Rate (USD/hour) |
| Phap, Nguyen Van | Scrum Master | 2 |
| Dat, Hoang Van Tien | Team Member | 2 |
| Cuong, Doan Viet | Team Member | 2 |
| Phong, Hoang Ba | Team Member | 2 |
| Quynh, Ho Dang | Team Member | 2 |

*Table 1. Cost person/hours*

|  |  |  |  |
| --- | --- | --- | --- |
| No | Criteria | Price | Total (USD) |
| 1 | Working hours | 2 | 3600 |
| 2 | Other cost | 100 | 500 |
|  |  |  | 4100 |

*Table 2. Total cost estimation*

|  |  |  |
| --- | --- | --- |
| Description | Amount | Unit |
| Number of members | 5 | Person |
| Number of working hours per day | 4 | Hours |
| The cost per hour per member | 2 | USD |
| The number of working days | 90 | Days |

*Table 3. Description*

- The explanation for the table

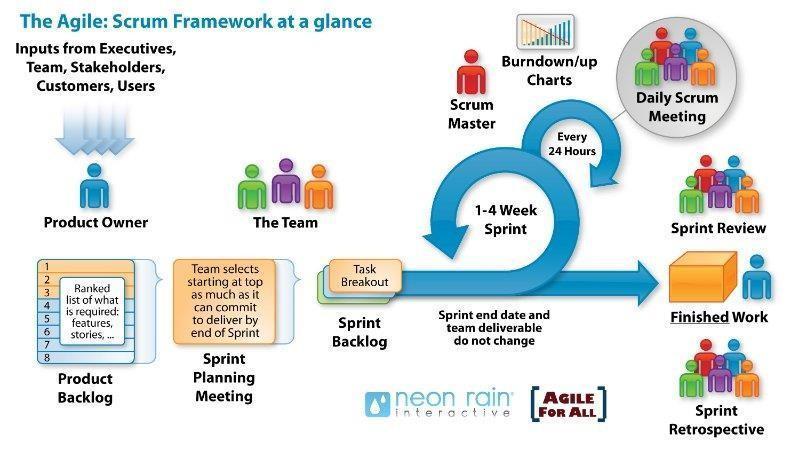
* Amount of working hours = 5 members \* 4 hours \* 90 days
* Other cost = 5 members \* 100 USD

**6.2. Tentative Schedule**

**6.2.1. Master Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| phase | Task Name | Duration | Start | Finish |
| Phase 1: Foundation & Planning | **Sprint 0:** Definition & Setup | 15 days (3 weeks) | 26/06/2025 | 16/07/2025 |
| **Phase 2:** Core MVP Development | **Sprint 1:** Build the "Text Pre-processing Module" | 15 days (3 weeks) | 17/07/2025 | 06/08/2025 |
|  | **Sprint 2:** Integrate the "LLM Brain" (First Pass) | 15 days (3 weeks) | 07/08/2025 | 27/08/2025 |
|  | **Sprint 3:** Build the Interactive Editors (Core) | 20 days (4 weeks) | 28/08/2025 | 25/09/2025 |
|  | **Sprint 4:** Complete the Editor Tool Suite | 20 days (4 weeks) | 26/09/2025 | 23/10/2025 |
| **Phase 3: Advanced Features** | **Sprint 5:** Build the Traceability Matrix | 20 days (4 weeks) | 24/10/2025 | 20/11/2025 |
| **Phase 4: Launch Preparation** | **Sprint 6.1:** Optimization & UAT | 10 days (2 weeks) | 21/11/2025 | 04/12/2025 |
|  | **Sprint 6.2:** Packaging & Documentation | 10 days (2 weeks) | 05/12/2025 | 18/12/2025 |

**6.2.2. Scrum Process**



* Scrum is an iterative and incremental agile software development framework for

managing software projects and product or application development.

* Scrum focuses on project management institutions where it is difficult to plan ahead.
* Mechanisms of empirical process control, where feedback loops that constitute

the core management technique is used as opposed to traditional command-and

-control management.

* Its approach to planning and managing projects is by bringing decision-making

authority to the level of operation properties and certainties.

* Benefit of the methodology:
  + - Project can respond easily to change.
    - Problems are identified early.
    - Customers get the most beneficial work first.
    - Work done will better meet the customer’s needs.
    - Improved productivity.
    - Ability to maintain a predictable schedule for delivery.

**7. Project Constraints**

|  |  |  |
| --- | --- | --- |
| Constraint | Constraints Description | Guidelines for Acceptance |
| Economic | • One-month free trial available  • After trial period:   + Monthly renewal: $4.99/month   + Annual renewal: $2.99/month   + Lifetime access: $99.99 one-time  • Key cost factors include design, development,maintenance, operation, and pricing strategy | The pricing structure must be reasonable and sustainable. Design and operational costs should be justified and optimized for long-term use.   |  | | --- | |  | |
| Environmental | The system has no direct impact on the environment. However, indirect impacts (e.g., server energy consumption) may be considered. | Consider the environmental footprint of the infrastructure. Where applicable, ensure design is compatible with recycling practices and minimal environmental disruption. |
| Ethical | • User data is encrypted upon registration  • No sharing of user data with third parties  • Full respect for user privacy and data security  • No user tracking or profiling | Adhere to ethical standards in data handling, including user privacy, data encryption, and transparency. Avoid conflicts between business goals and user safety. |
| Public health, safety, and welfare | As a web-based application, prolonged use may lead to eye strain or fatigue. It is recommended that users take breaks after 180 minutes of continuous use. | Comply with digital well-being standards. Inform users about healthy usage habits and consider interface design that reduces fatigue. |
| Social and Global | The application is designed to assist beginners, students, and professionals in learning system design, enhancing communication and collaboration skills, and fostering a sense of community. | Consider accessibility, inclusiveness, and the positive societal impact of the tool. Promote responsible AI usage and equitable access. |
| Cultural | • Primary language: Vietnamese  • Option to switch interface between Vietnamese and English | Respect cultural context by providing localization options. Ensure interface design is adaptable to different cultural norms and user expectations. |
| Sustainability | • Development team: members from Duy Tan University  • Technologies used:   + Backend: ExpressJS (Node.js framework)   + Frontend: VueJS   + Database: MongoDB   + AI Processing/Integration: Python  • The system architecture is designed to be modular and maintainable, with clear separation between components (frontend, backend, database, and AI services). | Ensure the technology stack supports scalability, ease of maintenance, and long-term evolution. The system should be designed with clear documentation, reusability of components, and efficient resource usage in mind. |

**8. Conclusion**

This project aims to develop a smart assistant platform powered by Large Language Models (LLMs) to enhance and optimize the process of software system analysis and design. By leveraging natural language understanding and automated modeling suggestions, the application helps bridge the gap between stakeholders and technical teams, streamlining communication and improving design accuracy.

The solution is designed to foster collaboration, reduce manual effort, and assist both beginners and experienced developers in building software systems more efficiently. With a modular and sustainable architecture built using ExpressJS, VueJS, MongoDB, and Python, the product ensures maintainability, scalability, and ease of future expansion.

The project is expected to be completed within 3 months, with a total estimated cost of $4100. Upon completion, it will provide a valuable tool for software engineers, analysts, and students, contributing to the improvement of software development workflows through intelligent automation.

**9. References**

[1] IEEE. *Software Development Standards for the Guidance and Control Software Project*. IEEE Standards Association.

[2] Sommerville, Ian. *Software Engineering* (10th Edition). Pearson, 2015.

[3] ISO/IEC/IEEE 29148:2018. *Systems and Software Engineering — Life Cycle Processes — Requirements Engineering*.

[4] OpenAI. *GPT-4 Technical Report*.<https://openai.com/research/gpt-4>

[5] MongoDB Inc. *MongoDB Documentation*.<https://www.mongodb.com/docs/>

[6] Vue.js Core Team. *Vue.js Official Documentation*.<https://vuejs.org/>

[7] Express.js Team. *Express.js Documentation*.<https://expressjs.com/>

[8] Python Software Foundation. *Python 3 Documentation*.<https://docs.python.org/3/>

10. Attachment