Ho Chi Minh University of Science
Department of Information Technology

PROJECT PROPOSAL
GROUP 06

Course: **Introduction to Software Engineering**

Instructor: **Mr. Truong Phuoc Loc**

Class: **23CLC03**

Students' name: **Le Anh Duy – 23127011**

Tran Gia Huy – 23127199

Nguyen Thanh Luan – 23127296

Nguyen Thanh Tien – 23127539

Ho Chi Minh City, November 2025

Table of Contents

1 Member Contribution Assessment

ID	Name	Contribution (%)	Signature
23127011	Le Anh Duy	100%	
23127199	Tran Gia Huy	100%	
23127296	Nguyen Thanh Luan	100%	
23127539	Nguyen Thanh Tien	100%	

2 Preliminary Problem Statement

Nowadays, with the rise of image recognition technology such as optical character recognition (OCR) and machine translation, users can conveniently read manga from around the world without language difficulties. As the team leader for a manga reading application, we need to build a web-based auto-translation platform for manga consisting of two subsystems: **the reader subsystem**, which displays manga pages and provides an automatic translation option for each page; and **the admin subsystem**, which manages, uploads, modifies, approves, including OCR and machine translation.

The **admin subsystem** can manage both users (uploader) and content. Admins can **assign roles to users**, such as the **uploader role**, which allows them to submit manga chapters to the system. When a manga chapter needs to be uploaded to the system, the users (uploaders) first provide descriptive information about this: chapter title, manga title, genre, number of pages, original language, and short summary of the content. This information must be accurate and clear. After submitting the information, the uploaded manga chapter will be **sent to the admin for approval**. The **admin** reviews the content to ensure suitable contents before publishing it to the website. Besides, admin also can lock or temporarily suspend a user account to maintain platform security.

The admin can **set these manga chapters as "Highlighted" or "Recommended"** to feature them on the homepage or in special sections, which are based on number of views, user ratings, ... In this case, the system allows the admin to **set an end date for priority display** (e.g. with a default of 7 days.) and they can adjust the end date.

If the admin chooses to **integrate translations with third-party services**, the system will connect to the service's API to retrieve the translation, then store and display it for readers.

As for the reader **subsystem**, readers must register an account to use the website, which ensures that all information will be encrypted. When a user opens a manga chapter, the system automatically displays the pages and provides options for translation. If the user **can choose their desired language**, the user (readers) subsystem sends each page to the translation module and receives the translated text in real time. Then, the screen would show their language.

Moreover, the reader subsystem also keeps track of the user's **reading progress**, including the current chapter, last page read, and translation status, which is **synchronized with the uploader system,** and they can **comment and rate the chapters**, providing feedback. Comments are shown to other users, increasing interaction and discussion, while ratings help highlight popular or high-quality chapters for the community.

The reader subsystem allows users to **share manga chapters to external platforms**, increasing engagement across social networks. Additionally, readers have the option to **register to become an uploader** by submitting a request to the admin. Once approved, they become the **uploader**, which allows them to upload, manage, and edit manga chapters in the system. Finally, readers can **pay for specific manga** using various methods (such as credit cards, ZaloPay, etc.), and they only need to pay a fee to read.

The website is implemented using **Next.js** and **Tailwind CSS** to provide an interactive interface for users. The application uses **client-side rendering (CSR)** for smooth user experience. **Node.js** with the **ExpressJS** framework handles requests, manages data, and communicates with the data storage. **PostgreSQL** is used as the relational database to store and manage structured data.

In addition, the system integrates a **third-party model** running in a separate service for **text detection and automatic translation**. This model is responsible for recognizing text in manga pages using **OCR** and generating translations in the user's selected language.

3 Proposed Solution

3.1 Software

3.1.1. Features

Demand	*Functional requirement*	*Non-functional requirement*
As a director, I want to see the statistics of users visiting the website.	User visit statistics	Reports must be displayed in 5 seconds, statistical data must be accurate and

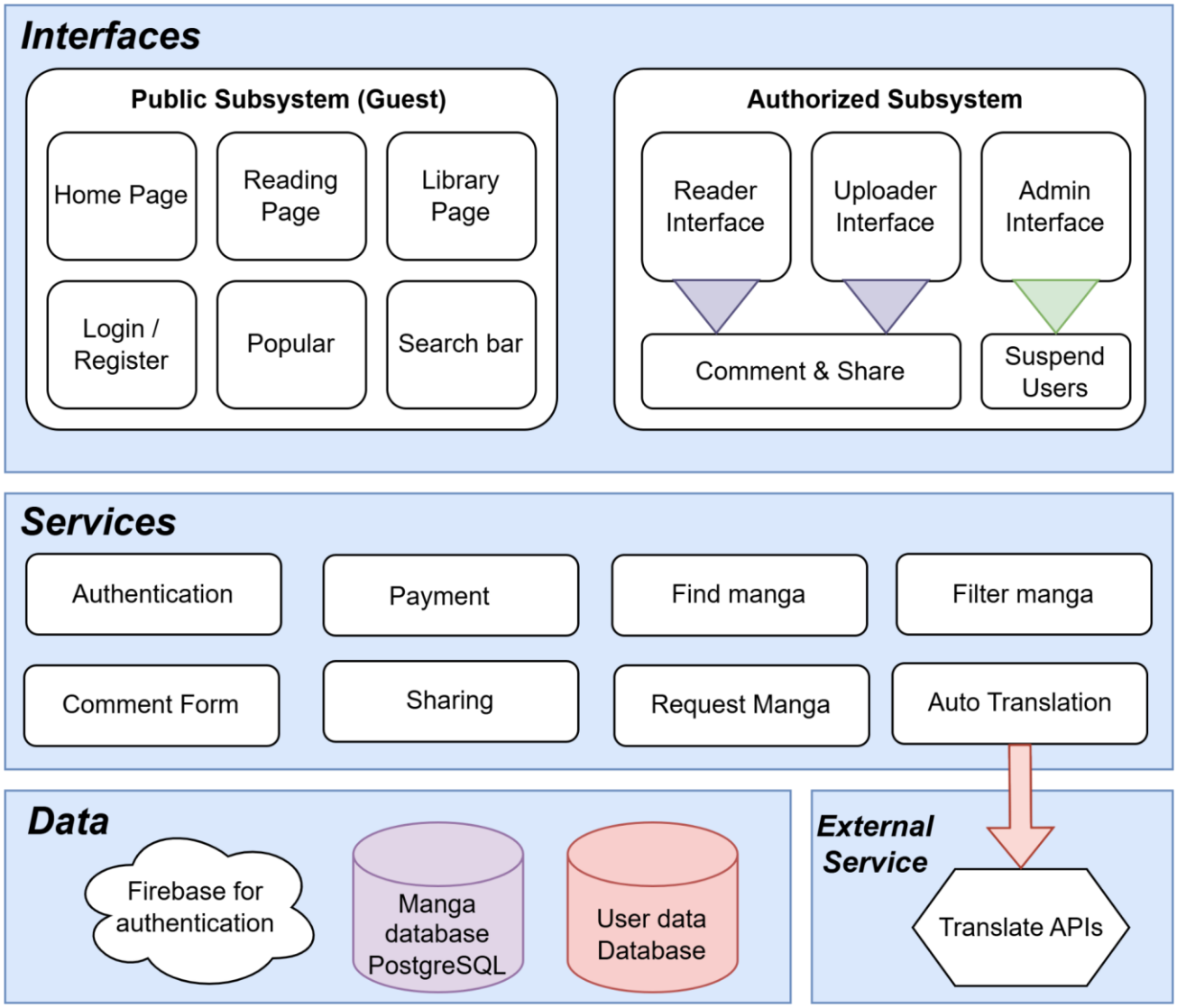
		updated in real time
As a director, I can see monthly and yearly revenue.	Revenue report	Fast report displays less than 5 seconds, accurate data
As an admin, I can add, update, and delete a manga.	Comic Management	The interface must be easy to operate, responding in 2 seconds
As an admin, I can assign roles to each member.	User authorization	The system must ensure that only administrators can authorize.
As an admin, approve new manga.	Review content before publishing	Ensure the story's content obeys the rules
As an admin, I can set featured or suggested stories to display on the home page	Manage featured and suggested mangas	Update changes displayed on the home page are less or equal than 3 seconds; only administrators have the right to operate
As an admin, I can lock or suspend user accounts	User management	Response time less than or equal to 2 seconds; record operation history
As a user, I can register to become an uploader to have the right to post	Register uploader	Send to the administrator for approval for less than a minute; response within 48 hours; only approved people have the right to post.
As an uploader, I can post stories.	Post mangas	Easy-to-use posting interface; upload stories less than 10MB/chapter; automatically send approval requests to administrators
As a user, I want to be able to login/register an account (google, Facebook,...)	Account Registration/Login	Authentication must take ≤ 3 seconds, login data is secured with HTTPS
As a registered user, I want the password to be encrypted and unreadable.	Password security	Passwords must be encrypted one-way (bcrypt or equivalent)
As a user, I want to find the author, genre, year of publication.	Search for comics	Search results returned in less than 2 seconds, supports case-insensitive search

As a reader, I want to have an automatic comic translation function.	Automatic comic translation	Processing time per page ≤ 5 seconds, ensuring translation accuracy ≥ 80%
As a reader, I want to be able to save comics to my favorites list.	Manage your favorites list	Favorite list data must be saved and not lost after logging out.
As a reader, I want to save the number of pages I am reading.	Save reading progress	Reading progress should be updated automatically every 30 seconds or when changing pages (avoid spam)
As a reader, I can comment and rate the quality.	Comments and reviews	Feedback must be displayed immediately after submission; the system must check offending content.
As a reader, I can pay to read the story (need to pay)	Payment function	Transactions must be safe and secure; payment less than 5 seconds; support multiple methods (cards, Zalo-pay, etc.)
As a user, I can share mangas on social networks	Share content	Share directly, respond within 2 seconds, via platforms (FB, Zalo,...)

3.1.2. Software Architecture

To meet the functional and non-functional requirements outlined in section 3.1.1, the team proposes building the system based on a clear 3-Tier Architecture. This architecture helps separate concerns, enhances maintainability, and allows for independent scaling of each component.

The system is divided into three main layers: **Interfaces, Services (Business Logic), and Data**, along with the **integration of third-party services.**



1. **Interfaces Layer:**
    - This is the layer responsible for all user interactions (UI/UX), developed using NextJS and Tailwind CSS to ensure a responsive and modern interface.
    - *This layer is divided into two main subsystems:*
        - **Public Subsystem (Guest):** For unauthenticated (Guest) users. It provides basic functionalities such as the Home Page, Reading Page, Library, and discovery features like the Popular page and Search bar. The Login/Register function also belongs to this subsystem.
        - **Authorized Subsystem:** Provides specialized interfaces based on the user's role after logging in, including: Reader Interface, Uploader Interface, and Admin Interface. Specific functions like "Comment & Share" or "Suspend Users" are tightly linked to these role-based interfaces.

2. **Services Layer:**
    - This is the core section of the system, handling all business logic. This layer is built with Node.js and ExpressJS to create API endpoints for the Interface layer to call.
    - *Key services include:*
        - Authentication: Handles login, registration, and session management.
        - Payment: Integrates payment gateways to process transactions for paid content and features.
        - Find/Filter Manga: Provides the logic for searching and filtering manga based on various criteria.
        - Comment Form & Sharing: Handles the submission, storage, and display of comments, as well as the logic for external sharing.
        - Auto Translation: This is the core service responsible for receiving translation requests from the reader. Which will be our paid feature.
3. **Data Layer and External Services:**
    - This final layer is responsible for data storage and connecting to external services.
    - *Internal Storage:*
        - Firebase for authentication: Uses Firebase Authentication to handle and store user authentication information securely and efficiently.
        - Manga database (PostgreSQL): The main relational (SQL) database, storing all information related to manga, chapters, pages, and other related data.
        - User data Database: A database (possibly a separate schema in PostgreSQL) to store user profile information, roles, permissions, reading progress, and comments/ratings.
    - *External Services:*
        - Translate APIs: Instead of building a costly OCR and Machine Translation model, the system's Auto Translation service will call third-party Translation APIs. The Interface layer will send the manga page to the "Auto Translation" service, which will process it (if necessary), call the external API, and then return the translated result to the user.
        - We might perform some lightweight OCR tasks, but the translation will be done with third-party services (Gemini, Google Translate, ...)

3.2 Hardware

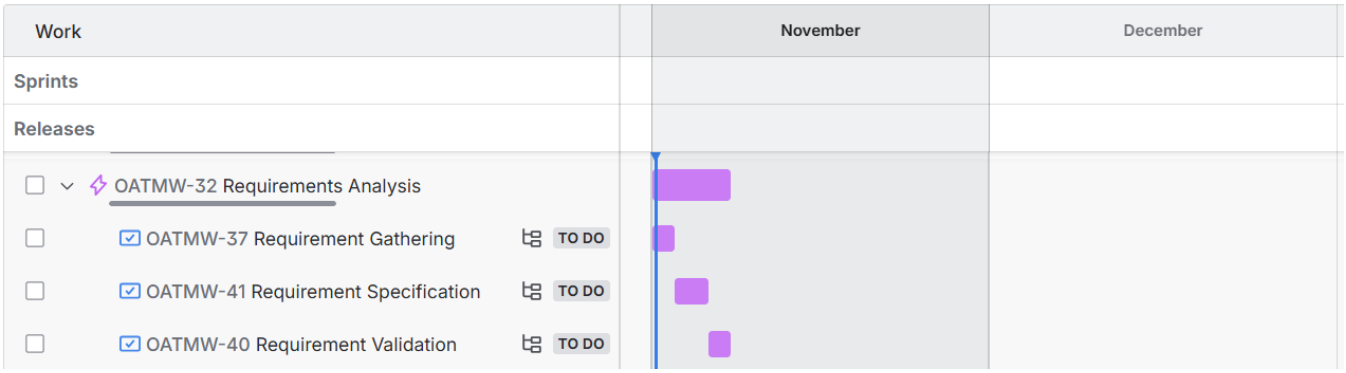
- Client:
    - Device: Computer, laptop or smartphone.
    - Browser: Modern web browsers such as Chrome, Firefox, Edge, or Safari.
    - Internet: Stable connection to load images and chapters smoothly.

- Server 1 (Backend)
  - Purpose: Handle API requests, business logic, database access, call third-party model server
  - CPU/RAM: 4 cores CPU, 8 GB RAM
  - Storage: SSD 50 GB (for build files, uploaded manga content, caching, and database storage)
  - Network: Public IP, stable internet to serve FE requests and connect to third-party model server
- Server 2 (Third-party model server – OCR + Translation)
  - Purpose: Run Machine Learning model for OCR and automatic translation
  - CPU: 4 cores CPU, 16 GB RAM
  - Storage: SSD 50 GB (to store model files, temporary inputs/outputs, and cache).
  - GPU: NVIDIA GTX 1650 / 1660 or RTX 2060, 4–6 GB VRAM, to accelerate
  - OS: Windows Server

4 Development Plan

4.1 Requirements Analysis

Goal: To understand what the system should do.



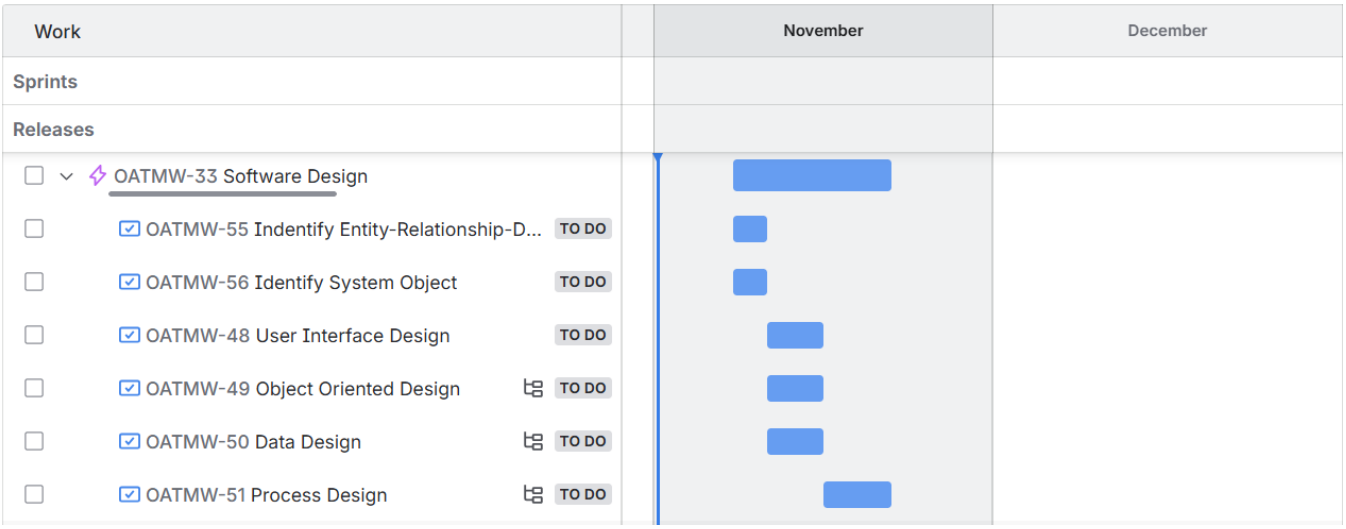
Deliverables:

- Table of questions and answers in the meeting.
- List of Functional requirements.
- List of Non-functional requirements.
- Define dependencies for each requirement.

- List of use cases.
- A prototype for each use cases.

4.2 Software Design

Goal: Describe how the system will be built.

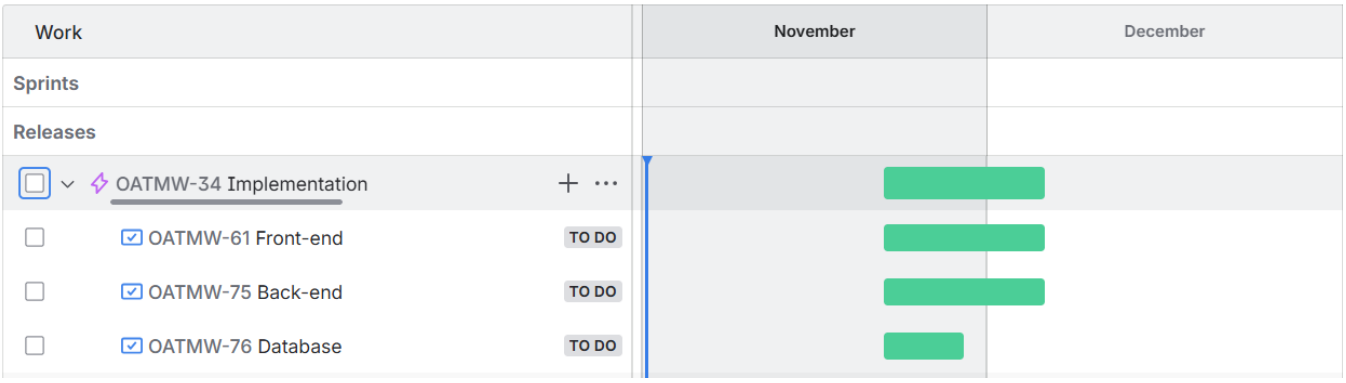


Deliverables:

- Entity-Relationship Diagram (ERD).
- System Object Diagram.
- UI Wireframes.
- Class diagram.
- Database schema.
- Document describes database.
- Process screen play for each use case.

4.3 Implementation

Goal: Implement follow system design.

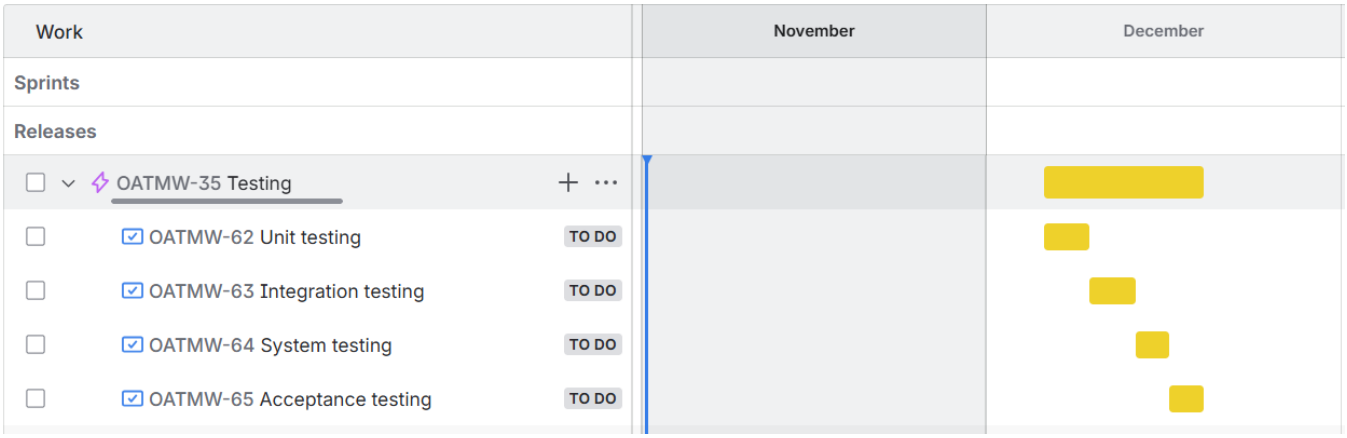


Deliverables:

- Sources code.
- Documents.
- Git version control log.

4.4 Testing

Goal: To ensure software meets requirements and error-free.

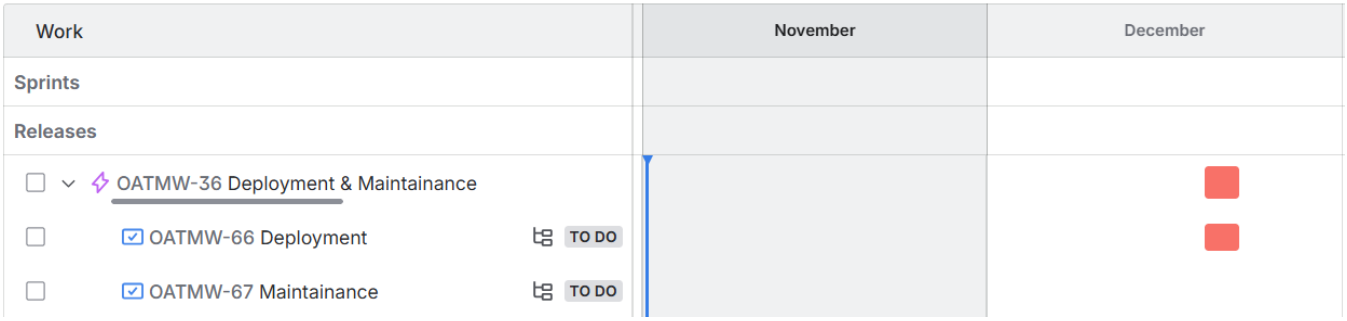


Deliverables:

- Test plan.
- Test cases.
- Issue reports.
- Final summary.

4.5 Deployment and Maintainance

Goal: Deploy software, keep it available and improve it over time.



Deliverables:

- User manual.
- Admin guide.
- Maintenance guide.

5 Human Resources & Costing Plan

5.1 Human Resources

5.1.1. *Requirements Analysis*

Role	Quantity	Key Responsibilities	Required Skills
Business Analyst	1	Requirements gathering, user story creation, prioritizing features for each development phase and validating features.	Communication skills, documentation skills and domain knowledge.

5.1.2. Software Design

Role	Quantity	Key Responsibilities	Required Skills
Database Engineer	1	Designs and manage database schema for users, manga, comments, roles, and payments.	MySQL / PostgreSQL, database normalization, security.
UI/UX Designer	2	Designs frames, mockups, and ensures smooth user interaction across web pages.	Wireframing, Prototyping (Figma/Sketch), User Research.

5.1.3. Implementation

Role	Quantity	Key Responsibilities	Required Skills
Backend Developer	2	Implement API endpoints, authentication, payment processing, and integration with third-party OCR/translation APIs.	Node.js, Express.js, REST API, database management, API integration.
Frontend Developer	2	Develop user interface for reader and admin subsystems using Next.js and Tailwind CSS.	Next.js, React, Tailwind CSS, responsive UI design.

5.1.4. Testing

Role	Quantity	Key Responsibilities	Required Skills

QA Tester	1	Test all subsystems, verify translation accuracy, performance and security. Write test cases.	Manual/Automated Testing, Test Case Design, Bug Tracking.

5.1.5. Deployment and Maintenance

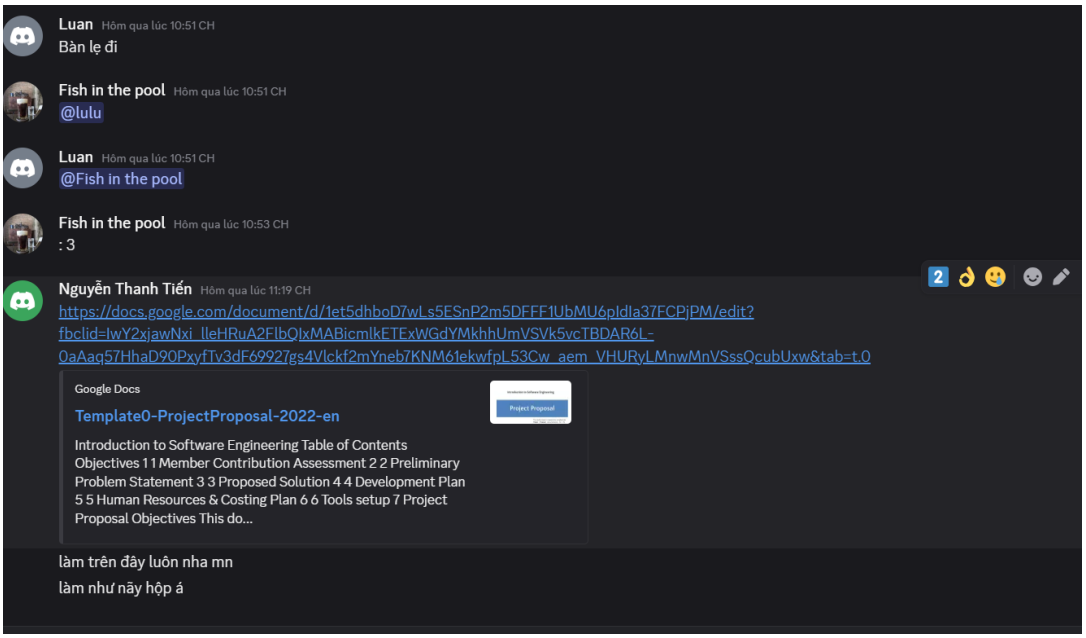
Role	Quantity	Key Responsibilities	Required Skills
DevOps	1	Manage server setup, deployment and continuous integration. Implement monitoring, backup, and recovery	AWS / Azure / Docker, CI/CD pipelines, Nginx.

5.2 Costing Plan

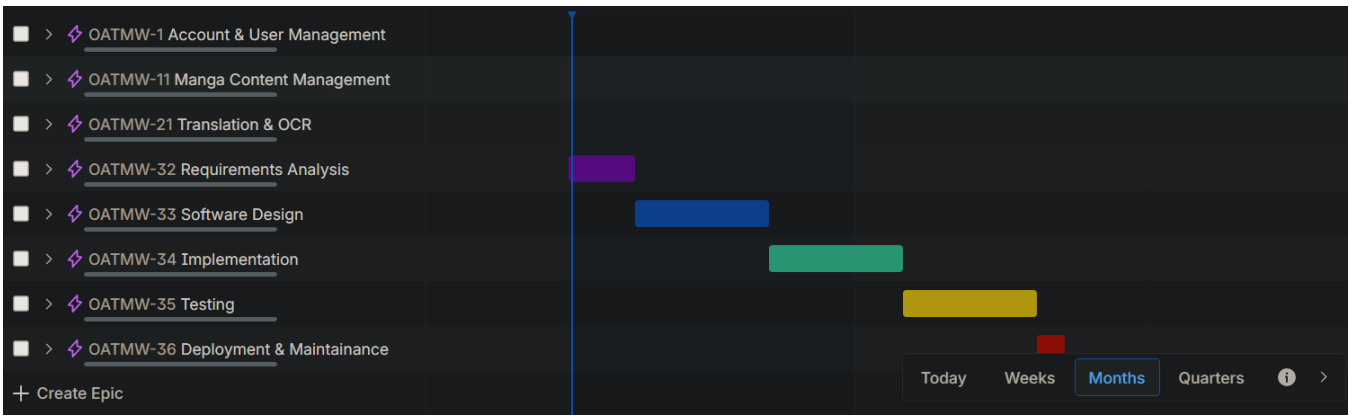
Category	Estimated Cost (2 months) (VND)	Description
Cloud Hosting	500,000	AWS EC2 / DigitalOcean server for backend + database.
Payment Gateway Integration	~2–3% per transaction	ZaloPay, MoMo.
Domain & SSL	400,000	Website domain and certificate.

6 Tools setup

6.1 Discord



6.2 *Jira*



6.3 *GitHub*

[GitHub Project]