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

Objectives 1

1 Member Contribution Assessment 2

2 Preliminary Problem Statement 3

3 Proposed Solution 4

4 Development Plan 5

5 Human Resources & Costing Plan 6

6 Tools setup 7

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
23127006	Trần Nguyễn Khải Luân	20%	
23127113	Nguyễn Trần Phú Quý	20%	
23127144	Đinh Đại Vũ	20%	
23127179	Nguyễn Bảo Duy	20%	
23127189	Trần Trọng Hiếu	20%	

2 Preliminary Problem Statement

2.1 Business Context and Problem

In the rapidly expanding world of online gaming and esports, players rely increasingly on data analytics to enhance their gameplay, understand strategies, and stay competitive. One of the most popular strategy-based games today, Teamfight Tactics (TFT), continuously evolves with new patches, units, and compositions. To remain competitive, players need accurate and up-to-date information about the current meta unit performance, and optimal team compositions.

However, this information is often fragmented across various online sources, requiring players to manually gather and analyze data. Many existing platforms present static or outdated data, lack intuitive interfaces, or fail to visualize trends in a user-friendly way.

2.2 Preliminary Solution

This project aims to address these issues by developing a web-based analytical platform, similar in concept to MetaTFT, which provides statistical insights, composition recommendations, and visualization of gameplay trends.

Upon completion, the project will deliver a fully functional web-based analytics platform for Teamfight Tactics. Users will be able to view game statistics, analyze top-performing compositions, and gain insight into trends across different patches and ranks.

To achieve this, the main objectives of the project are to:

- Develop a responsive, intuitive front-end interface for data exploration and visualization.
- Implement a scalable back-end API capable of handling real-time requests efficiently.

- Design a normalized database schema to store and query statistical data.
- Ensure secure and efficient communication between client, server, and database.

2.3 Operating Environment

The system is designed to operate in the following environment:

- **Client:** Modern web browsers supporting HTML5 (e.g., Google Chrome, Firefox, Safari).
- **Server (VPS-based Architecture):**
  - The entire application stack (Frontend and Backend) will be deployed using Docker containers.
  - These containers will be hosted on a **Virtual Private Server (VPS)**.
- **Container Registry:** Docker Hub will be used to store the project's Docker images.
- **Database:** Supabase (Cloud-hosted PostgreSQL).

2.4 Design & Implementation Constraints

The project will adhere to the following technical and design constraints:

- **Front-end:** React (using TypeScript).
- **Back-end:** Express (running on Node.js).
- **Database:** Supabase (PostgreSQL).
- **Version Control:** All source code and documentation changes will be managed via GitHub.
- **Containerization:** Docker will be utilized to ensure a consistent development environment.
- **Document Standard:** All project documentation (SRS, Design, etc.) will follow the templates provided by the course and be stored in the /docs repository folder.

2.5 Non-functional Constraints

The system must meet the following key non-functional requirements:

- **Performance:** The website must have an initial page load time of less than 5 seconds.

- **Usability:** The interface must feature a responsive design, ensuring a consistent and effective user experience on both desktop and mobile devices.
- **Security:** All API endpoints accessing sensitive user data must be protected and require proper user authentication (e.g., via JWT).

3 Proposed Solution

3.1 *Software*

3.1.1. Features

a. Justifying with User Story Mapping

To ensure our features are necessary and logically grouped, we used the User Story Mapping technique. This map organizes all stakeholder needs (R1-R14) into a visual journey, grouping features into three main "Epics" or user activities. This map justifies why we need each feature set.

Project User Story Map

Epic	1. Account & Social Features	2. Game Analysis	3. Administration
User Stories (Stakeholder Needs)	(R9) Login/Sign-up	(R1) Check Top Comps (Meta)	(R12) Moderate user content
	(R11) Customize profile	(R2) Check Game Stats (Champions, Items)	(R13) Get statistics (Dashboard)

	(R10) View other user profiles	(R3) Player Information Lookup (Match History)	(R14) Manage user accounts
	(R6) Post discussions	(R4) Team Builder	
	(R7) React and Comment		
	(R5) Capture/Save favorite teams		

b. List of Requirements

Functional Requirements

Id	Stakeholder	User Requirement (Stakeholder Need)	System Requirement (Feature)
R1	End User	I want to know which team compositions are currently the strongest.	Top Comps Table: Shows the strongest team compositions based on data.

R2	End User	I want to know detailed information about Champions, Items, and Upgrades.	Game Stats Page: Provides detailed information about champions, items, etc.
R3	End User	I want to see my match history and ranking (and other players').	Player Information Lookup: Displays info, match history, and rankings.
R4	End User	I want a Team Builder so I can try building a team.	Team Builder: Allows users to create and test their own team compositions.
R5	End User	I want to save my favorite teams to review later.	Capture Team: Allows users to store their team by using pictures.
R6	End User	I want to post to discuss issues in the game.	Discussion Posting: Users can upload images and write captions.
R7	End User	I want to be able to interact and comment on posts.	Reactions and Comments Feature

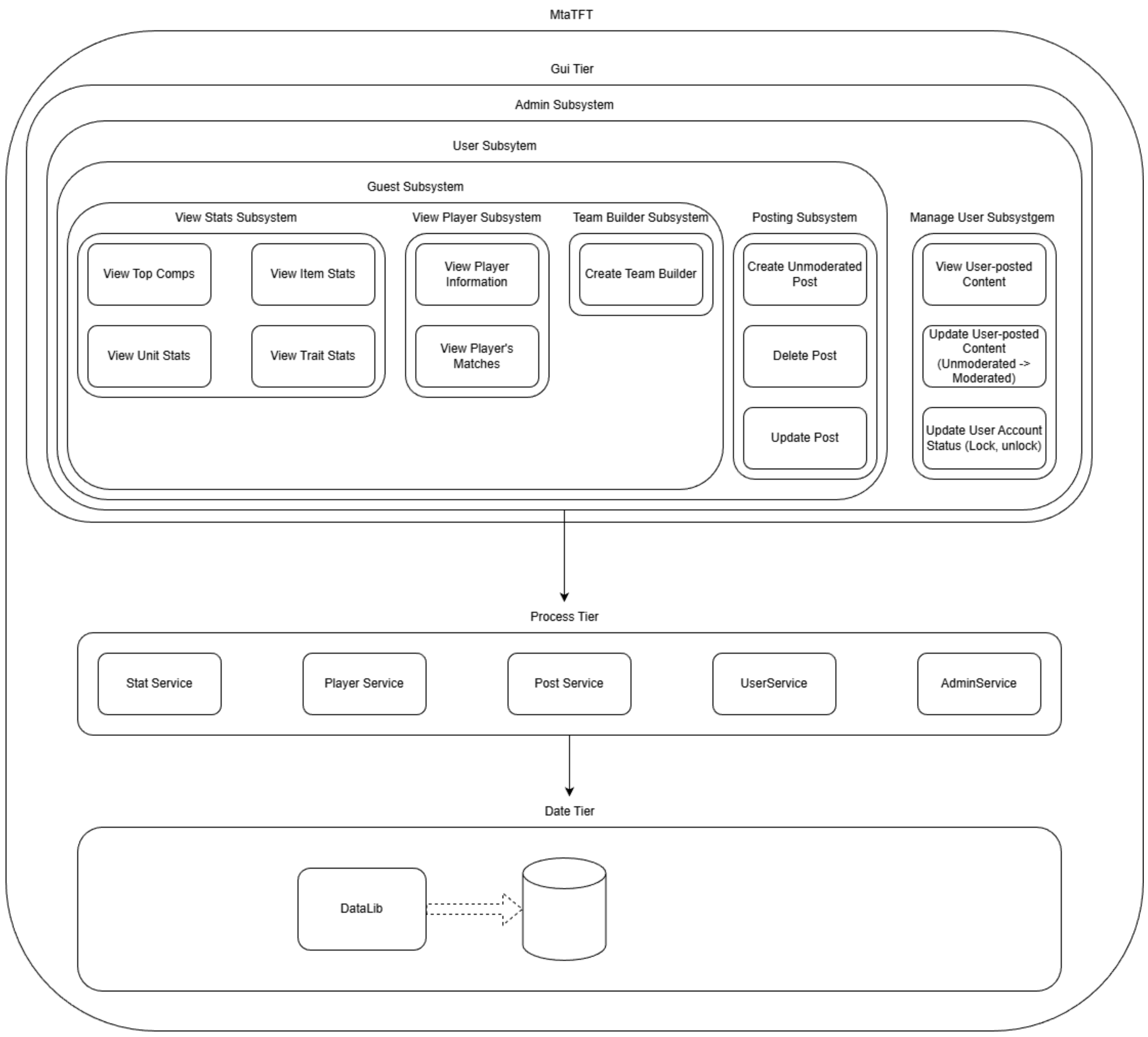
R9	End User	I want an account to post or save teams.	Authentication System: Login/Sign-up, password encryption, and profile management.
R10	End User	I want to follow the post history of my favorite users.	View Profile: Allows viewing the profile and post history of accounts.
R11	End User	I want to customize my personal information.	Customize Profile: Edit name, password, bio.
R12	Admin	I want to moderate user-submitted content.	Moderation Tools: Admin can review, edit, or delete violating posts.
R13	Admin	I want to get statistics on user activity and site visits.	Admin Dashboard: Displays user, post, and interaction statistics.
R14	Admin	I want to manage user accounts.	User Management: Admin can lock/unlock user accounts.

Non-functional Constraints

Id	Stakeholder	User Requirement	System Requirement
NR1	End User	I want the system to handle responses quickly.	Data response < 5 seconds
NR2	End User	I want the system to operate stably with few errors.	Requires clear testing
NR3	Riot Games	I want Riot API data to be used correctly.	Comply with Riot's API policies
NR4	End User	I want to ensure account information is managed securely.	No hardcoded keys/secrets
NR5	Supabase	We want data to be used properly and transparently.	Ensure data usage is transparent
NR6	End User	I do not want my data or account to be leaked.	Secure DB connection with SSL/TLS

3.1.2. Software Architecture

Architecture diagram



The project will be built using a **3-Tier Architecture**. This model is chosen because of the following reasons:

- **Security:** We can hide the Riot Games API Key and the JWT Secure Key on the server-side, preventing them from being exposed on the client.
- **Scalability:** If the logic in one tier is slow or needs an update, we can scale that single tier independently without affecting the others.

- **Separation of Concerns:** It separates the Client-side and Server-side code, making it easier to manage and better for teamwork. Changes to UI logic will not break the business logic or database operations.

Tier Overview:

- **GUI Tier (Presentation Tier):**
    - Displays the User Interface for user interaction, built with React.
    - Handles three main user types: Guest User, Logged-in User, and Admin.
    - Logged-in Users get additional posting features. Admins get moderation features and an "Admin" tag.
- **Process Tier (Application Tier):**
    - The Back-end responsible for handling all business logic.
    - It is broken into smaller services corresponding to features as shown in the architecture diagram.
- **Data Tier (Data Tier):**
    - Responsible for storing and retrieving all application data.
    - Uses a Supabase database.
    - The Process Tier communicates with this tier via a central DataLib to ensure security.

3.2 Hardware

3.2.1. End-User Equipment

The hardware requirements for the end-user are minimal:

- **Device:** Any personal computer (PC), laptop, or mobile device (smartphone/tablet).
- **Software:** A modern web browser supporting HTML5 (e.g., Google Chrome, Firefox, Safari).
- **Connection:** A stable internet connection is required to access the system.

3.2.2. Server-Side Hardware & Infrastructure

To ensure stable operation, the server-side infrastructure requirements are defined as follows:

- **Application Server:**
    - The system (e.g., deployed as Docker containers) will run on a **Cloud Linux Server (VPS)**.

  - The minimum required specifications for this server are: 1 core CPU, 2GB RAM and 20GB SSD.
- **Database:**
  - The project will utilize **Supabase** for its database needs.
  - The system requires a minimum of **500MB of cloud data storage**, which is covered by the Supabase Free Tier.

4 Development Plan

The team's development plan follows a plan-driven software process, dividing the project into five key phases. Each phase has a designated manager, a clear objective, and defined deliverables as outlined in the project schedule.

4.1 Requirements Analysis

- **Phase Managers: Vũ (BA Lead), Hiếu (QA/BA)**
- **Objective:** The goal is to define and document *what* the system must do, ensuring the team and instructors agree on the project scope (Specification).
- **Key Deliverables:**
  - Software Requirement Specification (SRS) Document
  - Completed User Story Map
  - Initial Product Backlog (as a User Story list) for JIRA

4.2 Software Design

- **Phase Managers: Vũ (UI/UX Designer), Duy (DB/Architecture)**
- **Objective:** The goal is to create the technical blueprint showing *how* the system will be built to meet the requirements from the SRS.
- **Key Deliverables:**
  - System Architecture Diagram (3-Tier)
  - Database ERD (Entity Relationship Diagram)
  - Completed Figma Mockups (UI/UX Design)
  - API Specification Document

4.3 Implementation

- **Phase Managers: Luân (FE Lead), Duy (BE Lead)**

- **Objective:** The goal is to write the code and build the working application based on the approved design blueprints.
- **Key Deliverables:**
    - Complete Source Code on GitHub
    - Docker Images (pushed to Docker Hub)
    - Unit Test Reports (if applicable)

4.4 Testing

- **Phase Manager: Hiếu (QA)**
- **Objective:** The goal is to validate the application against the SRS, ensuring it is stable, free of critical bugs, and meets user expectations (Validation).
- **Key Deliverables:**
    - A final report summarizing all testing activities, bug statistics, and confirming the application is "Ready for Deployment".
    - A complete record of all found bugs on JIRA.

4.5 Deployment and Maintenance

- **Phase Managers: Quý (DevOps)**
- **Objective:** The goal is to launch the application to production and establish a plan for its ongoing operation, including managing key operational risks.
- **Key Deliverables:**
    - The running application at www.mtatft.io.vn.
    - Configured CI/CD Pipeline: The automated GitHub Actions workflow.
    - A document outlining the maintenance process and the defined procedure for managing the Riot API key refresh.

4.6 Project Timeline

	Week 1	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**
Requirements Analysis	🟩	🟩						
Software Design			🟦					
Implementation				🟦	🟦	🟦		
Testing							🟦	
Deployment and Maintenance								🟪

5 Human Resources & Costing Plan

5.1 Human resources

No.	Student ID	Full Name	Role
1	23127006	Trần Nguyễn Khải Luân	Project Manager, Front-End
2	23127113	Nguyễn Trần Phú Quý	DevOps, Back-End, Tester
3	23127144	Đinh Đại Vũ	Business Analysis, Tester, UI/UX Designer
4	23127179	Nguyễn Bảo Duy	Front-End, Back-End, Database Architect
5	23127189	Trần Trọng Hiếu	Quality Assurance, Time Keeper, Main Tester

5.2 Costing plan

No.	*Object*	*Description*	*Estimated Price (thousand VND)*
1	Virtual Private Server	Server to run Docker containers (1 month) Specs: 1 core CPU, 2GB RAM, 20GB SSD	109
2	Supabase	Free Tier Storage Limit: 500 MB	0
3	Docker (Hub/Registry)	Using Free Tier to store images	0
4	Domain name	Cost to buy and maintain domain (1 year)	30
	Total		**139**

6 Tools setup

6.1 Tools Breakdown

Tool	Purpose
Moodle	**Submission:** The official platform for submitting all project deliverables for each Project Assignment (PA).
JIRA (link to board)	**Project Management:** Used for planning, tracking progress through Sprints, and managing all Tasks, Subtasks, and Bugs.
Slack (link to workspace)	**Communication:** The primary channel for all team discussions, quick meetings (huddles), and receiving automated notifications from JIRA/GitHub.
GitHub	**Source Code Management:** Used to host the entire project source code (`/src`) and all project documentation (`/docs`).

6.2 Repository Structure

We commit to strictly adhering to the required directory structure. The team's official GitHub repository is located at the following link:

Repository Link: mtatft

The detailed structure of the repository is as follows:

```
[mtatft]/
 │
 ├── /src/
 │    └── (Stores all source code for React, Node.js, etc.)
 │
 ├── /docs/
 │    ├── /management/
 │    │    └── (Stores the proposal file, weekly reports, progress reports, etc.)
 │    ├── /requirements/
 │    │    └── (Stores SRS document, User Stories, Use Cases, etc.)
 │    ├── /analysis and design/
 │    │    └── (Stores architecture diagrams, ERDs, UI/UX design files, etc.)
 │    └── /test/
 │         └── (Stores Test Plans, Test Cases, Test Reports, etc.)
 │
 └── /pa/
      └── (Subfolders for each Project Assignment (PA) submission)
```

6.3 Usage Commitment

We understand that the effective use of these tools is an important criterion for evaluating team performance. All members are committed to regularly updating their work on JIRA, Slack, Moodle, and GitHub to ensure transparency.