Smart Freelancing

Faculty of Computer and Information - Minia University - in Partial Fulfillment of the Requirements for the Degree of Bachelor of Computer Science.

By:

Abdelrahman Ashour abdelrahmanashour758@gmail.com	Ali Jamal Mohammed ag0355121@gmail.com
Abdelrahman Masoud abdelrahmanmasoud206@gmail.com	Ahmed Hassan ahmdhsnbas598@gmail.com
Amr Khaled khamr6524@gmail.com	Mohammed Ahmed m011236320@gmail.com

Supervised By:

Dr. Eman Mamdouh
@mu.edu.eg

Eng. Mostafa Mahmoud
@mu.edu.eg

Abstract

This project presents an intelligent freelance marketplace designed to enhance the experience of both clients and freelancers through advanced AI-driven capabilities...

ACKNOWLEDGEMENT

We are very thankful to everyone who supported us to complete this documentation effectively...

Contents

Chapter 1: Introduction

1.1 Main Idea

AI-powered freelancing platform that facilitates seamless interaction between clients and freelancers while enhancing the overall freelancing process through intelligent automation. The system not only enables users to post projects, submit proposals, and communicate securely, but also incorporates advanced AI features to evaluate freelancers' skills, analyze their profile strengths, and provide personalized recommendations for improvement.
Additionally, the platform empowers clients through smart proposal filtering, insightful analytics, and transparent project-progress tracking, enabling both parties to make better decisions while ensuring a more efficient, fair, and productive freelancing experience.

1.2 Project Scope

- Development of an AI-powered freelancing platform that connects clients and freelancers in a secure and user-friendly environment.
- Implementation of AI-based skill analysis to evaluate freelancers' profiles and assess their compatibility with posted jobs.
- Integration of a recommendation engine that suggests new or trending skills for freelancers to improve their profiles.
- Support for two user types – Client and Freelancer – each with personalized dashboards.
- Features for project posting, proposal submission, payments, and messaging between users.
- Integration of a secure payment gateway using PayPal with an Escrow system.
- Cloud-based file storage for project-related files and portfolio materials.
- AI-supported chat translation for multilingual communication between users.
- **Client Dashboard Insights:** Provides comprehensive project analytics, including the number of received proposals, the average proposal price, and estimated market pricing.

- **Proposal Filtering System:** Clients can filter freelancer proposals based on: Required budget range, Experience level, (Beginner, Intermediate, Expert), Delivery time, and Freelancer rating.
- **Project Progress Tracking:** Allows clients to monitor real-time project status, including task completion percentage, milestone updates, submitted files and approvals, as well as upcoming deadlines and late-task alerts.

Out-of-Scope:

- Advanced plagiarism or originality detection in submitted work.
- Automated code review or design validation beyond basic file checks.
- Blockchain-based payment or identity verification systems.
- Development of a dedicated mobile application (only responsive web supported).
- Training of proprietary deep-learning models (only pre-trained or fine-tuned models will be used).

1.3 Problem Statement

Freelance platforms face several challenges that reduce the efficiency of selecting freelancers and hinder the quality of collaboration between clients and professionals. Freelancers struggle to realistically evaluate and improve their profiles, and there is a lack of accurate mechanisms that enable them to know how suitable they are for different jobs. In addition, there is a clear gap between the skills listed in profiles and the actual skills required in the market, leading to a poor match between client needs and executors' capabilities. Clients also find it difficult to distinguish the best candidates when receiving a large number of applications, due to the absence of smart analytical tools that help them make decisions based on accurate data rather than personal judgment.

1.4 Objectives

This project aims to develop an independent AI-powered platform to improve the efficiency and transparency of the freelancer selection process. The objectives are summarized as follows:

1. Enhance freelancer profiles through intelligent tools that suggest edits, highlight strengths and weaknesses, and provide an accurate assessment of the profile's suitability for various roles.

2. Reduce the gap between stated skills and required skills through skill analysis models and evaluation techniques based on data and modern labor market standards.

3. Help clients make more accurate decisions through matching algorithms and analysis that assess the quality of applications and present the top candidates based on skills, experience, and actual project compatibility.

4. Enhance the experience for both parties (the client and the freelancer) through an easy-to-use interface and a recommendation system based on machine learning.

1.5 Methodology

Methodology (Agile Approach)

The development of the proposed system follows the **Agile methodology**, which emphasizes iterative development, continuous user feedback, and incremental improvement through short development cycles (Sprints). The methodology includes the following phases:

1. Planning

- Continuously gathering and analyzing requirements from both clients and freelancers.
- Identifying existing gaps in current freelancing platforms and establishing a prioritized backlog for each Sprint.

2. Design

- Designing the system architecture and decomposing it into core modules, including:
  - Skill Analysis Module
  - Profile Enhancement Module
  - Client–Freelancer Matching Module
- Creating data flow diagrams, database structure, and initial UI/UX prototypes.

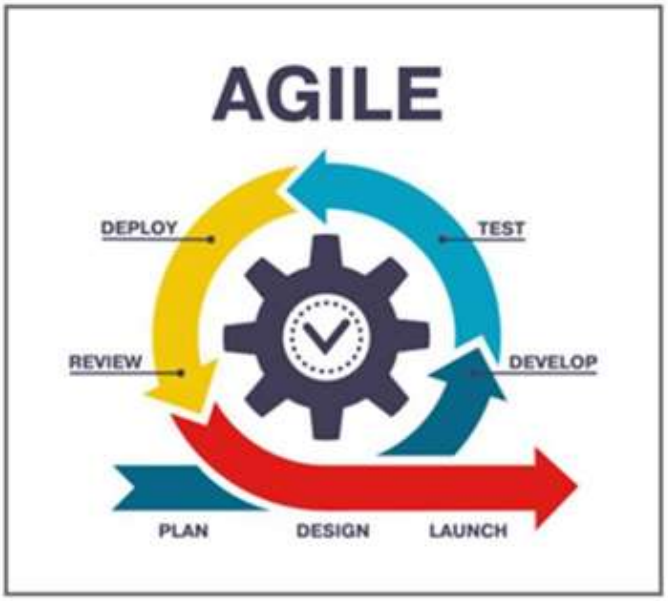


Figure 1.1 Agile Methodology.

3. Develop

- Implementing the features planned for each Sprint.
- Developing the AI components, including NLP-based skill extraction, profile analysis algorithms, and recommendation/classification models.
- Building and refining the user interface in parallel with backend development.

4. Test

- Conducting continuous testing throughout each Sprint.
- Evaluating:
  - The accuracy of AI recommendations
  - The performance of the skill analysis module
  - The usability and responsiveness of the interface
- Applying unit testing, integration testing, and scenario-based evaluation.

5. Deploy

- Deploying incremental versions of the system at the end of each Sprint.
- Ensuring that newly added features are smoothly integrated without disrupting existing functionality.

6. Review

- Reviewing Sprint outcomes with stakeholders and test users.

- Collecting feedback on performance, usability, and accuracy.
- Updating the backlog and defining improvement tasks for the next Sprint.

7. Launch (Incremental Release)

- Publishing improved and stable versions after completing multiple Sprints.
- Continuously enhancing the system to ensure high reliability, efficiency, and alignment with user needs.

Tools & Technologies:

- **Frontend:** Next.js (React)
- **Backend:** ASP.NET Core Web API
- **Database:** SQL Server
- **AI Models:** Python (scikit-learn / TensorFlow) – integrated via REST API
- **Authentication:** JWT (JSON Web Tokens)
- **Payment Gateway:** PayPal (Escrow system)
- **Cloud Storage:** Firebase / AWS S3
- **Deployment:** Docker + Azure Cloud Hosting

Architecture:

User (Client / Freelancer) → Next.js Frontend → .NET API Backend → SQL Server Database

↓

AI Module (Skill Analysis & Recommendation – Python API)

↓

Payment Gateway + Cloud Storage + Chat System

Chapter 2: Background and related work

2.1 Background

In recent years, freelancing platforms have grown rapidly as more professionals shift toward flexible and independent work models. However, despite this expansion, many platforms still rely on traditional keyword-based search, general ratings, or manual filtering to match freelancers with clients. These methods often fail to reflect true skill levels, understand the context of competencies, or accurately represent the alignment between a freelancer's profile and the actual requirements of the job market.

With advancements in modern technologies, ready-made AI APIs for text analysis and semantic understanding have become widely accessible. These tools enable deeper insight into user profiles, more accurate skill extraction, and improved evaluation processes without requiring developers to build NLP or machine-learning models from scratch. Such APIs provide a practical and efficient way to implement intelligent features within systems while significantly reducing development complexity.

This project leverages these capabilities by using pre-built AI services to analyze freelancer profiles, assess skill relevance, and extract semantic information from textual data. This approach allows the system to enhance matching accuracy, offer relevant insights, and improve the overall decision-making experience for both freelancers and clients.

2.2 Related Work

Several existing systems and research efforts have explored methods to enhance talent matching and skill assessment within freelancing and recruitment platforms. These works can generally be categorized in the following directions:

1. Keyword-based Matching in Traditional Platforms:

Platforms such as Upwork and Freelancer primarily rely on matching keywords between job descriptions and freelancer skills. Although simple, this method lacks semantic understanding and does not accurately measure skill quality or relevance.

2. Behavior-based Recommendation Systems:

Some studies have introduced recommendation systems based on user behavior, such as previous projects, browsing patterns, or client preferences. While effective in certain contexts, these systems often overlook the actual content and quality of freelancer skills.

3. Semantic Analysis for Job–Skill Matching:

A number of studies apply semantic processing to compare job requirements with freelancer skills. Although these methods improve analysis quality, they often focus solely on matching and rarely provide a comprehensive system that evaluates, scores, and improves the freelancer's profile.

The following table illustrates a comparison between the most popular traditional platforms:

Website-Name	**Advantages**	**Disadvantages**
Upwork	- Comprehensive system for project management and payments.- Large global community of clients and freelancers.- Supports long-term and specialized projects.	- Matching relies mainly on manual search and keywords.- High competition makes it difficult for new freelancers to be discovered.- No advanced analysis of freelancer skills or personalized recommendations.
Fiverr	- Simple interface for posting and browsing fixed-price services (Gigs).- Suitable for small and medium-sized tasks.- Clear categorization of services.	- No intelligent mechanism to match clients with the most suitable freelancer.- Relies heavily on ratings and sales history, not true skill assessment.- Hard for clients to evaluate quality without trial and error.

Freelancer.com	- Wide range of project categories.- Offers contests and bidding tools to attract freelancers.- Provides secure payment and milestone systems.	- Search and discovery are not enhanced by AI.- Competition is often unstructured, making selection difficult for clients.- Lacks tools for automatic profile improvement or performance analytics.

Contribution of This Project

This project distinguishes itself from previous work through:

- Integrating **ready-made AI APIs** to perform text analysis, skill extraction, and compatibility assessment efficiently.
- Focusing on **actual skill–job alignment**, not just textual similarity.
- Providing **profile evaluation and improvement suggestions**, offering actionable feedback for freelancers.
- Designing a **simple yet extensible system** suitable for academic work while delivering practical real-world value.

2.3 Summary

This chapter provided an overview of the theoretical foundations underlying the project, including the challenges faced by freelancing platforms and the limitations of traditional matching techniques. It highlighted the shift toward using ready-made AI tools to analyze text, extract skills, and understand user profiles with greater accuracy—offering a practical alternative to developing NLP or ML models from scratch.

The related-work review explored several approaches, such as keyword-based matching, behavior-driven recommendation systems, and AI-powered profile analysis. While each approach contributes valuable insights, most existing solutions address only part of the problem and often lack a fully integrated system for evaluating, improving, and matching freelancer profiles.

This chapter sets the foundation for the system design and methodology that follow, where the proposed solution integrates AI APIs to deliver more accurate skill analysis, improved profile recommendations, and enhanced matching for both clients and freelancers.

Chapter 3: Requirement analysis

3.1 Feasibility Study

Technical Feasibility:

The project is technically feasible using the proposed technologies:

- **Frontend:** Next.js with Redux Toolkit for state management, providing a responsive and dynamic user interface.
- **Backend:** ASP.NET Core Web API for handling business logic and secure client-freelancer interactions.
- **Database:** SQL Server to store user profiles, project details, proposals, and chat history.
- **AI/ML:** Python-based APIs (scikit-learn, TensorFlow) for skill analysis, profile evaluation, and recommendation engine.
- **Deployment:** Docker and Azure Cloud Hosting ensure scalability and smooth deployment.

Operational Feasibility:

- Users (clients and freelancers) can easily interact with the platform through a web interface.
- AI-based recommendations improve the decision-making process for both freelancers and clients.
- Secure payment integration with PayPal (Escrow) ensures trust and smooth transactions.

Economic Feasibility:

- Development costs are manageable within a student project scope.
- Using pre-trained AI models reduces training costs and computational resources.
- Cloud services and open-source frameworks lower infrastructure expenses.

3.2 Development Costs

Item	Estimated Cost	Notes
Frontend Development	Low	Next.js and Redux are open-source
Backend Development	Low	ASP.NET Core is free for students
AI/ML Model Integration	Medium	Using pre-trained models via Python APIs
Database & Hosting	Medium	SQL Server and Azure hosting (student plans)
UI/UX Design	Low	Figma
Payment Integration	Low	PayPal developer account is free
Total Estimated Cost	Low-Medium	Affordable for a graduation project

3.3 Organizational Study

- **Team Roles:**
  - Frontend Developers: Build UI components, integrate Redux state management.
  - Backend Developers: Create APIs, handle authentication, integrate AI and payment systems.
  - AI/ML Specialists: Implement skill analysis, profile evaluation, and recommendation engine.
  - UI/UX Designers: Design intuitive dashboards and responsive interfaces.
- **Communication Tools:** Slack or Microsoft Teams for coordination.
- **Project Management:** Agile methodology with 2-week sprints.

3.4 Proposed Features

Freelancer Features:

- Profile creation and skill tagging.
- AI-based skill evaluation and improvement suggestions.
- Proposal submission and tracking.
- Portfolio upload and file management.
- Chat with clients with AI-powered translation support.

Client Features:

- Project posting with detailed requirements.
- Proposal filtering by budget, experience, delivery time, and rating.
- Dashboard for analytics: average proposal price, number of proposals, top candidates.
- Real-time project progress tracking with milestones and deadlines.

Admin Features:

- User management and reporting.
- Monitoring AI recommendations and system performance.
- Payment oversight and dispute management.

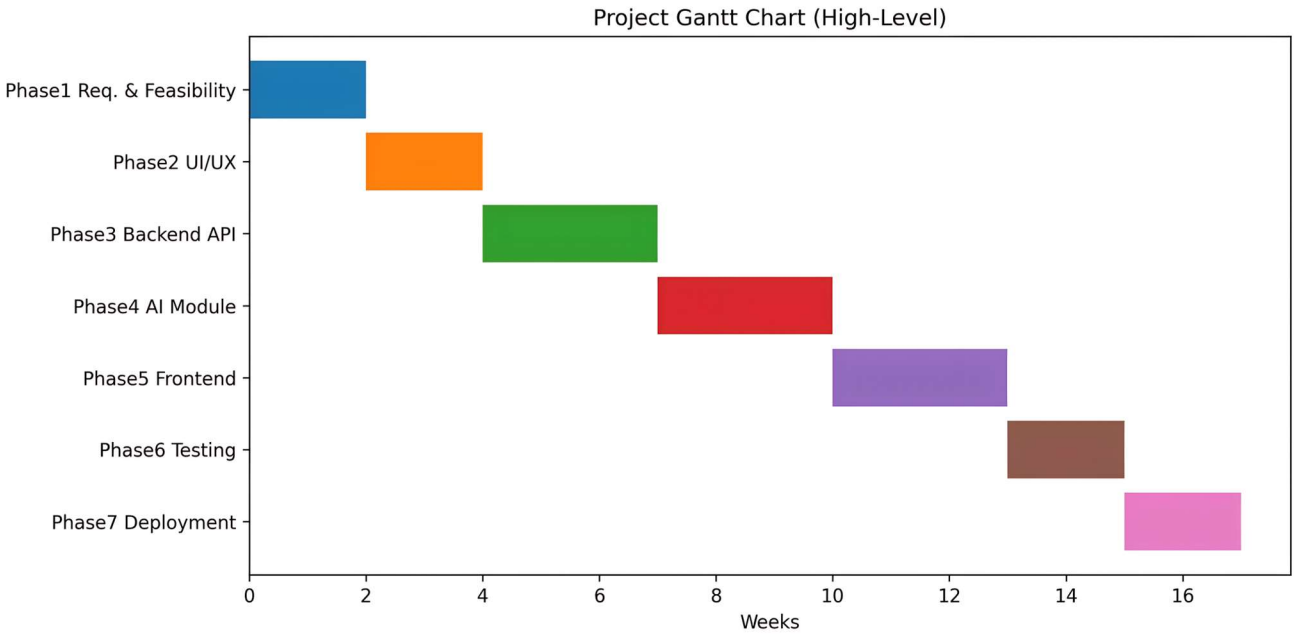
3.5 Risk Management

Risk	Impact	Mitigation
Delays in AI integration	High	Use pre-trained models and APIs
Cloud service downtime	Medium	Use redundant hosting and backup storage
Payment errors	High	Test PayPal sandbox and Escrow thoroughly
Feature creep	Medium	Stick to MVP scope for initial release
Security breaches	High	Implement JWT, encryption, and secure coding practices

3.6 Work plan

Phase	Description	Duration
Phase1	Requirement gathering, feasibility study, and backlog creation	2 weeks
Phase2	UI/UX design, prototyping, and user feedback	2 weeks
Phase3	Backend API development (user management, project management, payment integration)	3 weeks
Phase 4	AI module integration (skill analysis, profile recommendations, proposal ranking)	3 weeks
Phase 5	Frontend implementation with Redux state management and API integration	3 weeks
Phase 6	Testing and quality assurance (unit, integration, and scenario-based testing)	2 weeks
Phase 7	Deployment, documentation, and final review	1–2 weeks

3.7 Gantt chart



Chapter 4: Requirement Gathering

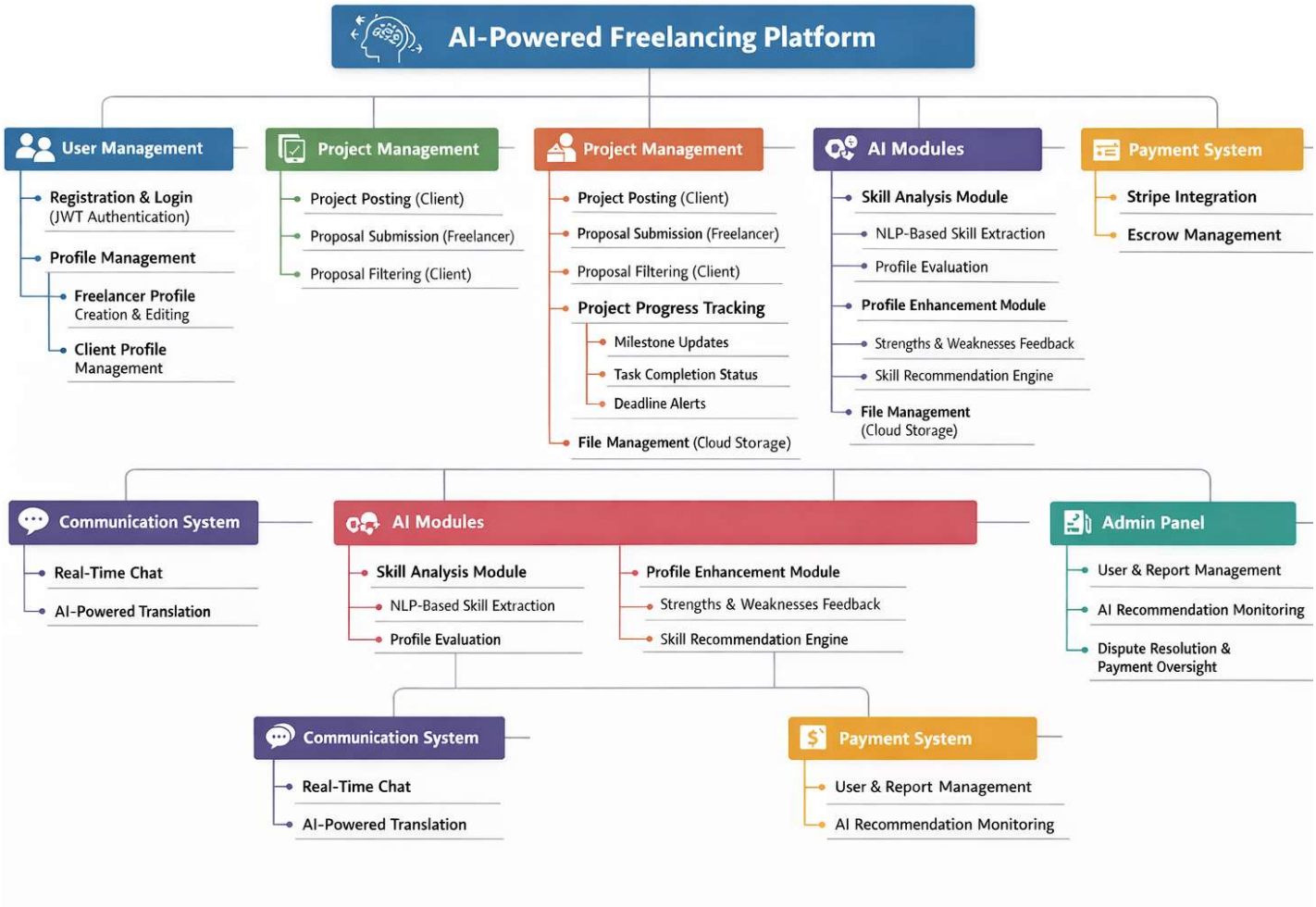
4.1 Functional Requirements

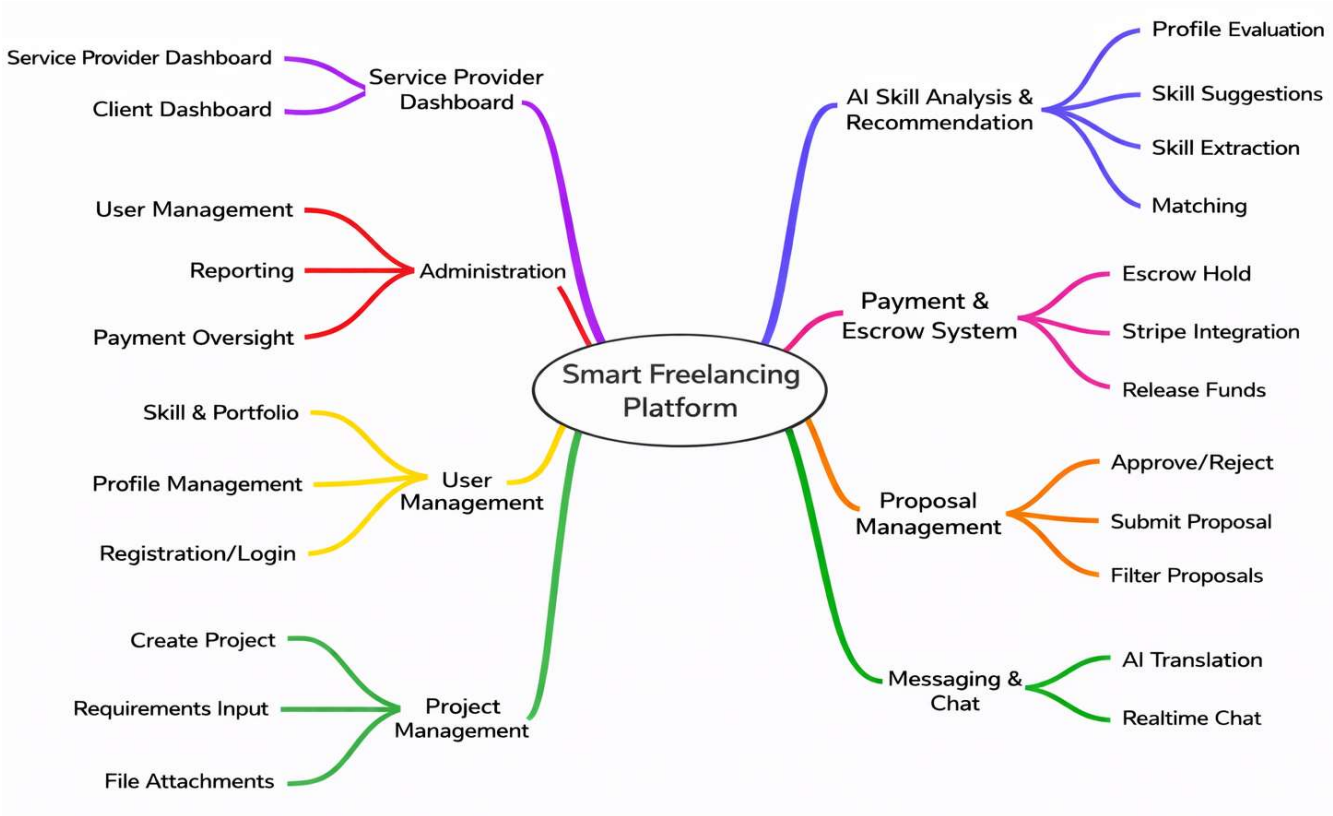
- Users can register and log in securely (JWT authentication).
- Freelancers can create and update profiles with skills and portfolio files.
- Clients can post projects with detailed specifications.
- AI module evaluates freelancer skills and recommends improvements.
- Proposal system allows freelancers to submit offers for projects.
- Clients can filter proposals based on multiple criteria.
- Real-time chat system with AI translation support.
- Payment system via PayPal Escrow for secure transactions.
- Dashboard analytics for clients and freelancers.

4.2 Non-functional Requirements

- **Performance:** Fast response time for API requests (<2 seconds).
- **Scalability:** Handle up to 10,000 concurrent users initially.
- **Reliability:** 99% uptime via Azure cloud hosting.
- **Security:** JWT authentication, encrypted communication (HTTPS), secure file storage.
- **Usability:** Responsive web design, intuitive navigation.

4.3 Functional (System) Decomposition Diagram




Smart Freelancing Platform
Service Provider Dashboard
Service Provider Dashboard
Client Dashboard
Administration
User Management
Reporting
Payment Oversight
User Management
Skill & Portfolio
Profile Management
Registration/Login
Project Management
Create Project
Requirements Input
File Attachments
AI Skill Analysis & Recommendation
Profile Evaluation
Skill Suggestions
Skill Extraction
Matching
Payment & Escrow System
Escrow Hold
Stripe Integration
Release Funds
Proposal Management
Approve/Reject
Submit Proposal
Filter Proposals
Messaging & Chat
AI Translation
Realtime Chat

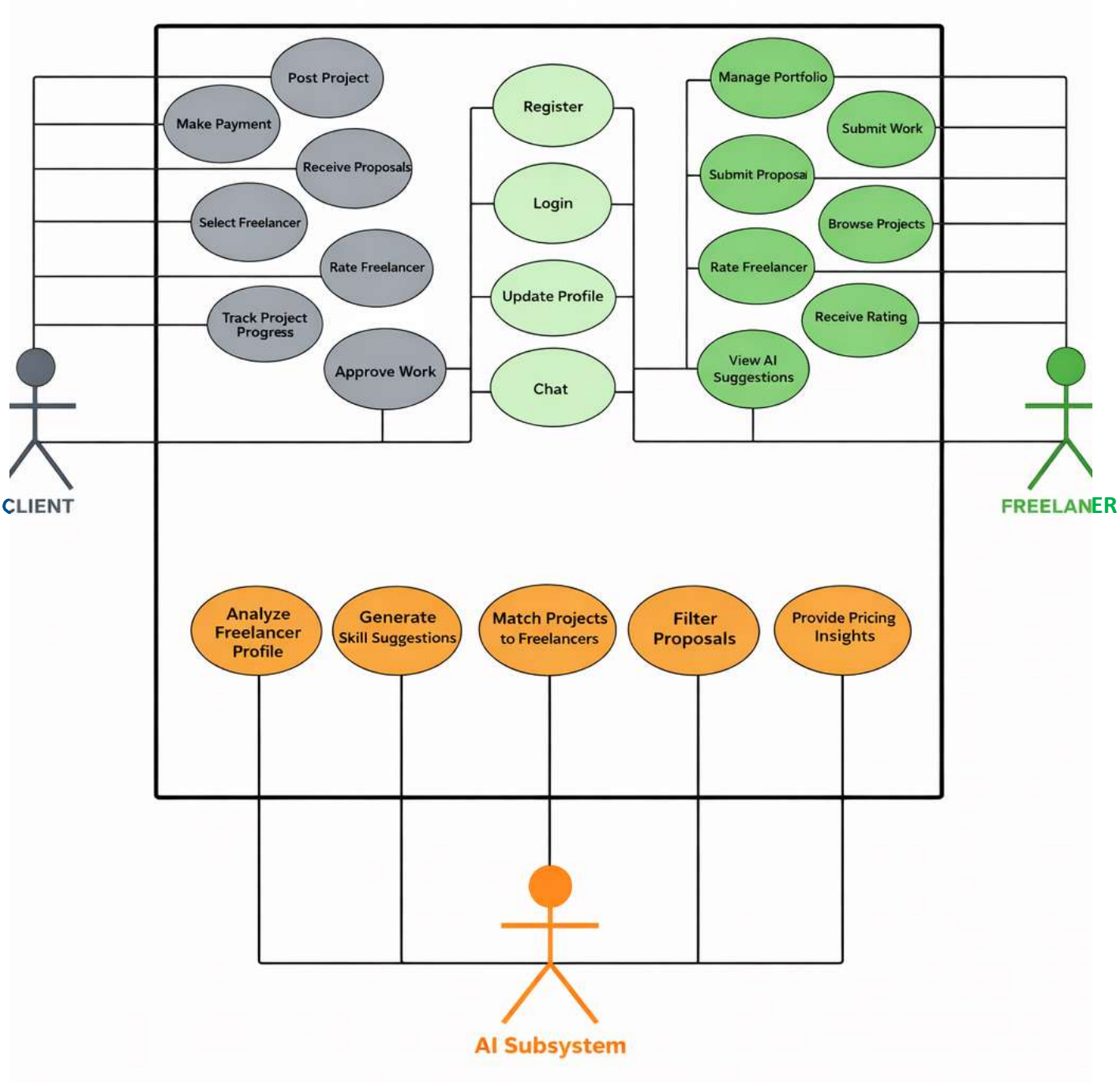
Chapter 5: System Architecture

5.1 Use Case Model

Actor	Use Cases
Client	• Register • Login • Update Profile • Post Project • Browse Freelancers • Receive & Filter Proposals • Select Freelancer • Track Project Progress • Approve Work • Make Payment • Chat with Freelancer • Rate Freelancer
Freelancer	• Register • Login • Update Profile • Manage Portfolio • Browse Projects • Submit Proposal • Submit Work • Receive Payment • Chat with Client • Receive Rating • View AI Suggestions

System (AI)	• Analyze Freelancer Profile • Generate Skill Recommendations • Match Projects to Freelancers • Filter Proposals • Provide Pricing Insights

5.2 Use Case Diagram



5.3 Use Case Formats

Brief Use Case Formats

1. **Register**

   A new user (client or freelancer) creates an account by providing email, password, and basic details.

2. **Login**

   A registered user logs into the platform using email and password.

3. **Update Profile**

   A user edits their personal information, profile picture, or contact details.

4. **Post Project**

   The client publishes a new project by entering title, description, budget, and deadline.

5. **Browse Freelancers**

   The client searches and views freelancer profiles based on skills, ratings, or experience.

6. **Receive & Filter Proposals**

   The client views proposals received for their project and filters them by price, rating, or delivery time.

7. **Select Freelancer**

   The client chooses a freelancer from the proposals to work on the project.

8. **Track Project Progress**

   The client monitors the real-time status, tasks, and milestones of an ongoing project.

9. Approve Work

   The client reviews the delivered work and approves it, releasing payment to the freelancer.

10. **Make Payment**

    The client pays for a project or milestone using the integrated payment gateway.

11. **Rate Freelancer**

The client rates and reviews the freelancer after project completion.

12. **Manage Portfolio**

The freelancer adds, edits, or removes work samples, skills, and experience from their profile.

13. Browse Projects

The freelancer searches and views available projects posted by clients.

14. Submit Proposal

The freelancer submits a proposal for a project, including price, timeline, and cover letter.

15. Submit Work

The freelancer delivers the completed project or a milestone to the client.

16. Receive Payment

The freelancer receives payment for an approved milestone or completed project.

17. Receive Rating

The freelancer receives a rating and review from the client after project completion.

18. View AI Suggestions

The freelancer views personalized skill and profile improvement recommendations from the AI subsystem.

19. Chat

The client and freelancer exchange messages in a real-time chat window.

20. Analyze Freelancer Profile

The AI subsystem evaluates a freelancer's profile and skills to generate a compatibility score.

21. **Generate Skill Recommendations**

The AI subsystem suggests new or trending skills for the freelancer to learn or add.

22. Match Projects to Freelancers

The AI subsystem recommends relevant projects to freelancers based on their skills and preferences.

23. Filter Proposals

The AI subsystem automatically filters and ranks proposals for the client based on relevance and quality.

24. Provide Pricing Insights

The AI subsystem suggests competitive price ranges for projects based on market trends and freelancer experience.

Casual Use Case Formats

Use Case: Post a Project

Actor: Client

Main Success Scenario:

1. The client logs into the platform using their email/username and password.
2. The client navigates to the "Post a Project" page from the dashboard.
3. The client enters the following required details:
    - Project title
    - Detailed description
    - Required skills/tags
    - Fixed budget (single amount)
    - Deadline for submission
    - Project category
4. The client optionally uploads any reference files or additional documents.

5. The client reviews the information and clicks the "Publish Project" button.
6. The system validates all inputs (e.g., title length, valid future date, positive budget).
7. The system saves the project and makes it visible in the public project listings.
8. The client receives a confirmation message: "Your project has been published successfully."

Alternative Scenarios:

A1. Missing Required Fields:
In Step 6, if any required field is empty, the system displays an error message:
"Please fill in all required fields: Title, Description, Skills, Budget, and Deadline."

A2. Invalid Budget (Zero or Negative):
In Step 6, if the budget is zero or a negative number, the system shows:
"Budget must be a positive amount."

A3. Past Deadline:
In Step 6, if the selected deadline is a past date, the system warns:
"Deadline must be a future date."

Use Case: Submit Proposal

Actor: Freelancer

Main Success Scenario:

1. The freelancer logs into the platform using their email/username and password.
2. The freelancer browses available projects and selects one to view details.
3. The freelancer clicks the "Submit Proposal" button on the project page.
4. The freelancer fills in the proposal form with:
    - Proposed price (can be equal to or different from the client's budget)
    - Estimated delivery time (in days)

- A cover letter explaining their approach and suitability
- Optional attachments (portfolio samples, previous work)

5. The freelancer reviews and clicks "Send Proposal."
6. The system validates the inputs (price > 0, delivery time reasonable).
7. The system saves the proposal and notifies the client.
8. The freelancer sees a confirmation: "Your proposal has been submitted successfully."

Alternative Scenarios:

A1. Project Not Accepting Proposals:
In Step 3, if the project is closed or no longer accepting proposals, the system shows:
"This project is no longer accepting new proposals."

A2. Invalid Price Input:
In Step 6, if the proposed price is zero or negative, the system displays:
"Proposed price must be a positive amount."

A3. Duplicate Proposal:
If the freelancer has already submitted a proposal for this project, the system warns:
"You have already submitted a proposal for this project. You can edit your existing proposal."

Use Case: Filter Proposals

Actor: Client

Main Success Scenario:

1. The client logs into the platform.
2. The client navigates to "My Projects" and selects a published project.
3. The client clicks on the "View Proposals" tab.
4. The client applies filters from the sidebar:

- **Budget range** (minimum and maximum price)
- **Minimum freelancer rating** (e.g., 4 stars and above)
- **Maximum delivery time** (e.g., within 14 days)

5. The client clicks "Apply Filters."
6. The system filters the list of proposals in real-time based on the selected criteria.
7. The client reviews the filtered and sorted proposals.

Alternative Scenarios:

A1. No Proposals Match Filters:

In Step 6, if no proposals meet all filter criteria, the system displays:

"No proposals match your current filters. Try broadening your criteria."

A2. AI-Assisted Filtering Toggle:

The client can enable "Smart Filter" (AI), which automatically prioritizes proposals based on:

- Skill match score
- Freelancer reliability (on-time delivery rate)
- Proposal relevance (cover letter quality)

When enabled, the system re-ranks proposals without manual filter input.

Use Case: Submit Work

Actor: Freelancer

Main Success Scenario:

1. The freelancer logs into the platform.
2. The freelancer goes to "My Projects" and selects an active project.
3. The freelancer clicks the "Submit Work" button.

4. The freelancer uploads the completed files (documents, code, designs, etc.).
5. The freelancer adds a delivery message describing what has been delivered.
6. The freelancer clicks "Deliver Work."
7. The system validates the upload (file type, size) and saves the delivery.
8. The client is notified immediately that work has been submitted.
9. The freelancer sees a confirmation: "Work submitted successfully. Waiting for client review."

Alternative Scenarios:

A1. No Files Attached:
In Step 7, if no file is attached, the system shows:
"Please attach at least one file to deliver work."

A2. Large File Size:
If the uploaded file exceeds the size limit, the system warns:
"File size exceeds the limit. Please compress or split the file."

A3. Early/Late Submission Note:
The system may display a note:
"Submitted [X days] before/after the deadline."

Use Case: Approve Work

Actor: Client

Main Success Scenario:

1. The client logs into the platform and sees a notification: "Work submitted for [Project Name]".
2. The client navigates to the project page and clicks "Review Work".

3. The client downloads and examines the delivered files.
4. If satisfied, the client clicks the "Approve & Pay" button.
5. The system releases the payment from escrow to the freelancer's account.
6. The project status is updated to "Completed".
7. Both the client and freelancer receive a confirmation notification and email.

Alternative Scenarios:

A1. Request Revision:

If the client wants changes, they click "Request Revision", add comments, and resubmit the request.

The freelancer is notified and can submit updated work.

Use Case: Chat

Actor: Client, Freelancer

Main Success Scenario:

1. The user (client or freelancer) logs into the platform.
2. The user navigates to the project page and clicks the "Chat" button.
3. The chat window opens, showing the message history (if any).
4. The user types a message in the input field and clicks "Send".
5. The system delivers the message instantly to the other user.
6. The other user sees a notification and can reply in the same chat.

Alternative Scenarios:

A1. File Attachment:

- The user can attach a file (image, document, etc.) before sending.
- The system validates file type and size before allowing the upload.

A2. Real-time Translation:

- If the users have different language preferences, the system can auto-translate messages.
- A small icon indicates that the message was translated.

Fully Dressed Use case formats

Use Case: Analyze Freelancer Profile (AI)

Primary Actor: AI Subsystem

Secondary Actors: Freelancer (receives suggestions)

Stakeholders and Interests:

- **Freelancer:** Wants to understand their skill level and how to improve.
- **System:** Needs reliable profile data to match freelancers with projects.

Preconditions:

- The freelancer has a profile with basic information (skills, bio).
- The AI service is available.

Postconditions:

- The freelancer receives a profile strength score.
- Improvement suggestions are saved and displayed in their dashboard.

Main Success Scenario:

1. The system collects the freelancer's profile data.
2. The system sends the data to the AI module.
3. The AI checks the skills listed and compares them with common job market requirements.
4. The AI calculates a simple **profile score** based on skill completeness and relevance.
5. The AI suggests a few missing but popular skills that the freelancer could add.
6. The system shows these results in the freelancer's "AI Insights" section.

Extensions (Alternative Scenarios):

- If the profile is very new or empty, the AI suggests completing basic information first.
- If the AI service is down, the system shows a message to try again later.

Special Requirements:

- The analysis should be simple and fast.
- Suggestions should be clear and easy to understand.

Frequency of Use:

- When a freelancer updates their profile.
- Once per week automatically.

Use Case: Match Projects to Freelancers (AI)

Primary Actor: AI Subsystem

Secondary Actors: Freelancer, Client

Stakeholders and Interests:

- **Freelancer:** Wants to find projects that fit their skills.
- **Client:** Wants to see freelancers who can do their project well.

Preconditions:

- There are active projects and freelancers with profiles.

Postconditions:

- Freelancers see a list of projects that might suit them.
- Clients see a list of freelancers who might be a good fit.

Main Success Scenario:

1. The system gathers new and active projects.
2. For each freelancer, the system checks their skills and past work.
3. The AI compares each project's required skills with each freelancer's skills.
4. The AI gives each freelancer–project pair a **match score**.

5. The system shows freelancers the projects with the highest match scores.
6. The system can also show clients which freelancers have high match scores for their project.

Extensions (Alternative Scenarios):

- If a project has rare skills, the system tells the client that matches may be limited.
- For new freelancers, matching relies more on listed skills than past history.

Special Requirements:

- Matching should be quick.
- The system should explain briefly why a project is recommended (e.g., "Matches your skills in Python").

Frequency of Use:

- Regularly, and when a new project is posted.

Use Case: Filter Proposals (AI)

Primary Actor: AI Subsystem

Secondary Actor: Client

Stakeholders and Interests:

- **Client:** Wants to see the best proposals first without reading all of them.

Preconditions:

- The client has a project with at least one proposal.

Postconditions:

- Proposals are sorted so the most relevant ones appear at the top.

Main Success Scenario:

1. The client opens the list of proposals for their project.
2. The system sends the proposals to the AI helper.

3. The AI looks at each proposal and checks:
    - Does the freelancer's skills match the project?
    - What is the freelancer's rating?
    - Is the proposed price reasonable for the project budget?
    - Is the cover letter well-written and relevant?
4. The AI gives each proposal a **relevance score**.
5. The system reorders the proposals by score (highest first).
6. The client sees the most promising proposals at the top.

Extensions (Alternative Scenarios):

- If all proposals have similar scores, they stay in the original order.
- The client can turn off AI sorting and see proposals as they came in.

Special Requirements:

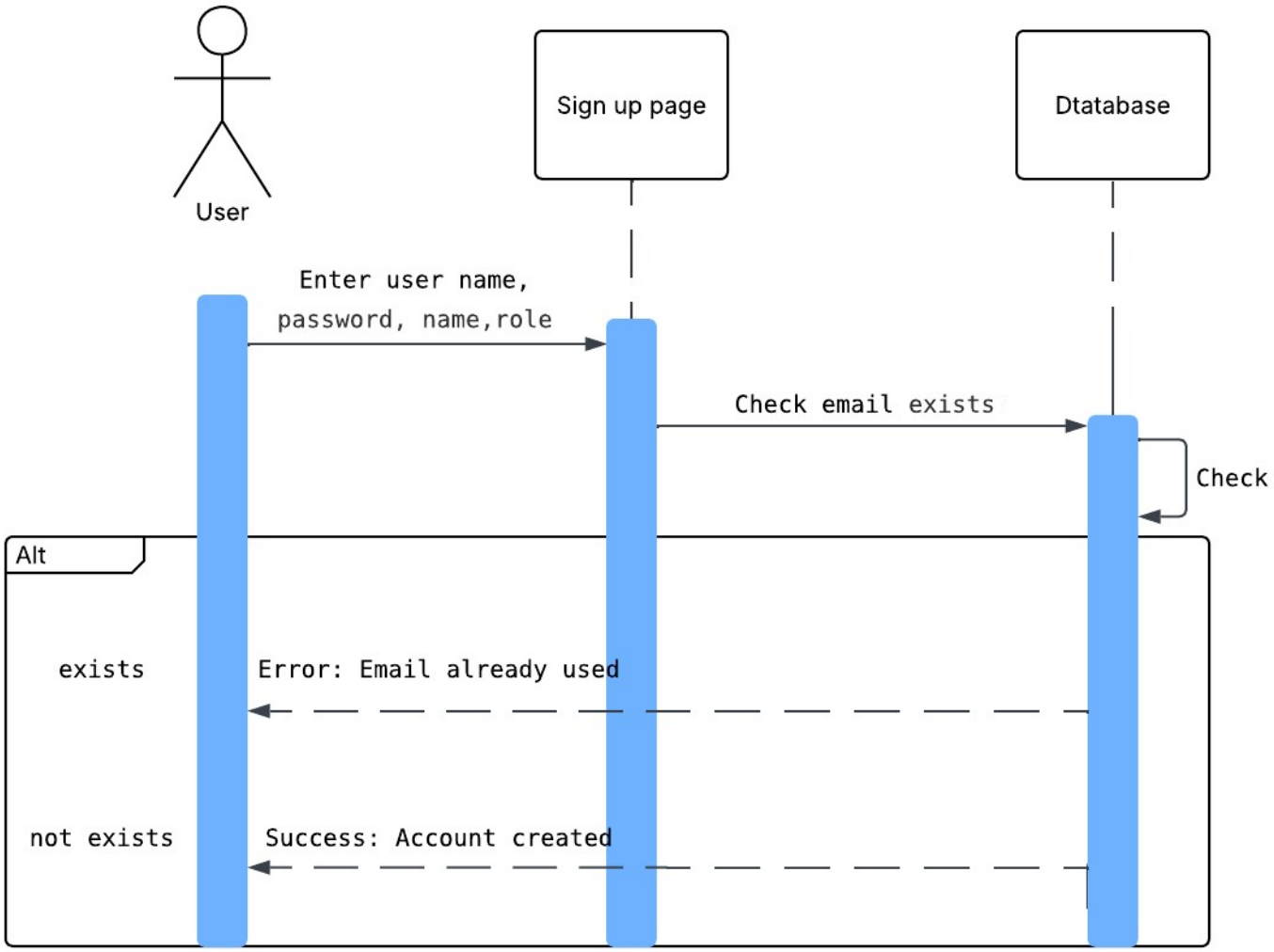
- Sorting should be fast.
- The client should understand why proposals are ordered that way.

Frequency of Use:

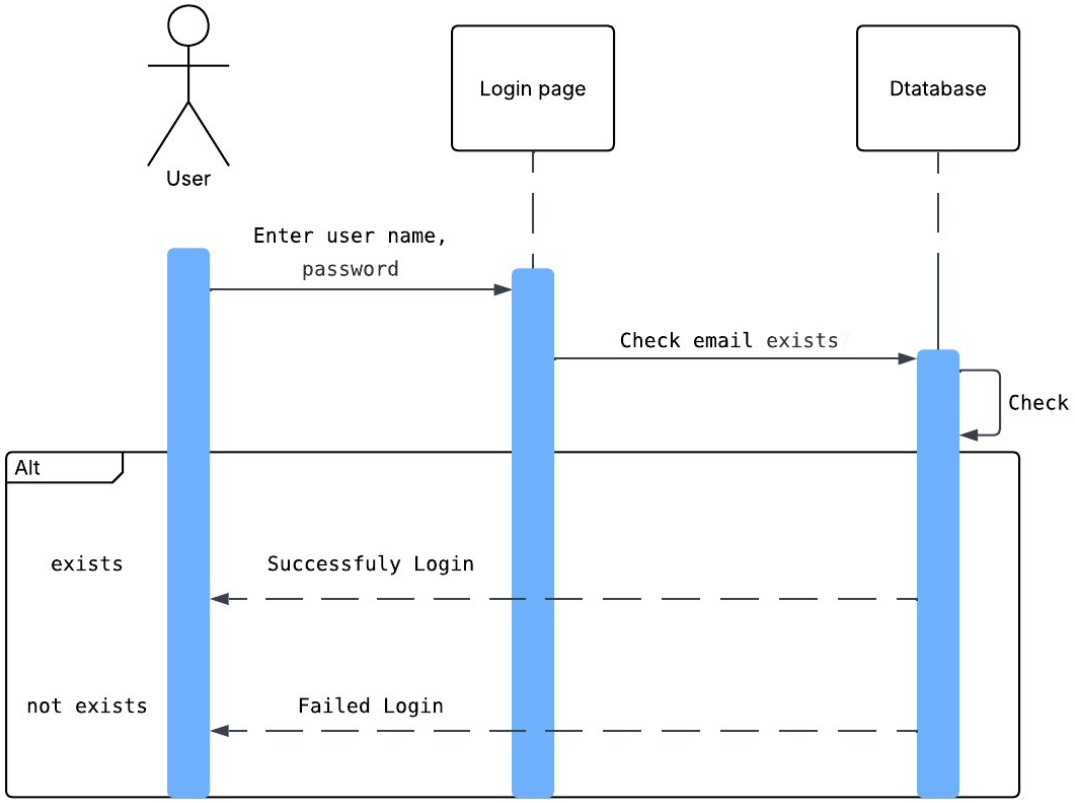
- Every time a client views their proposals.

5.4 Sequence Diagram

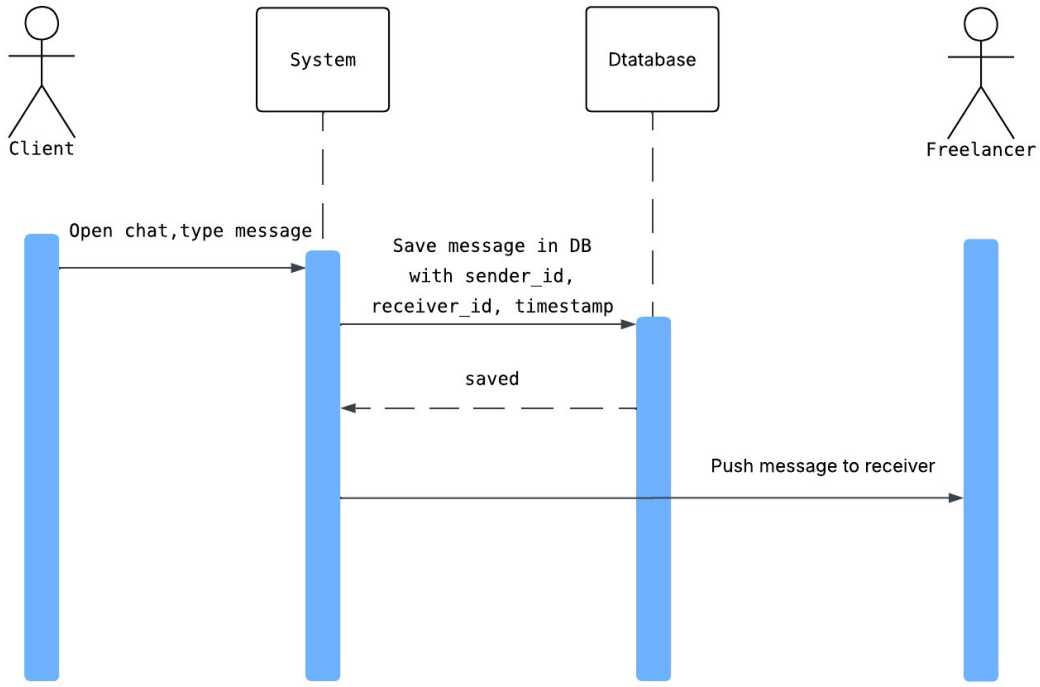
Sign up:



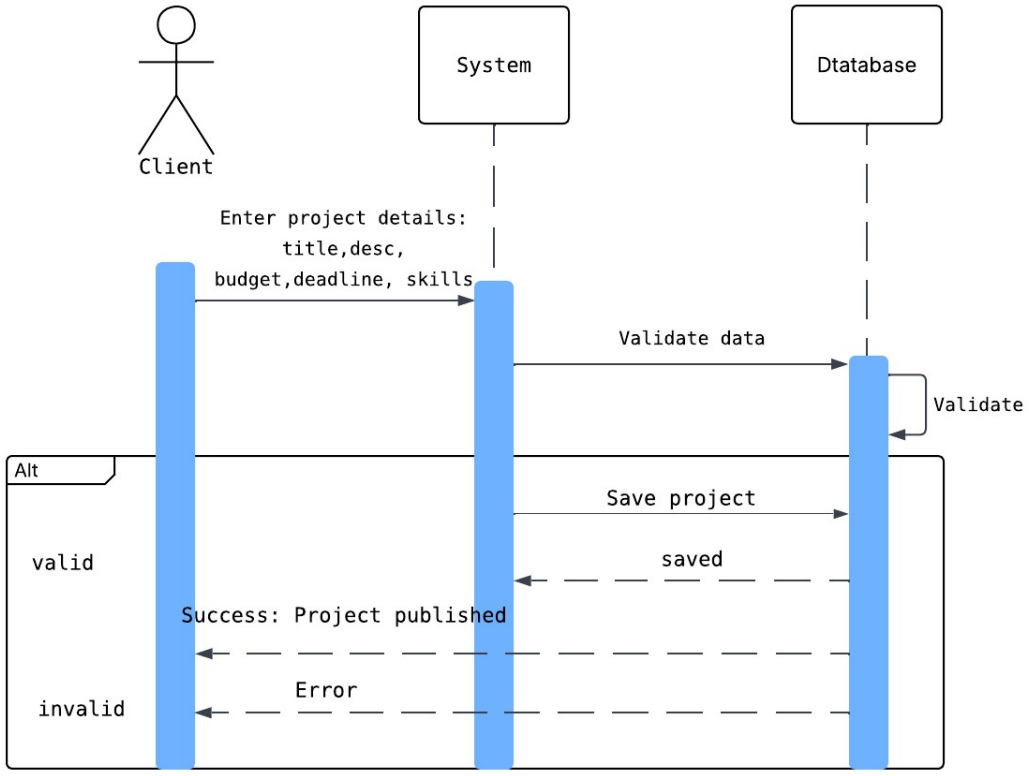
Login:


User
Login page
Dtatabase
Enter user name,
password
Check email exists
Check
Alt
exists
Successfuly Login
not exists
Failed Login

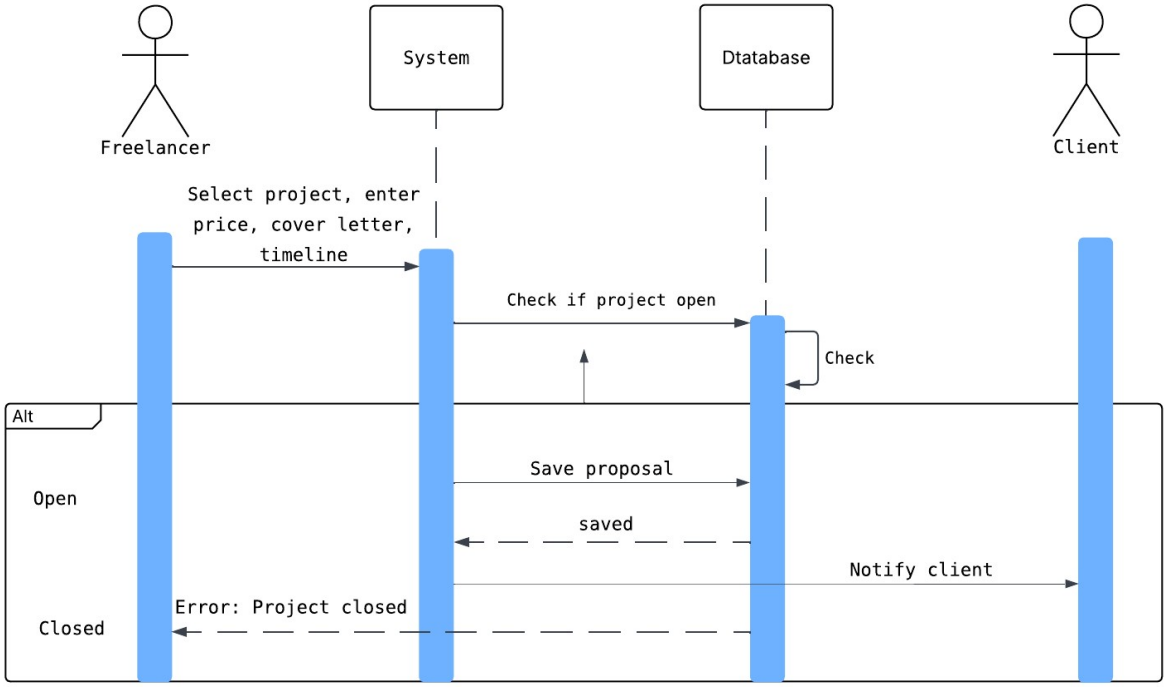
Chat:


Client
System
Dtatabase
Freelancer
Open chat,type message
Save message in DB
with sender_id,
receiver_id, timestamp
saved
Push message to receiver

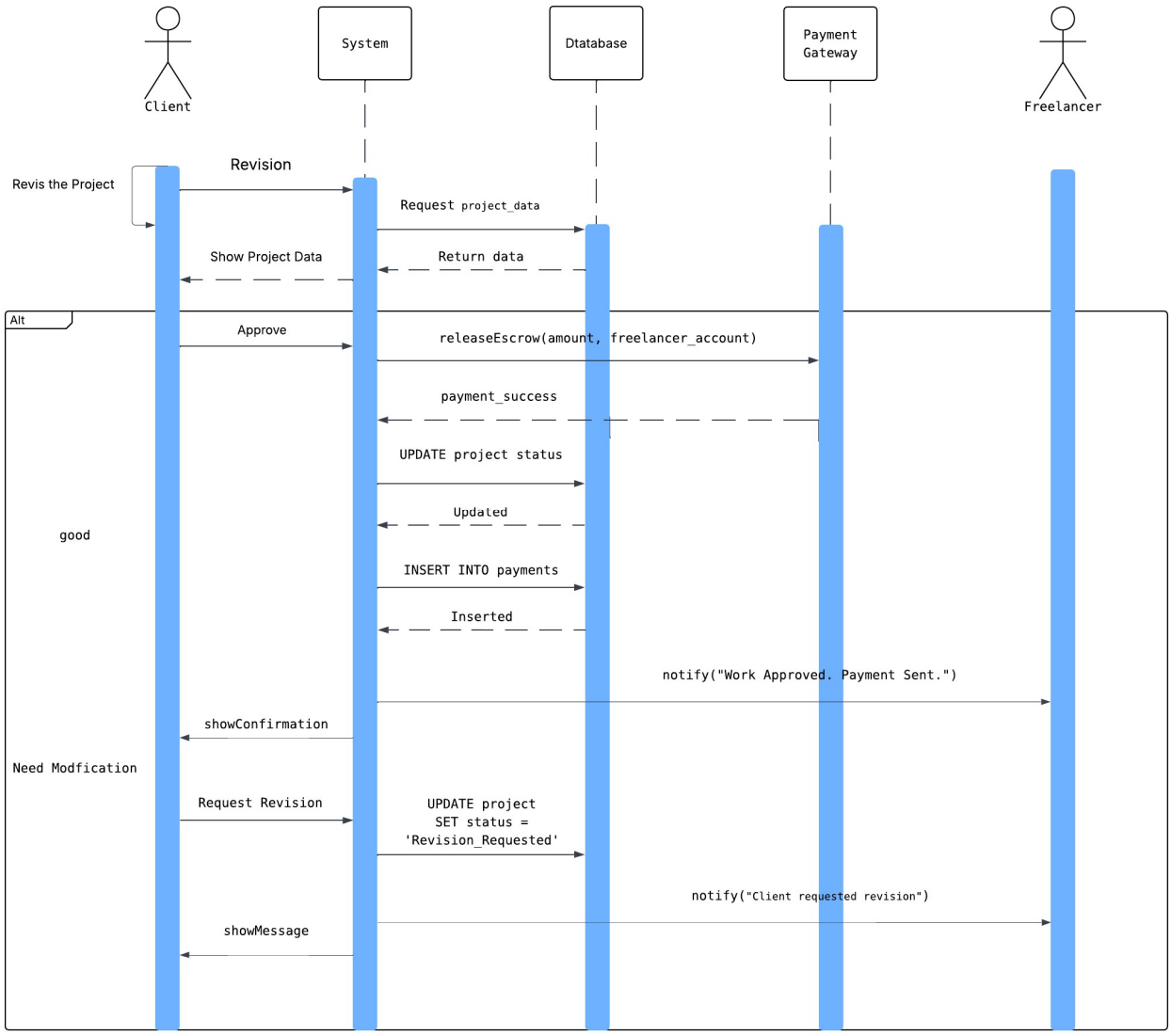
Post a Project



Submit Proposal:



Approve Work:

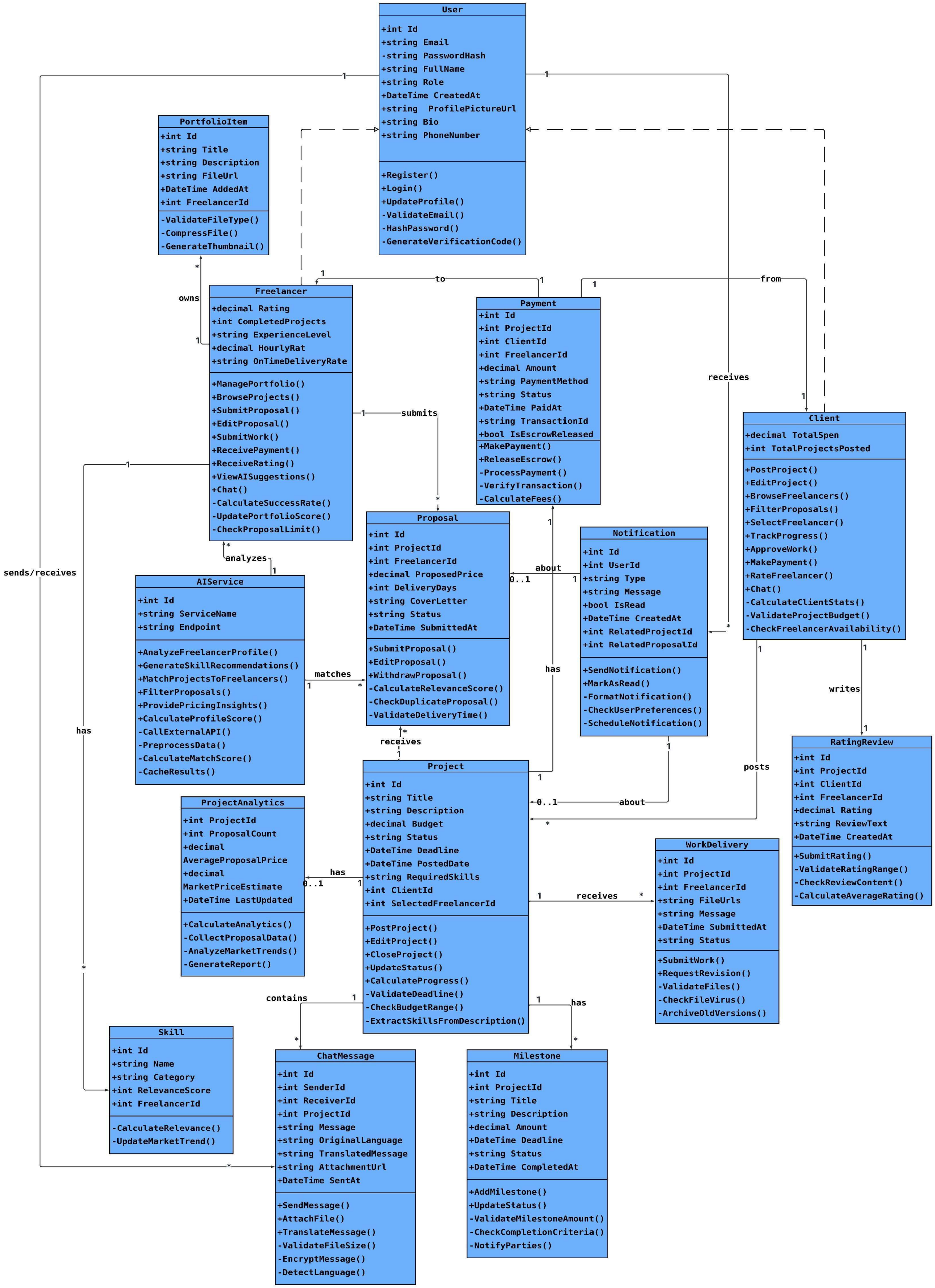


Filter Proposals (AI) ناقص

Analyze Freelancer Profile ناقص

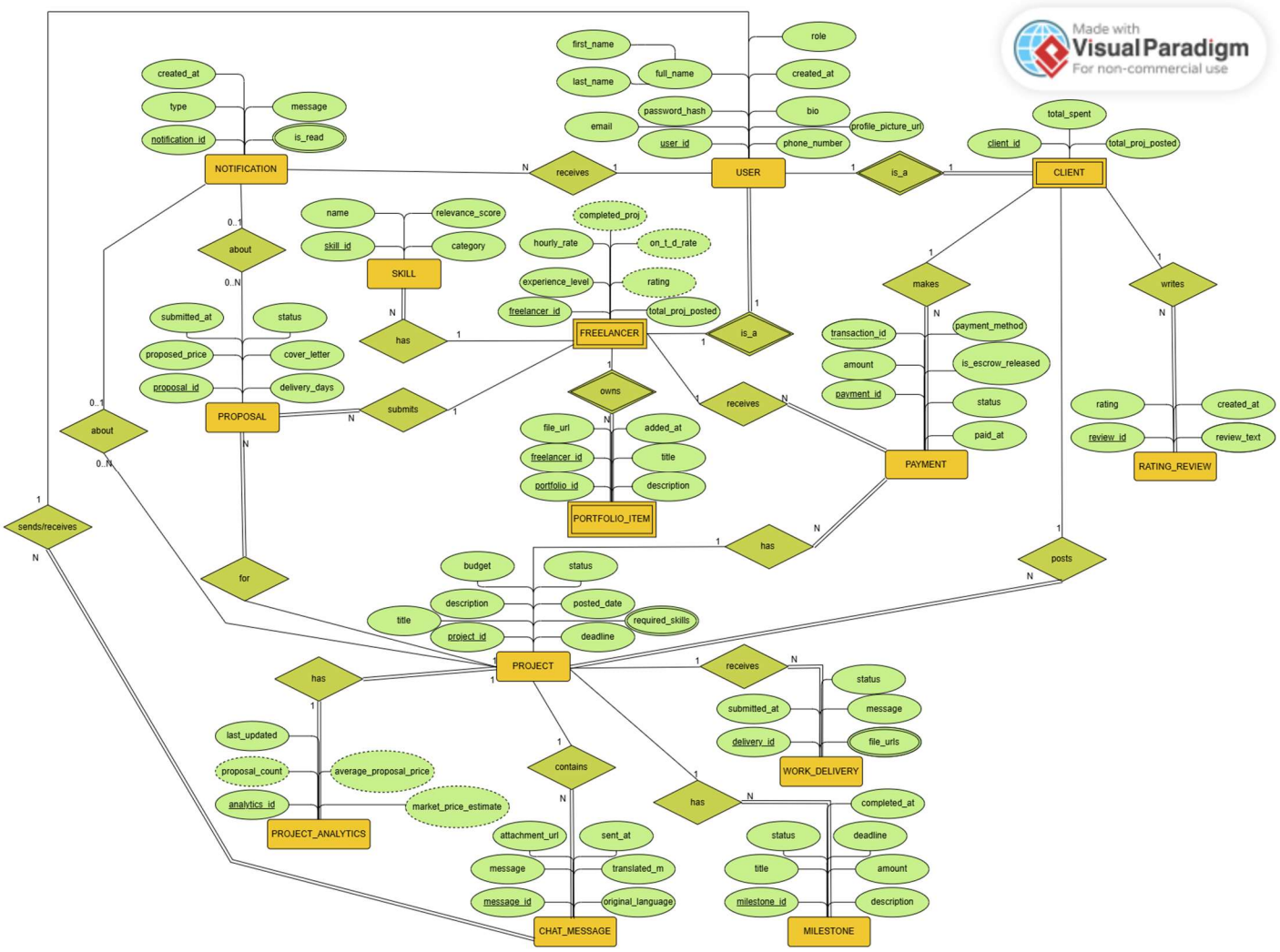
(AI) Match Projects to Freelancers (AI)ناقص

5.5 Class Diagram

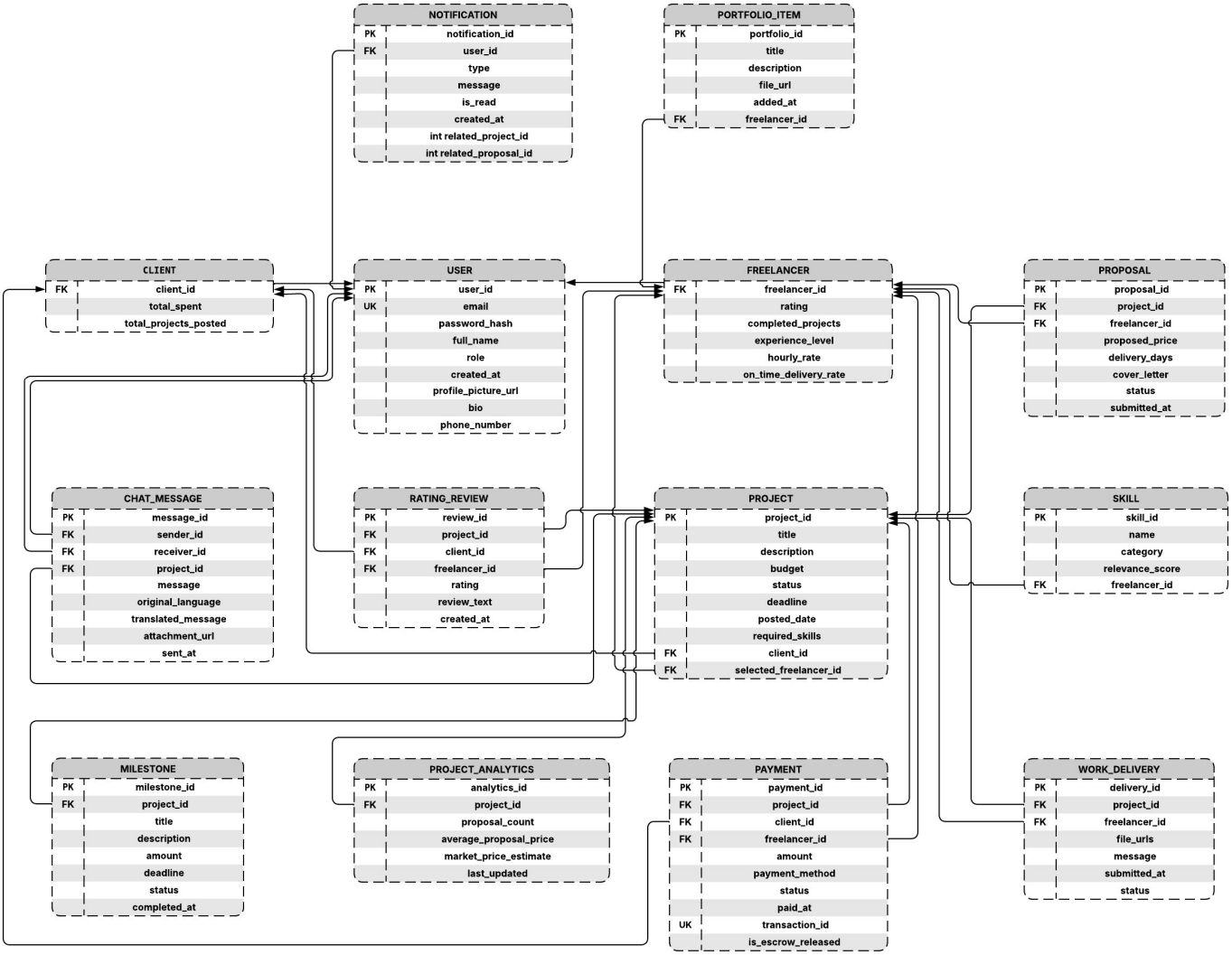

User
+int Id
+string Email
-string PasswordHash
+string FullName
+string Role
+DateTime CreatedAt
+string ProfilePictureUrl
+string Bio
+string PhoneNumber
+Register()
+Login()
+UpdateProfile()
-ValidateEmail()
-HashPassword()
-GenerateVerificationCode()
PortfolioItem
+int Id
+string Title
+string Description
+string FileUrl
+DateTime AddedAt
+int FreelancerId
-ValidateFileType()
-CompressFile()
-GenerateThumbnail()
owns
Freelancer
+decimal Rating
+int CompletedProjects
+string ExperienceLevel
+decimal HourlyRat
+string OnTimeDeliveryRate
+ManagePortfolio()
+BrowseProjects()
+SubmitProposal()
+EditProposal()
+SubmitWork()
+ReceivePayment()
+ReceiveRating()
+ViewAISuggestions()
+Chat()
-CalculateSuccessRate()
-UpdatePortfolioScore()
-CheckProposalLimit()
to
from
Payment
+int Id
+int ProjectId
+int ClientId
+int FreelancerId
+decimal Amount
+string PaymentMethod
+string Status
+DateTime PaidAt
+string TransactionId
+bool IsEscrowReleased
+MakePayment()
+ReleaseEscrow()
-ProcessPayment()
-VerifyTransaction()
-CalculateFees()
receives
Client
+decimal TotalSpen
+int TotalProjectsPosted
+PostProject()
+EditProject()
+BrowseFreelancers()
+FilterProposals()
+SelectFreelancer()
+TrackProgress()
+ApproveWork()
+MakePayment()
+RateFreelancer()
+Chat()
-CalculateClientStats()
-ValidateProjectBudget()
-CheckFreelancerAvailability()
submits
sends/receives
analyzes
Proposal
+int Id
+int ProjectId
+int FreelancerId
+decimal ProposedPrice
+int DeliveryDays
+string CoverLetter
+string Status
+DateTime SubmittedAt
+SubmitProposal()
+EditProposal()
+WithdrawProposal()
-CalculateRelevanceScore()
-CheckDuplicateProposal()
-ValidateDeliveryTime()
about
0..1
Notification
+int Id
+int UserId
+string Type
+string Message
+bool IsRead
+DateTime CreatedAt
+int RelatedProjectId
+int RelatedProposalId
+SendNotification()
+MarkAsRead()
-FormatNotification()
-CheckUserPreferences()
-ScheduleNotification()
AIService
+int Id
+string ServiceName
+string Endpoint
+AnalyzeFreelancerProfile()
+GenerateSkillRecommendations()
+MatchProjectsToFreelancers()
+FilterProposals()
+ProvidePricingInsights()
+CalculateProfileScore()
-CallExternalAPI()
-PreprocessData()
-CalculateMatchScore()
-CacheResults()
matches
has
writes
receives
Project
+int Id
+string Title
+string Description
+decimal Budget
+string Status
+DateTime Deadline
+DateTime PostedDate
+string RequiredSkills
+int ClientId
+int SelectedFreelancerId
+PostProject()
+EditProject()
+CloseProject()
+UpdateStatus()
+CalculateProgress()
-ValidateDeadline()
-CheckBudgetRange()
-ExtractSkillsFromDescription()
about
0..1
posts
RatingReview
+int Id
+int ProjectId
+int ClientId
+int FreelancerId
+decimal Rating
+string ReviewText
+DateTime CreatedAt
+SubmitRating()
-ValidateRatingRange()
-CheckReviewContent()
-CalculateAverageRating()
ProjectAnalytics
+int ProjectId
+int ProposalCount
+decimal AverageProposalPrice
+decimal MarketPriceEstimate
+DateTime LastUpdated
+CalculateAnalytics()
-CollectProposalData()
-AnalyzeMarketTrends()
-GenerateReport()
has
0..1
WorkDelivery
+int Id
+int ProjectId
+int FreelancerId
+string FileUrls
+string Message
+DateTime SubmittedAt
+string Status
+SubmitWork()
+RequestRevision()
-ValidateFiles()
-CheckFileVirus()
-ArchiveOldVersions()
receives
has
contains
has
Skill
+int Id
+string Name
+string Category
+int RelevanceScore
+int FreelancerId
-CalculateRelevance()
-UpdateMarketTrend()
ChatMessage
+int Id
+int SenderId
+int ReceiverId
+int ProjectId
+string Message
+string OriginalLanguage
+string TranslatedMessage
+string AttachmentUrl
+DateTime SentAt
+SendMessage()
+AttachFile()
+TranslateMessage()
-ValidateFileSize()
-EncryptMessage()
-DetectLanguage()
Milestone
+int Id
+int ProjectId
+string Title
+string Description
+decimal Amount
+DateTime Deadline
+string Status
+DateTime CompletedAt
+AddMilestone()
+UpdateStatus()
-ValidateMilestoneAmount()
-CheckCompletionCriteria()
-NotifyParties()

Chapter 6: Database Design

6.1 Entity Relationship Diagram



6.2 Schema


NOTIFICATION
PK notification_id
FK user_id
type
message
is_read
created_at
int related_project_id
int related_proposal_id
PORTFOLIO_ITEM
PK portfolio_id
title
description
file_url
added_at
FK freelancer_id
CLIENT
FK client_id
total_spent
total_projects_posted
USER
PK user_id
UK email
password_hash
full_name
role
created_at
profile_picture_url
bio
phone_number
FREELANCER
FK freelancer_id
rating
completed_projects
experience_level
hourly_rate
on_time_delivery_rate
PROPOSAL
PK proposal_id
FK project_id
FK freelancer_id
proposed_price
delivery_days
cover_letter
status
submitted_at
CHAT_MESSAGE
PK message_id
FK sender_id
FK receiver_id
FK project_id
message
original_language
translated_message
attachment_url
sent_at
RATING_REVIEW
PK review_id
FK project_id
FK client_id
FK freelancer_id
rating
review_text
created_at
PROJECT
PK project_id
title
description
budget
status
deadline
posted_date
required_skills
FK client_id
FK selected_freelancer_id
SKILL
PK skill_id
name
category
relevance_score
FK freelancer_id
MILESTONE
PK milestone_id
FK project_id
title
description
amount
deadline
status
completed_at
PROJECT_ANALYTICS
PK analytics_id
FK project_id
proposal_count
average_proposal_price
market_price_estimate
last_updated
PAYMENT
PK payment_id
FK project_id
FK client_id
FK freelancer_id
amount
payment_method
status
paid_at
UK transaction_id
is_escrow_released
WORK_DELIVERY
PK delivery_id
FK project_id
FK freelancer_id
file_urls
message
submitted_at
status

Chapter 7: Implementation

7.1 UI design

7.2 Front End Code

7.3 Back End Code

7.4 Machine Learning Model

Chapter 8: References