South Valley University
Faculty of Computers and Information

Graduation Project

Tradof

Submitted by

Ahmed Sabry Mahmoud

Mohamed Abdalrazek Abdo

Yasser Mohamed Abdel Hamid Abdel Eahim

Ahmed Nady Essa Mahamed

Ahmed Nasser Abdelhameed Mahmoud

Ahmed Kamal Youssef Mohamed

Youssef Ghareb Abbas Taha

Mohamed Hagag Mohamed Mostafa

Under Supervision

Dr. Nahla Fathy Ahmed

Dr. Randa Mohamed

Software Requirements Specification (SRS)

Team Members

Name	ID	Department
Ahmed Sabry Mahmoud	2021190035	Computer Science
Mohamed Abdalrazek Abdo	2021190164	Computer Science
Mohamed Hagag Mohamed	2021190027	Computer Science
Ahmed Nasser Abdelhameed	2021190037	Computer Science
Ahmed Nady Essa	2021190006	Computer Science
Yasser Mohamed Abdelhamid	2021190001	Computer Science
Ahmed Kamal Youssef	2021190165	Computer Science
Youssef Ghareb Abbas	2021190088	Computer Science

List of figures

Revision history

Date	Version	description	Author
25/12/2024	1.0	SRS 0.1	Tradof Team

1. Introduction

1.1 Document Purpose

The purpose of this document is to specify the functional and non-functional requirements of the project management system for translation services. It aims to be a reference for developers, testers, and stakeholders to understand the system's behavior and capabilities clearly.

1.2 Product Scope

The product is a web and mobile application designed to facilitate translation project management. It caters to companys, freelancers, and administrators, offering features such as user registration, project creation, file sharing, real-time chat, and payment processing. The system streamlines the translation process, enhances communication, and provides analytics for performance evaluation.

1.3 Intended Audience and Document Overview

This SRS document is intended to read by

Software Developers: Use the SRS as a guide for building the system's functionalities.

Project Managers: Oversee project progress and ensure alignment with the requirements.

companys: Request translation services and manage projects via the application.

Freelancers: Utilize the platform to receive and complete assigned tasks.

Admins: Manage user roles and oversee overall platform operations.

Quality Assurance (QA) Team: Design test plans and validate that the application meets specified requirements.

Technical Writers: Create user manuals and documentation based on the system’s features.

Overview: This document covers the functional requirements, use case diagrams, and detailed descriptions of the main components, including project management, user handling, payment system, and reporting features.

1.4 Definitions, Acronyms and Abbreviations

SRS: Software Requirement Specification

IEEE Format: Format designed by Institute of Electrical and Electronics Engineers for a software requirement document

OTP: One-Time Password

ID: Identifier

UI: User Interface

API: Application Programming Interface

QA: Quality Assurance

1.5 Document Conventions

This document will use IEEE format. For clarity, acronyms and technical jargon, deemed uncommon by the author, will be included in the glossary. The format for headings is as followed: Major headings are in bold 18 pt font, and concurrent headings in bold 14 pt. font.

1.6 References and Acknowledgments

Standard SRS templates and guidelines from IEEE standards for software documentation.

2. System Environment

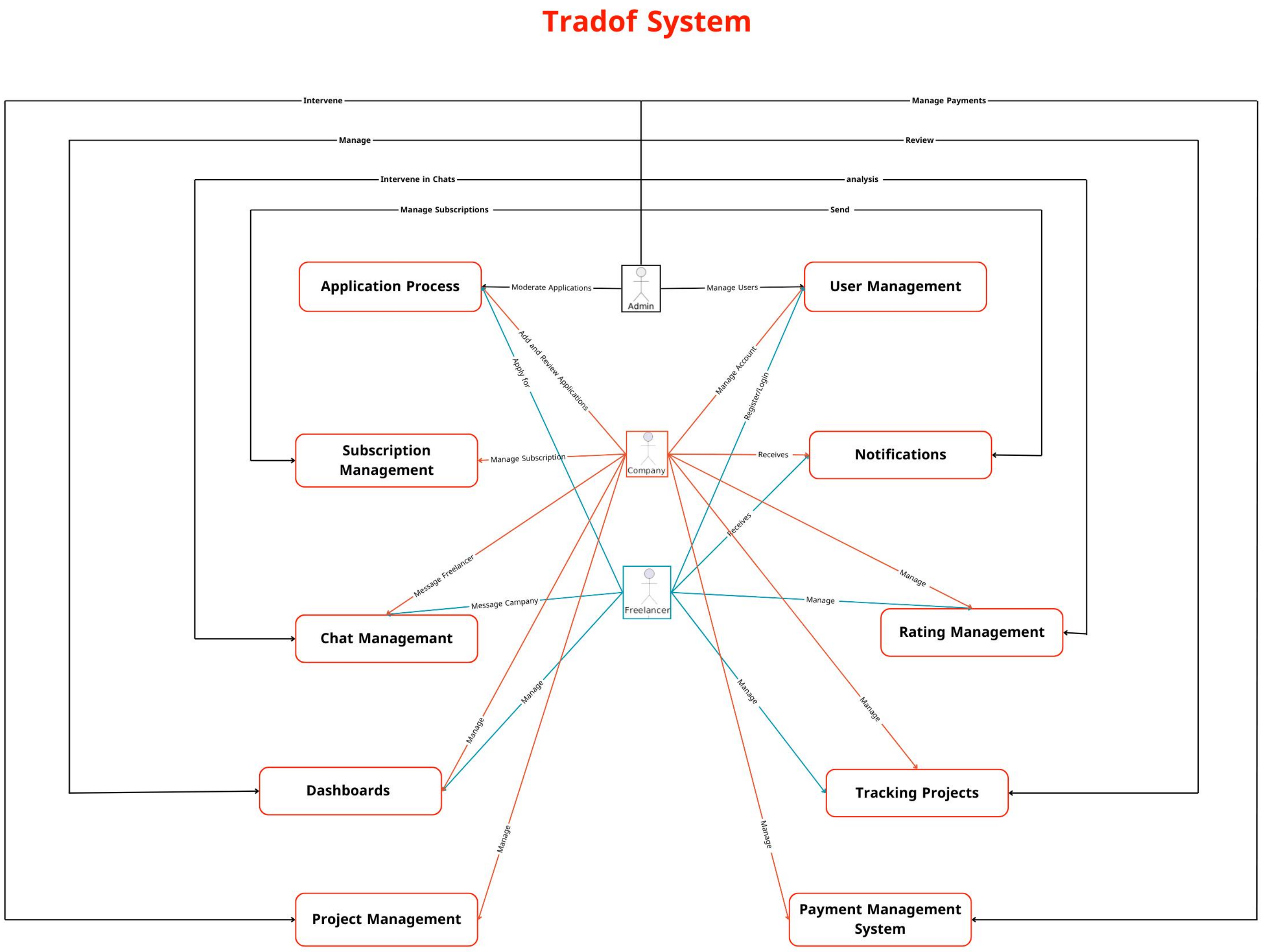


Figure 2.1 System Enviroment Diagram

3. Overall Description

3.1 Product Overview

The project management application is a centralized platform designed for companies and freelance translators to collaborate efficiently. Companies can create translation projects, assign freelancers, track progress, and manage payments, while freelancers can apply to projects, manage their tasks, track their earnings, and receive feedback. The platform includes dashboards and reports tailored to each user type—companies, freelancers, and administrators—facilitating real-time updates, transparent communications, and streamlined project management.

The system aims to:
- Simplify the translation project life-cycle, from initiation and assignment to completion and payment.

- Provide a transparent workspace for freelancers and companies to track tasks, deadlines, feedback, and performance.

- Ensure secure and efficient handling of project files, payments, and feedback.

- Enable platform administrators to monitor system health, user engagement, and financial activities.

3.2 Product Functionality

The key functions of the platform are as follows:

3.2.1 Project Management

Project Creation: Companies can create new projects with specifications (languages, skills), deadlines, and budgets.

Project Assignment: Projects are assigned to freelancers based on qualifications and availability.

Progress Tracking: Tracks real-time status and milestones of ongoing projects.

Quality Assurance & Approval: Allows for feedback and revisions before final approval.

Completion & Payment: Marks projects as complete and initiates payment processing.

3.2.2 User Management

Registration & Authentication: Users register, verify their identity, log in, and manage passwords.

Role Management: Admins assign roles to users (company/Company, Freelancer, Admin).

Profile Management: Users can view and edit their personal information.

Subscription Management

Plan Configuration: Admins configure subscription plans, setting features and usage limits.

Subscription Analytics: Provides insights into active subscriptions, revenue, and Compiny retention.

3.2.3 Dashboard & Reporting

Company Dashboard: Displays project summaries, payment status, and performance analytics.

Freelancer Dashboard: Summarizes active, completed, and pending projects, earnings, and feedback.

Admin Dashboard: Provides a platform-wide view of project statuses, user engagement, and revenue metrics.

Reports: Generates detailed reports on financials, project performance, freelancer ratings, and system health.

3.2.4 Payment Management

Invoice Generation: Creates invoices upon project completion and handles payments.

Refund Requests: Enables users to request refunds for unsatisfactory services.

3.2.5 Chat and Notifications

Messaging: Real-time chat for direct communication.

Notifications: Alerts users about new messages, project updates, and payment actions.

3.3 Design and Implementation Constraints

3.3.1 Web Application and Mobile Application

The system should be a web application and a mobile application.

3.3.2 Software Limitations

- The system must use both SQL and NoSQL databases.
- Frameworks to be used: Microsoft .NET Framework 6, Node.js v23.2.0, React.js v19, and Flutter.

3.3.3 Programming Language Limitation

- The C# language must be used for developing components related to the .
- NET Framework, utilizing Microsoft Visual Studio 2017.
- JavaScript must be used for developing components related to Node.js and React.js.
- Dart must be used for developing components related to Flutter.

3.3.4 Timing Constraints

System should be developed completely in 3 Months.

3.4 Assumptions and Dependencies

3.4.1 Assumptions:

- The code should be free with compilation errors/syntax errors.
- The product must have an interface which is simple enough to understand.

3.4.2 Dependencies:

- All necessary hardware and software are available for implementing and use of the tool.
- The proposed system would be designed, developed and implemented based on the software requirements specifications document.

4. System Features

F1: Project Management

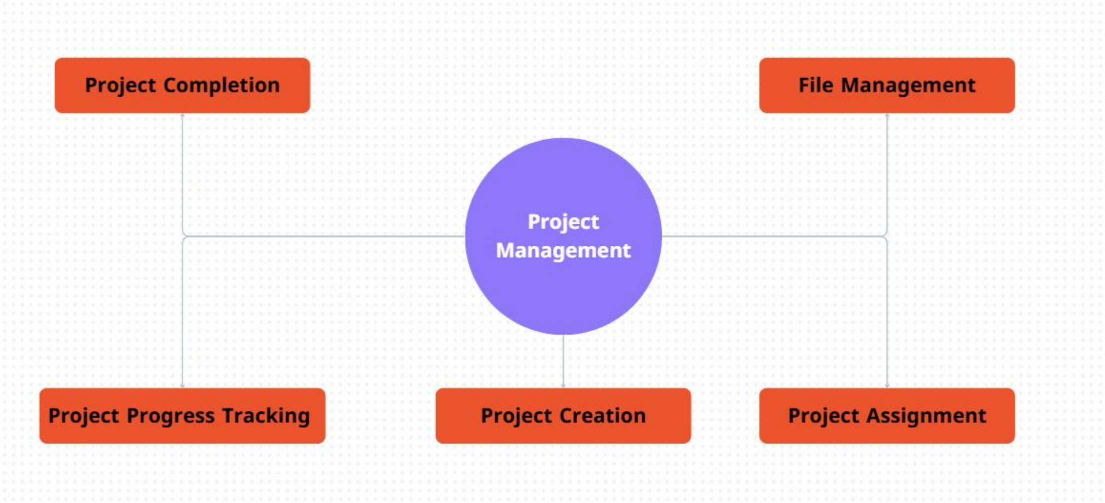


Figure 4.1 Project Management Context Diagram

1. Project Creation

Description

Allows Company to create a new project with detailed specifications, deadlines, and budget constraints for freelancers.

Input

Project title, description, deadline, budget, required Language pair , and Attachment files.

Output

Project ID, project details, and confirmation of successful creation.

2. Project Assignment

Description

Assigns projects to available freelancers based on their skillset and availability.

Input

Project ID, freelancer ID.

Output

Assignment confirmation, freelancer notifications.

3. File Management

Description

Manages project files, allowing users to upload, access, organize, and share files. Includes batch file upload for multiple files at once. **Batch File Upload:** Allows users to upload multiple files to a project in one action.

Input

Project ID, files (multiple file upload).

Output

List of uploaded files with details, file organization and sorting options, download and share links.

4. Project Progress Tracking

Description

Tracks the ongoing status of the project, milestones achieved.

Input

Project ID.

Output

Real-time project status, progress percentage.

5. Project Completion

Description

Marks a project as complete, generates a payment summary, and processes payment to freelancers.

Input

Project ID, completion confirmation, payment information.

Output

Payment summary, payment confirmation, and project closure notification.

F2: User Management

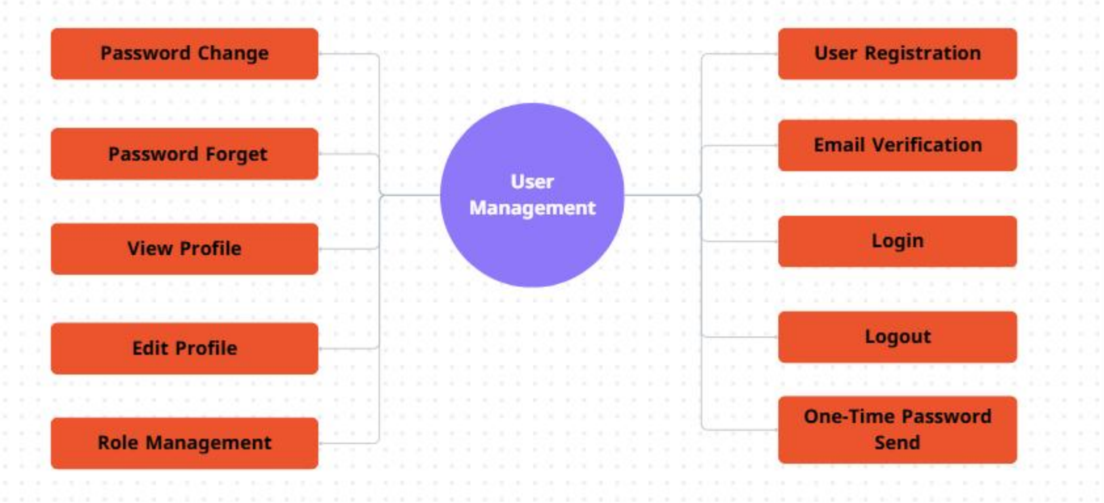


Figure 4.2 User Management Context Diagram

1. User Registration

Description

Registers a new user in the system with necessary details and assigns a role.

Input :

- Role: Role type (Super Admin,Admin, Company Employee ,Company Admin, Freelancer)
- FirstName
- LastName
- PhoneNumber
- Username
- Email
- Password
- Confirm Password
- Details data based Role
  - If it was a role freelancer
    - Gender
    - Work Experience

        - ◆ Social Media
        - ◆ Specialization
        - ◆ Language Pair
    - ■ If it was a role Company
        - ◆ Languages
        - ◆ Job Title
        - ◆ Created Date
        - ◆ Country
        - ◆ Specialization
        - ◆ Company User Type

Output :

Confirmation message indicating successful registration or error.

2. Email Verification

Description

Verifies the email of a newly registered user to confirm authenticity.

Input :

Email: Identifier of the user.

Output :

Message indicating email verification success or error.

3. Login

Description

Authenticates a user based on email and password, initiating a session.

Input :

Email: Identifier for the user.

Password: Password to verify the user.

Output :

Confirmation of successful login and session token, or an error message.

4. Logout

Description

Ends the user's session and logs them out of the system.

Input :

UserID: Identifier for the user.

Output :

Confirmation message indicating successful logout.

5. One-Time Password Send

Description

Allows users to receives the OTP for Change or Reset Password.

Input

Email: Email associated with the user's accoun.

Output

The OTP Messgae in Email are Sent consists of 6 Numbers.

6. Password Change

Description

Allows users to Change their password if has CurrentPassowrd.

Input

- CurrentPassword
- NewPassword
- ConfiremNewPassword

Output

Confirmation of successful Password Changed, or an error message .

7. Password Forget

Description

Allows users to reset their password if forgotten or compromised.

Input

Email and NewPassword and OTP.

Output

Confirmation of successful Password Reset, or an error message like OTP Not Correct.

8. View Profile

Description

Retrieves and displays a user’s profile information

Input

UserID: Identifier for the user.

Output

User profile details, including name, role, Phone, etc.

9. Edit Profile

Description

Allows a user to edit their profile information.

Input

- UserID: Identifier for the user.
- ProfileData: Data containing fields the user wants to update (e.g., name, phone, address).

Output

Confirmation message indicating success or error.

10. Role Management

Description

For Admin   Assigns or updates a user's role to define their permissions.

Input

- UserID: Identifier for the user.
- NewRole: New role to be assigned (Freelancer, company).

Output

Confirmation message indicating role assignment success or error.

F3: Admin Subscription Management

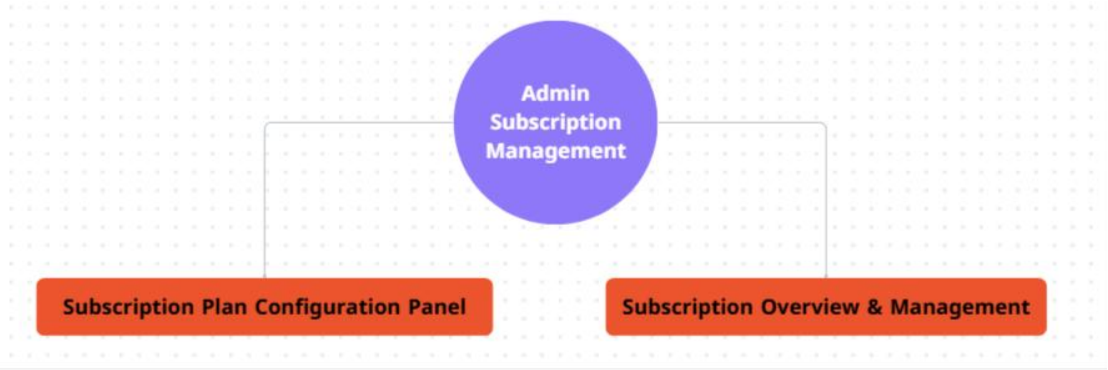

Admin Subscription Management
Subscription Plan Configuration Panel
Subscription Overview & Management

Figure 4.3 Admin Subscription Management Context Diagram

1. Subscription Plan Configuration Panel

Description : Allows admins to create, edit, and remove subscription plans with customization features and usage limits.

Input: Plan name, Features, Price, Duration, Usage limits.

Output: Subscription plan ID, confirmation of plan creation or update, and status (active/inactive).

2. Subscription Overview & Management

Description: Provides a comprehensive view of company subscriptions with options to manage and override automation when necessary.

Input: Company ID (optional for specific searches), Subscription filters (e.g., active, expired).

Output: Filtered list of subscriptions, details on individual plans, override actions for activation or renewal.

F4: Apply Process

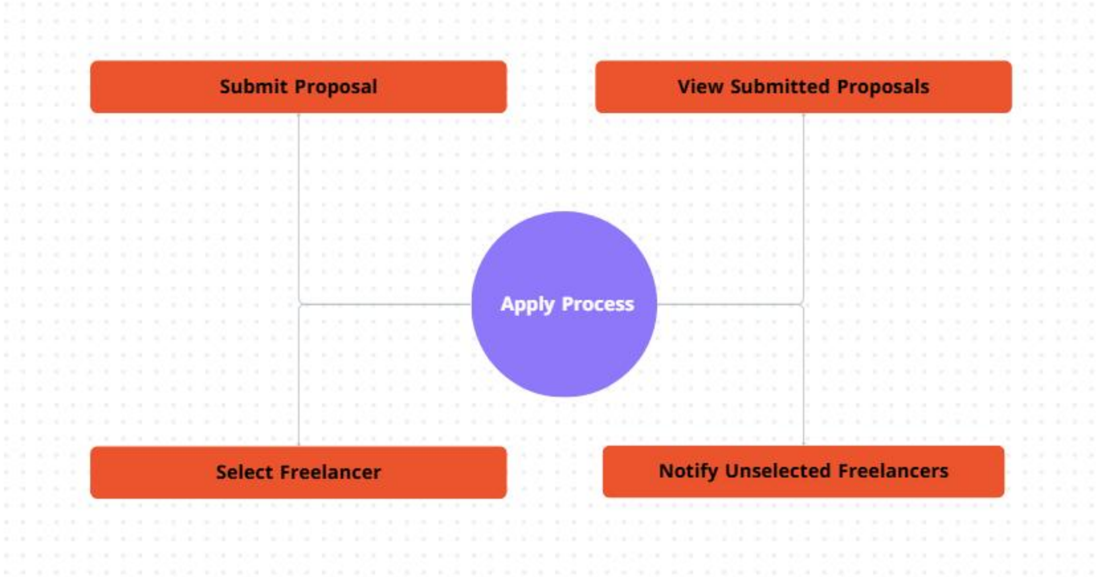


Figure 4.4 Apply Process Context Diagram

1. Submit Proposal

Description: Freelancers submit a proposal for a translation project, including details such as pricing, time estimates, and any additional notes.

Input:

- Freelancer ID
- Project ID

- Proposal details (price, estimated completion time, additional comments)

Output:

- ✓ Confirmation of proposal submission
- ✓ Updated project status to "Proposal Submitted"

2. View Submitted Proposals

Description: The company can view all the submitted proposals from freelancers interested in the project, comparing details like price, time estimates, and ratings.

Input:

- Company ID
- Project ID

Output:

- ✓ List of submitted proposals with freelancer details, prices, and timelines

3. Select Freelancer

Description: The company selects the most suitable freelancer for their project based on the submitted proposals, finalizing the selection process.

Input:

- Company ID
- Project ID
- Selected Freelancer ID

Output:

- ✓ Confirmation of freelancer assignment
- ✓ Project status updated to "In Progress"
- ✓ Notification to the selected freelancer

4. Notify Unselected Freelancers

Description: Automatically notify all freelancers who were not selected for the project, thanking them for their interest and informing them that the project has been assigned to someone else.

Input:

- Project ID
- List of Freelancer IDs (unselected)

Output:

- ✓ Email or in-app notification sent to unselected freelancers

F5: Dashboards

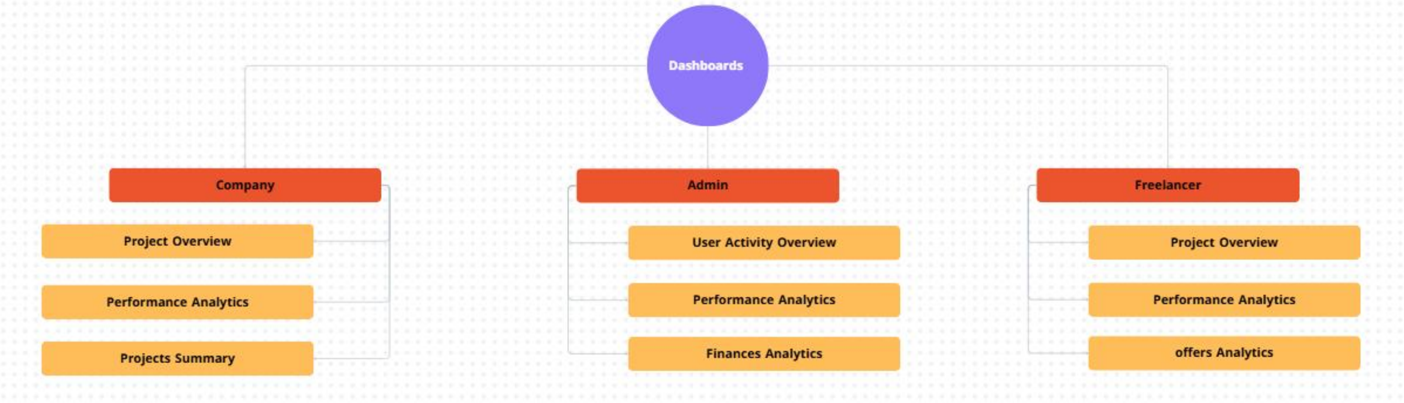


Figure 4.5 Dashboard Context Diagram

1. company

1. Project Overview

Description: Displays a detailed view of all ongoing, pending, and completed project.

Inputs

- Project ID : Identifier for the Project.

Output:

- ✓ Projects Detailed
- ✓ Freelance Detailed
- ✓ Chatting with Freelancer
- ✓ workspace

2. Performance Analytics

Description: Provides analytics on counter of projects in month.

Inputs:

- Company ID: Identifier for the company.

Output:

- ✓ Graphs and charts displaying projects in month.

3. Projects Summary

Description : Provides an overview of all start and upcoming projects.

Inputs:

- Company ID: Identifier for the company.

Output:

- ✓ list of all start and upcoming projects.

2. Admin

1. User Activity Overview

Description: Shows user activity levels, including companies and freelancers engaged on the platform.

Input:

None (data pulled from the database)

Output:

- ✓ Total number of active companies and freelancers.
- ✓ Engagement metrics (e.g., logins, actions taken).
- ✓ Comparison of activity levels over time.
- ✓ Visual representation of revenue trends over time.

2. Performance Analytics

Description: You will receive statistics such as the number of projects, the number of freelancers, the number of partnerships, and other statistics.

Input:

- None (data pulled from the database)

Output:

- ✓ Graphs and charts displaying all of statistics.

2. Finances Analytics

Description: View project financial statistics.

Input:

- ● None (data pulled from the database)

Output:

- ✓ Mind maps to display financial statistics, subscriptions and their details.

3. Freelancer

1. Project Overview

Description: Displays a detailed view of all ongoing, pending, and completed project.

Input

- ● Project ID : Identifier for the Project.

Output

- ✓ Projects Detailed
- ✓ Project Card
- ✓ Owner Detailed
- ✓ Chatting with Owner
- ✓ Workspace

2. Performance Analytics

Description: Provides analytics on status of projects .

Input:

- ● Freelancer ID: Identifier for the Freelancer.

Output:

- ✓ Graphs and charts displaying status of projects .

3. offers Analytics

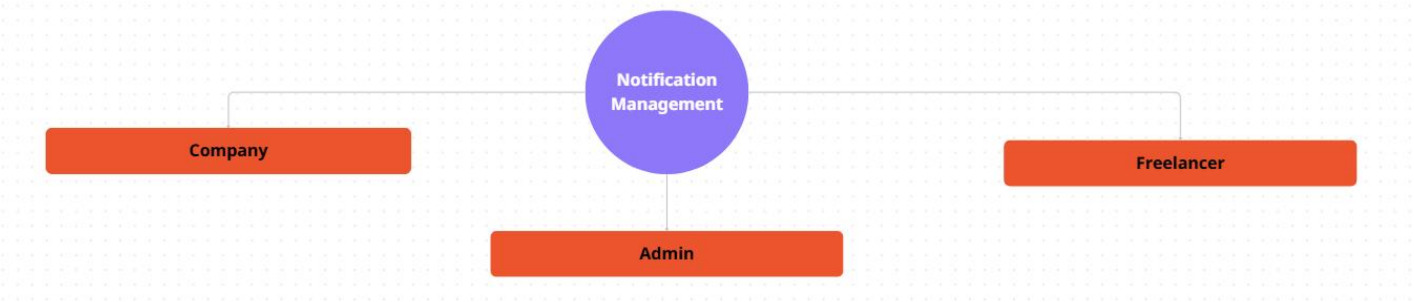
Description: Provides analytics on offers of projects in month .

Inputs:

- Freelancer ID: Identifier for the Freelancer.

Output:

✓ Graphs and charts displaying offers of projects in month.



F6: Notification Management

Figure 4.6 Notification Management Context Diagram

1. Freelancer Notification

Description: Show all alerts regarding technical support, offers provided on projects, a message from the project owner, and a message regarding delivery or tasks.

Inputs:

- Freelancer ID: Identifier for the Freelancer.

Output:

✓ View details Alerts

2. Admin Notification

Description: Show all technical support alerts and monthly reports.

Inputs:

No Data

Output:

✓ View details Alerts

3. company Notification

Description: Show all alerts related to technical support, project offers, message from freelancer owner, message regarding delivery or tasks and monthly reports.

Inputs:

- company ID: Identifier for the company.

Output:

✓ View details Alerts

F7: Tracking Projects

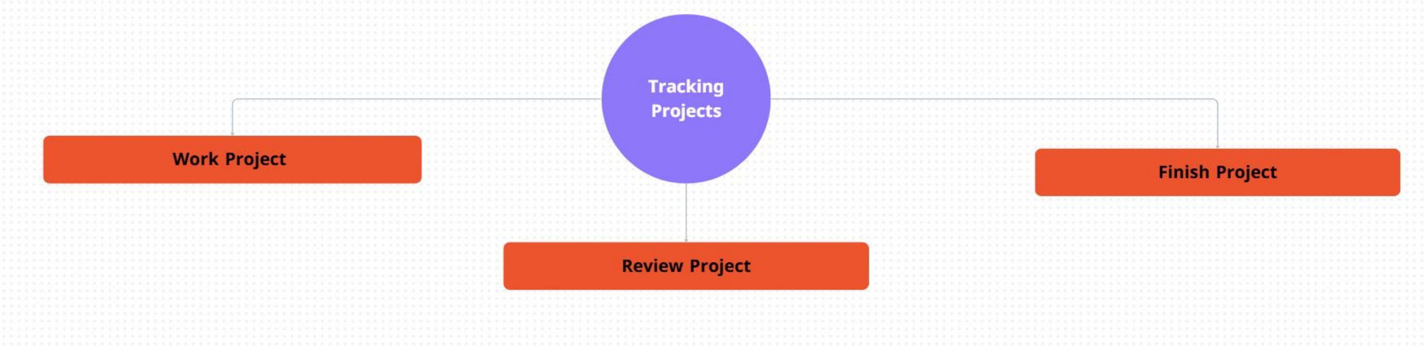


Figure 4.7 Tracking Projects Context Diagram

1. Work Project

Description: After the freelancer selects the appropriate offer, the project enters the work or planning phase.

Inputs:

- Project ID
- Freelancer ID

Output:

✓ Show working stage

2. Review Project

Description: The freelancer requests a review of his work on the project by the company and the company begins to review his work.

Inputs:

- Project ID
- Freelancer ID

Output:

✓ Work review phase completed

3. Finish Project

Description: After reviewing the work, the company moves to the project completion stage..

Inputs:

- Project ID
- Freelancer ID

Output:

✓ The project is completed and the project funds are transferred to the freelancer.

F8: Rating System

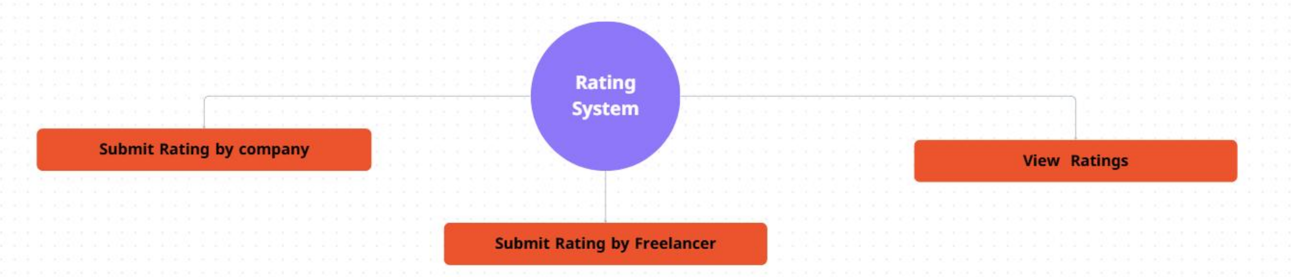


Figure 4.5 Rating System Context Diagram

1. Submit Rating by company

Description: After a project is completed, the user can submit a rating (1-5 stars) and a review for the freelancer.

Input:

- Company ID: Identifier for the user submitting the rating.

- Freelancer ID: Identifier for the freelancer receiving the rating.
- Rating: Numeric value (1-5 stars).
- Review: Written feedback from the user.

Output:

- ✓ Confirmation message indicating success or error.

2. Submit Rating by Freelancer

Description: After the project is completed, the freelancer can submit a rating for the user based on their experience.

Inputs:

- Freelancer ID: Identifier for the freelancer submitting the rating.
- Company ID: Identifier for the user receiving the rating.
- Rating: Numeric value (1-5 stars).
- Review: Written feedback from the freelancer.

Output:

- ✓ Confirmation message indicating success or error.

3. View Ratings

Description: Displays the average rating and all reviews for a specific freelancer or specific company .

Inputs:

- Freelancer ID or Company ID .

Output: A list of ratings and reviews, along with an average rating.

F9: Chat Management

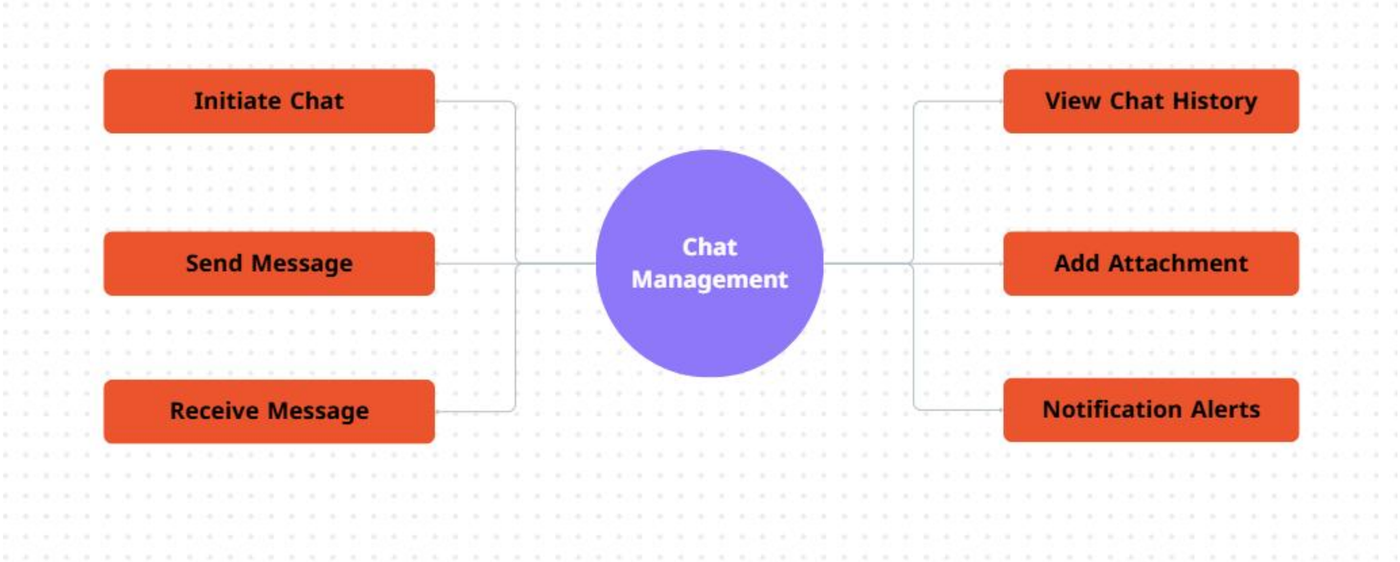


Figure 4.9 Chat Management Context Diagram

1. Initiate Chat

Description: Allows a user to start a conversation with another user based on their roles (e.g., freelancer, project manager).

Input: Sender ID, Receiver ID, Message Content

Output: Chat session initiated with confirmation message.

2. Send Message

Description: Enables a user to send a message within an active chat session.

Input: Sender ID, Message Content, Receiver ID

Output: Message sent successfully, notification status.

3. Receive Message

Description: Receives and displays a message in the user’s chat window in real-time.

Input: Receiver ID, Sender ID

Output: Message displayed in chat with timestamp.

4. View Chat History

Description: Allows users to view past messages within a chat session.

Input: Receiver ID, Sender ID

Output: Display of chat history with timestamps.

5. Add Attachment

Description: Allows users to attach and send files (e.g., PDFs, images) in the chat.

Input: Receiver ID, Sender ID, File Attachment

Output: File uploaded and sent in chat.

6. Notification Alerts

Description: Sends a notification when a new message arrives .

Input: Receiver ID, Sender ID, Notification Type

Output: Notification sent to the user's device.

F10: Payment Management System

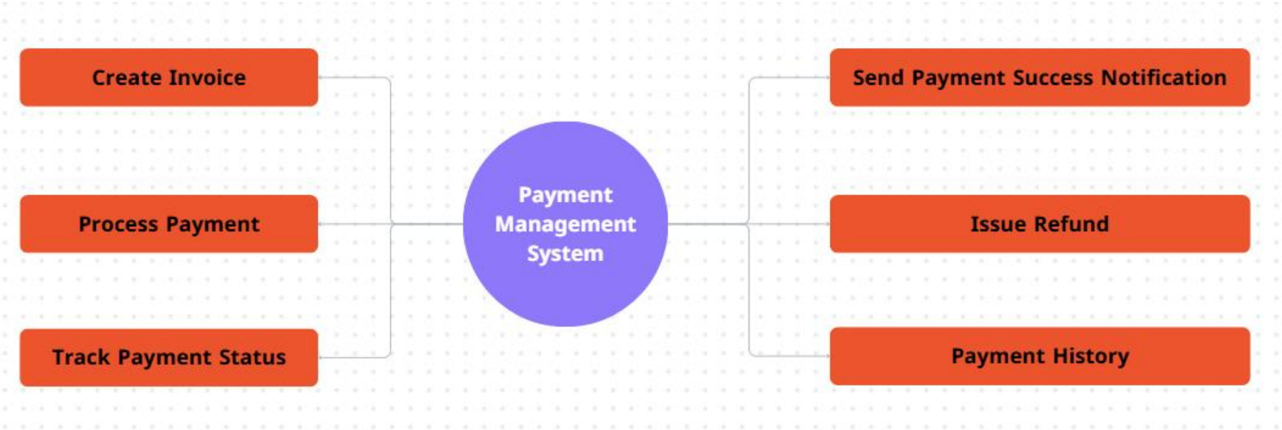


Figure 4.10 Payment Management Context Diagram

1. Create Invoice

Description: Automatically generates an invoice for a completed project, detailing the amount to be paid by the user to the freelancer.

Inputs:

- Project ID: Identifier for the completed project.
- Freelancer ID: Identifier for the freelancer who completed the project.
- Total Amount: The total amount the user needs to pay.

Output

- ✓ A generated invoice in a downloadable format (e.g., PDF).

2. Process Payment

Description: Handles the transaction of funds from the user to the freelancer via a secure payment gateway.

Inputs:

- Invoice ID: Identifier of the invoice.
- Payment Method: Selected payment method (e.g., credit card, PayPal).
- Amount: The payment amount.

Output: Confirmation of a successful or failed payment transaction.

3. Track Payment Status

Description : Tracks whether the payment has been made, is pending, or has failed.

Inputs :

- Invoice ID: Identifier for the invoice being tracked.

Output : Current status of the payment (Paid, Pending, Failed).

4. Send Payment Success Notification

Description: Sends a notification to both the freelancer and user after a payment is successfully processed.

Inputs:

- Invoice ID: Identifier of the related invoice.
- User ID and Freelancer ID: Identifiers for the recipients.

Output: Notification sent to both parties confirming successful payment.

5. Issue Refund

Description: Allows users to request refunds if the service provided does not meet expectations.

Inputs:

- Project ID: Identifier for the project.
- Refund Reason: Explanation for why a refund is being requested.

Output: Confirmation of refund status (Approved, Denied).

6. Payment History

Description : Displays a detailed history of all financial transactions for both users and freelancers.

Inputs :

- User ID or Freelancer ID: Identifier of the user or freelancer.

Output : List of past payments, including amounts, dates, and statuses.

5. Other Nonfunctional Requirements

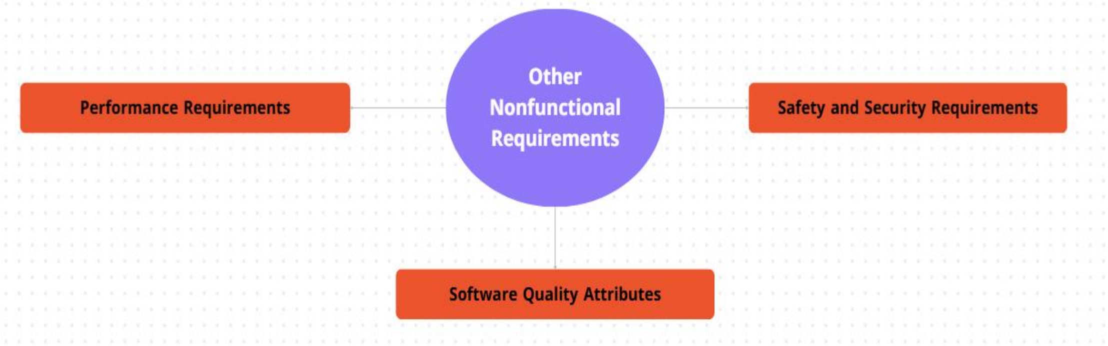

Performance Requirements
Other Nonfunctional Requirements
Safety and Security Requirements
Software Quality Attributes

Figure 5.1 Other Non-functional Diagram

1. Performance Requirements

1. System Response Time: The Tradof platform should respond to user actions (e.g., page loading, form submission, project creation) within 2 seconds under normal operating conditions.

2. Scalability: The system should be able to handle an increase in concurrent users, supporting at least 500 users without performance degradation.

3. Data Processing Speed: Importing or exporting data (e.g., project files, company databases) should complete within 5 seconds for files up to 50 MB.

4. Database Performance: Queries and database transactions must complete within 1 second, ensuring that search and filter operations on projects, companys, and freelancers are efficient.

2. **Safety and Security Requirements**

1. User Authentication and Authorization: Tradof must use a secure authentication mechanism (e.g., OAuth 2.0, JWT) to verify users and ensure that data access permissions are strictly enforced based on user roles (e.g., admin, project manager, company).

2. Data Encryption: All sensitive data, including company information, project details, and financial transactions, must be encrypted both at rest (AES-256) and in transit (TLS 1.2 or higher).

3. Data Backup and Recovery: Automated daily backups of all critical data must be stored securely off-site. In case of system failure, a full data recovery must be achievable within 1 hour.

4. Audit Logging: All user actions related to data modifications, access to sensitive information, and system errors should be logged. These logs should be accessible only to administrators and stored for at least one year.

5. Account Locking and Intrusion Detection: After five failed login attempts, the user account should be temporarily locked, and an alert sent to the account owner's registered email. Unusual access patterns (e.g., frequent logins from different IPs) should trigger alerts to the security team.

3. Software Quality Attributes

1. Usability: The platform should have an intuitive interface with clear navigation, ensuring that new users can perform essential actions (e.g., project setup, freelancer assignment) without extensive training.

2. Reliability: The system must have an uptime of at least 99.9% to ensure continuous access for users, especially in high-traffic periods or critical deadlines.

3. Maintainability: The codebase should be modular and well-documented to allow efficient updates, bug fixes, and the addition of new features without disrupting existing functionality.

4. Extensibility: The architecture should support the easy addition of new modules or integrations (e.g., third-party translation tools or payment gateways) with minimal refactoring.

5. Interoperability: Tradof should support integrations with external tools commonly used in translation and project management, such as CAT tools, accounting systems, and CRM platforms, through APIs.

6. Accessibility: The platform should comply with WCAG 2.1 Level AA standards to ensure accessibility for users with disabilities, including screen reader compatibility and keyboard navigation.

6. Oher Services

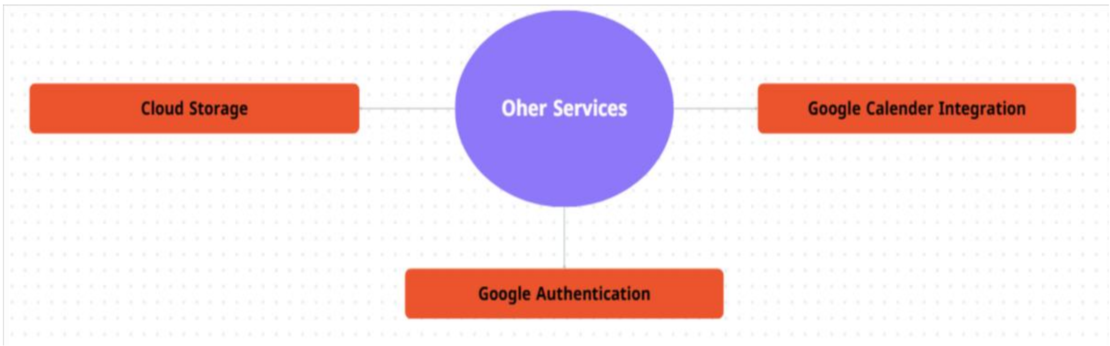


Figure 6.1 Other Services Diagram

1. Cloud Storage

Cloud storage is an essential service that enables users to securely store, manage, and access data over the internet. It offers scalability, allowing storage capacity to grow with user needs.

Key benefits:

- **Accessibility:** Access files from anywhere on any device with an internet connection.
- **Security:** Advanced encryption and secure login mechanisms protect user data.
- **Collaboration:** Facilitates real-time collaboration by sharing files with teams or Companys.
- **Cost-effectiveness:** Pay-as-you-go pricing eliminates the need for large upfront investments.

2. Google Calender Integration

Google Calendar Integration allows applications to interact seamlessly with Google Calendar, providing users with advanced scheduling and event management features. By leveraging Google Calendar APIs, apps can:

- **Create and manage events:** Schedule meetings, set reminders, and invite attendees.
- **Real-time synchronization:** Ensure updates are reflected across devices and platforms.
- **Notifications:** Send timely alerts for upcoming events or changes to schedules.
- **Custom workflows:** Automate scheduling based on specific triggers or actions within the app.

7. External Interface Requirements (User interface)

1. User Interfaces

1. **Login Screen:**

    1. A simple screen that includes:

        1. Fields to input username and password.
        2. A "Login" button.
        3. A link for "Create a New Account" for both translators and companies.
        4. Error messages displayed in case of incorrect input.

    2. **Note:**

        1. The available screens and options may vary depending on the user's role (translator or company).

2. company Interface:

1. New Translation Project Form:

1. Text fields for entering project details:

    1. Project name.
    Source language and target language.
    Content field (Legal, Medical, Marketing, etc.).
    Deadline.

2. **Auto-generated field:** A unique project ID is automatically generated.
3. 3. "Post Project" button.

2. Project Management Screen:

1. table displaying:

1. A list of current projects (status: In Progress, Completed, Awaiting Approval).
2. Buttons to edit or cancel a project.

2. **"View Details" button:**

1. Displays a list of translators who have applied for the project.
2. Allows the company to accept a bid or communicate with the translator.

3. Invoice Generation Form:

1. Fields to enter:
    1. Project ID.
    2. Translator fees.
    3. Additional charges (e.g., printing fees).
2. "Generate Invoice" button.

3. Freelancer Interface:

1. Apply for a New Project Screen:

1. A table displaying available projects.
2. An "Apply" button next to each project.
3. The application form includes:
    1. Proposed fee.
    2. Estimated delivery time.
    3. A message for the company.

2. Project Management Screen:

1. A list of current projects:
    1. Project details (name, deadline, status).
    2. "Upload File" button for project submission upon completion.
2. Completed projects displayed with ratings and feedback.

3. Profile Update Form:

1. Fields to edit:
    1. Languages.
    2. Specializations.
    3. Work experience.

4. Upload work samples.

4. Admin Interface:

A. User Management:

1. A table displaying all users (companies and translators).
2. Buttons to activate/deactivate accounts.
3. Search functionality by name or role.

B. Project Management:

1. Displays all published projects.
2. Admin intervention options for resolving disputes.

C. Payment Settings:

1. Displays all transactions between parties.
2. Approves withdrawal requests.

8. Other Requirements (Analysis Models)

1. Use Case diagrams

F1 : Project Management System

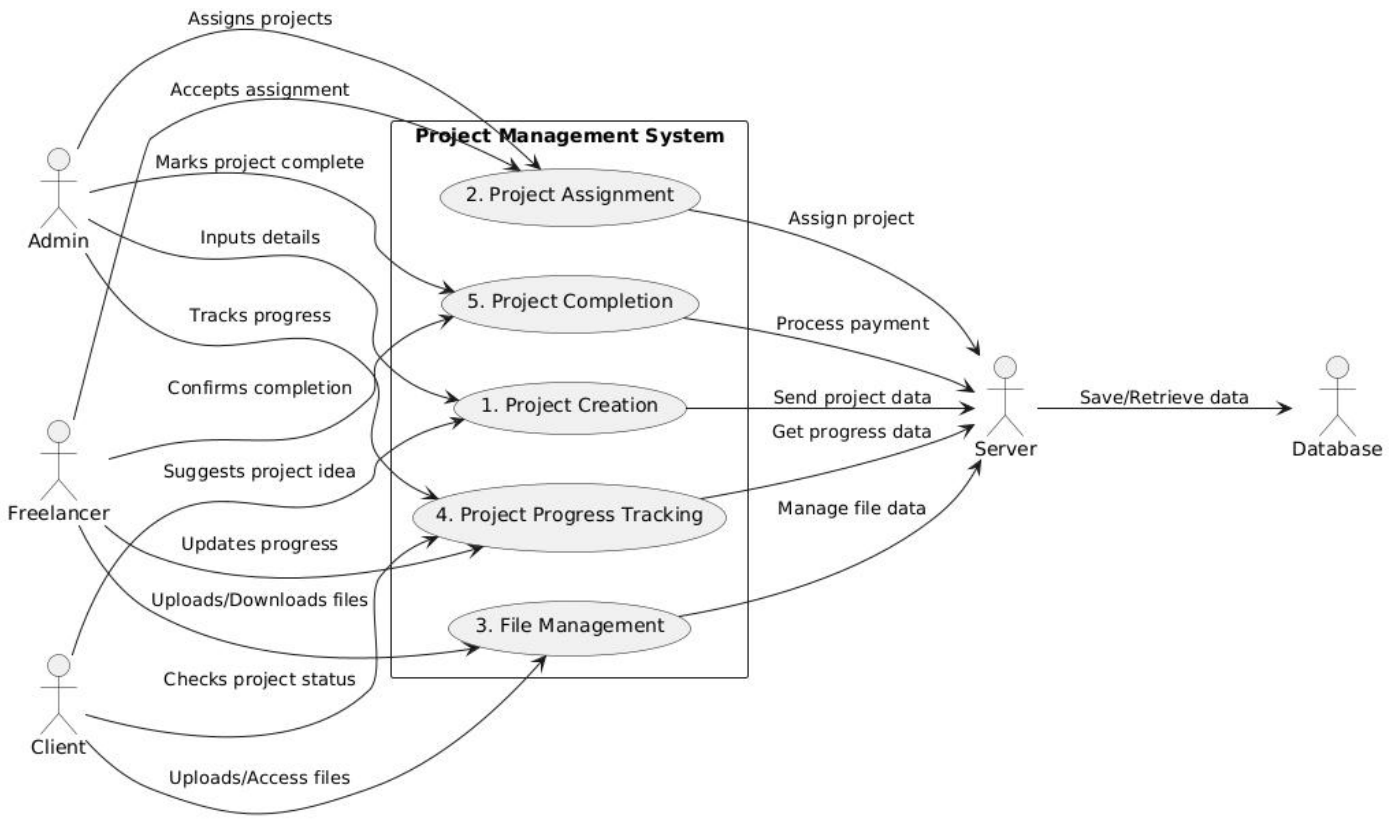


Figure 8.1 Uase Case Project Management Diagram

F2: User Management System

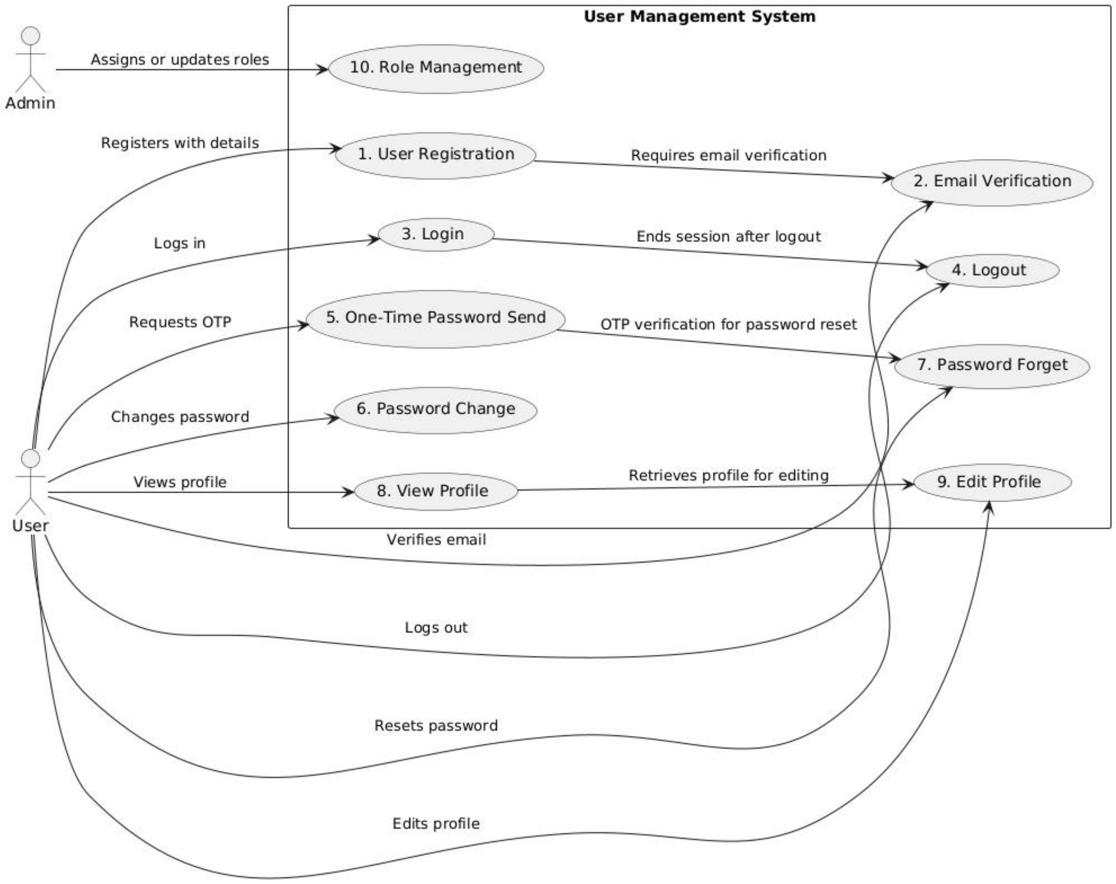


Figure 8.2 Use Case User Management Diagram

F3: Subscription Management System

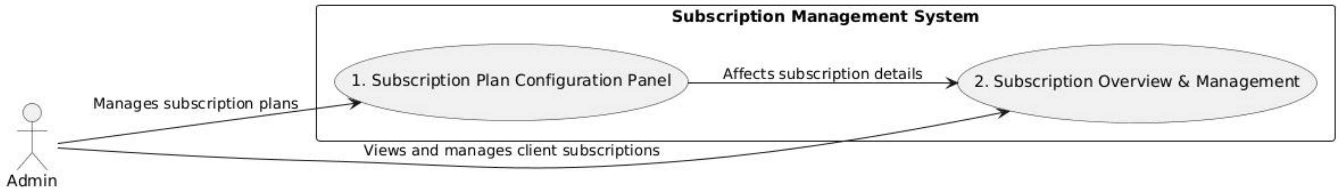

Subscription Management System
1. Subscription Plan Configuration Panel
Affects subscription details
2. Subscription Overview & Management
Manages subscription plans
Views and manages client subscriptions
Admin

Figure 8.3 Use Case Subscription Management Diagram

F4: Apply Process

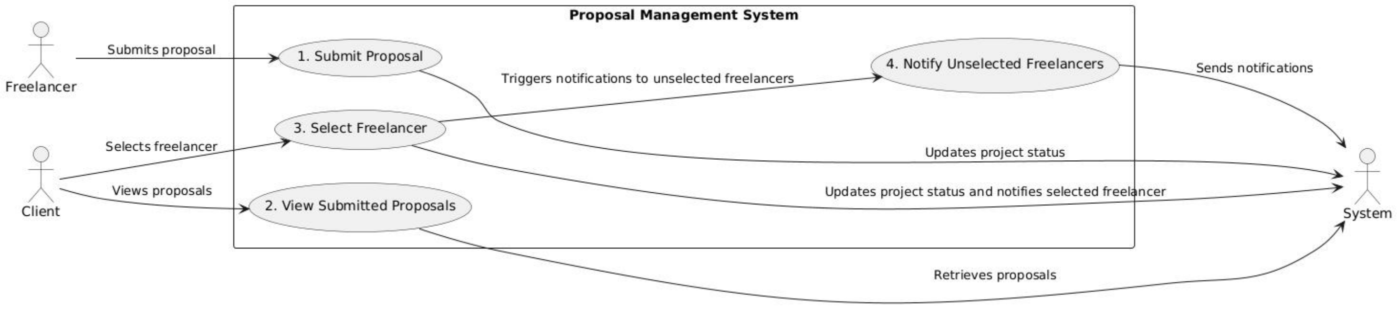

Proposal Management System
Freelancer
Client
System
Submits proposal
Selects freelancer
Views proposals
1. Submit Proposal
3. Select Freelancer
2. View Submitted Proposals
4. Notify Unselected Freelancers
Triggers notifications to unselected freelancers
Sends notifications
Updates project status
Updates project status and notifies selected freelancer
Retrieves proposals

Figure 8.4 Use Case Apply Process Diagram

F5: Dashboard

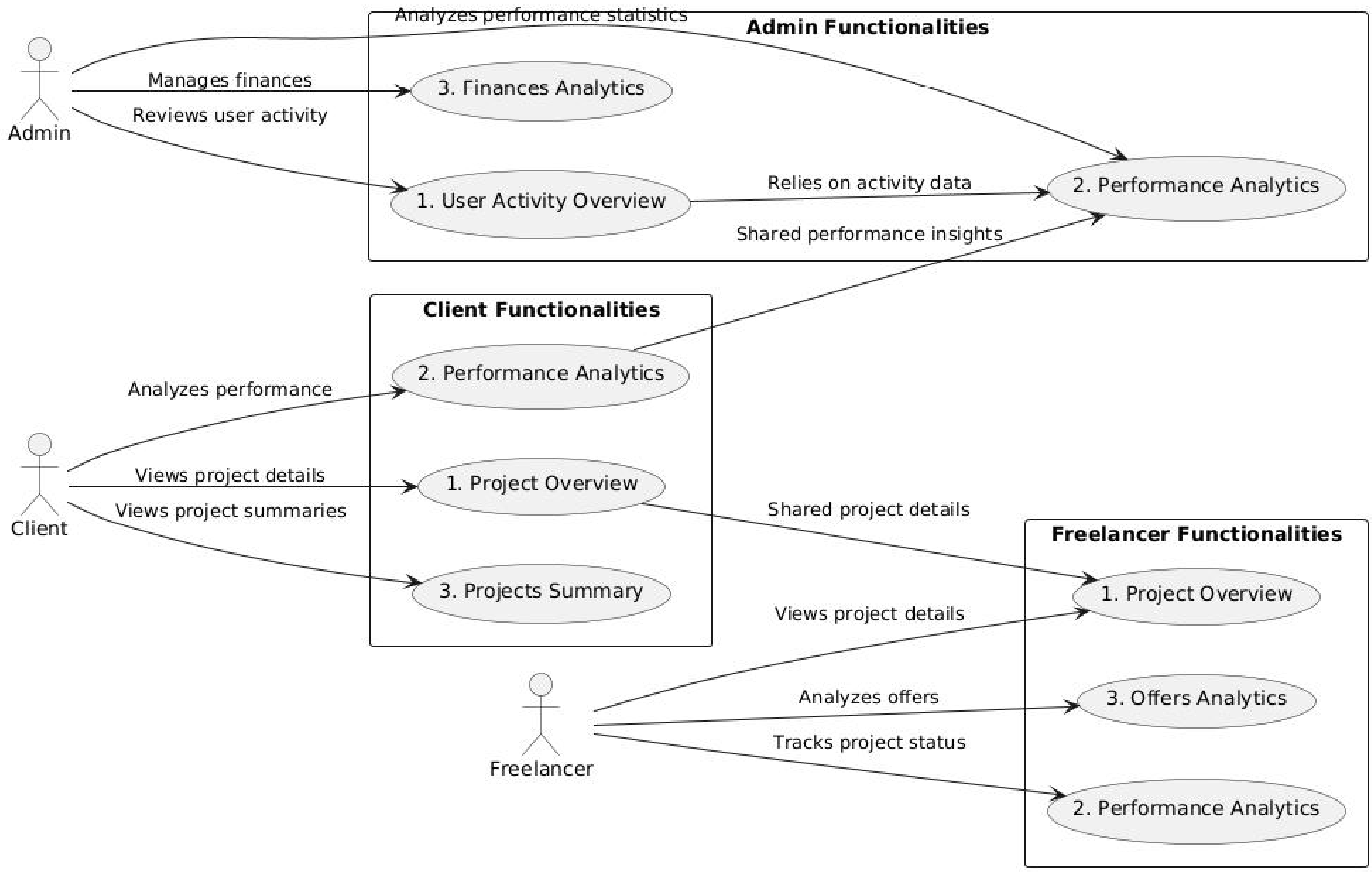


Figure 8.5 Use Case Dashboard Diagram

F6: Notification System

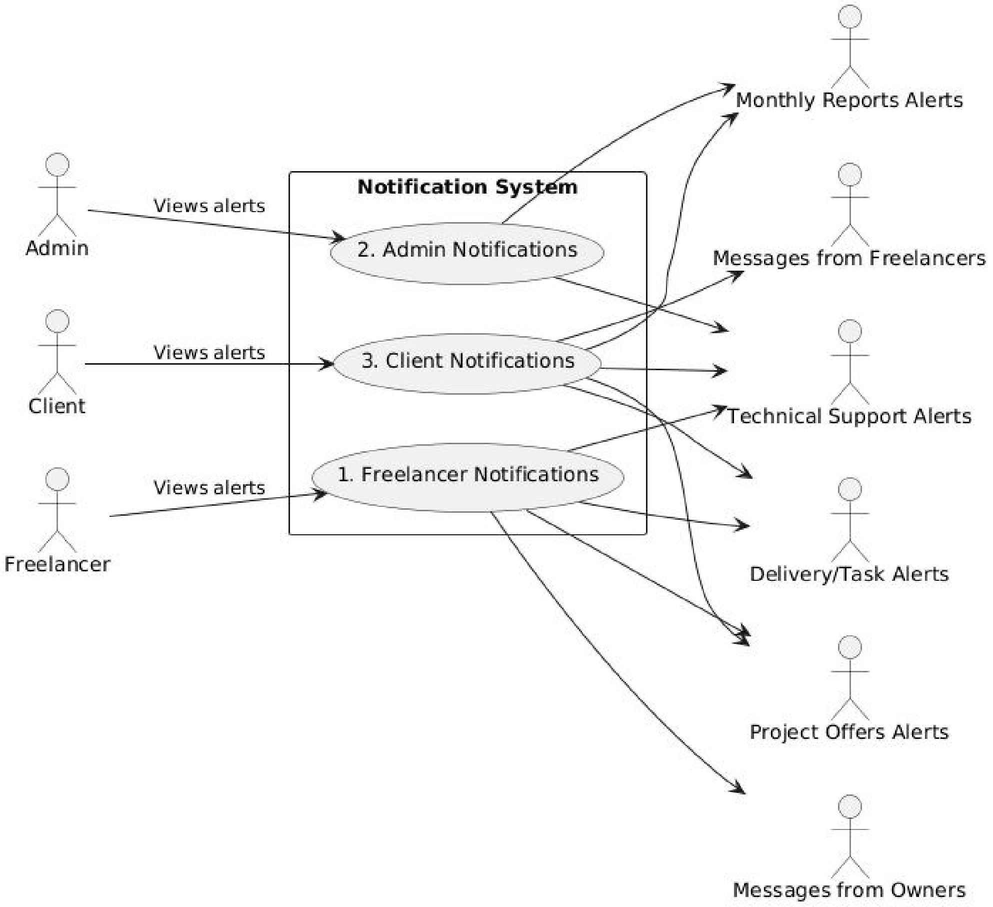


Figure 8.6 Use Case Notification System Diagram

F7: Tracking Projects

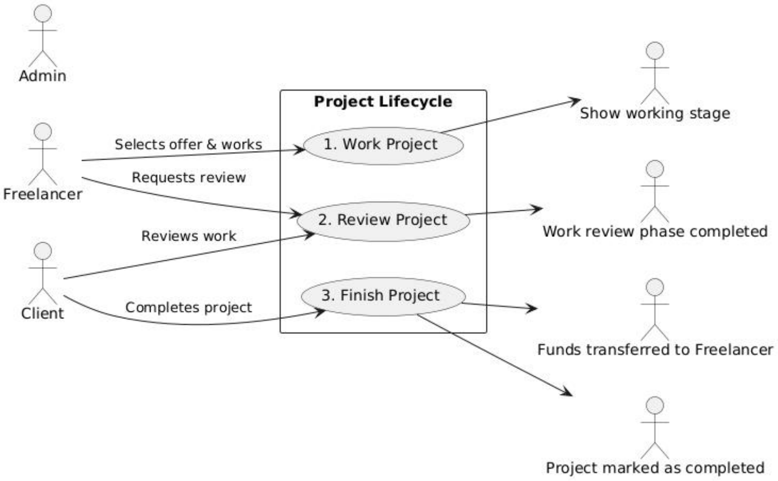


Figure 8.7 Use Case Tracking Projects Diagram

F8: Rating System

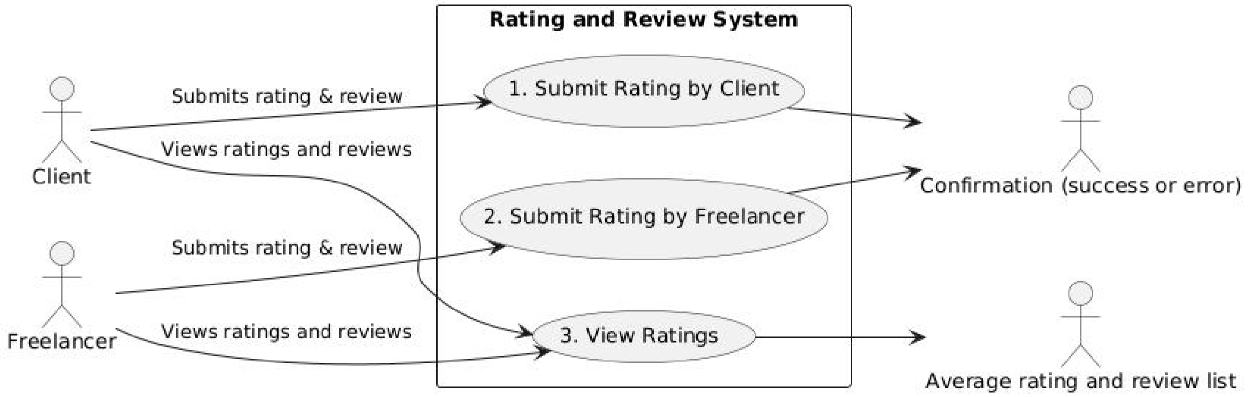

Rating and Review System
Client
Freelancer
Submits rating & review
Views ratings and reviews
Submits rating & review
Views ratings and reviews
1. Submit Rating by Client
2. Submit Rating by Freelancer
3. View Ratings
Confirmation (success or error)
Average rating and review list

Figure 8.8 Use Case Rating System Diagram

F9:Chat System

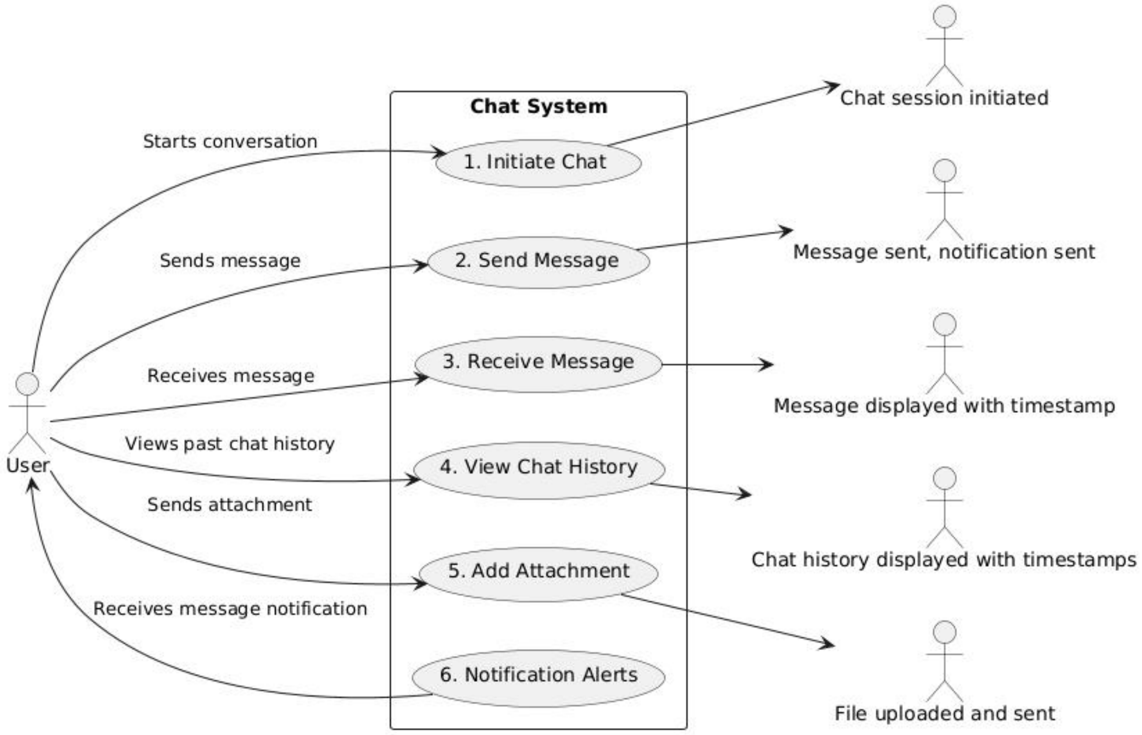


Figure 8.9 Use Case Chat System Diagram

F10: Payment System

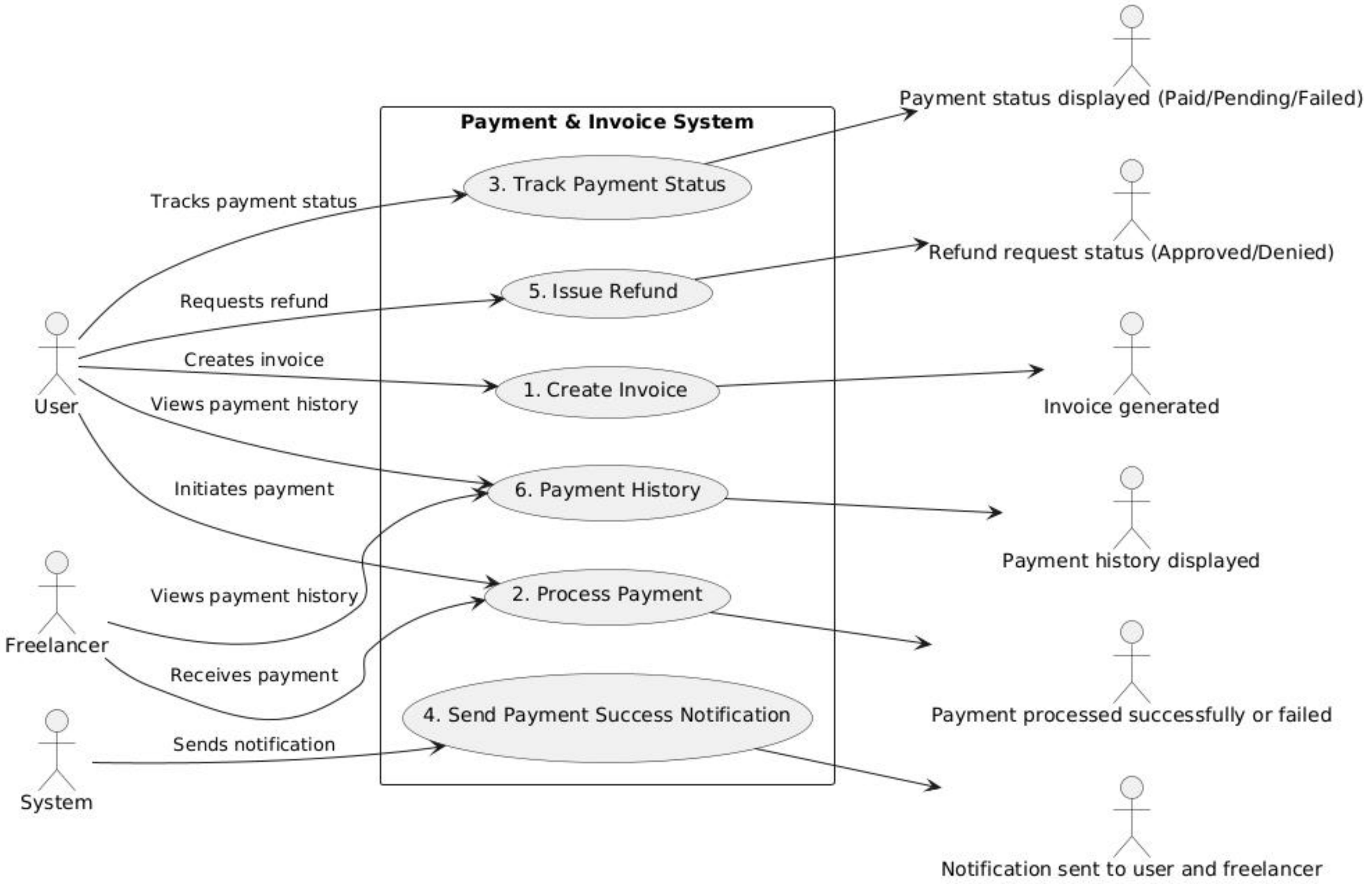


Figure 8.10 Use Case Payment System Diagram

2. Class diagram (Database)

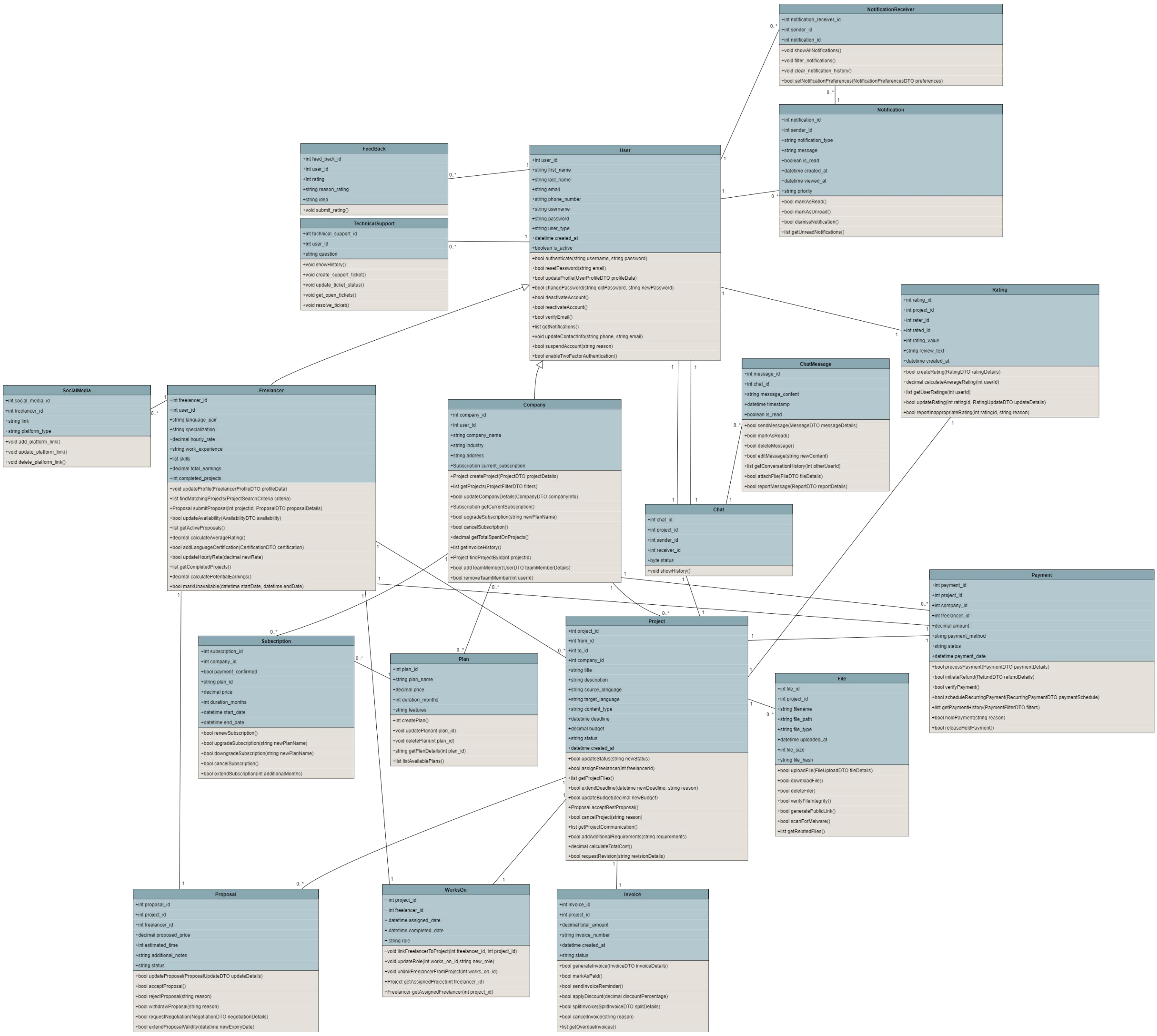
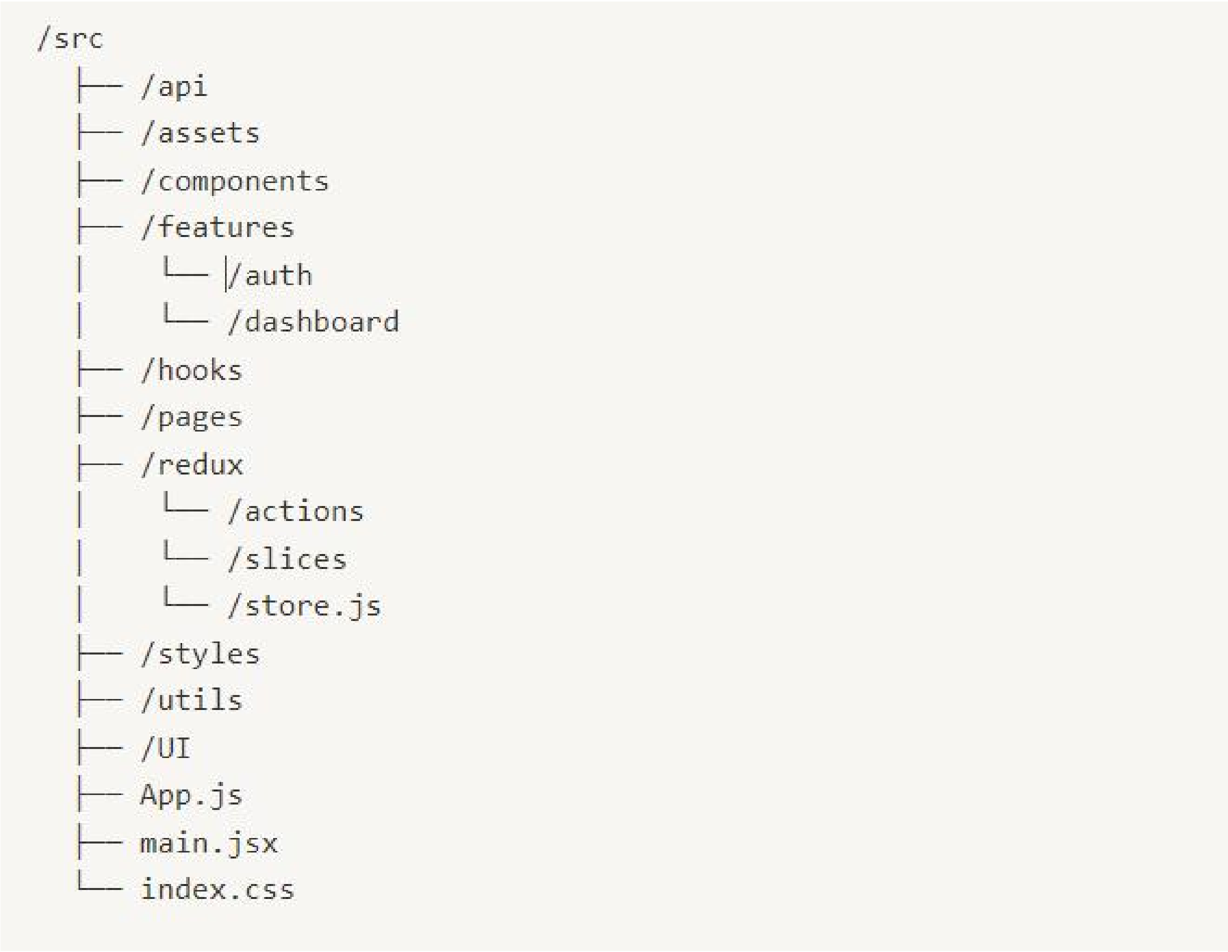

NotificationReceiver
+int notification_receiver_id
+int sender_id
+int notification_id
+void showAllNotifications()
+void filter_notifications()
+void clear_notification_history()
+bool setNotificationPreferences(NotificationPreferencesDTO preferences)
Notification
+int notification_id
+int sender_id
+string notification_type
+string message
+boolean is_read
+datetime created_at
+datetime viewed_at
+string priority
+bool markAsRead()
+bool markAsUnread()
+bool dismissNotification()
+list getUnreadNotifications()
FeedBack
+int feed_back_id
+int user_id
+int rating
+string reason_rating
+string idea
+void submit_rating()
TechnicalSupport
+int technical_support_id
+int user_id
+string question
+void showHistory()
+void create_support_ticket()
+void update_ticket_status()
+void get_open_tickets()
+void resolve_ticket()
User
+int user_id
+string first_name
+string last_name
+string email
+string phone_number
+string username
+string password
+string user_type
+datetime created_at
+boolean is_active
+bool authenticate(string username, string password)
+bool resetPassword(string email)
+bool updateProfile(UserProfileDTO profileData)
+bool changePassword(string oldPassword, string newPassword)
+bool deactivateAccount()
+bool reactivateAccount()
+bool verifyEmail()
+list getNotifications()
+void updateContactInfo(string phone, string email)
+bool suspendAccount(string reason)
+bool enableTwoFactorAuthentication()
Rating
+int rating_id
+int project_id
+int rater_id
+int rated_id
+int rating_value
+string review_text
+datetime created_at
+bool createRating(RatingDTO ratingDetails)
+decimal calculateAverageRating(int userId)
+list getUserRatings(int userId)
+bool updateRating(int ratingId, RatingUpdateDTO updateDetails)
+bool reportInappropriateRating(int ratingId, string reason)
ChatMessage
+int message_id
+int chat_id
+string message_content
+datetime timestamp
+boolean is_read
+bool sendMessage(MessageDTO messageDetails)
+bool markAsRead()
+bool deleteMessage()
+bool editMessage(string newContent)
+list getConversationHistory(int otherUserId)
+bool attachFile(FileDTO fileDetails)
+bool reportMessage(ReportDTO reportDetails)
SocialMedia
+int social_media_id
+int freelancer_id
+string link
+string platform_type
+void add_platform_link()
+void update_platform_link()
+void delete_platform_link()
Freelancer
+int freelancer_id
+int user_id
+string language_pair
+string specialization
+decimal hourly_rate
+string work_experience
+list skills
+decimal total_earnings
+int completed_projects
+void updateProfile(FreelancerProfileDTO profileData)
+list findMatchingProjects(ProjectSearchCriteria criteria)
+Proposal submitProposal(int projectId, ProposalDTO proposalDetails)
+bool updateAvailability(AvailabilityDTO availability)
+list getActiveProposals()
+decimal calculateAverageRating()
+bool addLanguageCertification(CertificationDTO certification)
+bool updateHourlyRate(decimal newRate)
+list getCompletedProjects()
+decimal calculatePotentialEarnings()
+bool markUnavailable(datetime startDate, datetime endDate)
Company
+int company_id
+int user_id
+string company_name
+string industry
+string address
+Subscription current_subscription
+Project createProject(ProjectDTO projectDetails)
+list getProjects(ProjectFilterDTO filters)
+bool updateCompanyDetails(CompanyDTO companyInfo)
+Subscription getCurrentSubscription()
+bool upgradeSubscription(string newPlanName)
+bool cancelSubscription()
+decimal getTotalSpentOnProjects()
+list getInvoiceHistory()
+Project findProjectById(int projectId)
+bool addTeamMember(UserDTO teamMemberDetails)
+bool removeTeamMember(int userId)
Chat
+int chat_id
+int project_id
+int sender_id
+int receiver_id
+byte status
+void showHistory()
Payment
+int payment_id
+int project_id
+int company_id
+int freelancer_id
+decimal amount
+string payment_method
+string status
+datetime payment_date
+bool processPayment(PaymentDTO paymentDetails)
+bool initiateRefund(RefundDTO refundDetails)
+bool verifyPayment()
+bool scheduleRecurringPayment(RecurringPaymentDTO paymentSchedule)
+list getPaymentHistory(PaymentFilterDTO filters)
+bool holdPayment(string reason)
+bool releaseHeldPayment()
Subscription
+int subscription_id
+int company_id
+bool payment_confirmed
+string plan_id
+decimal price
+int duration_months
+datetime start_date
+datetime end_date
+bool renewSubscription()
+bool upgradeSubscription(string newPlanName)
+bool downgradeSubscription(string newPlanName)
+bool cancelSubscription()
+bool extendSubscription(int additionalMonths)
Plan
+int plan_id
+string plan_name
+decimal price
+int duration_months
+string features
+int createPlan()
+void updatePlan(int plan_id)
+void deletePlan(int plan_id)
+string getPlanDetails(int plan_id)
+list listAvailablePlans()
Project
+int project_id
+int from_id
+int to_id
+int company_id
+string title
+string description
+string source_language
+string target_language
+string content_type
+datetime deadline
+decimal budget
+string status
+datetime created_at
+bool updateStatus(string newStatus)
+bool assignFreelancer(int freelancerId)
+list getProjectFiles()
+bool extendDeadline(datetime newDeadline, string reason)
+bool updateBudget(decimal newBudget)
+Proposal acceptBestProposal()
+bool cancelProject(string reason)
+list getProjectCommunication()
+bool addAdditionalRequirements(string requirements)
+decimal calculateTotalCost()
+bool requestRevision(string revisionDetails)
File
+int file_id
+int project_id
+string filename
+string file_path
+string file_type
+datetime uploaded_at
+int file_size
+string file_hash
+bool uploadFile(FileUploadDTO fileDetails)
+bool downloadFile()
+bool deleteFile()
+bool verifyFileIntegrity()
+bool generatePublicLink()
+bool scanForMalware()
+list getRelatedFiles()
Proposal
+int proposal_id
+int project_id
+int freelancer_id
+decimal proposed_price
+int estimated_time
+string additional_notes
+string status
+bool updateProposal(ProposalUpdateDTO updateDetails)
+bool acceptProposal()
+bool rejectProposal(string reason)
+bool withdrawProposal(string reason)
+bool requestNegotiation(NegotiationDTO negotiationDetails)
+bool extendProposalValidity(datetime newExpiryDate)
WorksOn
+ int project_id
+ int freelancer_id
+ datetime assigned_date
+ datetime completed_date
+ string role
+void linkFreelancerToProject(int freelancer_id, int project_id)
+void updateRole(int works_on_id,string new_role)
+void unlinkFreelancerFromProject(int works_on_id)
+Project getAssignedProject(int freelancer_id)
+Freelancer getAssignedFreelancer(int project_id)
Invoice
+int invoice_id
+int project_id
+decimal total_amount
+string invoice_number
+datetime created_at
+string status
+bool generateInvoice(InvoiceDTO invoiceDetails)
+bool markAsPaid()
+bool sendInvoiceReminder()
+bool applyDiscount(decimal discountPercentage)
+bool splitInvoice(SplitInvoiceDTO splitDetails)
+bool cancelInvoice(string reason)
+list getOverdueInvoices()

Figure 8.11 class Diagram

3. Architecture

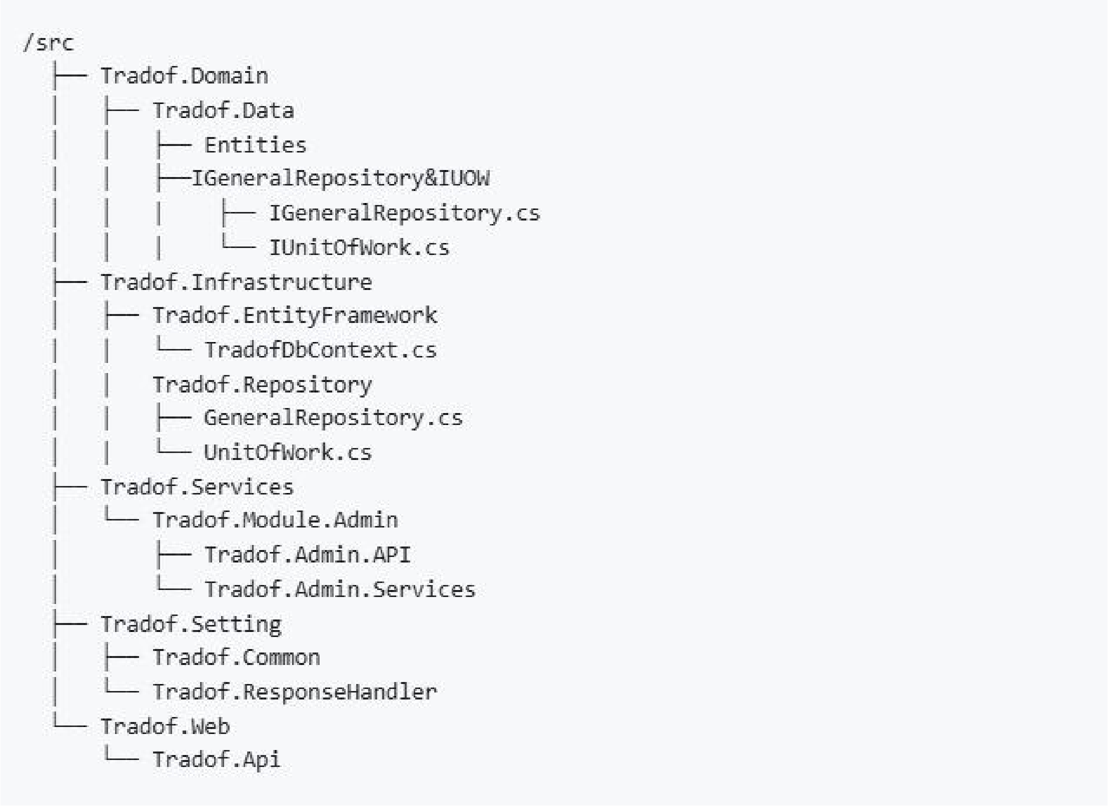
1. Front-end (React.JS) Architecture:



```
/src
  ├── /api
  ├── /assets
  ├── /components
  ├── /features
  │     └── /auth
  │     └── /dashboard
  ├── /hooks
  ├── /pages
  ├── /redux
  │     └── /actions
  │     └── /slices
  │     └── /store.js
  ├── /styles
  ├── /utils
  ├── /UI
  ├── App.js
  ├── main.jsx
  └── index.css
```

Figure 8.13 Front End Architecture

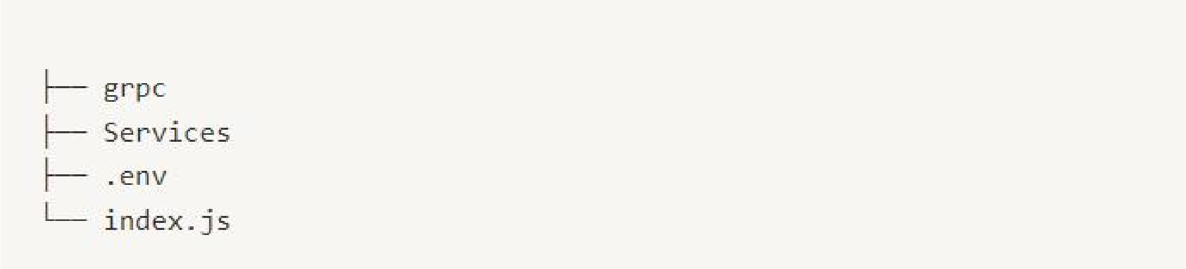
2. Back-end (.Net)



```
/src
  ├── Tradof.Domain
  │   ├── Tradof.Data
  │   │   ├── Entities
  │   │   ├──IGeneralRepository&IUOW
  │   │   │     ├── IGeneralRepository.cs
  │   │   │     └── IUnitOfWork.cs
  ├── Tradof.Infrastructure
  │   ├── Tradof.EntityFramework
  │   │   └── TradofDbContext.cs
  │   │   Tradof.Repository
  │   │   ├── GeneralRepository.cs
  │   │   └── UnitOfWork.cs
  ├── Tradof.Services
  │   └── Tradof.Module.Admin
  │       ├── Tradof.Admin.API
  │       └── Tradof.Admin.Services
  ├── Tradof.Setting
  │   ├── Tradof.Common
  │   └── Tradof.ResponseHandler
  └── Tradof.Web
      └── Tradof.Api
```

Figure 8.14 Back-end (.Net) Architecture

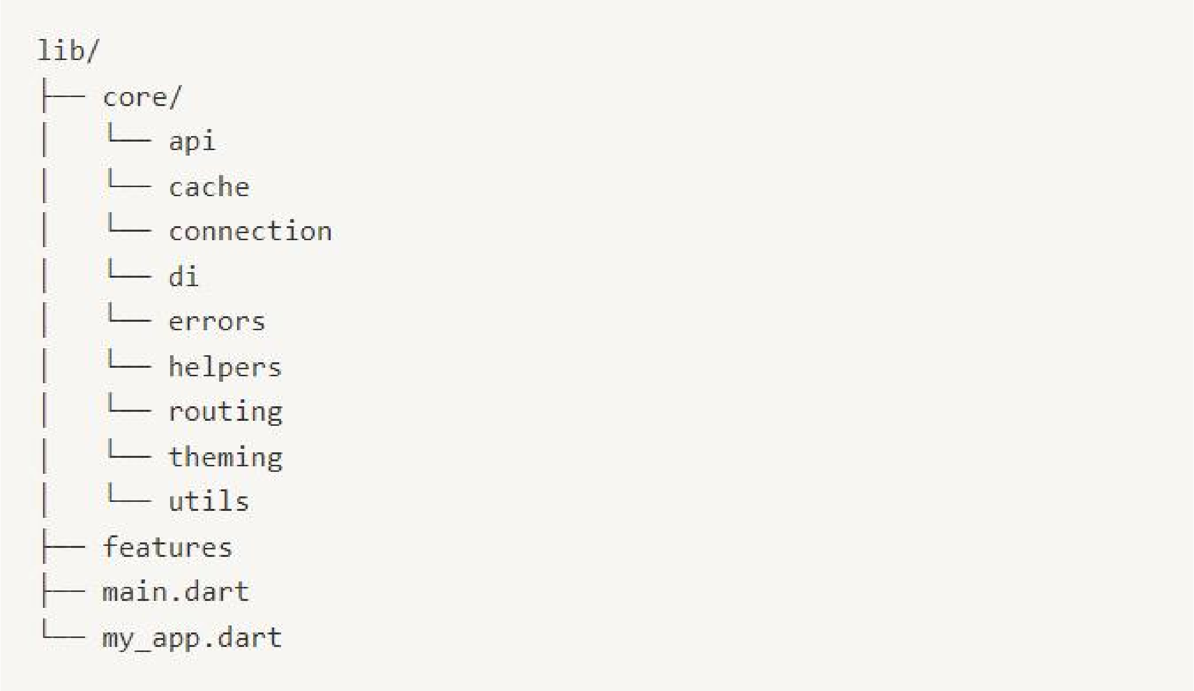
3. Back-end (Node.JS)



```
├── grpc
├── Services
├── .env
└── index.js
```

Figure 8.15 Back-end (Node) Architecture

4. Mobile App (Flutter)



```
lib/
├── core/
│   └── api
│   └── cache
│   └── connection
│   └── di
│   └── errors
│   └── helpers
│   └── routing
│   └── theming
│   └── utils
├── features
├── main.dart
└── my_app.dart
```

Figure 8.16 Mobile App Architecture