FREELENSO

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT

OF REQUIREMENT

FOR THE AWARD OF THE DEGREE

MASTER OF COMPUTER APPLICATIONS (MCA)

OF

MAHATMA GANDHI UNIVERSITY, KOTTAYAM

BY

JAYASURYA SUDHAKARAN

Reg No: 24PMC123

MARIAN COLLEGE
KUTTIKKANAM
AUTONOMOUS
MAKING COMPLETE

Marian College Kuttikanam Autonomous

Peermade, Kerala – 685 531

2025

FREELENSO

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF REQUIREMENT

FOR THE AWARD OF THE DEGREE

MASTER OF COMPUTER APPLICATIONS (MCA)

OF

MAHATMA GANDHI UNIVERSITY, KOTTAYAM

BY

JAYASURYA SUDHAKARAN

Reg No: 24PMC123

MARIAN COLLEGE
KUTTIKKANAM
AUTONOMOUS
MAKING COMPLETE

Marian College Kuttikanam Autonomous

Peermade, Kerala – 685 531

2025

A Project Report on

FREELENSO

SUBMITTED IN PARTIAL FULFILMENT OF REQUIREMENT

FOR THE AWARD OF THE DEGREE

MASTER OF COMPUTER APPLICATIONS (MCA)

OF

MAHATMA GANDHI UNIVERSITY, KOTTAYAM

BY

JAYASURYA SUDHAKARAN

Reg No: 24PMC123

Under the guidance of

Mr. Robins A Kattoor

Assistant Professor

PG Department of Computer Applications

Marian College Kuttikkanam Autonomous

Marian College Kuttikanam Autonomous

Peermade, Kerala – 685 531

2025

PG DEPARTMENT OF COMPUTER APPLICATIONS

Marian College Kuttikkanam Autonomous

MAHATMA GANDHI UNIVERSITY, KOTTAYAM

KUTTIKKANAM – 685 531, KERALA.

CERTIFICATE

This is to certify that the project work entitled.

FREELENSO

is a bonafide record of work done by

JAYASURYA SUDHAKARAN

Reg. No. 24PMC123

In partial fulfillment of the requirements for the award of Degree of

MASTER OF COMPUTER APPLICATIONS [MCA]

During the academic year 2024-2026

Mr. Robins A Kattoor
Assistant Professor
PG Department of Computer Applications
Marian College Kuttikkanam Autonomous

Mr. Win Mathew John
Head of the Department
PG Department of Computer Applications
Marian College Kuttikkanam Autonomous

External Examiner

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who contributed to the successful completion of this project.

First and foremost, I am grateful to **God Almighty** for His divine guidance, grace, and strength throughout this journey, without which this project would not have been possible.

I would like to extend my heartfelt thanks to **Dr. Ajimon George**, Principal of Marian College Kuttikkanam (Autonomous), and **Dr. Mendus Jacob**, Director of the PG Department of Computer Applications, for their continuous support and encouragement.

A special mention goes to **Mr. Win Mathew John**, Head of the PG Department of Computer Applications, whose leadership and motivation helped me stay focused and driven throughout the project.

I am deeply thankful to my **project guide, Mr. Robins A Kattoor**, Assistant Professor, PG Department of Computer Applications, for her expert guidance, patience, and valuable insights, which played an essential role in shaping this project.

I would also like to express my appreciation to all the faculty members of the **PG Department of Computer Applications**, whose support and timely assistance enriched my learning experience during the project.

Finally, I would like to thank my **family and friends** for their unwavering support, love, and encouragement, which gave me the strength to overcome challenges and complete this project successfully

JAYASURYA SUDHAKARAN

ABSTRACT

FREELENSO

This project, **FREELENSO**, is a Django-based web application designed to provide a secure and efficient digital platform where freelancers and clients can connect, collaborate, and complete projects online. The system supports various service categories such as web development, graphic design, and content writing. FREELENSO offers a seamless and user-friendly experience, allowing clients to post projects and hire professionals, while freelancers can showcase their portfolios, bid on jobs, and receive payments safely through an escrow system. Additional features such as real-time notifications, messaging, and review systems ensure smooth communication and transparency throughout the workflow.

Statement about the Problem: Traditional freelance workflows often suffer from trust issues, lack of payment security, poor communication, and difficulty in finding skilled professionals. Informal methods of collaboration can lead to misunderstandings, payment delays, or incomplete projects. FREELENSO addresses these challenges by providing a structured digital platform that ensures accountability, secure transactions, and smooth communication between freelancers and clients.

Objective and Scope of the Project The primary objective of FREELENSO is to build a robust and easy-to-use freelance platform that supports efficient project management and secure payments. The scope of the project includes two types of users:

1. Freelancers:

- Create and manage profile
- Communicate with clients
- Submit completed work
- Receive payments securely

2. Clients :

- Manage laundry item listings
- Process user requests
- Monitor service status

- Handle user feedback
- Generate service reports

‘

Methodology: The project leverages modern web technologies to ensure scalability, performance, and ease of use. The backend is built using Python and Django, which provides robust features for handling authentication, request management, and database interactions. The frontend uses HTML5, CSS3, and JavaScript to create a visually appealing and responsive user interface.

Hardware & Software:

• **Hardware:** Standard development and deployment systems
• **Software:**

- **Backend:** Python and Django
- **Frontend:** HTML5, CSS3, JavaScript
- **Database:** SQLite (or MySQL for production)
- **Tools:** Git for version control, VS Code / PyCharm for development

Testing Technologies:
To ensure a stable and reliable application, FREELENSO will be tested using:
• **Unit Testing:** To verify individual components (e.g., bidding, messaging)
• **Integration Testing:** To check the interaction between modules (e.g., user registration and project bidding)
• **User Acceptance Testing (UAT):** To validate end-to-end functionality and usability from a real user's perspective

Contribution of the Project: FREELENSO supports the freelance economy by offering a complete solution that connects talent with opportunity in a secure digital space. It benefits freelancers with better exposure and reliable payments, helps clients find the right professionals, and equips admins with tools to manage the ecosystem effectively.

Conclusion: The development of FREELENSO meets the growing demand for a structured freelance platform that ensures transparency, efficiency, and security. With features tailored for all stakeholders—freelancers, clients, and administrators—FREELENSO delivers a professional online environment that simplifies project collaboration and builds trust across users.

TABLE OF CONTENTS

TABLE INDEX

FIGURE INDEX

Figure Page No.

1. INTRODUCTION

1. INTRODUCTION

1.1 PROBLEM STATEMENT

In today's evolving gig economy, freelancers and clients face numerous challenges when connecting through existing online platforms. Many freelance marketplaces suffer from issues such as high commission fees, lack of secure payment mechanisms, ineffective dispute resolution, and difficulty in finding the right talent or projects. Additionally, users often encounter complex interfaces that hinder seamless interaction.

FREELENSO aims to address these challenges by providing a user-friendly, secure, and efficient freelance platform. With features such as an escrow payment system, real-time notifications, advanced search and filtering, and a transparent rating and review system, FREELENSO ensures a seamless and trustworthy experience for both freelancers and clients. The platform is designed to minimize complexity while maximizing efficiency, creating a space where professionals can connect, collaborate, and complete projects with confidence.

.

1.2 PROPOSED SYSTEM

FREELENSO is a scalable and user-friendly freelance marketplace designed to connect clients with skilled freelancers in a secure and efficient environment. The platform features a clean, intuitive interface built using Django for the backend, with a responsive frontend powered by HTML, CSS, and JavaScript.Users can create accounts through a validated registration process, set up detailed profiles showcasing their skills and portfolios, and seamlessly browse or post freelance projects. Clients can post projects with specific requirements, budgets, and deadlines, while freelancers can submit proposals and negotiate terms.To ensure secure transactions, FREELENSO integrates an escrow payment system that holds funds until project completion, protecting both parties. Real-time notifications keep users informed of project updates, bid responses, and payment statuses. The system also incorporates a rating and review mechanism to maintain credibility and trust within the community.Administrators have access to comprehensive user and transaction management tools, allowing them to monitor activities, handle disputes, and enforce

platform policies. Advanced search and filtering options enhance usability, making it easier for clients to find the right talent and for freelancers to discover relevant opportunities. FREELENSO prioritizes security and transparency, implementing robust authentication measures and secure data handling. The responsive design ensures seamless performance across different devices, providing a smooth user experience without unnecessary complexity. By balancing functionality with ease of use, FREELENSO aims to revolutionize the freelance marketplace, fostering a productive and trustworthy ecosystem for both freelancers and clients.

1.3 FEATURES OF THE PROPOSED SYSTEM

FREELENSO incorporates a comprehensive set of features designed to streamline the freelance hiring and project execution process while maintaining a user-friendly and secure environment. The platform supports secure user registration with robust form validation, enabling both clients and freelancers to create and customize detailed profiles that highlight skills, portfolios, and reviews.The project management system allows clients to post projects with defined budgets and timelines, while freelancers can browse opportunities, submit proposals, and communicate with potential employers. An integrated bidding mechanism facilitates transparent negotiations and project selection.FREELENSO ensures transactional security through an escrow-based payment system, where funds are held until project completion and client approval, offering peace of mind to both parties. A two-way rating and review system promotes trust and accountability across the platform.Real-time notification systems keep users informed of important updates such as new bids, project milestones, and payment releases. The interface supports responsive layouts, clean navigation, and intuitive controls, ensuring accessibility across all devices.Admin-level functionalities include user account management, dispute resolution tools, transaction monitoring, and handling of user reports, ensuring platform integrity and safety. Advanced filtering and search options enhance discoverability of relevant projects or talent, while real-time communication tools support efficient collaboration.All features are integrated into a cohesive and accessible system, combining performance, security, and simplicity—making FREELENSO a reliable and efficient solution for freelancers and clients alike.

2. FUNCTIONAL REQUIREMENTS

2. FUNCTIONAL REQUIREMENTS

2.1 User Authentication and Access Control

Users can create accounts with validated data, log in securely, and recover/change passwords when needed. Email verification ensures account security. The system implements client-side validation for all form fields to prevent invalid submissions. Password fields feature visibility toggles to help users verify their entries while maintaining security.

2.2 Profile Management

Users can customize profiles with personal information, profile pictures, and view their activity history. Account deletion functionality is available. The profile interface provides intuitive editing options with immediate visual feedback. Users can manage their personal data through dedicated modal interfaces with confirmation steps for critical actions.

2.3 Messaging System

Users can send and receive messages in individual and group conversations, with attachments and formatting options. Message history is maintained with timestamps. The messaging interface organizes conversations chronologically and visually distinguishes between sent and received messages. Users can scroll through conversation history with all past exchanges preserved.

2.4 Message Management

Users can delete messages (for themselves or everyone), pin important messages, and report inappropriate content through custom confirmation dialogs. Message context menus provide quick access to management options with appropriate confirmations for irreversible actions. Pinned messages remain visible at the top of conversations for quick reference.

2.5 Contact and Privacy Controls

Users can add/remove contacts, block users, and archive conversations. Privacy settings allow control over who can initiate conversations. The blocking system prevents unwanted communication while maintaining a record of past exchanges. Archived conversations can be restored when needed while keeping the main interface uncluttered..

2.6 Notification System

Users receive notifications for new messages, friend requests, and system updates.

Unread message indicators highlight conversations requiring attention.

The notification system provides visual cues without interrupting the user experience, maintaining counts of unread items across the platform.

2.7 User Interface

The system provides responsive dark-themed interfaces with intuitive navigation, context menus, and form validation feedback for enhanced user experience. The design maintains consistent styling and interaction patterns throughout the application, with accessible color contrasts and readable typography

3. NON-FUNCTIONAL REQUIREMENTS

3.NON-FUNCTIONAL REQUIREMENTS

Performance Requirements

- The system shall support concurrent access by multiple users without performance degradation.
- The system shall be capable of handling voting sessions involving hundreds of students simultaneously.
- Page load times shall not exceed 2 seconds under normal operating conditions.

Scalability

- The system shall allow migration from SQLite to more robust databases like PostgreSQL for handling larger data volumes in the future.

Usability

- The system shall offer a clean, intuitive, and responsive user interface suitable for users with basic technical skills.
- The user interface shall be mobile-responsive to ensure accessibility across devices (phones, tablets, desktops).
- Error messages and validation prompts shall be clearly communicated to users in a user-friendly manner.

Reliability

System Availability:

The Freelenso platform shall be available 99.5% of the time during peak hours and normal usage periods, ensuring consistent access for both clients and freelancers.

Data Integrity:

All data submitted by users — such as project postings, proposals, transactions, and reviews — shall be accurately stored and processed without loss or corruption.

Backup Mechanism:

The system shall perform regular automated backups of the database and media files to prevent data loss in case of server failures or application errors.

Maintainability

- The system shall follow modular and organized code structure using Django's MVC (Model-View-Controller) architecture.
- The codebase shall be documented to support future enhancements and debugging.
- Admins shall be able to perform user and department management without modifying the backend code.

Portability

- The application shall be deployable on any server supporting Python and Django.
- Static and media files shall be organized and easily configurable for different environments (development/production).

Compatibility

- The system shall be compatible with major web browsers including Chrome, Firefox, Edge, and Safari.
- The frontend shall be built using Bootstrap 5 to ensure cross-platform responsiveness and visual consistency.

4. UML DIAGRAMS

4.1 USE CASES

4.1.1 Student Login

- **Use Case ID:** FR_01
- **Use Case Name:** Register
- **Date Created:** 2025-05-14
- **Description:** Allows new users (clients or freelancers) to create an account on Freelenso.
- **Primary Actor:** User
- **Secondary Actor:** None
- **Precondition:** User is not registered.
- **Postcondition:** Account is created and user is redirected to login.
- **Main Flow:**
    1. User clicks “Register”.
    2. Fills in details (name, email, password, role).
    3. Submits the registration form.
    4. System validates and saves the data.
    5. User is redirected to the login page.
- **Alternative Flows:**
    1. Invalid input: Validation errors shown on the form.

4.1.2 login

- **Use Case ID:** FR_02
- **Use Case Name:** Login
- **Date Created:** 2025-05-14
- **Description:** Allows registered users to log in to their accounts.
- **Primary Actor:** User
- **Secondary Actor:** None
- **Precondition:** Valid user account exists.
- **Postcondition:** User is redirected to their dashboard.
- **Main Flow:**

1. User enters email and password.
2. Clicks “Login”.
3. System validates credentials.
4. If valid, redirects to dashboard.
5. If invalid, remains on login with error.

- **Alternative Flows:**
  1. Invalid credentials: System shows error message.

4.1.4 Create Profile

- **Use Case ID:** FR_03
- **Use Case Name:** Create Profile
- **Date Created:** 2025-05-14
- **Description:** Allows users to set up and update their profile with relevant information.
- **Primary Actor:** User (Client or Freelancer)
- **Secondary Actor:** None
- **Precondition:** User is logged in.
- **Postcondition:** Profile information is saved.
- **Main Flow:**
  1. User navigates to “Profile” section.
  2. Enters or updates profile data.
  3. Clicks save.
  4. System stores updated information.
- **Alternative Flows:**
  1. Validation errors shown for missing or invalid fields.

4.1.5 Post Project

- **Use Case ID:** FR_04
- **Use Case Name:** Post Project
- **Date Created:** 2025-05-14
- **Description:** Enables clients to post new projects with details like budget and deadline.

- **Primary Actor:** Client
- **Secondary Actor:** None
- **Precondition:** Client is logged in.
- **Postcondition:** Project is posted and visible to freelancers.
- **Main Flow:**
    1. Client selects “Post Project”.
    2. Fills in project details.
    3. Clicks “Submit”.
    4. System validates and saves project.
- **Alternative Flows:**
    1. Missing required fields: Form displays error messages.

4.1.6 Submit Proposal

- **Use Case ID:** FR_05
- **Use Case Name:** Submit Proposal
- **Date Created:** 2025-05-14
- **Description:** Allows freelancers to bid on posted projects.
- **Primary Actor:** Freelancer
- **Secondary Actor:** None
- **Precondition:** Freelancer is logged in.
- **Postcondition:** Proposal is submitted to client.
- **Main Flow:**
    1. Freelancer views a project.
    2. Clicks “Submit Proposal”.
    3. Enters proposal details and submits.
    4. System stores and notifies the client.
- **Alternative Flows:**
    1. Invalid proposal fields: Error shown.

4.1.7 Make Payment

- **Use Case ID:** FR_06
- **Use Case Name:** Make Payment

- **Date Created:** 2025-05-14
- **Description:** Enables clients to make payments securely through the platform.
- **Primary Actor:** Client
- **Secondary Actor:** Payment Gateway
- **Precondition:** Client has accepted a freelancer proposal.
- **Postcondition:** Payment is recorded and processed via escrow.
- **Main Flow:**
    1. Client clicks “Make Payment”.
    2. Enters payment details.
    3. Payment gateway processes it.
    4. System confirms and holds payment.
- **Alternative Flows:**
    1. Payment failure: Prompt to retry.

4.2 USECASE DIAGRAM

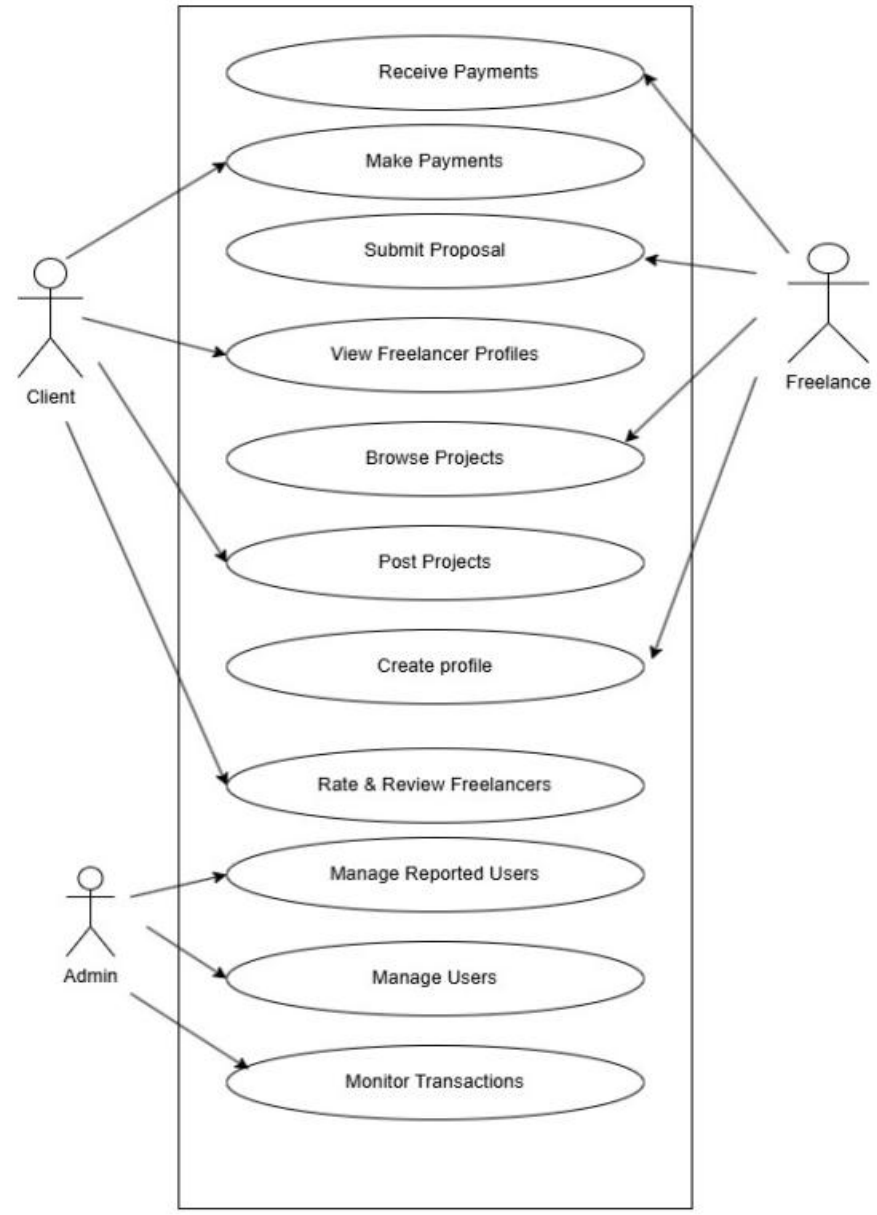


Fig 4.2.1 Use case

4.3 ACTIVITY DIAGRAM

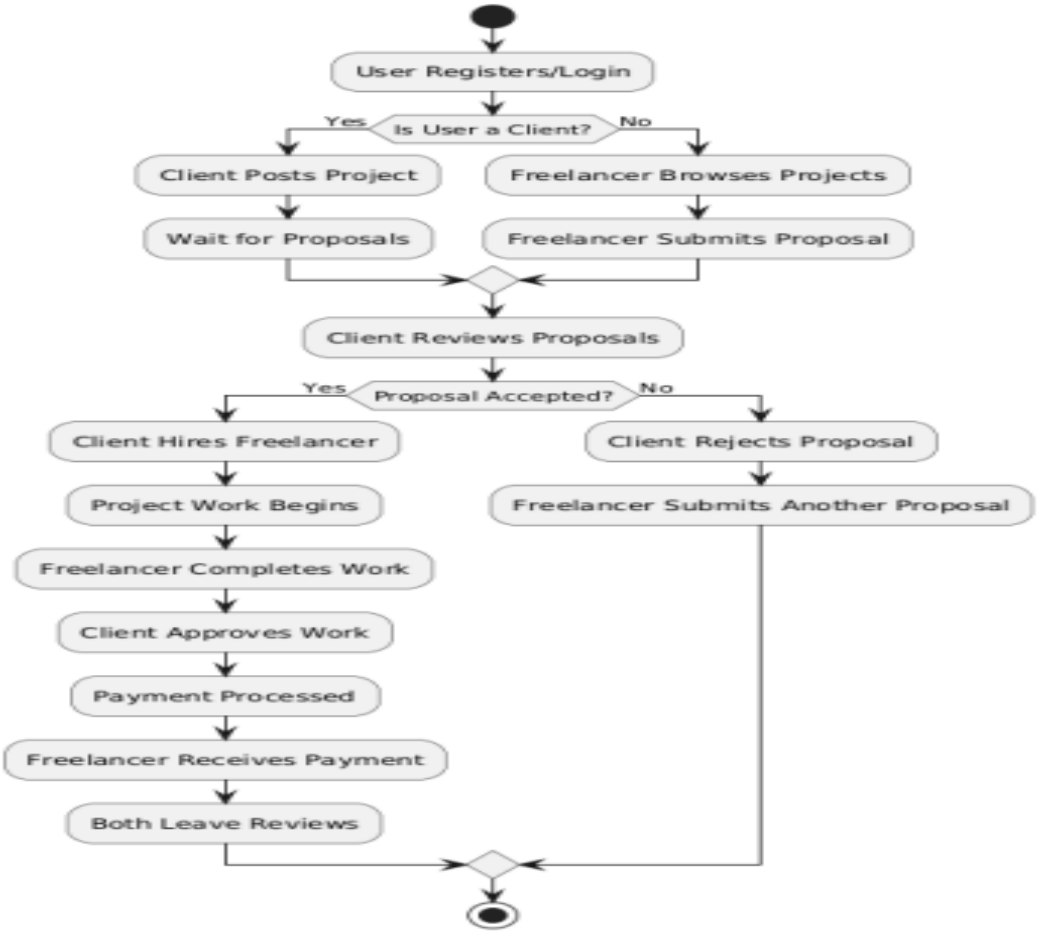


Fig 4.3.1 Activity Diagram

4.4 CLASS DIAGRAM

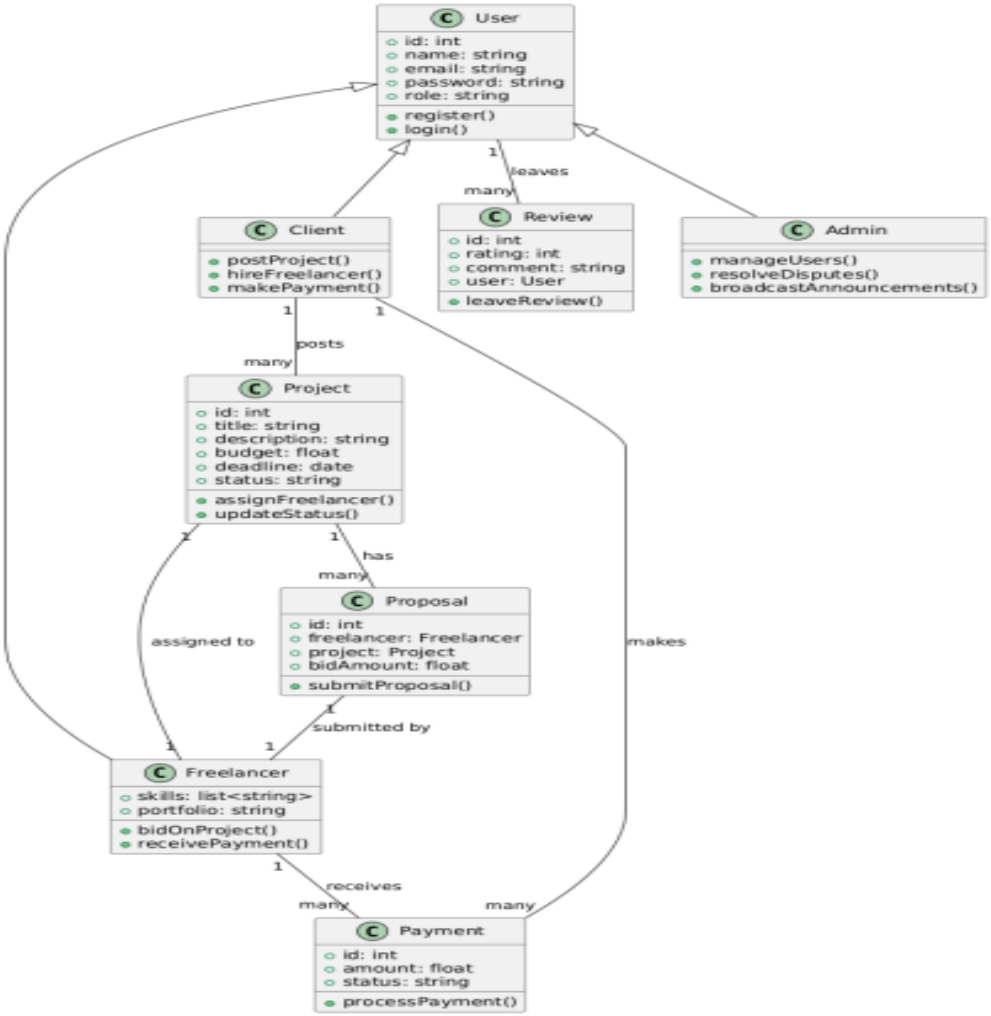


Fig 4.4.1 Class Diagram

5. TEST CASES

5.1 USER

5.2 Login

SR_NO	TEST CASE	FEATURE	DESCRIPTION	STEPS TO EXECUTE	EXPECTED RESULTS
1	TC-01	UI	Check all input fields and buttons	1. Open login page 2. Inspect layout	UI elements are aligned and responsive
2	TC-02	Required Fields	Leave fields blank	1. Leave username and password blank 2. Click "Login"	Error messages for required fields
3	TC-03	User Login	Valid username, invalid password	1. Enter correct username 2. Enter wrong password 3. Click "Login"	Error: Incorrect password
4	TC-04	User Login	Valid login	1. Enter correct credentials 2. Click "Login"	Redirect to dashboard

5.3 Apply a Project

SR_NO	TEST CASE	FEATURE	DESCRIPTION	STEPS TO EXECUTE	EXPECTED RESULTS
1	TC-01	Application Form	Apply without description	1. Open project page 2. Leave proposal field blank 3. Click "Apply"	Error message for blank proposal field
2	TC-02	Application Form	Valid application	1. Enter valid proposal 2. Click "Apply"	Application submitted and listed
1	TC-01	Application Form	Apply without description	1. Open project page 2. Leave proposal field blank 3. Click "Apply"	Error message for blank proposal field
2	TC-02	Application Form	Valid application	1. Enter valid proposal 2. Click "Apply"	Application submitted and listed

5.4 Post a Project

SR_NO	TEST CASE	FEATURE	DESCRIPTION	STEPS TO EXECUTE	EXPECTED RESULTS
1	TC-01	Required Fields	Check missing project details	1. Open "Post Project" 2. Leave fields blank 3. Click "Submit"	Error messages for all mandatory fields
2	TC-02	Project Creation	Post with valid inputs	1. Fill all details 2. Click "Submit" Project saved and listed on platform	Project saved and listed on platform

6. INPUT DESIGN AND OUTPUT DESIGN

6.1 INPUT DESIGN

The forms in Freelenso are designed to offer a smooth and intuitive experience for both freelancers and clients. The system enforces data validation, user guidance, and accessibility to ensure accurate and complete user input. Below are the key forms in the project:

6.1.1 Student Input Design:

Registration Form:

The registration form collects essential user information such as full name, email address, user type (freelancer or client), and a password. Freelancers may optionally upload a profile picture and provide a short bio or skill set. The form includes validations to ensure email uniqueness, strong password policies, and proper format of uploaded files. Clear error messages guide users to correct any mistakes during registration.

Login Form:

This form allows users to log into the system using their registered email. As passwords are not stored, an alternative authentication method (like OTP or token-based login) is implemented. If credentials are incorrect or missing, appropriate error messages such as “Invalid email or token” are shown. The system also includes a "Resend Token" option for accessibility and ease of login.

Project Posting Form:

Clients can post new freelance projects using this form. It collects details like project title, description, budget range, skills required, and deadlines. Optional attachments can be included for reference. The form ensures mandatory fields are validated and highlights any missing or incorrectly formatted inputs, helping clients post clear and effective job listings.

Application Form:

Freelancers apply to posted projects via this form. It includes a short proposal field, estimated delivery time, and bid amount. The form validates numeric input for the bid and

time, ensuring submissions meet client requirements. It also checks if the freelancer has already applied to avoid duplicate applications.

Chat & Milstone Forms:

The chat interface allows users to communicate in real-time, supporting both text and file uploads. All messages are logged securely. The milestone creation form enables clients to break projects into phases. Inputs include milestone name, amount, and deadline. Validation ensures proper structure and timeline clarity.6.1.2 HOD Input Design:

Wallet Transaction Form:

Users can perform transactions like adding funds, withdrawing earnings, and reviewing balance history. Inputs are verified to prevent invalid payment details and transaction errors. Confirmation prompts ensure secure operations.

Search & Filter Form:

To improve discoverability of projects and freelancers, the search form accepts input such as keywords, skills, budget, and project type. Dynamic filters enhance precision, and results update in real time, making the system fast and user-friendly.

6.2 OUTPUT DESIGN

- The output design of **Freelenso** focuses on providing users with relevant, accurate, and user-friendly information.
- Outputs are structured to be clear, concise, and easy to understand.
- Each output is tailored to match the user role—freelancer, client, or admin
- Dashboards are used to summarize key data like active projects, wallet balance, and notifications.
- All outputs are designed with responsiveness and accessibility in mind, suitable for both desktop and mobile devices.
- Real-time updates are incorporated for dynamic outputs such as chats, project status, and notifications.
- Error messages and success messages are shown clearly to guide users after actions like submitting proposals, creating milestones, or adding funds.
- Project listings are presented in card or table formats, displaying title, budget, deadline, and skill tags.

- Application lists allow clients to view freelancer proposals with names, bid amounts, estimated delivery time, and a short message.
- Chat outputs are displayed in a conversational layout with timestamps and file sharing previews.
- Milestone details are shown in a structured table format including name, amount, due date, and current status.
- Wallet outputs include transaction history with clear icons for credits and debits, and filters for better visibility.
- Admin panel outputs are visualized using cards, tables, and charts for statistics like total users, active jobs, and transaction summaries.
- Overall, the output system enhances the usability of the platform and helps users take informed actions.

7. SYSTEM IMPLEMENTATION

7.1 Introduction

The implementation phase of Freelenso – Freelance Website focuses on transforming the planned functionalities and design into a working web application. This phase includes building both the frontend and backend of the system using Django's MVT (Model-View-Template) architecture. The platform is developed to offer a smooth experience for different user types, namely: Admins, Clients, and Freelancers.

Freelenso is developed using the Django framework to take advantage of its secure authentication system, scalability, and clean separation of logic. Bootstrap and custom CSS are utilized for responsive design, while JavaScript provides interactivity. Django templates are used for rendering dynamic content.

Key aspects of implementation include:

- Code modularity and reusability
- Secure login and payment processing
- Real-time notification and status updates
- Escrow-based payment flow
- Thorough validation and error handling

Development Environment:

- Python 3.9
- Django 4.x (Backend)
- SQLite (Development DB) / PostgreSQL (Production)
- HTML5, CSS3, Bootstrap 5 (Frontend)
- JavaScript (Interactivity)

7.2 System Components

Backend (Django):

- Uses Django's MVT pattern
- Implements user roles (Client, Freelancer, Admin)
- Includes business logic for project posting, bidding, and payments
- Authentication handled via Django's auth system

Frontend:

- Developed using HTML, Bootstrap 5, and custom CSS
- Responsive user interfaces for each role
- Templates for dashboard, profiles, projects, and transactions
- Font Awesome used for UI icons
- JavaScript handles dynamic elements like live notifications and form validation

Database:

- SQLite used during development; PostgreSQL recommended for production
- Django ORM handles all interactions
- Models defined for Users, Profiles, Projects, Proposals, Transactions, Reviews, Reports

Static & Media Handling:

- Static files: CSS, JS, and images served via Django's static files framework
- Media files: Uploaded files (e.g., profile pictures) managed via Django media settings

7.3 Project Structure

The project follows Django's standard project structure, with a clear separation of concerns for the frontend, backend, and static resources. The structure is organized as follows:

```
freelenso/
├── manage.py
├── freelenso/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
│   └── asgi.py
```

```
├── core/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── views.py
│   ├── urls.py
│   ├── forms.py
│   ├── tests.py
│   ├── migrations/
│   │   └── __init__.py
│   ├── static/
│   │   ├── css/
│   │   ├── js/
│   │   └── images/
│   └── templates/
│       ├── client/
│       ├── freelancer/
│       ├── admin/
│       ├── shared/
│       └── base.html
```

7.4 Database Design

7.4.1 User

Field Name	Data Type	Constraints	Description
user	INT	PRIMARY KEY, AUTO_INCREMENT	Connects profile to a single User account
is_freelance	BOOLEAN	DEFAULT FALSE	user is a freelancer
is_client	BOOLEAN	DEFAULT FALSE	user is a client

profile_picture	Image field	NULL	URL to profile image

7.4.2 Project Attachment

Field Name	Data Type	Constraints	Description
attachment_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for the attachment
project_id	INT	FOREIGN KEY	Associated project
file_path	VARCHAR(255)	NOT NULL	File path or filename
uploaded_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Upload timestamp
attachment_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for the attachment
project_id	INT	FOREIGN KEY	Associated project

7.4.3 Project Application

Field Name	Data Type	Constraints	Description
application_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for the application
project_id	INT	NOT NULL	Linked project
freelancer_id	INT	NOT NULL	Freelancer applying
cover_letter	TEXT	NOT NULL	Application letter

proposed _budget	DECIMAL(10,2)	CHECK (proposed_budget >= 0)	Freelancer's proposed budget

7.4.4 Project Milestone

Field Name	Data Type	Constraints	Description
milestone_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each milestone
project_id	INT	NOT NULL	Linked project
title	VARCHAR(200)	NOT NULL	Milestone title
description	TEXT	NOTNULL	Milestone details

7.4.5 Project Activity

Field Name	Data Type	Constraints	Description
activity_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each activity
project_id	INT	NOT NULL	Associated project
user_id	INT	NOT NULL	User who performed the activity
activity_type	VARCHAR(50)	NOT NULL, Enum check for allowed types	Type of project-related activity
description	TEXT	NOTNULL	Additional details

created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Timestamp when the activity was recorded
activity_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each activity
project_id	INT	NOT NULL	Associated project
user_id	INT	NOT NULL	User who performed the activity
activity_type	VARCHAR(50)	NOT NULL, Enum check for allowed types	Type of project-related activity
description	TEXT	NOTNULL	Additional details
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Timestamp when the activity was recorded
activity_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each activity
project_id	INT	NOT NULL	Associated project
user_id	INT	NOT NULL	User who performed the activity
activity_type	VARCHAR(50)	NOT NULL, Enum check for allowed types	Type of project-related activity

7.4.6 Chatroom

Field Name	Data Type	Constraints	Description
room_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique chat room ID
project_id	INT	NOT NULL	Linked project
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Chat room creation timestamp
last_message_at	DATETIME	DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	Timestamp of last activity/message
room_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique chat room ID
project_id	INT	NOT NULL	Linked project
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Chat room creation timestamp

last_message_at	DATETIME	DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	Timestamp of last activity/message
room_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique chat room ID

7.4.7 Transaction

Field Name	Data Type	Constraints	Description
id	INT	Auto-generated primary key	PRIMARY KEY, AUTO_INCREMENT
transaction_id	VARCHAR(100)	Unique identifier for transaction	NOT NULL, UNIQUE
wallet_id	INT	Linked wallet	NOT NULL, FOREIGN KEY
project_id	INT	Linked project	FOREIGN KEY
milestone_id	INT	Linked milestone	FOREIGN KEY

amount	DECIMAL(10, 2)	Transaction amount	NOT NULL
fee_amount	DECIMAL(10, 2)	Platform fee	DEFAULT
transaction_type	VARCHAR	Type of transaction	NOT NULL
payment_method	VARCHAR	Method used for payment	NOT NULL,
status	VARCHAR	Status of the transaction	DEFAULT
description	TEXT	Optional description	Nullable
reference_id	VARCHAR	External payment	Nullable

		referen ce	
created_at	DATETIME	Creatio n timesta mp	DEFAULT CURRENT_TIMES TAMP
updated_at	DATETIME	Last update timesta mp	DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

7.4 URL Routing and Views

URL paths are modularized using app-specific urls.py files and linked through the main freelenso/urls.py for maintainability.

Example URLs:

- /login/ – User login (Freelancer or Client)
- /register/ – User registration
- /dashboard/ – User dashboard based on role
- /post-project/ – Client posts a new project
- /projects/ – Freelancers browse available projects
- /project/<id>/bid/ – Submit bid on a project
- /project/<id>/review/ – Client reviews bids
- /transactions/ – View payment history
- /profile/edit/ – Update user profile

Views:

- login_view() – Authenticates users

- register_view() – Handles account creation
- dashboard_view() – Role-based homepage
- post_project_view() – Clients post projects
- browse_projects_view() – Freelancers browse listings
- submit_proposal_view() – Freelancers bid on projects
- manage_bids_view() – Clients view and accept proposals
- payment_view() – Handles escrow payment
- review_view() – Allows rating after project completion

HTTP Methods:

- **GET:** Render pages like dashboards, project listings, profiles
- **POST:** Submit forms like login, registration, bids, reviews, payments

7.5 Templates and Frontend Integration

Templates:

- Django's template engine is used to render dynamic content such as forms, user data, and results.
- Conditional logic and loops manage role-based content display and dynamic UI updates.

Template Inheritance:

- base.html holds shared layout: header, footer, and navbar.
- Other pages like student_dashboard.html, nomination_form.html extend this base for UI consistency.

Static Files:

- Stored in static/ directory.
- Includes:
  - CSS for styling pages
  - JavaScript for interactive elements

- Images for user profiles and results display

7.6 User Authentication and Authorization

Authentication:

- All users (clients, freelancers) authenticate using Django's built-in User model.
- Admins have separate access via the Django admin panel.

Authorization & Access Levels:

- Role-based permissions use custom decorators and view-level logic.
- Unauthorized access is blocked and redirected with warning messages.

7.7 Forms and Validation

Django Forms:

Custom forms are created for login, registration, project posting, bidding, payments, and reviews.

Validation:

- **Required Fields:** All necessary fields (e.g., username, budget, deadline) must be filled
- **Email Validation:** Unique and correctly formatted email
- **Password Validation:** Minimum 8 characters, not too common, confirm match
- **Proposal Validation:** Ensure no duplicate bids and valid bid amount
- **Payment Validation:** Valid transaction data, escrow conditions met

Custom Error Messages:

Helpful feedback like:

- “Email already exists”
- “You’ve already submitted a proposal for this project”
- “Passwords do not match”

7.8 Business Logic and Core Functionality

Role-Based Logic:

- **Admin:** Full platform management, user monitoring, reporting
- **Client:** Can post projects, review bids, select freelancers, initiate payment
- **Freelancer:** Can browse projects, submit proposals, communicate with clients, receive payment

Core Functionalities:

- **User Management:** Registration, profile updates, role control
- **Project Lifecycle:**
  - Clients post projects with scope and deadline
  - Freelancers submit proposals
  - Clients select a freelancer and start work
- **Escrow Payment System:**
  - Clients deposit funds
  - Freelancers receive payment after work is marked completed
- **Review System:** Both users can leave reviews post-completion
- **Notifications:** Real-time alerts for bids, messages, and project updates

7.9 Testing and Debugging

Testing:

- **Unit Tests:** Verify model behavior for User, Project, Proposal, and Transaction
- **Functional Tests:** Test views like project creation, bid submission, user login
- **Integration Tests:** Ensure forms work together properly and routes are protected

Debugging:

- Django debug toolbar and browser developer tools used during testing
- Logging and exception handling added to catch runtime errors

Error Handling:

- Clear error pages for 404, 403, 500
- User input errors are caught and explained on forms

8. FUTURE ENHANCEMENTS

8. Future Enhancements

Future enhancements for Freelenso aim to make the platform more user-friendly, professional, and engaging for both freelancers and clients.

A **project sharing** feature will be introduced, allowing clients to share their project postings directly via platforms like LinkedIn, WhatsApp, or email. This will help attract a wider range of talented freelancers and increase platform visibility.

To improve accessibility, a **dark mode** option will be added, offering a visually comfortable experience for users who work during late hours or prefer a darker theme.
Advanced search functionality is another priority, including features like autocomplete, skill-based filtering, budget ranges, and sorting by deadline or popularity. This will make it easier for users to find suitable projects or freelancers quickly.
Community interaction will be improved by introducing **comment threads and replies** in project discussions and reviews, enabling more engaging and helpful communication between users.
An **offline mode** will be implemented, allowing users to view saved project details, chat history, or downloaded invoices without an internet connection, ensuring continuous access even in limited connectivity scenarios.
To make the platform more interactive and educational, **video profiles and tutorials** will be enabled for freelancers. This will allow them to showcase their work, explain their skills, or walk through case studies—helping clients make more informed hiring decisions.
These enhancements will significantly enrich the platform's capabilities, providing a more seamless, interactive, and professional experience for all users of **Freelenso**.

9. CONCLUSION

9. Conclusion

The **Freelenso** freelance platform has been designed and developed to bridge the gap between clients and freelancers by providing a secure, efficient, and user-friendly environment for collaboration. With features such as user registration, project postings, applications, milestone tracking, real-time chat, notifications, and a wallet system, the platform ensures smooth project execution and effective communication.

Throughout the development process, emphasis was placed on usability, functionality, and responsiveness to meet the needs of both freelancers and clients. The modular design allows for scalability, enabling the integration of future enhancements such as advanced search, dark mode, video profiles, and offline access.

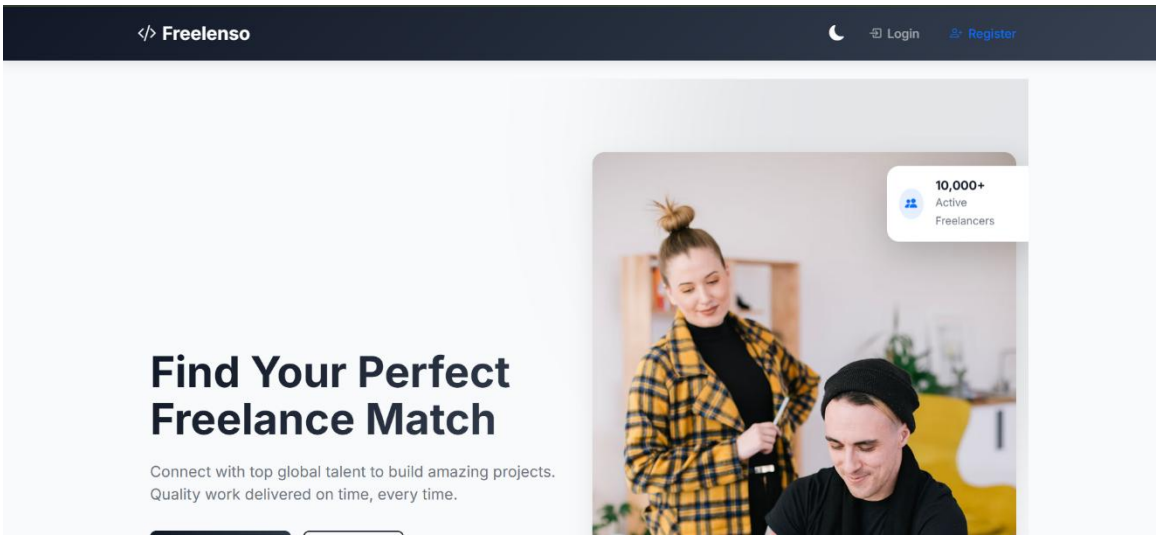
Freelenso not only simplifies the freelance hiring process but also promotes transparency, timely delivery, and quality outcomes. The platform serves as a complete freelance management system that is reliable, flexible, and adaptable to the evolving needs of the digital workforce.

In conclusion, **Freelenso** stands as a promising solution to modern freelance challenges, offering a professional ecosystem for managing freelance work effectively and efficiently.

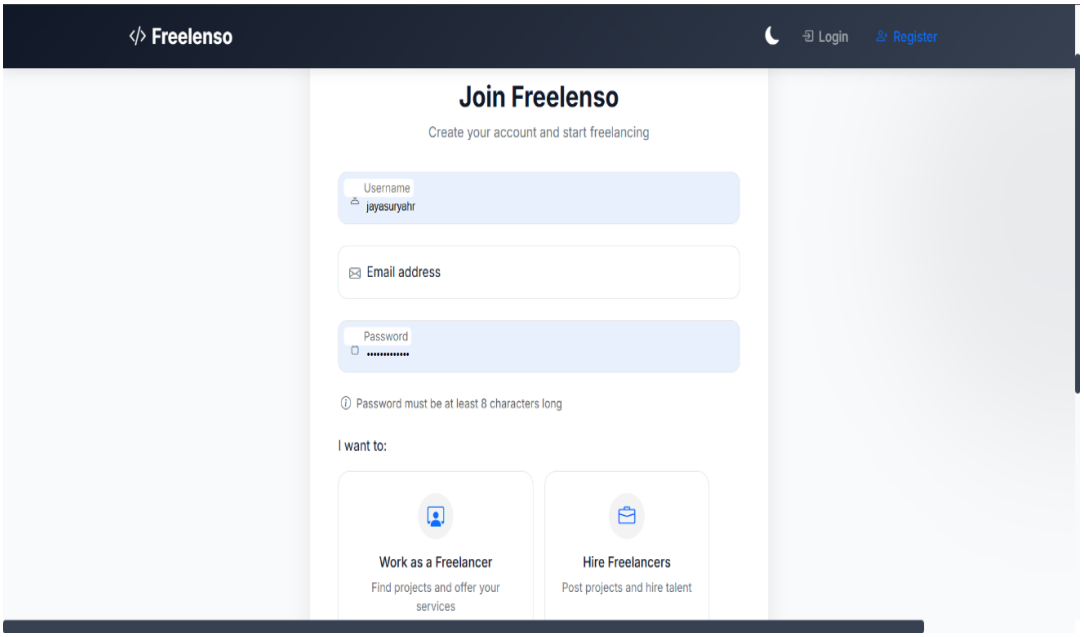
10. ANNEXURE

10. SCREENSHOTS

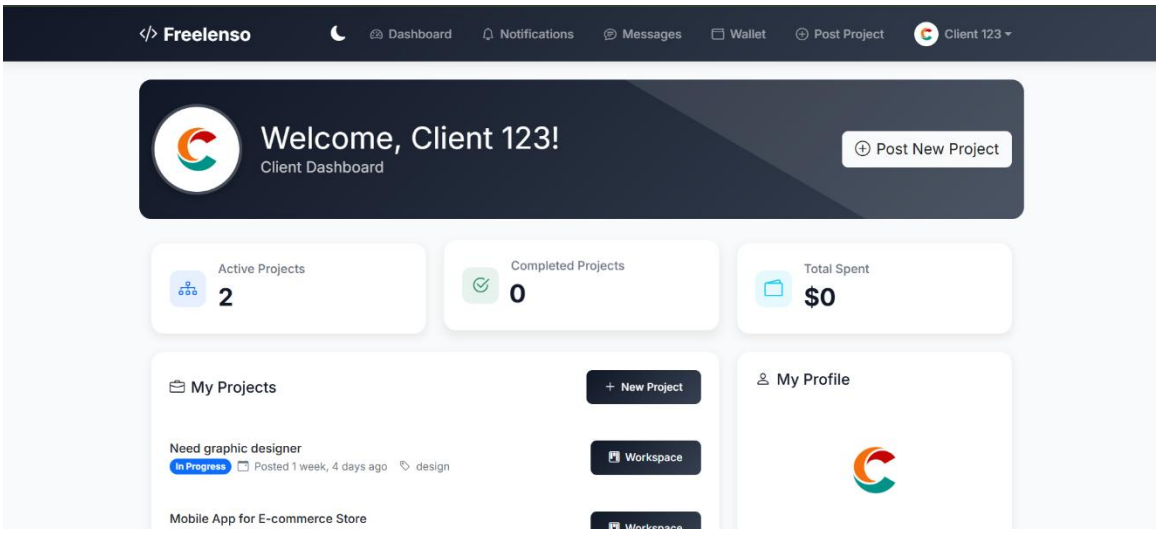
10.1 *Homepage*



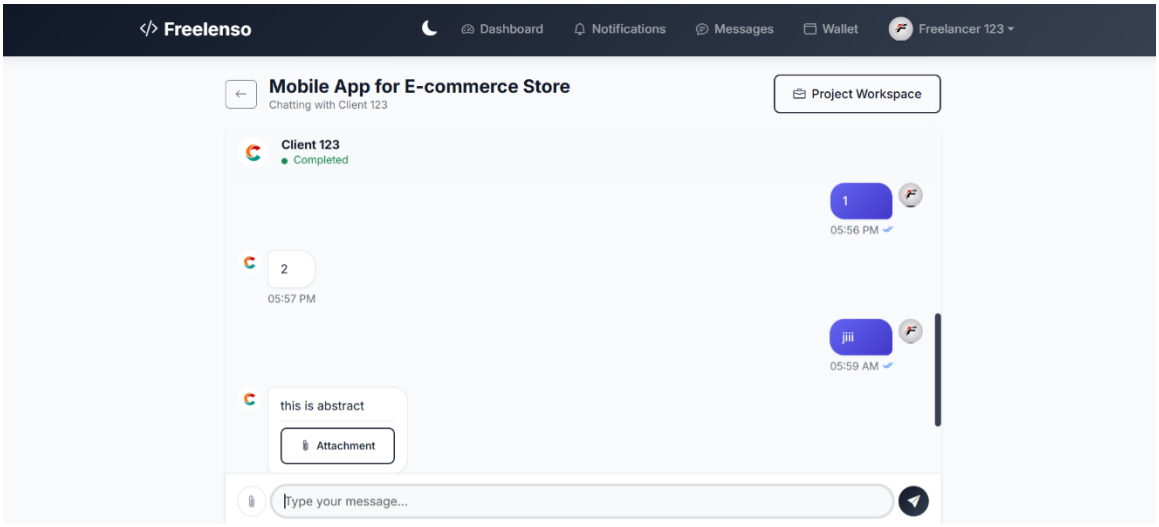
10.2 ***Registration page***



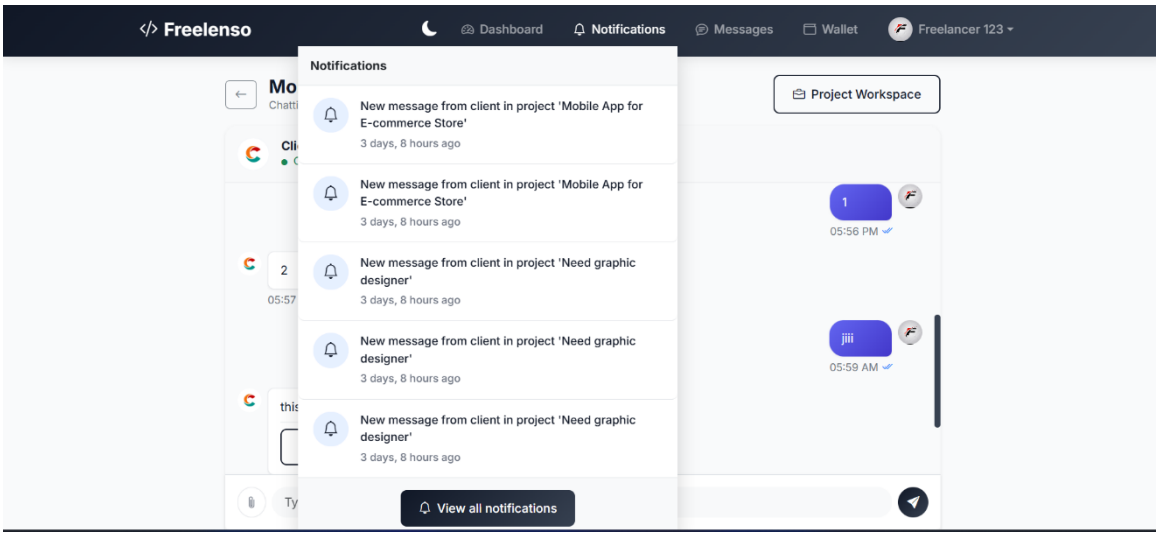
10.3 Client Dashboard



10.4 Chat Room page



10.5 Notification Page



10.6 wallet dashboard