

Project Report

On

“Crowdsourcing Platform-PROCONNECT”

SUBMITTED TO

ROURKELA INSTITUTE OF MANAGEMENT STUDIES

(As a Partial fulfilment of the requirement for the award of degree)

FOR

“MASTER IN COMPUTER APPLICATION “

(2023-25)

SUBMITTED BY

ANJANI SHAW

Registration Roll No:- 2305260004

MCA 4th SEMESTER

ROURKELA INSTITUTE OF MANAGEMENT STUDIES

(Affiliated to Biju Patnaik University of Technology, Odisha)

Rourkela – 769015



Rourkela Institute of Management Studies

Rourkela

Department of Computer Science

Rourkela Institute of Management Studies

Chhend, Rourkela-15, Odisha

Phone:0661 2480482

Fax:91-0661-1480665

Mail: rkl_rimsgrol@sancharnet.in

Visit: www.rims-edu.com

CERTIFICATE OF EXAMINATON

This is to certify that this project report entitled “CROWDSOURCING PLATFORM-PROCONNECT” submitted by ANJANI SHAW of 4th semester, Rourkela Institute of Management Studies, Rourkela, is accepted as partial fulfillment of requirements for the degree in Master in Computer Applications, under Biju Patnaik University of Technology, Rourkela this has been verified by us and found be original up to our satisfaction.

External Examiner



Rourkela Institute of Management Studies

Rourkela

Department of Computer Science

Rourkela Institute of Management Studies

Chhend, Rourkela-15, Odisha

Phone:0661 2480482

Fax:91-0661-1480665

Mail: rkl_rimsgrol@sancharnet.in

Visit: www.rims-edu.com

CERTIFICATE OF EXAMINATON

This is to certify that this project report entitled “CROWDSOURCING PLATFORM-PROCONNECT” submitted by ANJANI SHAW of 4th semester, Rourkela Institute of Management Studies, Rourkela, is accepted as partial fulfillment of requirements for the degree in Master in Computer Applications, under Biju Patnaik University of Technology, Rourkela this has been verified by us and found be original up to our satisfaction.

Internal Examiner



Rourkela Institute of Management Studies

Rourkela

Department of Computer Science Rourkela Institute of Management Studies Chhend, Rourkela-15, Odisha
Phone:0661 2480482

Fax:91-0661-1480665

Mail: rkl_rimsgrol@sancharnet.in Visit: www.rims-edu.com

CERTIFICATE

This is to certify that this project entitled “CROWDSOURCING PLATFORM” has been and submitted by ANJANI SHAW, M.C.A 2023-2025, Rourkela Institute of Management Studies, Rourkela, has been examined by us.

He is found fit and approved for the award of “Master in Computer Application “Degree.

To the best my knowledge this work has not been submitted for the award of any other degree.

I wish all success in his life.

DEAN ACADEMICRIMS, ROURKELA



Prof. Bibhudendu Panda Head of The Department, MCA

Rourkela Institute of Management Studies Rourkela.

CERTIFICATE

This is to certify that **ANJANI SHAW** student of **M.C.A, Rourkela Institute of Management Studies, Rourkela, Odisha** of Session 2023-2025 has completed the project successfully.

I wish all success in his life.

(Prof. Bibhudendu Panda)



DECLARATION

I, ANJANI SHAW, hereby declare that the project report entitled "CROWDSOURCING PLATFORM-PROCONNECT" is of my work. The above work I submitted to "Biju Patnaik University of Technology, Rourkela" for the award of "Master in Computer Applications" Degree.

To the best of my knowledge, this work has not been submitted or published anywhere for the award of any degree.

ANJANI SHAW

ACKNOWLEDGEMENT

I would like to express my gratitude to **Mr. RASHMI KANT DASH (JAVA TECHNOCRATE) AND MR. SMRUTI RANJAN NAYAK** for his guidance and support during the project work.

I am deeply indebted to **Rourkela Institute of Management Studies, Chhend, Rourkela**, for providing me an opportunity to undertake a project work entitled “CROWD SOURCING PLATFORM-PROCONNECT”.

I am grateful to my project guide **Prof. Bibhudendu Panda** without his guidance it would not have been possible on my part to complete the project.

I acknowledge the help and co-operation received from all my team members in making this project.

I consider myself fortunate that I have successfully completed this project; I acknowledge my sincere gratitude to all those works and ideas that had helped me in writing this project.

ANJANI SHAW

University Roll No: 2305260004 MCA (2023-2025)

Rourkela Institute of Management Studies, Rourkela.

ABSTRACT

The rise of the digital economy has significantly transformed the traditional job market, enabling professionals to work remotely and businesses to access global talent. This project aims to develop **ProConnect**, a feature-rich **crowdsourcing platform** designed to connect clients with skilled freelancers across various domains such as software development, graphic design, content writing, marketing, and more. The platform will provide an efficient, secure, and user-friendly environment for project collaboration, contract management, and payment processing.

ProConnect will offer a seamless experience for both clients and freelancers through **key features**, including **job posting, bidding mechanisms, project tracking, secure payments, rating & review systems**. An intelligent matching algorithm will recommend the most suitable freelancers for job postings based on their skills, expertise, and previous work history.

A **dedicated admin panel** will allow administrators to monitor platform activity, manage disputes, verify user identities, and oversee financial transactions. To further enhance communication, **ProConnect** will enable clients and freelancers to discuss project details efficiently within their dashboards.

Security and trust will be at the core of the platform, incorporating **identity verification and encrypted transactions** to prevent fraud and disputes. Additionally, the platform will leverage cloud-based infrastructure and modern web technologies to ensure scalability and high performance, even under heavy user traffic.

By providing a **reliable and transparent freelancing ecosystem**, **ProConnect** aims to bridge the gap between businesses seeking top talent and professionals looking for flexible, well-paid job opportunities. This platform aspires to redefine how freelancing is conducted in the digital age, fostering a collaborative and trustworthy work environment.

TABLE OF CONTENT

<u>Sl.No.</u>	<u>Title</u>	<u>Page No.</u>
1	FRONT PAGE	01
2	CERTIFICATION OF EXAINATION	02
3	CERTIFICATE EXTERNAL EXAMINAR	03
4	CERTIFICATE INTERNAL EXAMINAR	04
5	CERTIFICATE BY DEAN	05
5	DECLARATION	06
6	ACKNOWLEDGEMENT	07
6	ABSTRACT	08
7	TABLE OF CONTENT	9-10
7	CHAPTER 1 INTRODUCTION	11-19
8	CHAPTER-2 LITERATURE SURVEY	20-23
9	CHAPTER-3 PROJECT PLANNING	24-26
10	CHAPTER-4 ENVIRONMENT SETTINGS	27-38
11	CHAPTER-5 SELECTED SOFTWARE PLATFORM	39-42
12	CHAPTER-6 SELECTED DEVELOPMENT LIFE CYCLE	43-48
13	CHAPTER-7 METHODOLOGY	49-52
14	CHAPTER-8 SYSTEM DESIGN	52-59

15	CHAPTER-9 SECURITY AND AUTHENTICATION	60-133
16	CHAPTER-10 TESTING	134-150
17	CHAPTER-11 CONCLUSION AND LIMITATION	151-152
18	CHAPTER-12 CONCLUSION AND LIMITATIONS	153-154
19	CHAPTER-13 FUTURE SCOPE AND ENHANCEMENT	155-157
20	CHAPTER-14 REFRERANCES	158-159

CHAPTER-1

INTRODUCTION

Preface

In today's fast-evolving digital economy, freelancing and crowdsourcing have become essential components of the global workforce. Businesses seek skilled professionals to complete specialized tasks, while talented individuals look for flexible work opportunities. **ProConnect** is designed to bridge this gap by providing a seamless and efficient platform for both clients and freelancers.

Background

This project aims to revolutionize the way businesses and freelancers interact, creating an ecosystem that fosters productivity, trust, and success for all stakeholders.

ProConnect is an innovative **crowdsourcing and freelancing platform** designed to bridge the gap between businesses and skilled professionals. It provides a seamless and efficient environment where **clients** can post projects, **freelancers** can showcase their expertise, and **admins** can manage the overall platform operations.

Built with **Spring Boot (Java), HTML, and CSS**, ProConnect ensures a robust, secure, and scalable architecture, offering an intuitive user experience for all stakeholders. It streamlines job posting, bidding, payments, and communication, making the freelancing process more transparent and efficient.

The global workforce is undergoing a rapid transformation, with digital platforms playing a crucial role in redefining traditional employment models. The rise of freelancing has been fueled by technological advancements, increased internet accessibility, and the growing preference for flexible work arrangements. Businesses, ranging from startups to large enterprises, are increasingly leveraging **freelancing platforms** to access specialized talent on demand while reducing operational costs.

Freelancing platforms provide a **structured marketplace** where clients can post projects, and freelancers can bid based on their expertise and experience. These platforms help bridge the gap between talent supply and demand by offering a **transparent, efficient, and scalable** hiring process. However, many existing freelancing platforms face challenges such as **high commission fees, lack of trust, inefficient dispute resolution, and limited communication tools**.

To address these challenges, **ProConnect** aims to develop an **innovative and user-centric freelancing platform** that enhances collaboration, security, and efficiency. The platform will provide features such as **intelligent job matching, milestone-based payments, real-time messaging, and an escrow system**, ensuring a fair and seamless experience for both clients and freelancers.

Furthermore, with the increasing demand for **remote work and gig-based employment**, a well-structured freelancing platform like **ProConnect** can contribute to the **global digital economy**, offering opportunities for professionals to work from anywhere while helping businesses scale efficiently. By integrating modern web technologies and secure transaction methods, **ProConnect** aims to establish itself as a reliable and competitive freelancing marketplace, fostering professional growth and economic opportunities for users worldwide.

Understanding the Problem Statement:

The rise of freelancing platforms has created immense opportunities for businesses and independent professionals. However, several challenges exist that hinder the smooth operation of such platforms. ProConnect is designed to address these key issues and provide an efficient solution for clients, freelancers, and administrators.

Key Problems in Freelancing Platforms:

1. Lack of Trust and Transparency:
 - Clients often struggle to find reliable freelancers, while freelancers face difficulties in securing genuine projects and timely payments.
 - Fake profiles and job scams create an unreliable environment.
2. Inefficient Job Matching:
 - Clients may find it difficult to identify the right talent due to poor filtering mechanisms.
 - Freelancers face competition from an overwhelming number of bidders, making it harder to secure projects.
3. Payment and Dispute Resolution Issues:
 - Unclear payment terms and lack of escrow systems lead to delayed or failed payments.
 - Disputes between clients and freelancers often lack proper resolution mechanisms.
4. Communication Barriers:
 - Many freelancing platforms lack an efficient communication system, leading to misunderstandings and project delays.
 - Clients and freelancers require a secure and structured way to discuss project details, share updates, and collaborate.
5. Platform Management and Security Concerns:
 - Managing user profiles, job postings, and financial transactions requires a strong admin panel for better control and fraud prevention.
 - Security risks such as data breaches, spam, and fake users need to be effectively managed.

ProConnect as a Solution

ProConnect is developed to overcome these challenges by providing:

- Secure user authentication and verification to build trust.
- Efficient job-matching algorithms for better project allocation.
- A structured escrow-based payment system for secure transactions.
- A built-in messaging system to enhance communication.
- A robust admin panel for monitoring and managing platform activities.

By addressing these critical pain points, ProConnect aims to create a reliable, transparent, and efficient freelancing ecosystem that benefits all users.

The growing **gig economy** has transformed how businesses and professionals collaborate, with freelancing platforms playing a pivotal role in connecting skilled individuals with global opportunities. However, many existing platforms face challenges such as **trust issues, inefficient job allocation, delayed payments, and poor communication systems**. This motivated us to develop **ProConnect**, a freelancing and crowdsourcing platform that overcomes these limitations.

Key Factors Behind the Motivation:

1. Bridging the Gap Between Clients and Freelancers

- Many businesses struggle to find **reliable and skilled** freelancers, while professionals face difficulties securing legitimate projects.
- ProConnect aims to create a **trustworthy and efficient** environment where both parties can collaborate seamlessly.

2. Enhancing Security and Transparency

- Fake profiles, job scams, and payment frauds are common issues in freelancing.
- We aim to implement **secure authentication, profile verification, and escrow-based payments** to ensure transparency and reliability.

3. Providing a Fair and Competitive Platform

- Freelancers often compete with thousands of bidders, making it harder for talented individuals to stand out.
- ProConnect introduces **smart job-matching algorithms** to connect the right talent with the right projects efficiently.

4. Improving Communication and Collaboration

- Many platforms lack a **structured communication system**, leading to misunderstandings and project delays.
- We aim to integrate a **real-time chat system** to streamline discussions between clients and freelancers.

5. Building an Efficient Admin Management System

- Effective platform management requires strong administrative controls for **user monitoring, dispute resolution, and security enforcement**.
- Our **admin panel** will help in managing user activities, job postings, payments, and reviews efficiently.

Our Vision

With ProConnect, we envision a **secure, efficient, and user-friendly freelancing ecosystem** that empowers businesses to find top talent and enables freelancers to grow their careers without hurdles. This project is a step toward revolutionizing the freelancing industry by ensuring **fairness, security, and transparency** in online work collaborations.

Project overview and purpose:

ProConnect is a **crowdsourcing and freelancing platform** designed to connect **clients** with skilled **freelancers**, while allowing **admins** to manage platform operations efficiently. The platform provides a **secure, transparent, and efficient** environment for businesses to post projects, hire professionals, and manage work seamlessly.

Key Users of ProConnect:

- **Admin:** Oversees platform operations, manages users, jobs, payments, and ensures security.
- **Client:** Posts projects, reviews freelancer profiles, assigns tasks, and makes payments.
- **Freelancer:** Bids on projects, collaborates with clients, and delivers quality work.

Technologies Used:

ProConnect is developed using a **robust tech stack** to ensure scalability and security:

- **Backend:** Spring Boot (Java) for secure and efficient processing.
- **Frontend:** HTML and CSS for a user-friendly and responsive interface.
- **Database:** Structured data management for storing user profiles, job details, and transactions.

Core Features of ProConnect:

- User Authentication & Role-Based Access:** Secure login system for admins, clients, and freelancers.
- Job Posting & Bidding:** Clients post jobs, freelancers bid, and clients select the best match.
- Escrow-Based Payments:** Ensuring secure transactions and reducing payment disputes.
- Messaging System:** Real-time chat for smooth communication between clients and freelancers.
- Admin Dashboard:** Comprehensive control panel for managing users, jobs, payments, and reports.

Project Objectives

- ◆ **To develop a secure and efficient freelancing platform** that connects clients and freelancers.
- ◆ **To streamline the job-matching process** through an intuitive project posting and bidding system.
- ◆ **To ensure secure and timely payments** using an escrow-based payment system.
- ◆ **To enhance communication** between clients and freelancers with a built-in messaging system.
- ◆ **To empower admins** with tools to manage users, monitor transactions, and resolve disputes.
- ◆ **To provide a scalable and user-friendly solution** that supports future enhancements and business growth.

By achieving these objectives, **ProConnect** aims to **revolutionize the freelancing industry** by making it **transparent, reliable, and efficient** for all users.

Project Goals:

- ◆ To create a **trusted** and **efficient** freelancing platform.
- ◆ To **simplify** the hiring process for businesses and **enhance** work opportunities for freelancers.
- ◆ To implement **secure payments** and **efficient communication** features.
- ◆ To develop an **admin panel** for smooth platform management.

With **ProConnect**, we aim to **redefine freelancing** by ensuring fairness, transparency, and efficiency, making it easier for businesses and professionals to collaborate successfully.

Requirement Specification

The requirement specification outlines the functional and non-functional requirements for **ProConnect**, ensuring a structured and efficient development process.

A. Functional Requirements

These define the key features and functionalities of the platform:

1. User Authentication & Role Management

- Secure registration and login for **Admins, Clients, and Freelancers**.
- Role-based access control with distinct permissions.

2. Job Posting and Bidding System

- Clients can **post jobs** with detailed descriptions and budgets.
- Freelancers can **bid on projects** with proposals and expected timelines.
- Clients can review bids and **hire the most suitable freelancer**.

3. Payment Processing

- Secure **based payment** to ensure fair transactions.
- Automated release of payments upon project completion.
- Payment history tracking for both clients and freelancers.

4 . Admin Panel for Platform Management

- Dashboard for monitoring users, jobs, payments, and reviews.
- User verification and fraud prevention measures.
- Dispute resolution system for handling conflicts.

5 . Review & Rating System

- Clients and freelancers can rate and review each other after project completion.
- A reputation system to enhance trust and credibility.

6 . Reports and Analytics

- Job success rates, freelancer earnings, and platform statistics.

B. Non-Functional Requirements

These define the system's performance, security, and usability:

1. **Scalability** – The platform should handle an increasing number of users and projects.
 2. **Security** – Data encryption, secure login, and payment protection mechanisms.
 3. **Performance** – Fast response times for job searches, messaging, and transactions.
 4. **User-Friendly Interface** – Intuitive design for easy navigation.
 5. **Reliability** – Ensure system availability with minimal downtime.
-

Project Specification:

The project specification outlines the **technology stack** and **system architecture** used to develop ProConnect.

A. Technology Stack

Component	Technology Used
Backend	Spring Boot (Java)
Frontend	HTML, CSS
Database	MySQL
Authentication	Spring Security, JWT
Payment Processing	Stripe / PayPal API
Version Control	Git / GitHub

B. System Architecture

1. **Client-Side (Frontend)**
 - Developed using **HTML and CSS** for a responsive interface.
 - Handles **user interactions, job postings, and messaging**.
2. **Server-Side (Backend)**
 - Built using **Spring Boot (Java)** for processing business logic.
 - Manages **user authentication, job matching, payments, and communication**.

3. Database Layer

- MySQL to store user profiles, job data, and transactions.
- Optimized queries for fast data retrieval.

4. Third-Party Integrations

- Payment Gateway (Stripe/PayPal) for secure transactions.
- Cloud Storage for storing user files and documents.

C. Deployment Strategy

- The platform will be deployed on a cloud server (AWS/Digital Ocean) for scalability.
- Continuous Integration and Deployment (CI/CD) using GitHub Actions.

Software Specification

The **Software Specification** defines the tools, technologies, and frameworks used to develop **ProConnect**, ensuring a structured and efficient development process.

A. Software Requirements

1. Development Environment

- **Operating System:** Windows 10/11, Linux (Ubuntu), macOS
- **IDE:** IntelliJ IDEA, VS Code
- **Version Control:** Git & GitHub
- **Build Tool:** Maven / Gradle
- **Dependency Management:** Spring Boot Starter Dependencies

2. Backend Technologies

- **Programming Language:** Java (Spring Boot Framework)
- **Framework:** Spring Boot (for MVC architecture and backend logic)
- **Database:** MySQL / PostgreSQL (for storing users, jobs, and payments)
- **Security:** Spring Security, JWT Authentication (for secure access control)
- **REST API Development:** Spring Boot REST Controller

3. Frontend Technologies

- **Languages:** HTML, CSS
- **UI Frameworks:** Bootstrap (for responsive design)
- **JavaScript (Optional Enhancement):** For dynamic UI interactions

4. Database Management

- **DBMS:** MySQL / PostgreSQL
- **ORM Tool:** Hibernate (for database interaction)
- **Database Backup:** Automated SQL dumps for data security

5. Payment Integration

- **Payment Gateway:** Stripe / PayPal API for secure transactions
- **Escrow System:** Funds held securely until project completion

6. Hosting & Deployment

- **Cloud Hosting:** AWS, Digital Ocean, or Firebase
- **Web Server:** Apache Tomcat (for hosting Spring Boot application)
- **Containerization (Optional):** Docker for scalable deployments

7. Third-Party APIs & Libraries

- **Logging & Monitoring:** Log4j, Spring Actuator
 - **Email Notifications:** SMTP / Spring Mail
 - **Cloud Storage (Optional):** AWS S3 / Firebase Storage
-

B. System Requirements

1. Minimum Hardware Requirements

Component	Specification
Processor	Intel i7 / AMD Ryzen 5 or higher
RAM	8GB (Recommended: 16GB)
Storage	50GB free space (For database and application files)
Internet	Stable connection for hosting and API integration

2. Software Requirements

Component	Software
OS	Windows 10/11, Ubuntu 20.04+, macOS
JDK	Java Development Kit (JDK 11 or above)
Database	MySQL 8.0 / PostgreSQL
Web Server	Apache Tomcat 9+

Component	Software
Browser	Chrome, Firefox, Edge

Conclusion

The software specifications ensure that ProConnect is built using modern, scalable, and secure technologies. The combination of Spring Boot, MySQL, HTML, and CSS ensures a high-performance and user-friendly freelancing platform.

CHAPTER-2

LITRATURE REVIEW

The **literature review** provides an overview of existing freelancing platforms, their challenges, and how **ProConnect** aims to improve upon them. It explores previous research, industry trends, and technological advancements in the freelancing ecosystem.

1. Introduction to Freelancing Platforms

Freelancing platforms have revolutionized the global workforce by enabling professionals to work remotely and businesses to access a diverse talent pool. Popular platforms such as **Upwork**, **Freelancer**, **Fiverr**, and **Toptal** have established themselves as industry leaders. These platforms provide an **online marketplace** where clients post jobs, freelancers bid on projects, and transactions occur securely.

Despite their success, freelancing platforms face significant **challenges**, including **payment security, job authenticity, unfair bidding, high commission rates, and communication gaps**. Several studies have analyzed these issues and proposed improvements, which ProConnect aims to implement.

2. Existing Freelancing Platforms and Their Challenges

Platform	Strengths	Challenges
Upwork	Large talent pool, secure contracts, job tracking tools	High service fees, competitive bidding makes it hard for new freelancers
Freelancer	Wide range of job categories, milestone-based payments	Spam job postings, delayed payments
Fiverr	Quick gig-based hiring, user-friendly interface	Limited flexibility in project scope, high commission
Toptal	Exclusive high-quality freelancers, strict screening process	Expensive for clients, limited opportunities for new freelancers

Research indicates that these platforms lack **personalized job matching, transparent fee structures, efficient dispute resolution, and real-time collaboration tools**. These shortcomings serve as the foundation for **ProConnect's** development.

3. Research Studies on Freelancing Platforms

Several academic and industry studies highlight key trends in freelancing:

- **Trust and Security Issues:**

- A study by **Kittur et al. (2013)** on online labor markets found that **scams and fake job postings** reduce freelancer trust in platforms.
 - ProConnect addresses this through **strict user verification, escrow-based payments, and an admin-moderated environment**.
- **Job Matching and Bidding Challenges:**
 - Research by **Lehdonvirta et al. (2018)** suggests that the **bidding process often favors low-cost freelancers**, making it difficult for skilled professionals to earn fair wages.
 - ProConnect introduces **smart job-matching algorithms** that connect clients with the most relevant freelancers based on **skills, experience, and reviews**.
- **Payment Disputes and Commission Issues:**
 - Studies by **Malik & Wahaj (2020)** emphasize that **delayed payments and high service fees** frustrate freelancers.
 - ProConnect mitigates this with **escrow payments, fair commission policies, and automated dispute resolution mechanisms**.
- **Communication Gaps Between Clients and Freelancers:**
 - A report by **Singh et al. (2019)** highlights that many platforms lack **effective collaboration tools**, leading to project delays and misunderstandings.
 - ProConnect integrates a **real-time chat system**, improving **workflow transparency and project coordination**.

4. Technological Advancements in Freelancing Platforms

Recent technological innovations have shaped freelancing platforms:

- **Artificial Intelligence (AI) in Job Matching:**
 - AI-powered algorithms improve **job recommendations** and **automate candidate selection**.
 - ProConnect employs **AI-driven job-matching** to enhance project allocation efficiency.
- **Blockchain for Secure Payments:**
 - Blockchain ensures **transparent and tamper-proof transactions**.
 - While ProConnect starts with **escrow-based payments**, it has the potential to integrate **blockchain-based contracts** in future iterations.
- **Cloud-Based Infrastructure for Scalability:**
 - Cloud computing enables freelancing platforms to **scale efficiently** as user demand increases.
 - ProConnect leverages **AWS/Digital Ocean** for hosting and data storage.

5. ProConnect's Contribution and Innovations

Based on the review of existing platforms and research studies, ProConnect introduces **several enhancements**:

1. **User Verification & Fraud Prevention** – Ensuring **legitimate users and genuine job postings** through strict verification processes.
2. **Smart Job Matching System** – AI-based recommendations for **better freelancer-client pairing**.
3. **Escrow-Based Payments** – Secure **funds release only after project approval**, reducing fraud risks.
4. **Real-Time Communication Tools** – Integrated **chat system for direct freelancer-client interaction**.
5. **Fair & Transparent Commission Structure** – **Lower fees** compared to competitors, promoting freelancer satisfaction.
6. **Comprehensive Admin Panel** – Efficient monitoring of **user activities, payments, and dispute resolution**.

CHAPTER-3

PROJECT PLANNING

Project Planning

Project planning is a crucial phase in the development of **ProConnect**, ensuring that tasks are organized, resources are allocated efficiently, and timelines are met. This section outlines the **development methodology, phases, timeline, and risk management strategies** for the project.

1.9.1 Development Methodology

ProConnect follows the **Agile Software Development Methodology**, specifically the **Scrum framework**, due to its iterative approach, flexibility, and focus on user feedback.

Key Agile Principles Applied in ProConnect:

1. **Incremental Development:** Features are developed in iterations (sprints).
2. **Continuous Feedback:** Regular updates and feedback ensure improvements.
3. **Prioritized Backlog:** Features are developed based on priority and feasibility.
4. **Collaborative Approach:** Regular team discussions, stand-up meetings, and sprint reviews.

1.9.2 Project Development Phases

The project is divided into **six major phases**, each with well-defined deliverables.

Phase 1: Requirement Analysis & Planning (Week 1-2)

Objective: Define the project's functional and non-functional requirements.

Tasks:

- Conduct market research on freelancing platforms.
- Define user roles (Admin, Client, Freelancer).
- Prepare Software Requirement Specification (SRS).

Deliverables:

- Requirement Specification Document
- Initial Project Plan

Phase 2: System Design (Week 3-4)

Objective: Create a blueprint of the system architecture.

Tasks:

- Design database schema for users, jobs, payments, and reviews.
- Develop wireframes and UI/UX designs for the website.
- Finalize system flow diagrams and APIs.

Deliverables:

- System Architecture Diagram
- Database Schema
- Wireframes & UI Designs

Phase 3: Backend & Database Development (Week 5-7)

Objective: Build the core logic, data handling, and APIs using **Spring Boot**.
Tasks:

- Develop user authentication and role-based access control.
- Implement database connections and ORM (Hibernate).
- Develop job posting, bidding, and messaging APIs.

Deliverables:

- Secure Authentication System
- REST APIs for User Management, Job Posting, and Bidding
- Database Setup

Phase 4: Frontend Development (Week 8-10)

Objective: Implement a **responsive UI** using HTML, CSS, and Bootstrap.
Tasks:

- Develop client and freelancer dashboards.
- Implement job posting and application interfaces.
- Integrate messaging system and notification panel.

Deliverables:

- Interactive UI with navigation
- Job Posting & Application Modules
- Integrated Chat & Notification System

Phase 5: Integration & Testing (Week 11-12)

Objective: Ensure smooth functionality, security, and performance of the platform.
Tasks:

- Integrate frontend with backend APIs.
- Conduct unit testing, integration testing, and user acceptance testing (UAT).
- Optimize for security, performance, and scalability.

Deliverables:

- Fully Integrated System
- Test Reports and Bug Fixes

Phase 6: Deployment & Maintenance (Week 13-14)

Objective: Deploy the application and ensure continuous maintenance.
Tasks:

- Host the platform on AWS/Digital Ocean.
- Implement SSL encryption and security patches.
- Provide post-deployment support and performance monitoring.

Deliverables:

- ✓ Live
 - ✓ Security & Production
 - ✓ Ongoing Maintenance Plan Deployment Optimization
-

1.9.3 Project Timeline (Gantt Chart Representation)

Phase	Week 1-2	Week 3-4	Week 5-7	Week 8-10	Week 11-12	Week 13-14
Requirement Analysis	✓					
System Design		✓				
Backend Development			✓			
Frontend Development				✓		
Integration & Testing					✓	
Deployment & Maintenance						✓

1.9.4 Risk Management

To ensure smooth execution, potential risks are identified along with mitigation strategies:

Risk Factor	Impact	Mitigation Strategy
Scope Creep	High	Define a clear scope and prioritize core features.
Security Threats	High	Implement Spring Security, JWT authentication, and encrypted payments.
Delayed Deliverables	Medium	Follow Agile sprints and set clear deadlines.
Database Overload	Medium	Optimize queries and use caching techniques.
Technical Issues	Medium	Perform frequent testing and integration reviews.

1.9.5 Resource Allocation

Resource	Role	Responsibilities
Project Manager	Oversees development	Planning, tracking progress, resolving roadblocks
Backend Developer	Develops server-side Spring Boot development, API creation, database logic	management

Resource	Role	Responsibilities
Frontend Developer	Designs UI/UX	HTML, CSS, Bootstrap development
Database Administrator	Manages database	Database design, optimization, and backups
QA Engineer	Tests software	Unit testing, integration testing, UAT
DevOps Engineer	Manages deployment	Cloud hosting, security, and monitoring

1.9.6 Tools & Technologies Used

Category	Tool/Technology
Version Control	Git, GitHub
Development Frameworks	Spring Boot (Java), HTML, CSS
Database	MySQL / PostgreSQL
Testing	JUnit, Postman (API Testing)
Deployment	AWS, Digital Ocean
Security	Spring Security, JWT Authentication

CHAPTER-4

ENVIRONMENTAL SETUP

1. VS CODE
2. GIT
3. GITHUB
4. JAVA
5. HTM,CSS,JAVASCRIPT
6. SPRING BOOT
7. Maven (MVC)

Developed any Project need a IDLE (Integrated Development Environment), so Developer can easily Create, Run, Compile and Execute Programs.

1.How to Install Visual Studio Code on Windows?

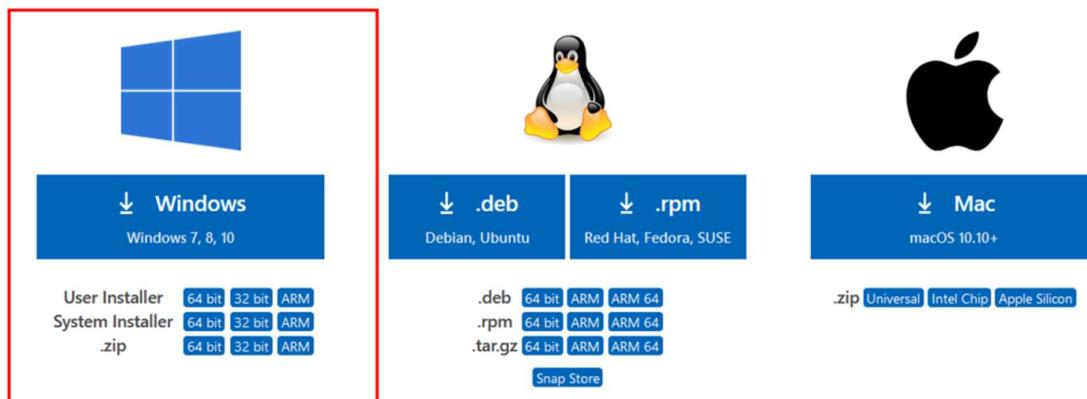
Firstly, download the Visual Studio Code installer for Windows. Once it is downloaded, run the installer (*VSCodeUserSetup-{version}.exe*). It will only take a minute.

Visit: <https://code.visualstudio.com/download>

Download Visual Studio Code

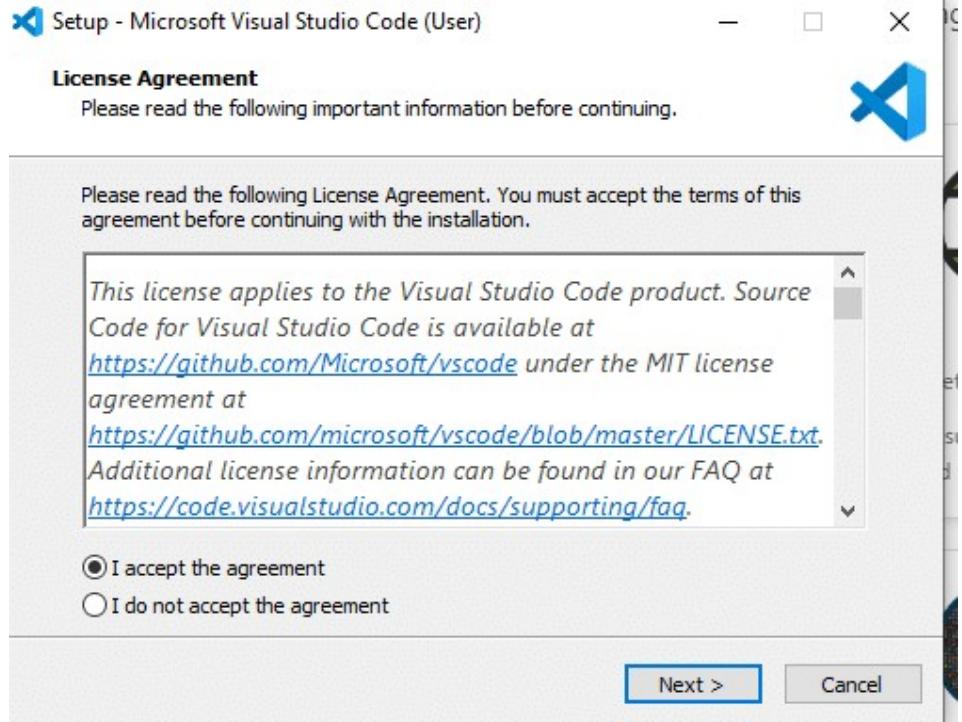
Free and built on open source. Integrated Git, debugging and extensions.

How to Install Visual Studio Code on Windows and Run Python Programs

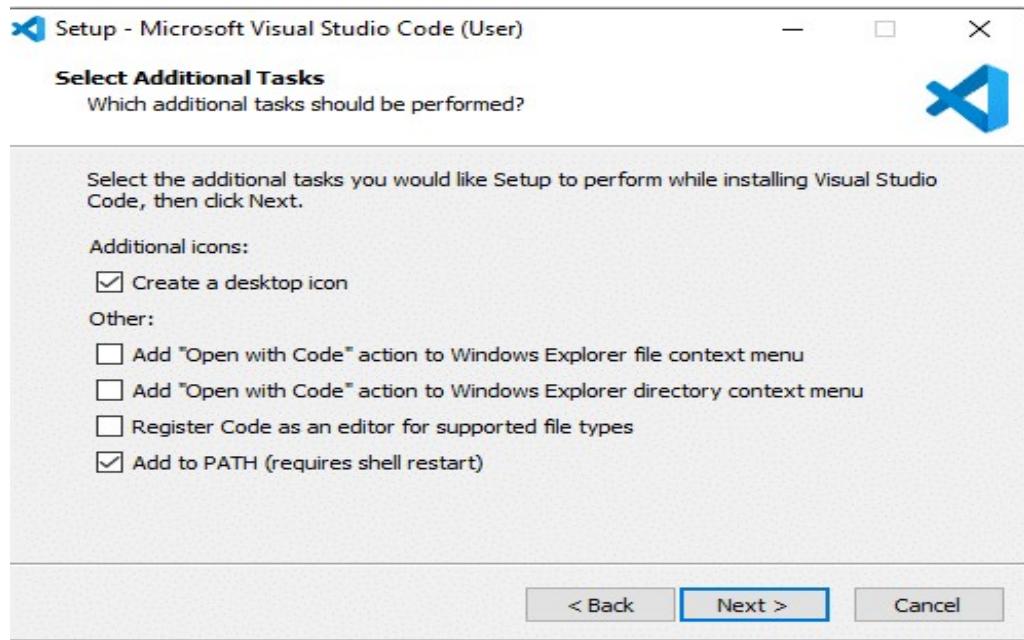


Then double Click on Download folder .exe file and Follow the step by step instruction Process .

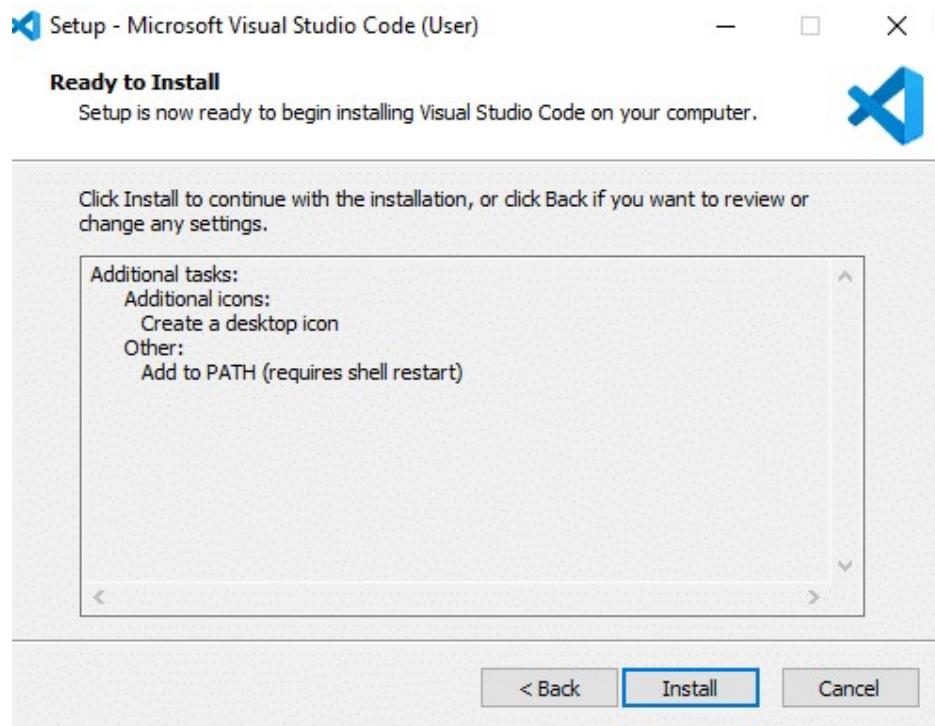
Secondly, accept the agreement and click on next.



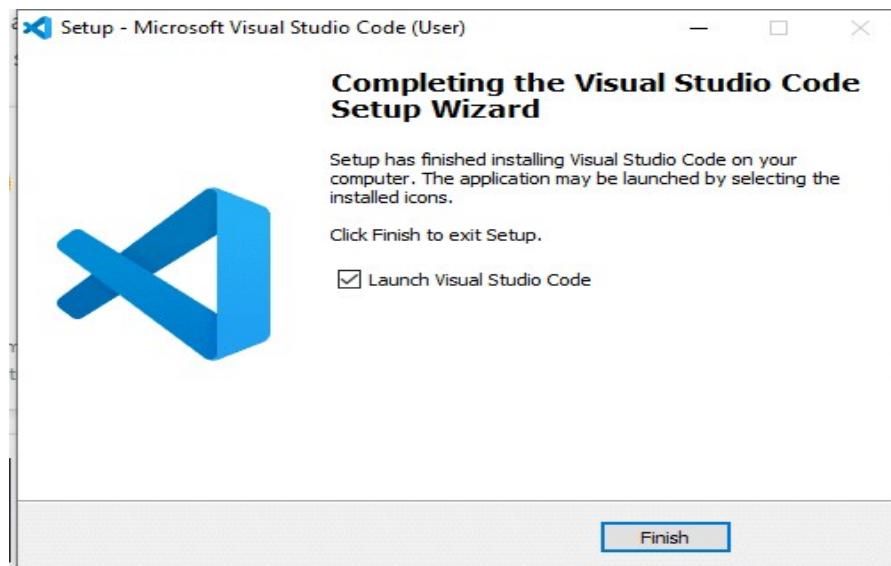
Thirdly, click on “create a desktop icon” so that it can be accessed from desktop and click on Next.



After that, click on the install button.



Finally, after installation completes, click on the finish button, and the visual studio code will get open.



2.Git and GitHub download steps:

Steps to Download and Install Git & GitHub on Windows

Step 1: Download Git

1. Open your browser and go to the **official Git website**: <https://git-scm.com/downloads>.
2. Click on **Windows** (Git will automatically detect your OS).
3. The **Git setup file (.exe)** will start downloading.

Step 2: Install Git

1. Locate the downloaded .exe file and **double-click** to open it.
2. Click **Next** to continue the installation.
3. Choose the installation folder (default is recommended) and click **Next**.
4. Select **components** (keep the default selection) and click **Next**.
5. Select the **default editor for Git** (choose Notepad++ or VS Code if installed) and click **Next**.
6. Choose **Git from the command line and 3rd-party software** (default option) and click **Next**.
7. Select **Use the OpenSSL library** for HTTPS connections and click **Next**.
8. Keep the default settings for line endings and click **Next** until installation begins.
9. Once installed, click **Finish** to complete the process.

Step 3: Verify Git Installation

1. Open **Command Prompt** (Press Win + R, type cmd, and hit Enter).
2. Type the following command and press Enter:

```
git --version
```

3. If installed successfully, it will display the installed Git version.

Steps to Download and Install GitHub Desktop

Step 1: Download GitHub Desktop

1. Go to the **official GitHub Desktop website**: <https://desktop.github.com/>.
2. Click on **Download for Windows** (or Mac, depending on your OS).

Step 2: Install GitHub Desktop

1. Locate the downloaded .exe file and **double-click** to install.
2. Follow the **on-screen instructions** and click **Finish**.

Step 3: Sign In to GitHub

1. Open **GitHub Desktop**.
2. Click **Sign in to GitHub.com** and enter your GitHub credentials.

- Once signed in, you can start managing repositories.

Next Steps

- Create a GitHub Repository:** Click **File > New Repository** to create a new repo.
- Clone an Existing Repository:** Click **Clone a Repository** and paste the GitHub link.
- Use Git Commands** in Command Prompt or Git Bash.

3. How to Download and Set Up Java in VS Code

Since you already have **VS Code installed**, follow these steps to download and configure Java for development.

Step 1: Install Java (JDK)

- Download Java JDK**
 - Visit: <https://www.oracle.com/java/technologies/javase-downloads.html>.
 - Click **Download** for the latest JDK version.
 - Select the appropriate installer for **Windows/Linux/Mac**.
- Install Java JDK**
 - Open the downloaded installer and follow the setup wizard.
 - Accept the **license agreement** and click **Next** until installation completes.
- Verify Java Installation**
 - Open **Command Prompt** (Win + R, type cmd, and press Enter). `java -version`
 - If installed successfully, it should show the Java version (e.g., java version "17.0.2").

Step 2: Install Java Extension Pack in VS Code

- Open **VS Code**.
- Click on **Extensions** (Ctrl + Shift + X).
- Search for "**Java Extension Pack**".
- Click **Install** (This installs Java support, debugging, and Maven tools).
- Restart VS Code after installation.

Step 3: Configure Java in VS Code

- Open VS Code and go to **Settings** (Ctrl + ,).
- Search for **Java: Home** and ensure the correct JDK path is set.
- Open **Terminal** (Ctrl + ~) and type:

```
javac -version
```

- If it shows a Java compiler version, the setup is complete.
-

Step 4: Create and Run a Java Program in VS Code

1. Create a New Java File

- Open VS Code.
- Click **File > New File** and save it as HelloWorld.java.

2. Write a Simple Java Program

3. Run the Program

Download and Install Maven (Build Tool)

Spring Boot uses **Maven** or **Gradle** for project management.

1. Download Maven

- Visit: <https://maven.apache.org/download.cgi>.
- Download the latest **Binary zip** file.

2. Install Maven

- Extract the zip file to C:\Program Files\Apache\Maven.
- Add C:\Program Files\Apache\Maven\bin to **System Environment Variables (Path)**.

3. Verify Maven Installation

- Open **Command Prompt** and type:

```
mvn -version
```

- If installed, it will display the Maven version.
-

Step 3: Create a Spring Boot Project

1. Open **VS Code**.
2. Press **Ctrl + Shift + P** to open the Command Palette.
3. Type "**Spring Initializr: Generate a Maven Project**" and select it.
4. Choose the following settings:
 - **Language:** Java
 - **Spring Boot Version:** Latest Stable Version
 - **Project Type:** Maven
 - **Group:** com.example

- **Artifact:** proconnect
 - **Dependencies:** Spring Web, Spring Boot DevTools, Spring Data JPA, MySQL Driver
5. Click **Generate Project**, extract the ZIP file, and open it in **VS Code**.
-

Step 4: Install Spring Boot Extension Pack

1. In **VS Code**, click on **Extensions** (Ctrl + Shift + X).
 2. Search for "**Spring Boot Extension Pack**".
 3. Click **Install**.
 4. Restart VS Code after installation.
-

Step 5: Run the Spring Boot Application

1. Open **Terminal** (Ctrl + ~) in **VS Code**.
2. Navigate to the project folder:
cd proconnect
3. Run the Spring Boot application using:
mvn spring-boot:run
4. If successful, you will see:
Tomcat started on port 8080.

4. How to Set Up a Spring Boot Maven MVC Project in VS Code

Spring Boot MVC (Model-View-Controller) is used for building **web applications** with a structured pattern. Follow these steps to create an **Maven-based Spring Boot MVC project in VS Code**.

Step 1: Prerequisites

Before setting up the project, ensure you have the following installed:

<input checked="" type="checkbox"/>	Java	JDK	(17 or later)	→	Download JDK
<input checked="" type="checkbox"/>	Apache	Maven	→	Download Maven	
<input checked="" type="checkbox"/>	VS Code with Java Extensions				

Step 2: Create a Spring Boot MVC Project with Spring Initializr

1. **Open VS Code**.
2. **Press Ctrl + Shift + P** to open the Command Palette.
3. Type "**Spring Initializr: Generate a Maven Project**" and select it.

4. Choose the following options:

- **Language:** Java
- **Spring Boot Version:** Latest Stable Version
- **Project Type:** Maven
- **Group:** com.example
- **Artifact:** mvc-demo
- **Packaging:** jar
- **Java Version:** 17+

5. **Add Dependencies:**

- Spring Web
- Thymeleaf (for Views)
- Spring Boot DevTools (for Live Reload)

6. Click **Generate**, extract the ZIP file, and open it in **VS Code**.

Step 3: Project Structure

After creating the project, the structure will look like this:

```
mvc-demo
|--- src
|   |--- main
|   |   |--- java/com/example/mvcdemo
|   |   |   |--- controller
|   |   |   |   |--- HomeController.java
|   |   |   |--- MvcDemoApplication.java
|   |   |--- resources
|   |   |   |--- templates (for Thymeleaf)
|   |   |   |   |--- index.html
|   |   |   |--- static (for CSS/JS)
|   |   |   |--- application.properties
|--- pom.xml
```

Step 4: Create a Controller

Inside src/main/java/com/example/mvcdemo/controller, create a new file: **AdminController.java**

Step 5:Create a View (HTML Page)

1. Inside src/main/resources/templates/, create a file **index.html**.
2. Add the following Thymeleaf-based HTML code:

Step 6: Configure application.properties

1. Open src/main/resources/application.properties and add:
properties

Step 7: Run the Spring Boot MVC Application

1. Open **Terminal** in VS Code (Ctrl + ~).

2. Navigate to the project folder:

```
cd mvc-demo
```

3. Run the application:

```
mvn spring-boot:run
```

4. Open a browser and go to:

```
http://localhost:1211/
```

You should see the **Welcome Message** from index.html! 

5. MySQL Download and Database Management:

1. MySQL Download & Installation

Step 1: Download MySQL

1. Visit the official MySQL website: <https://dev.mysql.com/downloads/>.
2. Click on **MySQL Community Server** (which is free).
3. Select your operating system (Windows, macOS, Linux).
4. Download the installer (either **MSI Installer** for Windows or **DMG** for macOS).

Step 2: Install MySQL

1. Run the installer and choose **Custom Installation** (recommended) or **Default Installation**.
2. Select **MySQL Server**, **MySQL Workbench**, and **MySQL Shell** for installation.
3. Set up a root password when prompted.
4. Complete the installation process.

Step 3: Verify Installation

- Open **Command Prompt (Windows) / Terminal (macOS/Linux)** and run:

```
mysql --version
```

If MySQL is installed correctly, it will display the version.

2. Setting Up MySQL Database

Step 1: Open MySQL Server

- If you installed **MySQL Workbench**, open it and connect using **root** credentials.
- If using the **command line**, run:

```
mysql -u root -p
```

Enter your root password when prompted.

Step 2: Create a New Database

To create a database for the ProConnect project, run:

```
CREATE DATABASE proconnect;
```

To use this database:

```
USE proconnect;
```

Step 3: Create Tables

Example: Creating a **Users** table with id, name, and email

```
CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL
);
```

MySQL Download and Database Management

1. What is MySQL?

MySQL is an open-source relational database management system (RDBMS) used to store and manage structured data. It is widely used in web applications due to its scalability, security, and performance.

2. MySQL Download & Installation

To use MySQL for your project, you need to install the **MySQL Community Server**, which includes the database server and command-line tools, along with **MySQL Workbench**, a graphical user interface for managing databases.

Installation Steps:

1. **Download** MySQL from the official website.

2. **Install MySQL** by selecting the required components (MySQL Server, Workbench, and MySQL Shell).
 3. **Configure MySQL** by setting up a root password and choosing the default server settings.
 4. **Verify Installation** by checking the MySQL service status and running basic queries.
-

3. Database Management in MySQL

Once MySQL is installed, you can create and manage databases using either the **MySQL Command Line** or **MySQL Workbench**.

Key Tasks in Database Management:

- **Creating a Database:** You need to set up a database to store project data.
 - **Creating Tables:** Tables store data in rows and columns, organized by data type.
 - **Inserting, Updating, and Deleting Data:** You can add, modify, or remove records from tables.
 - **Retrieving Data:** Queries allow you to fetch specific records from the database.
-

4. Connecting MySQL to Spring Boot

For backend integration, you will connect MySQL with **Spring Boot** using JDBC (Java Database Connectivity) or JPA (Java Persistence API). This allows your application to interact with the database seamlessly.

Configuration Steps:

1. **Add MySQL Connector** to your Spring Boot project.
 2. **Configure Database Properties** in the application settings.
 3. **Create Entity Classes** to define the database schema in Java.
 4. **Use Repository Layer** to interact with the database through queries.
-

5. MySQL Workbench for GUI-Based Management

MySQL Workbench is a **graphical tool** that allows users to visually create, manage, and query databases without using the command line.

Features of MySQL Workbench:

- **Database Design & Modeling:** Create database schemas visually.
- **SQL Query Execution:** Run queries and analyze results.
- **User & Privilege Management:** Assign permissions to users securely.

- **Data Export & Import:** Backup and restore database files easily.
-

6. Backup and Restore in MySQL

To ensure data security, MySQL provides options to **export** and **import** databases.

Backup Options:

- **Manual Export** using MySQL Workbench or command-line tools.
- **Automated Backups** scheduled at regular intervals.

Restoration Methods:

- **Import Data** from backup files.
 - **Recover Tables** in case of accidental deletion.
-

7. Why MySQL for ProConnect?

MySQL is ideal for the **ProConnect freelancing platform** due to:

- **Scalability:** Handles multiple user interactions efficiently.
- **Security:** Supports user authentication and access control.
- **Performance:** Optimized for high-speed transactions.
- **Integration:** Works seamlessly with **Spring Boot**, the backend framework for ProConnect.

CHAPTER-5

SELECTED SOFTWARE PLATFORMS

1. VS Code (Visual Studio Code)

- **Overview:** A lightweight, cross-platform source code editor developed by Microsoft. It supports a wide range of programming languages, including Java, JavaScript, and HTML/CSS, and has excellent support for Spring Boot development through extensions.
- **Key Features:**
 - Intelligent code completion (IntelliSense).
 - Built-in Git support for version control.
 - Debugging capabilities.
 - Extensions for Java, Spring Boot, and Thymeleaf.
 - Integrated terminal for running code and commands.

2. Git & GitHub

- **Git:**
 - **Overview:** A distributed version control system (VCS) to manage source code history. Git helps track changes in your project, manage different versions of files, and collaborate on code with other developers.
 - **Key Features:**
 - Version control allows you to save code snapshots at various stages.
 - Branching to work on multiple features in parallel.
 - Merging changes made in different branches.
 - **Git Commands:**
 - `git init`: Initialize a new Git repository.
 - `git commit`: Record changes in the repository.
 - `git push`: Upload changes to a remote repository (e.g., GitHub).
 - `git pull`: Fetch changes from the remote repository.
- **GitHub:**
 - **Overview:** A cloud-based platform built on top of Git to host repositories and collaborate on code. It supports version control, issue tracking, and CI/CD pipelines.
 - **Key Features:**
 - Hosting repositories with version control.

- Collaboration through pull requests and code reviews.
- GitHub Actions for automating workflows (CI/CD).

3. HTML (HyperText Markup Language)

- **Overview:** The standard markup language for creating web pages. HTML defines the structure of a web page using elements like headings, paragraphs, tables, forms, and links.
- **Key Features:**
 - **Elements:** Tags like `<h1>`, `<p>`, `<a>`, `` to define web page content.
 - **Attributes:** Provide additional information about elements (e.g., `href` for links).
 - **HTML5:** The latest version of HTML with new elements such as `<article>`, `<section>`, and `<footer>` for semantic structure.

4. CSS (Cascading Style Sheets)

- **Overview:** A style sheet language used to describe the presentation of HTML documents. CSS controls the layout, colors, fonts, spacing, and overall visual design of a web page.
- **Key Features:**
 - **Selectors:** Define which elements to style. E.g., `h1`, `.header`, `#id`.
 - **Properties:** Define what to change about the element. E.g., `color`, `font-size`, `margin`.
 - **Responsive Design:** Media queries allow styling for different screen sizes (e.g., mobile vs. desktop).
 - **CSS Frameworks:** Tools like Bootstrap and Tailwind CSS help create responsive and well-styled pages quickly.

5. JavaScript (JS)

- **Overview:** A client-side scripting language that makes web pages interactive. It allows manipulation of HTML elements, handling events, and creating dynamic user interfaces.
- **Key Features:**
 - **DOM Manipulation:** JavaScript can modify the Document Object Model (DOM) to change the structure and content of web pages.
 - **Event Handling:** JavaScript listens for user interactions (clicks, form submissions, etc.) and responds accordingly.
 - **AJAX:** Asynchronous JavaScript allows for dynamic content updates without reloading the entire page.

6. Java

- **Overview:** A high-level, object-oriented programming language used for backend development. It's platform-independent, thanks to the Java Virtual Machine (JVM), and widely used for web applications, Android apps, and enterprise-level applications.
- **Key Features:**

- **Object-Oriented:** Supports classes, objects, inheritance, polymorphism, and encapsulation.
- **Cross-Platform:** Java programs run on any system with the JVM installed.
- **Robust:** Strong memory management, exception handling, and garbage collection make Java programs reliable.

7. Spring Boot MVC (Model-View-Controller)

- **Overview:** A framework within the Spring ecosystem that simplifies Java web development by providing built-in configurations and reducing boilerplate code. Spring Boot makes it easy to set up a Spring application with embedded servers (like Tomcat) and minimal configuration.
- **Key Features:**
 - **MVC Pattern:** Separates the application into three components—Model (data), View (UI), and Controller (business logic).
 - **Auto-Configuration:** Spring Boot automatically configures components based on the project's dependencies.
 - **Embedded Server:** Spring Boot uses embedded servers like Tomcat, so you don't need to deploy the app on an external server.

8. Thymeleaf

- **Overview:** A modern templating engine for Java-based applications, especially integrated with Spring Boot. It is used to create dynamic web pages by rendering HTML templates.
- **Key Features:**
 - **Server-Side Rendering:** Thymeleaf allows dynamic generation of HTML by embedding Java expressions inside the template.
 - **HTML5 Compliant:** Thymeleaf templates are well-formed HTML files, which can be opened directly in a browser for testing.
 - **Conditional Logic:** Supports loops, conditions, and variables to manipulate the HTML view.

9. Hibernate & Entity Classes

- **Overview:** Hibernate is an Object-Relational Mapping (ORM) framework that simplifies database interactions by mapping Java objects to database tables.
- **Entity Classes:** These are Java classes marked with `@Entity`, representing a table in the database. Hibernate automatically handles the mapping between Java objects and database records.
- **Key Features:**
 - **CRUD Operations:** Automatically generates SQL queries for Create, Read, Update, and Delete operations.
 - **Lazy Loading:** Hibernate loads data only when it is needed, improving performance.

- **Transaction Management:** Hibernate provides built-in support for transactions.

10. Application Properties (`application.properties`)

- **Overview:** This is a configuration file used by Spring Boot to store application-level settings, such as database configurations, server settings, and custom properties.
- **Key Features:**
 - **Database Configuration:** Define the datasource, URL, username, and password for database connections.
 - **Server Configuration:** Set server port, context path, and other server-related properties.
 - **Custom Properties:** You can define custom application-specific settings here.

11. Security Config & Authentication

- **Overview:** **Spring Security** is a framework for securing Java web applications by implementing authentication (verifying identity) and authorization (defining user roles).
- **Key Features:**
 - **Authentication:** Verifies the identity of a user via login, OAuth, or LDAP.
 - **Authorization:** Grants or denies access to resources based on roles (e.g., Admin, User).
 - **JWT (JSON Web Tokens):** Used for token-based authentication in Spring Boot apps.
 - **CSRF Protection:** Prevents cross-site request forgery attacks.

These tools and frameworks provide a comprehensive environment for full-stack Java web development. With **Spring Boot MVC** handling the backend, **Thymeleaf** rendering views, and **Hibernate** managing database operations, you're well-equipped to build secure, dynamic, and scalable web applications.

CHAPTER-6

SYSTEM DESIGN

System Design for ProConnect Freelancing Platform

The **system design** phase outlines how different components of the ProConnect platform interact, ensuring smooth communication between **Admins, Clients, and Freelancers**. This section includes:

1. **Use Case Diagram** – showing user interactions with the system.
2. **ER Diagram** – depicting the database structure and relationships between entities.

1. Use Case Diagram

A **Use Case Diagram** illustrates how users (Admin, Client, and Freelancer) interact with various features of the freelancing platform. It helps in understanding system functionalities from a user perspective.

Actors in the System:

1. **Admin**
 - Manages users, jobs, payments, and disputes.
 - Generates reports and oversees system activities.
2. **Client**
 - Posts job requirements.
 - Reviews freelancer profiles and selects a candidate.
 - Makes payments after job completion.
3. **Freelancer**
 - Browses available jobs and submits proposals.
 - Communicates with clients via chat.
 - Completes assigned jobs and receives payments.

Main Use Cases:

- **User Authentication** (Register, Login, Logout)
- **Job Management** (Post, Apply, Approve, Track Status)
- **Payment Processing** (Escrow, Transaction, Withdrawal)
- **Chat System** (Real-time Messaging)
- **Review & Rating** (Feedback System)

2. ER Diagram (Entity-Relationship Diagram)

Purpose of ER Diagram

The ER Diagram defines the database structure of the platform by representing entities, attributes, and relationships. It ensures data integrity, relationships, and constraints.

1. User Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for users
Name	VARCHAR(255)	Full name of the user
Email	VARCHAR(255)	Email address (unique)
password	VARCHAR(255)	Encrypted password
Role	ENUM	User role (Admin, Client, Freelancer)
createdAt	TIMESTAMP	Account creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

2. Client Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for client
userId	INT (FK)	Foreign key from User table
companyName	VARCHAR(255)	Name of the client's company
contactNumber	VARCHAR(20)	Contact number
Location	VARCHAR(255)	Client's location
createdAt	TIMESTAMP	Account creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

3. Freelancer Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for freelancer
userId	INT (FK)	Foreign key from User table
Skills	TEXT	List of skills

Column Name	Data Type	Description
experience	INT	Years of experience
portfolioLink	VARCHAR(255)	Freelancer's portfolio URL
Rating	DECIMAL(3,2)	Average rating
createdAt	TIMESTAMP	Account creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

4. Profile Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for profile
userId	INT (FK)	Foreign key from User table
Bio	TEXT	Short bio of the user
profilePicture	VARCHAR(255)	Profile image URL
resumeLink	VARCHAR(255)	Resume/CV link
socialLinks	TEXT	Social media links
createdAt	TIMESTAMP	Profile creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

5. Project Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for project
clientId	INT (FK)	Foreign key from Client table
Title	VARCHAR(255)	Title of the project
description	TEXT	Project description
Budget	DECIMAL(10,2)	Budget allocated for the project
deadline	DATE	Project deadline
Status	ENUM	Project status (Open, Completed)

Column Name	Data Type	Description
createdAt	TIMESTAMP	Project creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

6. ProjectRequest Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for request
projectId	INT (FK)	Foreign key from Project table
freelancerId	INT (FK)	Foreign key from Freelancer table
Status	ENUM	Request status (Pending, Approved)
message	TEXT	Message from freelancer to client
createdAt	TIMESTAMP	Request creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

7. ServiceRequest Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for request
clientId	INT (FK)	Foreign key from Client table
freelancerId	INT (FK)	Foreign key from Freelancer table
serviceDetails	TEXT	Description of service request
Status	ENUM	Status of request (Approved, Pending)
createdAt	TIMESTAMP	Request creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

8. Category Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for category

Column Name	Data Type	Description
Name	VARCHAR(255)	Category name (e.g., Web Development, Design)
description	TEXT	Category description
createdAt	TIMESTAMP	Category creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

9. Payment Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for payment
clientId	INT (FK)	Foreign key from Client table
freelancerId	INT (FK)	Foreign key from Freelancer table
Amount	DECIMAL(10,2)	Payment amount
transactionDate	TIMESTAMP	Date and time of transaction
Status	ENUM	Payment status (Pending, Completed)
createdAt	TIMESTAMP	Payment creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

10. ContactMessage Table

Column Name	Data Type	Description
Id	INT (PK)	Unique identifier for message
senderId	INT (FK)	Foreign key from User table (Sender)
receiverId	INT (FK)	Foreign key from User table (Receiver)
message	TEXT	Message content
timestamp	TIMESTAMP	Time when the message was sent

11. Auditable Table

This is a **base table** used for tracking timestamps across multiple entities.

Column Name	Data Type	Description
createdAt	TIMESTAMP	Record creation timestamp
updatedAt	TIMESTAMP	Last update timestamp

Relationships Between Entities

1. A Client can post multiple Jobs.
2. A Freelancer can submit multiple Proposals for different Jobs.
3. A Job can have multiple Proposals from different Freelancers.
4. A Payment is linked to a Job upon completion.
5. Clients and Freelancers can exchange Messages.
6. Clients can leave Reviews for Freelancers after job completion.

CHAPTER-7

METHODOLOGY

Methodology for ProConnect Freelancing Website

The methodology outlines the development approach, techniques, models, and tools used in creating ProConnect, a freelancing platform. This structured approach ensures efficient project execution, error handling, and iterative improvements.

1. Development Methodology

Agile Methodology (SCRUM)

The project follows the Agile SCRUM methodology due to its flexibility, incremental progress, and continuous feedback loop. Agile ensures that the platform is developed in small, manageable iterations called sprints, allowing regular updates and improvements.

Agile Process Flow:

1. Requirement Gathering → Identify system needs from clients, freelancers, and admin.
2. Sprint Planning → Divide the project into smaller tasks and assign priorities.
3. Design & Development → Implement each sprint with frontend, backend, and database integration.
4. Testing & Debugging → Each sprint undergoes rigorous testing to ensure error-free functionality.
5. Deployment & Review → Deploy functional modules and gather stakeholder feedback.
6. Repeat Iterations → Improve and refine the project in subsequent sprints.

2. System Architecture

MVC (Model-View-Controller) Architecture

The project follows the MVC (Model-View-Controller) architecture, ensuring structured, scalable, and maintainable development.

1. Model (Backend & Database)
 - Defines business logic and database operations.
 - Uses Spring Boot, Hibernate ORM, and MySQL.
2. View (Frontend)
 - Handles **UI and user interaction** using **HTML, CSS, JavaScript, and Thymeleaf**.
3. Controller (Logic & Routing)
 - Manages **request handling, authentication, and processing** using **Spring Boot MVC**.

3. Software Development Life Cycle (SDLC)

The SDLC model ensures a step-by-step approach to development. The project follows a 7-stage SDLC model:

SDLC Stages:

1. Planning
 - Define project scope and feasibility.
 - Identify user roles: Admin, Client, Freelancer.
2. Requirement Analysis
 - Functional requirements (user registration, job posting, bidding system).
 - Non-functional requirements (scalability, security, performance).
3. Design
 - Create Use Case Diagrams, ER Diagrams, Wireframes.
 - Define system architecture and data flow.
4. Implementation (Coding & Development)
 - Develop backend with Java Spring Boot MVC & Hibernate.
 - Design frontend with HTML, CSS, JavaScript, and Thymeleaf.
5. Testing
 - Conduct Unit Testing, Integration Testing, User Acceptance Testing (UAT).
6. Deployment
 - Deploy on a cloud server (AWS, DigitalOcean, Heroku).
 - Ensure production readiness.
7. Maintenance & Updates
 - Regular security updates and bug fixes.
 - Improve based on user feedback.

4. Tools & Technologies Used

Category	Tools Used
Code Editor	Visual Studio Code
Version Control	Git, GitHub
Frontend Development	HTML, CSS, JavaScript, Thymeleaf
Backend Development	Java, Spring Boot MVC
Database	MySQL, Hibernate ORM

Category	Tools Used
Security	Spring Security, Authentication & Authorization
Testing	JUnit, Postman (for API testing)
Deployment	AWS / Digital Ocean / Heroku

5. Testing Methodology

Testing is **critical** to ensure the freelancing platform is **error-free and reliable**.

Types of Testing Conducted:

1. **Unit Testing**
 - Tests individual components (e.g., **login module, job posting system**).
2. **Integration Testing**
 - Tests interactions between **frontend, backend, and database**.
3. **System Testing**
 - Evaluates overall system performance.
4. **User Acceptance Testing (UAT)**
 - Clients and freelancers test the platform for **usability and functionality**.
5. **Security Testing**
 - Ensures **data encryption, authentication, and protection** against attacks.

6. Project Workflow

Step-by-Step Project Execution Flow

1. **User Authentication** (Signup/Login using Spring Security).
2. **Client Dashboard** (Post projects, hire freelancers).
3. **Freelancer Dashboard** (Search jobs, bid, get hired).
4. **Payment System** (Secure transactions and withdrawal options).
5. **Messaging System** (Chat feature for clients & freelancers).
6. **Admin Panel** (Manage users, jobs, transactions, reviews).

7. Deployment Strategy

The deployment process includes:

1. Code Management → Push updates to GitHub.
2. Build & Test → Ensure system passes all tests.
3. Deploy to Cloud → AWS / Digital Ocean / Heroku.

4. Monitor & Maintain → Performance tracking and updates.

The ProConnect Freelancing Platform follows an Agile methodology with a structured SDLC approach. Using MVC architecture, it integrates secure authentication, job posting, freelancer hiring, and payments efficiently. Testing ensures a robust and user-friendly experience, while cloud deployment makes the platform scalable.

CHAPTER-8

SYSTEM ANALYSIS

The **System Analysis** phase is crucial for understanding the **ProConnect Freelancing Platform**, ensuring efficient design, implementation, and performance. This chapter defines the **proposed system, web application structure, user requirements, admin functionalities, and inventory control**.

1. Proposed System

The **ProConnect Freelancing Platform** is an online system that connects **clients and freelancers**, enabling seamless project management and collaboration.

Objectives of the Proposed System

- Provide a user-friendly freelancing platform for **clients** to hire freelancers and **freelancers** to offer services.
- Ensure secure authentication and authorization using **Spring Security**.
- Enable real-time project management with functionalities like **job posting, bidding, and payments**.
- Facilitate communication between clients and freelancers through a **chat system**.
- Ensure smooth financial transactions using **secure payment gateways**.

Core Features

1. **Client Panel**
 - Dashboard for **managing projects**.
 - Ability to **post jobs, review freelancer proposals, and hire**.
 - Secure **payment and contract handling**.
2. **Freelancer Panel**
 - Dashboard for **browsing and bidding on projects**.
 - Profile creation to **showcase skills and past work**.
 - Secure **payment withdrawal options**.
3. **Admin Panel**
 - Manage **users, projects, and transactions**.
 - Monitor platform activity for **security and compliance**.

2.Web Application

The **ProConnect platform** is built as a **web-based application** using **Spring Boot MVC**, providing a robust and scalable architecture.

Technology Stack

Component	Technology
Frontend	HTML, CSS, JavaScript, Thymeleaf
Backend	Java, Spring Boot MVC
Database	MySQL (with Hibernate ORM)
Security	Spring Security (Authentication & Authorization)
Version Control	Git & GitHub

Hosting (Optional) AWS, DigitalOcean, Heroku

Architecture of the Web Application

- **Client-Side:** Uses **HTML, CSS, and JavaScript** for UI.
- **Server-Side:** Uses **Spring Boot** for business logic and request handling.
- **Database Layer:** Uses **MySQL with Hibernate ORM** to store **user data, projects, and transactions**.

3.User Requirement

The system supports three user types: **Clients, Freelancers, and Admins**.

Functional Requirements

1. **Clients**
 - Register and log in securely.
 - Post projects and manage proposals.
 - Hire freelancers and track project progress.
 - Make secure payments and review freelancers.

2. **Freelancers**
 - Register and create a **professional profile**.
 - Browse available projects and submit proposals.
 - Manage ongoing projects and communicate with clients.
 - Receive payments after project completion.

3. Admin

- Manage **user registrations, approvals, and bans.**
- Monitor and regulate **project postings.**
- Oversee **financial transactions.**
- Generate reports on **platform performance.**

Non-Functional Requirements

- **Security:** Data protection and encrypted authentication.
- **Scalability:** Efficient handling of high user traffic.
- **Performance:** Fast response time for page loading and user actions.

4. Admin Panel

The Admin Panel enables ProConnect administrators to monitor, manage, and maintain the platform.

Key Features

1. User Management
 - Approve or disable client and freelancer accounts.
 - Verify identity and service quality.
2. Project Oversight
 - Review project postings and remove fraudulent listings.
3. Transaction Monitoring
 - Oversee payments and withdrawals.
 - Resolve payment disputes.
4. Security Management
 - Implement fraud detection and spam control.
 - Enforce terms and policies.
5. Reports & Analytics
 - Generate user activity, earnings, and engagement reports.

5. Inventory Control (For Resource Management)

Although ProConnect does not handle physical inventory, it manages digital resources, such as:

- Database Records: User profiles, project listings, payments.
- Digital Assets: Freelancer portfolios, attachments, and documents.

Inventory Control Strategies

1. Data Storage & Optimization

- Efficient database queries for fast response time.
 - Regular data backups to prevent loss.
2. Server Resource Allocation
- Dynamic load balancing for high-traffic handling.
 - Cloud hosting solutions for scalability.
3. Performance & Security Monitoring
- Real-time tracking of server performance.
 - Regular audits to maintain data integrity.

The System Analysis outlines the key functionalities of the ProConnect Freelancing Platform, ensuring secure project management, smooth financial transactions, and efficient user interaction. The web application's robust architecture and admin controls make it a reliable and scalable freelancing solution.

CHAPTER-9

SECURITY AND AUTHENTICATION

Security Configuration and Authentication

Security is a critical aspect of the ProConnect freelancing platform as it involves sensitive data such as user profiles, projects, and financial transactions. The platform uses Spring Security to ensure authentication, authorization, and data protection.

1. Security Configuration in ProConnect

Spring Security is used for handling **user authentication and role-based access control (RBAC)**. The configuration is defined in a **SecurityConfig.java** class, which includes:

Key Security Features

1. User Authentication

- Uses **JWT (JSON Web Token) authentication** for secure login.
- Passwords are **hashed using BCrypt** for security.

2. Authorization and Role Management

- **Role-Based Access Control (RBAC):**
 - **Admin:** Full control over users, projects, and transactions.
 - **Client:** Can post projects, hire freelancers, and make payments.
 - **Freelancer:** Can bid on projects, complete tasks, and receive payments.
- Routes are protected based on roles (e.g., only admins can access /admin/**).

3. Session and Token Management

- Uses **JWT tokens** instead of traditional sessions.
- Users receive a **token upon login**, which is used to access resources.

4. CSRF (Cross-Site Request Forgery) Protection

- Enabled for all **non-API-based** requests to prevent unauthorized actions.

5. CORS (Cross-Origin Resource Sharing) Handling

- Configured to allow requests from the frontend to the backend securely.

6. Encryption & Data Protection

- **User credentials are hashed and stored securely.**
- **Sensitive user data (e.g., payment details) is encrypted.**

2. Authentication in ProConnect

Authentication ensures that **only registered users** can access the system. It follows the **JWT-based authentication mechanism**.

Authentication Process

1. User Login

- User submits **email and password**.
- Credentials are validated against the database.
- If valid, a **JWT token** is generated and sent to the user.

2. Access Control

- JWT token is included in every request.
- Backend validates the token before processing the request.

3. Token Expiry & Logout

- JWT tokens have an expiration time.
- Logout clears the stored token from the client side.

3. Role-Based Access Control (RBAC)

User roles and access restrictions are defined as follows:

Role	Permissions
------	-------------

Admin	Manage users, approve/reject projects, oversee payments.
-------	--

Client	Post projects, hire freelancers, make payments.
--------	---

Freelancer	Bid on projects, complete work, receive payments.
------------	---

Example URL Restrictions

- /admin/** → Accessible only by Admin.
- /client/** → Accessible by Clients.
- /freelancer/** → Accessible by Freelancers.

4. Security Best Practices Implemented

✓	Strong	Password	Encryption	(BCrypt)
✓	JWT	Authentication	for	Secure API Calls
✓	Role-Based	Access	Control	(RBAC)
✓	CSRF	&	CORS	Protection
✓	Secure Payment Processing.			

The ProConnect platform follows a secure authentication and authorization model using Spring Security with JWT. It ensures that users can access only what they are authorized for, protecting sensitive data from unauthorized access.

Implementation – Primary Implementation and Exploring Database in ProConnect

1. Primary Implementation

The **ProConnect freelancing platform** follows a structured implementation process that includes **database setup, backend development with Spring Boot, frontend integration, and security implementation.**

Key Aspects of Implementation:

- 1. Frontend Development (HTML, CSS, JavaScript)**
 - Designed UI for **Admin, Client, and Freelancer** dashboards.
 - Implemented **Thymeleaf** for dynamic content rendering.
 - Created **responsive and user-friendly interfaces.**
- 2. Backend Development (Spring Boot MVC)**
 - Built APIs for **user authentication, project management, and payments.**
 - Used **Spring Boot** for business logic implementation.
 - Integrated **JWT-based authentication for security.**
- 3. Database Setup & ORM (Hibernate, MySQL)**
 - Defined **Entity Classes** for database tables.
 - Used **Hibernate ORM** for seamless interaction with MySQL.
 - Designed **ER diagram** for relationship mapping.
- 4. Security Implementation (Spring Security)**
 - Configured **JWT authentication** for secure logins.
 - Implemented **role-based access control (RBAC)** for Admin, Client, and Freelancer.
- 5. Testing & Deployment**
 - Performed **unit testing, integration testing, and security testing.**
 - Deployed application on a **local server** for testing before final deployment.

2. Exploring Database in ProConnect

Database Management in MySQL

The ProConnect database is structured to store and manage data efficiently for clients, freelancers, projects, payments, and reviews.

Key Tables in the Database

Table Name	Description
User	Stores details of all users (Admin, Client, Freelancer).

Table Name	Description
Client	Stores client-specific details.
Freelancer	Stores freelancer details like skills and profile.
Project	Contains project details, budget, and deadlines.
ProjectRequest	Manages client project requests.
ServiceRequest	Handles freelancer service requests.
Payment	Records all transactions between clients and freelancers.
ReviewRating	Stores feedback and ratings given by clients and freelancers.

Database Relationships (ER Diagram)

- One-to-Many: A client can post multiple projects.
- One-to-Many: A freelancer can work on multiple projects.
- One-to-One: A user has a profile.
- Many-to-One: Multiple reviews belong to a single project.

Data Flow in Database

1. User logs in → Authenticated via User table.
2. Client posts a project → Stored in Project table.
3. Freelancer bids on project → Tracked in ProjectRequest table.
4. Payment is made → Stored in Payment table.
5. Review & Rating submitted → Added to ReviewRating table.

The implementation of ProConnect follows a well-structured approach with secure database management, efficient backend development, and responsive frontend design. The database plays a crucial role in handling user interactions, project management, and financial transactions securely.

CHAPTER-10

CODING

As we know we have 3 section client ,admin, freelancer and each folder contain it's dashboard code along with its all functional codes in template folder.

CLIENT FOLDER:-

COMMON.HTML

```
<!DOCTYPE html>

<html lang="en">
<head>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PROCONNECT</title>
    <link rel="stylesheet" href="/style/fragment/common.css">
    <link rel="stylesheet" href="/style/client.home.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css"
          integrity="sha512-Evv84Mr4kqVGRNSglGL/F/aIDqQb7xQ2vcrdlwxjThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg=="
          crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>

<body>
<header>

    <div class="logo">
        <a href="/client/home"></a>
    </div>
    <form action="/category/category/search">
        <div class="search-container">
            <input type="text" class="search-input" name="search" placeholder="Search you are looking for ...">
            <button class="search-btn">
                <span class="search-icon">#128269;</span>
            </button>
        </div>
    </form>
</header>
<main>
    <div class="content">
        <h1>Welcome to PROCONNECT</h1>
        <p>Your one-stop solution for all your business needs!</p>
        <ul>
            <li>Category 1</li>
            <li>Category 2</li>
            <li>Category 3</li>
            <li>Category 4</li>
            <li>Category 5</li>
        </ul>
        <div class="cta">
            <button>Get Started</button>
        </div>
    </div>
</main>
<footer>
    <div>
        <img alt="Footer Logo" />
        <p>Copyright © 2024 PROCONNECT. All Rights Reserved.</p>
        <ul>
            <li>About Us</li>
            <li>Contact Us</li>
            <li>FAQ</li>
            <li>Terms of Service</li>
            <li>Privacy Policy</li>
        </ul>
    </div>
</footer>

```

```

        </div>
    </form>
<div class="sign">
    <button class="sign-in"><span><i class="fa-solid fa-right-to-bracket"></i></span>
        <a href="/client/sign_in">Sign-In</a></button>
    <button class="call"><span><i class="fa-solid fa-phone"></i></span>
        <a href="/client/contact">Call-Us</a></button>
    <form method="post" action="/logout" data-sec-authorize="isAuthenticated()">
        <button class="logout" type="submit "><span><i class="fas fa-sign-out-alt"></i></span>
            Logout
        </button>
        <input type="hidden" name="_csrf" data-th-value="${_csrf.token}">
    </form>
</div>
</header>
<nav>
    <ul class="nav-links">
        <li onclick="toggleDropdown()">
            <div class="dashboard-container">
                <span id="dropdownText"><b>Dashboard</b></span>
                <i class="fas fa-chevron-down "></i>
                <div class="dropdown hidden">
                    <div class="dropdown-content" id="dashboardDropdown">
                        <ul>
                            <li>
                                <a href="/client/clientdashboard">Client Dashboard</a>
                            </li>
                            <li>
                                <a href="/admin/admindashboard">Admin Dashboard</a>
                            </li>
                            <li>

```

```

        <a href="/freelancer/freelancerdashboard">Freelancer Dashboard</a>
    </li>
</ul>
</div>
</div>
</div>
</li>
<li><a href="/client/solution"> <b>Our Solutions</b></a></li>
<li><a href="#about"><b>About Us</b></a></li>
<li><a href="/category/category"><b>Categories</b></a></li>
<li><a href="/freelancer/freelancer"><b>Freelancer</b></a></li>
<li><a href="/client/howitworks"><b>How it works</b></a></li>
</ul>
</nav>
<footer class="footer">
<div class="footer-container">
<div class="footer-section about">
<h2>ProConnect</h2>
<p>Connecting freelancers with businesses worldwide. Grow your career with ProConnect.</p>
</div>

<div class="footer-section links">
<h3>Quick Links</h3>
<ul>
<li><a href="/client/home">Home</a></li>
<li><a href="#">Browse Jobs</a></li>
<li><a href="/admindashboard/admindashboard">Post a Job</a></li>
<li><a href="/client/howitworks">How It Works</a></li>
<li><a href="/contact/contact">Contact Us</a></li>
</ul>
</div>

```

```
<script>

    function toggleDropdown() {
        document.querySelector(".dropdown").classList.toggle('hidden');
    }

</script>

<div class="footer-section support">
    <h3>Support</h3>
    <ul>
        <li><a href="#">Help Center</a></li>
        <li><a href="#">Terms & Conditions</a></li>
        <li><a href="#">Privacy Policy</a></li>
        <li><a href="#">FAQs</a></li>
    </ul>
</div>

<div class="footer-section social">
    <h3>Follow Us</h3>
    <div class="social-icons">
        <a href="#"><i class="fab fa-facebook"></i></a>
        <a href="#"><i class="fab fa-twitter"></i></a>
        <a href="#"><i class="fab fa-linkedin"></i></a>
        <a href="#"><i class="fab fa-instagram"></i></a>
    </div>
</div>
</div>

<div class="footer-bottom">
    <p>&copy; 2025 ProConnect. All Rights Reserved.</p>
</div>
</footer>
</body>
```

```

</html>

HOME.HTML

<!DOCTYPE html>

<html lang="en">

<head data-th-replace="~{fragment/common :: head}"></head>

<body>

    <header data-th-replace="~{fragment/common :: header}"></header>

    <nav data-th-replace="~{fragment/common :: nav}"></nav>

    <!-- Hero Section -->

    <section id="hero">

        <div class="hero-content">

            <h1>Welcome to PROCONNECT</h1>

            <p>Your gateway to top-notch freelance professionals.</p>

            <a href="#services" class="cta-btn">Get Started</a>

            <h1> Find & Hire <br>Expert Freelancers </h1>

            <h3>Work with the best freelance talent from around the world on our secure,<br class="u-blockLargeOnly">flexible and cost-effective platform.

            </h3>

        </div>

        <div>

        </div>

        <div class="button-container">

            <button class="btn" id="post-job"><a href="/client/clientdashboard">Post a project</a></button>

            <button class="btn" id="get-job"><a href="/freelancer/freelancerdashboard">Get a project</a></button>

        </div>

    </section>

<main>

    <div class="top_freelancer">

        <h1>PINPOINT LEADING SKILLS</h1>

```

```
</div>

<div class="container1">
    <div class="box">
        <i class="fas fa-code"></i>
        <p>Programming & Development</p>
    </div>
    <div class="box">
        <i class="fas fa-pen"></i>
        <p>Writing & Translation</p>
    </div>
    <div class="box">
        <i class="fas fa-paint-brush"></i>
        <p>Design & Art</p>
    </div>
    <div class="box">
        <i class="fas fa-user-tie"></i>
        <p>Administrative & Secretarial</p>
    </div>
    <div class="box">
        <i class="fas fa-chart-line"></i>
        <p>Sales & Marketing</p>
    </div>
    <div class="box">
        <i class="fas fa-drafting-compass"></i>
        <p>Engineering & Architecture</p>
    </div>
    <div class="box">
        <i class="fas fa-briefcase"></i>
        <p>Business & Finance</p>
    </div>
    <div class="box">
```

```
<i class="fas fa-chalkboard-teacher"></i>
<p>Education & Training</p>
</div>
<div class="box">
<i class="fas fa-balance-scale"></i>
<p>Legal</p>
</div>
</div>
</main>
<div class="add-content">
<a href="/category/category">All Categories</a>
</div>
<video src="/video/v2.mp4" autoplay muted loop></video>

<!-- About Section -->
<section id="about">
<div class="container">
<h2>About Us</h2>
<p>Learn more about our journey and how we connect talented freelancers with great opportunities!</p>
</div>
</section>

<!-- Our Mission -->
<section class="mission">
<div class="container">
<h2>Our Mission</h2>
<p>Our mission is to empower freelancers across the globe by providing them with an easy-to-use platform that connects them with quality clients and projects. We strive to offer an environment that encourages growth,</p>

```

```
        creativity, and professional development.</p>
    </div>
</section>

<!-- Our Story -->
<section class="story">
    <div class="container">
        <h2>OUR STORY</h2>
        <p>We started as a small group of freelancers who shared a vision of creating a platform that would enable individuals to showcase their skills and find new opportunities. Over time, our platform has grown to serve thousands of freelancers and clients worldwide, creating a community built on trust, collaboration, and innovation.</p>
    </div>
</section>

<!-- Team Section -->
<section class="team">
    <div class="container">
        <h2>MEET OUR TEAM</h2>
        <div class="team-members">
            <div class="team-member">
                
                <h3>Anjani Shaw</h3>
                <p>Founder & CEO</p>
            </div>
            <div class="team-member">
                
                <h3>Subhranjali Panda</h3>
                <p>Chief Operating Officer</p>
            </div>
        </div>
    </div>
</section>
```

```
<div class="team-member">
    
    <h3>Soumendra Behera</h3>
    <p>Lead Developer</p>
</div>
<div class="team-member">
    
    <h3>Smruti Ranjan Nayak</h3>
    <p>Marketing Director</p>
</div>
</div>
</div>
</section>
<!-- Testimonials Section -->
<section class="testimonials">
    <div class="container">
        <h2>What Our Users Say</h2>
        <div class="testimonial">
            <blockquote>
                "This platform has been a game changer for my freelance career. It made it so easy to find high-quality clients and manage my projects efficiently."
            </blockquote>
            <p>- Sarah Lee, Freelancer</p>
        </div>
        <div class="testimonial">
            <blockquote>
                "As a client, I've been able to hire top talent for all my projects. I highly recommend this platform to anyone looking to find skilled freelancers."
            </blockquote>
            <p>- Mike Green, Client</p>
        </div>
    </div>

```

```
</div>

<!-- Contact Section -->

<section class="contact-section">

    <div class="container">

        <h1>Get in Touch with ProConnect</h1>

        <p>Whether you're a freelancer or a business, we're here to help. Please contact us using any of the methods below.</p>

        <div class="contact-methods">

            <div class="contact-box">

                <h2>Email Us</h2>

                <p>For general inquiries, reach out to us at:</p>

                <a href="mailto:support@proconnect.com">support@proconnect.com</a>

                <p>If you're a business looking for freelancer solutions:</p>

                <a href="mailto:projects@proconnect.com">projects@proconnect.com</a>

            </div>

            <div class="contact-box">

                <h2>Message Form</h2>

                <form action="#" method="POST">

                    <input type="name" id="name" placeholder="Name"></input>

                    <textarea name="messageus" id="msg">message</textarea>

                    <button type="submit">Submit</button>

                </form>

            </div>

            <div class="contact-box">

                <h2>Phone Support</h2>

                <p>If you need immediate assistance, feel free to call us at:</p>

                <p><strong>+1 (800)-7735870981 </strong></p>

                <p>Our team is available Monday to Friday, from 9 AM to 6 PM.</p>

            </div>

            <div class="contact-box">

                <h2>Follow Us</h2>
```

<p>Stay connected with us through social media:</p>

Facebook <i class="fa-brands fa-facebook"></i>

Twitter <i class="fa-brands fa-square-twitter"></i>

LinkedIn <i class="fa-brands fa-linkedin"></i>

Instagram <i class="fa-brands fa-instagram"></i>

</div>

<div class="contact-box">

<p>If you'd like to meet us in person, our office is located at:</p> <p>123 Freelancer Lane, proconnect , California- Suite 200, Tech City, USA</p> </div> <div> <p>our website not only give platform to everyone but also it provide essence to connect through out the world. </p> <p>ProConnect website our concern to connect people and grow.</p> </div> </div> </div> </div> </section> <!--designer --> <section class="designer-list">

```
<h2>MEET OUR DEVELOPERS</h2>

<div class="designer-cards">

    <div class="designer-card">

        <h3>Jane Doe</h3>

        <p>UI/UX Designer</p>

        <p>Expert in creating beautiful and functional user interfaces.</p>

        <button class="hire-btn">Hire Now</button>

    </div>

    <div class="designer-card">

        <h3>John Smith</h3>

        <p>Web Designer</p>

        <p>Specializes in modern and responsive websites.</p>

        <button class="hire-btn">Hire Now</button>

    </div>

    <div class="designer-card">

        <h3>Emily Brown</h3>

        <p>Branding Expert</p>

        <p>Experienced in logo and brand identity design.</p>

        <button class="hire-btn">Hire Now</button>

    </div>

    <div class="designer-card">

        <h3>David Lee</h3>

        <p>Graphic Designer</p>

        <p>Creating stunning graphics for various digital platforms.</p>

        <button class="hire-btn">Hire Now</button>

    </div>

</div>
```

```
</section>

<div class="add-content1">
    <a href="/freelancer/freelancer">See All</a>
</div>

<!--make it all happen with freelancer-->

<div class="makehappen">
    <h2>Make it all happen with freelancer</h2>
<div class="benefits">
    <div class="benefit">
        <div class="icon"><i class="fa-solid fa-people-group"></i></div>
        <p>Access a pool of top talent across 700 categories</p>
    </div>
    <div class="benefit">
        <div class="icon">✓</div>
        <p>Enjoy a simple, easy-to-use matching experience</p>
    </div>
    <div class="benefit">
        <div class="icon">⚡</div>
        <p>Get quality work done quickly and within budget</p>
    </div>
    <div class="benefit">
        <div class="icon">฿</div>
        <p>Only pay when you're happy</p>
    </div>
</div>
<button class="cta-button">Join now</button>
</div>
<div class="q_a_section">
    <h2>Q & A</h2>
    <div class="faq">
        <div class="faq-item" onclick="toggleAnswer(this)">
```

```
<h3>What can I sell?</h3>
<p>You can sell digital services like design, writing, marketing, and more.</p>
</div>
<div class="faq-item" onclick="toggleAnswer(this)">
    <h3>How much time will I need to invest?</h3>
    <p>You decide your working hours based on your availability.</p>
</div>
<div class="faq-item" onclick="toggleAnswer(this)">
    <h3>How much money can I make?</h3>
    <p>Your earnings depend on your skills and the number of projects you complete.</p>
</div>
<div class="faq-item" onclick="toggleAnswer(this)">
    <h3>How do I price my service?</h3>
    <p>Research the market and set competitive pricing based on your skills.</p>
</div>
<div class="faq-item" onclick="toggleAnswer(this)">
    <h3>How much does it cost?</h3>
    <p>Signing up is free, but the platform takes a small commission from your earnings.</p>
</div>
<div class="faq-item" onclick="toggleAnswer(this)">
    <h3>How do I get paid?</h3>
    <p>Payments are processed securely through multiple withdrawal options.</p>
</div>
</div>
</div>
<script src="/script/home.js"></script>
<footer data-th-replace="~{fragment/common :: footer}"></footer>
</body>
</html>
```

CATEGORY.HTML

```

<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Freelancing Categories</title>
    <link rel="stylesheet" href="/style/fragment/common.css">
    <link rel="stylesheet" href="/style/client.category.css">
      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv8Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
  </head>
  <body>
    <header data-th-replace="~{fragment/common :: header}"></header>
    <nav data-th-replace="~{fragment/common :: nav}"></nav>
    <section class="heading">
      <div class="hero-section">
        <h1>Find the Best Freelancers for Every Project</h1>
        <p>Explore categories and connect with top experts in every field.</p>
        <a href="#categories" class="cta-button">Browse Categories</a>
      </div>
    </section>
    <form action="/project/project/search">
      <section id="search">
        <input name="search" type="text" id="search-input" placeholder="Search projects...">
      </section>
    </form>
    <section id="categories">
      <h2>Browse Categories</h2>
      <div class="categories-container">
        <a data-th-href="|/category/freelancer?categoryId=${category.categoryId}|">

```

```

        data-th-each="category:${categories}">
        <div class="category-card" data-category="web-dev">
            
            <h3 data-th-text="${category.categoryName}"></h3>
            <p data-th-text="${category.categoryDescription}"></p>
        </div>
    </a>
</div>
</section>
<!-- Modal for Category Details -->
<div id="category-modal" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <h3 id="modal-title">Category Details</h3>
        <p id="modal-description">Category description will appear here.</p>
    </div>
</div>
<footer data-th-replace="~{fragment/common :: footer}"></footer>
</body>
</html>

```

FREELANCER.HTML

```

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Hire Developers</title>
        <link rel="stylesheet" href="/style/client.freelancer.css">
        <link rel="stylesheet" href="/style/fragment/common.css">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css"
integrity="sha512-

```

```

Evv84Mr4kqVGRNSgIGL/F/aIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" 
crossorigin="anonymous" referrerpolicy="no-referrer" />

</head>

<body>

    <header data-th-replace="~{fragment/common :: header}"></header>

    <nav data-th-replace="~{fragment/common :: nav}"></nav>

    <h1>Hire the best <span> developers</span></h1>

    <form action="/freelancer/freelancer/search">

        <section id="search">

            <input name="search" type="text" id="search-input" placeholder="Search freelancers...">

        </section>

    </form>

    <div class="developers-container">

        <div class="developer-card" th:each="freelancer : ${freelancers}">

                  <h3 th:text="${freelancer.userName}"></h3>

            <p class="rating" th:text="|${freelancer.rating}/5|"></p>

            <p class="rating" th:text="|Experience: ${freelancer.yearsExperience} years|"></p>

            <div class="skills-container">

                <div class="skills">

                    <span th:each="skill : ${#strings.arraySplit(freelancer.skills, ',')}" class="bg-gray-200 text-gray-800 py-1 px-3 rounded" th:text="${skill}">
                    </span>

                </div>

            </div>

            <a data-th-href="|/freelancer/pdp?id=${freelancer.userId}|" class="see-more">See more</a>

        </div>

    </div>

    <footer data-th-replace="~{fragment/common :: footer}"></footer>

</body>

</html>

```

CLIENTDASHBOARD.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ProConnect Client Dashboard</title>
    <link rel="stylesheet" href="/style/client.clientdashboard.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
        <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerPolicy="no-referrer" />
</head>

<body>
    <div class="dashboard-container">
        <aside class="sidebar">
            <h2><a href="/client/home">ProConnect</a></h2>
            <ul>
                <li><a href="#"><i class="fa-solid fa-house"></i> Dashboard</a></li>
                <li><a href="/client/myproject"><i class="fas fa-tasks"></i> My Projects</a></li>
                <li><a href="/client/request"><i class="fas fa-users"></i> Service Request</a></li>
                <li><a href="/client/projectRequest"><i class="fas fa-users"></i> Project Request</a></li>
                <li><a href="/client/profile"><i class="fas fa-money-bill"></i> Profile</a></li>
                <li><a href="/client/reviewrating"><i class="fa-solid fa-star"></i>Review and Rating</a></li>
                <li>
                    <form method="post" action="/logout" data-sec-authorize="isAuthenticated()">
                        <button>
                            <i class="fas fa-sign-out-alt"></i> Logout

```

```
</button>

<input type="hidden" name="_csrf" data-th-value="$_csrf.token">

</form>

</li>

</ul>

</aside>

<main class="content">

    <header>

        <h1>Client Dashboard</h1>

    </header>

    <div class="search-container">

        <input type="text" id="pageSearchBar" placeholder="Search categories..." class="search-bar">

    </div>

    <section class="stats">

        <div class="stat-box">

            <i class="fas fa-briefcase card-icon"></i>

            <h3>Active Projects</h3>

            <p>5</p>

        </div>

        <div class="stat-box">

            <i class="fas fa-dollar-sign card-icon"></i>

            <h3>Pending Payments</h3>

            <p>$2,300</p>

        </div>

        <div class="stat-box">

            <i class="fas fa-file-alt card-icon"></i>

            <h3>Freelancers Hired</h3>

            <p>12</p>

        </div>

    </section>
```

```
<section class="projects">
    <h2>My Projects</h2>
    <table>
        <thead>
            <tr>
                <th>Project Name</th>
                <th>Freelancer</th>
                <th>Budget</th>
                <th>Status</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Website Development</td>
                <td>John Doe</td>
                <td>$3,000</td>
                <td>Ongoing</td>
            </tr>
            <tr>
                <td>Logo Design</td>
                <td>Jane Smith</td>
                <td>$500</td>
                <td>Completed</td>
            </tr>
            <tr>
                <td>SEO Optimization</td>
                <td>Michael Brown</td>
                <td>$1,200</td>
                <td>Pending Approval</td>
            </tr>
        </tbody>
```

```

        </table>

    </section>

</main>

</div>

</body>

</html>

MYPROJECT.HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Myproject</title>

<link rel="stylesheet" href="/style/client.clientdashboard.css">

<link rel="preconnect" href="https://fonts.googleapis.com">

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/F/aIDqQb7xQ2vcrdlwxfjThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerpolicy="no-referrer" />

</head>

<body>

<aside data-th-replace="~{client/clientdashboard :: aside}"></aside>

<link rel="stylesheet" href="/style/client.myproject.css">

<!-- Main Content -->

<main class="content">

<header>

<h1>Welcome, Client</h1>

</header>

<section class="projects">

<h2>My Projects</h2>

```

```

<button class="add-project-btn" onclick="openModal()">Add New Project</button>

<table>
    <thead>
        <tr>
            <th>Project Title</th>
            <th>Description</th>
            <th>Listed Date</th>
            <th>Image</th>
            <th>Price</th>
            <th>Status</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody id="projectTable" th:if="${user != null and user.projects != null}">
        <tr data-th-each="project:${user.projects}">
            <td data-th-text="${project.projectTitle}"></td>
            <td data-th-text="${project.projectDescription}"></td>
            <td data-th-text="${project.createdDate}"></td>
            <td>
                
            </td>
            <td data-th-text="${project.projectPrice}"></td>
            <td data-th-text="${project.projectStatus}"></td>
            <td>
                <span
                    data-th-if="${project.projectStatus == 'OPEN' or project.projectStatus == 'CANCELED'}">
                    <button class="edit-btn"
                        data-th-onclick="openModal([[${project.projectId}]])">Edit</button>
                    <a data-th-href="|/client/myproject/delete?project=${project.projectId}|" class="delete-btn">Delete</a>
                </span>
            </td>
        </tr>
    </tbody>

```

```

<span data-th-if="${project.projectStatus == 'IN_PROGRESS'}">
    <i class="fa-solid fa-ban"></i>
</span>
<span data-th-if="${project.projectStatus == 'COMPLETED'}">
    <i class="fa-solid fa-check-double"></i>
</span>
</td>
</tr>
</tbody>
</table>
</section>
</main>
</div>

<!-- Add Project Modal -->
<div id="projectModal" class="modal">
    <div class="modal-content">
        <span class="close-btn" onclick="closeModal()">&times;</span>
        <h2>Add New Project</h2>
        <form id="projectForm" action="/client/myproject" method="post" enctype="multipart/form-data">
            <input type="hidden" id="projectId" name="projectId">
            <input type="hidden" id="projectImageName" name="projectImageName">
            <input type="text" id="projectTitle" name="projectTitle" placeholder="Project Name">
            <input type="text" id="projectPrice" name="projectPrice" placeholder="Project Price">
            <input type="file" id="projectImage" name="projectImage" placeholder="Project Image">
            <textarea name="projectDescription" id="projectDescription"
                placeholder="Describe the Project Details"></textarea>
            <select name="category" id="category">
                <option value="" disabled selected>Select Category</option>
                <option data-th-value="${category.categoryId}" data-th-each="category:${categories}"
                    data-th-text="${category.categoryName}"></option>
            </select>
        </form>
    </div>
</div>

```

```

        </select>

        <input type="hidden" name="_csrf" data-th-value="${_csrf.token}">
        <button type="submit">Add Project</button>
    </form>
</div>
</div>
<script src="/script/myproject.js"></script>
</body>
</html>

```

PROJECTPDP.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Project Description - Freelancing Website</title>
</head>
<body>
    <main>
        <section class="project-description">
            <h2 data-th-text="|Project Title: ${project.projectTitle}|"></h2>
            <div>
                
            </div>
        <div class="project-details">
            <p><strong>Client:</strong> <span data-th-text="${project.client.userName}"></span></p>
            <p><strong>Budget:</strong> <span data-th-text="${project.projectPrice}"></span></p>
            <p><strong>Listed Date:</strong> <span data-th-text="${project.createdDate}"></span></p>
        </div>
        <div class="description">

```

```

        <h3>Project Description</h3>
        <p data-th-text="${project.projectDescription}"></p>
    </div>
    <div class="apply">
        <a data-th-href="|/serviceRequest/apply?id=${project.projectId}|">Apply for this Project</a>
    </div>
</section>
</main>
</body>
</html>

```

PROJECTREQUEST.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Service Request</title>
    <link rel="stylesheet" href="/style/admin/admindashboard.css">
    <link rel="stylesheet" href="/style/client.request.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerPolicy="no-referrer" />
</head>

<body>
    <aside data-th-replace="~{client/clientdashboard :: aside}"></aside>
    <main class="content">
        <header>

```

```

<h1>Welcome, Client</h1>

</header>

<section class="projects">
    <h1>Your Project Requests</h1>
    <!-- <button class="add-project-btn" onclick="openModal()">Add New Project</button> -->
    <table>
        <thead>
            <tr>
                <th>Freelancer Name</th>
                <th>Project Title</th>
                <th>Image</th>
                <th>Price</th>
                <th>Category</th>
                <th>Status</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody id="projectTable">
            <tr data-th-each="request : ${user.projectRequests}">
                <td data-th-text="${request.freelancer.userName}"></td>
                <td data-th-text="${request.project.projectTitle}"></td>
                <td>Image</td>
                <td data-th-text="${request.project.projectPrice}"></td>
                <td data-th-text="${request.project.category.categoryName}"></td>
                <td data-th-text="${request.projectRequestStatus}"></td>
                <td>
                    <span
                        data-th-if="${request.projectRequestStatus.name() == 'APPROVED' ||
                        request.projectRequestStatus.name() == 'REJECTED' || request.projectRequestStatus.name() == 'PENDING'}">
                        <a class="cancel-btn">
                            ...
                        </a>
                    </span>
                </td>
            </tr>
        </tbody>
    </table>
</section>

```

```

        data-th-
Approve</a>

        <a class="cancel-btn" data-th-
Reject</a>

```

PAYMENT.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Freelancer Payment</title>
    <link rel="stylesheet" href="/style/fragment/common.css">
    <link rel="stylesheet" href="/style/client.payment.css">
    <link rel="stylesheet" href="/script/payment.js">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css"

```

```

integrity="sha512-
Evv84Mr4kqVGRNSgIGL/F/aIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg=="
crossorigin="anonymous" referrerpolicy="no-referrer" />

</head>

<body>

<header data-th-replace="~{fragment/common :: header}"></header>

<nav data-th-replace="~{fragment/common :: nav}"></nav>

<div class="payment-container">

<h2>█ Secure Payment</h2>

<!-- Order Summary -->

<div class="order-summary">

<h3>█ Order Summary</h3>

<p><strong>Freelancer:</strong> <span data-th-text="${serviceRequest.freelancer.userName}"></span></p>
<p><strong>Service:</strong> <span data-th-text="${serviceRequest.project.projectTitle}"></span></p>
<p class="total"><strong>Price:</strong><span data-th-text="|${serviceRequest.project.projectPrice}|"></span>
</p>
</div>

<form id="payment-form">

<!-- Billing Info -->

<h3>█ Billing Details</h3>
<!-- Payment Method -->

<h3>█ Payment Method</h3>

<div class="input-group">
<label for="payment-method">Select Payment Method</label>
<select id="payment-method">
<option value="credit-card">Credit Card</option>
<option value="paypal">PayPal</option>
<option value="crypto">Cryptocurrency</option>

```

```

        </select>

    </div>

    <!-- Credit Card Details -->

    <div id="card-details" class="hidden">

        <div class="credit-card-preview">

            <div class="card-number-display">***** ***** ***** </div>

            <div class="card-name-display">Cardholder Name</div>

        </div>

        <div class="input-group">

            <label for="card-number">Card Number</label>

            <input type="text" id="card-number" placeholder="1234 5678 9012 3456" required>

        </div>

        <div class="input-group">

            <label for="card-name">Name on Card</label>

            <input type="text" id="card-name" placeholder="John Doe" required>

        </div>

        <div class="input-row">

            <div class="input-group">

                <label for="expiry">Expiry Date</label>

                <input type="text" id="expiry" placeholder="MM/YY" required>

            </div>

            <div class="input-group">

                <label for="cvv">CVV</label>

                <input type="text" id="cvv" placeholder="123" required>

            </div>

        </div>

    </div>

    <div class="paynow">

        <a type="submit" href="|/serviceRequest/approve?id=${serviceRequest.serviceRequestId}&status=approve|">

```

data-th-

 href="|/serviceRequest/approve?id=\${serviceRequest.serviceRequestId}&status=approve|"/>

```

        Pay Now</a>

    </div>
</form>
</div>
<footer data-th-replace="~{fragment/common :: footer}"></footer>
</body>
</html>

SIGHUP.HTML

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css">
<title>Login Page</title>
</head>
<body>
<div class="container" id="container">
<div class="form-container sign-upper">
<form action="/client/sign_up" method="post" enctype="multipart/form-data">
<h1>Create Account as a Client</h1>
<div class="social-icons">
<a href="#" class="icon"><i class="fa-brands fa-google-plus-g"></i></a>
<a href="#" class="icon"><i class="fa-brands fa-facebook-f"></i></a>
<a href="#" class="icon"><i class="fa-brands fa-github"></i></a>
<a href="#" class="icon"><i class="fa-brands fa-linkedin-in"></i></a>
</div>
<input type="text" placeholder="Name" name="userName">
<input type="email" placeholder="Email" name="userEmail">
<input type="password" placeholder="Password" name="userPassword">
<input type="password" placeholder="Confirm Your Password">

```

```

<input type="tel" id="phone" placeholder="Enter Contact Information" name="userPhone"
       pattern="[0-9]{10}" required>

<input type="text" placeholder="Add Bio" name="userBio">

<input type="file" name="userProfilePic" id="userProfilePic">

<textarea name="location" id="location" placeholder="Enter your Location"></textarea>

<input type="hidden" name="_csrf" id="" data-th-value="$_csrf.token">

<input type="submit" value="Sign Up" style="background-color: royalblue;">

</form>

</div>

<div class="form-container sign-up">

<form action="/freelancer/signup" method="post" enctype="multipart/form-data">

<h1>Create Account as a Freelancer</h1>

<div class="social-icons">

<a href="#" class="icon"><i class="fa-brands fa-google-plus-g"></i></a>

<a href="#" class="icon"><i class="fa-brands fa-facebook-f"></i></a>

<a href="#" class="icon"><i class="fa-brands fa-github"></i></a>

<a href="#" class="icon"><i class="fa-brands fa-linkedin-in"></i></a>

</div>

<input type="text" placeholder="Name" name="userName">

<input type="email" placeholder="Email" name="userEmail">

<input type="password" placeholder="Password" name="userPassword">

<input type="password" placeholder="Confirm Your Password">

<input type="tel" id="phone" placeholder="Enter Contact Information" name="userPhone"
       pattern="[0-9]{10}" required>

<input type="text" placeholder="Add Bio" name="userBio">

<input type="file" name="userProfilePic" id="userProfilePic">

<textarea name="language" id="language" placeholder="like english, hindi, etc...."></textarea>

<input type="text" placeholder="Specialization like: Web Development" name="specialization">

<input type="text" placeholder="Years of Experience (in number)" name="yearsExperience">

<textarea name="skills" id="skills" placeholder="Skills: like java, python, etc...."></textarea>

<textarea name="location" id="location" placeholder="Enter your Location"></textarea>

```

```

        <input type="hidden" name="_csrf" id="" data-th-value="$_csrf.token">
        <input type="submit" value="Sign Up" style="background-color: royalblue;">
    </form>
</div>
<div class="toggle-container">
    <div class="toggle">
        <div class="toggle-panel toggle-left">
            <h1>Hello, Friend!</h1>
            <p>Enter your personal details to use all of site features for freelancers </p>
            <button class="hidden" id="login">Sign Up</button>
        </div>
        <div class="toggle-panel toggle-right">
            <h1>Hello, Friend!</h1>
            <p>Register with your personal details to use all features for clients</p>
            <button class="hidden" id="register">Sign Up</button>
        </div>
    </div>
</div>
</div>
<script src="/script/sign_up.js"></script>
</body>
</html>

```

ADMIN FOLDER:-

ADMINDASHBOARD.HTML

```

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>ProConnect Admin Dashboard</title>
        <link rel="stylesheet" href="/style/admin/admindashboard.css">

```

```

<link rel="preconnect" href="https://fonts.googleapis.com">
<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxJfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerPolicy="no-referrer" />
</head>
<body>
<div class="dashboard-container">
<aside class="sidebar">
<h2>ProConnect</h2>
<ul>
<li><a href="#"><i class="fa fa-home" aria-hidden="true"></i> Dashboard</a></li>
<li><a href="/admin/category"><i class="fas fa-users"></i> Category</a></li>
<li><a href="/admin/usermanagement"><i class="fas fa-briefcase"></i> User Management</a></li>
<li><a href="/admin/freelancermanagement"><i class="fas fa-briefcase"></i> Freelancer Management</a></li>
<li><a href="/admin/report"><i class="fa-solid fa-chart-simple"></i> Report and Analytics</a></li>
<li><a href="#"><i class="fas fa-sign-out-alt"></i> Logout</a></li>
</ul>
</aside>
<main class="content">
<header>
<h1>Admin Dashboard</h1>
</header>
<section class="stats">
<div class="stat-box">
<h3>Users</h3>
<p>1,245</p>
</div>
<div class="stat-box">

```

```
<h3>Active Jobs</h3>
<p>320</p>
</div>
<div class="stat-box">
    <h3>Earnings</h3>
    <p>$45,210</p>
</div>
<div class="stat-box">
    <h3>Disputes</h3>
    <p>12</p>
</div>
</section>
<section class="jobs">
    <h2>Recent Job Listings</h2>
    <table>
        <thead>
            <tr>
                <th>Job Title</th>
                <th>Client</th>
                <th>Budget</th>
                <th>Status</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Web Developer Needed</td>
                <td>John Doe</td>
                <td>$1,500</td>
                <td>Active</td>
            </tr>
            <tr>
```

```

        <td>Graphic Designer</td>
        <td>Jane Smith</td>
        <td>$800</td>
        <td>Completed</td>
    </tr>
    <tr>
        <td>SEO Specialist</td>
        <td>Michael Brown</td>
        <td>$1,200</td>
        <td>Active</td>
    </tr>
    <tr>
        <td>Tester </td>
        <td>Mikal jaikson</td>
        <td>$1,000</td>
        <td>Active</td>
    </tr>
</tbody>
</table>
</section>
</main>
</div>
</body>
</html>

```

CATEGORY.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

```

```

<link rel="stylesheet" href="../style/admin/admindashboard.css">
<link rel="stylesheet" href="../style/admin/category.css">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>
<body>
<aside data-th-replace="~{admin/admindashboard :: aside}"></aside>
<main class="content">
<h1>Admin Dashboard</h1>
<div class="search-container">
<input type="text" id="pageSearchBar" placeholder="Search categories..." class="searchbar">
</div>
<button onclick="openModal()">Add Category</button>
<div id="categoryModal" class="modal" style="display: none;">
<h2>Add Category</h2>
<form id="categoryForm" method="post" action="/admin/admin/category" enctype="multipart/form-data">
<input type="hidden" id="categoryId" name="categoryId">
<input type="hidden" id="categoryImages" name="categoryImages">
<label for="categoryName">Category Name:</label>
<input type="text" name="categoryName" id="categoryName">
<label for="categoryDescription">Category Description:</label>
<input type="text" name="categoryDescription" id="categoryDescription">
<label for="categoryImage">Category Image:</label>
<input type="file" name="categoryImageFile" id="categoryImage" accept="image/png, image/jpeg, image/jpg">
<label for="categoryStatus">Category Status:</label>

```

```

<select id="categoryStatus" name="categoryStatus">
    <option value="AVAILABLE">Available</option>
    <option value="UNAVAILABLE">Unavailable</option>
</select>

<input type="hidden" name="_csrf" id="" data-th-value="$_csrf.token">
<input type="submit" value="Add">
</form>

<button onclick="closeModal()">Cancel</button>
</div>

<table border="1">
    <thead>
        <tr>
            <th>Category Name</th>
            <th>Category Description</th>
            <th>Image</th>
            <th>Status</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody id="categoryTableBody">
        <tr data-th-each="category:${categories}">
            <td data-th-text="${category.categoryName}"></td>
            <td data-th-text="${category.categoryDescription}"></td>
            <td></td>
            <td data-th-text="${category.categoryStatus}"></td>
            <td>
                <button class="edit-btn" data-th-onclick="openModal([[${category.categoryId}]])">Edit</button>
                <a data-th-href="|/admin/category/delete?category=${category.categoryId}|" class="delete-btn">Delete</a>
            </td>
        </tr>
    </tbody>
</table>

```

```

        </tbody>
    </table>
</main>
<script src="/script/admin_category.js"></script>
</body>
</html>

```

FREELANCERMANAGMENT.HTML

```

<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Freelancer Managment</title>
    <link rel="stylesheet" href="/style/admin/admindashboard.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
    <link rel="stylesheet" href=" ../style/admin/usermanagment.css">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxvjThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>
<body>
    <aside data-th-replace="~{admin/admindashboard :: aside}"></aside>
    <main class="content">
        <h1>Freelancer Management</h1>
        <p>Manage platform users, edit details, or remove accounts.</p>
        <table class="user-table">
            <tr>
                <th>Name</th>
                <th>Email</th>

```

```

<th>Completed project</th>
<th>Total client</th>
<th>Language</th>
<th>Year of experience</th>
<th>Specialization</th>
<th>Skill</th>
<th>Location</th>
</tr>
<tr data-th-each="freelancer:${freelancers}">
    <td data-th-text="${freelancer.userName}"></td>
    <td data-th-text="${freelancer.userEmail}"></td>
    <td>10</td>
    <td>12</td>
    <td data-th-text="${freelancer.language}"></td>
    <td data-th-text="${freelancer.yearsExperience}"></td>
    <td data-th-text="${freelancer.specialization}"></td>
    <td data-th-text="${freelancer.skills}"></td>
    <td data-th-text="${freelancer.location}"></td>
</tr>
</table>
</main>
</body>
</html>

```

USERMANAGMENT.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Management</title>
    <link rel="stylesheet" href="/style/admin/admindashboard.css">

```

```

<link rel="preconnect" href="https://fonts.googleapis.com">
<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
<link rel="stylesheet" href="/style/admin/usermanagment.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerPolicy="no-referrer" />
</head>
<body>
<aside data-th-replace="~{admin/admindashboard :: aside}"></aside>
<main class="content">
<h1>User Management</h1>
<p>Manage platform users, edit details, or remove accounts.</p>
<table class="user-table">
<tr>
<th>Name</th>
<th>Email</th>
<th>Role</th>
<th>Bio</th>
<th>Phone</th>
<th>Image</th>
<th>Location</th>
</tr>
<tr data-th-each="user: ${users}">
<td data-th-text="${user.userName}"></td>
<td data-th-text="${user.userEmail}"></td>
<td data-th-text="${user.userType}"></td>
<td data-th-text="${user.userBio}"></td>
<td data-th-text="${user.userPhone}"></td>
<td></td>
<td data-th-text="${user.location}"></td>

```

```

        </tr>
    </table>
</main>
</body>
</html>

FREELANCER FOLDER:-

FREELANCERDASHBOARD.HTML

<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ProConnect Admin Dashboard</title>
    <link rel="stylesheet" href="../style/admin/admindashboard.css">
    <link rel="stylesheet" href="../style/admin/category.css">
    <link rel="stylesheet" href="../style/freelancer/freelancerdashboard.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/F/aIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>
<body>
    <div class="dashboard-container">
        <aside class="sidebar">
            <h2>ProConnect</h2>
            <ul>
                <li><a href="#"><i class="fa fa-home" aria-hidden="true"></i>Dashboard</a></li>
                <li><a href="/freelancer/project"><i class="fas fa-project-diagram"></i> Projects</a></li>
                <li><a href="/freelancer/earning"><i class="fas fa-dollar-sign"></i> Earnings</a></li>
            
        </aside>
    
```

```

        <li><a href="/freelancer/servicerequest"><i class="fas fa-users"></i>Service Request</a></li>
        <li><a href="/freelancer/projectRequest"><i class="fas fa-briefcase"></i>Project Complete
Req.</a></li>
        <li><a href="/freelancer/profile"><i class="fas fa-money-bill"></i> Profile</a></li>
        <li><a href="#"><i class="fas fa-sign-out-alt"></i>Logout</a></li>
    </ul>
</aside>
</div>
<main class="content">
    <header>
        <h1>Freelancer Dashboard</h1>
    </header>
    <section class="stats">
        <div class="stat-box">
            <i class="fas fa-briefcase card-icon"></i>
            <h3>Ongoing Projects</h3>
            <p>5</p>
        </div>
        <div class="stat-box">
            <i class="fas fa-dollar-sign card-icon"></i>
            <h3>Pending Payments</h3>
            <p>$2,300</p>
        </div>
        <div class="stat-box">
            <i class="fas fa-file-alt card-icon"></i>
            <h3>Completed project</h3>
            <p>12</p>
        </div>
    </section>
</body>
</html>

```

PROJECT.HTML

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Freelancer Dashboard</title>
    <link rel="stylesheet" href="/style/admin/admindashboard.css">
    <link rel="stylesheet" href="../style/admin/category.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerPolicy="no-referrer" />
  </head>
  <body>
    <aside data-th-replace="~{freelancer/freelancerdashboard :: aside}"></aside>
    <main class="content">
      <h1>Freelancer Dashboard</h1>
      <div class="search-container">
        <input type="text" id="pageSearchBar" placeholder="Search categories..." class="search-bar">
      </div>
    </div>
    <table border="1">
      <thead>
        <tr>
          <th>Project Name</th>
          <th>Project Description</th>
          <th>Image</th>
          <th>Status</th>
        </tr>
      </thead>
```

```

<th>Action</th>
</tr>
</thead>
<tbody id="projectTableBody">
    <tr data-th-if="${freelancer != null and freelancer.projects != null and #lists.isEmpty(freelancer.projects)}" style="color: red;">
        No Project Yet!!
    </tr>
    <tr data-th-each="project : ${freelancer != null and freelancer.projects != null ? freelancer.projects : {}}">
        <td data-th-text="${project.projectTitle}"></td>
        <td data-th-text="${project.projectDescription}"></td>
        <td></td>
        <td data-th-text="${project.projectStatus}"></td>
        <td>
            <span data-th-if="${project.projectStatus != 'COMPLETED'}">
                <a data-th-href="|/projectRequest/apply?id=${project.projectId}">Completion Request</a>
            </span>
            <span data-th-if="${project.projectStatus == 'COMPLETED'}">
                <i class="fa-solid fa-check-double"></i>
            </span>
        </td>
    </tr>
</tbody>
</table>
</main>
<script src="/script/project.js"></script>
</body>
</html>

```

PROJECTREQUEST.HTML

```

<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Service Request</title>
    <link rel="stylesheet" href="../style/admin/admindashboard.css">
    <link rel="stylesheet" href="/style/client.request.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxvjThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerPolicy="no-referrer" />
  </head>
  <body>
    <aside data-th-replace="~{freelancer/freelancerdashboard :: aside}"></aside>
    <main class="content">
      <header>
        <h1>Welcome, Freelancer</h1>
      </header>
      <section class="projects">
        <h1>Your Project Requests</h1>
        <!-- <button class="add-project-btn" onclick="openModal()">Add New Project</button> -->
        <table>
          <thead>
            <tr>
              <th>Client Name</th>
              <th>Project Title</th>
              <th>Image</th>
              <th>Price</th>
            </tr>
          </thead>
        </table>
      </section>
    </main>
  </body>
</html>

```

```

<th>Category</th>
<th>Status</th>
<th>Action</th>
</tr>
</thead>
<tbody id="projectTable">
<tr data-th-each="request : ${freelancer.projectRequests}">
<td data-th-text="${request.client.userName}"></td>
<td data-th-text="${request.project.projectTitle}"></td>
<td>Image</td>
<td data-th-text="${request.project.projectPrice}"></td>
<td data-th-text="${request.project.category.categoryName}"></td>
<td data-th-text="${request.projectRequestStatus}"></td>
<td>
<!-- <span data-th-if="${request.projectRequestStatus.name() != 'CANCELED'}"></span> -->
<span data-th-if="${request.projectRequestStatus.name() == 'PENDING'}">
<a class="cancel-btn"
data-th-
href="/projectRequest/approve?id=${request.projectRequestId}&status=canceled | ">Cancel</a>
</span>
<span
data-th-if="${request.projectRequestStatus.name() == 'CANCELED' or
request.projectRequestStatus.name() == 'REJECTED'}">
<i class="fa-solid fa-ban"></i>
</span>
<span data-th-if="${request.projectRequestStatus.name() == 'APPROVED'}">
<i class="fa-solid fa-circle-check"></i>
</span>
</td>
</tr>
</tbody>
</table>

```

```

        </section>
    </main>
</body>
</html>

SERVICEREQUEST.HTML

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Service Request</title>
    <link rel="stylesheet" href="../style/admin/admindashboard.css">
    <link rel="stylesheet" href="/style/client.request.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
        <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGRNSgIGL/FaIDqQb7xQ2vcrdlwxjfThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>
<body>
    <aside data-th-replace="~{freelancer/freelancerdashboard :: aside}"></aside>
    <main class="content">
        <header>
            <h1>Welcome, Freelancer</h1>
        </header>
        <section class="projects">
            <h1>Your Service Requests</h1>
            <!-- <button class="add-project-btn" onclick="openModal()">Add New Project</button> -->
            <table>
                <thead>

```

```

<tr>
    <th>Client Name</th>
    <th>Project Title</th>
    <th>Image</th>
    <th>Price</th>
    <th>Category</th>
    <th>Status</th>
    <th>Action</th>
</tr>
</thead>
<tbody id="projectTable">
    <tr data-th-each="request : ${freelancer.serviceRequests}">
        <td data-th-text="${request.client.userName}"></td>
        <td data-th-text="${request.project.projectTitle}"></td>
        <td>Image</td>
        <td data-th-text="${request.project.projectPrice}"></td>
        <td data-th-text="${request.project.category.categoryName}"></td>
        <td data-th-text="${request.serviceRequestStatus}"></td>
        <td>
            <span data-th-if="${request.serviceRequestStatus.name() == 'PENDING'}">
                <a class="cancel-btn"
                    href="/serviceRequest/approve?id=${request.serviceRequestId}&status=canceled | ">Cancel</a>
            </span>
            <span
                data-th-if="${request.serviceRequestStatus.name() == 'CANCELED' || request.serviceRequestStatus.name() == 'REJECTED'}">
                <i class="fa-solid fa-ban"></i>
            </span>
            <span data-th-if="${request.serviceRequestStatus.name() == 'APPROVED'}">
                <i class="fa-solid fa-circle-check"></i>
            </span>
        <td>
            <span data-th-if="${request.serviceRequestStatus.name() == 'APPROVED'}">
                <i class="fa-solid fa-circle-check"></i>
            </span>
        </td>
    </tr>
</tbody>

```

```
</td>
</tr>
</tbody>
</table>
</section>
</main>
</body>
</html>
```

JAVA ENTITY FOLDER FOR TABLES :-

AUDITABLE.JAVA

```
package edu.rims.pro_connect.entity;

import jakarta.persistence.Column;
import jakarta.persistence.MappedSuperclass;
import java.time.LocalDate;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@MappedSuperclass
public class Auditable {

    @Column(name = "created_date", updatable = false)
    private LocalDate createdDate;

    @Column(name = "updated_date")
    private LocalDate updatedDate;

    @Column(name = "created_by")
    private String createdBy;

    @Column(name = "updated_by")
    private String updatedBy;
}
```

CATEGORY.JAVA

```
package edu.rims.pro_connect.entity;
```

```
import java.util.List;
import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
@Entity
@Getter
@Setter
@Table(name = "category")
public class Category extends Auditable {
    @Id
    @Column(name = "category_id")
    @GeneratedValue(strategy = GenerationType.UUID)
    private String categoryId;
    @Column(name = "category_name", nullable = false)
    private String categoryName;
    @Column(name = "category_description", nullable = false, columnDefinition = "TEXT")
    private String categoryDescription;
    @Column(name = "category_image_url", length = 250)
    private String categoryImageUrl ;
    @Column(name = "category_status", nullable = false, columnDefinition = "TEXT")
    private String categoryStatus;
    @OneToMany(mappedBy = "category")
    private List<Project> projects;
}
```

CLIENT.JAVA

```
package edu.rims.pro_connect.entity;

import java.util.ArrayList;
import java.util.List;

import jakarta.persistence.CascadeType;
```

```
import jakarta.persistence.Entity;
import jakarta.persistence.FetchType;
import jakarta.persistence.OneToMany;
import jakarta.persistence.PrimaryKeyJoinColumn;
import jakarta.persistence.Table;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter
@Setter
@Table(name = "client")
@PrimaryKeyJoinColumn(name = "user_id")
public class Client extends User {

    @OneToMany(mappedBy = "client", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<Project> projects = new ArrayList<>();

    @OneToMany(mappedBy = "client", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<ServiceRequest> serviceRequests = new ArrayList<>();

    @OneToMany(mappedBy = "client", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<ProjectRequest> projectRequests = new ArrayList<>();

    public void addProjectRequest(ProjectRequest projectRequest) {
        if (getProjectRequests() == null) {
            setProjectRequests(new ArrayList<>());
        }
        getProjectRequests().add(projectRequest);
        projectRequest.setClient(this);
    }
}
```

```
}
```

PROJECT.JAVA

```
package edu.rims.pro_connect.entity;

import java.util.List;

import jakarta.persistence.*;

import lombok.Getter;
import lombok.Setter;

@Entity
@Getter
@Setter
@Table(name = "project")

public class Project extends Auditable {

    @Id
    @GeneratedValue(strategy = GenerationType.UUID) // Generates UUID automatically
    @Column(name = "project_id", length = 36, updatable = false, nullable = false)
    private String projectId;

    @Column(name = "project_title", nullable = false, length = 255)
    private String projectTitle;

    @Column(name = "project_description", columnDefinition = "TEXT")
    private String projectDescription;

    @Column(name = "project_image_url", length = 250)
    private String projectImageUrl;

    @Column(name = "project_status", nullable = false, length = 50)
    private String projectStatus;

    @ManyToOne
    @JoinColumn(name = "client_id", nullable = false)
    private Client client;

    @ManyToOne
    @JoinColumn(name = "freelancer_id", nullable = true) // Nullable because a project might not have a
freelancer//
```

```
// assigned initially

private Freelancer freelancer;

@ManyToOne

@JoinColumn(name = "category_id", nullable = false)

private Category category;

@OneToOne(mappedBy = "project", cascade = CascadeType.ALL, orphanRemoval = true)

private Payment payment;

@Column(name = "project_price")

private Double projectPrice;

@OneToMany(mappedBy = "project")

private List<ServiceRequest> serviceRequests;

@OneToMany(mappedBy = "project")

private List<ProjectRequest> projectRequests;

}
```

PROJECTREQUEST.JAVA

```
package edu.rims.pro_connect.entity;

import edu.rims.pro_connect.constant.ProjectRequestStatus;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import lombok.Getter;
import lombok.Setter;
```

```
@Entity
@Getter
@Setter
public class ProjectRequest extends Auditable {
```

```
@Id
```

```

@GeneratedValue(strategy = GenerationType.UUID)
private String projectRequestId;

@ManyToOne
private Client client;

@ManyToOne
private Freelancer freelancer;

@ManyToOne
private Project project;

private ProjectRequestStatus projectRequestStatus = ProjectRequestStatus.PENDING;
}

```

SERVICEREQUEST.JAVA

```

package edu.rims.pro_connect.entity;

import edu.rims.pro_connect.constant.ServiceRequestStatus;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;
import lombok.Getter;
import lombok.Setter;

@Entity
@Getter
@Setter
public class ServiceRequest extends Auditable {
    @Id

```

```

    @GeneratedValue(strategy = GenerationType.UUID)
    private String serviceRequestId;

    @ManyToOne
    private Client client;

    @ManyToOne
    private Freelancer freelancer;

    @ManyToOne
    private Project project;

    private ServiceRequestStatus serviceRequestStatus = ServiceRequestStatus.PENDING;
}

```

CONTROLLERS :-

ADMINDASHBOARDCONTROLLER.JAVA

```

package edu.rims.pro_connect.controller;

import java.io.FileInputStream;
// import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.List;
import java.util.UUID;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

```

```
import org.springframework.web.bind.annotation.ResponseBody;

import edu.rims.pro_connect.constant.CategoryStatus;
import edu.rims.pro_connect.constant.UserType;
import edu.rims.pro_connect.dto.CategoryResponseDTO;
import edu.rims.pro_connect.entity.Category;
import edu.rims.pro_connect.entity.Freelancer;
import edu.rims.pro_connect.entity.User;
import edu.rims.pro_connect.repository.CategoryRepository;
import edu.rims.pro_connect.repository.FreelancerRepository;
import edu.rims.pro_connect.repository.UserRepository;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
//import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.multipart.MultipartFile;

@Controller
@RequestMapping("/admin")
public class AdmindashboardController {

    private final FreelancerRepository freelancerRepository;

    @Autowired
    private CategoryRepository categoryRepository;

    @Autowired
    private UserRepository userRepository;
```

```

AdmindashboardController(FreelancerRepository freelancerRepository) {
    this.freelancerRepository = freelancerRepository;
}

@GetMapping("/admindashboard")
String admindashboard() {
    return "admin/admindashboard";

}

@GetMapping("/category")
String admincategory(Model model) {
    List<Category> categories = categoryRepository.findAll();
    model.addAttribute("categories", categories);
    return "admin/category";
}

@PostMapping("/admin/category")
public String categoryAdd(@ModelAttribute Category category, @RequestParam("categoryImageFile")
    MultipartFile file, @RequestParam(required = false) String categoryImages)
    throws IOException {

    String categoryId = category.getCategoryId();
    category.setCategoryId(categoryId == null || categoryId.isEmpty() ? null : categoryId);
    if (!file.isEmpty()) {
        String originalFilename = file.getOriginalFilename();
        String extName = originalFilename.substring(originalFilename.lastIndexOf("."));
        String fileName = "upload_images/" + UUID.randomUUID().toString() + extName;
        FileOutputStream fileOutputStream = new FileOutputStream(fileName);
        fileOutputStream.write(file.getBytes());
        fileOutputStream.close();
    }
}

```

```

category.setCategoryImageUrl(fileName);

}else if(categoryImages != null) {

    category.setCategoryImageUrl(categoryImages);

}

categoryRepository.save(category);

return "redirect:/admin/category";

}

}

@GetMapping(value = "/category/image/{categoryId}", produces = { "image/jpg", "image/jpeg", "image/png"})
)

@ResponseBody

public byte[] getImage(@PathVariable String categoryId) throws IOException {

    Category category = categoryRepository.findById(categoryId).orElseThrow();

    String categoryImageUrl = category.getCategoryImageUrl();

    if(categoryImageUrl == null || categoryImageUrl.startsWith("http")) {

        return null;

    }

    FileInputStream fis = new FileInputStream(categoryImageUrl);

    byte[] image = fis.readAllBytes();

    fis.close();

    return image;

}

}

@GetMapping("/category/delete")

public String deleteCategory(@RequestParam("category") String categoryId) {

    Category category = categoryRepository.findById(categoryId).orElseThrow();

    category.setCategoryStatus(CategoryStatus.UNAVAILABLE.toString());

    categoryRepository.save(category);

    return "redirect:/admin/category";

}

```

```

    @GetMapping("/usermanagment")
    String adminusermanagment(Model model) {
        List<User> users = userRepository.findByUserType(UserType.CLIENT);
        model.addAttribute("users", users);
        return "admin/usermanagment";
    }

    @GetMapping("/freelancermanagment")
    String adminFreelancermanagment(Model model) {
        List<Freelancer> freelancers = freelancerRepository.findByUserType(UserType.FREELANCER);
        model.addAttribute("freelancers", freelancers);
        return "admin/freelancermanagment";
    }

    @GetMapping("/report")
    String adminreport() {
        return "admin/report";
    }

    @GetMapping("/category/{categoryId}")
    @ResponseBody
    public CategoryResponseDTO getcategory(@PathVariable String categoryId){
        Category category=categoryRepository.findById(categoryId).orElseThrow();
        CategoryResponseDTO dto = new CategoryResponseDTO();
        dto.setCategoryId(categoryId);
        dto.setCategoryName(category.getCategoryName());
        dto.setCategoryDescription(category.getCategoryDescription());
        dto.setCategoryImageUrl(category.getCategoryImageUrl());
        dto.setCategoryStatus(category.getCategoryStatus());
        return dto;
    }

```

```
}
```

```
}
```

CLIENTDASHBOARDCONTROLLER.JAVA

```
package edu.rims.pro_connect.controller;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.Principal;
import java.time.LocalDate;
import java.util.List;
import java.util.UUID;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;

import edu.rims.pro_connect.constant.CategoryStatus;
import edu.rims.pro_connect.constant.ProjectStatus;
import edu.rims.pro_connect.dto.ProjectResponseDTO;
import edu.rims.pro_connect.dto.ProjectResponseDTO.CategoryResponse;
import edu.rims.pro_connect.entity.Category;
import edu.rims.pro_connect.entity.Client;
```

```
import edu.rims.pro_connect.entity.Project;
import edu.rims.pro_connect.entity.User;
import edu.rims.pro_connect.repository.CategoryRepository;
import edu.rims.pro_connect.repository.ClientRepository;
import edu.rims.pro_connect.repository.ProjectRepository;
import edu.rims.pro_connect.repository.UserRepository;
import edu.rims.pro_connect.service.UserService;

@Controller
@RequestMapping("/client")
public class ClientdashboardController {
    @Autowired
    private CategoryRepository categoryRepository;

    @Autowired
    private ProjectRepository projectRepository;

    @Autowired
    private ClientRepository clientRepository;

    @Autowired
    private UserService userService;

    @GetMapping("/clientdashboard")
    String clientdashboard() {
        return "client/clientdashboard";
    }

    @GetMapping("/profile")
    String clientprofile(Principal principal, Model model) {
        Client client = clientRepository.findByUserEmail(principal.getName());
    }
}
```

```

model.addAttribute("user", client);

return "client/profile";

}

@GetMapping("/request")
String clientrequest(Model model) {

    User user = clientRepository.findById(1).orElseThrow();

    model.addAttribute("user", user);

    return "client/request";

}

@GetMapping("/myproject")
String clientmyproject(Model model, Principal principal) {

// User user = userService.getUser(principal.getName());

    Client user1 = clientRepository.findByUserEmail(principal.getName());

    List<Category> categories = categoryRepository.findAll();

    model.addAttribute("user", user1);

    model.addAttribute("categories", categories);

    return "client/myproject";

}

@PostMapping("/myproject")
String clientmyproject(@ModelAttribute Project project, @RequestParam("projectImage") MultipartFile file,
                      @RequestParam("projectImageName") String imageName, Principal principal)
throws IOException {

    String projectId = project.getProjectId();

    project.setProjectId(projectId == null || projectId.isEmpty() ? null : projectId);

    if (!file.isEmpty()) {

        String originalFilename = file.getOriginalFilename();

```

```

        String extName = originalFilename.substring(originalFilename.lastIndexOf("."));

        String fileName = "upload_images/" + UUID.randomUUID().toString() + extName;
        FileOutputStream fileOutputStream = new FileOutputStream(fileName);
        fileOutputStream.write(file.getBytes());
        fileOutputStream.close();

        project.setProjectImageUrl(fileName);

    } else if (imageName != null) {
        project.setProjectImageUrl(imageName);
    }

    int userId = userService.getUserId(principal.getName());
    Client client2 = clientRepository.findById(userId).orElseThrow();
    project.setClient(client2);
    project.setCreatedDate(LocalDate.now());
    project.setProjectStatus(ProjectStatus.OPEN.toString());
    projectRepository.save(project);
    return "redirect:/client/myproject";
}

@GetMapping("/myproject/{projectId}")
@ResponseBody
byte[] getImage(@PathVariable String projectId)
    throws IOException {
    Project project = projectRepository.findById(projectId).orElseThrow();
    String imageUrl = project.getProjectImageUrl();
    if (imageUrl == null) {
        return null;
    }
    FileInputStream fis = new FileInputStream(imageUrl);
    return fis.readAllBytes();
}

```

```

@GetMapping("/myproject/delete")
public String deletemyproject(@RequestParam("project") String projectId) {
    Project project = projectRepository.findById(projectId).orElseThrow();
    project.setProjectStatus(ProjectStatus.CANCELED.toString());
    projectRepository.save(project);
    return "redirect:/client/myproject";
}

@GetMapping("/reviewrating")
String clientreviewrating() {
    return "client/reviewrating";
}

@GetMapping("/projectRequest")
String clientProjectRequest(Model model) {
    User user = clientRepository.findById(1).orElseThrow();
    model.addAttribute("user", user);
    return "client/projectRequest";
}

@GetMapping("/myprojectedit/{id}")
@ResponseBody
ProjectResponseDTO myProjectEdit(@PathVariable String id) {
    Project project = projectRepository.findById(id).orElseThrow();

    ProjectResponseDTO dto = new ProjectResponseDTO();
    dto.setProjectId(project.getProjectId());
    dto.setProjectTitle(project.getProjectTitle());
    dto.setProjectPrice(project.getProjectPrice());
}

```

```

        dto.setProjectDescription(project.getProjectDescription());
        dto.setProjectImageUrl(project.getProjectImageUrl());
        dto.setProjectStatus(project.getProjectStatus());
        CategoryResponse categoryResponse = dto.new CategoryResponse();
        categoryResponse.setCategoryId(project.getCategory().getCategoryId());
        categoryResponse.setCategoryName(project.getCategory().getCategoryName());
        dto.setCategory(categoryResponse);
        return dto;
    }
}

```

FREELANCERBOARDCONTROLLER.JAVA

```

package edu.rims.pro_connect.controller;

import java.security.Principal;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import edu.rims.pro_connect.entity.Freelancer;

import edu.rims.pro_connect.entity.ServiceRequest;

import edu.rims.pro_connect.entity.User;

import edu.rims.pro_connect.repository.FreelancerRepository;

import edu.rims.pro_connect.repository.ProjectRepository;

import edu.rims.pro_connect.repository.ServiceRequestRepository;

import jakarta.transaction.Transactional;

@Controller

@RequestMapping("/freelancer")

public class FreelancerdashboardController {

```

```

@Autowired
private ProjectRepository projectRepository;

@Autowired
private FreelancerRepository freelancerRepository;

@Autowired
private ServiceRequestRepository serviceRequestRepository;

@GetMapping("/freelancerdashboard")
String freelancerdashboard() {
    return "freelancer/freelancerdashboard";
}

@Transactional
@GetMapping("/project")
String freelancerproject(Principal principal, Model model) {
    Freelancer freelancer = freelancerRepository.findByUserEmail(principal.getName());
    System.out.println(freelancer.getProjects().size());
    model.addAttribute("freelancer", freelancer);
    return "freelancer/project";
}

@GetMapping("/earning")
String freelancerearning(Principal principal, Model model) {
    Freelancer freelancer = freelancerRepository.findByUserEmail(principal.getName());
    model.addAttribute("projects", freelancer.getProjects());
    return "freelancer/earning";
}

@Transactional
@GetMapping("/servicerequest")
String freelancerServiceRequest(Principal principal, Model model) {
    Freelancer freelancer = freelancerRepository.findByUserEmail(principal.getName());
    model.addAttribute("freelancer", freelancer);
    System.out.println(freelancer.getServiceRequests().size());
    return "freelancer/servicerequest";
}

```

```

}

@Transactional
@GetMapping("/projectRequest")

String freelancerProjectRequest(Principal principal, Model model) {
    Freelancer freelancer = freelancerRepository.findByUserEmail(principal.getName());
    model.addAttribute("freelancer", freelancer);
    System.out.println(freelancer.getServiceRequests().size());
    return "freelancer/projectRequest";
}

@GetMapping("/profile")

String freelancerprofile(Principal principal, Model model) {
    Freelancer freelancer = freelancerRepository.findByUserEmail(principal.getName());
    model.addAttribute("freelancer", freelancer);
    return "freelancer/profile";
}

}

```

CLIENTCONTROLLER.JAVA

```

package edu.rims.pro_connect.controller;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.time.LocalDate;
import java.util.UUID;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;
import edu.rims.pro_connect.constant.UserType;
import edu.rims.pro_connect.entity.Client;

```

```

import edu.rims.pro_connect.entity.User;
import edu.rims.pro_connect.repository.ClientRepository;
import edu.rims.pro_connect.repository.UserRepository
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
@Controller
@RequestMapping("/client")
public class ClientController {
    @Autowired
    private ClientRepository clientRepository;
    @GetMapping("/home")
    String clienthome() {
        return "client/home";
    }
    @GetMapping("/solution")
    public String solution() {
        return "client/solution";
    }
    @GetMapping("/howitworks")
    public String howitworks() {
        return "client/howitworks";
    }
    @GetMapping("/contact")
    public String contact() {
        return "client/contact";
    }
    @PostMapping("/signup")
    String signUpClient(@ModelAttribute Client client, @RequestParam("userProfilePic") MultipartFile file)
        throws IOException {

```

```

if (!file.isEmpty()) {

    String originalFilename = file.getOriginalFilename();
    String extName = originalFilename.substring(originalFilename.lastIndexOf("."));

    String fileName = "upload_images/" + UUID.randomUUID().toString() + extName;
    FileOutputStream fileOutputStream = new FileOutputStream(fileName);
    fileOutputStream.write(file.getBytes());
    fileOutputStream.close();

    client.setUserProfilePicture(fileName);
}

client.setCreatedDate(LocalDate.now());
client.setUserType(UserType.CLIENT);
clientRepository.save(client);

return "redirect:/login/login";
}

@GetMapping("/image/{id}")
@ResponseBody
byte[] clientGetImage(@PathVariable int id) throws IOException {
    User user = clientRepository.findById(id).orElseThrow();
    String image = user.getUserProfilePicture();

    FileInputStream fileInputStream = new FileInputStream(image);

    return fileInputStream.readAllBytes();
}
}

```

CATEGORY.JAVA

```

package edu.rims.pro_connect.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

```

```
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

import edu.rims.pro_connect.constant.CategoryStatus;
import edu.rims.pro_connect.constant.ProjectStatus;
import edu.rims.pro_connect.entity.Category;
import edu.rims.pro_connect.entity.Project;
//import edu.rims.pro_connect.entity.Freelancer;
import edu.rims.pro_connect.repository.CategoryRepository;
import edu.rims.pro_connect.repository.ProjectRepository;

@Controller
@RequestMapping("/category")
public class CategoryController {
    @Autowired
    private CategoryRepository categoryRepository;

    @Autowired
    private ProjectRepository projectRepository;

    @GetMapping("/category")
    public String category(Model model) {
        List<Category> categories = categoryRepository.findAll();

        model.addAttribute("categories", categories);
        return "client/category";
    }
    @GetMapping("/category/search")
}
```

```

public String searchCategory(@RequestParam("search") String categoryName, Model model) {

    List<Category> categories = categoryRepository.
        findByCategoryNameContainingIgnoreCaseAndCategoryStatus(categoryName,
        CategoryStatus.AVAILABLE.toString());

    model.addAttribute("categories", categories);

    return "client/category";
}

@GetMapping("/freelancer")
public String categoryProjectList(Model model, @RequestParam String categoryId) {

    List<Project> projects = projectRepository.findByCategoryCategoryIdAndProjectStatus(categoryId,
        ProjectStatus.OPEN.toString());

    model.addAttribute("projects", projects);

    return "client/project";
}

```

SECURITY CONFIGURATION FOR AUTHENTICATION

SECURITYCONFIG.JAVA

```

package edu.rims.pro_connect.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.Customizer;
// import org.springframework.security.config.Customizer;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import edu.rims.pro_connect.constant.UserType;

```

```

import edu.rims.pro_connect.entity.User;
import edu.rims.pro_connect.repository.UserRepository;
@Configuration
public class SecurityConfig {
    @Autowired
    private UserRepository userRepository;
    @Bean
    SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.authorizeHttpRequests(
            request -> request
                .requestMatchers("/client/sign_in", "/style/**", "/images/*", "/script/**",
                    "/category/category", "/freelancer/freelancer", "/client/home",
                    "/category/category/search", "/freelancer/freelancer/search", "/admin/category/image",
                    "/client/howitworks", "/freelancer/pdp",
                    "/client/solution", "/client/project",
                    "/client/contact", "/client/category/project", "/client/sign_up", "/freelancer/signup")
                .permitAll()
                .requestMatchers("/admin/*").hasRole(UserType.ADMIN.toString())
                .requestMatchers("/freelancer/*").hasRole(UserType.FREELANCER.toString())
                .requestMatchers("/client/**").hasRole(UserType.CLIENT.toString())
                .anyRequest().authenticated());
        http.formLogin(form -> form.loginPage("/client/sign_in").defaultSuccessUrl("/client/home"));
        http.logout(Customizer.withDefaults());
        return http.build();
    }
    @Bean
    PasswordEncoder encoder() {
        return new BCryptPasswordEncoder(12);
    }
}

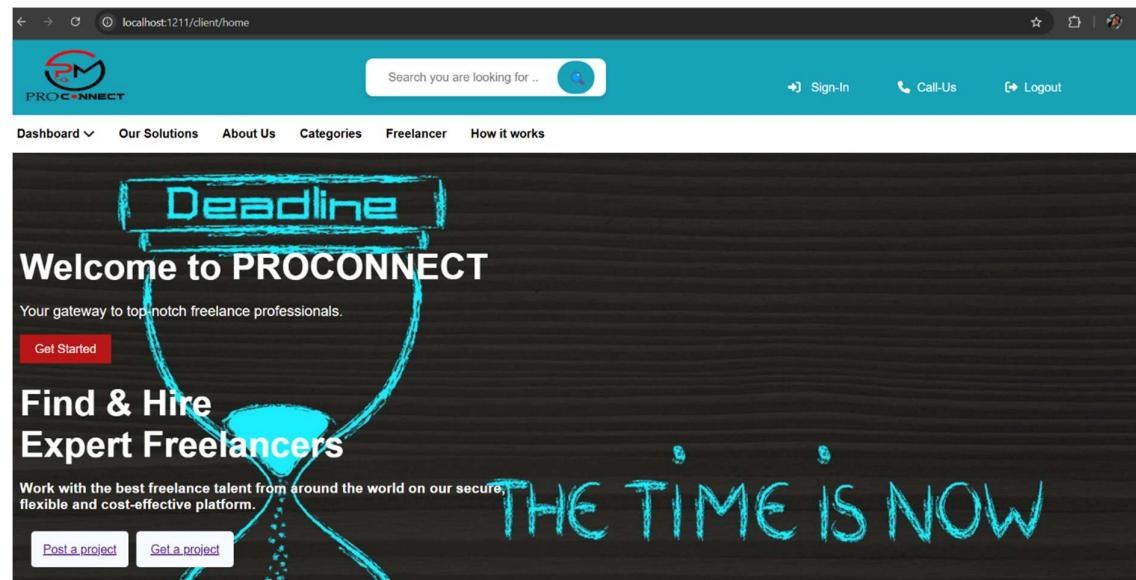
```

```
@Bean  
UserDetailsService detailsService() {  
    return (username) -> {  
        User user = userRepository.findByUserEmail(username)  
            .orElseThrow(() -> new UsernameNotFoundException("userNotFound"));  
  
        return org.springframework.security.core.userdetails.User.builder()  
            .username(username)  
            .password(user.getUserPassword())  
            .roles(user.getUserType().toString())  
            .build();  
    };  
}  
}
```

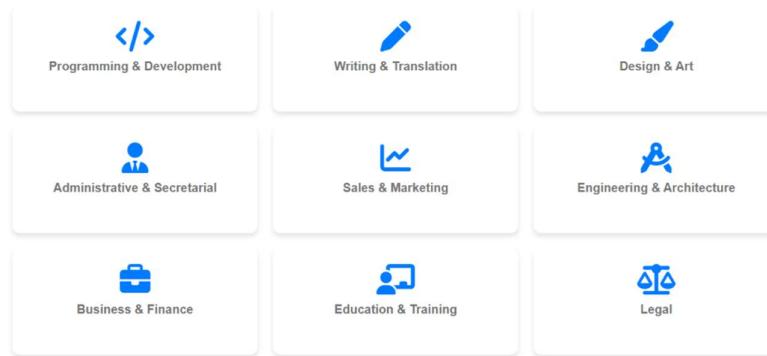
CHAPTER-13

SNOPTSHORT

Homepage



PINPOINT LEADING SKILLS



[ALL CATEGORIES](#)



Get in Touch with ProConnect

Whether you're a freelancer or a business, we're here to help. Please contact us using any of the methods below.

Email Us

For general inquiries, reach out to us at:

support@proconnect.com

If you're a business looking for freelancer solutions:

projects@proconnect.com

Message Form

Phone Support

If you need immediate assistance, feel free to call us at:

+1 (800)-7735870981

Our team is available Monday to Friday, from 9 AM to 6 PM.

Follow Us

Stay connected with us through social media:

Facebook Twitter LinkedIn Instagram

Visit Our Office

If you'd like to meet us in person, our office is located at:

123 Freelancer Lane, proconnect, California- Suite 200, Tech City, USA

Our website

our website not only give platform to everyone but also it provide essence to connect through out the world.

ProConnect website our concern to connect people and grow.

About Us

Learn more about our journey and how we connect talented freelancers with great opportunities!

Our Mission

Our mission is to empower freelancers across the globe by providing them with an easy-to-use platform that connects them with quality clients and projects. We strive to offer an environment that encourages growth, creativity, and professional development.

OUR STORY

We started as a small group of freelancers who shared a vision of creating a platform that would enable individuals to showcase their skills and find new opportunities. Over time, our platform has grown to serve thousands of freelancers and clients worldwide, creating a community built on trust, collaboration, and innovation.

MEET OUR TEAM



Anjani Shaw
Founder & CEO



Subhranjali Panda
Chief Operating Officer



Soumendra Behera
Lead Developer



Smruti Ranjan Nayak
Marketing Director

What Our Users Say

"This platform has been a game changer for my freelance career. It made it so easy to find high-quality clients and manage my projects efficiently."

- Sarah Lee, Freelancer

"As a client, I've been able to hire top talent for all my projects. I highly recommend this platform to anyone looking to find skilled professionals."

Get in Touch with ProConnect

Whether you're a freelancer or a business, we're here to help. Please contact us using any of the methods below.

Email Us

For general inquiries, reach out to us at:

support@proconnect.com

If you're a business looking for freelancer solutions:

projects@proconnect.com

Message Form

Name

message

Phone Support

If you need immediate assistance, feel free to call us at:

+1 (800)-7735870981

Our team is available Monday to Friday, from 9 AM to 6 PM.

Follow Us

Stay connected with us through social media:

[Facebook](#) [Twitter](#) [LinkedIn](#) [Instagram](#)

Visit Our Office

If you'd like to meet us in person, our office is located at:

123 Freelancer Lane, proconnect California - Suite 200, Tech City, USA

Our website

our website not only give platform to everyone but also it provide essence to connect through out the world.

ProConnect website our concern to connect people and grow.

MEET OUR DEVELOPERS



Jane Doe
UI/UX Designer

Expert in creating beautiful and functional user interfaces.

[Hire Now](#)



John Smith
Web Designer

Specializes in modern and responsive websites.

[Hire Now](#)



Emily Brown
Branding Expert

Experienced in logo and brand identity design.

[Hire Now](#)



David Lee
Graphic Designer

Creating stunning graphics for various digital platforms.

[Hire Now](#)

[SEE ALL](#)

Make it all happen with freelancer



Access a pool of top talent across 700 categories



Enjoy a simple, easy-to-use matching experience



Get quality work done quickly and within budget



Only pay when you're happy

[Join now](#)

Q & A

Q & A

What can I sell?	How much time will I need to invest?
How much money can I make?	How do I price my service?
How much does it cost?	How do I get paid?

ProConnect
 Connecting freelancers with businesses worldwide. Grow your career with ProConnect.

Quick Links

- [Home](#)
- [Browse Jobs](#)
- [Post a Job](#)
- [How It Works](#)
- [Contact Us](#)

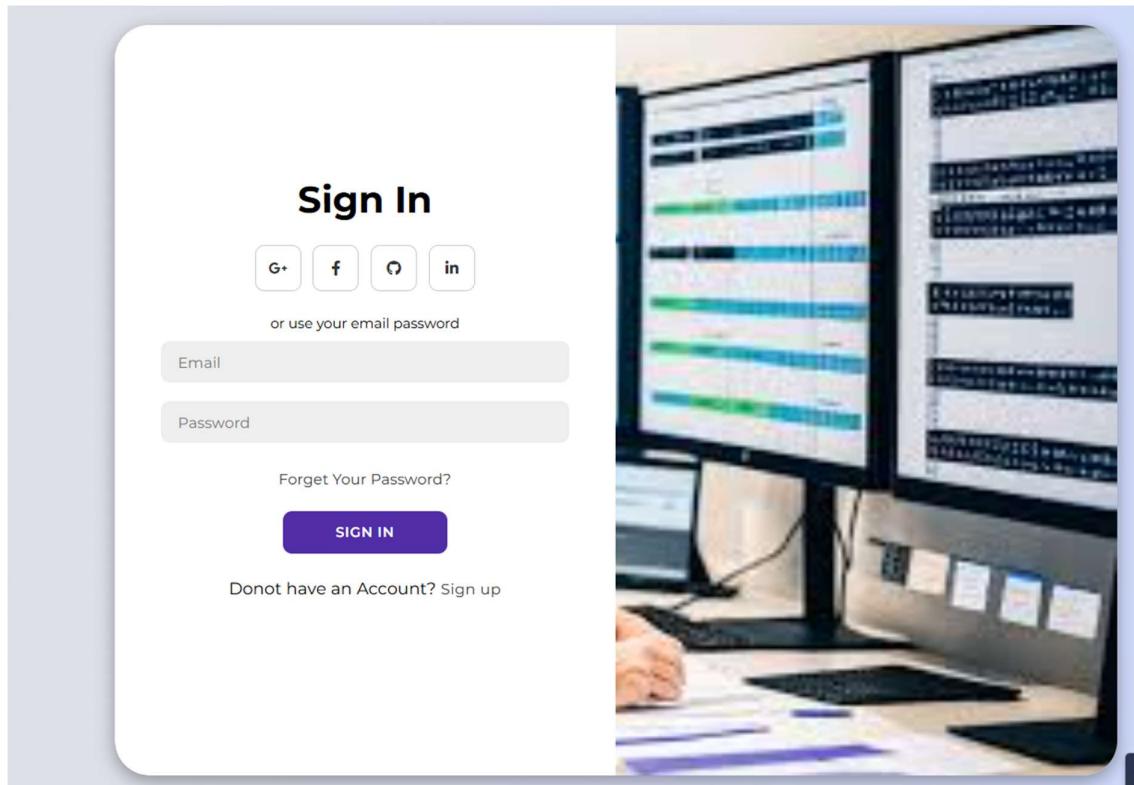
Support

- [Help Center](#)
- [Terms & Conditions](#)
- [Privacy Policy](#)
- [FAQs](#)

Follow Us

© 2025 ProConnect. All Rights Reserved.

SIGN IN PAGE



CALL US PAGE :

localhost:1211/client/contact

Contact Us

Freelancing made easy—reach out and let's make magic happen! "Got an idea? Let's turn it into reality—drop us a message!" "We're just a message away from making your project a success!"

[location] Address
405544 Sugar Camp Road,
Owatonna, Minnesota,
22545521

[phone] Phone
000-000-0000

[envelope] Email
temporaryEmail@dummy.domain

Connect with us

[Facebook] [Twitter] [Instagram] [LinkedIn]

Send Message

Full Name

Email

Type your Message...

SOLUTION PAGE

The screenshot shows a web browser window with the URL localhost:1211/client/solution. The page has a dark header with navigation links: Dashboard, Our Solutions, About Us, Categories, Freelancer, and How it works. A teal sidebar on the left contains sections for Solutions for Freelancers and Solutions for Clients, each with icons and descriptions. The main content area features a section titled "Empowering Freelancers and Clients" with a sub-section for "Solutions for Freelancers".

Our Solutions
Discover the powerful solutions we offer to help both freelancers and clients succeed.

Empowering Freelancers and Clients

Our platform connects freelancers with clients worldwide, offering innovative tools, secure payments, and a seamless experience.

Solutions for Freelancers

Work on Your Terms Choose the projects that align with your skills and interests. Set your own schedule and work from anywhere.

Build a Strong Portfolio Showcase your best work with ease. Create a professional portfolio to attract clients and increase your visibility.

Build a Strong Portfolio Showcase your best work with ease. Create a professional portfolio to attract clients and increase your visibility.

Get Paid Securely Enjoy fast and secure payment processing with multiple payment methods available. Track your earnings effortlessly.

Collaborate and Communicate Use our integrated messaging system to communicate with clients. Stay connected and organized throughout your projects.

Solutions for Clients

Find the Right Talent Search for top freelancers with the skills and experience that match your needs. Filter results based on expertise, location, and more.

Easy Collaboration Collaborate with freelancers seamlessly. Use our project management tools to assign tasks, track progress, and share feedback.

Secure Payments

Ensure that payments are made securely and only after satisfactory completion of milestones. Use our escrow service for added peace of mind.

Quality Assurance

Rely on our rating and review system to find freelancers who deliver quality work on time. Leave feedback to help others make informed decisions.

👍 Why Choose Us?

Our platform offers unique advantages to both freelancers and clients, making it the ideal place for meaningful collaborations.

Global Reach

Connect with clients and freelancers from around the world. We provide a global platform for people to work together no matter where they are located.

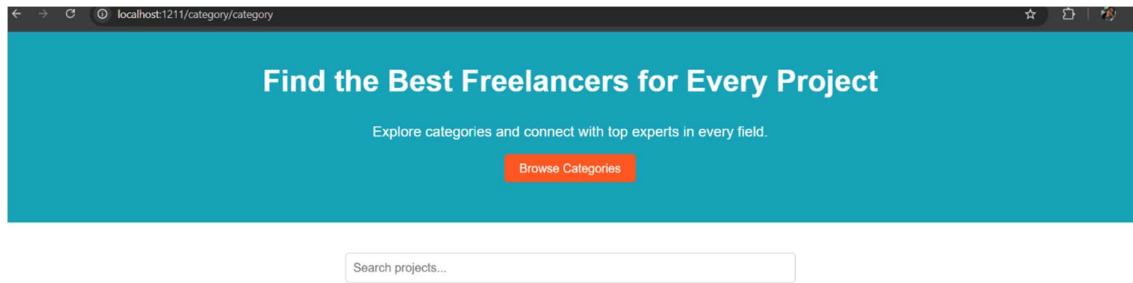
Comprehensive Tools

We provide a wide range of tools such as project management, time tracking, and secure messaging to streamline your work experience.

Secure Transactions

All payments are processed securely through our platform, ensuring safe transactions and fair compensation for freelancers.

CATEGORY PAGE:



localhost:1211/category/category

Find the Best Freelancers for Every Project

Explore categories and connect with top experts in every field.

Browse Categories

Search projects...

Browse Categories



 Graphic Design Creative design services, including logos and branding	 Frontend development Using HTML CSS JS	 Graphic Design jufhfd	 Web development API,JPA REACT	 Logo design logo design for any website to show case its best version.
--	---	--	---	---

FREELANCER PAGE:

localhost:1211/freelancer/freelancer

Hire the best developers

Search freelancers...

Puja dash
null\$
Experience: 2 years
HTML CSS

[See more](#)

priya sahu
null\$
Experience: 2 years
java, spring boot, php, javascript

[See more](#)

HOWITWORKS PAGE:-

localhost:1211/client/howitworks

How It Works

ProConnect is a powerful freelancing platform that connects clients with talented freelancers worldwide. Our platform ensures smooth project management, secure transactions, and professional collaboration.

1. Create an Account
Sign up for free and set up your freelancer or client profile. Customize your portfolio, skills, and experience.

[Get Started](#)

2. Post or Find Jobs
Clients post jobs with detailed descriptions, and freelancers browse and apply for suitable projects.

3. Submit Proposals
Freelancers craft compelling proposals with pricing, estimated completion time, and previous work samples.

4. Hire and Work
Clients review proposals, chat with freelancers, and hire the best fit. Freelancers then begin working on the project.

5. Secure Payments
Payments are held securely in escrow and released when milestones or projects are completed successfully.

6. Review & Ratings
Clients and freelancers provide feedback and ratings to maintain a trusted community of professionals.

7. Dispute Resolution
Our support team assists in resolving disputes and ensuring fair transactions between freelancers and clients.

8. Project Management Tools
Utilize built-in tools such as real-time chat, file sharing, progress tracking, and deadline reminders for better collaboration.

9. Skill Certification
Enhance credibility by completing skill tests and earning verified certifications on your profile.

ProConnect

Connecting freelancers with businesses worldwide. Grow your career with ProConnect.

Quick Links

- [Home](#)
- [Browse Jobs](#)
- [Post a Job](#)
- [How It Works](#)
- [Contact Us](#)

Support

- [Help Center](#)
- [Terms & Conditions](#)
- [Privacy Policy](#)
- [FAQs](#)

Follow Us

CLIENT DASHBOARD:-

The screenshot shows the ProConnect Client Dashboard. On the left sidebar, there are links for Dashboard, My Projects, Service Request, Project Request, Profile, and Review and Rating, along with a Logout button. The main area features a "Client Dashboard" header and three cards: "Active Projects" (5), "Pending Payments" (\$2,300), and "Freelancers Hired" (12). Below these is a section titled "My Projects" with a table:

Project Name	Freelancer	Budget	Status
Website Development	John Doe	\$3,000	Ongoing
Logo Design	Jane Smith	\$500	Completed
SEO Optimization	Michael Brown	\$1,200	Pending Approval

MY PROJECT PAGE:-

The screenshot shows the "Welcome, Client" page under the "My Projects" section. The sidebar is identical to the Client Dashboard. The main area has a "My Projects" header and a table:

Project Title	Description	Listed Date	Image	Price	Status	Action
logo design	logo designing is one of the important aspect of a website it should be very attractive	2025-03-26		2400.0	COMPLETED	
Freelancing website	Freelancing is a platform where it provide various things like part-time job with permanent job	2025-03-26		9000.0	IN_PROGRESS	

SERVICE REQUEST PAGE:-

The screenshot shows a web browser window with the URL `localhost:1211/client/request`. The page title is "Welcome, Client". On the left, there is a dark sidebar menu with the following items: "Dashboard", "My Projects", "Service Request" (which is highlighted in blue), "Project Request", "Profile", "Review and Rating", and "Logout". The main content area is titled "Your Project Requests". It contains a table with the following columns: Freelancer Name, Project Title, Image, Price, Category, Status, and Action. There are four rows in the table, each representing a project request from "Puja dash".

Freelancer Name	Project Title	Image	Price	Category	Status	Action
Puja dash	Freelancing website		9000.0	Web development	CANCELED	Cancel
Puja dash	logo design		2400.0	Logo design	APPROVED	Approve cancel
Puja dash	Freelancing website		9000.0	Web development	APPROVED	Approve cancel
Puja dash	logo design		2400.0	Logo design	APPROVED	Approve cancel

PROJECT REQUEST PAGE:-

The screenshot shows a web browser window with the URL `localhost:1211/client/projectRequest`. The page title is "Welcome, Client". The sidebar menu is identical to the previous screenshot. The main content area is titled "Your Project Requests". It contains a table with the same structure as the previous screenshot, but the "Action" column now contains two buttons: "Approve" and "Reject". There are two rows in the table, each representing a project request from "Puja dash".

Freelancer Name	Project Title	Image	Price	Category	Status	Action
Puja dash	Freelancing website	Image	9000.0	Web development	PENDING	Approve Reject
Puja dash	logo design	Image	2400.0	Logo design	APPROVED	Approve Reject

PROFILE PAGE:-

The screenshot shows a web application interface. On the left is a dark sidebar with white icons and text for navigation: Dashboard, My Projects, Service Request, Project Request, Profile, Review and Rating, and Logout. The main content area has a header "ProConnect". It displays a user profile for "Soumendra behera" with a placeholder "User Picture". Below the profile picture, the user's name is shown in large bold letters. Underneath the name, there is contact information: "web developer", "Location:Cuttack", "Email:s@gmail.com", and "Phone:7657787987". A blue "Contact" button is located below this information. A section titled "Skills" lists "HTML", "CSS", "JavaScript", "React", and "Node.js" in blue boxes. Another section titled "Experience" lists bullet points: "5+ years in Web Development", "Worked with 100+ clients", and "Specializing in Frontend & Backend". A "Portfolio" section is partially visible at the bottom.

REVIEW AND RATING PAGE:-

The screenshot shows a web application interface. On the left is a dark sidebar with white icons and text for navigation: Dashboard, My Projects, Service Request, Project Request, Profile, Review and Rating, and Logout. The main content area has a header "ProConnect". It displays a section titled "Client Dashboard - Reviews & Ratings". Below this, there is a form titled "Submit a Review" with fields for "Your Name", "Your Email (Optional)", "Service Quality" (a dropdown menu), and a text area for "Write your review here...". A green "Submit Review" button is located at the bottom of the form. Below the form, there is a section titled "Recent Reviews" showing a single review entry: "John Doe (Service Quality)" with a yellow five-star rating icon, followed by the text "Excellent service and prompt response."

ADMINDASHBOARD PAGE:-

The screenshot shows the Admin Dashboard for ProConnect. On the left is a dark sidebar with navigation links: Dashboard, Category, User Management, Freelancer Management, Report and Analytics, and Logout. The main area is titled "Admin Dashboard" and contains four cards: "Users" (1,245), "Active Jobs" (320), "Earnings" (\$45,210), and "Disputes" (12). Below these is a section titled "Recent Job Listings" with a table:

Job Title	Client	Budget	Status
Web Developer Needed	John Doe	\$1,500	Active
Graphic Designer	Jane Smith	\$800	Completed
SEO Specialist	Michael Brown	\$1,200	Active
Tester	Mikal jaikson	\$1,000	Active

CATEGORY PAGE:

The screenshot shows the Category page of the Admin Dashboard. The sidebar is identical to the Admin Dashboard. The main area is titled "Admin Dashboard" and includes a search bar and a "Add Category" button. Below is a table of categories:

Category Name	Category Description	Image	Status	Action
Graphic Design	Creative design services, including logos and branding		AVAILABLE	<button>Edit</button> <button>Delete</button>
Frontend development	Using HTML CSS JS		AVAILABLE	<button>Edit</button> <button>Delete</button>
Graphic Design	jufhfd		AVAILABLE	<button>Edit</button> <button>Delete</button>
Web development	API,JPA REACT		AVAILABLE	<button>Edit</button> <button>Delete</button>
Logo design	logo design for any website to show case its best version.		AVAILABLE	<button>Edit</button> <button>Delete</button>

USERMANAGEMENT PAGE:-

ProConnect

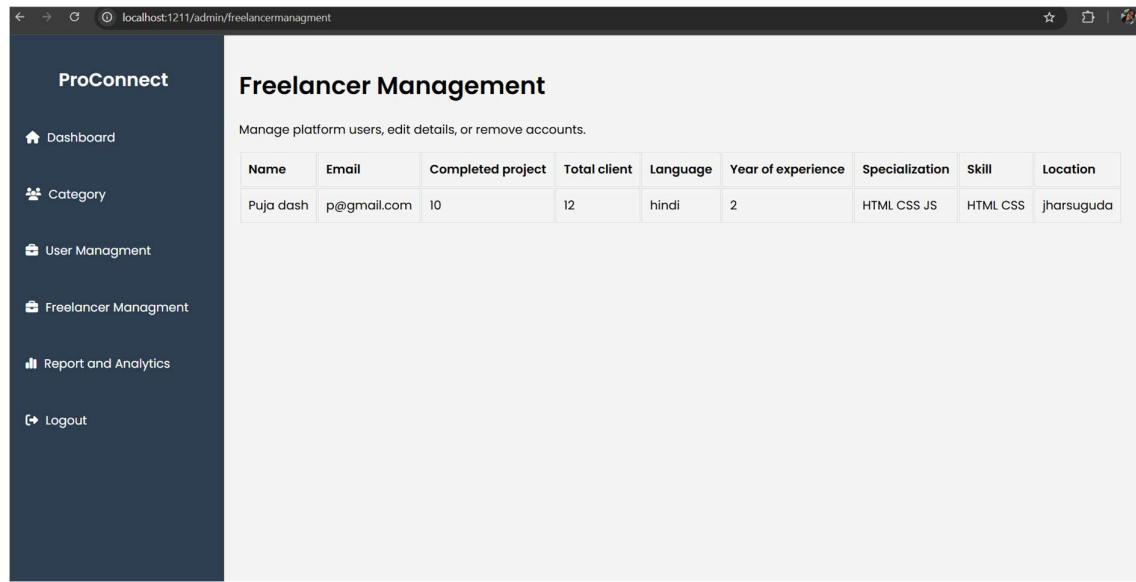
- [*Dashboard*](#)
- [*Category*](#)
- [*User Management*](#)
- [*Freelancer Management*](#)
- [*Report and Analytics*](#)
- [*Logout*](#)

User Management

Manage platform users, edit details, or remove accounts.

Name	Email	Role	Bio	Phone	Image	Location
Soumendra behera	s@gmail.com	CLIENT	web developer	7657787987		Cuttack
Smurti ranjan nayak	sr@gmail.com	CLIENT	senior developer	7657787987		Bhubaneswar

FREELANCER PAGE:-



The screenshot shows a browser window with the URL `localhost:1211/admin/freelancermanagement`. The page title is "Freelancer Management". The left sidebar is identical to the User Management page, featuring the ProConnect logo and navigation links. The main content area displays a table titled "Manage platform users, edit details, or remove accounts." with one row of data:

Name	Email	Completed project	Total client	Language	Year of experience	Specialization	Skill	Location
Puja dash	p@gmail.com	10	12	hindi	2	HTML CSS JS	HTML CSS	jharsuguda

REPORT AND ANALYTIC PAGE:

localhost:1211/admin/report

ProConnect

- Dashboard
- Category
- User Management
- Freelancer Management
- Report and Analytics
- Logout

Reports & Analytics

\$

Total Spend

\$12,540

Active Projects

4

Completed Projects

18

Spending Trend (Chart Placeholder)

Expense Breakdown (Chart Placeholder)

Invoices & Payments

Date	Project	Amount	Status
2025-03-15	Website Development	\$2,000	Paid
2025-03-10	Logo Design	\$500	Pending

Export as PDF Export as Excel Export as CSV

FREELANCER DASHBOARD PAGE:-

localhost:1211/freelancer/freelancerdashboard

ProConnect

- Dashboard
- Projects
- Earnings
- Service Request
- Project Complete Req.
- Profile
- Logout

Freelancer Dashboard

💼

Ongoing Projects

5

\$

Pending Payments

\$2,300

📝

Completed project

12

PROJECT PAGE:-

The screenshot shows the 'Earnings Overview' section with four colored boxes: Total Earnings (\$5000), Pending Earnings (\$700), Available Balance (\$3000), and Withdrawn (\$2000). Below this is the 'Transaction History' table:

Project Name	Date	Amount	Status
logo design	2025-03-26	2400.0	NA
Freelancing website	2025-03-26	9000.0	NA

EARNING PAGE:-

The screenshot shows the 'Freelancer Dashboard' with a table of projects:

Project Name	Project Description	Image	Status	Action
logo design	logo designing is one of the important aspect of a website it should be very attractive		COMPLETED	✓
Freelancing website	Freelancing is a platform where it provide various things like part-time job with permanent job		IN_PROGRESS	Completion Request

SERVICE REQUEST PAGE:-

The screenshot shows the 'Your Service Requests' table:

Client Name	Project Title	Image	Price	Category	Status	Action
Soumendra behera	Freelancing website	Image	9000.0	Web development	CANCELED	🔗
Soumendra behera	logo design	Image	2400.0	Logo design	PENDING	Cancel
Soumendra behera	Freelancing website	Image	9000.0	Web development	APPROVED	✓
Soumendra behera	logo design	Image	2400.0	Logo design	APPROVED	✓

PROJECT COMPLETE REQUEST PAGE:-

The screenshot shows the 'Your Project Requests' section of the ProConnect platform. On the left is a dark sidebar with navigation links: Dashboard, Projects, Earnings, Service Request, Project Complete Req. (which is highlighted in blue), Profile, and Logout. The main content area has a header 'Welcome, Freelancer' and a sub-header 'Your Project Requests'. Below this is a table with one row:

Client Name	Project Title	Image	Price	Category	Status	Action
Soumendra behera	logo design	Image	2400.0	Logo design	APPROVED	

PROFILE PAGE:-

The screenshot shows the profile page for a freelancer named Puja dash. The sidebar on the left is identical to the previous screenshot. The main content area displays the freelancer's profile information: name (Puja dash), title (frontend developer), location (Jharsuguda), email (p@gmail.com), and phone number (7978707224). There is a 'Contact' button below this information. Below the profile is a 'Skills' section with buttons for HTML, CSS, JavaScript, React, and Node.js. Further down is an 'Experience' section listing: 5+ years in Web Development, Worked with 100+ clients, and Specializing in Frontend & Backend. At the bottom is a 'Client Reviews' section containing a single review: 'Jane Smith: Great work, very professional!'.

PAYMENT PAGE:-

The screenshot shows a payment interface for a service named "Secure Payment". At the top, there is a navigation bar with the PROCONNECT logo, a search bar, and links for Sign-In, Call-Us, and Logout. Below the navigation bar, a menu bar includes Dashboard, Our Solutions, About Us, Categories, Freelancer, and How it works. The main content area features a "Secure Payment" header and a "Order Summary" section. The summary details a freelancer named Puja dash providing a logo design service at a price of \$2400.0. Below this, there are sections for "Billing Details" and "Payment Method", with a dropdown menu set to "Credit Card". A prominent green button labeled "Pay Now" with a credit card icon is at the bottom.

CHAPTER-12

TESTING

1. Introduction

Testing is a critical phase in the **ProConnect freelancing platform** to ensure that all functionalities work as expected, data integrity is maintained, and the user interface functions properly. The testing process ensures that:

- The **Hibernate Entity layer** properly maps and retrieves data from the database.
- The **Thymeleaf templates** display correct information dynamically.
- The **Spring Boot application** handles user interactions, authentication, and transactions seamlessly.

2. Unit Testing

Definition

Unit testing in **ProConnect** focuses on validating individual modules, such as **database operations, web templates (Thymeleaf), and business logic in Spring Boot**.

Tools Used:

- **Hibernate Entity Testing** – Ensures the proper mapping and retrieval of data.
- **Thymeleaf Rendering Tests** – Validates dynamic content rendering in HTML pages.
- **Spring Boot Test** – Used for integration testing of services, repositories, and controllers.

Unit Testing for Major Modules

Module	Tested Components	Expected Outcome
User Authentication	Hibernate Entity for User	User data is stored and retrieved correctly.
Project Management	Hibernate Entity for Projects	Project details are saved and fetched without errors.
Payment Processing	Payment Entity & Service	Transactions are stored securely.
Profile Management	Thymeleaf Profile Page	Correct user information is displayed.
Messaging System	Thymeleaf Message Templates	Messages are displayed dynamically.

3. Testing Scenarios

Functional Testing Scenarios

Test Scenario	Expected Result
User registers with valid details	Account is created successfully in the database.

Test Scenario	Expected Result
User logs in with correct credentials	Dashboard loads with accurate user details.
Client posts a new project	Project appears in the freelancer's dashboard.
Freelancer applies for a project	Request is stored in the ServiceRequest entity.
Client makes a payment	Transaction data is stored in the Payment entity.
Admin generates reports	The system retrieves correct data from the database.

Performance Testing Scenarios

Test Scenario	Expected Result
Thousands of database transactions	Hibernate handles data efficiently without lag.
Multiple users load the same page	Thymeleaf renders data quickly.
High volume of messaging	No delay in message display.

Security Testing Scenarios

Test Scenario	Expected Result
Unauthorized access to the admin panel	Access is denied.
SQL Injection attempt	Hibernate prevents direct database access.
Unauthenticated user tries to access a project	Redirected to the login page.

Testing ensures that **ProConnect** functions **efficiently, securely, and reliably**. With **Hibernate Entity testing, Thymeleaf validation, and Spring Boot integration tests**, the system is well-prepared for deployment.

Testing scenarios define different cases to verify that the **ProConnect freelancing platform** functions correctly across all modules, ensuring a seamless user experience. These scenarios cover **functional, performance, and security testing**.

CHAPTER-13

CONCLUSION

The **ProConnect Freelancing Platform** has been successfully developed as a **web-based application** that connects **clients and freelancers**, facilitating smooth project posting, bidding, collaboration, and payments. The system provides a structured and secure environment where:

- **Clients** can post projects, hire freelancers, and manage project progress efficiently.
- **Freelancers** can browse projects, submit proposals, and communicate with clients seamlessly.
- **Admin** can manage users, monitor transactions, and ensure platform integrity.

The project is built using **Spring Boot**, **Java**, **Hibernate**, **Thymeleaf**, **MySQL**, **HTML**, **CSS**, and **JavaScript**. It integrates **authentication mechanisms**, **secure data management**, and a **dynamic user interface**, making it a **reliable and scalable freelancing platform**.

Through **systematic testing and analysis**, the platform has been validated for functionality, performance, and security. The **user-friendly interface**, **structured workflow**, and **real-time communication system** make ProConnect an effective marketplace for freelance services.

Limitations

Despite its success, the project has certain limitations that can be addressed in future enhancements:

1. Limited Payment Integration

- Currently, the system lacks a **fully automated** payment gateway. Payments are recorded manually, which may limit real-time transactions.
- **Future Improvement:** Integrate third-party payment gateways like **PayPal**, **Stripe**, or **Razorpay** for seamless transactions.

2. Lack of Mobile Application

- The platform is **optimized for web browsers**, but there is no dedicated **mobile app**.
- **Future Improvement:** Develop a mobile app for both **Android** and **iOS** to increase accessibility and user engagement.

3. Basic Messaging System

- The current messaging system allows basic text-based communication but lacks features like **file sharing**, **real-time notifications**, and **video calls**.
- **Future Improvement:** Implement **real-time chat features** using **WebSockets** and **media sharing** options.

4. Performance Under Heavy Load

- While the system performs well under normal conditions, **high concurrent users** may slow down responses due to database queries.

- **Future Improvement:** Optimize **database queries**, use **caching mechanisms**, and consider **server load balancing** to improve scalability.

5. User Verification and Security

- The system lacks advanced **identity verification** mechanisms, which could lead to **fake profiles or fraudulent activities**.
- **Future Improvement:** Implement **KYC (Know Your Customer)** verification using email, phone number OTP verification, and government ID checks.

6. Limited Admin Features

- The admin panel **manages users and projects** but lacks **detailed analytics and reports**.
- **Future Improvement:** Develop a **dashboard with analytics** to track user engagement, revenue, and system performance.

CHAPTER-14

FUTURE ENHANCEMENT AND SCOPE

Future Scope

The ProConnect platform has great potential for future expansion. Enhancements such as AI-based job recommendations, automated invoicing, advanced filters, and international payment support can further improve user experience.

By addressing these limitations, ProConnect can evolve into a comprehensive and competitive freelancing platform, ensuring better opportunities for both clients and freelancers.

Future Enhancements

The ProConnect Freelancing Platform has been successfully developed, but there are several areas for future enhancements to improve user experience, security, and platform scalability. The following improvements can be considered in upcoming versions:

1. Advanced Payment Integration

- Current Limitation: Payments are manually recorded, with limited automation.
- Enhancement:
 - Integrate secure third-party payment gateways like PayPal, Stripe, Razorpay, or UPI for seamless and real-time transactions.
 - Implement automated invoice generation for freelancers.
 - Introduce an escrow system to ensure secure payments between clients and freelancers.

2. Mobile Application Development

- Current Limitation: The platform is only web-based and lacks a dedicated mobile app.
- Enhancement:
 - Develop a cross-platform mobile app (Android & iOS) using Flutter or React Native.
 - Enable push notifications for project updates and messages.
 - Optimize UI/UX for mobile users.

3. AI-Based Job Recommendation System

- Current Limitation: Job recommendations are manual, and users must search for projects.
- Enhancement:
 - Use Artificial Intelligence (AI) and Machine Learning (ML) to recommend projects based on:
 - Freelancer skills.

- Past work history.
 - Client preferences.
- Implement a smart bidding system to suggest optimal bids based on market trends.

4. Real-Time Chat & Video Conferencing

- Current Limitation: Basic text-based messaging without real-time updates.
- Enhancement:
 - Implement WebSockets for real-time messaging.
 - Add file-sharing support for documents and images.
 - Integrate video conferencing for better client-freelancer communication.

5. Enhanced Security & Verification

- Current Limitation: User verification is basic, and fraudulent profiles can be created.
- Enhancement:
 - Implement KYC (Know Your Customer) verification with email, phone number OTP, and government ID verification.
 - Strengthen data encryption for secure transactions and user privacy.
 - Add two-factor authentication (2FA) for extra security.

6. Advanced Analytics & Reports for Admin Panel

- Current Limitation: Admins can manage users but lack detailed insights.
- Enhancement:
 - Develop a comprehensive admin dashboard with:
 - User activity reports (active freelancers, projects completed).
 - Financial analytics (total transactions, freelancer earnings).
 - Performance monitoring (system uptime, page load speed).

7. Multi-Language & Global Payment Support

- Current Limitation: The platform supports only a single language and limited payment options.
- Enhancement:
 - Introduce multi-language support for a global audience.
 - Allow payments in multiple currencies to expand international reach.

8. Freelancer Skill Assessment & Certification

- Current Limitation: There is no verification of freelancer skills.
- Enhancement:
 - Introduce built-in skill assessments to certify freelancers.
 - Allow clients to filter freelancers based on certifications.

9. AI-Powered Dispute Resolution System

- Current Limitation: Dispute handling is manual and time-consuming.
- Enhancement:
 - Develop an AI-powered dispute resolution system that:
 - Analyzes chat and project history.
 - Suggests fair resolutions automatically.
 - Provide mediation tools for admins to intervene in disputes.

10. Blockchain-Based Contract & Payment Security

- Future Innovation:
 - Implement blockchain smart contracts for secure transactions.
 - Store contract details and payments on a decentralized network to prevent fraud.

These future enhancements will significantly improve the efficiency, security, and scalability of the ProConnect freelancing platform. By implementing AI-based job recommendations, real-time chat, advanced payment integration, and mobile applications, the platform can provide a seamless experience for clients, freelancers, and administrators.

CHAPTER-15

REFERENCES

Project has been done under guidance of Java technocrate by the help of the teachers and a senior developer and The following references were utilized to gather technical knowledge, best practices, and implementation strategies for the ProConnect Freelancing Platform:

1. Search Engines & Documentation

1. Google – Used for general research, troubleshooting, and finding relevant documentation.
 - o <https://www.google.com>
2. Spring Boot Official Documentation – Comprehensive reference for Spring Boot, MVC, security, and REST API development.
 - o <https://spring.io/projects/spring-boot>
3. MDN Web Docs – Used for HTML, CSS, JavaScript, and frontend-related concepts.
 - o <https://developer.mozilla.org>

2. Java, Spring Boot, Hibernate, and Database Resources

4. JavaTpoint (JT) – Referenced for Java programming, Spring Boot, Hibernate, and MySQL integration.
 - o <https://www.javatpoint.com>
5. GeeksforGeeks – In-depth explanations on Java, Spring Boot, Hibernate, and authentication mechanisms.
 - o <https://www.geeksforgeeks.org>
6. Baeldung – Advanced tutorials on Spring Boot, JPA, Hibernate, and REST APIs.
 - o <https://www.baeldung.com>
7. Oracle Java Documentation – Official guide for Java SE and EE development.
 - o <https://docs.oracle.com/en/java>
8. MySQL Documentation – Used for database setup, management, and SQL queries.
 - o <https://dev.mysql.com/doc>

3. Security & Authentication References

9. Spring Security Official Guide – Used for authentication, authorization, and role-based access control.
 - o <https://spring.io/guides/gs/securing-web>
10. OAuth 2.0 & JWT Authentication – For securing API endpoints.
 - o <https://jwt.io/introduction>
11. OWASP Security Practices – Used for web application security and vulnerabilities.

- <https://owasp.org>

4. UI/UX & Frontend Development

12. W3Schools – Quick references for HTML, CSS, JavaScript, and responsive design.

- <https://www.w3schools.com>

13. Bootstrap Official Documentation – Used for creating a responsive UI.

- <https://getbootstrap.com>

14. Tailwind CSS – Utility-first CSS framework used for styling.

- <https://tailwindcss.com>

5. Project Management & Version Control

15. Git & GitHub Documentation – Used for version control, repository management, and collaboration.

- <https://git-scm.com/doc>
- <https://docs.github.com>

6. YouTube Tutorials & Online Learning

16. Java Techie (YouTube Channel) – Used for Spring Boot, Hibernate, REST APIs, and authentication tutorials.

- <https://www.youtube.com/@JavaTechie>

17. Code With Durgesh (YouTube Channel) – Spring Boot and Hibernate tutorials.

- <https://www.youtube.com/@CodeWithDurgesh>

18. Programming with Mosh – Java, frontend, and backend development tutorials.

- <https://www.youtube.com/c/programmingwithmosh>

7. AI-Based Assistance & Research

19. ChatGPT (OpenAI) – Used for structuring project documentation, refining explanations, and debugging.

- <https://chat.openai.com>

20. Stack Overflow – Used for resolving coding issues and best practices in development.

- <https://stackoverflow.com>

8. Open-Source & Online Projects for Reference

21. GitHub Open-Source Freelancing Projects – Used to study the architecture of existing freelancing platforms.

- <https://github.com>

22. Online Freelancing Platform Case Studies – Research papers and real-world examples.

- [Upwork](#)

