Project Title

FreelanceFinder: Discovering Opportunities, Unlocking Potential

Team Members:

Here List team members and their roles:

1. **K. Ajay Samuel (Full Stack Developer):** Combines both frontend and backend responsibilities, ensuring smooth communication between the two. This role also handles bug fixing, feature integration, and overall system performance.
2. **CH. Basava Sambasivarao (Full Stack Developer):** Bridges frontend and backend tasks to maintain seamless interaction between them. This position also manages debugging, feature implementation, and optimizing system performance.
3. **M. Dinesh (Froentend Developer):** Responsible for designing the user interface using React.js. This role focuses on ensuring a responsive, user-friendly design, as well as integrating the frontend with backend APIs.
4. **J. Akhil (Backend Administration):** Develops the backend server using Node.js and Express.js, ensuring the creation of secure, scalable RESTful APIs, as well as handling authentication, data processing, and business logic.
5. **K. Viswandh Babu (Database Administration):** Manages the MongoDB database, focusing on schema design, data integrity, and database optimization to ensure efficient data storage and retrieval.

1 INTRODUCTION

In today's fast-paced digital landscape, the demand for freelance services has surged, creating a need for efficient platforms that connect clients with skilled professionals. FreelanceFinder emerges as a comprehensive solution tailored to bridge this gap, providing a user-friendly environment where freelancers can showcase their talents and clients can find the right expertise for their projects.

The platform is designed with both buyers and sellers in mind, ensuring that users have access to the tools necessary for effective collaboration. By offering features such as profile creation, portfolio management, and project tracking, FreelanceFinder enables freelancers to market their services while empowering clients to make informed hiring decisions. With an emphasis on security and user experience, FreelanceFinder implements robust authentication protocols to protect user information.

Additionally, the platform fosters community through a feedback system, allowing clients to share their experiences and help others in their hiring journey. As the freelance economy continues to evolve, FreelanceFinder stands at the forefront, facilitating meaningful connections and optimizing the way freelancers and clients engage. By streamlining the process of finding, hiring, and managing freelance talent, FreelanceFinder aims to revolutionize the freelance marketplace and enhance the overall quality of service delivery.

Furthermore, FreelanceFinder prioritizes accessibility, ensuring that the platform is easy to use for individuals across all levels of technical expertise. Regular updates and enhancements are planned to keep the platform aligned with emerging industry trends. Freelancers benefit from visibility into potential earnings and project timelines, empowering them to make better career choices. FreelanceFinder's commitment to fostering a supportive freelance ecosystem highlights its dedication to the professional growth of freelancers and the satisfaction of clients alike.

1.1 PROJECT IDEA

The **Freelance Finder** project aims to create an intuitive, accessible platform that bridges the gap between freelancers and clients, allowing them to connect easily for project-based work. The platform addresses the challenges freelancers face in finding relevant gigs and helps clients discover skilled professionals quickly. Through user-friendly features like profile creation, gig postings, search and filter options, and in-app messaging, the project facilitates smooth communication and efficient project management. Designed with a secure and scalable MERN (MongoDB, Express.js, React, Node.js) stack, Freelance Finder seeks to streamline the hiring process, saving time and improving outcomes for both freelancers and clients.

1.2 MOTIVATION OF PROJECT

MOTIVATION:

The primary motivation for developing the FreelanceFinder platform is to simplify and enhance the hiring process for freelancers and clients. In the traditional freelance marketplace, clients often face challenges such as lengthy searches, miscommunication, and difficulties in evaluating freelancer credentials. This can lead to frustration and inefficiency, ultimately impacting project timelines and outcomes. By creating a centralized system where freelancers can showcase their skills and clients can easily find and hire talent, we aim to eliminate these barriers. The goal is to facilitate a smoother and more efficient collaboration process that benefits both freelancers and clients alike. We believe that a user-friendly platform will empower freelancers to present their work effectively while providing clients with the tools they need to make informed decisions. Through FreelanceFinder, we aspire to create a thriving freelance ecosystem that promotes productivity and fosters positive working relationships.

IDEA BEHIND THE PROJECT:

The idea behind developing the FreelanceFinder platform centers on creating a centralized marketplace that integrates and streamlines the hiring process for freelancers and clients. This system aims to simplify the way clients search for and engage freelance talent by providing a comprehensive database of freelancer profiles, showcasing their skills, experiences.

1.3 PROBLEM STATEMENT

In today's dynamic gig economy, freelancers and clients encounter significant challenges that hinder effective collaboration and project execution. The traditional methods of finding and hiring freelance talent are often inefficient and cumbersome, leading to a range of issues:

- **Fragmented Communication**: Clients struggle to communicate their project needs clearly and effectively, resulting in misunderstandings and misaligned expectations. Freelancers may also find it difficult to showcase their skills and portfolios to potential clients.
- **Inefficient Search Processes**: Clients often waste time sifting through numerous profiles across different platforms, leading to frustration and missed opportunities. This disorganized approach can deter clients from engaging with freelancers altogether.
- **Lack of Trust and Accountability**: In the absence of a structured feedback and rating system, clients face challenges in assessing the reliability and quality of freelancers. This uncertainty can lead to hesitancy in hiring, ultimately affecting project timelines and outcomes.
- **Administrative Burden**: Freelancers frequently deal with administrative tasks such as invoicing, project tracking, and payment processing, diverting their focus from their core competencies. This inefficiency can impact their ability to deliver high-quality work in a timely manner.

Given these challenges, there is a pressing need for a comprehensive FreelanceFinder platform that streamlines the process of connecting freelancers with clients, enhances communication, and fosters a trusted environment for collaboration. By addressing these issues, FreelanceFinder aims to improve user experience and optimize the overall efficiency of the freelance marketplace.

1.4 STATEMENT OF SCOPE

The FreelanceFinder platform is designed to streamline the connection between freelancers and clients while ensuring a secure and efficient environment for both parties. This system will enable freelancers to showcase their skills, portfolios, and expertise, thereby attracting potential clients seeking specific services. FreelanceFinder will cater to various users, including freelancers across multiple industries and clients looking for specialized services, while implementing a robust user management system to ensure secure storage of user data and maintain privacy. The platform will feature user-friendly and intuitive modules that facilitate easy navigation and access to information. By incorporating tools for project management, messaging, and payment processing, FreelanceFinder aims to improve the overall efficiency of the freelance hiring process. Clients will benefit from the ability to search for freelancers based on specific criteria and access ratings and reviews from previous clients.

1.5 GOALS AND OBJECTIVES

The main goal of this project is to enhance the freelance marketplace by connecting clients with skilled freelancers efficiently and effectively. The primary objectives of this website development can be defined as follows:

- Facilitate and simplify the process of hiring freelancers to improve user experience.
- Centralize freelancer profiles and enable seamless communication between clients and freelancers.
- Provide tools for project management, allowing both parties to track progress and deadlines.
- Enhance client satisfaction by offering convenient access to a diverse range of freelance services.
- Automate key processes, reducing administrative burden and improving overall efficiency in freelance transactions.

2.1 LITERATURE SURVEY

The *Freelance Finder* project aims to develop a user-friendly platform connecting freelancers and clients for various project needs. The literature highlights the rapid growth of online freelancing platforms and the increasing demand for secure, reliable, and efficient gig-matching systems. Existing solutions like Upwork and Fiverr serve as benchmarks, showcasing critical features such as user authentication, profile verification, secure messaging, and feedback systems. However, research points to areas for improvement, such as data privacy, enhanced user experience, and personalization. Leveraging the MERN stack, this project will focus on creating a scalable and accessible system that addresses these gaps, emphasizing smooth navigation, secure data handling, and seamless communication between users. This project also aims to incorporate feedback and rating features to build trust and transparency on the platform.

2.2 EXISTING SYSTEM:

The existing systems, such as Upwork and Fiverr, have established themselves as leaders in the freelancing industry, offering comprehensive platforms where clients can find, hire, and manage freelancers for a wide variety of tasks. Upwork provides a structured process for job postings, bidding, and milestone-based payments, which helps both clients and freelancers track project progress. Fiverr, on the other hand, uses a gig-based model where freelancers advertise predefined services, allowing clients to browse and purchase services instantly. While these platforms are widely successful, they also present challenges, including high fees, intense competition, and occasional lack of personalization in the hiring process. These gaps present opportunities for improvement, such as offering more customized matching, enhanced data privacy, and reduced transaction costs, all of which are considerations in developing *Freelance Finder*. Additionally, both Upwork and Fiverr face issues with overcrowding, making it difficult for new freelancers to stand out and for clients to find suitable talent efficiently. Some users also report a lack of direct communication, as messaging and negotiation options are often limited until a project starts. These pain points highlight the need for a more streamlined, user-friendly platform that emphasizes transparency, ease of access, and effective client-freelancer communication, which *Freelance Finder* aims to address.

2.3 DISADVANTAGES OF EXSITING SYSTEM:

The existing freelance platforms, such as Upwork and Fiverr, have several disadvantages that can hinder the user experience for both freelancers and clients:

High Fees: Many platforms charge significant service fees, which can reduce freelancers' earnings and deter clients from hiring.

Overcrowding: The large number of freelancers competing for the same projects makes it challenging for new entrants to get noticed and secure work.

Limited Communication: Users often face restrictions on communication before a contract is established, leading to misunderstandings and misaligned expectations.

Quality Control Issues: Variability in the quality of freelancers can make it difficult for clients to find reliable talent. Poor experiences can affect overall trust in the platform.

Inconsistent Support: Many existing platforms have slow or ineffective customer support, leaving users frustrated when issues arise.

Rigid Project Structures: Platforms may enforce strict project guidelines, limiting creativity and flexibility in how freelancers can deliver their work.

2.4 FESABILITY STUDY:

A feasibility study is essential for assessing the practicality of the *Freelance Finder* project. This study will cover several key aspects to determine whether the project can be successfully developed and launched.

1.Technical Feasibility:

Technology Stack: The project will utilize the MERN (MongoDB, Express.js, React, Node.js) stack, which is widely used for building modern web applications. This technology is suitable for creating a responsive and dynamic platform.

Development Skills: The development team possesses the necessary skills in JavaScript, React, and backend technologies, ensuring the project can be executed effectively.

Integration: The system will integrate APIs for features such as payment processing and communication tools, which are feasible within the chosen technology.

2. Economic Feasibility

Cost Analysis: Initial costs will include development, hosting, marketing, and ongoing maintenance. A preliminary budget will be outlined to ensure sufficient funding.

Revenue Model: Potential revenue streams may include subscription fees for premium services, commission on transactions, or advertising, providing a clear path to profitability.

3. Operational Feasibility

User Experience: The platform aims to improve user satisfaction through an intuitive interface and streamlined processes for both freelancers and clients.

Market Demand: Research indicates a growing demand for freelance work, with many users dissatisfied with existing platforms. This presents an opportunity for *Freelance Finder* to capture market share.

4. Legal Feasibility

Compliance: The project will adhere to legal regulations, including data protection laws (e.g., GDPR) to ensure user data is handled responsibly.

Contractual Framework: Clear terms of service and user agreements will be established to protect both freelancers and clients.

3.1 PROPOSED SYSTEM:

The proposed Freelance Finder System aims to create a streamlined platform for connecting freelancers with clients. This system will facilitate communication between users, allowing clients to post job listings and freelancers to showcase their skills and bid on projects. The key features of the system will include:

- **User Registration and Profiles:** Both clients and freelancers can create accounts to manage their profiles, including personal details, work experience, and project portfolios.
- **Job Listings and Bidding:** Clients can post job opportunities, and freelancers can browse these listings, submit proposals, and communicate directly with clients to discuss project details.
- **Appointment Scheduling:** Users will have the ability to schedule meetings and consultations, ensuring efficient coordination between clients and freelancers.
- **Feedback and Ratings:** A feedback system will be implemented to allow clients and freelancers to rate each other based on their experiences, helping to build trust within the platform.

This proposed system eliminates the need for payment processing, focusing solely on fostering collaboration and effective communication between freelancers and clients

3.2 METHODOLY:

The development of the Freelance Finder System will follow the Agile methodology, enabling iterative progress through regular feedback and collaboration. Requirements will be gathered from potential users to ensure the system meets their needs effectively. The project will be divided into sprints, focusing on specific features such as user registration, job postings, and communication tools. Continuous testing will be implemented to identify and resolve issues promptly. Finally, user training and documentation will be provided to ensure a smooth transition to the new system.

3.3 ADVANTAGES:

The Freelance Finder System offers a user-friendly platform that connects clients with freelancers seamlessly, fostering efficient project collaboration. It eliminates the need for premium subscriptions, allowing users to access all features at no cost. The system provides real-time communication tools, enhancing interaction between clients and freelancers. Additionally, it supports a transparent feedback mechanism, helping users make informed decisions based on previous experiences. With its intuitive interface, users can easily navigate job postings and applications, streamlining the hiring process. Overall, the platform promotes accessibility and efficiency in the freelance marketplace.

3.4 APPROCHES:

The development of the Freelance Finder System will adopt a user-centric approach, focusing on creating an intuitive interface that enhances user experience for both clients and freelancers. Agile methodology will be employed to allow for iterative development, enabling regular feedback and quick adjustments based on user needs and market trends. Additionally, the project will leverage modern web technologies, including the MERN stack, to ensure scalability and responsiveness across various devices. Security measures will be prioritized to protect user data and maintain trust within the platform. Finally, a robust testing strategy will be implemented to ensure the system's functionality and reliability before the launch.

4.1 SOFTWARE REQUIREMENTS:

The Freelance Finder project requires a set of software tools and platforms to ensure smooth functionality and performance. The main software requirements include:

1. **Node.js**: A JavaScript runtime environment for building the backend server.
2. **Express.js**: A web application framework for Node.js that simplifies the server-side coding.
3. **MongoDB**: A NoSQL database to store user data, gigs, and other relevant information.
4. **Mongoose**: An ODM (Object Data Modeling) library for MongoDB and Node.js, providing a straightforward way to model data.
5. **React**: A JavaScript library for building user interfaces, particularly for the frontend of the application.
6. **SCSS**: A preprocessor scripting language that is interpreted into Cascading Style Sheets (CSS), allowing for more efficient styling of the application.

4.2 TECHNOLOGIES USED:

The technologies used in the Freelance Finder project focus on enhancing the user experience and ensuring efficient system operations. These include:

- **Frontend Technologies**: React for building dynamic and responsive user interfaces, coupled with SCSS for styling, ensuring a modern and visually appealing design.
- **Backend Technologies**: Node.js and Express.js to create a robust server-side environment, enabling the handling of HTTP requests and API development.
- **Database Technology**: MongoDB as a flexible database solution that accommodates the non-relational data structure required for user and gig management.
- **Version Control**: Git for tracking changes in the codebase and collaborating with other developers, ensuring a smooth workflow and version management.
- **API Testing**: Postman for testing API endpoints and ensuring they function correctly, providing a seamless interaction between the frontend and backend components.

4.3 HARDWARE REQUIREMENTS:

The hardware requirements for the Freelance Finder project ensure that both the development and deployment environments can support the necessary software and applications effectively. These include:

Development Environment:

- **Processor**: A dual-core processor (Intel i3 or equivalent) or higher to handle coding, compiling, and running local servers.
- **RAM**: At least 8 GB of RAM to support running multiple applications and processes simultaneously without lag.
- **Storage**: A minimum of 256 GB SSD for faster read/write speeds, which is beneficial for loading applications and managing project files.
- **Network**: A stable internet connection for downloading dependencies, accessing online resources, and collaborating on cloud platforms.

5.1 SYSTEM ARCHITECTURE:

The Freelance Finder project employs a **client-server architecture** based on the MERN stack (MongoDB, Express.js, React, Node.js). This architecture separates the front-end and back-end components, allowing for efficient communication and scalability. Below is an overview of the system architecture:

1. Client Side (Front-End):

- **Framework**: React
  - Manages the user interface and user interactions.
  - Handles routing with React Router for seamless navigation between different pages (e.g., Home, Login, Registration, Dashboard).
- **State Management**: Redux (if used)
  - Manages global application state, making it easier to share data across components.

2. Server Side (Back-End):

- **Framework**: Node.js with Express.js
  - Handles API requests from the client side.
  - Processes user authentication, manages sessions, and serves data.
- **Database**: MongoDB
  - Stores user data (clients and freelancers), project listings, and application-related information.
  - Utilizes collections for organized data management.

3. Communication:

- **RESTful API**:
  - The front end communicates with the back end through RESTful API endpoints.
  - Requests and responses are typically in JSON format, facilitating easy data interchange.

5.2 MODULE DESCRIPTION:

The Freelance Finder project consists of two primary modules: **Client Module** and **Freelancer Module**. Each module is designed to provide specific functionalities tailored to the needs of its respective users.

1. Client Module

This module allows clients (employers) to find and hire freelancers for various projects. Key functionalities include:

- **User Registration**: Clients can create an account by providing necessary details such as name,

email, and password.

- **Profile Management**: Clients can update their profiles, manage personal information, and set preferences for project notifications.
- **Post Projects**: Clients can create and post new projects, specifying the project details, budget, and deadlines.
- **Search and Filter**: Clients can search for freelancers based on skills, ratings, and availability. They can also filter results to find suitable candidates quickly.
- **View Proposals**: Clients can review proposals submitted by freelancers for their posted projects and communicate with them through the platform.
- **Contact Freelancers**: Clients can directly contact freelancers through messaging or email options to discuss project specifics.

2. Freelancer Module

This module is designed for freelancers to showcase their skills, apply for projects, and manage their work. Key functionalities include:

- **User Registration**: Freelancers can sign up by providing their personal information, including skills and areas of expertise.
- **Profile Setup**: Freelancers can create detailed profiles, including portfolio samples, hourly rates, and previous work experiences.
- **Browse Projects**: Freelancers can search and browse available projects, filtering by categories, budget, and deadlines.
- **Submit Proposals**: Freelancers can submit proposals for projects they are interested in, outlining their approach, budget, and timelines.
- **Manage Applications**: Freelancers can track the status of their submitted proposals and communicate with clients regarding project requirements.
- **Review and Rating System**: After project completion, freelancers can receive ratings and feedback from clients, helping them build a strong reputation on the platform.

5.3 DATA FLOW DIAGRAM (DFD)

5.3.1 Level-0

The context diagram (DFD level 0) shows the system's overall function in a single process. It is used to pinpoint the very purpose of the freelance finder system.

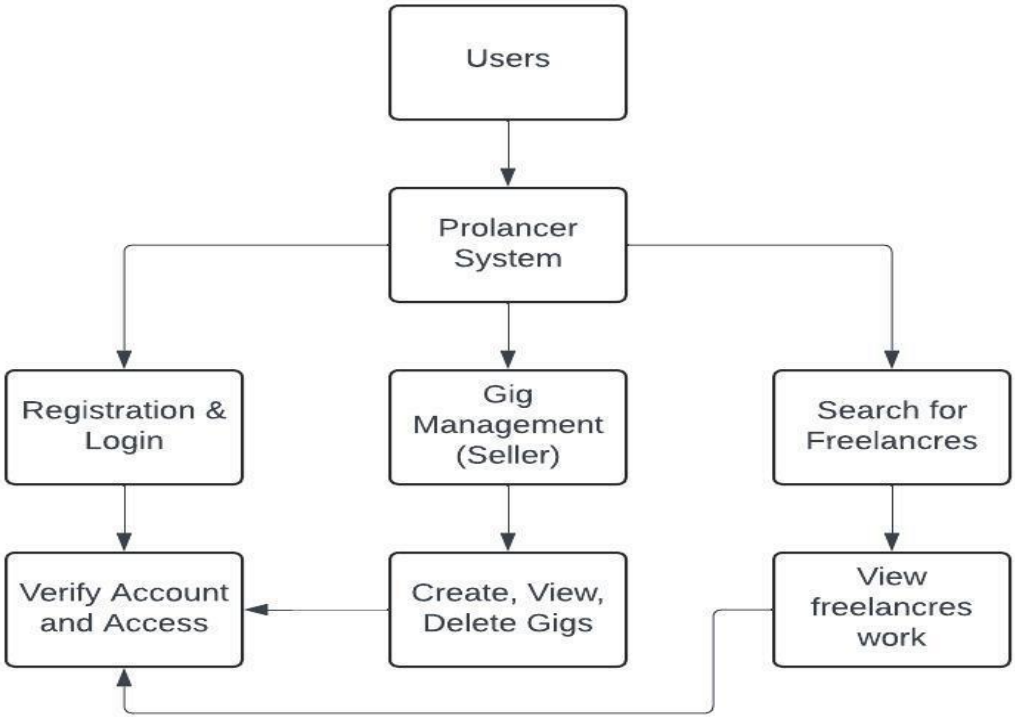


Fig – 5.3.1 Level-0 DFD Diagram for FREELACEFINDER

In the FreelanceFinder platform, the main function is invoked by inputs from external entities, including admins, clients (buyers), and freelancers (sellers). The data flow begins as soon as an input is provided, ensuring a seamless interaction between users and the system.

5.3.2 Level – 1:

FreelanceFinder Platform Sub-processes:

5.3.2.1 User Registration and Authentication

5.3.2.2 Gig Management

5.3.2.3 Freelance Search

5.3.2.4 Communication

These processes specify the paths (flows) of data that may enter and exit the system.

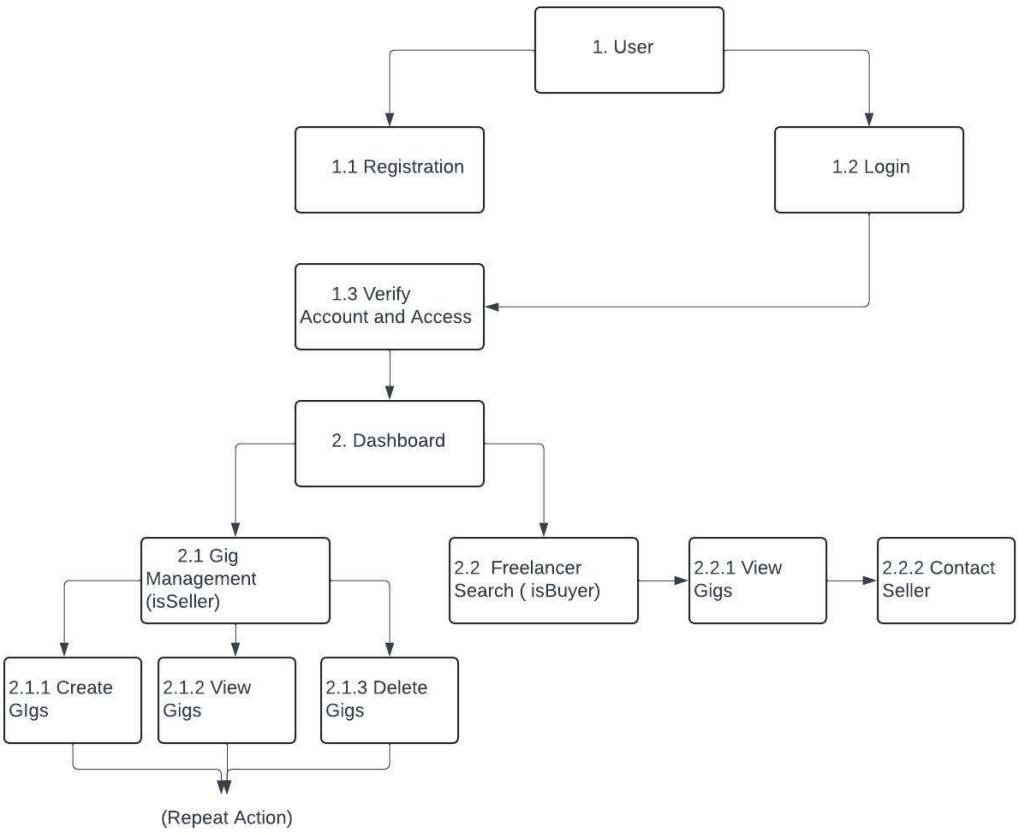


Fig – 5.3.2 Level – 1 DFD Diagram for FREELANCEFINDER

5.4 UML DIAGRAM:

5.4.1Class Diagram:

FreelanceFinder Platform Class Diagram - is a designed diagram the shows the system's relationships and classes. This UML Class Diagram is made to guide programmers along with the freelance system development.

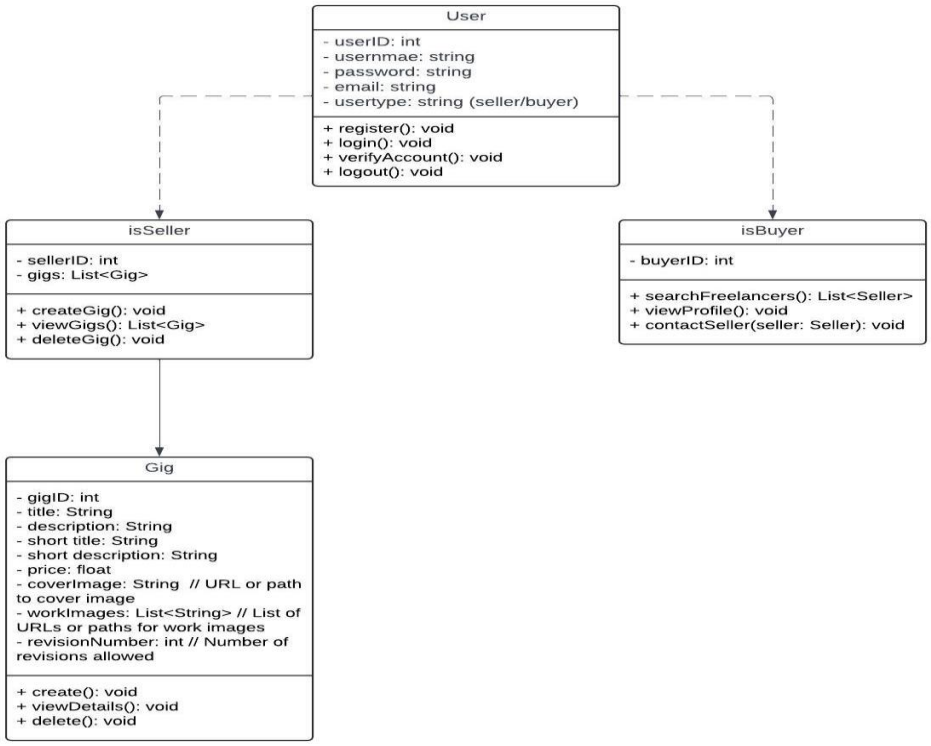


FIG 5.4.1 Class Diagram for FREELANCEFINDER

It contains the class attributes, methods as well as the relationships between classes. These mentioned contents make sure that your freelance finder system development must inline with what should be its functions.

5.4.2Sequence Diagram:

Freelace finding System Sequence Diagram - shows the sequence of events that should be present in the Freelanceing Management.

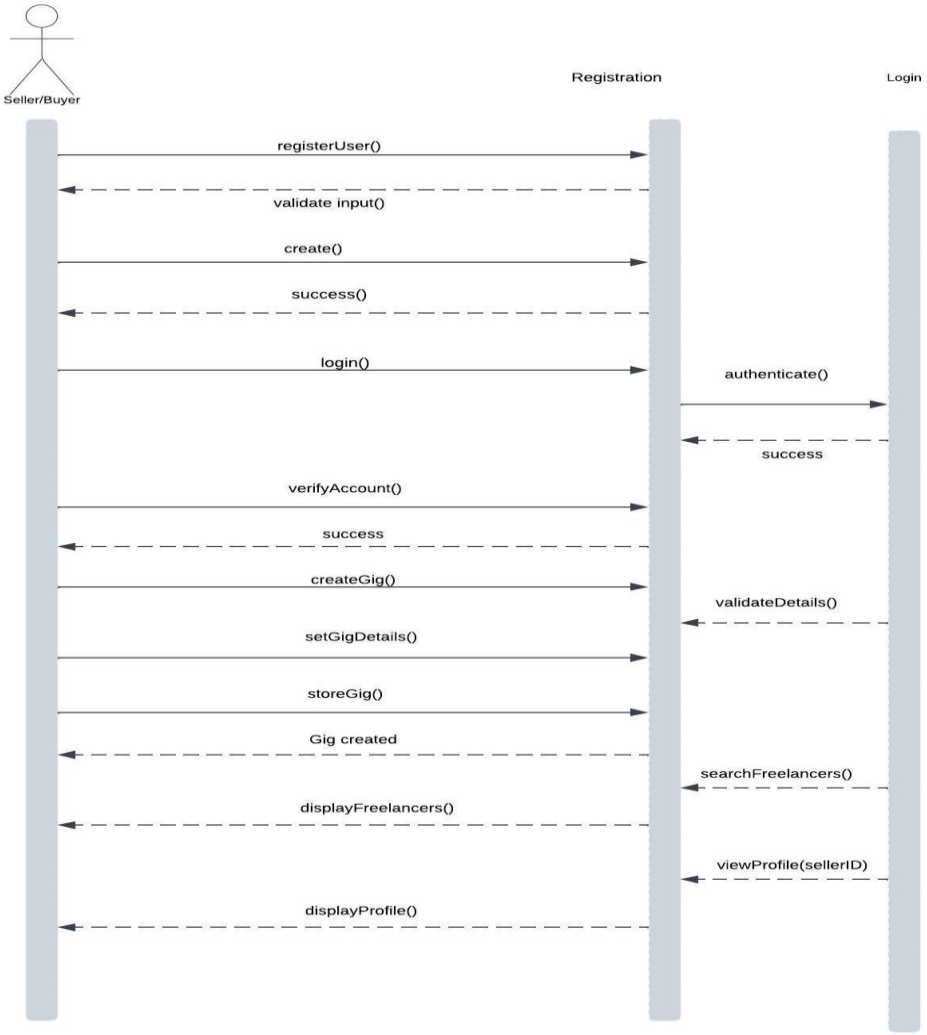


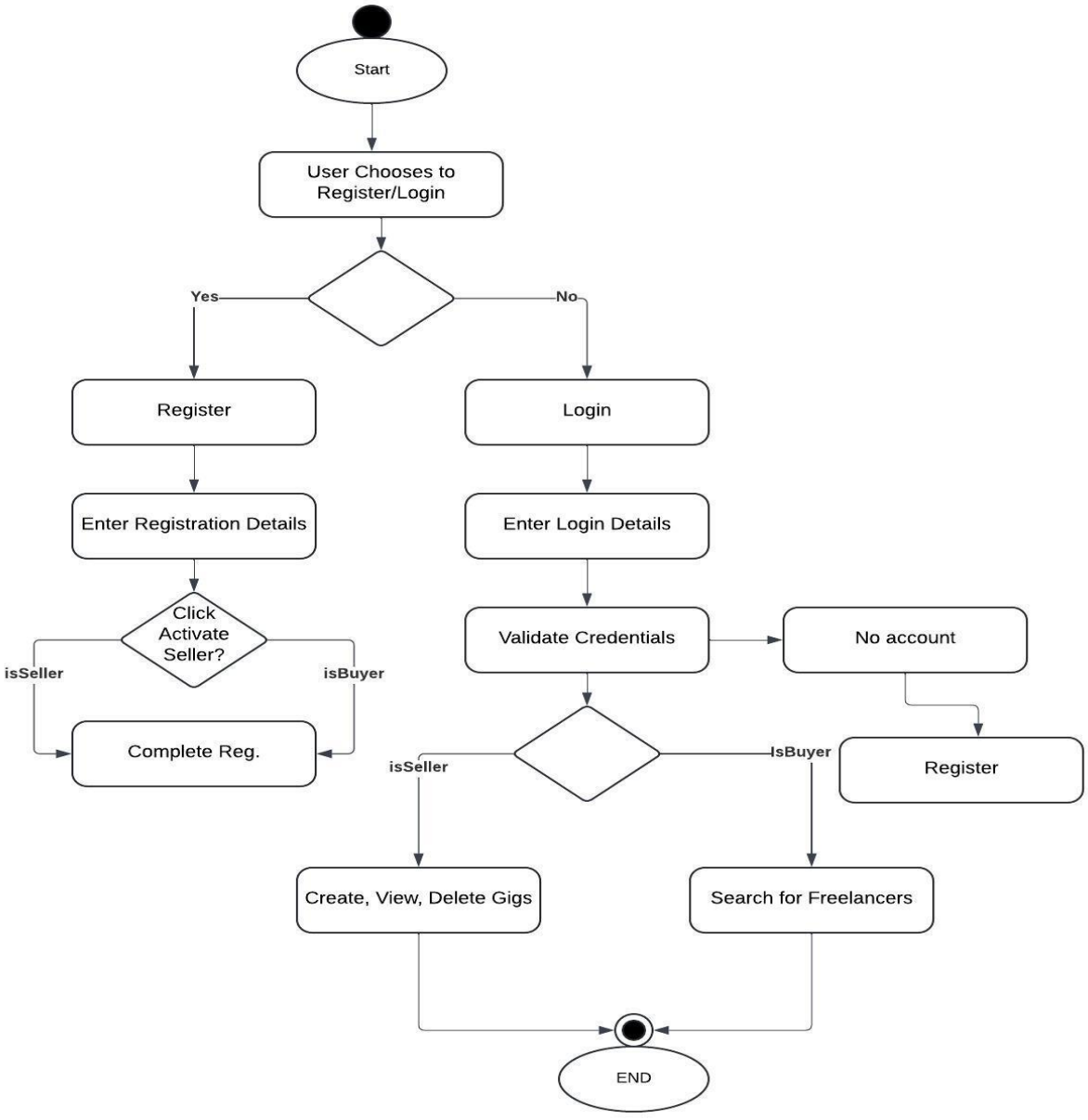
FIG 5.4.2 Sequence Diagram For FREELANCEFINDER

This diagram gives enlightenment and guide to the programmers and developers on how should they build the system. The idea presented in a sequence diagram will give efficiency to Freelance Finder system development.

5.4.3 Activity Diagram:

Freelance Finder System Activity Diagram - is one of the UML diagrams built to give the proponents the right ideas on how to develop the said software.

FIG 5.4.3 Activity Diagram for FREELANCEFINDER



5.4.4 Use Case Diagram:

Freelance Finder System Use Case Diagram - is a visual representation of how a user might interact with a program. A use case diagram depicts the system's numerous use cases and differentsorts of users. The circles or ellipses are used to depict the use cases.

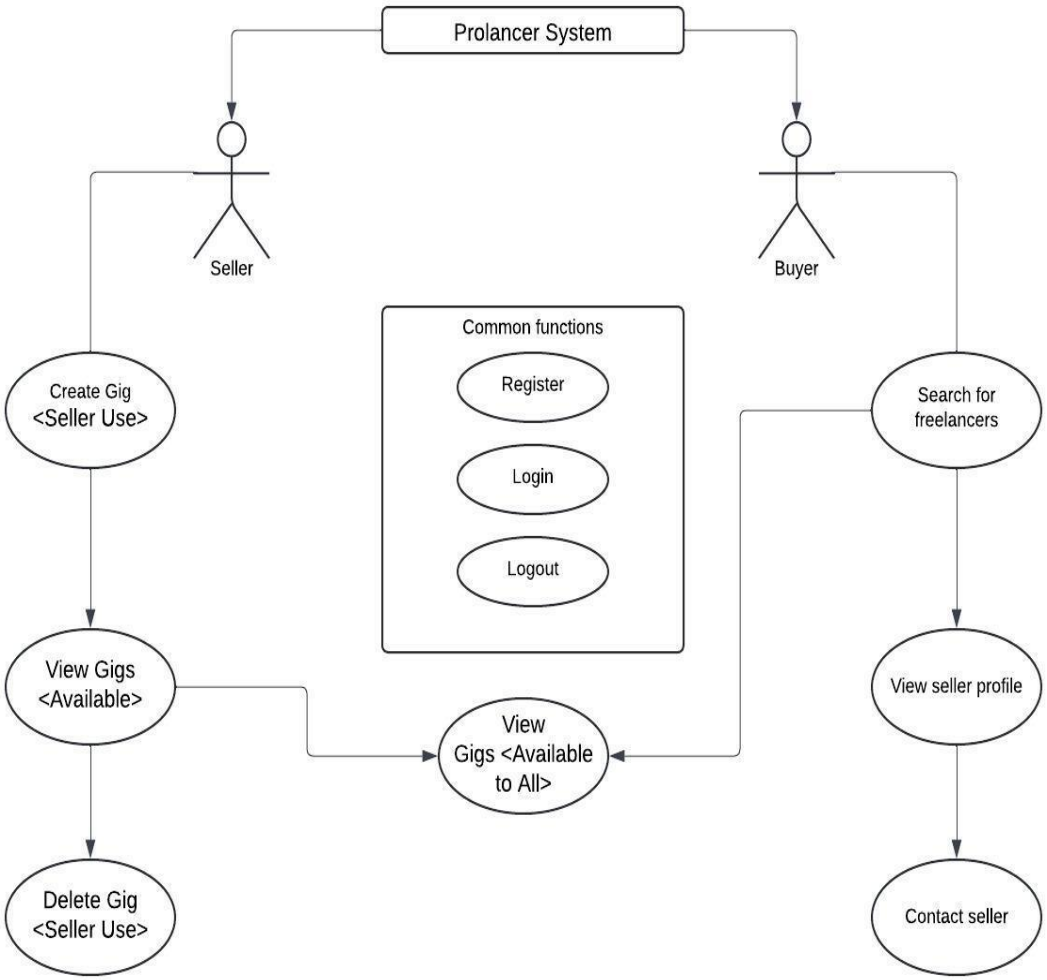


FIG 5.4.4 Use Case Diagram for FREELANCEFINDER

5.5 Database Design

Freelance Finder System Database Design – The Freelance Finder system database design is created to manage functions and data related to freelance services. It enables the admin to manage user accounts, while freelancers (sellers) can create and update their gig offerings, and clients (buyers) can view and search for services. The database securely stores user information, gig details, and communication records. Admins have access to add, view, edit, update, or delete data as needed to maintain the platform's integrity and support smooth operations

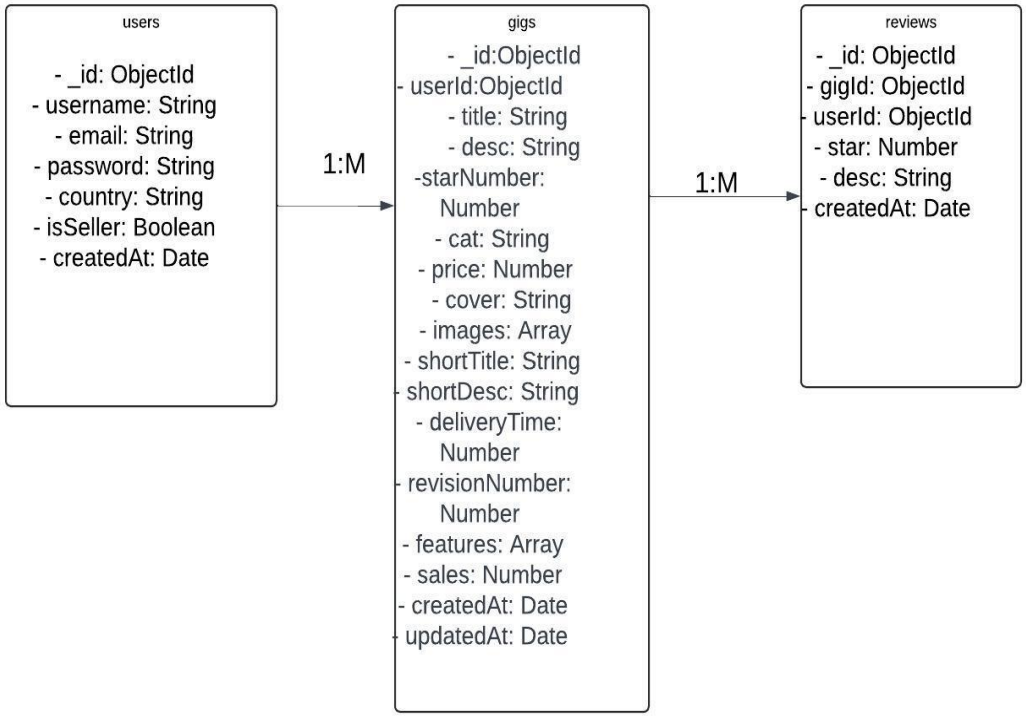


FIG 5.5 FREELANCE FINDER Database Design

The purpose of the Freelance Finder system database design is to provide a secure and efficient way of storing information about users, gigs, communication records, and client-freelancer interactions. It supports smooth data management, ensuring that freelancer profiles, service offerings, and client inquiries are accessible, organized, and securely stored.

5.6 E-R DIAGRAM:

We have ER diagram that has 3 different entities and each entity have its own-own attributes and relationship between them

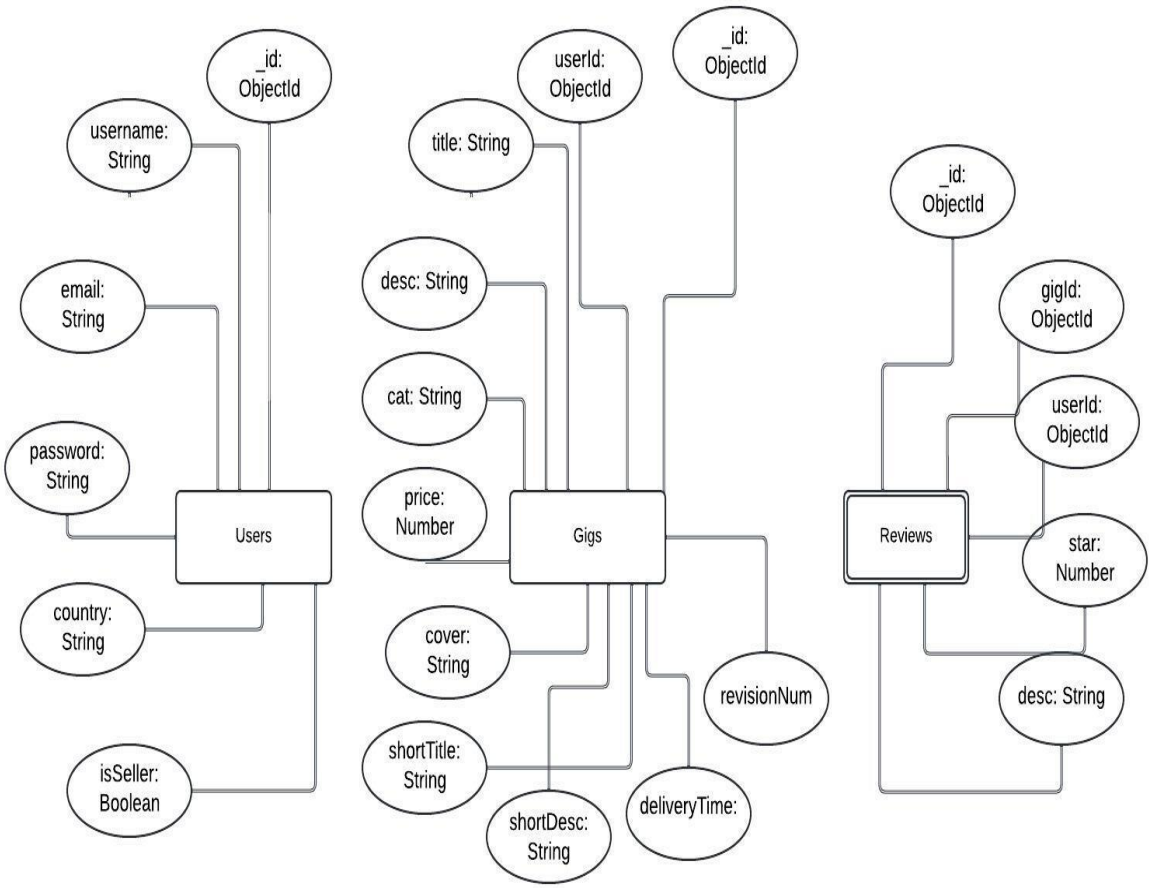


FIG 5.6 E-R DIAGRAM for FREELANCEFINDER

5.7 Work Flow Diagram

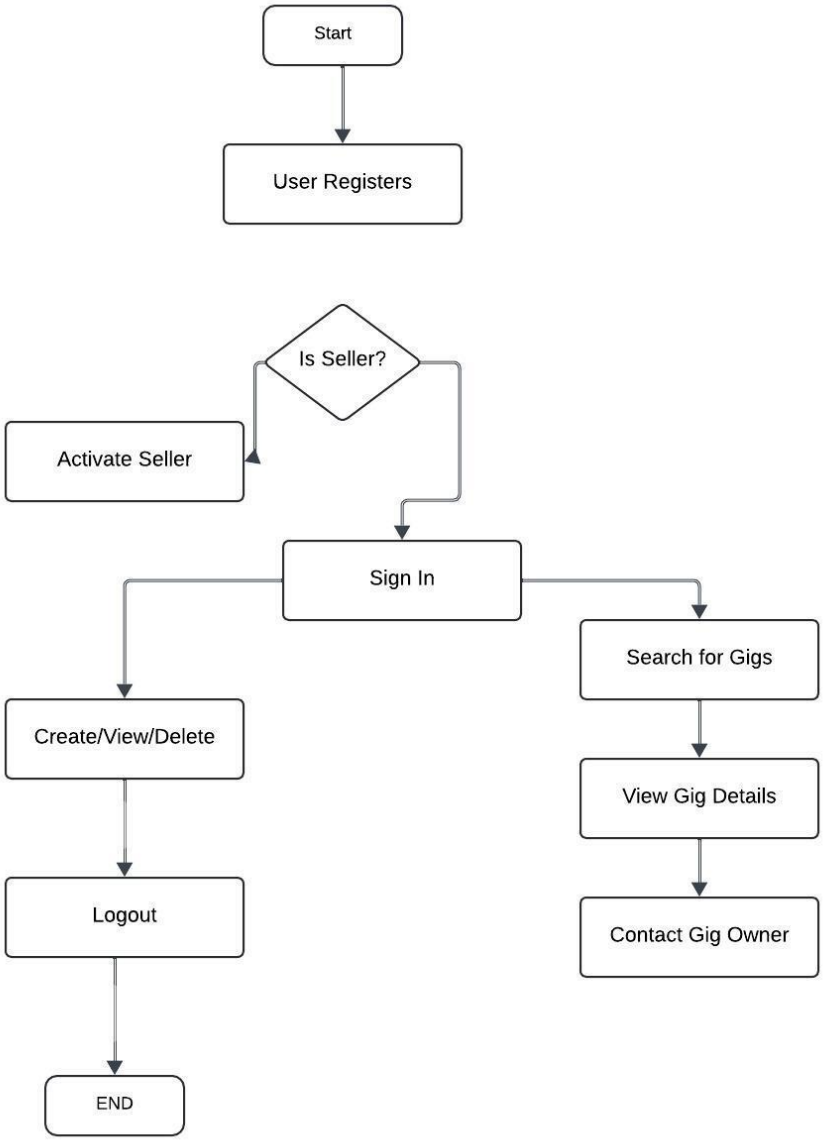


FIG 5.7 Workflow Diagram for FREELANCEFINDER

6.1IMPLEMENTATION:

The FreelanceFinder system has a Home page where clients (buyers) and freelancers (sellers) can log into their accounts by selecting the appropriate tabs.

The Project consists of 2 modules:

- Client Module
- Freelancer Module

Client Module:

- This module allows clients (buyers) to create an account on FreelanceFinder by clicking the 'Register' button. After registration, they are redirected to their personalized Dashboard.
- **Dashboard**: Once logged in, clients can perform key operations:
    1. **Search and View Gigs** - Clients can browse available gigs posted by freelancers, view details of services offered, and check images, revisions, and other relevant information.
    2. **Contact Freelancer -** By clicking on 'Continue' within a gig, clients can initiate contact with the freelancer to discuss project details and requirements.

- **Login Access**: Once a client has created an account, they can log in to their account anytime to access their Dashboard and explore services without the need to register again.
- Overall, this module enables clients to register or log in, explore freelancer profiles and services, and initiate communication with freelancers through a seamless interface.

Freelancer Module:

- **Login and Access**: Freelancers can log into their account by selecting the ‘Freelancer’ tab on the login page. Only registered freelancers can access this section, as new freelancer registrations must be set up initially through the client registration process with the ‘Activate Seller’ option.
- **Dashboard Access**: Upon logging in, freelancers are redirected to their dashboard, where they can manage their gigs and view recent client inquiries.
- **Gig Management**: Freelancers can create new gigs by detailing their skills, services, pricing, and revision offerings. They can also edit or delete existing gigs as needed to keep their offerings up-to-date.
- **Contact Management**: Freelancers can manage all client contacts received through their gigs. This helps them organize inquiries and follow up efficiently to build relationships with potential clients.
- **Logout and Security**: Once done, freelancers can securely log out of their account. The system ensures secure authentication and access, preventing unauthorized use.

In summary, the Freelancer Module allows freelancers to log in, manage their gigs, view and respond to client inquiries, and update their profile information, offering a streamlined interface for effective client interaction and profile management.

6.2 Coding

6.2.1 Code for HOME PAGE

```
import React from "react";
import "./Home.scss";
import Featured from "../../components/featured/Featured";
import TrustedBy from "../../components/trustedBy/TrustedBy";
import Slide from "../../components/slide/Slide";
import CatCard from "../../components/catCard/CatCard";
import ProjectCard from "../../components/projectCard/ProjectCard";
import { cards, projects } from "../../data";

function Home() {
  return (
    <div className="home">
      <Featured />
      <TrustedBy />
      <Slide slidesToShow={5} arrowsScroll={5}>
        {cards.map((card) => (
          <CatCard key={card.id} card={card} />
        ))}
      </Slide>
      <div className="features">
        <div className="container">
          <div className="item">
            <h1>Unlock a global pool of expert freelancers ready to bring your ideas to life</h1>
            <div className="title">
              <img src="./img/check.png" alt="" />
              The best for every budget
            </div>
            <p>
            Access top-tier talent at any price range—no hourly fees, just clear project-based rates.
            </p>
            <div className="title">
              <img src="./img/check.png" alt="" />
              Quality work done quickly
            </div>
            <p>
            Get started fast—find the perfect freelancer and begin your project in minutes.
            </p>
            <div className="title">
              <img src="./img/check.png" alt="" />
              Protected payments, every time
            </div>
            <p>
            Pay only when you're satisfied. Transparent pricing with secure, milestone-based payments.
            </p>
            <div className="title">
              <img src="./img/check.png" alt="" />
```

```
            24/7 support
          </div>
          <p>
          Around-the-clock assistance whenever you need it, ensuring a smooth experience from start to
finish.
          </p>
        </div>
        <div className="item">
          <video src="./img/video.mp4" controls />
        </div>
      </div>
    </div>
    <div className="explore">
      <div className="container">
        <h1>Explore the marketplace</h1>
        <div className="items">
          <div className="item">
            <img
              src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/graphics-design.d32a2f8.svg"
              alt=""
            />
            <div className="line"></div>
            <span>Graphics & Design</span>
          </div>
          <div className="item">
            <img
              src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/online-marketing.74e221b.svg"
              alt=""
            />
            <div className="line"></div>

            <span>Digital Marketing</span>
          </div>
          <div className="item">
            <img
              src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/writing-translation.32ebe2e.svg"
              alt=""
            />
            <div className="line"></div>
            <span>Writing & Translation</span>
          </div>
          <div className="item">
            <img
              src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/video-animation.f0d9d71.svg"
              alt=""
            />
```

```
            <div className="line"></div>
            <span>Video & Animation</span>
          </div>
          <div className="item">
            <img
              src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/music-audio.320af20.svg"
              alt=""
            />
            <div className="line"></div>
            <span>Music & Audio</span>
          </div>
          <div className="item">
            <img
              src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/programming.9362366.svg"
              alt=""
            />
            <div className="line"></div>
            <span>Programming & Tech</span>
          </div>
          <div className="item">
            <img
              src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/business.bbdf319.svg"
              alt=""
            />
            <div className="line"></div>
            <span>Business</span>
          </div>
          <div className="item">
            <img
              src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/lifestyle.745b575.svg"
              alt=""
            />
            <div className="line"></div>
            <span>Lifestyle</span>
          </div>
          <div className="item">
            <img
              src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/data.718910f.svg"
              alt=""
            />
            <div className="line"></div>
            <span>Data</span>
          </div>
          <div className="item">
            <img
```

```
            src="https://fiverr-res.cloudinary.com/npm-
assets/@fiverr/logged_out_homepage_perseus/apps/photography.01cf943.svg"
            alt=""
           />
           <div className="line"></div>
           <span>Photography</span>
          </div>
         </div>
        </div>
       </div>
       <div className="features dark">
        <div className="container">
         <div className="item">
          <h1>
           ProLancer <i>business</i>
          </h1>
          <h1>
           A business solution designed for <i>teams</i>
          </h1>
          <p>
           Upgrade to a curated experience packed with tools and benefits,
           dedicated to businesses
          </p>
          <div className="title">
           <img src="./img/check.png" alt="" />
           Connect to freelancers with proven business experience
          </div>

          <div className="title">
           <img src="./img/check.png" alt="" />
           Get matched with the perfect talent by a customer success manager
          </div>

          <div className="title">
           <img src="./img/check.png" alt="" />
           Manage teamwork and boost productivity with one powerful workspace
          </div>
          <button>Explore ProLancer Business</button>
         </div>
         <div className="item">
          <img
           src="https://fiverr-
res.cloudinary.com/q_auto,f_auto,w_870,dpr_2.0/v1/attachments/generic_asset/asset/d9c17ceebda44764
b591a8074a898e63-1599597624768/business-desktop-870-x2.png"
           alt=""
          />
         </div>
        </div>
       </div>
       <Slide slidesToShow={4} arrowsScroll={4}>
```

```
        {projects.map((card) => (
          <ProjectCard key={card.id} card={card} />
        ))}
      </Slide>
    </div>
  );
}

export default Home;
```

6.2.2 Login page CODE

```
import React, { useState } from "react";
import "./Login.scss";
import newRequest from "../../utils/newRequest";
import { useNavigate, Link } from "react-router-dom";

function Login() {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [captchaInput, setCaptchaInput] = useState("");
  const [generatedCaptcha, setGeneratedCaptcha] = useState(generateCaptcha());
  const [showPassword, setShowPassword] = useState(false); // To toggle password visibility
  const [error, setError] = useState(null);

  const navigate = useNavigate();

  function generateCaptcha() {
    const characters =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    let result = "";
    for (let i = 0; i < 6; i++) {
      result += characters.charAt(Math.floor(Math.random() * characters.length));
    }
    return result;
  }
```

```
const handleSubmit = async (e) => {
  e.preventDefault();

  if (captchaInput !== generatedCaptcha) {
    setError("Invalid CAPTCHA. Please try again.");
    setGeneratedCaptcha(generateCaptcha());
    return;
  }

  try {
    const res = await newRequest.post("/auth/login", { username, password });
    localStorage.setItem("currentUser", JSON.stringify(res.data));
    navigate("/");
  } catch (err) {
    setError(err.response.data);
  }
};

const togglePasswordVisibility = () => {
  setShowPassword(!showPassword);
};

return (
  <div className="login">
    <form onSubmit={handleSubmit}>
      <h1>Sign in</h1>
```

```
<label htmlFor="username">Username</label>
<input
  name="username"
  type="text"
  placeholder="johnwick"
  onChange={(e) => setUsername(e.target.value)}
  required
/>

<label htmlFor="password">Password</label>
<div className="password-container">
  <input
    name="password"
    type={showPassword ? "text" : "password"}
    onChange={(e) => setPassword(e.target.value)}
    required
  />
  <span className="password-toggle" onClick={togglePasswordVisibility}>
    {showPassword ? "🙈" : "👁️"}
  </span>
</div>

<label htmlFor="captcha">Enter CAPTCHA: {generatedCaptcha}</label>
<input
  name="captcha"
  type="text"
```

```
          placeholder="Enter the CAPTCHA above"
          onChange={(e) => setCaptchaInput(e.target.value)}
          required
        />

        <button type="submit">Login</button>
        {error && <div className="error-message">{error}</div>}

        <div className="register-link">
          <p>Don't have an account? <Link to="/register">Join Us</Link></p>
        </div>
      </form>
    </div>
  );
}

export default Login;
```

6.2.3 Authentication Routes CODE

```
  import express from "express";
import { register, login, logout } from "../controllers/auth.controller.js";

const router = express.Router();
```

```
router.post("/register", register)
router.post("/login", login)
router.post("/logout", logout)

export default router;
```

6.2.4 Middleware Code

```
import jwt from "jsonwebtoken";
import createError from "../utils/createError.js";

export const verifyToken = (req, res, next) => {
  const token = req.cookies.accessToken;
  if (!token) return next(createError(401,"You are not authenticated!"))

  jwt.verify(token, process.env.JWT_KEY, async (err, payload) => {
    if (err) return next(createError(403,"Token is not valid!"))
    req.userId = payload.id;
    req.isSeller = payload.isSeller;
    next()
  });
};
```

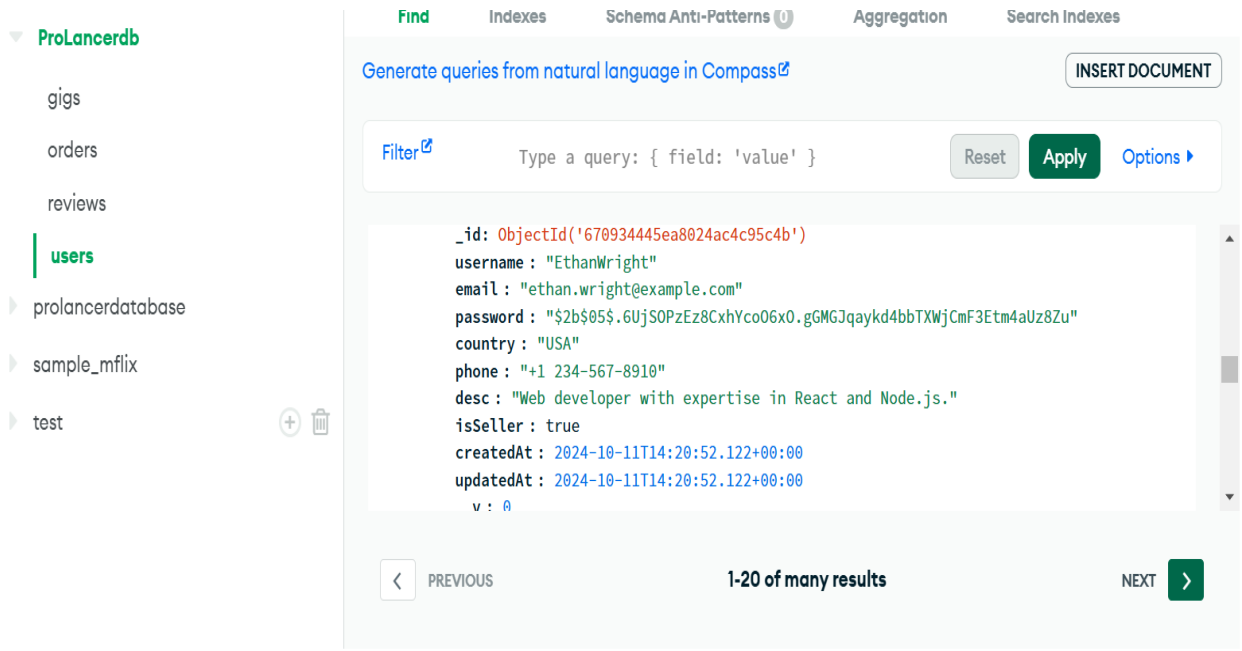
6.5 DATABASE COLLECTIONS

In MongoDB, data is stored in collections, which are the equivalent of tables in a relational database. Collections hold groups of documents, each representing an entry within the collection, and the data is organized in a flexible, schema-less format.

Each document is stored in **JSON-like** format, consisting of key-value pairs that allow for easy representation of complex data structures. This flexibility enables the **FreelanceFinder** system to efficiently store and retrieve information, such as user profiles, gigs, client inquiries, and user feedback, without the strict structure required by relational databases

6.5.1 Users Data Collection

FIG 6.5.1 Users data



6.5.2 Gigs Collections

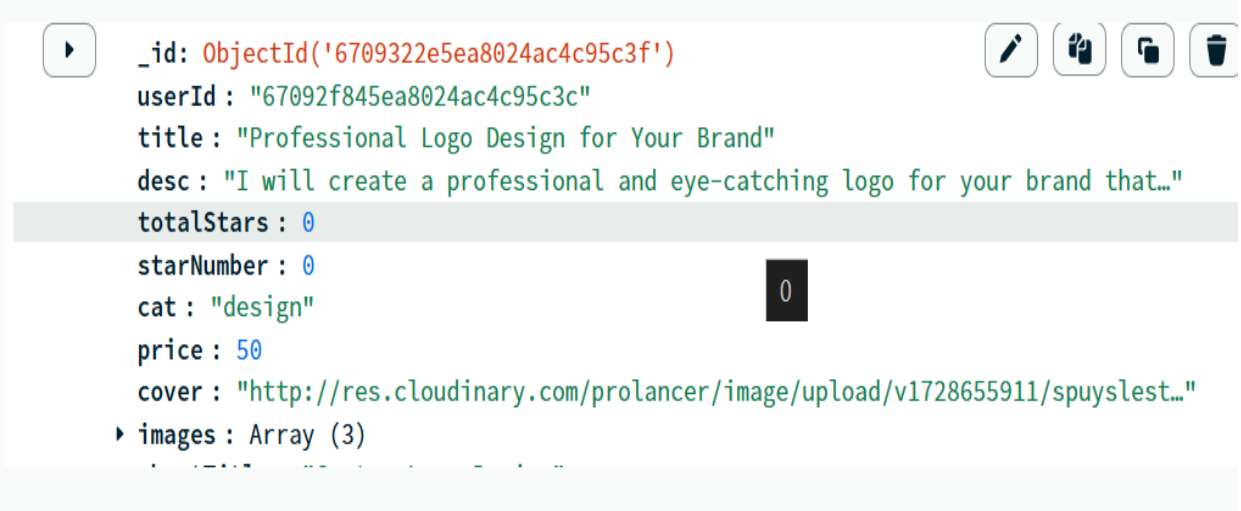


FIG 6.5.2 Gigs Data

6.5.3 Reviews Collections

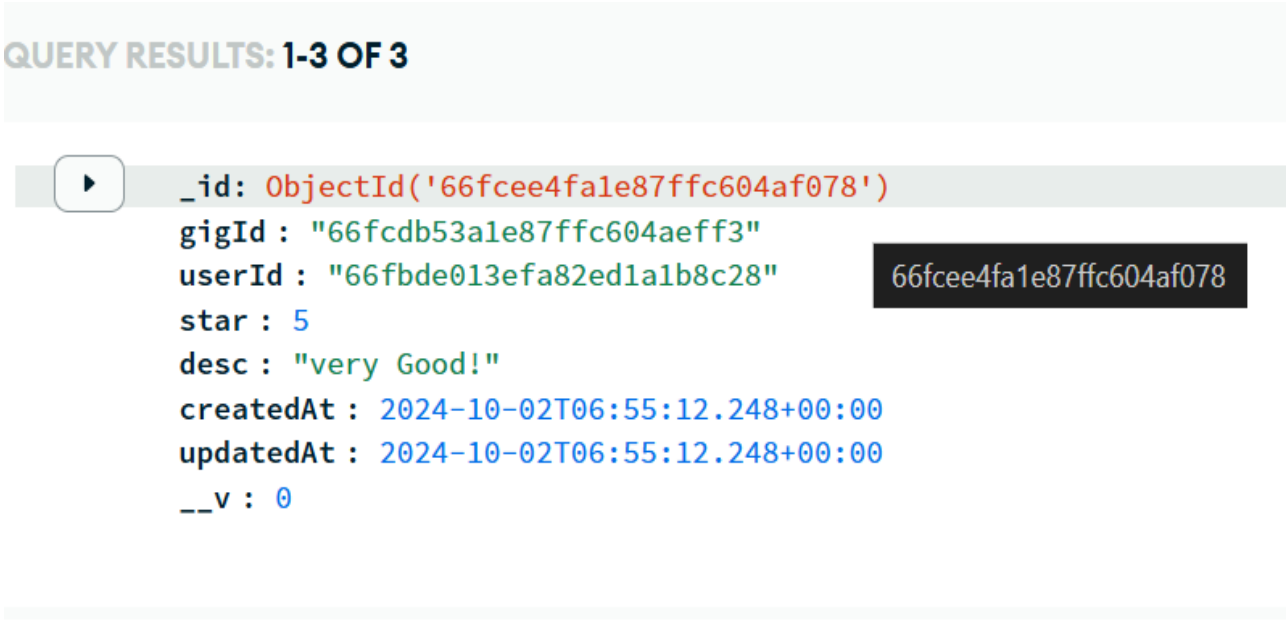


FIG 6.5.3 Reviews Data

6.5.4 MONGODB Connection Code:

```
const connect = async () => {
  try {
    await mongoose.connect(process.env.MONGO);
    console.log("Connected to mongoDB!");
  } catch (error) {
    console.log(error);
  }
};
```

Mongoose: Mongoose is a popular ODM library that provides a straightforward way to model application data in MongoDB. It includes features like schema validation, query building, and middleware support, making it easier to interact with MongoDB.

Async/Await: The use of async and await makes it easier to write asynchronous code in a synchronous manner, improving code readability and maintainability.

7.1INTRODUCTION TO SYSTEM TESTING:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.2 TYPES OF TESTING:

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. Test strategy and approach Field testing will be performed manually and functional tests will be written in detail.

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Features to be tested

Verify that the entries are of the correct format.

No duplicate entries should be allowed

All links should take the user to the correct page.

7.2.1 Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up software applications at the company level interact without error.

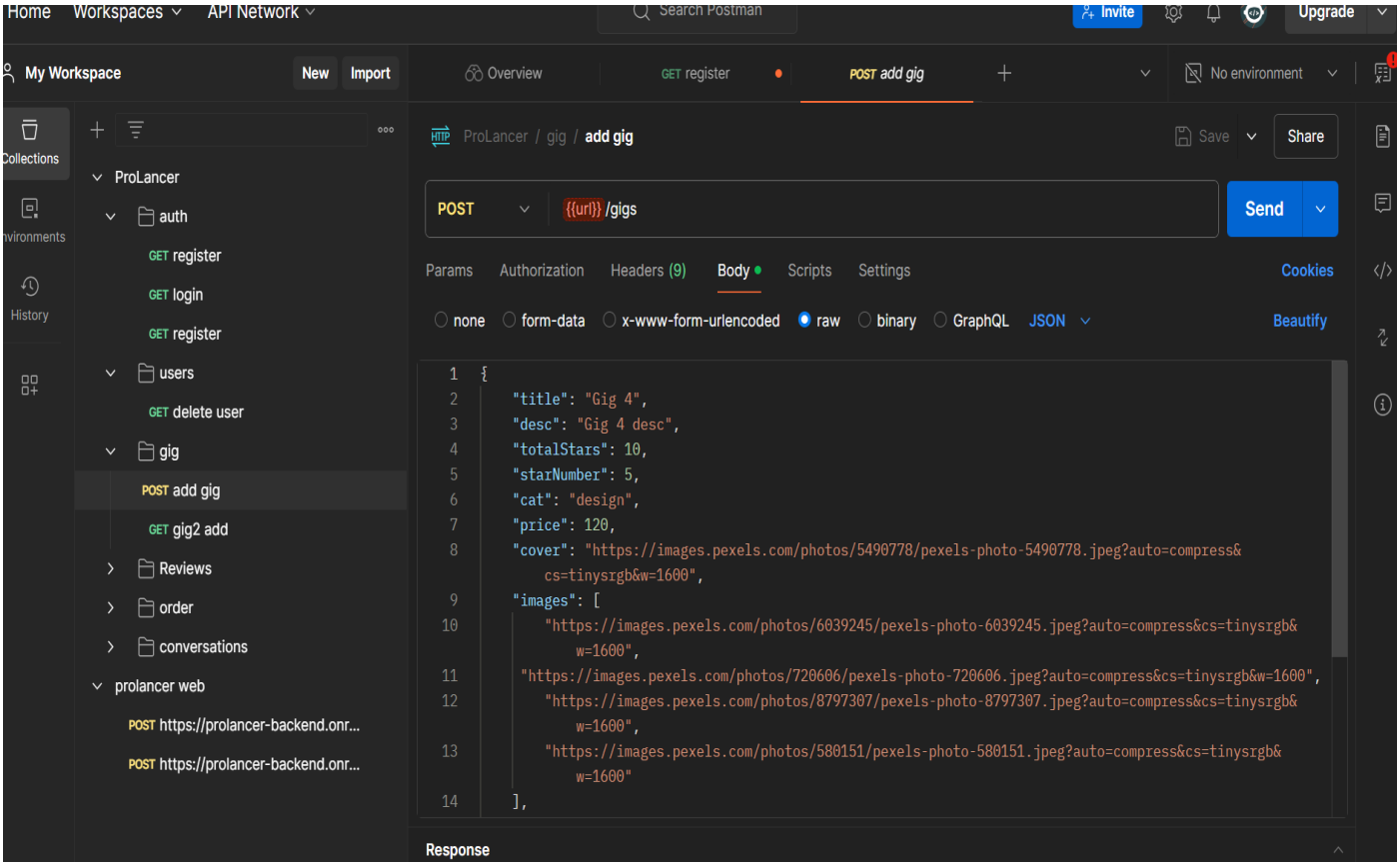
7.2.2 System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.2.3API Test Tool:

Postman for API Testing: Postman is a popular API testing tool that allows developers to send requests to web servers and inspect the responses. It provides an intuitive interface for creating, testing, and documenting APIs, making it a valuable tool for ensuring that your backend services function as intended.

FIG 7.2.4 API Testing



8.1 HOME PAGE

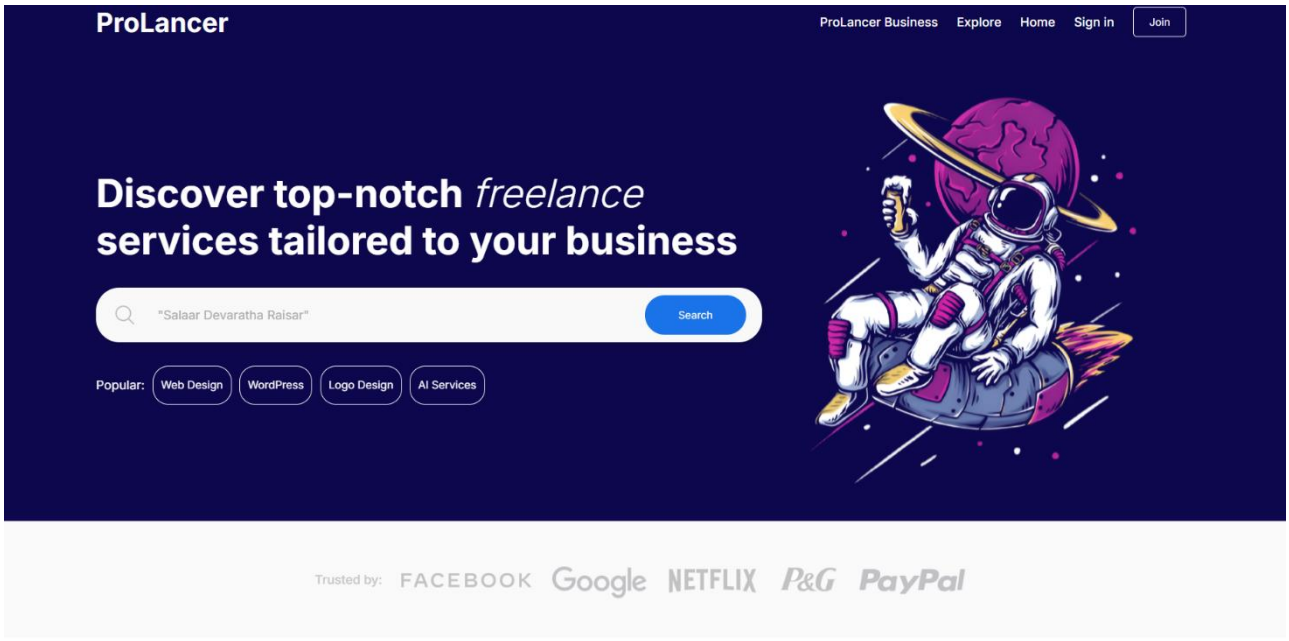
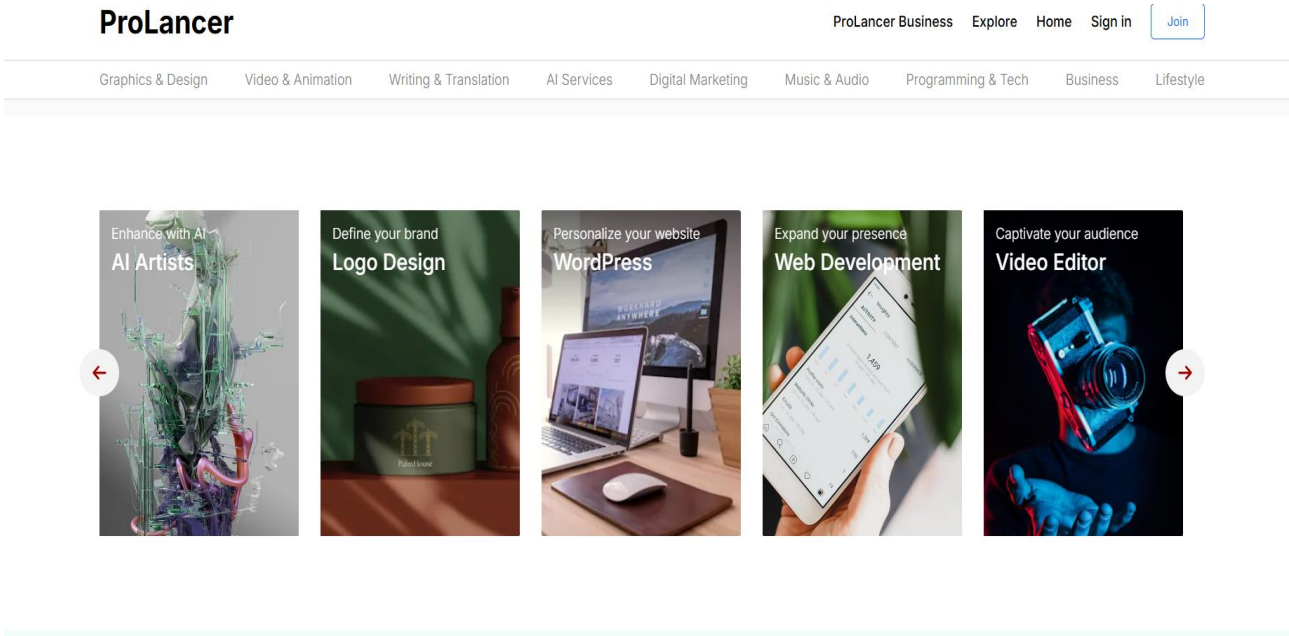


FIG 8.1 Home Page

8.2 GIG CARDS

FIG 8.2 Gig Cards



8.3 REGISTRATION

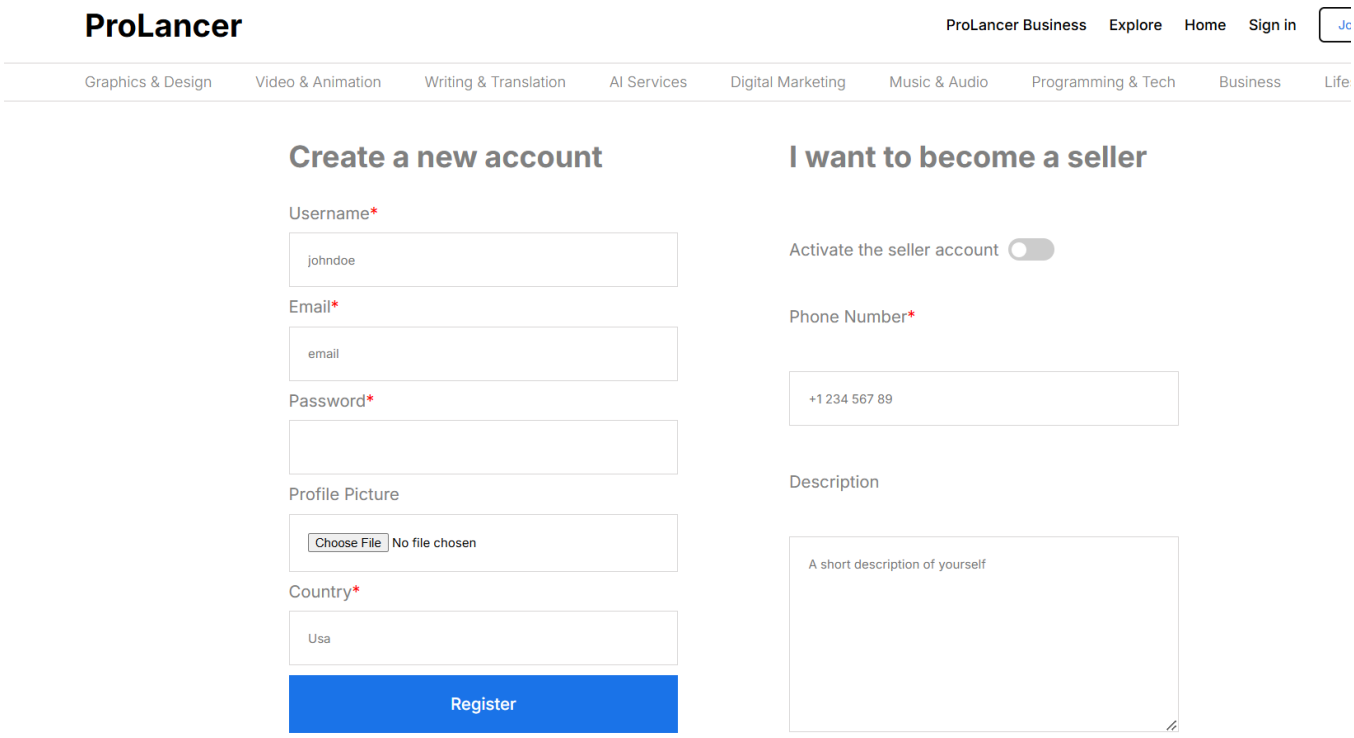


FIG 8.3 Registration

8.4LOGIN

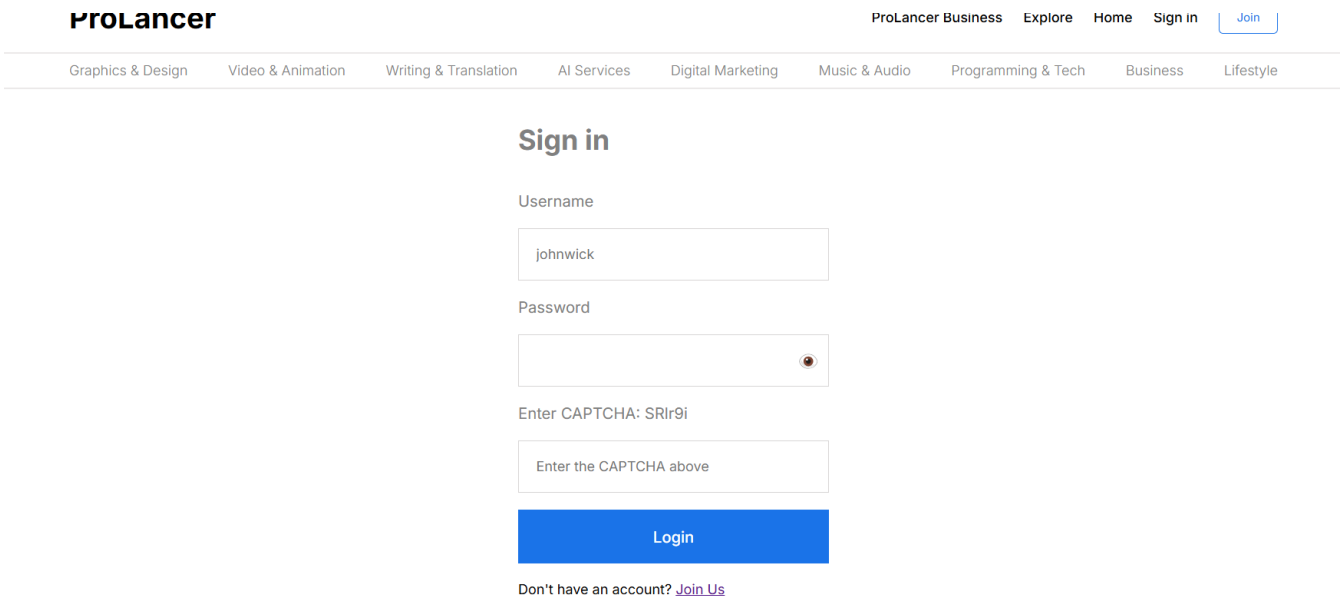


FIG 8.4 Login Page

8.5GIGS

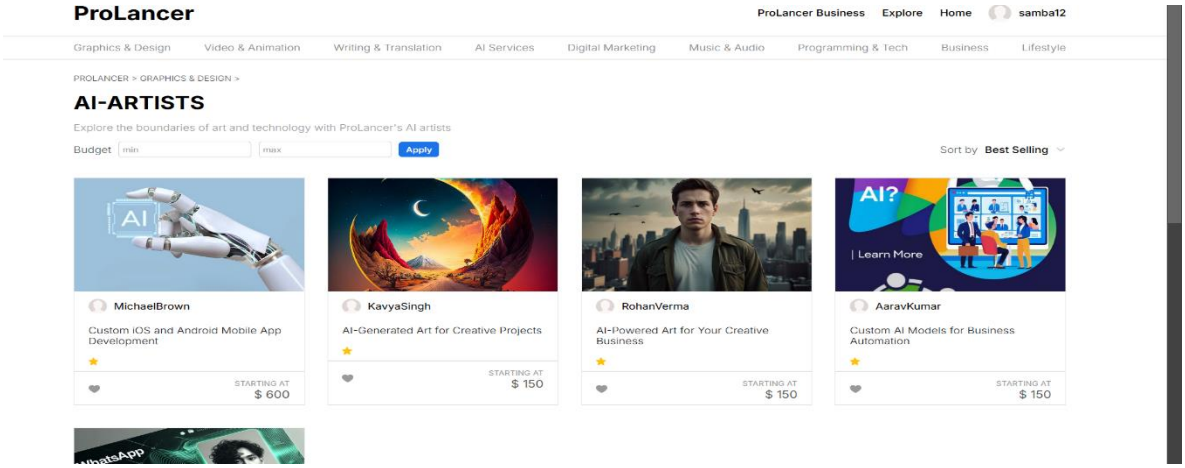


FIG 8.5.1 Gigs

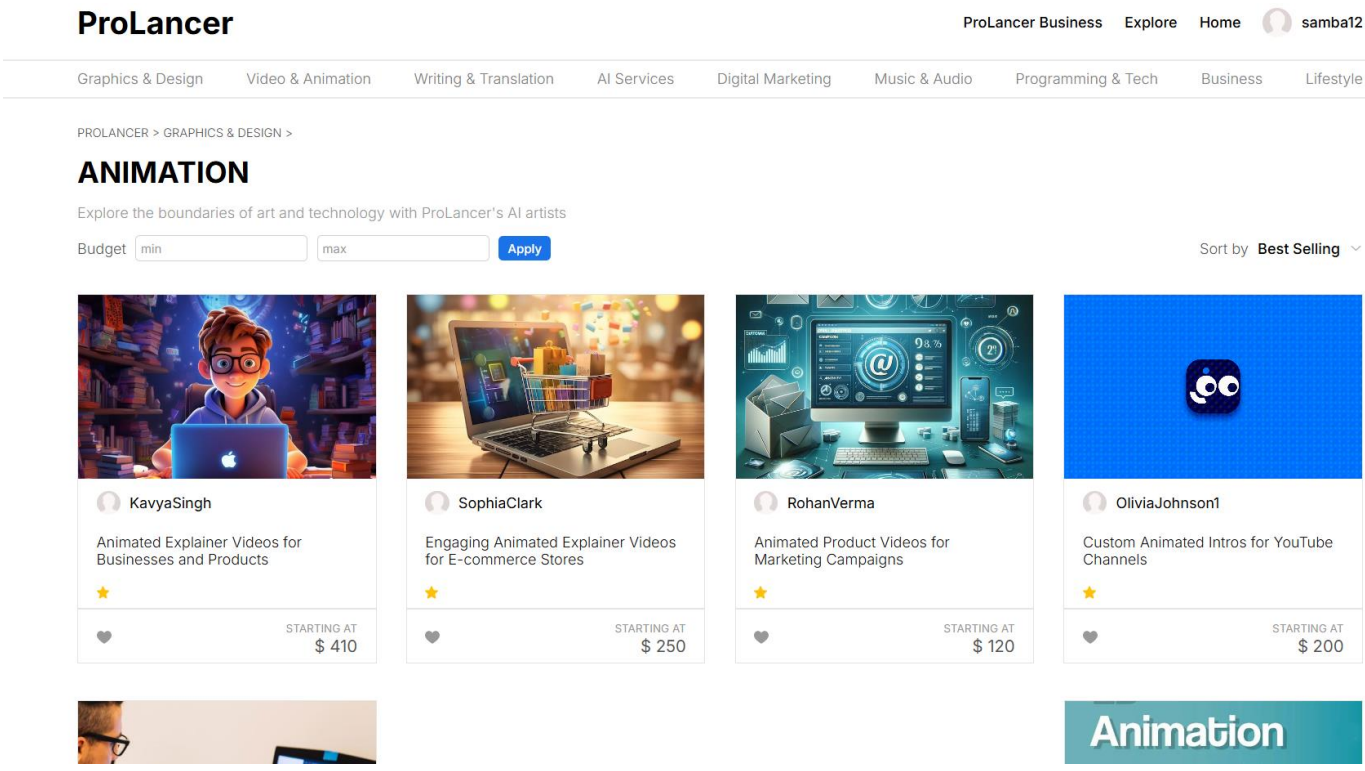


FIG 8.5.2 Gigs

8.6 GIG Page

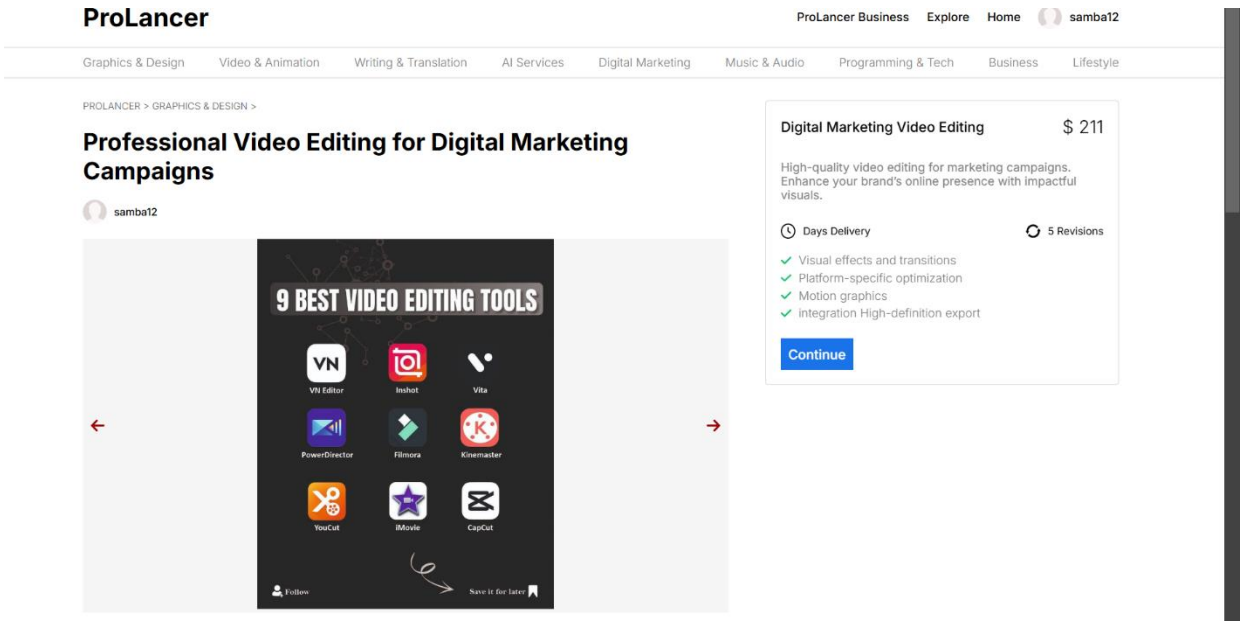


FIG 8.6.1 Gig Page

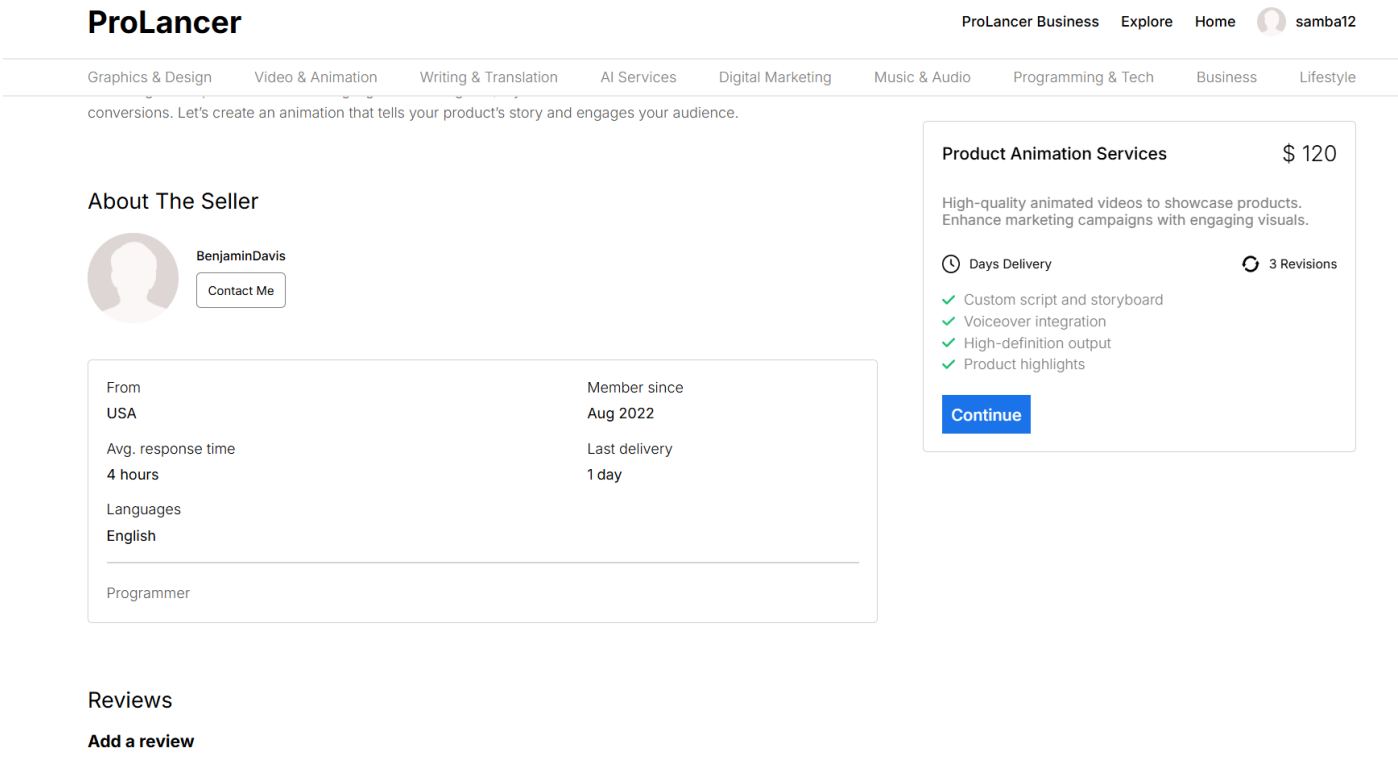


FIG 6.6.2 Gig Page

8.7 Features

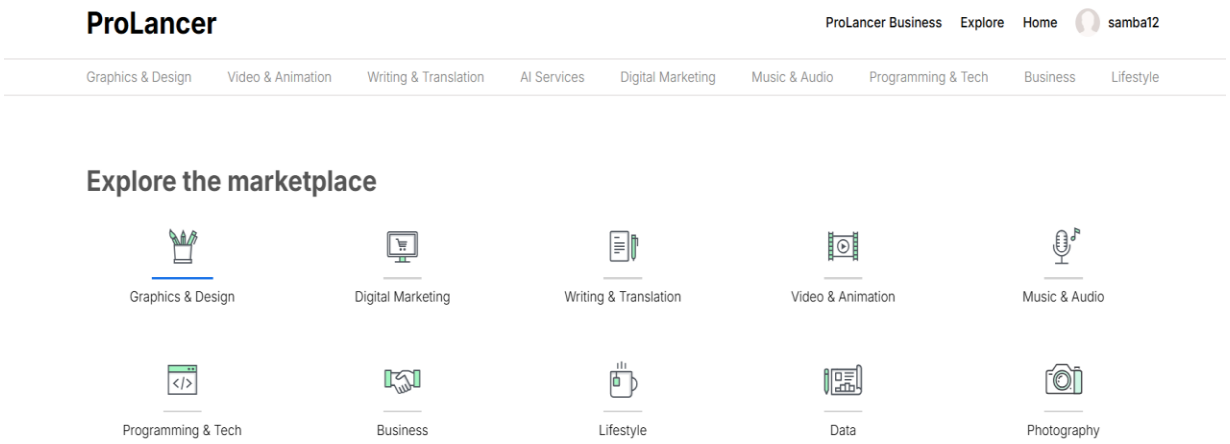


FIG 8.7 Features

8.8 FOOTER

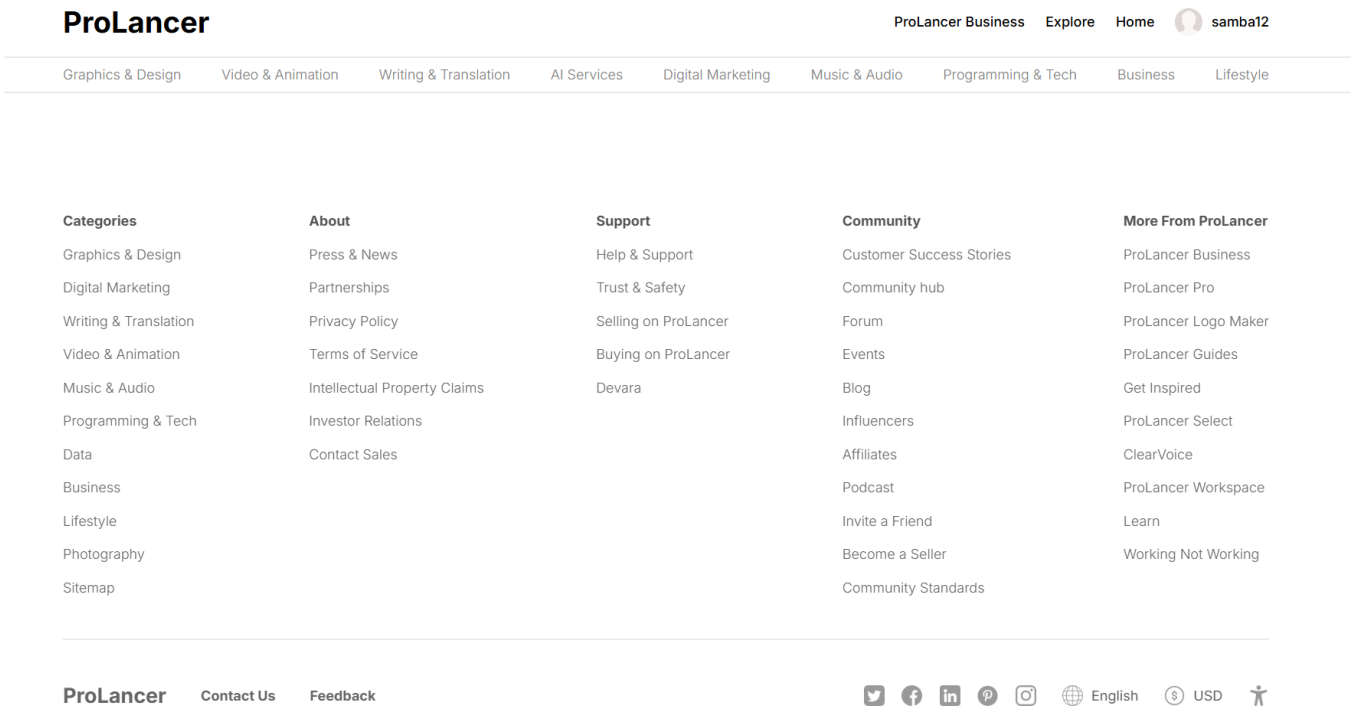


FIG 8.8 Footer

8.9 EXPLORE

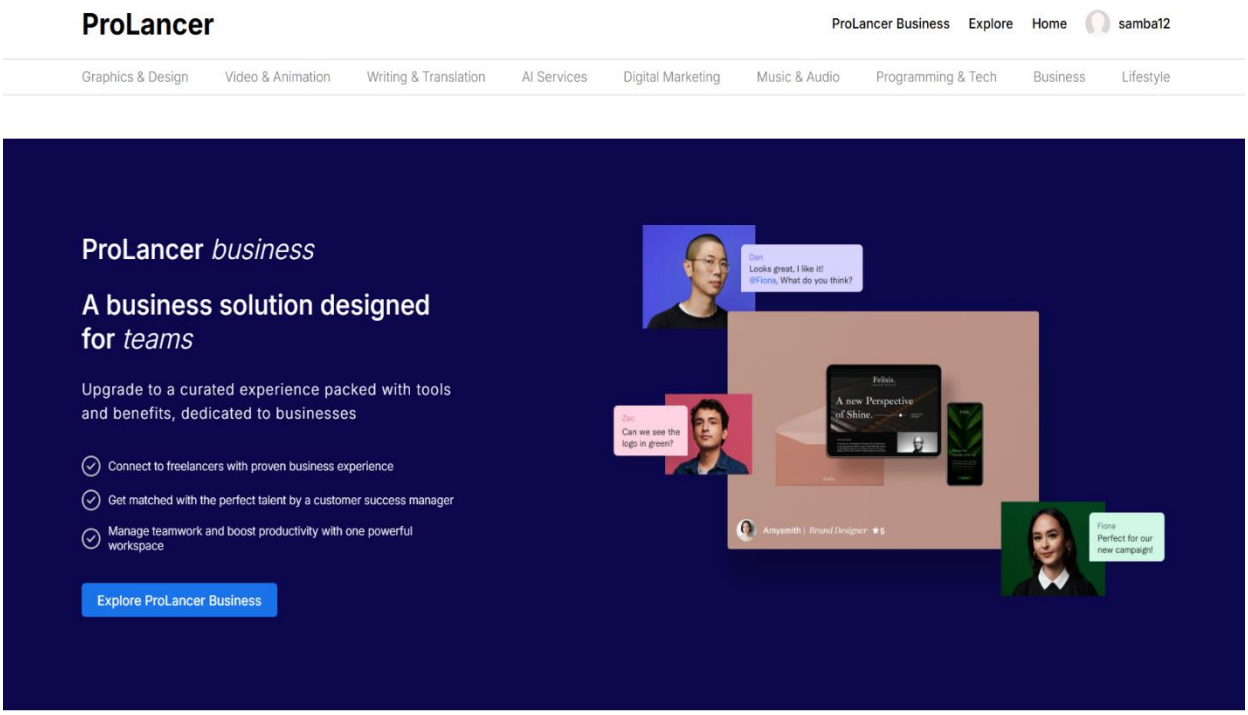


FIG 8.9 Explore

Week 1: Project Planning

- **Define Project Scope:**

Outline the key features and functionality of the Freelance Finder platform.

Identify target users (clients and freelancers) and their needs.

- **Research Existing Platforms:**

Analyze competitor platforms such as Upwork and Fiverr to understand their strengths and weaknesses.

Document findings on user experience, features, and business models.

- **Requirements Gathering:**

Conduct surveys or interviews with potential users to gather insights on desired features.

Compile a list of functional and non-functional requirements for the system.

Week 2: Design Phase

- **Wireframing:**

Create wireframes for the Client Module and Freelancer Module, focusing on user navigation and layout.

Use tools like Figma or Sketch for visual representation.

- **Mockups:**

Develop high-fidelity mockups based on wireframes to showcase the UI design.

Ensure design consistency across different pages and modules.

- **Feedback Gathering:**

Present mockups to stakeholders (e.g., peers, mentors) and gather feedback for improvements.

Iterate on designs based on feedback received.

Week 3: Development Environment Setup

- **Initialize Repository:**

Create a GitHub repository for version control and collaboration.

Set up the initial project structure for frontend and backend.

- **Select Tech Stack:**

Finalize the MERN stack (MongoDB, Express.js, React, Node.js) for development.

Install necessary dependencies and configure development tools (e.g., ESLint, Prettier).

- **Environment Setup:**

Set up local development environments for team members.

Ensure all team members have access to necessary resources and documentation.

Week 4: Develop Client Module

- **User Registration:**

Implement the registration functionality allowing clients to create accounts with personal information.

Ensure data validation and secure password storage.

- **Appointment Booking:**

Develop the feature for clients to view available freelancers and book appointments.

Implement alerts for successful bookings.

- **Appointment History:**

Create a feature for clients to view their past appointments, including details such as date, time, and freelancer name.

Week 5: Develop Freelancer Module

- **Login Functionality:**

Implement secure login for freelancers with password recovery options.

Ensure the registration of freelancers is handled by the admin only.

- **Appointment Management:**

Develop features for freelancers to view their scheduled appointments.

Allow freelancers to accept or decline appointments based on availability.

- **Patient Search:**

Implement a search functionality for freelancers to find and view client profiles.

Week 6: Backend Development

- **Database Setup:**

Design the database schema in MongoDB to store user data, appointments, and feedback.

Implement models for Client, Freelancer, and Appointment.

- **API Development:**

Create RESTful API endpoints for user registration, login, appointment management, and data retrieval.

Ensure proper error handling and response formatting.

- **Authentication:**

Implement JSON Web Token (JWT) for user authentication and session management.

Week 7: Integration

- **Frontend and Backend Integration:**

Connect the frontend React components to the backend APIs.

Ensure smooth data flow between client-side and server-side functionalities.

- **UI Testing:**

Conduct user interface testing to ensure all features are working as expected.

Test for responsiveness and usability across different devices.

Week 8: Testing Phase

- **Unit Testing:**

Write unit tests for individual components and functions in both frontend and backend.

Use testing libraries like Jest and Mocha for effective unit testing.

- **Integration Testing:**

Test interactions between different modules to ensure compatibility and functionality.

Use Postman for testing API endpoints and validating responses.

- **Bug Identification:**

Log and categorize any bugs or issues found during testing for future resolution.

Week 9: Bug Fixing

- **Addressing Issues:**

Prioritize and fix bugs based on severity and impact on user experience.

Refactor code where necessary to improve performance and maintainability.

- **Final Testing:**

Conduct another round of testing to ensure all issues have been resolved.

Perform regression testing to verify that new changes haven't broken existing functionality.

Week 10: Final Review

- **Feedback Gathering:**

Present the completed project to stakeholders and gather feedback.

Document suggestions for future improvements or additional features.

- **Final Adjustments:**

Make any last-minute changes based on feedback received.

Ensure all documentation is up to date, including user manuals and technical specifications.

- **Project Presentation:**

Prepare for the final project presentation, summarizing the development process, challenges faced, and lessons learned.

Highlight the key features of the Freelance Finder platform and its benefits for users.

CONCLUSION

The Freelance Finder project provides a comprehensive platform for connecting freelancers with clients, streamlining the process of job postings, applications, and communication. By leveraging modern web technologies, we have created an efficient and user-friendly application that meets the needs of both freelancers and clients. As we move forward, the focus will be on refining the system based on user feedback and incorporating the latest technological advancements to enhance functionality.

The system was implemented using React for the frontend, ensuring a dynamic and responsive user experience, while MongoDB was utilized for the backend, providing a robust and flexible database solution. With features like user registration, appointment booking, and easy access to user profiles, the platform simplifies the entire freelancing process.

By implementing this web-based application, both clients and freelancers can easily manage their interactions without the need for cumbersome paperwork. User information is securely stored in the database, allowing for quick access to profiles and previous communications. This streamlined approach minimizes time consumption, making it convenient for users to find opportunities or hire talent efficiently.

The Freelance Finder project ultimately aims to enhance communication between clients and freelancers, reduce manual efforts, and improve overall productivity in the freelancing ecosystem. With continuous improvements and updates, we strive to make the platform even more effective in meeting the evolving needs of its users.

Future Enhancements for Freelance Finder:

Payment Integration:

- Implement a secure payment gateway to facilitate financial transactions between clients and freelancers. This would allow clients to pay for services directly through the platform, enhancing convenience and security.

Advanced Search Filters:

- Introduce more refined search options for clients to filter freelancers based on skills, ratings, price range, and availability. This will help users find the most suitable freelancers for their projects more efficiently.

Review and Rating System:

- Develop a robust review and rating system where clients can leave feedback for freelancers after a project is completed. This will enhance trust and provide valuable insights for future clients.

Chat and Messaging Feature:

- Implement a real-time messaging system for clients and freelancers to communicate directly within the platform. This will streamline communication, making it easier to discuss project details and requirements.

Mobile Application:

- Create a mobile app for both Android and iOS platforms, allowing users to access the Freelance Finder on the go. This will increase user engagement and accessibility.

REFERENCES

- **"MERN Quick Start Guide" by Eddy Wilson**: This book provides a hands-on approach to building applications with the MERN stack, making it a great resource for beginners and intermediate developers alike.

- **"Fullstack React: The Complete Guide to ReactJS and Friends" by Accomazzo, Murray, and Lerner**: This book covers React and its ecosystem, providing insights into building full-stack applications using React with Node.js and Express.

- **"Learning MongoDB: A Beginner's Guide" by Abhishek Kumar**: This guide introduces MongoDB, focusing on its core features and how to effectively manage data in your applications.

- **YouTube Channel: LamaDev**: This channel offers tutorials on MERN stack development, providing step-by-step guidance on building full-stack applications.

- **"Node.js Design Patterns":** Node.js Official Documentation: The official documentation is a comprehensive resource for understanding Node.js features and APIs. W3Schools - Node.js Tutorial: W3Schools provides a beginner-friendly tutorial covering the basics of Node.js, including its architecture, modules, and web server creation.

- **"React :** React Official Documentation: The official React documentation provides an excellent foundation for learning about components, props, state, and more . W3Schools - React Tutorial: This tutorial introduces React, covering components, props, state management, and more in a simple and easy-to-understand format. freeCodeCamp - Learn React: freeCodeCamp offers a comprehensive curriculum to learn React, including practical exercises and projects.