COMS3009 Project Report

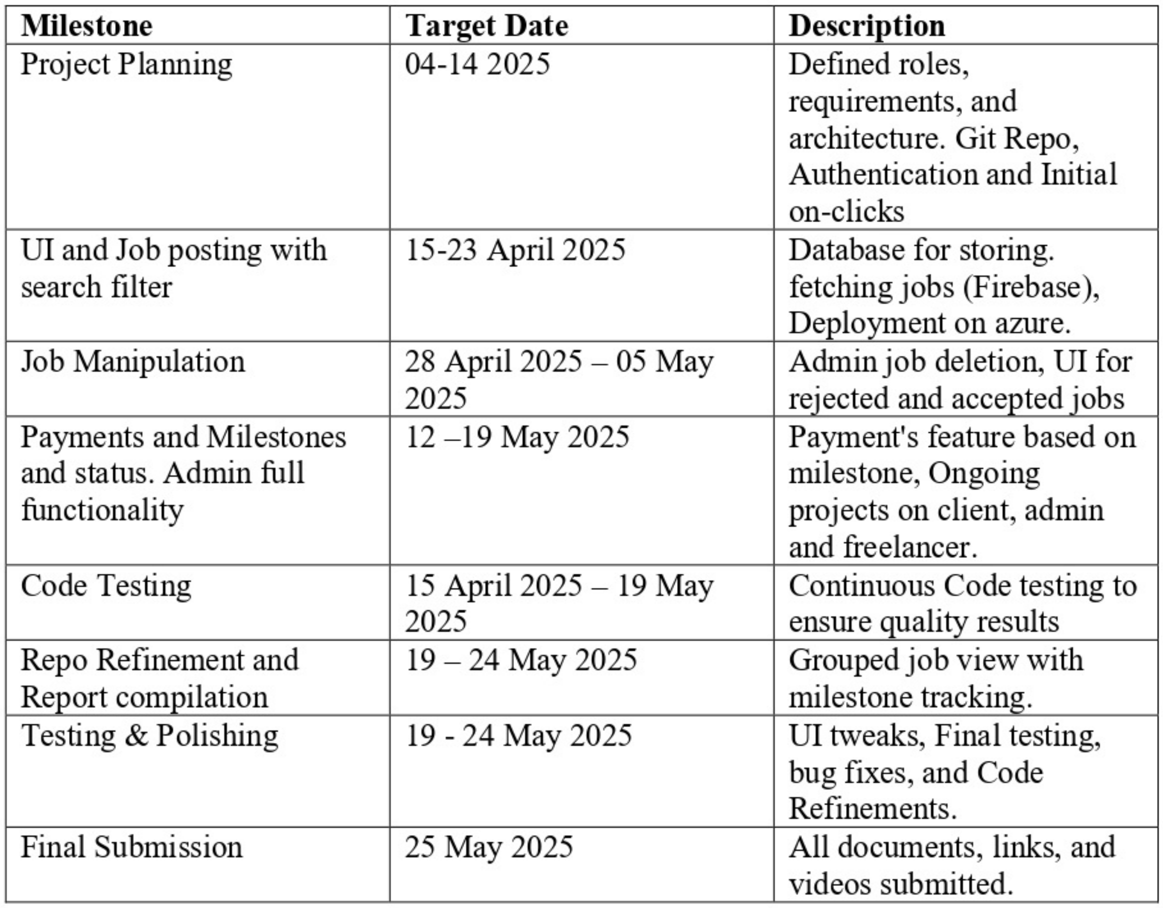
Sesihle Goniwe, 2680440
Mareme Mogoru, 2675094
Wakhiwe Ndzimandze, 2694083
Khuselo Sefohlo, 2729931
Salimata Mbaye, 2662485
Doyen Simango, 1602758

25 May 2025

Project Overview

The **Hustlr** platform connects clients and freelancers. Clients post jobs, freelancers apply for them, and admins manage platform activity and data integrity. The aim of this project is to allow job matching with real-time updates while providing an intuitive user experience.

Timeline with Milestones



Milestone	Target Date	Description
Project Planning	04-14 2025	Defined roles, requirements, and architecture. Git Repo, Authentication and Initial on-clicks
UI and Job posting with search filter	15-23 April 2025	Database for storing. fetching jobs (Firebase), Deployment on azure.
Job Manipulation	28 April 2025 – 05 May 2025	Admin job deletion, UI for rejected and accepted jobs
Payments and Milestones and status. Admin full functionality	12 –19 May 2025	Payment's feature based on milestone, Ongoing projects on client, admin and freelancer.
Code Testing	15 April 2025 – 19 May 2025	Continuous Code testing to ensure quality results
Repo Refinement and Report compilation	19 – 24 May 2025	Grouped job view with milestone tracking.
Testing & Polishing	19 - 24 May 2025	UI tweaks, Final testing, bug fixes, and Code Refinements.
Final Submission	25 May 2025	All documents, links, and videos submitted.

Figure 1: Table for Milestones

Team Roles

- **Scrum Master**: Salimata Mbaye
- **Frontend**: Wakhiwe Ndzimandze, Salimata Mbaye
- **Backend**: Doyen Simango, Sesihle Goniwe, Mareme Mogoru, Khuselo Sefohlo

Tools for Implementation

- **Frontend**: React
- **Backend**: Node.js
- **Database**: Firebase Realtime Database
- **Authentication**: Firebase Authentication (Google and Microsoft)
- **Deployment**: Microsoft Azure
- **Code Coverage**: Jest

Goals & Deliverables

- Authentication with role-based routing
- Job posting & viewing system
- Application system with persistent status
- Milestone tracking for accepted jobs
- Admin dashboard categorizing jobs by milestone progress
- Responsive UI with real-time updates

Architecture Diagram: Hustlr

MVC Architecture Overview

The **Model-View-Controller (MVC)** architecture is a widely adopted design pattern that separates an application into three main components: Model, View, and Controller. The **Model** is responsible for handling data and business logic, including storage, retrieval, and validation. The **View** manages the user interface, displaying data and reflecting changes in real time. The **Controller** acts as the bridge between the Model and the View, processing user inputs, updating data, and determining the appropriate response or interface change. This architecture provides a clear visual and logical framework for understanding the software's main components and their interactions, making it easier to manage user flows, data processing, and UI rendering in a modular and maintainable way.

Sequence Diagrams and 4+1 View Model

In designing our system, we adopted the **4+1 View Model** to structure and communicate key architectural decisions. This model allowed us to address the system's functionality, runtime behavior, implementation organization, and deployment environment while ensuring all stakeholders' concerns were addressed.

Use Case View

The Use Case View outlines the primary interactions between external actors and the system. Our key actors include:

- **Client**: Posts jobs, creates contracts, and tracks progress.
- **Freelancer**: Applies for jobs and updates milestones.
- **Administrator**: Manages users, job data, and platform integrity.

Key use case scenarios include:

- Job posting and search
- Freelancer application submission
- Contract creation and milestone assignment
- Milestone payment and status updates

Logical (Design) View

The Logical View models the functional building blocks of the system and their interactions. This view is realized using class diagrams and sequence diagrams. Key components include:

- **User Management Module**: Handles authentication and role-based access.
- **Job Management Module**: Supports job creation, editing, and filtering.
- **Contract Module**: Manages contract creation, linking users and jobs.
- **Milestone and Payment Module**: Handles milestone tracking and linked payments.

Process View

The Process View addresses the system's dynamic behavior. It captures how runtime elements interact and manage concurrent events such as:

- Simultaneous job applications by multiple freelancers.
- Real-time updates to job status and milestones.
- Concurrent authentication and access to dashboard features.

This view ensures system responsiveness, fault tolerance, and proper state management under concurrent usage.

Development (Implementation) View

This view presents the static organization of the software modules, showing how the system is divided into components and layers:

- **Frontend Layer**: Built with React, includes pages, components, and services.
- **Backend Layer**: Uses Node.js to manage APIs, logic, and Firebase operations.
- **Database**: Real-time data storage using Firebase Realtime Database.
- **Authentication Layer**: Firebase Authentication handling Google and Microsoft logins.

Physical (Deployment) View

The Physical View describes how the system is deployed in the physical environment:

- **Web App Hosting**: Deployed on Microsoft Azure cloud.
- **Database**: Firebase Realtime Database hosted in the cloud.
- **Third-Party Services**: Firebase Authentication and payment services integrated for user login and transaction support.

Together, these views helped us align the technical architecture of the **Hustlr** platform with its functional requirements, performance expectations, and deployment constraints.

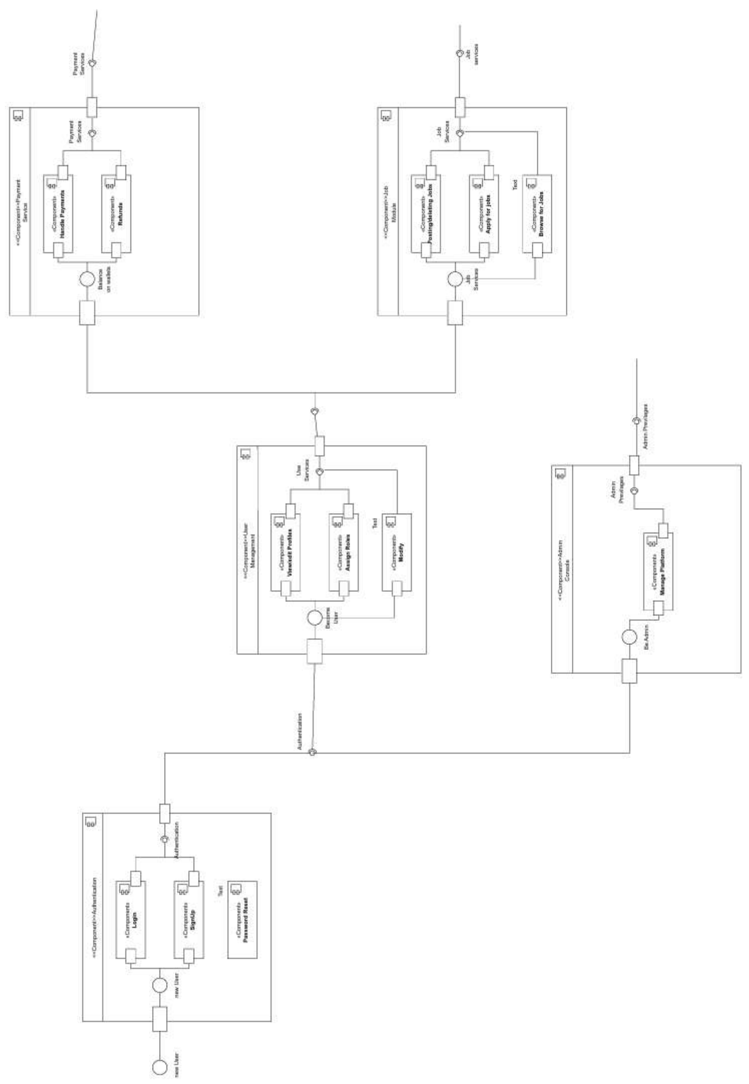

Payment Services
<<Component>>Payment Service
<<Component>> Handle Payments
<<Component>> Refunds
Balance on wallets
Job services
Job Services
<<Component>>Job Module
<<Component>> Posting/deleting Jobs
<<Component>> Apply for jobs
<<Component>> Browse for Jobs
Text
User Services
<<Component>>User Management
<<Component>> View/edit Profiles
<<Component>> Assign Roles
<<Component>> Modify
Become User
Admin Privileges
<<Component>>Admin Console
<<Component>> Manage Platform
Be Admin
Authentication
<<Component>>Authentication
<<Component>> Login
<<Component>> SignUp
<<Component>> Password Reset
new User

Figure 2: Development View

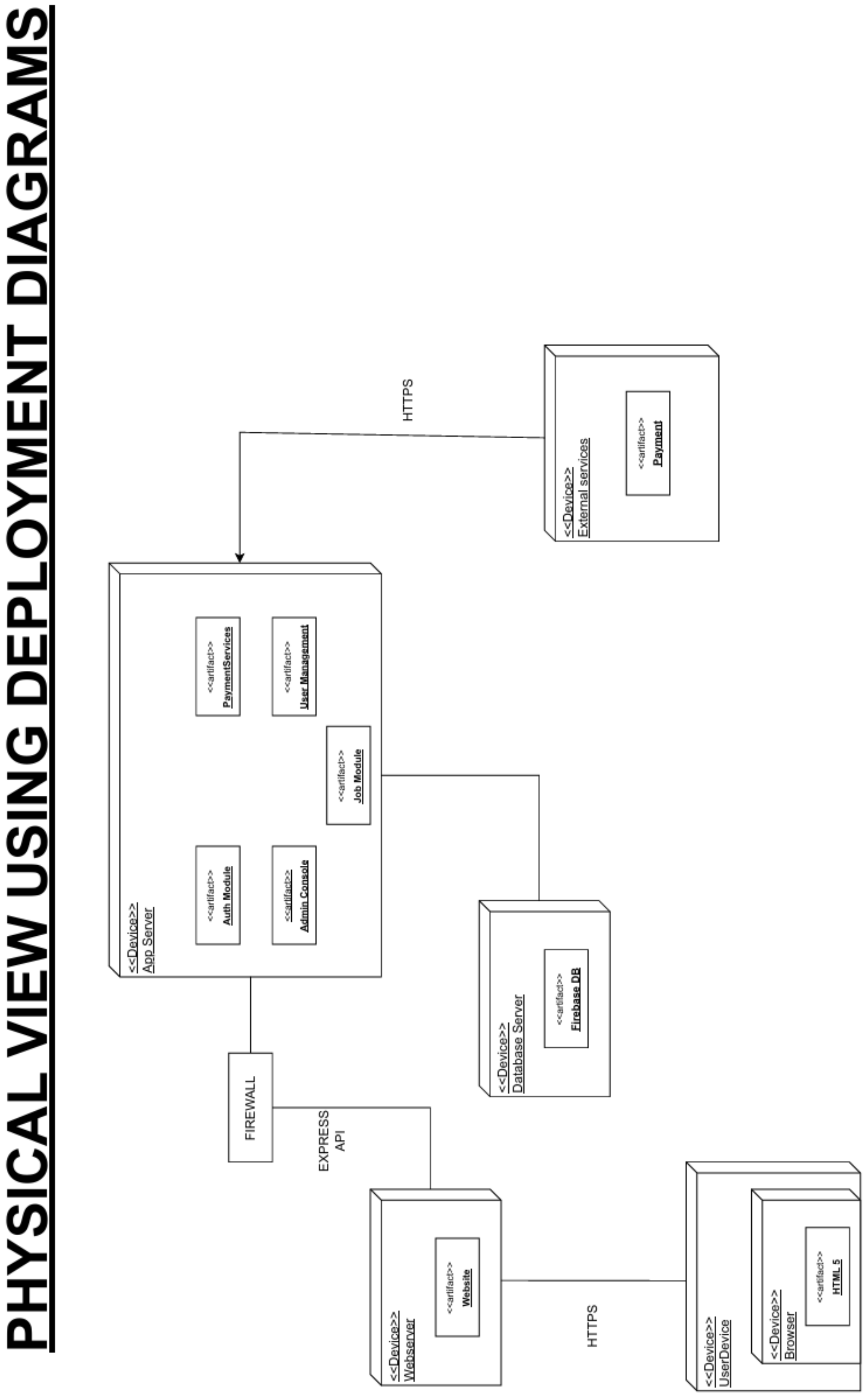


Figure 3: Physical View

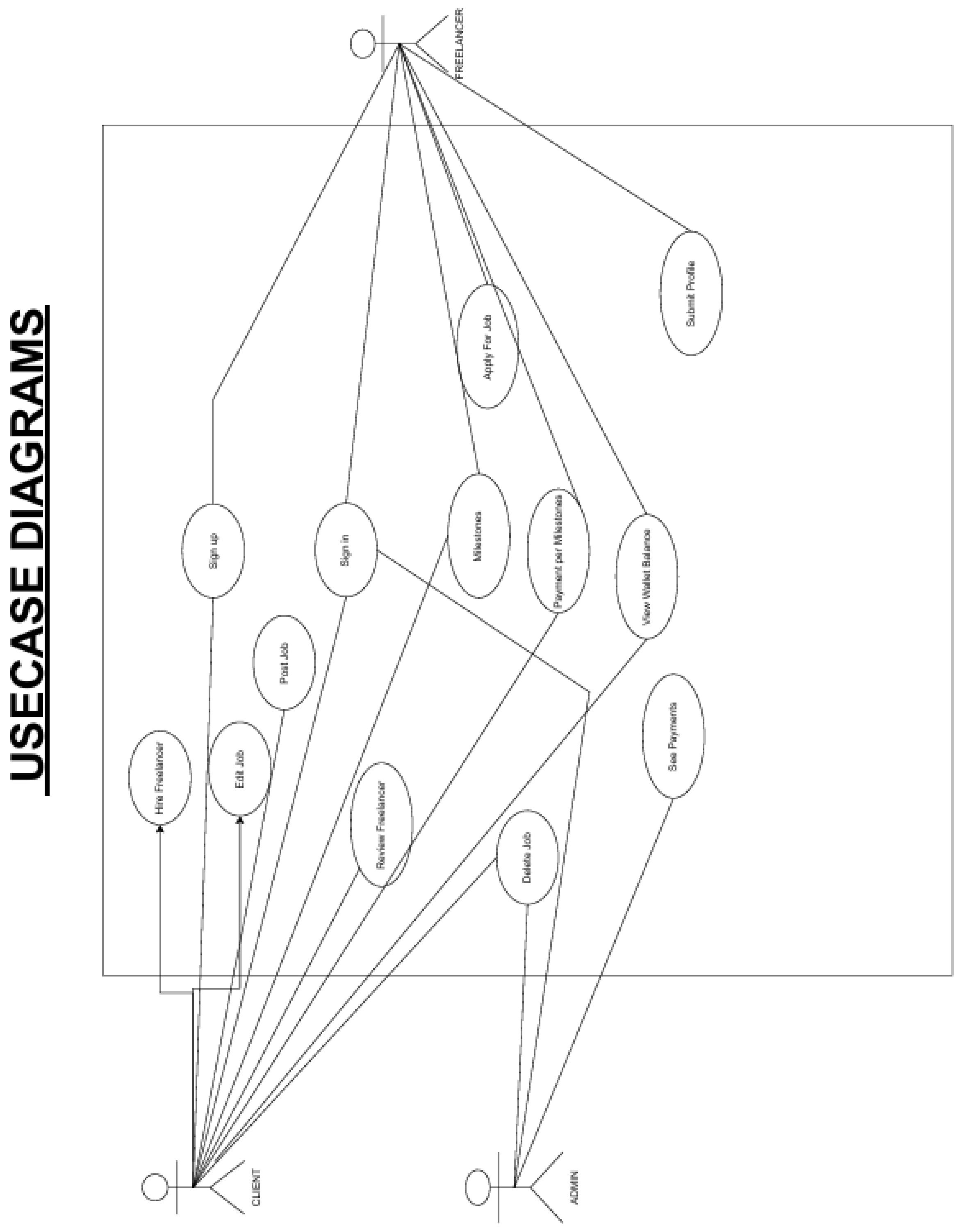


Figure 4: USE case View

Design Documents: Hustlr

Class Diagrams

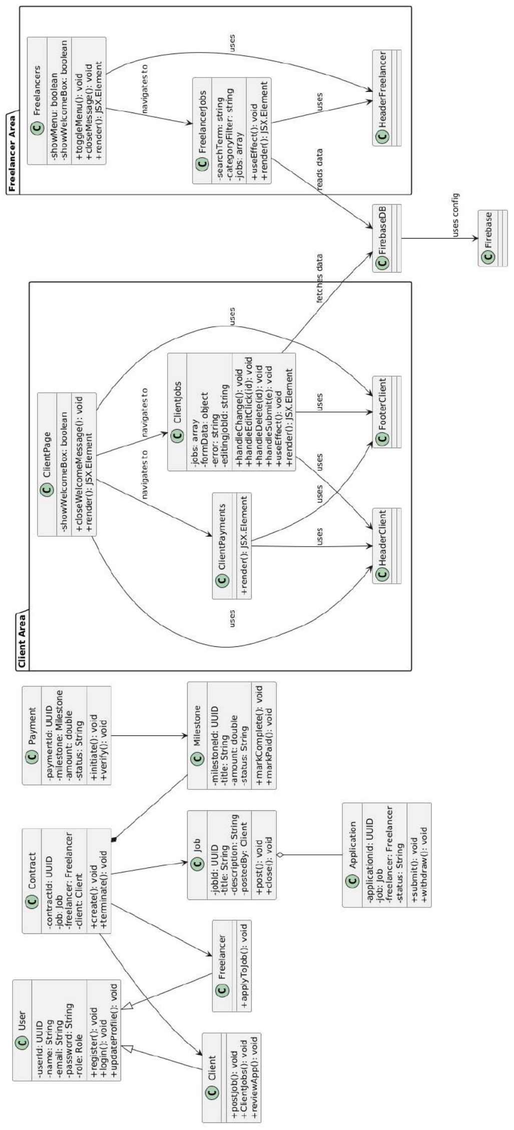


Figure 5: UML Diagram

Relationships Between Classes

This diagram represents the overall structure of the **Hustlr** system, depicting the relationships between the primary classes in the application. We first identified the logical connections between the entities involved in job posting, freelancing, and contract management.

Relationship Descriptions

- **User** has one **Profile**.
- **Client** (a type of User) can post a **Job**.
- **Freelancer** (a type of User) can apply to a **Job**.
- A **Job** has many **Applications**.
- A **Contract** links a **Client**, a **Freelancer**, and a **Job**.
- A **Contract** has many **Milestones**.
- Each **Milestone** is linked to one **Payment**.
- A **User** can generate many **Reports**.

UML Relationship Notation

Based on the above relationships, we modeled the following UML relationships:

- **Inheritance (Generalization)**:
  - `User` <|- `Freelancer`
  - `User` <|- `Client`
- **Aggregation (Empty Diamond)**:
  - `Client` – `Job` (A client "has" many jobs.)
- **Composition (Filled Diamond)**:
  - `Contract` – `Milestone` (Milestones cannot exist without the contract.)
- **Association (Plain Arrows)**:
  - `Job` $\leftrightarrow$ `Application`
  - `Contract` $\leftrightarrow$ `Client`, `Freelancer`, and `Job`
  - `Milestone` $\leftrightarrow$ `Payment`

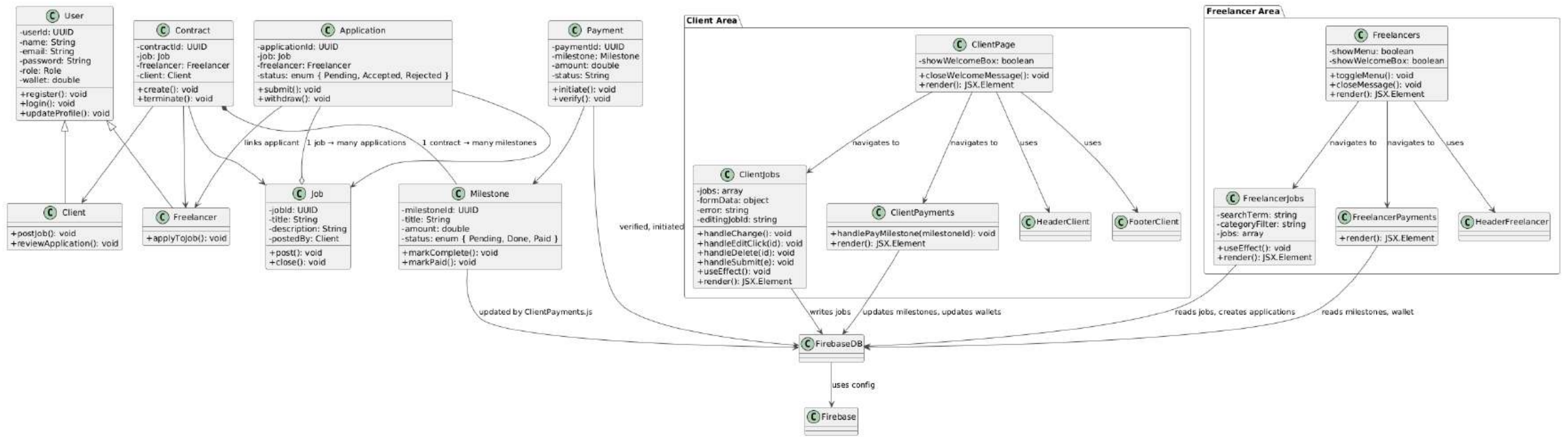


Figure 6: UML Diagram

Use case Diagrams

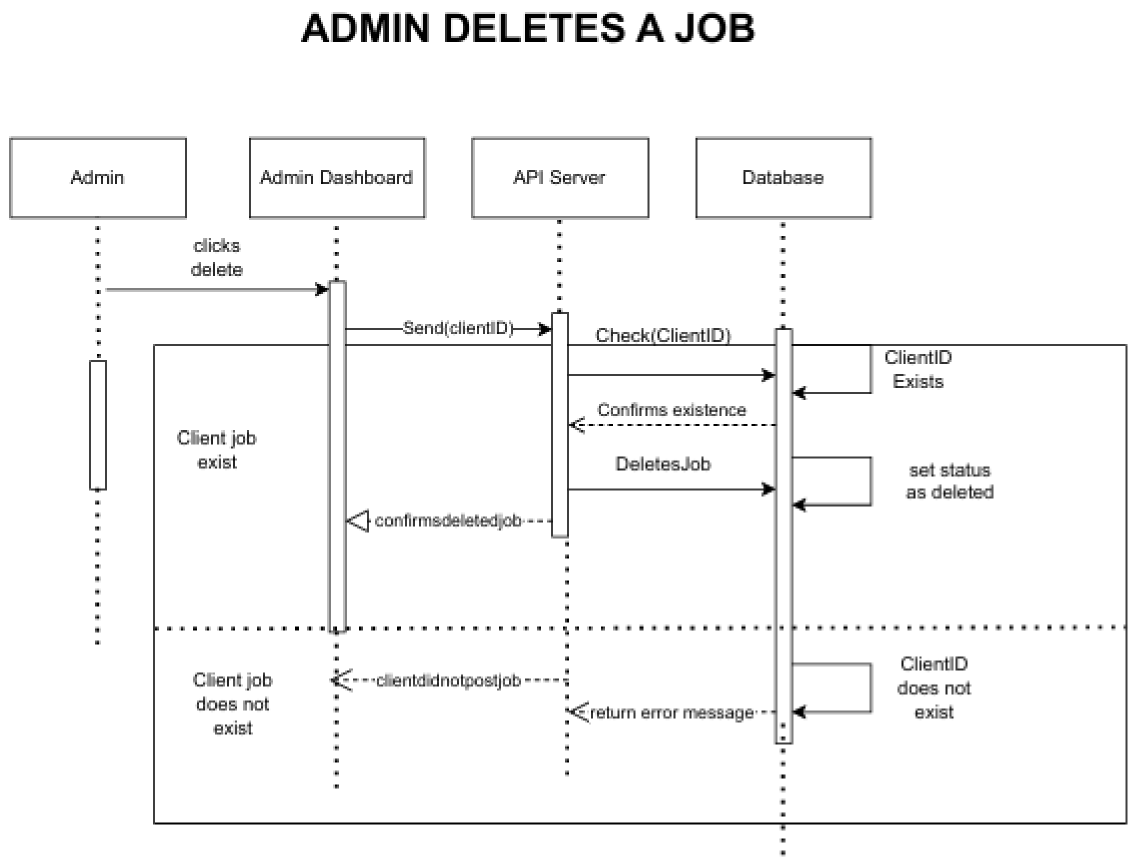


Figure 7: UML Diagram

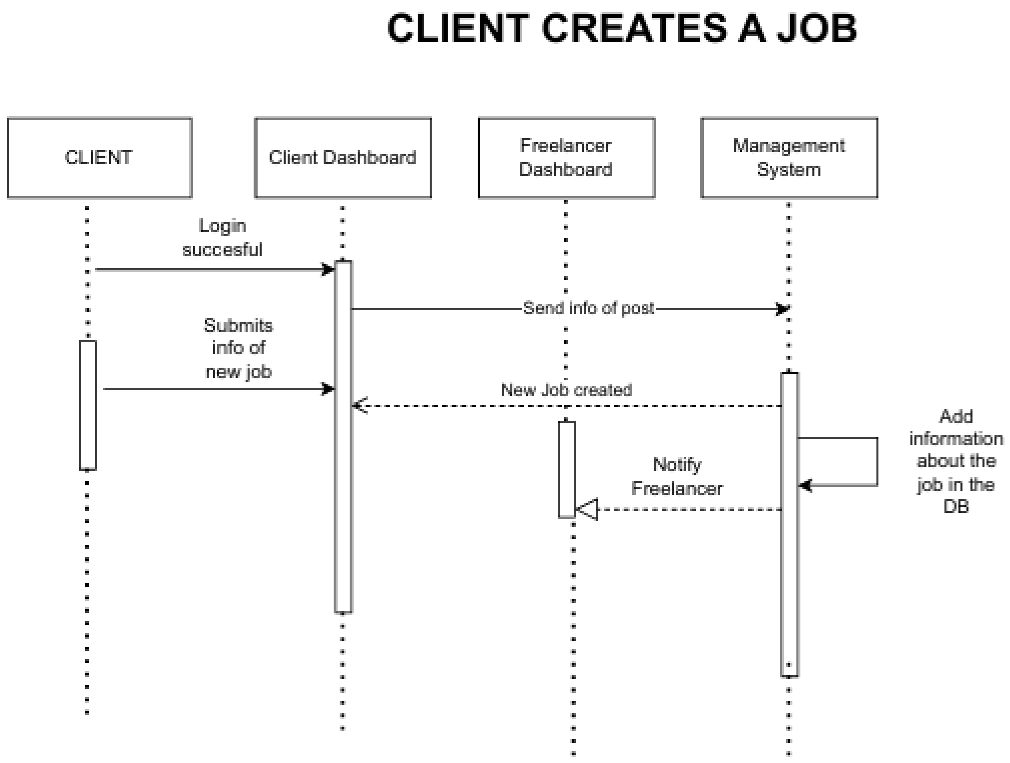


Figure 8: UML Diagram

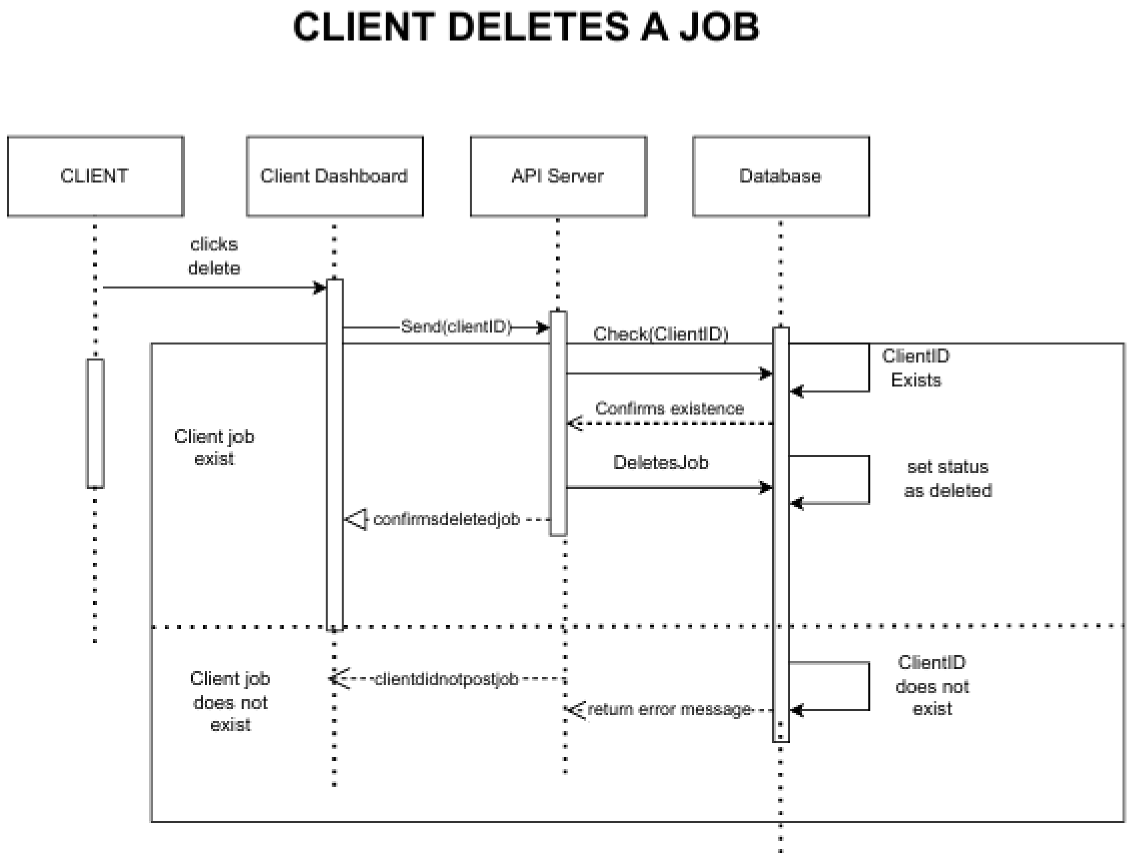


Figure 9: UML Diagram

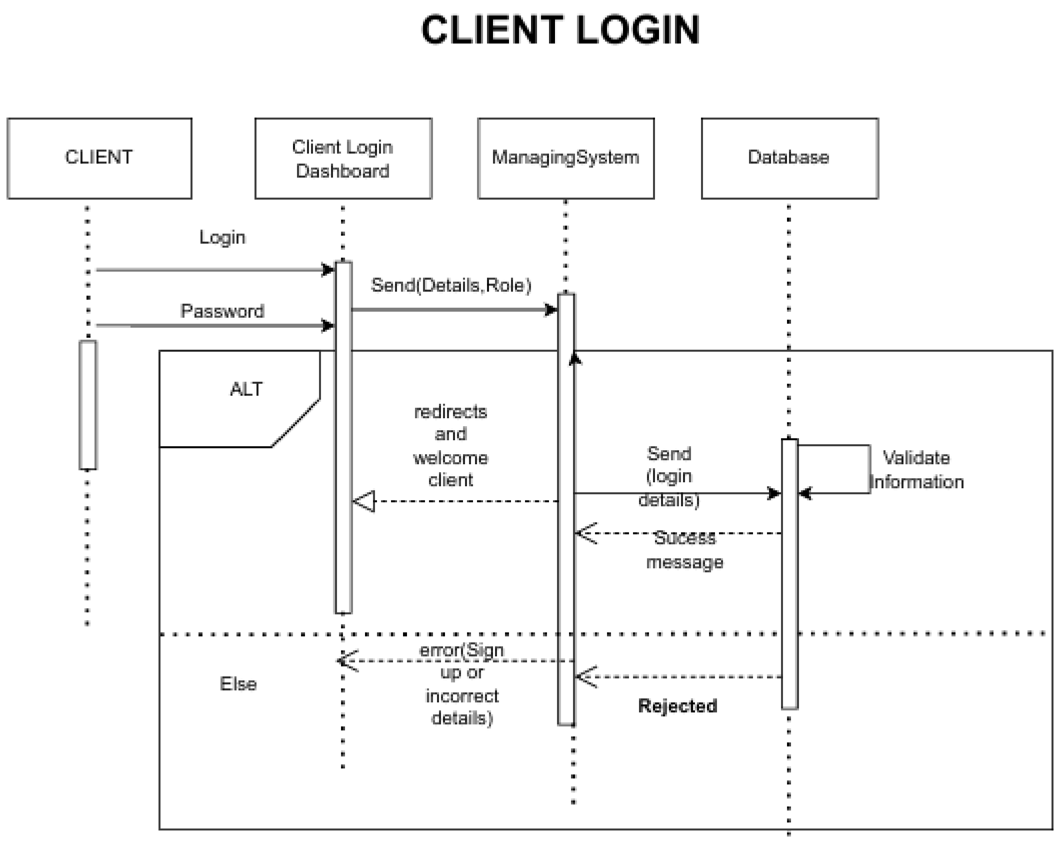


Figure 10: UML Diagram

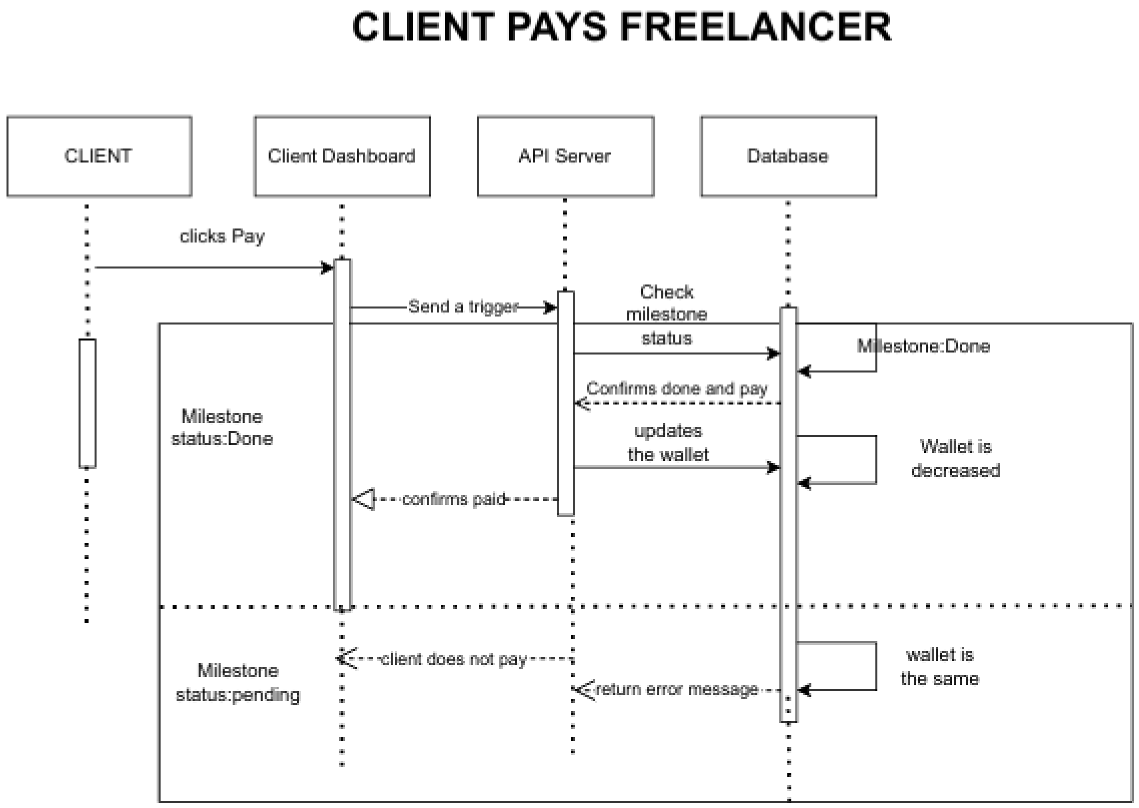


Figure 11: UML Diagram

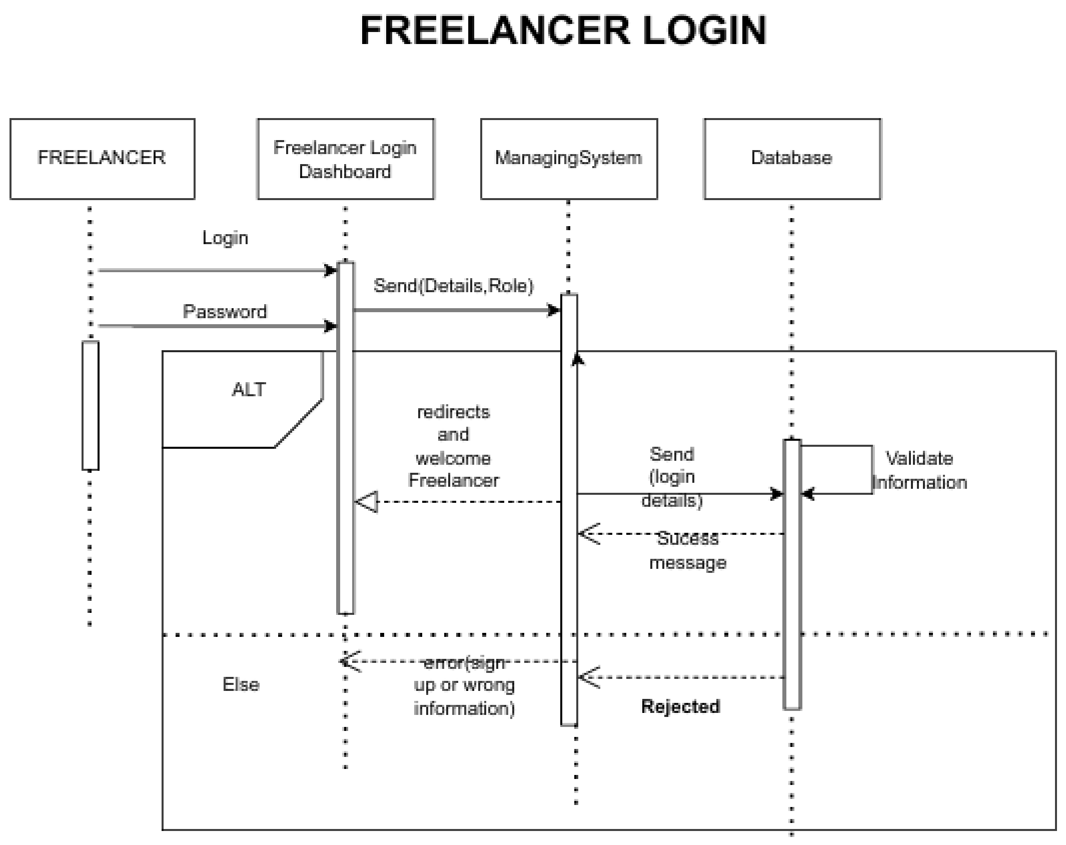


Figure 12: UML Diagram

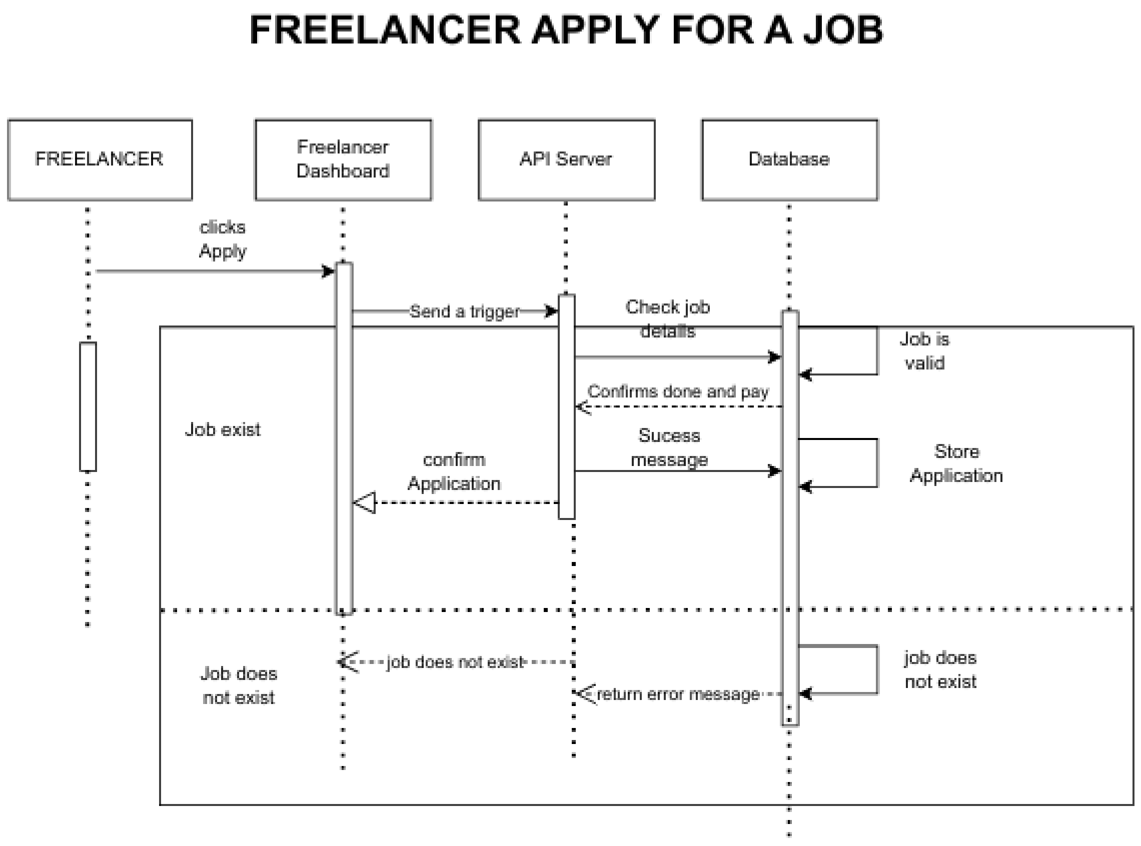


Figure 13: UML Diagram

Wireframes

Wireframing and Interface Design

For our **Freelancer and Client Management System**, we utilized **Freehand by InVision** along with references from existing internet wireframes to guide our interface design process. These wireframes played a crucial role in visualizing user interactions and aligning the user experience with our system's core functionalities such as job posting, application tracking, contract management, and milestone-based payments.

We developed distinct wireframes for:

- **Freelancer Dashboard**: Focused on browsing jobs, applying to posts, and managing accepted contracts.
- **Client Dashboard**: Enabled job posting, application review, and contract creation.
- **Admin Dashboard**: Provided views for monitoring jobs, users, and milestone progression across the platform.

Design considerations emphasized:

- **Usability**: Clear layout with minimal learning curve.
- **Clarity**: Visual hierarchy and clean design for better readability.
- **Intuitive Navigation**: Easy access to key user flows such as posting, applying, and reviewing milestones.

The use of internet-sourced templates helped us benchmark against industry standards and significantly accelerated the design process. **Freehand by InVision** enabled real-time collaboration among team members, facilitating rapid iteration and feedback during Agile sprints.

This hybrid approach allowed us to:

- Prototype key user flows early.
- Validate interface ideas with stakeholders.
- Ensure visual and functional consistency across UI components.

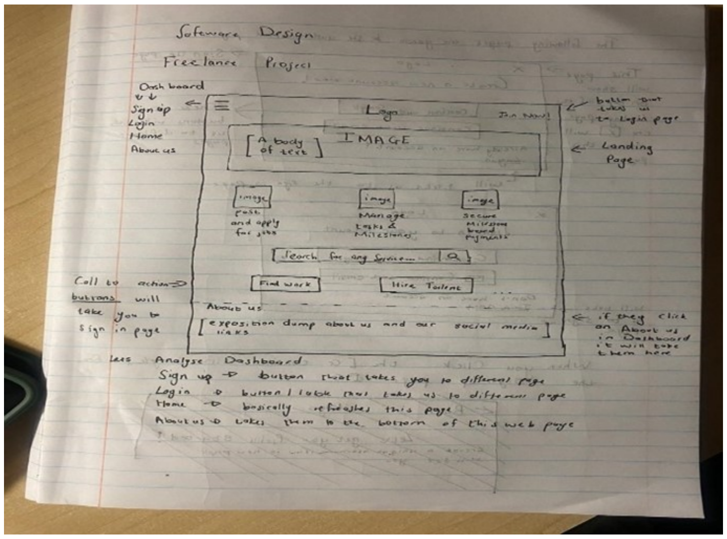

Software Design
Freelance Project
Dashboard
Sign up
Login
Home
About us
Logo
Join Now!
button that takes us to Login page
A body of text
IMAGE
Landing Page
image
post and apply for jobs
image
Manage tasks & Milestones
image
Secure milestone based payments
Search for any Service...
Find work
Hire Talent
Call to action buttons will take you to sign in page
About us
exposition dump about us and our social media links
if they click on About us in Dashboard it will take them here
Lets Analyse Dashboard
Sign up -> button that takes you to different page
Login -> button / label that takes us to different page
Home -> basically refreshes this page
About us -> takes them to the bottom of this web page

Figure 14: Home page Wireframe

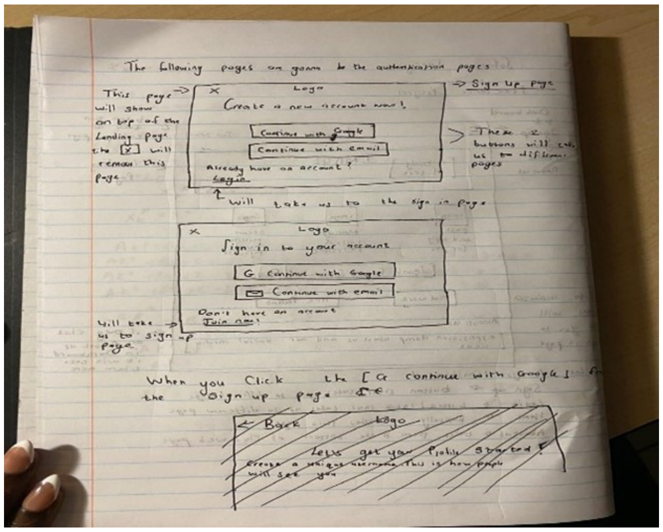

The following pages are gonna be the authentication pages
This page will show on top of the Landing page the X will remove this page
Logo
Create a new account now!
Continue with Google
Continue with email
Already have an account?
Login
Sign Up page
These 2 buttons will take us to different pages
will take us to the sign in page
Logo
Sign in to your account
G Continue with Google
Continue with email
Don't have an account
Join now!
Will take us to sign up page
When you Click the [G continue with Google] from the Sign up page
Back
Logo
Lets get your Profile Started!
Create a unique username. This is how people will see you

Figure 15: Authentication page Wireframe

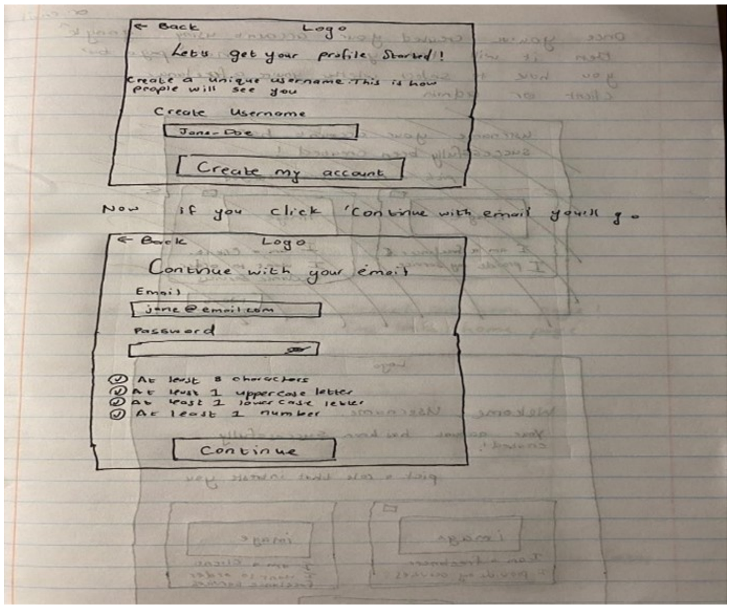

← Back
Logo
Lets get your profile Started!
Create a unique username. This is how people will see you
Create Username
Jane-Doe
Create my account
Now if you click 'Continue with email' you'll go
← Back
Logo
Continue with your email
Email
jane@email.com
Password
At least 8 characters
At least 1 uppercase letter
At least 1 lower case letter
At least 1 number
Continue

Figure 16: Authentication page Wireframe cont.

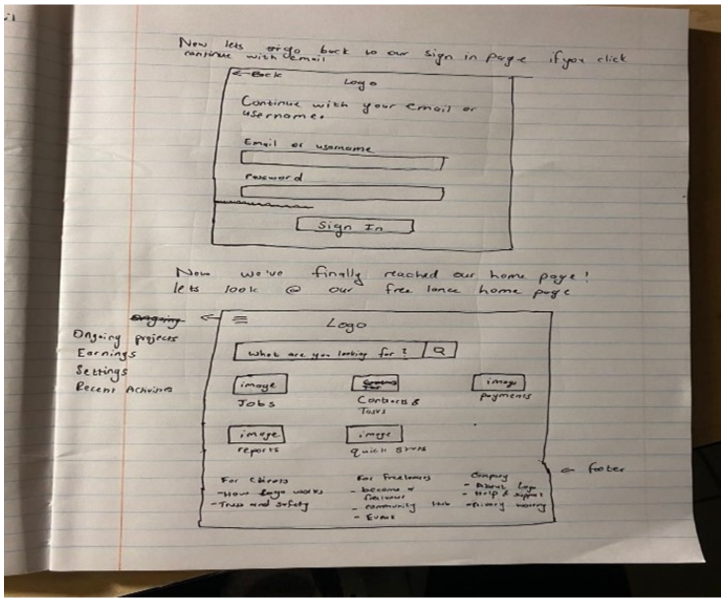


Figure 17: Dashboard page Wireframe

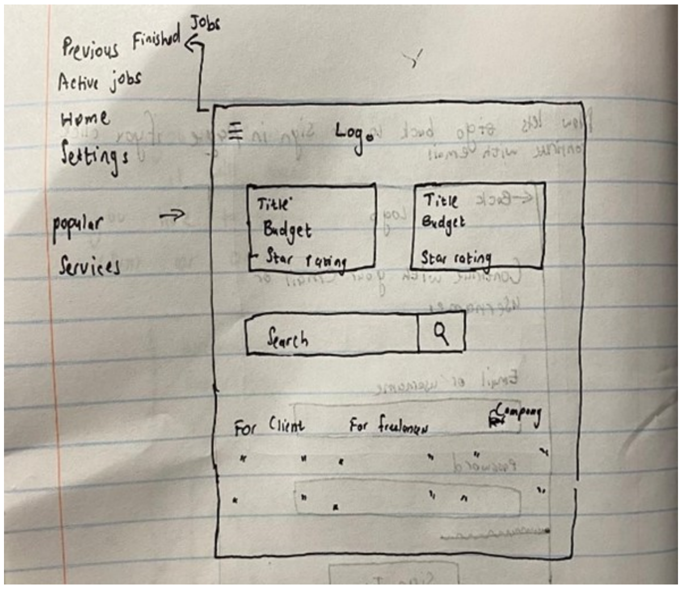


Figure 18: Jobs page Wireframe

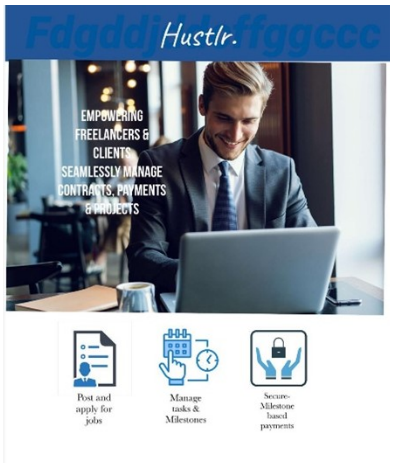

Hustlr.
EMPOWERING FREELANCERS & CLIENTS
SEAMLESSLY MANAGE CONTRACTS, PAYMENTS & PROJECTS
Post and apply for jobs
Manage tasks & Milestones
Secure-Milestone based payments

Figure 19: Jobs page Wireframe conti

Test Plan and Results: Hustlr

Testing Strategy

Our testing strategy includes unit testing and component testing using **Jest**. The approach is detailed as follows:

1. **Unit Testing**
   - **Purpose**: To verify the correctness of individual utility functions and modules in isolation.
   - **Tools**: Jest
   - **Examples**: Input sanitization, date conversions, filtering logic.
   - **Benefit**: Catches core logic bugs early.
2. **Component Testing**
   - **Purpose**: To test React components in isolation.
   - **Tools**: React Testing Library
   - **Examples**: UI rendering, interaction testing.
   - **Benefits**: Verifies user interaction and state changes.
3. **Mocking External Dependencies**
   - **Tools**: `jest.mock()`
   - **Examples**: Mock API and Firebase methods.
   - **Benefits**: Fast, safe, and deterministic testing.
4. **Edge Case & Negative Testing**
   - **Scenarios**: Empty inputs, network failures, invalid actions.
   - **Outcome**: Improved error handling and robustness.
5. **Code Coverage & Reporting**
   - **Tool**: Jest + Codecov
   - **Strategy**: Ensure key flows are covered and identify gaps.
   - **Notes**: GitHub Actions vs Codecov discrepancies due to config differences and file inclusion.

Test Environment

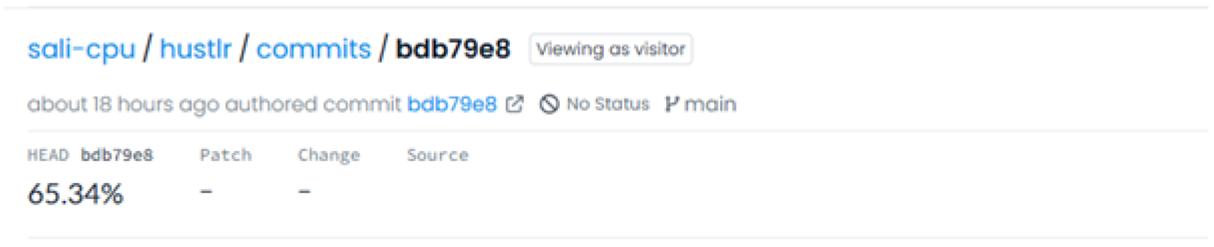
- **Framework**: Jest
- **Component Testing**: React Testing Library
- **Mocking Tools**: `jest.fn()`, `jest.mock()`

Code Coverage Summary

Metric	Coverage
Statements	64.67%
Branch	51.76%
Functions	63.52%
Lines	65.26%

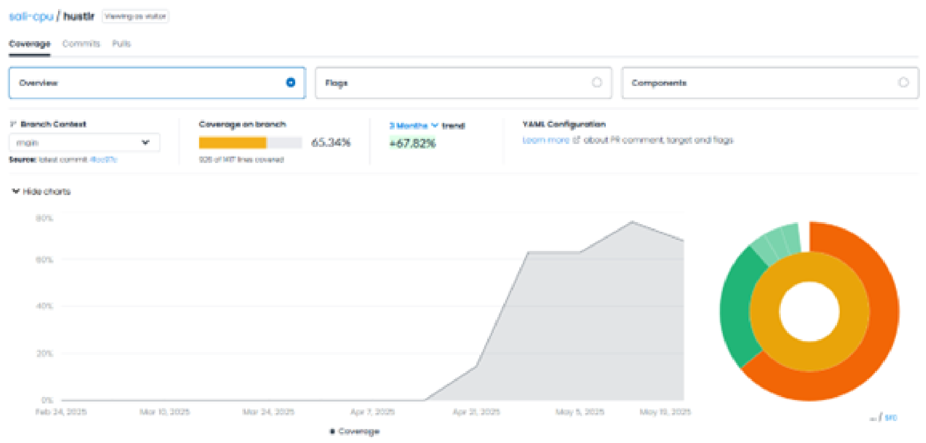
Note: Codecov and GitHub Actions show different metrics due to config and scope differences.

Sample Screenshots The code was also uploaded to Codecov and it showed the following results. The code coverage percentage on Github was different from the one uploaded to Github. GitHub Actions might show only line coverage (e.g., from Jest), while Codecov could factor in branch, statement, or function coverage, leading to variations. Codecov also analyses all source files, even those not explicitly tested, while Github only show tested files. For example, files excluded by .gitignore, jest.config.js, or nyc.config.js may not be uniformly applied.

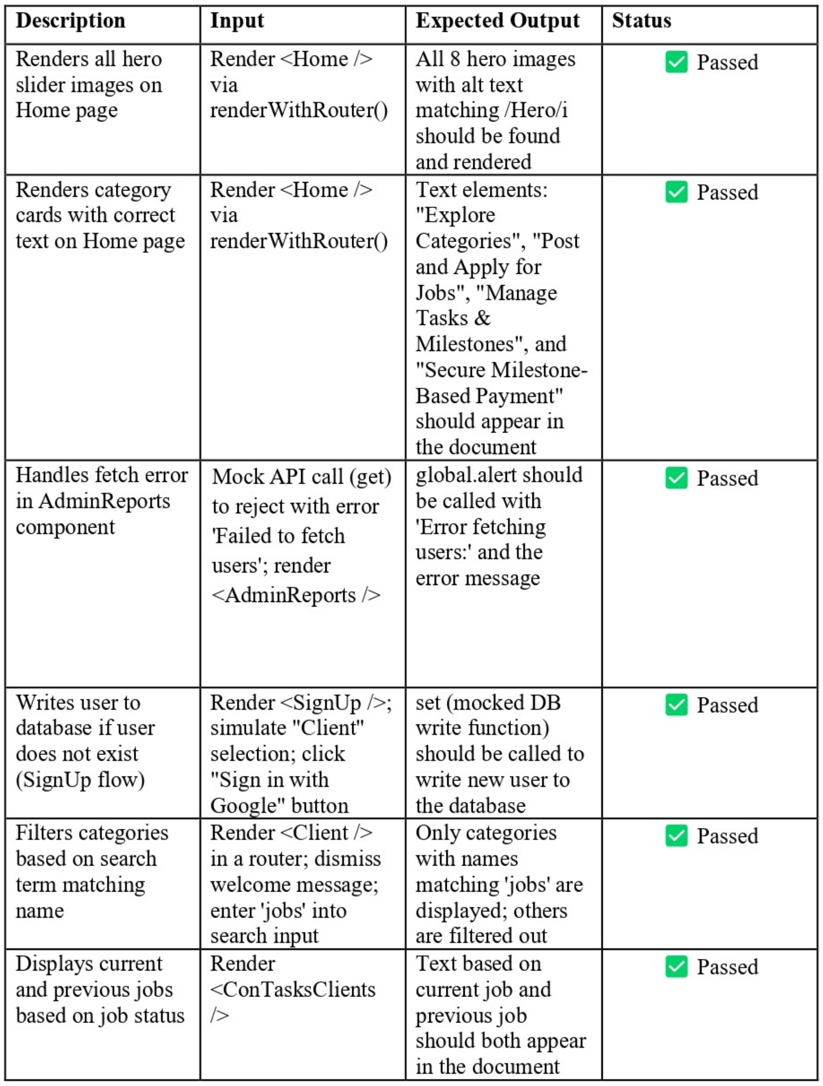

sali-cpu / hustlr / commits / bdb79e8 Viewing as visitor

about 18 hours ago authored commit bdb79e8 No Status main

HEAD bdb79e8	Patch	Change	Source
65.34%	–	–	

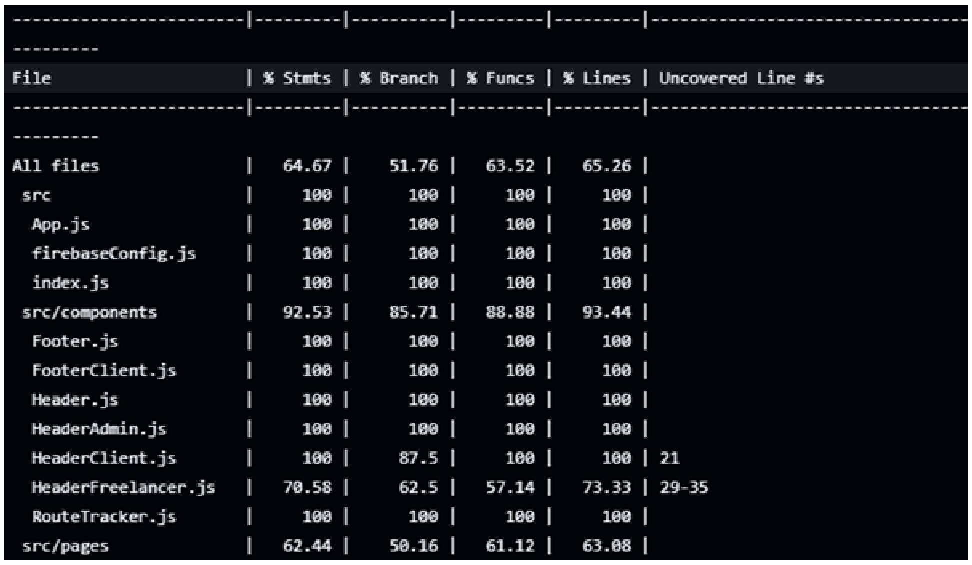


Test Cases

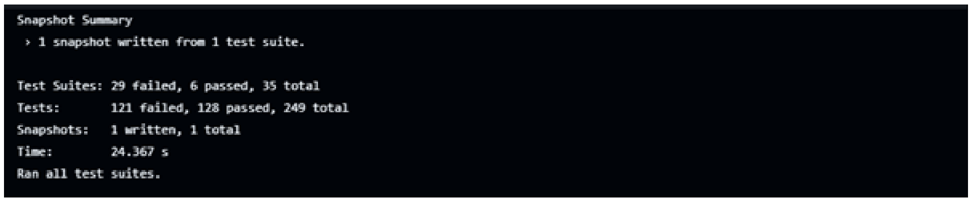


Description	Input	Expected Output	Status
Renders all hero slider images on Home page	Render <Home /> via renderWithRouter()	All 8 hero images with alt text matching /Hero/i should be found and rendered	✓ Passed
Renders category cards with correct text on Home page	Render <Home /> via renderWithRouter()	Text elements: "Explore Categories", "Post and Apply for Jobs", "Manage Tasks & Milestones", and "Secure Milestone-Based Payment" should appear in the document	✓ Passed
Handles fetch error in AdminReports component	Mock API call (get) to reject with error 'Failed to fetch users'; render <AdminReports />	global.alert should be called with 'Error fetching users:' and the error message	✓ Passed
Writes user to database if user does not exist (SignUp flow)	Render <SignUp />; simulate "Client" selection; click "Sign in with Google" button	set (mocked DB write function) should be called to write new user to the database	✓ Passed
Filters categories based on search term matching name	Render <Client /> in a router; dismiss welcome message; enter 'jobs' into search input	Only categories with names matching 'jobs' are displayed; others are filtered out	✓ Passed
Displays current and previous jobs based on job status	Render <ConTasksClients />	Text based on current job and previous job should both appear in the document	✓ Passed

Figure 20: Table Test cases



```
-----------------------|---------|----------|---------|---------|---------------------------------
---------
File                   | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------------------|---------|----------|---------|---------|---------------------------------
---------
All files              |   64.67 |    51.76 |   63.52 |   65.26 |
 src                   |     100 |      100 |     100 |     100 |
  App.js               |     100 |      100 |     100 |     100 |
  firebaseConfig.js    |     100 |      100 |     100 |     100 |
  index.js             |     100 |      100 |     100 |     100 |
 src/components        |   92.53 |    85.71 |   88.88 |   93.44 |
  Footer.js            |     100 |      100 |     100 |     100 |
  FooterClient.js      |     100 |      100 |     100 |     100 |
  Header.js            |     100 |      100 |     100 |     100 |
  HeaderAdmin.js       |     100 |      100 |     100 |     100 |
  HeaderClient.js      |     100 |     87.5 |     100 |     100 | 21
  HeaderFreelancer.js  |   70.58 |     62.5 |   57.14 |   73.33 | 29-35
  RouteTracker.js      |     100 |      100 |     100 |     100 |
 src/pages             |   62.44 |    50.16 |   61.12 |   63.08 |
```



```
Snapshot Summary
 › 1 snapshot written from 1 test suite.

Test Suites: 29 failed, 6 passed, 35 total
Tests:       121 failed, 128 passed, 249 total
Snapshots:   1 written, 1 total
Time:        24.367 s
Ran all test suites.
```