UE22CS352B - Object Oriented Analysis & Design

Mini Project Report

GigUp - A Freelancing Website

Submitted by:

PES2UG22CS231 (Janvi Munshi)
PES2UG22CS239 (Jova Varghese)
PES2UG22CS241 (Jumanah Maheen)
PES2UG22CS202 (Gowri MB)

Semester: VI   Section: D

Suja C M

January - May 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

Problem Statement

GigUp is a freelancing platform that connects clients with freelancers for project-based work. The main problems it addresses are:

1. Finding a platform for clients to post projects and find skilled freelancers
2. Providing freelancers with opportunities to showcase their skills and bid on projects
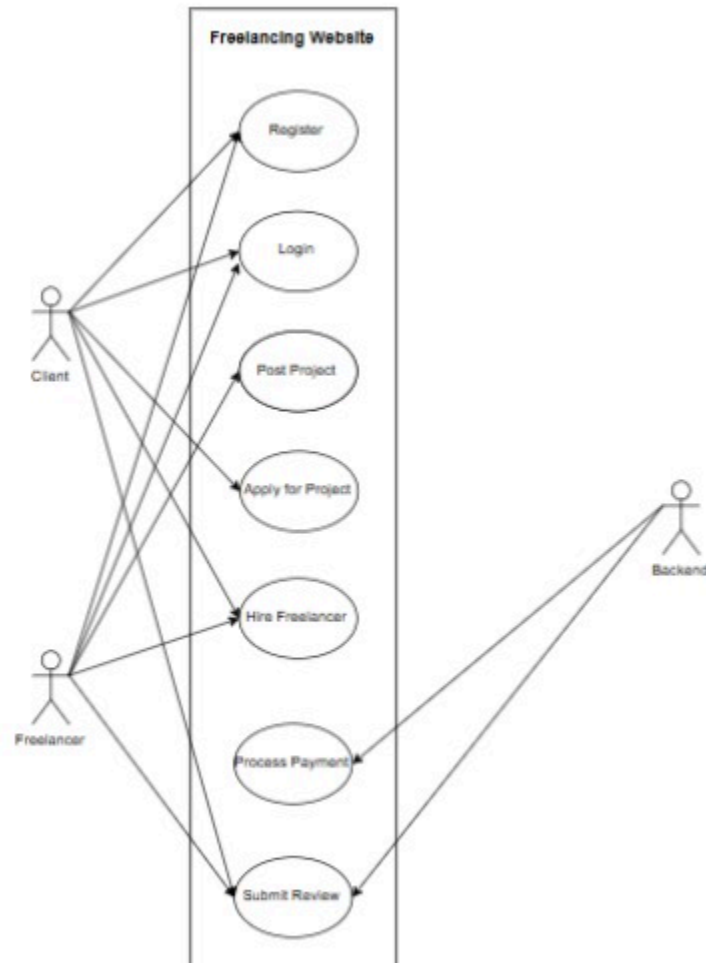3. Managing the bidding process and project lifecycle

Key Features:

1. User Management
   - Registration and authentication
   - Role-based access control
   - Profile management

2. Project Management

   - Project creation and listing
   - Project status tracking
   - Skill-based project categorization

3. Bidding System

   - Bid placement and management
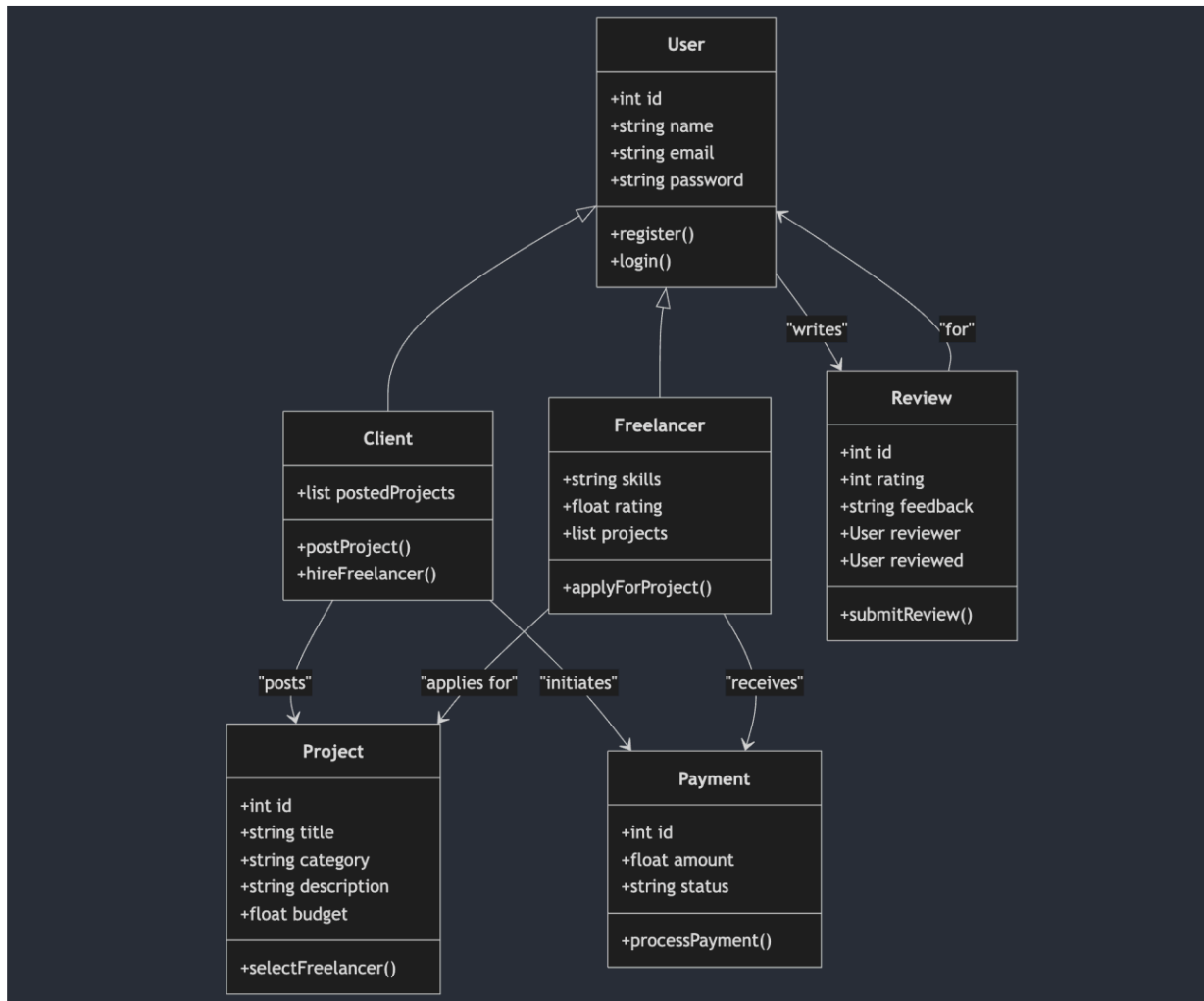   - Bid status tracking
   - Budget management

Technical Stack:

- Backend: Spring Boot, Spring Security, Spring Data JPA
- Frontend: Thymeleaf, Bootstrap, JavaScript
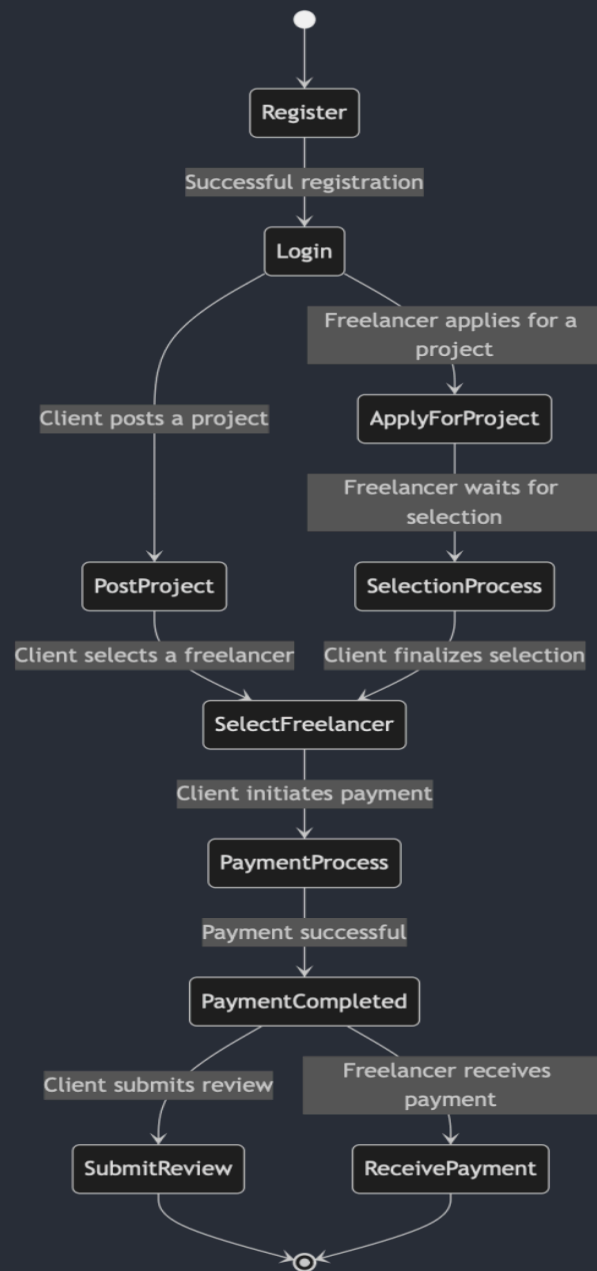- Database: MySQL
- Build Tool: Maven

Models:

1. <u>Use Case Diagram:</u>

2. Class Diagram:

User

+int id
+string name
+string email
+string password

+register()
+login()

"writes"   "for"

Review

+int id
+int rating
+string feedback
+User reviewer
+User reviewed

+submitReview()

Client

+list postedProjects

+postProject()
+hireFreelancer()

Freelancer

+string skills
+float rating
+list projects

+applyForProject()

"posts"   "applies for"   "initiates"   "receives"

Project

+int id
+string title
+string category
+string description
+float budget

+selectFreelancer()

Payment

+int id
+float amount
+string status

+processPayment()

3. <u>State Diagram:</u>

4. <u>Activity Diagrams:</u>

Architecture Patterns

Model – View – Controller Pattern (MVC)

Design Principles

1. Separation of Concerns (SoC)
    - Clear separation between controllers, services, and repositories
    - Distinct layers for business logic, data access, and presentation

2. Single Responsibility Principle (SRP)

    - Each class has a single responsibility
    - Controllers handle HTTP requests
    - Services contain business logic
    - Repositories handle data access

3. DRY (Don't Repeat Yourself)

    - Reusable components and services
    - Common functionality centralized in service classes

4. Security by Design

    - Spring Security implementation
    - Authentication and authorization
    - Secure password handling with BCrypt

Design Patterns

1. MVC (Model-View-Controller)
    - Controllers: Handle HTTP requests and responses
    - Models: Represent data entities (User, Project, Bid, etc.)
    - Views: Thymeleaf templates for UI

2. Repository Pattern

    - JPA repositories for data access
    - Abstraction of database operations
    - Consistent data access layer

3. Service Layer Pattern

- Business logic encapsulation
- Transaction management
- Service interfaces and implementations

4. Factory Pattern

- Object creation through constructors
- Lombok annotations for object creation

5. Observer Pattern

- Event handling in Spring
- Notification system for project updates

Github link to the Codebase:

https://github.com/Strange-14M605/GigUp

OUTPUT SCREENSHOTS

LOGIN/SIGNUP

HOME PAGE

BROWSE PROJECTS

PLACING A BID
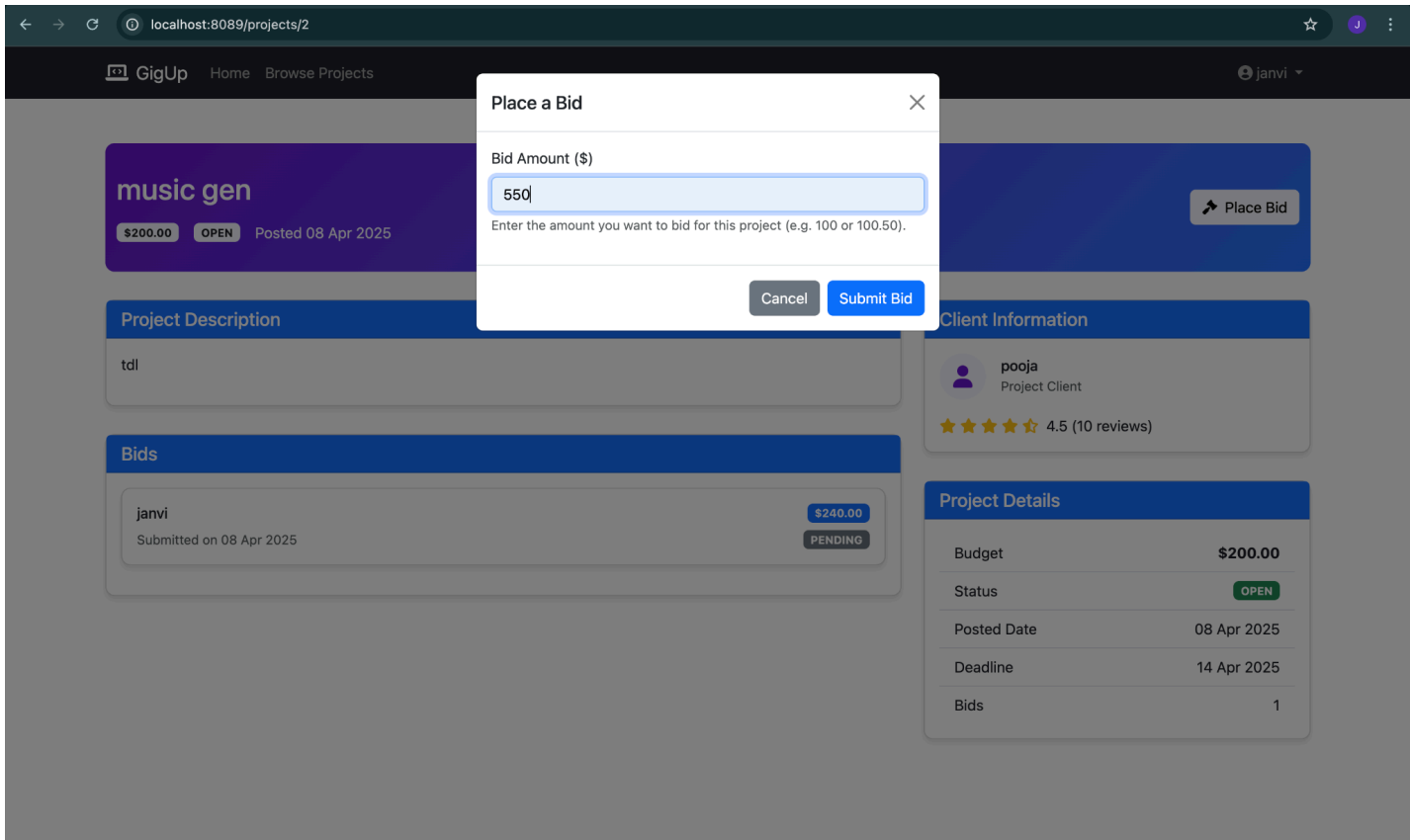
PROJECT CREATOR VIEWING BIDS

PROJECT CREATOR ACCEPTED BID

CREATING A PROJECT

VIEWING PROJECT CREATED

Individual contributions of the team members:

Name	Module worked on
JANVI MUNSHI	BIDDING
JOVA VARGHESE	CREATING PROJECT
JUMANAH MAHEEN	SIGN UP/ LOG IN
GOWRI M B	SIGN UP/ LOG IN