Freelance Bidding Platform – Mini Fiverr Clone – (By Aditya Sharma)

Abstract

This project presents the design and development of a freelance bidding platform that simulates a simplified version of Fiverr. The system enables clients to post projects and freelancers to place competitive bids. Once a bid is accepted, a contract is generated, and milestones can be tracked until completion. A messaging system supports communication, while optional payment integration through Stripe Test API demonstrates how secure transactions can be incorporated. The application is built using the MERN stack with Supabase-based authentication and showcases how modern technologies can be combined to create a functional, scalable service marketplace.

Introduction

With the rapid growth of the gig economy, freelancing platforms have become essential in connecting businesses with skilled professionals. Traditional platforms like Fiverr and Upwork inspired this project, which seeks to provide a **mini freelance marketplace** with core functionalities such as project posting, bidding, contract management, and messaging.

The objective was to build a system that is **scalable, modular, and secure**, while still lightweight enough for demonstration purposes. To achieve this, the project was implemented using the **MERN stack (MongoDB, Express, React, Node.js)** with additional integrations like Supabase for authentication and Stripe for mock payments.

The platform is designed around **two primary roles**:

- **Clients** – post projects, review bids, accept offers, and manage contracts.
- **Freelancers** – browse projects, place bids, communicate with clients, and deliver work.

This end-to-end flow replicates the lifecycle of freelance projects in real-world platforms.

Tools Used

- **Frontend**: React 18, TypeScript, Vite, TailwindCSS → fast, modular, and styled UI development.
- **Backend**: Node.js with Express.js and TypeScript → REST API with clear route design.
- **Database**: MongoDB with Mongoose ODM → schema-based modeling for users, projects, bids, contracts, milestones, and messages.
- **Authentication**: Supabase Auth → role-based authentication (client/freelancer).

- **Payments (optional)**: Stripe Test API → server generates payment intents, though full client-side integration is pending.
- **Development Tools**: Nodemon and ts-node for efficient backend development.

This stack was selected for its popularity, community support, and ease of extending features in the future.

Steps Involved in Building the Project

1. **Project Initialization**
   - Configured frontend with React, Vite, and TailwindCSS.
   - Set up backend with Node.js, Express, TypeScript, and MongoDB connection.
2. **Authentication**
   - Integrated Supabase for user sign-up/sign-in.
   - Implemented role-based access (client/freelancer) to control UI visibility.
3. **Project Posting & Browsing**
   - Clients post projects with title, description, budget, duration, and skills.
   - Freelancers browse/search/filter open projects in real-time.
4. **Bidding System**
   - Freelancers submit bids with amount, delivery time, and proposal text.
   - Server stores bids and links them to both project and freelancer.
5. **Contract Creation**
   - Clients accept a bid, triggering automatic contract creation.
   - Status updates: project moves to *in-progress*, accepted bid marked *active*, other bids marked *rejected*.
6. **Milestone Tracking**
   - Contracts support multiple milestones with status updates (*pending*, *in-progress*, *completed*, *approved*).
   - Provides transparency in work delivery.
7. **Messaging System**
   - Implemented conversations between clients and freelancers.
   - Messages stored in MongoDB with read/unread status for reliability.
8. **Payment Integration (optional)**
   - Backend implements Stripe's Payment Intent API in **test mode**.
   - Currently, the server can generate `clientSecret` tokens for secure transactions, but full client-side payment flow is yet to be completed.
9. **User Interface & Experience**
   - TailwindCSS applied for clean design and responsive layout.
   - Added loading states, filters, and conditional UI (role-based visibility).

Conclusion

The freelance bidding platform successfully replicates the **core mechanics of a service marketplace** on a smaller scale. The system allows clients and freelancers to interact seamlessly through project posting, bidding, contracts, and messaging. The backend is designed with scalability in mind, while the frontend provides an intuitive experience.