**Frontend: HTML, CSS, and JavaScript with Figma for UI Design**

For the frontend of our freelancer management platform, we chose to use core web technologies—HTML, CSS, and JavaScript—rather than relying on frontend frameworks. This choice provides greater control over the structure and behaviour of our interface, which is essential during the early development stages of our platform. Since many views in the platform—such as freelancer profiles, service listings, and dashboards—can be built as modular static components, we avoid the added complexity of frameworks while still achieving a smooth and responsive user experience. By using designs created in Figma, our development team can easily convert visual layouts into code, ensuring accurate implementation of UI/UX designs.

**Backend: Firebase**

Firebase is well-suited for our backend due to its serverless architecture and ease of integration with the rest of our stack. The platform allows us to manage backend logic—such as posting gigs, sending messages, and updating job statuses—through Firebase Cloud Functions. This eliminates the need to manage traditional server infrastructure, which saves time and reduces overhead. Since our freelancer platform requires real-time updates (e.g., status changes, notifications), Firebase’s ecosystem offers us the scalability and performance we need, without sacrificing development speed or security.

**Database: Firebase Firestore**

For data storage, we selected Firebase Firestore, which aligns perfectly with the dynamic and semi-structured data model of our freelancer platform. The platform must manage a variety of data types, including user profiles, job listings, reviews, and messages. Firestore’s NoSQL document model offers the flexibility to store this data in a nested and scalable format. Additionally, its real-time syncing capabilities make it ideal for implementing live chat and instant job status updates—features essential for smooth communication between freelancers and clients. Firestore also integrates seamlessly with Firebase Functions and Firebase Auth, enabling secure, consistent data handling.

**Hosting: GitHub Pages and Azure**

To deploy our frontend, we use GitHub Pages for static content hosting, ensuring fast and cost-effective delivery of HTML, CSS, and JavaScript files. For broader application hosting and future backend services, we deploy to Azure, which provides enterprise-grade scalability and integration. This hybrid setup is especially practical during development: GitHub Pages serves as a quick deployment solution, while Azure supports growth and more robust services as the platform matures. Azure also supports continuous deployment pipelines, making it a reliable choice for long-term scalability.

**Authentication and Security: Firebase Authentication**

Firebase Authentication is our choice for handling user sign-up, login, and identity verification. Our freelancer platform requires secure user accounts for both clients and freelancers, with role-based access control. Firebase Auth provides built-in support for common authentication methods such as email/password and Google login, and it integrates tightly with Firestore security rules. This means we can enforce data access restrictions directly at the database level, improving both security and consistency across the system.

**DevOps & CI/CD: GitHub Actions with Azure Deployment**

For continuous integration and deployment (CI/CD), we use GitHub Actions to automate our workflow from code to deployment. This setup enables us to automatically test and deploy code to Azure whenever updates are pushed to the main branch. This approach ensures reliability and reduces manual errors, allowing the team to focus on building new features rather than managing deployments. It also supports a collaborative environment where multiple developers can work on the platform simultaneously while maintaining stability.

**Testing: Jest and Postman**

To ensure code reliability and functional correctness, we use Jest for frontend unit testing and Postman for backend API testing. Jest allows us to validate frontend logic, such as form validation and dynamic interactions, which are common in user-facing parts of our platform. Postman is used to test Firebase Functions and database interactions, ensuring that endpoints handle inputs correctly and return expected results. This testing setup helps us catch bugs early and maintain a high-quality user experience.