Compute Labs Fullstack Engineer Take Home Challenge

Realestate RWA Crowdfunding Platform - Frontend Focus

Project Overview

Design and implement the frontend for a crowdfunding platform for real estate projects. The platform should allow users to browse and invest in fractional ownership of properties, track their investments, and participate in project governance. While the backend and blockchain integration can be mocked (big bonus if you can actually implement!), the focus should be on creating a high-quality, performant, and user-friendly frontend experience.

Recommended Tech Stack

All the techstack mentioned here and later in the requirement is not a hard requirement, you can use your preferred solution.

- Frontend: Next.js, TypeScript, Tailwind CSS, GSAP
- State Management: React Query, Zustand (or your preferred solution)
- UI Components: Radix UI, Headless UI, or custom components (or your preferred solution)
- Testing: Jest, React Testing Library
- Performance Monitoring: Lighthouse, Web Vitals

Core Features

1. User Authentication and Profile Management
2. Property Listing and Details
3. Investment Simulation

4. Dashboard for Tracking Investments
5. Governance and Voting System UI (Bonus)
6. Admin Panel for Property Management (Bonus)

Detailed Requirements

Frontend

1. Create a responsive and intuitive user interface
2. Implement the following pages/components:
    - Home page with featured properties
    - Property listing page with search and filter functionality
    - Property detail page with investment options
    - User dashboard for tracking investments and returns
    - Governance page for viewing and voting on proposals
    - Admin panel for property and user management
3. Use GSAP for smooth animations and transitions (or your preferred solution)
4. Implement efficient state management using React Query for server state and Zustand for client state (Or any your preferred solution)
5. Ensure accessibility compliance (Bonus)
6. Implement proper error handling and user feedback
7. Use TypeScript for improved code quality and developer experience
8. Optimize for performance, aiming for high Lighthouse scores

Mock API and Data

1. Create a mock API using tools like MSW (Mock Service Worker) or json-server (or your preferred solution)
2. Design and implement mock data structures for:
    - User profiles

    - Property details
    - Investment records
    - Governance proposals and votes (Bonus)
3. Simulate API latency and error scenarios for realistic testing (Bonus)

Testing and Quality Assurance (Bonus)

1. Implement unit tests for components and utility functions using Jest and React Testing Library
2. Write integration tests for key user flows
3. Implement end-to-end tests using Cypress or Playwright
4. Set up continuous integration for automated testing

Performance Optimization

1. Implement code splitting and lazy loading for optimal bundle sizes
2. Use Next.js Image component for optimized image loading
3. Implement proper caching strategies for API requests
4. Optimize CSS for minimal footprint (e.g., PurgeCSS with Tailwind)
5. Implement virtualization for long lists (e.g., property listings)

UX/UI Design

1. Create a cohesive and modern design system
2. Implement responsive designs that work well on desktop, tablet, and mobile devices
3. Use micro-interactions and animations to enhance user experience
4. Ensure consistent styling and component usage across the application

Blockchain Integration (Bonus)

While the primary focus of this assignment is on frontend development, and blockchain functionality can be mocked, candidates who can implement actual blockchain integration will receive significant bonus points. This can be done using either EVM (Ethereum Virtual Machine) or Solana(Preferred):

EVM Integration (Bonus)

- Implement smart contracts using Solidity
- Integrate with Web3.js or Ethers.js for blockchain interactions
- Implement wallet connection (e.g., MetaMask integration)
- Create functions for minting property tokens and handling investments

Solana Integration (Bonus and Preferred)

- Implement smart contracts using Rust and the Anchor framework
- Integrate with @solana/web3.js for blockchain interactions
- Implement wallet connection (e.g., Phantom wallet integration)
- Create functions for minting property tokens and handling investments

Candidates choosing to implement blockchain functionality should:

1. Ensure proper error handling for blockchain transactions
2. Implement a fallback to mock data when blockchain connection is unavailable
3. Consider gas fees and transaction confirmation UX
4. Implement proper security measures for handling private keys and signatures

Note: While blockchain integration is impressive, it's not required. The primary evaluation will still focus on frontend skills, performance, and user experience.

Bonus Features (Optional)

1. Implement a dark/light mode toggle to switch between dark and light theme
2. Add real-time updates using WebSockets or Server-Sent Events
3. Implement multi-language support
4. Create an interactive property exploration feature (e.g., 3D views, virtual tours)
5. Implement Animations with three.js

6. Implement advanced filtering and sorting options for property listings

Evaluation Criteria

Your project will be evaluated based on the following criteria:

1. Code Quality and Organization
    - Clean, well-documented, and maintainable TypeScript code
    - Proper use of React hooks and functional components
    - Effective error handling and user feedback
2. Frontend Architecture
    - Efficient state management
    - Proper separation of concerns
    - Reusable component design
3. User Experience and Interface Design
    - Intuitive and responsive user interface
    - Smooth animations and transitions using GSAP
    - Accessibility compliance
4. Performance Optimization
    - Fast initial load times and time-to-interactive
    - Efficient data loading and state updates
    - High Lighthouse scores (Performance, Accessibility, Best Practices, SEO)
5. Testing and Quality Assurance
    - Comprehensive unit and integration tests
    - Proper mocking of API calls and external dependencies
6. Creative Problem-Solving
    - Innovative approaches to UX challenges
    - Thoughtful handling of edge cases and error states

7. Documentation and Setup
    - Clear README with setup instructions
    - Comprehensive component documentation (e.g., using Storybook)
    - Clear code comments and type definitions

Submission Guidelines

1. Provide a GitHub repository with your complete project code.
2. Include a comprehensive README file with:
    - Project overview
    - Setup and installation instructions
    - Notes on mock API usage
    - Any additional explanations or design decisions
3. Deploy your application and provide a live demo URL (e.g., using Vercel).
4. Prepare a brief (3-10 minutes) video walkthrough of your project, highlighting key features, UX decisions, and any challenges you faced.
5. Submit your work via email (With Title, `Frontend Take Home Assignment <your_name>`) to x@computelabs.ai and n@computelabs.ai.
    a. You will be invited to a short Q&A call to go over your project.

Time Frame

You will have 1 week to complete this assignment. (Email us if you need an extension) Focus on creating a polished, performant, and user-friendly frontend experience. Quality is valued over quantity, so prioritize core features and ensure they are implemented robustly.

This take-home assignment focuses on frontend development skills, emphasizing UI/UX design, performance optimization, and code quality. It provides an opportunity to showcase your expertise in creating engaging and efficient web applications. Feel free to reach out to x@computelabs.ai if you have any questions. We look forward to seeing your amazing solutions!