FSD Mini Project Report

**Title: Elite Housing**

2. Team Members

- Mohit Khade

- Siddharth Patilo

- Trupti Hande

- Jinal Gulhane

**3. Overview of Various Full Stack Technologies Used in the Mini Project**

**Frontend Technologies:**

- Framework/Language:[ React.js]

- Styling: [ CSS]

- Package Manager: [Yarn]

Backend Technologies:

- Server Framework: [ Node.js]

- Database: [MongoDB]

- APIs: [Express]

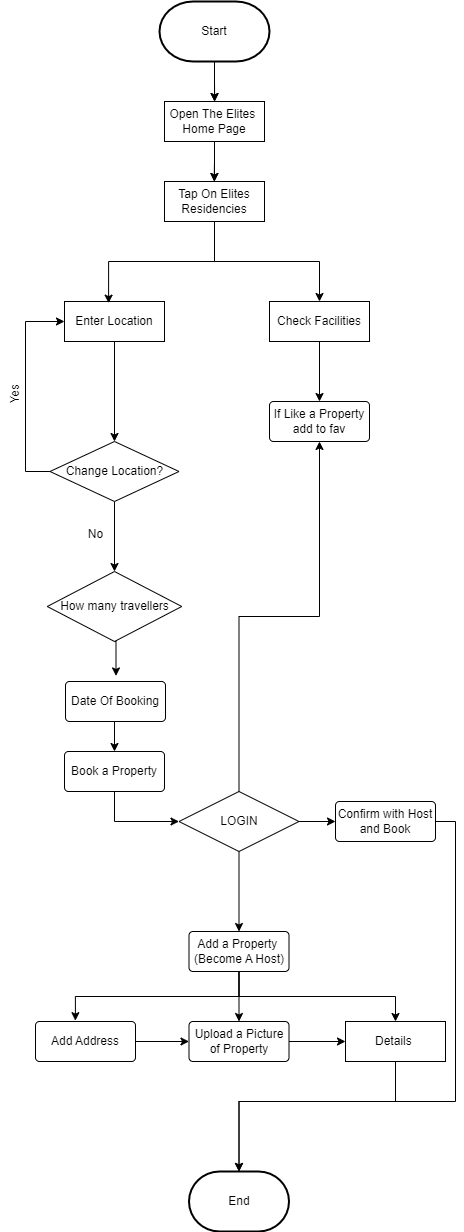
- Authentication: [ JWT, OAuth]

Deployment and Version Control:

Version Control:[Git]

Hosting/Cloud Service: [Vercel]

**4. Workflow/Architecture Diagram with Explanation**



Workflow Diagram:

Explanation:

**User Interaction:**

A user interacts with the application through the frontend, which is built using a frontend framework React Js.

The frontend consists of user interfaces and components to provide a seamless user experience.

1. **Frontend Requests:**

When a user performs an action, such as searching for listings, selecting dates, or making a booking, the frontend sends HTTP requests or API calls to the backend.

These requests are initiated by components within the frontend application, and they include relevant data (e.g., search criteria, user authentication tokens).

1. **Backend Receives Requests:**

The backend server, built using a server framework Express and Node.js receives the HTTP requests from the frontend.

It is responsible for handling the business logic, authentication, and authorization.

1. **Authentication with Auth0:**

The backend uses the Auth0 service to verify the JWT. This involves validating the token's signature, checking its expiration, and confirming that it was issued by a trusted Auth0 authority.

Auth0 provides a secure and standardized way to authenticate users, offering features like social logins, multi-factor authentication, and user management.

1. **Business Logic:**

The backend contains business logic that manages various aspects of the application, such as:

* **Listing Management**: Handling the creation, update, and retrieval of property listings.
* **Booking Management**: Processing and confirming bookings, checking availability, and managing reservations.
* **User Management**: Handling user profiles, authentication, and authorization.

**API Endpoints:**

The backend exposes API endpoints that the frontend interacts with. These endpoints are URLs that correspond to specific functionalities (e.g., /search, /bookings, /listings).

Each endpoint may require different data parameters and return specific responses.

1. **Database Interaction:**

The backend interacts with the database to perform CRUD (Create, Read, Update, Delete) operations.

For example, when a user searches for listings, the backend queries the database for matching properties based on the search criteria.

Similarly, when a user makes a booking, the backend updates the database to reflect the new reservation.

1. **Data Transformations:**

Data is transformed as it flows between the frontend and backend. This includes serializing data for transport (e.g., converting JavaScript objects to JSON for API requests) and deserializing data on the receiving end.

Data transformations also involve formatting and structuring data appropriately for the frontend components to consume and display.

1. **Backend Sends Responses:**

Once the backend processes the request and interacts with the database, it sends responses back to the frontend.

Responses contain relevant data, such as search results, property details, or booking confirmations.

1. **Frontend Updates**:

The frontend receives responses from the backend and updates the user interface accordingly.

Components re-render with the updated data, providing real-time feedback to the user.

**5. Future Scope and Conclusion:**

1. **Enhanced Search and Filtering:**
   * Implement advanced search options, such as filters for amenities, property types, and neighbourhood preferences, to provide users with more personalized search results.
2. **User Reviews and Ratings:**
   * Integrate a user review and rating system to allow guests to share their experiences and help hosts build credibility. Implement features like verified reviews and ratings to enhance trust.
3. **Instant Booking:**
   * Implement an instant booking feature that allows users to make reservations without host approval, streamlining the booking process and providing convenience for both guests and hosts.
4. **Integrated Messaging System:**
   * Develop an integrated messaging system to facilitate communication between guests and hosts. Include real-time notifications, message history, and multimedia support.
5. **Multi-Language Support:**
   * Expand the application's reach by incorporating multi-language support, enabling users from diverse linguistic backgrounds to access and use the platform more effectively.
6. **Mobile Application Development:**
   * Develop native mobile applications (iOS and Android) to enhance accessibility and provide a seamless user experience on mobile devices.
7. **AI-Powered Recommendations:**
   * Implement machine learning algorithms to analyse user preferences, behaviour, and historical data. Provide personalized recommendations for listings, destinations, and travel experiences.

**Conclusion:**

Overcoming challenges related to authentication and authorization, especially when transitioning to Auth0 and JWT, required meticulous implementation and testing. Addressing security concerns and ensuring a smooth user login experience was a significant focus.

The achievements and challenges overcome during the development process have contributed to the overall success of the project.

**6. References**

1. **Vite.js:**
   * Vite Documentation. (<https://vitejs.dev/>)
   * GitHub Repository: <https://github.com/vitejs/vite>
2. **Express.js:**
   * Express.js Documentation. (<https://expressjs.com/>)
   * GitHub Repository: <https://github.com/expressjs/express>
3. **React:**
   * React Documentation. (<https://reactjs.org/>)
   * GitHub Repository: <https://github.com/facebook/react>
4. **Leaflet:**
   * Leaflet Documentation. (<https://leafletjs.com/>)
   * GitHub Repository: <https://github.com/Leaflet/Leaflet>
5. **MongoDB Connection in React:**
   * MongoDB Atlas Documentation. (<https://docs.atlas.mongodb.com/>)
   * MongoDB Node.js Driver Documentation. (<https://docs.mongodb.com/drivers/node/>)
6. **Auth0 Documentation:**

* Auth0 Documentation is an essential resource for understanding and implementing authentication and authorization using Auth0.
* ([Auth0 Documentation](https://auth0.com/docs/))

1. **Mantine Documentation:**

* The official documentation for Mantine provides detailed information about the components, styles, and utilities offered by the library.
* [Mantine Documentation](https://mantine.dev/docs/getting-started/)

1. **Mantine GitHub Repository:**

* The Mantine library is an open-source project, and its GitHub repository is the central hub for contributions, issues, and updates.
* [Mantine GitHub Repository](https://github.com/mantinedev/mantine)

1. **Prisma Documentation:**

* The official Prisma documentation provides comprehensive guidance on setting up and using Prisma, including tutorials, API references, and best practices.
* [Prisma Documentation](https://www.prisma.io/docs/)

1. **MDN Web Docs - Cross-Origin Resource Sharing (CORS):**

* The Mozilla Developer Network (MDN) provides a comprehensive guide on CORS, covering concepts, headers, and server-side configurations.
* [MDN Web Docs - CORS](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS)

1. **Express Connection frontend and backend:**

* <https://youtube.com/playlist?list=PL4cUxeGkcC9iJ_KkrkBZWZRHVwnzLIoUE&feature=shared>

**7. Final Output Screenshots of Application Built:**

