## PropExchange

## Objective

Develop a full-stack MERN (MongoDB, Express.js, React.js, Node.js) application for a modern real estate marketplace, focusing on database connection, user authentication, and CRUD operations, while utilizing Redux for state management.

## Table of Contents

1. Introduction  
2. Functional Requirements  
3. Non-functional Requirements  
4. Architecture Overview  
5. Database Design  
6. API Design  
7. Frontend Design  
8. Deployment Strategy  
9. Technologies and Tools  
10. Timeline  
11. Conclusion

## 1. Introduction

This project aims to create a user-friendly platform for real estate listings, enabling buyers and sellers to connect efficiently. The marketplace will support advanced search functionalities, user authentication, and image uploads while ensuring a responsive and secure interface.

## 2. Functional Requirements

- User Authentication:  
 - Login, signup, forgot password  
 - Google OAuth integration  
  
- CRUD Operations:  
 - Manage real estate listings  
  
- Search and Filter Functionality:  
 - Filter by type, price, location, etc.  
  
- Role-based Access Control:  
 - Define roles for buyers, sellers, and admin  
  
- Real-time Updates:  
 - Notify users of new listings and inquiries  
  
- Image Upload:  
 - Use Firebase for secure image handling  
  
- Authentication Security:  
 - Password hashing and JWT-based token authentication

## 3. Non-functional Requirements

- Scalability:  
 - Handle high user traffic efficiently  
  
- Security:  
 - Implement JWT authentication and secure data handling  
  
- Usability:  
 - Intuitive UI/UX design  
  
- Performance:  
 - Ensure low latency  
  
- Accessibility:  
 - Cater to all user types

## 4. Architecture Overview

**High-Level Architecture**

* **Frontend:** React.js
* **Backend:** Express.js, Node.js
* **Database:** MongoDB

**Data Flow**

1. User interacts with the React frontend.
2. Frontend sends API requests to the Express.js backend.
3. Backend processes the request and interacts with MongoDB.
4. Response is sent back to the frontend.

## 5. Database Design

**Schema Overview**

**Users:**

* userId
* username
* email
* password (hashed)
* avatar

**Listings:**

* listingId
* name
* description
* address
* regularPrice
* discountPrice
* bathrooms
* bedrooms
* furnished
* parking
* offer
* type
* imageUrls
* userRef

## 6. API Design

**Endpoints**

**User Authentication:**

* POST /api/auth/signup
* POST /api/auth/login
* POST /api/auth/google
* POST /api/auth/forgot-password
* POST /api/auth/logout

**CRUD Operations:**

* GET /api/listings
* POST /api/listings
* PUT /api/listings/:id
* DELETE /api/listings/:id

**Search:**

* GET /api/listings/search (with filters: type, price, location, etc.)

**User Operations:**

* GET /api/users/:id
* PUT /api/users/:id
* DELETE /api/users/:id
* GET /api/users/:id/listings

## 7. Frontend Design

**UI/UX Wireframes**

* Login Page
* Dashboard
* Listings Page
* Listing Detail Page

**Component Hierarchy**

* App
  + Navbar
  + Footer
  + Routes
    - LoginPage
    - Dashboard
    - ListingsPage
    - ListingDetailPage
    - NotFound

**Pages and Component Progress**

* **About.jsx:** Complete about page
* **CreateListing.jsx:** Complete listing page
* **Home.jsx:** Complete home page
* **Listing.jsx:** Create the listing item component and show listings
* **Profile.jsx:** Complete update listing functionality
* **Search.jsx:** Add "show more listings" functionality
* **SignIn.jsx:** Create and add OAuth component
* **SignUp.jsx:** Create and add OAuth component
* **UpdateListing.jsx:** Complete update listing functionality

**Styling**

* **Tailwind CSS:**
  + Modern and responsive design with utility-first classes

## 8. Deployment Strategy

**Hosting**

* **Frontend:** Vercel or Netlify
* **Backend:** Render or AWS

## 9. Technologies and Tools

* **MongoDB:** Database
* **Express.js:** Backend framework
* **React.js:** Frontend framework
* **Node.js:** Runtime
* **Firebase:** Image uploads
* **bcryptjs:** Password hashing
* **JWT:** Token-based authentication
* **Git/GitHub:** Version control
* **Postman:** API testing
* **Redux Toolkit:** State management
* **Tailwind CSS:** Styling

## 10. Timeline

**Milestones**

1. Requirement gathering and planning
2. Set up backend and database
3. Develop frontend components
4. Integration and testing
5. Deployment and final testing

## 11. Conclusion

The real estate marketplace project leverages the MERN stack to deliver a secure, scalable, and user-friendly platform. Key highlights include JWT authentication, Redux for state management, and advanced search features. Post-deployment, continuous monitoring and enhancements will ensure long-term success.