Property Listing Backend

Overview

Relevant source files:

- README.md
- package.json

Purpose and Scope

This document provides a comprehensive overview of the Property Listing System, a full-stack web application designed for managing real estate property listings with advanced user features. This overview covers the system's architecture, core components, and key functionalities at a high level.

For detailed information about specific components:

- For API endpoint specifications, see API Reference
- For development environment setup, see Development Setup
- For implementation details of authentication and caching, see Implementation Details

System Description

The Property Listing System is a modern web application built on Next.js that provides a complete backend infrastructure for real estate property management. The system supports property CRUD operations, user authentication, advanced filtering capabilities, and social features like property favorites and recommendations.

The application follows a serverless architecture pattern using Next.js API routes, with MongoDB as the primary data store and Redis for performance optimization through caching. The system includes a comprehensive data import mechanism for bulk property data ingestion from CSV sources.

Sources: README.md5-24 package.json1-11

System Architecture

The following diagram illustrates the high-level system architecture, mapping system components to their actual code implementations:

System Architecture with Code Entities

Sources: README.md18-24 package.json12-22

Core Components

The system consists of four primary functional areas:

Component	Purpose	Key Technologies
Authentication System	User registration, login, and session management	`jsonwebtoken`, `bcryptjs`
Property Management	CRUD operations for property listings with filtering	`mongoose`, MongoDB
User Features	Favorites and recommendations functionality	MongoDB relationships
Caching Layer	Performance optimization for data retrieval	`redis`

Authentication System

The authentication system implements JWT-based user sessions with secure password hashing. User credentials are validated against MongoDB storage, with tokens managed through HTTP-only cookies for security.

Property Management

The core business logic handles property listings with support for advanced filtering across multiple attributes including location, price range, property type, and amenities. All property data is stored in MongoDB with optimized indexing for search performance.

User Features

Social features allow users to mark properties as favorites and recommend properties to other users. These features create engagement layers on top of the basic property browsing experience.

Caching Strategy

Redis caching is implemented to reduce database load and improve response times for frequently accessed property data and user sessions.

Sources: README.md9-17 package.json12-23

API Structure

The following diagram shows the actual API route structure as implemented in the codebase:

API Routes and Handlers

Sources: README.md25-51

Technology Stack

The system is built using modern web development technologies optimized for performance and developer experience:

Core Framework

- **Next.js 14.2.15** with App Router for full-stack React development
- **TypeScript** for type safety and improved developer experience
- **React 18** for the frontend user interface

Backend Technologies

- **Node.js** runtime environment
- **Mongoose 8.2.3** for MongoDB object modeling
- **JWT** (`jsonwebtoken`) for authentication
- **bcryptjs** for password hashing

Data Storage

- **MongoDB** for primary data persistence
- **Redis 4.6.13** for caching and session storage

Development Tools

- **Tailwind CSS** for styling
- **ESLint** for code quality
- **csv-parse** and **node-fetch** for data import functionality

Build Process

The application uses Next.js's built-in build system with TypeScript compilation, PostCSS processing for Tailwind CSS, and ESLint for code quality checks.

Sources: package.json12-37 README.md18-24

Data Import System

The system includes a sophisticated data import mechanism for bulk property data ingestion. The `import-data` script (`scripts/import-data.js`) processes CSV files from external sources, creates an administrative user account, and populates the MongoDB database with property listings in batches for optimal performance.

This import system enables rapid deployment with real-world property data and provides a foundation for testing and development activities.

Sources: README.md74-82 package.json10