Real-Estate Project

Project Overview

- Objective: Build a modern real estate marketplace with JWT authentication and Redux Toolkit.
- Target Audience: Users interested in buying, selling, or renting properties.

Functional Requirements

- Authentication:
  - Implement JWT authentication for secure user login and registration.
  - Include user profile management features (update, delete, sign-out).
- Listings:
  - Users can create new property listings with details such as images, description, price, etc.
  - Implement CRUD functionality for listings (Create, Read, Update, Delete).
  - Enable users to search for listings based on various criteria.
- User Interaction:
  - Users can contact the landlord for a specific listing.
  - Implement a messaging or notification system for communication between users.
- Frontend:
  - Use React.js for the frontend development.
  - Implement a responsive and user-friendly design with Tailwind CSS.

Non-Functional Requirements

- Performance: Ensure the application is responsive and can handle a reasonable number of concurrent users.
- Security: Implement secure authentication and protect user data.

Technology Stack

- MongoDB for the database
- Express.js as the backend framework
- React.js for the frontend
- Node.js as the runtime environment for the backend
- Mongoose as the ODM (Object Data Modeling) library for MongoDB
- JWT for authentication
- Redux Toolkit for state management
- Tailwind CSS for styling

Deployment

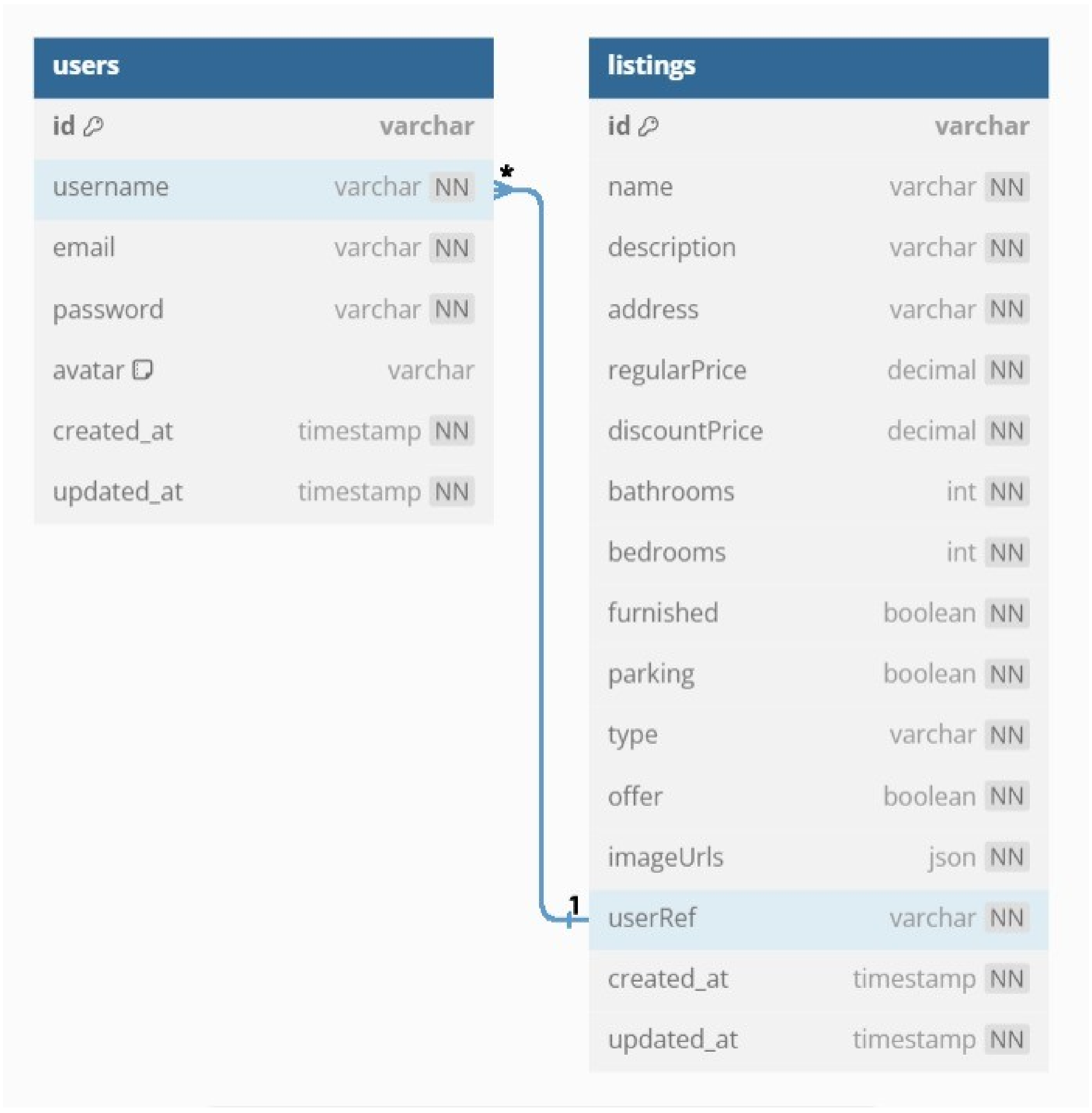
- Deploy the application to a hosting service (e.g., Render).

Design Document

System Architecture

- Client-Server architecture with React.js as the frontend and Express.js/Node.js as the backend.
- Use Redux Toolkit for state management.

Database Schema



users		
id	**varchar**	
username	varchar	NN
email	varchar	NN
password	varchar	NN
avatar	varchar	
created_at	timestamp	NN
updated_at	timestamp	NN

listings		
id	**varchar**	
name	varchar	NN
description	varchar	NN
address	varchar	NN
regularPrice	decimal	NN
discountPrice	decimal	NN
bathrooms	int	NN
bedrooms	int	NN
furnished	boolean	NN
parking	boolean	NN
type	varchar	NN
offer	boolean	NN
imageUrls	json	NN
userRef	varchar	NN
created_at	timestamp	NN
updated_at	timestamp	NN

Authentication Flow

- User registration and login with JWT tokens.
- Secure routes and API endpoints requiring valid JWT tokens.

User Interface Design

- Design responsive and intuitive UI using React.js and Tailwind CSS.
- Include pages for listings, user profile, search, etc.

API Endpoints

- Define API endpoints for user authentication, CRUD operations on listings, messaging, etc.

Error Handling

- Implement a consistent error-handling mechanism for API requests and form submissions.

Deployment Strategy

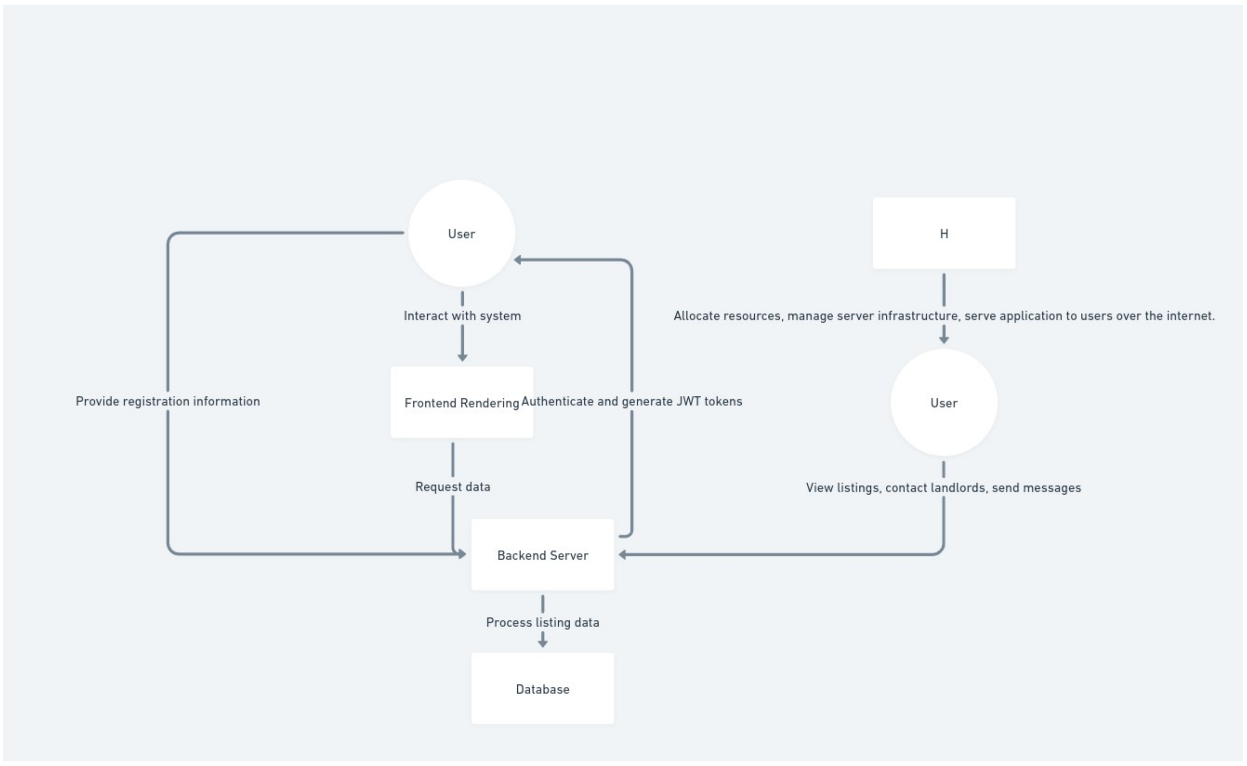
- Choose a deployment platform (e.g., Render, Vite, Github) and outline the deployment process.

Testing

- Define testing strategies for both frontend and backend components.

Scalability

- Consider potential future scalability requirements and design the system to handle increased user loads.


User
H
Interact with system
Allocate resources, manage server infrastructure, serve application to users over the internet.
Provide registration information
Frontend Rendering
Authenticate and generate JWT tokens
User
Request data
View listings, contact landlords, send messages
Backend Server
Process listing data
Database