Full Stack Development with MERN

Database Design and Development Report

Date	11-July-2024
Team ID	SWTID1720075141
Project Name	House Hunt
Maximum Marks	5 Marks

Project Title: Rent Ease – House Hunt

Date: 11-July-2024

Prepared by: A Bhavana & Korru Kiranmayee

Objective

The objective of this report is to outline the database design and implementation details for the House hunt project, including schema design and database management system (DBMS) integration.

Technologies Used

- **Database Management System (DBMS):** MongoDB
- **Object-Document Mapper (ODM):** Prisma ORM

Design the Database Schema

The database schema is designed to accommodate the following entities and relationships:

1. **Users**

- Attributes: _id, email, username, password, avatar, createdAt

2. **Posts**

- Attributes: _id, title, price, images, address, city, bedroom, bathroom, latitude, longitude, type, property, createdAt, user (references User), postDetail (references PostDetail), savedPosts (references SavedPost)

3. **PostDetails**

- Attributes: _id, desc, utilities, pet, income, size, school, bus, restaurant, post (references Post)

4. **SavedPosts**

- Attributes: _id, user (references User), post (references Post), createdAt

5. **Chats**

- Attributes: _id, users (references User), createdAt, seenBy, messages (references Message), lastMessage

6. **Messages**

- Attributes: _id, text, userId, chat (references Chat), createdAt

Implement the Database using MongoDB

The MongoDB database is implemented with the following collections and structures:

Database Name: rent_ease

1. Collection: users

  - Schema:

    ```

```
{
  "_id": "ObjectId",
  "email": "String",
  "username": "String",
  "password": "String",
  "avatar": "String",
  "createdAt": "Date"
}
```

2. Collection: posts

  - Schema:

    ```

```
 {
  "_id": "ObjectId",
```

```
  "title": "String",
  "price": "Int",
  "images": ["String"],
  "address": "String",
  "city": "String",
  "bedroom": "Int",
  "bathroom": "Int",
  "latitude": "String",
  "longitude": "String",
  "type": "String",
  "property": "String",
  "createdAt": "Date",
  "userId": "ObjectId",
  "postDetailId": "ObjectId",
  "savedPostIds": ["ObjectId"]
}
```

3. Collection: postDetails

   - Schema:

```
   {
  "_id": "ObjectId",
  "desc": "String",
  "utilities": "String",
  "pet": "String",
  "income": "String",
  "size": "Int",
  "school": "Int",
  "bus": "Int",
  "restaurant": "Int",
  "postId": "ObjectId"
```

}   ```

4. Collection: savedPosts

  - Schema:

    ```

{

  "_id": "ObjectId",

  "userId": "ObjectId",

  "postId": "ObjectId",

  "createdAt": "Date"

}    ```

5. Collection: chats

  - Schema:

    ```

{

  "_id": "ObjectId",

  "userIds": ["ObjectId"],

  "createdAt": "Date",

  "seenBy": ["ObjectId"],

  "messageIds": ["ObjectId"],

  "lastMessage": "String"

}

    ```

6. Collection: messages

  - Schema:

    ```

{

  "_id": "ObjectId",

  "text": "String",

"userId": "ObjectId",

"chatId": "ObjectId",

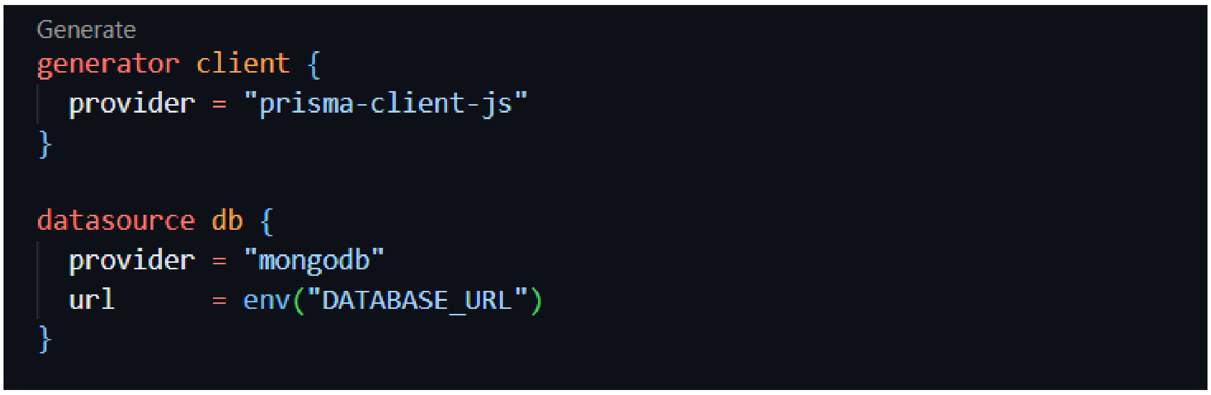
"createdAt": "Date"

}

```

**Integration with Backend**

- Database connection: Screenshot of Database connection done using Prisma
  In schema.prisma file



```
Generate
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "mongodb"
  url   = env("DATABASE_URL")
}
```

- The backend APIs interact with MongoDB using Prisma ORM. Key interactions include:

  - User Management: CRUD operations for users.
  - Post Management: CRUD operations for posts, with user authentication.
  - Post Detail Management: CRUD operations for post details associated with posts.
  - Saved Post Management: CRUD operations for saved posts, allowing users to save and retrieve their favorite posts.
  - Chat Management: CRUD operations for chats, including managing user interactions and messages.
  - Message Management: CRUD operations for messages within chats.
```