Full Stack Development with MERN

Database Design and Development Report

Date	13-07-2024
Team ID	SWTID1720073159
Project Name	Project - TuneTrail
Maximum Marks	

Project Title: Music Streaming Webapp
Date: 13-07-2024
Prepared by: : Jaiaditya Mathur, Saket DB, Riya Autade, Savithri Nair

Objective

The objective of this report is to outline the database design and implementation details for the [Your Project Title] project, including schema design and database management system (DBMS) integration.

Technologies Used

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

Design the Database Schema

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

1. admins

- Attributes: _id, name, email, password, __v

2. playlistitems

- Attributes: _id, itemId, userId, username, title, genre, singer, image, songUrl, __v

3. songs

- Attributes: _id, title, genre, singer, image, songUrl, __v

4. songs

- Attributes: _id, name, email, password, __v

Implement the Database using MongoDB

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

Database Name: MusicPlayer

1. Collection: admins

- Schema:

```
{
  _id: ObjectId,
  name: String,
  email: String,
  password: String,
  __v: integer,
}
```

2. Collection: playlistitems

- Schema:

```
{
  _id: ObjectId,
  itemId: ObjectId,
  userId: ObjectId,
  userName: String
  title: String,
  genre: String,
  singer: String,
  image: String,
  songUrl: String,
  __v: integer
}
```

```

3. Collection: songs
   - Schema:
   ```
   {
     _id: ObjectId,
     title: String,
     genre: String,
     singer: String,
     image: String,
     songUrl: String,
     __v: integer
   }
   ```
4. Collection: users
   - Schema:
   ```
   {
     _id: ObjectId,
     name: String,
     email: String,
     password: String,
     __v: integer,
   }
   ```
   ```

Integration with Backend

- Database connection:

Code Blame 15 lines (12 loc) · 403 Bytes

```
const mongoose = require('mongoose');

const MONGO_URI = 'mongodb://localhost:27017/MusicPlayer';

const connectDB = async () => {
    try {
        await mongoose.connect(MONGO_URI);
        console.log('MongoDB connected');
    } catch (err) {
        console.error('MongoDB connection error:', err.message);
        process.exit(1); // Exit process with failure
    }
};

module.exports = connectDB;
```

- The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
    - User Management: CRUD operations for users.

Code Blame 15 lines (11 loc) · 286 Bytes

```
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
    name:String,
    email: String,
    password: String,
    userId:{
        type:mongoose.Schema.Types.ObjectId,
        ref:"user",

    }
})

module.exports =mongoose.model('users',UserSchema)
```

- Admin Management: CRUD operations for admins.

Code Blame 14 lines (11 loc) · 284 Bytes

```
const mongoose =require('mongoose')

const UserSchema = new mongoose.Schema({
    name:String,
    email: String,
    password: String,
    userId:{
        type:mongoose.Schema.Types.ObjectId,
        ref:"admin",

    }
})

module.exports =mongoose.model('admin',UserSchema)
```

- Song Management: CRUD operations to add songs.

Code Blame 12 lines (9 loc) · 334 Bytes

```
const mongoose=require('mongoose')

const addsongschema =new mongoose.Schema({
    title:{type:String,required:true},
    genre:{type:String,required:true},
    singer:{type:String,required:true},
    image:{type:String,required:true},
    songUrl:{type:String,required:true},

})

module.exports=mongoose.model('songs',addsongschema)
```

- Playlist Management: CRUD operation for the creation and operations involved in a playlist.

Code Blame 14 lines (12 loc) · 535 Bytes

```
1   const mongoose = require('mongoose');
2
3   const wishlistItemSchema = new mongoose.Schema({
4       itemId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
5       userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
6       userName:{type:String,required:true},
7       title:{type:String,required:true},
8       genre:{type:String,required:true},
9       singer:{type:String,required:true},
10      image:{type:String,required:true},
11      songUrl:{type:String,required:true},
12  });
13
14  module.exports = mongoose.model('PlaylistItem', wishlistItemSchema);
```