**Full Stack Development with MERN**

**Database Design and Development Report**

|  |  |
| --- | --- |
| Date | 18-07-2024 |
| Team ID | SWTID1720167264c |
| Project Name | Cab Booking App |
| Maximum Marks |  |

**Project Title: Cab Booking App**

Date: 18-07-2024

Prepared by: NANDAM SAKETHRAM

**Objective**

The objective of this report is to outline the database design and implementation details for the Cab Booking App 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. Users**

- Attributes: \_id, name, email, password

**2. Bookings**

- Attributes: \_id, selectedPickupState, selectedPickupCity, selectedDropState, selectedDropCity, pickupdate, pickuptime, dropdate, droptime, drivername, fare, carname, carno, price, userID, userName, bookeddate

**3. Cabs**

- Attributes: \_id, drivername, carname, cartype, carno, carImage, price

**4. Admins**

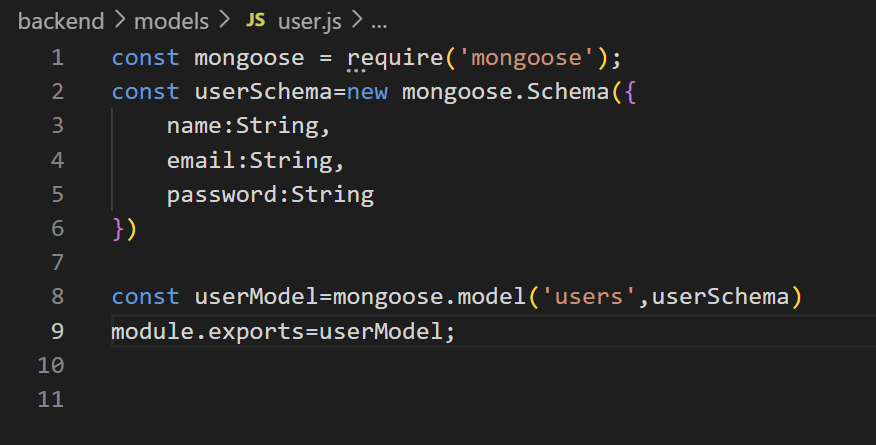
- Attributes: \_id, AdminID, password

**Implement the Database using MongoDB**

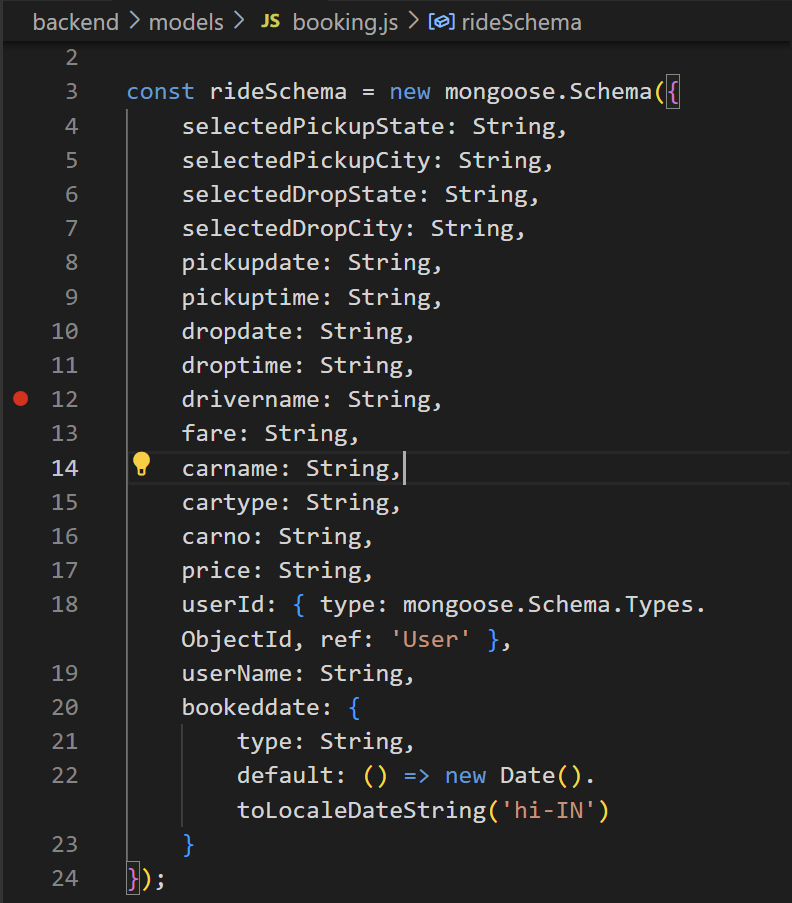
The MongoDB database is implemented with the following collections and structures:

Database Name: Cab\_Booking\_System

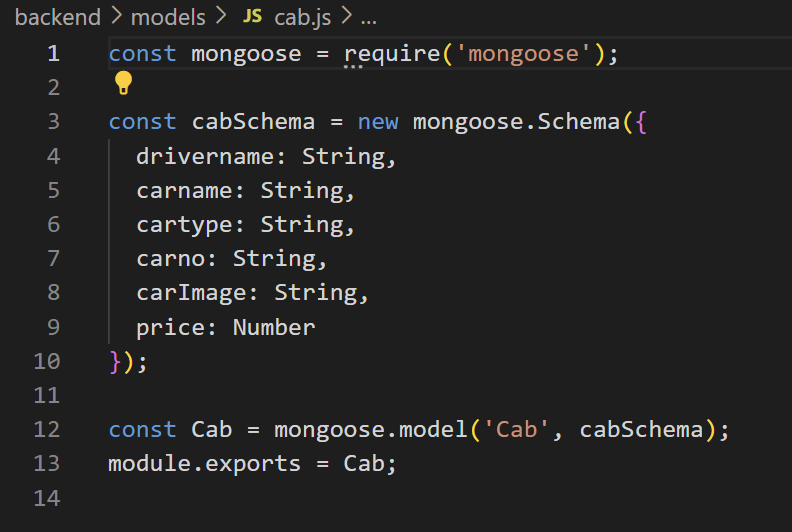
1. User Schema :



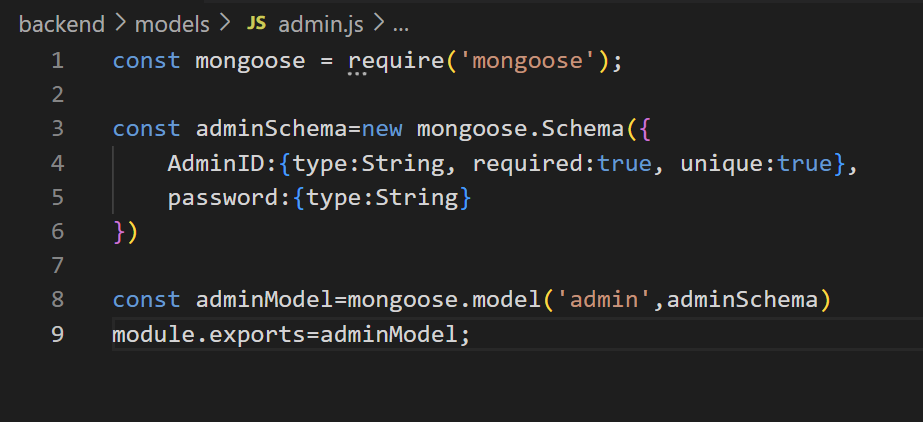
2. Bookings Schema



3. Cabs Schema

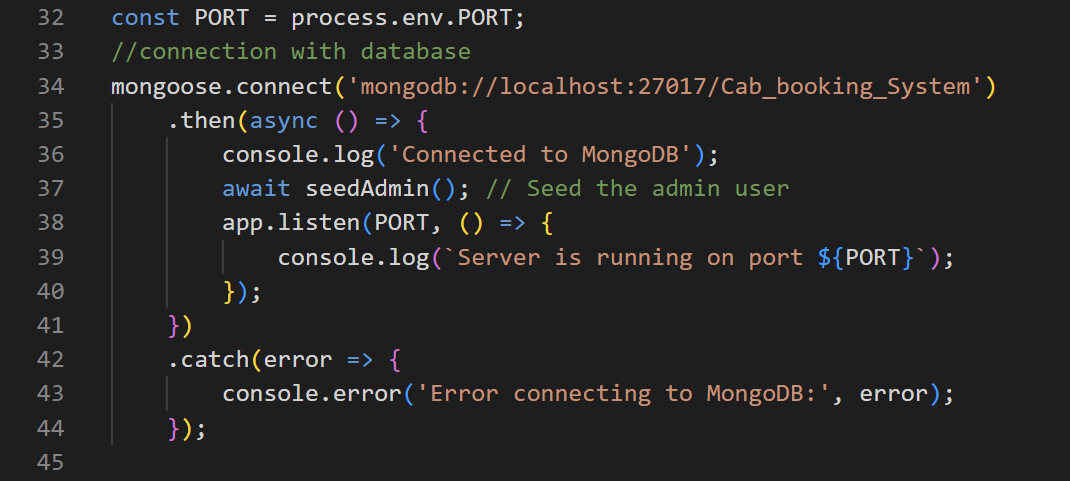


4. Admins Schema



**Integration with Backend**

* Database connection: Give Screenshot of Database connection done using Mongoose



* The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
  + User Management: CRUD operations for users.
* const userModel = require('../models/user');
* exports.getAllUsers = async (req, res) => {
* try {
* const users = await userModel.find();
* res.status(200).json(users);
* } catch (error) {
* console.error('Error fetching data ', error);
* res.status(500).json({ error: 'Internal server error' });
* }
* }
* exports.getuserbyId = async (req, res) => {
* const id = req.params.id;
* try {
* const users = await userModel.findById({ \_id: id });
* res.json(users);
* } catch (err) {
* res.status(500).json({ error: err.message });
* }
* }
* exports.updateUserById = async (req, res) => {
* const id = req.params.id;
* userModel.findByIdAndUpdate({ \_id: id }, {
* name: req.body.name,
* email: req.body.email
* })
* .then((users) => {
* res.json(users)
* }).catch((err) => {
* res.json(err)
* })
* }
* exports.postUser = async (req, resp) => {
* const { name, email, password } = req.body;
* try {
* // Check if user already exists
* const existingUser = await userModel.findOne({ email: email });
* if (existingUser) {
* return resp.json({ Status: "Error", message: "User already exists" });
* }
* // Create a new user
* const newUser = new userModel({
* name: name,
* email: email,
* password: password // Store password as plain text
* });
* // Save the user to the database
* await newUser.save();
* resp.json({ Status: "Success", user: { id: newUser.id, name: newUser.name, email: newUser.email } });
* } catch (error) {
* console.error(error);
* resp.status(500).json({ Status: "Error", message: "Server error" });
* }
* };
* exports.checkUser = async (req, resp) => {
* const { email, password } = req.body;
* userModel.findOne({ email: email })
* .then(user => {
* if (user) {
* if (user.password === password) {
* return resp.json({ Status: "Success", user: { id: user.id, name: user.name, email: user.email } })
* } else {
* resp.json("login fail")
* }
* } else {
* resp.json("no user")
* }
* })
* }
* exports.deleteUser= async (req,res)=>{
* try {
* await userModel.findByIdAndDelete(req.params.id);
* res.sendStatus(200);
* } catch (error) {
* res.status(500).json({ error: 'Internal server error' });
* }
* }
  + Bookings Management: CRUD operations for Bookings.
* const bookingModel = require('../models/booking');
* exports.bookCab = async (req, res) => {
* const {
* selectedPickupState, selectedPickupCity, selectedDropState, selectedDropCity,
* pickupdate, pickuptime, dropdate, droptime, bookeddate, userId, userName,
* drivername, fare, carname, cartype, carno, price
* } = req.body;
* try {
* const booking = new bookingModel({
* selectedPickupState, selectedPickupCity, selectedDropState, selectedDropCity,
* pickupdate, pickuptime, dropdate, droptime, bookeddate, userId, userName,
* drivername, fare, carname, cartype, carno, price
* });
* await booking.save();
* res.status(201).json(booking);
* } catch (err) {
* res.status(400).json(err);
* }
* }
* exports.getAllBookings = async (req, res) => {
* try {
* const rides = await bookingModel.find();
* res.json(rides);
* } catch (error) {
* console.error(error);
* res.status(500).send('Server Error');
* }
* }
* exports.getMyBookings = async (req, res) => {
* const userId = req.params.userId;
* try {
* console.log(`Fetching bookings for user ID: ${userId}`);
* const bookings = await bookingModel.find({ userId });
* if (!bookings || bookings.length === 0) {
* return res.json({ message: 'No bookings found for this user.' });
* }
* res.status(200).json(bookings);
* } catch (error) {
* console.error('Error fetching bookings:', error);
* res.status(500).json({ error: 'Internal server error. Please try again later.' });
* }
* };
* exports.updateRideById = async (req, res) => {
* try {
* const updatedBooking = await bookingModel.findByIdAndUpdate(req.params.id, req.body, { new: true });
* res.status(200).json(updatedBooking);
* } catch (error) {
* res.status(500).json({ error: 'Internal server error' });
* }
* };
* exports.deleteRidebyId = async (req, res) => {
* try {
* await bookingModel.findByIdAndDelete(req.params.id);
* res.status(200).json({ message: 'Ride deleted successfully' });
* } catch (error) {
* res.status(500).json({ error: 'Internal server error' });
* }
* }
  + Cabs Management: CRUD operations for Cabs .
* const Cab = require('../models/cab');
* exports.createCar = async (req, res) => {
* const { drivername, carname, cartype, carno, price } = req.body;
* const carImage = req.file ? req.file.path : null; // Check if file exists
* try {
* const car = new Cab({ drivername, carImage, carname, cartype, carno, price });
* await car.save();
* res.status(201).json(car);
* } catch (err) {
* res.status(400).json({ error: 'Failed to create car' });
* }
* };
* exports.getCabs = async (req, res) => {
* try {
* const cabs = await Cab.find();
* res.json(cabs);
* } catch (error) {
* console.error(error);
* res.status(500).send('Server Error');
* }
* };
* exports.getCabById = async (req, res) => {
* const id = req.params.id;
* try {
* const cab = await Cab.findById(id);
* if (!cab) return res.status(404).json({ error: 'Cab not found' });
* res.json(cab);
* } catch (err) {
* res.status(500).json({ error: err.message });
* }
* };
* exports.editCabById = async (req, res) => {
* const id = req.params.id;
* try {
* const updatedCab = await Cab.findByIdAndUpdate(
* id,
* {
* drivername: req.body.drivername,
* carname: req.body.carname,
* cartype: req.body.cartype,
* carno: req.body.carno,
* price: req.body.price
* },
* { new: true }
* );
* if (!updatedCab) return res.status(404).json({ error: 'Cab not found' });
* res.json(updatedCab);
* } catch (err) {
* res.status(500).json({ error: err.message });
* }
* };
* exports.deleteCabById = async (req, res) => {
* try {
* const deletedCab = await Cab.findByIdAndDelete(req.params.id);
* if (!deletedCab) return res.status(404).json({ error: 'Cab not found' });
* res.sendStatus(200);
* } catch (error) {
* res.status(500).json({ error: 'Internal server error' });
* }
* };
  + Admin Management : for Admin authentication
* const adminModel = require('../models/admin');
* exports.checkAdmin = async (req, res) => {
* const { AdminID, password } = req.body; // Ensure the field names match
* try {
* const admin = await adminModel.findOne({ AdminID: AdminID });
* if (admin) {
* if (admin.password === password) {
* return res.json({ Status: "Success", admin: { AdminID: admin.AdminID } });
* } else {
* return res.status(401).json({ Status: "Fail", message: "Incorrect password" });
* }
* } else {
* return res.status(404).json({ Status: "Fail", message: "Admin not found" });
* }
* } catch (error) {
* console.error('Error checking admin:', error);
* return res.status(500).json({ Status: "Error", message: "Internal server error" });
* }
* };