Mongo DB code

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const mongoose = require('mongoose');

const app = express();
const PORT = 3000;

// Connect to MongoDB
mongoose.connect('mongodb://localhost/car-sharing-platform', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

// Create car schema
const carSchema = new mongoose.Schema({
  make: String,
  model: String,
  year: Number,
  pricePerDay: Number,
  description: String,
});

const Car = mongoose.model('Car', carSchema);

// Create booking schema
const bookingSchema = new mongoose.Schema({
  carId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Car',
  },
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
  },
  bookingDate: Date,
});

const Booking = mongoose.model('Booking', bookingSchema);

// Create user schema
const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  password: String,
});

const User = mongoose.model('User', userSchema);

// Middleware
app.use(bodyParser.json());
app.use(cors());

// API endpoints
app.get('/api/cars', async (req, res) => {
  const cars = await Car.find();
  res.json(cars);
});

app.get('/api/bookings', async (req, res) => {
  const bookings = await Booking.find();
  res.json(bookings);
});

app.post('/api/bookings', async (req, res) => {
  const { carId, bookingDate, userId } = req.body;

  // Check if car is available for booking on the given date
  const conflictingBooking = await Booking.findOne({ carId, bookingDate });
  if (conflictingBooking) {
    res.status(409).json({ error: 'Car already booked on the given date' });
    return;
  }

  // Create new booking
  const newBooking = await Booking.create({ carId, bookingDate, userId });

  res.json(newBooking);
});

app.post('/api/users', async (req, res) => {
  const { name, email, password } = req.body;

  // Check if user already exists with the same email
  const existingUser = await User.findOne({ email });
  if (existingUser) {
    res.status(409).json({ error: 'User with the same email already exists' });
    return;
  }

  // Create new user
  const newUser = await User.create({ name, email, password });


  res.json(newUser);
});


// Start server
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```