SERVER WITH NODEJS AND MONGODB

server.js

```
const express = require("express");
const bodyParser = require("body-parser");
const dotenv = require("dotenv");
const connectDB = require("./database/db");
const userRoutes = require("./routes/user.routes");
const cors = require("cors");
dotenv.config();

const app = express();

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

const corsOptions = {
  origin: "http://localhost:8100", // Allow requests from this origin (your frontend's URL)
  methods: ["GET", "POST", "PUT", "DELETE"], // Allow these HTTP methods
  allowedHeaders: ["Content-Type"], // Allow these headers in the requests
};
app.use(cors(corsOptions));

// Connect to MongoDB
connectDB();

// Routes
app.use("/api/", userRoutes);

// Start the server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

db.js

```
const mongoose = require("mongoose");
require("dotenv").config();

const connectDB = async () => {
try {
await mongoose.connect(process.env.MONGODB_URI, {
useNewUrlParser: true,
useUnifiedTopology: true,
});
console.log("MongoDB connected!");
} catch (err) {
console.error("Error connecting to MongoDB:", err.message);
process.exit(1); // Exit the process with failure
}
};

module.exports = connectDB;
```

user.controller.js

```
const User = require("../models/user");
const bcrypt = require("bcrypt");

// Create a new user
exports.createUser = async (req, res) => {
  try {
    const { email, password } = req.body;
    const hashPass = await bcrypt.hash(password, 10);
    const newUser = new User({ email, password: hashPass });
    await newUser.save();
    res.status(201).json(newUser);
  } catch (err) {
    res.status(500).json({ message: "Server error", error: err.message });
  }
};

// Get all users
exports.getUsers = async (req, res) => {
  try {
    const users = await User.find();
    res.status(200).json(users);
  } catch (err) {
    res.status(500).json({ message: "Server error", error: err.message });
  }
};
```

```
// Get a single user by ID
exports.getUserById = async (req, res) => {
  try {
    const user = await User.findOne(req.params.email);
    if (!user) {
      return res.status(404).json({ message: "User not found" });
    }
    res.status(200).json(user);
  } catch (err) {
    res.status(500).json({ message: "Server error", error: err.message });
  }
};

// Update a user by ID
exports.updateUser = async (req, res) => {
  try {
    const user = await User.findByIdAndUpdate(req.params.id, req.body, {
      new: true,
    });
    if (!user) {
      return res.status(404).json({ message: "User not found" });
    }
    res.status(200).json(user);
  } catch (err) {
    res.status(500).json({ message: "Server error", error: err.message });
  }
};

// Delete a user by ID
exports.deleteUser = async (req, res) => {
  try {
    const user = await User.findOneAndDelete(req.params.email);
    if (!user) {
      return res.status(404).json({ message: "User not found" });
    }
    res.status(200).json({ message: "User deleted" });
  } catch (err) {
    res.status(500).json({ message: "Server error", error: err.message });
  }
};

exports.loginUser = async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });
    if (!user) {
      return res.status(400).json({ message: "Invalid Credentials" });
    }
    const isMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) {
      return res.status(400).json({ message: "Invalid Credentials" });
    }
    res.status(200).json({ email: user.email, id: user._id });
  } catch (err) {
    res.status(500).json({ message: "Server error", error: err.message });
  }
};
```

router.js

```
const express = require("express");
const router = express.Router();
const {
  createUser,
  getUsers,
  getUserById,
  updateUser,
  deleteUser,
  loginUser,
} = require("../controller/user.controller");

// Route to create a new user
router.post("/create", createUser);

// Route to get all users
router.get("/getAllUsers", getUsers);

// Route to get a single user by ID
router.get("/:id", getUserById);

// Route to update a user by ID
router.put("/:id", updateUser);

// Route to delete a user by ID
router.delete("/:id", deleteUser);

router.post("/login", loginUser);
module.exports = router;
```

model.js

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

const userSchema = new mongoose.Schema({
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
});

module.exports = mongoose.model("User", userSchema);
```