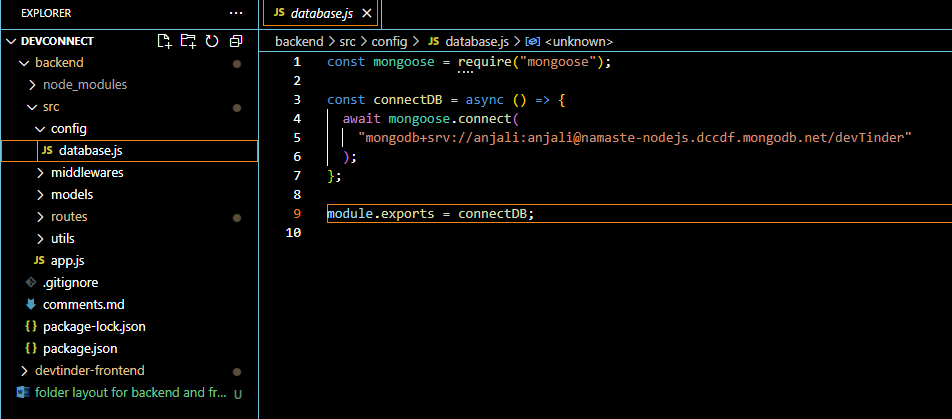
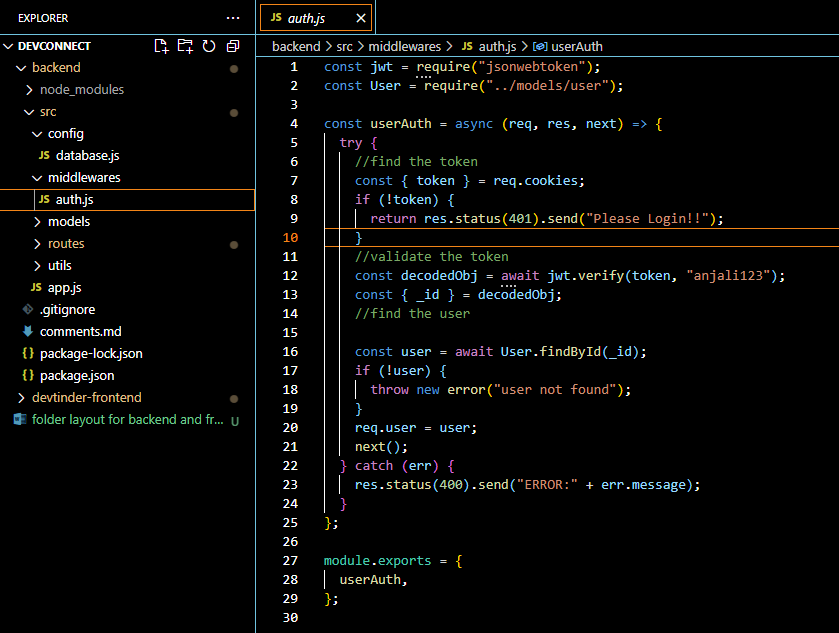


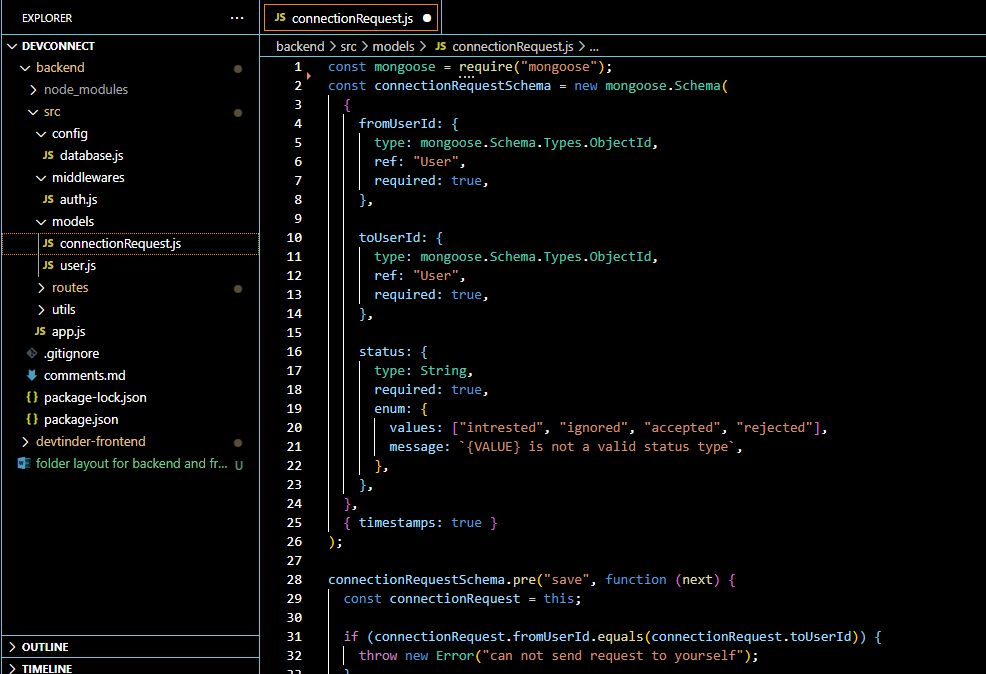
(1):database.js :

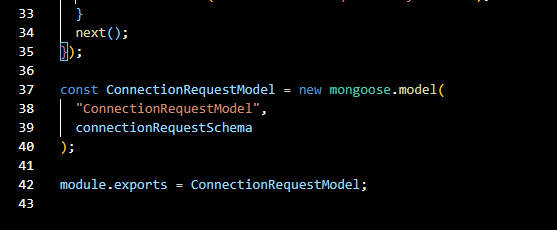


( 2 ) auth.js :



( 3 ) connectionRequest.js :





( 4 ):

Models-->User.js

const jwt = require("jsonwebtoken");

const mongoose = require("mongoose");

var validator = require("validator");

const bcrypt = require("bcrypt");

const userSchema = new mongoose.Schema(

  {

    firstName: {

      type: String,

      required: true,

      minLength: 4,

      maxLength: 10,

    },

    lastName: {

      type: String,

      required: true,

    },

    emailId: {

      type: String,

      required: true,

      unique: true,

      lowercase: true,

      trim: true,

      validate(value) {

        if (!validator.isEmail(value)) {

          throw new Error("invalid email id" + value);

        }

      },

    },

    age: {

      type: Number,

      min: 18,

      max: 60,

    },

    password: {

      type: String,

      required: true,

      validate(value) {

        if (!validator.isStrongPassword(value)) {

          throw new Error(" please Enter a strong password" + value);

        }

      },

    },

    about: {

      type: String,

      default: "This is a default value",

    },

    skills: {

      type: [String],

    },

    photoUrl: {

      type: String,

    },

    gender: {

      type: String,

      validate(value) {

        if (!["male", "female", "others"].includes(value)) {

          throw new Error("gender is not valid");

        }

      },

    },

  },

  {

    timestamps: true,

  }

);

(userSchema.methods.getJWT = async function () {

  const user = this;

  const token = await jwt.sign({ \_id: user.\_id }, "anjali123", {

    expiresIn: "7d",

  });

  return token;

}),

  (userSchema.methods.validatePassword = async function (passwordInputByUser) {

    const user = this;

    const passwordHash = user.password;

    const isPasswordValid = await bcrypt.compare(

      passwordInputByUser,

      passwordHash

    );

    return isPasswordValid;

  });

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

( 5 ):

routes🡪auth.js

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

const { validateSignUpData } = require("../utils/validation");

const bcrypt = require("bcrypt");

const express = require("express");

const authRouter = express.Router();

authRouter.post("/signup", async (req, res) => {

  try {

    //validation of data

    validateSignUpData(req);

    const {

      firstName,

      lastName,

      emailId,

      password,

      age,

      gender,

      about,

      skills,

      photoUrl,

    } = req.body;

    //encrypt the password

    const passwordHash = await bcrypt.hash(password, 10);

    console.log(passwordHash);

    //creating a new instance of User model

    const user = new User({

      firstName,

      lastName,

      emailId,

      age,

      gender,

      about,

      skills,

      photoUrl,

      password: passwordHash,

    });

    const savedUser = await user.save();

    const token = await savedUser.getJWT();

    res.cookie("token", token, {

      expires: new Date(Date.now() + 8 \* 3600000),

    });

    res.json({ message: "user added successfully", data: savedUser });

  } catch (err) {

    res.status(400).send("ERROR:" + err.message);

  }

});

authRouter.post("/login", async (req, res) => {

  try {

    const { emailId, password } = req.body;

    //email id is present or not in DB

    const user = await User.findOne({ emailId: emailId });

    if (!user) {

      throw new Error("invalid credentials");

    }

    const isPasswordValid = await user.validatePassword(password);

    if (isPasswordValid) {

      //create a jwt token

      const token = await user.getJWT();

      //add the token to cookie and send the response back to user

      res.cookie("token", token, {

        expires: new Date(Date.now() + 7 \* 3600000),

      });

      res.send(user);

    } else {

      throw new Error("Invalid credentials");

    }

  } catch (err) {

    res.status(400).send("Error:" + err.message);

  }

});

authRouter.post("/logout", async (req, res) => {

  res.cookie("token", null, {

    expires: new Date(Date.now()),

  });

  res.send("logout successfull");

});

module.exports = authRouter;

( 6 ): profile.js

const { userAuth } = require("../middlewares/auth");

const express = require("express");

const { validateEditProfileData } = require("../utils/validation");

const profileRouter = express.Router();

profileRouter.get("/profile/view", userAuth, async (req, res) => {

  try {

    const user = req.user;

    res.send(user);

  } catch (err) {

    res.status(400).send("Error:" + err.message);

  }

});

profileRouter.patch("/profile/edit", userAuth, async (req, res) => {

  try {

    if (!validateEditProfileData(req)) {

      throw new Error("invalid edit request");

    }

    const loggedInUser = req.user;

    Object.keys(req.body).forEach((key) => {

      loggedInUser[key] = req.body[key];

    });

    await loggedInUser.save();

    res.json({

      message: "edited successfully",

      data: loggedInUser,

    });

  } catch (err) {

    res.status(400).send("Error:" + err.message);

  }

});

profileRouter.patch("/profile/password", userAuth, async (req, res) => {

  try {

  } catch (err) {}

});

module.exports = profileRouter;

( 7 ): request.js

const { userAuth } = require("../middlewares/auth");

const express = require("express");

const connectionRequestModel = require("../models/connectionRequest");

const requestRouter = express.Router();

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

requestRouter.post(

  "/request/send/:status/:toUserId",

  userAuth,

  async (req, res) => {

    try {

      const fromUserId = req.user.\_id;

      const toUserId = req.params.toUserId;

      const status = req.params.status;

      const allowedStatus = ["intrested", "ignored"];

      if (!allowedStatus.includes(status)) {

        res.status(401).send("invalid status type");

      }

      const toUser = await User.findById(toUserId);

      if (!toUser) {

        throw new Error("user not found");

      }

      //created a new instance of connectionRequestModel

      const connectionRequest = new connectionRequestModel({

        fromUserId,

        toUserId,

        status,

      });

      const existingConnection = await connectionRequestModel.findOne({

        $or: [

          { fromUserId, toUserId },

          { fromUserId: toUserId, toUserId: fromUserId },

        ],

      });

      if (existingConnection) {

        throw new Error("connection request alreasy exists");

      }

      const data = await connectionRequest.save();

      res.json({

        message: "connection request sent successfully",

        data,

      });

    } catch (err) {

      res.status(400).send("ERROR: " + err.message);

    }

    res.send(User.firstName + "is sending connection request");

  }

);

requestRouter.post(

  "/request/review/:status/:requestId",

  userAuth,

  async (req, res) => {

    try {

      const loggedInUser = req.user;

      const { status, requestId } = req.params;

      const allowedStatus = ["accepted", "rejected"];

      if (!allowedStatus.includes(status)) {

        throw new Error("status is not valid");

      }

      const connectionRequest = await connectionRequestModel.findOne({

        \_id: requestId,

        toUserId: loggedInUser.\_id,

        status: "intrested",

      });

      if (!connectionRequest) {

        throw new Error("connection request is not present");

      }

      connectionRequest.status = status;

      const data = await connectionRequest.save();

      res.json({

        message: "connection request " + status,

        data,

      });

    } catch (err) {

      res.status(401).send("ERROR: " + err.message);

    }

  }

);

module.exports = requestRouter;

( 8 ): user.js

const express = require("express");

const ConnectionRequestModel = require("../models/connectionRequest");

const userRouter = express.Router();

const { userAuth } = require("../middlewares/auth");

const USER\_SAFE\_DATA = "firstName lastName age gender about skills photoUrl";

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

userRouter.get("/user/request/received", userAuth, async (req, res) => {

  try {

    const loggedInUser = req.user;

    const connectionRequests = await ConnectionRequestModel.find({

      toUserId: loggedInUser.\_id,

      status: "intrested",

    }).populate("fromUserId", USER\_SAFE\_DATA);

    res.json({

      message: "data fatched successfully",

      data: connectionRequests,

    });

  } catch (err) {

    res.status(401).send("ERROR: " + err.message);

  }

});

userRouter.get("/user/connections", userAuth, async (req, res) => {

  try {

    const loggedInUser = req.user;

    const connectionRequests = await ConnectionRequestModel.find({

      $or: [

        { toUserId: loggedInUser.\_id, status: "accepted" },

        { fromUserId: loggedInUser.\_id, status: "accepted" },

      ],

    })

      .populate("fromUserId", USER\_SAFE\_DATA)

      .populate("toUserId", USER\_SAFE\_DATA);

    const data = connectionRequests.map((row) => {

      if (row.fromUserId.\_id.toString() === loggedInUser.\_id.toString()) {

        return row.toUserId;

      }

      return row.fromUserId;

    });

    res.json({ data });

  } catch (err) {

    res.send("ERROR " + err.message);

  }

});

userRouter.get("/feed", userAuth, async (req, res) => {

  try {

    const loggedInUser = req.user;

    const page = parseInt(req.query.page) || 1;

    let limit = parseInt(req.query.limit) || 10;

    limit = limit > 50 ? 50 : limit;

    const skip = (page - 1) \* limit;

    //find all the users who have sent you request or you have sent them request

    const connectionRequests = await ConnectionRequestModel.find({

      $or: [{ fromUserId: loggedInUser.\_id }, { toUserId: loggedInUser.\_id }],

    }).select("fromUserId toUserId");

    const hiddenUsersFromFeed = new Set();

    connectionRequests.forEach((req) => {

      hiddenUsersFromFeed.add(req.fromUserId.toString());

      hiddenUsersFromFeed.add(req.toUserId.toString());

    });

    const users = await User.find({

      $and: [

        { \_id: { $nin: Array.from(hiddenUsersFromFeed) } },

        { \_id: { $ne: loggedInUser.\_id } },

      ],

    })

      .select(USER\_SAFE\_DATA)

      .skip(skip)

      .limit(limit);

    res.send(users);

  } catch (err) {

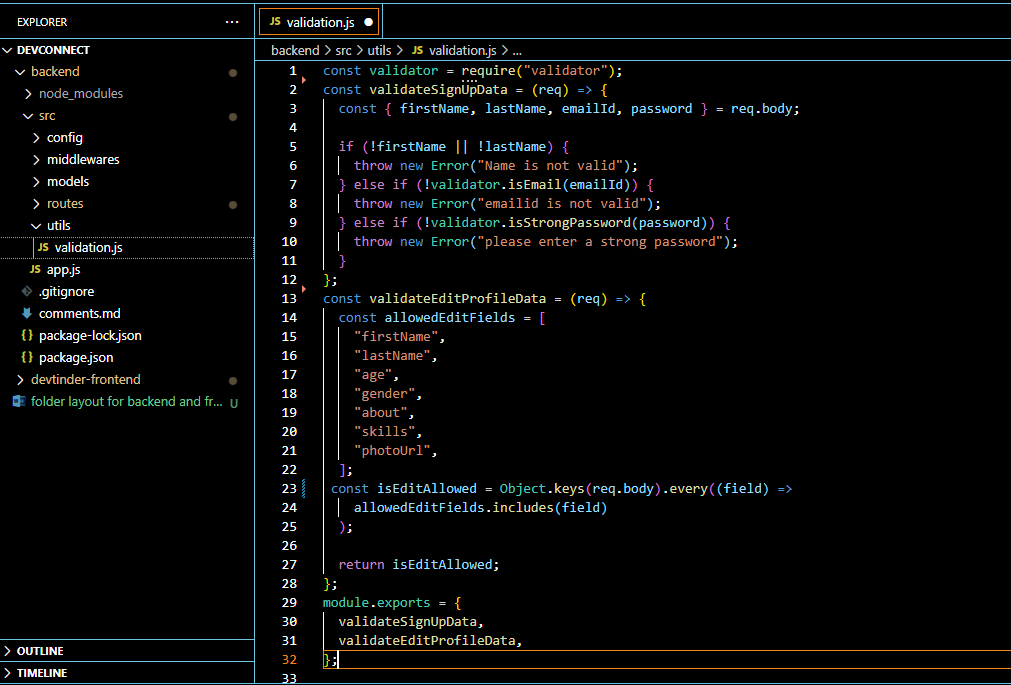
    res.status(401).send("ERROR " + err.message);

  }

});

module.exports = userRouter;

( 9 ): validation.js:



( 10 ): App.js:

const express = require("express");

const connectDB = require("./config/database");

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

const cookieParser = require("cookie-parser");

const cors = require("cors");

const authRouter = require("./routes/auth");

const profileRouter = require("./routes/profile");

const requestRouter = require("./routes/request");

const userRouter = require("./routes/user");

const app = express();

app.use(express.json());

app.use(cookieParser());

app.use(

  cors({

    origin: "http://localhost:5173",

    credentials: true,

  })

);

app.use("/", authRouter);

app.use("/", profileRouter);

app.use("/", requestRouter);

app.use("/", userRouter);

connectDB()

  .then(() => {

    console.log("database connected successfully");

    app.listen(7777, () => {

      console.log("server is connected to port 7777...");

    });

  })

  .catch((err) => {

    console.error("unable to connect" + err.message);

  });