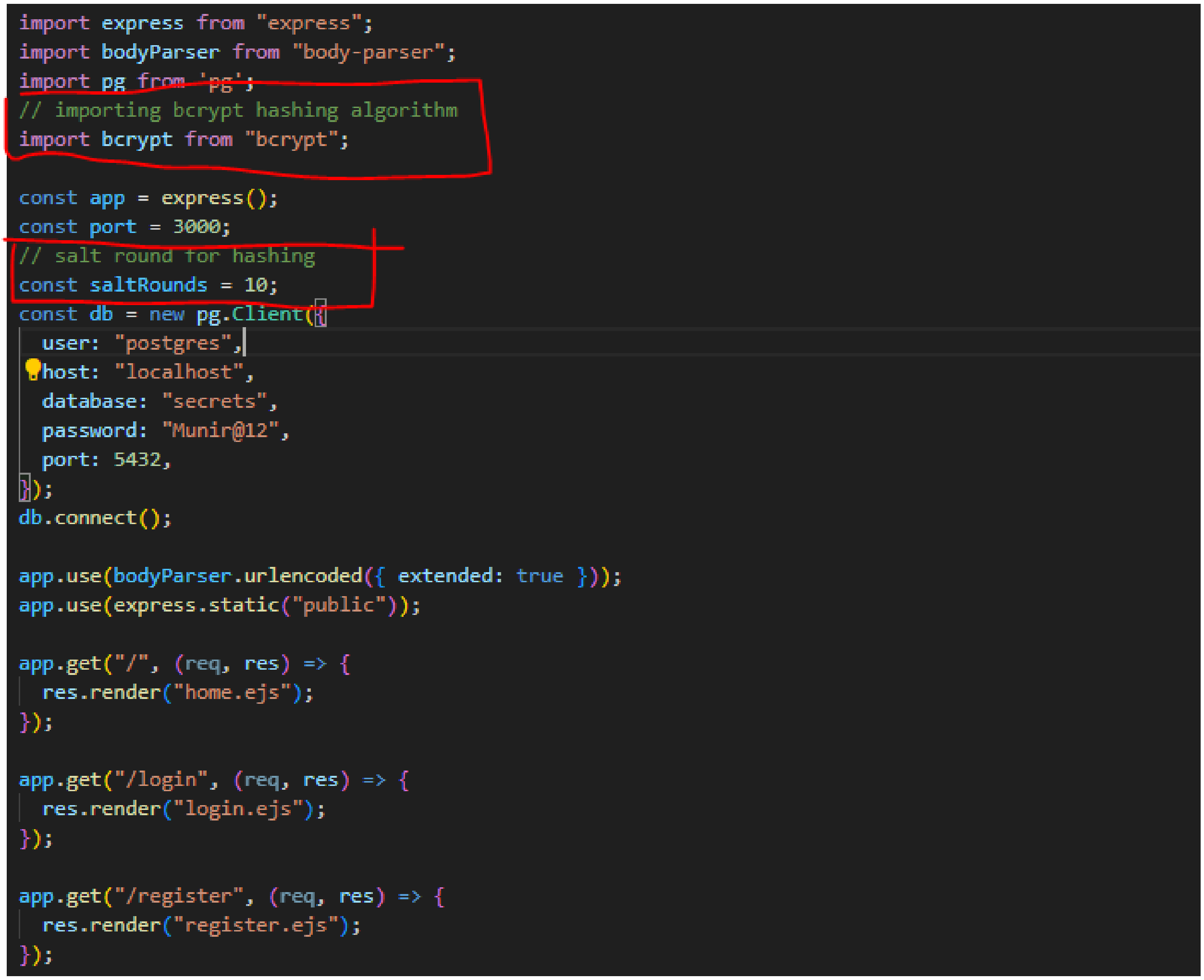
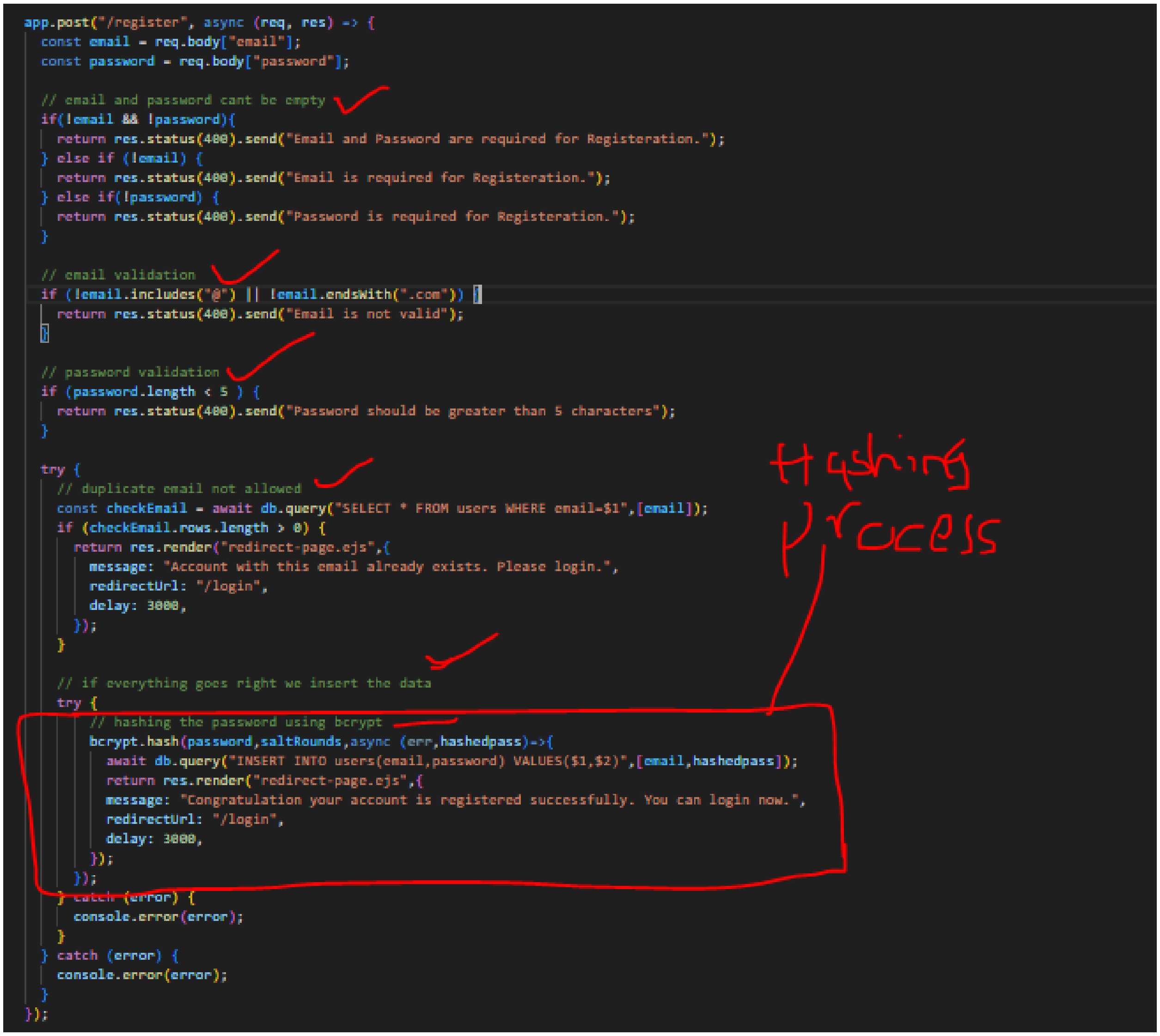
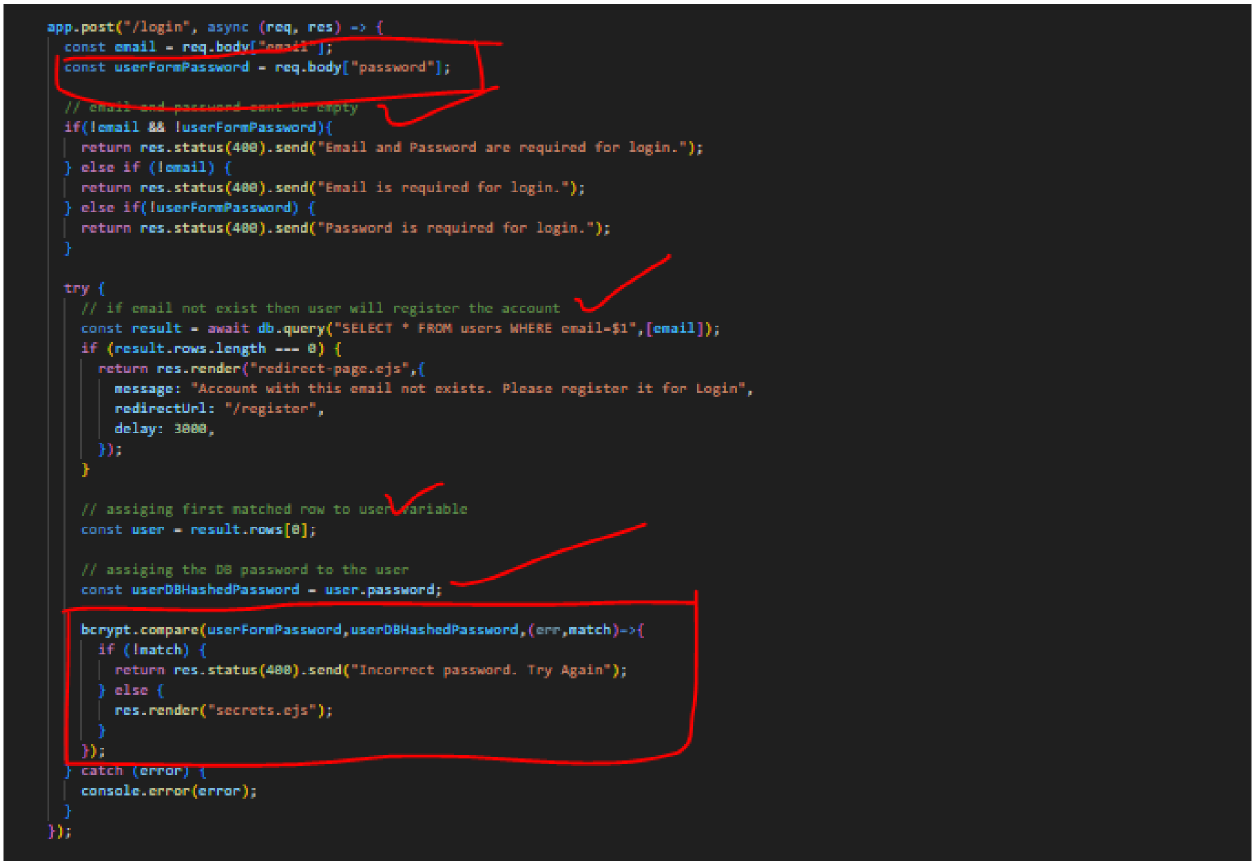
LOGIN and REGISTER with Bcrypt Hashing Algorithm for Password and Email and Password Validations.



```
import express from "express";
import bodyParser from "body-parser";
import pg from 'pg';
// importing bcrypt hashing algorithm
import bcrypt from "bcrypt";

const app = express();
const port = 3000;
// salt round for hashing
const saltRounds = 10;
const db = new pg.Client({
  user: "postgres",
  host: "localhost",
  database: "secrets",
  password: "Munir@12",
  port: 5432,
});
db.connect();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static("public"));

app.get("/", (req, res) => {
  res.render("home.ejs");
});

app.get("/login", (req, res) => {
  res.render("login.ejs");
});

app.get("/register", (req, res) => {
  res.render("register.ejs");
});
```



```
app.post("/register", async (req, res) => {
  const email = req.body["email"];
  const password = req.body["password"];

  // email and password cant be empty
  if(!email && !password){
    return res.status(400).send("Email and Password are required for Registeration.");
  } else if (!email) {
    return res.status(400).send("Email is required for Registeration.");
  } else if(!password) {
    return res.status(400).send("Password is required for Registeration.");
  }

  // email validation
  if (!email.includes("@") || !email.endsWith(".com")) {
    return res.status(400).send("Email is not valid");
  }

  // password validation
  if (password.length < 5 ) {
    return res.status(400).send("Password should be greater than 5 characters");
  }

  try {
    // duplicate email not allowed
    const checkEmail = await db.query("SELECT * FROM users WHERE email=$1",[email]);
    if (checkEmail.rows.length > 0) {
      return res.render("redirect-page.ejs",{
        message: "Account with this email already exists. Please login.",
        redirectUrl: "/login",
        delay: 3000,
      });
    }

    // if everything goes right we insert the data
    try {
        // hashing the password using bcrypt
        bcrypt.hash(password,saltRounds,async (err,hashedpass)=>{
          await db.query("INSERT INTO users(email,password) VALUES($1,$2)",[email,hashedpass]);
          return res.render("redirect-page.ejs",{
          message: "Congratulation your account is registered successfully. You can login now.",
          redirectUrl: "/login",
          delay: 3000,
        });
      });
    } catch (error) {
      console.error(error);
    }
  } catch (error) {
    console.error(error);
  }
});
```



```
app.post("/login", async (req, res) => {
  const email = req.body["email"];
  const userFormPassword = req.body["password"];

  // email and password cant be empty
  if(!email && !userFormPassword){
    return res.status(400).send("Email and Password are required for login.");
  } else if (!email) {
    return res.status(400).send("Email is required for login.");
  } else if(!userFormPassword) {
    return res.status(400).send("Password is required for login.");
  }

  try {
    // if email not exist then user will register the account
    const result = await db.query("SELECT * FROM users WHERE email=$1",[email]);
    if (result.rows.length === 0) {
      return res.render("redirect-page.ejs",{
        message: "Account with this email not exists. Please register it for Login",
        redirectUrl: "/register",
        delay: 3000,
      });
    }

    // assiging first matched row to user variable
    const user = result.rows[0];

    // assiging the DB password to the user
    const userDBHashedPassword = user.password;

    bcrypt.compare(userFormPassword,userDBHashedPassword,(err,match)=>{
      if (!match) {
        return res.status(400).send("Incorrect password. Try Again");
      } else {
        res.render("secrets.ejs");
      }
    });
  } catch (error) {
    console.error(error);
  }
});
```