Backend Development
DA217A
Lab 3
Kassem Alsheikh

Starting with making the lab folder and installing the all needed packages such as: express, cors, bcrypt, dotenv, ejs, jsonwebtoken, nodemon and sqlite3. I made a file for the database with the name database.js where I made some function in order to set up the database with the users table that has an id and username and password and four other functions to be able to make a simple login system. The first function is to register a user with username and password and insert a new user to the database. The second one is to check if the user exists by passing the username to the function. The third is responsible to validate the login by passing the username and the password. The last one is getting the password by passing the name to be able to get the encrypted password that has been stored in the database. Moving to the views I made four views with simple css and elements. The views are: register.ejs, login.ejs, fail.ejs and start.ejs. Dotenv file has been made and has the environment variable: TOKEN with the token key that has been generated manually. Moving to the server.js starting with setting up the server with express and setting the view engine to be able to use the ejs files and route between them. Setting up bcrypt, jwt and dotenv for after use. And importing the function from the database.js. Five routes have been made, three of them are get and two are post. The get routes: '/' redirect to the login route, '/login' renders the login.ejs, '/register' renders register.ejs. The post routes, I started the /register post route. First getting the username and password from the req.body and then encode the password with bcrypt and register the user to the database. After registering the user in the database you will be redirected to the login route to be able to log in. The login post route: we get the username

and the password from the req.body and then we use the function that returns the encoded password from the database to compare it with the given password, we check if the user exists in the database! If not we render fail.ejs if the user exists we compare the password given with the encoded one and we validate the user in login if the password does not match we render fail.ejs otherwise we create a token with jwt by passing the username and the environment variable and we logge it then we render start.ejs. Everything works as it should.