Lab Report SE

CS-165L

Submitted To:

Sir Laeeq

Submitted By:

Usman Asgher **2021-CS-46**

Lab Assignment

University of Engineering and Technology

Lahore

1. Complete Code

1.1 Server

```
const express = require("express");
const cors = require("cors");
const bodyparser = require("body-parser");
const port = 8080;
require("./utils/db");
const userroute = require("./routes/UserRoutes");
const app = express();
app.use(bodyparser.json());
app.use("/api", userroute);
app.use(cors());
app.listen(port, () => {
  console.log(`Port is host at local host ${port}`);
});
```

2. Models

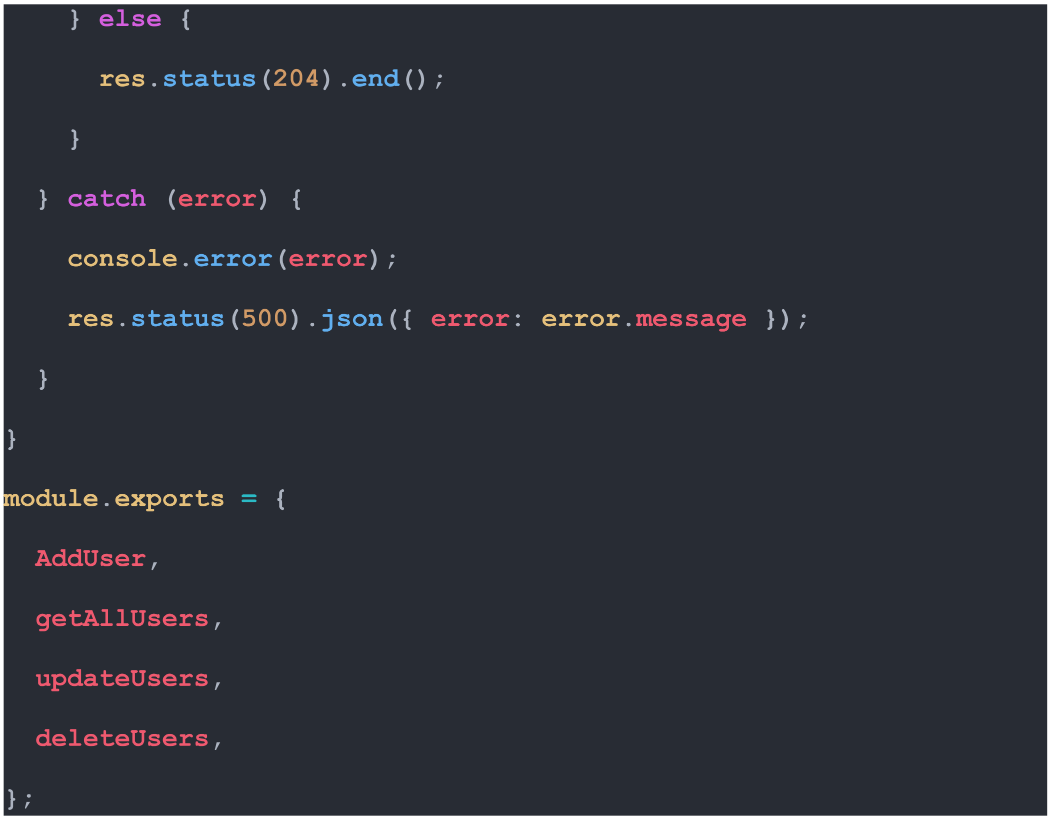
2.1 UserModel

```
const mongoose = require("mongoose");
const UserSchema = new mongoose.Schema(
  {
    FirstName: String,
    LastName: String,
    UserName: {
      type: String,
      unique: true,
    },
    Email: {
      type: String,
      unique: true,
    },
    Password: {
      type: String,
      unique: true,
    },
    Role: String,
    DateOfBirth: Date,
  },
  { timestamps: true }
);
module.exports = mongoose.model("Users", UserSchema);
```

3. Controllers

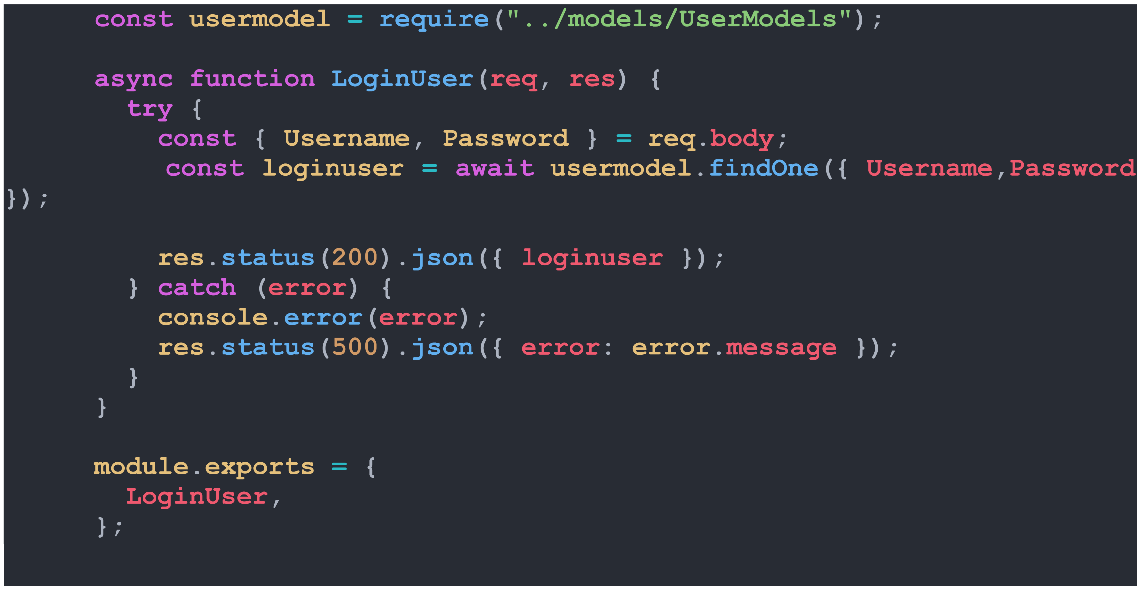
3.1 UserController

```
const usermodel = require("../models/UserModels");
async function AddUser(req, res) {
  try {
    const Users = await usermodel.create(req.body);
    res.status(200).json({ Users });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: error.message });
  }
}
async function getAllUsers(req, res) {
  try {
    const Users = await usermodel.find();
    res.status(200).json({ Users });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: error.message });
  }
}
async function updateUsers(req, res) {
  try {
    const userId = req.params.id;
     const updateUser = await usermodel.findByIdAndUpdate(userId,
req.body, {
      new: true,
    });
    if (!updateUser) {
      return res.status(404).json({ error: "User Not Found" });
    } else {
      res.status(200).json({ updateUser });
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: error.message });
  }
}
async function deleteUsers(req, res) {
  try {
    const userId = req.params.id;
                          const       deleteUsers       =       await
usermodel.findByIdAndDelete(userId);
    if (!deleteUsers) {
      return res.status(404).json({ error: "User Not Deleted" });
```



```
    } else {

      res.status(204).end();

    }

  } catch (error) {

    console.error(error);

    res.status(500).json({ error: error.message });

  }

}

module.exports = {

  AddUser,

  getAllUsers,

  updateUsers,

  deleteUsers,

};
```

3.2 Login Controller



```
      const usermodel = require("../models/UserModels");

      async function LoginUser(req, res) {
        try {
          const { Username, Password } = req.body;
           const loginuser = await usermodel.findOne({ Username,Password
});

          res.status(200).json({ loginuser });
        } catch (error) {
          console.error(error);
          res.status(500).json({ error: error.message });
        }
      }

      module.exports = {
        LoginUser,
      };
```

4. Routes

4.1 UserRoutes

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

const route = express.Router();
const usercontroller = require("../controller/UserController");
const logincontroller = require("../controller/LoginController");

route.post("/users", usercontroller.AddUser);
route.get("/users", usercontroller.getAllUsers);
route.put("/users/:id", usercontroller.updateUsers);
route.delete("/users/:id", usercontroller.deleteUsers);
route.post("/login", logincontroller.LoginUser);

module.exports = route;
```

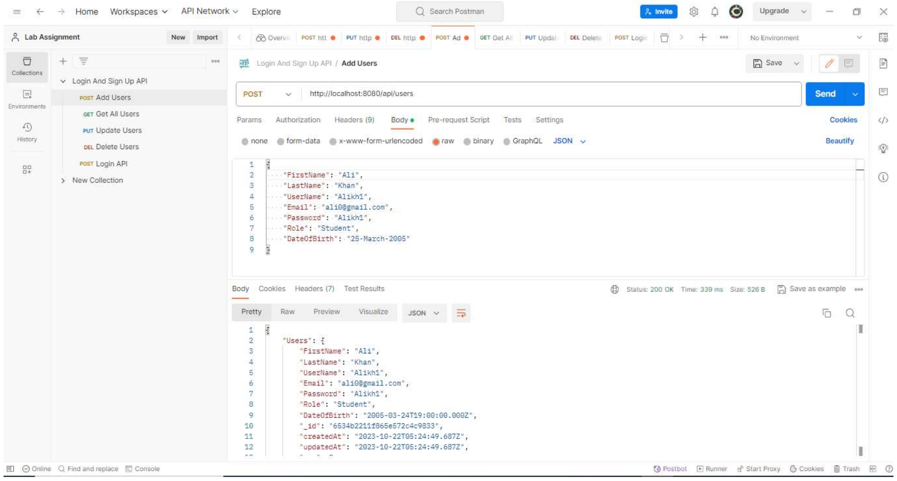
5. Utils

5.1 db

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

mongoose.set("strictQuery", true);

mongoose.connect("mongodb://127.0.0.1:27017/User-APIS", {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

const db = mongoose.connection;

db.on("error", () => {
  console.log("Failed to Connect MongoDb");
});

db.once("open", () => {
  console.log("Connected To MongoDB");
});
```

6. Testing

6.1 Post API



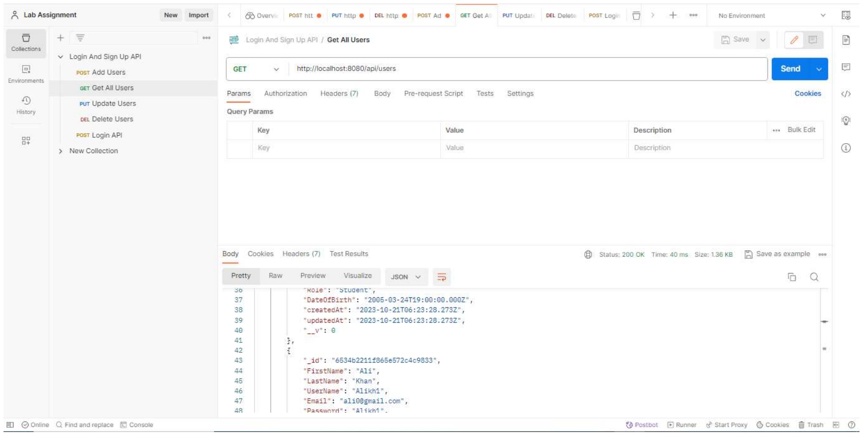
POST http://localhost:8080/api/users

```
{
    "FirstName": "Ali",
    "LastName": "Khan",
    "UserName": "Alikh1",
    "Email": "ali0@gmail.com",
    "Password": "Alikh1",
    "Role": "Student",
    "DateOfBirth": "25-March-2005"
}
```

Status: 200 OK Time: 339 ms Size: 526 B

```
{
    "Users": {
        "FirstName": "Ali",
        "LastName": "Khan",
        "UserName": "Alikh1",
        "Email": "ali0@gmail.com",
        "Password": "Alikh1",
        "Role": "Student",
        "DateOfBirth": "2005-03-24T19:00:00.000Z",
        "_id": "6534b2211f865e572c4c9833",
        "createdAt": "2023-10-22T05:24:49.687Z",
        "updatedAt": "2023-10-22T05:24:49.687Z",
```

6.2 Get API



Login And Sign Up API / Get All Users

GET http://localhost:8080/api/users

Params · Authorization · Headers (7) · Body · Pre-request Script · Tests · Settings

Query Params

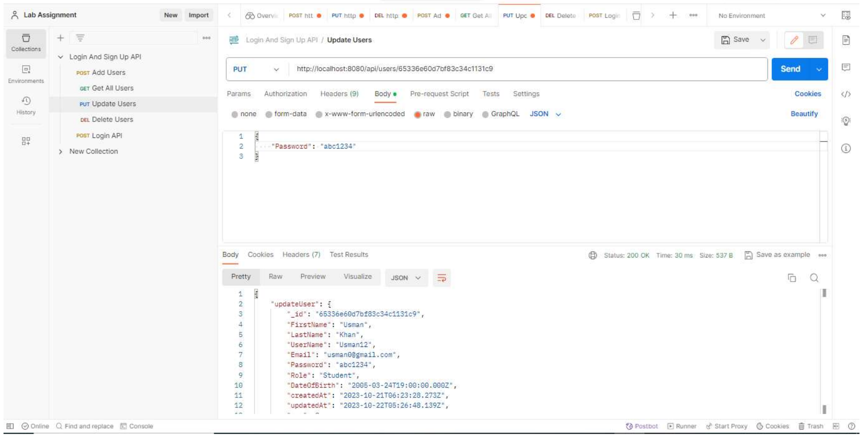
Key	Value	Description
Key	Value	Description

Body · Cookies · Headers (7) · Test Results — Status: 200 OK Time: 40 ms Size: 1.36 KB

Pretty · Raw · Preview · Visualize · JSON

```
            "Role": "Student",
            "DateOfBirth": "2005-03-24T19:00:00.000Z",
            "createdAt": "2023-10-21T06:23:28.273Z",
            "updatedAt": "2023-10-21T06:23:28.273Z",
            "__v": 0
        },
        {
            "_id": "6534b2211f865e572c4c9833",
            "FirstName": "Ali",
            "LastName": "Khan",
            "UserName": "Alikh1",
            "Email": "ali0@gmail.com",
            "Password": "Alikh1",
```

6.3 Put API



Login And Sign Up API / Update Users

PUT http://localhost:8080/api/users/65336e60d7bf83c34c1131c9

Params · Authorization · Headers (9) · Body ● · Pre-request Script · Tests · Settings

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

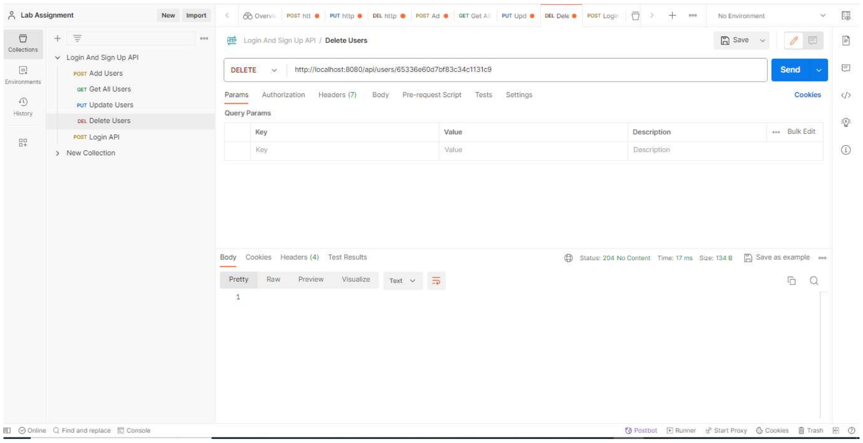
```
{
    "Password": "abc1234"
}
```

Body · Cookies · Headers (7) · Test Results — Status: 200 OK Time: 30 ms Size: 537 B

Pretty · Raw · Preview · Visualize · JSON

```
{
    "updateUser": {
        "_id": "65336e60d7bf83c34c1131c9",
        "FirstName": "Usman",
        "LastName": "Khan",
        "UserName": "Usman12",
        "Email": "usman0@gmail.com",
        "Password": "abc1234",
        "Role": "Student",
        "DateOfBirth": "2005-03-24T19:00:00.000Z",
        "createdAt": "2023-10-21T06:23:28.273Z",
        "updatedAt": "2023-10-22T05:26:48.139Z",
```

6.4 Delete API



Login And Sign Up API / Delete Users

DELETE http://localhost:8080/api/users/65336e60d7bf83c34c1131c9

Params · Authorization · Headers (7) · Body · Pre-request Script · Tests · Settings

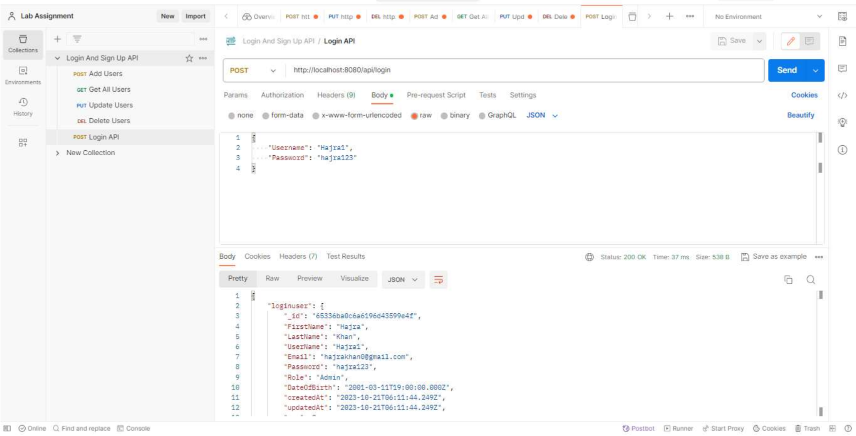
Query Params

Key	Value	Description
Key	Value	Description

Body · Cookies · Headers (4) · Test Results — Status: 204 No Content Time: 17 ms Size: 134 B

Pretty · Raw · Preview · Visualize · Text

6.5 Login API



POST http://localhost:8080/api/login

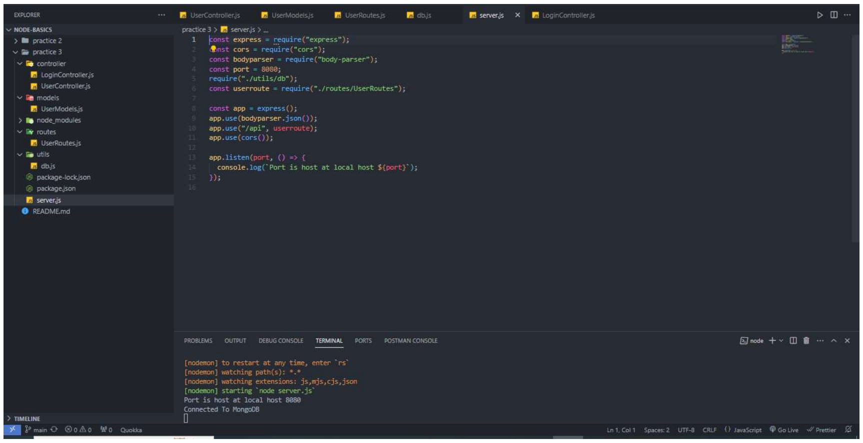
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "Username": "Hajra1",
    "Password": "hajra123"
}
```

Status: 200 OK Time: 37 ms Size: 538 B

```
{
    "loginuser": {
        "_id": "65336ba0c6a6196d43599e4f",
        "FirstName": "Hajra",
        "LastName": "Khan",
        "UserName": "Hajra1",
        "Email": "hajrakhan0@gmail.com",
        "Password": "hajra123",
        "Role": "Admin",
        "DateOfBirth": "2001-03-11T19:00:00.000Z",
        "createdAt": "2023-10-21T06:11:44.249Z",
        "updatedAt": "2023-10-21T06:11:44.249Z",
```

7. File Structure



```
NODE-BASICS
  > practice 2
  v practice 3
    v controller
        LoginController.js
        UserController.js
    v models
        UserModels.js
    > node_modules
    v routes
        UserRoutes.js
    v utils
        db.js
      package-lock.json
      package.json
      server.js
    README.md
```

practice 3 > server.js > ...

```
const express = require("express");
const cors = require("cors");
const bodyparser = require("body-parser");
const port = 8080;
require("./utils/db");
const userroute = require("./routes/UserRoutes");

const app = express();
app.use(bodyparser.json());
app.use("/api", userroute);
app.use(cors());

app.listen(port, () => {
  console.log(`Port is host at local host ${port}`);
});
```

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Port is host at local host 8080
Connected To MongoDB
```