

Assignment 3B

Index: server.js

```
const express =
require("express");
const bodyParser = require("body-
parser");
const app = express();
app.use(bodyParser.urlencoded({ e
xtended: true }));

app.use(bodyParser.json());
const UserRoute =
require("../routes/User");
app.use("/user", UserRoute);

const dbConfig =
require("../config/database.config
.js");
const mongoose =
require("mongoose");
mongoose.Promise =
global.Promise;
mongoose
  .connect(dbConfig.url, {
    useNewUrlParser: true,
  })
  .then(() => {
    console.log("Database
Connected Successfully!!");
  })
  .catch((err) => {
    console.log("Could not
connect to the database", err);
    process.exit();
  });
app.get("/", (req, res) => {
  res.json({ message: "Hello Crud
Node Express" });
});
app.listen(3000, () => {
  console.log("Server is
listening on port 3000");
});
```

Routes: User.js

```
const express =
require("express");
const UserController =
require("../controllers/User");
const router = express.Router();

router.get("/",
UserController.findAll);
router.get("/:id",
UserController.findOne);
router.post("/",
UserController.create);
router.patch("/:id",
UserController.update);
router.delete("/:id",
UserController.destroy);
module.exports = router;
```

Model: User.js

```
var mongoose =
require("mongoose");
var schema = new
mongoose.Schema({
  email: { type: String,
required: true, unique: true },
  firstName: { type: String,
default: "" },
  lastName: { type: String,
default: "" },
  phone: String,
});
var user = new
mongoose.model("User", schema);
module.exports = user;
```

Output:

Create User:

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:3000/user/`
- Method:** `POST`
- Body:** `x-www-form-urlencoded`
- Parameters:**

Key	Value
email	abc@gmail.com
firstName	ABC
lastName	XYZ
phone	85586966
- Response:**

```
1 {
2   "message": "User created successfully!!",
3   "user": {
4     "email": "abc@gmail.com",
5     "firstName": "ABC",
6     "lastName": "XYZ",
7     "phone": "85586966",
8     "_id": "6434eb2184665bc514a6dbf4",
9     "__v": 0
10  }
11 }
```

Display Single User:

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:3000/user/6434eb2184665bc514a6dbf4`
- Method:** `GET`
- Response:**

```
1 {
2   "_id": "6434eb2184665bc514a6dbf4",
3   "email": "abc@gmail.com",
4   "firstName": "ABC",
5   "lastName": "XYZ",
6   "phone": "85586966",
7   "__v": 0
8 }
```

Update User:

http://localhost:3000/user/6434eb2184665bc514a6dbf4

PATCH http://localhost:3000/user/6434eb2184665bc514a6dbf4

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

	Key	Value
<input checked="" type="checkbox"/>	phone	88955
	Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1
2  "message": "User updated successfully."
3
```

Delete User:

http://localhost:3000/user/6434eb2184665bc514a6dbf4

DELETE http://localhost:3000/user/6434eb2184665bc514a6dbf4

Params Authorization Headers (6) **Body** Pre-request Script Tests

Query Params

	Key
	Key

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1
2  "message": "User deleted successfully!"
3
```