

Assignment 3B

Name: Atharva Agrawal

RollNo: 33303

Batch: K-11

Index: server.js

```
const express = require("express");
const bodyParser = require("body-parser");

const app = express();
app.use(bodyParser.urlencoded({ extended: 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);
```

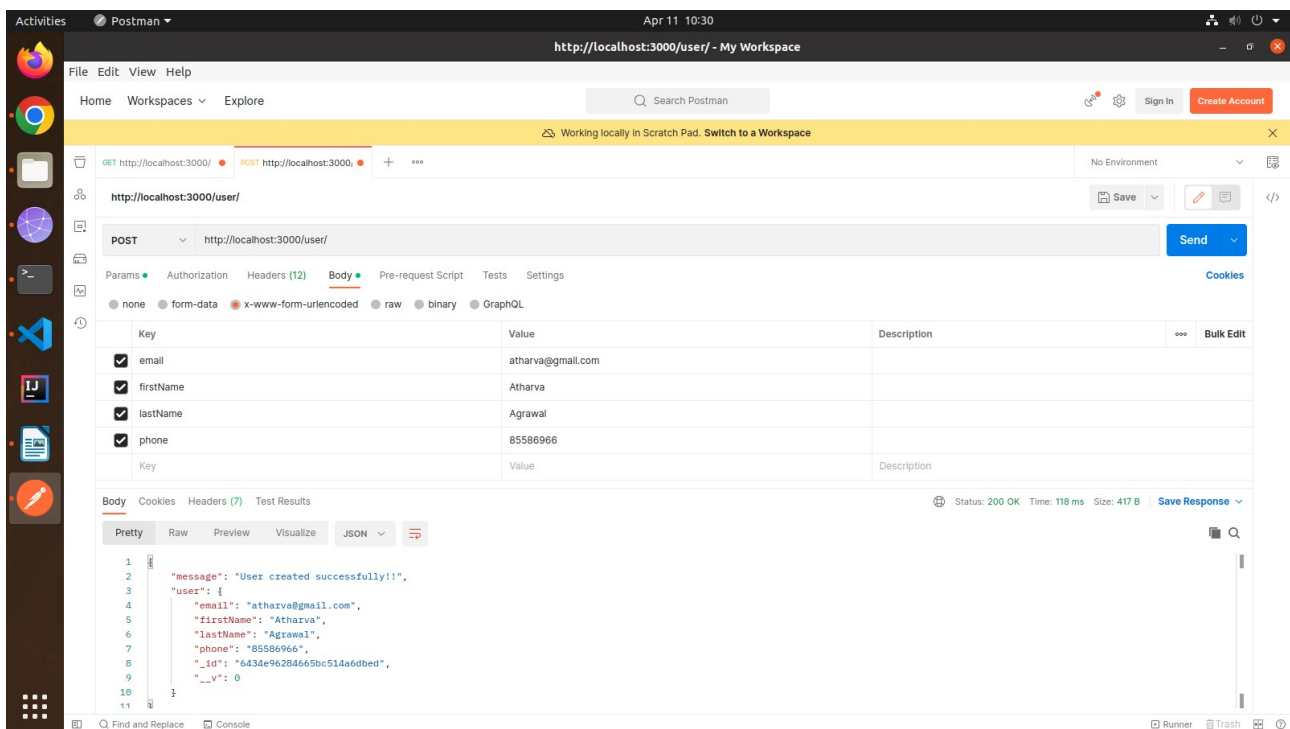
```
module.exports = router;
```

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:



Display Single User:

The screenshot shows the Postman application interface. The URL bar displays `http://localhost:3000/user/6434e96284665bc514a6dbed - My Workspace`. The request method is set to **GET**, and the URL is `http://localhost:3000/user/6434e96284665bc514a6dbed`. The response status is **200 OK** with a time of **15 ms** and a size of **368 B**. The response body is displayed in JSON format:

```
{
  "_id": "6434e96284665bc514a6dbed",
  "email": "atharva@gmail.com",
  "firstName": "Atharva",
  "lastName": "Agrawal",
  "phone": "85586966",
  "___v": 0
}
```

Update User:

The screenshot shows the Postman application interface. The URL bar displays `http://localhost:3000/user/6434e96284665bc514a6dbed - My Workspace`. The request method is set to **PATCH**, and the URL is `http://localhost:3000/user/6434e96284665bc514a6dbed`. The request body is set to **x-www-form-urlencoded** with the following data:

Key	Value	Description
phone	888958	

The response status is **200 OK** with a time of **36 ms** and a size of **275 B**. The response body is displayed in JSON format:

```
{
  "message": "User updated successfully."
}
```

Display All User:

The screenshot shows the Postman interface with a GET request to `http://localhost:3000/user/`. The response is a 200 OK status with a JSON body containing user information.

Key	Value	Description
Key	Value	Description

```
1 {
2   "id": "6434e96284665bc514a6dbed",
3   "email": "atharva@gmail.com",
4   "firstName": "Atharva",
5   "lastName": "Agrawal",
6   "phone": "8889958",
7   "v": 0
8 }
```

Delete User:

The screenshot shows the Postman interface with a DELETE request to `http://localhost:3000/user/6434e96284665bc514a6dbed`. The response is a 200 OK status with a JSON body containing a success message.

Key	Value	Description
Key	Value	Description

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