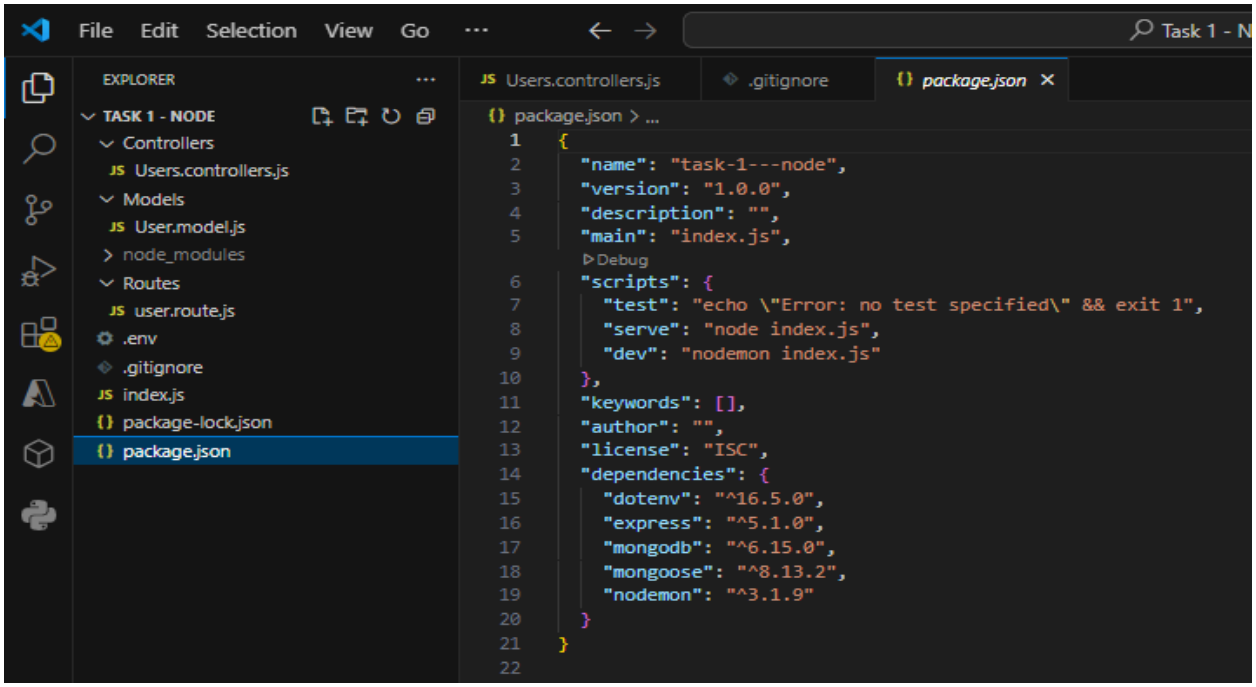
NODE JS - TASK 1

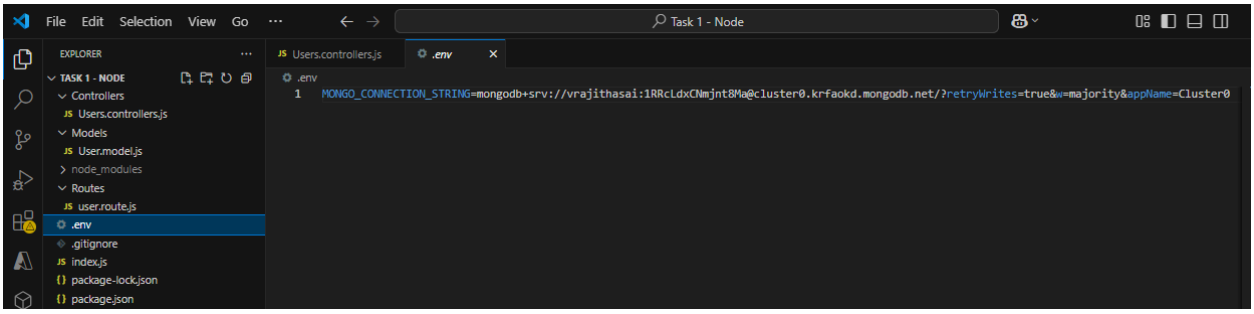
To build Node.js CRUD Application, Testing using Postman and Pushing code to the Github Repository.

<u>To build Node.js CRUD Application - USER DETAILS APPLICATION</u>

- Initially downloading the necessary packages and checking in package.json if the dependencies are updated correctly.



- Mongodb Connectivity [.env is used to store the string that was taken from mongodb]



- The whole connectivity could be seen in index.js
  index.js contains the details of,
  Routes (API, Test)
  To start the server,
  Mongo db connectivity

```node_modules/index.js
const express = require('express');
const app = express();
const userRoute = require('./Routes/user.route.js');
const mongoose = require('mongoose');
const dotenv = require('dotenv');

dotenv.config();

// middleware to convert the json
app.use(express.json());

// API routes
app.use("/api/users", userRoute);

// Test route
app.get('/', (req, res) => {
    res.send('This is the response from a Node API which is updated');
});

// Start server
app.listen(3000, () => {
    console.log('Server is running on port 3000');
});

// MongoDB connection
mongoose
  .connect(process.env.MONGO_CONNECTION_STRING)
  .then(() => {
    console.log('Connected to MongoDB!');
  })
  .catch((err) => {
    console.error('Connection Failed!', err);
  });
```

- Creating Schema [USER SCHEMA - name, age, contact_number, working]

```Models/User.model.js
const mongoose = require('mongoose');

const UserSchema = mongoose.Schema(
    {
        name: {
            type: String,
            required: [true, "Please enter your name"]
        },
        age: {
            type: Number,
            required: true,
            default: 0
        },
        contact_number: {
            type: Number,
            required: true,
            default: 0
        },
        working: {
            type: String,
            required: true
        }
    },
    {
        timestamps: true
    }
);

const User = mongoose.model("Users", UserSchema);

module.exports = User;
```

- Put all the APIs [GET, POST, PUT, DEL]
  The details has to be put in the controllers which means it contains functioning or working of each API.

File Edit Selection View Go ··· Task 1 - Node

EXPLORER

TASK 1 - NODE
- Controllers
  - Users.controllers.js
- Models
- node_modules
- Routes
- .env
- .gitignore
- index.js
- package-lock.json
- package.json

OUTLINE
TIMELINE
NPM SCRIPTS

Users.controllers.js

Controllers > Users.controllers.js > ...

```javascript
// Update a user
const updatedUser = async (req, res) => {
    try {
        const { id } = req.params;
        const user = await User.findByIdAndUpdate(id, req.body, { new: true });

        if (!user) {
            return res.status(404).json({ message: "User not found" });
        }

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

// Delete a user
const deleteUser = async (req, res) => {
    try {
        const { id } = req.params;
        const user = await User.findByIdAndDelete(id);

        if (!user) {
            return res.status(404).json({ message: "User not found" });
        }

        res.status(200).json({ message: "User Deleted Successfully" });
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
};

module.exports = {
    getUsers,
    getUser,
    createUser,
    updatedUser,
    deleteUser
};
```

File Edit Selection View Go ··· Task 1 - Node

EXPLORER

TASK 1 - NODE
- Controllers
  - Users.controllers.js
- Models
- node_modules
- Routes
- .env
- .gitignore
- index.js
- package-lock.json
- package.json

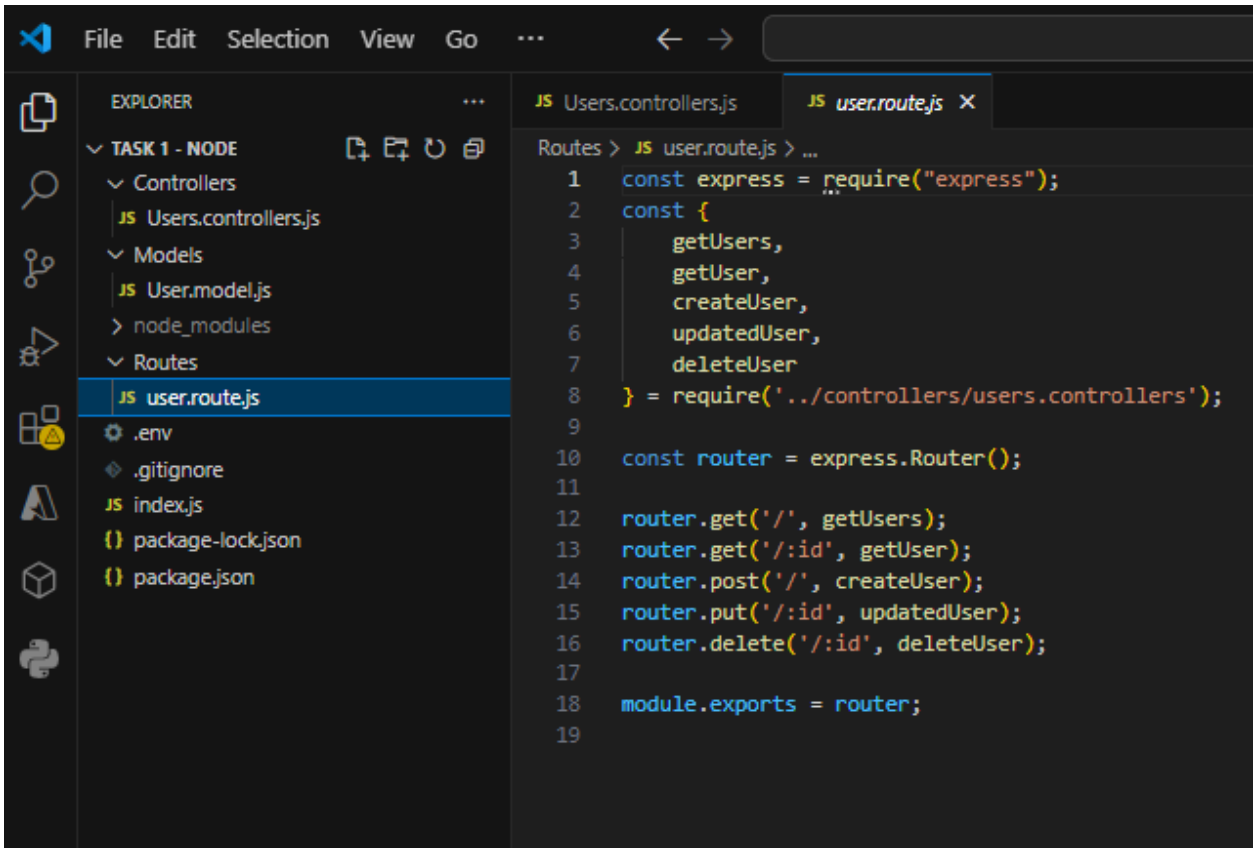
OUTLINE
TIMELINE
NPM SCRIPTS

Users.controllers.js

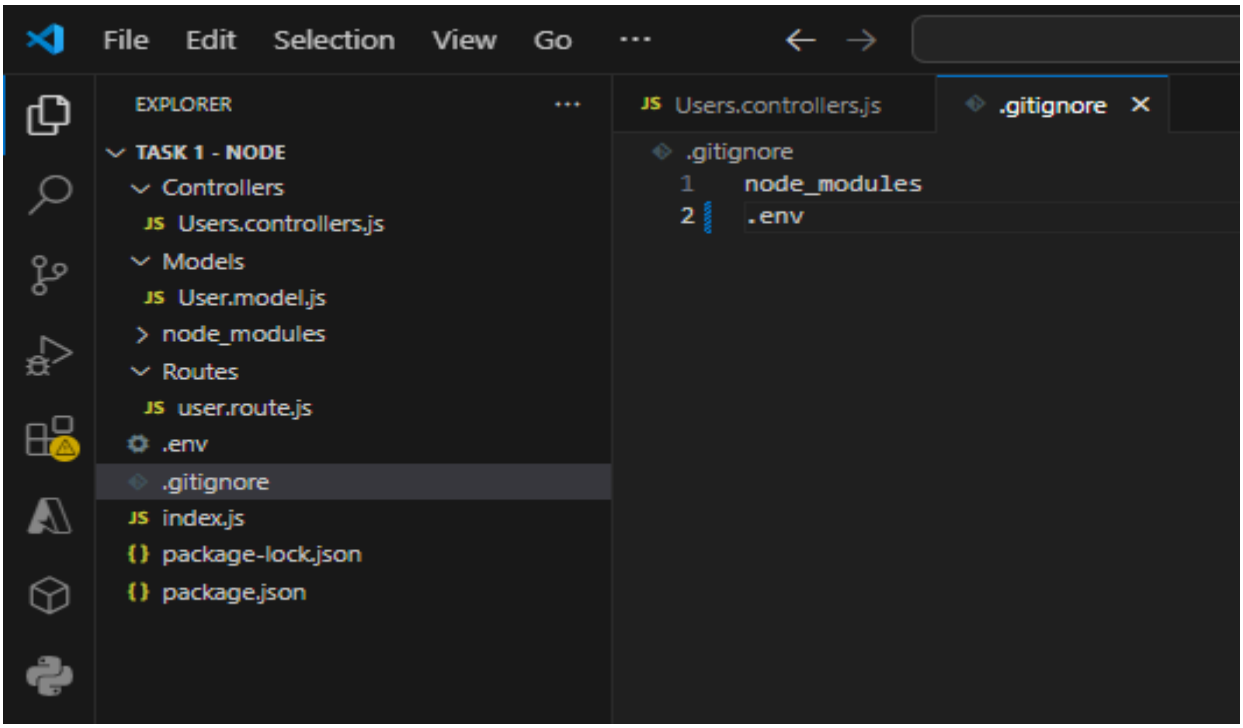
Controllers > Users.controllers.js > ...

```javascript
// Update a user
const updatedUser = async (req, res) => {
    try {
        const { id } = req.params;
        const user = await User.findByIdAndUpdate(id, req.body, { new: true });

        if (!user) {
            return res.status(404).json({ message: "User not found" });
        }

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

// Delete a user
const deleteUser = async (req, res) => {
    try {
        const { id } = req.params;
        const user = await User.findByIdAndDelete(id);

        if (!user) {
            return res.status(404).json({ message: "User not found" });
        }

        res.status(200).json({ message: "User Deleted Successfully" });
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
};

module.exports = {
    getUsers,
    getUser,
    createUser,
    updatedUser,
    deleteUser
};
```

- And now connect all files through Routes



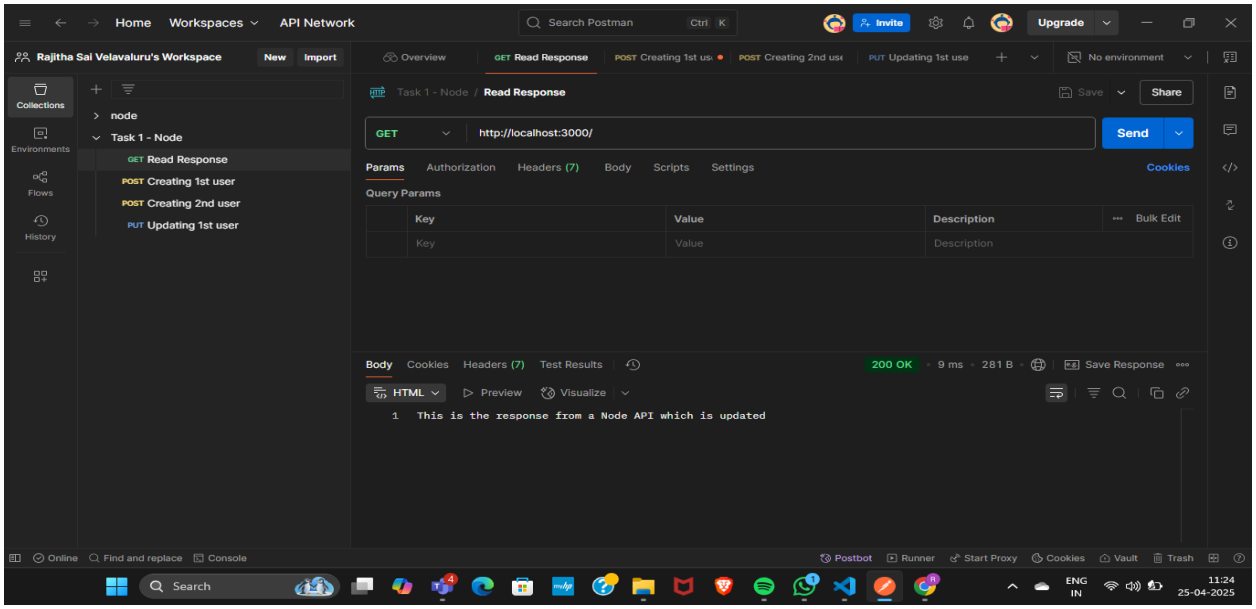
- .gitignore file is used - if any files are mentioned in this file, that particular file will not be pushed to github [node_modules - contains more files, .env - contains string which is used for connectivity that has username and password]



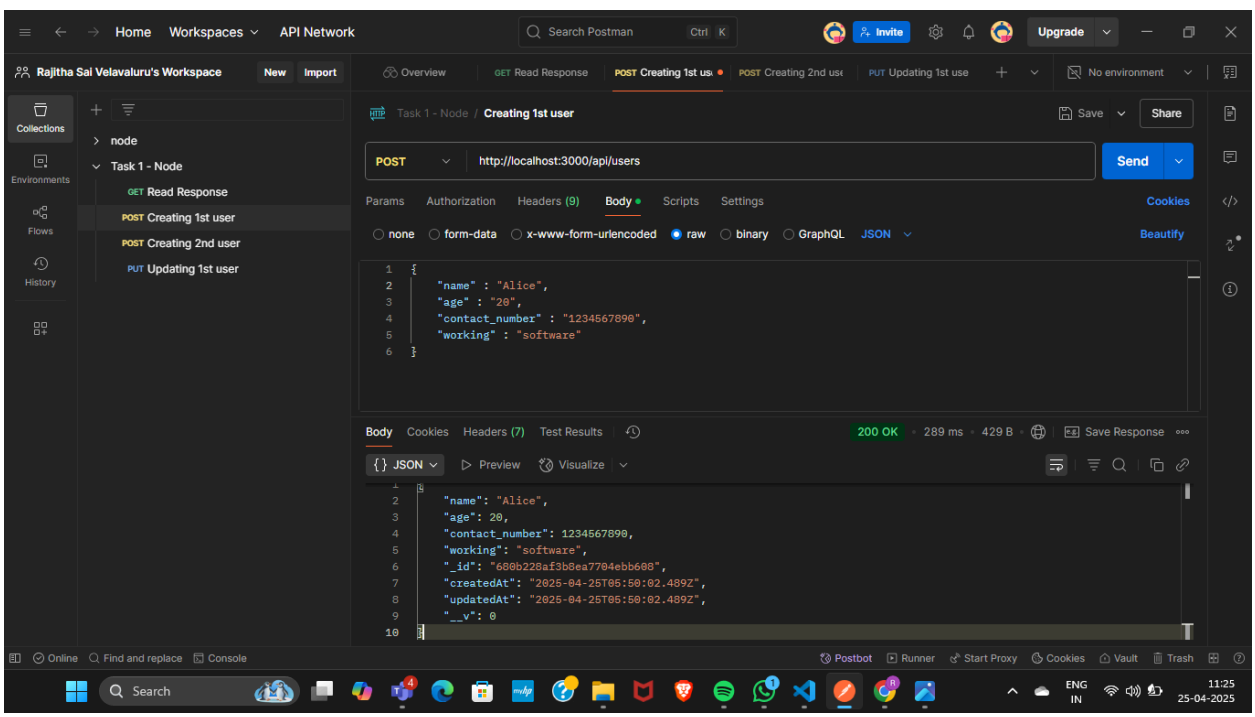
Testing APIs using POSTMAN

Testing includes Creating, Reading, Updating and Deleting of the user

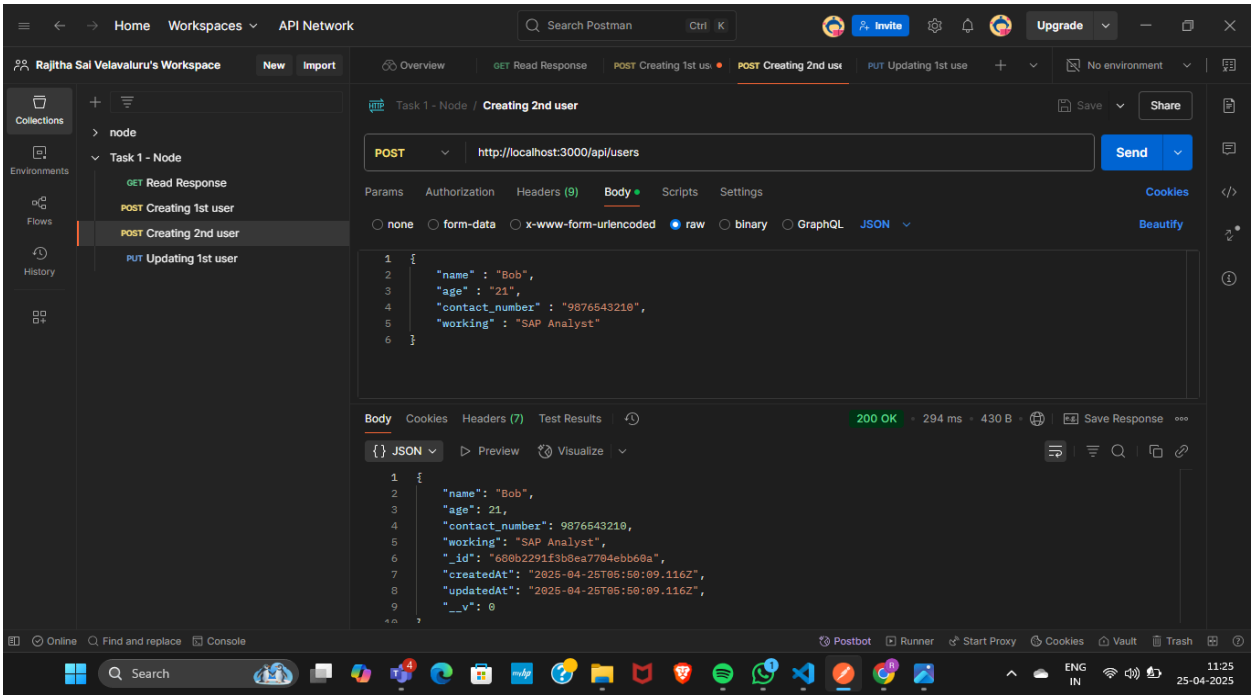
- To check with the response GET method is used



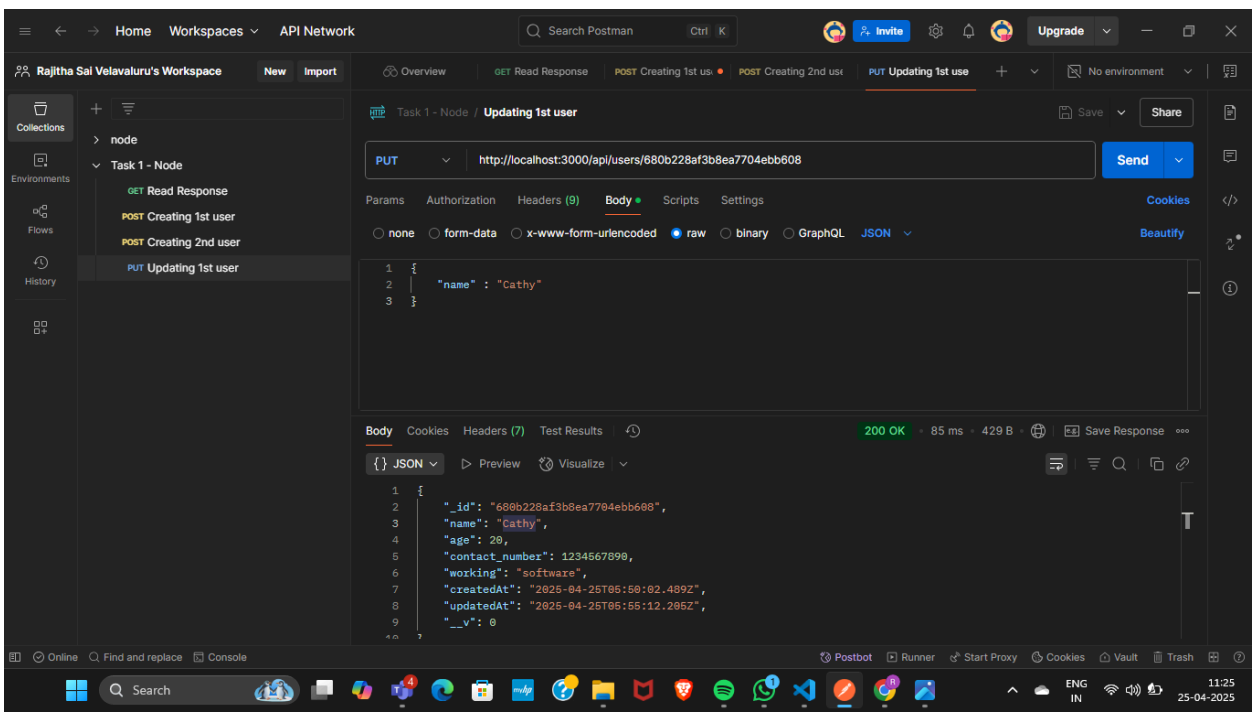
- To post the user details POST method is used [creating 2 users]
  1st User - Alice



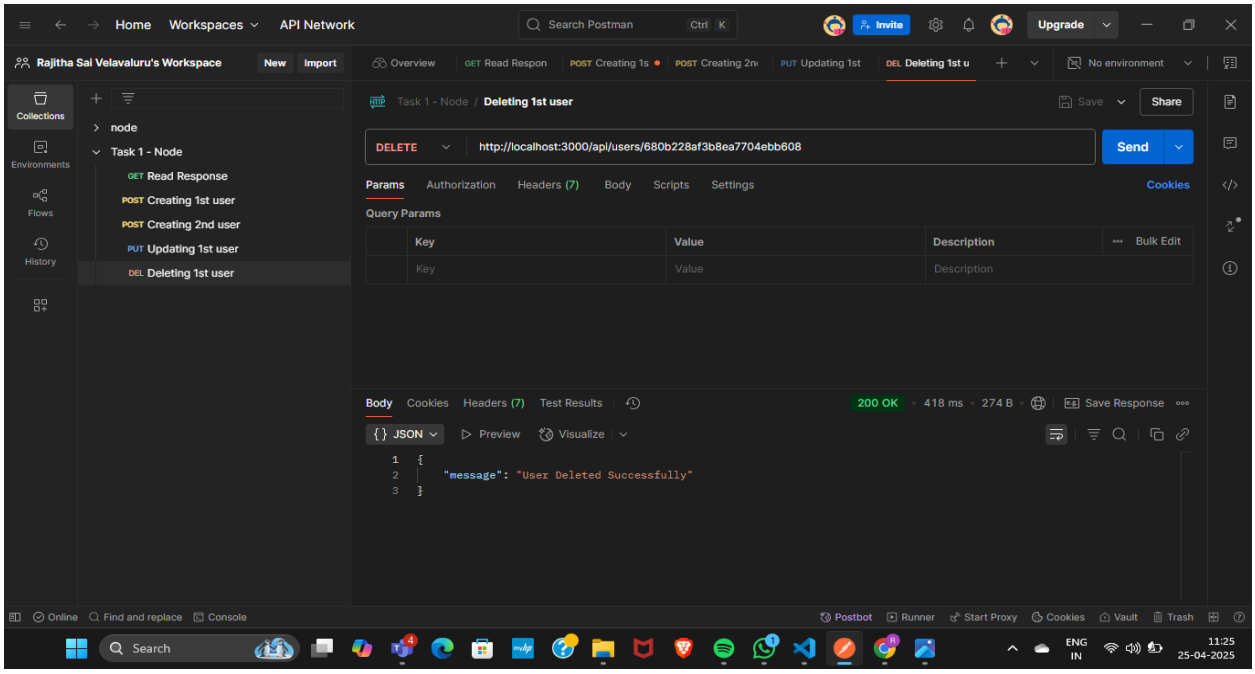
2nd User - Bob



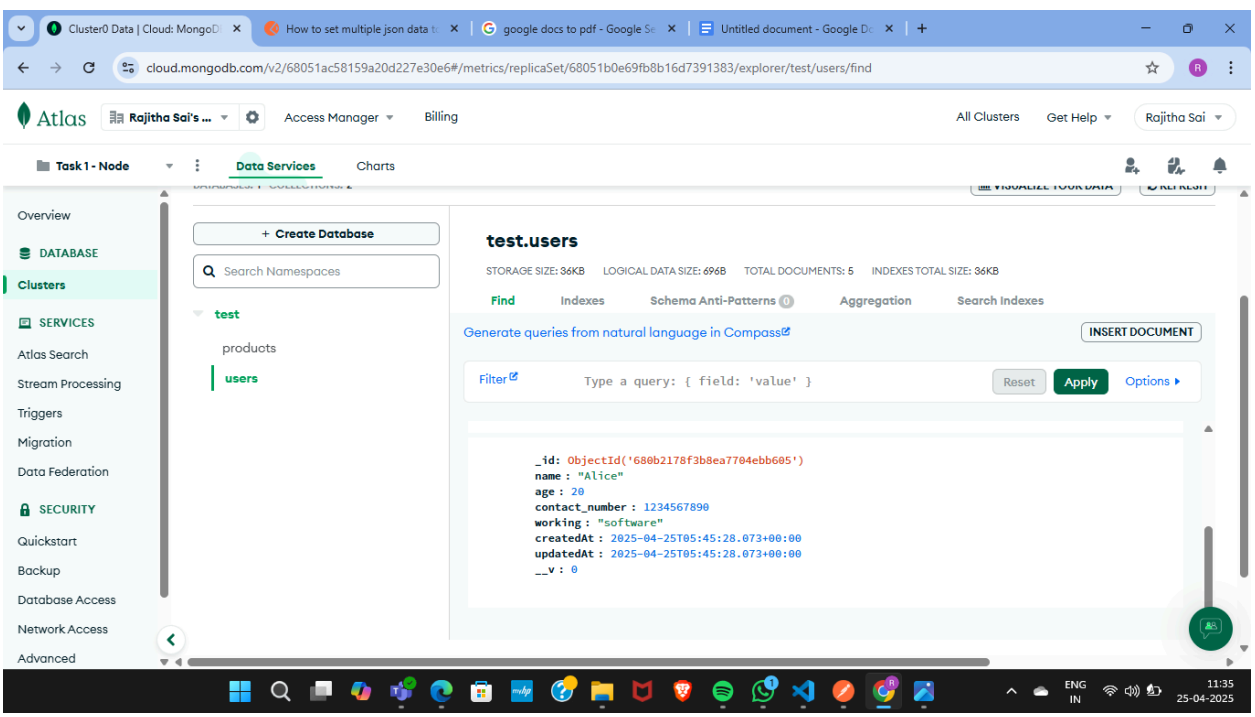
- To update the user details PUT method is user [1st user's name is updated as Cathy using the id]

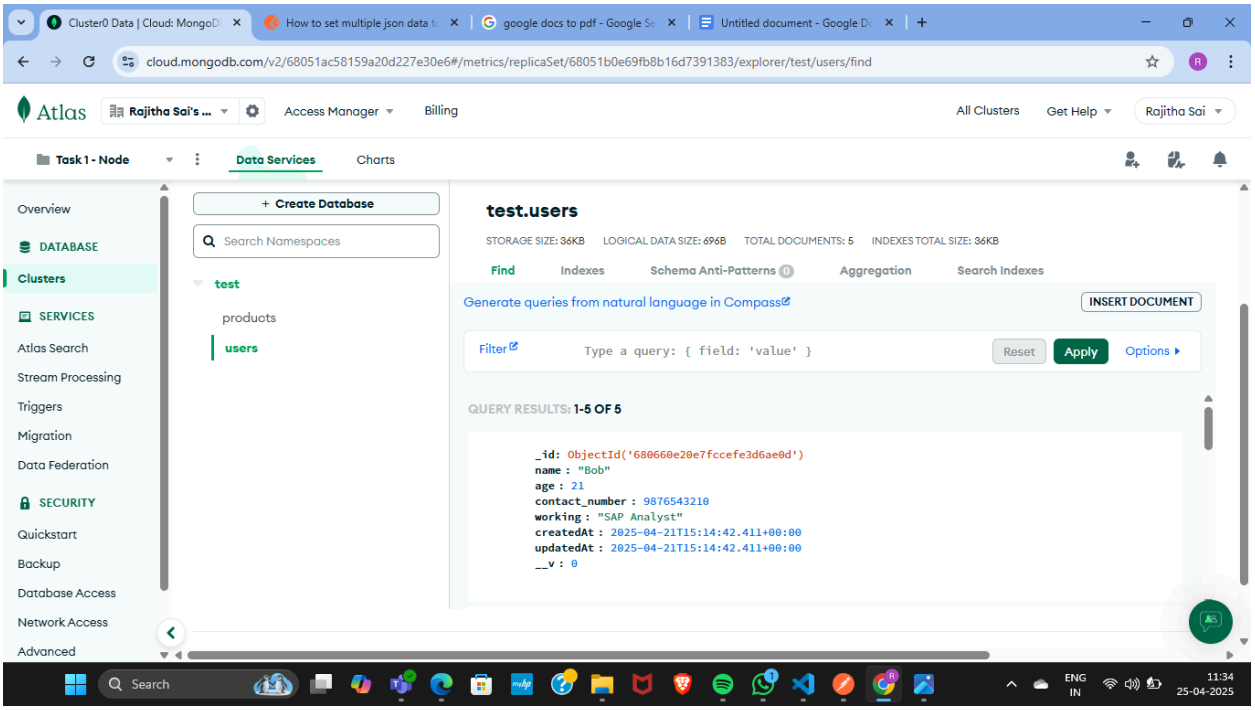


- To delete the user DEL method is used [deleting the 1st user using the id]



User Details can be seen in Mongodb





Pushing code into Github Repository

Using git commands the codes are pushed into the Repository

Repository Link : https://github.com/rajithasaivelavaluru/task-1-node-CRUD