Practical 1

* + Create web server in Node

CODE

*const* http = require('http');

*const* fs = require('fs');

*const* myServer = http.createServer((req, res) *=>* {

*const* log = `${Date.now()}: New Request for ${req.url}\n`;

    fs.appendFile('log.txt', log, (err, data) *=>* {

        console.log("NEW REQUEST RECEIVED");

*switch* (req.url) {

*case* '/': res.end("Welcome to Home Page");

*break*;

*case* '/about': res.end("Hey I am jatinRSP");

*break*;

*case* '/contact': res.end("Contact me at 9824304318");

*break*;

*default*: res.end("404 Page Not Found");

        }

    });

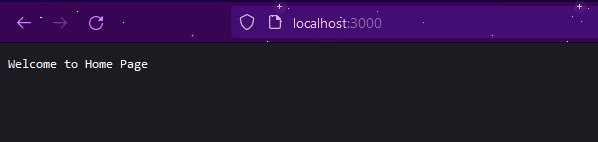
});

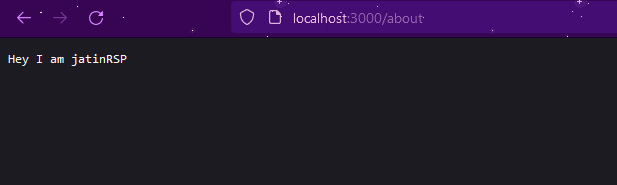
myServer.listen(3000, () *=>* {

    console.log("Server started on http://localhost:3000");

});

OUTPUT





Practical 2

* + Enhance your node web server by using Express

CODE

*const* express = require('express');

*const* fs = require('fs');

*const* app = express();

app.get('/', (req, res) *=>* {

    res.send('Welcome to Home Page');

});

app.get('/about', (req, res) *=>* {

    res.send('Hey there! Jatin RSP here');

});

app.get('/contact', (req, res) *=>* {

    res.send('Contact me at 9824304318');

});

app.get('/Shop', (req, res) *=>* {

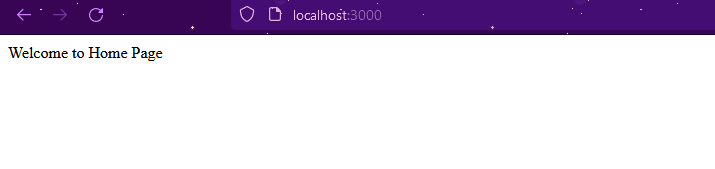
    res.send('Shop our products');

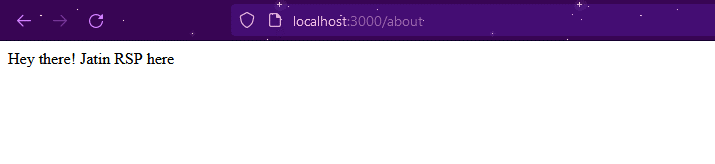
});

app.listen(3000, () *=>* {

    console.log('Server is running at port 3000');

});

OUTPUT



Practical 3

* + Setting up a MongoDB Database (Connecting MongoDB to your application)

**Experiment 3:** Create a JavaScript file with MongoDB queries for operations such as insert, update, and delete while also establishing a connection to the MongoDB database.

**Hint:** Ensure that your MongoDB server is running and accessible at localhost:27017 or replace it with the appropriate connection string if it's hosted elsewhere.

**Note: Please include snapshots of all commands, terminal sessions, localhost outputs, and Mongo compass output in your documentation with all necessary steps.**

**Code:-**

*const* mongoose = require('mongoose');

*const* validator = require('validator');

*// Connection URI*

*const* uri = 'mongodb+srv://jatinRSP:jatinRSP@cluster0.lmhkvus.mongodb.net/jatinRSP?retryWrites=true&w=majority';

*// Connect to MongoDB*

mongoose.connect(uri, { useNewUrlParser: *true*, useUnifiedTopology: *true* })

  .then(() *=>* console.log('Connected to MongoDB'))

  .catch(err *=>* console.error('Error connecting to MongoDB:', err));

*// Define friend schema*

*const* friendSchema = *new* mongoose.Schema({

  username: { type: String, required: *true* },

  email: { type: String, required: *true*, validate: [validator.isEmail, 'Invalid email'] },

  age: { type: Number, required: *true* },

});

*// Create model*

*const* Friend = mongoose.model('Friend', friendSchema);

*// Main function to perform CRUD operations*

*const* main = *async* () *=>* {

*try* {

*// Insert document*

*const* ins = {

      username: "jatinRSP",

      email: "jatinrsp575@gmail.com",

      age: 21

    };

*await* Friend.create(ins);

*// Update document*

*await* Friend.updateOne({ name: "jatinRSP" }, { $set: { age: 20 } });

*// Delete document*

*await* Friend.deleteOne({ name: "jatinRSP" });

*// Read document*

*const* read = *await* Friend.findOne({ name: "jatinRSP" });

    console.log(read.username);

  } *catch* (err) {

    console.error(err);

  } *finally* {

*// Close MongoDB connection*

*await* mongoose.connection.close();

  }

};

*// Execute main function*

main();

OUTPUT

Practical 4

* + Building models with Mongoose

**Experiment 4:** Create a database schema and model using mongoose and perform MongoDB queries for operations such as insert, update, and delete while also establishing a connection to the MongoDB database.

**Hint:** Ensure that your MongoDB server is running and accessible at localhost:27017 or replace it with the appropriate connection string if it's hosted elsewhere.

**Note:** Please include snapshots of all commands, terminal sessions, localhost outputs, and Mongo compass output in your documentation with all necessary steps

**Code:-**

*const* mongoose = require('mongoose');

*const* validator = require('validator');

*// Connection URI*

*const* uri = 'mongodb+srv://jatinRSP:jatinRSP@cluster0.lmhkvus.mongodb.net/jatinRSP?retryWrites=true&w=majority';

*// Connect to MongoDB*

mongoose.connect(uri, { useNewUrlParser: *true*, useUnifiedTopology: *true* })

  .then(() *=>* console.log('Connected to MongoDB'))

  .catch(err *=>* console.error('Error connecting to MongoDB:', err));

*// Define friend schema*

*const* friendSchema = *new* mongoose.Schema({

  name: String,

  age: Number,

  type: String,

  active: Boolean,

});

*// Sample document*

*const* ins = {

  name: "jatinRSP",

  age: 21,

  type: "friend",

  active: *true*

};

*// Create model*

*const* Friend = mongoose.model("Friend", friendSchema);

*// Main function to perform CRUD operations*

*const* main = *async* () *=>* {

*try* {

*// Insert document*

*await* Friend.insertMany(ins);

*// Update document*

*await* Friend.findOneAndUpdate({ name: "lisa" }, { $set: { age: 26 } });

*// Delete document*

*await* Friend.findOneAndDelete({ name: "lisa" });

*// Read document*

*const* read = *await* Friend.find({ name: "jatinRSP" });

    console.log(read[0].name);

  } *catch* (err) {

    console.log(err);

  } *finally* {

    mongoose.connection.close();

  }

};

*// Execute main function*

main();

Practical 5

* + Building an API in NODE

Code:

*const* express = require('express');

*const* app = express();

*const* PORT = 3000;

*// Sample data*

*const* projects = [

  { id: 1, name: 'Project 1' },

  { id: 2, name: 'Project 2' },

  { id: 3, name: 'Project 3' }

];

*// Middleware to parse JSON bodies*

app.use(express.json());

*// Route to get all projects*

app.get('/api/projects', (req, res) *=>* {

  res.json(projects);

});

*// Route to get a specific project by ID*

app.get('/api/projects/:id', (req, res) *=>* {

*const* projectId = parseInt(req.params.id);

*const* project = projects.find(project *=>* project.id === projectId);

*if* (project) {

    res.json(project);

  } *else* {

    res.status(404).json({ message: 'Project not found' });

  }

});

*// Route to create a new project*

app.post('/api/projects', (req, res) *=>* {

*const* { name } = req.body;

*if* (!name) {

*return* res.status(400).json({ message: 'Name is required for creating a project' });

  }

*const* newProject = { id: projects.length + 1, name };

  projects.push(newProject);

  res.status(201).json(newProject);

});

*// Route to update an existing project*

app.put('/api/projects/:id', (req, res) *=>* {

*const* projectId = parseInt(req.params.id);

*const* { name } = req.body;

*if* (!name) {

*return* res.status(400).json({ message: 'Name is required for updating a project' });

  }

*const* project = projects.find(project *=>* project.id === projectId);

*if* (!project) {

*return* res.status(404).json({ message: 'Project not found' });

  }

  project.name = name;

  res.json(project);

});

*// Route to delete a project*

app.delete('/api/projects/:id', (req, res) *=>* {

*const* projectId = parseInt(req.params.id);

*const* projectIndex = projects.findIndex(project *=>* project.id === projectId);

*if* (projectIndex === -1) {

*return* res.status(404).json({ message: 'Project not found' });

  }

  projects.splice(projectIndex, 1);

  res.json({ message: 'Project deleted successfully' });

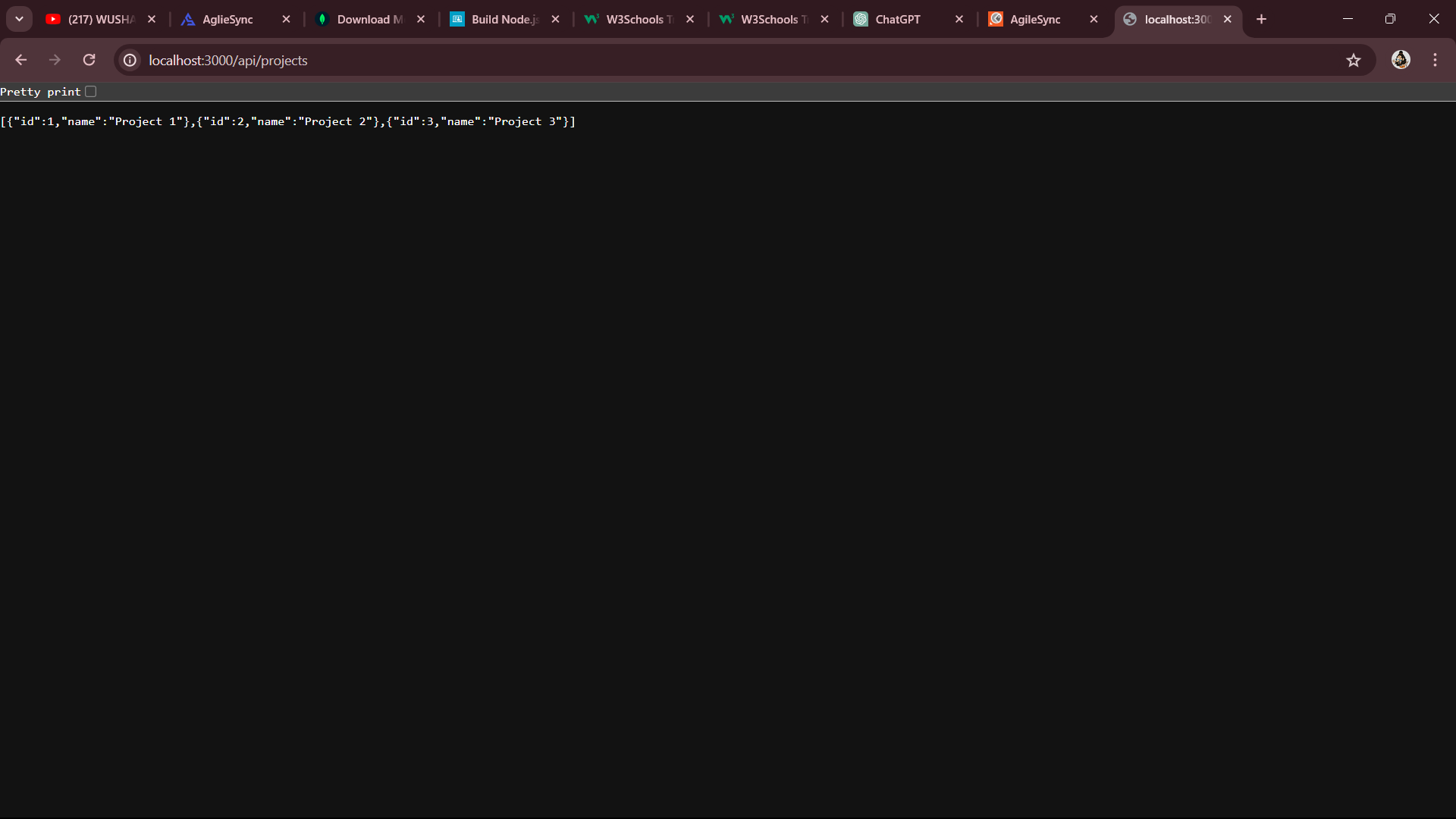
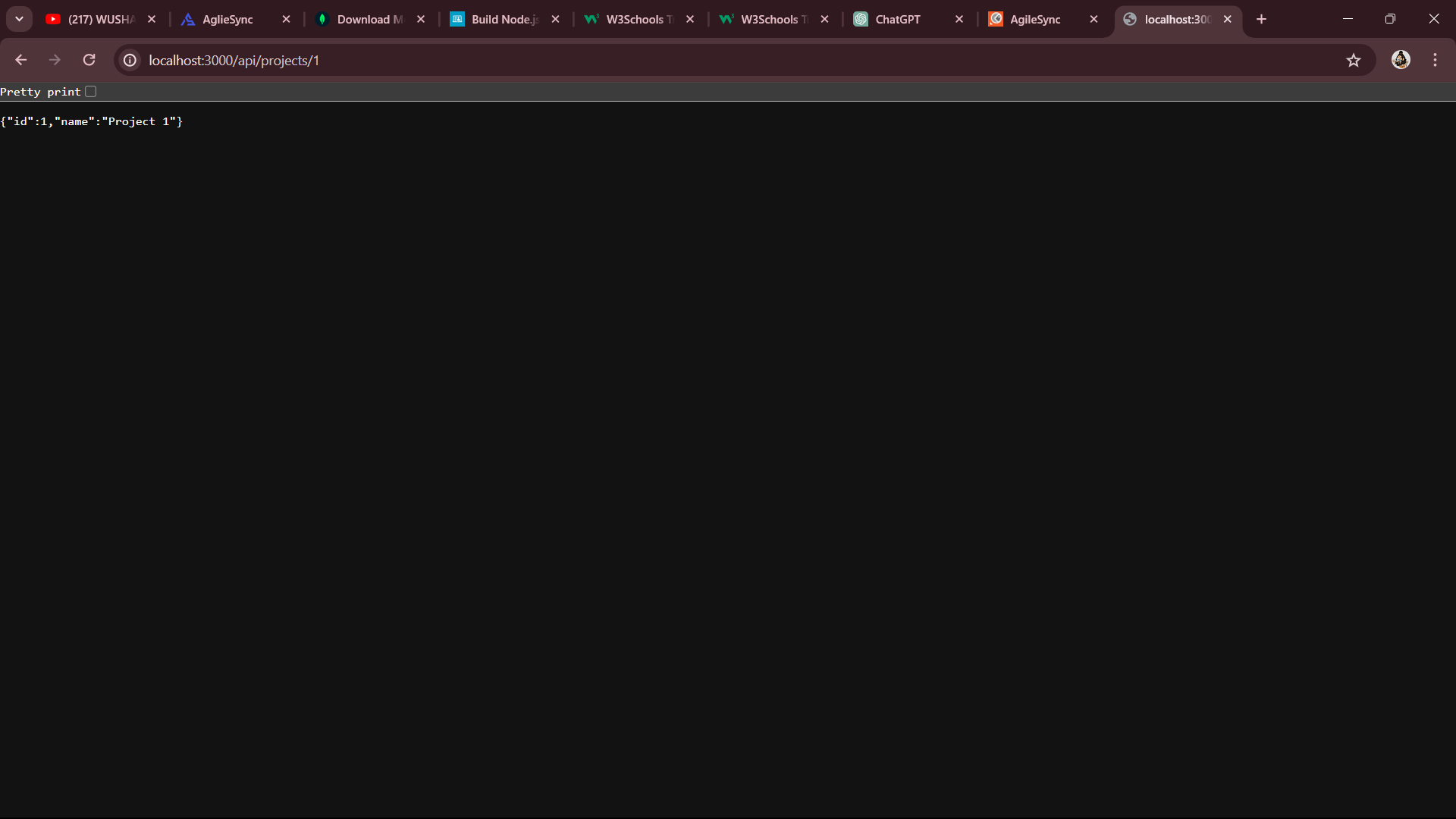
});

*// Start the server*

app.listen(PORT, () *=>* {

  console.log(`Server is running on http://localhost:${PORT}`);

});

OUTPUT

Practical 6

* + Deploy your application

Step 1: Create react app

App.jsx

*import* { useState } *from* "react";

*function* App() {

*return* (

    <>

      <h1>Deploy web app on Vercel</h1>

    </>

  );

}

*export* *default* App;

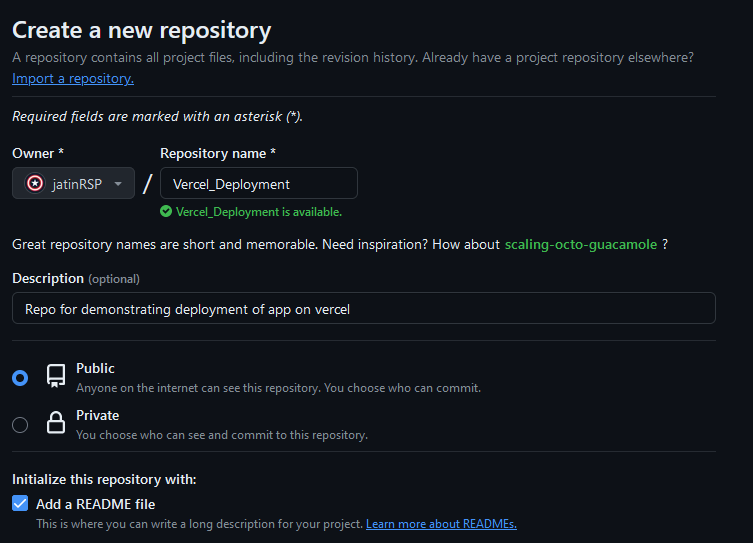
Main.jsx

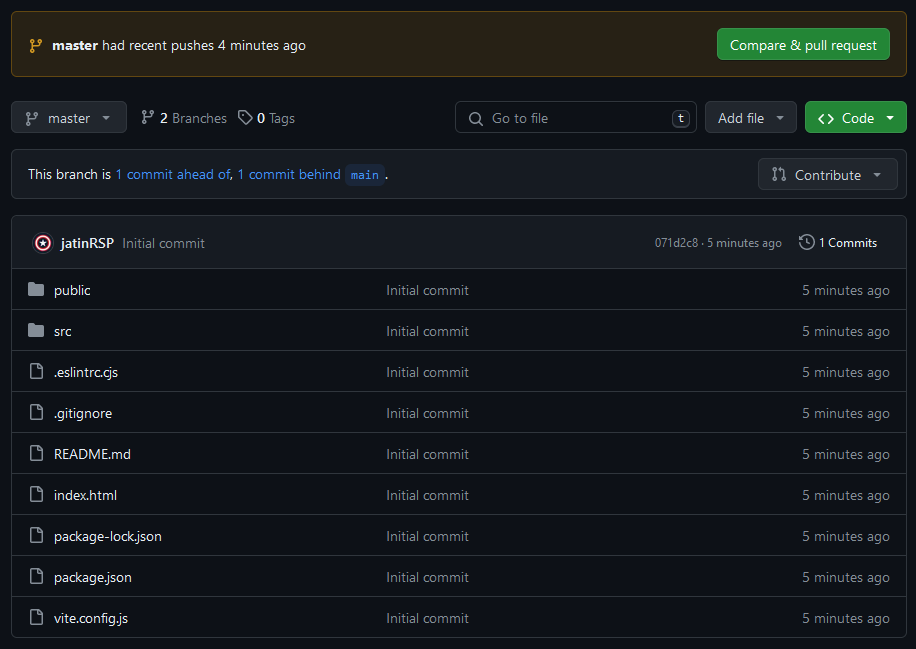
*import* React *from* "react";

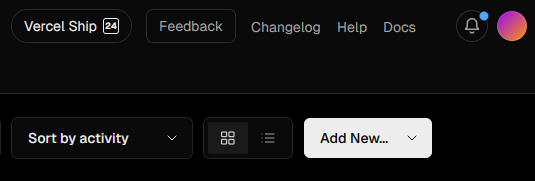
*import* ReactDOM *from* "react-dom/client";

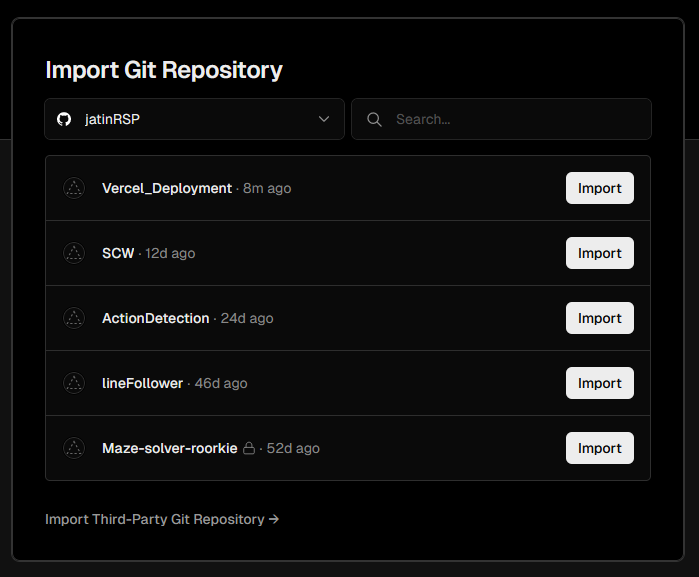
*import* App *from* "./App.jsx";

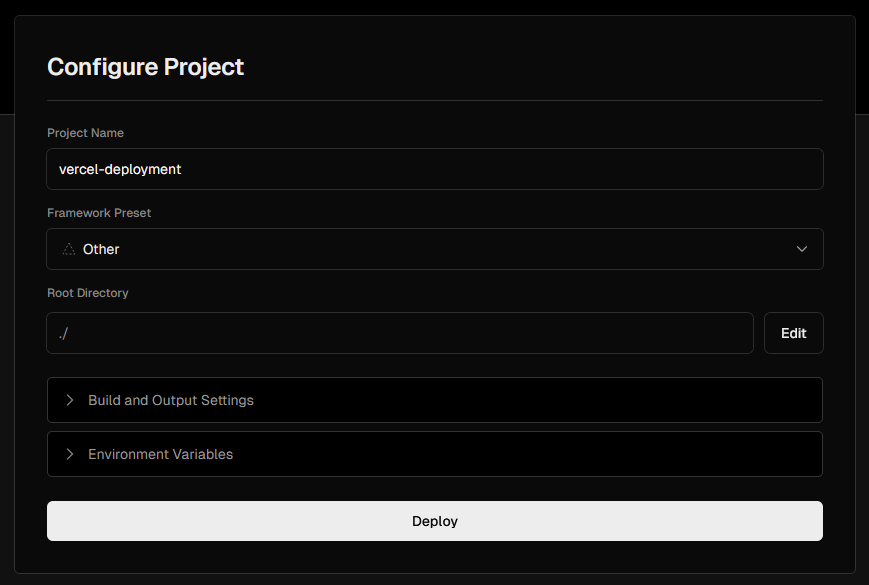
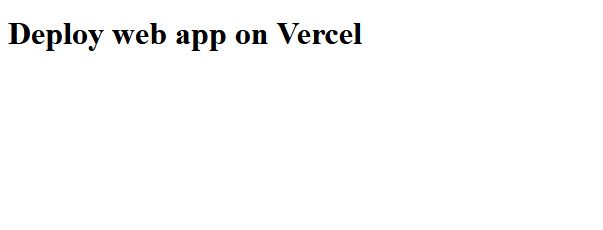
ReactDOM.createRoot(document.getElementById("root")).render(<App />);

Step 2: Create github repo and add that react app into it









Practical 7

* + Create your first React code: hello world

App.jsx

*import* { useState } *from* "react";

*function* App() {

*const* [count, setCount] = useState(0);

*return* (

    <>

      <h1>Hello World</h1>

    </>

  );

}

*export* *default* App;

Main.jsx

*import* React *from* "react";

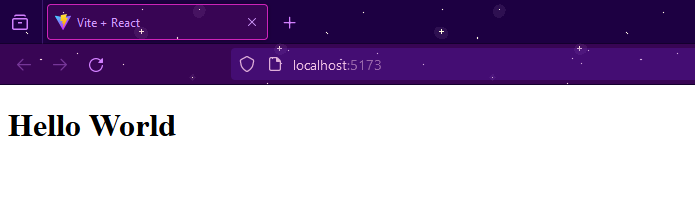
*import* ReactDOM *from* "react-dom/client";

*import* App *from* "./App.jsx";

ReactDOM.createRoot(document.getElementById("root")).render(

    <App />

);

OUTPUT

Practical 8

* + Working with properties in React.

Student.jsx

*import* React *from* "react";

*function* student(props) {

*return* (

    <div>

      <h1>Student Details</h1>

      <hr />

      <h2>{props.name}</h2>

      <h2>{props.roll}</h2>

      <h2>{props.age}</h2>

    </div>

  );

}

*export* *default* student;

App.jsx

*import* React *from* "react";

*import* Student *from* "./Student.jsx";

*function* App() {

*return* (

    <>

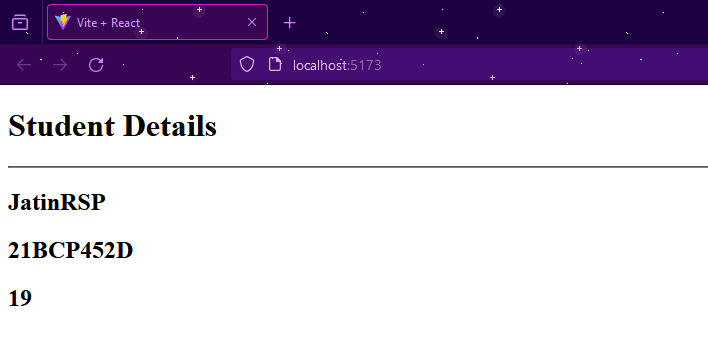
      <Student *name*="JatinRSP" *roll*="21BCP452D" *age*="19" />

    </>

  );

}

*export* *default* App;

OUTPUT

Practical 9

* + Working with react states

Form.jsx

*import* React, { useState } *from* "react";

*function* Form() {

*const* [formData, setFormData] = useState({

    name: "",

    email: "",

    message: "",

  });

*const* [submittedData, setSubmittedData] = useState(*null*);

*const* handleChange = (e) *=>* {

*const* { name, value } = e.target;

    setFormData((prevState) *=>* ({

      ...prevState,

      [name]: value,

    }));

  };

*const* handleSubmit = (e) *=>* {

    e.preventDefault();

*// Store the form data in submittedData state*

    setSubmittedData(formData);

  };

*return* (

    <div>

      <h2>Simple Form</h2>

      <form *onSubmit*={handleSubmit}>

        <div>

          <label *htmlFor*="name">Name:</label>

          <input

*type*="text"

*id*="name"

*name*="name"

*value*={formData.name}

*onChange*={handleChange}

*required*

          />

        </div>

        <div>

          <label *htmlFor*="email">Email:</label>

          <input

*type*="email"

*id*="email"

*name*="email"

*value*={formData.email}

*onChange*={handleChange}

*required*

          />

        </div>

        <div>

          <label *htmlFor*="message">Message:</label>

          <textarea

*id*="message"

*name*="message"

*value*={formData.message}

*onChange*={handleChange}

*required*

          ></textarea>

        </div>

        <button *type*="submit">Submit</button>

      </form>

      {submittedData && (

        <div>

          <h3>Form Data</h3>

          <p>Name: {submittedData.name}</p>

          <p>Email: {submittedData.email}</p>

          <p>Message: {submittedData.message}</p>

        </div>

      )}

    </div>

  );

}

*export* *default* Form;

App.jsx

*import* React *from* "react";

*import* Form *from* "./Form.jsx";

*function* App() {

*return* (

    <>

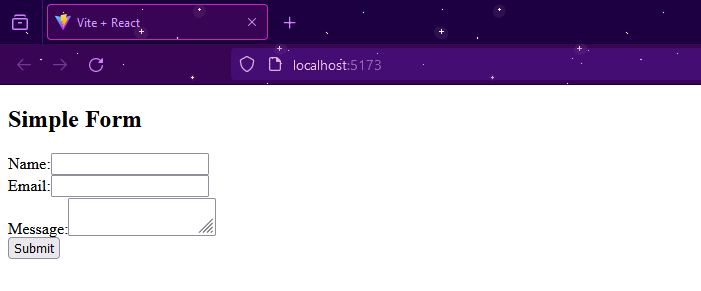
      <Form />

    </>

  );

}

*export* *default* App;

OUTPUT

Practical 10

* + Working with React Forms

Form.jsx

*import* React, { useState } *from* "react";

*function* Form() {

*const* [formData, setFormData] = useState({

    name: "",

    email: "",

    message: "",

  });

*const* [submittedData, setSubmittedData] = useState(*null*);

*const* handleChange = (e) *=>* {

*const* { name, value } = e.target;

    setFormData((prevState) *=>* ({

      ...prevState,

      [name]: value,

    }));

  };

*const* handleSubmit = (e) *=>* {

    e.preventDefault();

*// Store the form data in submittedData state*

    setSubmittedData(formData);

  };

*return* (

    <div>

      <h2>Simple Form</h2>

      <form *onSubmit*={handleSubmit}>

        <div>

          <label *htmlFor*="name">Name:</label>

          <input

*type*="text"

*id*="name"

*name*="name"

*value*={formData.name}

*onChange*={handleChange}

*required*

          />

        </div>

        <div>

          <label *htmlFor*="email">Email:</label>

          <input

*type*="email"

*id*="email"

*name*="email"

*value*={formData.email}

*onChange*={handleChange}

*required*

          />

        </div>

        <div>

          <label *htmlFor*="message">Message:</label>

          <textarea

*id*="message"

*name*="message"

*value*={formData.message}

*onChange*={handleChange}

*required*

          ></textarea>

        </div>

        <button *type*="submit">Submit</button>

      </form>

      {submittedData && (

        <div>

          <h3>Form Data</h3>

          <p>Name: {submittedData.name}</p>

          <p>Email: {submittedData.email}</p>

          <p>Message: {submittedData.message}</p>

        </div>

      )}

    </div>

  );

}

*export* *default* Form;

App.jsx

*import* React *from* "react";

*import* Form *from* "./Form.jsx";

*function* App() {

*return* (

    <>

      <Form />

    </>

  );

}

*export* *default* App;

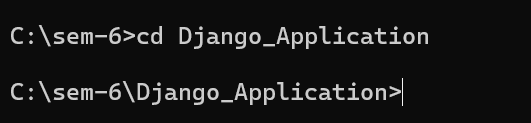
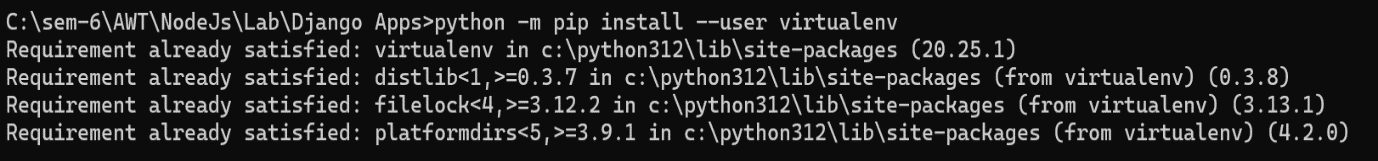
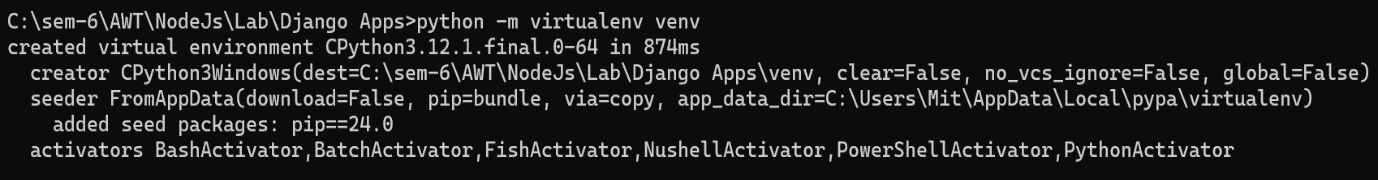
OUTPUT

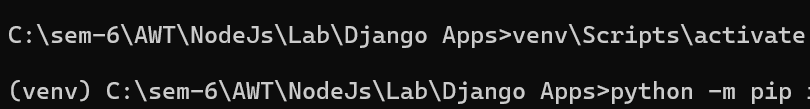
Practical 11

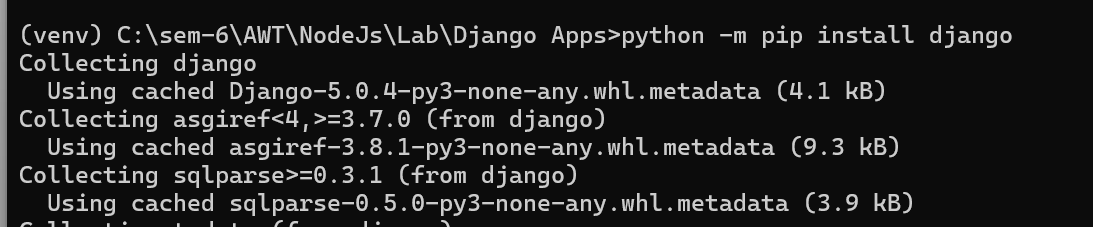
* + Creating basic Django web app

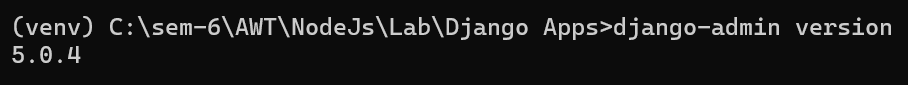
**Aim:** Build a simple webapp using Django.

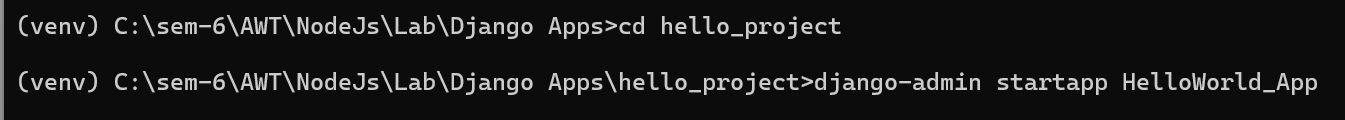
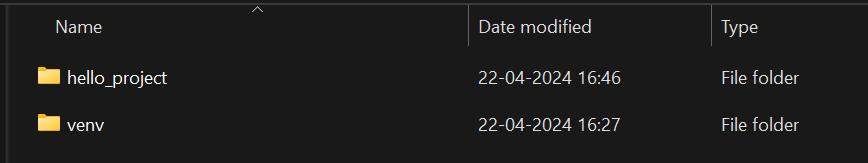
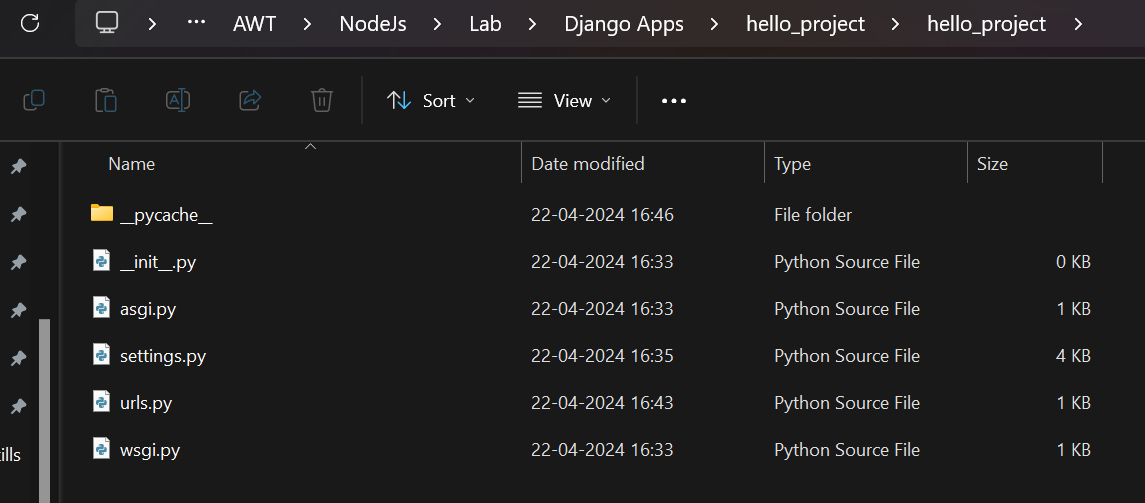
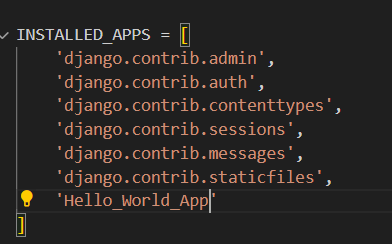
**Setup & Code & Output:**

1. Open the cmd at desired path and make a directory named ‘ Django\_Apps’.
2. ****Change directory to that Django\_Apps.
3. Create the virtual Environment.
4. Activate the virtual environment.

****

1. ****Install the Django.

****Django Verison checking:

1. ****Create the Django Project.
2. Go inside this Django Project and create the Django App.
3. ****Now, Inside Django Project -> settings.py -> add app name into INSTALLED\_APPS.
4. Write the webapp code, Django project -> Django App -> views.py

**Views.py:**

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

def HelloWorld(request):

    return HttpResponse('<h1>Hello JatinRSP Whatsapp?? </h1>')

1. Make the connect between the Django project and Django app by adding the app to Django project-> url.py.

url.py (Project’s url file)

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('',include('Hello\_World\_App.urls'))

]

1. **Make the url.py inside the Django App and import the response function (which give response on http request.**

url.py (Django App’s url file)

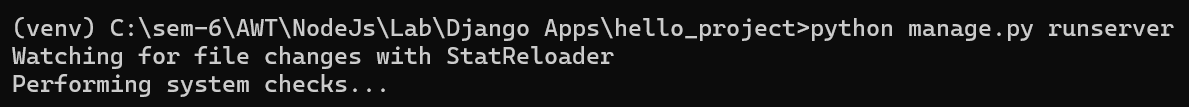
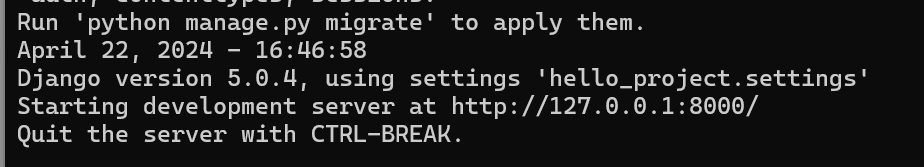
from django.urls import path,include

from .views import HelloWorld

urlpatterns = [

    path('',HelloWorld,name='HelloWorld'),

]

1. Run the webapp : ‘ python manage.py runserver ’

**Output:**

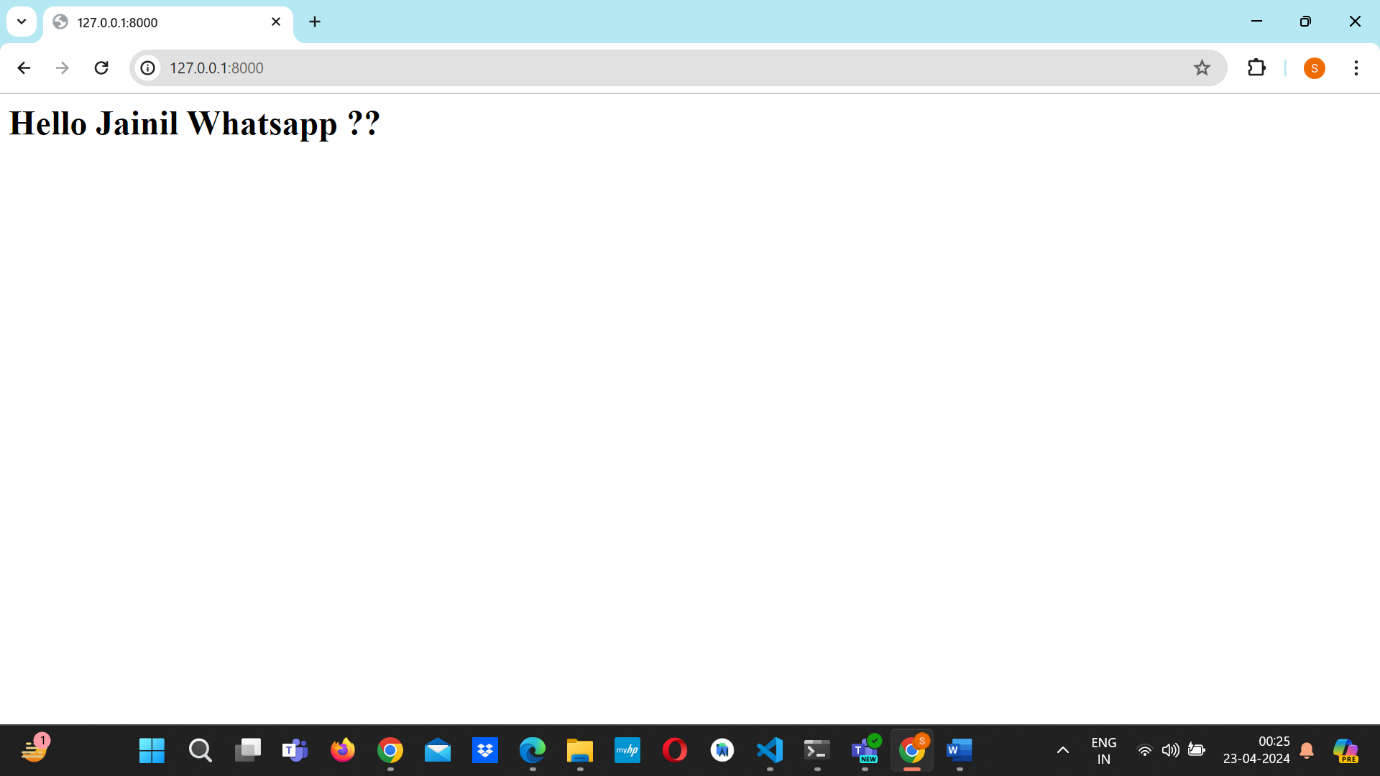
Webapp-2: Creating the webapp to login authorization webapp.

Setup & Code & Output:

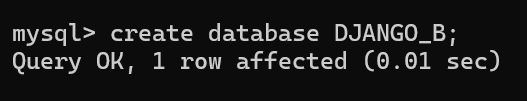
* Installation of modules:

pip install mysqlclient

pip install mysql-connector-python



**Hello Jatin What is Going ON??**

* ****Create the database:

Practical 12

* + Build a simple flask app

# Code:

*from* flask *import* Flask

*# Create a Flask application instance*

app = Flask(\_\_name\_\_)

*# Define a route for the root URL '/'*

@app.route('/')

*def* hello\_world():

*return* 'Hello, World!'

*# Run the Flask application*

*if* \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=*True*)

# Output:

