Here’s a step-by-step guide to creating an online code editor with IDE functionalities using HTML, CSS, ReactJS, and a backend API:

1. Project Structure

Backend API

- **Technologies**: Node.js, Express.js
- **Purpose**: To take code and language as input, execute the code on the server, and return the output.

Frontend Editor

- **Technologies**: React.js, HTML, CSS
- **Purpose**: To allow users to choose a language, write, edit, and modify code, and submit it to the backend API for execution.

2. Backend Setup

1. Initialize Node.js Project

sh

Copy code

mkdir code-editor-backend

cd code-editor-backend

npm init -y

2. Install Dependencies

sh

Copy code

npm install express body-parser child_process

- express: For creating the server.
- body-parser: For parsing incoming request bodies.
- child_process: For executing shell commands.

3. Create server.js

js

Copy code

const express = require('express');

const bodyParser = require('body-parser');

const { exec } = require('child_process');

```
const app = express();
const PORT = process.env.PORT || 5000;

app.use(bodyParser.json());

app.post('/run', (req, res) => {
  const { language, code } = req.body;
  let command;

  switch (language) {
    case 'python':
      command = `python -c "${code.replace(/"/g, '\\"')}"`;
      break;
    case 'javascript':
      command = `node -e "${code.replace(/"/g, '\\"')}"`;
      break;
    // Add more languages as needed
    default:
      return res.status(400).send('Language not supported');
  }

  exec(command, (error, stdout, stderr) => {
    if (error) {
      return res.status(500).send(stderr);
    }
    res.send(stdout);
  });
});

app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
```

});

4. Run the Server

sh

Copy code

node server.js

3. Frontend Setup

1. Initialize React Project

sh

Copy code

npx create-react-app code-editor-frontend

cd code-editor-frontend

2. Install Axios

sh

Copy code

npm install axios

3. Create Components

App.js

js

Copy code

```
import React, { useState } from 'react';
import axios from 'axios';
import './App.css';

function App() {
  const [code, setCode] = useState('');
  const [language, setLanguage] = useState('javascript');
  const [output, setOutput] = useState('');

  const runCode = async () => {
    try {
      const response = await axios.post('http://localhost:5000/run', { code, language });
```

```
      setOutput(response.data);
    } catch (error) {
      setOutput(error.response ? error.response.data : 'Error');
    }
  };

  return (
    <div className="App">
      <h1>Online Code Editor</h1>
      <select value={language} onChange={(e) => setLanguage(e.target.value)}>
        <option value="javascript">JavaScript</option>
        <option value="python">Python</option>
        {/* Add more languages as needed */}
      </select>
      <textarea
        value={code}
        onChange={(e) => setCode(e.target.value)}
        rows="20"
        cols="100"
      ></textarea>
      <button onClick={runCode}>Run</button>
      <pre>{output}</pre>
    </div>
  );
}

export default App;
```

App.css

css

Copy code

```
.App {
```

```
    font-family: sans-serif;
    text-align: center;
    margin: 20px;
}

textarea {
    width: 80%;
    margin: 20px 0;
}

button {
    margin: 20px;
    padding: 10px 20px;
}

pre {
    background-color: #f5f5f5;
    padding: 20px;
    width: 80%;
    margin: 20px auto;
    text-align: left;
}
```

4. Run the Frontend

sh

Copy code

npm start

5. Deployment

1. Deploy Backend

- Use services like Heroku, AWS, or DigitalOcean to deploy your backend API.

2. Deploy Frontend

- Use services like Netlify, Vercel, or GitHub Pages to deploy your frontend application.

3. Update API Endpoint

- Once deployed, update the backend API endpoint in the React application to point to the deployed backend service.

This setup provides a complete and functional online code editor with basic IDE functionalities. Let me know if you need more detailed guidance on any part!