

Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

Invoice Reminder and Follow-up Automation with Zapier Integration

Description:

Your task is to develop a system that allows users to log in using Google OAuth, view the details of due invoices in a SaaS platform, and automate the process of sending past-due invoice reminders and follow-up sequences. The system should integrate with Zapier to trigger automation actions. This assignment assesses your ability to integrate third-party services, implement OAuth authentication, and design a microservices architecture

Requirements:

Backend Microservice (Node.js):

Implement a Node.js backend microservice that provides the following API endpoints:

- User authentication: Enable users to log in using Google OAuth.
- Invoice details: Allow users to view details of their due invoices, including the invoice amount, due date, and recipient information.
- Integration with Zapier: Create endpoints to trigger automation actions in response to past-due invoices.
- Integrate with Zapier to automate past-due invoice reminders and follow-up sequences. Set up workflows in Zapier to send email reminders and follow-up notifications based on past-due invoice data.

Frontend (React):

Develop a React frontend that includes the following features:

- Google OAuth integration: Implement a user interface for users to log in using their Google accounts.
- Invoice details: Display a list of due invoices with relevant details.
- Trigger automation: Provide a button or option for users to manually trigger the automation process using Zapier.

Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

Table of Contents

1. [Project Overview](#)
2. [Technologies Used](#)
3. [Backend Documentation](#)
 - [Setup Instructions](#)
 - [API Endpoints](#)
 - [Google OAuth Implementation](#)
 - [Invoice Details](#)
 - [Zapier Integration](#)
4. [Frontend Documentation](#)
 - [Setup Instructions](#)
 - [Project Structure](#)
 - [Google OAuth Integration](#)
 - [Invoice Display](#)
 - [Trigger Automation](#)

Project Overview

The Invoice Reminder System allows users to log in using Google OAuth, view due invoices, and automate sending past-due invoice reminders and follow-up sequences through Zapier.

Technologies Used

- Backend: Node.js, Express, Passport.js, MongoDB
- Frontend: React, Axios
- Authentication: Google OAuth 2.0
- Automation: Zapier

Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

Backend Setup Instructions

1. Prerequisites:

- Node.js
- MongoDB

Installation:

To implement the required backend microservice using Node.js, we need to set up a project that handles user authentication via Google OAuth, manages invoice details, and integrates with Zapier for automation. Below is a step-by-step guide to achieve this:

1. Project Setup

Initialize the Project:

Code:

```
mkdir invoice-microservice
cd invoice-microservice
npm init -y
```

Install Required Packages:

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS C:\Users\User\tencargo_project\frontend\frontend> npm install express axios mongoose passport passport-google-oauth20 jsonwebtoken dotenv node-cron
>> █
```

Create Folder Structure:

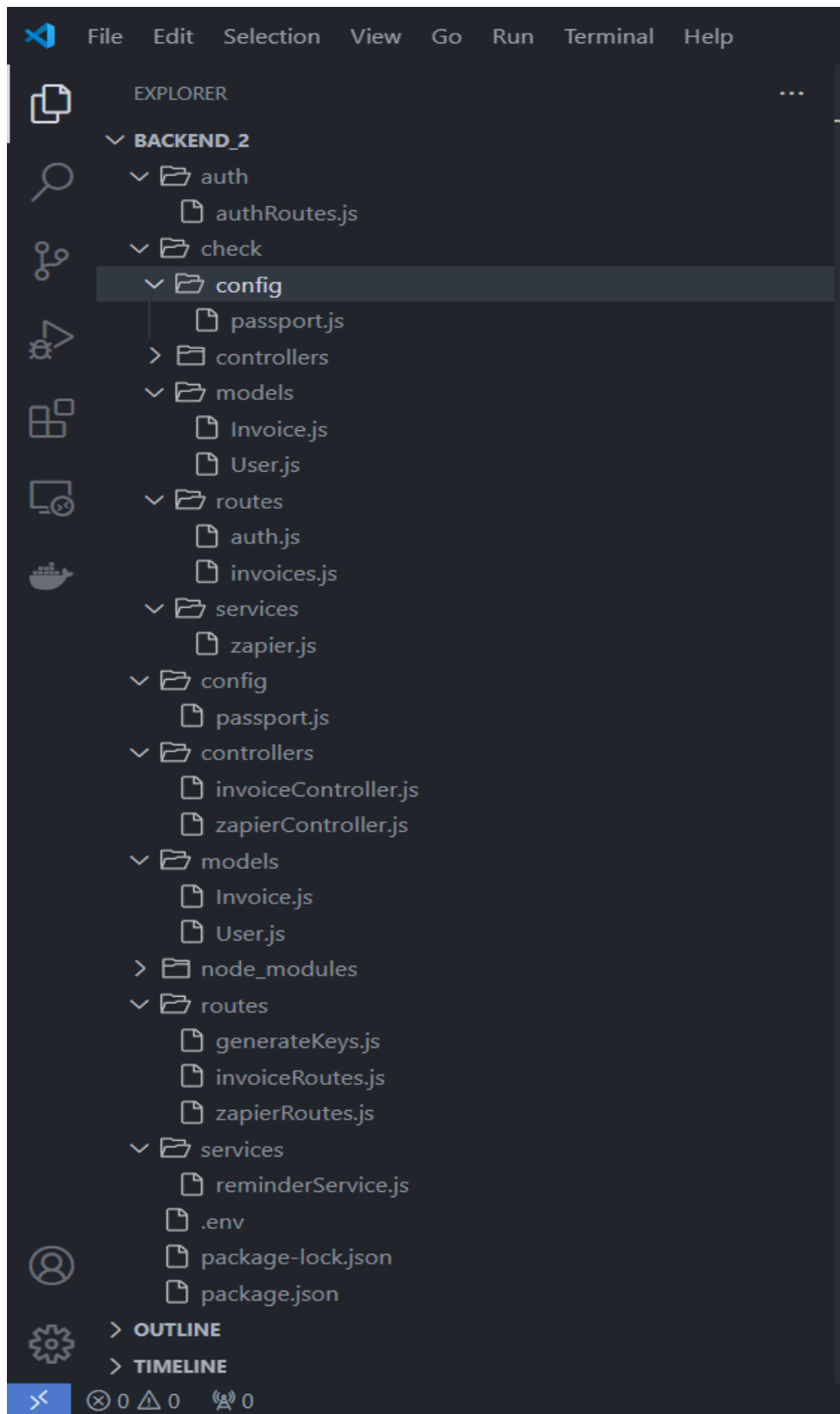
backend

```
|— auth/
|   |— authRoutes.js
|— check/
|— config/
|   |— passport.js
|— controllers/
|   |— invoiceController.js
|   |— zapierController.js
|— models/
|   |— Invoice.js
|   |— User.js
|— routes/
|   |— auth.js
|   |— generateKeys.js
|   |— invoiceRoutes.js
|   |— zapierRoutes.js
|— services/
|   |— reminderService.js
|   |— zapier.js
|— .env
```

Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

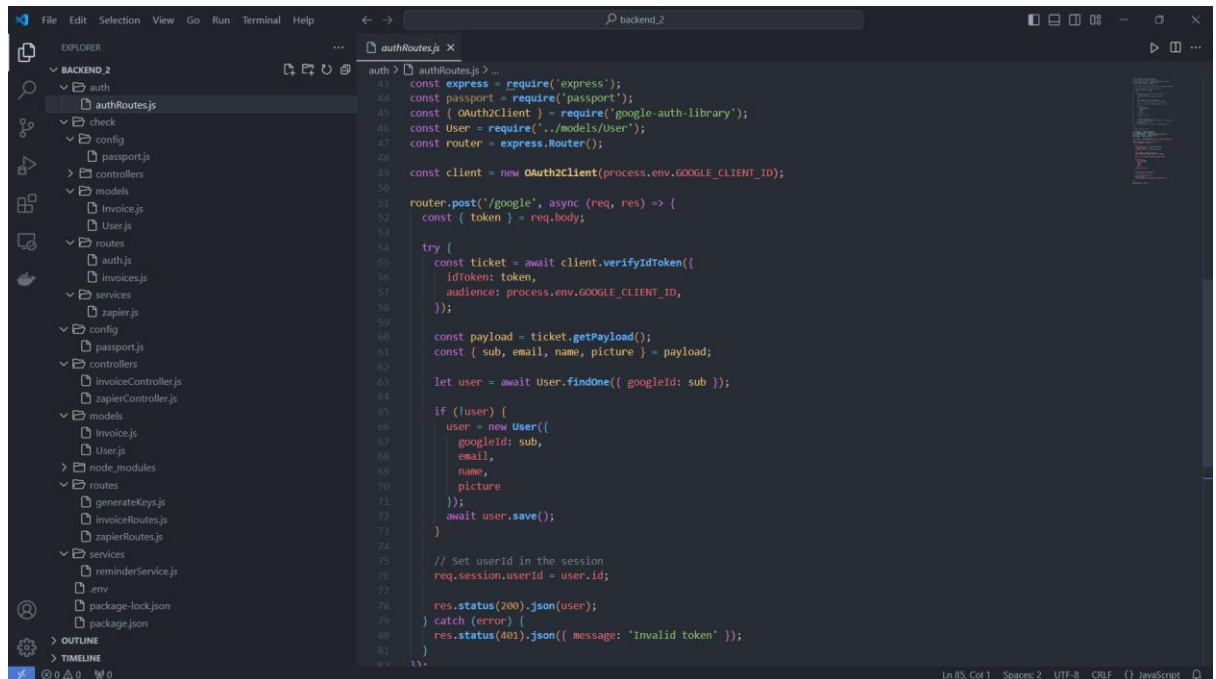
- package-lock.json
- package.json



Name: Thiyaagu.E

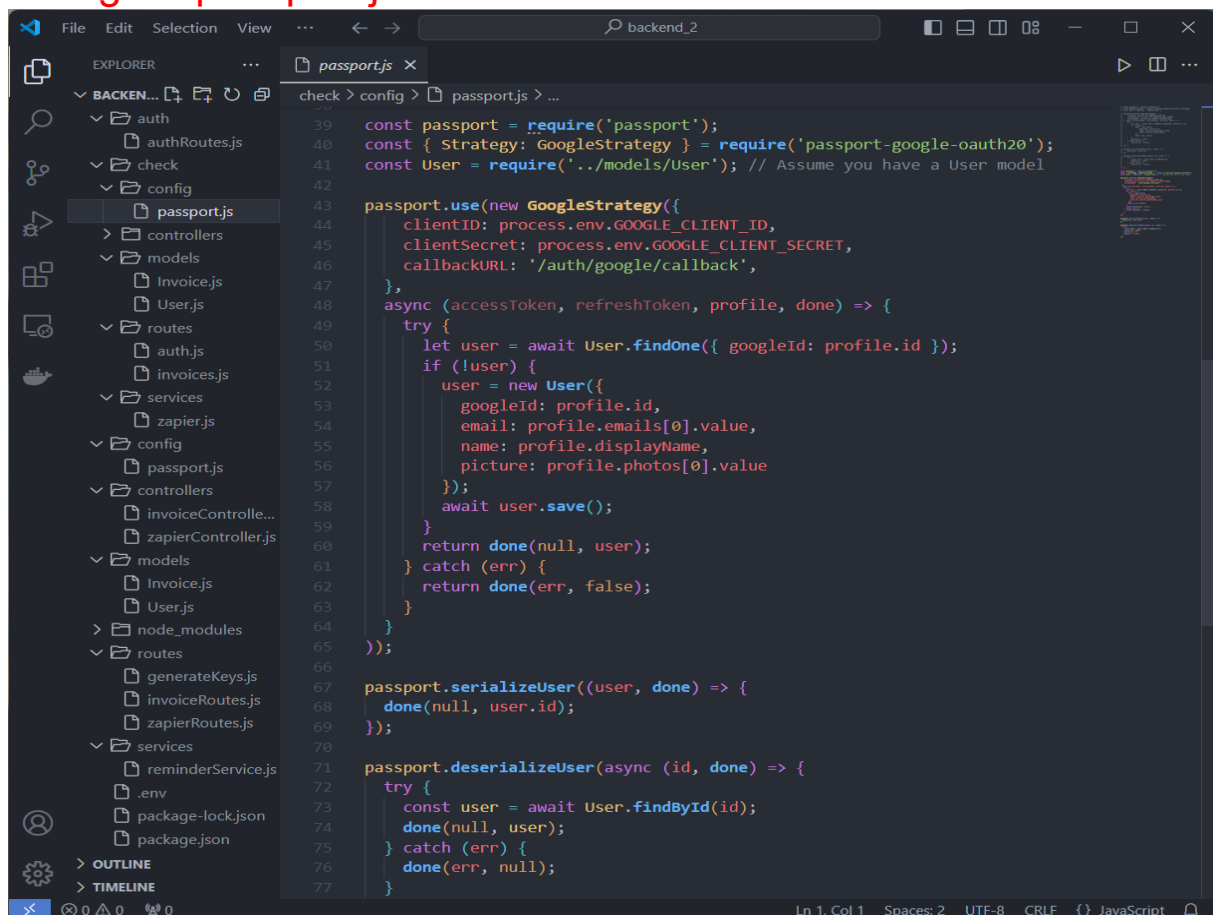
G-Mail: thiyaaguvit@gmail.com

Auth => authRoutes.js



```
auth > authRoutes.js > ...
43 const express = require('express');
44 const passport = require('passport');
45 const { OAuth2Client } = require('google-auth-library');
46 const User = require('../models/User');
47 const router = express.Router();
48
49 const client = new OAuth2Client(process.env.GOOGLE_CLIENT_ID);
50
51 router.post('/google', async (req, res) => {
52   const { token } = req.body;
53
54   try {
55     const ticket = await client.verifyIdToken({
56       idToken: token,
57       audience: process.env.GOOGLE_CLIENT_ID,
58     });
59
60     const payload = ticket.getPayload();
61     const { sub, email, name, picture } = payload;
62
63     let user = await User.findOne({ googleId: sub });
64
65     if (!user) {
66       user = new User({
67         googleId: sub,
68         email,
69         name,
70         picture
71       });
72       await user.save();
73     }
74
75     // Set userId in the session
76     req.session.userId = user.id;
77
78     res.status(200).json(user);
79   } catch (error) {
80     res.status(401).json({ message: 'Invalid token' });
81   }
82 })
```

config => passport.js

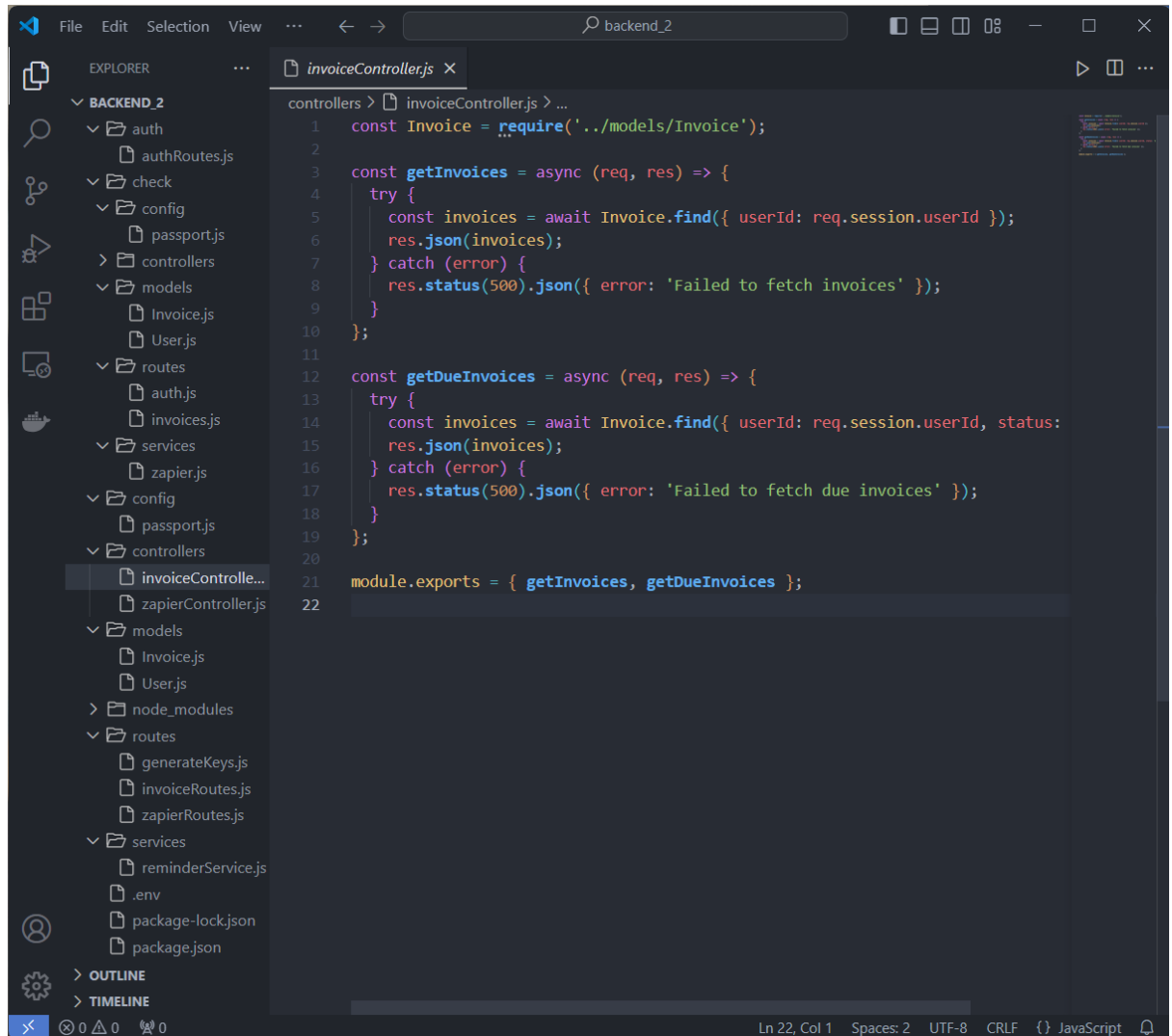


```
check > config > passport.js > ...
39 const passport = require('passport');
40 const { Strategy: GoogleStrategy } = require('passport-google-oauth20');
41 const User = require('../models/User'); // Assume you have a User model
42
43 passport.use(new GoogleStrategy({
44   clientID: process.env.GOOGLE_CLIENT_ID,
45   clientSecret: process.env.GOOGLE_CLIENT_SECRET,
46   callbackURL: '/auth/google/callback',
47 },
48 async (accessToken, refreshToken, profile, done) => {
49   try {
50     let user = await User.findOne({ googleId: profile.id });
51     if (!user) {
52       user = new User({
53         googleId: profile.id,
54         email: profile.emails[0].value,
55         name: profile.displayName,
56         picture: profile.photos[0].value
57       });
58       await user.save();
59     }
60     return done(null, user);
61   } catch (err) {
62     return done(err, false);
63   }
64 });
65
66 passport.serializeUser((user, done) => {
67   done(null, user.id);
68 });
69
70 passport.deserializeUser(async (id, done) => {
71   try {
72     const user = await User.findById(id);
73     done(null, user);
74   } catch (err) {
75     done(err, null);
76   }
77 })
```

Name: Thiyagu.E

G-Mail: thiyaguvit@gmail.com

Controllers => invoiceController.js:

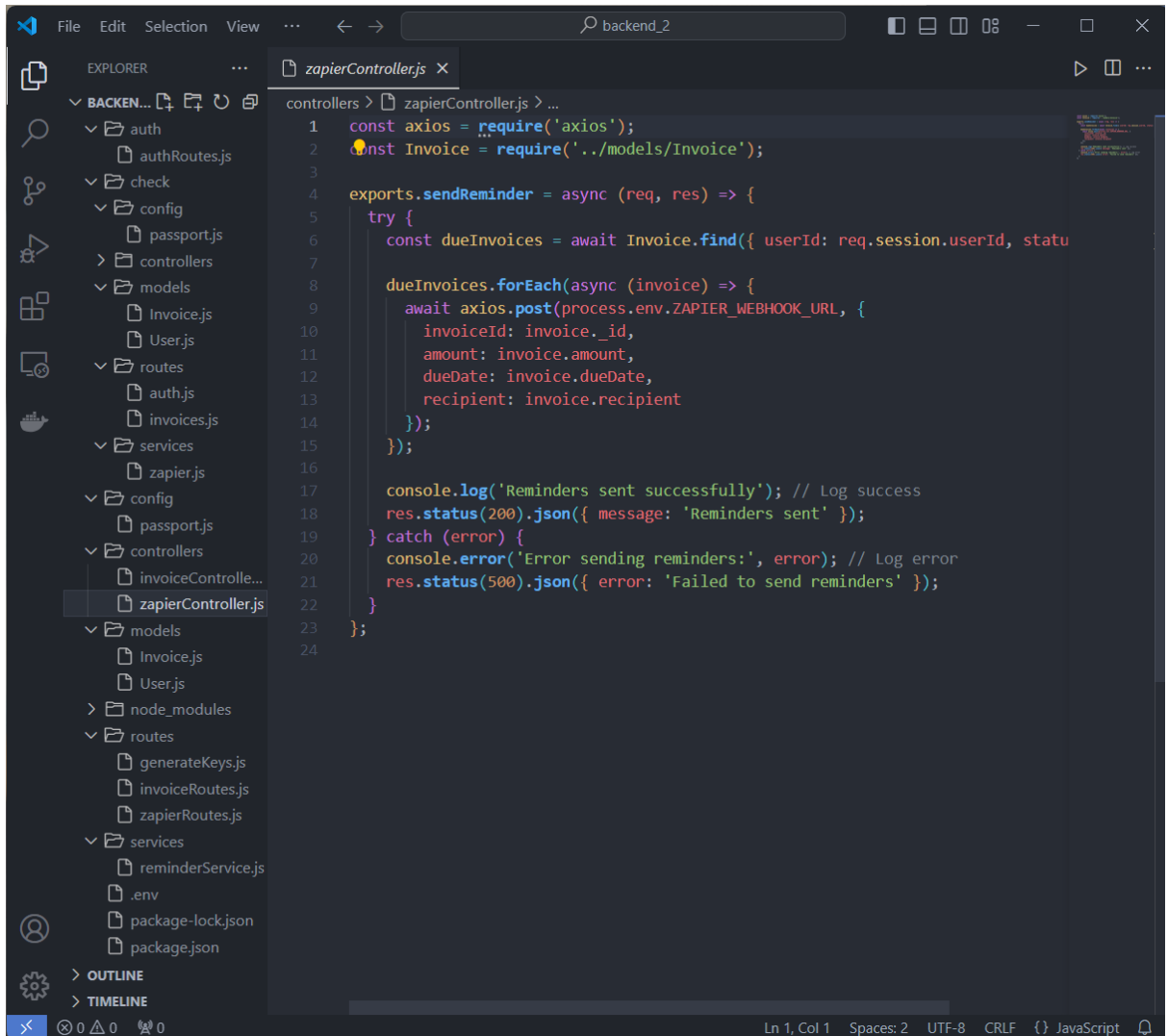


```
1  const Invoice = require('../models/Invoice');
2
3  const getInvoices = async (req, res) => {
4    try {
5      const invoices = await Invoice.find({ userId: req.session.userId });
6      res.json(invoices);
7    } catch (error) {
8      res.status(500).json({ error: 'Failed to fetch invoices' });
9    }
10   };
11
12  const getDueInvoices = async (req, res) => {
13    try {
14      const invoices = await Invoice.find({ userId: req.session.userId, status:
15      res.json(invoices);
16    } catch (error) {
17      res.status(500).json({ error: 'Failed to fetch due invoices' });
18    }
19   };
20
21  module.exports = { getInvoices, getDueInvoices };
22
```

Name: Thiyagu.E

G-Mail: thiyaguvit@gmail.com

zapierController.js:

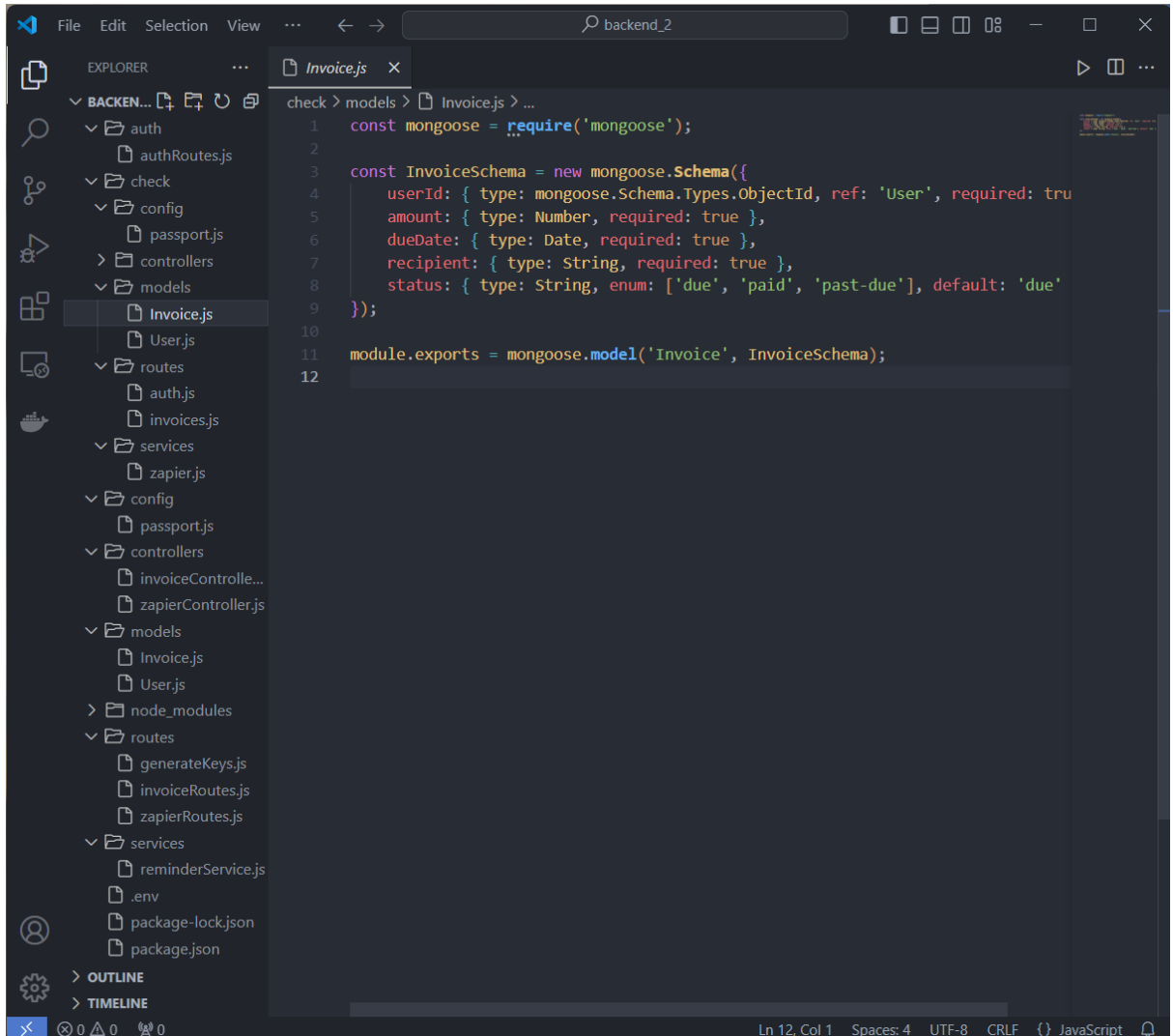


```
1 const axios = require('axios');
2 const Invoice = require('../models/Invoice');
3
4 exports.sendReminder = async (req, res) => {
5   try {
6     const dueInvoices = await Invoice.find({ userId: req.session.userId, statu
7
8     dueInvoices.forEach(async (invoice) => {
9       await axios.post(process.env.ZAPIER_WEBHOOK_URL, {
10         invoiceId: invoice._id,
11         amount: invoice.amount,
12         dueDate: invoice.dueDate,
13         recipient: invoice.recipient
14       });
15     });
16
17     console.log('Reminders sent successfully'); // Log success
18     res.status(200).json({ message: 'Reminders sent' });
19   } catch (error) {
20     console.error('Error sending reminders:', error); // Log error
21     res.status(500).json({ error: 'Failed to send reminders' });
22   }
23 };
24
```

Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

Models => Invoice.js:

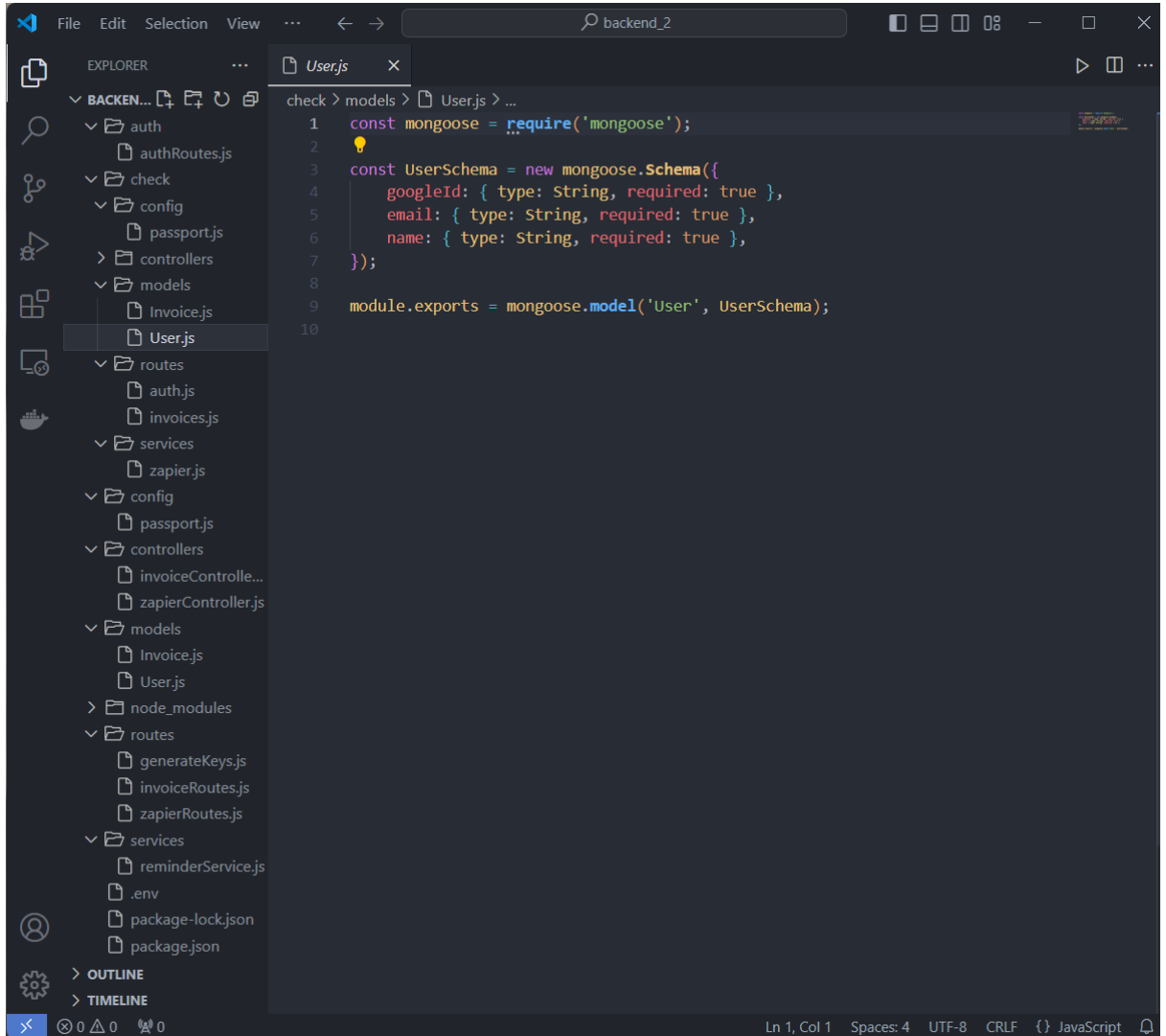


```
File Edit Selection View ... backend_2
EXPLORER
  BACKEN...
    auth
      authRoutes.js
    check
      config
      passport.js
    controllers
    models
      Invoice.js
      User.js
    routes
      auth.js
      invoices.js
    services
      zapier.js
    config
      passport.js
    controllers
      invoiceControlle...
      zapierController.js
    models
      Invoice.js
      User.js
    node_modules
    routes
      generateKeys.js
      invoiceRoutes.js
      zapierRoutes.js
    services
      reminderService.js
    .env
    package-lock.json
    package.json
  OUTLINE
  TIMELINE
  Invoice.js
    check > models > Invoice.js > ...
    1 const mongoose = require('mongoose');
    2
    3 const InvoiceSchema = new mongoose.Schema({
    4   userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    5   amount: { type: Number, required: true },
    6   dueDate: { type: Date, required: true },
    7   recipient: { type: String, required: true },
    8   status: { type: String, enum: ['due', 'paid', 'past-due'], default: 'due'
    9 });
    10
    11 module.exports = mongoose.model('Invoice', InvoiceSchema);
    12
```


Name: Thiyaqu.E

G-Mail: thiyaquvit@gmail.com

User.js:

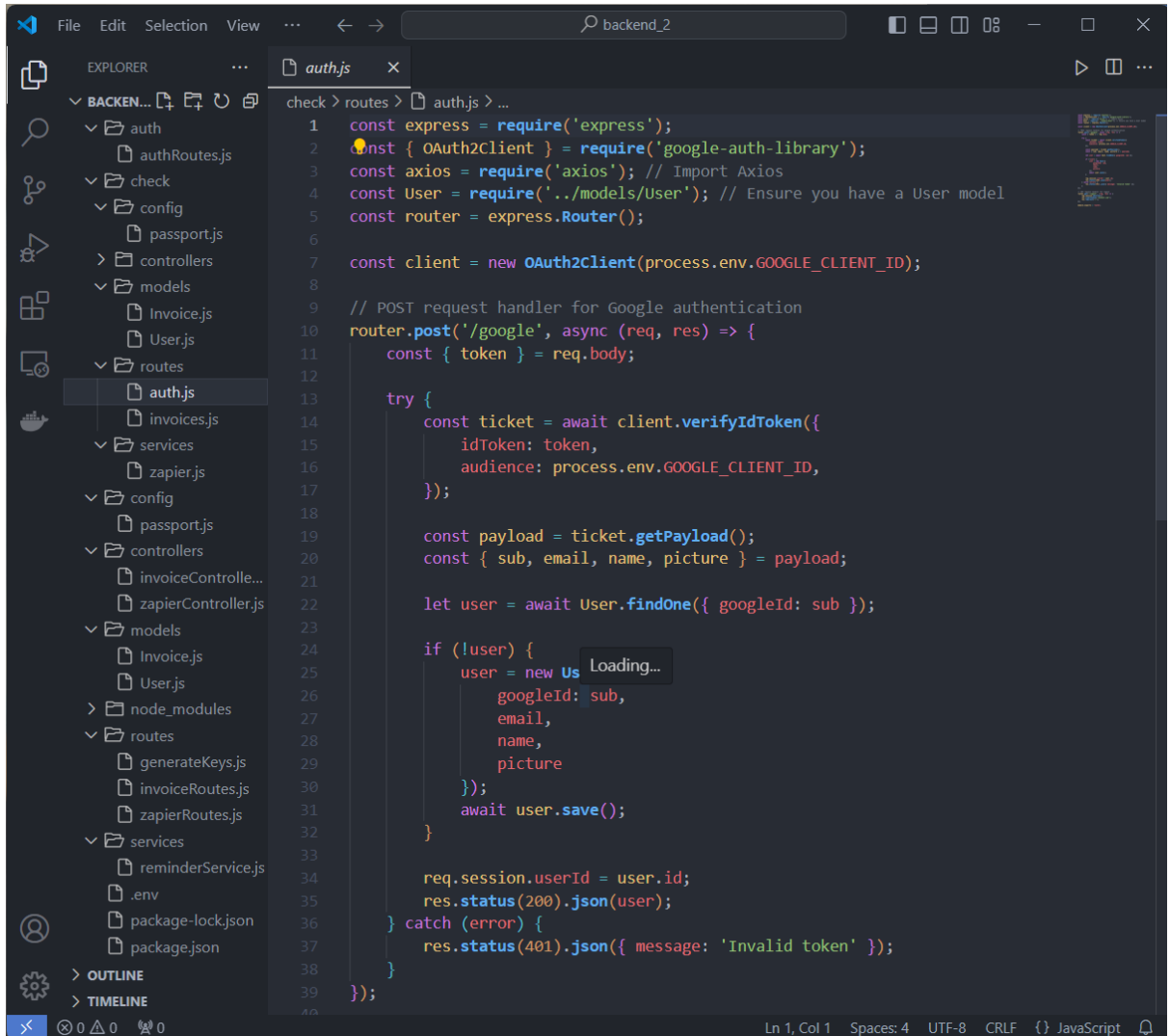


```
1  const mongoose = require('mongoose');
2
3  const UserSchema = new mongoose.Schema({
4    googleId: { type: String, required: true },
5    email: { type: String, required: true },
6    name: { type: String, required: true },
7  });
8
9  module.exports = mongoose.model('User', UserSchema);
10
```

Name: Thiyaгу.E

G-Mail: thiyaгuvit@gmail.com

Routes => auth.js

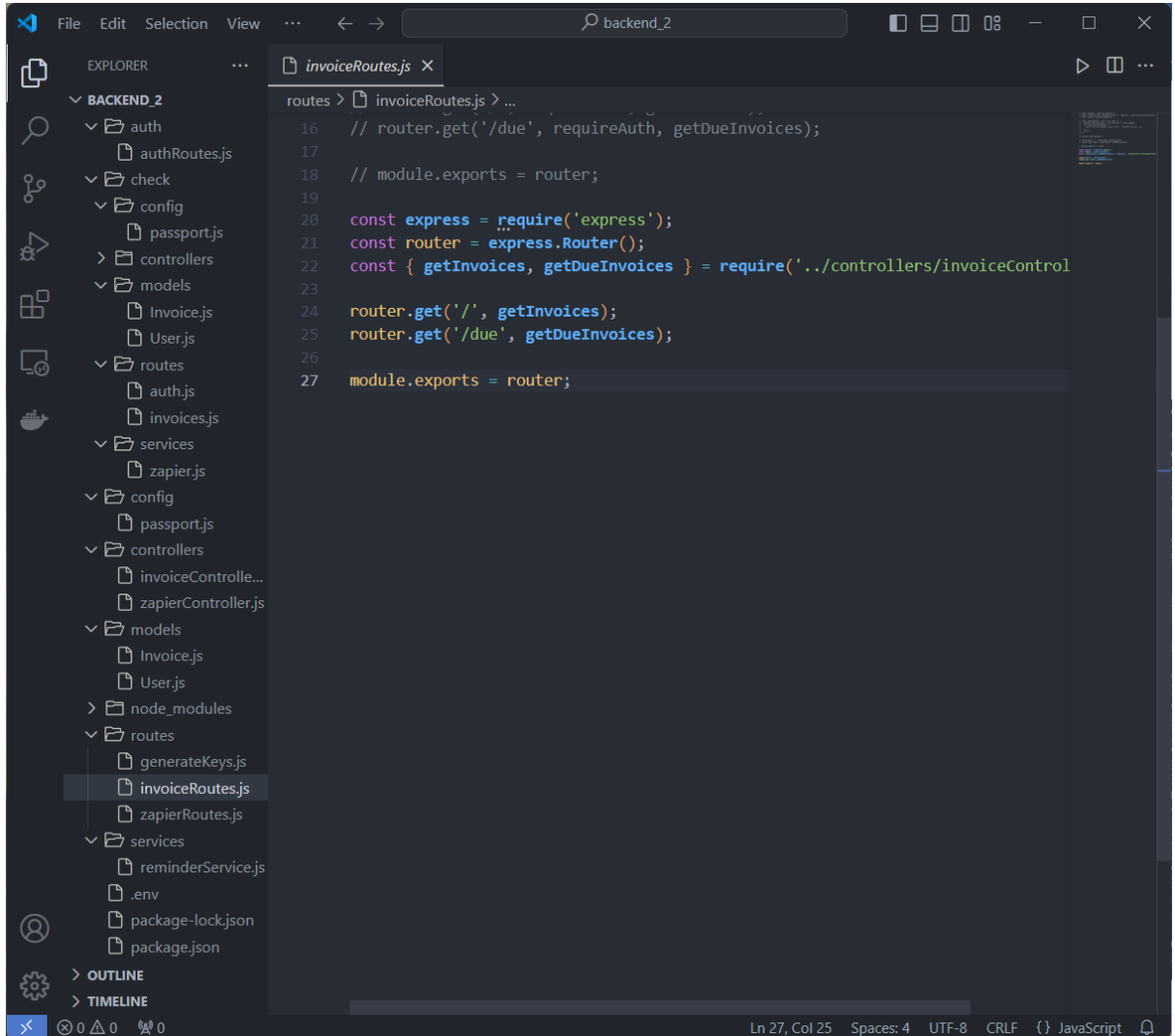


```
1  const express = require('express');
2  const { OAuth2Client } = require('google-auth-library');
3  const axios = require('axios'); // Import Axios
4  const User = require('../models/User'); // Ensure you have a User model
5  const router = express.Router();
6
7  const client = new OAuth2Client(process.env.GOOGLE_CLIENT_ID);
8
9  // POST request handler for Google authentication
10 router.post('/google', async (req, res) => {
11   const { token } = req.body;
12
13   try {
14     const ticket = await client.verifyIdToken({
15       idToken: token,
16       audience: process.env.GOOGLE_CLIENT_ID,
17     });
18
19     const payload = ticket.getPayload();
20     const { sub, email, name, picture } = payload;
21
22     let user = await User.findOne({ googleId: sub });
23
24     if (!user) {
25       user = new User({
26         googleId: sub,
27         email,
28         name,
29         picture
30       });
31       await user.save();
32     }
33
34     req.session.userId = user.id;
35     res.status(200).json(user);
36   } catch (error) {
37     res.status(401).json({ message: 'Invalid token' });
38   }
39 });
```

Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

invoiceRoutes.js

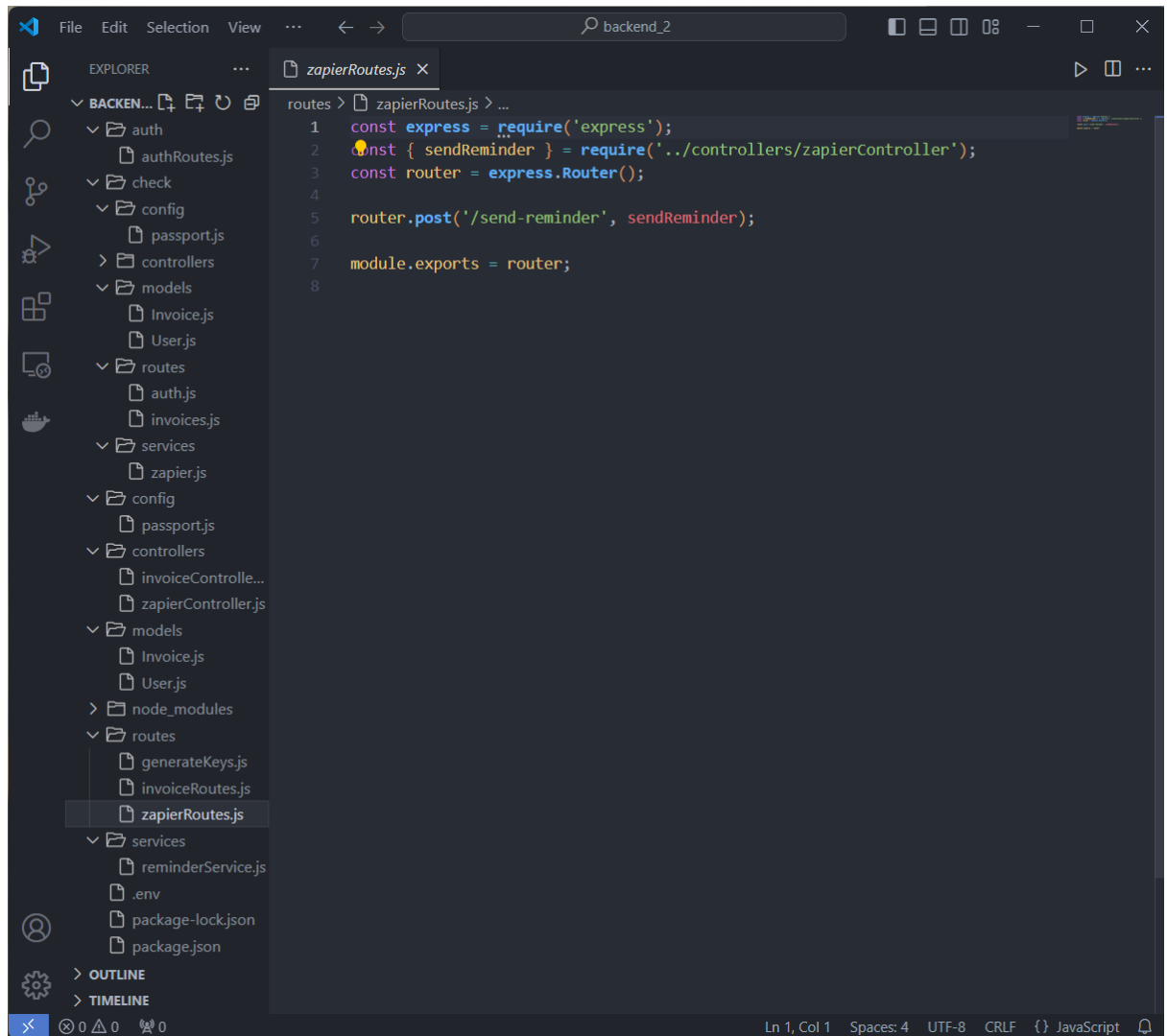


```
16 // router.get('/due', requireAuth, getDueInvoices);
17
18 // module.exports = router;
19
20 const express = require('express');
21 const router = express.Router();
22 const { getInvoices, getDueInvoices } = require('../controllers/invoiceControl
23
24 router.get('/', getInvoices);
25 router.get('/due', getDueInvoices);
26
27 module.exports = router;
```

Name: Thiyagu.E

G-Mail: thiyaguvit@gmail.com

zapierRoutes.js:



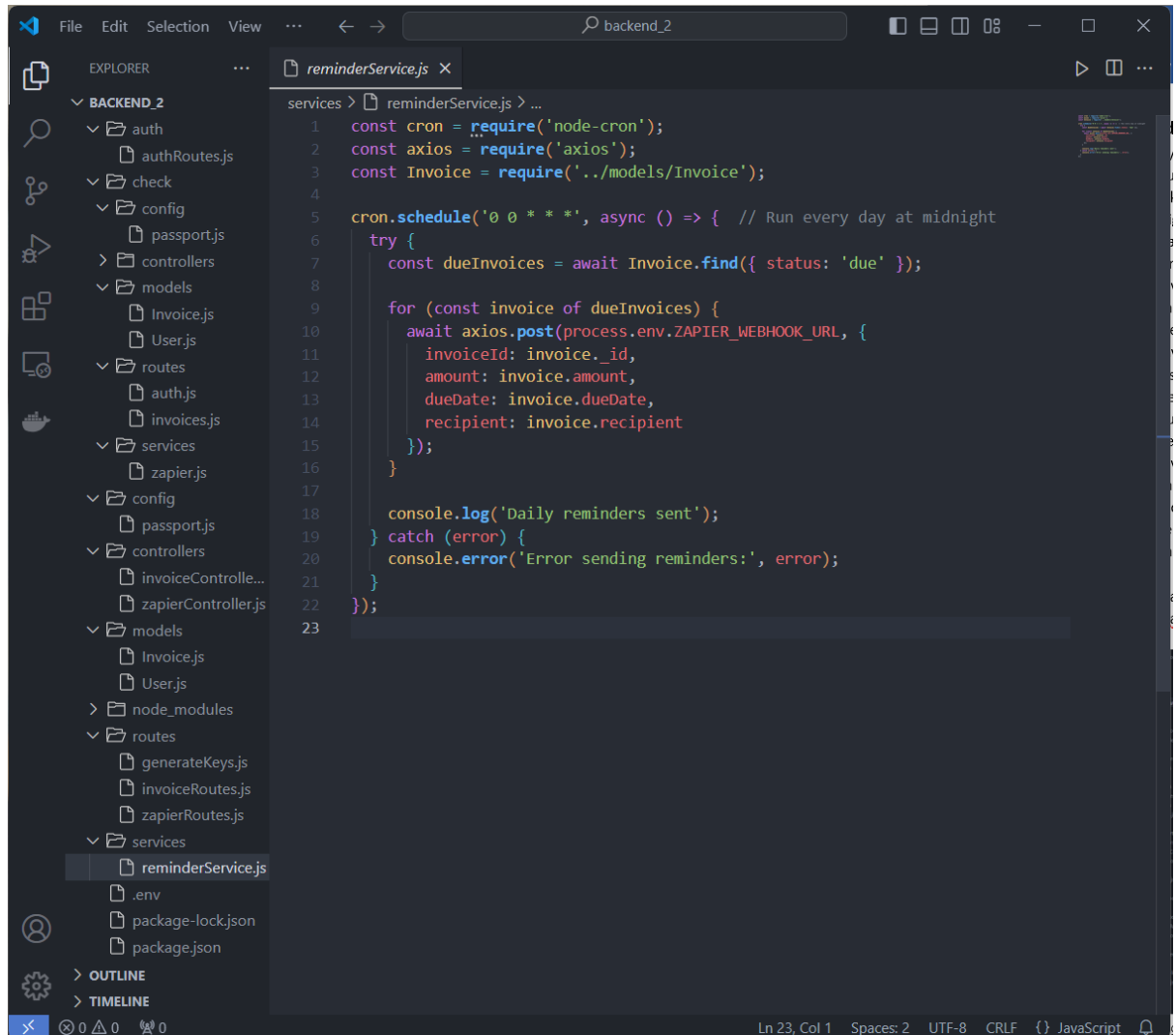
```
1 const express = require('express');
2 const { sendReminder } = require('../controllers/zapierController');
3 const router = express.Router();
4
5 router.post('/send-reminder', sendReminder);
6
7 module.exports = router;
8
```

Name: Thiyagu.E

G-Mail: thiyaguvit@gmail.com

Services

reminderService.js :

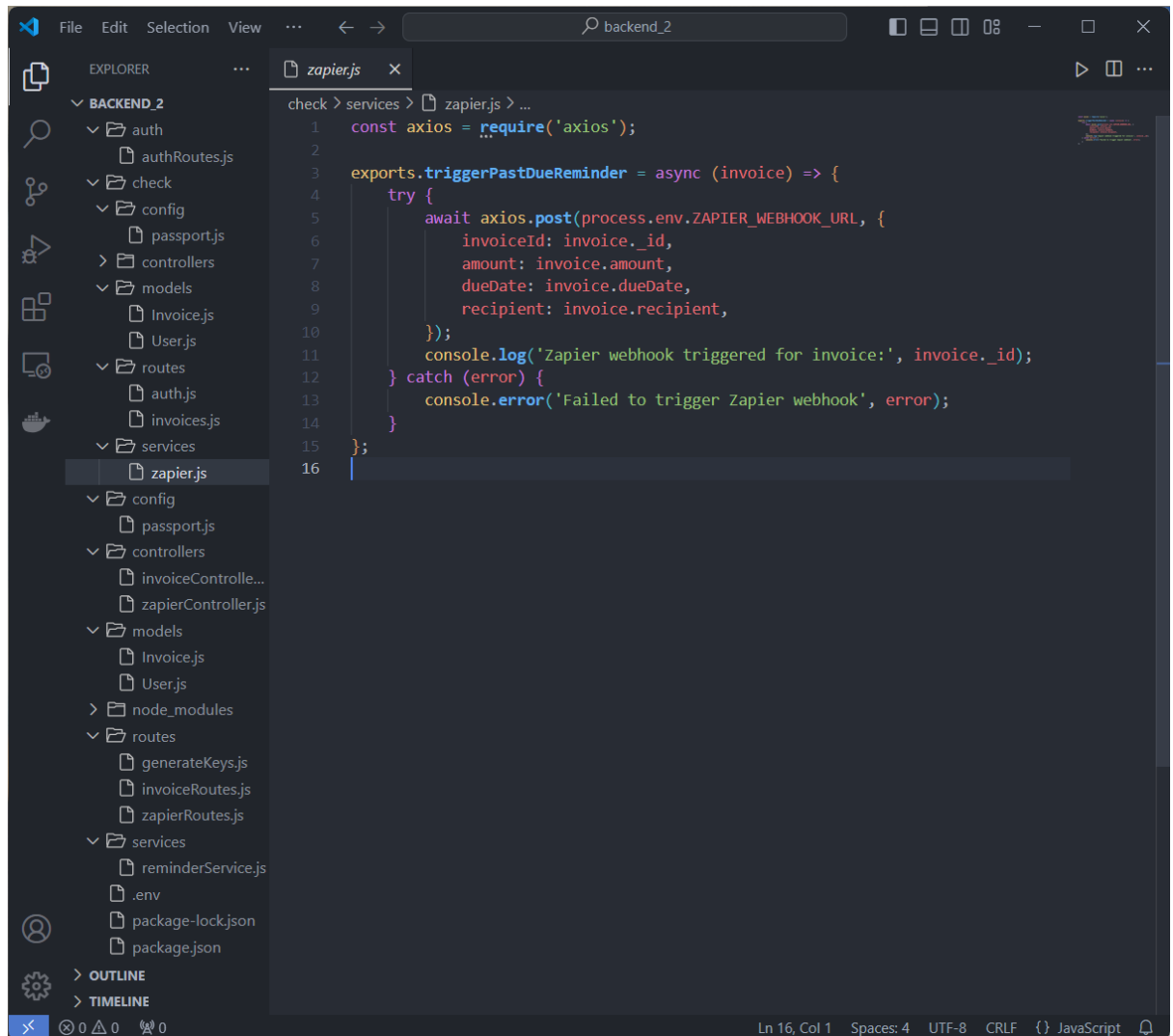


```
1  const cron = require('node-cron');
2  const axios = require('axios');
3  const Invoice = require('../models/Invoice');
4
5  cron.schedule('0 0 * * *', async () => { // Run every day at midnight
6    try {
7      const dueInvoices = await Invoice.find({ status: 'due' });
8
9      for (const invoice of dueInvoices) {
10        await axios.post(process.env.ZAPIER_WEBHOOK_URL, {
11          invoiceId: invoice._id,
12          amount: invoice.amount,
13          dueDate: invoice.dueDate,
14          recipient: invoice.recipient
15        });
16      }
17
18      console.log('Daily reminders sent');
19    } catch (error) {
20      console.error('Error sending reminders:', error);
21    }
22  });
23
```

Name: Thiyaagu.E

G-Mail: thiyaguvit@gmail.com

zapier.js



```
1  const axios = require('axios');
2
3  exports.triggerPastDueReminder = async (invoice) => {
4    try {
5      await axios.post(process.env.ZAPIER_WEBHOOK_URL, {
6        invoiceId: invoice.id,
7        amount: invoice.amount,
8        dueDate: invoice.dueDate,
9        recipient: invoice.recipient,
10     });
11     console.log('Zapier webhook triggered for invoice:', invoice.id);
12   } catch (error) {
13     console.error('Failed to trigger Zapier webhook', error);
14   }
15 };
16
```

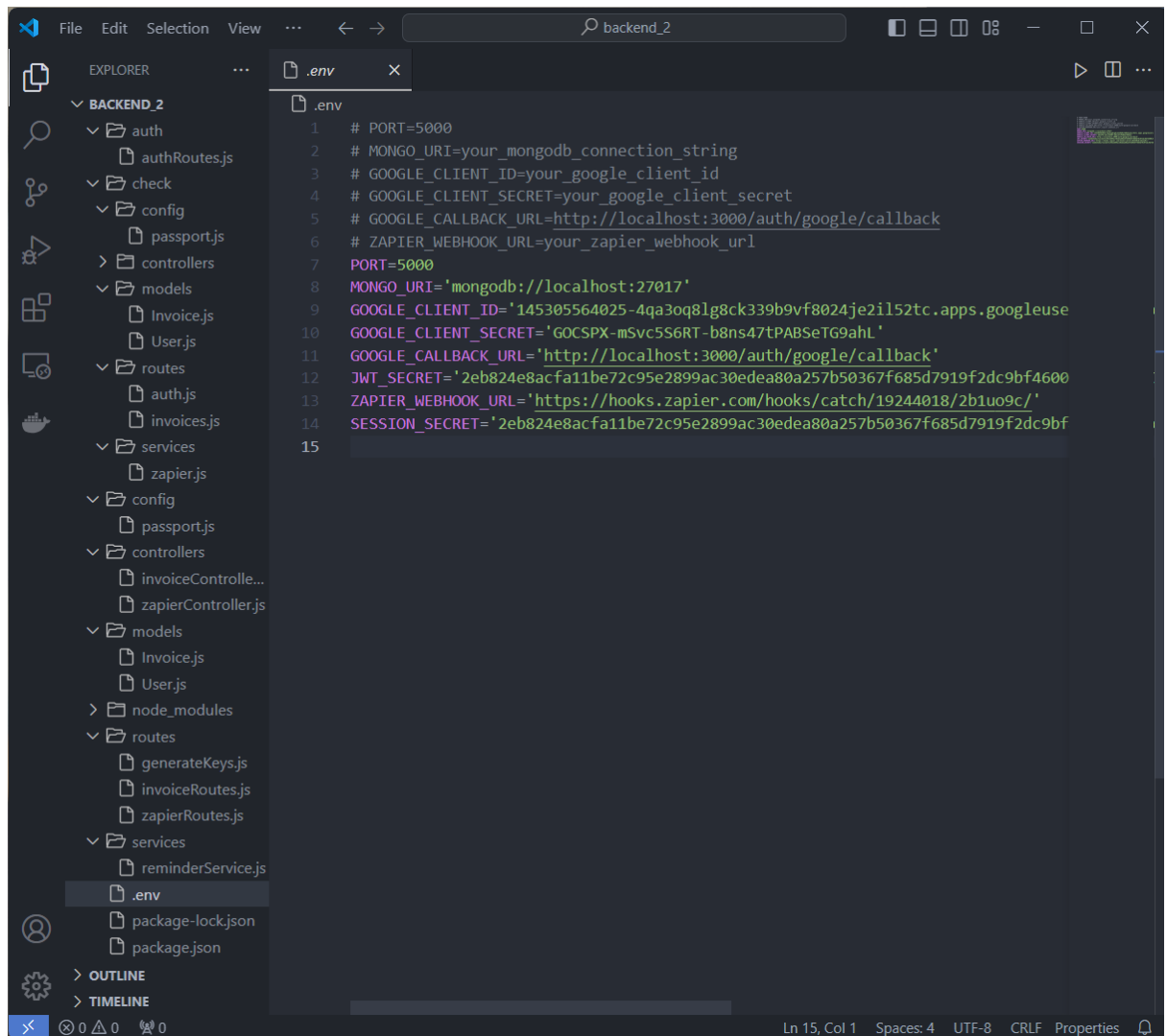
The screenshot shows a VS Code editor window with a project named 'backend_2'. The Explorer sidebar on the left shows the project structure, including folders for 'auth', 'check', 'config', 'controllers', 'models', 'routes', and 'services'. The 'zapier.js' file is selected in the 'services' folder. The main editor area displays the code for 'zapier.js', which defines an asynchronous function 'triggerPastDueReminder' that uses 'axios' to send a POST request to a webhook URL. The code includes error handling with a 'catch' block. The status bar at the bottom indicates the file is at line 16, column 1, using UTF-8 encoding with CRLF line endings.

Name: Thiyagu.E

G-Mail: thiyaguvit@gmail.com

.env:

Environment Variables:



The screenshot shows a Visual Studio Code editor window with the file explorer on the left and the .env file open in the editor. The file explorer shows a project structure for 'BACKEND_2' with various folders like 'auth', 'check', 'config', 'controllers', 'models', 'routes', 'services', and 'config'. The .env file contains the following environment variables:

```
1 # PORT=5000
2 # MONGO_URI=your_mongodb_connection_string
3 # GOOGLE_CLIENT_ID=your_google_client_id
4 # GOOGLE_CLIENT_SECRET=your_google_client_secret
5 # GOOGLE_CALLBACK_URL=http://localhost:3000/auth/google/callback
6 # ZAPIER_WEBHOOK_URL=your_zapier_webhook_url
7 PORT=5000
8 MONGO_URI='mongodb://localhost:27017'
9 GOOGLE_CLIENT_ID='145305564025-4qa3oq8lg8ck339b9vf8024je2il52tc.apps.googleusercontent.com'
10 GOOGLE_CLIENT_SECRET='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'
11 GOOGLE_CALLBACK_URL='http://localhost:3000/auth/google/callback'
12 JWT_SECRET='2eb824e8acfa11be72c95e2899ac30edea80a257b50367f685d7919f2dc9bf4600'
13 ZAPIER_WEBHOOK_URL='https://hooks.zapier.com/hooks/catch/19244018/2b1uo9c/'
14 SESSION_SECRET='2eb824e8acfa11be72c95e2899ac30edea80a257b50367f685d7919f2dc9bf'
15
```

The status bar at the bottom indicates the current position is Line 15, Column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and Properties view.

Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

API Endpoints

1. User Authentication:

- **Endpoint:** /auth/google
- **Method:** GET
- **Description:** Redirects the user to Google for authentication.

2. Invoice Details:

- **Endpoint:** /api/invoices
- **Method:** GET
- **Description:** Retrieves the list of due invoices for the authenticated user.

3. Trigger Zapier Automation:

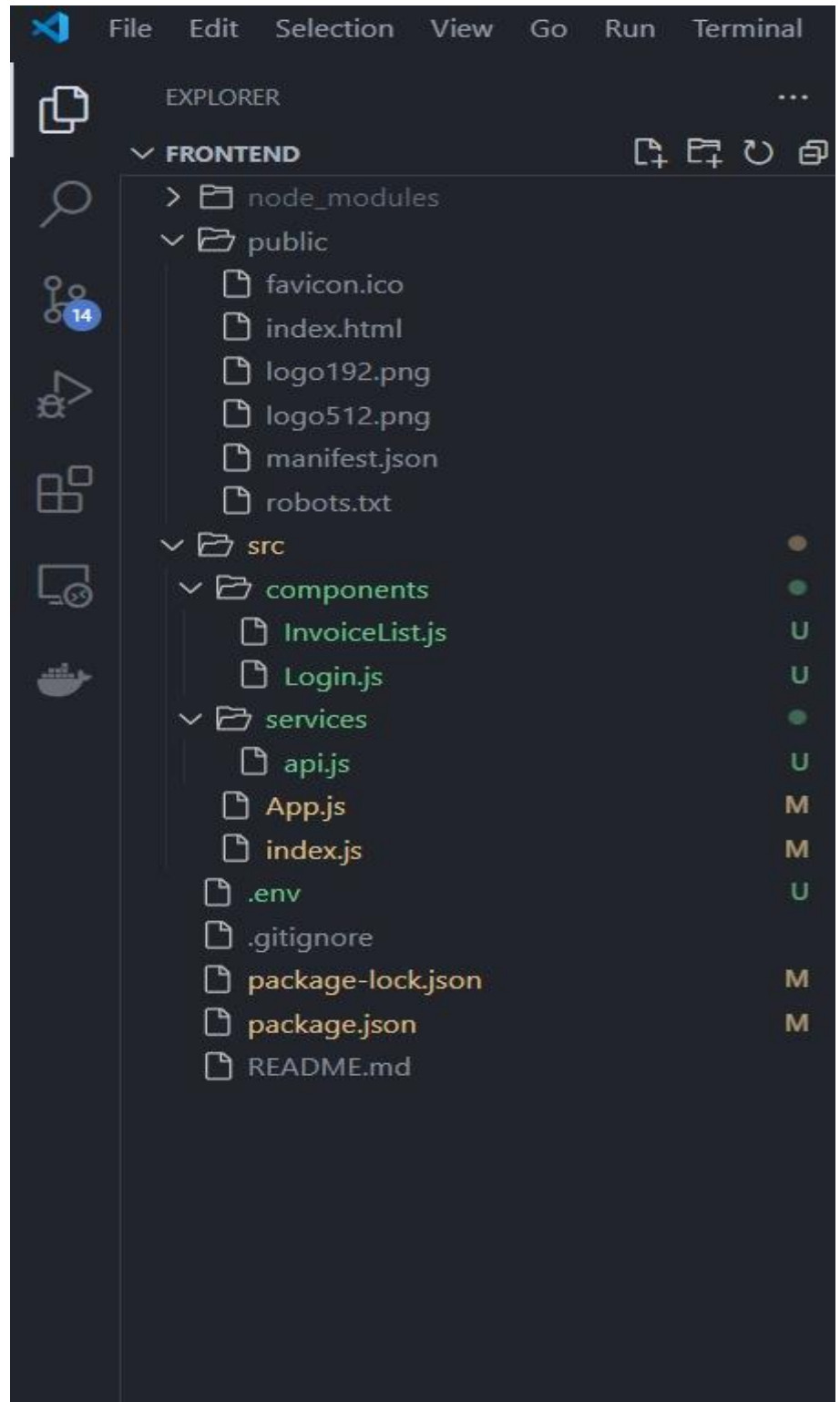
- **Endpoint:** /api/trigger-automation
- **Method:** POST
- **Description:** Triggers the Zapier workflow for past-due invoices.

Frontend Project Structure

Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

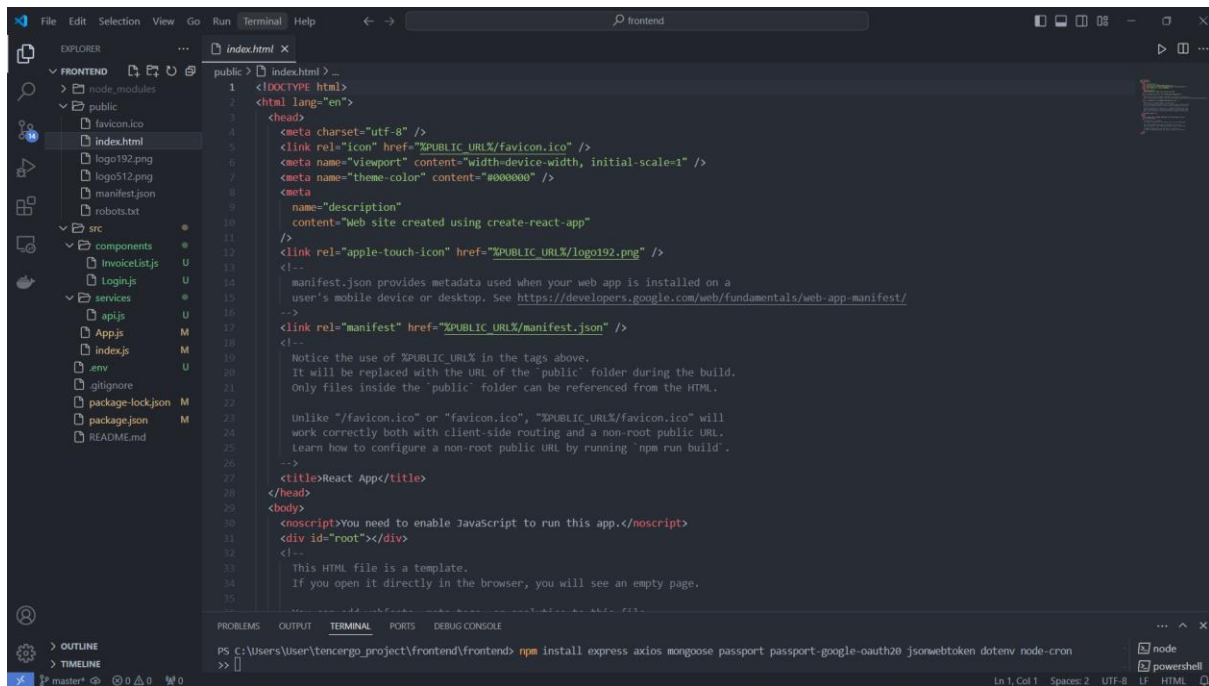
```
├── public/
│   ├── index.html
│   ├── favicon.ico
│   ├── logo192.png
│   ├── logo512.png
│   ├── manifest.json
│   └── robots.txt
├── src/
│   ├── components/
│   │   ├── InvoiceList.js
│   │   └── Login.js
│   ├── services/
│   │   ├── api.js
│   │   └── App.js
│   ├── index.js
│   ├── .env
│   ├── .gitignore
│   ├── package-lock.json
│   └── package.json
```



Name: **Thiyagu.E**

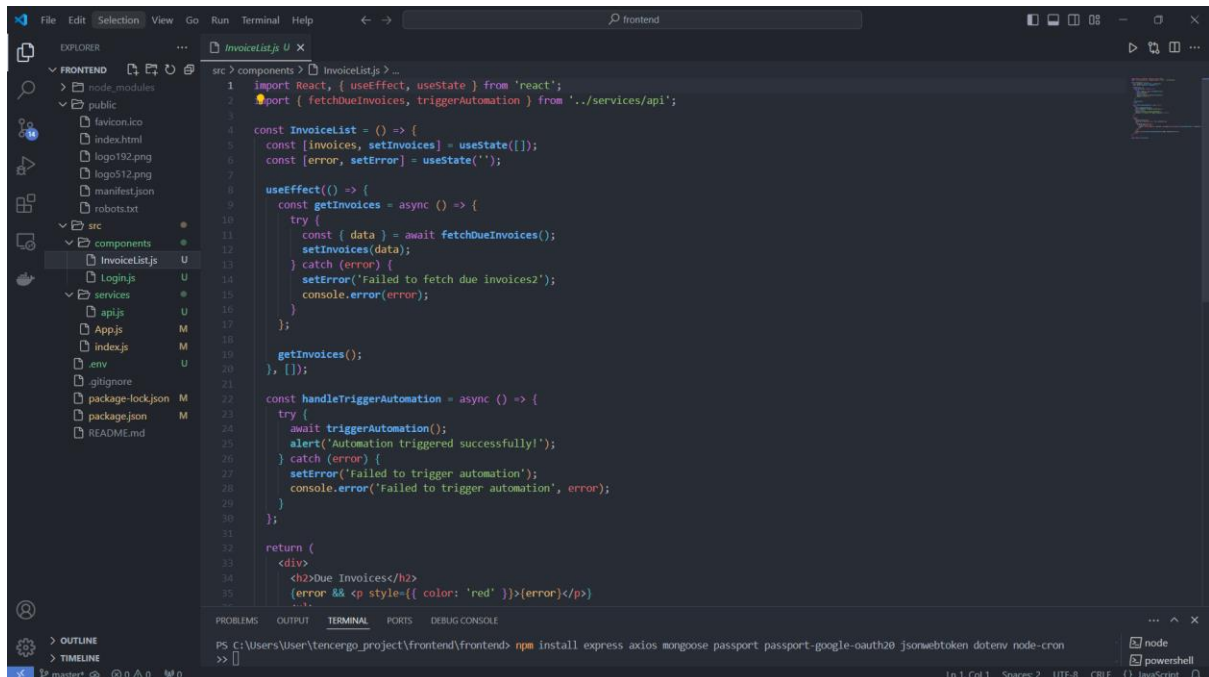
G-Mail: **thiyaguvit@gmail.com**

Frontend/public/index.html:



```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6     <meta name="viewport" content="width=device-width, initial-scale=1" />
7     <meta name="theme-color" content="#000000" />
8     <meta
9       name="description"
10      content="Web site created using create-react-app"
11    />
12    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13    <!--
14      manifest.json provides metadata used when your web app is installed on a
15      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
16    -->
17    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
18    <!--
19      Notice the use of %PUBLIC_URL% in the tags above.
20      It will be replaced with the URL of the 'public' folder during the build.
21      Only files inside the 'public' folder can be referenced from the HTML.
22
23      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
24      work correctly both with client-side routing and a non-root public URL.
25      Learn how to configure a non-root public URL by running `npm run build`.
26    -->
27    <title>React App</title>
28  </head>
29  <body>
30    <noscript>you need to enable JavaScript to run this app.</noscript>
31    <div id="root"></div>
32    <!--
33      This HTML file is a template.
34      If you open it directly in the browser, you will see an empty page.
35    -->
36  </body>
37</html>
```

Src/components/InvoiceList.js

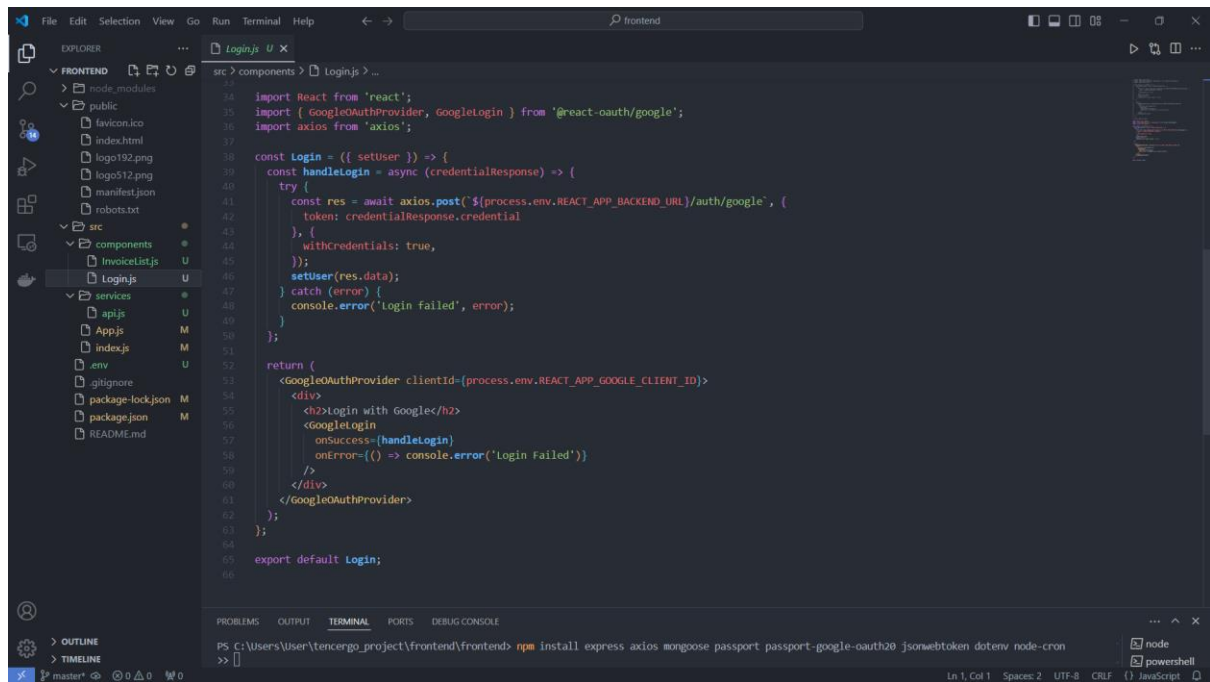


```
1 import React, { useEffect, useState } from 'react';
2 import { fetchDueInvoices, triggerAutomation } from '../services/api';
3
4 const InvoiceList = () => {
5   const [invoices, setInvoices] = useState([]);
6   const [error, setError] = useState('');
7
8   useEffect(() => {
9     const getInvoices = async () => {
10       try {
11         const { data } = await fetchDueInvoices();
12         setInvoices(data);
13       } catch (error) {
14         setError('Failed to fetch due invoices');
15         console.error(error);
16       }
17     };
18     getInvoices();
19   }, []);
20
21   const handleTriggerAutomation = async () => {
22     try {
23       await triggerAutomation();
24       alert('Automation triggered successfully!');
25     } catch (error) {
26       setError('Failed to trigger automation');
27       console.error('Failed to trigger automation', error);
28     }
29   };
30
31   return (
32     <div>
33       <h2>Due Invoices</h2>
34       <div>
35         {error && <p style={{ color: 'red' }}>{error}</p>}
```

Name: **Thiyagu.E**

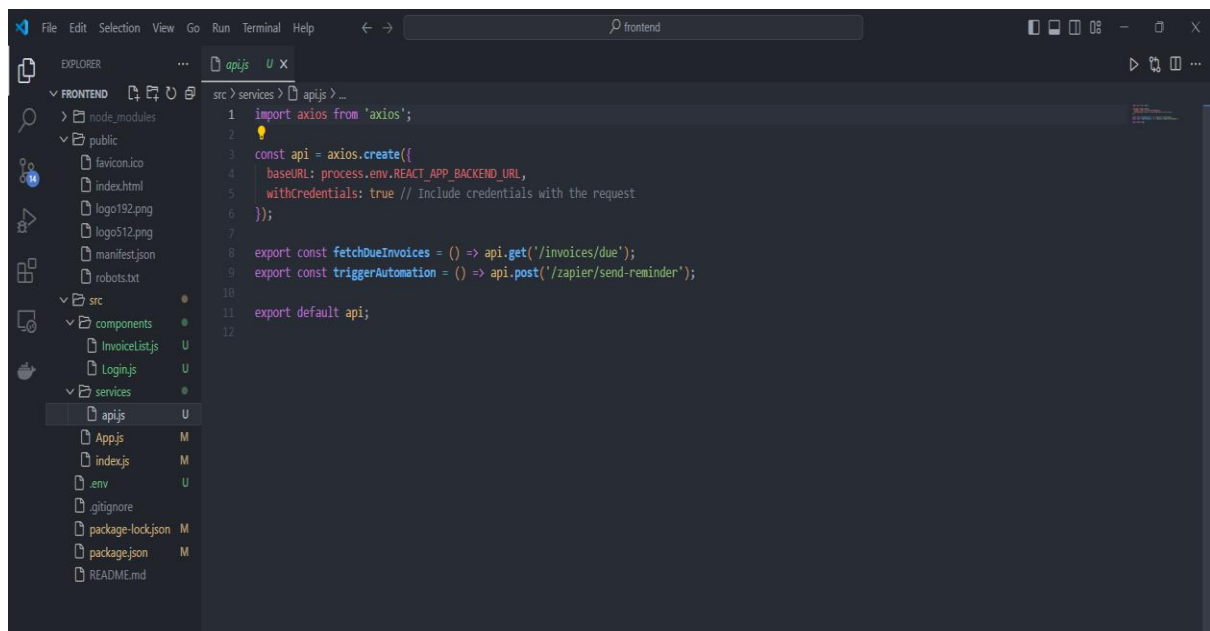
G-Mail: **thiyaguvit@gmail.com**

Login.js



```
14 import React from 'react';
15 import { GoogleAuthProvider, GoogleLogin } from '@react-oauth/google';
16 import axios from 'axios';
17
18 const Login = ({ setUser }) => {
19   const handleLogin = async (credentialResponse) => {
20     try {
21       const res = await axios.post(`${process.env.REACT_APP_BACKEND_URL}/auth/google`, {
22         token: credentialResponse.credential
23       }, {
24         withCredentials: true,
25       });
26       setUser(res.data);
27     } catch (error) {
28       console.error('Login failed', error);
29     }
30   };
31
32   return (
33     <GoogleAuthProvider clientId={process.env.REACT_APP_GOOGLE_CLIENT_ID}>
34     <div>
35       <h2>Login with Google</h2>
36       <GoogleLogin
37         onSuccess={handleLogin}
38         onError={() => console.error('Login Failed')}
39       />
40     </div>
41     </GoogleAuthProvider>
42   );
43 };
44
45 export default Login;
```

services/api.js:

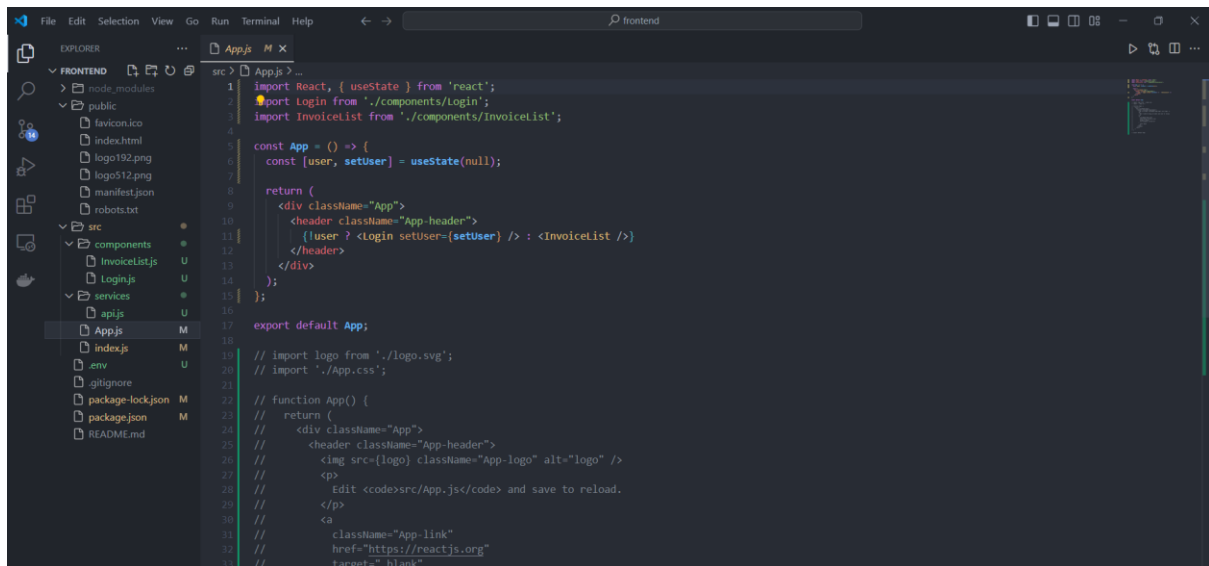


```
1 import axios from 'axios';
2
3 const api = axios.create({
4   baseURL: process.env.REACT_APP_BACKEND_URL,
5   withCredentials: true // Include credentials with the request
6 });
7
8 export const fetchDueInvoices = () => api.get('/invoices/due');
9 export const triggerAutomation = () => api.post('/zapier/send-reminder');
10
11 export default api;
```

Name: Thiyagu.E

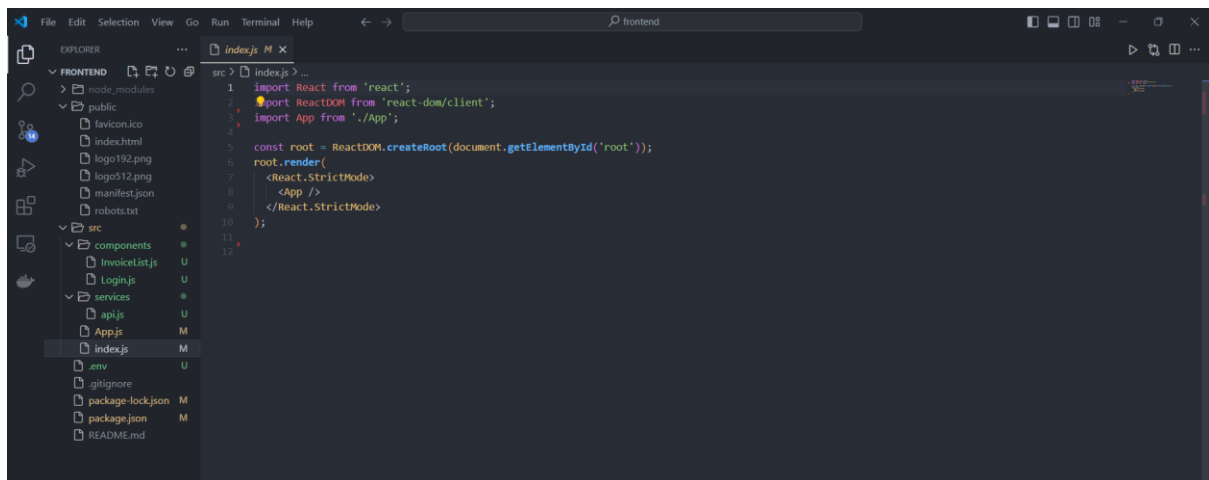
G-Mail: thiyaguvit@gmail.com

App.js



```
1 import React, { useState } from 'react';
2 import Login from './components/Login';
3 import InvoiceList from './components/InvoiceList';
4
5
6 const App = () => {
7   const [user, setUser] = useState(null);
8
9   return (
10     <div className="App">
11       <header className="App-header">
12         {user ? <Login setUser={setUser} /> : <InvoiceList />}
13       </header>
14     </div>
15   );
16
17 export default App;
18
19 // Import logo from './logo.svg';
20 // Import './App.css';
21
22 // function App() {
23 //   return (
24 //     <div className="App">
25 //       <header className="App-header">
26 //         <img src={logo} className="App-logo" alt="logo" />
27 //         <p>
28 //           Edit <code>src/App.js</code> and save to reload.
29 //         </p>
30 //         <a
31 //           className="App-link"
32 //           href="https://reactjs.org"
33 //           target="blank"
```

index.js:

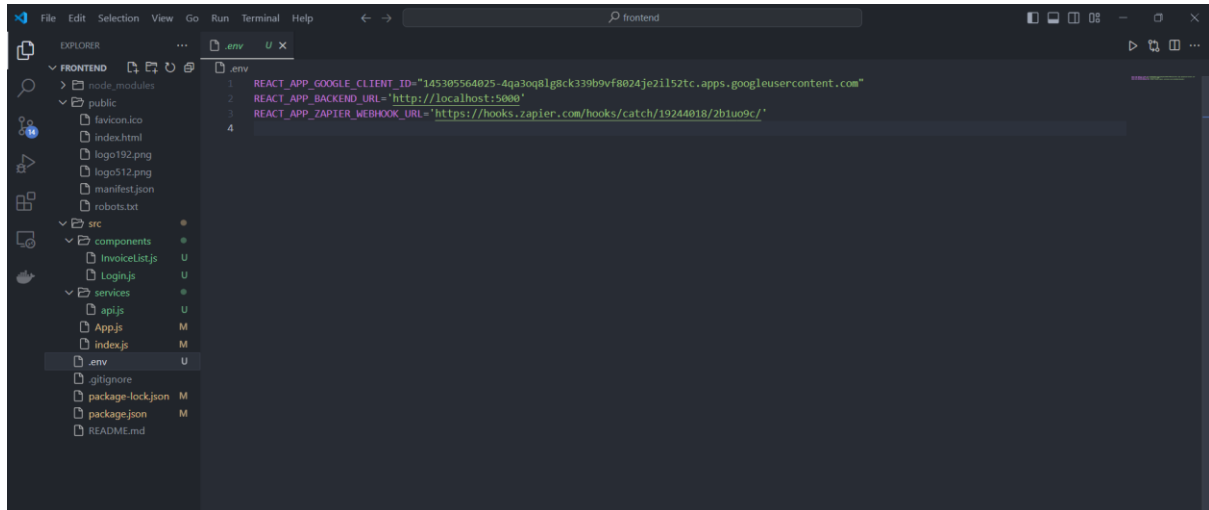


```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import App from './App';
4
5 const root = ReactDOM.createRoot(document.getElementById('root'));
6 root.render(
7   <React.StrictMode>
8     <App />
9   </React.StrictMode>
10 );
```

Name: **Thiyagu.E**

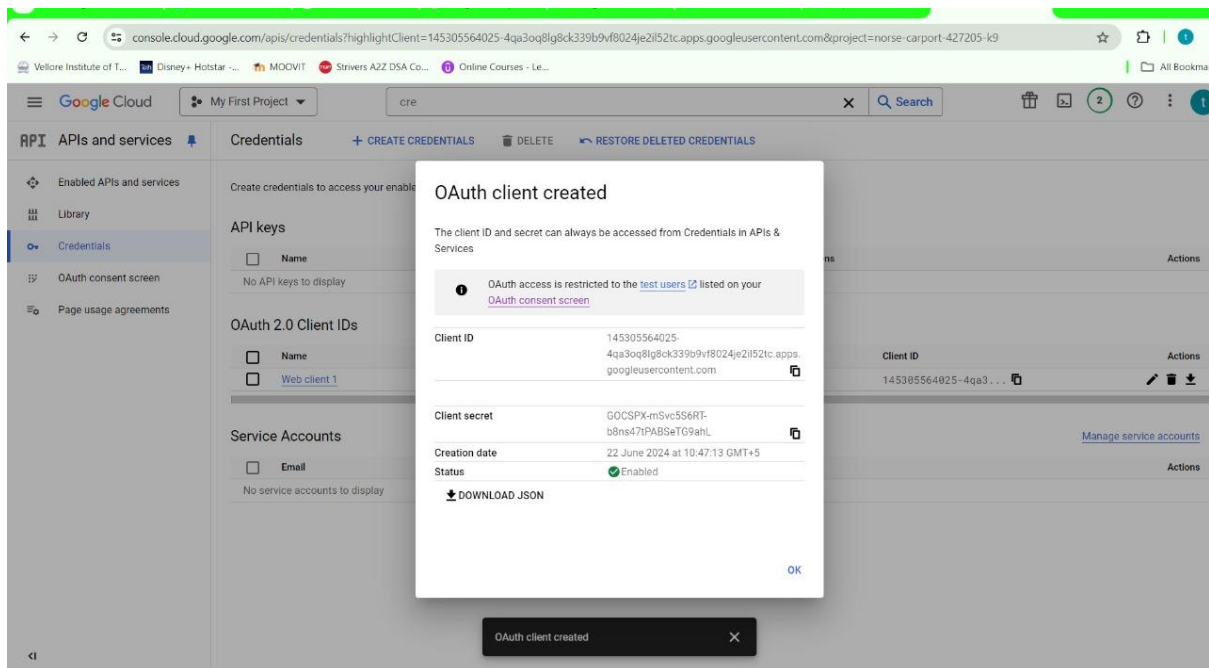
G-Mail: **thiyaguvit@gmail.com**

Env:



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'public', 'src', and 'services'. The code editor is open to a file named '.env' with the following content:

```
1 REACT_APP_GOOGLE_CLIENT_ID="145305564025-4qa3oq8lg8ck339b9vf8024je2il52tc.apps.googleusercontent.com"
2 REACT_APP_BACKEND_URL="http://localhost:5000"
3 REACT_APP_ZAPIER_WEBHOOK_URL="https://hooks.zapier.com/hooks/catch/19244018/2b1uo9c/"
4
```



Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

Step-by-Step Guide to Set Up Zapier Integration

Step 1: Set Up Your Zapier Account

1. Sign Up/Login to Zapier:

- Go to zapier.com and sign up for a Zapier account if you haven't already.
- Log in to your Zapier account.

Step 2: Create a New Zap

1. Create a New Zap:

- Click on the 'Make a Zap!' button in your Zapier dashboard to start creating a new automation.

Step 3: Set Trigger Event (Trigger)

1. Choose Trigger App:

- Search and select "FreshBooks" as your trigger app.
- Select the trigger event based on when you want the zap to initiate. For past-due invoices, you might choose "New Invoice" or "Invoice Updated".
- Authenticate your FreshBooks account with Zapier.

2. Set Trigger Conditions (if applicable):

- Depending on your selection, set up filters to trigger the zap only for invoices that are past their due date. You can filter based on invoice status or due date fields provided by FreshBooks.

Step 4: Set Action Event (Action)

1. Choose Action App:

- Search and select "Gmail" as your action app (or your preferred email service provider).
- Select the action event, such as "Send Email".
- Authenticate your Gmail account with Zapier.

2. Customize Email Template:

- Set up the email template that will be sent as a reminder. You can use data from FreshBooks (like customer name, invoice number, amount due, etc.) using Zapier's placeholders.

Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

Example email template:

csharp

Subject: Reminder: Payment Due for Invoice {{invoice_number}}

Hi {{client_name}},

This is a friendly reminder that invoice {{invoice_number}} for {{amount_due}} is now past due. Please make the payment at your earliest convenience.

Thank you,

[Your Name]

[Your Company Name]

Step 5: Test Your Zap

1. Test Your Zap:

- Once you've set up your trigger and action, run a test to ensure that Zapier correctly integrates FreshBooks with Gmail and processes the data as expected.
- Zapier will use sample data from FreshBooks to simulate the trigger and send a test email through Gmail.

Step 6: Turn On Your Zap

1. Turn On Your Zap:

- After successful testing, turn on your zap to start automating past-due invoice reminders based on your specified conditions in FreshBooks.

Step 7: Monitor and Adjust

1. Monitor Zap Performance:

- Keep an eye on your Zapier dashboard to monitor how your zaps are performing.
- Adjust your zap settings or email template as needed based on feedback or changes in your workflow.

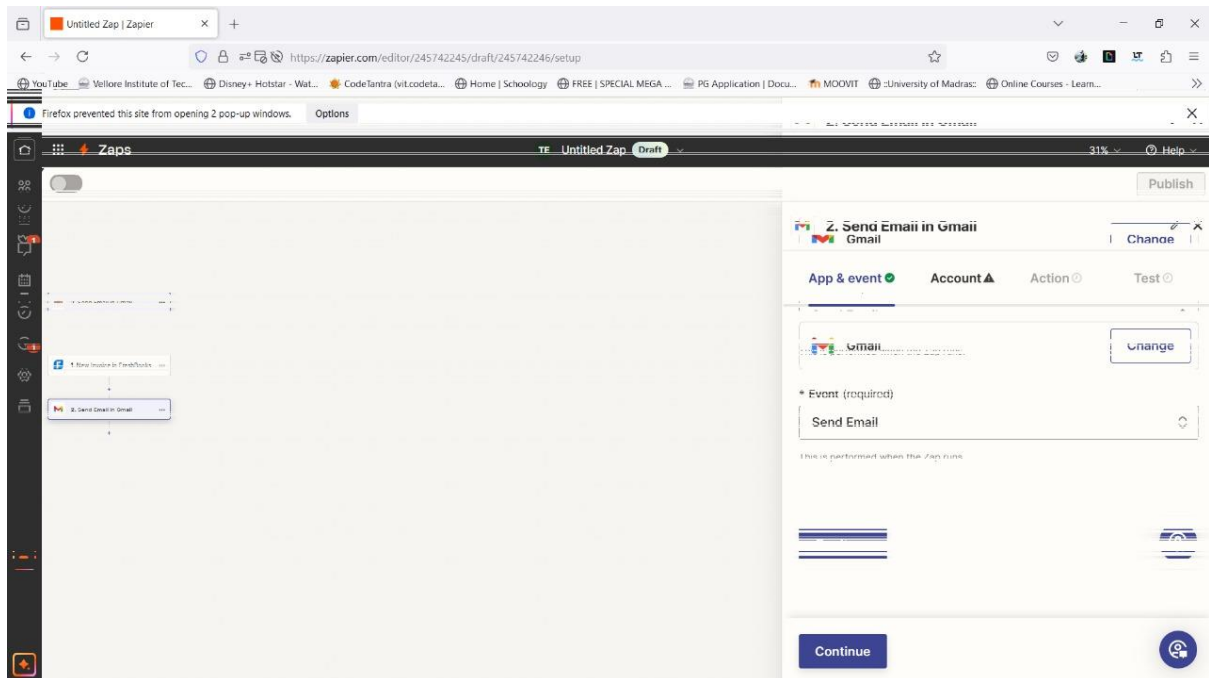
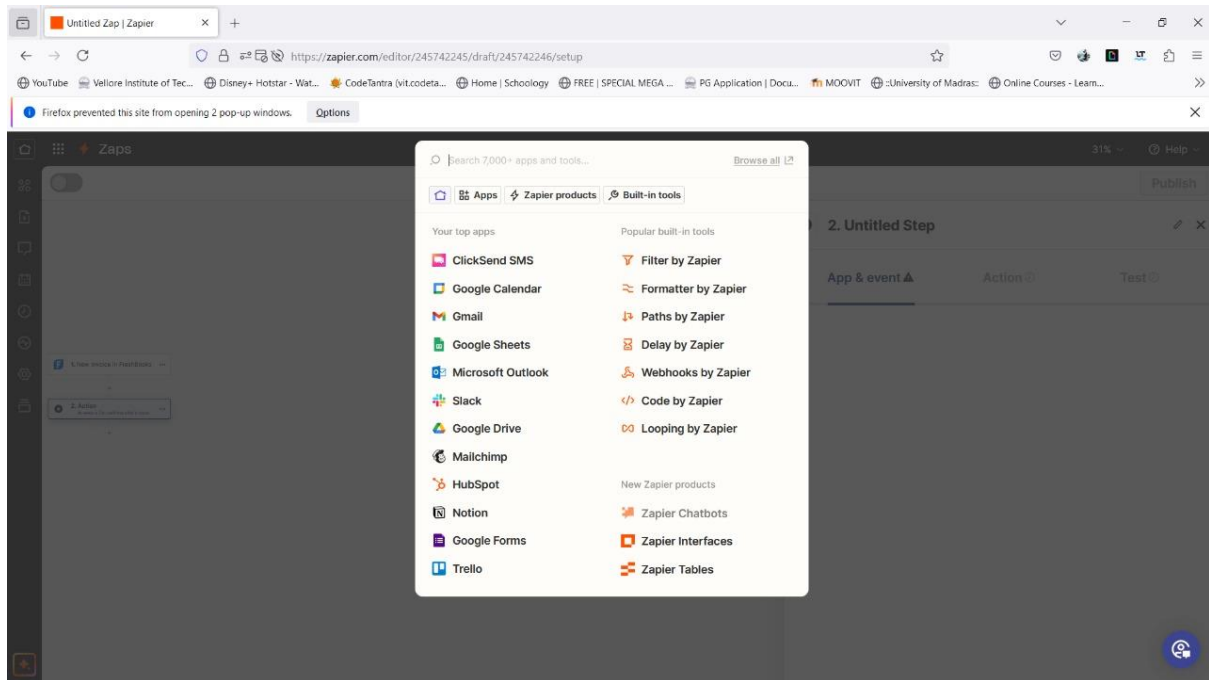
Additional Tips:

- **Advanced Options:** Explore Zapier's advanced features like delays or filters to send reminders at specific times or based on custom criteria.
- **Multi-Step Zaps:** Consider creating multi-step zaps to send escalating reminders or notifications for severely overdue invoices.

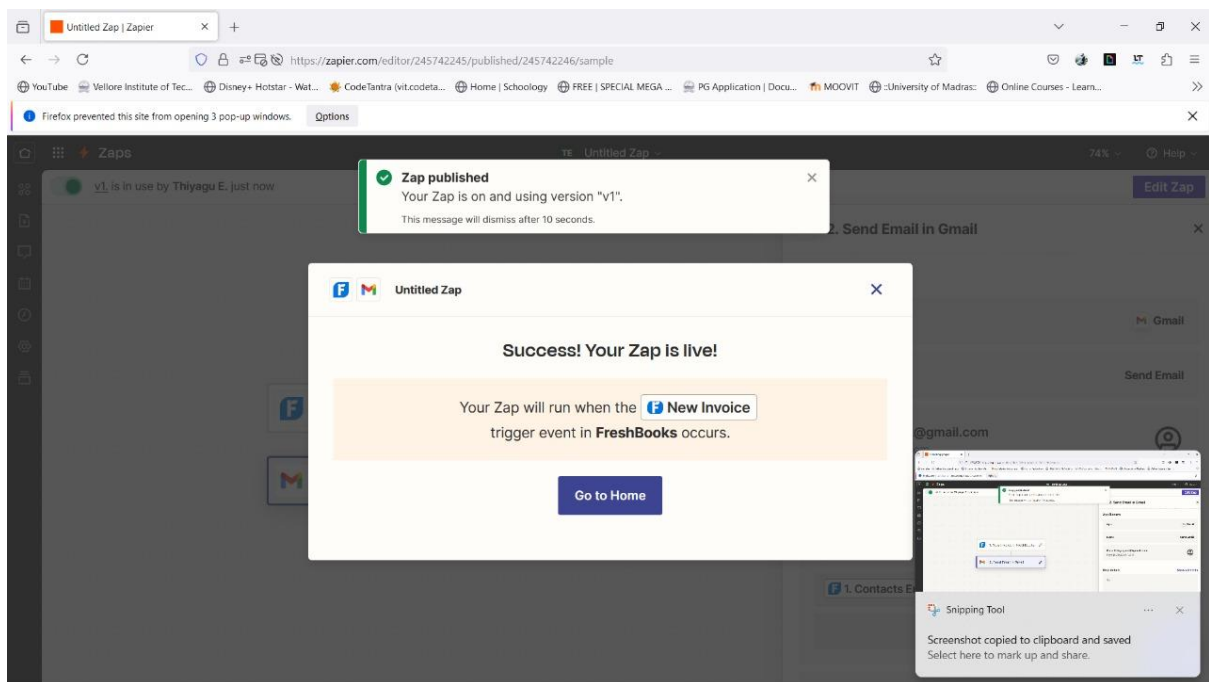
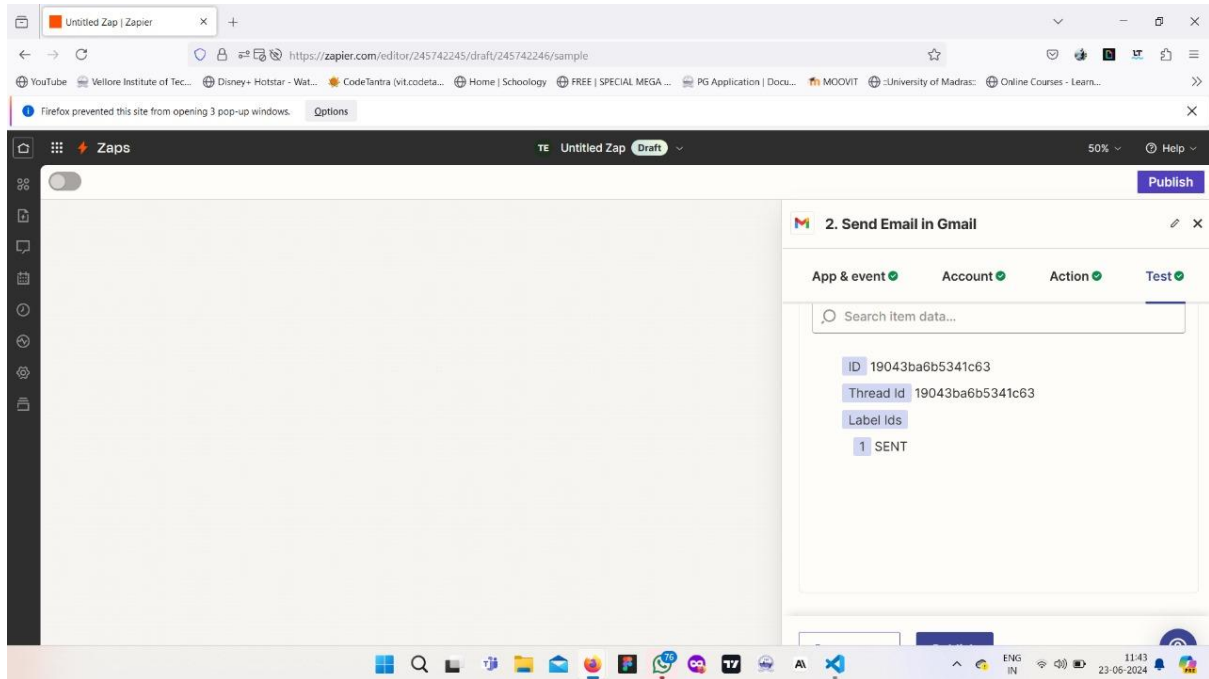
Name: [Thiyagu.E](#)

G-Mail: thiyaguvit@gmail.com

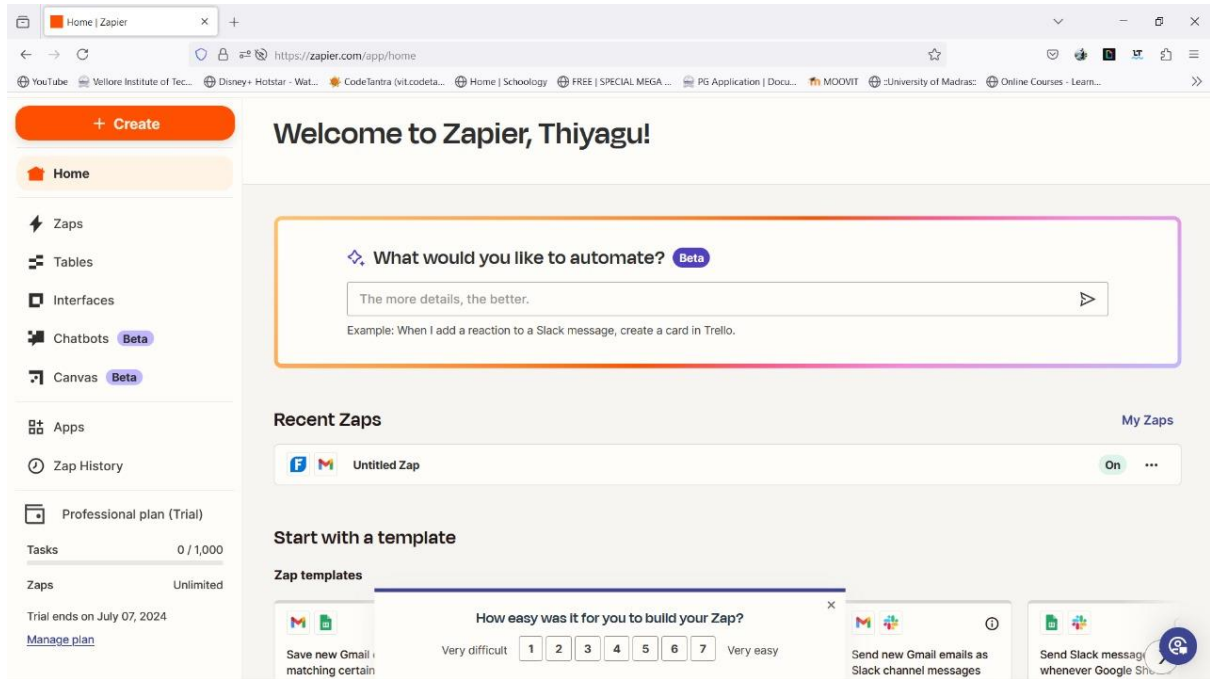
By following these steps, you can effectively automate past-due invoice reminders and follow-up sequences using Zapier, integrating FreshBooks for invoice data and Gmail for sending automated emails to clients. Adjust the settings and email content as necessary to fit your specific business needs and branding guidelines.



Name: **Thiyagu.E**
G-Mail: **thiyaguvit@gmail.com**

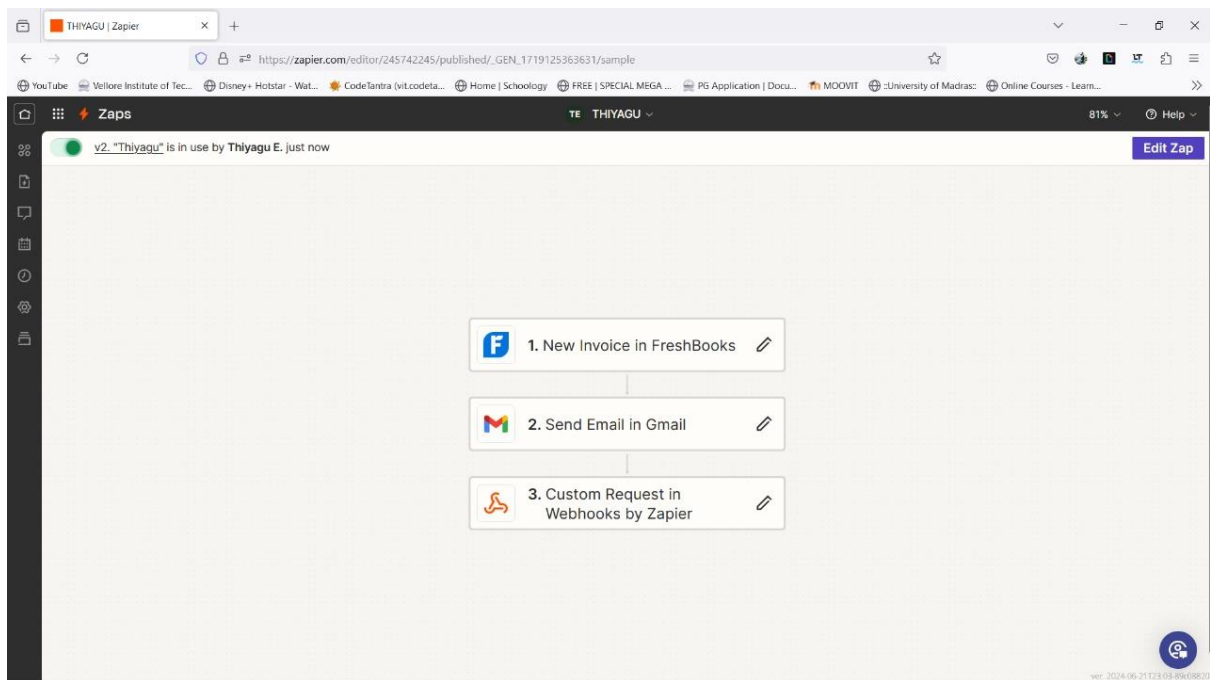


Name: **Thiyagu.E**
G-Mail: **thiyaguvit@gmail.com**



The screenshot shows the Zapier home page in a web browser. The browser's address bar displays "https://zapier.com/app/home". The left sidebar contains a navigation menu with options: Home, Zaps, Tables, Interfaces, Chatbots (Beta), Canvas (Beta), Apps, Zap History, and Professional plan (Trial). The main content area features a "Welcome to Zapier, Thiyagu!" message, a "What would you like to automate?" prompt, and a "Recent Zaps" section showing an "Untitled Zap". Below this is a "Start with a template" section with a "Zap templates" tab. A survey pop-up titled "How easy was it for you to build your Zap?" is visible, with a rating scale from 1 to 7. The bottom of the page shows a trial end date of July 07, 2024, and a "Manage plan" link.

]



The screenshot shows the Zapier Zap editor interface. The browser's address bar displays "https://zapier.com/editor/245742245/published/_GEN_1719125363631/sample". The top bar shows the user's name "TE THYAGU" and a "Help" button. The left sidebar contains a "Zaps" section with a "v2 'Thiyagu'" status. The main content area displays a workflow with three steps: 1. New Invoice in FreshBooks, 2. Send Email in Gmail, and 3. Custom Request in Webhooks by Zapier. The bottom right corner shows a version number "ver: 2024.06.21T12:53:53.000000000".

G-Mail: thiyaguvit@gmail.com

<https://zapier.com/editor/245742245/draft/245742245/sample>

⚙️

🔌

📅

🗨️

📅

🕒

🔄

⚙️

📄

🔌 Zaps

TE Untitled Zap Draft

31%

Help

Publish

F

1. New Invoice in FreshBooks

✎

✕

App & event ✓

Account ✓

Trigger ✓

Test ✓

✓

We found records in your FreshBooks account. We will load up to 3 most recent records, that have not appeared previously. [Learn more about test records.](#)

🔍 Search

invoice A

original record pulled on Jun 23, 2024

Accountid p7qJAe

Accounting Systemid p7qJAe

Address

amount

Amount Amount 10000.00

Amount Code INR

Auto Bill false

Autobill Status

Basecampid 0

City

Code

Contacts

Find new records

✎

Continue with selected record

Name: Thiylagu.E
G-Mail: thiylaguvit@gmail.com

Sample invoice template: freshbook.com:

thiyagu's Company
Owner

Dashboard

Clients

Estimates

Invoices

- Recurring Templates
- Retainers

Payments

Expenses

Projects

Time Tracking

Accounting

Reports

Apps

Team Members

Items and Services

Bank Connections

F

There are 30 days left in your trial. Upgrade Account

TE

Recurring Templates

New Recurring Template

All Recurring Templates

Search

| Client / Recurring Template Number | Last Issued | Frequency / Duration | Amount / Status |
|---|-------------|-------------------------|-------------------------------|
| <input type="checkbox"/> name1 00000008969 | 2024-06-23 | Every Month Infinite | Rs.10,000.00 INR Auto-Sent |

1-1 of 1

View Archived Recurring Templates
or deleted

thiyagu's Company
Owner

Dashboard

Clients

Estimates

Invoices

- Recurring Templates
- Retainers

Payments

Expenses

Projects

Time Tracking

Accounting

Reports

Apps

Team Members

Items and Services

Bank Connections

F

There are 30 days left in your trial. Upgrade Account


TE

Recurring Templates

Recurring Template 00000008969

More Actions Edit

Automatically Sent This template automatically sends an invoice every month.



thiyagu's Company
8825952624

Billed To
John Doe
name1
123 Main Street Anytown, USA

Date of Issue
2024-07-23
Due Date
2024-08-22

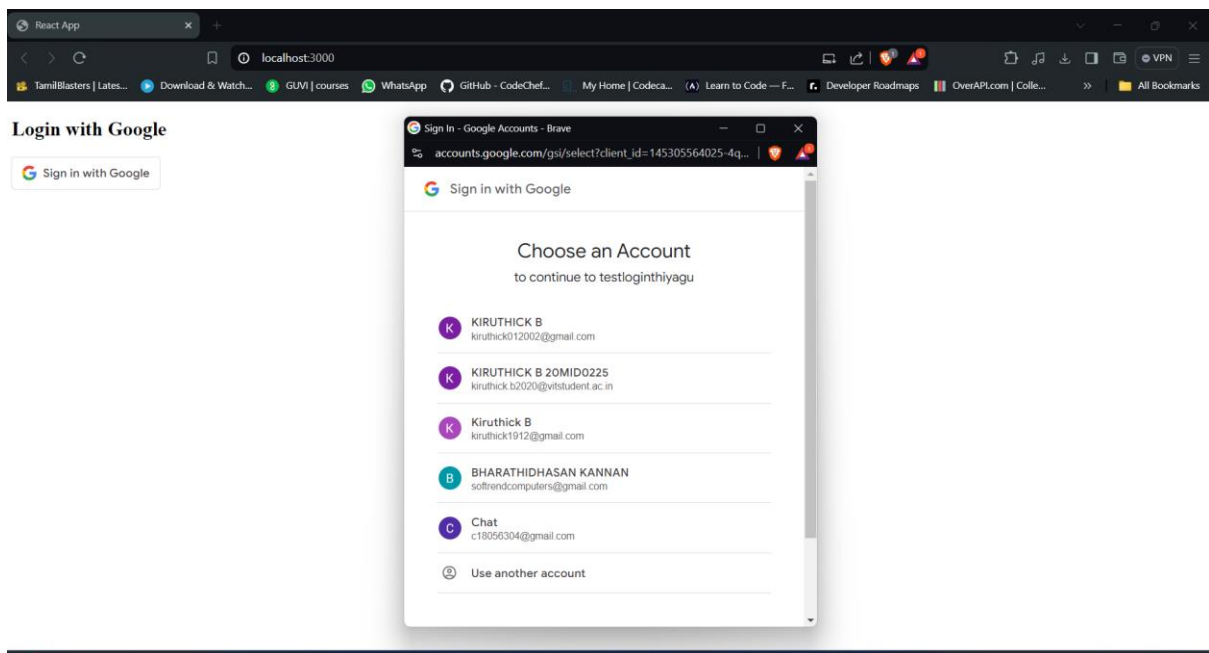
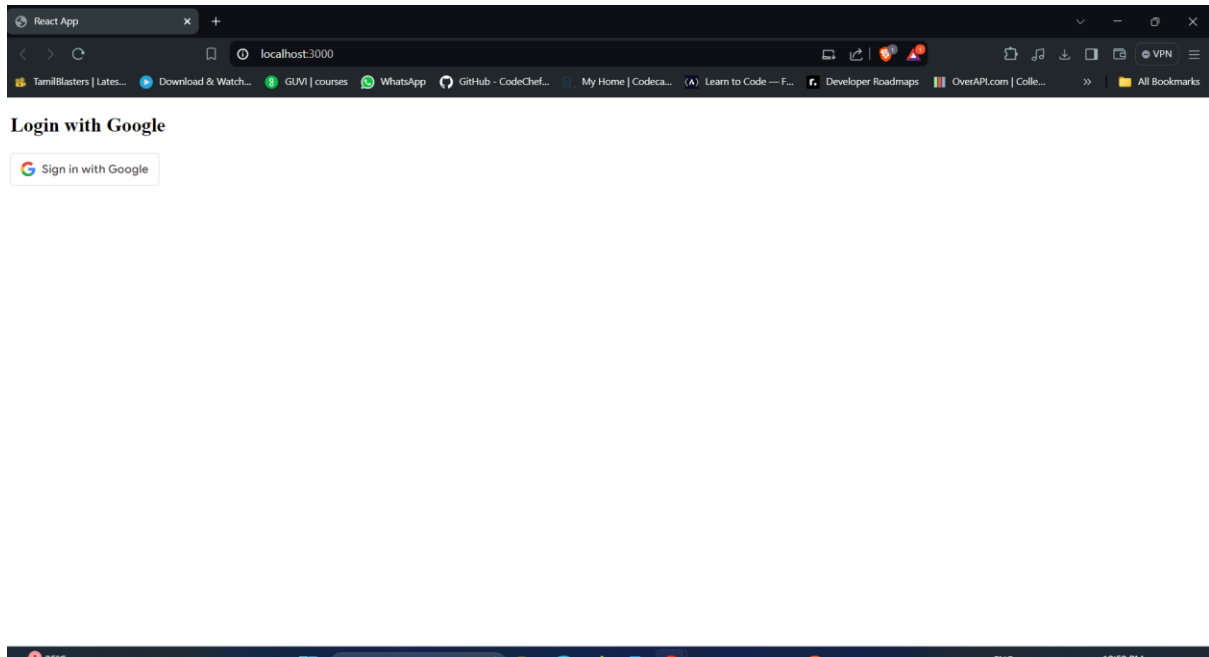
Invoice Number
0000002

Amount Due (INR)
Rs.10,000.00

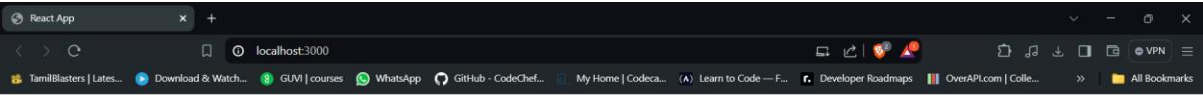
| Description | Rate | Qty | Line Total |
|-----------------|-------------|-----|-------------|
| Web Development | Rs.333.3333 | 15 | Rs.5,000.00 |

Name: [Thiyagu.E](#)
G-Mail: thiyaguvit@gmail.com

Project Output:



Name: Thiyagu.E
G-Mail: thiyaguvit@gmail.com



Due Invoices

- Amount: 75.75 | Due Date: 30/06/2024 | Recipient: Client C
- Amount: 500 | Due Date: 05/07/2024 | Recipient: Client E

Trigger Automation



Due Invoices

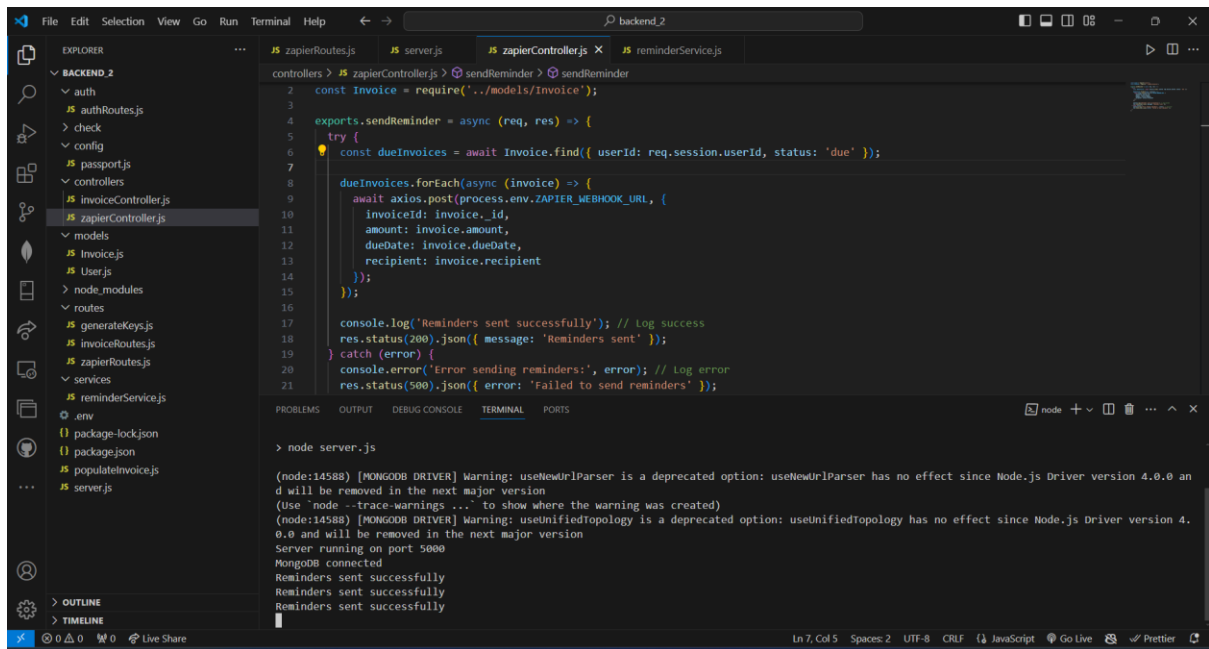
- Amount: 75.75 | Due Date: 30/06/2024 | Recipient: Client C
- Amount: 500 | Due Date: 05/07/2024 | Recipient: Client E

Trigger Automation



Name: Thiyaagu.E

G-Mail: thiyaaguvit@gmail.com



The screenshot shows a VS Code editor window with a project named 'backend_2'. The Explorer sidebar on the left shows the file structure, including 'controllers', 'models', 'routes', and 'services'. The main editor area displays the 'zapierController.js' file, which contains a REST client for the 'sendReminder' endpoint. The code defines a REST client with a base URL of 'http://localhost:3000' and a path of '/api/sendReminder'. The client is configured with a 'POST' method and a 'Content-Type' of 'application/json'. The request body is a JSON object with 'invoiceId', 'amount', 'dueDate', and 'recipient' fields. The response is a JSON object with 'message' and 'status' fields. The terminal at the bottom shows the output of running 'node server.js', which includes warnings about deprecated options and a successful message: 'Reminders sent successfully'.

```
const Invoice = require('../models/Invoice');

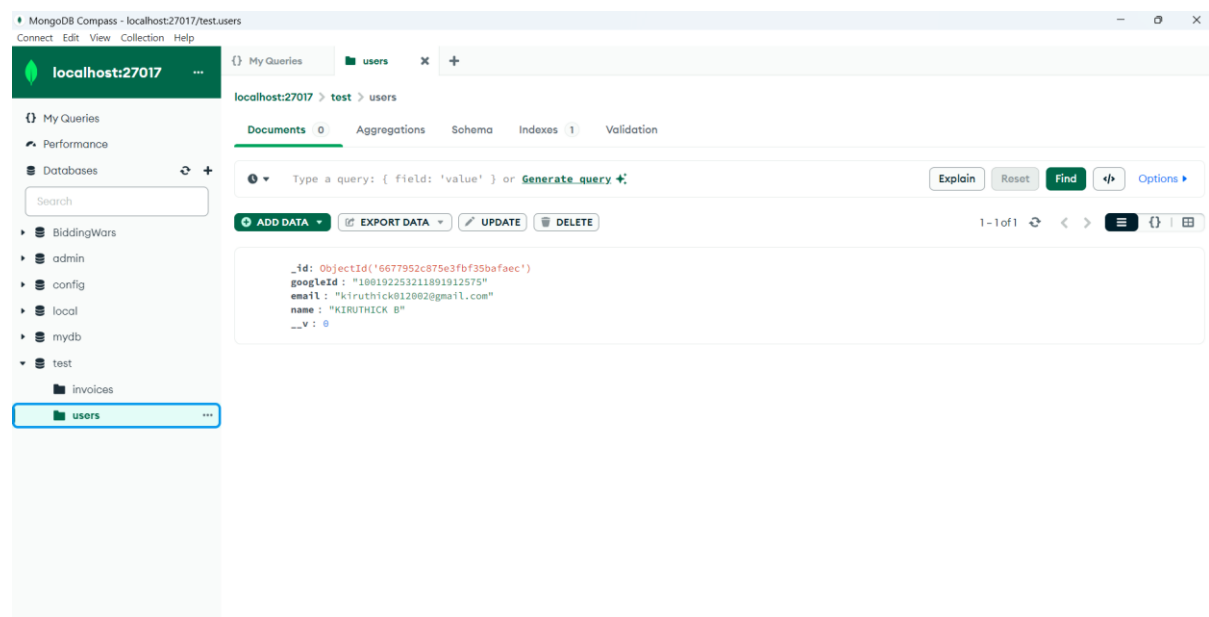
exports.sendReminder = async (req, res) => {
  try {
    const dueInvoices = await Invoice.find({ userId: req.session.userId, status: 'due' });

    dueInvoices.forEach(async (invoice) => {
      await axios.post(process.env.ZAPIER_WEBHOOK_URL, {
        invoiceId: invoice_id,
        amount: invoice.amount,
        dueDate: invoice.dueDate,
        recipient: invoice.recipient
      });
    });

    console.log('Reminders sent successfully'); // Log success
    res.status(200).json({ message: 'Reminders sent' });
  } catch (error) {
    console.error('Error sending reminders:', error); // Log error
    res.status(500).json({ error: 'Failed to send reminders' });
  }
}
```

```
> node server.js

(node:14588) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and
d will be removed in the next major version
(node:14588) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.
0.0 and will be removed in the next major version
Server running on port 5000
MongoDB connected
Reminders sent successfully
Reminders sent successfully
Reminders sent successfully
```



Name: **Thiyagu.E**

G-Mail: **thiyaguvit@gmail.com**

MongoDB Compass - localhost:27017/test:invoices

Connect Edit View Collection Help

localhost:27017

My Queries Performance Databases Search

BiddingWars admin config local mydb test

Invoices

users

localhost:27017 > test > Invoices

Documents 0 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

EXPLAIN RESET FIND Options

ADD DATA EXPORT DATA UPDATE DELETE

1 - 5 of 5

```
{
  "_id": ObjectId('6677b4abe723752215e8411'),
  "userId": ObjectId('6677952c875e3fbf35bafaec'),
  "amount": 150.5,
  "dueDate": "2024-07-01T00:00:00.000+00:00",
  "recipient": "Client A",
  "status": "past-due",
  "__v": 0
}
```

```
{
  "_id": ObjectId('6677b4abe723752215e8412'),
  "userId": ObjectId('6677952c875e3fbf35bafaec'),
  "amount": 200,
  "dueDate": "2024-06-25T00:00:00.000+00:00",
  "recipient": "Client B",
  "status": "past-due",
  "__v": 0
}
```

```
{
  "_id": ObjectId('6677b4abe723752215e8413'),
  "userId": ObjectId('6677952c875e3fbf35bafaec'),
  "amount": 75.75,
  "dueDate": "2024-06-30T00:00:00.000+00:00",
  "recipient": "Client C",
  "status": "due",
  "__v": 0
}
```

mail.google.com/mail/u/0/?tab=rm&ogbl#sent/FMfcgZQVxHhSrchHnSdrRSDxNhfMZC

Vellore Institute of T... Disney+ Hotstar ... MOOVIT Strivers A2Z DSA Co... Online Courses - Le...

Gmail in:sent

1 of 46

000000100000001 [inbox x](#)

Thiyagu.E <thiyaguvit@gmail.com>
to thiyagusunbeam, thiyaguvit

Pay the bill soon sir

11:43 (7 hours ago)