Yes, I remember the step-by-step guide I provided earlier for creating the **Email Management Program** using **Node.js**. Here's a quick recap and reorganization for your reference:

### **Project Overview**

You are building a web application to:

1. Authenticate users via **Google Gmail API** and **Microsoft Graph API**.
2. Allow users to **read, send, and reply to emails**.
3. Manage emails (e.g., moving emails from spam/promotions to the inbox).
4. Store metadata (like sender, subject, timestamp) in a **MongoDB database**.

### **Steps to Build the Project**

#### **1. Project Setup**

**Create a new project directory:** mkdir email-management-app && cd email-management-app

npm init -y

**Install necessary dependencies:** npm install express mongoose googleapis @microsoft/microsoft-graph-client dotenv body-parser ejs

**Directory structure:** project/

├── public/ # Static files (CSS/JS)

├── views/ # EJS templates for frontend

├── routes/ # API routes

├── src/ # Service functions (Gmail, Microsoft)

├── app.js # Main server file

├── .env # Environment variables

└── package.json

#### **2. Configure .env File**

Add sensitive credentials to the .env file (don’t forget to include .env in .gitignore):

MONGO\_URI=mongodb://localhost:27017/emailManagement

GMAIL\_CLIENT\_ID=your-gmail-client-id

GMAIL\_CLIENT\_SECRET=your-gmail-client-secret

GMAIL\_REDIRECT\_URI=http://localhost:3000/auth/google/callback

MICROSOFT\_CLIENT\_ID=your-microsoft-client-id

MICROSOFT\_CLIENT\_SECRET=your-microsoft-client-secret

MICROSOFT\_REDIRECT\_URI=http://localhost:3000/auth/microsoft/callback

#### **3. Backend Implementation**

##### **a. Connect to MongoDB**

In your app.js, set up MongoDB:

const mongoose = require('mongoose');

mongoose.connect(process.env.MONGO\_URI)

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

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

##### **b. Gmail Service**

Create a src/gmailService.js to handle Gmail API logic:

const { google } = require('googleapis');

const OAuth2 = google.auth.OAuth2;

const oAuth2Client = new OAuth2(

process.env.GMAIL\_CLIENT\_ID,

process.env.GMAIL\_CLIENT\_SECRET,

process.env.GMAIL\_REDIRECT\_URI

);

const authenticateGmail = (req, res) => {

const authUrl = oAuth2Client.generateAuthUrl({

access\_type: 'offline',

scope: ['https://www.googleapis.com/auth/gmail.readonly', 'https://www.googleapis.com/auth/gmail.send'],

});

res.redirect(authUrl);

};

const handleGmailCallback = async (req, res) => {

try {

const code = req.query.code;

const { tokens } = await oAuth2Client.getToken(code);

oAuth2Client.setCredentials(tokens);

res.send('Authentication successful! You can now access Gmail.');

} catch (err) {

console.error('Error during token exchange:', err);

res.status(500).send('Authentication failed.');

}

};

module.exports = { authenticateGmail, handleGmailCallback };

##### **c. Define Routes**

Create routes/emailRoutes.js for API endpoints:

const express = require('express');

const { authenticateGmail, handleGmailCallback } = require('../src/gmailService');

const router = express.Router();

router.get('/gmail/authenticate', authenticateGmail);

router.get('/auth/google/callback', handleGmailCallback);

module.exports = router;

##### **d. Set Up the Main App**

In app.js, configure the server:

require('dotenv').config();

const express = require('express');

const emailRoutes = require('./routes/emailRoutes');

const app = express();

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use('/emails', emailRoutes);

app.set('view engine', 'ejs');

app.listen(3000, () => console.log('Server running on http://localhost:3000'));

#### **4. Frontend Implementation**

Add a simple EJS template (views/index.ejs) for testing:  
  
 <!DOCTYPE html>

<html>

<head>

<title>Email Management</title>

</head>

<body>

<h1>Email Management App</h1>

<a href="/emails/gmail/authenticate">Authenticate with Gmail</a>

</body>

</html>

Update the home route in app.js:  
  
 app.get('/', (req, res) => {

res.render('index');

});

#### **5. Test and Debug**

Run the app:  
 node app.js

1. Visit http://localhost:3000 in your browser.
2. Authenticate Gmail and handle tokens using the /auth/google/callback route.

### **Next Steps**

1. Add features for **sending emails** and **retrieving email lists**.
2. Integrate Microsoft Graph API similarly for Outlook.
3. Deploy the app using **Heroku**, **Vercel**, or any cloud provider.
4. Secure the app for production by storing environment variables safely (e.g., using cloud secret managers).

This is the full roadmap I provided earlier. Let me know if you’d like detailed help with any specific step! 😊