Troubleshooting my issues:-

Verify the Setup

- **Test Nginx Proxy**: Open your browser and navigate to `http://<YOUR_EC2_PUBLIC_IP>`. If everything is configured correctly, the backend API should respond.
- **Check Nginx Logs**: If there are any issues, review the logs:

```
bash
Copy code
sudo tail -f /var/log/nginx/error.log
```

1 . virithi024@MSI:/mnt/TravelMemory/backend$ sudo npm install npm WARN EBADENGINE Unsupported engine { npm WARN EBADENGINE package: 'bson@5.3.0', npm WARN EBADENGINE required: { node: '>=14.20.1' }, npm WARN EBADENGINE current: { node: 'v12.22.9', npm: '8.5.1' } npm WARN EBADENGINE } npm WARN EBADENGINE Unsupported engine { npm WARN EBADENGINE package: 'mongodb@5.5.0', npm WARN EBADENGINE required: { node: '>=14.20.1' }, npm WARN EBADENGINE current: { node: 'v12.22.9', npm: '8.5.1' } npm WARN EBADENGINE } npm WARN EBADENGINE Unsupported engine { npm WARN EBADENGINE package: 'mongoose@7.2.4', npm WARN EBADENGINE required: { node: '>=14.20.1' }, npm WARN EBADENGINE current: { node: 'v12.22.9', npm: '8.5.1' } npm WARN EBADENGINE } npm WARN EBADENGINE Unsupported engine { npm WARN EBADENGINE package: 'mquery@5.0.0', npm WARN EBADENGINE required: { node: '>=14.0.0' }, npm WARN EBADENGINE current: { node: 'v12.22.9', npm: '8.5.1' } npm WARN EBADENGINE } up to date, audited 118 packages in 2s 13 packages are looking for funding run `npm fund` for details 13 vulnerabilities (3 low, 1 moderate, 8 high, 1 critical) To address issues that do not require attention, run: npm audit fix To address all issues (including breaking changes), run: npm audit fix --force Run `npm audit` for details.
virithi024@MSI:/mnt/TravelMemory/backend$ sudo node index.js
/mnt/TravelMemory/backend/node_modules/mongodb/lib/operations/add_user.js:16
this.options = options ?? {}; ^ SyntaxError: Unexpected token '?' at wrapSafe (internal/modules/cjs/loader.js:915:16) at Module._compile (internal/modules/cjs/loader.js:963:27) at Object.Module._extensions..js (internal/modules/cjs/loader.js:1027:10) at Module.load (internal/modules/cjs/loader.js:863:32) at Function.Module._load (internal/modules/cjs/loader.js:708:14) at Module.require (internal/modules/cjs/loader.js:887:19) at require (internal/modules/cjs/helpers.js:85:18) at Object.<anonymous> (/mnt/TravelMemory/backend/node_modules/mongodb/lib/admin.js:4:20) at

Module._compile (internal/modules/cjs/loader.js:999:30) at Object.Module._extensions..js (internal/modules/cjs/loader.js:1027:10) virithi024@MSI:/mnt/TravelMemory/backend$

Step 1.2: Install the Latest Node.js Version

1. Add the NodeSource repository for Node.js 16.x or 18.x (recommended):

   ```
   curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -
   ```

   For Node.js 18:

   ```
   curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
   ```

2. Install Node.js:

   ```
   sudo apt install -y nodejs
   ```

3. Verify the installation:

   ```
   node -v
   npm -v
   ```

   Ensure the Node.js version is **14.20.1 or higher** and npm is updated.

---

2. how to check my mongodb authentication for below travelmemory url from mongo db compass. do i need to create new cluster.
buntu@ip-172-31-56-174:~/TravelMemory/backend$ sudo node index.js Server started at http://52.207.24.26:3000 DB ERROR: MongoServerError: bad auth : authentication failed at Connection.onMessage (/home/ubuntu/TravelMemory/backend/node_modules/mongodb/lib/cmap/connection.js:202:26) at MessageStream.<anonymous> (/home/ubuntu/TravelMemory/backend/node_modules/mongodb/lib/cmap/connection.js:61:60) at MessageStream.emit (node:events:513:28) at processIncomingData (/home/ubuntu/TravelMemory/backend/node_modules/mongodb/lib/cmap/message_stream.js:124:16) at MessageStream._write (/home/ubuntu/TravelMemory/backend/node_modules/mongodb/lib/cmap/message_stream.js:33:9) at writeOrBuffer (node:internal/streams/writable:391:12) at _write (node:internal/streams/writable:332:10) at MessageStream.Writable.write (node:internal/streams/writable:336:10) at TLSSocket.ondata (node:internal/streams/readable:754:22) at TLSSocket.emit (node:events:513:28) { ok: 0, code: 8000, codeName: 'AtlasError', connectionGeneration: 0, [Symbol(errorLabels)]: Set(2) { 'HandshakeError', 'ResetPool' } } node:internal/process/promises:279 triggerUncaughtException(err, true /* fromPromise */); ^ MongoServerError: bad auth : authentication failed at Connection.onMessage (/home/ubuntu/TravelMemory/backend/node_modules/mongodb/lib/cmap/connection.js:202:26) at MessageStream.<anonymous> (/home/ubuntu/TravelMemory/backend/node_modules/mongodb/lib/cmap/connection.js:61:60) at MessageStream.emit (node:events:513:28) at processIncomingData (/home/ubuntu/TravelMemory/backend/node_modules/mongodb/lib/cmap/message_stream.js:124:16) at MessageStream._write (/home/ubuntu/TravelMemory/backend/node_modules/mongodb/lib/cmap/message_stream.js:33:9) at writeOrBuffer (node:internal/streams/writable:391:12) at _write (node:internal/streams/writable:332:10) at MessageStream.Writable.write (node:internal/streams/writable:336:10) at TLSSocket.ondata (node:internal/streams/readable:754:22) at TLSSocket.emit (node:events:513:28) { ok: 0, code: 8000, codeName: 'AtlasError', connectionGeneration: 0, [Symbol(errorLabels)]: Set(2) { 'HandshakeError', 'ResetPool' } } ubuntu@ip-172-31-56-174:~/TravelMemory/backend$

The error **`bad auth : authentication failed`** indicates that your MongoDB Atlas connection string has incorrect or missing authentication details, such as the username, password, or database name.

Here's how to troubleshoot and resolve the issue step by step:

. Test the Connection in MongoDB Compass

You can use **MongoDB Compass** to test your connection.

1. **Open MongoDB Compass**.
2. **Enter Your Connection String**:
    - Use the connection string from your `.env` file:

      ```
      mongodb+srv://vika2k:<password>@cluster0.b7mwy.mongodb.net/traveldatabase?retryWrites=true&w=majority
      ```

    - Replace `<password>` with the actual password for `vika2k`.
3. **Click "Connect"**:
    - If the connection is successful, you will see the collections in your database.
    - If it fails, double-check your connection string.

---

3. Correct the `.env` File

Ensure your `.env` file has the correct values:

```
env

MONGO_URI="mongodb+srv://vika2k:<password>@cluster0.b7mwy.mongodb.net/traveldatabase?retryWrites=true&w=majority"

PORT=3000
```

- Replace `<password>` with the correct password for the `vika2k` user.
- Ensure there are no spaces in the connection string.

---

4. Restart the Application

After updating the `.env` file, restart the backend application:

```
node index.js
```

---

3. Fix Vulnerabilities

1. Run this command to fix vulnerabilities:

   ```
   npm audit fix --force
   ```

2. If there are still critical vulnerabilities, review them manually:

   ```
   npm audit
   ```

3. 024/11/24 05:57:51 [emerg] 2177#2177: unexpected "}" in /etc/nginx/sites-enabled/default:66 nginx: configuration file /etc/nginx/nginx.conf test failed ubuntu@ip-172-31-56-174:~/TravelMemory/backend$ ^C y test connection is failing . check my udo nano /etc/nginx/sites-available/default ubuntu@ip-172-31-56-174:~/TravelMemory/backend$ - as given in picture

The error message **`unexpected "}"`** in your Nginx configuration indicates there is an extra or misplaced closing brace (`}`) in the configuration file. Let's debug and fix this step by step:

Steps to Fix Nginx Configuration

1. **Inspect the Configuration File** Open the Nginx configuration file:

   ```
   Copy codesudo nano /etc/nginx/sites-available/default
   ```

2. **Check the Structure** Look for unmatched or misplaced braces (`{` and `}`). A proper configuration file structure should look like this:

   ```
   nginx
   Copy code
   server {
         listen 80;

         server_name 52.207.24.26;

         location / {
         proxy_pass http://localhost:3000;
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection 'upgrade';
         proxy_set_header Host $host;
         proxy_cache_bypass $http_upgrade;
         }
   }
   ```

   Ensure:

   - Each opening brace `{` has a matching closing brace `}`.
   - There are no extra or misplaced braces.

3. **Correct Any Syntax Issues**
   - If there is an extra `}` at the end of the file or in any location block, remove it.

- Verify indentation to make it easier to spot errors.

4. **Test the Configuration** Once corrected, test the Nginx configuration:

```
sudo nginx -t
```

   - If the configuration is valid, you'll see:

```
bash
Copy code
nginx: configuration file /etc/nginx/nginx.conf test is
successful
```

5. **Reload Nginx** If the configuration test is successful, reload Nginx to apply the changes:

```
sudo systemctl reload nginx
```

---

4. Domain Setup with Cloudflare: - Connect your custom domain to the application using Cloudflare. - Create a CNAME record pointing to the load balancer endpoint. - Set up an A record with the IP address of the EC2 instance hosting the front end. how to perform above

Here's how you can set up your **custom domain with Cloudflare** and configure **CNAME** and **A** records to connect your application:

Step-by-Step Guide for Domain Setup with Cloudflare

---

1. Add Your Domain to Cloudflare

1. **Log in to Cloudflare**:
   - Go to Cloudflare and log in (or create an account if you don't have one).
2. **Add Your Domain**:
   - Click on **Add a Site**.
   - Enter your domain name (e.g., `example.com`) and click **Add Site**.
3. **Select a Plan**:
   - Choose the **Free Plan** (or upgrade if needed).
4. **Update DNS Nameservers**:
   - Cloudflare will provide you with **two nameservers** (e.g., `ns1.cloudflare.com` and `ns2.cloudflare.com`).
   - Update these nameservers in your domain registrar (e.g., GoDaddy, Namecheap, AWS Route 53):
     - Log in to your registrar account.
     - Locate your domain's DNS settings.
     - Replace the existing nameservers with the ones provided by Cloudflare.
   - Wait for the changes to propagate (this can take a few minutes to 24 hours).

---

2. Configure DNS Records in Cloudflare

Once your domain is connected to Cloudflare, configure the necessary DNS records:

2.1. Create a CNAME Record

A **CNAME** record maps your domain (e.g., `www.example.com`) to the **Load Balancer endpoint**.

1. Go to the **DNS** section in Cloudflare.
2. Click **Add Record**.
3. Enter the following details:
    - **Type**: CNAME
    - **Name**: `www` (or leave blank for the root domain).
    - **Target**: Your **load balancer endpoint** (e.g., `lb-endpoint.amazonaws.com`).
    - Enable **Proxy Status** to take advantage of Cloudflare's performance and security features.
4. Click **Save**.

2.2. Create an A Record

An **A** record maps your domain (e.g., `example.com`) to the **IP address of your EC2 instance**.

1. Click **Add Record**.
2. Enter the following details:
    - **Type**: A
    - **Name**: `@` (or leave blank for the root domain).
    - **IPv4 Address**: The public IP address of your EC2 instance hosting the frontend.
    - Enable **Proxy Status** to secure and optimize your connection.
3. Click **Save**.

---

3. Test the Domain Connection

1. Open your browser and enter your custom domain (e.g., `example.com` or `www.example.com`).
2. It should resolve to your application.

---

4. Enable HTTPS (Optional but Recommended)

1. In Cloudflare, go to the **SSL/TLS** settings.
2. Choose **Full** or **Full (Strict)** mode for SSL.
    - **Full**: If your backend doesn't have an SSL certificate.
    - **Full (Strict)**: If your backend has a valid SSL certificate.
3. Cloudflare will issue a free SSL certificate for your domain, ensuring secure HTTPS connections.

---

5. Verify Everything

1. Use tools like https://dnschecker.org/ to verify the DNS propagation of your domain.
2. Ensure the CNAME and A records are pointing correctly.

Steps to Access and Use Cloudflare

1. **Visit Cloudflare**:
    - Open your browser and go to https://www.cloudflare.com.
2. **Create an Account** (if you don't already have one):
    - Click on **Sign Up**.
    - Provide your email address and create a password.
    - Complete the sign-up process.
3. **Log In** (if you already have an account):
    - Click **Log In** at the top-right corner of the page.
    - Enter your credentials and access your dashboard.

Using Cloudflare

1. Once logged in, you can:
    - Add your domain to Cloudflare.
    - Configure DNS records (CNAME, A, etc.).
    - Manage security settings (SSL/TLS, firewall rules, etc.).
    - Optimize your website's performance (cache, CDN).
2. Use the Cloudflare dashboard to manage your DNS records and set up your custom domain for the application.

Update DNS Nameservers: Cloudflare will provide you with two nameservers (e.g., ns1.cloudflare.com and ns2.cloudflare.com). Update these nameservers in your domain registrar (e.g., GoDaddy, Namecheap, AWS Route 53): Log in to your registrar account. Locate your domain's DNS settings. Replace the existing nameservers with the ones provided by Cloudflare. Wait for the changes to propagate (this can take a few minutes to 24 hours).

Here's a detailed explanation of **how to update your DNS nameservers with your domain registrar** after setting up Cloudflare:F

What Are Nameservers?

- **Nameservers** are like an address book for the internet. They tell the browser where to find the services (like websites) associated with your domain name.
- When you use Cloudflare, they give you **two nameservers** (e.g., `ns1.cloudflare.com` and `ns2.cloudflare.com`) that need to replace the current nameservers in your **domain registrar** settings.

Where to Update Nameservers?

- This step needs to be done in your **domain registrar's account** (e.g., GoDaddy, Namecheap, AWS Route 53, Google Domains, etc.).
- Your domain registrar is where you purchased your domain name.

Step-by-Step Instructions

Step 1: Get Cloudflare Nameservers

1. After adding your domain to Cloudflare:
    - Cloudflare will display **two nameservers**. These will look like:
        - `ns1.cloudflare.com`
        - `ns2.cloudflare.com`
2. Copy these nameservers (you'll use them in the next step).

Step 2: Log in to Your Domain Registrar

1. Go to the website of your **domain registrar** (e.g., GoDaddy, Namecheap, AWS Route 53).
2. Log in with your credentials.

Step 3: Locate DNS Settings

1. After logging in, navigate to the **DNS Settings** or **Domain Settings** for the domain you're updating.
    - This is typically found under a "Manage Domains" or "My Domains" section.
2. Find the option to **edit nameservers**.

Step 4: Replace Existing Nameservers

1. Delete the current nameservers.
2. Replace them with the two nameservers provided by Cloudflare:
    - e.g., `ns1.cloudflare.com`
    - e.g., `ns2.cloudflare.com`

Step 5: Save Changes

1. Save the changes and confirm.
2. Your registrar may ask for confirmation (via email or a click).

Step 6: Wait for Propagation

- DNS changes may take **a few minutes to 24 hours** to propagate globally.
- You can check if your nameservers have updated using tools like DNSChecker.

Next Steps to Update Nameservers

1. **Copy the Nameservers**:
    - `courtney.ns.cloudflare.com`
    - `lochlan.ns.cloudflare.com`
2. **Log In to Your Domain Registrar**:
    - Visit the website of the company where you purchased your domain (e.g., GoDaddy, Namecheap, AWS Route 53, etc.).
    - Log in with your credentials.

3. **Locate Your Domain Settings**:
    - Go to the "Manage Domains" or "Domain Settings" section.
    - Select the domain for which you want to update the nameservers.
4. **Replace Existing Nameservers**:
    - Look for the **Nameservers** section.
    - Delete the current nameservers.
    - Add the Cloudflare nameservers (`courtney.ns.cloudflare.com` and `lochlan.ns.cloudflare.com`).
5. **Save Changes**:
    - Confirm and save the changes.
    - Some registrars might ask for additional confirmation (e.g., email verification).
6. **Wait for Propagation**:
    - DNS changes may take **a few minutes to 24 hours** to propagate globally.
    - You can monitor the status using tools like DNSChecker.
