Southampton Solent University

Computer Science
2025

Web Technologies

Student ID: 10423836

Table of Contents

*1. **Proof conformity to W3C standards** 10*

*2. **Contact Form with AJAX submission (Client–Server Database Interaction)** 10*

*3. **Live Event Search with AJAX (Client–Server Database Interaction)** 13*

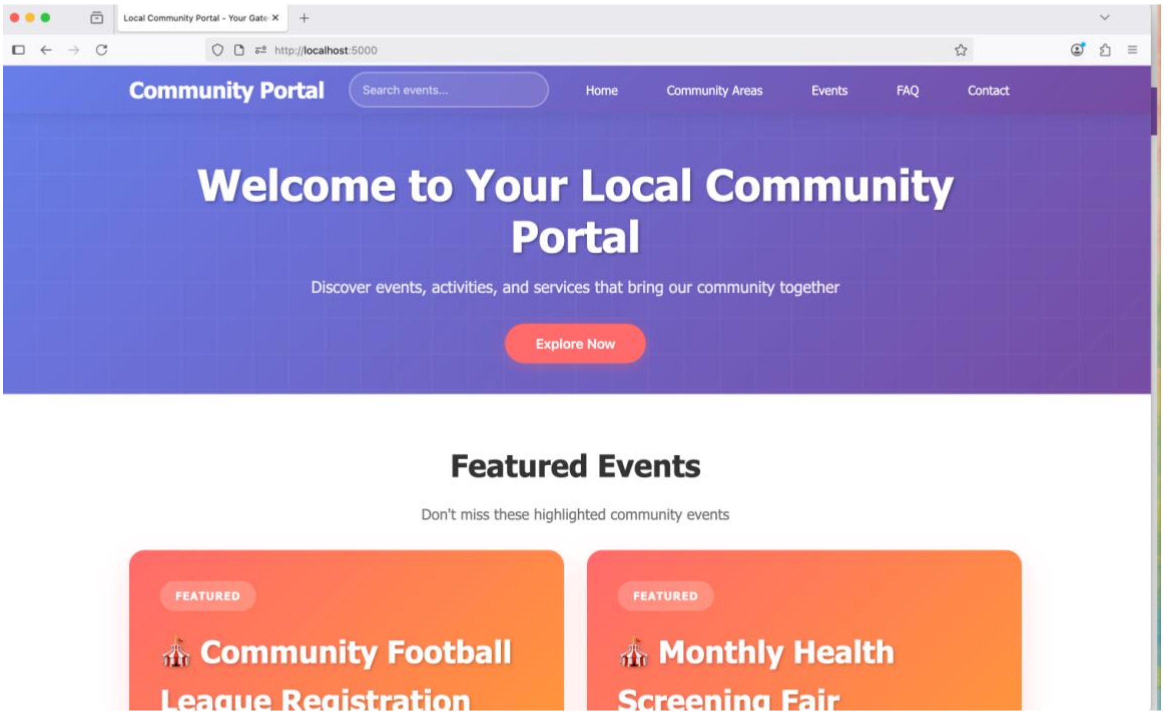
*4. **Security considerations** 14*

*5. **ERD showing the database structure** 14*

*6. **AI usage** 16*

List of Figures

Figure 1: Homepage on Mozilla Firefox ... 4
Figure 2: Community areas on Mozilla Firefox ... 4
Figure 3: Homepage on chrome ... 5
Figure 4: community areas chrome ... 5
Figure 5: Homepage on Safari ... 6
Figure 6: Mobile style community portal tested on Safari ... 8
Figure 7: Community event on safari ... 9
Figure 8: CSS Validator ... 10
Figure 9: HTML Validator ... 10
Figure 10: valid data input on database ... 12
Figure 11: showing error validating the form ... 12
Figure 12: Console showing successful and unsuccessful fetch ... 13
Figure 13: searching on the website ... 14
Figure 14: queries generated on console log ... 14
Figure 15: ERD database structure generated on mermaidchart.com ... 15



Welcome to Your Local Community Portal

Discover events, activities, and services that bring our community together

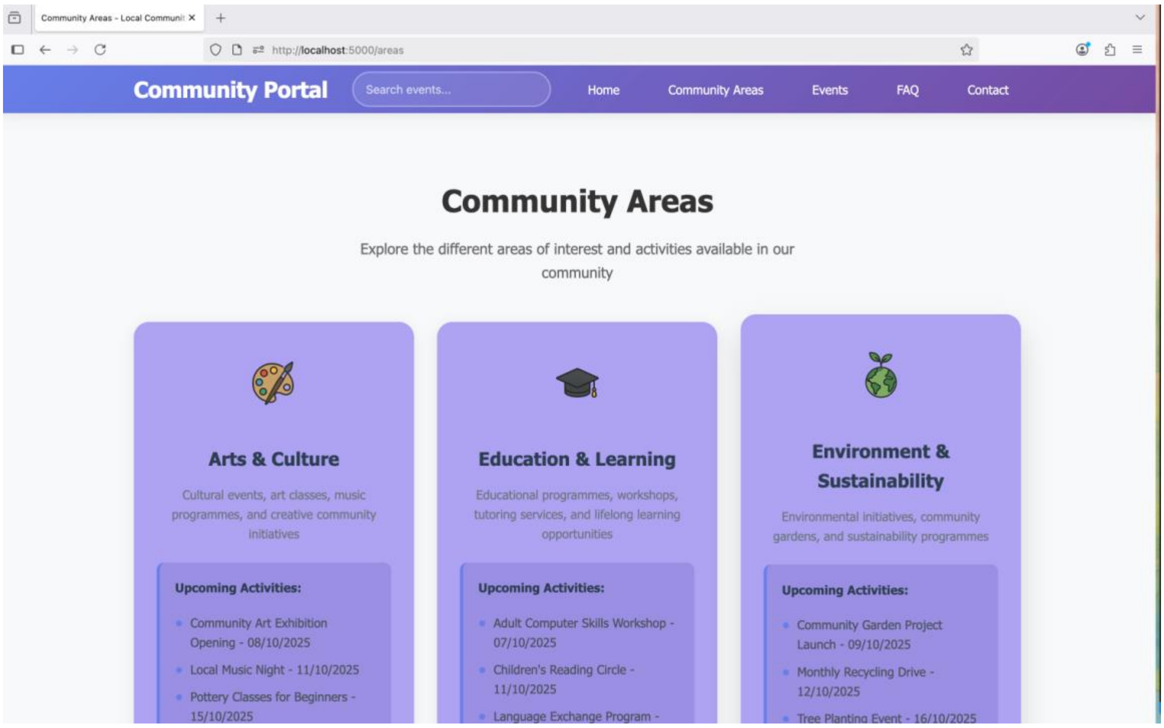
Featured Events

Don't miss these highlighted community events

Community Football League Registration

Monthly Health Screening Fair

Figure 1: Homepage on Mozilla Firefox



Community Areas

Explore the different areas of interest and activities available in our community

Arts & Culture

Cultural events, art classes, music programmes, and creative community initiatives

Upcoming Activities:

- Community Art Exhibition Opening - 08/10/2025
- Local Music Night - 11/10/2025
- Pottery Classes for Beginners - 15/10/2025

Education & Learning

Educational programmes, workshops, tutoring services, and lifelong learning opportunities

Upcoming Activities:

- Adult Computer Skills Workshop - 07/10/2025
- Children's Reading Circle - 11/10/2025
- Language Exchange Program -

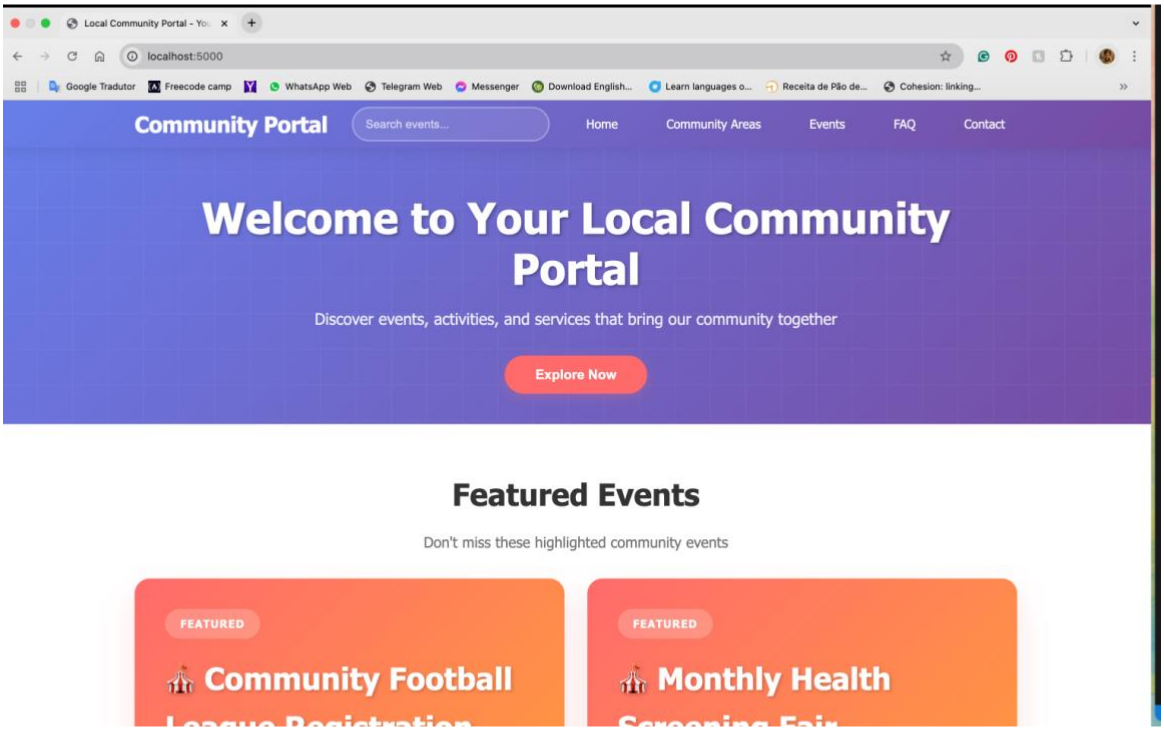
Environment & Sustainability

Environmental initiatives, community gardens, and sustainability programmes

Upcoming Activities:

- Community Garden Project Launch - 09/10/2025
- Monthly Recycling Drive - 12/10/2025
- Tree Planting Event - 16/10/2025

Figure 2: Community areas on Mozilla Firefox



Welcome to Your Local Community Portal

Discover events, activities, and services that bring our community together

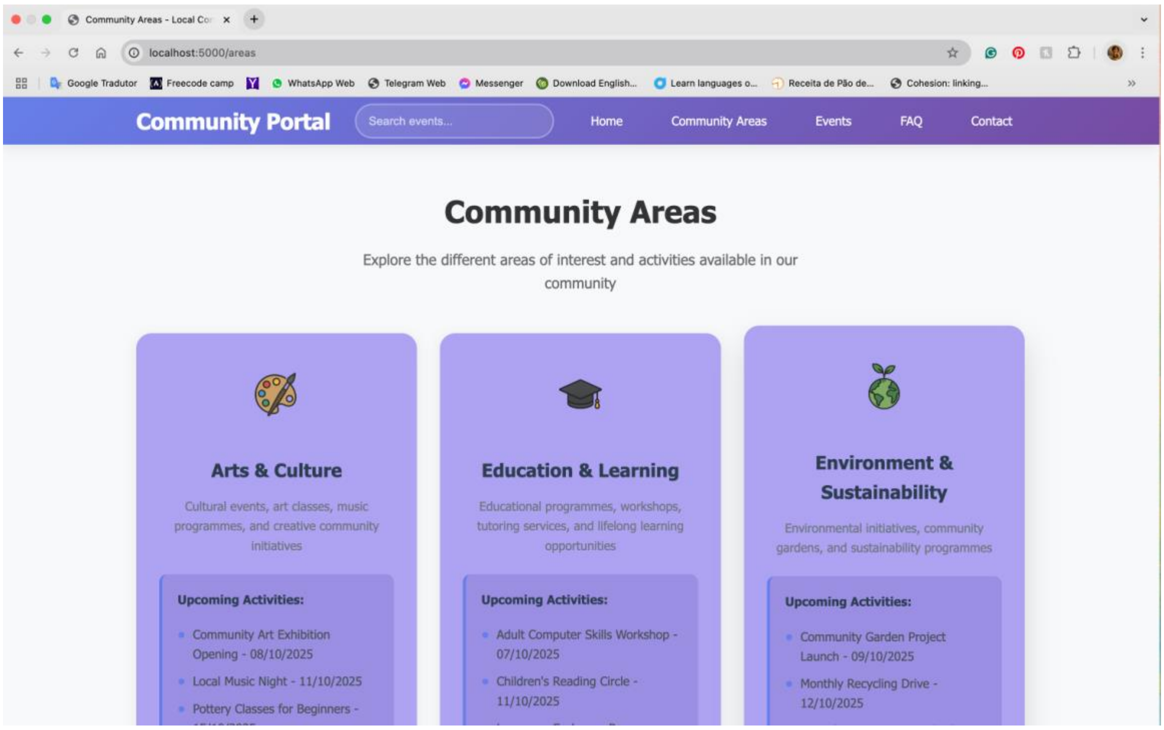
Featured Events

Don't miss these highlighted community events

Community Football

Monthly Health

Figure 3: Homepage on chrome



Community Areas

Explore the different areas of interest and activities available in our community

Arts & Culture

Cultural events, art classes, music programmes, and creative community initiatives

Upcoming Activities:

- Community Art Exhibition Opening - 08/10/2025
- Local Music Night - 11/10/2025
- Pottery Classes for Beginners -

Education & Learning

Educational programmes, workshops, tutoring services, and lifelong learning opportunities

Upcoming Activities:

- Adult Computer Skills Workshop - 07/10/2025
- Children's Reading Circle - 11/10/2025

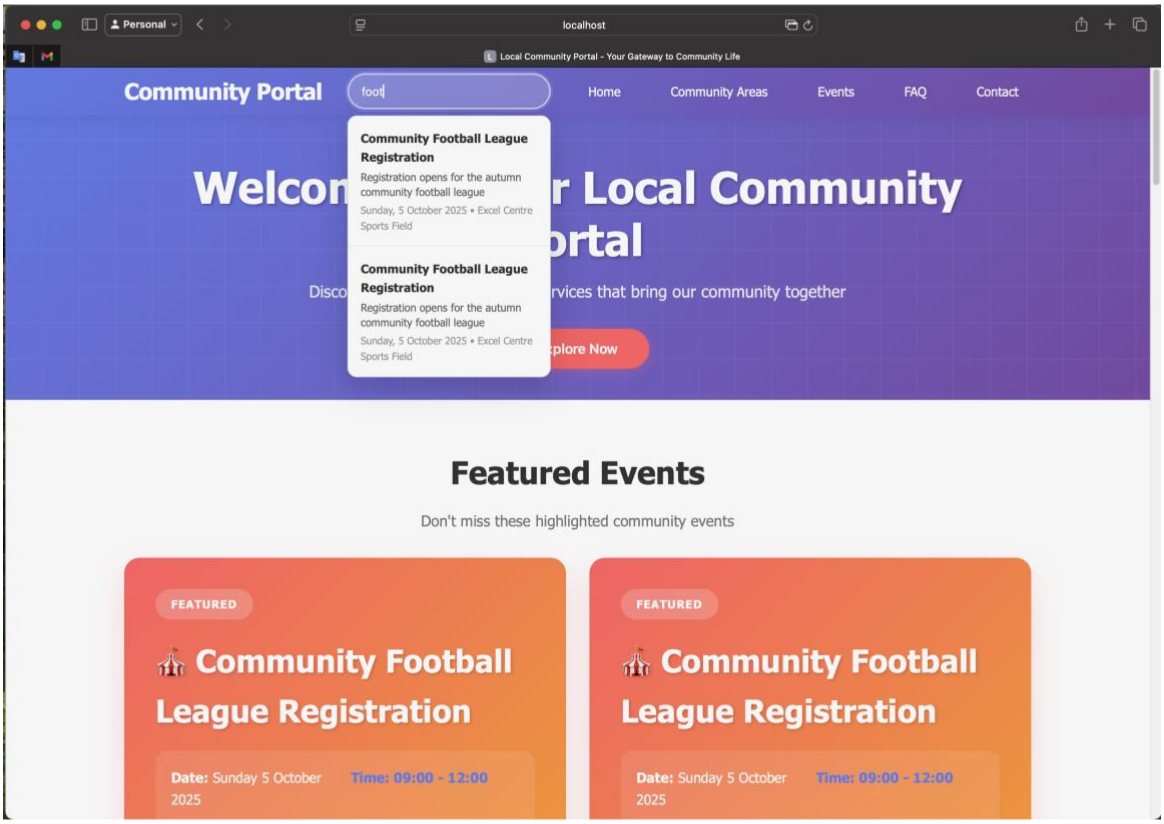
Environment & Sustainability

Environmental initiatives, community gardens, and sustainability programmes

Upcoming Activities:

- Community Garden Project Launch - 09/10/2025
- Monthly Recycling Drive - 12/10/2025

Figure 4: community areas chrome



Community Football League Registration
Registration opens for the autumn community football league
Sunday, 5 October 2025 • Excel Centre Sports Field

Community Football League Registration
Registration opens for the autumn community football league
Sunday, 5 October 2025 • Excel Centre Sports Field

Welcon r Local Community ortal

Disco rvices that bring our community together

Featured Events

Don't miss these highlighted community events

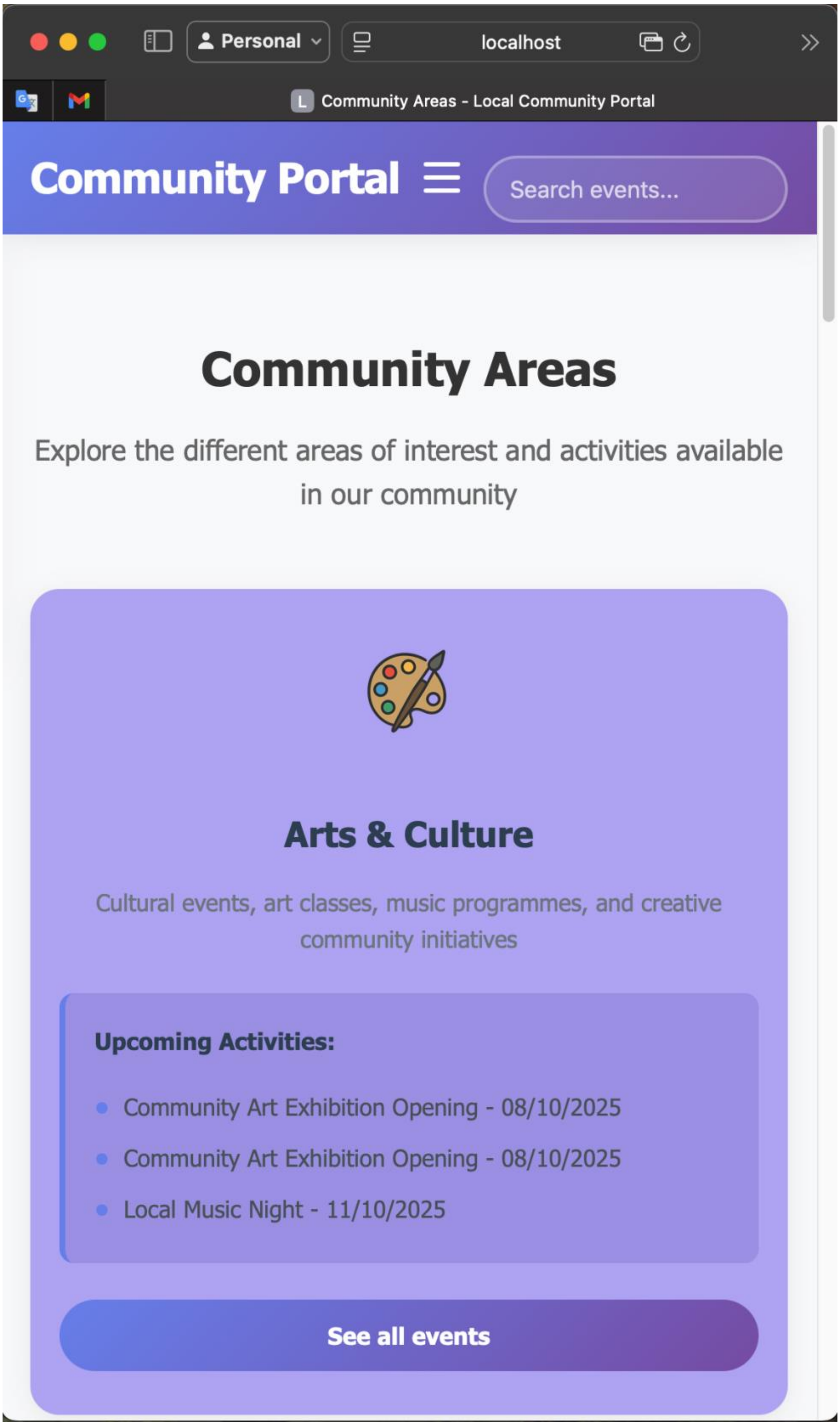
Community Football League Registration

Date: Sunday 5 October 2025 **Time: 09:00 - 12:00**

Community Football League Registration

Date: Sunday 5 October 2025 **Time: 09:00 - 12:00**

Figure 5: Homepage on Safari



Community Areas

Explore the different areas of interest and activities available in our community

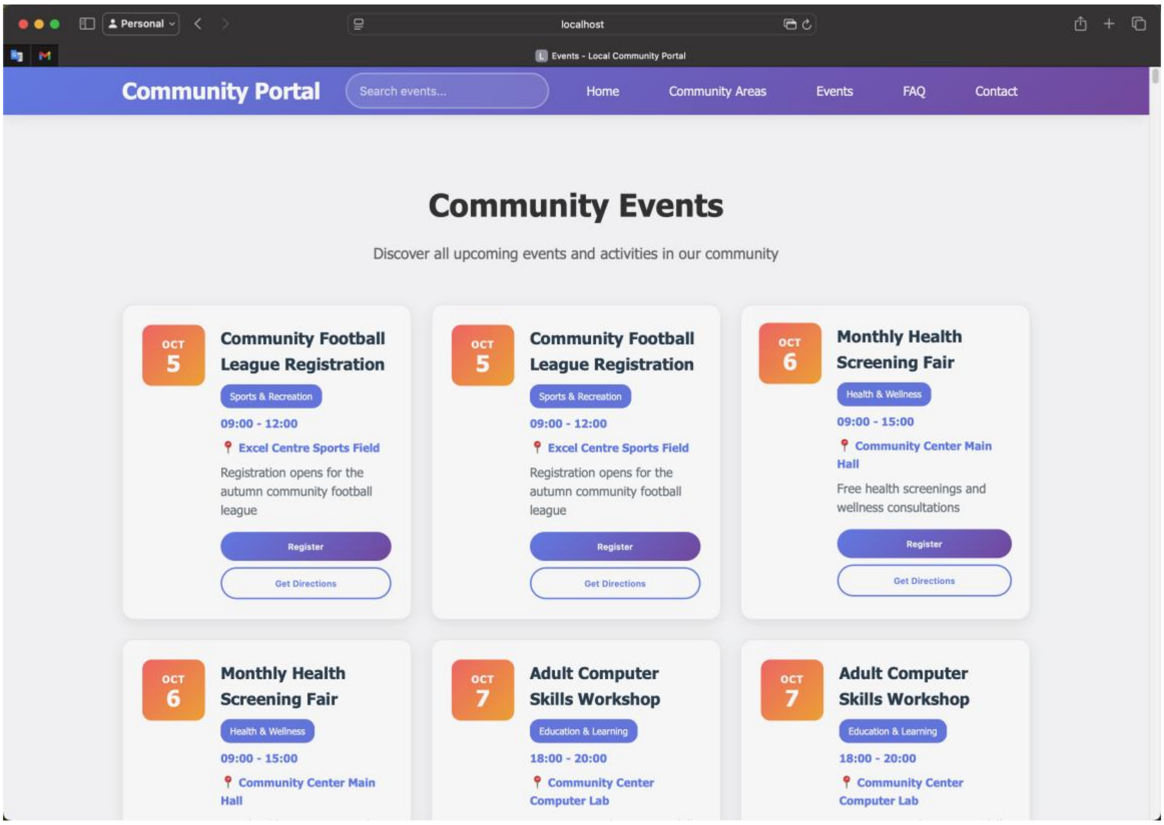
Arts & Culture

Cultural events, art classes, music programmes, and creative community initiatives

Upcoming Activities:

- Community Art Exhibition Opening - 08/10/2025
- Community Art Exhibition Opening - 08/10/2025
- Local Music Night - 11/10/2025

Figure 6: Mobile style community portal tested on Safari



Community Events

Discover all upcoming events and activities in our community

OCT 5
Community Football League Registration
Sports & Recreation
09:00 - 12:00
📍 Excel Centre Sports Field
Registration opens for the autumn community football league
Register
Get Directions

OCT 5
Community Football League Registration
Sports & Recreation
09:00 - 12:00
📍 Excel Centre Sports Field
Registration opens for the autumn community football league
Register
Get Directions

OCT 6
Monthly Health Screening Fair
Health & Wellness
09:00 - 15:00
📍 Community Center Main Hall
Free health screenings and wellness consultations
Register
Get Directions

OCT 6
Monthly Health Screening Fair
Health & Wellness
09:00 - 15:00
📍 Community Center Main Hall

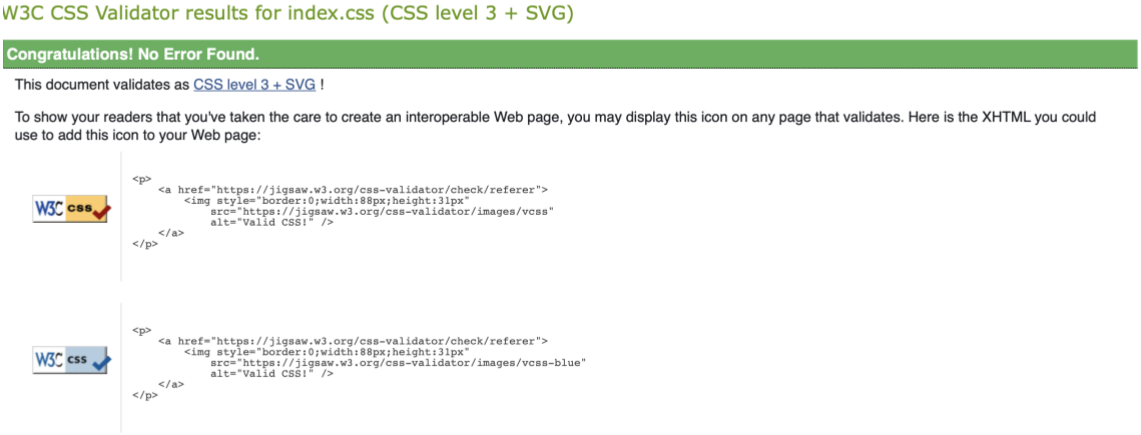
OCT 7
Adult Computer Skills Workshop
Education & Learning
18:00 - 20:00
📍 Community Center Computer Lab

OCT 7
Adult Computer Skills Workshop
Education & Learning
18:00 - 20:00
📍 Community Center Computer Lab

Figure 7: Community event on safari

1. Proof conformity to W3C standards

The screenshots below, shows the HTML and CSS structures comply with the W3C Standards



W3C CSS Validator results for index.css (CSS level 3 + SVG)

Congratulations! No Error Found.

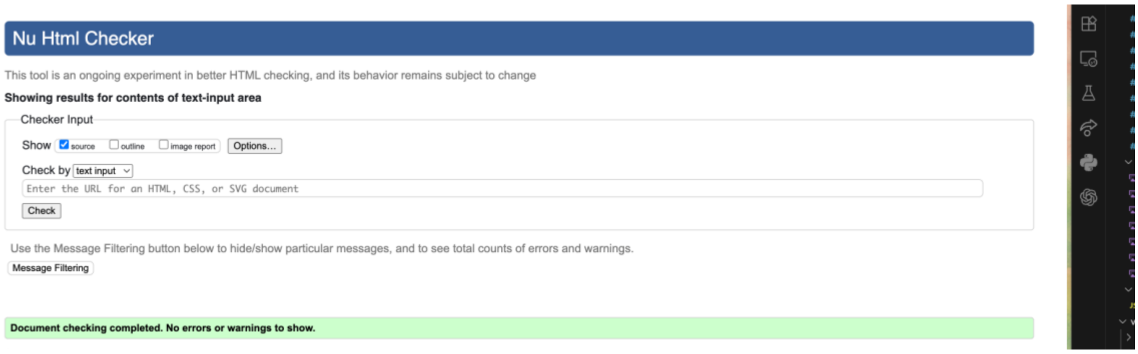
This document validates as CSS level 3 + SVG !

To show your readers that you've taken the care to create an interoperable Web page, you may display this icon on any page that validates. Here is the XHTML you could use to add this icon to your Web page:

```
<p>
   <a href="https://jigsaw.w3.org/css-validator/check/referer">
      <img style="border:0;width:88px;height:31px"
          src="https://jigsaw.w3.org/css-validator/images/vcss"
          alt="Valid CSS!" />
   </a>
</p>
```

```
<p>
   <a href="https://jigsaw.w3.org/css-validator/check/referer">
      <img style="border:0;width:88px;height:31px"
          src="https://jigsaw.w3.org/css-validator/images/vcss-blue"
          alt="Valid CSS!" />
   </a>
</p>
```

Figure 8: CSS Validator



Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for contents of text-input area

Checker Input

Show ☑ source ☐ outline ☐ image report

Check by text input

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Document checking completed. No errors or warnings to show.

Figure 9: HTML Validator

However, if we test the entire will show some syntax errors, that happens because the project is using EJS format which the validator does not understand the EJS syntax.

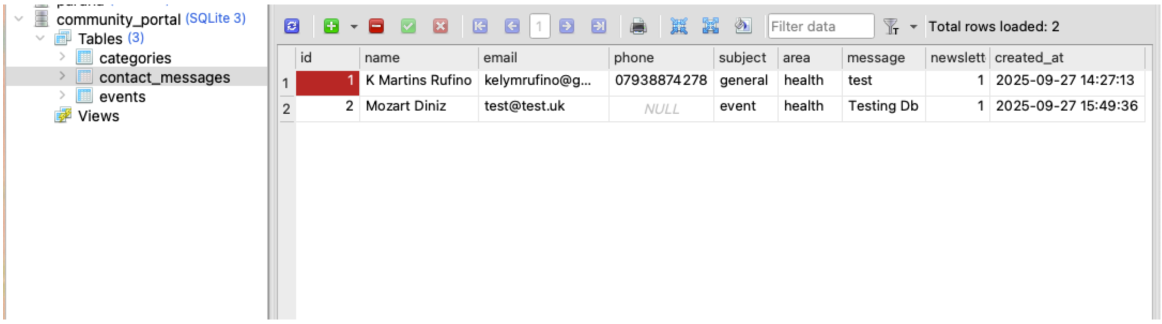
2. Contact Form with AJAX submission (Client-Server Database Interaction)

Client-side code controls what the user sees and how they interact, including layout, live validation, and the actual sending of data. The server-side code securely stores the message, performs final validation, and returns a response. This separation enables the interface to remain fast for users while ensuring data integrity and security on the backend.

The contact form is built using a mix of HTML, CSS, and JavaScript on the client side, alongside database and API functionality on the server side.

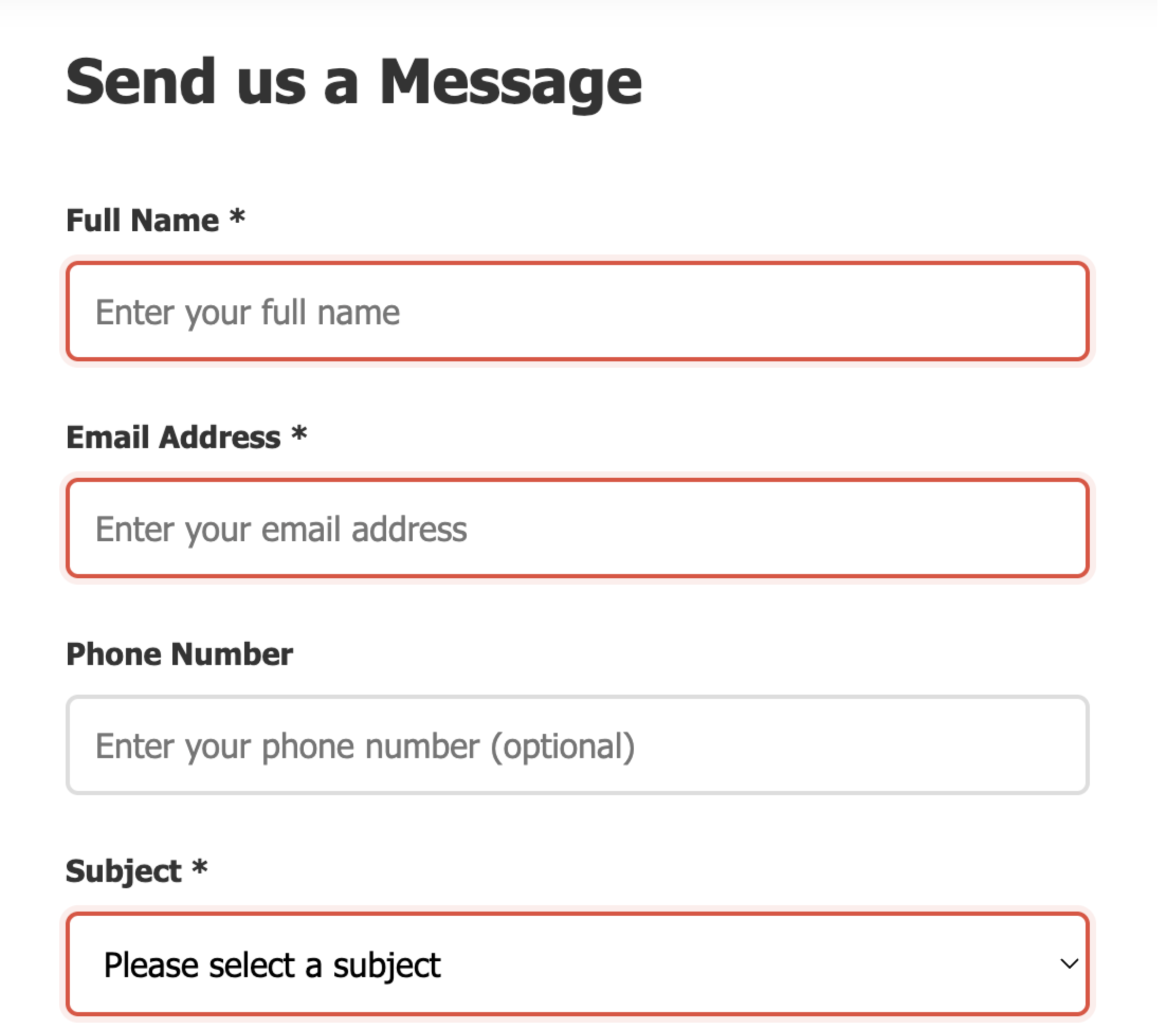
- Client-Side:
  The HTML in the contact EJS outlines the visible form, featuring text fields for the user's name, email, and message, along with optional inputs for phone number and community area.
  It also includes checkboxes for signing up for a newsletter and agreeing to privacy terms, as well as buttons to submit or reset the form, plus a hidden area for success messages.
  The design is made using CSS (contact.css), creating a responsive layout with red borders for errors, blue highlights when fields are focused, and green animations for successful submissions.
  JavaScript plays a crucial role in real-time validation of the form. It checks that all required fields are completed, verifies the email format using a regular expression, and ensures that the privacy agreement is accepted.
  Event listeners help erase error messages as the user types.
  Once everything is validated, an AJAX call (using fetch()) sends the data to the server in JSON format, showing a "Sending…" message while waiting for a response.
  This whole layer—the layout, styling, instant feedback, and the AJAX call—is handled entirely on the client side, operating directly in the user's browser.
- Server-Side:
  When the AJAX request is made, it reaches a backend route like */message*. A server framework, such as Express.js, takes in the JSON data, performs further validation and sanitization, and adds a new entry to the *contact_messages* database table.
  The server is responsible for managing logs, handling errors, and implementing security measures to prevent threats such as SQL injection and cross-site scripting.
  After processing the data, it replies with a JSON response indicating whether the operation was successful or not

  Below the screenshot showing the database updating according to data typed on contact form.



	id	name	email	phone	subject	area	message	newslett	created_at
1	1	K Martins Rufino	kelymrufino@g...	07938874278	general	health	test	1	2025-09-27 14:27:13
2	2	Mozart Diniz	test@test.uk	NULL	event	health	Testing Db	1	2025-09-27 15:49:36

Figure 10: valid data input on database



Send us a Message

Full Name *

Enter your full name

Email Address *

Enter your email address

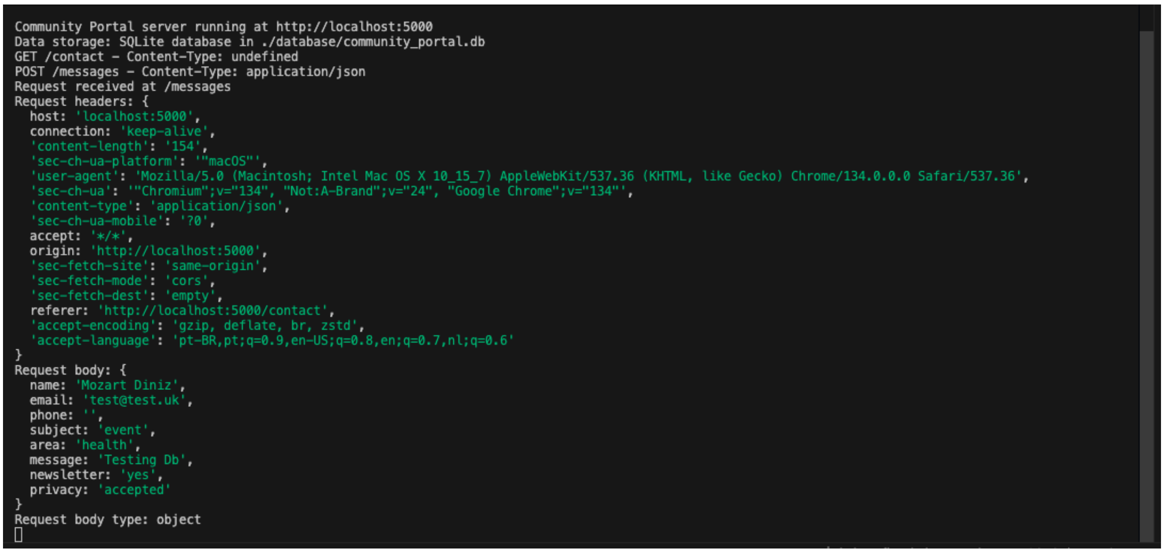
Phone Number

Enter your phone number (optional)

Subject *

Please select a subject

Figure 11: showing error validating the form



```
Community Portal server running at http://localhost:5000
Data storage: SQLite database in ./database/community_portal.db
GET /contact - Content-Type: undefined
POST /messages - Content-Type: application/json
Request received at /messages
Request headers: {
  host: 'localhost:5000',
  connection: 'keep-alive',
  'content-length': '154',
  'sec-ch-ua-platform': '"macOS"',
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36',
  'sec-ch-ua': '"Chromium";v="134", "Not:A-Brand";v="24", "Google Chrome";v="134"',
  'content-type': 'application/json',
  'sec-ch-ua-mobile': '?0',
  accept: '*/*',
  origin: 'http://localhost:5000',
  'sec-fetch-site': 'same-origin',
  'sec-fetch-mode': 'cors',
  'sec-fetch-dest': 'empty',
  referer: 'http://localhost:5000/contact',
  'accept-encoding': 'gzip, deflate, br, zstd',
  'accept-language': 'pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7,nl;q=0.6'
}
Request body: {
  name: 'Mozart Diniz',
  email: 'test@test.uk',
  phone: '',
  subject: 'event',
  area: 'health',
  message: 'Testing Db',
  newsletter: 'yes',
  privacy: 'accepted'
}
Request body type: object
▯
```

Figure 12: Console showing successful and unsuccessful fetch

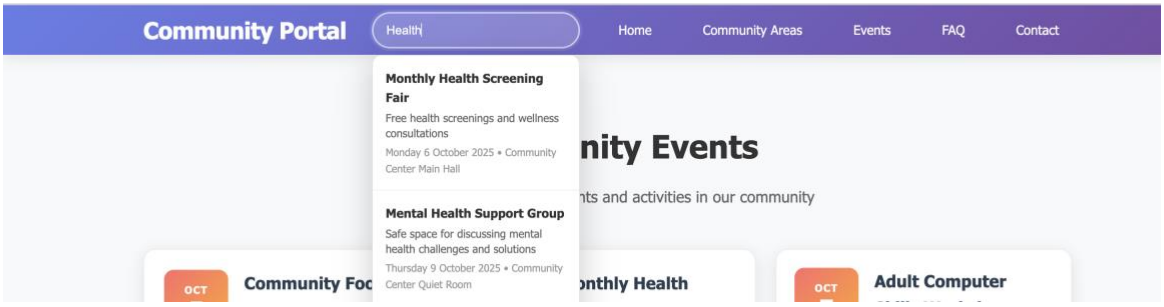
3. Live Event Search with AJAX (Client-Server Database Interaction)

Another feature that has been used client-server database was to search the live events on the page, the client side handles the user interface, detects typing, sends requests, and displays results in real-time. Meanwhile, the server side verifies input, queries the database, filters the outcomes, and generates the JSON response. Together, they deliver immediate and seamless search feedback without needing to reload the page.

- Server-Side (Node.js/Express)
The backend provides an endpoint at /api/search. When a request is received, Express processes the query parameters—q for the search term and an optional category. It invokes *DatabaseService.searchEvents(),* which executes SQL queries to find matches in event titles, descriptions, and locations, and can also search within community-area data. If a category is specified, the server narrows down the results accordingly. A JSON response containing all relevant records is then returned. The server also handles input validation, manages unforeseen errors effectively, and mitigates security risks, such as SQL injection. All interactions with the database and error logging take place here, remaining unseen by the user.

- Client-Side(JavaScript/AJAX)

  On the client side, JavaScript monitors the user's input in the search bar. To prevent overwhelming the server, it uses a technique called debouncing, which involves waiting approximately 300 milliseconds after the user has stopped typing before making a request. Once activated, the client submits an HTTP GET request to */api/search?q=searchterm&category=optional* using *fetch()* or a similar AJAX technique. When the JSON data is received, the script modifies the DOM of the page to immediately present the updated search results without the need to refresh. In cases where the request encounters an error, the client displays a user-friendly error notification.



Monthly Health Screening Fair
Free health screenings and wellness consultations
Monday 6 October 2025 • Community Center Main Hall

Mental Health Support Group
Safe space for discussing mental health challenges and solutions
Thursday 9 October 2025 • Community Center Quiet Room

nity Events

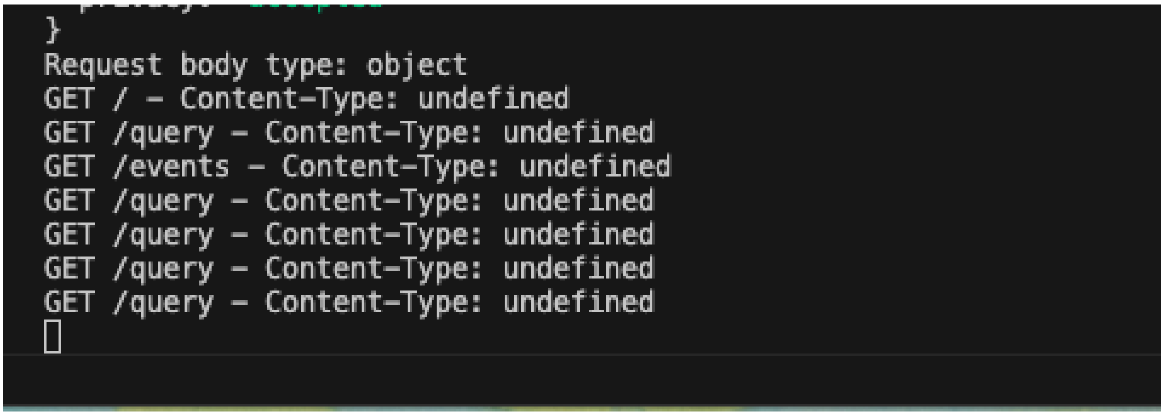
nts and activities in our community

Community Foc

onthly Health

Adult Computer

Figure 13: searching on the website



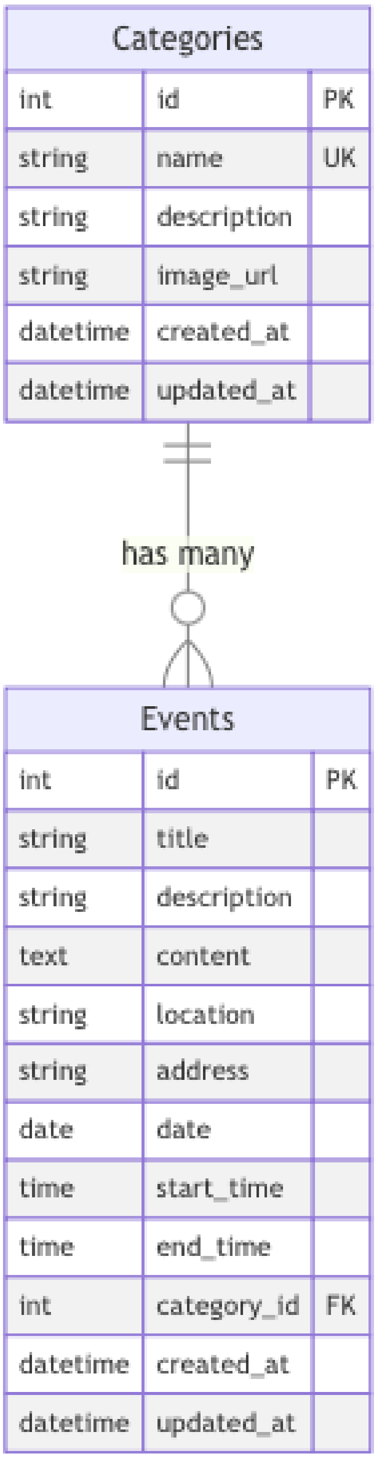
```
}
Request body type: object
GET / - Content-Type: undefined
GET /query - Content-Type: undefined
GET /events - Content-Type: undefined
GET /query - Content-Type: undefined
GET /query - Content-Type: undefined
GET /query - Content-Type: undefined
GET /query - Content-Type: undefined
```

Figure 14: queries generated on console log

4. Security considerations

The images on folder images were AI generated on Chat GPT.
The codebase exhibits a clear awareness of privacy, accessibility, and security. This includes features such as a mandatory privacy checkbox, screen-reader helpers, and responsive CSS.

5. ERD showing the database structure



Categories		
int	id	PK
string	name	UK
string	description	
string	image_url	
datetime	created_at	
datetime	updated_at	

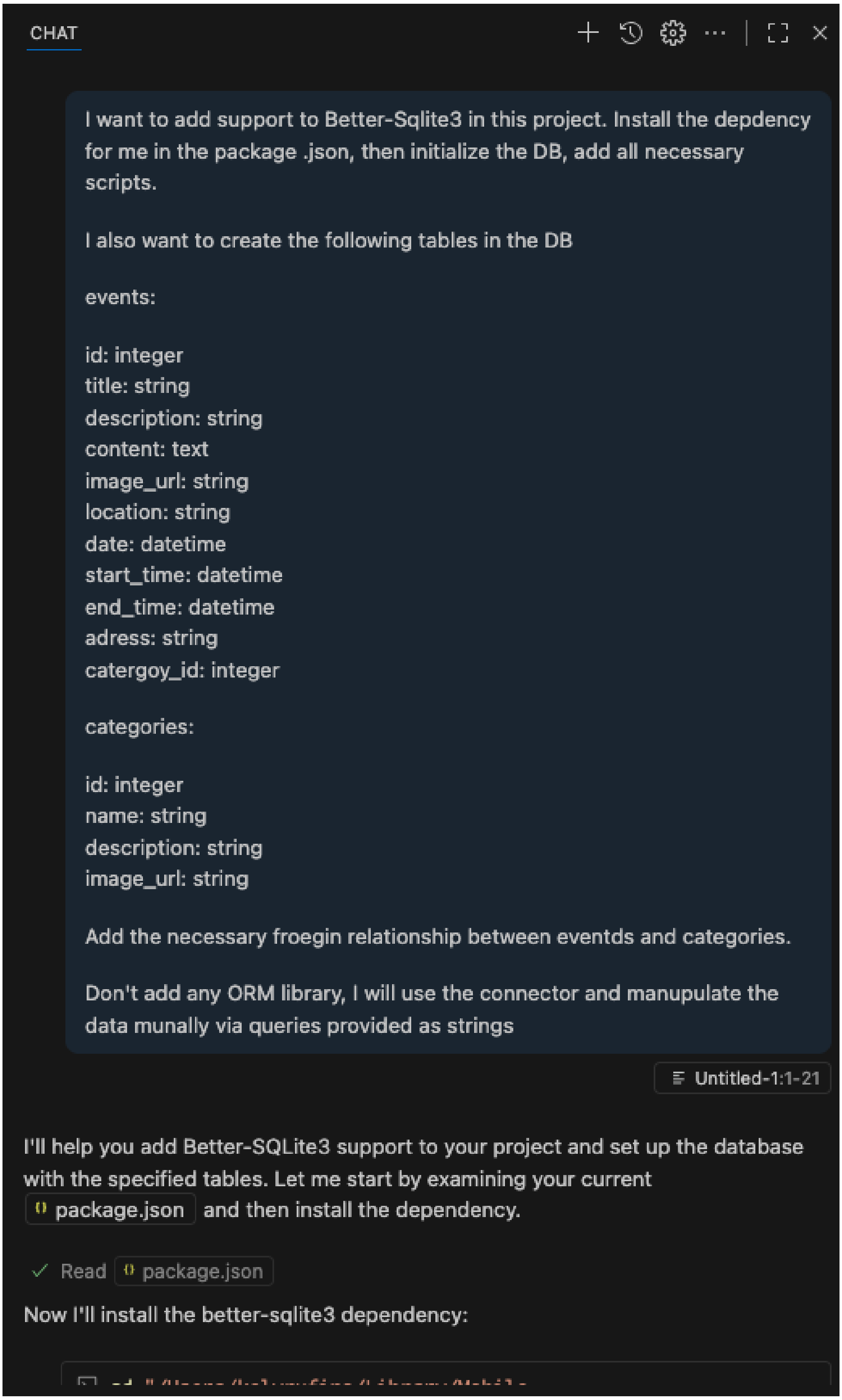
Events		
int	id	PK
string	title	
string	description	
text	content	
string	location	
string	address	
date	date	
time	start_time	
time	end_time	
int	category_id	FK
datetime	created_at	
datetime	updated_at	

Figure 15: ERD database structure generated on mermaidchart.com

6. AI usage

On this project I have use AI to help me to create a database and to generate random events to show on events page.

Below the screenshots showing the prompts and results from AI



I want to add support to Better-Sqlite3 in this project. Install the depdency for me in the package .json, then initialize the DB, add all necessary scripts.

I also want to create the following tables in the DB

events:

id: integer
title: string
description: string
content: text
image_url: string
location: string
date: datetime
start_time: datetime
end_time: datetime
adress: string
catergoy_id: integer

categories:

id: integer
name: string
description: string
image_url: string

Add the necessary froegin relationship between eventds and categories.

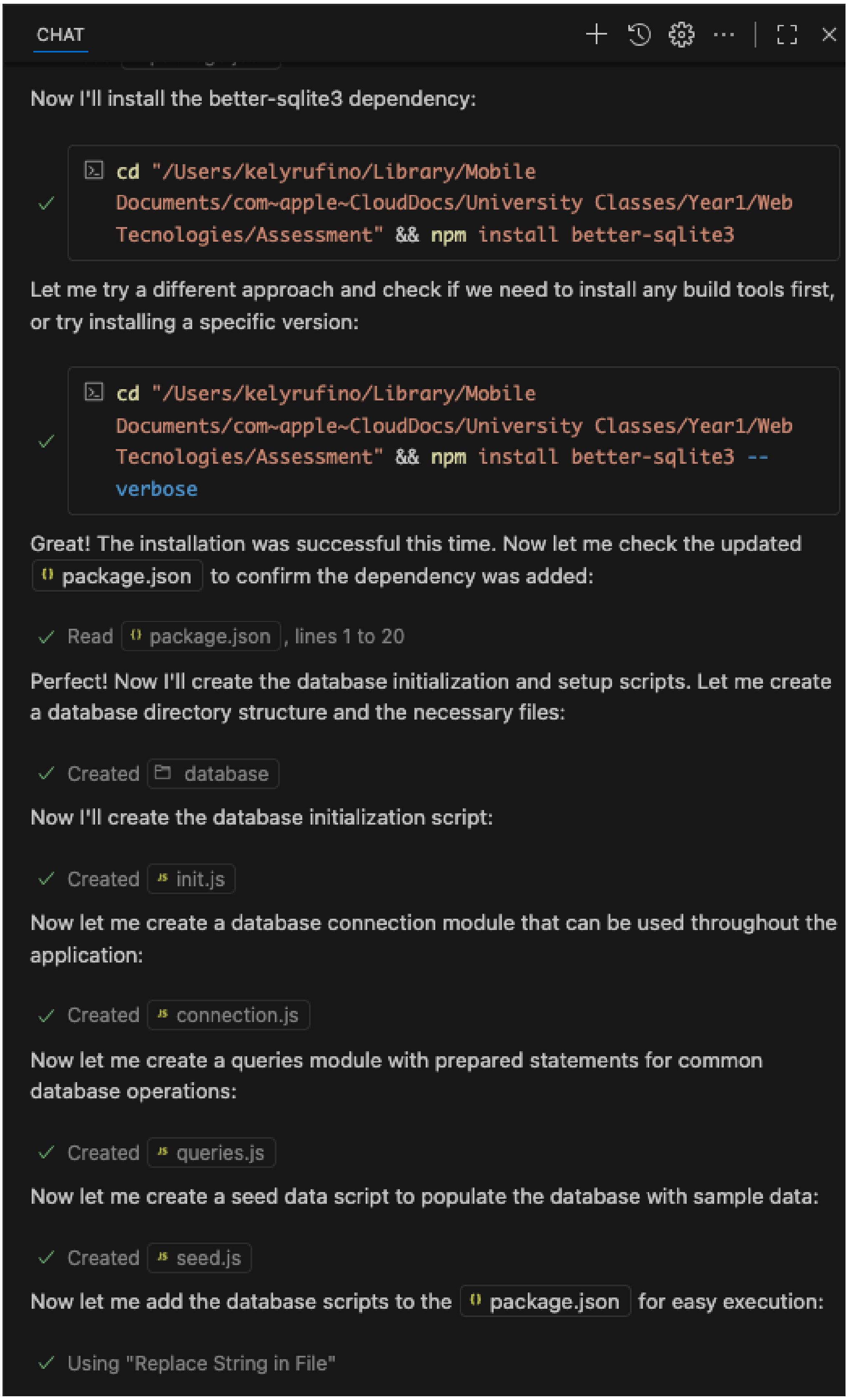
Don't add any ORM library, I will use the connector and manupulate the data munally via queries provided as strings

Untitled-1:1-21

I'll help you add Better-SQLite3 support to your project and set up the database with the specified tables. Let me start by examining your current `package.json` and then install the dependency.

✓ Read `package.json`

Now I'll install the better-sqlite3 dependency:



Now I'll install the better-sqlite3 dependency:

```
cd "/Users/kelyrufino/Library/Mobile Documents/com~apple~CloudDocs/University Classes/Year1/Web Tecnologies/Assessment" && npm install better-sqlite3
```

Let me try a different approach and check if we need to install any build tools first, or try installing a specific version:

```
cd "/Users/kelyrufino/Library/Mobile Documents/com~apple~CloudDocs/University Classes/Year1/Web Tecnologies/Assessment" && npm install better-sqlite3 --verbose
```

Great! The installation was successful this time. Now let me check the updated `package.json` to confirm the dependency was added:

✓ Read `package.json`, lines 1 to 20

Perfect! Now I'll create the database initialization and setup scripts. Let me create a database directory structure and the necessary files:

✓ Created `database`

Now I'll create the database initialization script:

✓ Created `init.js`

Now let me create a database connection module that can be used throughout the application:

✓ Created `connection.js`

Now let me create a queries module with prepared statements for common database operations:

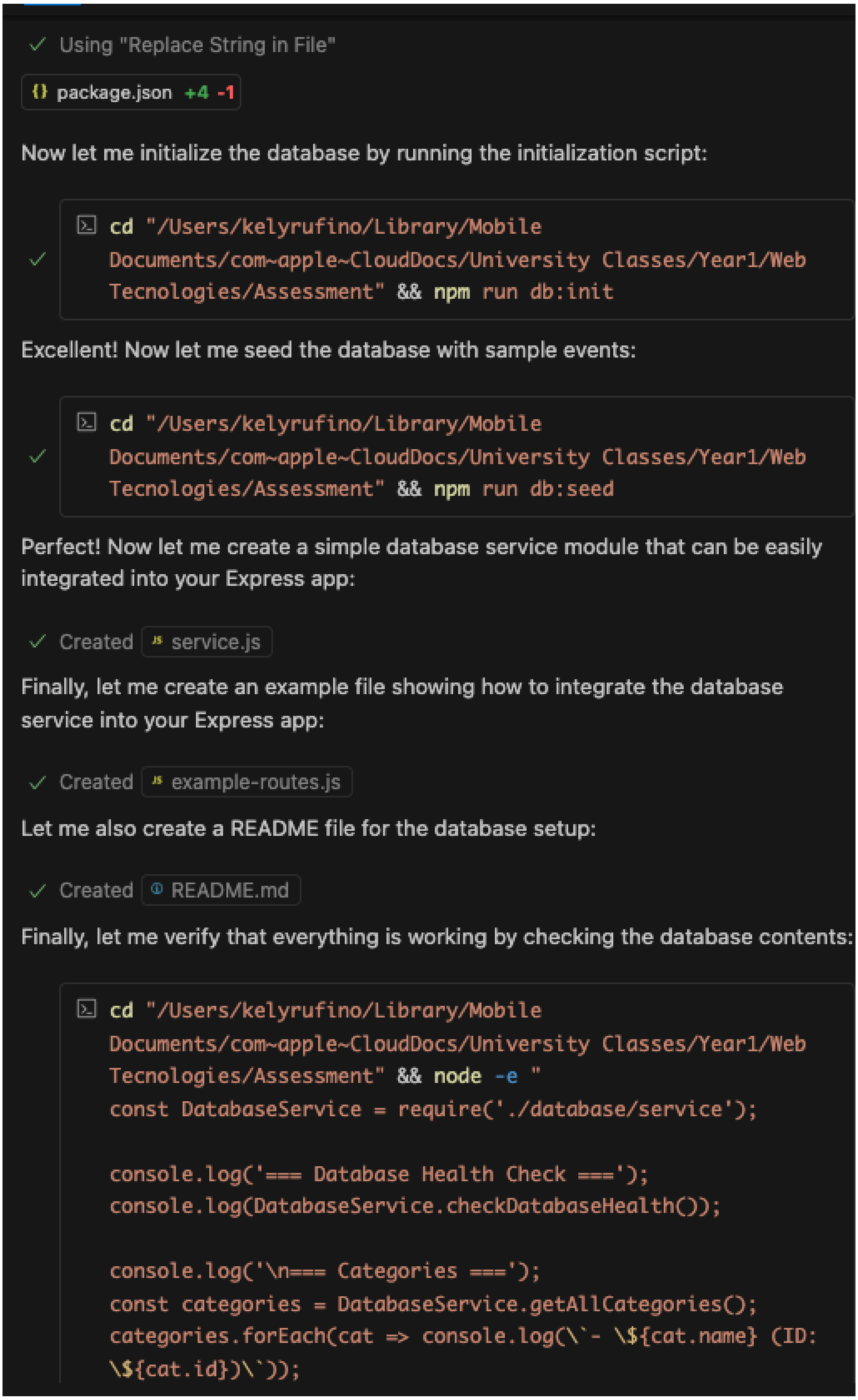
✓ Created `queries.js`

Now let me create a seed data script to populate the database with sample data:

✓ Created `seed.js`

Now let me add the database scripts to the `package.json` for easy execution:

✓ Using "Replace String in File"



✓ Using "Replace String in File"

{} package.json +4 -1

Now let me initialize the database by running the initialization script:

```
cd "/Users/kelyrufino/Library/Mobile Documents/com~apple~CloudDocs/University Classes/Year1/Web Tecnologies/Assessment" && npm run db:init
```

Excellent! Now let me seed the database with sample events:

```
cd "/Users/kelyrufino/Library/Mobile Documents/com~apple~CloudDocs/University Classes/Year1/Web Tecnologies/Assessment" && npm run db:seed
```

Perfect! Now let me create a simple database service module that can be easily integrated into your Express app:

✓ Created service.js

Finally, let me create an example file showing how to integrate the database service into your Express app:

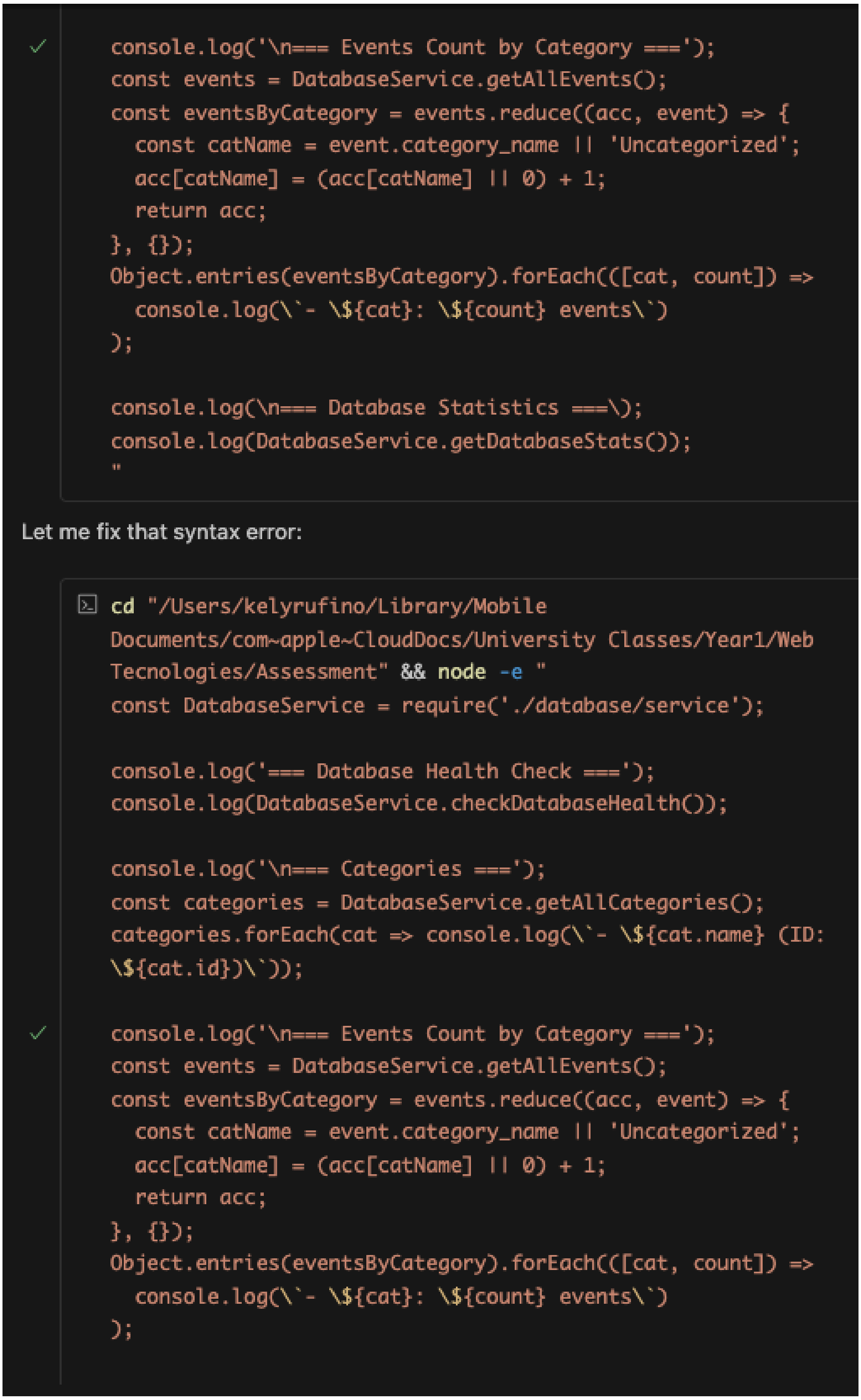
✓ Created example-routes.js

Let me also create a README file for the database setup:

✓ Created README.md

Finally, let me verify that everything is working by checking the database contents:

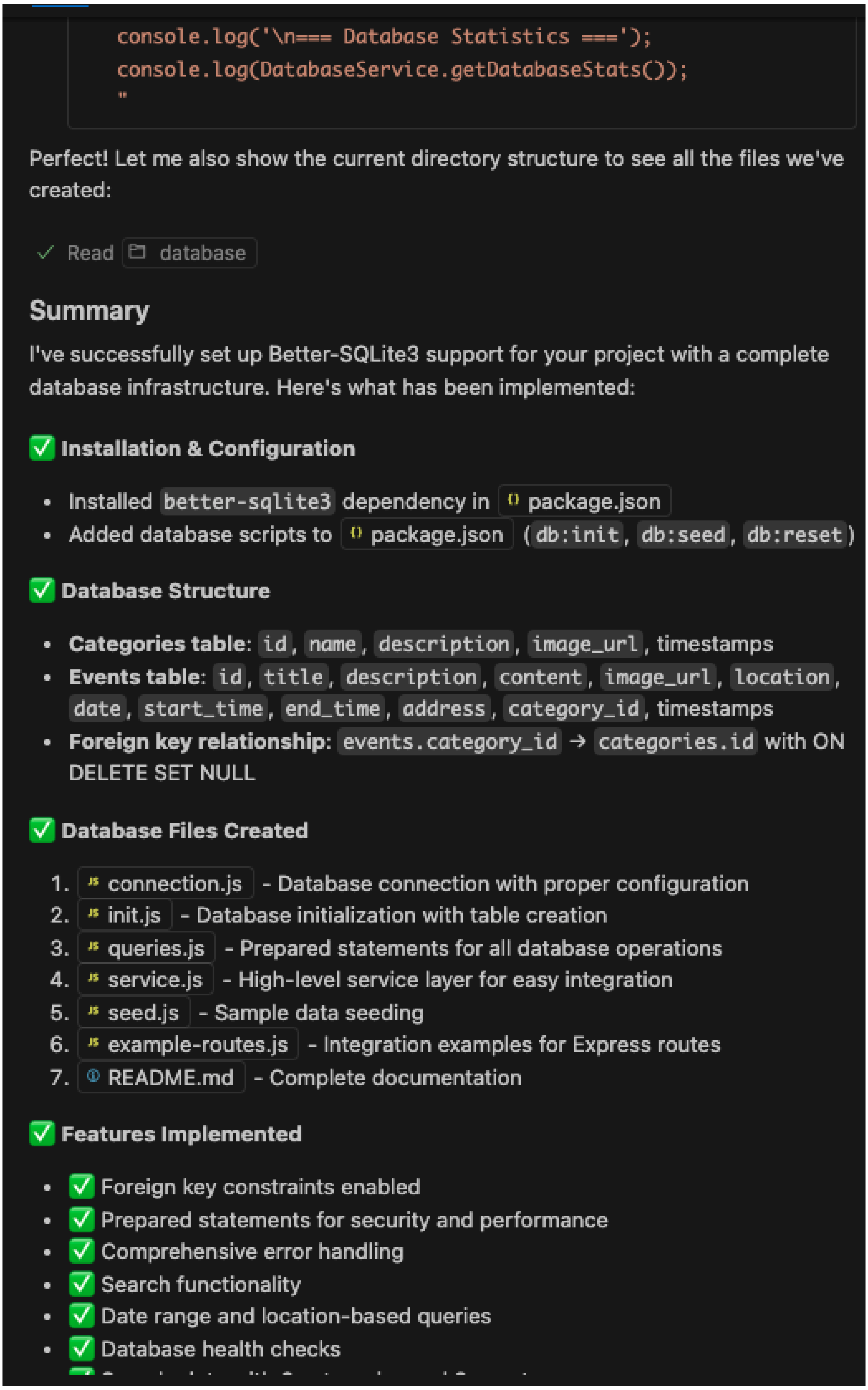
```
cd "/Users/kelyrufino/Library/Mobile Documents/com~apple~CloudDocs/University Classes/Year1/Web Tecnologies/Assessment" && node -e "
const DatabaseService = require('./database/service');

console.log('=== Database Health Check ===');
console.log(DatabaseService.checkDatabaseHealth());

console.log('\n=== Categories ===');
const categories = DatabaseService.getAllCategories();
categories.forEach(cat => console.log(\`- \${cat.name} (ID: \${cat.id})\`));
```



```
console.log('\n=== Events Count by Category ===');
const events = DatabaseService.getAllEvents();
const eventsByCategory = events.reduce((acc, event) => {
  const catName = event.category_name || 'Uncategorized';
  acc[catName] = (acc[catName] || 0) + 1;
  return acc;
}, {});
Object.entries(eventsByCategory).forEach(([cat, count]) =>
  console.log(\`- \${cat}: \${count} events\`)
);

console.log(\n=== Database Statistics ===\);
console.log(DatabaseService.getDatabaseStats());
"
```

Let me fix that syntax error:

```
cd "/Users/kelyrufino/Library/Mobile
Documents/com~apple~CloudDocs/University Classes/Year1/Web
Tecnologies/Assessment" && node -e "
const DatabaseService = require('./database/service');

console.log('=== Database Health Check ===');
console.log(DatabaseService.checkDatabaseHealth());

console.log('\n=== Categories ===');
const categories = DatabaseService.getAllCategories();
categories.forEach(cat => console.log(\`- \${cat.name} (ID:
\${cat.id})\`));

console.log('\n=== Events Count by Category ===');
const events = DatabaseService.getAllEvents();
const eventsByCategory = events.reduce((acc, event) => {
  const catName = event.category_name || 'Uncategorized';
  acc[catName] = (acc[catName] || 0) + 1;
  return acc;
}, {});
Object.entries(eventsByCategory).forEach(([cat, count]) =>
  console.log(\`- \${cat}: \${count} events\`)
);
```



```
    console.log('\n=== Database Statistics ===');
    console.log(DatabaseService.getDatabaseStats());
    "
```

Perfect! Let me also show the current directory structure to see all the files we've created:

✓ Read database

Summary

I've successfully set up Better-SQLite3 support for your project with a complete database infrastructure. Here's what has been implemented:

✅ Installation & Configuration

- Installed `better-sqlite3` dependency in package.json
- Added database scripts to package.json (`db:init`, `db:seed`, `db:reset`)

✅ Database Structure

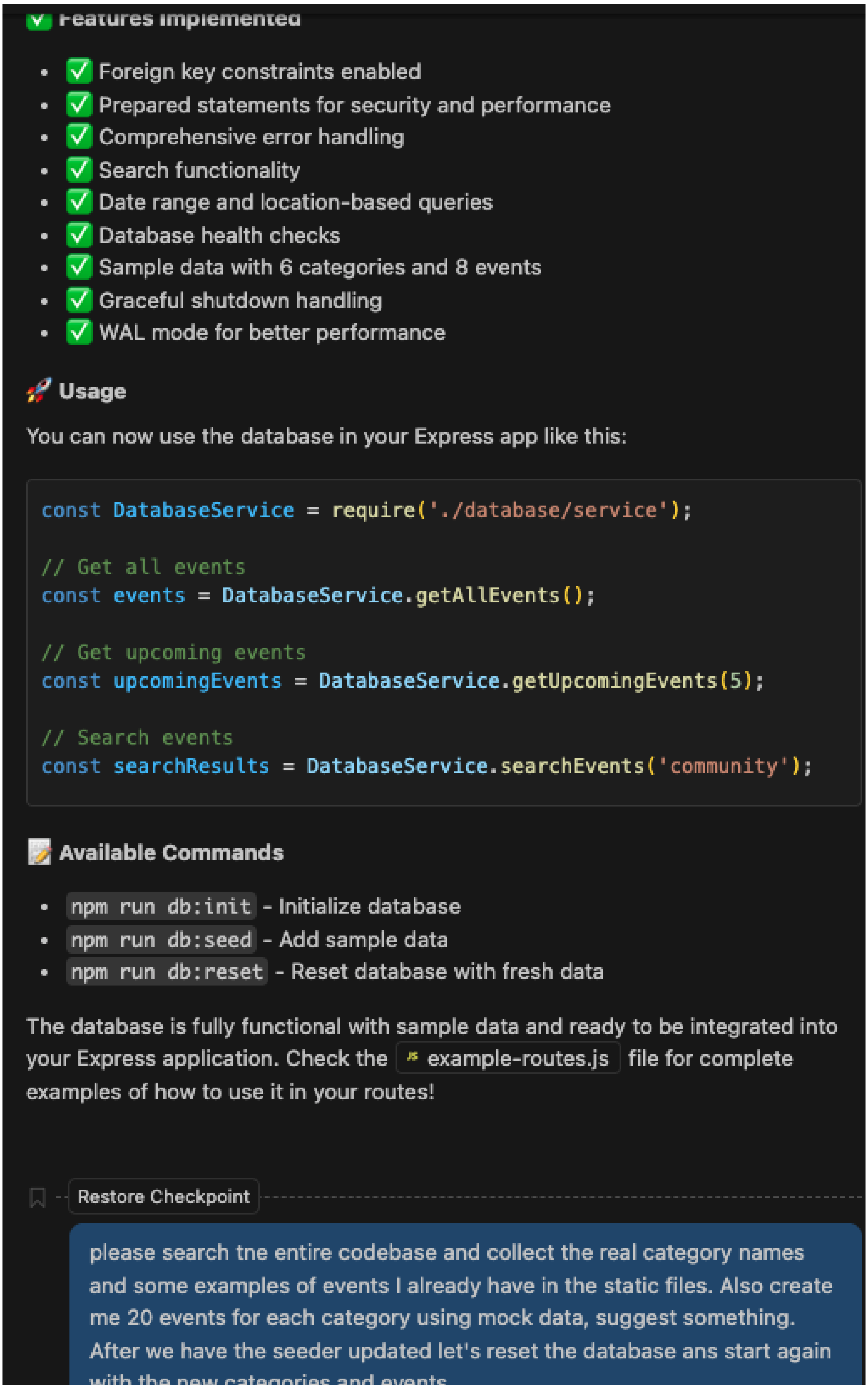
- **Categories table**: `id`, `name`, `description`, `image_url`, timestamps
- **Events table**: `id`, `title`, `description`, `content`, `image_url`, `location`, `date`, `start_time`, `end_time`, `address`, `category_id`, timestamps
- **Foreign key relationship**: `events.category_id` → `categories.id` with ON DELETE SET NULL

✅ Database Files Created

1. connection.js - Database connection with proper configuration
2. init.js - Database initialization with table creation
3. queries.js - Prepared statements for all database operations
4. service.js - High-level service layer for easy integration
5. seed.js - Sample data seeding
6. example-routes.js - Integration examples for Express routes
7. README.md - Complete documentation

✅ Features Implemented

- ✅ Foreign key constraints enabled
- ✅ Prepared statements for security and performance
- ✅ Comprehensive error handling
- ✅ Search functionality
- ✅ Date range and location-based queries
- ✅ Database health checks



✅ Features Implemented

- ✅ Foreign key constraints enabled
- ✅ Prepared statements for security and performance
- ✅ Comprehensive error handling
- ✅ Search functionality
- ✅ Date range and location-based queries
- ✅ Database health checks
- ✅ Sample data with 6 categories and 8 events
- ✅ Graceful shutdown handling
- ✅ WAL mode for better performance

🚀 Usage

You can now use the database in your Express app like this:

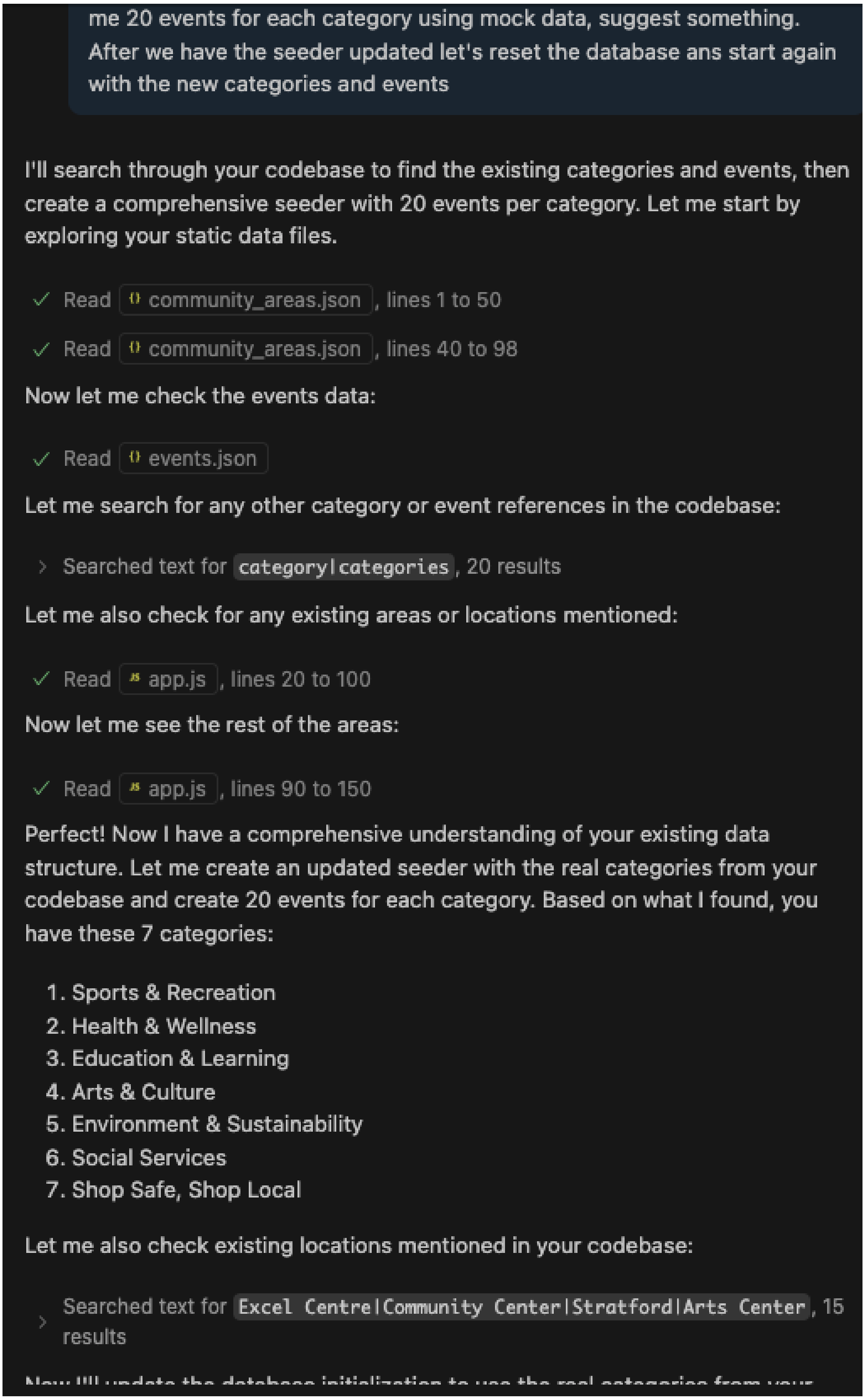
```
const DatabaseService = require('./database/service');

// Get all events
const events = DatabaseService.getAllEvents();

// Get upcoming events
const upcomingEvents = DatabaseService.getUpcomingEvents(5);

// Search events
const searchResults = DatabaseService.searchEvents('community');
```

📝 Available Commands

- `npm run db:init` - Initialize database
- `npm run db:seed` - Add sample data
- `npm run db:reset` - Reset database with fresh data

The database is fully functional with sample data and ready to be integrated into your Express application. Check the `example-routes.js` file for complete examples of how to use it in your routes!

please search tne entire codebase and collect the real category names and some examples of events I already have in the static files. Also create me 20 events for each category using mock data, suggest something. After we have the seeder updated let's reset the database ans start again with the new categories and events



me 20 events for each category using mock data, suggest something. After we have the seeder updated let's reset the database ans start again with the new categories and events

I'll search through your codebase to find the existing categories and events, then create a comprehensive seeder with 20 events per category. Let me start by exploring your static data files.

✓ Read `community_areas.json`, lines 1 to 50

✓ Read `community_areas.json`, lines 40 to 98

Now let me check the events data:

✓ Read `events.json`

Let me search for any other category or event references in the codebase:

› Searched text for `category|categories`, 20 results

Let me also check for any existing areas or locations mentioned:

✓ Read `app.js`, lines 20 to 100

Now let me see the rest of the areas:

✓ Read `app.js`, lines 90 to 150

Perfect! Now I have a comprehensive understanding of your existing data structure. Let me create an updated seeder with the real categories from your codebase and create 20 events for each category. Based on what I found, you have these 7 categories:

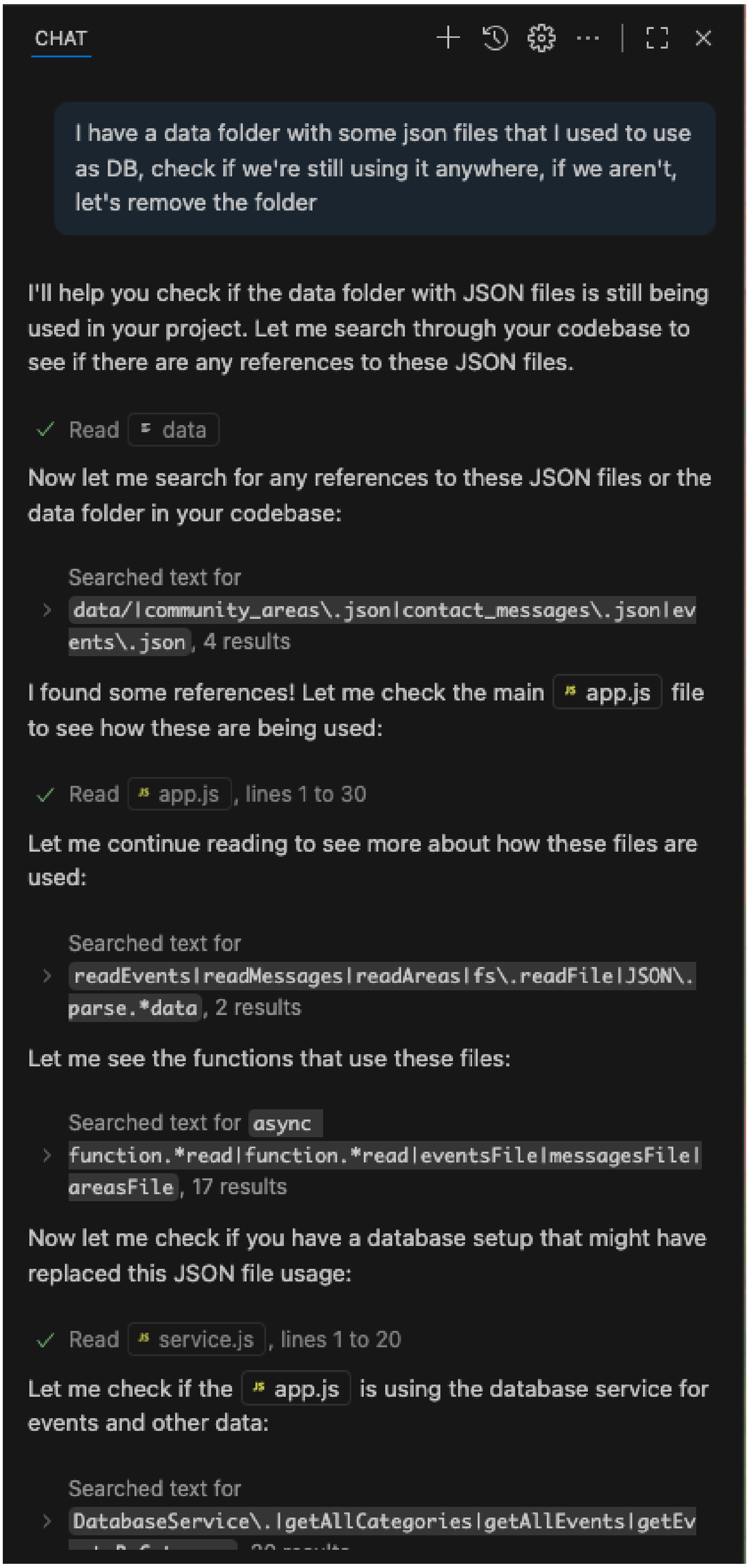
1. Sports & Recreation
2. Health & Wellness
3. Education & Learning
4. Arts & Culture
5. Environment & Sustainability
6. Social Services
7. Shop Safe, Shop Local

Let me also check existing locations mentioned in your codebase:

› Searched text for `Excel Centre|Community Center|Stratford|Arts Center`, 15 results

Now I'll update the database initialization to use the real categories from your

The Ai was used to remove the JSON files as used to work as database, when I migrate to SQLITE3 I asked to AI remove the folders containing JSON files



I have a data folder with some json files that I used to use as DB, check if we're still using it anywhere, if we aren't, let's remove the folder

I'll help you check if the data folder with JSON files is still being used in your project. Let me search through your codebase to see if there are any references to these JSON files.

✓ Read data

Now let me search for any references to these JSON files or the data folder in your codebase:

Searched text for `data/|community_areas\.json|contact_messages\.json|events\.json`, 4 results

I found some references! Let me check the main app.js file to see how these are being used:

✓ Read app.js, lines 1 to 30

Let me continue reading to see more about how these files are used:

Searched text for `readEvents|readMessages|readAreas|fs\.readFile|JSON\.parse.*data`, 2 results

Let me see the functions that use these files:

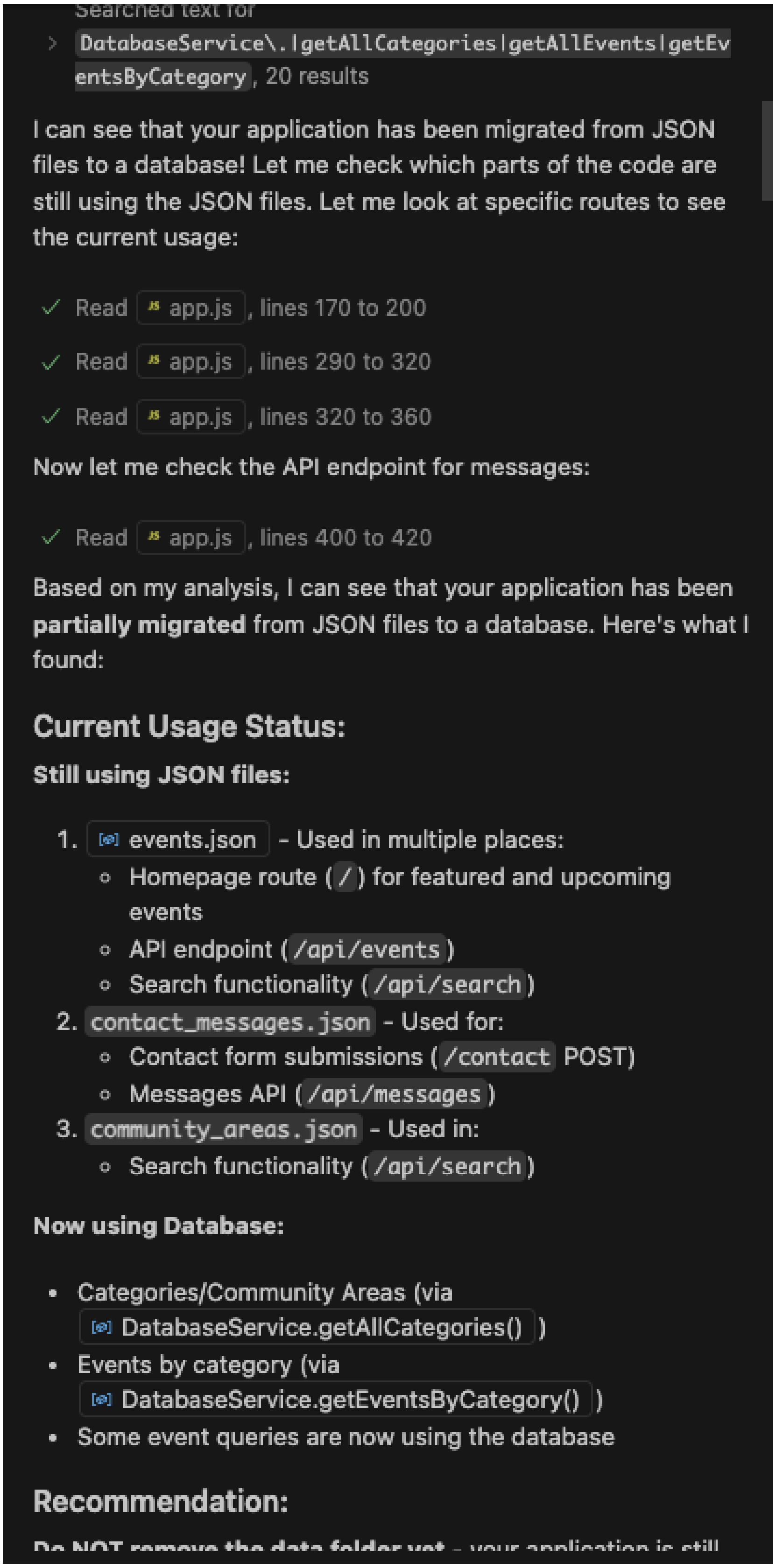
Searched text for `async function.*read|function.*read|eventsFile|messagesFile|areasFile`, 17 results

Now let me check if you have a database setup that might have replaced this JSON file usage:

✓ Read service.js, lines 1 to 20

Let me check if the app.js is using the database service for events and other data:

Searched text for `DatabaseService\.|getAllCategories|getAllEvents|getEv`



Searched text for `DatabaseService\.|getAllCategories|getAllEvents|getEventsByCategory`, 20 results

I can see that your application has been migrated from JSON files to a database! Let me check which parts of the code are still using the JSON files. Let me look at specific routes to see the current usage:

✓ Read app.js, lines 170 to 200

✓ Read app.js, lines 290 to 320

✓ Read app.js, lines 320 to 360

Now let me check the API endpoint for messages:

✓ Read app.js, lines 400 to 420

Based on my analysis, I can see that your application has been **partially migrated** from JSON files to a database. Here's what I found:

Current Usage Status:

Still using JSON files:

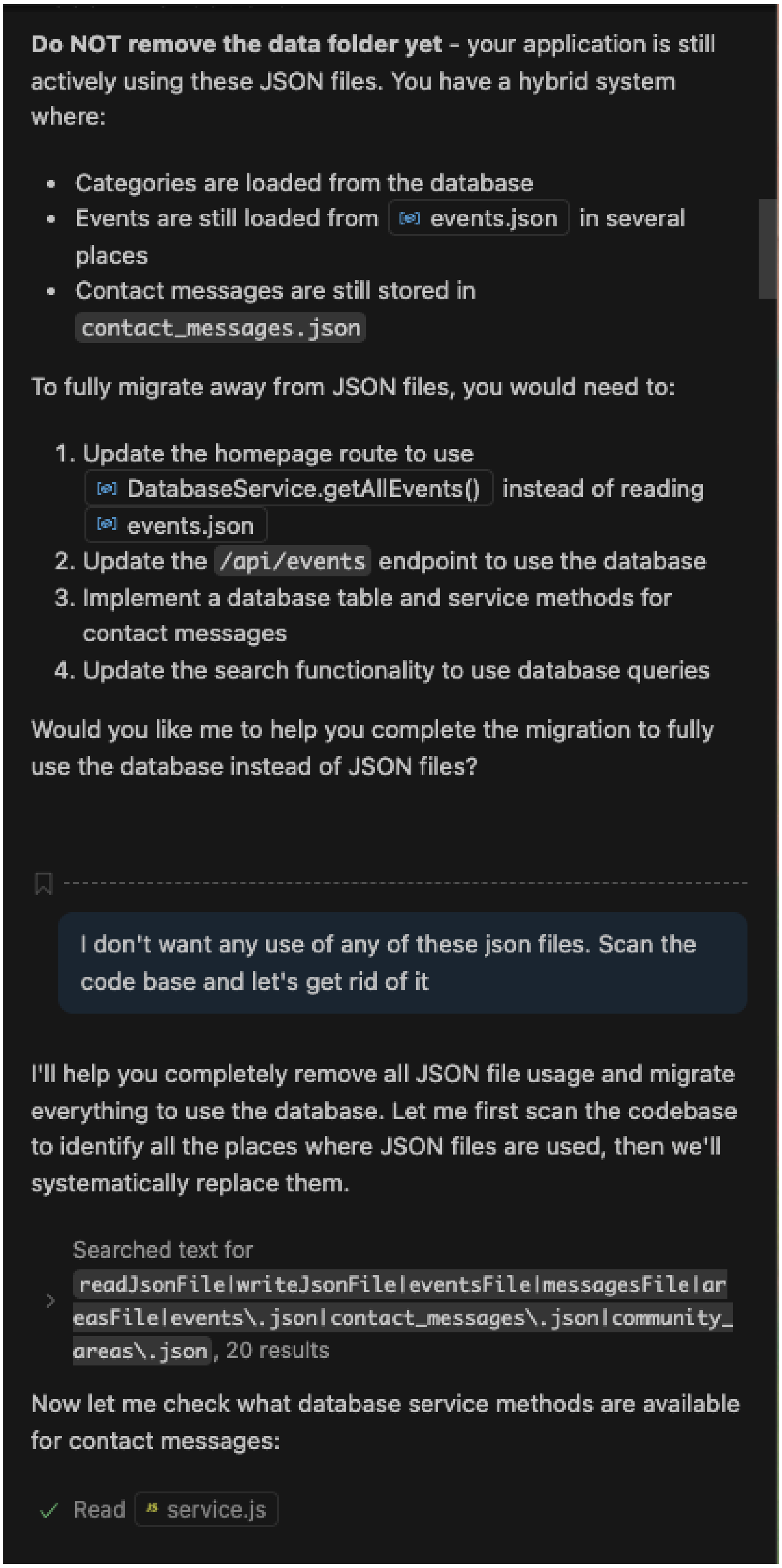
1. events.json - Used in multiple places:
   - Homepage route (`/`) for featured and upcoming events
   - API endpoint (`/api/events`)
   - Search functionality (`/api/search`)
2. `contact_messages.json` - Used for:
   - Contact form submissions (`/contact` POST)
   - Messages API (`/api/messages`)
3. `community_areas.json` - Used in:
   - Search functionality (`/api/search`)

Now using Database:

- Categories/Community Areas (via DatabaseService.getAllCategories())
- Events by category (via DatabaseService.getEventsByCategory())
- Some event queries are now using the database

Recommendation:

Do NOT remove the data folder yet – your application is still



Do NOT remove the data folder yet - your application is still actively using these JSON files. You have a hybrid system where:

- Categories are loaded from the database
- Events are still loaded from `events.json` in several places
- Contact messages are still stored in `contact_messages.json`

To fully migrate away from JSON files, you would need to:

1. Update the homepage route to use `DatabaseService.getAllEvents()` instead of reading `events.json`
2. Update the `/api/events` endpoint to use the database
3. Implement a database table and service methods for contact messages
4. Update the search functionality to use database queries

Would you like me to help you complete the migration to fully use the database instead of JSON files?

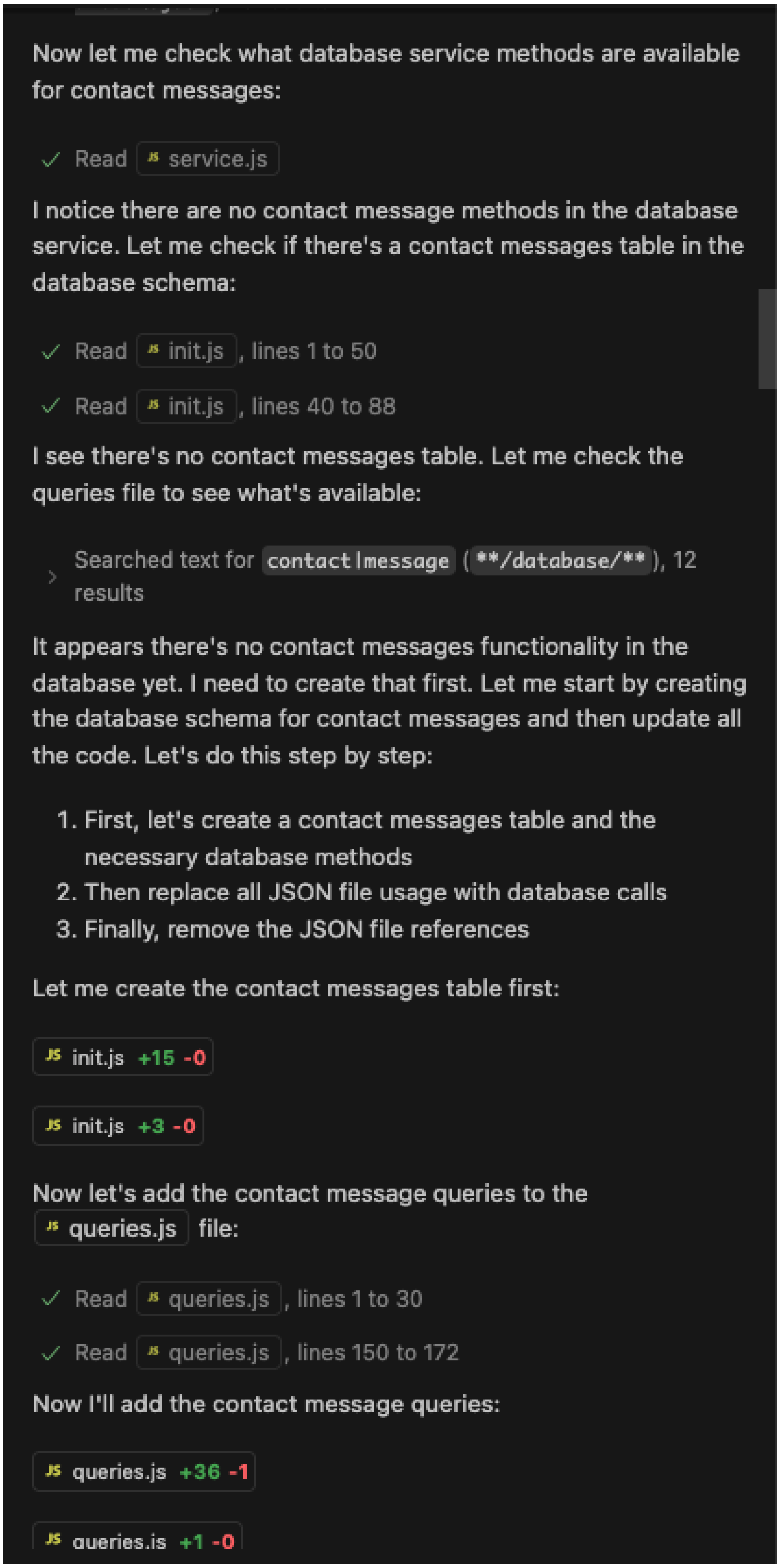
I don't want any use of any of these json files. Scan the code base and let's get rid of it

I'll help you completely remove all JSON file usage and migrate everything to use the database. Let me first scan the codebase to identify all the places where JSON files are used, then we'll systematically replace them.

Searched text for `readJsonFile|writeJsonFile|eventsFile|messagesFile|areasFile|events\.json|contact_messages\.json|community_areas\.json`, 20 results

Now let me check what database service methods are available for contact messages:

✓ Read `service.js`



Now let me check what database service methods are available for contact messages:

✓ Read service.js

I notice there are no contact message methods in the database service. Let me check if there's a contact messages table in the database schema:

✓ Read init.js, lines 1 to 50

✓ Read init.js, lines 40 to 88

I see there's no contact messages table. Let me check the queries file to see what's available:

› Searched text for `contact|message` (`**/database/**`), 12 results

It appears there's no contact messages functionality in the database yet. I need to create that first. Let me start by creating the database schema for contact messages and then update all the code. Let's do this step by step:

1. First, let's create a contact messages table and the necessary database methods
2. Then replace all JSON file usage with database calls
3. Finally, remove the JSON file references

Let me create the contact messages table first:

init.js +15 -0

init.js +3 -0

Now let's add the contact message queries to the queries.js file:

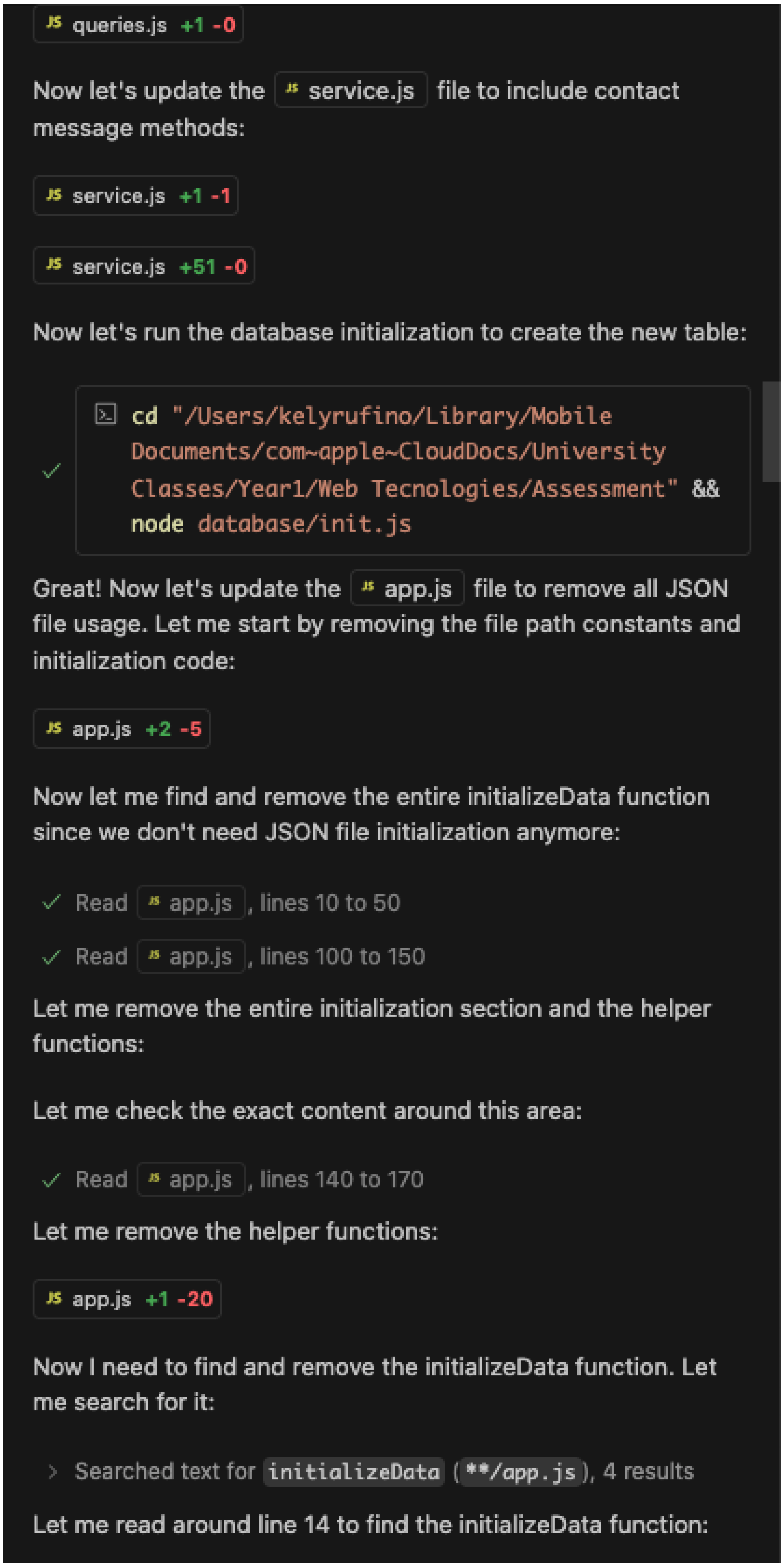
✓ Read queries.js, lines 1 to 30

✓ Read queries.js, lines 150 to 172

Now I'll add the contact message queries:

queries.js +36 -1

queries.js +1 -0



queries.js +1 -0

Now let's update the service.js file to include contact message methods:

service.js +1 -1

service.js +51 -0

Now let's run the database initialization to create the new table:

✓

```
cd "/Users/kelyrufino/Library/Mobile Documents/com~apple~CloudDocs/University Classes/Year1/Web Tecnologies/Assessment" && node database/init.js
```

Great! Now let's update the app.js file to remove all JSON file usage. Let me start by removing the file path constants and initialization code:

app.js +2 -5

Now let me find and remove the entire initializeData function since we don't need JSON file initialization anymore:

✓ Read app.js, lines 10 to 50

✓ Read app.js, lines 100 to 150

Let me remove the entire initialization section and the helper functions:

Let me check the exact content around this area:

✓ Read app.js, lines 140 to 170

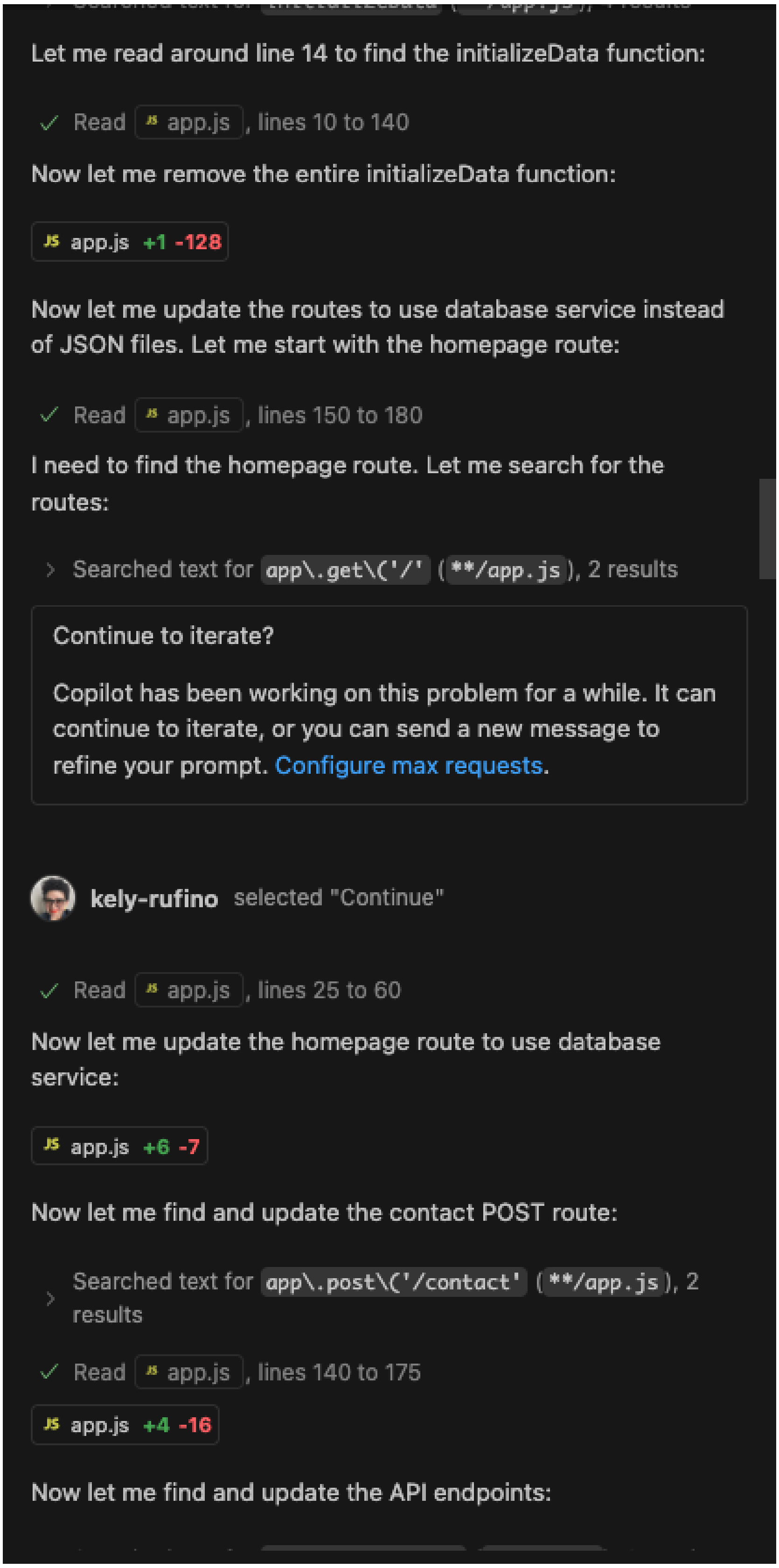
Let me remove the helper functions:

app.js +1 -20

Now I need to find and remove the initializeData function. Let me search for it:

› Searched text for `initializeData` (`**/app.js`), 4 results

Let me read around line 14 to find the initializeData function:



Let me read around line 14 to find the initializeData function:

✓ Read `app.js`, lines 10 to 140

Now let me remove the entire initializeData function:

app.js +1 -128

Now let me update the routes to use database service instead of JSON files. Let me start with the homepage route:

✓ Read `app.js`, lines 150 to 180

I need to find the homepage route. Let me search for the routes:

Searched text for `app\.get\('/'` (`**/app.js`), 2 results

Continue to iterate?

Copilot has been working on this problem for a while. It can continue to iterate, or you can send a new message to refine your prompt. Configure max requests.

kely-rufino selected "Continue"

✓ Read `app.js`, lines 25 to 60

Now let me update the homepage route to use database service:

app.js +6 -7

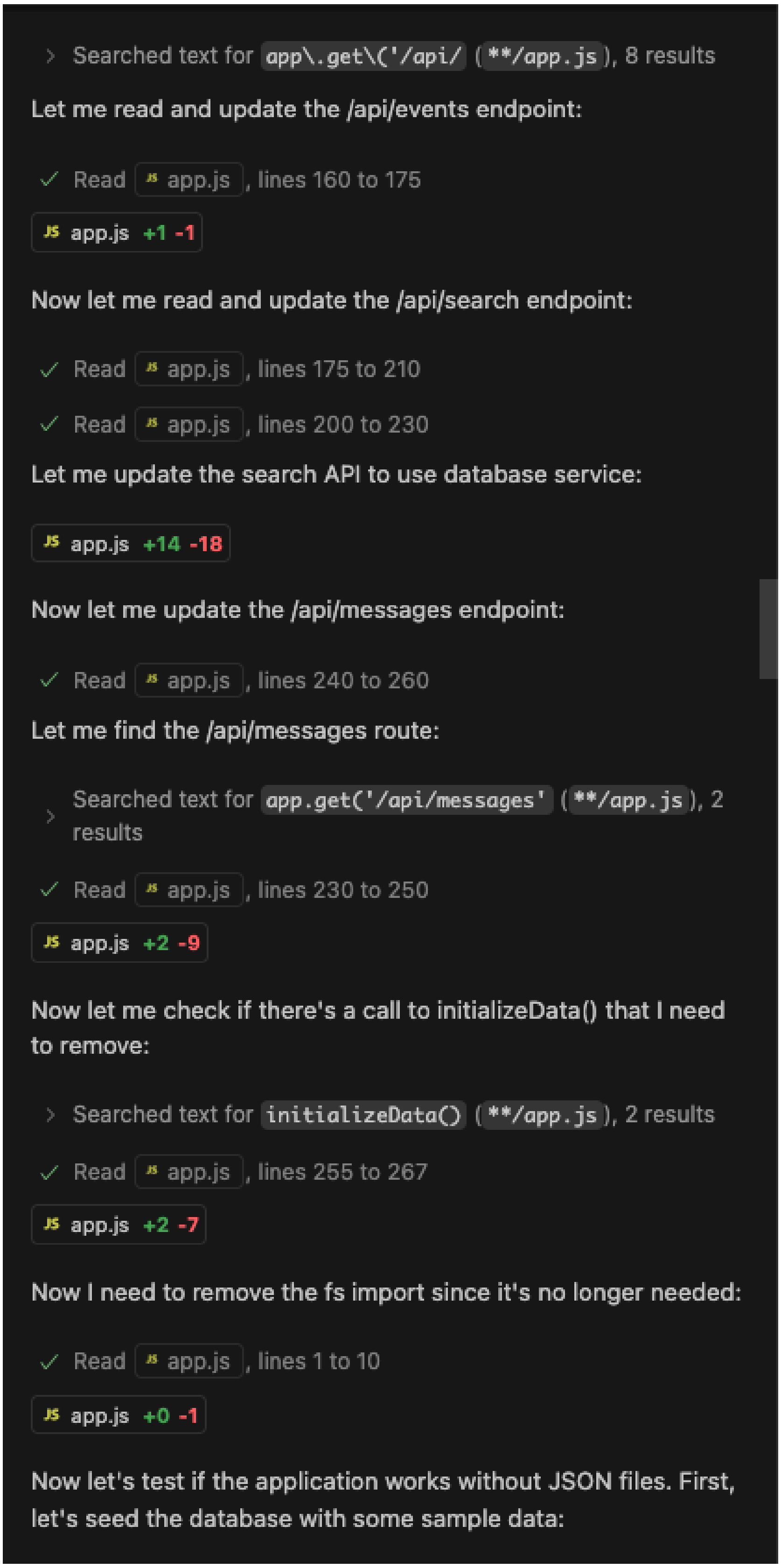
Now let me find and update the contact POST route:

Searched text for `app\.post\('/contact'` (`**/app.js`), 2 results

✓ Read `app.js`, lines 140 to 175

app.js +4 -16

Now let me find and update the API endpoints:



> Searched text for `app\.get\('/api/` (`**/app.js`), 8 results

Let me read and update the /api/events endpoint:

✓ Read `app.js`, lines 160 to 175

app.js +1 -1

Now let me read and update the /api/search endpoint:

✓ Read `app.js`, lines 175 to 210

✓ Read `app.js`, lines 200 to 230

Let me update the search API to use database service:

app.js +14 -18

Now let me update the /api/messages endpoint:

✓ Read `app.js`, lines 240 to 260

Let me find the /api/messages route:

> Searched text for `app.get('/api/messages'` (`**/app.js`), 2 results

✓ Read `app.js`, lines 230 to 250

app.js +2 -9

Now let me check if there's a call to initializeData() that I need to remove:

> Searched text for `initializeData()` (`**/app.js`), 2 results

✓ Read `app.js`, lines 255 to 267

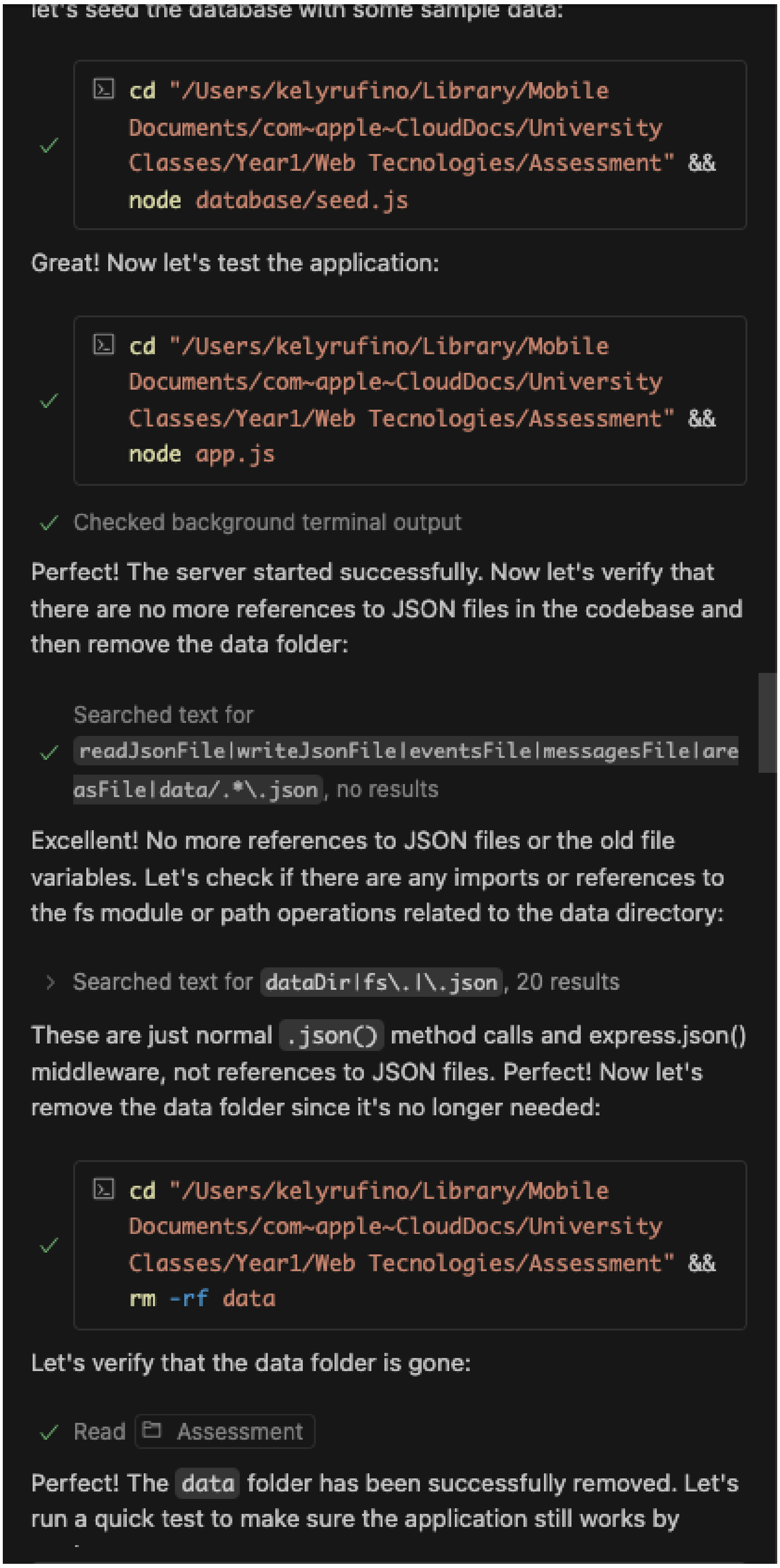
app.js +2 -7

Now I need to remove the fs import since it's no longer needed:

✓ Read `app.js`, lines 1 to 10

app.js +0 -1

Now let's test if the application works without JSON files. First, let's seed the database with some sample data:



let's seed the database with some sample data:

```
cd "/Users/kelyrufino/Library/Mobile Documents/com~apple~CloudDocs/University Classes/Year1/Web Tecnologies/Assessment" && node database/seed.js
```

Great! Now let's test the application:

```
cd "/Users/kelyrufino/Library/Mobile Documents/com~apple~CloudDocs/University Classes/Year1/Web Tecnologies/Assessment" && node app.js
```

✓ Checked background terminal output

Perfect! The server started successfully. Now let's verify that there are no more references to JSON files in the codebase and then remove the data folder:

✓ Searched text for `readJsonFile|writeJsonFile|eventsFile|messagesFile|areasFile|data/.*\.json`, no results

Excellent! No more references to JSON files or the old file variables. Let's check if there are any imports or references to the fs module or path operations related to the data directory:

Searched text for `dataDir|fs\.|\.json`, 20 results

These are just normal `.json()` method calls and express.json() middleware, not references to JSON files. Perfect! Now let's remove the data folder since it's no longer needed:

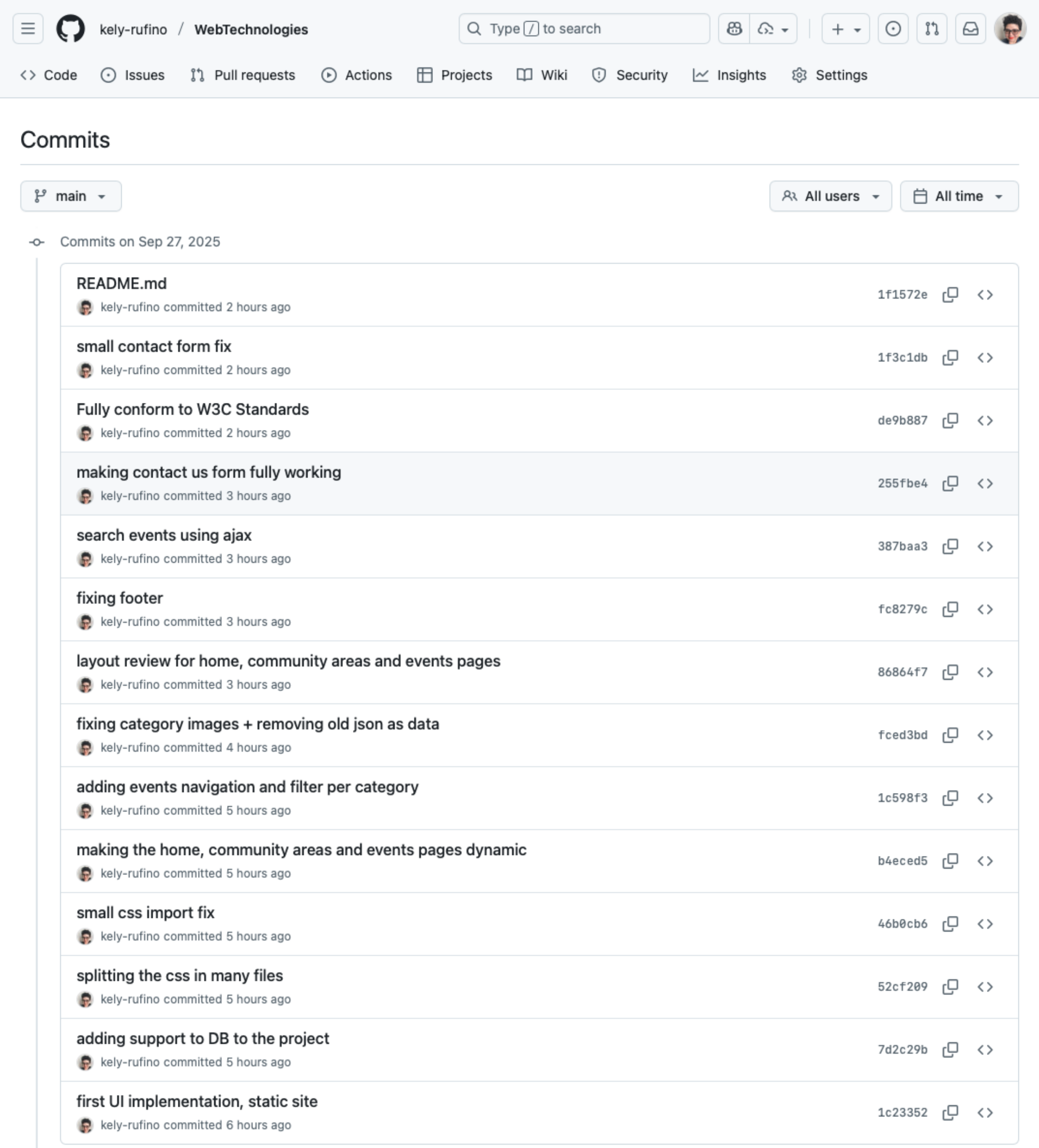
```
cd "/Users/kelyrufino/Library/Mobile Documents/com~apple~CloudDocs/University Classes/Year1/Web Tecnologies/Assessment" && rm -rf data
```

Let's verify that the data folder is gone:

✓ Read Assessment

Perfect! The `data` folder has been successfully removed. Let's run a quick test to make sure the application still works by

The GitHub repository can be seen on: **https://github.com/kely-rufino/WebTechnologies**



Commits

Commits on Sep 27, 2025

README.md
kely-rufino committed 2 hours ago 1f1572e

small contact form fix
kely-rufino committed 2 hours ago 1f3c1db

Fully conform to W3C Standards
kely-rufino committed 2 hours ago de9b887

making contact us form fully working
kely-rufino committed 3 hours ago 255fbe4

search events using ajax
kely-rufino committed 3 hours ago 387baa3

fixing footer
kely-rufino committed 3 hours ago fc8279c

layout review for home, community areas and events pages
kely-rufino committed 3 hours ago 86864f7

fixing category images + removing old json as data
kely-rufino committed 4 hours ago fced3bd

adding events navigation and filter per category
kely-rufino committed 5 hours ago 1c598f3

making the home, community areas and events pages dynamic
kely-rufino committed 5 hours ago b4eced5

small css import fix
kely-rufino committed 5 hours ago 46b0cb6

splitting the css in many files
kely-rufino committed 5 hours ago 52cf209

adding support to DB to the project
kely-rufino committed 5 hours ago 7d2c29b

first UI implementation, static site
kely-rufino committed 6 hours ago 1c23352

AI was not used to create the main code on this project.