Sprints

– Task 27 –

REST API Development and Apache Configuration

Name: Basel Amr Barakat
Email: baselamr52@gmail.com

Project Overview

- Build a REST API using Flask with GET and Post methods
- Handle JSON responses and errors gracefully
- Configure Apache Virtual Hosts to serve the API

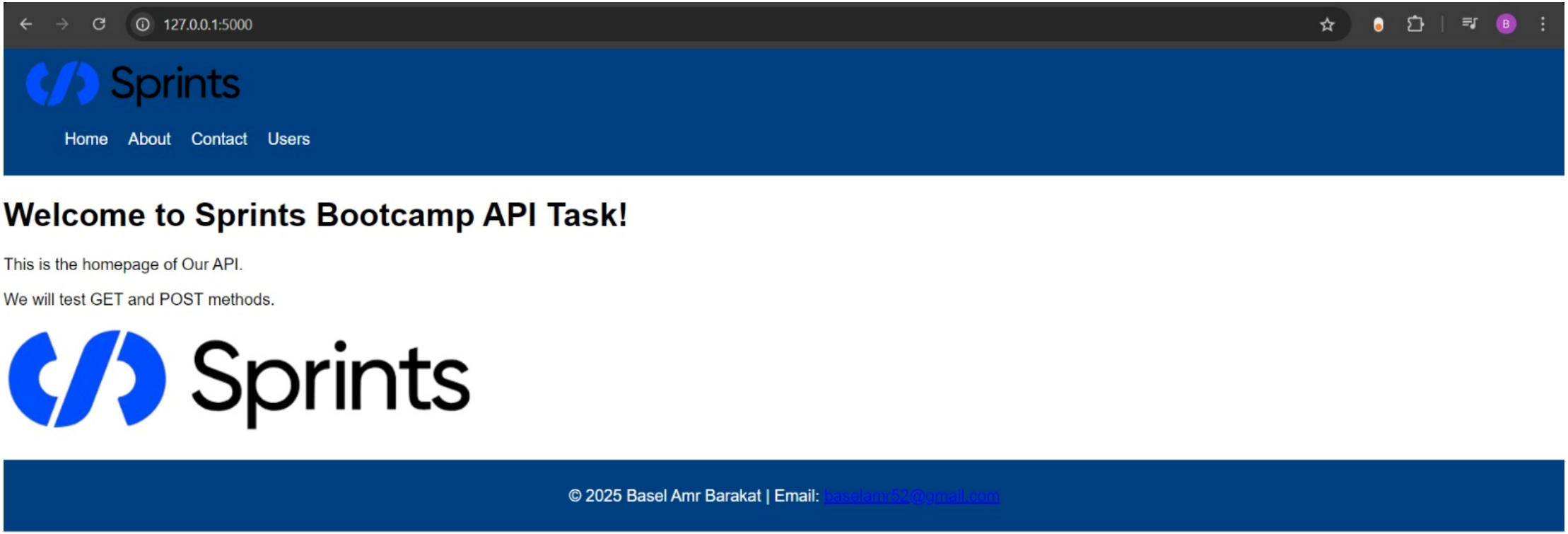
API Design

- **/users (GET) :** List all users
- **/users (POST) :** Add a new user
- **/users/<id> (DELETE) :** Delete a user
- **/users (Search)** : Search for a user by name or id

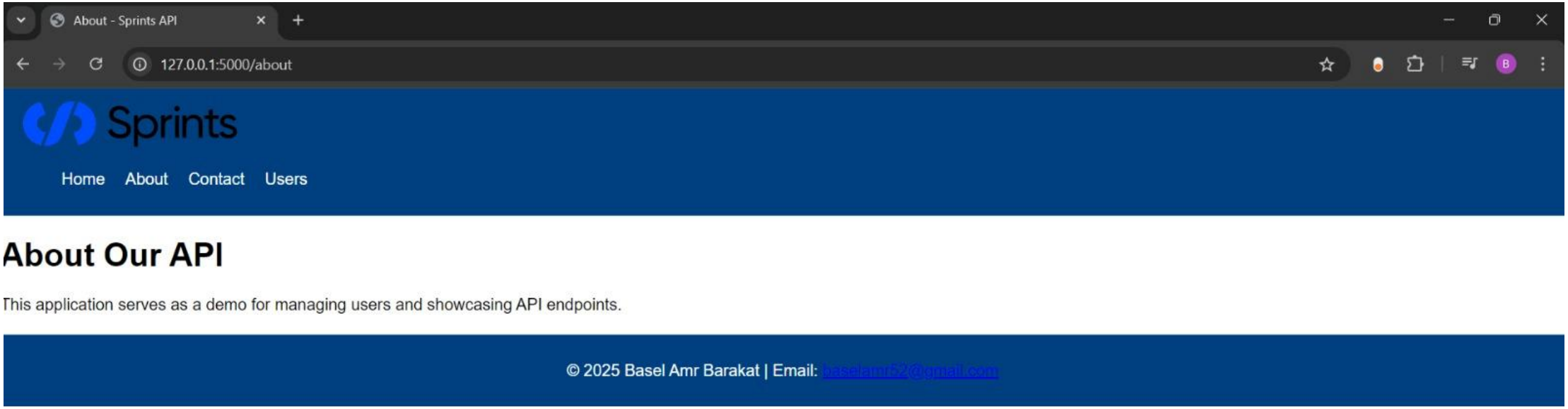
API Features

- SQLite database integration
- Input validation for user data (ID, name, email)
- JSON and HTML response handling
- Error handling with try-except blocks

Home Page

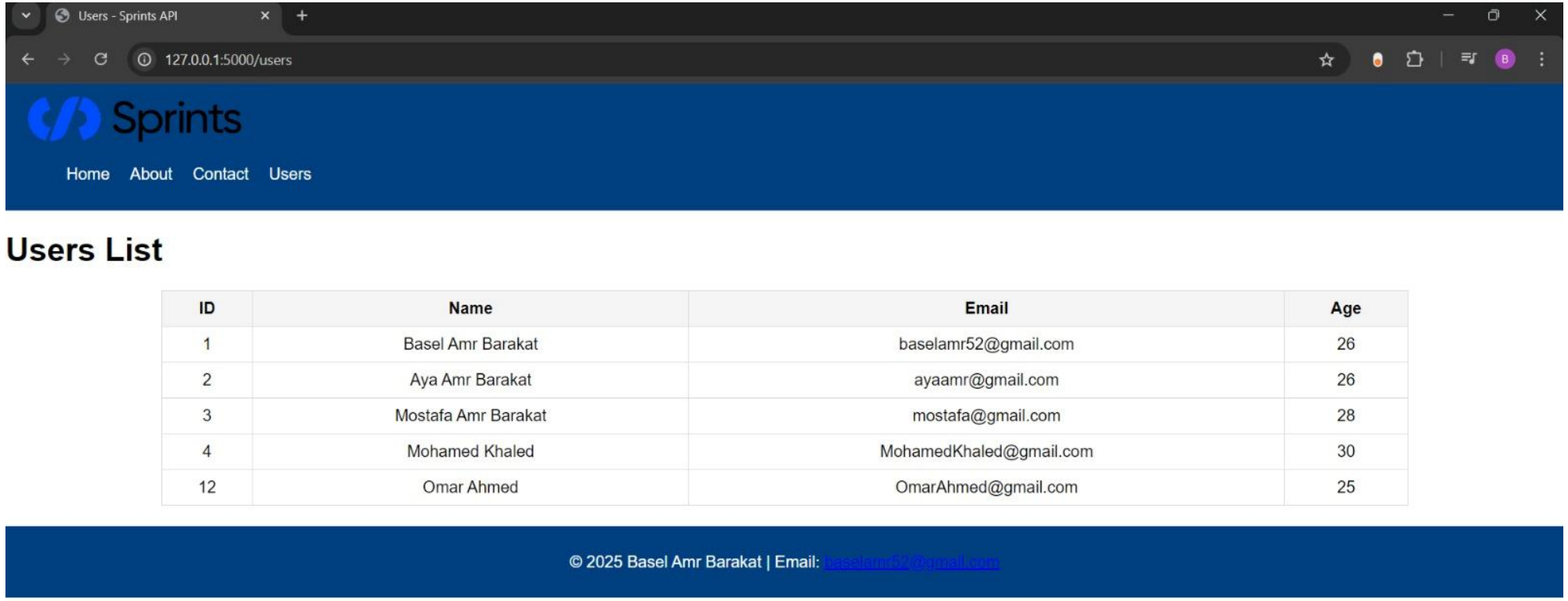

127.0.0.1:5000
Sprints
Home
About
Contact
Users
Welcome to Sprints Bootcamp API Task!
This is the homepage of Our API.
We will test GET and POST methods.
Sprints
© 2025 Basel Amr Barakat | Email: baselamr52@gmail.com

About Page


About - Sprints API
127.0.0.1:5000/about
Sprints
Home
About
Contact
Users
About Our API
This application serves as a demo for managing users and showcasing API endpoints.
© 2025 Basel Amr Barakat | Email: baselamr52@gmail.com

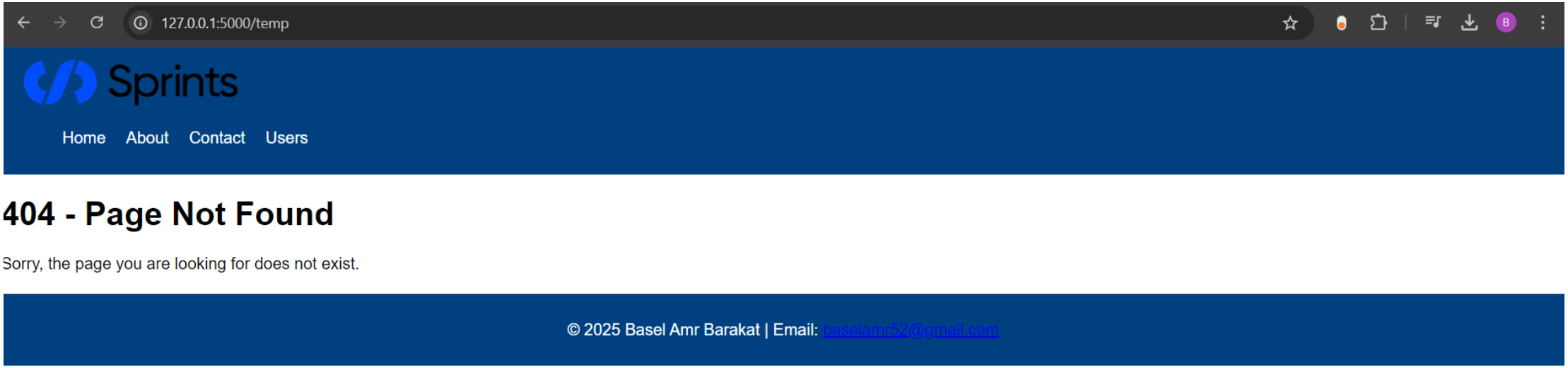
Contact Page

User List Page



ID	Name	Email	Age
1	Basel Amr Barakat	baselamr52@gmail.com	26
2	Aya Amr Barakat	ayaamr@gmail.com	26
3	Mostafa Amr Barakat	mostafa@gmail.com	28
4	Mohamed Khaled	MohamedKhaled@gmail.com	30
12	Omar Ahmed	OmarAhmed@gmail.com	25

404 Page


127.0.0.1:5000/temp
Sprints
Home
About
Contact
Users
404 - Page Not Found
Sorry, the page you are looking for does not exist.
© 2025 Basel Amr Barakat | Email: baselamr52@gmail.com

Some Code Screen Shoots

01_Get_All_Users

```python
# GET Route: Retrieve all users
@app.route('/users', methods=['GET'])
def get_all_users():
    """
    Users Page Route
    Description: Displays all users in an HTML page or returns JSON data based on the request.
    Query Parameters:
        format (str): 'json' to return JSON response, otherwise returns HTML.
    Returns:
        HTML: User list rendered in a template.
        JSON: User list in JSON format.
    """
    try:
        conn = sqlite3.connect('users.db')
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users")
        users = cursor.fetchall()
        conn.close()

        # Convert users list to a dictionary format
        user_list = [{"id": user[0], "name": user[1], "email": user[2], "age": user[3]} for user in users]

        # Check if 'format=json' is in the query string
        if request.args.get('format') == 'json':
            return jsonify(user_list), 200
        else:
            return render_template('users.html', users=users)

    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

02_Add_User

```python
# POST Route: Add a new user
@app.route('/users', methods=['POST'])
def add_user():
    """
    Add a new user to the SQLite database.
    Args:
        JSON: A JSON payload containing "id", "name", and "email".
    Description:
        - Validates required fields.
        - Checks for unique 'id' and 'email'.
        - Validates 'name' contains only alphabetic characters.
        - Validates 'email' format.
    Returns:
        JSON: A JSON response containing the status of the operation.
    """
    try:
        new_user = request.json  # Get JSON data from the POST request

        # Validate required fields
        if not new_user.get('id') or not new_user.get('name') or not new_user.get('email') or not new_user.get('age'):
            missing_fields = [field for field in ['id', 'name', 'email', 'age'] if field not in new_user]
            return jsonify({"error": f"Missing required field(s): {', '.join(missing_fields)}"}), 400  # 400: Bad Request

        # Validate name format (only letters and spaces)
        if not re.match("^[A-Za-z ]+$", new_user['name']):
            return jsonify({"error": "The 'name' field must only contain alphabetic characters and spaces."}), 400

        # Validate email format
        if not re.match(r"[^@]+@[^@]+\.[^@]+", new_user['email']):
            return jsonify({"error": "Invalid email format."}), 400
```

```python
        # Validate email format
        if not re.match(r"[^@]+@[^@]+\.[^@]+", new_user['email']):
            return jsonify({"error": "Invalid email format."}), 400

        # Check if ID is unique
        conn = sqlite3.connect('users.db')
        cursor = conn.cursor()
        cursor.execute("SELECT id FROM users WHERE id=?", (new_user['id'],))
        if cursor.fetchone():
            conn.close()
            return jsonify({"error": "A user with this 'id' already exists."}), 400

        # Check if email is unique
        cursor.execute("SELECT email FROM users WHERE email=?", (new_user['email'],))
        if cursor.fetchone():
            conn.close()
            return jsonify({"error": "A user with this 'email' already exists."}), 400

        # Add the new user to the database
        cursor.execute("INSERT INTO users (id, name, email, age) VALUES (?, ?, ?, ?)",
                       (new_user['id'], new_user['name'], new_user['email'], new_user['age']))
        conn.commit()
        conn.close()

        return jsonify({"message": "User added successfully"}), 201  # 201: Created
    except Exception as e:
        return jsonify({"error": str(e)}), 500  # 500: Internal Server Error
```

03_Delete_User

```python
@app.route('/users/<int:id>', methods=['DELETE'])
def delete_user(id):
    """
    Delete a user by ID.
    Args:
        id (int): The ID of the user to delete.
    Description:
        This endpoint deletes a user with the specified ID from the database.
    Returns:
        JSON: A JSON response with a success or error message.
    """
    try:
        # Connect to the database
        conn = sqlite3.connect('users.db')
        cursor = conn.cursor()

        # Check if the user exists
        cursor.execute("SELECT * FROM users WHERE id=?", (id,))
        user = cursor.fetchone()
        if not user:
            conn.close()
            return jsonify({"error": f"User with ID {id} not found."}), 404  # 404: Not Found

        # Delete the user
        cursor.execute("DELETE FROM users WHERE id=?", (id,))
        conn.commit()
        conn.close()

        return jsonify({"message": f"User with ID {id} deleted successfully."}), 200  # 200: OK
    except sqlite3.Error as db_error:
        return jsonify({"error": f"Database error: {str(db_error)}"}), 500  # 500: Internal Server Error
    except Exception as e:
        return jsonify({"error": f"Unexpected error: {str(e)}"}), 500  # 500: Internal Server Error
```

04_SearchUser

```python
# Search Route: Search for users by name or email
@app.route('/search', methods=['GET'])
def search_users():
    """
    Search for users by name or email.
    Args:
        query (str): The search query parameter.
    Description:
        This endpoint searches for users by name or email using a case-insensitive search.
    Returns:
        JSON: A list of matching users.
    """
    try:
        search_query = request.args.get('query')
        if not search_query:
            return jsonify({"error": "Query parameter is required"}), 400  # 400: Bad Request

        conn = sqlite3.connect('users.db')
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE name LIKE ? OR email LIKE ?",
                       ('%' + search_query + '%', '%' + search_query + '%'))
        users = cursor.fetchall()
        conn.close()

        if not users:
            return jsonify({"error": "No users found matching the search criteria"}), 404  # 404: Not Found

        # Convert users list to a dictionary format
        user_list = [{"id": user[0], "name": user[1], "email": user[2], "age": user[3]} for user in users]
        return jsonify(user_list), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 500  # 500: Internal Server Error
```

Pages Codes

Error and User Pages

```python
# Custom 404 Page
@app.errorhandler(404)
def page_not_found(e):
    """
    404 Error Page Route
    Description: Custom page for 404 errors.
    Returns:
        HTML: 404 error page content.
    """
    return render_template('404.html'), 404

# Users Page
@app.route('/users')
def list_users():
    """
    Users Page Route
    Description: Displays all users in a table format.
    Returns:
        HTML: List of users.
    """
    conn = sqlite3.connect('users.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM users")
    users = cursor.fetchall()
    conn.close()
    return render_template('users.html', users=users)
```

About and Settings Pages

```python
# About Page
@app.route('/about')
def about():
    """
    About Page Route
    Description: Provides information about the application.
    Returns:
        HTML: About page content.
    """
    return render_template('about.html')

# Settings Page
@app.route('/settings')
def settings():
    """
    Settings Page Route
    Description: Allows users to adjust application preferences
    Returns:
        HTML: Settings page.
    """
    return render_template('settings.html')
```

Home and Docs Pages

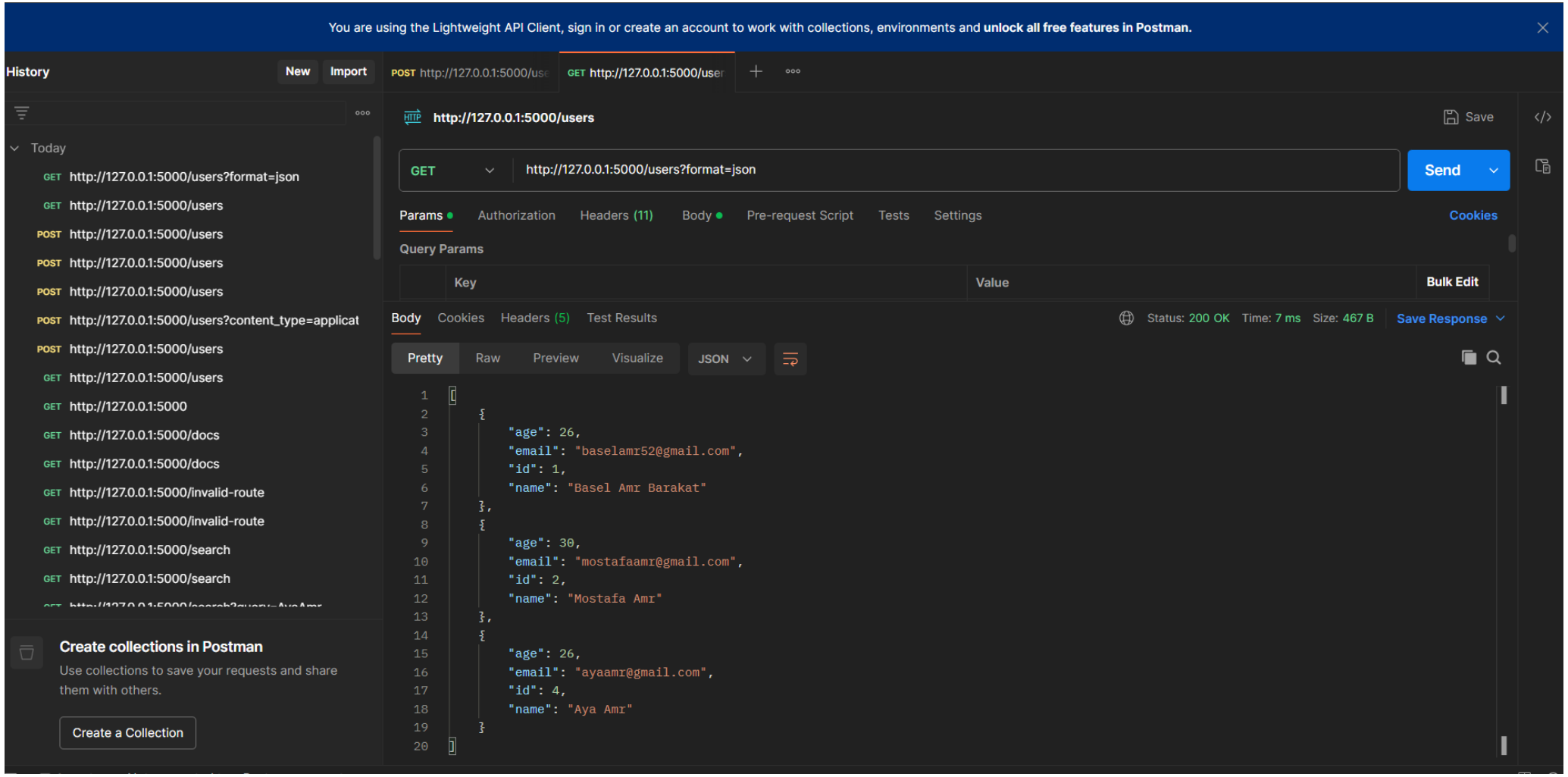
```python
# Home Page
@app.route('/')
def home():
    """
    Home Page Route
    Description: Displays a welcome message with an image.
    Returns:
        HTML: Welcome page with an image.
    """
    return render_template('index.html')

# API Documentation Page - /docs
@app.route('/docs')
def docs():
    return render_template('docs.html')
```

Invalid Pages

```python
# Handle invalid routes (404 error)
@app.errorhandler(404)
def page_not_found(e):
    """
    Handle 404 errors for undefined routes.
    Provides a custom error message and lists valid routes.

    Args:
        e: The exception object for the 404 error.

    Returns:
        JSON: List of valid routes.
    """
    valid_routes = [
        {"method": "GET", "route": "/"},
        {"method": "GET", "route": "/users"},
        {"method": "POST", "route": "/users"},
        {"method": "DELETE", "route": "/users/<id>"},
        {"method": "GET", "route": "/search"}
    ]
    return jsonify({
        "error": "Page not found",
        "message": "The route you entered does not exist.",
        "valid_routes": valid_routes
    }), 404
```

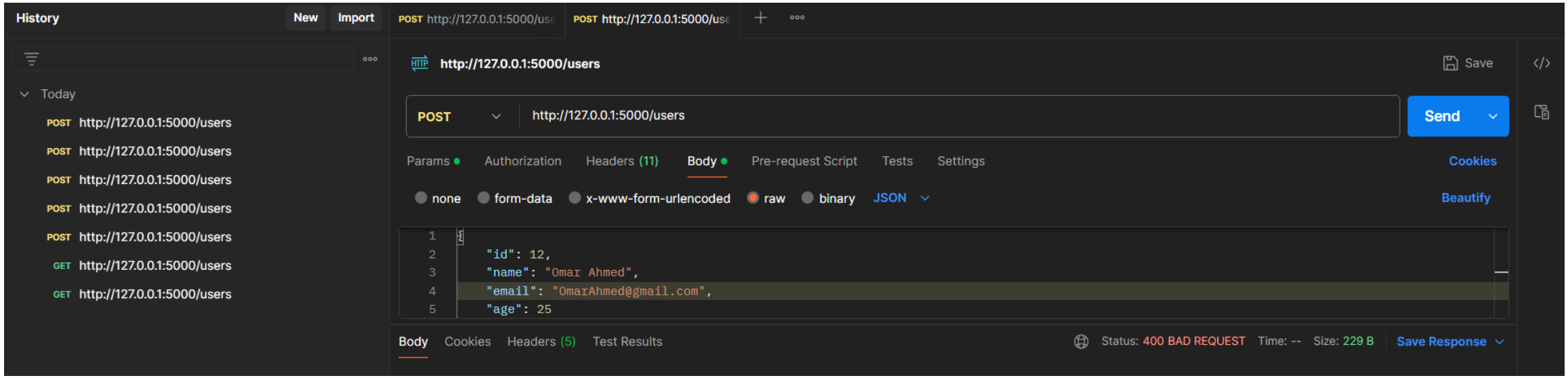
Testing GET Method

1.1 Successfully Getting All Users Postman



Testing POST Method

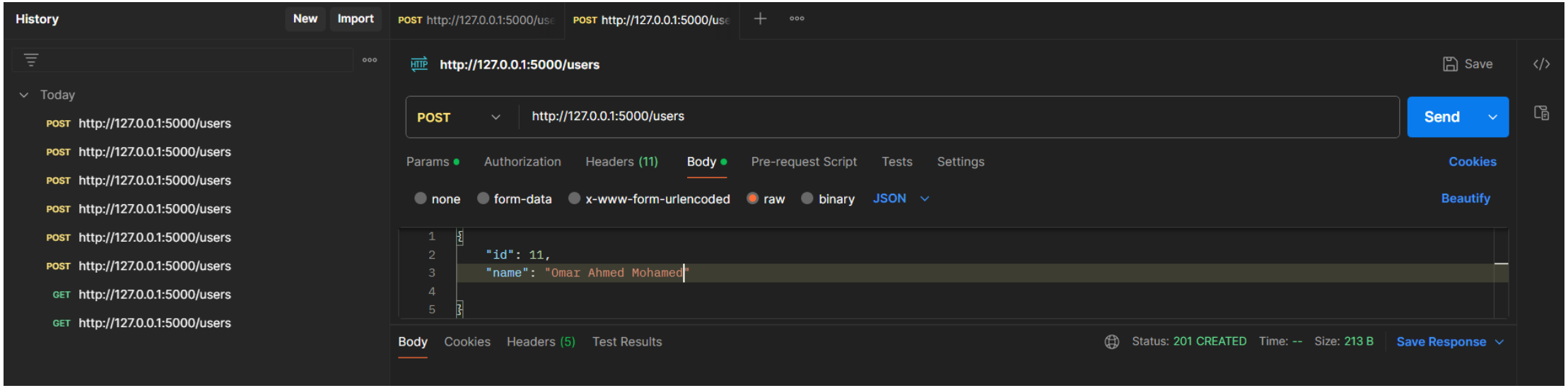
2.1 Successful User Creation

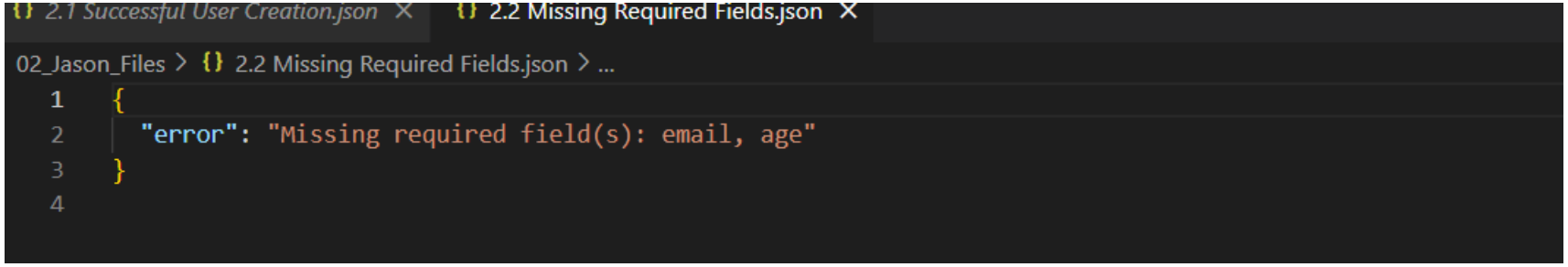


02_Jason_Files > {} 2.1 Successful User Creation.json > ...

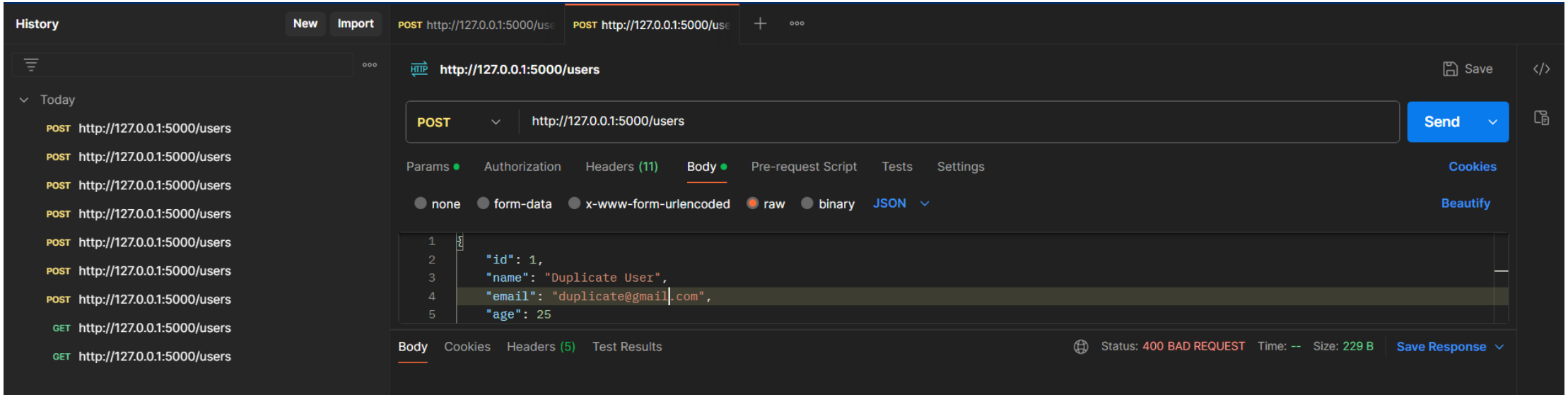
```
{
  "message": "User added successfully"
}
```

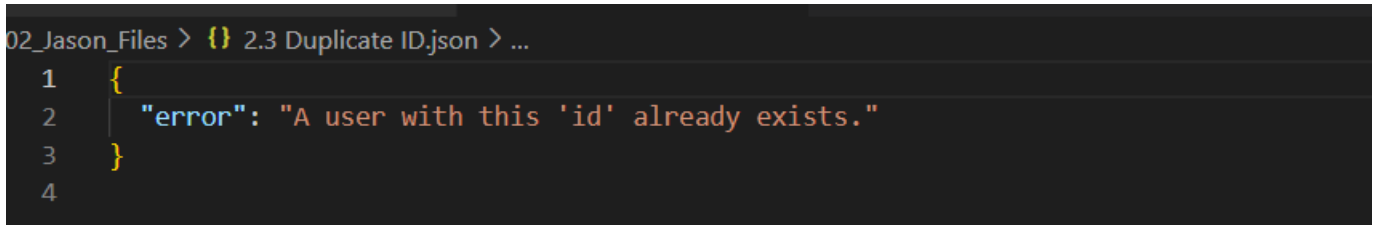
2.2 Missing Required Fields



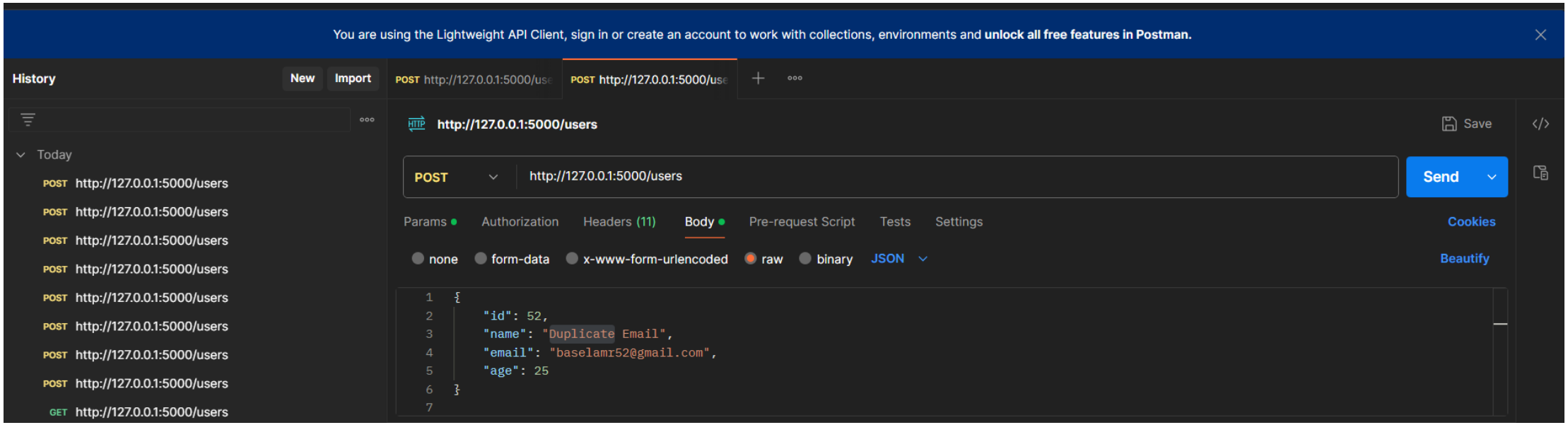


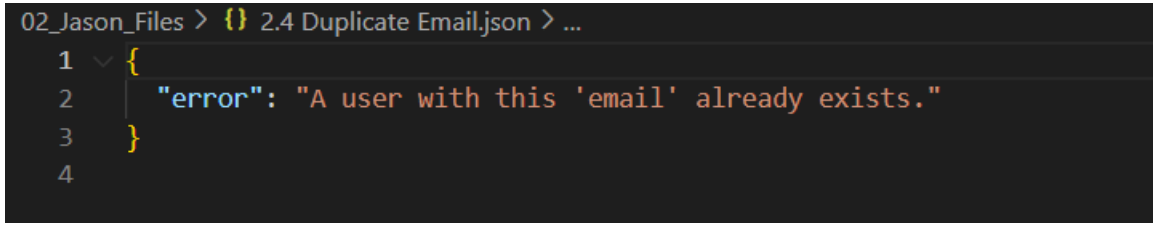
2.3 Duplicate ID





2.4 Duplicate Email





2.5 Invalid Email Format

History New Import

Today

POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
GET http://127.0.0.1:5000/users
GET http://127.0.0.1:5000/users

POST http://127.0.0.1:5000/use POST http://127.0.0.1:5000/use

http://127.0.0.1:5000/users Save

POST http://127.0.0.1:5000/users Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```
1 {
2     "id": 6,
3     "name": "Invalid Email",
4     "email": "invalidemail.com",
5     "age": 25
6 }
7
```

Body Cookies Headers (5) Test Results Status: 400 BAD REQUEST Time: -- Size: 232 B Save Response

02_Jason_Files > {} 2.4 Duplicate Email.json > ...

```
1 {
2   "error": "A user with this 'email' already exists."
3 }
4
```

2.6 Invalid Name Format

History New Import

Today

POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
POST http://127.0.0.1:5000/users
GET http://127.0.0.1:5000/users
GET http://127.0.0.1:5000/users

POST http://127.0.0.1:5000/use POST http://127.0.0.1:5000/use

http://127.0.0.1:5000/users Save

POST http://127.0.0.1:5000/users Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```
1 {
2     "id": 7,
3     "name": "Invalid@Name",
4     "email": "valid.email@example.com",
5     "age": 25
6 }
7
```

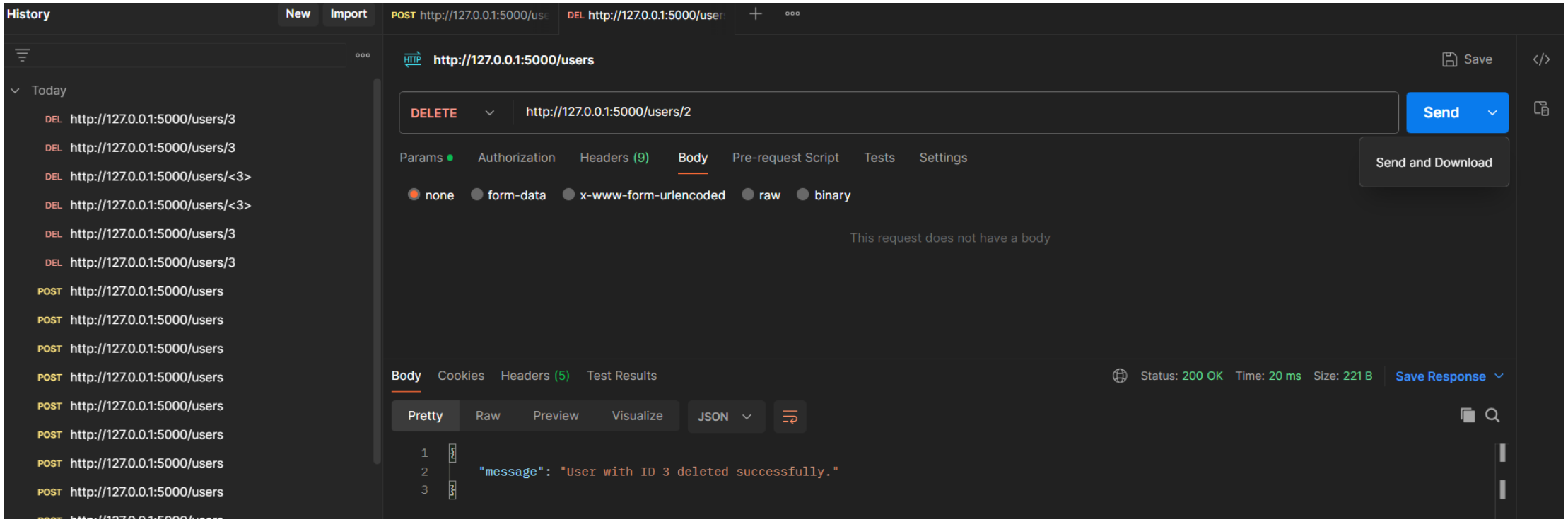
Body Cookies Headers (5) Test Results Status: 400 BAD REQUEST Time: -- Size: 213 B Save Response

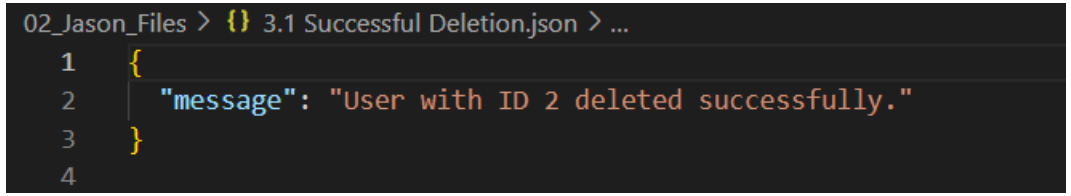
02_Jason_Files > {} 2.6 Invalid Name Format.json > ...

```
1 {
2   "error": "The 'name' field must only contain alphabetic characters and spaces."
3 }
4
```

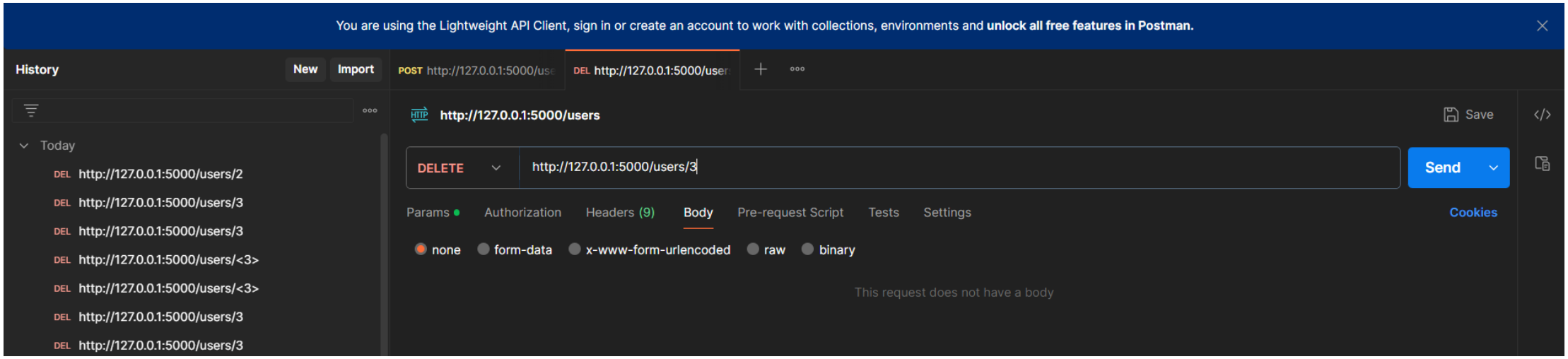
Testing DELETE Method

3.1 Successful Deletion





3.2 User Not Found

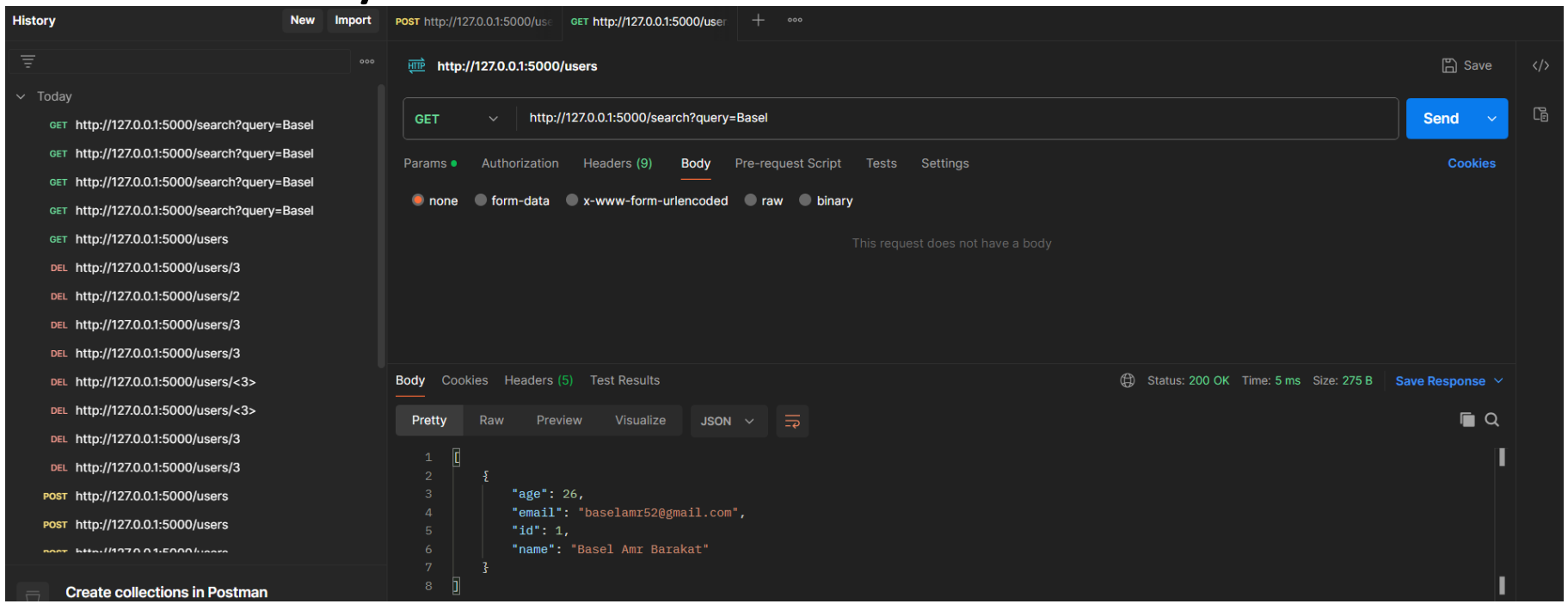


02_Jason_Files > {} 3.2 User Not Found.json > ...

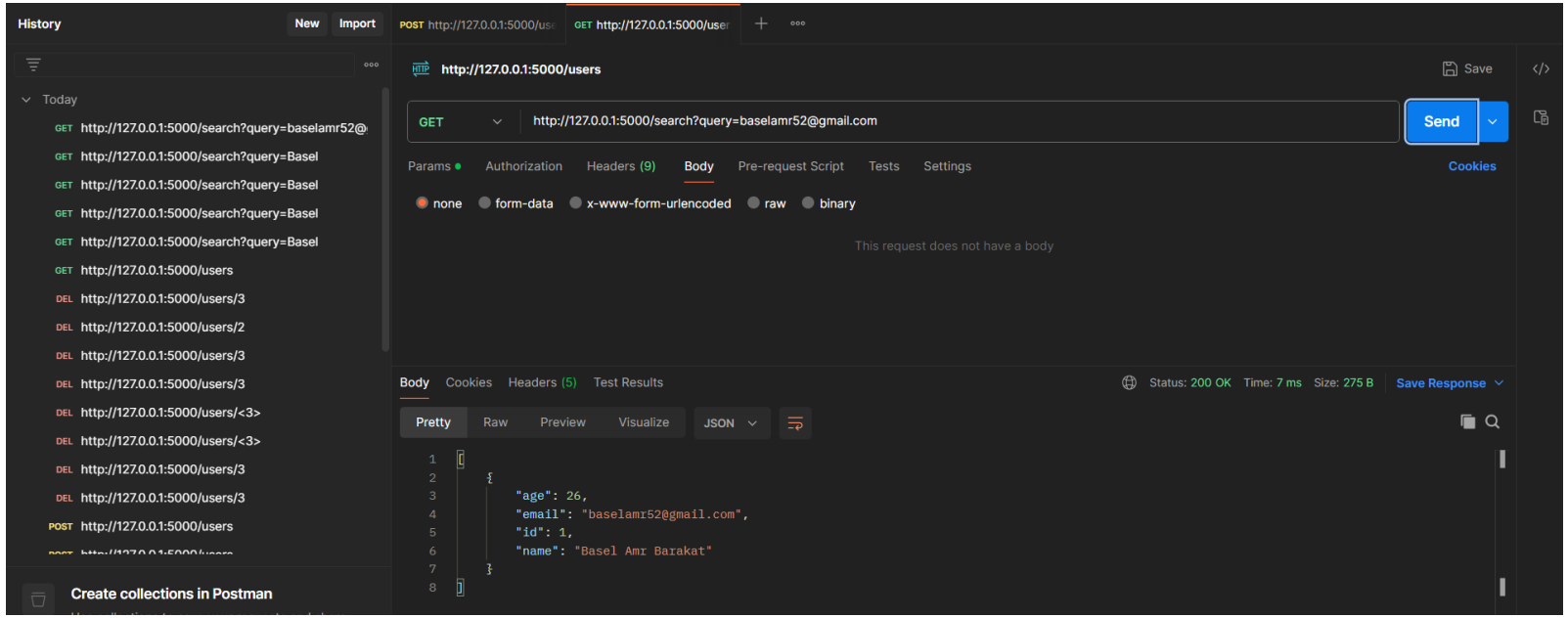
```
1  {
2    "error": "User with ID 3 not found."
3  }
4
```

Testing SEARCH Method

4.1 Search by Name



4.2 Search by Email



4.3 No Results Found

History | New | Import

POST http://127.0.0.1:5000/use | GET http://127.0.0.1:5000/user

Today

- GET http://127.0.0.1:5000/search?query=AyaAmr
- GET http://127.0.0.1:5000/search?query=baselamr52@
- GET http://127.0.0.1:5000/search?query=baselamr52@
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/users
- DEL http://127.0.0.1:5000/users/3
- DEL http://127.0.0.1:5000/users/2
- DEL http://127.0.0.1:5000/users/3
- DEL http://127.0.0.1:5000/users/3
- DEL http://127.0.0.1:5000/users/<3>
- DEL http://127.0.0.1:5000/users/<3>
- DEL http://127.0.0.1:5000/users/3

http://127.0.0.1:5000/users | Save

GET | http://127.0.0.1:5000/search?query=AyaAmr | Send

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary

This request does not have a body

Body | Cookies | Headers (5) | Test Results | Status: 404 NOT FOUND | Time: 7 ms | Size: 233 B | Save Response

Pretty | Raw | Preview | Visualize | JSON

```
1 {
2     "error": "No users found matching the search criteria"
3 }
```

02_Jason_Files > {} 4.3 No Results Found.json > ...

```
1 {
2     "error": "No users found matching the search criteria"
3 }
4
```

4.4 Missing Query Parameter

History | New | Import

POST http://127.0.0.1:5000/use | GET http://127.0.0.1:5000/user

Today

- GET http://127.0.0.1:5000/search
- GET http://127.0.0.1:5000/search?query=AyaAmr
- GET http://127.0.0.1:5000/search?query=AyaAmr
- GET http://127.0.0.1:5000/search?query=baselamr52@
- GET http://127.0.0.1:5000/search?query=baselamr52@
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/users
- DEL http://127.0.0.1:5000/users/3
- DEL http://127.0.0.1:5000/users/2
- DEL http://127.0.0.1:5000/users/3
- DEL http://127.0.0.1:5000/users/3
- DEL http://127.0.0.1:5000/users/<3>

http://127.0.0.1:5000/users | Save

GET | http://127.0.0.1:5000/search | Send

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary

This request does not have a body

Body | Cookies | Headers (5) | Test Results | Status: 400 BAD REQUEST | Time: 5 ms | Size: 219 B | Save Response

Pretty | Raw | Preview | Visualize | JSON

```
1 {
2     "error": "Query parameter is required"
3 }
```

02_Jason_Files > {} 4.4 Missing Query Parameter.json > ...

```
1 {
2     "error": "Query parameter is required"
3 }
4
```

4.4 Missing Query Parameter

History | New | Import

POST http://127.0.0.1:5000/use | GET http://127.0.0.1:5000/user

Today

- GET http://127.0.0.1:5000/search
- GET http://127.0.0.1:5000/search?query=AyaAmr
- GET http://127.0.0.1:5000/search?query=AyaAmr
- GET http://127.0.0.1:5000/search?query=baselamr52@
- GET http://127.0.0.1:5000/search?query=baselamr52@
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/search?query=Basel
- GET http://127.0.0.1:5000/users
- DEL http://127.0.0.1:5000/users/3
- DEL http://127.0.0.1:5000/users/2
- DEL http://127.0.0.1:5000/users/3
- DEL http://127.0.0.1:5000/users/3
- DEL http://127.0.0.1:5000/users/<3>

http://127.0.0.1:5000/users | Save

GET | http://127.0.0.1:5000/search | Send

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary

This request does not have a body

Body | Cookies | Headers (5) | Test Results | Status: 400 BAD REQUEST | Time: 5 ms | Size: 219 B | Save Response

Pretty | Raw | Preview | Visualize | JSON

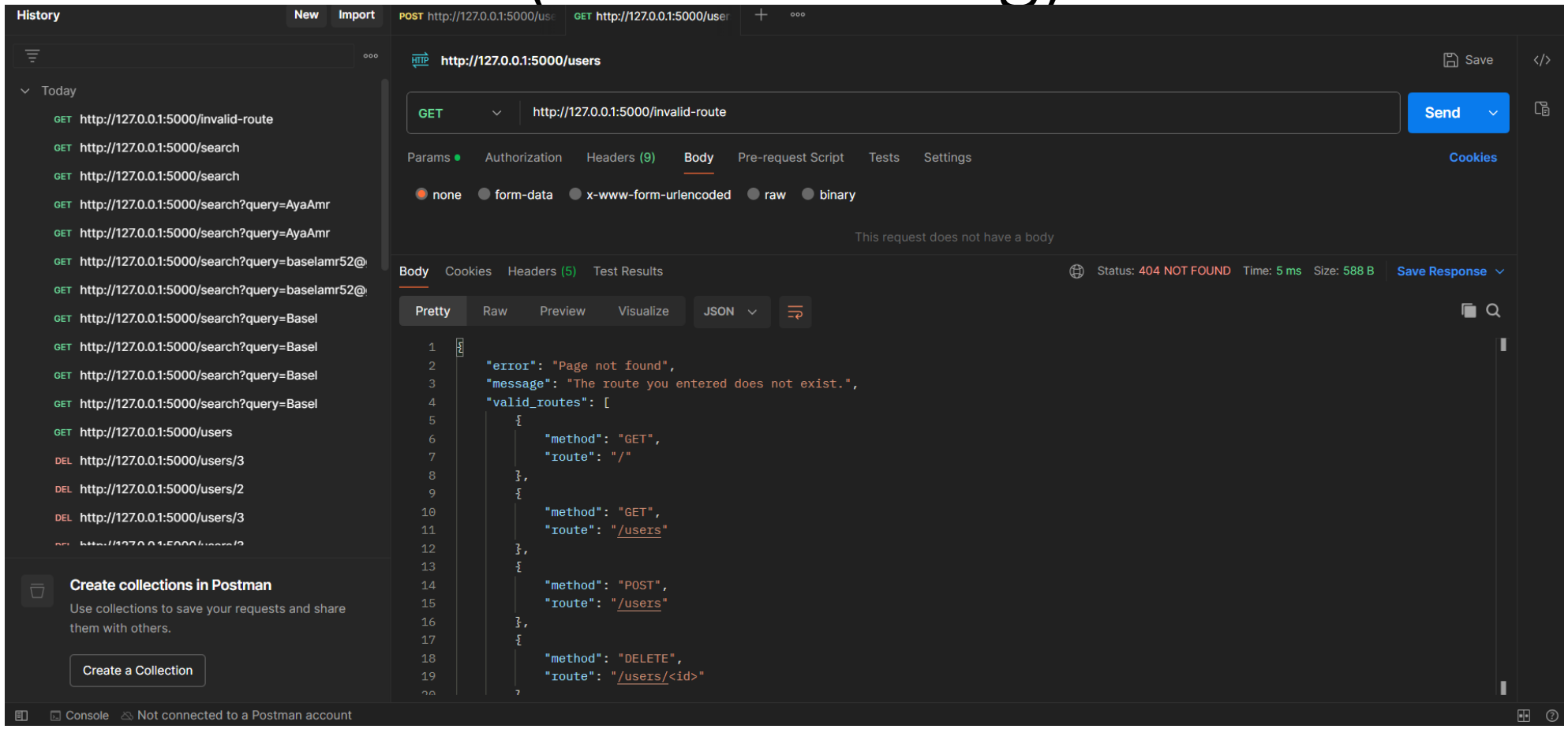
```
1 {
2     "error": "Query parameter is required"
3 }
```

02_Jason_Files > {} 4.4 Missing Query Parameter.json > ...

```
1 {
2     "error": "Query parameter is required"
3 }
4
```

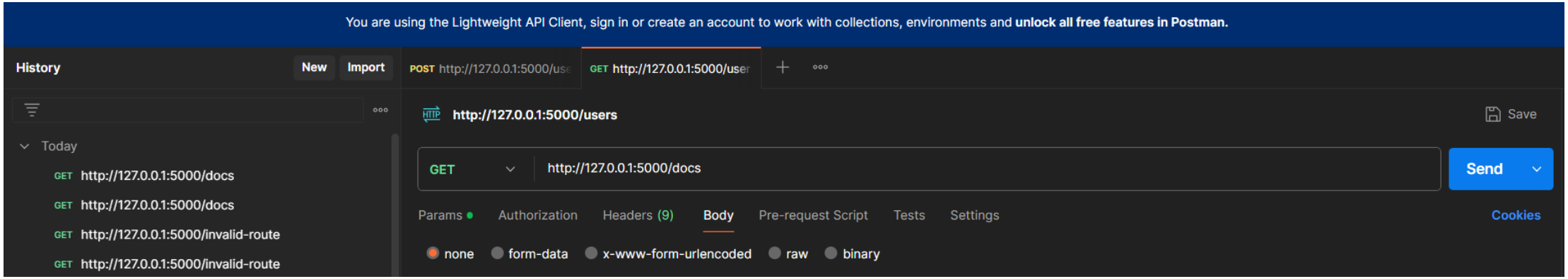
Invalid Route

5. Invalid Route (404 Handling)

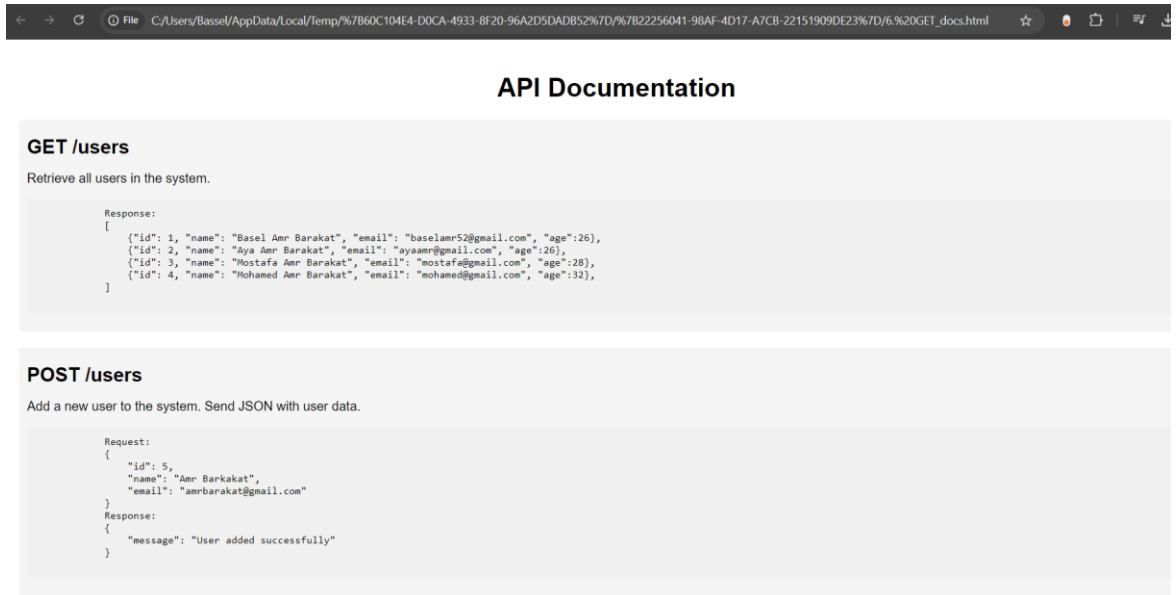


API Documentation

6. GET_docs



6. GET_docs.html



Setting up Flask with Apache using WGSI

1. Prerequisites for WSGI Deployment

- Install Apache for windows
- Install `mod_wsgi` for Apache
- Verify installation by checking mod_wsgi version

```
C:\Windows\System32>mod_wsgi-express module-config
LoadFile "C:/Users/Bassel/AppData/Local/Programs/Python/Python313/python313.dll"
LoadModule wsgi_module "C:/Users/Bassel/AppData/Local/Programs/Python/Python313/Lib/site-packages/mod_wsgi/server/mod_ws
gi.cp313-win_amd64.pyd"
WSGIPythonHome "C:/Users/Bassel/AppData/Local/Programs/Python/Python313"
```

2. Open the **http.conf** file located in the **conf** directory of my Apache installation and add the following lines

```
#1. Add Server Name
ServerName localhost:80
```

```
#2. Load configuration details
LoadFile "C:/Users/Bassel/AppData/Local/Programs/Python/Python313/python313.dll"
LoadModule wsgi_module "C:/Users/Bassel/AppData/Local/Programs/Python/Python313/Lib/site-packages/mod_wsgi/server/mod_wsgi.cp313-win_amd64.pyd"
WSGIPythonHome "C:/Users/Bassel/AppData/Local/Programs/Python/Python313"
```

3. Create a WSGI File

- 1. Create a file named flaskapp.wsgi in the same directory as our api.py file
- 2. Add the following content

```
import sys
import os
from 27_BaselAmr_Task3_API import app as application

# Add the project directory to the Python path
sys.path.insert(0, os.path.dirname(__file__))
```

4. Setting Up Apache Virtual Host

- 1. Open the httpd-vhost-conf file located in the conf/extra directory of our Apache installation
- 2. Add the following content

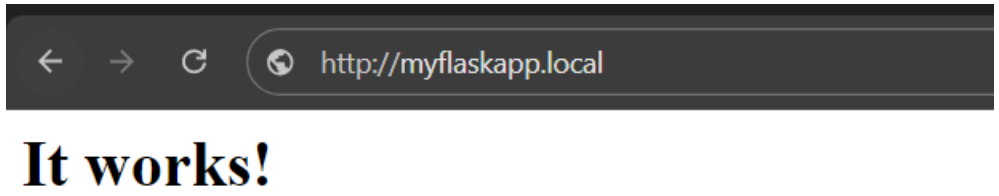
```
<VirtualHost *:80>
    ServerName firstflaskapp.localhost
    DocumentRoot "D:/Education/Courses/Sprints/AL_and_ML_Sprints_BootCamp/04-Tasks/05_FromDatatoAI/27_BaselAmr_Task3_API"
    WSGIScriptAlias / "D:/Education/Courses/Sprints/AL_and_ML_Sprints_BootCamp/04-Tasks/05_FromDatatoAI/27_BaselAmr_Task3_API/api.wsgi"

    <Directory "D:/Education/Courses/Sprints/AL_and_ML_Sprints_BootCamp/04-Tasks/05_FromDatatoAI/27_BaselAmr_Task3_API">
        Require all granted
    </Directory>

    ErrorLog "logs/flaskapp-error.log"
    CustomLog "logs/flaskapp-access.log" common
</VirtualHost>
```

5. Updating the host file

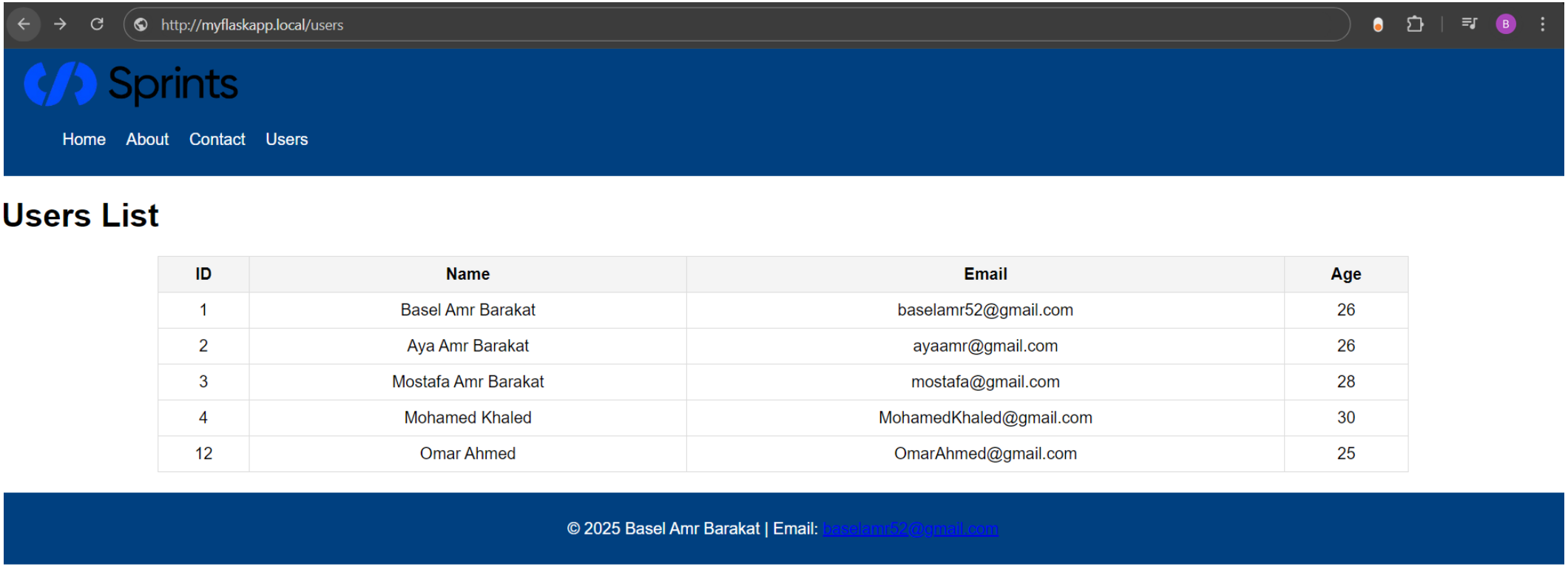
- 1. Open the hosts file with administrative priveleges
- 2. Add a line to map our hostname to myflask.local

```
# localhost name resolution is handled within DNS itself.
127.0.0.1 myflaskapp.local
```

6. Test the hostname by opening the browser and navigation to http://myflaskapp.local



7. Testing Users



ID	Name	Email	Age
1	Basel Amr Barakat	baselamr52@gmail.com	26
2	Aya Amr Barakat	ayaamr@gmail.com	26
3	Mostafa Amr Barakat	mostafa@gmail.com	28
4	Mohamed Khaled	MohamedKhaled@gmail.com	30
12	Omar Ahmed	OmarAhmed@gmail.com	25