URL Compressor Application

https://github.com/user-attachments/assets/77b6ba55-ed5f-47a1-859f-fa6752f08761

This project consists of both a **UrlCompressorApp** and a **UrlCompressorApi** that work together to provide a full-featured URL shortening service. The **UrlCompressorApi** is built with **Python** and **FastAPI**, while the **UrlCompressorApp** is built with **React**.

Overview

The **UrlCompressorApi API** handles the core functionality of the URL shortening service. It receives long URLs from the UrlCompressorApp, generates unique short codes using Base62 encoding, stores the mappings in a database (SQLite), and redirects users to the original URL when they access the compressed URL.

The **UrlCompressorApp application** provides a user-friendly interface where users can input long URLs and receive compressed URLs. It communicates with the UrlCompressorApi API to send the long URLs and display the generated short URLs to the user.

Getting Started with the URL Compressor Project in Visual Studio

This guide provides step-by-step instructions for cloning the URL Compressor project from Git and building both the backend and UrlCompressorApp using **Visual Studio** on Windows. You will set up the **Python FastAPI** backend and the **React** UrlCompressorApp within Visual Studio, leveraging its integrated development environment features.

Table of Contents

  - 2. Install Node.js Dependencies
  - 3. Run the UrlCompressorApp Application
- Testing the Application
- Additional Tips
- Troubleshooting

Prerequisites

1. **Visual Studio 2019 or Visual Studio 2022** (Community, Professional, or Enterprise edition).
   - **Workloads to Install**:
     - **Python development** workload.
     - **Node.js development** workload.
   - **Installation Steps**:
     - Open the Visual Studio Installer.
     - Modify your existing installation.
     - Check the **Python development** and **Node.js development** workloads.
     - Click **Modify** to install the selected workloads.
2. **Git** installed on your system.
   - Download Git
3. **Node.js** (version 12 or higher).
   - Download Node.js
4. **Python 3.7 or higher**.
   - Download Python

Cloning the Repository

First, you need to clone the project repository from GitHub.

Steps:

1. **Open Command Prompt or Git Bash**.
2. **Navigate to the Directory Where You Want to Clone the Project**:

   ```
   cd path\to\your\projects\directory
   ```

3. **Clone the Repository**:

```
git clone https://github.com/ShaneKirkbride/UrlCompressor.git
```

4. **Navigate to the Project Directory**:

   ```
   cd UrlCompressor
   ```

   The repository should contain two main directories:

   - `UrlCompressorApi` (Python FastAPI project)
   - `UrlCompressorApp` (React project)

Setting Up the Backend (Python FastAPI)

1. Open the Backend Folder in Visual Studio

1. **Launch Visual Studio**.
2. **Open the Project**:
   - Click on **Open a local folder** on the start window.
   - Navigate to the cloned repository, then select the `UrlCompressorApi` folder.
   - Click **Select Folder**.

   Visual Studio will load the folder and recognize it as a Python project.

2. Create a Virtual Environment

Creating a virtual environment isolates the project dependencies.

1. **Open the Command Prompt in Visual Studio**:
   - Go to **View** > **Terminal** or press `Ctrl +` (backtick).
2. **Create the Virtual Environment**:

   ```
   python -m venv venv
   ```

   This creates a virtual environment named `venv` in the project directory.

3. Install Dependencies

3. **Activate the Virtual Environment**:
   - On Windows:

     ```
     venv\Scripts\activate
     ```

   - The command prompt should now start with `(venv)`.
4. **Install Required Packages**:

   If your project includes a `requirements.txt` file, install the dependencies:

```
pip install -r requirements.txt
```

If you don't have a `requirements.txt`, you can install packages individually:

```
pip install fastapi uvicorn sqlalchemy pydantic alembic
```

5. **Update `requirements.txt`** (Optional):

```
pip freeze > requirements.txt
```

4. Configure Python Environment in Visual Studio

1. **Set the Python Interpreter**:
   - Go to **View** > **Other Windows** > **Python Environments**.
   - In the **Python Environments** window, click on **Add Environment** (if `venv` is not automatically detected).
   - Select **Existing environment**.
   - Set the **Prefix path** to the `venv` folder in your project.
   - Name the environment (e.g., `venv`).
   - Click **Add**.
2. **Select the Virtual Environment**:
   - Ensure that the virtual environment you just added is selected as the default for the project.

5. Run the Backend Application

1. **Create a Debug Configuration**:
   - Go to **Debug** > **Add Configuration**.
   - In the `launch.json` file (if prompted), set up the following configuration:

```
{
  "name": "Python: FastAPI",
  "type": "python",
  "request": "launch",
  "module": "uvicorn",
  "args": [
    "UrlCompressorApi:app",
    "--host",
    "127.0.0.1",
    "--port",
    "8000",
    "--reload"
  ],
```

```
    "jinja": true
}
```

   - Replace `UrlCompressorApi:app` with the correct module and app name if different.

2. **Start Debugging**:
   - Press `F5` or go to **Debug** > **Start Debugging**.
   - The terminal should display that Uvicorn is running the server.
3. **Verify the Backend is Running**:
   - Open a web browser and navigate to http://127.0.0.1:8000/docs.
   - You should see the interactive API documentation provided by Swagger UI.

Setting Up the UrlCompressorApp (React)

1. Open the UrlCompressorApp Folder in Visual Studio

1. **Open the Project**:
   - In the same soluiton click *Add* then *Existing Item*
   - Click on **Open a local folder**.
   - Navigate to the cloned repository, then select the `UrlCompressorApp` folder.
   - Click **Select Folder**.

   Visual Studio will recognize it as a Node.js project.

2. Install Node.js Dependencies

1. **Open the Terminal in Visual Studio**:
   - Go to **View** > **Terminal**.
2. **Install Dependencies**:

   ```
   npm install
   ```

   This will install all the packages listed in `package.json`.

3. Run the UrlCompressorApp Application

1. **Start the Development Server**:

   ```
   npm start
   ```

2. **Verify the UrlCompressorApp is Running**:
   - Open a web browser and navigate to http://localhost:3000.

- You should see the URL Compressor UrlCompressorApp interface.

3. **Debugging in Visual Studio** (Optional):
   - Visual Studio can debug JavaScript applications.
   - Set breakpoints in your `.jsx` or `.js` files.
   - Use **Debug** > **Start Debugging** to launch the debugger.

Testing the Application

With both the backend and UrlCompressorApp running, you can now test the full application.

1. **Access the UrlCompressorApp**:
   - Navigate to http://localhost:3000.
2. **Shorten a URL**:
   - Enter a valid URL (e.g., `https://www.example.com`) into the input field.
   - Click **"Shorten URL"**.
3. **View the Shortened URL**:
   - The shortened URL should be displayed.
   - Click **"Copy"** to copy it to your clipboard.
4. **Test Redirection**:
   - Paste the shortened URL into your browser.
   - You should be redirected to the original URL.

Troubleshooting

Backend Issues

- **Module Not Found Errors**:
  - Ensure all Python dependencies are installed in your virtual environment.
  - Activate the virtual environment before running the backend.
- **Port Already in Use**:
  - If port `8000` is in use, you can change the port in the Uvicorn arguments:

```
"args": [
  "UrlCompressorApi:app",
```

```
        "--host",
        "127.0.0.1",
        "--port",
        "8001",
        "--reload"
    ],
```

- **Database Errors**:
  - Ensure the database tables are created by running `Base.metadata.create_all(bind=engine)` in your `main.py` or `database.py`.

UrlCompressorApp Issues

- **Dependencies Not Found**:
  - Run `npm install` to install all Node.js dependencies.
- **API Endpoint Errors**:
  - Verify that the UrlCompressorApp is pointing to the correct backend API URL.
  - Check for any CORS errors in the browser console.
- **Port Conflicts**:
  - If port `3000` is in use, you can start the UrlCompressorApp on a different port:

    ```
    set PORT=3001 && npm start
    ```

General Issues

- **Firewall or Antivirus Interference**:
  - Ensure that your firewall or antivirus software is not blocking the ports used by the applications.
- **Version Compatibility**:
  - Ensure that the versions of Python, Node.js, and dependencies are compatible.
- **Visual Studio Extensions**:
  - Install helpful extensions like **Python**, **ESLint**, and **Prettier** for better development experience.

Conclusion

By following these steps, you should be able to clone the URL Compressor project from Git and set it up in Visual Studio, running both the backend and UrlCompressorApp applications. Visual Studio provides a powerful environment for developing, debugging, and running both Python and JavaScript projects.

If you have any questions or encounter issues, feel free to reach out for assistance or consult the documentation for Visual Studio, FastAPI, and React.

Additional Resources

- **Visual Studio Documentation**:
  - Python in Visual Studio
  - Node.js in Visual Studio
- **FastAPI Documentation**:
  - FastAPI
- **React Documentation**:
  - React
- **Git Documentation**:
  - Git

How They Work Together

1. **User Interaction**:
   - The user visits the UrlCompressorApp application and enters a long URL (and optionally, an expiration time) into the input form.
2. **API Request**:
   - Upon form submission, the UrlCompressorApp sends a **POST** request to the UrlCompressorApi API's `/shorten` endpoint with the long URL and expiration time.
3. **URL Shortening**:
   - The UrlCompressorApi API processes the request:
     - Validates the input URL.
     - Stores the original URL in the database with a unique integer ID.
     - Generates a unique short code by encoding the ID using Base62 encoding.
     - Constructs the compressed URL using the short code.

  - Returns the compressed URL to the UrlCompressorApp.

4. **Displaying the Short URL**:
   - The UrlCompressorApp receives the response from the UrlCompressorApi and displays the compressed URL to the user.
   - The user can copy the compressed URL to the clipboard.
5. **Redirection**:
   - When someone accesses the compressed URL, the request is handled by the UrlCompressorApi API's `/{short_code}` endpoint.
   - The UrlCompressorApi decodes the short code to retrieve the corresponding original URL from the database.
   - If the URL has not expired, the UrlCompressorApi redirects the user to the original URL.
6. **Error Handling**:
   - Both UrlCompressorApp and UrlCompressorApi handle errors gracefully.
     - The UrlCompressorApi returns appropriate HTTP status codes and error messages for invalid requests or expired URLs.
     - The UrlCompressorApp displays error messages to inform the user of any issues.

Technology Stack Interaction

- **HTTP Communication**:
  - The UrlCompressorApp and UrlCompressorApi communicate over HTTP using RESTful API endpoints.
  - CORS (Cross-Origin Resource Sharing) is configured on the UrlCompressorApi to allow requests from the UrlCompressorApp domain.
- **Data Exchange**:
  - Data is exchanged in JSON format.
  - The UrlCompressorApp sends JSON payloads containing the original URL and expiration time.
  - The UrlCompressorApi responds with JSON containing the compressed URL.
- **State Management**:
  - The UrlCompressorApp uses React's state management to handle user inputs and API responses.
- **Security Considerations**:
  - Input validation is performed on both the UrlCompressorApp (basic checks) and UrlCompressorApi (robust validation) to ensure URLs

are valid.
  - The UrlCompressorApi avoids security vulnerabilities by properly handling user inputs and errors.

URL Compressor UrlCompressorApp Deep Dive

This is the UrlCompressorApp application for a URL Compressor built with **React**. The application provides a user interface for entering long URLs and receiving compressed URLs from the UrlCompressorApi API.

Table of Contents

Features

- **URL Input Form**: Allows users to enter a long URL and an optional expiration time.
- **Short URL Display**: Shows the compressed URL generated by the UrlCompressorApi API.
- **Copy to Clipboard**: Users can copy the compressed URL with a single click.
- **Responsive Design**: The UI is responsive and works on various screen sizes.
- **Error Handling**: Displays error messages for invalid inputs or server errors.

Technology Stack

- **React** - A JavaScript library for building user interfaces.
- **Axios** - Promise-based HTTP client for making API requests.

- **JavaScript (ES6+)** - Modern JavaScript features.
- **CSS** - Styling components.
- **Create React App** - A comfortable environment for learning React and building a single-page application.

Prerequisites

- **Node.js** (version 12 or higher)
- **npm** or **yarn** package manager

Running the Application

1. Start the Development Server

Using **npm**:

```
npm start
```

Or using **yarn**:

```
yarn start
```

- The application will run in development mode.
- Open http://localhost:3000 to view it in the browser.
- The page will reload if you make edits.

2. Build for Production

Using **npm**:

```
npm run build
```

Or using **yarn**:

```
yarn build
```

- Builds the app for production to the `build` folder.
- It correctly bundles React in production mode and optimizes the build for the best performance.

Project Structure

```
url-Compressor-UrlCompressorApp/
  public/
      index.html
      favicon.ico
```

```
src/
    components/
        ShortenURLForm.jsx
        URLResult.jsx
    App.js
    App.css
    index.js
    index.css
package.json
README.md
.gitignore
```

Key Files and Directories

- **`public/`**: Contains the HTML file so you can tweak it, for example, to set the page title.
- **`src/`**: Contains the JavaScript code for your application.
  - **`components/`**: Contains the React components used in the application.
    - **`ShortenURLForm.jsx`**: Component for the URL input form.
    - **`URLResult.jsx`**: Component for displaying the compressed URL.
  - **`App.js`**: The root component that ties everything together.
  - **`App.css`**: Styling for the `App` component.
  - **`index.js`**: Entry point of the application.
- **`package.json`**: Contains project metadata and dependencies.

Available Scripts

In the project directory, you can run:

`npm start`

Runs the app in development mode. Open http://localhost:3000 to view it in the browser.

`npm run build`

Builds the app for production to the `build` folder.

`npm test`

Launches the test runner in the interactive watch mode.

`npm run eject`

Note: this is a one-way operation. Once you `eject`, you can't go back!

Environment Variables

If your UrlCompressorApi API is hosted on a different URL or port, you can configure the API endpoint by creating a `.env` file in the root of your project:

```
REACT_APP_API_URL=http://localhost:8000
```

In your components, you can access this variable using:

```
const API_URL = process.env.REACT_APP_API_URL || 'http://localhost:8000';
```

Contributing

Contributions are welcome! If you'd like to contribute, please follow these steps:

1. **Fork the repository.**
2. **Create a new branch.**

   ```
   git checkout -b feature/your-feature-name
   ```

3. **Make your changes and commit them.**

   ```
   git commit -m "Add some feature"
   ```

4. **Push to the branch.**

   ```
   git push origin feature/your-feature-name
   ```

5. **Open a pull request.**

Additional Information

Dependencies

- **React**: A JavaScript library for building user interfaces.
- **Axios**: Promise-based HTTP client for the browser and Node.js.

Installing Dependencies

If you need to install dependencies manually, you can run:

```
npm install react react-dom axios
```

Styling

You can customize the styles by modifying the CSS files or using a CSS-in-JS solution. Consider using libraries like **Material-UI** or **Bootstrap** for enhanced UI components.

Troubleshooting

- **CORS Issues**: Ensure that the UrlCompressorApi API has proper CORS configurations to allow requests from the UrlCompressorApp.
- **API Endpoint Configuration**: If the UrlCompressorApp cannot communicate with the UrlCompressorApi, verify that the API URL is correct and that the UrlCompressorApi server is running.
- **Dependency Errors**: If you encounter errors related to missing modules, run `npm install` to install all dependencies.

Future Enhancements

- **Form Validation**: Improve input validation to provide instant feedback to the user.
- **Error Messages**: Display more informative error messages from the UrlCompressorApi.
- **History of compressed URLs**: Allow users to see a list of their recently compressed URLs.
- **QR Code Generation**: Generate a QR code for the compressed URL.

Example Usage

1. Enter a Long URL

- Open the application in your browser.
- In the input field, enter the long URL you wish to shorten.
- Optionally, enter an expiration time in seconds.

2. Submit the Form

- Click the **"Shorten URL"** button.
- The application sends a request to the UrlCompressorApi API to create a compressed URL.

3. View and Use the compressed URL

- The compressed URL is displayed below the form.
- Click the **"Copy"** button to copy it to your clipboard.
- Use the compressed URL in your browser or share it with others.

Project Structure Details

`src/App.js`

The main application component that renders the header, form, and result components.

`src/components/ShortenURLForm.jsx`

Handles user input for the long URL and expiration time, and sends a POST request to the UrlCompressorApi API.

`src/components/URLResult.jsx`

Displays the compressed URL and provides a button to copy it to the clipboard.

URL Compressor UrlCompressorApi API Deep Dive

This is the UrlCompressorApi API for a URL Compressing application built with Python and FastAPI. The API allows users to shorten long URLs and redirect to the original URL when the compressed URL is accessed.

Table of Contents

- Contributing
- License

Features

- Shorten long URLs to compact, shareable links.
- Set custom expiration times for each compressed URL.
- Automatic redirection to the original URL when the short URL is accessed.
- Efficient Base62 encoding for generating short codes.
- SQLite database for storing URL mappings.

Technology Stack

- **Python 3.9**
- **FastAPI** - A modern, fast (high-performance) web framework for building APIs.
- **Uvicorn** - An ASGI web server implementation for Python.
- **SQLAlchemy** - An SQL toolkit and Object-Relational Mapping (ORM) library.
- **Pydantic** - Data validation and settings management using Python type annotations.
- **SQLite** - A lightweight disk-based database.

Prerequisites

- **Python 3.7 or higher** installed on your system.
- **pip** package manager.
- **Virtual Environment** (optional but recommended).

Installation

1. Clone the Repository

```
git clone https://github.com/yourusername/url-Compressor-UrlCompressorApi.git
cd url-Compressor-UrlCompressorApi
```

2. Create a Virtual Environment

```
python -m venv venv
```

- **Activate the Virtual Environment**

- On macOS/Linux:

  ```
  source venv/bin/activate
  ```

- On Windows:

  ```
  venv\Scripts\activate
  ```

3. Install Dependencies

```
pip install -r requirements.txt
```

Database Setup

The application uses **SQLite** as the database. The database file (`urlCompressor.db`) will be automatically created in the project directory when you run the application for the first time.

Running the Application

1. Start the FastAPI Application

```
uvicorn UrlCompressorApi:app --reload
```

- Replace `main` with the name of your Python file if different.
- The `--reload` flag enables auto-reloading when code changes.

2. Access the API Documentation

- Open your browser and navigate to `http://localhost:8000/docs` to view the interactive API documentation provided by **Swagger UI**.
- Alternatively, access the ReDoc documentation at `http://localhost:8000/redoc`.

API Endpoints

Base URL

```
http://localhost:8000
```

1. Create Short URL

- **Endpoint**: `/shorten`
- **Method**: `POST`
- **Description**: Creates a compressed URL for the provided original URL.

Request Body

```
{
  "original_url": "https://www.example.com",
  "expiration_time": 3600
}
```

- `original_url` (string, required): The long URL to be compressed.
- `expiration_time` (integer, optional): Time in seconds after which the short URL will expire. If not provided, the URL will not expire.

Response

- **Status Code**: `200 OK`

```
{
  "short_url": "http://localhost:8000/abc123"
}
```

- `short_url` (string): The generated compressed URL.

Possible Errors

- **Status Code**: `422 Unprocessable Entity` - If the input data is invalid.

2. Redirect to Original URL

- **Endpoint**: `/{short_code}`
- **Method**: `GET`
- **Description**: Redirects to the original URL associated with the given short code.

URL Parameters

- `short_code` (string, required): The code at the end of the compressed URL.

Behavior

- **Successful Redirect**: Redirects to the original URL with a `307 Temporary Redirect` status.
- **Errors**:
  - **Status Code**: `404 Not Found` - If the short code is invalid or does not exist.
  - **Status Code**: `410 Gone` - If the short URL has expired.

Testing the API

You can test the API using tools like **cURL**, **Postman**, or the **Swagger UI** interface.

Using cURL

1. Create a Short URL

```
curl -X POST "http://localhost:8000/shorten" \
-H "Content-Type: application/json" \
-d '{"original_url": "https://www.example.com", "expiration_time": 3600}'
```

2. Response

```
{
  "short_url": "http://localhost:8000/abc123"
}
```

3. Access the Short URL

- Open the `short_url` in your web browser to be redirected to `https://www.example.com`.

Project Structure

```
UrlCompressorApi/
   UrlCompressorApi.py  # Main application file (FastAPI app)
   models.py            # Database models (SQLAlchemy ORM)
   database.py          # Database connection and session management
   schemas.py           # Pydantic models for request and response validation
   services.py          # URLCompressorService class for encoding and decoding
   requirements.txt     # Python dependencies
   urlCompressor.db      # SQLite database file (created automatically)
```

Dependencies

- **FastAPI**: Web framework for building APIs.
- **Uvicorn**: ASGI server for running the application.
- **SQLAlchemy**: ORM for database interactions.
- **Pydantic**: Data validation library.
- **aiofiles**: For asynchronous file operations.
- **databases**: Async database support (if used).

Troubleshooting

- **CORS Issues**: Ensure that CORS is properly configured in `main.py` to allow requests from your UrlCompressorApp application.

```
from fastapi.middleware.cors import CORSMiddleware

  app.add_middleware(
      CORSMiddleware,
      allow_origins=["http://localhost:3000"],  # Specify your UrlCompressorApp origin
      allow_credentials=True,
      allow_methods=["POST", "OPTIONS"],  # Explicitly include POST and OPTIONS methods
      allow_headers=["*"],
  )
```

- **Database Errors**: If you encounter database errors, make sure that the database tables are created by running `Base.metadata.create_all(bind=engine)` and that your models are correctly defined.
- **Dependencies Not Found**: Ensure all dependencies are installed by running `pip install -r requirements.txt`.

Future Enhancements

- **User Authentication**: Implement user accounts to track URLs per user.
- **Analytics**: Track the number of clicks for each compressed URL.
- **Custom Aliases**: Allow users to specify custom short codes.
- **API Rate Limiting**: Prevent abuse by limiting the number of requests per user/IP.

Additional Information

Contributing

Contributions are welcome! If you find any issues or have suggestions for improvements, please open an issue or submit a pull request.

License

This project is licensed under the MIT License.

Contact

- **Author**: Shane Kirkbrde

- **Email**: shane.kirkbride@keysight.com
- **GitHub**: ShaneKirkbride

Acknowledgements

- UrlCompressorApi Acknowlegements
- **FastAPI Documentation**: https://fastapi.tiangolo.com/
- **SQLAlchemy Documentation**: https://www.sqlalchemy.org/
- **Pydantic Documentation**: https://pydantic-docs.helpmanual.io/
- **Simple React Example**: [https://github.com/aditya-sridhar/simple-reactjs-app/
- UrlCompressorApp Acknowlegements
- **React**: https://reactjs.org/
- **Axios**: https://axios-http.com/
- **Create React App**: https://create-react-app.dev/
