- Simple Search AI
  - Overview
  - Features
  - Project Structure
  - Prerequisites
  - Installation
    - Step 2: Set Up API Keys
    - Step 3: Set Up the Server
  - Usage
  - How It Works
    - Backend (PHP):
    - Frontend (HTML + JavaScript):
  - Filters
  - Error Handling
  - Troubleshooting
  - Contributing

Simple Search AI

Overview

This project is a simple search application that retrieves product information from multiple platforms (eBay, Google, SERP, and Amazon) based on a user's search query. The application compares search results from these platforms by price and other features. It also provides filter options to refine the search results.

This project is built using PHP on the server side, with a front-end UI powered by Bootstrap.

Features

- **Search Product** : Search for a product across four platforms: eBay, Google, SERP, and Amazon.

- **Comparison of Results** : The results from all platforms are displayed, allowing users to compare prices and other details.
- **Filters** : The application includes options to filter results based on specific criteria.
- **Dynamic Search** : The results update dynamically based on the entered search query.
- **Responsive Layout** : The UI is built using Bootstrap to be fully responsive across devices.

Project Structure

```
/simple-search-ai ├── .env # Configuration file for API keys ├── api/ │ ├── ebay.php # eBay API handler │ ├── google.php # Google API handler │ ├── serp.php # SERP API handler │ └── amazon.php # Amazon API handler ├── assets/ │ ├── css/ │ ├── js/ │ └── images/ ├── index.php # Main landing page ├── .gitignore # Ignore unnecessary files from Git └── README.md # Project documentation
```

Prerequisites

- **PHP 7.0+** installed on your server
- **cURL** enabled in PHP for API requests
- Valid API keys for the platforms (eBay, Google, SERP, and Amazon)
- Basic knowledge of running a PHP server

Installation

Step 2: Set Up API Keys

Create a `.env` file in the root directory of the project (same directory as `index.php`). You need to add the API keys for eBay, Google, SERP, and Amazon in this file.

Example `.env` file:

```
EBAY_API_KEY=your_ebay_api_key GOOGLE_API_KEY=your_google_api_key
SERP_API_KEY=your_serp_api_key AMAZON_API_KEY=your_amazon_api_key
```

Make sure you replace `your_ebay_api_key`, `your_google_api_key`, `your_serp_api_key`, and `your_amazon_api_key` with actual API keys.

Step 3: Set Up the Server

Make sure your server is running PHP. If you're using Apache, place the project folder in your web server's root directory (`/var/www/html` for Ubuntu, for example).

Alternatively, you can use PHP's built-in server to test locally:

```
php -S localhost:8000
```

This will start a local server at `http://localhost:8000`.

Usage

1. **Navigate to `index.php`** : Open your browser and navigate to the root URL of the project (e.g., `http://localhost:8000/index.php` if running locally).
2. **Enter a Search Query** : Type the name of a product into the search input field.
3. **Click "Search"** : Click the "Search" button to initiate the search.
4. **View Results** : The results will be displayed from eBay, Google, SERP, and Amazon, showing product details such as title, price, and a link to view the product on the respective platform.

How It Works

The search functionality works by making API calls to each of the platforms (eBay, Google, SERP, and Amazon) and retrieving product data such as price, title, and product link. The results are then dynamically displayed on the front-end.

Backend (PHP):

- Each platform has its own PHP file (e.g., `ebay.php`, `google.php`, etc.) that handles the respective API request and fetches results.
- The API keys are stored securely in the `.env` file and accessed by the backend code using PHP.
- The front-end calls the `performSearch()` function to initiate the search, and the backend sends the results back as JSON.

Frontend (HTML + JavaScript):

- The user enters a product name and submits the query.
- The `performSearch()` function makes an asynchronous request to the backend PHP files.
- The results are then displayed under the respective headings (eBay, Google, SERP, Amazon).

Filters

Filters can be implemented in the search UI for additional refinement (e.g., filter by price range, product category). To add new filters:

1. Add the HTML filter controls (e.g., dropdowns, checkboxes) in the `index.php` file.
2. Modify the backend API handling code to account for the filter parameters.
3. Pass the selected filter values to the respective API request functions.

Error Handling

The application includes basic error handling:

- If no API key is provided, or if an API request fails, the search results will display an error message indicating that no results could be retrieved.
- If the user doesn't enter a search query, the interface will display a prompt to enter a search term.

Troubleshooting

1. **Invalid API Key** : Ensure your API keys are correctly added to the `.env` file. If you get an error related to authentication, check the API documentation for the correct key format or any additional requirements.
2. **API Limits** : Many of these APIs have rate limits. If you exceed the number of allowed requests in a given period, you may need to wait before making additional requests or contact the API provider to increase the limit.
3. **cURL Errors** : Ensure that your PHP installation has cURL enabled, as it's required for making the API requests.
4. **Blank Results** : If the search query returns no results, ensure the API responses are being properly parsed. You can use `var_dump()` or `error_log()` for debugging.

Contributing

If you'd like to contribute improvements or fix bugs in the project, feel free to fork the repository, make your changes, and submit a pull request.