

# **Microservices for Daily Life**

CS4471 Software Design and Architecture

Professor Nazim H. Madhavji

Department of Computer Science

Western University

Group 3

Le (Leon) Zhu

Colin Brown

Xiaolin (Eric) Wang

Kaiyi Hou

Zoe Kaute

## Table of Contents

Table of Contents .....	<b>Error! Bookmark not defined.</b>
Introduction .....	3
Requirements .....	3
Registry Requirements .....	3
S-01 Requirements .....	3
S-02 Requirements .....	4
S-03 Requirements .....	5
S-04 Requirements .....	5
S-05 Requirements .....	6
Architecture and Design .....	7
Context Model .....	7
Overall System Architecture .....	8
Component Connector View .....	8
Implementation Details .....	9
Testing .....	10
Operational Evidence .....	10
Registry .....	10
S-01 .....	14
S-02 .....	18
S-03 .....	21
S-04 .....	25
S-05 .....	27
Appendix .....	30
References .....	31

# Introduction

This system comprises of a core Registry service which is designed to interface with a flexible number of external microservices.

## Requirements

### Registry Requirements

#### Functional Requirements

- **Display Microservices:** Display a real-time list of all microservices currently registered in the registry. The list must be dynamically updated to reflect the addition, removal, or unavailability of services.
- **Service Discovery:** Ability to search through all the registered microservices for service discovery.
- **Service Management:** Handles register, deregister, and heartbeat POST requests from the microservices and provides information back to them about the request.
- **Service Availability Monitoring:** Monitor the availability of registered microservices by tracking heartbeat signals. A microservice will be marked as unavailable if it fails to send two consecutive heartbeats.

#### Quality Requirements

- **Timely updates:** The registry must frequently refresh the list of registered microservices to ensure users and systems have access to accurate, up-to-date information. Updates must account for new registration, deregistration, and changes in service availability.
- **Search Performance:** The registry must process search queries quickly and reliably to enable efficient service discovery.

#### ASRs

- **Availability:** Should always be available so that you can always tell if the microservices are available or not. As a critical component of the system, the registry's downtime directly impacts the ability of microservices to function, making availability critical for the overall architecture.

## S-01 Requirements

#### Functional Requirements

- **Site Explanation and Input Method:** Clearly explain the purpose of the site and provide a straightforward method for users to enter a stock symbol and submit their request.
- **Market Share Calculation:** Calculate the market share of a stock by identifying its market and determining the percentage of the market cap it represents.

- **Additional Stock Information:** Return additional relevant information about the stock, such as changes in its market share over time, to provide a comprehensive overview.
- **Display Processed Information:** Display the calculated market share and additional stock data accurately for the user.
- **Registry Interaction:** Be able to register and deregister with the service registry and send periodic heartbeat signals to confirm the service's active status.

## Quality Requirements

- **Clear Interface:** Present the returned stock data in a clear, visually appealing, and informative manner for the user within a web browser.
- **Intuitive:** The purpose of the service is clear to the user when they first view the website and not confusing.
- **Fast Response:** After sending a request, the server takes no more than 2 seconds for the user to have the returned data.

## ASRs

- **Performance:** Aim to return the stock data within 2 seconds of receiving a request. Quick response times are critical for maintaining a positive user experience and supporting real-time decision-making. Delays could frustrate users, reduce system usability, and undermine trust in the service's efficiency.

## S-02 Requirements

### Functional Requirements

- **Stock Symbol Input:** Provide two text input forms for users to enter stock symbols, with validation to notify users if the symbols are invalid or unrecognized.
- **Stock Comparison:** Service compares two stocks provided by the user, compare the difference in their market cap, stock price, and profit margins.
- **Input Validation:** Service does not try to compare a stock that does not exist or compare a stock before the user enters two symbols.
- **Registry Interaction:** Be able to register and deregister with the service registry and send periodic heartbeat signals to confirm the service's active status.

### Quality Requirements

- **Responsive Design:** Ensure the interface adapts attractively to various screen sizes and device types, maintaining usability and visual appeal.
- **Clear Information Formatting:** Format the displayed information consistently, grouping related data together and aligning it neatly, preferably on the same line for easy comparison.
- **Quick Comparisons:** Present information in a way that allows users to identify which stock is performing better at a glance.

## ASRs

- Usability: The service must have a responsive design that works seamlessly across all platforms and device types. A responsive design ensures accessibility for a diverse user base, enhancing user satisfaction and broadening the service's usability.
- Usability: The service must present information in a way that is easy for users to understand and interpret. Clear and intuitive information presentation reduces cognitive load for users, enabling faster decision-making and improving the overall user experience.

## S-03 Requirements

### Functional Requirements

- Emissions Calculation: Given two cities, output estimated Co2 emission values for travelling between them by car, rail, or plane.
- City Pair Input: Accept city pairs in plaintext format (e.g., "City, Country") as input for the calculation.
- Emission Reduction Suggestions: Provide users with actionable suggestions to reduce emissions based on the specific city pair.
- Error Handling: Errors in city pairs or calculations are handled gracefully and alternatives are found.
- Registry Interaction: Be able to register and deregister with the service registry and send periodic heartbeat signals to confirm the service's active status.

### Quality Requirements

- Efficient Calculations: Calculations are performed in under 5 seconds by the server.
- Outlier Handling: Automatically detect and address outliers in CO2 emission estimates to ensure the accuracy and reliability of the output.

## ASRs

- Performance: Perform all calculations within 2 seconds of receiving the input. Fast calculations are essential for a smooth user experience, especially when users rely on quick results for decision-making. A delay could discourage use and diminish the perceived reliability of the service.

## S-04 Requirements

### Functional Requirements

- Daily Weather Summary: Provide a summary of the current day's weather for a given city based on user input.
- City Input Format: Accept city names in the format "City, Country" for weather lookup.

- Clothing Suggestions: Given the summary of the daily weather, output a suggestion for weather-appropriate clothing.
- Registry Interaction: Be able to register and deregister with the service registry and send periodic heartbeat signals to confirm the service's active status.

### Quality Requirements

- Weather Data Accuracy: Accuracy of the daily weather is not guaranteed as external API will be used. However, outliers and errors will be addressed.
- Efficient Clothing Suggestions: The server will return clothing results in less than 5 seconds, including API wait time.
- Reliability: To guarantee reliability (99% when running), a rolling backup will be implemented for common cities.

### ASRs

- Performance: Deliver clothing suggestions based on weather data within 2 seconds of receiving a valid input. Quick response times improve the overall user experience, ensuring interactions with the service feel seamless and efficient. This makes the service a dependable tool for daily planning while also encouraging users to rely on it frequently.

## S-05 Requirements

### Functional Requirements

- Exercise Plan Generation: Given a set of preferences (e.g. muscle group), output a plan of possible exercises to complete.
- User Profile Management: Enable users to view, save, delete, and create new exercise plans, ensuring their preferences and actions are remembered.
- Plan Completion Tracking: Allow users to mark exercise plans as complete using a simple flag to track their progress.
- Registry Interaction: Be able to register and deregister with the service registry and send periodic heartbeat signals to confirm the service's active status.

### Quality Requirements

- Efficient Plan Generation: Exercise plans are calculated in less than 1 second by the server.
- Plan Accuracy: Output the plan with a 95% accuracy rate (inaccurate meaning obvious errors with given exercise).
- Data Persistence: Save user profiles and exercise plans in a database to ensure continuity across microservice reboots.

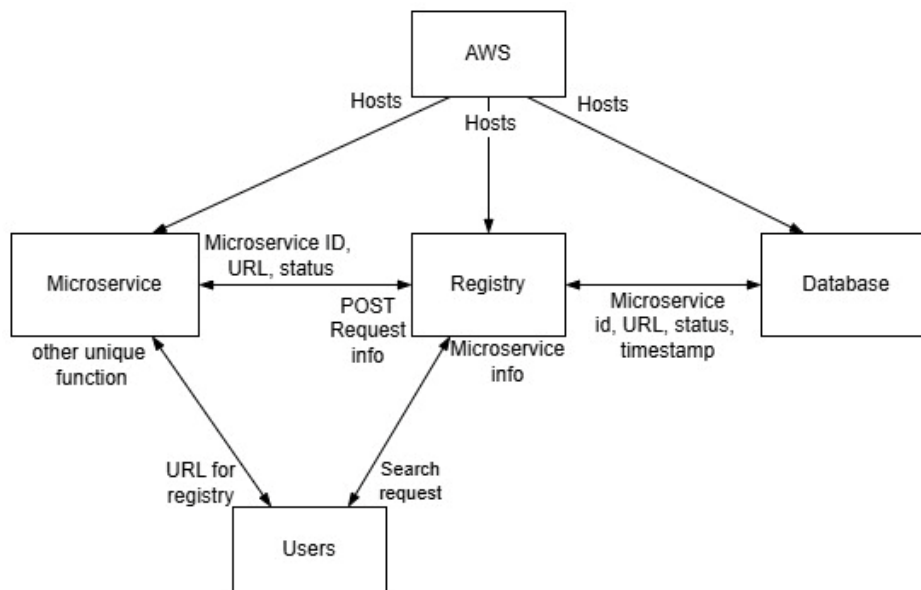
### ASRs

- Usability: User data, including profiles and exercise plans, must persist across microservice reboots. Fitness tracking relies on continuity, as users need consistent

access to their saved exercise plans, progress, and preferences to effectively manage their workouts. Data persistence is critical for maintaining a seamless user experience, ensuring users do not lose progress or customizations due to system restarts or updates.

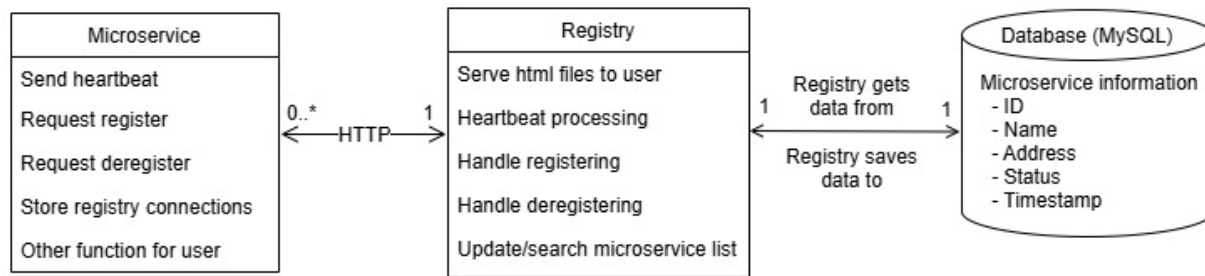
## Architecture and Design

### Context Model



The registry is the central node as it is the most important. It will send microservice ids, URLs, status', and timestamps between it and the database to provide to the user and update status based on heartbeats. The registry will send registered microservice info to the user and POST request info to the microservices about the outcome of their requests when they make requests with the registry. The microservices will send their ID, URL, and status to the registry so it can register it and facilitate heartbeats. It will also provide some function to the user. The user can send search requests to the registry and URLs of registries to the microservice that they want to register or deregister from. Finally, AWS will host all the software systems.

## Overall System Architecture



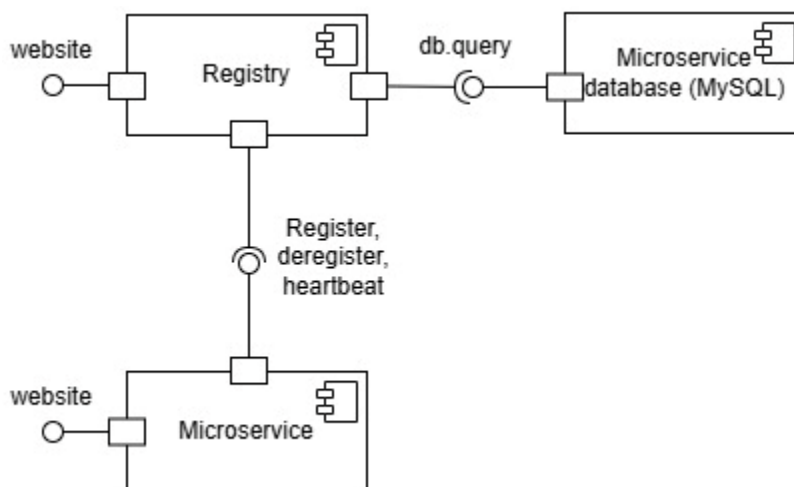
The module diagram is focused on the registry. In this way, the microservices are interchangeable from the eyes of the registry.

The registry will have a function to handle the heartbeats it receives from the microservices and update the microservice's availability accordingly. If it does not get a heartbeat from a microservice for 35 seconds, it will list it as unavailable. It will also provide a function to accept register and deregister requests from a microservice, updating its database and sending acknowledgements to the microservice.

The microservice will have a function to send heartbeats to the registry. It will do this every 15 seconds and include information differentiating it from other microservices. It will also implement a method to send register and deregister requests to the registry service over HTTP. These methods should wait for a response and then update its own internal storage accordingly. Finally, the microservice will provide some other function to the user that is to do with its own individual specification.

The database will store the information of each registered microservice including a unique id, name, address, status, and timestamp of the status for each microservice.

## Component Connector View



The microservice will provide an interface to send data between it and the registry. This interface will provide information about itself so that the registry can properly register,



deregister, and know its availability status. It will also provide a website interface for the user so they can use some function of it or have it register or deregister from a registry.

The registry will have two required interfaces. One for accepting information from a microservice to handle the requests it makes and a second to query the MySQL database that stores information about the registered microservices. It will also provide a website interface so the user can see what microservices are registered, search through them, view their status, and visit them if they choose.

The MySQL database will have a provided interface that the registry can use to query the data in it.

## Implementation Details

The registry was implemented using Node.js and a MySQL database.

The registry is hosted on an Amazon Web Services EC2 Instance at <http://3.135.187.132/>

The S-01 service was implemented using Node.js and the AlphaVantage API.

The S-01 service is hosted on an Amazon Web Services EC2 Instance at <http://3.133.139.3/>

The S-02 service was implemented using Node.js and the AlphaVantage API.

The S-02 service is hosted on an Amazon Web Services EC2 Instance at <http://18.190.176.152/>

The S-03 service was implemented using Python Streamlit.

The S-03 service is hosted on an Amazon Web Services EC2 Instance. This is best as Streamlit is always running and is more suited for a dedicated VM rather than serverless compute.

The S-04 service was implemented using Vite, Typescript React, Postgres, and Express.js

The S-04 service is hosted on a Replit Autoscale cluster. This allows it to automatically scale up according to demand. However, a dedicated VM is used as an alternative for cost reasons.

The S-05 service was implemented using Vite, Typescript React, Postgres, and Express.js

The S-05 service is designed to be hosted on a Replit Autoscale cluster. This allows it to automatically scale up according to demand. However, a dedicated VM is used as an alternative for cost reasons.

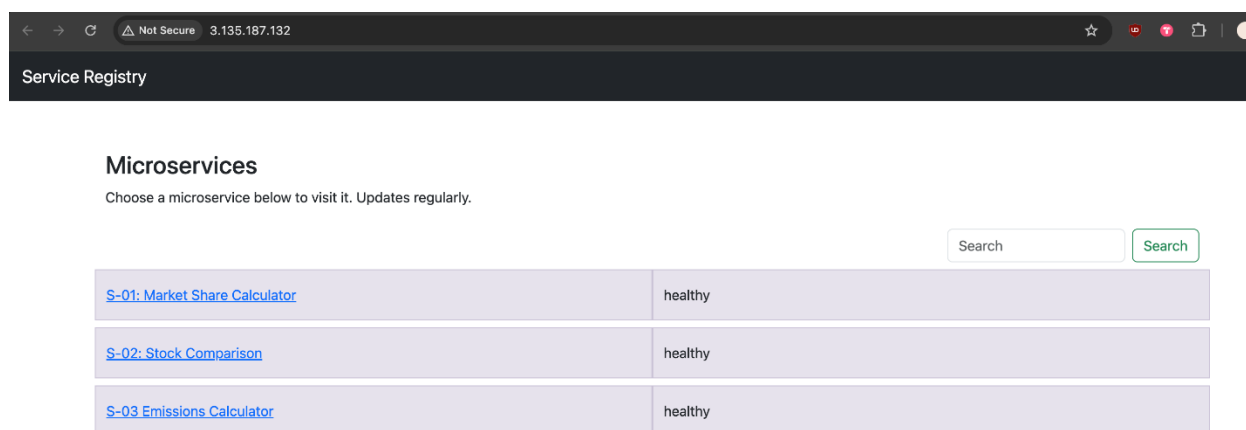
## Testing

The Quality Assurance for this system is composed of manual black-box tests. Each component is run through a set number of scenarios.

As an example, we go through each combination of registration and deregistration for every microservice and the registry. As we are doing it manually, we do Decision Table Testing which has the least amount of test cases but the most amount of thought involved.

## Operational Evidence

### Registry



User is presented with the service registry page where different services are listed. However, it is possible for none to be listed.

## Microservices

Choose a microservice below to visit it. Updates regularly.

<a href="#">S-01: Market Share Calculator</a>	healthy
---	---------

The search functionality performs a substring match. "m" would return all of the services here as it is a substring of all of the strings. But "moo" would not return any.

← → ↻ ⚠ Not Secure 3.135.187.132 ☆ 🔒 🔍

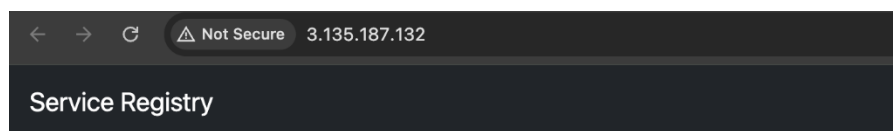
Service Registry

## Microservices

Choose a microservice below to visit it. Updates regularly.

Search returned zero results

The output when the search matches none of the service IDs.



## Microservices

Choose a microservice below to visit it. Updates regularly.

[S-01: Market Share Calculator](#)

[S-02: Stock Comparison](#)

[S-03 Emissions Calculator](#)

3.133.139.3

Each service is clickable and will bring you to the external service's IP.

← → ↻ Not Secure 18.190.176.152/settings

S-02: Stock Comparison Home Settings

## Settings

### Register in a Registry

Enter the URL of a registry below to register and start sending heartbeats to it.

Registry URL

### Deregister from a Registry

Select the URL of a registry below to deregister from it.

Removal of a service from the registry happens at the microservice side. There are logging messages.

← → ↻ Not Secure 3.135.187.132 ☆ 🔒 🔍 📄 🌐

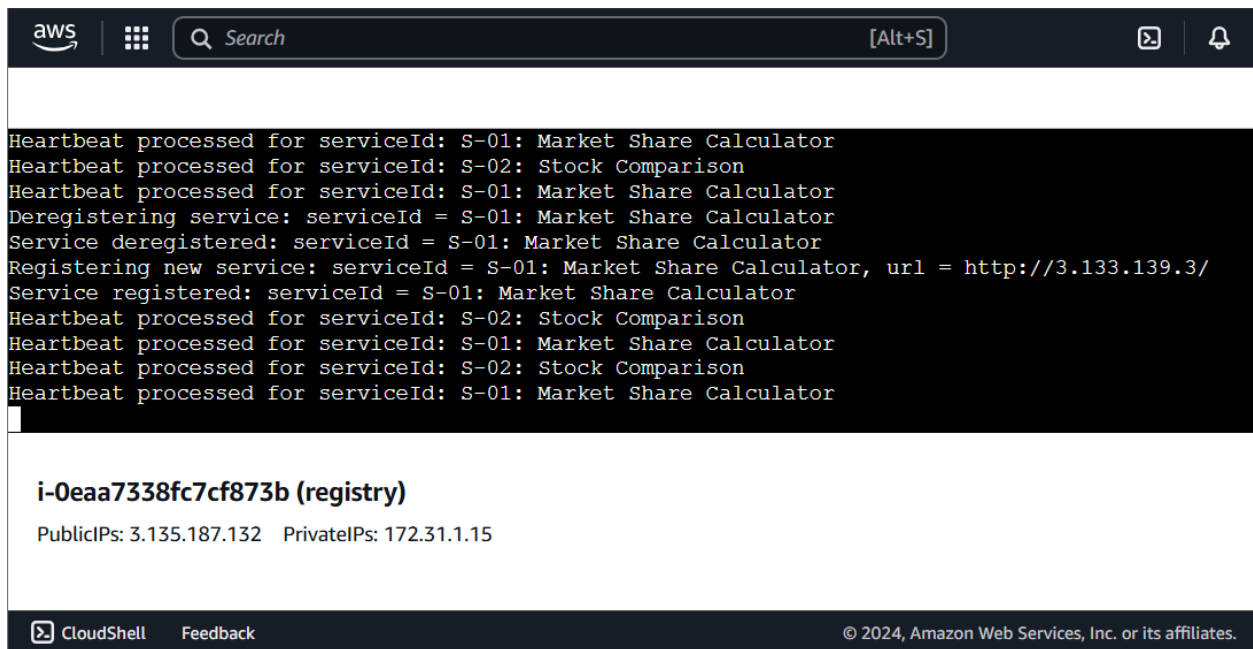
Service Registry

## Microservices

Choose a microservice below to visit it. Updates regularly.

<a href="#">S-01: Market Share Calculator</a>	healthy
<a href="#">S-03 Emissions Calculator</a>	healthy

Updated registry with the S-02 service freshly removed.



The screenshot shows the AWS CloudShell interface. At the top, there's a search bar with the text "Search" and a keyboard shortcut "[Alt+S]". Below the search bar, a terminal window displays the following log output:

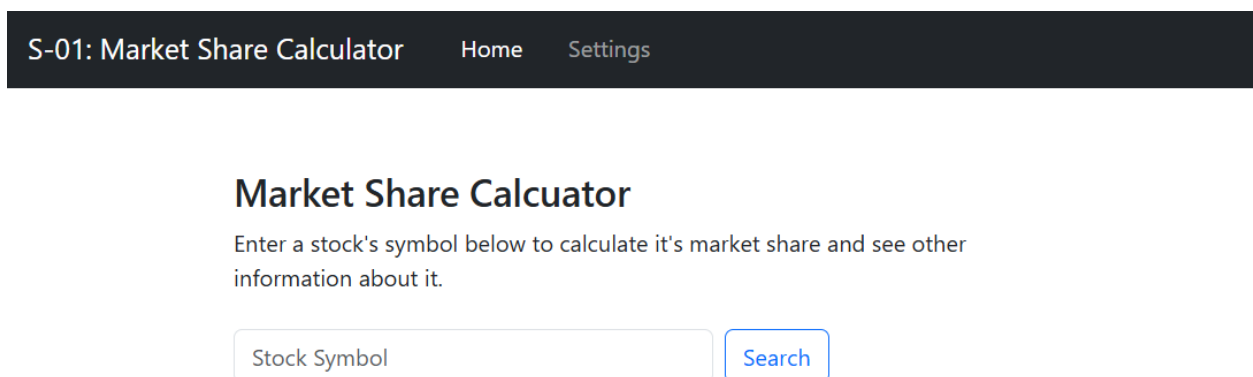
```
Heartbeat processed for serviceId: S-01: Market Share Calculator
Heartbeat processed for serviceId: S-02: Stock Comparison
Heartbeat processed for serviceId: S-01: Market Share Calculator
Deregistering service: serviceId = S-01: Market Share Calculator
Service deregistered: serviceId = S-01: Market Share Calculator
Registering new service: serviceId = S-01: Market Share Calculator, url = http://3.133.139.3/
Service registered: serviceId = S-01: Market Share Calculator
Heartbeat processed for serviceId: S-02: Stock Comparison
Heartbeat processed for serviceId: S-01: Market Share Calculator
Heartbeat processed for serviceId: S-02: Stock Comparison
Heartbeat processed for serviceId: S-01: Market Share Calculator
```

Below the terminal output, the instance ID **i-0eaa7338fc7cf873b (registry)** is shown, along with its public and private IP addresses: **PublicIPs: 3.135.187.132 PrivateIPs: 172.31.1.15**.

At the bottom of the CloudShell interface, there are links for "CloudShell" and "Feedback", and a copyright notice: "© 2024, Amazon Web Services, Inc. or its affiliates."

Console output shows S01 being deregistered and re-registered as well as a number of received heartbeats

## S-01



The screenshot shows the user interface for the "S-01: Market Share Calculator". At the top, there's a navigation bar with the title "S-01: Market Share Calculator" and two links: "Home" and "Settings".

Below the navigation bar, the main heading is "Market Share Calculator". Underneath this heading, a prompt reads: "Enter a stock's symbol below to calculate it's market share and see other information about it."

At the bottom of the form, there is a text input field labeled "Stock Symbol" and a blue "Search" button.

User is prompted to enter a ticket symbol

## Market Share Calculator

Enter a stock's symbol below to calculate it's market share and see other information about it.

[Search](#)

Company:	Apple Inc
Sector:	Technology
Price:	\$242.65
Market Capitalization:	\$3,667,848,593,000
Percent of Market:	6.645%

Program outputs the details of the stock

## Settings

### Register in a Registry

Enter the URL of a registry below to register and start sending heartbeats to it.

[Add to Registry](#)

### Deregister from a Registry

Select the URL of a registry below to deregister from it.

[Deregister](#)

User can navigate to settings page to register/deregister.

## Settings

### Register in a Registry

Enter the URL of a registry below to register and start sending heartbeats to it.

[Add to Registry](#)

### Deregister from a Registry

Select the URL of a registry below to deregister from it.

[Deregister](#)

Registration by inputting IP



## Settings

### Register in a Registry

Enter the URL of a registry below to register and start sending heartbeats to it.

Add to Registry

### Deregister from a Registry

Select the URL of a registry below to deregister from it.

Deregister

Successfully registered



Registration. Alerts are given for success/errors.

## Settings

### Register in a Registry

Enter the URL of a registry below to register and start sending heartbeats to it.

[Add to Registry](#)

### Deregister from a Registry

Select the URL of a registry below to deregister from it.

[Deregister](#)

Successfully deregistered



Successful deregistration from registry.

## S-02

## Stock Comparison

Enter two stock's symbols below to compare them.

[Search](#)

User lands on the homepage

## Stock Comparison

Enter two stock's symbols below to compare them.

<input type="text" value="aapl"/>	<input type="text" value="ibm"/>	<input type="button" value="Search"/>
-----------------------------------	----------------------------------	---------------------------------------

User enters two companies publicly traded in the United States (e.g. NYSE/NASDAQ).

## Stock Comparison

Enter two stock's symbols below to compare them.

<input type="text" value="aapl"/>	<input type="text" value="ibm"/>	<input type="button" value="Search"/>
-----------------------------------	----------------------------------	---------------------------------------

Company:	Apple Inc	International Business Machines
Sector:	Technology	Technology
Market Cap:	\$3,667,848,593,000	\$211,743,703,000
Stock Price:	\$242.650	\$229.000
Profit Margin:	24.00%	10.20%

The program outputs a comparison of the two companies' financial statistics.

## Settings

### Register in a Registry

Enter the URL of a registry below to register and start sending heartbeats to it.

[Add to Registry](#)

### Deregister from a Registry

Select the URL of a registry below to deregister from it.

[Deregister](#)

The Settings page allows the user to enter in a registry URL in order to register the service onto the registry.

## Settings

### Register in a Registry

Enter the URL of a registry below to register and start sending heartbeats to it.

[Add to Registry](#)

### Deregister from a Registry

Select the URL of a registry below to deregister from it.

[Deregister](#)[Successfully registered](#)

The output is a success/error message for the registration. The user can also deregister from a specific registry.

## Settings

### Register in a Registry

Enter the URL of a registry below to register and start sending heartbeats to it.

[Add to Registry](#)

### Deregister from a Registry

Select the URL of a registry below to deregister from it.

[Deregister](#)

Successfully deregistered



An error or success message is the output for a deregistration.

## S-03

Eco Travel Calculator Service Registry

### Eco Travel Calculator

Compare the carbon footprint of different transportation methods for your journey. Enter your starting point and destination to get started.

Starting City

Destination City

🔥 Enter your journey details above to calculate the carbon footprint.

Data sources:

- Transport emission factors: European Environment Agency
- Distance calculations: GeoPy

User is able to enter a starting city and destination city.

## Eco Travel Calculator

Compare the carbon footprint of different transportation methods for your journey. Enter your starting point and destination to get started.

Starting City

Toronto, Canada

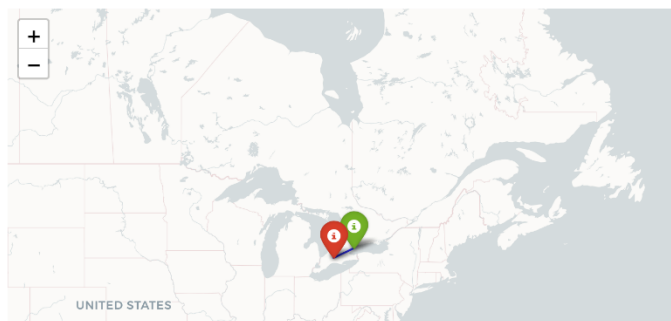
Destination City

London, Canada

### Journey Details

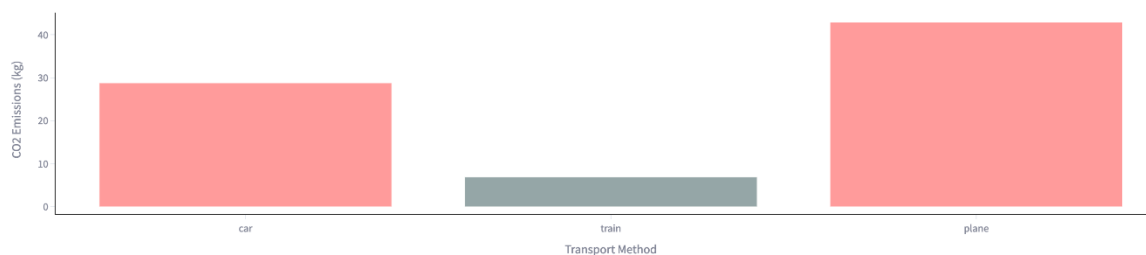
Total Distance: 168.21 km

### Route Visualization






After entry, the output comprises details and visualizations of the journey.

CO2 Emissions by Transport Method (kg)



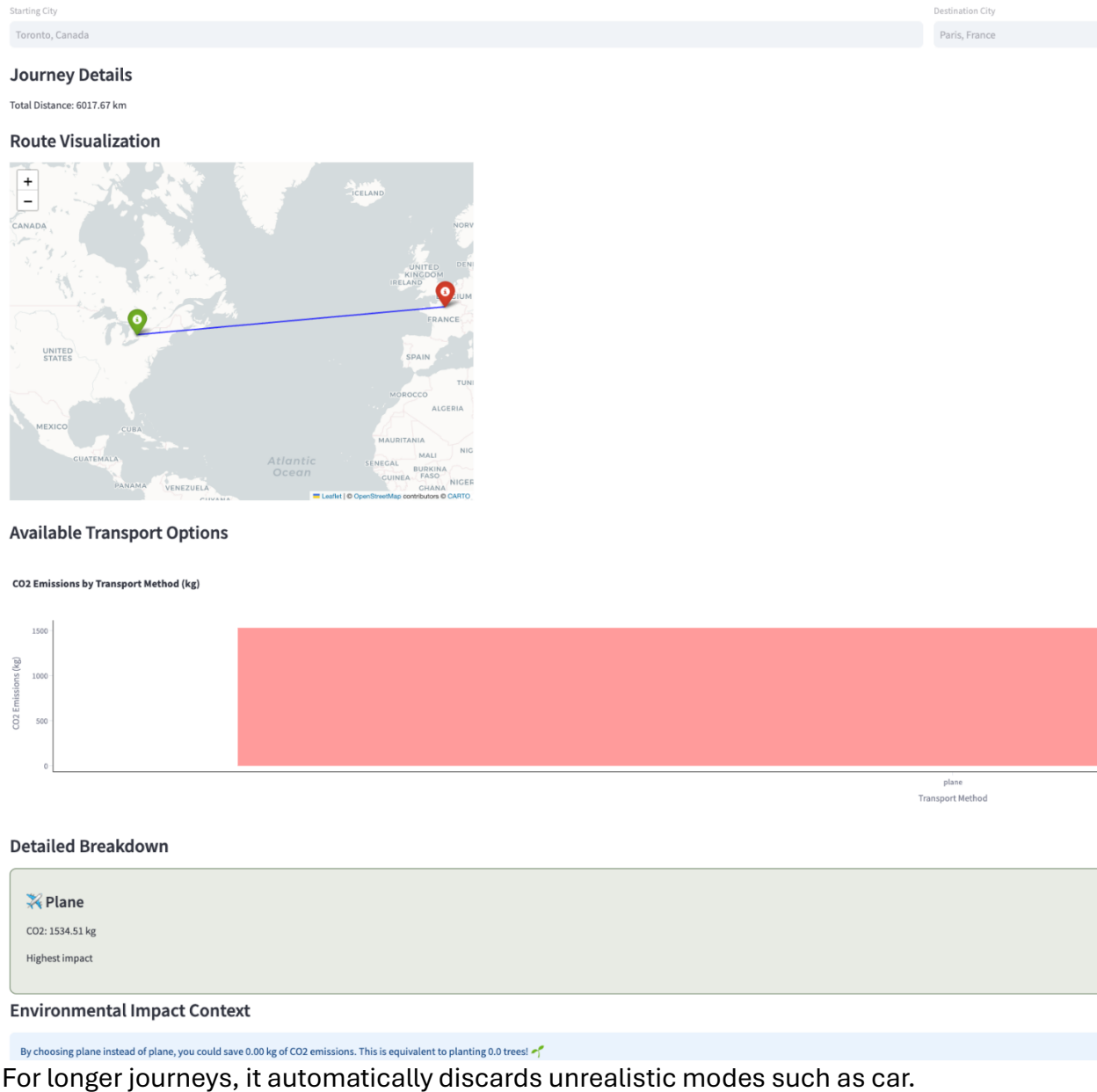
### Detailed Breakdown

 <b>Car</b> CO2: 28.76 kg High impact	 <b>Train</b> CO2: 6.90 kg Medium impact	 <b>Plane</b> CO2: 42.89 kg Highest impact
--	---	---

### Environmental Impact Context

By choosing train instead of plane, you could save 35.99 kg of CO2 emissions. This is equivalent to planting 1.7 trees! 🌱

The output also includes emissions details divided between different modes of transport.



## Service Registry

Register and deregister services in the registry.

### Register Service

Service ID

S-03 Emissions Calculator

Registry URL

http://3.135.187.132/

Register Service

Service registered successfully!

The registry settings page operates the same way, but for this service you can list the service with a custom ID.

Service Registry

### Microservices

Choose a microservice below to visit it. Updates regularly.

Search

Search

<a href="#">S-02: Stock Comparison</a>	healthy
<a href="#">S-01: Market Share Calculator</a>	healthy
<a href="#">S-03 Emissions Calculator</a>	healthy

### Registry View



## Deregister Service

Service ID to Deregister

S-03 Emissions Calculator

Registry URL

http://3.135.187.132/

Deregister Service

Service deregistered successfully!

Deregistration works the same way, the user just has to input the correct service ID.

### S-04

#### Weather & Outfit Recommendations

Registry Connections

Location

Enter city (e.g., London or London, GB)

Get Weather

The user is presented with landing page that asks for city.

## Weather & Outfit Recommendations

[Registry Connections](#)

Location

[Get Weather](#)

### London, Canada

# -1°C

Feels like: -1°C

Moderate snow fall

Humidity: 85%

### Outfit Recommendation

**Top:**

Insulated winter coat

**Bottom:**

Snow pants

**Accessories:**

- Winter boots
- Warm hat
- Gloves
- Scarf

*Snow protection gear*

Given a city, the program will output the current weather and some outfit recommendations.

## Registry Connections

[Weather & Outfits](#)

Registry URL

[Add Registry](#)

<http://3.135.187.132/>

[Deactivate](#)[Deregister](#)

<http://3.135.187.132/>

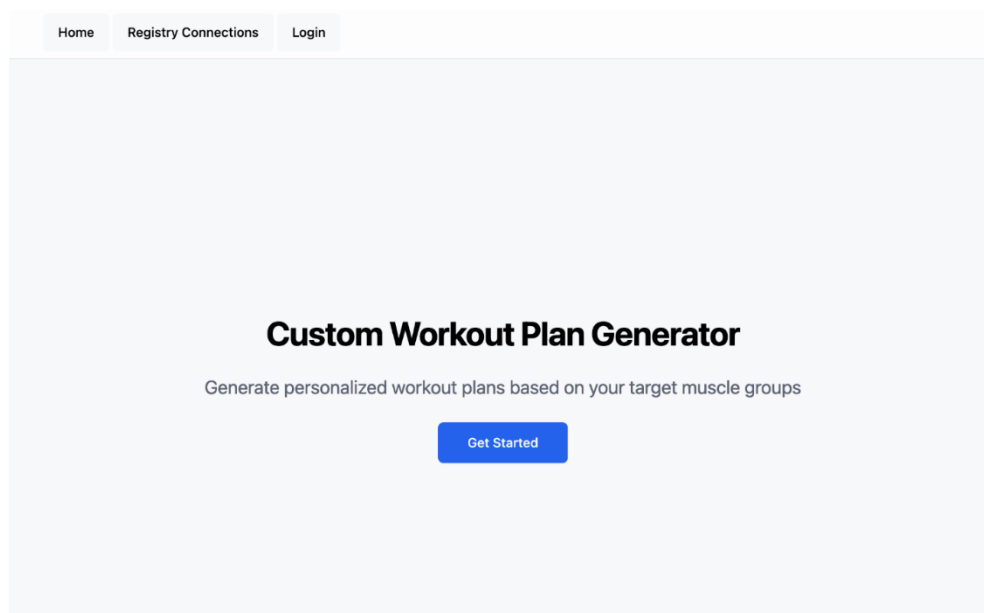
[Deactivate](#)[Deregister](#)

<http://3.135.187.132/>

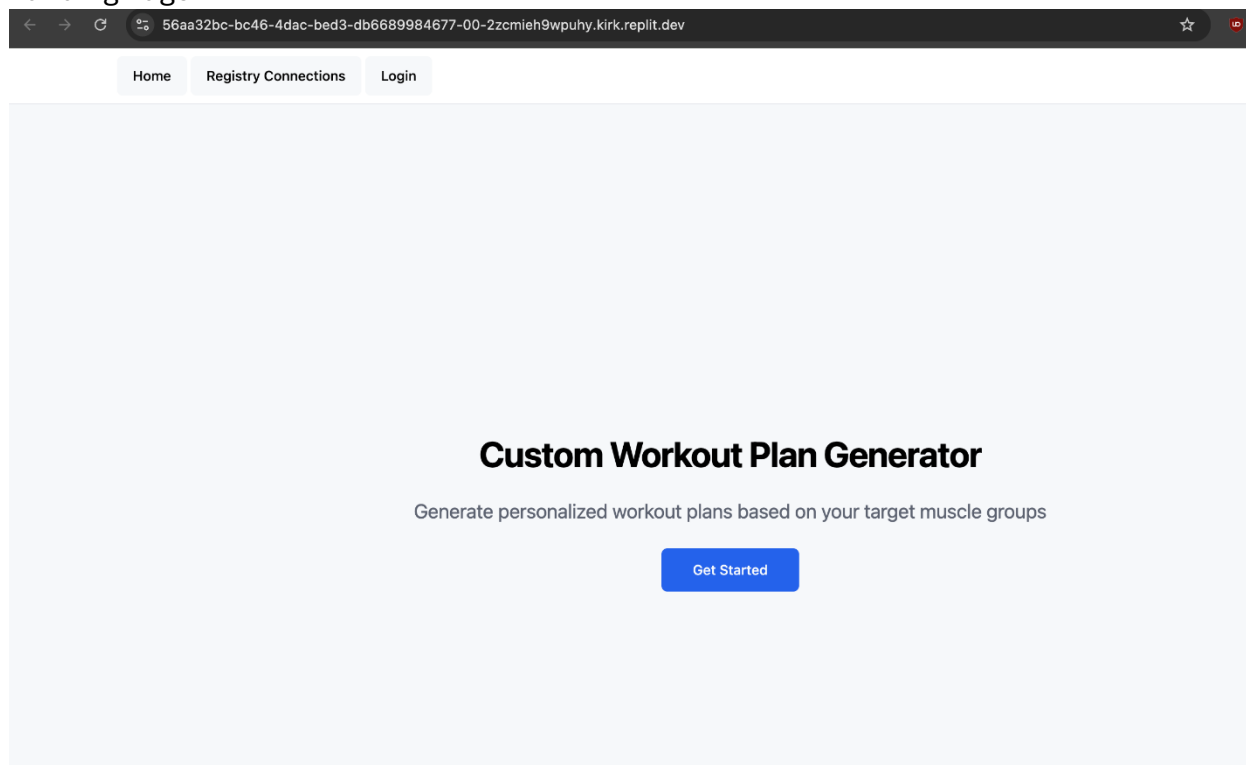
[Deactivate](#)[Deregister](#)

The registry connection page allows registration and deregistration of registry connections.

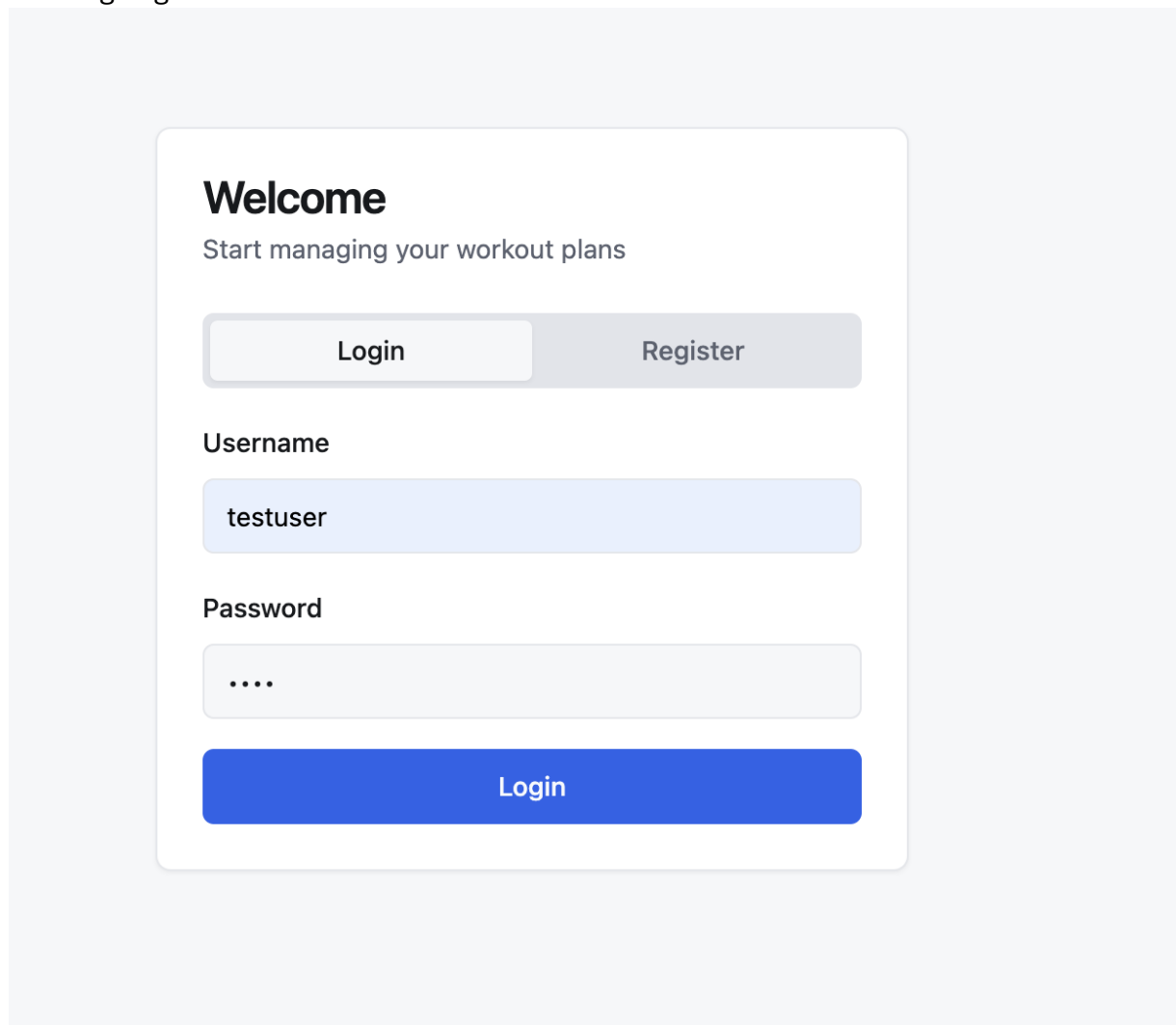
## S-05



## Landing Page



## Landing Page



The image shows a landing page with a light gray background. In the center is a white rounded rectangle containing the login form. At the top of the form is the heading 'Welcome' in bold black text, followed by the subtext 'Start managing your workout plans'. Below this are two buttons: 'Login' (light gray) and 'Register' (medium gray). Underneath the buttons are two input fields. The first is labeled 'Username' and contains the text 'testuser'. The second is labeled 'Password' and contains four dots. At the bottom of the form is a large blue button labeled 'Login'.

**Welcome**

Start managing your workout plans

Login Register

Username

testuser

Password

....

Login

## Login page

Home Dashboard Registry Connections

Welcome, testuser! [Logout](#)

### Generate New Workout

Workout Name  Select Category ▼

[Generate Workout](#)

#### Lower

lower

**Deadlifts**  
3 sets x 8 reps

**Leg Press**  
3 sets x 10 reps

**Bulgarian Split Squats**  
4 sets x 8 reps

**Box Jumps**  
3 sets x 14 reps

[Completed](#) [Delete](#)

#### Upper

upper

**Overhead Press**  
5 sets x 12 reps

**Lat Pulldowns**  
4 sets x 13 reps

**Skull Crushers**  
5 sets x 15 reps

**Hammer Curls**  
3 sets x 10 reps

**Bench Press**  
4 sets x 14 reps

[Mark as Complete](#) [Delete](#)

#### Neither

combined

**Curls**  
5 sets x 10 reps

**Lateral Raises**  
5 sets x 13 reps

**Lunges**  
4 sets x 12 reps

**Lat Pulldowns**  
5 sets x 13 reps

**Push-ups**  
4 sets x 14 reps

[Mark as Complete](#) [Delete](#)

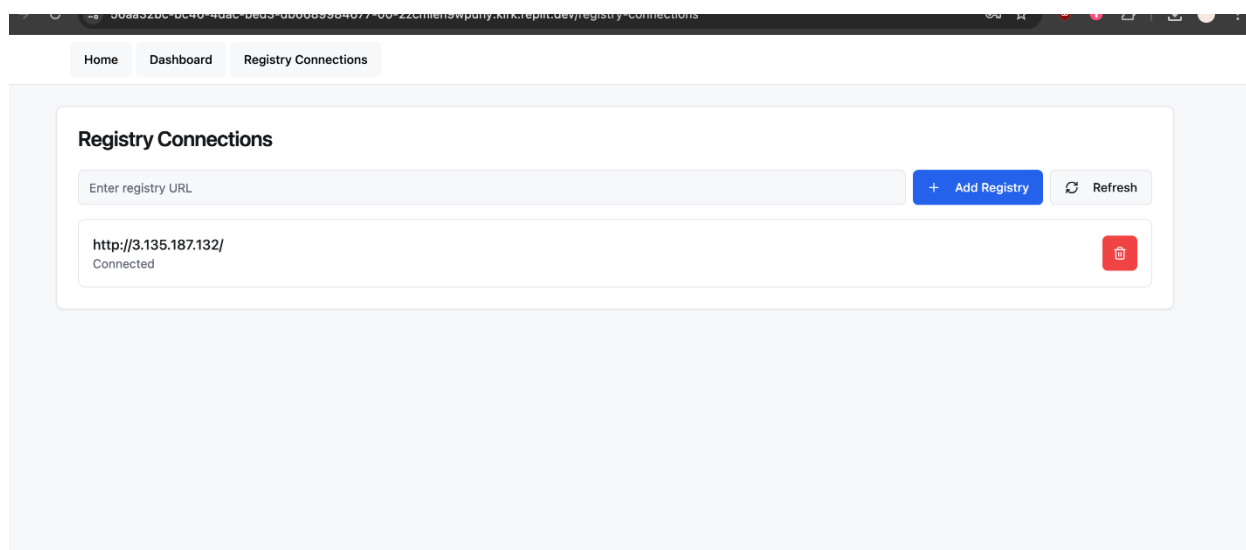
User can create new workout plans by entering a name and body section. A new plan is generated as output. Users can mark it as complete or delete the plan. These are all saved in a Postgres Database bundled with the program.

Home Dashboard Registry Connections

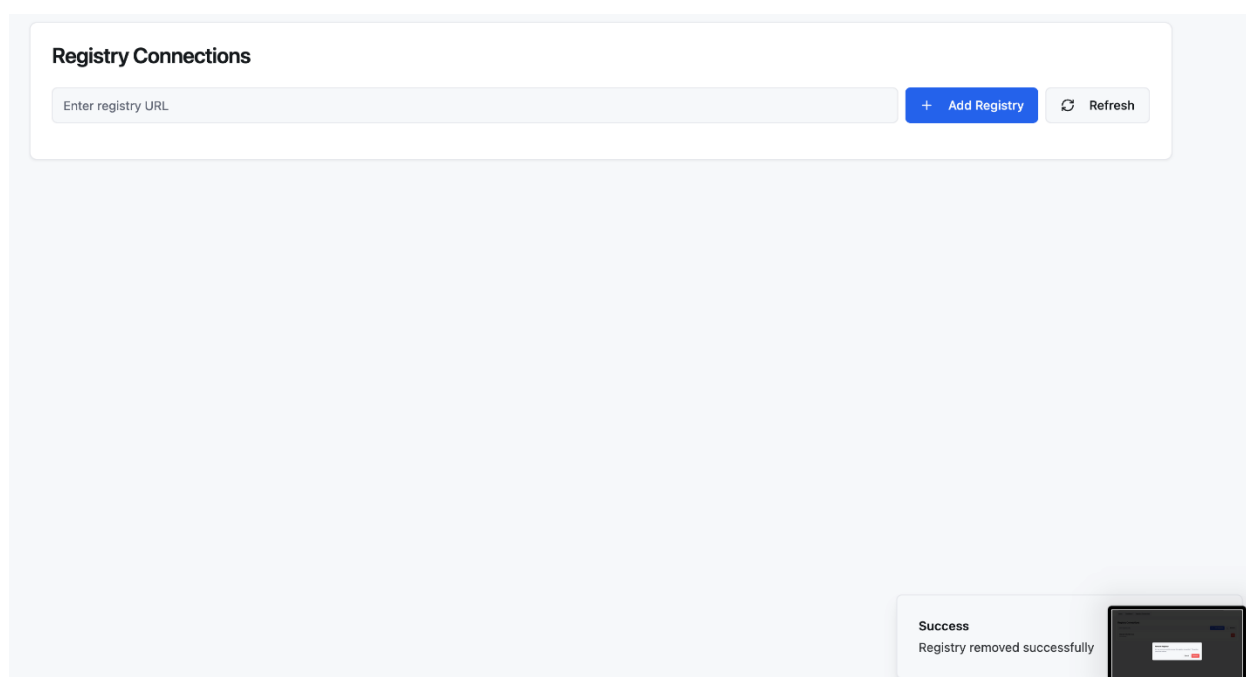
### Registry Connections

[+ Add Registry](#) [Refresh](#)

Registry Connections are a feature.



User can view all current connections to registries.



The user can also remove registry connections.

## Appendix

### Task Breakdown

## Task Breakdown

Colin Brown	Registry & S-01, 02 Implementations, Architectural Diagrams, ASRs
Xiaolin (Eric) Wang	AWS and MySQL Implementation, Registry improvements
Kaiyi Hou	Reports, S-03 Implementation
Le (Leon) Zhu	S-03, 04, 05 Implementation, PowerPoints, Reports
Zoe Kaute	Reports, Project Planning

## References

Bass, L., Clements, P., Kazman, R., & Klein, M. (2021). *Software architecture in practice* (4th ed.). Addison-Wesley Professional.