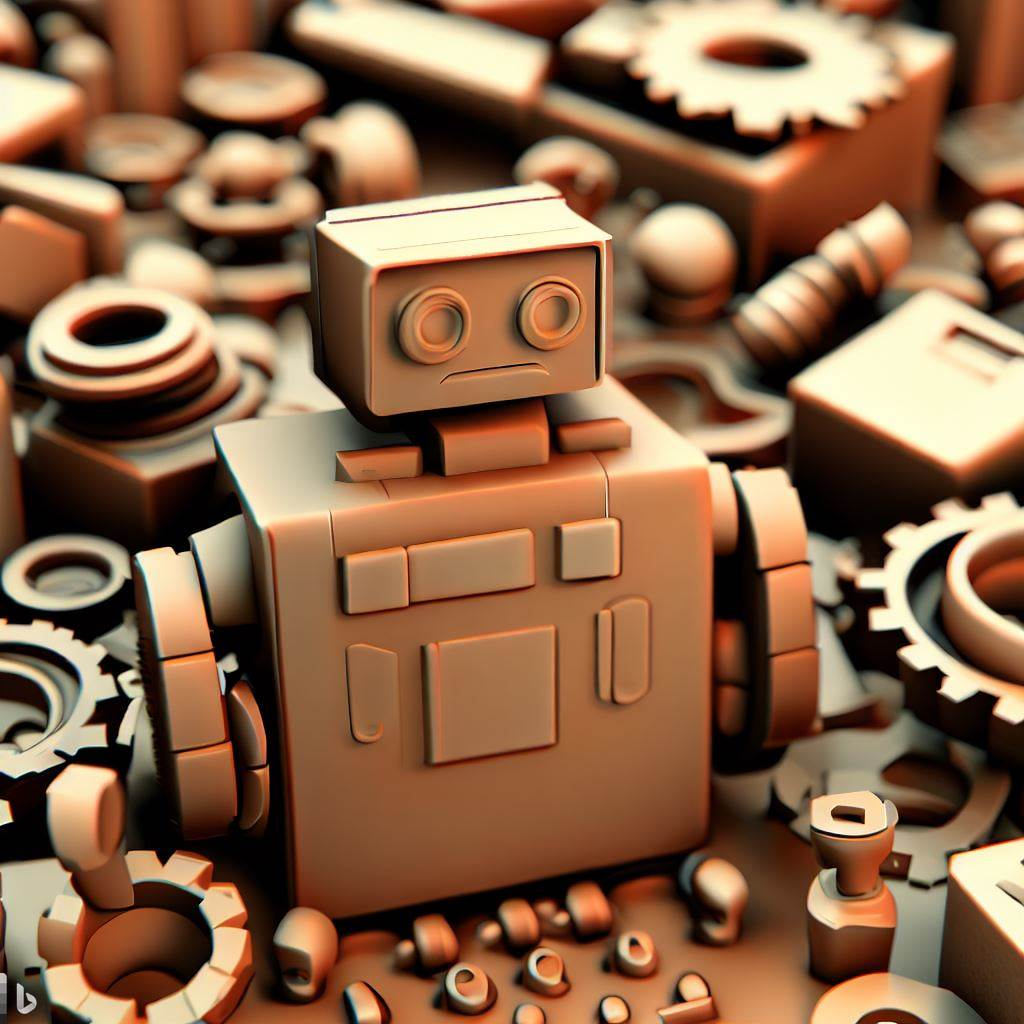
The Accu Bot Factory



# Introduction

Welcome to the Accu Bot Factory Challenge! This coding challenge is part of the interview process for the PHP Developer position at Accu. The purpose of the challenge is to assess your skills and expertise in PHP and Laravel, and to evaluate your ability to build a web application that uses external data sources.

The challenge requires you to build a web application that loads order data from a CSV file and component data from an engineering component API, and uses this information to generate amusing robot names based on the most prevalent category of components in each order. You will also need to display the results in a table on a web page.

To successfully complete the challenge, you will need to use Laravel to build the application and implement additional features to earn bonus points. You will also be evaluated based on your code quality, use of Laravel's database integration and command-line interface, and user interface design.

Read on for more detailed requirements and example inputs and outputs. Good luck!

# Requirements

To complete the Accu Bot Factory Challenge, you must fulfil the following requirements:

* Load order data from a CSV file containing order ID, customer name, SKU, and quantity fields.
* Retrieve component data from an engineering component API that returns a paginated JSON response with the following fields: SKU, description, category, and weight.
* Calculate the total weight of each order based on the components in the order.
* Generate an amusing robot name for each order based on the most prevalent category of components in the order, and display the name in a table on a web page.
* Allow users to click on an order to view all the components and edit the robot name of an order.
* Use Laravel to build the application and implement database integration and command-line interface functionality.
* Write clear and concise documentation for how to set up and run the application.

You will be awarded bonus points for implementing any of the following features:

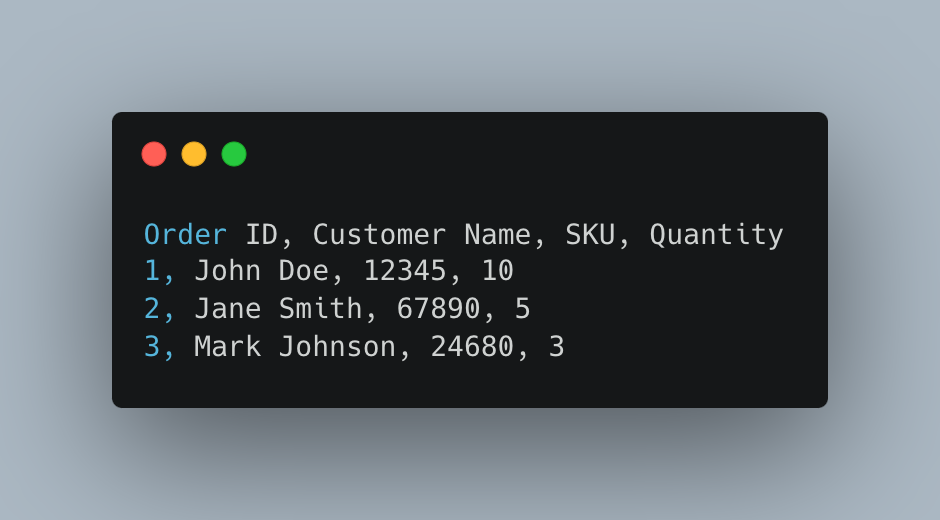
* Implement search functionality to search for orders by customer name.
* Use Laravel queues to process the order data and component data in the background.
* Implement user authentication and authorization.
* Implement testing for the application using PHPUnit or another testing framework.
* Use the service container to enable certain implementations to be swapped out in the future.
* Load the orders CSV into the database and the products from the API synced into the database using artisan commands.

Keep in mind that your application should be well-designed and easy to use, and should follow best practices for PHP and Laravel development.

# 

# Example Inputs

The orders CSV will contain the following columns in the specified order: order ID, customer name, SKU, and quantity. Here's an example:



The component data can be retrieved from the following API URL:

<https://nt5gkznl19.execute-api.eu-west-1.amazonaws.com/prod/products>

The API returns a paginated JSON response with the following fields: SKU, description, category, and weight. Here's an example of what the response might look like:

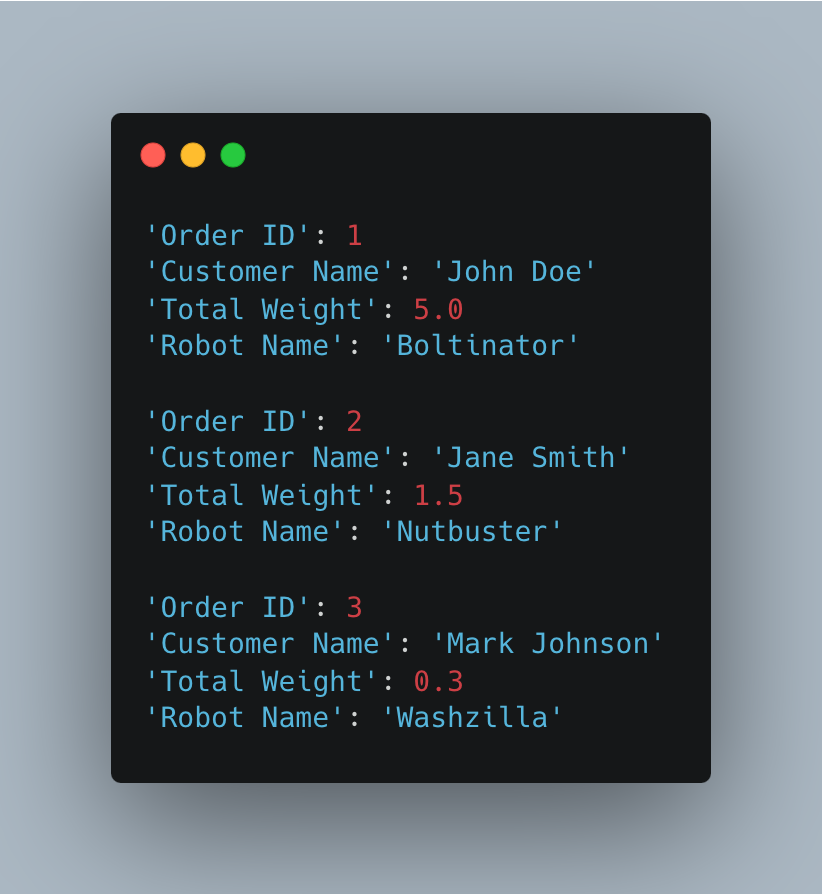


Note that the example response only includes 5 products. In the actual response, there will be a total of 100 products returned, with 10 products per page.

# 

# Example Outputs

Here are some example outputs for the Accu Bot Factory web application, based on the example inputs provided above:



In the above example, the robot names are intended to be amusing and creative, based on the most prevalent category of components in each order. The actual robot names generated by your implementation may vary depending on the data in the orders CSV and the engineering component API, so feel free to be as creative as possible when generating the robot names.

# Bonus Points

In addition to the basic requirements outlined above, you can earn bonus points by implementing any of the following additional features:

* Implement Laravel queues to process time-consuming tasks asynchronously, such as loading in orders or syncing the products from the engineering component API.
* Enable the web application to view and edit the robot name of an order by clicking on the order in the orders table.
* Use Laravel's service container to enable certain implementations to be swapped out in the future, such as the order repository or the component service.
* Implement a search functionality to allow users to search for orders by customer name.
* Implement unit or feature tests for your code using PHPUnit or Pest.
* Use React, Vue.js or another JavaScript framework to enhance the user interface and experience.

Feel free to add your own creative ideas for bonus points as well!

Note that bonus points are optional, and the main focus of the challenge is to demonstrate your skills and expertise in PHP and Laravel, as well as your ability to design and implement a working web application based on the provided requirements.

# 

# Evaluation Criteria

Your submission will be evaluated based on the following criteria:

* Implementation of the basic requirements outlined in the challenge, including loading in orders from a CSV file, retrieving component data from the engineering component API, generating amusing robot names based on the most prevalent category of components in each order, and displaying the results in a table on a web page.
* Code quality and adherence to best practices in PHP and Laravel, including proper use of Laravel's database integration and command-line interface.
* Creativity and quality of the generated robot names.
* Implementation of bonus features, if applicable.
* User interface design and user experience.

Please ensure that your code is well-documented and easy to understand, and that your web application is user-friendly and intuitive to use.

We're excited to see what you come up with! Good luck!