# Demo Script

## 1.Introduction:

Good morning everyone, we are from lab SCS2 and today we are here to present our project ChompTrack.

These are the topics which we would be covering in our presentation

## 1.1 Problem

Singapore is a fast paced society, where adults are often too busy to keep track of what they eat on the daily, and would often opt for convenient foods such as fast foods or ready to eat processed foods as they are easily stored and can be prepared on the fly. However in the long run, this would be detrimental to the person's health as they would not only fail to meet the daily macronutrient requirements, but would also have a build up of chemicals from these processed food.

Another problem is also that it is difficult to maintain a budget as costs of living in Singapore have been rising year on year with grocery prices constantly fluctuating in today's economy.

## 1.2 Solution

hence , Welcome to ChompTrack, your all-in-one meal planner website, crafted to make healthy eating and budgeting simpler. Our platform offers an extensive range of easy-to-follow recipes, helping users plan daily meals while tracking nutritional intake to meet their dietary goals. Additionally, ChompTrack includes grocery price tracking, empowering users to manage their food budget effectively.

## 1.3 API Used

The API we used is spoonacular API. The key features include:

- A vast array of recipes
- Access to detailed nutritional data
- And real-time groceries prices

## 1.4 Tech Stack

Our tech stack is organized into three main components: the front-end, back-end, and database. The front-end is built using HTML, CSS, and JavaScript to create an user interface. For the back-end, we used Python and Flask, where Flask serves as both the server framework and a REST API layer, connecting the front-end with the back-end functionality. Data is stored in a MySQL database, which we access using the `mysql.connector` library. This architecture allows seamless communication between the user interface and the database, enabling efficient data

management and dynamic interactions.

## 1.5 Main Functionalities

Our app has several core features:

- **Authentication** for secure Login and Registration.
- **Profile Creation** which includes calculating personalised daily nutrition intake.
- **Meal Planning** lets users add, complete, and remove meals as needed.
- **Nutrition Tracking** helps users compare daily intake against their recommended goals.
- **Recipe Selection** includes viewing, filtering recipes
- **Past Recipe View** include searching past recipes by name or date.
- **Grocery List** allows users to view their shopping list and mark off ingredients as they shop.

These features make meal planning and nutrition tracking simple and effective.

## 1.6 Use Case Diagram

Our system's main features are depicted in our use-case diagram as shown here.

At the center, we see the main actions available to the **User**, such as **Registering** and **Logging in**, which also allow them to **Create a Profile** and **View Suggested Nutrition Intake**.

The user can **Manage Meal Plans** by adding, completing, or removing meals. They also have access to **Statistics** for tracking their nutrition.

For recipes, they can **View a Recipe List**, **Filter Recipes**, and get **Recipe Details**. Additionally, they can **View History** and **Search Past Meal Plans**.

Finally, there's **Grocery List Management**, where users can mark items with a strikethrough as they shop.

Now, we will move on to the live demo.

## 1.7. Live Demo

RegisterInterface
This is the landing page of our website. As a new user. They would have to register for an account first. We can do that by clicking on the "New User" button

Press New User
Username:
Email: user1@gmail.com

Password: #Pass123
Confirm Password: #Pass123
Press Register

If there are missing fields, the System will prompt the user to fill in the blank fields

Username: user1
Press Register

Now that we filled in the blanks, if the username or email already exists, the System will display an error message and inform the user that the registration failed

Username: newUser

The system also checks if the email is valid.
Here are some invalid inputs:

The email here does not have @gmail.com
Email: user
Press Register

The email here does not have gmail.com
Email: user@
Press Register

The email here does not have .com
Email: user@gmail
Press Register

As shown, if the email is invalid, the system will prompt the user to enter a valid email.

This is the format of a valid email

Email: newuser@gmail.com

The requirements of the password are: 8-16 characters, 1 uppercase letter and 1 special character

Here are some invalid inputs of the password:

The password here is missing a special character (read out the password)
Password: Pass123
Confirm Password: Pass123
Press Register

The password here is missing an uppercase letter (read out the password)
Password: #pass123
Confirm Password: #pass123
Press Register

The password here does not have at least 8 character (read out the password)
Password: #Pass12
Confirm Password: #Pass12
Press Register

Hence, if the password does not meet the requirements, the System will display an error message and prompt the user to abide by the password requirements

Password: #Pass123
Confirm password: Pass123
Press Register

If the password and confirm password does not match, as shown here where the confirm password does not have the hashtag symbol, the system will inform the user that the passwords do not match

Confirm Password: #Pass123
Press Register

Now that everything is valid, we will press Register and the system will bring us to the profile creation page

ProfileCreationInterface
Here, the user will have to fill in their name, age, gender, height, weight, activity level and specify any diet type or allergy.

If there are missing fields, the system will show an error message and prompt the user to fill in the blank fields

Press Next

If the user inputs a negative value for age, height or weight, the system will inform the user that the value must be greater than or equal to 1

The value for age, weight and height here are all invalid
Name: user
Age: -1

Gender: Female
Height: -1
Weight: -1
Activity level: Lightly active
Press Next

Now, The age is valid but height and weight is still invalid
Age: 20
Height: -1
Press Next

The weight here is invalid
Height: 160
Weight: -1
Press Next

Weight: 50
Press Next

Now that all the information is valid, we will press Next and the system will show that the profile has been updated successfully

SuggestedIntakeInterface
Based on the information that the user has entered, the system will calculate and show the suggested estimated daily nutrition intake. This is to help users plan their meal in a way that meets their personalised nutritional requirements.

LoginInterface
After pressing the ‘Let’s start!’ button, the user will be directed to the ‘My Plan’ page. At the top is the navigation bar which enables the user to toggle between the ‘My Plan’ page, ‘Discover’ page, ‘Grocery List’ page and ‘History’ page. The user is also able to log out.

Press Log Out

Now, we will try logging in with the account that we have just created. As we already have an account, we will press existing user

Press Existing User

Press Next

If there are missing fields. The system will display an error message

If the username does not exist or if the password does not match the username, the system will display the same error message.

Here are the two cases:

Username: user0
Password: #Pass123
Press Next

Username: newUser
Password: Pass123
Press Next

Now, we will input the correct details.

Username: newUser
Password: #Password123
Press Next

After verifying the information, the user is logged in and directed to the ‘My Plan’ page

MyPlanInterface
For the ‘My Plan’ page, there is a customised greeting for each user based on the name they entered when they created their profile and the date below changes accordingly everyday. The suggested estimated daily nutrients intake is also reflected.

On the right, there is a daily progress section that updates users on how much they have achieved for their daily nutrition in terms of percentages and this serves as an easy way for users to know immediately how far away they are from achieving their suggested nutrition intake.

At the bottom is an interactive calendar for users can toggle between different weeks

Press Left & Right arrows

We can view the meal plans for a specific date by pressing on the date

Press different dates

As you can see here, there is currently no meal planned as we have not added any recipes yet. We are able to add recipes by pressing on this ‘Add recipe’ button. This would bring us to the ‘Discover page’

Press ‘Add recipe’ button

DiscoverInterface

This is the ‘Discover’ page where the system draws the recipe list from the API and users can find the different recipes that suit their preference.

Scroll down the recipe list & Press Load More

Users are able to filter the recipes based on cost, estimated preparation time and cuisine

For cost, the user can input the minimum and maximum amount in dollars.

Min $ 0
Max $ 3
Show the recipes

This will show all the recipes that cost between 0 to 3 dollars.

The user can also input the minimum and maximum time in minutes.

Min (minutes) 0
Max (minutes) 45
Show the recipes

This will show all the recipes that require 0 to 45 minutes to prepare.

The user is also able to choose one or multiple cuisines

Tick American
Show the recipe

Tick Mediterranean
Show the recipes

This will show all the recipes categorised under the specific cuisines chosen

The user is also able to choose the course

Remove filters for cuisine

Click on Breakfast
Show recipe

Click on Lunch

Show recipe

Remove filter for course

RecipeInterface

If the user is interested in the recipe, clicking on the image will display more information about the recipe.

The cost, estimated preparation time, cuisine, ingredients, nutrition and a link to the recipe's instructions will be displayed. To view the step by step instructions of the recipe, the user can click on the link.

Should the user decide to add the meal in his meal plan, he can click on the 'Add to Meal Plan' button, there will be a popup for the user to select the date and course for the meal plan.

Press 'Add to Meal Plan' button

If the user does not select a date or a meal type and presses confirm, the system will display an error message and prompt the user to select both a date and a meal type for the meal plan.

If the user only chose the date, the system will prompt the user to select a course

Date: 07/11/2024 - 7th Nov
Press Confirm

If the user only chose the course, the system will prompt the user to select a date

Clear the date
Course: Lunch
Press Confirm

Now, we will enter all the required information and press Confirm

Date: 07/11/2024 - 7th Nov
Press Confirm

This will bring us to the 'My Plan' page.

Press on 7 Nov on the calendar
Show recipe

The recipe we added will show up on the date we selected and the course we chose

Now, we will try adding another recipe on the same date and course

Go to Discover Page
Press on a recipe
Press ‘Add to Meal Plan’ button
Date: 07/11/2024 - 7th Nov
Course: Lunch
Press Confirm

As there is already a meal planned on that date and course, the system will notify the user that the Meal Plan for this date and course exists

GroceryListInterface
After the user has added his meal plan, the system will add all the ingredients needed for the recipes that we added to the meal plan to the grocery list.

Press on Grocery List

The list shows all the ingredients needed for all the meal plans for the week. The list shows the item, quantity, price and total cost. The user is able to toggle between different weeks

Press on left & right arrow

After the ingredient has been purchased, the user can click on the ingredient name and the ingredient will be striked through, signifying that the purchase was completed. The total cost will also update accordingly

Press on an item to strike through

The user is also able to unstrike the item

Press on the same item to unstrike

Now, we will trying adding another recipe to our meal plan and the grocery list should be updated

Go to ‘My Plan’ page
Press on Add recipe
Click on a recipe
Press ‘Add to Meal Plan’ button
Date: 07/11/2024 - 7th Nov
Meal Type: Dinner
Press Confirm

Go to Grocery List page

The list should be updated with the ingredients needed for the recipe that we have just added in. The total cost will also update accordingly

HistoryInterface
Users are able to view past recipes they have used

Go to History page

If no past recipes are used, the system will notify the users that no past recipes are used

After the user has cooked and consumed the meal, he can go to the 'My Plan' page and mark the meal as completed by clicking on the tick icon and the recipe will turn green. The daily progress section is also updated to reflect the nutrients the user has consumed from their meal.

Go to My Plan page
Press on 7th Nov
Click on 'tick' icon on the recipes (Lunch & Dinner)
Show pie charts under daily progress
Go to History Page

The recipes that we just mark as completed will turn up on the History page

The user is able to filter recipes based on date and through the Search Bar

Select Date: 06/10/2024

As we did not mark any meal completed on this date, no recipe is shown

Select Date: 07/10/2024

The user is also able to find recipes through the search bar

Search bar: *Input name of first recipe*

This will show all recipes with name that matches the input in the search bar
The user is able to view the recipe details by clicking on the image

Finally, if users have made a mistake in assigning the meal plan to a particular date and course. They can navigate to the my plan page and click on the cross button of the meal plan that was a mistake.

Click on 'cross' icon on the recipes (Lunch)

That is all for the demonstration.

## 2. Good SWE Practices

Next, we will share some good SWE practices that our team has adopted.

### 2.1 Documentation code comments & Pydoc

Firstly, we ensure the system is well documented by writing clear code comments. This is crucial for maintaining code readability and fostering team collaboration. By detailing the purpose, attributes, return type of each function and class, the comments help our team better understand the code, making it easier to communicate and work together effectively.

The comments are then translated into Pydoc, allowing us to easily and quickly view documentation for Python modules, functions, classes, and methods. This documentation helps developers navigate the code more efficiently, ultimately enhancing future extensibility by making it easier to understand and build upon the existing codebase.

### 2.2 Reusability & Refactoring

Another practice is reusing functions to reduce duplication. For instance, many of our pages require a navigation bar, so we created a reusable navigation bar template. This template can be easily incorporated into other pages simply by referencing its JavaScript and CSS files, streamlining our development process. This helps with the maintainability and modularity of our system. If there's a change needed in the nav bar, you can modify the template once, and it will be reflected across all pages, saving time.

## 3. System Design

Next, we will share more about our system design.

### 3.1 System Architecture

Firstly, we have adopted a layered architecture for our system. The communication goes from the top presentation layer down to the persistent data layer. This enhances the **extensibility** of the system by allowing each layer to evolve independently without affecting the others. For instance, if we need to change the way data is stored or add a new database feature, we can do so in the persistent data layer without impacting the presentation layer or business logic.

## 3.2 Class Diagram

Our class diagram also showcases this layered architecture.

## 3.3 Design Pattern

We have also implemented Factory Pattern design which simplifies database management by centralizing query creation through a single DBFactory class. The create_db_connection method dynamically provides the right query type—like User, Recipe, Ingredient, or MealPlan—based on the input, making our code modular and scalable. Each entity only needs to reference DatabaseQueries to handle database interactions, which keeps our system clean and reduces dependencies. This approach makes it easy to add new query types in the future without disrupting the existing code. Overall, the Factory Pattern ensures efficient and organized database handling

Another design pattern we used is the **Facade Pattern** for most of our control classes. An example of this is the RecipeCtrl class. The RecipeCtrl class is responsible for fetching data from the API, storing the recipe data as a Recipe object, and then storing the ingredient data in an Ingredient object. To simplify the code and make it easier to follow, we instantiate the RecipeCtrl and IngredientCtrl classes within the RecipeCtrl class and invoke their respective methods. This approach reduces the complexity of the RecipeCtrl class, making the code more understandable and easier to maintain.

This also helps in adhering to the **Single Responsibility Principle** (SRP), as each class has a clearly defined role and the RecipeCtrl does not need to manage the low-level details of recipe or ingredient handling.

## 3.4 Design Principles

We have also integrated key design principles into our system to enhance maintainability and scalability.

First, adhering to the **Single Responsibility Principle (SRP)**, we have organised our system into different packages, each containing specific classes that focus on a single task or functionality. This ensures that each class has one clear responsibility, as seen in our previous example for RecipeCtrl, making the code easier to understand and maintain.

Second, we embrace the **Open-Closed Principle (OCP)**. Through the use of the **Factory** and **Facade** patterns, we leverage abstraction to keep the system open for extension but closed for modification. For instance, if new object types need to be created, we can extend the Factory to produce them without altering existing code.

Lastly, we apply the **Dependency Injection Principle (DIP)**. By using the Factory pattern, we introduce an interface between high-level modules and low-level modules, which promotes low coupling and enhances flexibility. This setup allows for easier testing, maintenance, and modification of components without affecting other parts of the system.

## 4.1 Traceability

For traceability, we will look into the implementation of filter recipe list use cases.

## 4.1 Use-case Description

The main flow is for the system to display a list of filters in which the user can input parameters for each filter type. The system then updates the list of recipes based on the filter parameters.

## 4.2 Class Diagram

We have a set of filter objects but we do not know beforehand which filter type the user will use during runtime, hence we utilised the factory pattern. The filter type class implements the FilterInterface Class with the abstract method, apply() and the factory creates a filter type object at runtime, when the user uses a filter type.

## 4.3 Sequence Diagram

Here is the sequence of events. For example, when the user uses the cuisine filter, the FilterFactory will create a CuisineFilter object which will be added into the list of filter objects in the FilterCtrl class. The FilterCtrl class subsequently uses the apply_filters method to call upon the apply() method of the CuisineFilter object to filter recipes only of certain cuisine types.

As mentioned this allows for extensibility, a new filter type can easily be added just by implementing the FilterInterface Class and updating the object creation method in FilterFactory.

## 4.4 Testing: appy_filters()

We conducted white box testing for the apply_filters() function. This allows us to examine the internal code and structure of software, ensuring thorough coverage of all possible execution paths.

On the right side is the basis path testing. There are 4 binary decision points and hence the cyclomatic complexity is 5. For this function, there are 5 basis paths.

These test cases mirror the 5 different paths. The first path occurs when all the decision points are false, that is when no filter is applied by the user, resulting in all recipes being displayed. The second path occurs when the user uses the cuisine filter. The system will display recipes that match the cuisine filter. The third path occurs when the user uses the cost filter. The system will display recipes that match the cost filter. The fourth path occurs when the user uses the estimated preparation time filter. The system will display recipes that match the user's desired estimated preparation time filter. The last path occurs when the user uses the meal type filter. The system will display recipes that match the meal type filter.

We have come to the end of our presentation.

Thank you!