Sandwich Generator

Requirements Specification

Document

PREPARED FOR
Associate Professor Alex Pantaleev

PREPARED BY
Joshua Meritt

1.0 Introduction

1. 1 Purpose:

The purpose of this document is to specify the design, development, implementation, and testing of a standalone application that will generate random Subway sandwich to a User. This document will specify system requirements, goals, scenarios, and solutions of the system under development.

1. 2 Glossary:

User (n): A person with an account signed up for this application. Username and password are what authenticates a user.

Browse (v): To scroll through, visually scan, or otherwise peruse a group of displayed elements.

Bread (n): The list of choices: 9-Grain Wheat, Multi-grain Flatbread, Italian, Italian Herbs & Cheese, and Flatbread.

Meat (n): The list of 12 choices: Turkey Breast, Ham, Chicken Breast, Roast beef, Tuna, Turkey Salami, Beefsteak, Bacon, Meatballs, Genoa Salami, Turkey bologna, and Shaved steak.

Vegetables (n): The list of choices: Cucumbers, Green Peppers, Lettuce, Red Onions, Spinach, and Tomatoes.

Cheese (n): The list of choices: American and Monterey Cheddar.

Sauce (n): The list of choices: Chipotle Southwest, Light or Regular Mayonnaise, Ranch, Oil, and Subway vinaigrette.

Extras (n): The list of choices: Pepperoni and Bacon.

Sandwich (n): An object that consists of: Bread, possible vegetables, cheese, sauce, and possible extras.

View (v): To see what a specific sandwich consisted of.

Options (n): The possible choices found at every Subway location.

Sandwich history (n): A list of the previous generated sandwiches for a User.

Favorite sandwiches (n): A list of a User's favorite sandwiches, chosen by the User.

2.0 Product Design

2. 1 Project Overview:

A User must login/create an account if they want access to the application so their data can be stored for future reference. The User is then able to generate a random sandwich combination from the possible choices at Subway. Additionally, a User can generate a sandwich and reroll if the User does not like the generated sandwich. If the User really enjoys the sandwich, they can add it to a favorites section and if they already closed the application, then the User can go into their history of sandwiches and add it to their favorites. The favorites section will allow Users to remove a sandwich from the category if no longer desired in that area. Once a User exits or logs out then that session is complete.

2. 2 Endpoints:

List Account usernames of all Users

Endpoint:

GET /Account/all

Description:

Returns a list of usernames for all Users in the database.

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
Username		String	Username of an Account

Responses:

200 OK

```
{ status : 200
 "Accounts:
"
 "username": Josh_Admin,
    "username": newUser,
    "username": test,
}
```

503 Service Unavailable

List Information

Endpoint:

GET /Account/Info/{name}

Description:

Returns the information about a User. The information includes: Username, Password, Size of History list, Size of Favorite list.

Note: If truly giving out the password, there would need to be authentication or keep it hidden.

Path Parameters:

Field Name	Required	Type	Description
Name	True	String	The username of the User to search for

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
Account			
Username		String	Username of an Account
Password		String	Password of an Account
History Sandwich List Size		Int	Size of history list of an Account
Favorite Sandwich List size		Int	Size of favorite list of an Account

Responses:

200 OK


```
{ status : 200
 "Account": {
        "username": Josh_Admin,
        "password (Should be a secret though)": secure_password,
        "History Sandwich list size: ": 10,
        "Favorite Sandwich list size: ": 3
    }
```


503 Service Unavailable

Get Favorites

Endpoint:

GET /Account/Favorites/{name}

Description:

Returns a list of the favorite sandwiches a User has.

Path Parameters:

Field Name	Required	Type	Description
Name	True	String	The username of the User to search for

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
Favorites			
Username		String	Username of an Account
Id		String	Id of a sandwich
Bread		String	Bread of a Sandwich
Meat		String	Meat of a Sandwich
Cheese		String	Cheese of a Sandwich
Sauce		String	Sauce of a Sandwich
Veggie		String	Veggie of a Sandwich
Extra		String	Extra of a Sandwich

Responses:

200 OK


```
{ status : 200
 "Favorites": "Username": Josh_Admin

    "Sandwich": {
        "id": 1,
        "Bread": Multi-grain Flatbread,
        "Meat": Turkey breast,
        "Cheese": American,
        "Sauce": Light Mayonnaise,
        "Veggie": Red onions,
        "Extras": Bacon,
    },
    "Sandwich": {
        "id": 2,
        "Bread": Flatbread,
        "Meat": Roast Beef,
        "Cheese": American,
        "Sauce": Oil,
        "Veggie": Spinach,
        "Extras": ,
    },
    "Sandwich": {
        "id": 957,
        "Bread": 9-Grain Wheat,
        "Meat": Chicken breast,
        "Cheese": Monterey Cheddar,
        "Sauce": Light Mayonnaise,
        "Veggie": Cucumbers,
        "Extras": Pepperoni,
    },
}
```


503 Service Unavailable

Get History

Endpoint:

GET /Account/History/{name}

Description:

Returns a list of the Sandwiches a User has generated.

Path Parameters:

Field Name	Required	Type	Description
Name	True	String	The username of the User to search for

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
History			
Username		String	Username of an Account
Id		String	Id of a sandwich
Bread		String	Bread of a Sandwich
Meat		String	Meat of a Sandwich
Cheese		String	Cheese of a Sandwich
Sauce		String	Sauce of a Sandwich
Veggie		String	Veggie of a Sandwich
Extra		String	Extra of a Sandwich

Responses:

200 OK


```
{ status : 200 "History": "Username": newUser

    "Sandwich": {
        "id": 17,
        "Bread": Multi-grain Flatbread,
        "Meat": Turkey breast,
        "Cheese": Monterey Cheddar,
        "Sauce": Oil,
        "Veggie": Cucumbers,
        "Extras": Bacon,
    },
    "Sandwich": {
        "id": 521,
        "Bread": 9-Grain Wheat,
        "Meat": Chicken breast,
        "Cheese": American,
        "Sauce": Light Mayonnaise,
        "Veggie": Tomatoes,
        "Extras": Pepperoni,
    },
    "Sandwich": {
        "id": 789,
        "Bread": Flatbread,
        "Meat": Tuna,
        "Cheese": American,
        "Sauce": Oil,
        "Veggie": Green Peppers,
        "Extras": Bacon,
    },
}
```


503 Service Unavailable

Create Account

Endpoint:

POST /Account/CreateAccount/

Description:

Creates an account for a User.

Query Parameters:

Field Name	Required	Type	Description
Name	True	String	Username of the possible new User
Password	True	String	Password of the possible new User

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
Success		String	If Account creation was success

Responses:

200 OK

```
{ status : 200
 Success : Account created with username: IAmNewUser
}
```

503 Service Unavailable

Login

Endpoint:

Post /Account/Login/

Description:

Logs in a User.

Query Parameters:

Field Name	Required	Type	Description
Name	True	String	Username of the User
Password	True	String	Password of the User

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
Success		String	If Account login was success

Responses:

200 OK

```
{ status : 200
 IAmNewUser has a successful login.
}
```

401 UnAuthorized

Note: This is for if a User is already logged in or the username/password is incorrect.

```
{ status : 401
  Login cannot be completed. Username or password may be incorrect.
}
```

503 Service Unavailable

Delete Account

Endpoint:

POST /Account/DeleteAccount/

Description:

Deletes a User's account.

Query Parameters:

Field Name	Required	Type	Description
Name	True	String	Username of the possible new User
Password	True	String	Password of the possible new User

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
Success		String	If Account deletion was success

Responses:

200 OK

```
{ status : 200
 Success : Account deleted with username: IAmNewUser
}
```

401 UnAuthorized

Note: This is for if a User is not logged in or the username/password is incorrect.

```
{ Status : 401
 User is not logged in. Why try to use someone else's account?
}
```

503 Service Unavailable

Generate Sandwich

Endpoint:

POST /Sandwich/Generate/{name}

Description:

Generates a random sandwich for a User.

Path Parameters:

Field Name	Required	Type	Description
Name	True	String	Username

Body Parameters:

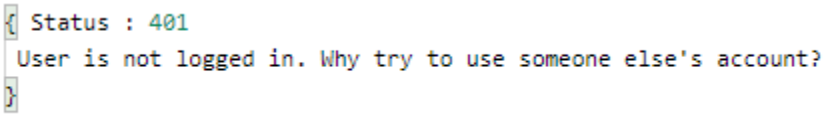
Field Name	Required	Type	Description
status	True	Int	Status of the request
Username		String	Username of an Account
Bread		String	Bread of a Sandwich
Meat		String	Meat of a Sandwich
Cheese		String	Cheese of a Sandwich
Sauce		String	Sauce of a Sandwich
Veggie		String	Veggie of a Sandwich
Extra		String	Extra of a Sandwich

Responses:

200 OK

401 UnAuthorized

Note: This is for if a User is not logged in or the username/password is incorrect.



503 Service Unavailable

Add a Favorite Sandwich to Favorites list

Endpoint:

POST /Account/AddFavorite/{name}

Description:

Adds a Sandwich to a User's favorite list.

Path Parameters:

Field Name	Required	Type	Description
Name	True	String	Username

Query Parameters:

Field Name	Required	Type	Description
Id	True	String	Id of the sandwich

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
Success		String	If Sandwich was added to favorites list

Responses:

200 OK

```
{ status : 200
 Success : Sandwich : 405 was added to Josh_Admin's Favorite list.
}
```

401 UnAuthorized

Note: This is for if a User is not logged in or the username/password is incorrect.

```
{ Status : 401
 User is not logged in. Why try to use someone else's account?
}
```

503 Service Unavailable

Remove a Favorite Sandwich to Favorites list

Endpoint:

POST /Account/DeleteFavorite/{name}

Description:

Removes a Sandwich to a User's favorite list.

Path Parameters:

Field Name	Required	Type	Description
Name	True	String	Username

Query Parameters:

Field Name	Required	Type	Description
Id	True	String	Id of the sandwich

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
Success		String	If Sandwich was removed from favorites list

Responses:

200 OK

```
{ status : 200
 Success : Sandwich : 405 was removed from Josh_Admin's Favorite list.
}
```

401 UnAuthorized

Note: This is for if a User is not logged in or the username/password is incorrect.

```
{ Status : 401
 User is not logged in. Why try to use someone else's account?
}
```

503 Service Unavailable

Logout

Endpoint:

Post /Account/Logout/

Description:

Logs out a User.

Query Parameters:

Field Name	Required	Type	Description
Name	True	String	Username of the User
Password	True	String	Password of the User

Body Parameters:

Field Name	Required	Type	Description
status	True	Int	Status of the request
Success		String	If Account logout was success

Responses:

200 OK

```
{ status : 200
 newUser has a successful logout.
}
```

401 UnAuthorized

Note: This is for if a User is already logged in or the username/password is incorrect.

```
{ status : 401
  Login cannot be completed. Username or password may be incorrect.
}
```

503 Service Unavailable

3.0 Extra

3.1 Additional Notes:

Not all the endpoints require authentication since I viewed parts of this as wanting to be more open and accessible to anyone. Anything that does not require altering a User's information is available for anyone to view, this includes viewing all account names, getting someone's account info (Password would be hidden if used by anyone other than professor Pantaleev, another professor, or myself), viewing a User's history or Favorites. Granted, I understand some people would want to hide their favorites or history so I would probably require authentication for those, but I would find it interesting to see so I kept those open. The ones that do require authentication include Logging in/out, adding a favorite sandwich, removing a favorite sandwich, or deleting an account.

3.2 Future improvements:

While the project has made a lot of progress since the start, I realize that it could still be improved in multiple ways. Even though this was created to be more of an API as compared to a website, one way it could be improved is having a more secure authentication and security. Currently, the system only makes sure that a User is logged in/out to check if their account can be altered. This means that if a User is logged in, anyone can hit the endpoint for generating a sandwich, or altering a User's favorites list. Additionally, a User is not logged out after a certain amount of time so to fix this I would think to calculate a logout time when a User logs in and store that in the database. If a User tries to access a feature that needs authentication, check the logout time and if a User does logout, the loggedIn field will be 0 so another User cannot access that endpoint. Another thing I would do is try to use a DAO library that is already implemented since I created an attempt at my own. The one I created hides all the CRUD operations that are performed but I can almost guarantee that it would be more efficient if I used an already created DAO such as jdbi. The reason I did not is because I spent a large amount of time trying to set it up using different dependencies and following multiple tutorials to no success, so I decided to make my own. Overall, the project completed and went further then I originally planned for creating my own backend API.