McDonalds Meal Generator

Team 3

Brandon Polite, Tiana, Mike, Enrich

**Software Requirements Specification**

**Document**

**Version: 0** **Date: (10/08/2020)**

**Table of Contents**

**1.** [**Introduction**](#7rtnzraaroe)

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, and Abbreviations

1.4 Overview

**2.** [**The Overall Description**](#816sot3e8sg4)

2.1 Product Perspective

2.1.1 System Interfaces

2.1.2 Interfaces

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

2.1.5 Communications Interfaces

2.1.6 Memory Constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

2.2 Product Functions

2.3 User Characteristics

2.4 Constraints

2.5 Assumptions and Dependencies

2.6 Apportioning of Requirements

**3.** [**Problem Statements**](#4tg63ko6osjh) **-**

3.1 Problem Statement One

3.1.1 Problem

3.1.2 Solution

3.2 Problem Statement Two

3.2.1 Problem

3.2.2 Solution

3.3 Problem Statement Three

3.3.1 Problem

3.3.2 Solution

3.4 Problem Statement Four

3.4.1 Problem

3.4.2 Solution 1

3.4.2 Solution 2

**4.** [**Input Requirements**](#r3tpckfcmyb) **-**

4.1 System Inputs

4.1.1 Meal Information

4.1.2 Location Information

4.2 User Inputs

4.2.1 Create Profile

4.2.2 Login Information

4.2.3 Meal Parameters

4.2.4 Favorite an item

4.2.5 Dislike an item

4.2.6 Location

4.2.7 Navigation

**5.** [**Process Requirements**](#9dmg78405xc) **-**

5.1 Data Validation

5.1.1 Errors

5.1.2 Default Values

5.2 Database

5.3 Randomization

5.3.1 User Interaction

5.4 Parsing

5.4.1 Formatting

**6.** [**Output Requirements**](#hfwb6o4kc8ms) **-**

6.1 User Database File Output

6.1.1 users.txt File Output

6.1.2 users.txt Errors

6.2 Username/Password Validation

6.3 User Favorites/Dislike Systems

6.3.1 Favorite/Dislike Edits

6.3.2 Favorite/Dislike Display

6.3.3 Favorite/Dislike Errors

6.4 Meal Information Output

6.5 Location Output

# **1.** [**Introduction**](#ev14p97rel0z)

## **1.1 Purpose**

The purpose of this SRS is to describe the requirements for the program and its intended audience. The intended audience is people who wish to eat at McDonald’s, locate the nearest McDonald’s or find the nutritional information of a meal.

## **1.2 Scope**

1. The software to be produced is the McDonald’s random meal generator.
2. This software will create a random McDonald’s meal, with predetermined user parameters. This software will also locate the nearest McDonald’s to the user, and also provide caloric and nutritional information of the meal.
3. The goal of this software is to make it easy to find out what McDonald’s meal you want to eat, locate the nearest McDonald’s, and also make the nutritional and dietary information of said meals available and easy to access.

## **1.3 Definitions, Acronyms, and Abbreviations.**

**1.3.1 Definitions**

n/a

**1.3.2 Acronyms**

n/a

**1.3.3 Abbreviations**

The following are abbreviations used in this SRS

1. MD, McDonald’s
2. RMG, Random meal generator or random meal generation
3. GUI, Graphical User Interface

## **1.4 References**

## n/a

## **1.5 Overview**

This SRS contains information about the McDoanald’s Meal Generator, how it is to be used, what data is to be collected and used, how it is to be developed, input requirements, potential problem statement along with example solutions, process requirements, and output requirements.

The SRS is organized in such a way that it makes finding specific information easy. Each section is labeled with what it contains, followed by subsections helighting different information through the section.

The [table of contents](#ev14p97rel0z) can be used to locate any information you may need to use, and it is hyperlinked to the section, so you may click whichever section you wish to see, and click the sections title to return back to the [table of contents](#ev14p97rel0z).

# 2. [The Overall Description](#ev14p97rel0z)

This program is designed to generate a meal, based on different parameters and likes/dislikes set by the user, and then provide nutrition information on the generated meal. This program should also be used to locate the nearest McDonalds to the user.

## 2.1 Product Perspective

The MD random meal generation program will be a self contained executable that does not rely on outside dependencies or systems. Although the system could extend McDonalds web systems, for the sake of simplicity and ease of processing, defining local classes and systems will be much more efficient.

### 2.1.1 System Interfaces

As this is a self contained program,It will not need to interface with external systems. However, it will need to interface with local data in the form of text documents that keep track of program information and dependencies. Thus, the System interfaces include

1. User Information Interface
2. Meal Information Interface
3. Location Information Interface

These interfaces serve to provide the system relevant information on user login and preference data, as well as meal information and McDonald's locations information.

### 

### 2.1.2 Interfaces

The system must provide interfaces to the user that allows them to:

1. Create an account or login to an existing account
2. Specify meal parameters and requirements
3. View randomly generated meals and their ingredient/nutritional information
4. Favorite or dislike a generated meal
5. View the nearest MD location to them (location scope pertains only to Delaware locations)
6. Navigate between pages of the program.

These interfaces will be presented to the user via a GUI

**2.1.3 Hardware Interfaces**

This program does not have any hardware interface requirements.

### 

### 2.1.4 Software Interfaces

The use of external software interfaces will not be necessary for the scope of this system. This software is self contained.

### 2.1.5 Communications Interfaces

Self contained system will not need to communicate over networks.

### 2.1.6 Memory Constraints

n/a

### 2.1.7 Operations

The system will have a number of operations that it must complete throughout its runtime such as:

(1) All users will have access to the same utilities and functions. Users will only be able to use the program after providing login information.

(2) The user will have to interface with the program to complete interactive operations described in section 2.1.2.

(3) At runtime, the system will gather data unattended and without user specification from the system interfaces described in section 2.1.1

(4) When closing the program, the system must back up/ export user information to a text document to ensure that user login information and preferences are kept.

### 2.1.8 Site Adaptation Requirements

As described in the previous section, upon runtime, the system must initialize users, meals, and locations from text documents stored on the host machine. This protocol will be universally applicable regardless of the hardware the system is running on.

## 2.2 Product Functions

The MD system can fulfil a variety of functions that will allow the users to perform the following:

1. Provide the user with an interface that allows them to design a meal from a variety of food items.
2. Allow users to favorite or dislike certain meals and retain the preferences of specific users.
3. Generate a meal based off of the preferences of the user.
4. Locate the nearest McDonalds and provide the user with directions.

## 2.3 User Characteristics

This system is designed to be utilized by any user. However, the user should have a general understanding of how programs work but doesn’t need any further technical expertise.

The general user may be someone who primarily prefers McDonald’s over other restaurant and food options as the system can only provide food items listed on McDonald’s main website.

Since our users may primarily prefer McDonalds over other options, our design should resemble that of McDonalds. Of course, our system will not completely copy the MD design, however, by incorporating a recognizable layout, users will be able to easily interact with the system.

## 2.4 Constraints

Our system doesn’t have many constraints that would affect its development. However, there must be limits put into place such as the number of meals that can be created and the number of users’ data the system can retain. This is because an excessive amount of data may slow down and cause errors in the system’s performance.

Besides that, other constraints may come straight from the user, such as the user’s location, dietary limits, specific likes and dislikes, and the saving of specific preferences. Section 3 goes into more detail about these problems but other than that, there are not many constraints that would affect the system.

## 2.5 Assumptions and Dependencies

During the development of the system, certain assumptions are made on how the system will operate and perform in different environments. Several assumptions include the following:

1. The operating system: It’s assumed that the users will be running the MD system on a specific operating system that can support the MD system.
2. Viewable location information: As part of locating the nearest McDonalds to the user, the system must gain the location information of the user. Without that data, the system can not perform that function.
3. Time constraints: Another feature of the MD system is that the user can create a meal consisting of breakfast items depending on the time of day. Once it reaches a certain time, the breakfast option will be locked. As a result, the system needs to have access to the user’s device time and will assume that time is correct.

## 2.6 Apportioning of Requirements.

As per the time restraints of development, some features and ideas that have been omitted from the initial release, but may be added in future iterations include:

1. An account management interface where the user can change their name and password if desired
2. An extensive dietary restriction system where the user could filter out random meals based on their ingredients or allergens.
3. A navigation system that could give a user directions to the nearest McDonalds, not just tell them the address.

**3.** [**Problem Statements**](#ev14p97rel0z)

This section will outline potential and prevalent problems with the software, and provide potential solutions to said outlined problems.

**3.1 Users location at a given time**

**3.1.1 Problem**

The problem arises of locating the user whenever they access the software. If a user does not update their location before using the nearest MD locator system, they will be sent to a MD that could potentially be further away then would be wanted.

**3.1.2 Solution**

When the user chooses the, “Find restaurant near me” option, they must be prompted to enter their current location before viewing results.

**3.2 Users meal/dietary restrictions**

**3.2.1 Problem**

The problem of the random meal generator creating meals is that some users may have dietary restrictions to which they need to be sure that a certain ingredient or cooking method is not used in any way.

**3.2.2 Solution**

The way to solve this is to detect when an ingredient restriction arises by checking the user inputs for either a dietary restriction or allergen, or by also displaying all ingredients in the meal before the user purchases the meal.

**3.3 Users meal likes and dislikes**

**3.3.1 Problem**

A potential problem with the likes and dislikes system could be that the system ignores the meal/dietary restriction system and that it generates a random meal without the users predetermined dietary restrictions.

**3.3.2 Solution**

The way to solve this is to detect when an ingredient restriction arises by checking the user inputs for either a dietary restriction or allergen, and apply those parameters before the user likes/dislikes system parameters are used to create a random meal.

**3.4 Dietary restrictions without a user login**

**3.4.1 Problem**

A potential problem is that some users may not wish to create and account, but they do have dietary restrictions.

**3.4.2 Solution**

A solution to this is to have the software show the user the ingredients in a meal before they purchase it. This will allow the user to see all of the ingredients in a meal, and judge for themselves if they can eat it without creating a log in.

**3.4.3 Solution 2**

Another solution to this problem is to make the program entirely log in based, and to not allow the use of the software without a user login. This will prevent the misuse of the app and will not allow users to order food without first checking if they will have adverse reactions to any ingredients in the meal. This is the easiest to implement and the most effective method of booth ensuring users do not order food that they cannot eat due to their dietary restrictions and saving memory.

**4.** [**Input Requirements**](#ev14p97rel0z)

This section will outline all of the expected and necessary inputs that will be required for the functionality of the MD random meal generation system.

**4.1 System Inputs**

The system will provide a set of inputs to itself upon runtime such as meal information as well as location information. The system will also be provided a list of users login information at runtime to allow existing users to login.

**4.1.1 Meal Information**

Meal information will be stored in a text document. Said text document names meal\_information.txt, will contain names of all MD menu items, as well as their corresponding nutritional information. The system will parse the text document at start up to initialize an array of meal objects. This will allow for new items to be easily added or removed by editing the text document rather than having to edit the source code directly.

As mentioned above, the system will parse meal\_information.txt to create an object array of Meals. The “Meal” Object should be defined with the following parameters

1. String mealName = The name of the meal.
2. String mealPrice = The price of the given meal in dollars.
3. String mealIngredients = The ingredients within the meal.
4. String mealCalories = The number of calories in the meal.
5. String mealType = Specifies if meal is breakfast or dinner.

As per the requirements of creating a Meal object, meal\_information.txt must be formatted in a consistent manner to allow the distinction between meals and parameters, formatting shown below.

----Empty Space----

Meal name

Meal price

Meal ingredients (delimited by commas: lettuce, onion, salt, etc)

Meal calories

Meal type (Breakfast or Dinner)

----Empty space----

Repeat for next meal item

The system must be able to catch and throw an error message if a food item within meal\_information.txt is formatted incorrectly, or if, for any other reason, information was unable to be parsed correctly.

There must also be a folder of images of all of the meals. The names of the images must be identical to the Meal name they are associated with so they can be called upon easily. Each meal object will contain an ImageIcon object which will hold the image of the meal to be displayed when generated.

**4.1.2 Location Information**

Location information for all MD Locations in Delaware will be stored in a text document, named, “location\_information.txt”. Upon Runtime, the system will parse the document to create Location objects.

Each Location Object will consist of the following parameters-

1. String city = The town/city the location is in
2. String address = The specific address of the location
3. String zipCode = the zip code the location resides in

As per the requirements of creating a Location object, location\_information.txt must be formatted in a consistent manner to allow the distinction between locations and their parameters, formatting shown below.

----Empty Space----

Location City

Location Address

Location Zip Code

----Empty Space----

Repeat for next location item

The system must be able to catch and throw an error message if a location within location\_information.txt is formatted incorrectly, or if, for any other reason, information was unable to be parsed correctly.

**4.1.3 Users**

At runtime, the system will be provided a text document titled users.txt which will contain login information of existing users and their favorite and dislike meals. The system must parse the information contained in users.txt to create an array of user objects.

Each user object will consist of the following parameters

1. String Username = Users username
2. String Password = Users password
3. ArrayList<String> Favorites = A list of the users favorited items
4. ArrayList<String> Dislikes = A list of the users disliked items

As per the requirements of the user object, the users.txt document must be formatted in a consistent manner to allow for the distinction between user information. Formatting shown below.

----Empty Space----

Username

ActualUsername

Password

ActualPassword

Favorites

FavoriteMeal1, FavoriteMeal2, FavoriteMeal3, etc

Dislikes

Dislike1, Dislike2, Dislike 3, etc

----Empty Space----

Repeat for next user

The system must be able to catch and throw an error message if a user within users.txt is formatted incorrectly, or if, for any other reason, information was unable to be parsed correctly.

**4.2 User Inputs**

Throughout the usage of the program a user will be required to offer a wide variety of inputs to the system in order to create a profile, log into an existing profile, disclose current location, and set specified parameters to generate a meal. Simple user inputs will also be required to like or dislike a meal, as well as to navigate the program.

**4.2.1 Create Profile**

When first entering the program, the user will be prompted to login, or create an account. If the user chooses to create a profile they will be prompted with text boxes to enter a username and password, as well as a verification for their password.

The username must be less than or equal to 16 characters, and must not be the same as an existing users username. The system must be able to alert the user if their chosen username is already taken (see section 6), and must allow them to enter a new one.

The users password must be at least 8 characters long. The user must enter the exact same password both in the password text field as well as the password verification field. If the two fields do not match, the system must alert the user that the passwords do not match (see section 6). And then must allow the user to correct the password fields.

The system must be able to export user login information to users.txt. See section 6- Output Requirements

**4.2.2 Login Information**

When first entering the program, the user will be prompted to login, or create an account. If the user chooses to login, they will be prompted by two text boxes to enter a username and password.

Username input must be limited to 16 characters. The system must be able to compare the entered username to the array of existing user accounts to check if it is a valid username. If it is not, the system must display an error message (see section 6).

Password input must be at least 8 characters long. System must be able to check if the password matches the user object of the same username. If the password is wrong the system must provide an error message(see section 6).

**4.2.3 Location**

If the user selects the system feature “Store Locator” the system must provide a drop down menu of Delaware's 55 towns and cities. The towns must be organized in alphabetical order. The user will scroll through the list of options and select the town they are currently in. Limiting user input in this way will prevent errors in input, and allow for less input validation.

The system will use the users selection to return the nearest MD location to their town, or a list of locations within their selected town (see section 6).

**4.2.4 Meal Parameters**

After logging in, the user will be prompted to enter a set of parameters for the system to follow in its generation of random meals. Meal parameters include, meal type, number of meals to generate, generate from favorites only.

The “Meal Type” input will be made from a drop down menu, which will consist of three options, breakfast, dinner, or All

The “Number of Meals” input will be made within a text box. Such input must be verified so that it is greater than 0. The input within the text box must be an integer.

The “Generate from favorites only” input must be represented as a check box. The option must only be available in the event that the user does, in fact, have favorites associated with their account. Checking the box will result in the randomly generated meal being generated within the scope of the users favorited meal items.

**4.2.5 Favorite an Item**

Once a user has generated food items, an option must be given to the user to favorite the item. This input will be represented as a check box.

If the “Favorite” checkbox is activated, the system must append the favorite food item to the user object so that the information will be saved.

**4.2.6 Dislike an Item**

Once a user has generated food items, an option must be given to the user to dislike the item. This input will be represented as a check box.

If the “Dislike” checkbox is activated, the system must append the disliked food item to the user object so that the information will be saved. disliked food items will be omitted from the generation of random food items.

**4.2.7 Navigation**

Throughout the runtime of the program, the user interface must provide a number of buttons and tabs for program navigation. Such buttons would include “previous” and “next”.

The “Next” button, when clicked, will allow the user to proceed to the next page in the program. If other user inputs are required before continuation, such as login, create profile, or specify meal parameters, the next button must be unable to be used or display an error message to alert the user to complete inputs. If there is no page proceeding the one the user is on, the next button will be unable to be used as well.

The “Previous” button, when clicked will allow the user to navigate to the last page they were on. If the user is on the login page, the previous button will not be displayed as their is no previous page to navigate to.

**5.** [**Process Requirements**](#ev14p97rel0z)

This section will list the specific requirements the MD menu system will be able to execute and a description of practices to help avoid faults through each requirements’ development.

**5.1 Data Validation**

As the system operates, it should have the capacity to accurately handle any errors that may occur. This includes mistakes on behalf of a user’s input or any coding or executing mistakes.

**5.1.1 Errors**

As errors can range from a wide variety of examples, good coding practices such as the use of exceptions, (E.I. try and catch), and utilizing the debugging tool with breakpoints and error messages will help maintain an organized workflow through each step of the developmental process.

**5.1.2 Default Values**

Most of the information to be set as a default value will most likely come from the data parsed from within the meal\_information.txt document. This may include the various ingredients found within the meal, the number of calories a meal may contain, and any additional details about the meal such as it being a breakfast or lunch/dinner meal.

These values must remain constant once parsed and will attribute to the user’s overall description of the meal they may create.

**5.2 Database**

After a user creates a meal, they have the option of favoriting or disliking said meal. Along with creating a randomized meal based on the user’s preferences, the system will save the data retained to any specific user. It will do so by adding items to a list implementation within the user class.

**5.3 Randomization**

As previously stated, the system will generate a randomized meal based on the preferences imputed by the user. This process must appeal to users’ likes and exclude any of the users’ dislikes.

**5.3.1 User Interaction**

When the user interacts with the system, they may utilize the favorite or dislike system options when creating different meals. From these parameters, the system will generate a randomized meal per the user’s tastes.

If a user does not have any meals favorited or disliked, the MC system is free to generate a meal without any sort of constraints.

**5.4 Parsing**

To gain the information for the meal items to be featured in the MC system, the text document meal\_information.txt will be parsed.

**5.4.1 Formating**

Per normal parsing data standards, the text document must be formatted correctly and straightforward enough for the system to read and evaluate.

Errors messages will help notify us whenever the data has not been parsed correctly, however, the parsing portion of the system’s development will be a very critical part of ensuring all the menu items and information are displayed properly.

**6.** [**Output Requirements**](#ev14p97rel0z)

This section will outline the necessary and expected outputs that the MD random meal generating system will perform.

**6.1 User Database File Output**

The system must keep track of user log ins and preferences. This section will outline the output file containing such information.

**6.1.1 users.txt File Output**

After the user has submitted their account information as per sections 4.2.1-4.2.2, the system shall update the users.txt file with the new account information, so that the system can access this new data now and in the future.

**6.1.2 users.txt Errors**

In the event that there is no users.txt file, the system shall create a file in a directory that is within the systems scope.

In the event that the users.txt file is inaccessible(due to formatting errors or other issues), and it is not possible to create or write to the users.txt file, the system shall display an error message specifying the exception that took place, the date and time at which the exception occurred, and write this data to a file entitled crash\_dump.txt.

**6.2 Username/Password Validation**

In the event that a user needs to log in, (as per section 4.2.2) and they enter in a username/password combination, the system will check against the currently accessed users.txt file and search for the user with the user name specified, and if that username is correct, and that the password that matches that username. If no username/password combination is found that matches the current users.txt file’s data, the system will display an error message telling the user that this username/password is incorrect.

**6.3 User Favorites/Dislike Systems**

A user must be able to like and dislike food items, this section will outline the outputs in the event that a user utilizes this function

**6.3.1 Favorite/Dislike Edits**

In the event that a user adds a new item to their favorites list or dislikes list (as per section 4.2.5 and 4.2.6), the system will append this new item to the user file, and display a message telling the user the item has been added to its respective list.

**6.3.2 Favorite/Dislike Display**

In the event that a user needs to view their favorites and/or dislikes the system will access the data from the users.txt file under the respective user, and cleanly display the data on screen.

**6.3.3 Favorite/Dislike Errors**

**I**n the event that there is a formatting error, or the data is corrupted in the favorites/dislikes system, the system will display an error message, telling the user that the “data cannot be accessed at this time”, and push this error to the crash\_dump.txt file, with its timestamp and exception.

**6.4 Meal Information Output**

In the event that a user generates a meal the system will output the meal and the meal’s information to a fullscreen display. This display includes the meal calorie and ingredient information found in themeal’s object data.

**6.4.2 Location Output**

In the event that a user needs to find the nearest McD’s location to them, the system will access the locations document and search for the closest address to the user’s given address.