M.O.M. Generator

Maddy Andersen Stella Oh

May 21, 2024

Video Presentation

Introduction

While Williams has multiple dining halls, sometimes it can be more convenient to order from one of the three mobile ordering options instead: Lee Snack Bar, 82 Grill, or Fresh n Go. But with this added convenience comes a daunting question: what exactly should you order? Each of the three locations has an extensive menu, with various categories containing different items that can be further customized. Because there are so many choices, ordering can be overwhelming. As a result, many people — ourselves included — stick to ordering the same meals again and again. Even worse, some people don't mobile order at all and instead go to one of the dining halls, feeling that a mediocre meal is better than having to make the decision about what to mobile order.

In an effort to help out indecisive individuals and those experiencing decision fatigue, our language generates a possible mobile order, following given specifications from the user about what day of the week it is, as well as about what kind of meal they are looking for. Designed for students who may or may not be familiar with mobile ordering, our language will help users mix up their order and find new options. It will also hopefully re-inspire students to mobile order their meals if they are unhappy with the dining hall options. Because our language is (ideally) easy to use, using our language to generate a meal will be less burdensome — and more fun! — for a user than determining what to order on their own.

Design Principles

This language is designed with simplicity and ease in mind. It is suitable for those who are already familiar with the types of mobile ordering items available from each location, as well as those who do not have much experience mobile ordering. Because this language is intended to be an alternative to the overwhelming task of deciding what to mobile order, it aims to be as straight-forward and stress-free as possible.

Examples

You'll find the following example programs in *code/examples*. They may be run with

```
dotnet run ../examples/<example#.txt>
```

at the project level. The output (the M.O.M. Generator's suggestions) are printed to the terminal.

1. **example1.txt**

   In this example, the user has provided the day, meal, location, category, and item. The M.O.M. Generator simply outputs a complement about this order (as there is nothing to suggest).

   ```
   monday, breakfast, lee, breakfast sandwiches, bagel supreme
   ```

   The expected output is:

   The Bagel Supreme from the Breakfast Sandwiches category for Breakfast at Lee on Monday is a great choice!

2. **example2.txt**

   This example explores ”any.” The first order in the request asks for any item, the second asks for any category and any item, and the third asks for any location, any category, and any item. For the first order, the M.O.M. Generator suggests a random item from the given category from the given location for the meal on the day. For the second order, the M.O.M. Generator suggests a random item from a random category from the given location for the meal on the day. For the third order, the M.O.M. Generator suggests a random item from a random category from a random location for the meal on the day.

   ```
   tuesday, lunch, fresh n go, build your own, any
   wednesday, dinner, 82 grill, any, any
   thursday, lunch, any, any, any
   ```

   A possible output is:

   For Lunch at Fresh n Go on Tuesday from the Build Your Own category, we recommend the Salad.
   For Dinner at 82 Grill on Wednesday, we recommend the Pasta Bowl from the Create Your Own category.
   For Lunch on Thursday, we recommend the Blt Club from the Hot Sandwiches category at Lee.

3. **example3.txt**

   This example shows the optional gluten-free tag. The first order includes this tag, while the second does not. The M.O.M. Generator will suggest a gluten-free item for the first order and a non-gluten-free item for the second.

   ```
   friday, mid day, any, any, any, gluten-free
   friday, mid day, any, any, any
   ```

   A possible output is:

   For Mid Day on Friday, we recommend the Grilled Chicken Salad from the Gluten-Free category at Lee.
   For Mid Day on Friday, we recommend the Sausage Egg & Cheese Biscuit from the Breakfast Sandwiches category at Lee.

4. **example4.txt**

   This last example demonstrates the error messages the user can receive if their order is invalid. For the first order, none of the mobile ordering locations are open on the weekends for lunch. For the second, 82 Grill is not open for breakfast. For the third, breakfast sandwiches is a category for dinner at Lee, but not for dinner at 82 Grill. For the fourth, the bagel supreme is in the breakfast sandwiches category, not the breakfast entrees category. And finally, for the fifth, the create your own category for late night at 82 Grill does not contain any gluten-free items (instead, the gluten-free category does).

   ```
   saturday, lunch, any, any, any
   monday, breakfast, 82 grill, any, any
   thursday, dinner, 82 grill, breakfast sandwiches, any
   tuesday, breakfast, lee, breakfast entrees, bagel supreme
   wednesday, late night, 82 grill, create your own, any, gluten-free
   ```

   The expected output is:

   No locations are open for given day and meal combination.
   Given location is not open for given day or meal.
   Given category does not exist for given meal for given location.
   Given item is not in given category.
   Given category does not contain gluten-free items.

Language Concepts

The user does not need to know much to successfully use our language. To be able to use it most easily, however, the user should have some familiarity with the three mobile ordering options on campus, as well as a general sense of their menus. To submit their request (which consists of one or more orders), the user will have to write their order(s) in a text-file, which they provide as input to the language. The user must know the components that go into an order, which are the day, the meal, the location, the category, and the item. The user will have to know the format of each order (<day>,<meal>,<location>,<category>,<item>) and the fact that each order must be on a new line. The user should also be aware of the optional gluten-free tag, especially if they interested in using it. To assist the user and to make our language as easy-to-use as possible, we provide the user with instructions when they run our language (dotnet run). These instructions have guidance on the syntax, as well as all of the possible values for each of the parts of the order.

Formal Syntax

The complete BNF for programs written in our language. Any place there is white space, one or more white space characters are allowed.

```
<request>  ::= <order> | <order> '\n' <request>
<order>    ::= <day>, <meal>, <location>, <category>, <item>, <isGlutenFree>
<day>      ::= Monday | Tuesday | Wednesday | Thursday | Friday |
               Saturday | Sunday
<meal>     ::= Breakfast | Lunch | MidDay | Dinner | LateNight
<location> ::= Lee | FnG | Grill | AnyLoc
<category> ::= "breakfast entrees" | "breakfast sandwiches" | "burgers" |
                "hot sandwiches" | "breakfast sandwiches" | "salads" |
                "parfait" | "specials" | "build your own" | "protein rich" |
                "gf" | "create your own" | "wings" | "any"
<item>     ::== "brick oven pizza" | "pasta bowl" | "mac & cheese bowl" |
                "nachos" | "toasted sandwich" | "pickle pizza" | "asian
                style sesame wings" | "garlic parmesan wings" | "buffalo
                hot wings" | "plain wings" | "deli sandwich" | "salad" |
                "grain bowl" | "kosher sandwich" | "grain bowl: PR" |
                "salad: PR" | "rise & shine" | "create your own omelet" |
                "yogurt & fresh berry parfait" | "bagel supreme" |
                "mcwilliams" | "sausage egg & cheese biscuit" | "burger" |
                "garden salad" | "grilled chicken salad" | "crispy chicken
                salad" | "grilled chicken breast" | "grilled cheese" |
                "tuna melt" | "fried chicken" | "blt club" | "chicken snack
                wrap" | "deluxe disco fries" | "spinach caprese grilled
                cheese" | "any"
<isGlutenFree> ::= True | False
```

Semantics

Syntax	Abstract Syntax	Prec./Assoc.	Meaning
<request>	List	n/a	*request* is a combining form and is a list of one or more *order*s. This allows the user to have multiple *order*s within one request.
<order>	Record	n/a	*order* is a combining form. It has fives parts: the day (of type *day*), the meal (of type *meal*), the location (of type *location*), the category (a string), and the item (another string). This allows us to store all of these pieces of information together.
<day>	Discriminated Union	n/a	*day* is a primitive that represents all of the days the user can order from (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday). To prettyprint the day to the terminal, day is turned into a string.
<meal>	Discriminated Union	n/a	*meal* is a primitive that represents a specific meal of the day (Breakfast, Lunch, Midday, Dinner, Late Night). To prettyprint a meal to the terminal, the given meal is turned into a string.
<location>	Discriminated Union	n/a	*location* is a primitive that represents all of the locations the user can order from: Lee, Fresh n Go, 82 Grill, or any location. When the location is any, the M.O.M. Generators randomly selects one of the three locations.
<category>	String	n/a	*category* represents the category from a location. Each location has a .txt file of all the different categories they have for different mealtimes.
<item>	String	n/a	*item* represents the item within a category of a location. Each location has a .txt file of all items falling under each category for different mealtimes.
<isGlutenFree>	Discriminated Union	n/a	*isGlutenFree* is a primitive that represents whether or not the order is gluten-free (True, False). If *isGlutenFree* is true, then the order generated will be gluten-free. If *isGlutenFree* is false, then the order generated will be not gluten-free. By default, *isGlutenFree* is false. *isGlutenFree* is never printed directly to the terminal.

Remaining Work

While we are happy with the current implementation of the M.O.M. Generator, there are a couple of additional features we hope to implement in the future. The first is ingredient customization. Currently, the M.O.M. Generator makes suggestions up to the item level, but if you've ever mobile ordered, you know there is the option to customize the ingredients within a given item. We'd like the M.O.M. Generator to not only be able to suggest items, but also suggest ingredients. Included in this, users would be able to request that certain ingredients be included or left out from their order. Additionally, we would love to create a webpage or some other type of user interface for the M.O.M. Generator

to boost the user experience. Finally, integrating our language into WSO would be a fantastic way of extending the reach of the M.O.M. Generator, allowing it to become a helpful language for the Williams community.