1. **Personal information**

   Lunch List Application; Daniel Stafford, 664452, Data Science Bachelor's, 1st year, 24 April 2019.

2. **General description**

   A Scala application that gathers the weekly menus of student restaurants in Otaniemi and displays them in a centralized manner (cf. Kanttiinit / ruokalistat.net). The application contains two restaurants from both Fazer and Sodexo. Additional features:

   - Filter menu items based on allergens.
   - Support for choice of favourite restaurants.
   - Inform user about menu items that contain a dish or a component (like fish) that the user has reported to like (show this GUI in some reasonable way).

   As I used a text-based user interface and not a graphical user interface, I met the basic easy requirements. Instead of using hard-coded JSON or XML, I used real-time XML data from the internet, so I went a bit beyond the easy requirements.

3. **User interface**

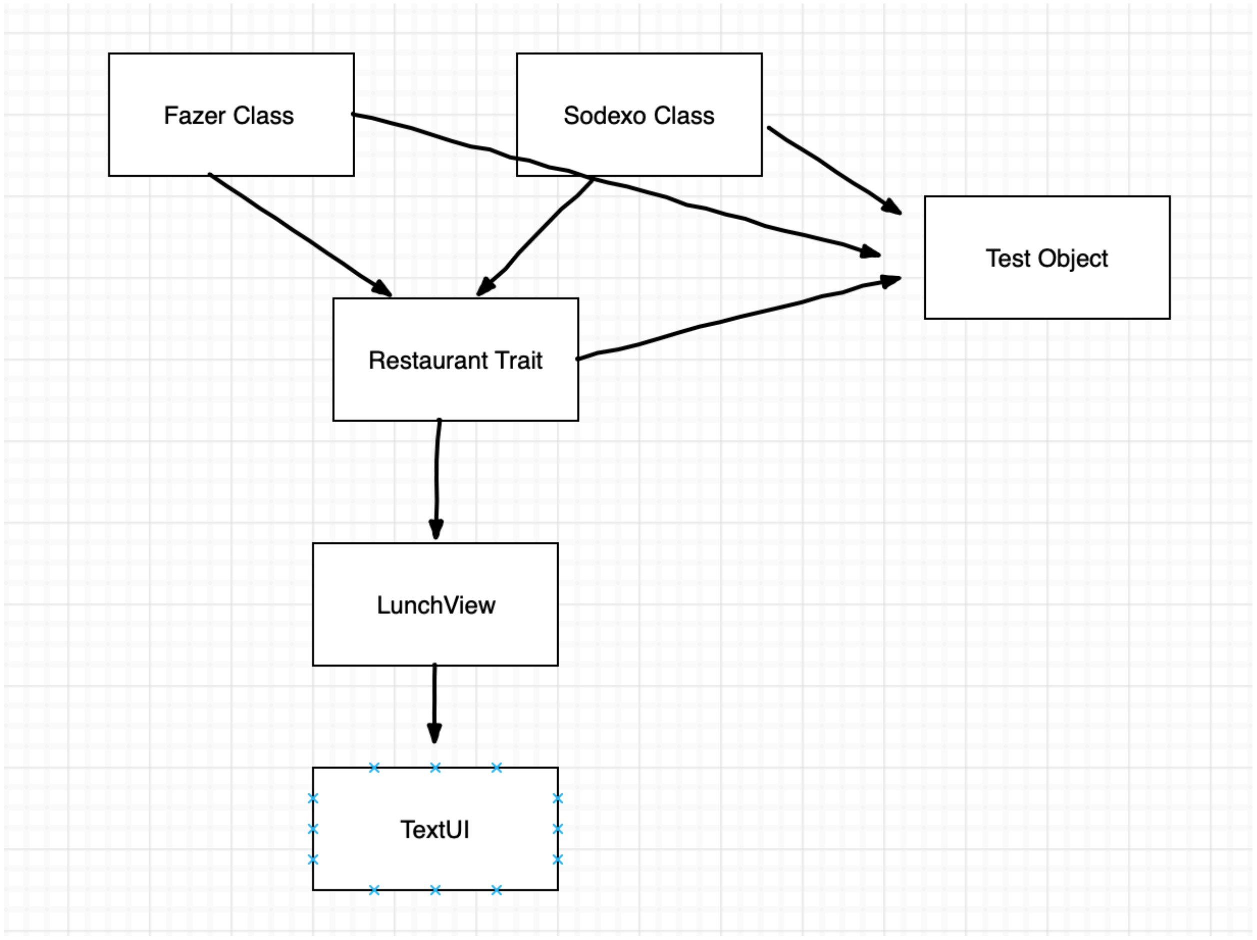
   The program is a simple text based UI. Directions on use:

   1. Import the lunch application zip file into Eclipse.
   2. You will see errors regarding the external libraries. Proceed to the next step.
   3. Add the following jar files to the build path. You can find them in the zip's JAR folder or you can download them yourself at the following links (choose the .jar file)
      a) JSoup: https://mvnrepository.com/artifact/org.jsoup/jsoup/1.11.3
      b) Scala XML https://mvnrepository.com/artifact/org.scala-lang.modules/scala-xml_2.12/1.2.0
   4. Once you've added the external JARs, delete the old duplicates(JSoup and Scala XML) with the red error marks next to them.
   5. Click apply and the errors should be gone.
   6. Ensure you are connected to the internet, as this project gathers real-time data from the internet.
   7. Within the newly imported project go to Lunch: src: lunch.ui. Right click on 'TextUI.scala' and choose 'Run as Scala Application'
   8. A console box should appear and after a few seconds, text should appear explaining how the program works.
   9. The user is asked three questions about favorite restaurants, allergies, and favorite foods.
   10. User input is by keyboard only. The user can only write nouns from a displayed list. No other commands are allowed.
   11. If the user doesn't follow directions properly (e.g. commits typos, omits commas, etc.) default values, as the text UI dialogue explains, are used.

12. Upon submitting each answer, the program will tell the user how their answer has been recorded.
13. After answering the three questions, a summary of the user's preferences are displayed.
14. After hitting enter, the current week's menu is printed, which includes only favorite restaurants, removes dishes with the specified allergy(s), and indicates whether the user's favorite dishes are served that week and lists any of those favorited dishes.
15. Finally, the program concludes and asks the user to run it again if different results are desired.

4. **Program structure**

The program was split into six different parts:



1. Restaurant trait containing all methods and variables common in Sodexo and Fazer classes.
   a) *Parsed, name, nameAndDate, getDay* are common to both Sodexo and Fazer but have different implementations.
   b) The rest of the methods have identical implementations for both Sodexo and Fazer.

   c) *getRawWeek* method return the entire week's menu without any filtering.
   d) *filterAllergies* methods takes in the buffer of allergies and, then filters out any dishes containing those allergies from the raw week.
   e) *filterFavs* methods acts similar to *filterAllergies,* takes in a buffer of favorite foods and filters out any dishes containing those foods from the raw menu. However, unlike *filterAllergies,* those filtered items are saved and printed at the top of the menu.
   f) *finalMenu* combines *filterAllergies* and *filterFavs* to return the final, filtered menu in buffer form.
2. Sodexo class extends the Restaurant trait and parses Sodexo XML data.
   a) As a parameter, the *Sodexo* class takes in the URL containing the desired restaurant's XML data.
   b) *Name* value (restaurant name), and *nameAndDate* value (restaurant name and the current date range) return strings, which will be used when displaying the menu.
   c) *getDay* method allows the program to parses the XML data to get dishes for each day of the week in buffer form, which will be used in the *getRawWeek* method in the Restaurant trait.
3. Fazer class extends the Restaurant trait and parses Fazer XML data.
   a) Contains the exact same parameter, variables and methods as the *Sodexo* class but with different implementation as Fazer XML data is formatted differently than the Sodexo XML data.
4. LunchView class containing most text UI logic.
   a) Creates all the available restaurants values via the *Fazer* and *Sodexo* classes
   b) Contains preset buffers containing available restaurants, allergies, and favorite dishes that the user can choose from.
   c) Contains empty preference buffers that user will populate (favorite restaurants, allergies, favorite foods).
   d) Contains test user input for debugging
   e) Contains various methods for converting user input into buffers that, in turn, acts as parameters for filtering methods (favorite restaurant, allergies, favorite foods)
   f) Contains stock text dialogue for the text UI.
5. Test object for running tests on individual restaurant objects.
   a) Object for testing Sodexo and Fazer classes separately and ensuring all restaurant trait methods are working.
6. TextUI object for running the text UI .
   a) Uses the *lunchView* classs for most methods and variables, as I didn't want the *TextUI* object to be cluttered with code.
   b) Uses *println* and *readLine* to print text for the user as well as record user input.

5. **Algorithms**

No advanced algorithms were used as this program didn't require anything complex. Filtering of for allergies and favorite foods was done using the standard Scala filter method which took the raw weekly menu, iterated over each element in the array (a dish) and then removed or kept a dish based on a Boolean condition. I felt this was appropriate as my weekly menus were not very long (perhaps 40 dishes maximum per week for each restaurant), so the standing filtering method was sufficient. If perhaps I had 100 restaurants over a year, I would have to find something a bit more efficient but, in this case, I was fine.

6. **Data Structure**

   I used mutable buffer arrays as my primary data structure because the number of elements in each buffer – whether a menu or a user preference was generally short. I used mutable buffers as it allowed me to combine, separate, and otherwise modify buffers without making a ton of new variables. As this program was simple, I didn't see the need for making an overly complex functional style set-up where all my variables with immutable.

7. **Files and network access**

   My program loads XML data from the internet using the Scala XML load method, which had to imported from an external jar. It also uses JSoup, an external library used to parse XML data.

8. **Testing**

   Testing was done on an ad-hoc basis using the Test object. As I was building method or variables in Fazer, Sodexo, or Restaurant, I checked that methods worked with various restaurants by printing them to the console. No unit tests were performed as printing to the console was sufficient. All tests discussed in the plan, such as checking for when the user gives incorrect input were tested.

9. **Known bugs and missing features**

   1. Favorite foods filters from the raw menu rather than from the allergy-free menu. Could lead to problems.
   2. No GUI.
   3. Allergy and favorite foods methods likely have either inappropriate and/or missing search terms that will lead to inappropriate filtering.

10. **3 best sides and 3 weaknesses**

    Strengths

    1. Using regular expressions along with the JSoup library to parse the XML data in the *getDay method* within the Sodexo class. While it may not look pretty, it works consistently from week to week. I learned a lot during this implementation as what CDATA is (non-traversable XML data), how regular expressions work, and how to get from a Java based library back to Scala (.asScala!)
    2. I can add new restaurants quite easily. In general, the available preferences are quite easy to modify without the need for a ton of editing.
    3. Restricting user input and limiting their ability to mess up the program

    Weaknesses:

1. My filtering methods in the Restaurant trait have a lot of repeat code. I tried to rewrite them as loops within loops but couldn't get it to work. I don't think it would have made much of a difference in speed and efficiency but would have simply looked cleaned.
2. If I want to add an allergy, or favorite food to the list of available preferences, I have to write code in more than one place. Would have been nice to create a chain of logic that allowed me to just have to write a new preference in just one place.
3. My menu display is a bit clunky. I could have displayed favorite foods with more detail (e.g. specific which day of the week a specific food is offered) and written a message like "Restaurant Closed" if no food is served on a particular day (instead I just have a blank day).

11. Deviations from the plan, realized process and schedule

Rather than start with choosing between a text based UI or GUI, I spent the few weeks trying to parse JSON vs. XML. This was the most difficult part of the project. Once I got things parsed, the project moved quite quickly and I moved faster than expected. My plan on using a GUI changed because I didn't feel the time investment was worth it. I suppose the parsing battle burned me out. My time estimate was a bit too slow, I could have planned to complete the project sooner.

12. Final evaluation

The project, I believe, is a solid one. It is relatively simple, well-organized, and while some methods could use some refactoring for cleaner looking code, there are no glaring shortcoming or undecipherable programming. I feel like my code is fairly easy to follow. I think the UI could be improved quite a bit but as this wasn't a big focus of the course, I didn't spend much time on it. If I were to do this project again, I would have jumped straight to XML and never bothered with JSON. I also would have planned on using traits earlier on, as it really helped structure my entire program.

13. **References**

- https://alvinalexander.com/scala/basic-xml-xpath-searching-in-scala
- https://www.scala-lang.org/api/2.9.1/scala/xml/XML$.html
- https://www.tutorialspoint.com/scala/scala_regular_expressions.htm
- https://stackoverflow.com/questions/33038040/how-to-iterate-through-elements-in-jsoup-using-scala/33051275
- https://alvinalexander.com/source-code/scala-how-convert-html-to-plain-text-jsoup
- https://docs.scala-lang.org/tour/traits.html
- XML data from Sodexo and Fazer restaurants. I got the XML link for each restaurant by clicking the RSS link on each restaurant's homepage. Here are some examples:
  a) alvari: https://www.fazerfoodco.fi/modules/MenuRss/MenuRss/CurrentWeek?costNumber=0190&language=en")

b) aBloc: https://www.fazerfoodco.fi/modules/MenuRss/MenuRss/CurrentWeek?costNumber=3087&language=en
c) valimo: https://www.sodexo.fi/ruokalistat/rss/weekly_rss/13918/en
d) kvarkki: https://www.sodexo.fi/ruokalistat/rss/weekly_rss/26521/en
e) csBuilding: https://www.sodexo.fi/ruokalistat/rss/weekly_rss/142/en

- I used the adventure game text UI from Programming 1 as inspiration for my own UI.
- External JARs:
    a) JSoup: https://mvnrepository.com/artifact/org.jsoup/jsoup/1.11.3
    b) Scala-XML:https://mvnrepository.com/artifact/org.scala-lang.modules/scala-xml_2.12/1.2.0

14. Appendixes

See the src folder for source code. Here's an example run of the application. User input is in red.

```
You look hungry...

Let's help you get some lunch.

To briefly explain, I'm a simple Scala application that collects real-time menu data
from student restaurants around Otaniemi.

To find you the perfect menu, I need to get some of your preferences first. You will be typing
out your answers.
Note: type carefully, as any typos will cause me to delete your entire command and record you
as merely hitting
the enter key.

Do you have a favorite place to eat? From which of the below restaurants would you like to see
menus?
Tell me by simply typing the restaurant name (e.g. Dipoli).
If you would like to see menus from more than one restaurant, seperate the restaurant names
by a common (e.g. Dipoli, Valimo)
If you are OK with the current list, simply press the enter key.

------------
Dipoli
Alvari
Valimo
Kvarkki
------------

Your favorite restaurants (type your answer): kvarkki, valimo, alvari ENTER KEY
```

```
OK, I'm saving your favorite resaurants as: Valimo, Kvarkki, Alvari

Next question: do you have any allergies to the below foods? If so, write it down!
If you have more than one allergy, please seperate them by a comma (e.g. chicken, fish)
If you don't have any allergies, just hit enter.

------------
fish
peanuts
wheat
shellfish
------------

your allergies:wheat, fish ENTER KEY
Thanks, I'm saving your allergies as: wheat, fish

Finally! Last question! What's your favorite food from the below list? Don't be shy, write it
down!
If you have more than one favorite food, please seperate them by a comma (e.g. pizza, pasta)
If you are strange and don't like any of below foods, just hit enter (I'm going to look for
pizza anyway because who doesn't like pizza :).

------------
mexican
asian
american
indian
------------

your favorite dishes: mexican, Indian ENTER KEY
Great. I'm saving your favorite dishes as: mexican, indian

Ok, I will get menus from Valimo and Kvarkki and Alvari restaurant(s), will note that your
allergies are the following: 'wheat and fish',
and I know that your favorite food is: mexican and indian.

Now press enter to get the menu! ENTER KEY

Valimo has no mexican dishes this week.

Valimo has no indian dishes this week.

 ---- Aalto Valimo 22.04.-28.04. ----
```

MONDAY:

CLOSED

---- Aalto Valimo 22.04.-28.04. ----

TUESDAY:

Salmon patties with egg sauce sauce
Root vegetable burgund
Chicken tandor salad
Champpion soup
Dessert quark

---- Aalto Valimo 22.04.-28.04. ----

WEDNESDAY:

Kofta meatballs and rice
Moroccon chickpea patties and bean salsa
Black salsify puree soup

---- Aalto Valimo 22.04.-28.04. ----

THURSDAY:

Chicken sticks with chili sauce and rice
Oats and cabbage casserole
Pea soup and pancake with jam and whipped cream
Soy pea soup, pancake, whipped cream and jam
Roast beef salad

---- Aalto Valimo 22.04.-28.04. ----

FRIDAY:

Kvarkki has the following mexican dishes this week:

Mexican minced meat soup

Kvarkki has the following indian dishes this week:

Chrispy chicken patty, curry mayonese and rice

Vegan chickpea and vegetable curry with rice

---- Aalto Kvarkki 22.04.-28.04. ----

MONDAY:

Chrispy chicken patty, curry mayonese and rice
Sausage stoganoff sauce with potatoes
Vegan chickpea and vegetable curry with rice
Potato and leek pure soup
Salad Garden & soup
Fruit quark

---- Aalto Kvarkki 22.04.-28.04. ----

TUESDAY:

Mexican minced meat soup
Minced salamon patty, egg sauce and potatoes
Vegan soy and root vegetable burgundy
Champion mushroom soup
Salad Garden & soup
Peach quark

---- Aalto Kvarkki 22.04.-28.04. ----

WEDNESDAY:

Pork cutlet, pepper sauce and mashed potatoes
Kofta meatballs and rice
Vegan maroccan chick pea patty, bean salsa and rice
Salsify soup
Salad Garden & soup
Lingonberry quark

---- Aalto Kvarkki 22.04.-28.04. ----

THURSDAY:

Pea soup and pancake with whipped cream and jam
Chicken fingers, sweet chili sauce and rice
Vegan cabbage and pulled oak casserole
Vegan pea soup
Roasted beef salad

Pancake, jam and whipped cream

---- Aalto Kvarkki 22.04.-28.04. ----

FRIDAY:

Alvari has no mexican dishes this week.

Alvari has the following indian dishes this week:

Vegetables in curry sauce (* ,G ,L ,M ,Veg)
Curry sauce (* ,A ,L)
Fish in curry and coconut sauce (* ,A ,G ,L ,M)

---- Alvari 22-04-2019 - 26-04-2019 ----

MONDAY:

---- Alvari 22-04-2019 - 26-04-2019 ----

TUESDAY:

Vegetable lunch:
Vegetables in curry sauce (* ,G ,L ,M ,Veg)
Whole grain rice
(* ,G ,L ,M ,Veg)
Vegetarian soup:
Pureed potato and leek soup
(* ,A ,L ,M ,Veg)
Lunch:
Sailor's casserole (pork) (* ,G ,L ,M)
Lunch:
Chicken nuggets (A ,L ,M)
Mustard relish sauce (A ,G ,L)
Mashed potatoes (* ,A ,G ,L)
Couscous and ginger salad (A ,L ,M ,Veg)
Roasted halloumi cheese (A ,G)
Honey melon (G ,L ,M ,Veg)

---- Alvari 22-04-2019 - 26-04-2019 ----

WEDNESDAY:

Vegetable lunch:
Bolognese and penne casserole (* ,A ,L ,M ,Veg)
Vegetarian soup:
Pureed sweet potato soup (* ,A ,G ,L)
Lunch:
Sweet and sour chicken sauce (* ,A ,G ,L ,M ,VS)
Whole grain wheat grits (* ,A ,L ,M ,Veg)
Lunch:
Saithe with mustard and leek (* ,A ,G ,L)
Boiled potatoes with parsley (* ,G ,L ,M ,Veg)
Flank steak (G ,L ,M)
Organic pasta salad and sun-dried tomatoes (A ,L ,M ,Veg ,VS)
Mozzarella (A ,G ,VL)

---- Alvari 22-04-2019 - 26-04-2019 ----

THURSDAY:

Vegetable lunch:
Dal (* ,A ,G ,L ,M ,Veg ,VS)
Whole grain wheat grits (* ,A ,L ,M ,Veg)
Vegetarian soup:
Vegetable borscht soup with yoghurt (* ,A ,G ,L)
Lunch:
Minced meat lasagnette (A ,VS)
Lunch:
Baked chicken (G ,L ,M)
Curry sauce (* ,A ,L)
Rice (G ,L ,M ,Veg)
Asparagus salad (A ,G ,L)
Green salad with pomegranate (G ,L ,M)
Potato salad (G ,L ,M ,Veg)
Á la carte:
Fried vendace (A ,L)
Lemon (G ,L ,M ,Veg)
Mashed potatoes (* ,A ,G ,L)

---- Alvari 22-04-2019 - 26-04-2019 ----

FRIDAY:

Vegetable lunch:
Carrot rissoles (A ,G ,L ,M ,Veg)

Vegan lemon mayonnaise (A ,G ,L ,M ,Veg)
Roasted potatoes (G ,L ,M ,Veg)
Vegetarian soup:
Corn soup (* ,A ,G ,L)
Lunch:
Whole grain rice
 (* ,G ,L ,M ,Veg)
Lunch:
Finnish hash (G ,L ,M)
Fried egg (A ,G ,L ,M)
Spinach ravioli salad with pine nuts (A ,VL)
Mixed salad (G ,L ,M ,Veg)
Prawns with ginger and spring onions (A ,G ,L ,M)
Herb-marinated organic tofu (A ,G ,L ,M ,Veg ,VS)

OK, that's it! If you are unhappy with your result, run this application again and try again!