Project Documentation

Name: Dang Hoang Phi

ID: 732080

Degree: Bachelor of Data Science

Year of studies: 2019 - 2024

Date: 24.04.2019

” Lunch list application”

I. General description:

Lunch list application

The project is 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 work enables displaying as many restaurants as desire, current GUI already has five available. Additional features:

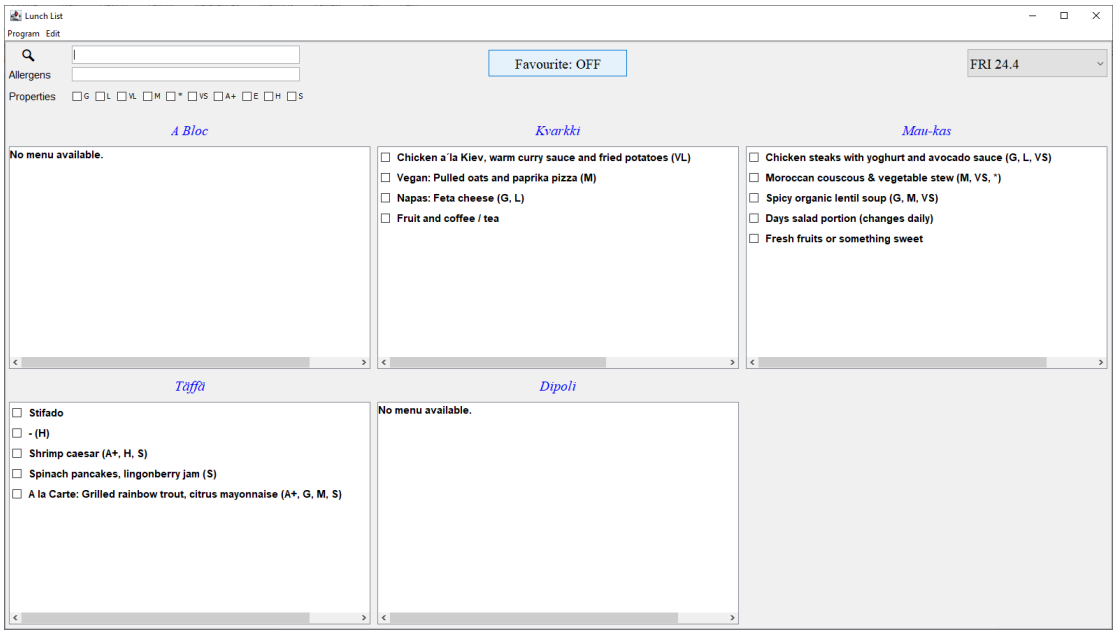
- Filter menu items based on allergens
- Illustrate only items with certain properties or ingredients
- Support for choice of favourite restaurants.
- Inform user about menu items that the user has reported to like.

There are three essential parts for the implementation:

1. Data structure for lunch list data
2. Import/Export files
3. GUI (Intermediate difficulty level)

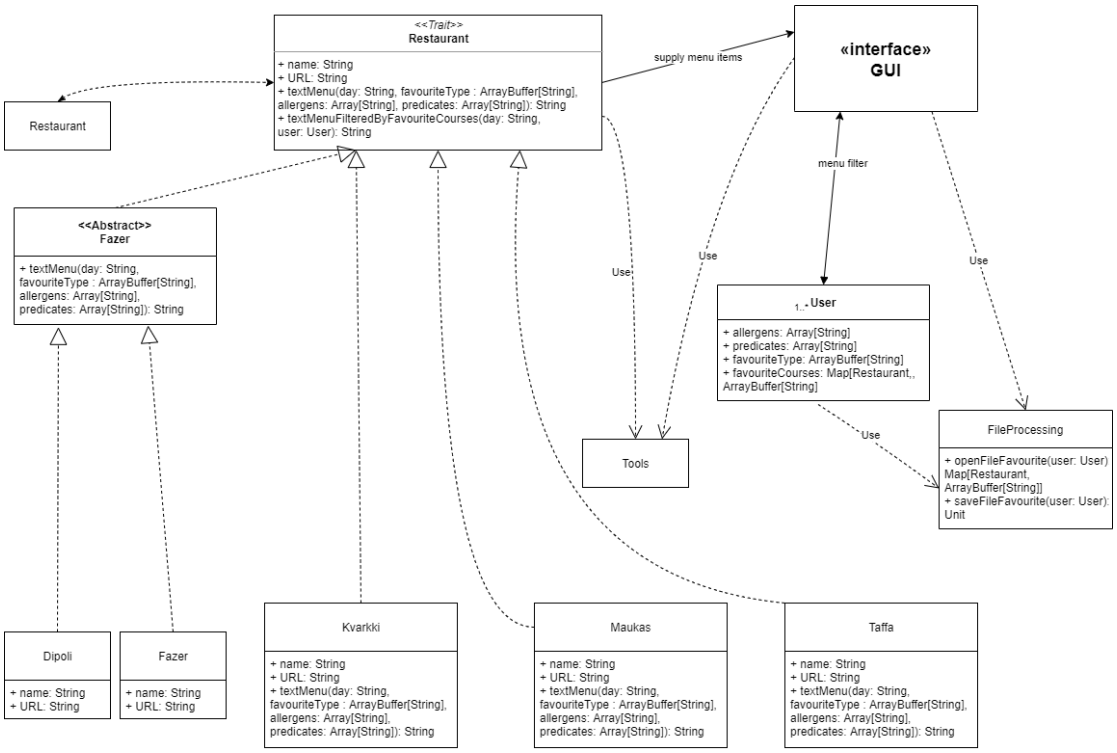
Level completed: Intermediate

II. User interface:



- The project is a graphic user interface that mainly comprises blocks of menu items of restaurants (here is the example of 5 Otaniemi restaurants)
- There is a menu bar that helps clear favorites data (elaborated below) or close the application.
- Search field: Show only menu items containing certain ingredients or belongs to certain category (in some cases, for example, a salad does not necessarily have the word ”salad” in the name), press Enter after done edited.
  Example: milk, beef, tea
- Allergens field: Filter menu items without certain ingredients or belongs to certain cetegory, press Enter after done edited.
  Example: milk, beef, tea
- Properties checkbox: Filter menu items based on certain properties
- User may add an menu item to favourites by ticking the corresponding checkbox
- Favourite ON/OFF: Present only favourite menu items
- Date drop-down menu: Dates selection

III. Program structure:



IV. Algorithm:

- Filter menu items by searching, allergens, properties, favourites: The features are variables in class “User” which can be accessed freely from other classes. Each time a user performs a change to any filter condition, these variables is updated right away. As a result, a “Restaurant” can produce a list of menu items with corresponding predicates and selected date. GUI has the list and updates itself.
- File processing: Each time close the application, the users can save data of their favourite courses, this is done by creating a text file and write each course in a line. When the application is re-opened, it does string processing to turn the text in the file into data again.

V. Data structure:

- Dynamic structure (ArrayBuffer) is used for preferred properties or courses, to simplify add/remove elements
- Static structure is used for list of restaurants, searching and allergens filter as they don’t change (searching and allergens filter need the whole string being processed)

- Map (mutable) is used for favourite courses, each elements is in a form (Restaurant -> ArrayBuffer[String]).

VI. Files and network access:

- The project create a file .txt in user's directory (if user choose to do so) to save "favourites" information, and turn the text in the file to usable data so that GUI would exhibit what user have chosen before. If user declines to save the records, text inside the file will be clear.
- Data is fetched by available JSON files on websites of restaurants

VII. Testing:

The program was tested straightly by using the GUI many times, doing trials and errors. During implementation phase, there have been careful observations and manual testing by a text UI. The program was founded no bugs so far. Unfortunately, there has not been big enough data to test the efficient running time as the plan.

- Filter: The GUI illustrates accurate menu items fitting the predicates. If there is no found item, a notification is pop up.
- Save and load program: After save, close, reopen, the program continues and display favourite items exactly same as the previous access.

There remains data that contains no menu items as the restaurant has been closed or the information has not been updated yet, the try – catch process works properly for this problem

The program also has been tested and work successfully under no internet connection condition.

VIII. Known bugs and missing features:

Through careful manual debugging process, all found bugs have been fixed.

The project can be improved by adding features to choose favourite restaurant, this is quite the same with favourite menu items selection feature: Have a collection to save which restaurants have been chosen and display only corresponding blocks. Another feature could be a button to access the link of websites or locations of restaurants.

IX. 3 best sides and 3 weaknesses:

3 best sides:

- A GUI with easy interactions to use.
- Many additional features.

- The class is reasonably well-structured which helps debugging and developing become easier.

3 weaknesses:

- Some feature of a class should be safer to access. For example, variable “allergens” in class User should be private, and such methods as “editAllergens” should be added.
- Some class or def is written in a good order which is hard to read and comprehend

X. Deviations from the plan, realized process and schedule:

Phase	Name	Expeted time period	Realiy time period	Expected working hour	Reality
1	Text-based interface	25/2 -> 8/3	25/2 -> 20/3	30	50
2	Make usable GUI interface	8/3 -> 23/3	20/3 -> 7/4	70	30
3	Make a better GUI	23/3 -> 1/4	7/4 -> 18/4	10	20
4	Debug and complete	1/4 -> 23/4	18/4 -> 23/4	30	10

The process deviated as the time to produce UI and GUI takes longer as expected because it is difficult to seek source that explain graphic libraries in a proper way. After finishing the GUI for lunch list, I am already now quite comfortable with those ones.

XI. Final evaluation:

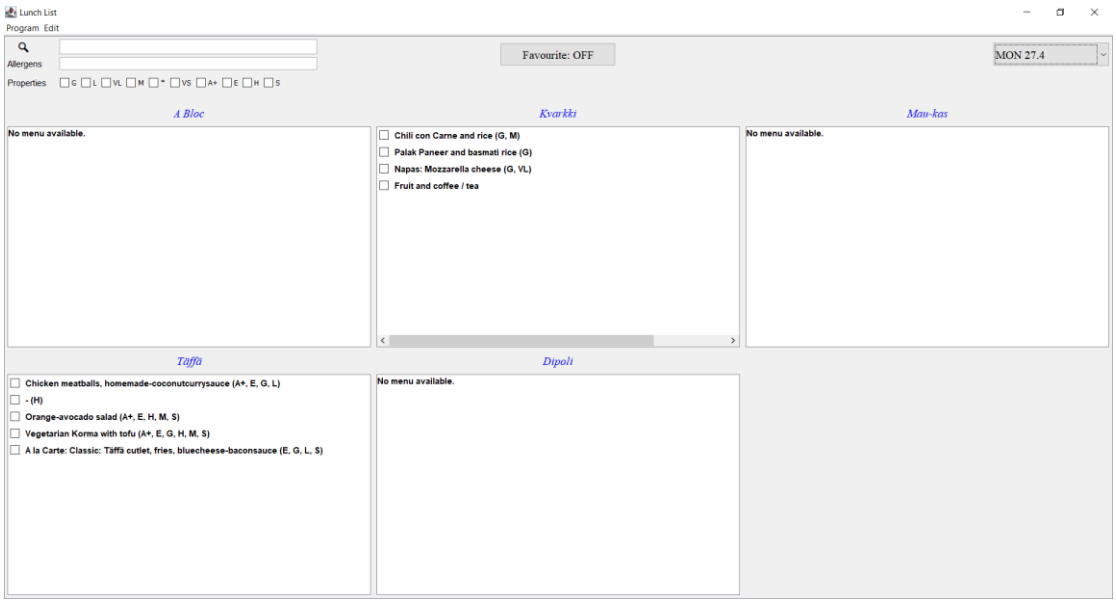
Al in all, the project has almost covered all requirements the topic, enable different preferred filter properties, and guarantee the accuracy of the output. The project can be improved in many aspects in the future, starting with improving GUI with many other functions that I have not been acquainted with. I would organize and implement the class in better way if I started the project again, and find a more general way to parse the data instead of expecting an available JSON file from the restaurant

XII. Reference:

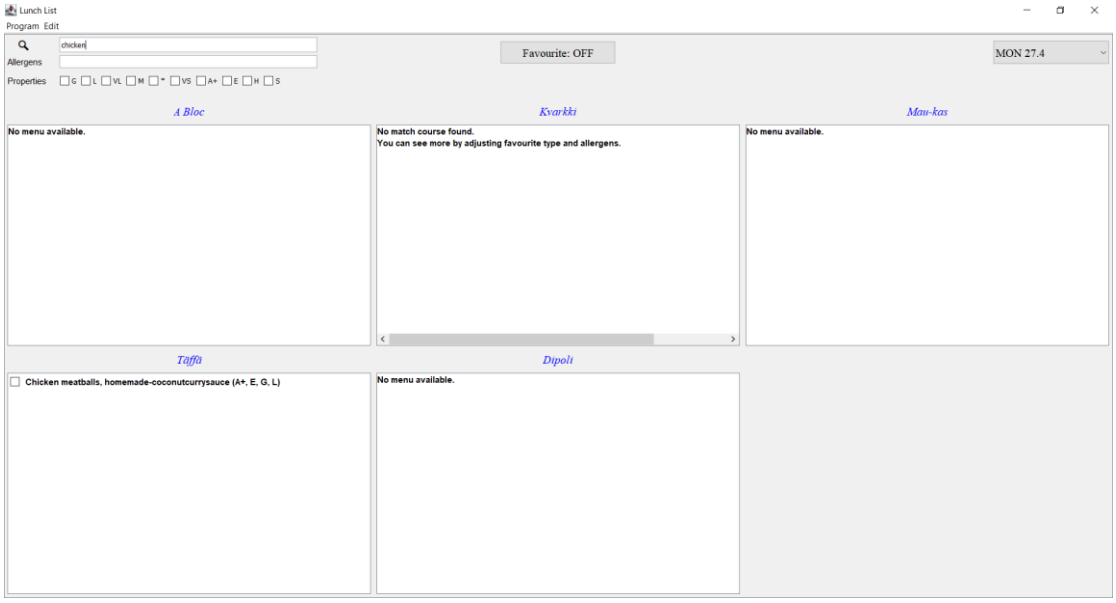
https://kanttiinit.fi/
http://otfried.org/scala/gui.html
http://www.furidamu.org/blog/2012/09/18/beautiful-json-parsing-in-scala/

XIII. Appendixes:

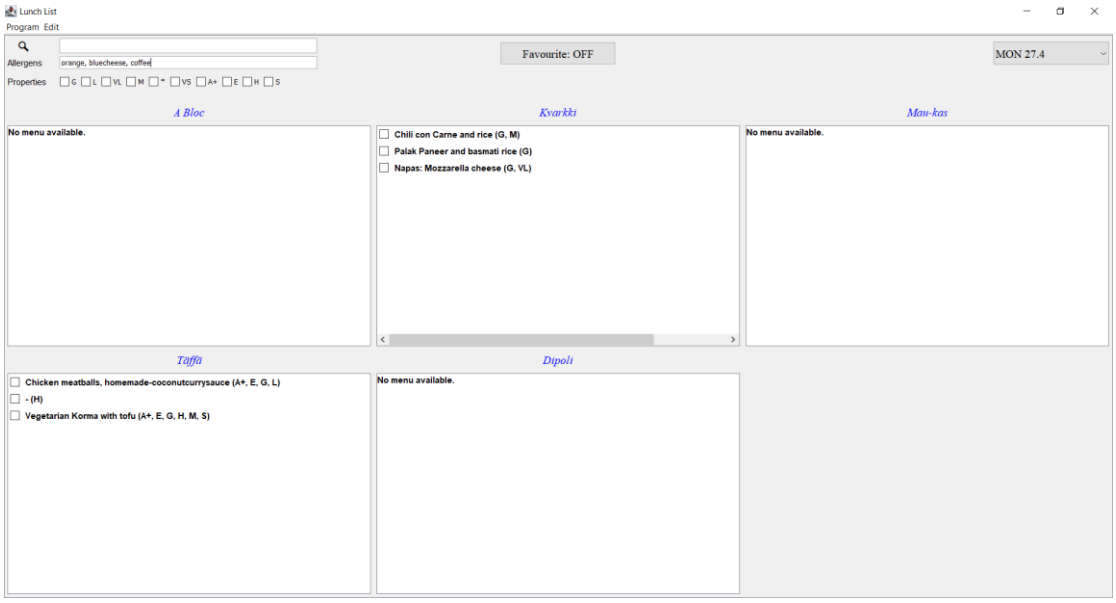
The graphic user interface starting like this:



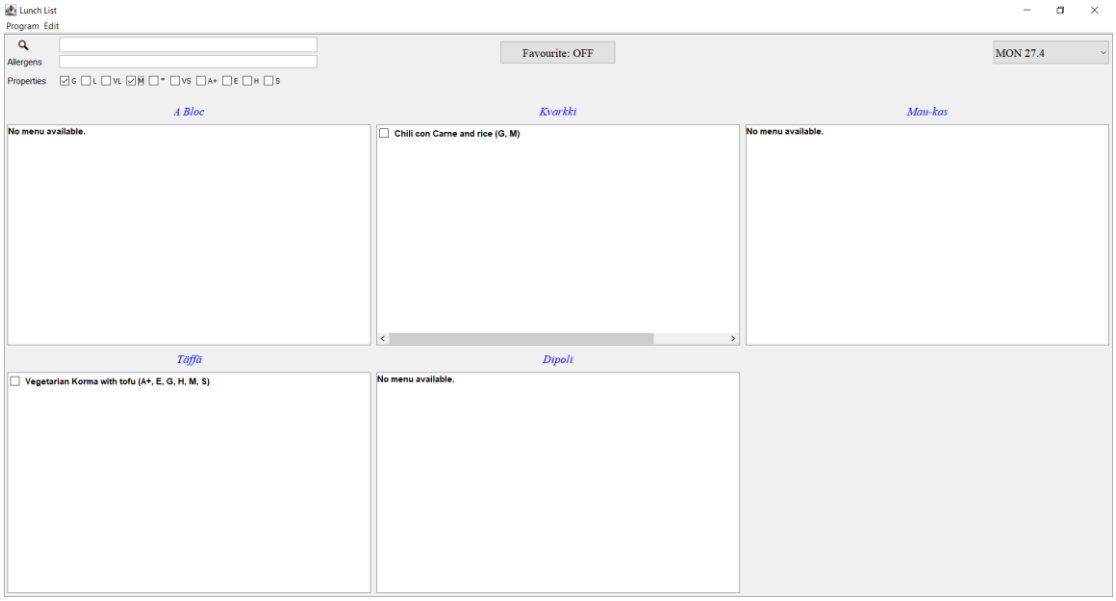
Search menu items containing chicken, by type “chicken” and press “Enter”:



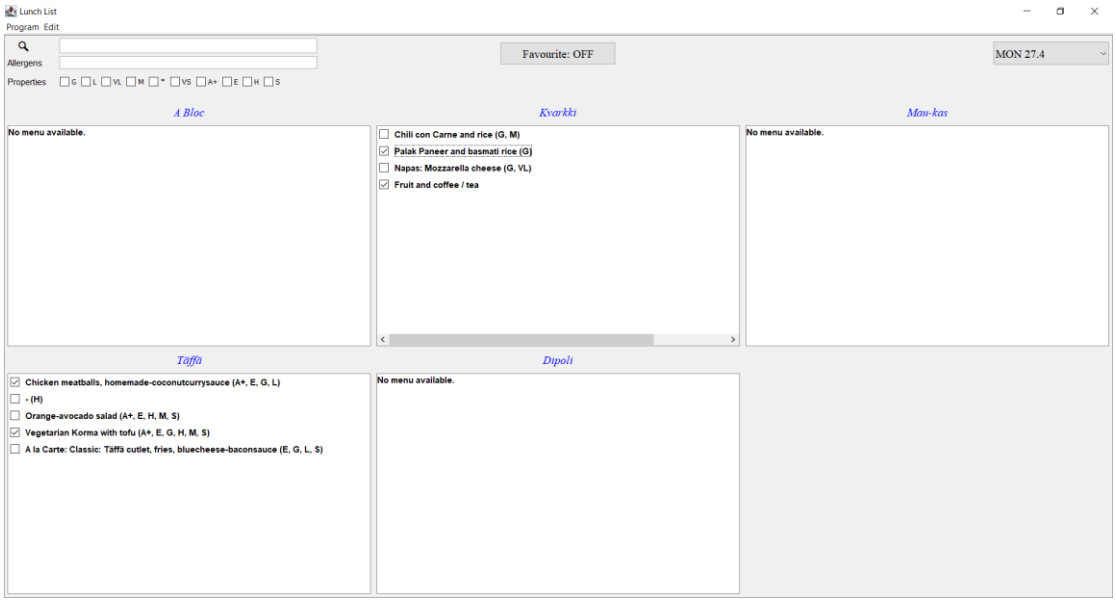
Same ways to filter items by allergens, multiple ingredients should be splitted by commas:



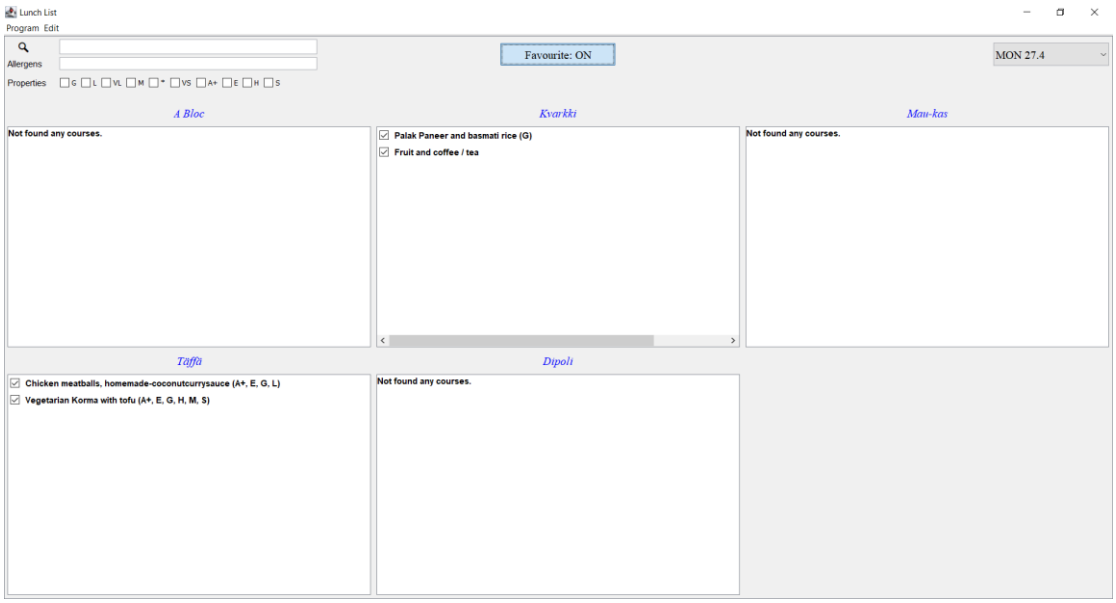
Filter by properties:



Select dishes to add to favourites:



Click the favourite button to display on favourite items:



Same things can be applied for the other days: