**Functional Requirements**

**RF1: Register new elements**Register new entries for every entity i.e. restaurants, clients, products, and restaurants. The user will select which entry will be added and their fields will be filled out with the data the user will specify.

**RF2: Update element information**  
Change the information of an specific, already created element in the program, as long as the change doesn’t imply that other elements are left with invalid references, such as changing the ID of a client, which would leave various orders stray.

**RF3:** **Seamlessly save program’s state**Saves the data of the program’s state every time something is changed. The user’s input has no involvement in this function, as the update happens after every update.

**RF4: Export order’s data**Creates a .csv file, which can be imported back, with the information of all orders.

**RF5: Display clients and restaurants**Displays the information of either all clients or all restaurants.

**RF6: Search for a client**Looks for a client using binary search. Then, displays its information and the time the search took.

**RF7: Import data**Allows for the importing of data from the specified .csv files.

**Non-functional requirements**

* The user can only update the status of an order forwards.
* The search function must be efficient
* The program must use at least two of the sorting algorithms bubble, selection and insertion.
* The program must do at least one sort using the Comparable interface and one using the Comparator interface, using the Collections or Arrays sort in both cases.
* Test must use a CSV file with 1000 entries.
* All clients must have exactly two names separated by a space.

**Class Diagram**

