Requirement specification based on Personas and Scenarios (plus some additions).

Note:
Items that are placed inside [...] should NOT be included in the application.
Items placed within {...} can be included at will, but they are not mandatory.
I have filtered some of the requirements (which is why they might not be included from the base documents).

The Flying Dutchman – Nataly Grinchenko

Functional requirements

The system should provide functionality that allows all customers to:

1. See available items (open menu)
2. Order beverages and simple food from servicing bartender at the table (on portable tablet)
   a. Single order
   b. Group order
      1. Group bill
      2. Split bill
3. Change order
4. Pay at bar, or to bartender/waiter/waitress at table
   a. See also 2.b.2 - splitting the bill
   b. **NOTE:** money transaction is not implemented more than symbolically
      - The new amount can simply be entered in a field (no card reader, Swish or similar equipment/method should be implemented)

VIP customers should also be able to:

5. Log in (at table)
   a. Simple login (user identification method at your choice)!
   b. **NOTE:** There are no requirements on security in this project.
6. Log out
7. Order food from a separate menu (at table)
8. See account balance (at table)
9. Order and Pay from account (at table)
10. Fetch special beer/drink from fridge or bar
    a. with combination lock
    b. code is changed between purchases
11. Add to account (at bar)

Bartender, Waiter and Waitress should be able to:

12. Log in
    a. Simple login!
    b. **NOTE:** There are no requirements on security in this project.
13. Log out
14. See availability of products (General req)
    a. Remove product (temporarily) from menu
    b. The bartender should immediately be notified when an item is running low
15. Modify/Recalculate price of single product as compensation for mistake. This is used in order to compensate customers, if we cannot offer it on the house (see 16).

    a. The Price modification has to be entered with a comment as to the reason.
    b. Different fixed categories of reasons

   Suggestion: Use a separate dialogue for this!

16. Offer product on the house or at discount
    a. Balancing the income/expenses
    b. Update number in stock
17. Notify security of problem (discretely)
18. Get order for certain table
    a. Change items on order
    b. Allow for splitting the bill
19. {Reserve table for group at specified time}
    a. {Remove reservation}
    b. {Print reservation note}

Bartender/Owner needs to be able to:

20. Manage stock
    a. Revise amounts
    b. Order refill of items
    c. Add/remove items from menu
21. [Manage prices.]
22. [Do accounting.]

General requirements

23. We need to be able to find products according to certain content
    a. Allergies - Gluten, Nuts, Lactose etc.
    b. Alcohol content
    c. Tannins (for wine)

Non-functional requirements

24. An order can consist of up to ten items at the same time (18)
25. Low limit is at five (five) items (14.b)
    a. A warning should be given when there are less than five items left of a certain type
26. The security notification should be discreet, but accessible within three seconds (17)

Additional details

27. The interface visual design <u>must</u> connect to the thematic background of the Pub
28. Drinks should be listed with the following details on the menu (**NOTE**: no id number):
    a. Beers
        - Name
        - Producer/Brewery
        - Country
        - Type (IPA, lager
        - Strength
        - Serving size (tap, bottle)
        - Price
    b. Wine
        - Name
        - Year
        - Producer
        - Type
        - Grape
        - Serving size (glass, bottle)
    c. Cocktails/Drinks

- Name
- Strength
- Contents/Recipe (for allergy purposes)
- Serving size

29. Food (presented on separate menu)
    a. Name of platter
    b. Ingredients (just listed as a long list
    c. Price
    {d. Picture}
30. For the bartender and manager, the number of remaining servings should also be shown
    a. It should be easy to see when an item is low in number

Notes

31. There should be one 9'' touch-screen at each table, and a large screen 27" – 30" at the bar.
    a. The system should work well on both sizes.
32. Waiters and Bartender should also have a small, portable tablet, 9", for taking orders at the tables

Course based requirements

These requirements have to be fulfilled in order for the project to be accepted. Should you have any problems with these you have to talk to the teacher as soon as possible. These requirements are also describing the most important knowledge that you have to know individually at the end of course (the individual examination).

33. **You should have fun programming!**
34. The system should be implemented using MVC (or a similar version)
35. The system should provide **three** distinct interface languages
    a. Dynamically changeable
    b. Remember choice over the whole session
    c. The system should implement Drag and Drop ***for all suitable actions***
    d. All functions with drag and drop should also have an alternative way to initiate (button or menu).
36. The system should implement an UNDO/REDO functionality
    a. **All** different actions that change the model should be undoable (e.g., add to order, remove from order, empty order, delete order).
    b. The undo should be "unlimited".
37. The system should be possible to resize between the bartender view (27") to table tablet (9"-10")
    a. NOTE: resizing can be directly from one size to the other, no gradual change needed
38. The system should be reasonably debugged
39. The code needs to be well documented (minimum according to the posted description).
40. In order to have an accepted project, you only need to be able to show that the system can run nicely on **one** platform you choose, and on **one** browser you choose. This means that if you can show it running on one computer, that will suffice.

Some useful (?) hints (unsorted)

A. Remember that you have a number of files with code examples uploaded to Studium. There you can find examples of how to implement most of the project requirements, e.g., MVC, UNDO-REDO and internationalization.

B. Make sure that you understand these code examples when you implement them in your system.

C. You may not use any other programming languages or libraries than:
   a. JavaScript, HTML, CSS and jQuery, or
   b. Python, Tkinter and pygame.

D. Try to divide the work between you in the group so that you all work on the user interface parts (but see F).

E. I suggest to work in pairs (pairwise programming), where you may change the pairs now and then. This will give you different perspectives on how different people think during programming.

F. Make sure that everybody in the group understands the code (educate each other).

G. Don’t forget to log your time.

H. Start the programming of the project with the following parts:
   a. Look at the requirements and make sketches of the different (sub-) displays: order, menu items (difference between customer, VIP, waiter and bartender).
   b. Go through the database that has been provided (Flying Dutchman Files) and select a subsample of the database (to test the system on) appr. 30 posts is good.
   c. Implement the functions that are needed to get the information you need (cf. H.a above).
      i. Get different beverage and user details from the static database.
      ii. Implement a simple “database” for the dynamic parts (orders, tables, etc.) and the functions needed to manage it.
   d. UNDO-REDO – implement it for the ones in ii) above.
   e. Make sure that you can run the scripts on the console (browser inspection mode) and that you get the correct answers.
   f. Connect the HTML functionality to the console functions (one at a time?)

I. When you create the interface as HTML page, start by implementing all the containers that are necessary, and add the CSS afterwards.
   a. This means that you will be able to focus on the content that has to be displayed, without being disturbed by the shapes and layouts.
   b. You can still work on the style sheet, but comment them when you work with the information design.

J. Make sure that you use a version management system (such as GIT) so that you have backups throughout the development

K. Don’t hesitate to ask for supervision (book a meeting or send a question on Studium).

L. HELP EACH OTHER, both inside the groups and between the groups!
   a. It is easy to ask: “How did you do to implement this!”.

M. **Remember:** **IT IS NOT A COMPETITION!**