**Relative contributions of members:**

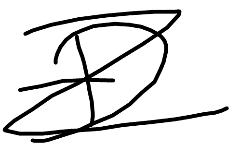
|  |  |  |  |
| --- | --- | --- | --- |
|  | Design/Planning and Coordination | Coding and testing | Other |
| Benjamin Griffiths | 14.25 | 18 | 14.3 |
| Diego Toledano | 14.25 | 13.8 | 14.3 |
| Gunjan Sahityani | 16 | 13.8 | 14.3 |
| Lorenzo Menegotto | 16 | 15.8 | 14.3 |
| Shah Alam | 12.25 | 12.8 | 14.3 |
| Sorin Olteanu | 16 | 15.8 | 14.3 |
| Uyi Fitzpatrick | 11.25 | 10 | 14.3 |

**Signatures**

**Ben Griffiths**

****

**Diego Toledano**

****

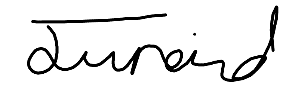
**Gunjan Sahityani**

****

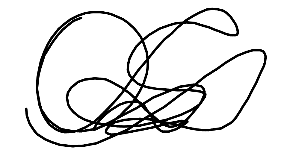
**Lorenzo Menegotto**

****

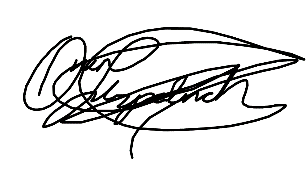
**Shah Alam**

****

**Sorin Olteanu**

****

**Uyi Fitzpatrick**

****

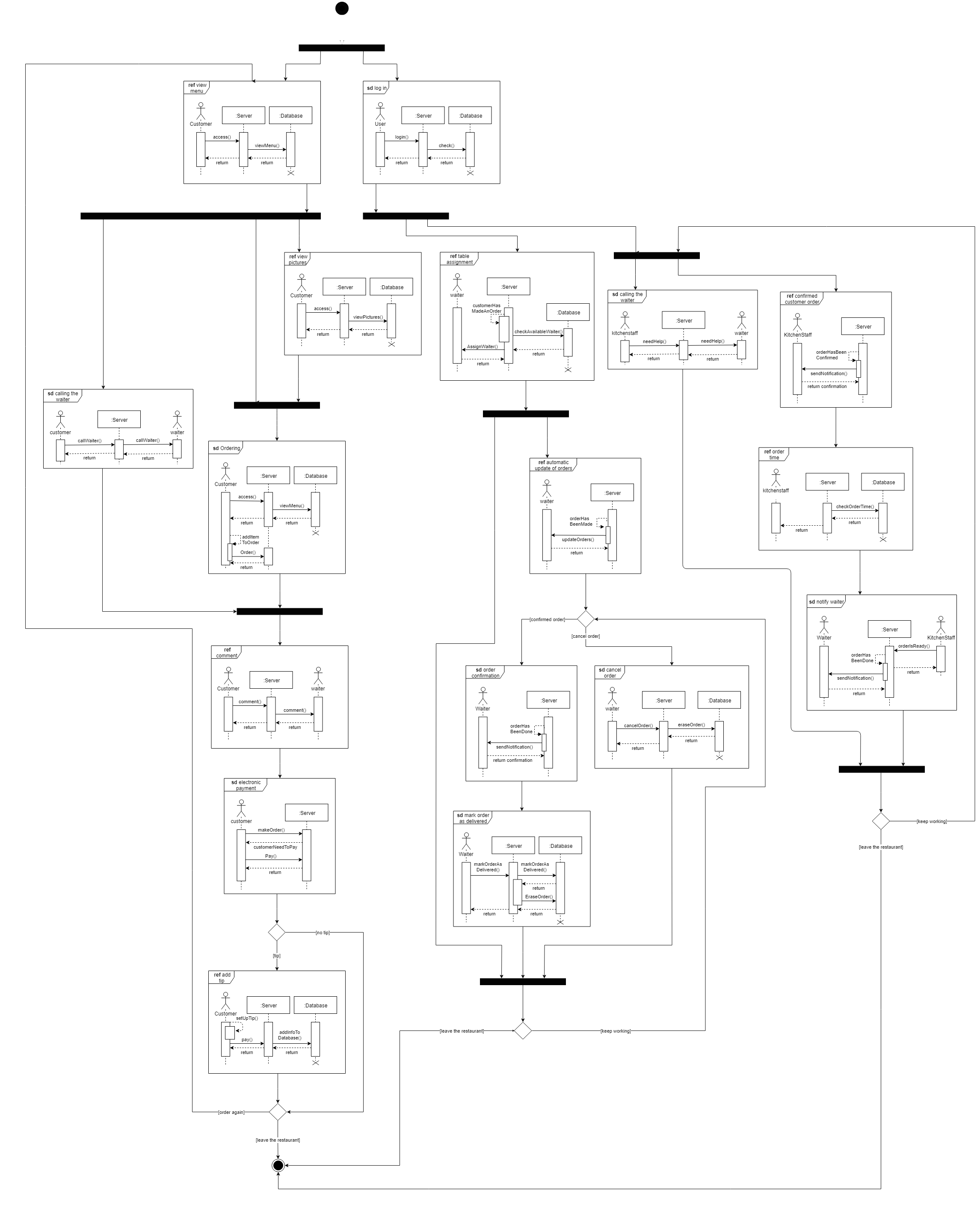
**Brief high-level description of the main components of your product: (include diagrams)**

Our project has a total of 3 pages depending on what the role of the user is. The default view is the menu, as the product assumes the general user will be a customer (and as customers do not require a log in). The menu includes a list of pulled items that are currently for sale, as well as their information (e.g. calories/price). The page also includes an interactive checkout basket which shows the updated price of the order. The user has the ability of confirming the order and inputting payment information to pay. The customer is then kept up to date with their order by interactive icons which display the status of their previous order.

Waiters have a list of live orders that have not yet been delivered however have been already confirmed by the customer. Each order has a timer to show how long the customer has been waiting for. As well as the updating of each orders status (e.g. Accept, Cancel, Delivered) the waiter also has the functionality of editing the current menu. This is done through a popup that allows the removal/addition of new menu items. Each waiter is automatically assigned to the appropriate tables/orders and can be notified for assistance (by either kitchen staff or customers).

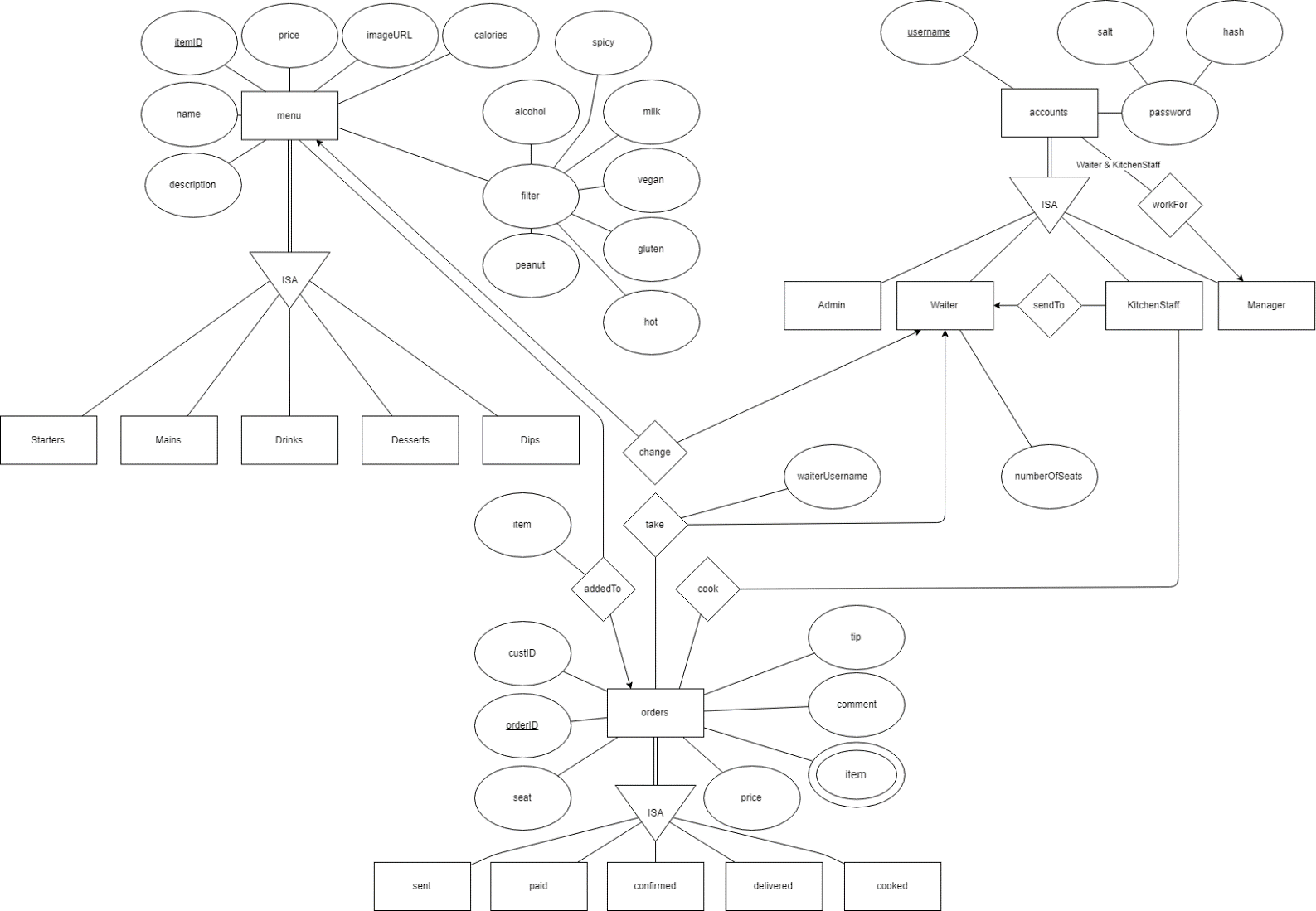
The kitchen staff page contains a list of orders that need to be cooked and they can be clicked to show that they have now been cooked. It’s a simple interface to maximize how intuitive the design is. The orders that need to be cooked have a timer to display how long the customer has been waiting, as well as a colour-changing scheme to display this numeric information in an easily interpreted visual manner. There also is a button that can call a free waiter if the kitchen staff requires any help.

The high level ER diagram for the pages described above can be found on the following page…

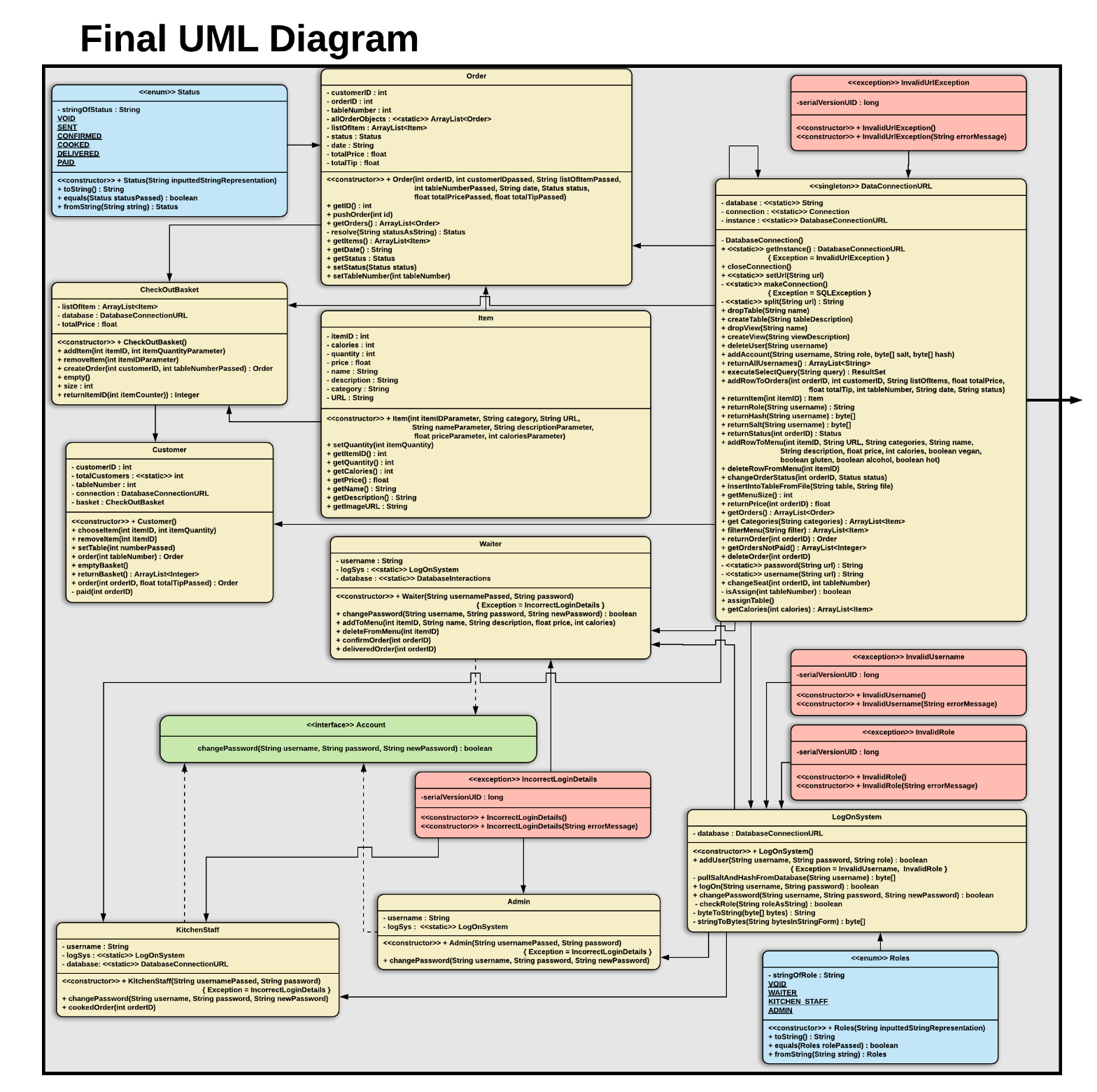


All of this data is kept in our postgreSQL database where we have 3 tables. One table (named ‘menu’) is dedicated to the current menu which holds all information about each menu item (containing a primary key, categories, imageURL, name, description, price, calories, and also Booleans linked to the attributed ‘peanut’, ‘vegan’, ‘gluten’, ‘spicy’, ‘alcohol’ and ‘hot’). Another table is the ‘orders’ table which holds all necessary data on each order that’s currently in use. Each order’s ID, list of items, table number, time, status, price and tip is saved. The waiter which is assigned to the order, as well as which customer ordered are also saved in this table. The final table (‘accounts’) holds the information necessary for a secure authentication system. This has columns such as ‘username’, ‘salt’ and ‘hash’ (which are the secured password) as well as ‘role’ which shows the authority of the linked account. If the account is a waiter, then the current number of tables waited are also stored in this database.

The database ER diagram can be found below:



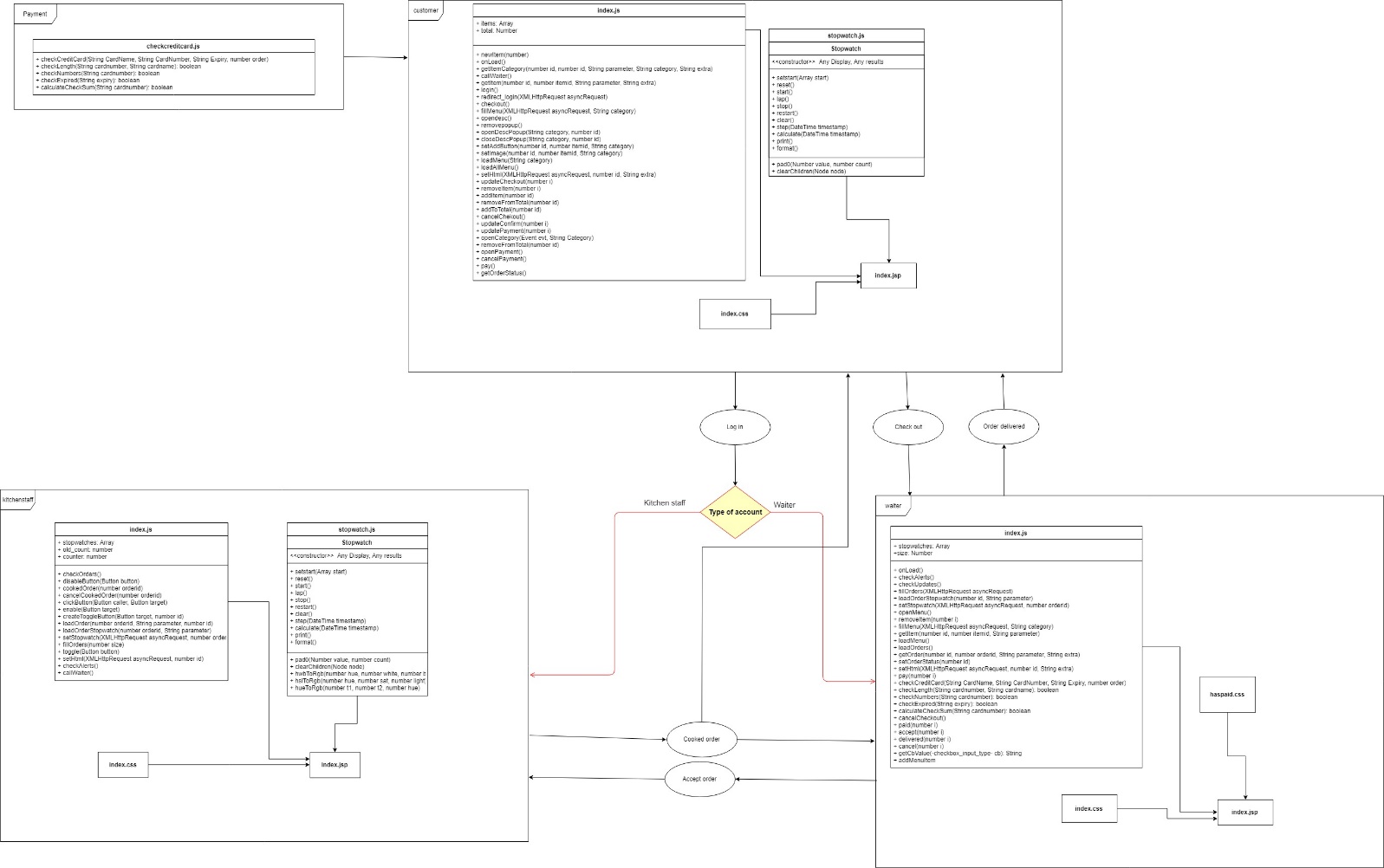
**A description of packages:**

We have 2 packages in use for our code and they are ‘BasicInteractions’ and ‘servlets’. The ‘BasicInteractions’ package contains all code that’s ‘back-end’ and contains the functionality of the ordering chain as well as the database updating/pulling. The most influential file in this package is the large “DatabaseConnectionURL” which contains all methods that push/pull from the database. This level of interaction simplifies any database interaction as well as ensuring one singular connection is needed rather than a connection per class (especially since this class is a singleton). The classes such as “KitchenStaff”, “Admin” and “Waiter” are linked to the interface “Account” however “Customer” isn’t as customers do not have accounts/logins. The log in system is called once instances of these classes are being (attempted to be) created. If failed, one of the multiple custom exceptions is thrown (which can be seen in the diagram below in red). The log in functionality is contained within the class “LogOnSystem” which interacts with the database through the “DatabaseConnectionURL” class. To ensure roles and statuses are within the allowed options, two enums have been made and are shown in the diagram in blue. Classes can contain ArrayLists of other class instances, such as “Order” having an ArrayList of the “Item” instances that is what the order consists of, and all relations between classes like this are shown with arrows, in the diagram below:

The servlets package contains all the code which links the backend to the front end. Each servlet handles a different category of requests. Each servlet requires different levels of permission to access or change different parts of the information, which is achieved with help from the login servlet. The login servlet handles user login requests and sends information to other servlets (such as user IP addresses for the alert system, whether a particular IP address is logged in and the username of the waiter with the least amount of tables). For the orders the GetOrders servlet is used, this servlet handles creation of orders, pulling order information from the database, editing order information and sending order information to the users. This servlet restricts most of these functions to the waiter and kitchenstaff pages, but allows the customers to create orders, check their order status and pay for their orders. For the menu the GetMenu servlet is used, this servlet handles the way information of the menu items in the database is retrieved as well as creating and removing menu items from the database. For the alerts the Alerts servlet is used, this servlet handles the creation of alerts, storing of alerts, and alerts the target users. Users are added to alerts via their IP address, and check for alerts using the same IP address. Alerts contain a custom message; defined by the web pages. The servlet package uses a servlet class which is a facade for java servlets making them easier to use. The class also contains methods which allows communication between servlets and interaction with the login system.

**[FRONT END PACKAGE EXPLANATION]**

The front-end UML diagram can be found below

****

**List of user stories that are fully implemented:**

* As a user, I’d like to be able to securely log into my corresponding page.
* As a customer I want to be able to see a menu so I can select my dishes.
* As a customer I want to be able to see images of the menu choices while selecting my dish.
* As a customer I want to be able to order food so I can eat.
* As a customer I want to be able to have a complete information on ingredients and calories for each dish so I can both avoid foods I am allergic to and follow my diet.
* As a customer I would like to have an intuitive way of submitting an order so that the process is straightforward.
* As a customer I want to be able to call the waiter at any time so I can get help with my order.
* As a customer I want to be able to include a tip in my order, for the waiter.
* As a customer I want to be able to electronically pay through the website.
* As a customer I want to add a comment to my order so I can give personal preferences.
* As a waiter, I would like to be able to change the menu, so I can show customers only currently available dishes.
* As a waiter, I want to be notified when a table is ready to order, so that I can confirm the order.
* As a waiter, I want to see the orders by the time they were placed, so that I can prioritise them.
* As a waiter, I want to be able to mark the order as delivered, so that the order progress is tracked correctly.
* As a waiter, I want to cancel the customer order, so that the kitchen staff knows when people change their mind.
* As a waiter, I want to be notified when the kitchen is ready with a dish, so that I can deliver it to the table.
* As a waiter, I want to be able to change the status of an order so that the customer is kept informed.
* As a waiter I want to be notified when the client needs help so I can assist them.
* As a waiter, I want to be automatically assigned to a table so I know which orders I must complete.
* As kitchen staff I want to be informed when a customer order is confirmed so I can start preparing it.
* As kitchen staff I want to be able to notify waiters once an order is ready so that they can deliver it.
* As kitchen staff I want to know the times at which the orders were made so I can tell if I am on track.
* As kitchen staff I want to be able to notify waiters for their attention when I need help.