NATIONAL COLLEGE OF IRELAND

BSC (HONS) IN BUSINESS INFORMATION SYSTEMS

2013/2014

ANDERSON AUGUSTO SIMISCUKA

X13115642

SUBSELF

TECHNICAL REPORT

National College of Ireland

Table of Contents

1. Executive Summary

Subself is primarily a system in which people can create their sandwiches step-by-step at sandwich stores without the need of waiting in a long queue and talking to an attendant. Users can use Subself on interactive kiosks in shops or use their own devices.

This system allows the customer to choose between different types of bread, salad, toppings, sauces and Drink selection.

The system contains other application to the kitchen and cashier staff, where they can see the orders and queue. And an application for managers, where they can see their shop stats and ingredients consumption using different timeframes.

The system will be web based and all applications will connect to the same database.

Subself was implemented using Java (Object Oriented applications), Eclipse platform – JavaServer Faces, customized Twitter Bootstrap for the mobile-desktop enabled display and MySQL. Model View Controller is used and Photoshop for customized buttons and logos.

2. Introduction

2.1. Background

There is a lack of interactive kiosks and apps for fast food ordering. Often you find yourself waiting to order a sandwich because of a customer that takes a really long time choosing their ingredients. Subself offers a solution for all these problems and also administrative tools for shop staff and managers. It is possible to use any module on desktops or on mobile devices, solving the problem of customers waiting in a long queue and talking to an attendant for a long time.

2.2. Environment

Subself offers a self-service solution for mobile and desktop users, customers in the shops and also administrative tools for shop staff and managers.

Subself allows the customer to choose between different types of bread, salad, toppings, sauces and Drink selection.

The system contains other application to the kitchen and cashier staff, where they can see the orders and queue. And an application for managers, where they can see their shop stats and ingredients consumption using different timeframes.

2.3. Technologies

Subself was implemented using Java (Object Oriented applications), Eclipse platform – JavaServer Faces (JSF), customized Twitter Bootstrap for the mobile-desktop enabled display and MySQL. Model View Controller is used and Photoshop for customized buttons and logos.

Model View Controller in Subself works like this: Model – Three classes: Connection Factory, the different objects classes and a class containing methods to handle the database according to the objects. View – XHTML pages to show contents. Controller – Managed Beans with all the methods that users will access through the XHTML pages. The managed beans also connect with model layer and controls view layer.

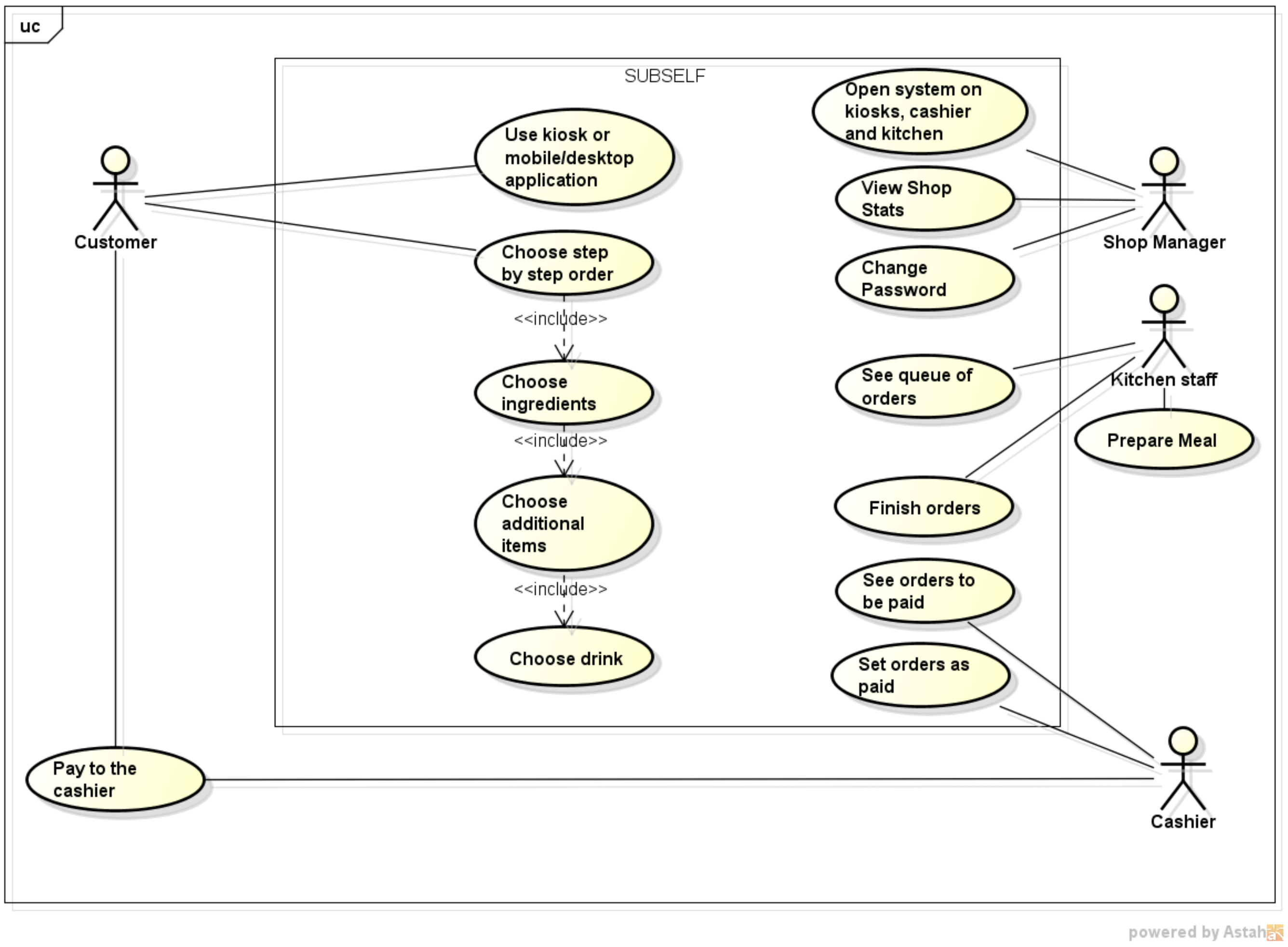
3. System

This section presents the technical details of the project. These details will be described in the sections Requirements, Design and Architecture, Implementation, Testing and Graphical User Interface.

3.1. Requirements

3.1.1. Functional Requirements

Functional Requirements will be explained according to this Use Case diagram:



Choose Step by Step Order / Kiosk-Mobile-Desktop applications

Customer can customise his/her order. In the Step by Step option, ingredients and additional items will be selected by the customer. In the mobile-desktop application, customers need to select a shop where they want to pay and pick their order.

Open System

Managers will have access to Subself Admin, the application where they can select:

- Subself Kiosk module to be opened on the interactive kiosks;
- Subself Cashier module to show to the cashier the orders to be paid by customers;
- Subself Kitchen module to show to kitchen staff the orders that were paid and need to be prepared and, finally;
- Subself Stats module, where the manager will see the amount of orders sold as well as each ingredient.

View Shop Stats

Shop Managers can use Subself Stats module, where the manager will see the amount of orders sold as well as each ingredient.

Change Password

Shop Managers will have access to Subself Admin, where they can change the shop password.

See Orders to Be Paid and Set Order as Paid

Through Subself Cashier module the cashier can see the orders to be paid and after customer payment, set these orders as paid, so they will be shown at kitchen screen.

Queue of Orders

Through Subself Kitchen module kitchen staff can see the orders that were paid and need to be prepared. Here kitchen staff can finish the orders too, to clean the queue.

3.1.2. Data Requirements

Subself relies on data transaction. Every order is stored on the MySQL database and cashier, kitchen and stats modules retrieve information from this database. The database also contains shops, their IDs and passwords.

When customers use the system, their orders are attached to a store. When Managers login, they need to enter they shop number and password, so all the information shown on cashier, kitchen and stats modules are related to his/her shop.

The kioks show the Subself Admin Subself Kiosk module, which is the same application used by customers, but is logged using the administrator login and all the orders will always be attached to this shop.

MySQL database contains two tables. Orders table contains order id (primary key), shop id (foreign key), sandwich ingredients, drinks, extras, timestamp and order status. Shops table contains shop id (primary key), shop name and shop password.

3.1.3. User Requirements

Any device capable of opening a web browser is needed to access the system if it is in the cloud. Subself is screen responsive and adapts to large, medium and small screens.

If it is running locally, it uses Glassfish server and needs the MySQL database running, so the user needs to have these technologies available.

3.1.4. Environmental Requirements

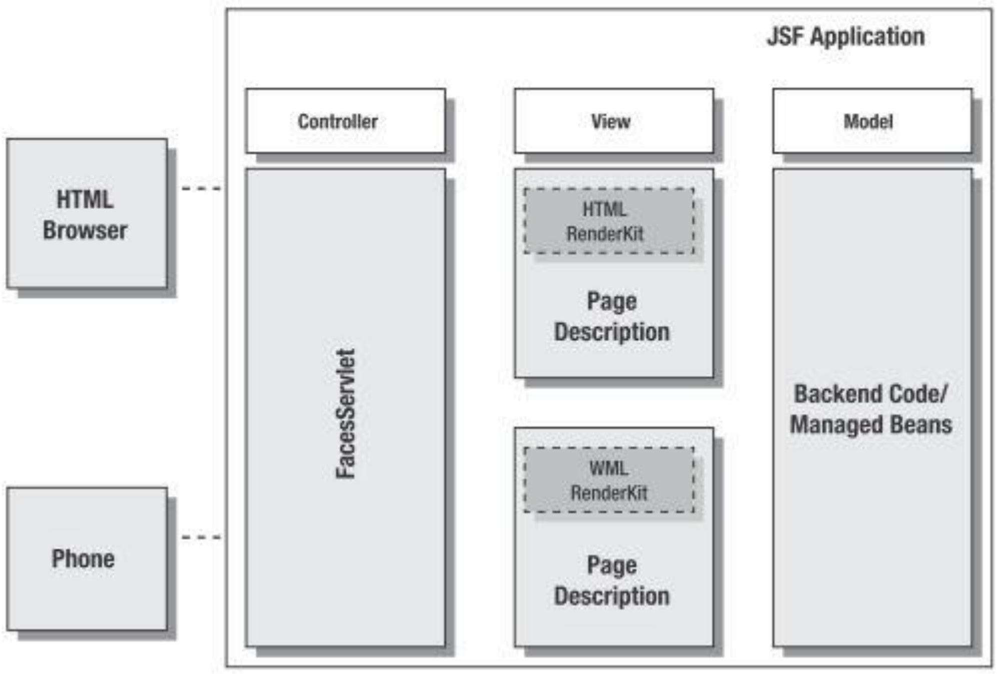
Eclipse was used with Glassfish server to implement Subself. Twitter Bootstrap to create screen responsive web pages. Photoshop used for icons, logos and buttons. MySQL used to store and provide data.

3.1.5. Usability Requirements

Responsive, as the system is desktop-mobile enabled, Understandable, as it needs to be user friendly, Learnable, as it needs to be fast used and Attractive, as users need to wish to use it to make the fast food process faster.

3.2. Design and Architecture

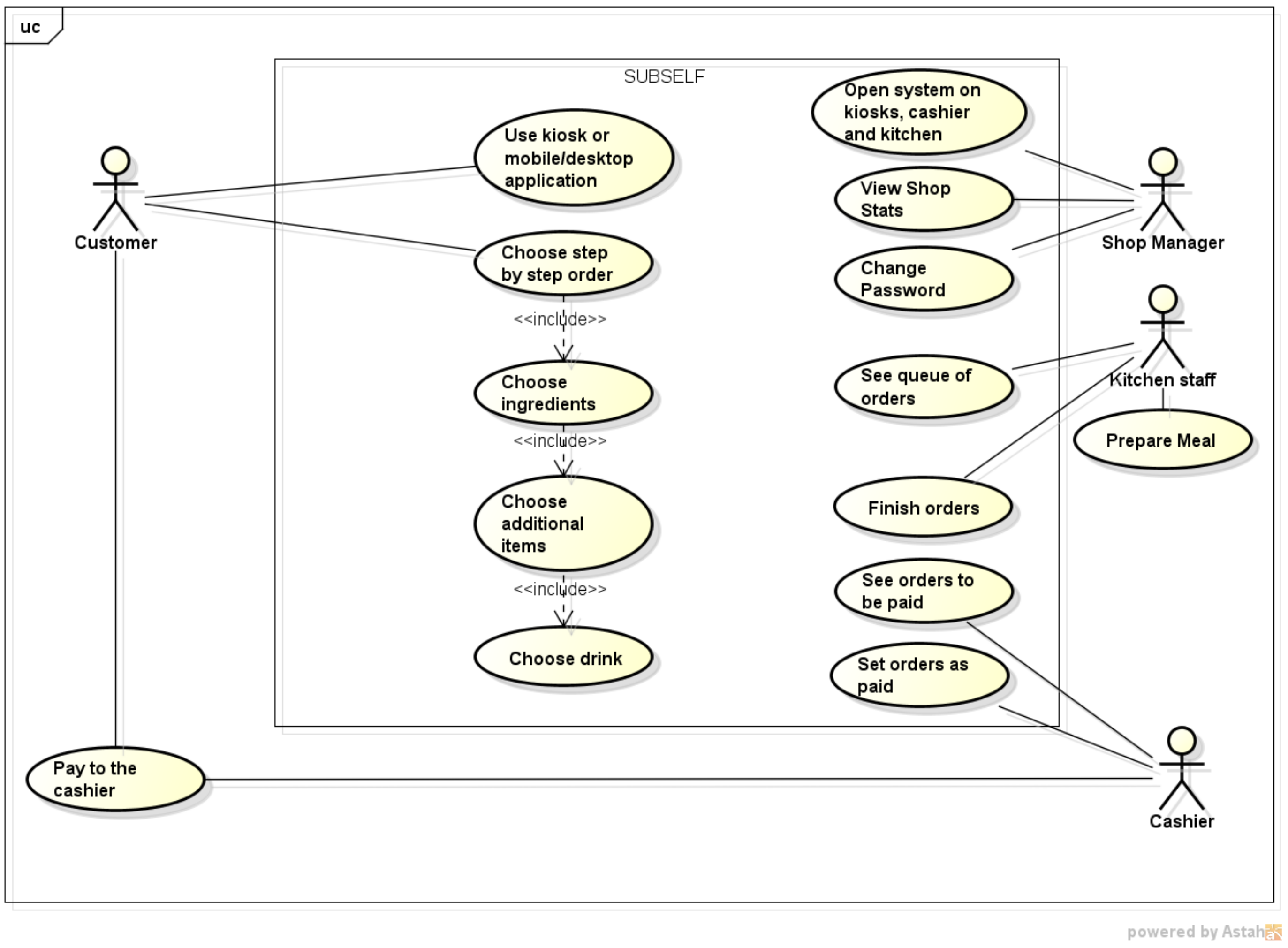
Model View Controller was used in Subself like this: Model – Three classes: Connection Factory, the different objects classes and a class containing methods to handle the database according to the objects. View – XHTML pages to show contents. Controller – Managed Beans with all the methods that users will access through the XHTML pages. The managed beans also connect with model layer and controls view layer.



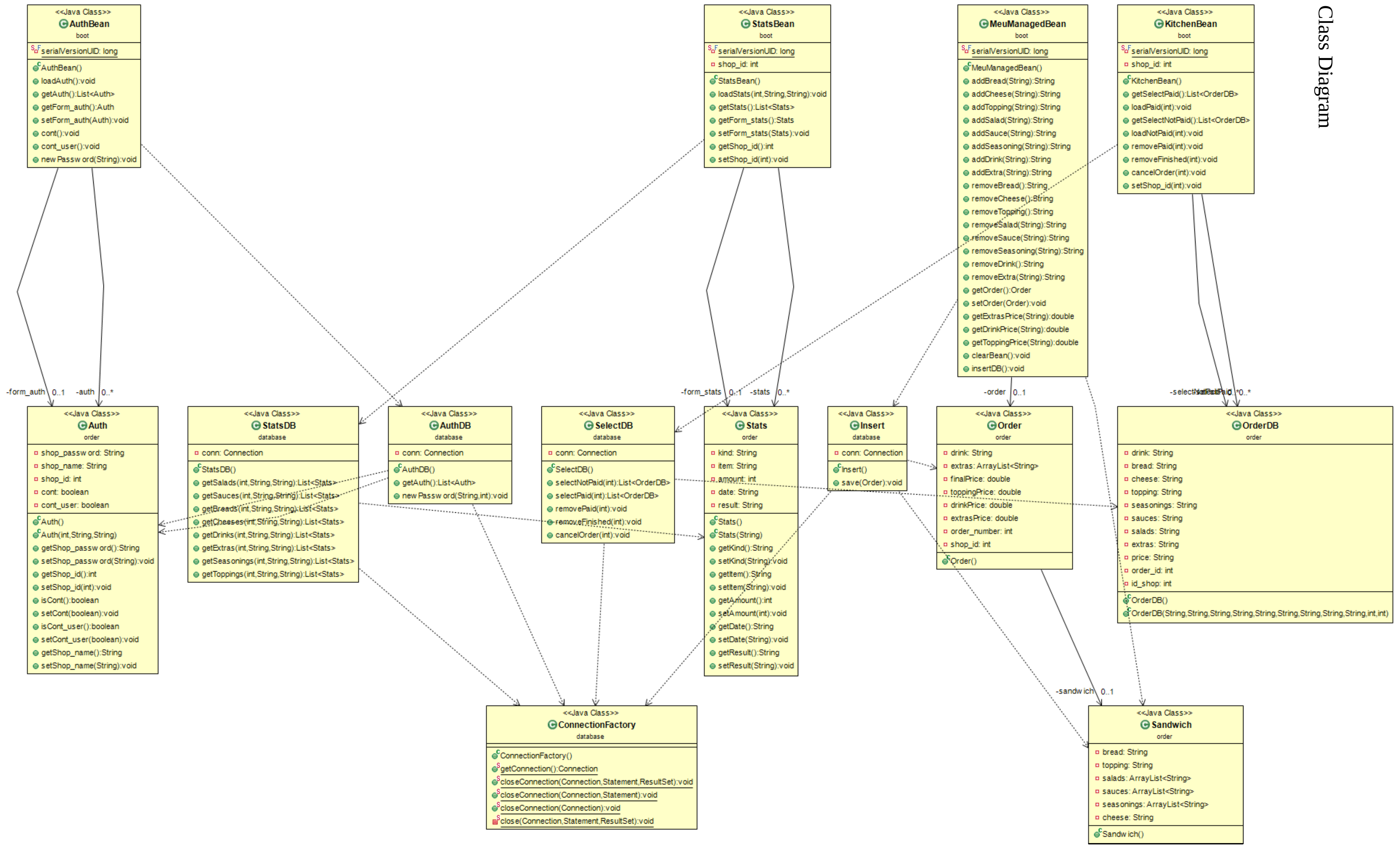
Source: http://karimdjaafar.wordpress.com/

3.2.1. UML

Use Case Diagram



Class Diagram



3.3. Implementation

I will explain the main implementation features of each module of Subself and its responsive design.

3.3.1. Subself Kiosk – Self Service application

Through ManagedBean, the ingredients are added and removed. Example:

```
public String addBread(String bread) {
          if (this.order.getSandwich().getBread() != null
                       &&
          this.order.getSandwich().getBread().equals(bread))
                return null;
          else
                this.order.getSandwich().setBread(bread);
          return null;
 }

 public String removeBread() {
          this.order.getSandwich().setBread(null);
          return null;
 }
```

Then, according to the action, the webpage render to the user on a selection list the items that he/she selected.

Also, this Bean is responsible to invoke methods in the Insert class, to insert the final order in the database when the customer is finished.

```
public void insertDB() throws Exception{
   Insert ins = new Insert();
    Order order = this.order;
   ins.save(order);
}
```

The webpages handle the bean according to this snippet:

```
<h:commandButton type="button"
          class="img-responsive btn btn-success"
          image="img/salads/Lettuce.png"
   action="#{meuManagedBean.addSalad('Lettuce')}" />
```

3.3.2. Subself Cashier / Subself Kitchen

These two modules share the same Bean, they differ in the database Select and the status of the order they display. The parameter id showed in the methods below, is the order_id. LoadPaid and LoadNotPaid select orders to show in the kitchen and cashier modules respectively.

```
public KitchenBean() {
	try {
		loadNotPaid(shop_id);
		loadPaid(shop_id);
	} catch (Exception e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
    }
public void removePaid(int id) throws Exception{
	SelectDB sel = new SelectDB();
	sel.removePaid(id);
}

public void removeFinished(int id) throws Exception{
	SelectDB sel = new SelectDB();
	sel.removeFinished(id);
}

public void cancelOrder(int id) throws Exception{
	SelectDB sel = new SelectDB();
	sel.cancelOrder(id);
    }
```

3.3.3. Subself Stats

This module allows managers to see stats accordingly to selected items and dates. The snippet below shows the StatsBean constructor. “Total of Salads” and “Total of Orders” are also available at this module. The StatsBean asks to the database class the proper MySQL Selects.

```
public StatsBean() {
	this.form_stats = new Stats();
	try {
		loadStats(shop_id, form_stats.getItem(),
		form_stats.getDate());

	}
```

3.3.4. Responsive Implementation

The snippet below shows how divs and buttons are responsive according to screen size. The properties lg, sm, and xs are related to large screens, medium screens and mobile screens respectively. The numbers are the space they represent in a grid with 12 slots.

In this example, in large screens, 12 buttons appear in a row, and only 3 on a phone screen.

```
<ui:repeat value="#{meuManagedBean.order.sandwich.salads}"
                                            var="salad">
  <div class="col-lg-1 col-sm-1 col-xs-4">
        <p>
        <h:commandButton action="#{meuManagedBean.removeSalad(salad)}"
        image="img/salads/#{salad}.png" class="btn-remove thumbnail" />
        </p>
  </div>
</ui:repeat>
```

Data Tables are also responsive:

```
<div class="table-responsive">
  <h:dataTable value="#{kitchenBean.selectNotPaid}"
  rendered="#{authBean.form_auth.cont}" var="dataItem"
  class="table table-striped table-condensed">
        <h:column>
              <f:facet name="header">
                    <h:outputText value="Order No." />
              </f:facet>
              <h:outputText value="#{dataItem.order_id}" />
        </h:column>
```

3.4. Testing and Customer Testing

The test plans consisted of selects on the database and comparison to results showed on the modules of Subself.

I asked roommates to use the Kiosk application and give me feedback of the usability.

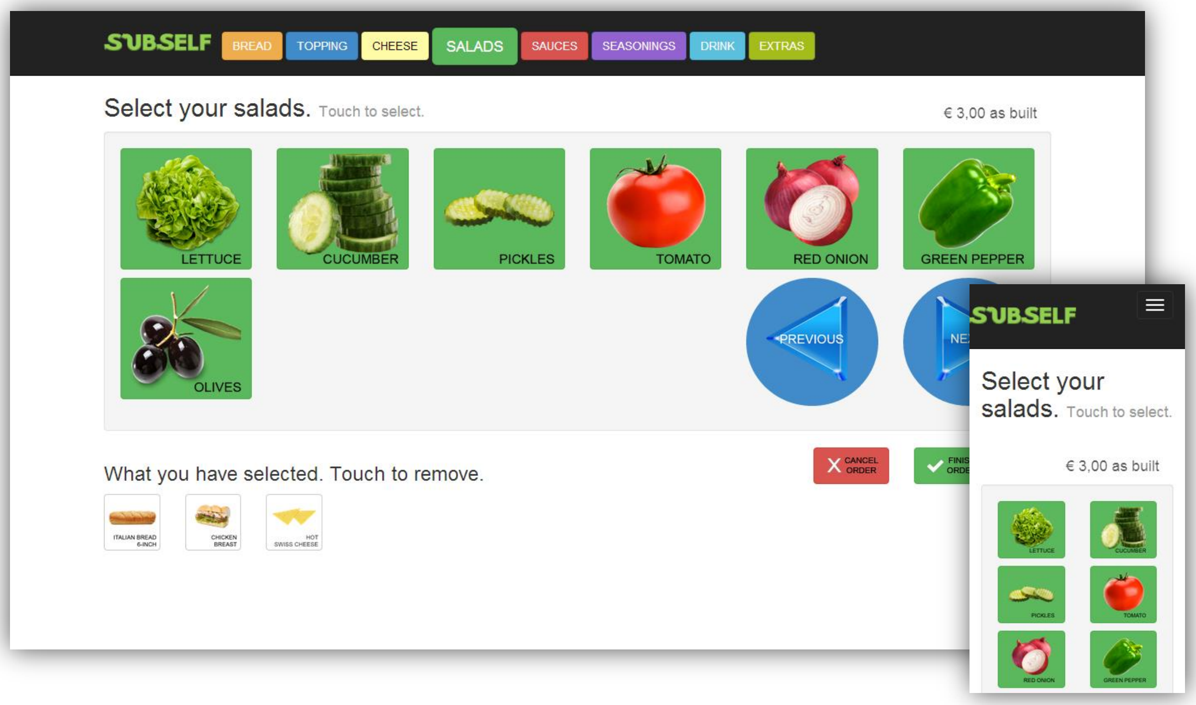
Initially, for example, to remove an item, you should touch the same item again. After user feedback, you remove an item in the selections list by touching it.

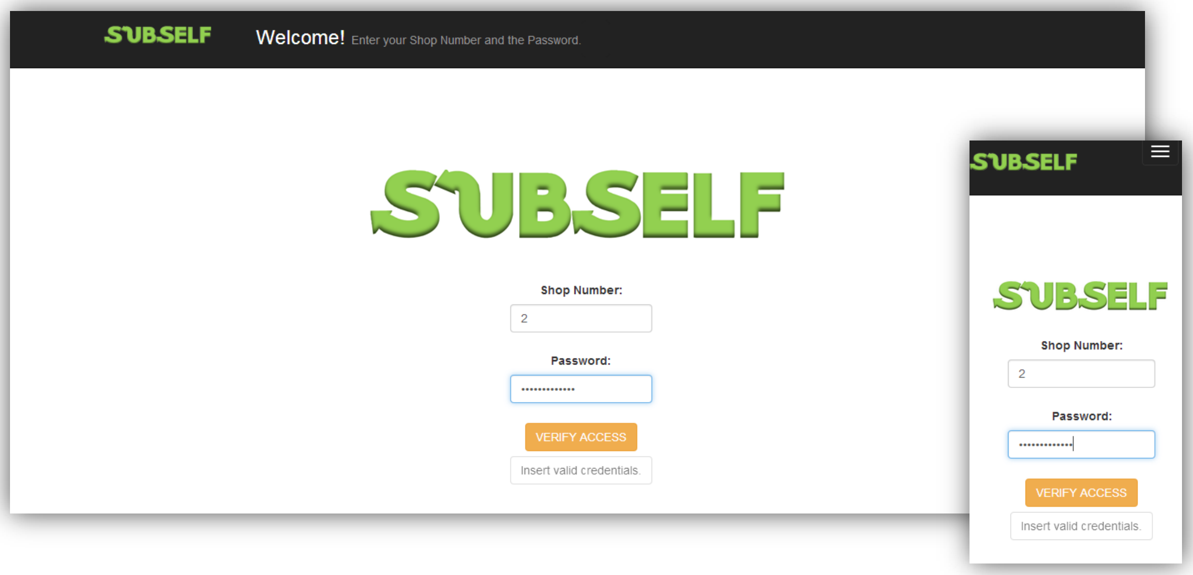
When touching it, the button gets red, to show you are going to delete it.

Authentication tests were extensive. The system rigorously asks for a shop selection in the beginning. No screens are available when the user do not select a shop.

In the Subself Admin mode, any module only display the logged shop information, and every screen always check if the admin user is correctly logged.

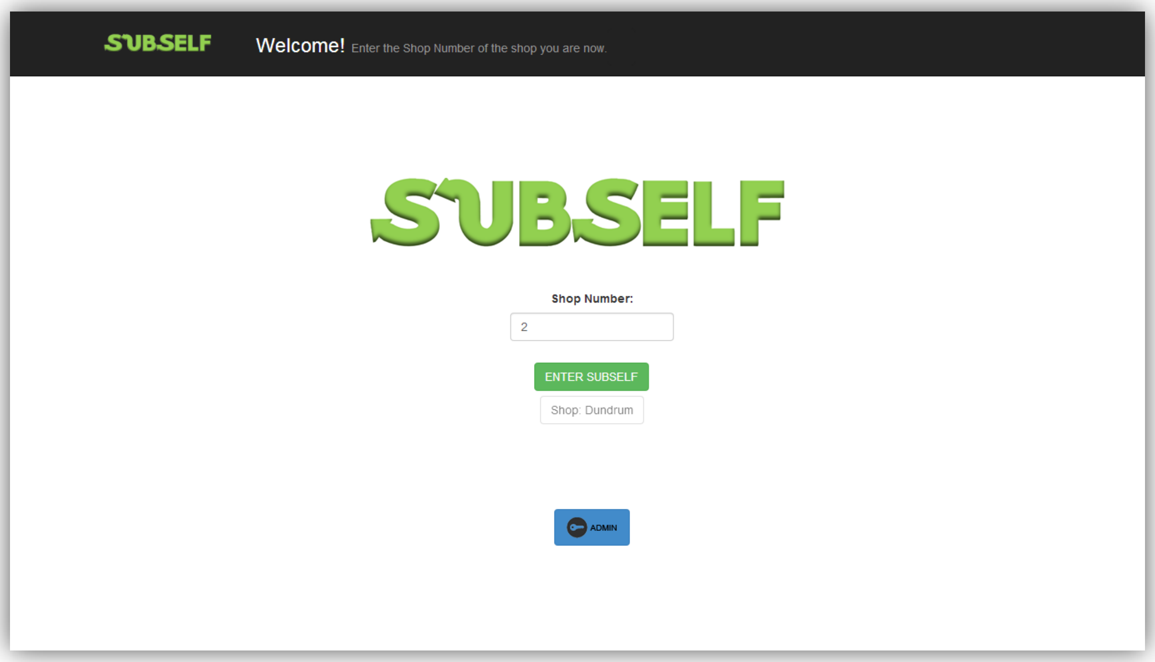
Because of the functionality of twitter bootstrap, all tests used Google Chrome as a browser. Every screen also was tested on a smaller window to show its responsiveness. Examples:



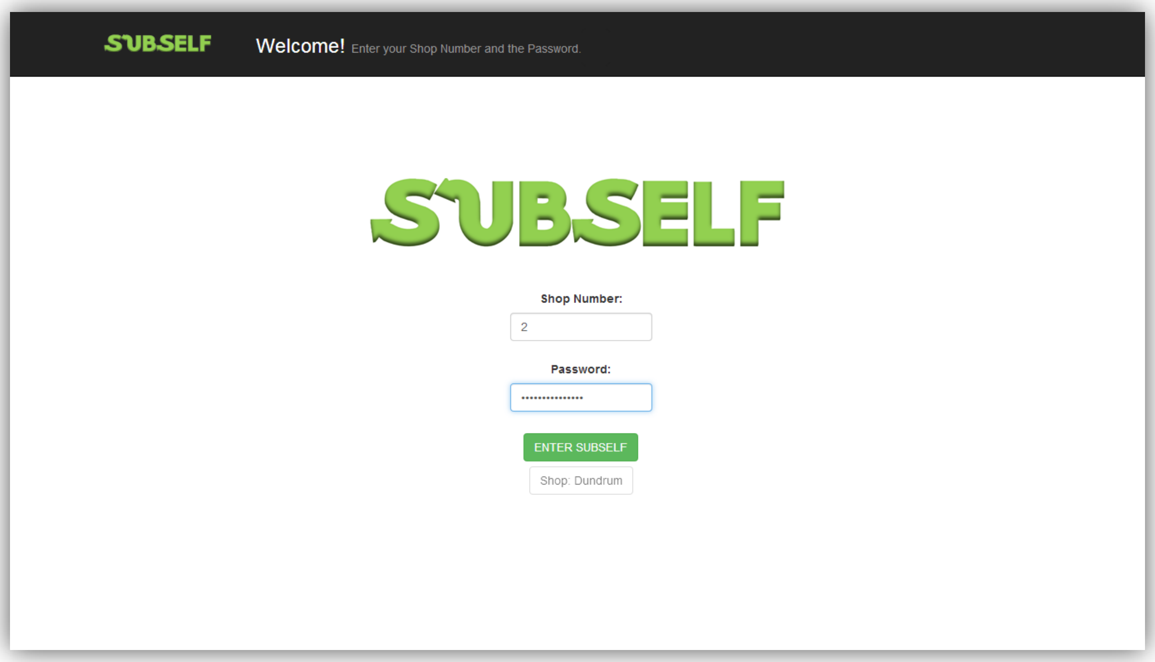

SUBSELF
Welcome! Enter your Shop Number and the Password.
Shop Number:
2
Password:
VERIFY ACCESS
Insert valid credentials.

3.5. GUI (Graphical User Interface)

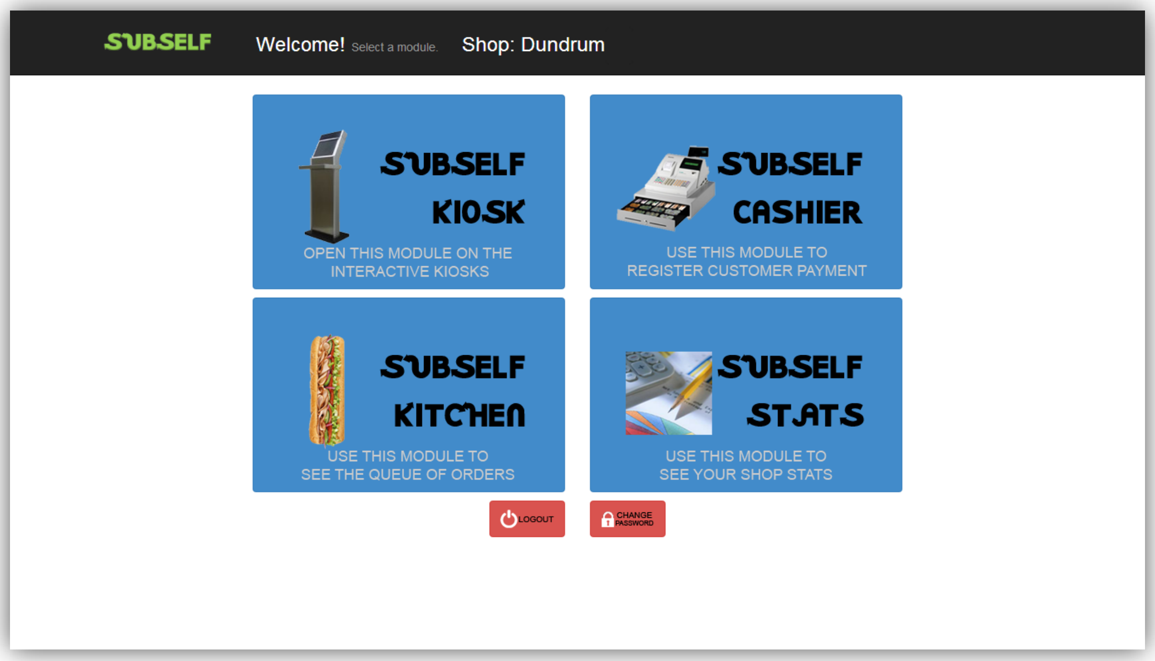
Twitter Bootstrap was used and customised for this project. Screenshots are shown below with a description of the screen.



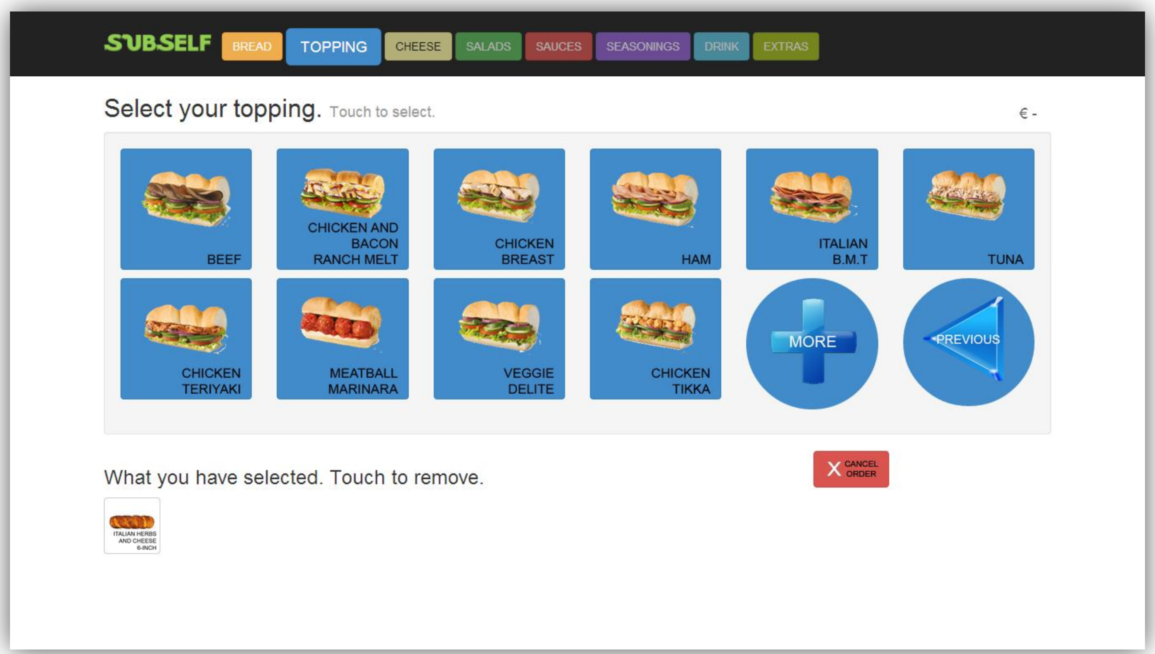
User login page. It only asks for a shop number.



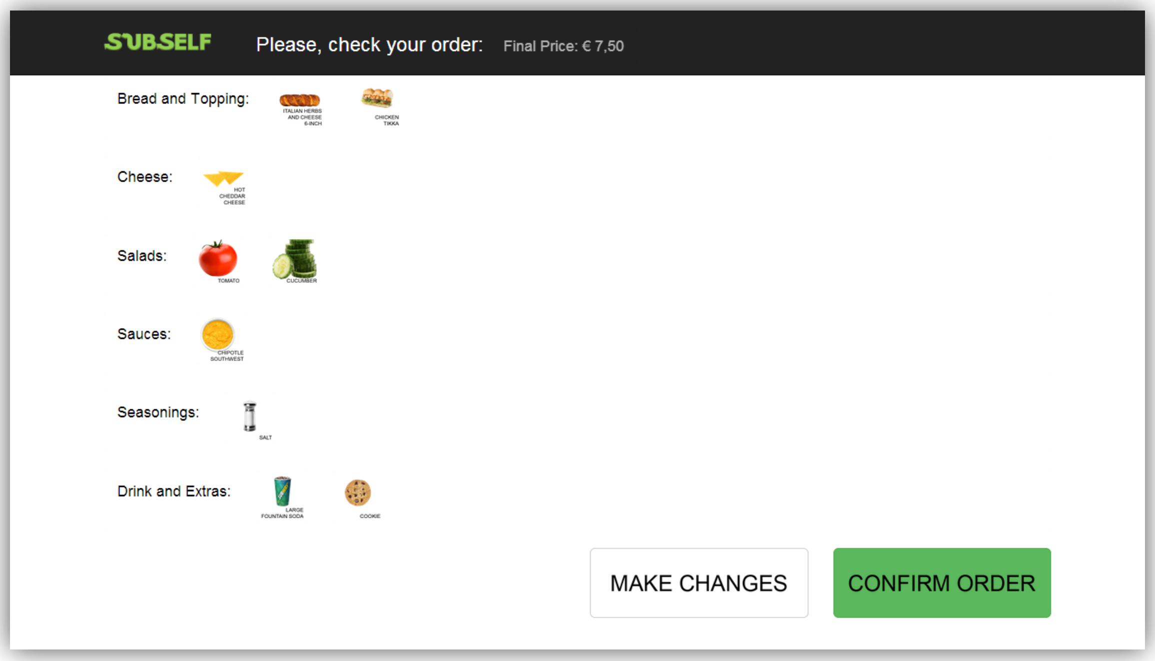
Shop login page. Subself Admin.


SUBSELF
Welcome! Select a module.
Shop: Dundrum
SUBSELF KIOSK
OPEN THIS MODULE ON THE INTERACTIVE KIOSKS
SUBSELF CASHIER
USE THIS MODULE TO REGISTER CUSTOMER PAYMENT
SUBSELF KITCHEN
USE THIS MODULE TO SEE THE QUEUE OF ORDERS
SUBSELF STATS
USE THIS MODULE TO SEE YOUR SHOP STATS
LOGOUT
CHANGE PASSWORD

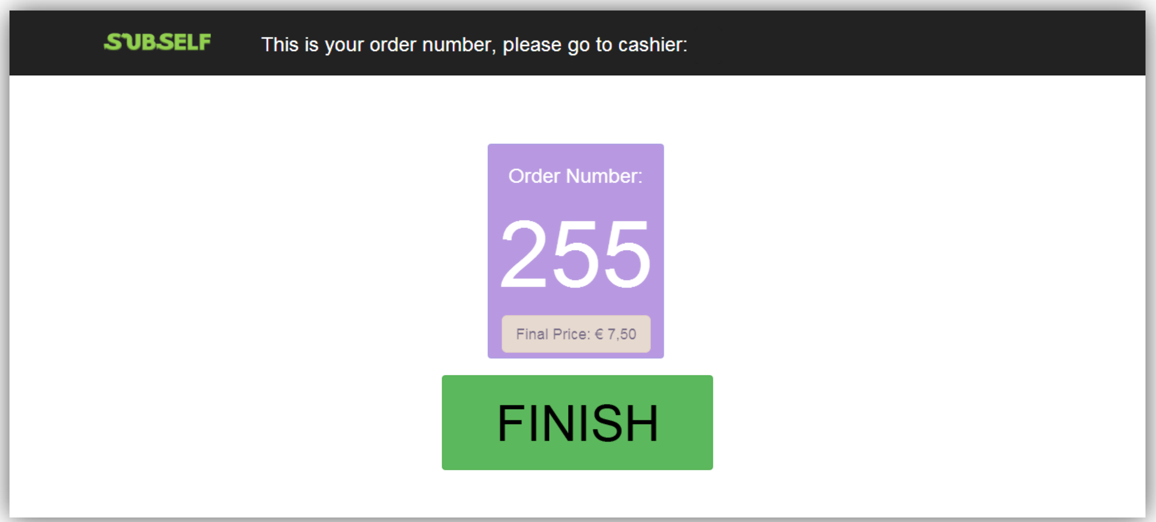
Subself Admin. Module selection page.



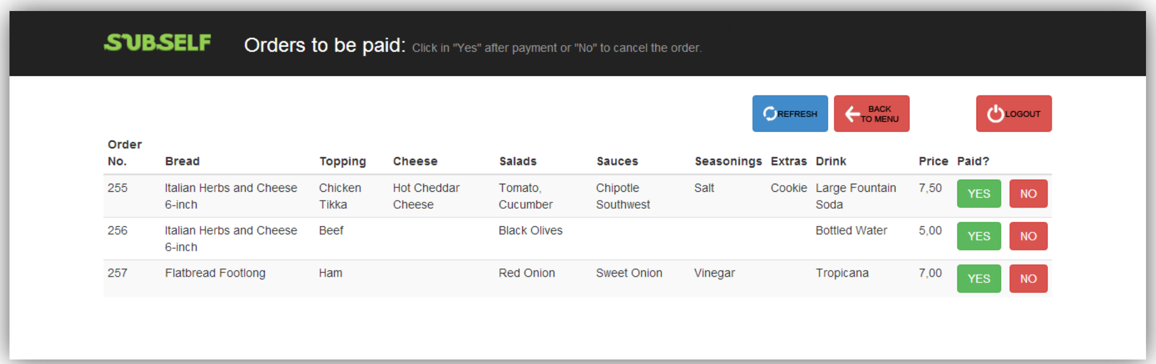
Toppings selection page.



Order confirmation.



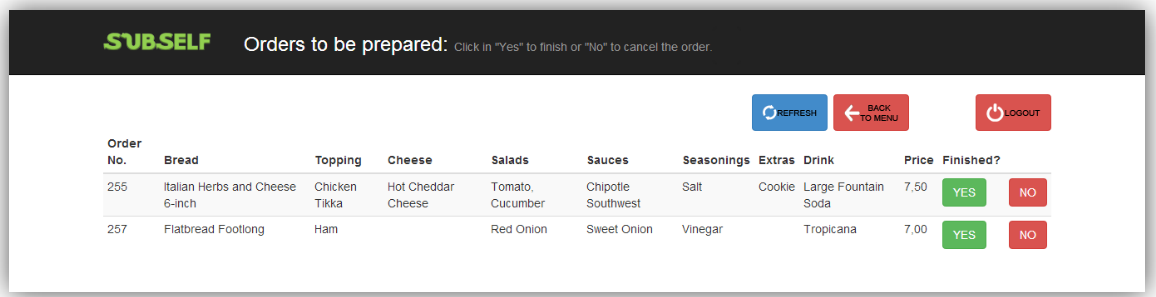
Order number.



Orders to be paid: Click in "Yes" after payment or "No" to cancel the order.

Order No.	Bread	Topping	Cheese	Salads	Sauces	Seasonings	Extras	Drink	Price	Paid?	
255	Italian Herbs and Cheese 6-inch	Chicken Tikka	Hot Cheddar Cheese	Tomato, Cucumber	Chipotle Southwest	Salt	Cookie	Large Fountain Soda	7,50	YES	NO
256	Italian Herbs and Cheese 6-inch	Beef		Black Olives				Bottled Water	5,00	YES	NO
257	Flatbread Footlong	Ham		Red Onion	Sweet Onion	Vinegar		Tropicana	7,00	YES	NO

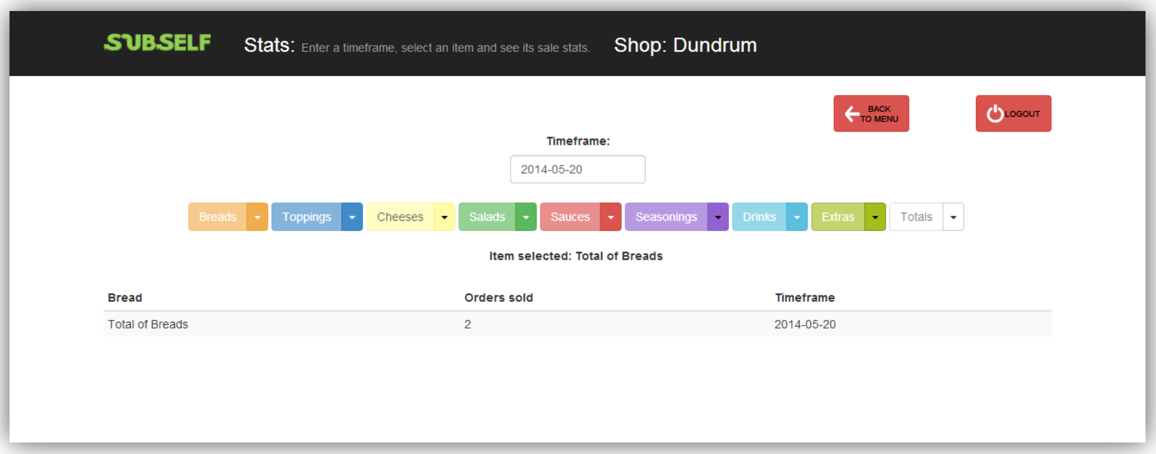
Subself Cashier.



Orders to be prepared: Click in "Yes" to finish or "No" to cancel the order.

Order No.	Bread	Topping	Cheese	Salads	Sauces	Seasonings	Extras	Drink	Price	Finished?
255	Italian Herbs and Cheese 6-inch	Chicken Tikka	Hot Cheddar Cheese	Tomato, Cucumber	Chipotle Southwest	Salt	Cookie	Large Fountain Soda	7,50	YES NO
257	Flatbread Footlong	Ham		Red Onion	Sweet Onion	Vinegar		Tropicana	7,00	YES NO

Subself kitchen.



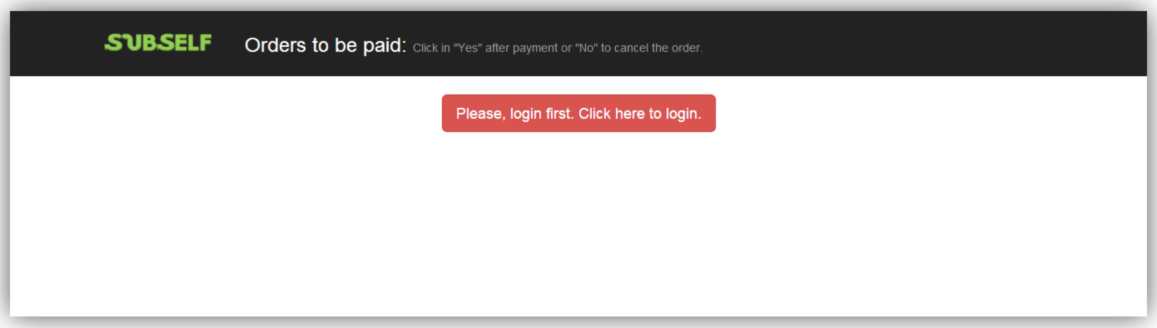
Stats: Enter a timeframe, select an item and see its sale stats. Shop: Dundrum

Timeframe: 2014-05-20

Item selected: Total of Breads

Bread	Orders sold	Timeframe
Total of Breads	2	2014-05-20

Subself Stats.



Orders to be paid: Click in "Yes" after payment or "No" to cancel the order.

Please, login first. Click here to login.

No user logged trying to access Subself Cashier.

4. Conclusions and Further Development

This report discussed the technical and also user related characteristics of Subself. I hope the reader may have understood the differences between Subself and Subself Admin with its administrator modules.

For future development, I think this system could generate the images and new items according to user insertion. They could also create their screens and the system could become a framework for “create it yourself” restaurants.

The development had some hurdles regarding college exams, but JSF was a smart choice for me.

In the beginning, Twitter Bootstrap and JSF was something totally new, as I could not find examples on the Internet to help me, but in the end both work together very well. JSF tags accept bootstrap CSS classes and it was very helpful.

5. References

JSF Tutorials (Accessed between 09/2013 – 05/2014):

- http://balusc.blogspot.ie/2011/01/jsf-20-tutorial-with-eclipse-and.html
- http://balusc.blogspot.ie/2006/06/communication-in-jsf.html

Others:

- Object Aid Eclipse Plugin and Astah Community for Diagrams
- www.subway.ie, www.subway.com, www.subway.com.br for pictures
- Twitter Bootstrap 3
- Eclipse Kepler
- Glassfish 4
- JSF 2
- Google Chrome

6. Appendix

6.1. Project Proposal

Project Proposal

SELF SERVICE SYSTEM FOR SANDWICH STORES

Anderson Simiscuka, 13115642, Anderson.simiscuka@student.ncirl.ie

BSc (Hons) in Business Information Systems

September 30th 2013

1. Objectives

The creation of a system in which people can create their sandwiches step-by-step at sandwich stores without the need of waiting in a long queue and talking to an attendant.

This system will allow the customer to choose between different types of bread, salad, toppings, sauces, Drink selection, Combo selection, Payment to the machine or print a receipt and take to the cashier.

The system can be displayed in different languages, so communication problems between customer and attendant, in cities with people from many different countries, could be solved.

The system will contain another application to the kitchen staff, where they can see the orders and queue.

2. Background

Restaurants like the Subway franchise generally are able to have only one queue of customers, and it takes more time than having, for example, three terminals where three customers could order their sandwiches at the same time.

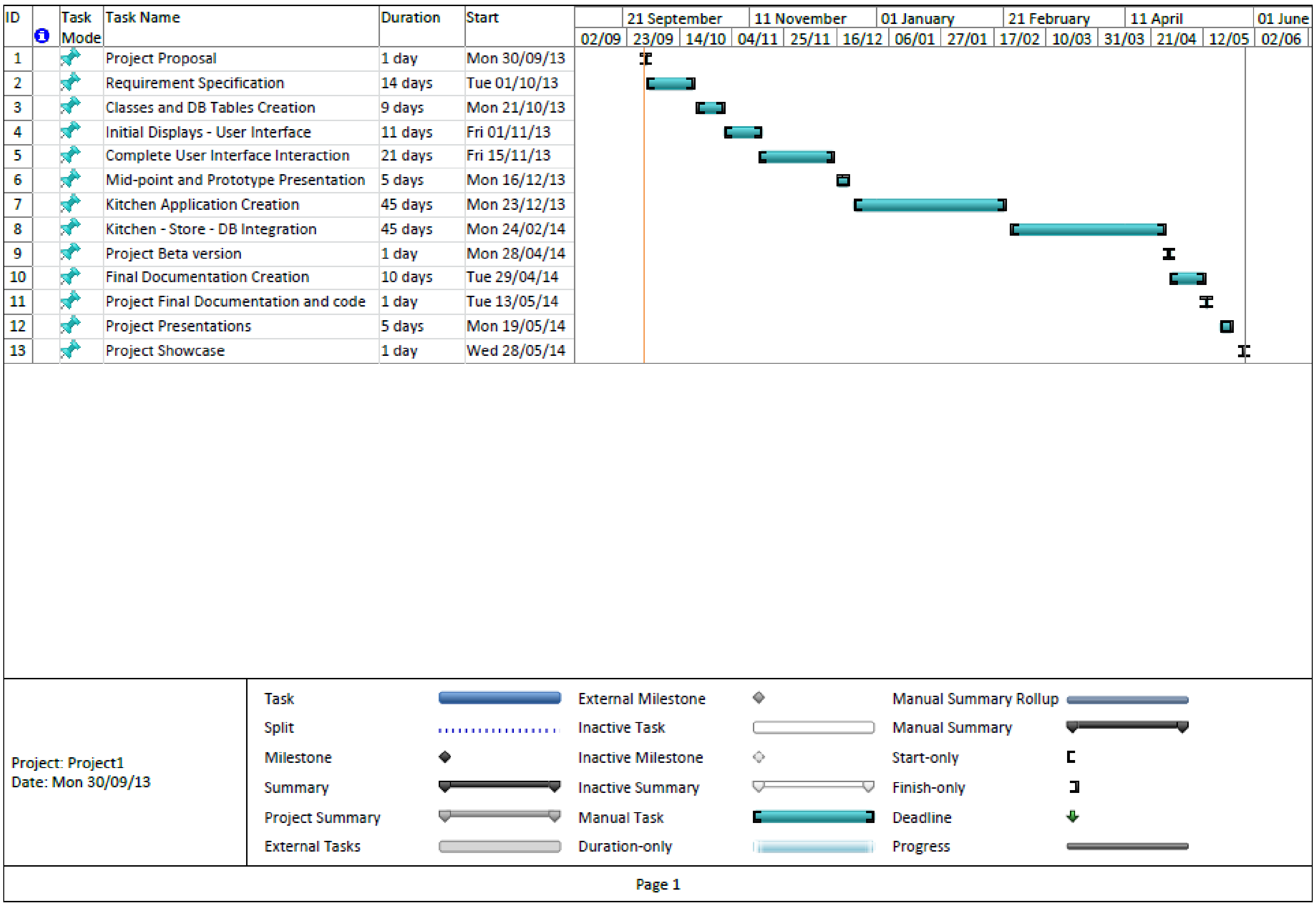
3. Technical Approach

The system will be web based and both applications, store and kitchen, will connect to the same database.

It will be implemented using Java (Object Oriented applications), Eclipse platform - JSF, Bootstrap for the display, open source database (i.e. MySQL).

The display is intended to have a similar Microsoft Metro User Interface navigation, and to be used in touch screens

4. Project Plan



ID	Task Mode	Task Name	Duration	Start
1		Project Proposal	1 day	Mon 30/09/13
2		Requirement Specification	14 days	Tue 01/10/13
3		Classes and DB Tables Creation	9 days	Mon 21/10/13
4		Initial Displays - User Interface	11 days	Fri 01/11/13
5		Complete User Interface Interaction	21 days	Fri 15/11/13
6		Mid-point and Prototype Presentation	5 days	Mon 16/12/13
7		Kitchen Application Creation	45 days	Mon 23/12/13
8		Kitchen - Store - DB Integration	45 days	Mon 24/02/14
9		Project Beta version	1 day	Mon 28/04/14
10		Final Documentation Creation	10 days	Tue 29/04/14
11		Project Final Documentation and code	1 day	Tue 13/05/14
12		Project Presentations	5 days	Mon 19/05/14
13		Project Showcase	1 day	Wed 28/05/14

Project: Project1
Date: Mon 30/09/13

Task		External Milestone	◆	Manual Summary Rollup	
Split		Inactive Task		Manual Summary	
Milestone	◆	Inactive Milestone	◇	Start-only	[
Summary		Inactive Summary		Finish-only	]
Project Summary		Manual Task		Deadline	⇩
External Tasks		Duration-only		Progress	

Page 1

5. Technical Details

Java (Object Oriented applications), Eclipse platform - JSF, Bootstrap for the display, open source database (i.e. MySQL).

6. Evaluation

Extensive black-box and white-box testing. User forms with opinions about the interface. Connection and data insertion testing.

7. Consultation with Project Specialisation Coordinator

Anu Sahni and Jonathan McCarthy. Positive feedback of the main idea.

8. Consultation with Academic Staff

The project Specialisation Coordinator will point you to a suitable staff member for consultation. Please include the name of the second academic staff member consulted and a summary of their feedback.

9. Proposed Supervisor

Names of academic staff member that has agreed to act as a supervisor for this project.

____

Signature of student and date

6.2. Requirements Specification

BSHC4, BSHCE4, BSHBIS4, BSHBISE4

Requirements Specification (RS)

Anderson Augusto Simiscuka
10/20/2013

Requirements Specification (RS)

Document Control

Revision History

Date	Version	Scope of Activity	Prepared	Reviewed	Approved
13/10/2013	1	Create			
20/10/2013	2	Update			

Distribution List

Name	Title	Version
Kyra McKenna	Lecturer / Project Supervisor	
Jonathan McCarthy	Lecturer	

Related Documents

Title	Comments
Title of Use Case Model	
Title of Use Case Description	

1 User

Purpose

The purpose of this document is to set out the requirements for the development of a Self Service System for Sandwich Stores.

The intended customers are stores that sell Sandwiches that the final customer choose each ingredient.

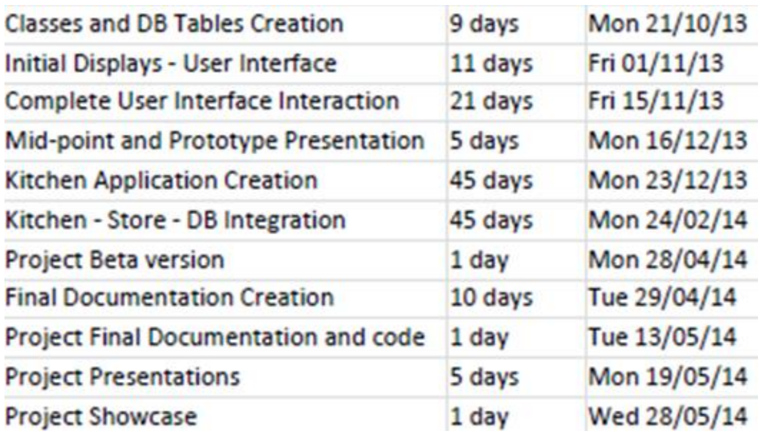
Project Scope

The scope of the project is to develop a Self Service System for Sandwich Stores. The system shall allow the customer to choose between different types of bread, salad, toppings, sauces, Drink selection, Combo selection, Payment to the machine or print a receipt and take to the cashier.

The system can be displayed in different languages, so communication problems between customer and attendant, in cities with people from many different countries, could be solved.

The system will contain another application to the kitchen staff, where they can see the orders and queue.

Schedule:



Classes and DB Tables Creation	9 days	Mon 21/10/13
Initial Displays - User Interface	11 days	Fri 01/11/13
Complete User Interface Interaction	21 days	Fri 15/11/13
Mid-point and Prototype Presentation	5 days	Mon 16/12/13
Kitchen Application Creation	45 days	Mon 23/12/13
Kitchen - Store - DB Integration	45 days	Mon 24/02/14
Project Beta version	1 day	Mon 28/04/14
Final Documentation Creation	10 days	Tue 29/04/14
Project Final Documentation and code	1 day	Tue 13/05/14
Project Presentations	5 days	Mon 19/05/14
Project Showcase	1 day	Wed 28/05/14

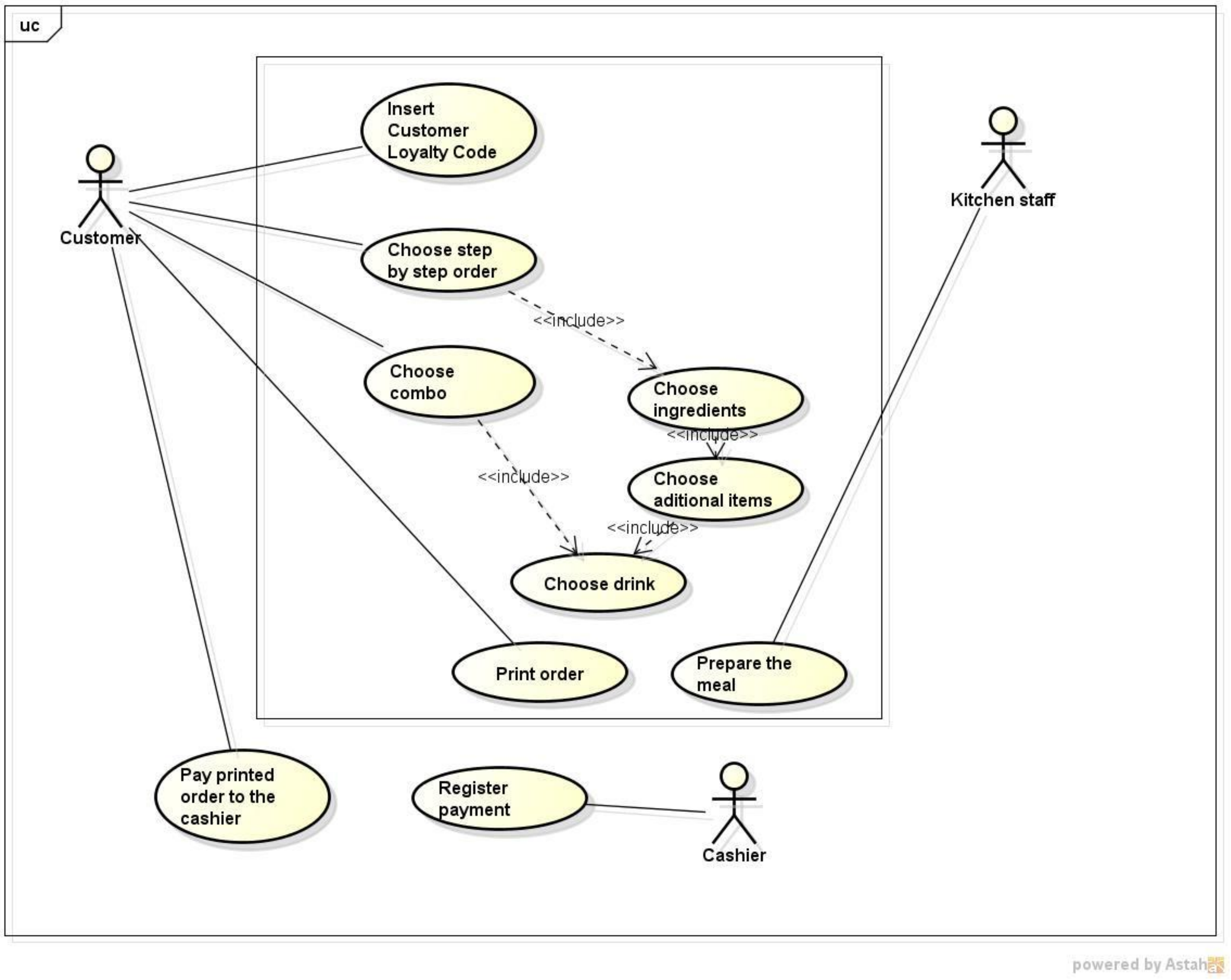
2 User Requirements Definition

Faster queues
More people being attended at the same time
Possibility to communicate in more than a language

3 Requirements Specification

Functional requirements

Use Case Diagram



Requirement 1: Insert Customer Loyalty Code

Description & Priority

When the customer start using the touch screen, selecting the language, it will be possible to register the customer code for future discounts or promotions. Low priority.

Use Case

Scope

How the customer will insert the loyalty code.

Description

In the first screen, the customer will choose between inserting or not the code. If code insertion selected, a keyboard will appear, and the code can be inserted. The information of the meal will go to the customer database and it can be used for discounts in the website, vouchers and promotion.

Use Case Diagram

Actor Customer and use case Insert Customer Loyalty Code.

Flow Description

Precondition

Customer started to interact to the touch screen.

Activation

This use case starts when a Customer selects to insert a Loyalty Code.

Main flow

The system identifies the option inserted

The Customer insert its code if decided to do so

The system upload this information to the database

Termination

The system presents the next screen (Type of meal)

Post condition

The system goes into a wait state

Requirement 2: Choose Step by Step Order

Description & Priority

Customer can choose between a regular combo or customise its order. In the Step by Step option, ingredients and additional items will be selected by the customer. High priority.

Use Case

Scope

Customer will select the ingredients of the meal and Drink.

Description

After selecting Step by Step Order, the customer needs to choose ingredients, additional items like crisps or cookies and the drink.

Use Case Diagram

Actor customer and use case Choose Step by Step Order and its inclusions.

Flow Description

Precondition

Step by Step Order selected on the screen

Activation

This use case starts when a Customer selected Step by Step Order

Main flow

The system identifies the option selected

The Customer selects the ingredients, then if wants additional items and finally the drink

Termination

The system presents the next screen (Print order)

Post condition

The system goes into a wait state

Requirement 3: Choose Combo

Description & Priority

Customer can choose between a regular combo or customise its order. In the Combo option, the customer can see regular sandwiches with included drinks and additional items. According to the day, combos can change. High priority.

Use Case

Scope

Customer will select the meal and Drink.

Description

After selecting Combo option, the customer needs to choose the combo.

Use Case Diagram

Actor customer and use case Choose Combo and its inclusions.

Flow Description

Precondition

Combo option selected on the screen

Activation

This use case starts when a Customer selected Combo option

Main flow

The system identifies the option selected

The Customer selects the combo

Termination

The system presents the next screen (Print order)

Post condition

The system goes into a wait state

Requirement 4: Print Order and Payment

Description & Priority

Customer will finalise the process printing the order with the price and items selected. High priority.

Use Case

Scope

Customer will finish the meal selection and print the order.

Description

The printed order needs to be presented to the cashier that will accept card or money. The kitchen already received the order from the system.

Use Case Diagram

Actors customer and cashier and use cases Pay Printed Order to the cashier and Register Payment.

Flow Description

Precondition

All previous screens completed.

Activation

This use case starts when a Customer finishes the order

Main flow

The system identifies the meal and its price

The Customer finishes the process and accept the price

The system prints the order and send the order to the kitchen

The Customer pays to the cashier

Termination

The system prints the order.

Post condition

The system goes into a wait state

Requirement 5: Prepare the Meal

Description & Priority

Kitchen staff have a system to see the orders. High priority.

Use Case

Scope

Customer will finish the meal selection and the kitchen see the order on a screen.

Description

The order is going to be prepared according to what was selected by the customer.

Use Case Diagram

Actor Kitchen Staff and use case Prepare the Meal.

Flow Description

Precondition

Order completed.

Activation

This use case starts when a Customer finishes the order

Main flow

The system prints the order and send the order to the kitchen

The Kitchen Staff prepares the meal and bring to the customer

Termination

The meal is prepared.

Post condition

The customer receives its meal.

Non-Functional Requirements

Specifies any other particular non-functional attributes required by the system. Examples are provided below.**Remove the requirement headings that are not appropriate to your project.**

Performance/Response time requirement

The system needs to be fluid and fast.

Availability requirement

At least one terminal needs to be working in the store.

Security requirement

Customer code needs to travel secure to the network.

Maintainability requirement

Combo prices change according to the day of the week.

Extendibility requirement

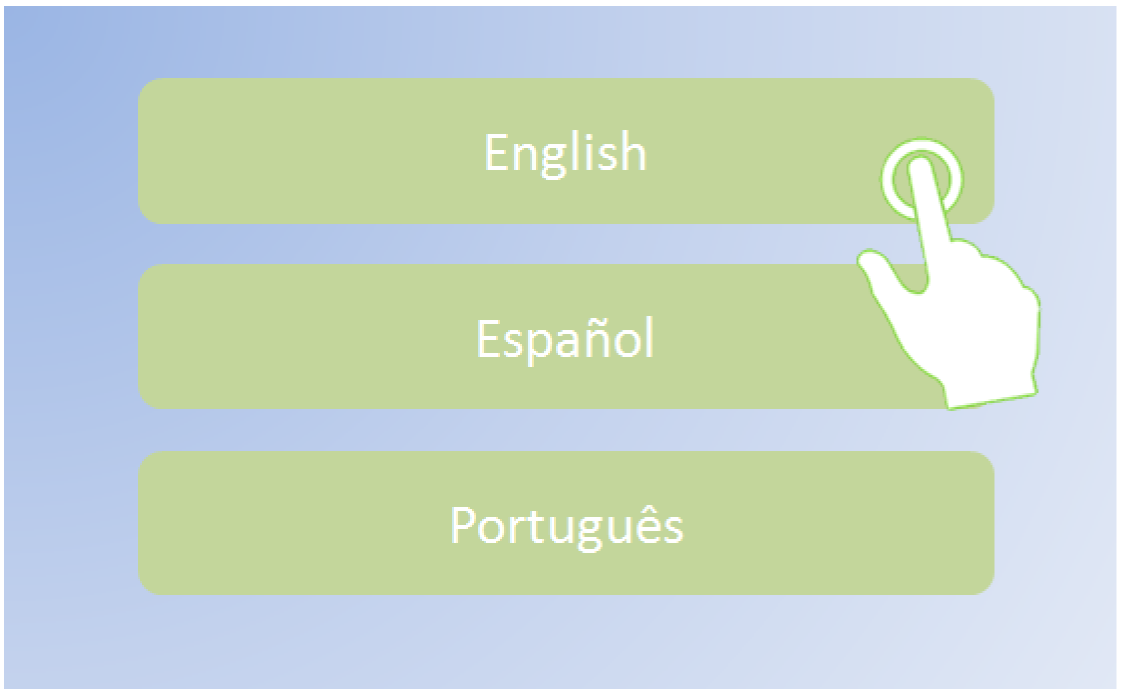
The same store has more one terminal that connect to the same database and kitchen system.

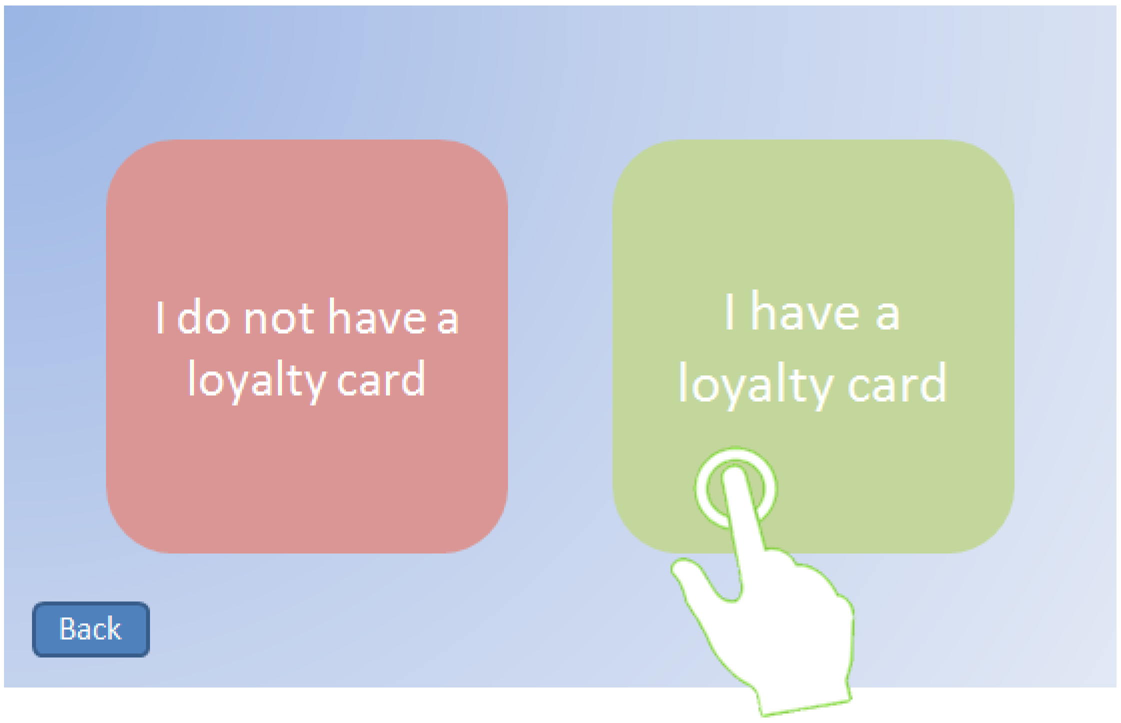
Reusability requirement

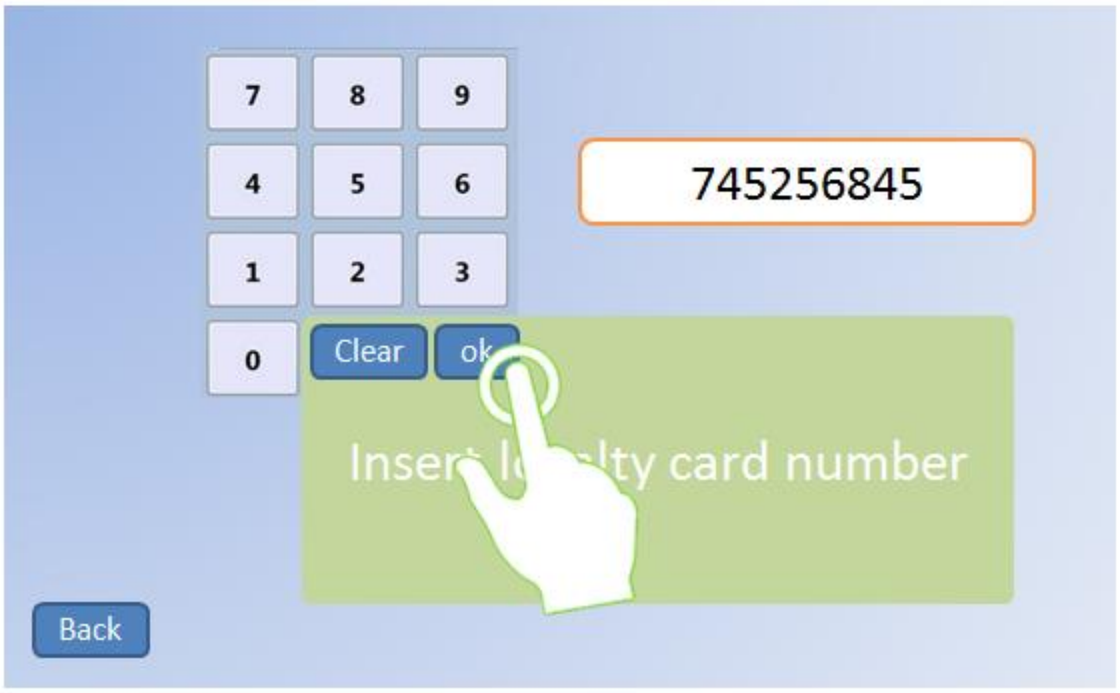
The system is the same in different terminals

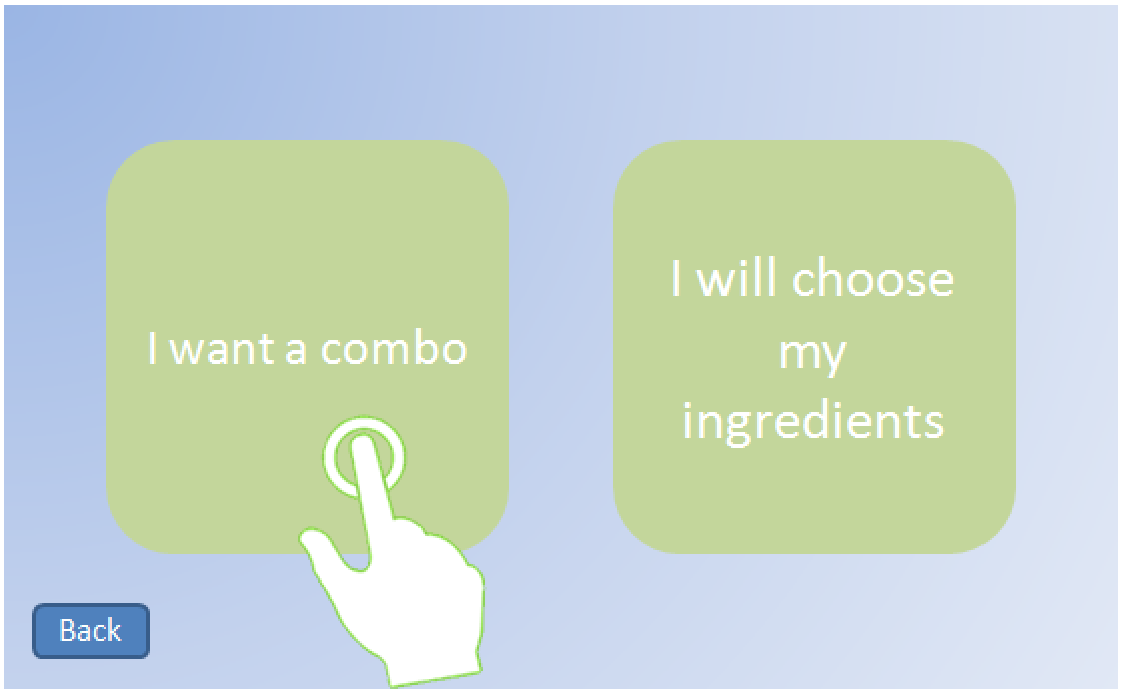
4 Interface requirements

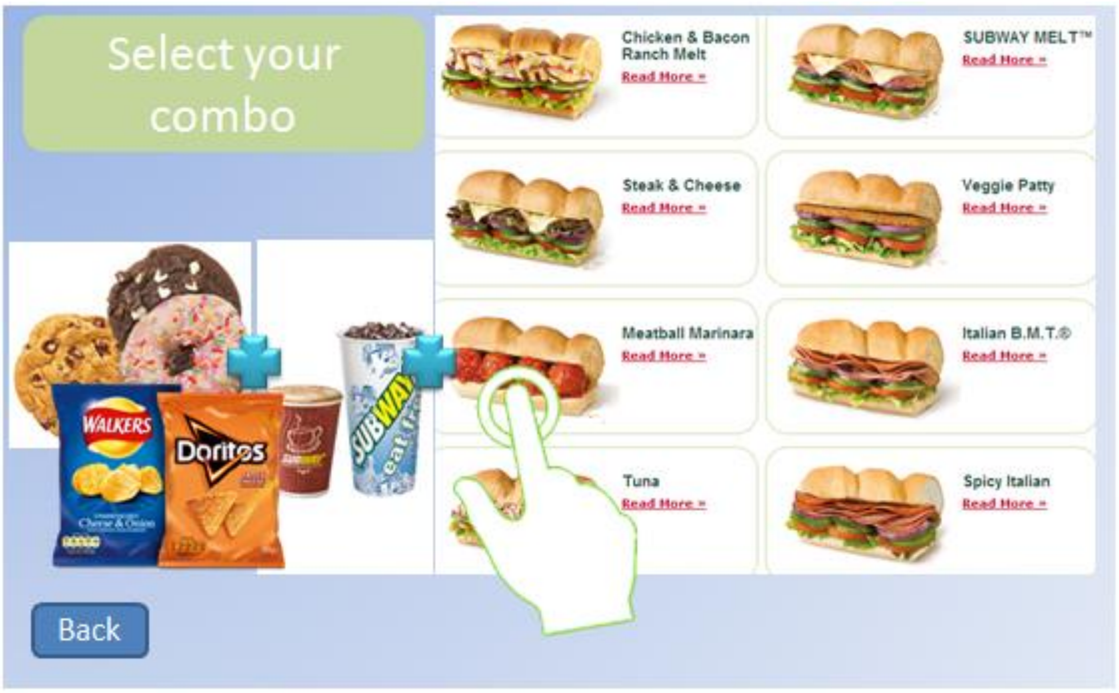
GUI

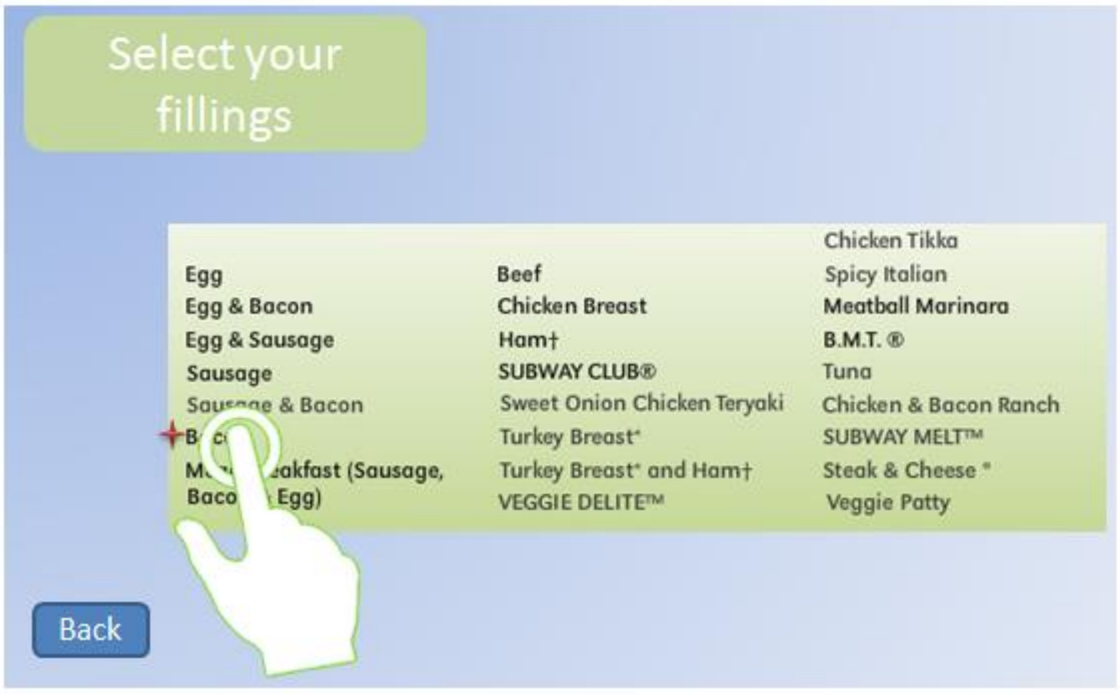





7
8
9
4
5
6
1
2
3
0
Clear
ok
745256845
Insert loyalty card number
Back


I want a combo
I will choose my ingredients
Back





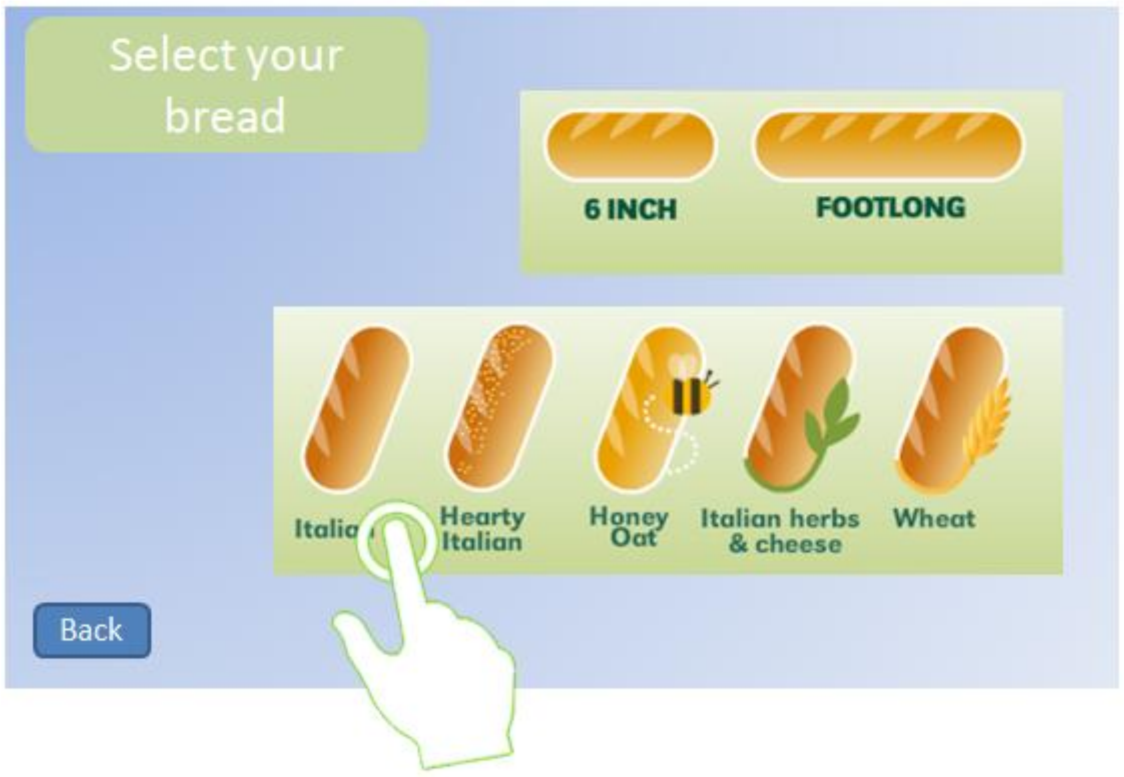
Select your fillings

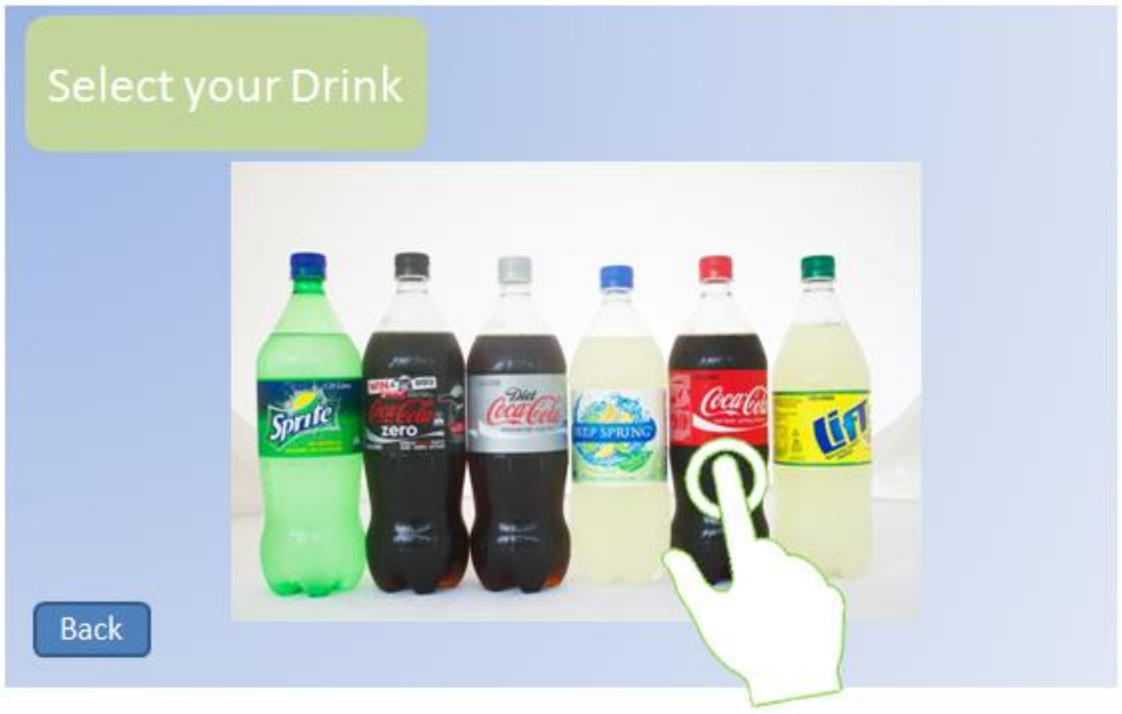
Egg
Egg & Bacon
Egg & Sausage
Sausage
Sau[illegible] & Bacon
B[illegible]c[illegible]
M[illegible]eakfast (Sausage, Baco[illegible] Egg)

Beef
Chicken Breast
Ham†
SUBWAY CLUB®
Sweet Onion Chicken Teryaki
Turkey Breast*
Turkey Breast* and Ham†
VEGGIE DELITE™

Chicken Tikka
Spicy Italian
Meatball Marinara
B.M.T. ®
Tuna
Chicken & Bacon Ranch
SUBWAY MELT™
Steak & Cheese *
Veggie Patty

Back

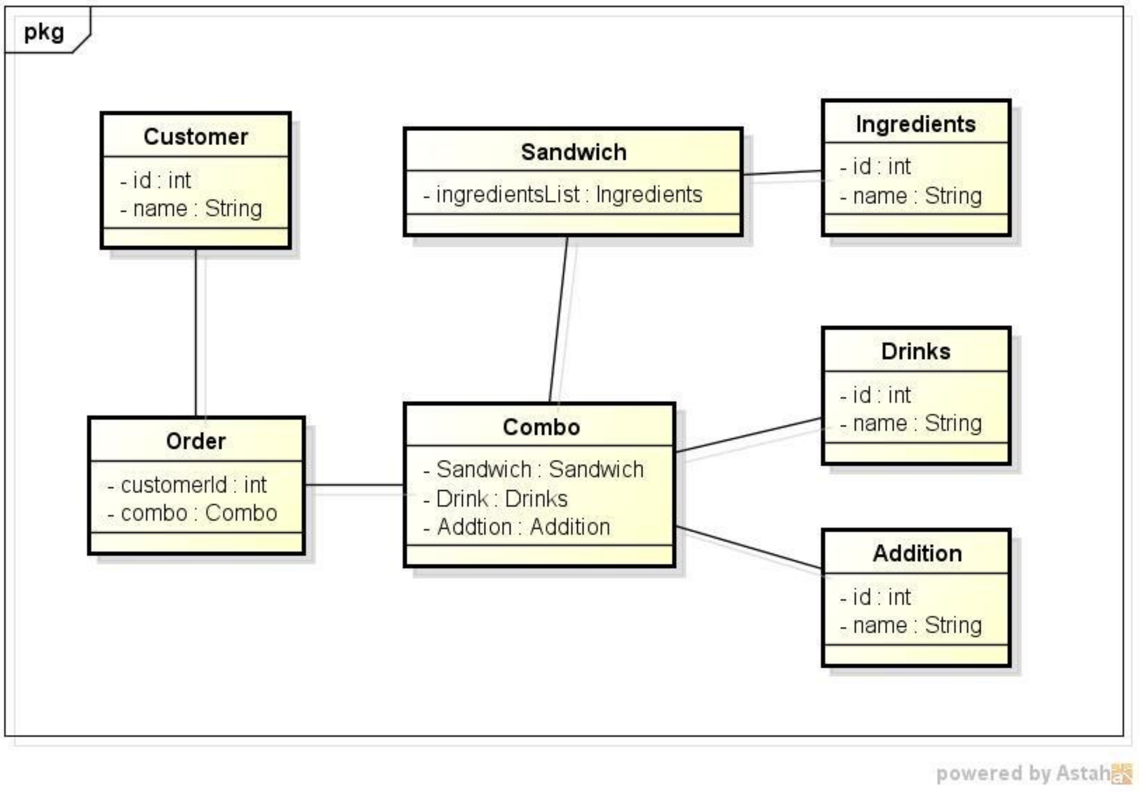




Application Programming Interfaces (API)

Printing interface

5 System Architecture



6 System Evolution

The system can accept new ingredients, combos and prices.

6.3. Reflective Journals

6.3.1. September

Reflective Journal

Student name: Anderson Augusto Simiscuka

Programme (e.g., BSc in Computing): BSc (Hons) in Business Information Systems

Month: September

1. My Achievements

This month, I was able to think of an idea that became my Project Proposal after that. I have focused on the first layout ideas of the Self Service System for Sandwich Stores.

The display is intended to have a similar Microsoft Metro User Interface navigation, and to be used in touch screens.

My contributions to the projects included defining the core ideas, writing down the first features and functionalities.

2. Reflection

I felt, it worked well to create the system concept and defining its functionalities.

3. Intended Changes

Next month, I will try to create diagrams, wireframes, and start coding the first functionalities.

6.3.2. April

Reflective Journal

Student name: Anderson Augusto Simiscuka

Programme (e.g., BSc in Computing): BSc (Hons) in Business Information Systems

Month: April

1.My Achievements

New modules are available. Kitchen and Cashier modules. User Authentication and Administrator Authentication.

Kitchen and Cashier modules retrieve data from the database according to their status and allow the cashier to insert payment on new orders and the kitchen employees to finish orders.

User Authentication is intended for mobile or desktop users to select a store, make their order and pick it up.

Administrator Authentication is intended to be used on shops. It also contains the Kiosk module. The Kiosk module is the same as the application of Users, where they create their sandwiches, however, to be used on kiosks/interactive totems in the shops.

2.Reflection

Those modules were challenging but JSF showed a nice tool to be used to create data tables and for user administration.

3.Intended Changes

Next month, I intend to create a new module, the Stats module. Managers will be able to see each item consumption using different timeframes.

CD

SUBSELF