School of Electronic Engineering and Computer Science
Queen Mary University of London

INTERIM REPORT 2020-21

Programme of Study:
Computer Science BSc(Hons)

Project Title:
allPOS – cloud based Electronic Point of Sale System

Supervisor:
Paulo Oliva

Student Name:
Anton Petrov Nyagolov

Queen Mary
University of London

Final Year
Undergraduate Project 2020/21

Date: 14 Nov 2020

Table of Contents

List of Figures

List of Tables

1. Introduction

During the past five years of residing within the UK, I have been in full-time employment, essentially working as a Head of Coffee in several companies around London. Head of Coffee is a management role in speciality coffee shops, including many administrative tasks such as the management of everyday operations, staff rota creation, stock management, and others. Furthermore, managers usually have a larger number of responsibilities than employees. They not merely should work and operate the same service as other employees, but managers likewise have to:

- Keep track of stock raw goods
- Order raw goods when low on stock
- Calculating raw material inventory
- Inventory optimisation
- Report generation
- Employee scheduling
- Cash reports
- Dealing with Staff Holiday Request
- Analyse financial data and conclude financial reports

Those are a couple of essential management duties. During the time I worked in those commercial environments, I noticed that few organisations were managing with those matters differently. Three primary types of software are utilised across the sector to automate a portion of the tasks referenced above:

- EPOS also know as Electronic Point of Sale System
- Employee Scheduling Software
- Stock Management Software

Every one of those three frameworks is sold independently as a solitary product. Premium bundles exist, which implements a combination of them. Notwithstanding, as the name of the product suggests (premium), it is highly-priced and unaffordable for most single store retailers.
There are several reasons behind the absence of full system implementation (3 systems used by one retailer) across retail and hospitality, but the primary ones are cost and cross-platform adaptability.
My idea is to build up a device-independent application that would unravel all those issues collectively, and besides, it will provide some extra functionalities. This project will combine an EPOS, ESS and SMS system into one simple to use web-based Django application. It will include all the main features that a native software supports. Additionally, the Stock Management Software would be able to gain data from past sale patterns and automate the ordering process of crude products with insignificant administration commitment.

Currently, I am taking a module in Web Programming learning the Django Framework. My software experience is mostly school-related, implying that I have not developed any complex application previously. The main challenge for this project will be the intricacy to consolidate the three sorts of frameworks with every one of their functionalities into one web application and execute them so that they exchange information.

1.1. Objectives

The project summarises four high-level objectives that the app will consist of:

- Implemented and fully functioning Electronic Point of Sale
- Implemented and fully functioning Employee Scheduling
- Implemented and fully functioning Stock Management
- Integrated decision making functionality to automate raw goods ordering

1.2. Scope

There is an extensive list of companies who develop software and hardware such as EPOS, ESS, and SMS systems for the hospitality and retail industry. Heretofore, I was unable to distinguish an organisation that has built up every one of the three strategies together.
The software industry is a highly dynamic and growing sector. New software is released every day, tackling various issues in innovative ways. Therefore, it ought to be considered that this project is undertaken under a tight time frame and resources. As a result, 5 of the most rated EPOS, ESS, and SMS available on the market were researched, studied and analysed.

Some of the EPOS companies provide custom hardware, designed specifically for their software. For instance, EPOS software is extendable to work with barcode scanners, cash drawers, receipt printers, card readers, voice control stations and several more.
Nonetheless, this research only takes into account the product execution of the systems discussed in the introduction section (page 5).

1.3. Overview

Chapter one, *Introduction*, introduces the project: "allPOS - cloud-based Electronic Point of Sale" and provide the reasoning behind the idea of undertaking this specific project.

Chapter two, *Research*, illustrates some basic concepts obtained from the background research to familiarise the user with the used technology and terminology and how this project fits in.

Chapter three, *Requirements Specification,* provides the functional and non-functional requirements of the project. Furthermore, use-case examples will be provided.

Chapter four, *Design and Implementation*, describes the different design approaches taken in the implementation process, high and low-level design, application walk-through and GUI.

Chapter five, *Testing*, walks the reader through all the testing techniques and results. Furthermore, it lists any bugs and flows found and they resolution.

Final Chapter, *Conclusion*, reports study limitations, outlines the scope for further studies and conclude the project.

2. Research

Research on EPOS systems and their architectural design has been conducted to identify the primary potential issues, acquire knowledge of the EPOS system market, research into existing software companies that provide similar solution and identify their strengths and weaknesses. Comprehensive software architecture research has been conducted to gain a comprehension of how complex systems are build and the diverse design patterns and principles they implement.

2.1. Django Web Framework

Before moving toward the aspects of what an EPOS system is and how it operates, we will proceed with a detailed introduction on the Django Web Framework. The introduction in this subsection would portray the technical aspects of the framework and familiarise the readers with the fundamental functionalities.

2.1.1. What is a Web Framework

For building applications that interfere with a server database and the web, developers are using web application frameworks. Web frameworks aim to encapsulate the low-level implementation of web services and provides tools and libraries to simplify the development process.

Nowadays, web applications are becoming more sophisticated than ever. A modern web application is capable of producing a dynamic based content, it communicates with many micro-services simultaneously, maintains a significant amount of data and many more. One of the most common frameworks is the Model-View-Controller design pattern which intends to divide the code for each application component into independent modules.

2.1.2. Django

A programming language named Python was initially created in the 1990s by Guido van Rossum as a member of the National Research Institute of Mathematics and Computer Science. Python is an interpreted, high-level programming language and was first released in 1991 [1].
'Django, on the other side, is a high-level Python web framework that enables rapid development of secure and maintainable websites' [5]. Django is free and open-sourced, with an incredible, thriving community, comprehensive documentation and free support.

2.1.3. Django architecture

Django utilises the Model-View-Controller (a.k.a MVC) architecture which distributes code responsibilities into different modules:

- The model outlines a relational database abstraction, and it is the logical low-level data structure. Models abstract the creation of the relational database and implement a simple to use interface for database queries. The interface acts as a mechanism between the high-level part of the framework and the database. Each model maps to a single database, and it is a definitive source of information about the data [2].

- The view also know as a Template, is the component of the Django application that represents the Graphical User Interface. The GUI is composed of HTML, CSS and JavaScript files to provide a dynamic and modern User Interface. Most Django applications use the view to render the model database[2].

- The controller is the most vital part of the application. In Django terms, the controller is known as a view. Views are python files that are responsible for managing web request and responses, manipulating data coming from the models, connecting the model's data with the user interface (view - the component above controller), user authentication, data sanitisation, managing cookies and user sessions, and many more [2].

Figure 2-1 illustrates how the Model-View-Controller generates a response based on some user request.
For instance, if we want to register on a website "A" when the user inserts its personal data into an HTML form, the data is sent with a "POST" request to the controller, the controller has the responsibility to sanitise the data, and substantiate that the data is valid. After validation, data is sent to the model. Once the data is received and processed, the model has the responsibility to send back to the controller an acknowledgement. The controller then gets the acknowledgement and transfers it to the view. The view renders the graphical representation of the model's response. For instance, in our case, a message "Registration Successful" could be printed onto the user's screen.

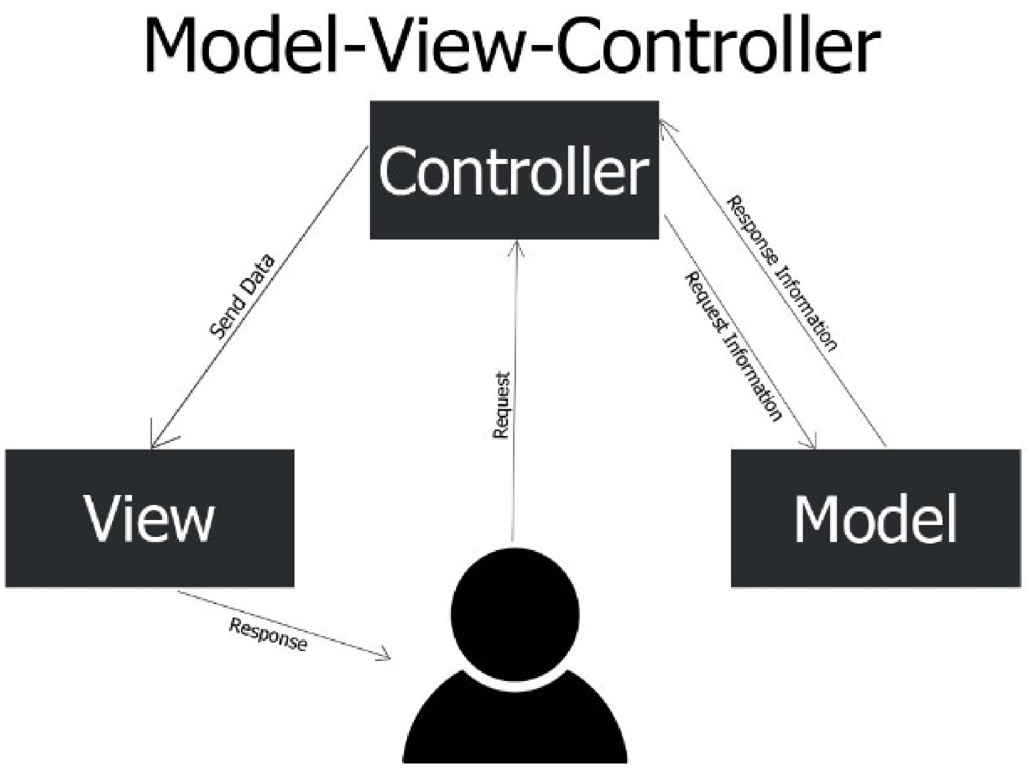


Figure 2-1: Model-View-Controller design pattern [3]

2.2. Electronic Point of Sale

An EPOS, also known as electronic point of sale system represents hardware and software that work collectively in an attempt to process transactions more efficiently. The hardware usually represents a central computer. Upon business prerequisites, companies can order additional devices identified as "peripherals" when connected to the computer, they incorporate software usability.
The system accommodates several software customisations so it can fit into the business requirements and help with specific analytics and data aggregation explicit to the company.
It is installed either directly to the computer, or connected to the cloud, allowing monitoring and analysation of sales across several locations.
Business performance examination shifts into a more affordable and easily obtainable assignment, as a result of the electronic tracking of sales and reports. At a glance, it can show what the store sells, where it sells, whom it sells to, and which of the salespeople sells it best. Moreover, whether a business is in the retail or hospitality sector, EPOS systems can modernise the processes and revolutionise the marketing divisions to suit the buying behaviour of the most loyal customers.

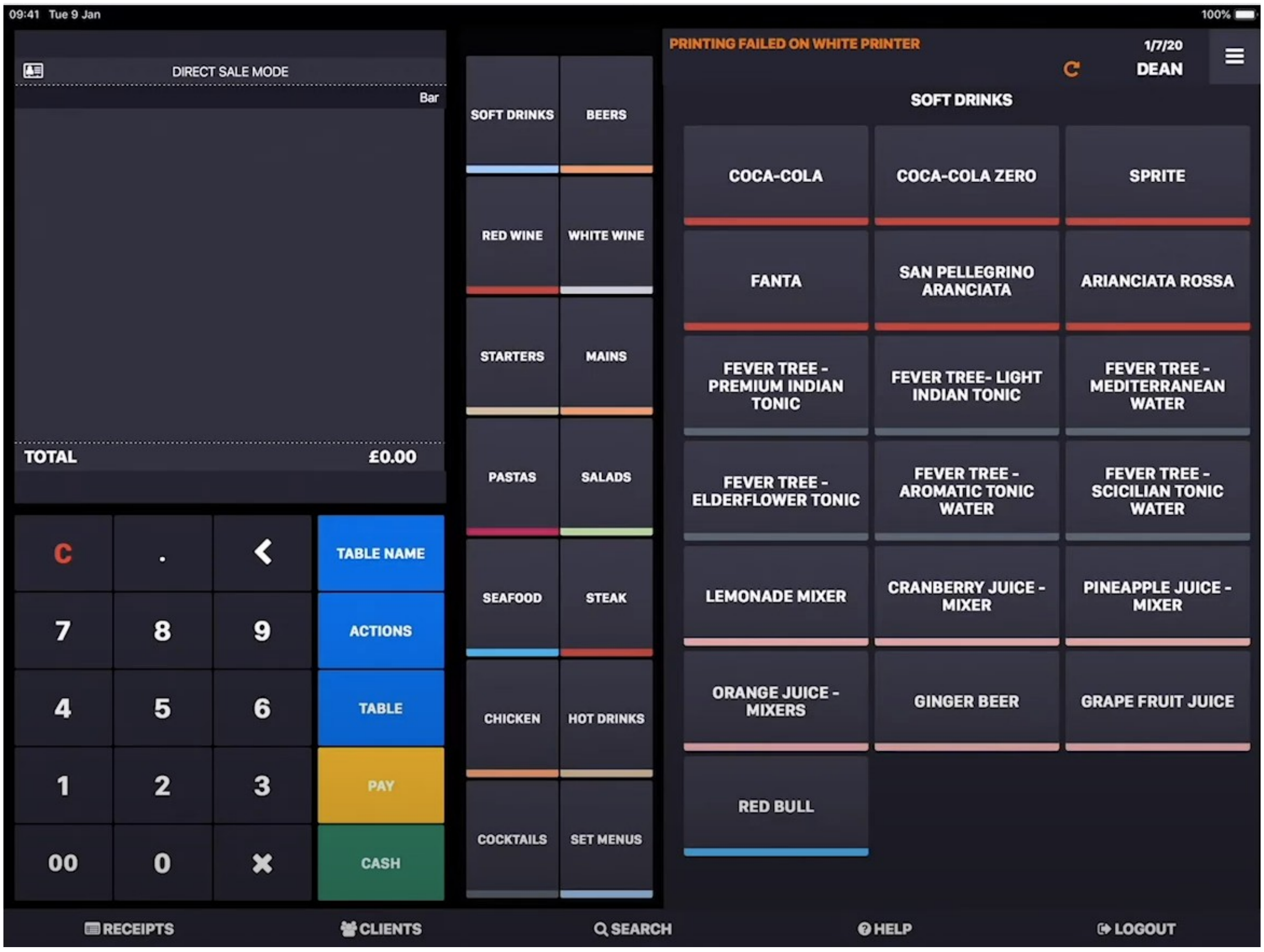


Figure 2-2: EPOS system User Interface

Another critical aspect of a successful EPOS system is the Graphical User Interface. 'Some time ago POS interfaces have replaced traditional cash registers in offline stores, and now they do a lot more than just process payments. Accordingly, POS system designers must embrace the modern context and offer complex solutions that benefit the customer and simplify things for a business.' [4]
Figure 2-2 demonstrates the sophistication behind the graphical user interface of an EPOS system. According to [4], those are the most common EPOS design principles:

- Many unnecessary UI elements could cause a cognitive load, therefore, simplicity is one of the main approaches during the EPOS design planning and implementation. 'In a nutshell, follow the KISS principle (Keep it Simple, Stupid) to keep everything right' [4]

- Usability, as a result of a user who utilises a specific product to accomplish a particular goal, the most critical elements to examine is the effectiveness, the efficiency and satisfaction from the results [6].

- Another aspect of an exemplary GUI is design consistency [4]. Implementation of multiple new elements on every single page does not account for good graphical design, and it often confuses users. Moreover, the reusability of a set of UI elements would significantly increase design optimisation.

- Page/view navigation should be straightforward to achieve. Furthermore, all tasks flow need to be logically designed. In this manner, it would promote the 'KISS' principle and incorporate Usability, providing an obvious and straightforward flow.

2.3. Employee Schedule

Retail and food alike businesses use an employee scheduling system to support the process of employee scheduling, enhancing punctuality of their employees while providing them with the flexibility of creating their working patterns. Furthermore, employee scheduling systems provide transparency by globally promoting every new schedule consisting of all the employees that are part of a particular department. As a result, it enhances communication between them.
In contrast, spreadsheets scheduling is very time-demanding and complex. As the business grows, the number of employees grow accordingly, making spreadsheet scheduling hard to manage and change on request.
Moreover, employee scheduling software also reduces labour expenses. According to a Hospitality Technology study (cited in [7]), an employee scheduling system accomplished an average annual saving of 2.2% labour cost.
Besides, by utilising shift booking frameworks, supervisors can diminish planning time, ease the stress, and forestall communication breakdowns among managers and staff. Particularly in the hospitality sector, where workers support is pivotal, zeroing in on time adaptability and representative prosperity will, at last, bring expanded consumer loyalty and better performance.

Hourly employees, for instance, students or workers with kids who cannot work all day, need adaptability in their work routines to study or take care of their youngsters. In any case, in the food-service business, it is customary for employees not to have control over their work routines. According to a study from Wonk and Ko (cited in [7]), it is a common issue for workers who experience the ill effects of burnout on account of the unpredictability of the time-sensitive clash and need to work through clashing requests in their personal lives.

By changing to a work planning framework, organisations can smooth out their labour force plans and improve the correspondence between hourly workers and their managers. Employees can get their timetables straightforwardly on their phones and can easily demand downtime with one click. Moreover, online employee booking has made greater adaptability to the hourly workers work routines since they have more control over their work hours and have more impact over their timetables. This, thus, could cause an individual to feel that work encounters improve the nature of their lives and that there is more prominent harmony among work and home balance [7].

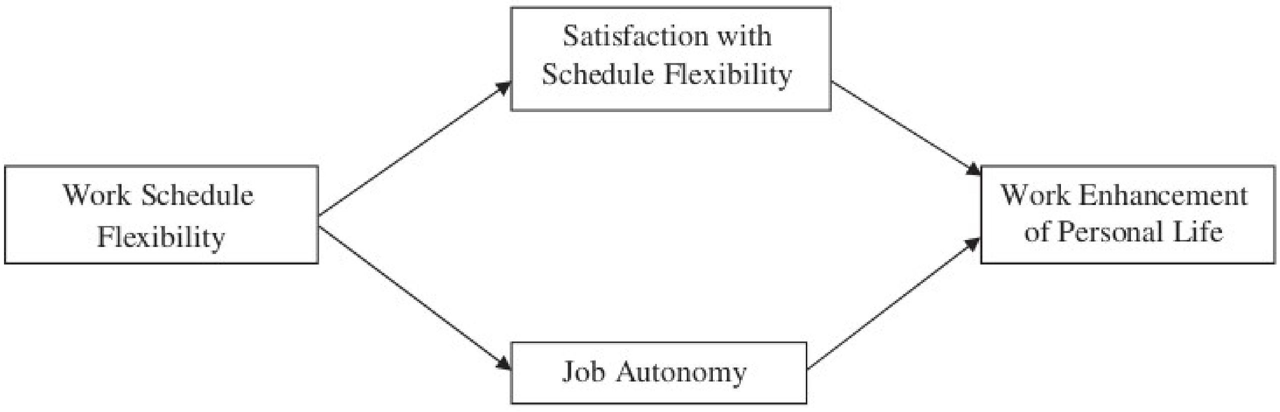

Satisfaction with Schedule Flexibility
Work Schedule Flexibility
Work Enhancement of Personal Life
Job Autonomy

Figure 2-3: Mechanisms for improving employees quality of life

Figure 2-3 illustrates the two essential components to improve employees personal satisfaction. It enhances schedule flexibility and provides job autonomy which is beneficial to employees personal life.

2.4. Stock Control

With regards to stock, the hospitality sector is one of the most disregarded industries yet as should be evident from the amazingly tough food-service market, what should matter most to any food business is the viable administration of orders, following menu items and in general stock management.

The term inventory is highly utilised within the accounting department. It refers to merchandise that is in different phases of being made available for sale, including:

- Completed products (that are accessible to be sold)
- Work-in-progress (products that are not yet completed)
- Crude materials (those are utilised to produce more competed goods)

Businesses ensure that all bookkeeping records are up-to-date and accurate by taking a stock count towards the end of each bookkeeping period, which is commonly quarterly or over year. Furthermore, companies that do a day by day stock count are considered to take interminable stock because their count is consistently current.
A strategy called "just-in-time" inventory is utilised to reduce the number of items that are in stock. By using this method, companies have stock just in time to meet customer needs. 'As a result, there is less inventory sitting around, waiting to be produced or sold.' [8].

According to [9], there are four types of inventory in the hospitality and retail business:

- The current inventory that a business has on a stock is called a "sitting inventory" Sitting inventory could be measured either by total weight, a quantity or as a monetary amount. [9]
- The amount of products a venue uses on a given time-frame is called "depletion", and its measurement is logged in the system likewise the sitting inventory. [9]
- When ordering goods, a business should know how long the sitting ordered inventory will last for. In this instance, the sitting inventory is divided by the average depletion time. Consequently, it abstains from running out of items, over-ordering and food wastage. [9]
- A difference within what a company ordered as a raw good and what the company sold as a single product is called a "variance". Variance is crucial, especially in the hospitality business. It is essential to understand how much of raw goods is wasted, as it helps get to the bottom of the wastage problems and reduce them. [9]

A stock management system implements an intuitive way to register all raw products that are received from a supplier. It allows for tracking their quantities and analysing how different items are used in a given time frame while reducing wastage.

2.5. Critical Review of existing applications

In this section, I will be evaluating similar applications that are available on the market and will be comparing their functionalities and features. However, there are limitations to this practice:

1. A functionality description or review does not cover the specific feature in-depth.

2. It only describes what the software does, but not how it does it.

3. GUI flows cannot be analysed.

2.5.1. Comparison to other EPOS systems

After extensive Google research of the top five EPOS systems, the following companies appear to provide a significantly large amount of functionalities. Besides, those companies are one of the most rated in the market:

- TouchBistro
- vendPOS
- Lightspeed Restaurant
- Square Point of Sale
- Epos Now

2.5.2. Non-functional EPOS comparison

Table 2-I: Non-functional EPOS comparison

Application name	allPOS (my project)	Touch Bistro	Vend POS	Lighspeed Retail	Square Point of Sale	Epos Now
1. Built for	Hospitality and Retail	Hospitality	Retail	Hospitality and Retail	Hospitality and Retail	Hospitality and Retail
2. Pricing	Initially free	£49+ per iPad/mo	£49+ per till/mo	£69+ per till/mo	Free	£25+ per till/ mo
3. Compatible devices	Any device that has internet browser and internet connection	Ipad	Ipad, Mac, Windows	Ipad, Mac, Windows	Ipad, iPhone, Android tablets and smartphones	Ipad, Android tablets, Mac, PC touchscreen

2.5.3. Inventory Management comparison

Table 2-II: Inventory Management comparison

Inventory Management	allPOS (my project)	Touch Bistro	Vend POS	Lighspeed Retail	Square Point of Sale	Epos Now
1.Categories and variants	Yes	Yes	Yes	Yes	Yes	Yes
2.Stock levels	Yes	Yes	Yes	Yes	Yes	Yes
3.Auto-supply	Yes	--	Yes	--	--	Yes
4.Ingredient Tracking	Yes	Yes	--	Yes	--	Yes
5.Automated decision-ordering	Yes	--	--	--	--	--

2.5.4. Reports and Analytics comparison

Table 2-III: Reports and analytics comparison

Reports and analytics	allPOS (my project)	Touch Bistro	Vend POS	Lighspeed Retail	Square Point of Sale	Epos Now
1.Daily Reports	Yes	Yes	Yes	Yes	Yes	Yes
2.Close/end-of-day report	Yes	Yes	Yes	Yes	--	Yes
3.Sales analytics	Yes	Yes	Yes	Yes	Yes	Yes

2.5.5. Employee and Store management comparison

Table 2-IV: Employee and store management comparison

Employee and store management	allPOS (my project)	Touch Bistro	Vend POS	Lighspeed Retail	Square Point of Sale	Epos Now
1.User logins	Yes	Yes	Yes	Yes	Yes	Yes
2.Different permission levels	Yes	Yes	Yes	Yes	Yes	Yes
3.Staff analytics	Yes	Yes	Yes	Yes	Yes	Yes
4.Shift-scheduling	Yes	--	--	--	--	--

2.6. Research conclusion

The tables above illustrate the different features of the reviewed applications. It can be seen that there is a gap between their functionalities, platform compatibility, and cost.
Three of the five companies provide "auto-supply" functionality. This feature allows managers to generate a list of of products with fixed quantity for each, that will be ordered on a given period. However, it needs to be noted that this feature is distinct from my idea of "auto-ordering based on previous sales patterns". While the "auto-supply" feature is capable of ordering automatically on a given time frame, it is ineffective in determining how many of the specific items need to order.
On the other side, I am planning to implement an intuitive system that will re-supply a different amount of different items based on some decisions.
Furthermore, three of the five systems are highly expensive, eliminating small and medium retailers from their customer targets.
From the reviewed applications, although all five systems are device-dependent, only one company provides multi-device comparability, however, it lacks one of the most critical features.
Moreover, none of the reviewed applications implements an employee scheduling system and automated decision-ordering feature natively, which could be the gap my project could fill in.

2.7. Summary

This chapter introduces the reader into different technologies, their high-level functionalities and specifications. It describes what a web framework is, Django Web framework, Django architecture, Moved-View-Controller.
Besides, it presents the user with a definition of the three systems that my project will be build of.

The concludes with a comparison research conclusion. It compares the top five rated systems on the market, analysing their features and non-functional requirements.

3. Requirements Specification

In this part, I will introduce the various requirements for this project. Functional requirements would be identified first, consisting of primary and secondary requirements. The majority of the functional requirements are introduced in the tables of chapter 2, subsection 2.4.
It will then continue with non-functional requirements. Non-functional requirements do not affect the functioning of the application but are essential for users.

3.1. Primary Functional Requirements

Primary functional requirements are the critical components of this project. Those are the functionalities that will achieve the aim of this project combining the three systems features into one whole application.

- **Main menu dashboard**: When a user logs in, the "**main menu dashboard**" screen will be shown.
  In the left-hand side, there will be the main menu where users would be able to swap between different parts of the application. The main functionalities in the left-hand side menu would be "**POS**" system screen, "**Employee Schedule**" screen and "**Stock Management**" screen. Additionally, each of those, mentioned earlier, will include subsection drop-downs so users can quickly navigate to the options of the three main applications.

- **POS:** Point of Sale screen will be the most used part of the application. Users (employees) will be using this screen to take orders from customers, generate their bills and take payments.
  The point of sale screen on one side will include the selection of items that the business sells and on the other side order number, items ordered and total amount. Employees would be able to delete added items or cancel the whole order. A pay button will be available, which will navigate the user to the "**payment**" screen.

- **Payment:** The payment screen will consist of a number keypad on the right-hand side where users could enter a custom amount to be paid (this is useful when customers want to split the bill). Furthermore, the right-hand side will consist of all the ordered items and a total amount. The bottom of the screen will include several buttons:

- Pay button will trigger a pop-up, enabling the user to pick the payment method - cash or card.
- Back button, which will navigate the user to the previous screen "POS" so users can add more items to the order or cancel the order.
- Add tips button, which will include a custom amount of tips to the total order amount.
- Add discount would display a pop-up. The pop-up will include several options:
    - Add promo code (discount based on promo code)
    - Add staff discount (fixed staff discount)
    - Add discount (general custom discount)

- **Employee Schedule:** The main screen of this applicationwill consist of a table showing:
    - the employees working today
    - employee start time
    - employee end-time
    - the time an employee logged in for the day
    - the time an employee logged out for the day
    - total hours worked for the day
    - the time an employee went in brake
    - then time an employee came back from the brake
    - The daily cost for each employee based on the time they worked and payment rate.

  User would be able to add new employees for the day. In addition it would be able to navigate to "**Schedule Employee Rota"** screen.

- **Schedule Employee Rota:** This screen will contain a weekly calendar. The user would be able to change between different weeks forward and backwards. An employee could be scheduled for each day, including working hours. Each calendar day cell should consist of the employee names working this day, otherwise empty.

- **Stock Management**: Stock management screen will list items based on their categories. Each category will be listing items in ascending order, meaning that the lowest items on stock will be listed at the top of the list. The user would be able to:
  Edit, delete or add new:

  - category
  - Item

  The stock management screen will include other stock management options such as stock analytics, suppliers list, automated decision ordering and several more.

3.2. Secondary Functional Requirements

Secondary requirements are crucial extra features of the primary requirements, nevertheless, secondary requirements would be implemented once the primary requirements are satisfied.

- **Employees Profiles:** the profiles would be used for several functionalities:
  - General employee information
  - Employee scheduling
  - Employee login/logout; daily login/daily logout
  - Order details – every order will consist of the employee who made the order and took the payment.
  - Sales analytics – which user sold the most today.

  User’s profile picture would be places in the top-tight corner within the nav-bar from where user would be able to access their profile settings.

- **Login:** A crustal part of the system would be to follow the activities of each employee. For instance, each order would include the employees name. When the application is not in use it would automatically log out the last user who used the application and a login screen would be shown.

- **Daily Login/Logout:** When employee starts/finishes it’s shift it needs to use this login/logout feature in order to record it’s starting/finishing shift time. Users who have not used the “**Daily Login**” would not be able to use the “**Login**”.

- **Analytics:** The “main menu dashboard” will consist of sales analytics charts. The charts would display daily income, weekly sales income and the most popular item during the day (the item that was sold the most). Additionally, a list with all working employees for that day would be rendered, alongside their starting and finishing hours.

- **Reports:** The application would be able to generate a sales report, including the items that were sold on that particular day, the total gross income and net.

- **Ingredient tracking:** allPOS would be able to build items our of ingredients. In this manner, hospitality businesses would be able to track their menu and the specific ingredient that they need. The stock system would store items, which items could be either build of ingredients or “solo items” meaning that they are already a product.

- **Permissions:** Three permission levels would be available:

  - Employee: employee permissions would allow users to take orders, take payments, view only their own schedule, view items low on stock.

  - Supervisor: supervisors would have access to the “employee” permissions. In addition, they would be able to view sales analytics, generate refunds, view all employees schedule and all items in stock. Furthermore, this permission level would be able to create new employee and supervisor permission users.

  - Manager: manager would be the likewise admin permission. It will give user access to “employee” and “supervisor” permissions. Furthermore, manager permission would allow the user to schedule employees, input employees holiday/day off requests, change users pay-rate, add/remove/delete items from the stock, generate sales reports, employees reports, end of the day reports and create users of all permission levels. Manager users would be capable of set-up automation ordering based on previous sales. Supplier details will also be a manager user responsibility.

3.3. Non-Functional Requirements

- **Compatibility**: allPOS should be available to any device that has a browser and an internet connection. Furthermore, the web app should be responsive for devices with smaller resolution. Despite the device used, it would have to behave in the same way across several devices. A PC, iPhone, Android phone, Android tablet will be used to test the compatibility and responsiveness of the app.

- **Reusability**: Django applications consist of several apps. Django app is a small library that represents an independent part from a larger project. Moreover, apps could be reused in another project.

- **Security**: All user data should be stored in an encrypted format. In addition, form inputs should be salted and sanitised to avoid injection attacks. Cookies should be inaccessible to JavaScript, HTTP only. Furthermore, a Cross-Site Request Forgery protection across all the forms in the website should be applied. Avoid session hijacking by storing user's IP address in session variables. Implement an HTTPS to prevent wiretapping and man-in-the-middle attacks.

- **Efficiency**: The system should take no more than 5 seconds to render different pages, update, add or remove any data from the database. Furthermore, the application should be able to support thousands of users working simultaneously and exchanging data with the main server.

- **Availability**: The system must be available at list 99% of the time. If there is any maintenance that needs to be conducted, all customers must be informed. In addition, maintenance should be led in the early morning of the first day of the week.

3.4. Use-Case Examples

This section will introduce the key use-cases of the application shown in Figure 3-1, 3-2, 3-3 and 3-4.

- **allPOS:** The use-case from figure 3-1 represents one of the most critical component for users to use allPOS system – they need to be registered first. It can be seen that a Manager can register all the employees that a supervisor can register. Additionally, only a manager can register another manager. A registered user provides personal data, profile image, NIN and has a permission level.

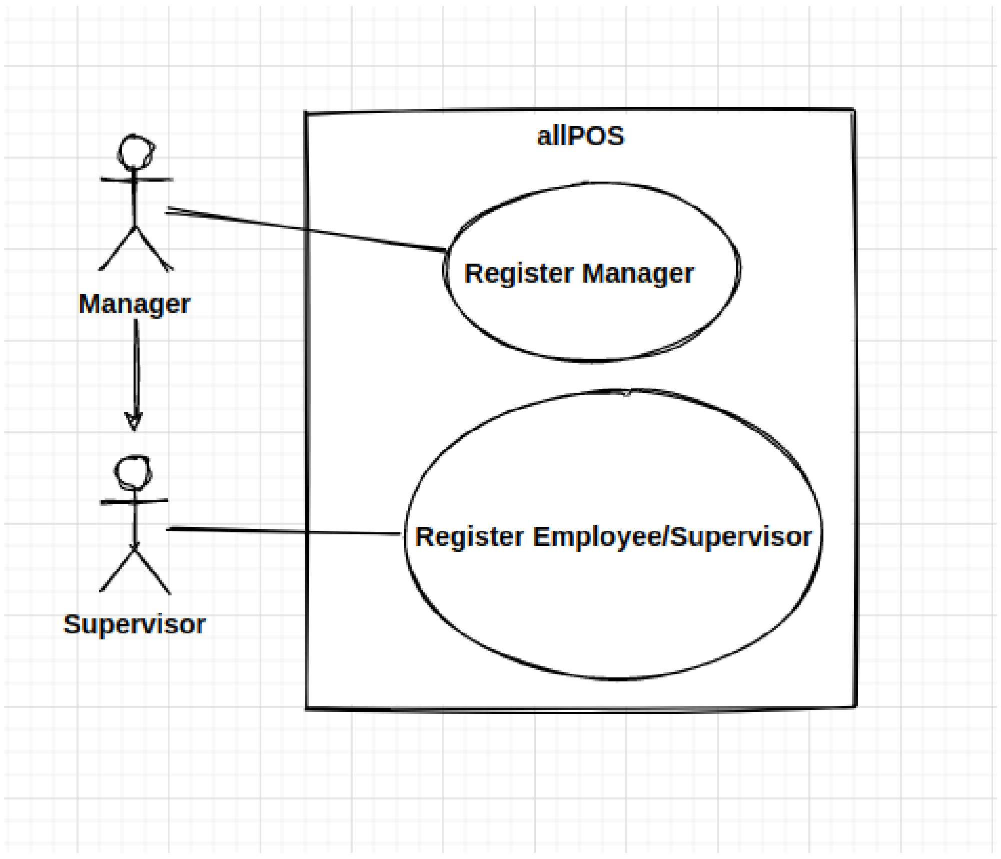


Figure 3-1: allPOS Use-Case

- **EPOS:** Figure 3-2 illustrates the most used component of allPOS. This is where employees take customer orders, add items, generate bills and take payments. Employees can take orders and take payments. Additionally, they have the extended functionalities of adding tips, apply discounts and to generate receipts. Supervisors can do all the task an employee can do. Additionally, they can refund customers from the payment screen, cancel an order and view weekly and daily sales analytics. Sales analytics would be placed on the main screen menu, but it would show up depending on the access level. Managers can perform all tasks a supervisor and an employee can. Furthermore, they can generate daily or weekly sales report and end-of-the-day sales report.

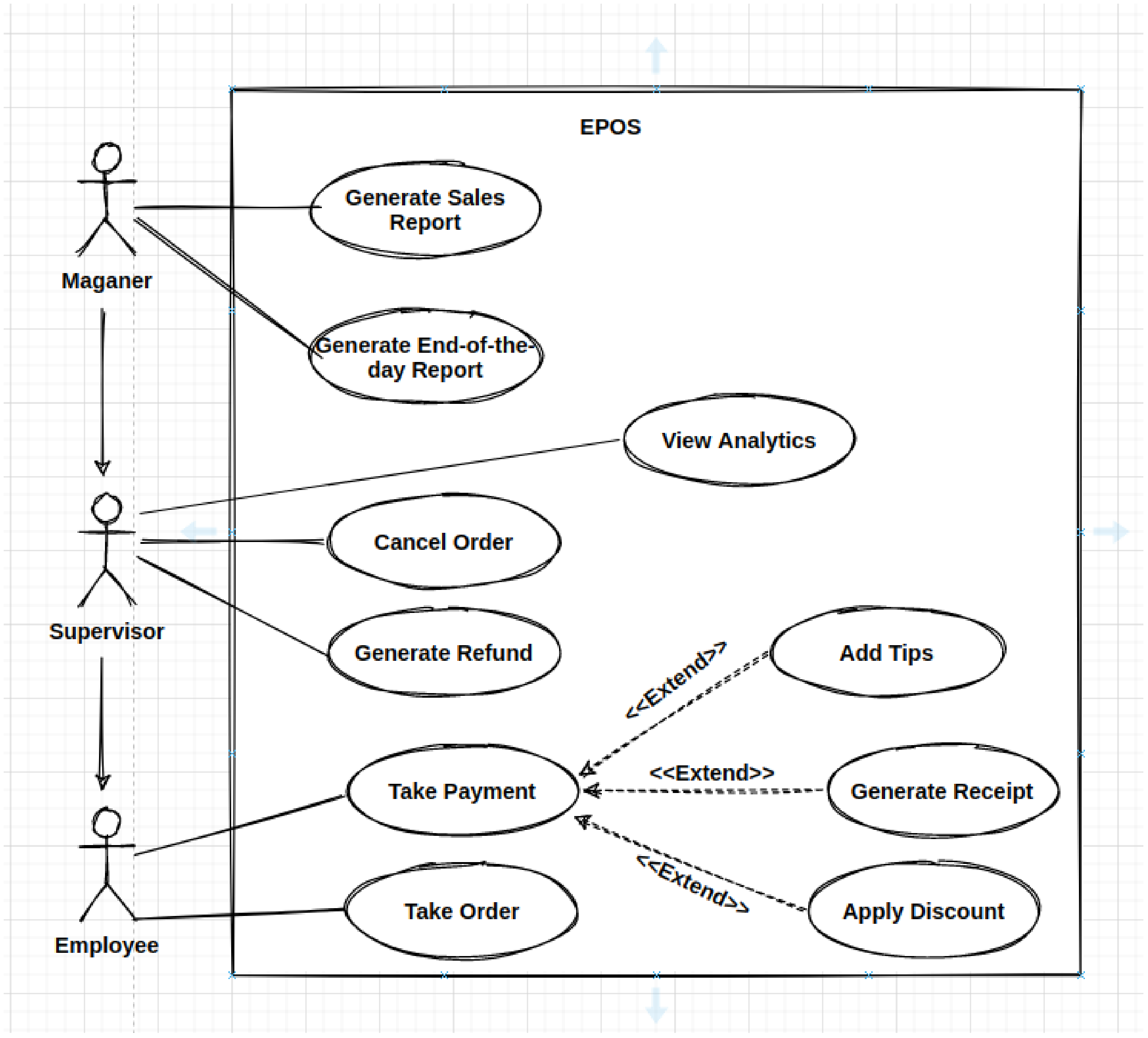


Figure 3-2: EPOS Use-Case

- **Employee Schedule:** Figure 3-3 represent the employee schedule use-case. It can be seen that an employee will only be able to see its own schedule for the day. The daily schedule will include time-started, how many hours an employee has worked so far, its pay-rate and finishing time. Supervisors inherit all the tasks employees can perform. Furthermore, supervisors would be able to view all the employees that are working today, including, starting-time, finishing-time, hours generated so far. Managers, on the other hand, would be able to create a schedule. They will have an admin calendar where more than one employee can be placed in a "day" cell. The cell will include all the employees added in the sell and their working hours.
  Managers would be able to edit schedules, remove schedules, input holiday and day off requests. Moreover, they will be able to change employees pay rate and generate a report for an individual, or all employees worked on a particular day. The report will include the cost of

having this employee and the net profit the employee-generated over the requested time frame.

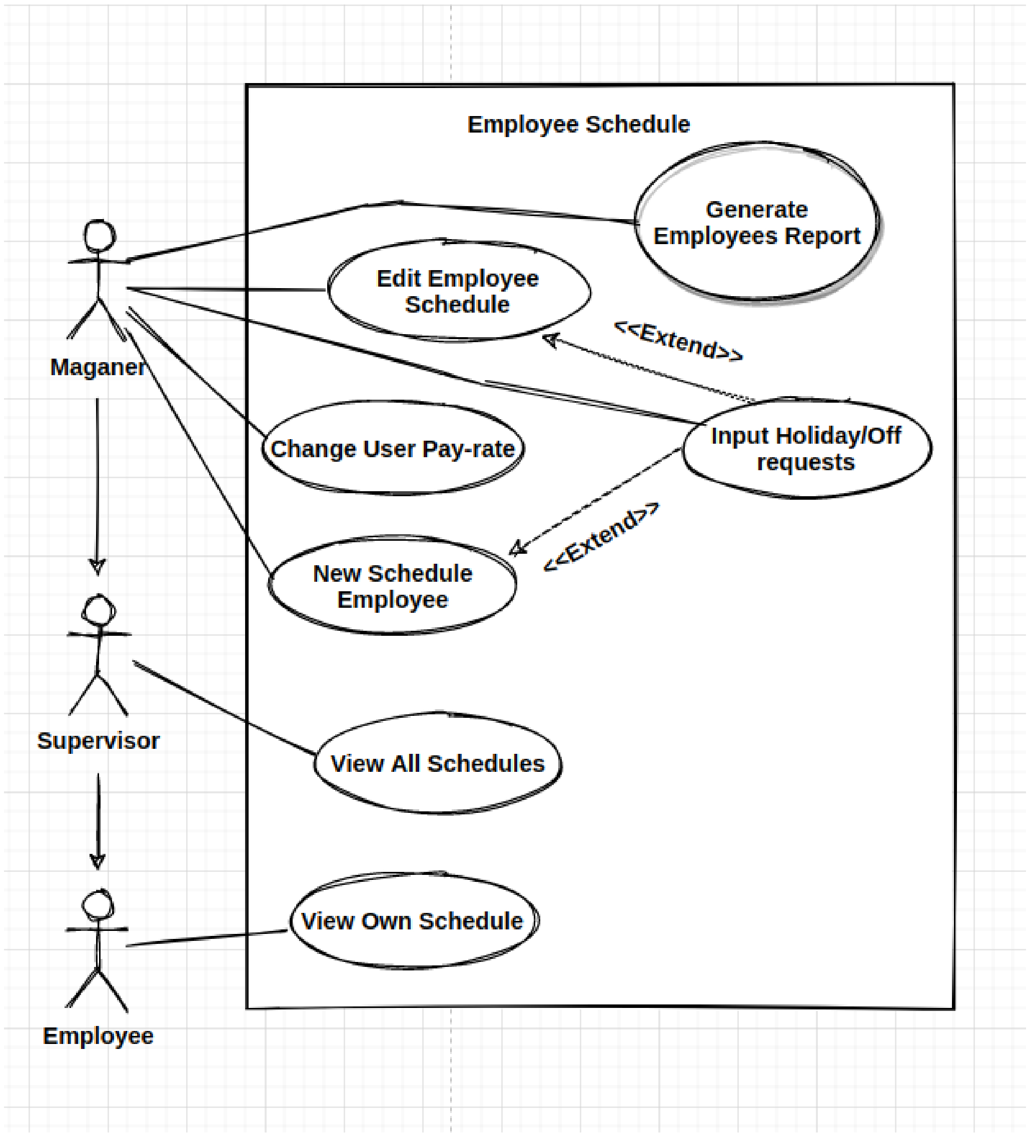


Figure 3-3: Employee Schedule Use-Case

- **Stock Control:** Figure 3-4 illustrate the Stock Control Use-Case. Employees can only see the items that are low on stock. In this way, they would be able to advise a supervisor or a manager. Supervisors, however, would be able only to view all stock items. Managers will have to add a supplier's account and items in stock. Once the automation order is activated, the module will regularly require information from the supplier's database accounts and items on stock. Once the module detects items low on stock, it will first calculate how long can the business last with the current stock based on previous sales. It will then generate a list of items that need to be ordered. Once the list is generated, it will be sent to the manager the night before the order it's placed. Manager can approve, cancel or edit the list. Once the list is approved, the system will send an order to each of the suppliers with the items.

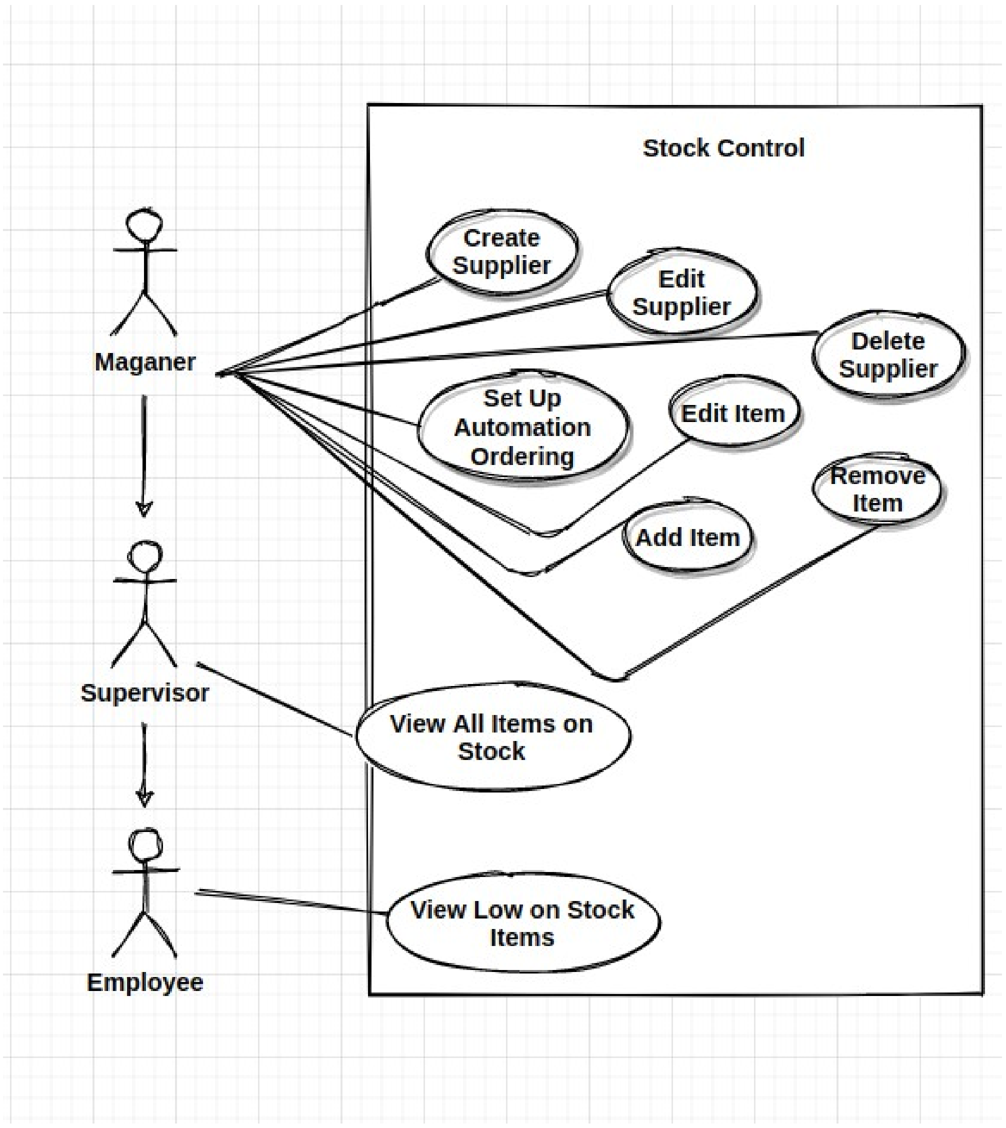


Figure 3-4: Stock Control Use-Case

3.5. Summary

The functional and non-functional requirements were described in this chapter. Functional remeasurements were divided in primarily and secondary requirements. The application functionality is not dependent on the non-functional requirements but those are required for the user experience.
The chapter concludes with the main use cases of the apps.

Reference List:

[1] John Wolfe (2018). A Brief History of Python.
Available from: https://medium.com/@johnwolfe820/a-brief-history-of-python-ca2fa1f2e99e [Accessed 10th November 2020]

[2] Timothy Ko (2017). A Quick Glance of Django.
Available from: https://medium.com/@timmykko/a-quick-glance-of-django-for-beginners-688bc6630fab#:~:text=robust%20and%20scalable.-,Django%20Architecture,such%20as%20MySql%2C%20Postgres). [Accessed 10th November 2020]

[3] Joseph Spinelli (2018). MVC Overview
Available from: https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5 [Accessed 11th November 2020]

[4] Sveta Yurkevich (2020). POS Interface Design Principles.
Available at: https://agentestudio.com/blog/design-principles-pos-interface [Accessed 10th November 2020]

[5] Hamish Willee (2020). Django Introduction
Available from:
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction [Accessed 10th November 2020]

[6] Xaview Ferre, Natalia Juristo (2001). Google Scholar: Usability basics of software developers.
Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=903160 [Accessed 10th November 2020]

[7] Shi Xu (2017) The effect of online scheduling on employees' quality of life
Available from:
https://www.tandfonline.com/doi/full/10.1080/15378020.2017.1364592?scroll=top&needAccess=true [Accessed 13th of November 2020]

[8] Shopify (unknown) Inventory definition.
Available from: https://www.shopify.co.uk/encyclopedia/inventory [Accessed 13th November 2020]

[9] Renae Smith (2019) Inventory management for hospitality businesses
Available from: https://www.myob.com/au/blog/inventory-management-software-hospitality/ [Accessed 13th November 2020]

Appendix

The final year project is a great way to demonstrate the skills that we have acquired at university for the last four years. Besides, it allows us to progress in a very niche direction of our choice, by working on a project that can influence our future employment and it's important to us.

To successfully implement my final year project and define a specific sector in which I wanted to dive in, I started to establish the scope of the project by reading different articles related to restaurant and retail systems. After the research stage, I had identified the gap in the hospitality system sector, and the scope of my project was established.

The second stage of the final year project was the literature review and literature research. It was carried for a couple of weeks by reading different research papers on the topic and processing the information that is relevant to my project. Additionally, software research was done alongside the literature review, examining different scenarios for the project implementation and design architecture. Implementation articles, libraries and templates were gathered to support the developing stage.

To maintain the monitoring of the project progress, I have decided to use a work management tool called Jira. Currently, I am defining the backlog with independent development milestones that are divided into tasks, also known as stories in the Jira ecosystem. Moreover, a project skeleton is currently under development and research of different libraries is in progress.

The project plan was clearly defined, and implementation goals were established in the following manner.
The development would be performed into three stages, as follows:

- **Stage One**: Point of Sale system
    - Start Date: 20/11/2020
    - End Date: 30/12/2020

- **Stage Two**: Employee Scheduling system
    - Start Date: 01/01/2020
    - End Date: 28/02/2020

- **Stage Three:** Stock Management system
    - Start Date: 01/03/2020
    - End Date: 30/04/2020

The project would be conducted in behaviour-driven development using a top-down approach. It defines the work in rapid, small iterations manner to increase the feedback of each module by continuous testing.
The starting point would be a user story implementation, testing and refactoring. BDD would be driven into weekly sprints, leaving the last sprint of each stage for final acceptance testing. Furthermore, a sprint would be starting with a user story, progressing into testing. If bugs are found it would continue with implementation and further testing.

Currently, the following requirements are organised into sprints for the first stage (Point of Sale system) of the implementation:

- **Sprint 1** (20/11 - 27/11):
    - Set up GitHub
    - Set up Jira
    - Set up OpenShift - deployment
    - Define all necessary libraries and front-end templates

- **Sprint 2** (27/11 - 04/12):
    - Define Relational Models
    - Implement Relational Model
    - Implement skeleton home page
    - Test models and home page

- **Sprint 3** (04/12 - 11/12):
    - Implement views
    - Implement front-end
    - Test views
    - Test front end

- **Sprint 4** (11/12 - 18/12):
    - Implement views
    - Implement front-end
    - Test views
    - Test front end

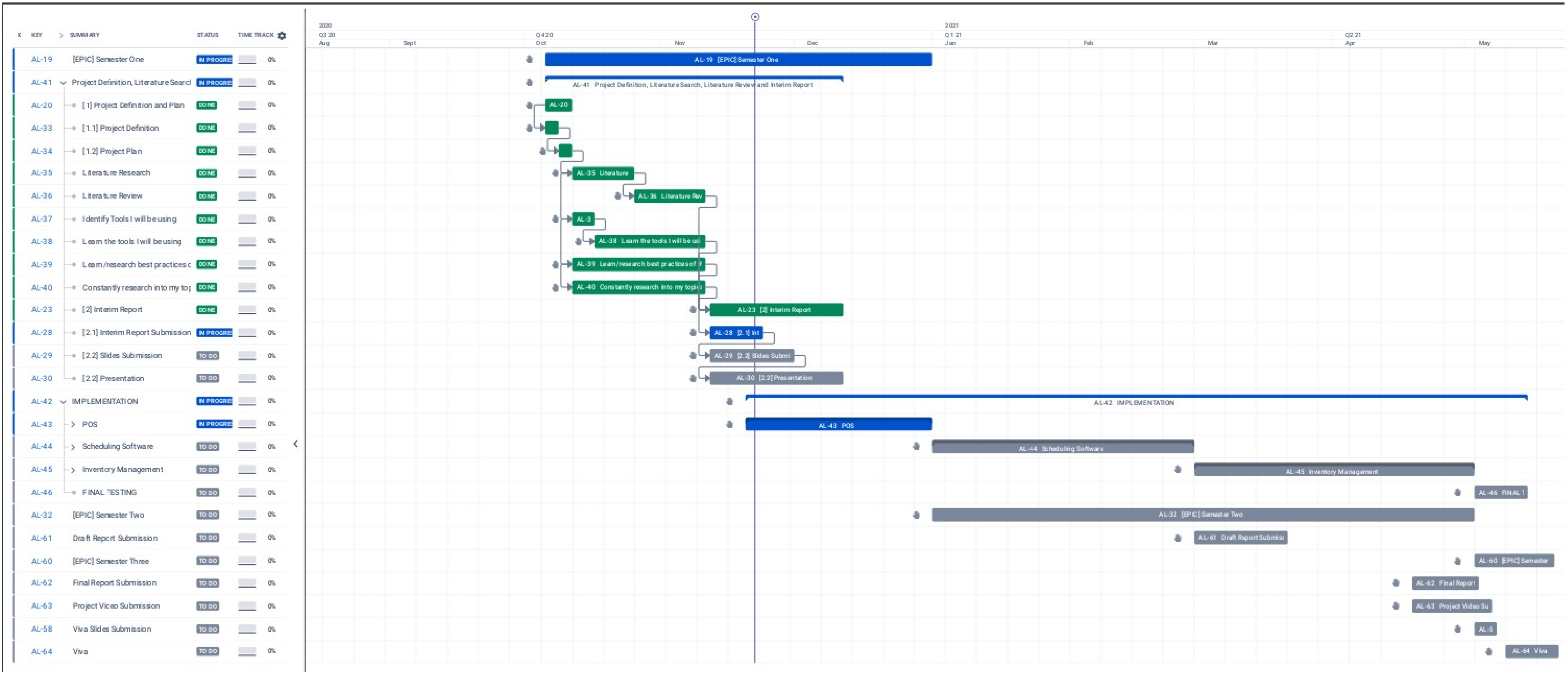
- **Sprint 5** (18/12 - 31/12)
    - Unity testing of each functionality
    - Smoke testing of the whole POS application
    - Integration Testing
    - System Testing

Employee Scheduling and Stock Management systems would be planned in the same manner. Consisting of individual sprints.

The first figure, bellow represents a Gantt chart for the overall project. It includes tasks that are done, tasks in progress and future task planning. In addition, due to the size of the Gantt chart, further parts of the graph would be provided bellow the overall chart of the project to illustrate details on each stage of the implementation.

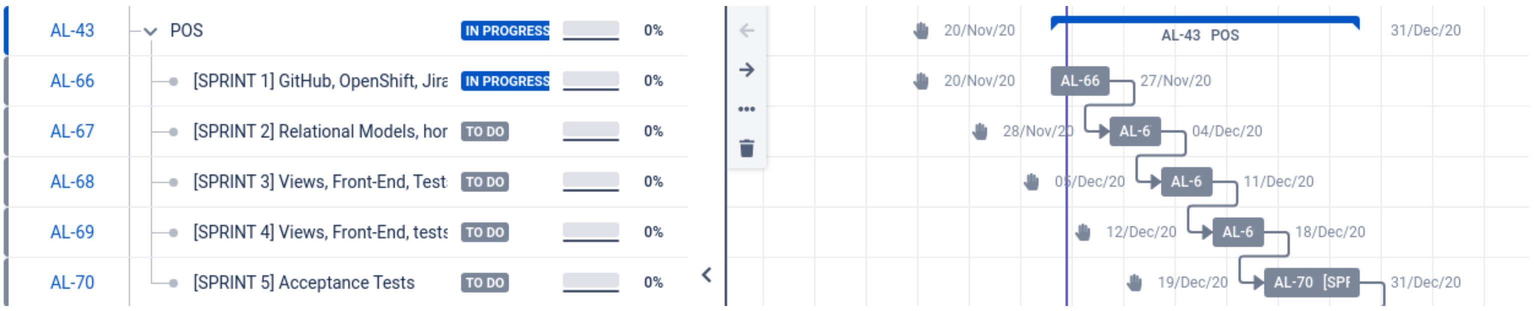
From the charts, it could be seen that different stages of the project are planned to be developed in a different manner. For instance, stage one has less number of sprints. That is due to the fact that it’s implementation is less complicated than Employee scheduling and Stock management systems.

Furthermore, Employee Scheduling system has less Back-End sprints than the Stock Management system. This is a component independent decision. The scheduling part of allPOS has more front-end functionalities, whereas the stock control module provides the business logic of allPOS, therefore, needing more back-end development.


2020
Q3 20
Aug
Sept
Q4 20
Oct
Nov
Dec
2021
Q1 21
Jan
Feb
Mar
Q2 21
Apr
May
AL-19 [EPIC] Semester One
AL-41 Project Definition, Literature Search, Literature Review and Interim Report
AL-20
AL-35 Literature
AL-36 Literature Rev
AL-3
AL-38 Learn the tools I will be us
AL-39 Learn/research best practices of
AL-40 Constantly research into my topic
AL-23 [2] Interim Report
AL-28 [2.1] Int
AL-29 [2.2] Slides Submi
AL-30 [2.2] Presentation
AL-42 IMPLEMENTATION
AL-43 POS
AL-44 Scheduling Software
AL-45 Inventory Management
AL-46 FINAL T
AL-32 [EPIC] Semester Two
AL-61 Draft Report Submiss
AL-60 [EPIC] Semester
AL-62 Final Report
AL-63 Project Video Su
AL-5
AL-64 Viva

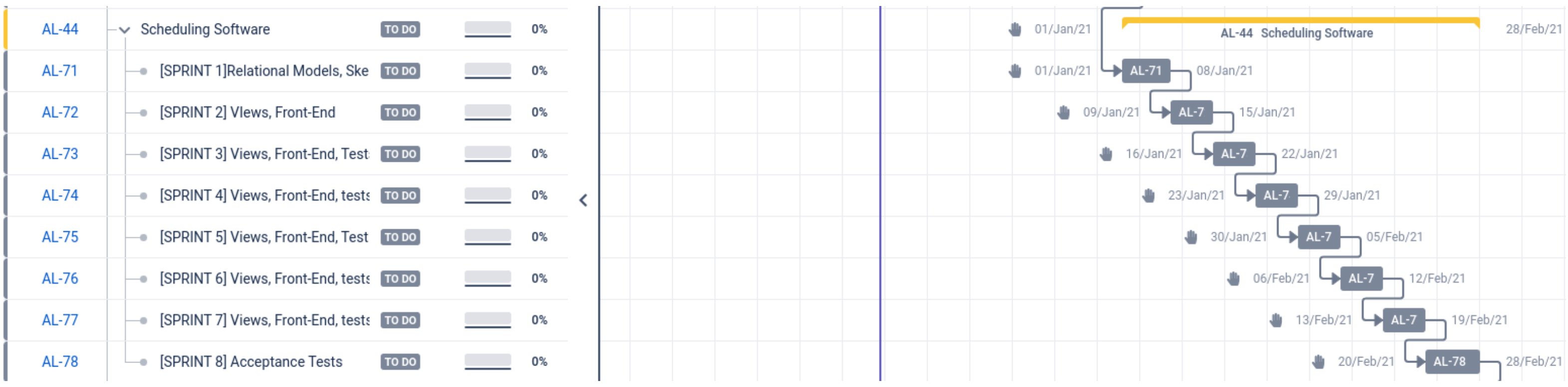
KEY	SUMMARY	STATUS	TIME TRACK	
AL-19	[EPIC] Semester One	IN PROGRE		0%
AL-41	Project Definition, Literature Searcl	IN PROGRE		0%
AL-20	[1] Project Definition and Plan	DONE		0%
AL-33	[1.1] Project Definition	DONE		0%
AL-34	[1.2] Project Plan	DONE		0%
AL-35	Literature Research	DONE		0%
AL-36	Literature Review	DONE		0%
AL-37	Identify Tools I will be using	DONE		0%
AL-38	Learn the tools I will be using	DONE		0%
AL-39	Learn/research best practices c	DONE		0%
AL-40	Constantly research into my to	DONE		0%
AL-23	[2] Interim Report	DONE		0%
AL-28	[2.1] Interim Report Submission	IN PROGRE		0%
AL-29	[2.2] Slides Submission	TO DO		0%
AL-30	[2.2] Presentation	TO DO		0%
AL-42	IMPLEMENTATION	IN PROGRE		0%
AL-43	POS	IN PROGRE		0%
AL-44	Scheduling Software	TO DO		0%
AL-45	Inventory Management	TO DO		0%
AL-46	FINAL TESTING	TO DO		0%
AL-32	[EPIC] Semester Two	TO DO		0%
AL-61	Draft Report Submission	TO DO		0%
AL-60	[EPIC] Semester Three	TO DO		0%
AL-62	Final Report Submission	TO DO		0%
AL-63	Project Video Submission	TO DO		0%
AL-58	Viva Slides Submission	TO DO		0%
AL-64	Viva	TO DO		0%

Figure A: Overall Gantt Chart



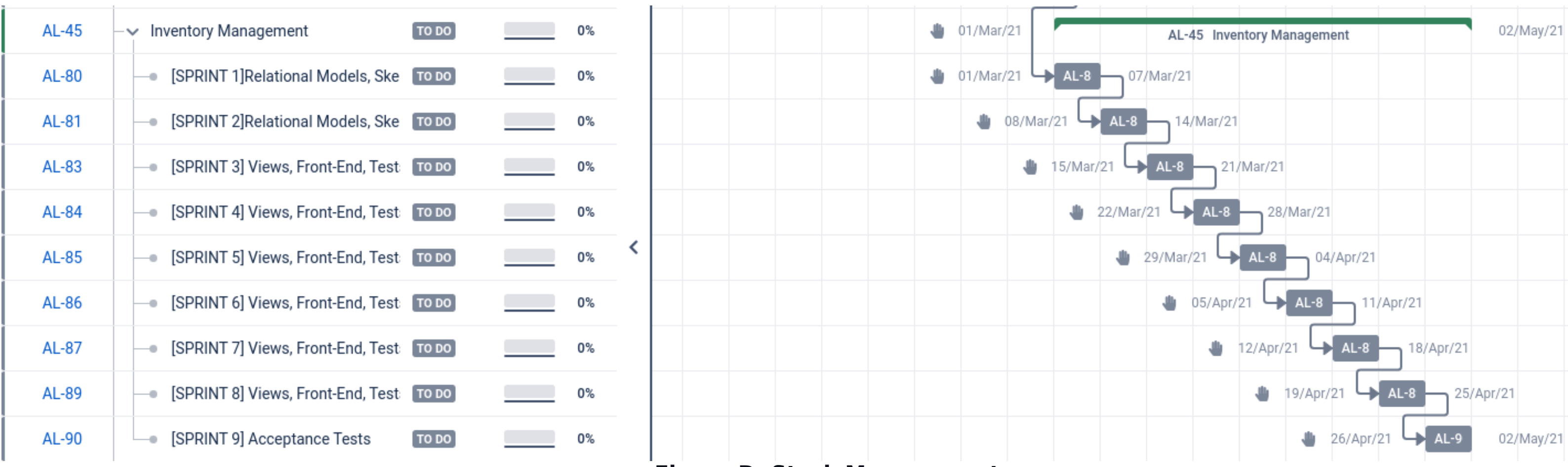
AL-43	POS	IN PROGRESS		0%
AL-66	[SPRINT 1] GitHub, OpenShift, Jira	IN PROGRESS		0%
AL-67	[SPRINT 2] Relational Models, hor	TO DO		0%
AL-68	[SPRINT 3] Views, Front-End, Test	TO DO		0%
AL-69	[SPRINT 4] Views, Front-End, tests	TO DO		0%
AL-70	[SPRINT 5] Acceptance Tests	TO DO		0%

Figure B: Point of Sale



AL-44	Scheduling Software	TO DO		0%
AL-71	[SPRINT 1]Relational Models, Ske	TO DO		0%
AL-72	[SPRINT 2] VIews, Front-End	TO DO		0%
AL-73	[SPRINT 3] Views, Front-End, Test	TO DO		0%
AL-74	[SPRINT 4] Views, Front-End, tests	TO DO		0%
AL-75	[SPRINT 5] Views, Front-End, Test	TO DO		0%
AL-76	[SPRINT 6] Views, Front-End, tests	TO DO		0%
AL-77	[SPRINT 7] Views, Front-End, tests	TO DO		0%
AL-78	[SPRINT 8] Acceptance Tests	TO DO		0%

Figure C: Employee Schedule



AL-45	Inventory Management	TO DO		0%
AL-80	[SPRINT 1]Relational Models, Ske	TO DO		0%
AL-81	[SPRINT 2]Relational Models, Ske	TO DO		0%
AL-83	[SPRINT 3] Views, Front-End, Test	TO DO		0%
AL-84	[SPRINT 4] Views, Front-End, Test	TO DO		0%
AL-85	[SPRINT 5] Views, Front-End, Test	TO DO		0%
AL-86	[SPRINT 6] Views, Front-End, Test	TO DO		0%
AL-87	[SPRINT 7] Views, Front-End, Test	TO DO		0%
AL-89	[SPRINT 8] Views, Front-End, Test	TO DO		0%
AL-90	[SPRINT 9] Acceptance Tests	TO DO		0%

Figure D: Stock Management