

|  |  |  |
| --- | --- | --- |
| **Module Code** | **:** | CT046-3-2-SDM –Systems Development Methods |
| **Intake Code** | **:** | APU2F2502CS(CYB) |
| **Lecturer Name** | **:** | Yogeswari Muthusamy |
| **Hand in Date** | **:** | 20 May 2025 (Week-11) |
| **Tutorial No.** | **:** | Tutorial 8 |
| **Group No.** | **:** | Group 15 |
| **Group Leader** | **:** | Yaw Kai Yuan |

|  |  |
| --- | --- |
| **Student ID** | **Student Name** |
| TP080121 | Yaw Kai Yuan |
| TP080661 | Gan Kai Hong |
| TP080254 | Er Chen Liang |
| TP080719 | Ting Jac Sheinn |
| TP077723 | Huang Xiao Shun |
| TP083088 | Chong Wai Kit |

Table of Contents

[1.0 Planning Phase 5](#_Toc198644462)

[1.1 Introduction 5](#_Toc198644463)

[1.1.1 Customer Company Details 5](#_Toc198644464)

[1.1.2 Current Business Process 5](#_Toc198644465)

[1.2 Problem Statement 6](#_Toc198644466)

[1.2.1 Single Payment Method 6](#_Toc198644467)

[1.2.2 Lack of Order Tracking 6](#_Toc198644468)

[1.2.3 Fragmented Menu Access 6](#_Toc198644469)

[1.2.4 No Centralized Monitoring or Analytics 6](#_Toc198644470)

[1.2.5 Order Miscommunication and Errors 7](#_Toc198644471)

[1.3 Proposed System 8](#_Toc198644472)

[1.4 Scopes and Objectives 10](#_Toc198644473)

[1.4.1 Project Scope 10](#_Toc198644474)

[1.4.2 Project Objective 10](#_Toc198644475)

[1.5 Project Scheduling 11](#_Toc198644476)

[2.0 Analysis Phase 12](#_Toc198644477)

[2.1 Product Backlog Creation 12](#_Toc198644478)

[2.2 Functional and Non-Functional Requirements 13](#_Toc198644479)

[3.0 Design Phase 14](#_Toc198644480)

[3.1 Use case and System Modelling 14](#_Toc198644481)

[4.0 Development Phase 18](#_Toc198644482)

[4.1 Prototype 18](#_Toc198644483)

[4.2 Testing Strategy 37](#_Toc198644484)

[4.3 Develop an initial Test Plan 38](#_Toc198644485)

[5.0 Deployment Phase 39](#_Toc198644486)

[5.1 Deployment Strategy 39](#_Toc198644487)

[5.2 Final Sprint Review and Retrospective 42](#_Toc198644488)

[6.0 User Story Creation and Processing 43](#_Toc198644489)

[6.1 Yaw Kai Yuan TP080121 43](#_Toc198644490)

[6.2 Er Chen Liang TP080254 44](#_Toc198644491)

[6.3 Chong Wai Kit TP080388 45](#_Toc198644492)

[6.4 Gan Kai Hong TP080661 46](#_Toc198644493)

[6.5 Ting Jac Sheinn TP080719 47](#_Toc198644494)

[6.6 Huang Xiao Shun TP077723 48](#_Toc198644495)

[7.0 Agile Principles and Reflection 49](#_Toc198644496)

[7.1 Yaw Kai Yuan TP080121 49](#_Toc198644497)

[7.1.1 Agile Principles 49](#_Toc198644498)

[7.1.2 IS Methodologies 50](#_Toc198644499)

[7.2 Er Chen Liang TP080254 52](#_Toc198644500)

[7.2.1 Agile Principles 52](#_Toc198644501)

[7.2.2 IS Methodologies 53](#_Toc198644502)

[7.3 Chong Wai Kit TP080388 55](#_Toc198644503)

[7.3.1 Agile Principles 55](#_Toc198644504)

[7.3.2 IS Methodologies 56](#_Toc198644505)

[7.4 Gan Kai Hong TP080661 59](#_Toc198644506)

[7.4.1 Agile Principles 59](#_Toc198644507)

[7.4.2 IS Methodologies 61](#_Toc198644508)

[7.5 Ting Jac Sheinn TP080719 63](#_Toc198644509)

[7.5.1 Agile Principles 63](#_Toc198644510)

[7.5.2 IS Methodologies 64](#_Toc198644511)

[7.6 Huang Xiao Shun TP077723 67](#_Toc198644512)

[7.6.1 Agile Principles 67](#_Toc198644513)

[7.6.2 IS Methodology 68](#_Toc198644514)

[7.7 Overall Comparison 69](#_Toc198644515)

[8.0 Design – System Design 71](#_Toc198644516)

[8.1 Yaw Kai Yuan TP080121 71](#_Toc198644517)

[8.2 Er Chen Liang TP080254 72](#_Toc198644518)

[8.3 Chong Wai Kit TP080388 74](#_Toc198644519)

[8.4 Gan Kai Hong TP080661 75](#_Toc198644520)

[8.5 Ting Jac Sheinn TP080719 77](#_Toc198644521)

[8.6 Huang Xiao Shun TP077723 78](#_Toc198644522)

[9. References 79](#_Toc198644523)

[10. Appendix - Workload Matrix 81](#_Toc198644524)

# 1.0 Planning Phase

## 1.1 Introduction

### 1.1.1 Customer Company Details

APU Food Court is a vibrant dining hub for university students, staff, and visitors. It is arguably the heart of campus life on a daily basis, serving a variety of food choices to a diverse range of campus needs and tastes. The main stakeholders in this ecosystem are university management, food stall owners & workers managing the ordering system. The food court provides seemingly endless options for breakfast, lunch and snacks during the day supported mainly by students and staff who use it regularly. The food court is also a popular aspect among visitors such as parents and campus guests, especially during days with a university event or campus tour.

### 1.1.2 Current Business Process

Currently, the APU Food Court uses a physical ordering system for students and staff to use their campus APCards to order and pay for food. This system works well for normal users, but it quickly falls apart for visitors and those without a APCard. Users order food at kiosks in the food court, but the company has found that it needs to improve the availability of real-time order notifications and the ease of use of its kiosk interface. Management of the food court is determined to improve the user experience by helping to fix these pain points, providing solutions that simplify the ordering process for the entirety of the patrons coming through, ensuring a faster more efficient solution.

## 1.2 Problem Statement

### 1.2.1 Single Payment Method

The current system only allows users to conduct cash transactions. E-wallets are now very popular, and many customers do not carry cash with them. It is very inconvenient for them. In addition, cash vendors need to manually make changes, which prolongs the transaction time with each customer and causes long queues. It also increases the risk of human error and loss, such as stall vendor calculating the wrong change for customers.

### 1.2.2 Lack of Order Tracking

After placing an order, customers can only wait near the stall without being informed when the food will be ready. The food may be finished earlier or later than expected, which may cause the food to get cold and taste bad. Since customers are not notified in real time and have to keep going to the stall to check if the food is ready, this may annoy customers. In addition, during peak hours, the crowds are even more crowded making it difficult for customers to figure out whose order is being prepared or to pick up someone else's order.

### 1.2.3 Fragmented Menu Access

Each food stall operates independently and does not have a unified menu. This causes customers to have to go to each stall to find out what dishes are sold, which wastes time and makes it impossible to compare prices. By the time customers find the dishes they want, they have already lost their patience, which affects their dining experience. In addition, the inconsistent menus confuse customers and require time to understand.

### 1.2.4 No Centralized Monitoring or Analytics

The current system APU management and vendors lack real-time data on popular foods, peak ordering hours, stall performance and sales. This makes it difficult for APU management to plan and improve the food court. Vendors cannot know which foods are popular and need to be prepared more and which foods are not selling well and need to be prepared less to avoid food waste. Without this real-time data, vendors need to reduce the amount of food to avoid waste, which results in customers not being able to buy the dishes they want and a drop in sales.

### 1.2.5 Order Miscommunication and Errors

The current system requires verbal orders to be placed at the stall, which may cause vendors to mishear or forget the dishes. It is easy to forget when there are many orders and customers require customized dishes. This is very frustrating for some customers because they may have health problems and allergies. When customers receive the wrong dishes or a food stall forgets a customer’s order, it will lead to dissatisfaction and reduce their willingness to come to APU food court again.

## 1.3 Proposed System

1.3.1 Proposed System Overview

The proposed APU Food Court Kiosk System is a thorough self-service solution that streamlines food ordering and payment processes within the APU campus. Kiosks that are indeed tactically installed will permit students as well as staff to browse food menus, place various orders, make cashless payments, and reload their meal credits through the utilization of their APU-issued ID cards. It diminishes manual errors, and it also betters user convenience. The aim of the system is to minimize queuing times. The users of Credit Reloading can reload the credits on their ID card with either cash or e-wallets by way of using the kiosk. Kiosks provide access to all the food menus from different vendors in the food court that is Digital Food Ordering. Customers, with the use of the E-cart System, can select items, review the items, and then confirm on orders. The ID card automatically deducts the payment and subsequently prints a receipt. The relevant food stalls fully receive orders routed electronically through Vendor Notification. APU administrators and stall vendors thoroughly handle menu management and pricing through a thorough CMS (Content Management System). They also do handle updates to the system by way of it. The students together with the staff at APU are considered primary users. Vendors: Food stall operators as administrators, APU staff handle the kiosk content and the backend configuration.

1.3.2 System Architecture

The proposed system is a modular, client-server-based architecture consisting of:

1. **Frontend (Kiosk Interface):** Touchscreen UI for order placement and reloading.
2. **Backend Server**: Manages orders, menus, user account balances, and transaction records.
3. **Database Layer**: Stores menu details, user profiles, transaction history, and stall information.
4. **Admin CMS Portal**: Web-based dashboard for admin and vendors to update menus and manage orders.
5. **Integration Layer**: Supports third-party services such as e-wallet payments (TnG, Boost, GrabPay, etc.

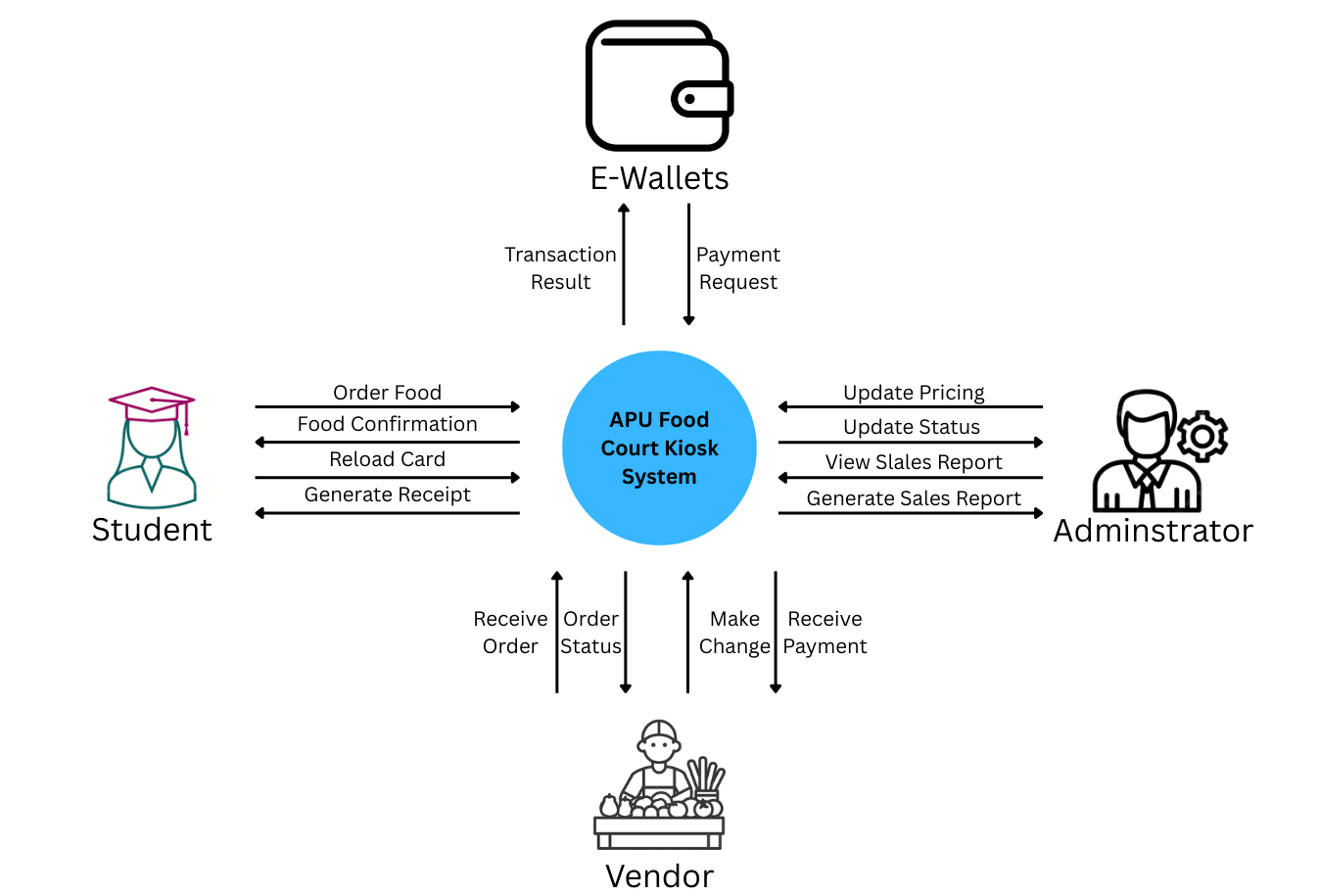


Figure 1 Context Diagram

## 1.4 Scopes and Objectives

### 1.4.1 Project Scope

The scope of the APU Food Court Kiosk System is to focus on enhancing the food ordering and payment process for all the students included staff and others. This project aims to analysis, system design, development, and implementation of the self-service kiosk system for APU new campus wing food court. Besides, to address the identified challenges effectively, we will need to find the most practical and implementable solution among these solutions included: Digital Ordering System, Multi-Payment Integration, Real-Time Order Notifications, and others, etc.

However, to ensure that the plan is not delayed, some function need to be excluded to avoid project creep. For example, table service, mobile app development, advanced data analytics dashboards and facial recognition or biometric login are out of scope for this phase.

### 1.4.2 Project Objective

The primary objective of this project is to develop and deliver a functional self-service kiosk within three months and a budget of RM80,000. The system aims to expand available payment methods and convenience of payment and improve customer satisfaction by enabling multiple secure and fast payment options.

The content management system allows APU administrators and respective stall personal to manage menus, analysis sales and other details to ensure data accuracy, security and speed.

## 1.5 Project Scheduling

**Project Start**: 31 March 2025

**Project End**: 7 May 2025

This project uses the scrum methodologies to apply project scheduling because scrum can deliver early, be customer-centric and continuously improve based on feedback. Each sprint takes about a week to launch new features.

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 2 Gantt Chart

# 2.0 Analysis Phase

## 2.1 Product Backlog Creation

A white sheet of paper with text

AI-generated content may be incorrect.

Table 1 Product Backlog

## 2.2 Functional and Non-Functional Requirements

**Functional Requirements**

1. The system must read and verify the identity and balance of the student or staff card.
2. Users can top up their card balance via cash or supported e-wallets at kiosk.
3. Users can view available food menus from all food stalls on the kiosk screen.
4. Users can select food items and add it to the e-cart.
5. The payment will be deducted from the card balance and the transaction will be confirmed
6. Generate and print receipt with order details and estimated preparation time.
7. Orders are sent electronically to the corresponding food stall for preparation.
8. Admins and vendors can update menus, prices, and availability via CMS.

**Non-Functional Requirements**

1. The system needs to respond to user input within two seconds.
2. Kiosk system must be operational 95% of campus hours, example 8AM to 10PM.
3. System must allow easy updates to menu and payment integration modules.
4. The system must ensure that orders are not lost or duplicated during processing.
5. The interface must be simple and intuitive at every step.
6. User data must be encrypted.
7. The system should handle at least 10 concurrent transactions at the kiosks during peak hours.
8. The system must be backed up daily and support data recovery in the event of a system crash.

# 3.0 Design Phase

## 3.1 Use case and System Modelling

A black and white screen with white ovals

AI-generated content may be incorrect.

Figure 3 Use Case Diagram of APU Food Court Kiosk System

**Use Case Specification**

A close-up of a document

AI-generated content may be incorrect.

Table 2 Confirm Order Cart Use Case Specification

A white sheet of paper with text

AI-generated content may be incorrect.

Table 3 Received Order Use Case Specification

A white and black document with text

AI-generated content may be incorrect.

Table 4 Monitor System and Analytics Use Case Specification

# 4.0 Development Phase

## 4.1 Prototype

A screen shot of a phone

AI-generated content may be incorrect.

Figure 4 Kiosk Main Page

A screenshot of a cell phone

AI-generated content may be incorrect.

Figure 5 Select Order Food or Top Up Page

A screenshot of a mobile payment

AI-generated content may be incorrect.

Figure 6 Top-Up Screen

A screenshot of a cash register

AI-generated content may be incorrect.

Figure 7 Cash Payment Method

A qr code on a white background

AI-generated content may be incorrect.

Figure 8 Payment - QR Scan Screen

A screenshot of a menu

AI-generated content may be incorrect.

Figure 9 Food Selection Screen

A screenshot of a menu

AI-generated content may be incorrect.

Figure 10 Order Details Screen

A screenshot of a mobile payment

AI-generated content may be incorrect.

Figure 11 Payment Success Confirmation

A screenshot of a phone

AI-generated content may be incorrect.

Figure 12 Top-Up Success Confirmation

A screenshot of a login form

AI-generated content may be incorrect.

Figure 13 CMS Login Page

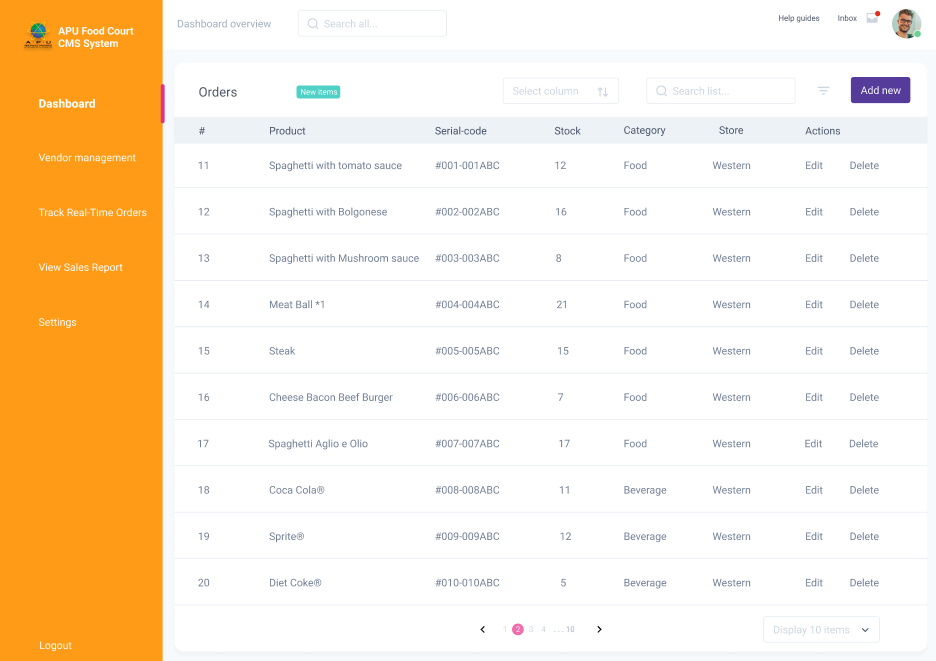


Figure 14 Admin Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

Figure 15 Admin Vendor Management

A screenshot of a computer

AI-generated content may be incorrect.

Figure 16 Admin Track Real-Time Orders

A screenshot of a computer

AI-generated content may be incorrect.

Figure 17 Admin View Sales Report

A screenshot of a computer

AI-generated content may be incorrect.

Figure 18 Vendor Dashboard

A screenshot of a web page

AI-generated content may be incorrect.

Figure 19 Vendor Profile

A screenshot of a computer

AI-generated content may be incorrect.

Figure 20 Vendor View Customer Feedback

A screenshot of a computer

AI-generated content may be incorrect.

Figure 21 Vendor Manage Menu

**Design choices / Design principles**

The design principles used in the APU Food Court Kiosk prototype are all screens use uniform fonts, colours, and button styles to maintain consistency, which reduces the cognitive burden on users and creates a predictable user experience. Important elements such as titles and prices are in bold, eye-catching fonts, while secondary actions are in smaller fonts, thus achieving visual hierarchy to attract users' attention. Next, the use of intuitive graphics and brand logos on the payment and menu pages allows users to understand without reading detailed text, realizing the principle of "recognition is better than recall". The layout of the interface is based on a simple concept design, such as large buttons that are easy for users to see and each screen only displays the most relevant elements, greatly reducing the possibility of users touching the wrong button. When users "pay successfully" or "recharge successfully", they will be redirected to the success screen to provide users with instant feedback, letting them know that the operation has been completed and the next step. Finally, the back button and editable electronic shopping cart on each screen consider that users may change their minds or press it accidentally.

## 4.2 Testing Strategy

**Static Testing**

Static testing is to review system documentation, design, and code without running the software in the early development process. For example, review the documentation of the APU food court kiosk system to ensure that the functional characteristics of ordering and recharging cards are correctly defined, and check whether the logic of the use case diagram, UI model, and design model is consistent with the project. The purpose of static testing is to find errors as early as possible before the code is executed to prevent defects after entering the development environment and increase costs. It also helps to verify the integrity of the system specifications.

**Dynamic Testing**

Dynamic testing is to execute the software code after the system is built to find defects. It includes unit testing, integration testing, system testing, etc. In the APU food court kiosk system, the purpose of unit testing is to ensure that each small function can run independently, such as testing the function of deducting money from the member card. Integration testing is to ensure that components or subsystems interact smoothly without errors, such as checking whether the order module correctly sends data to the vendor terminal. System testing is to ensure that the entire system meets functional and non-functional requirements, such as testing students inserting membership cards, recharging, ordering food, paying, and receiving receipts. The purpose of dynamic testing is to find errors that may occur during operation and verify that the system behaviour is the same as expected.

**Regression Testing**

Regression testing is the process of updating or fixing bugs to test whether the system's previous functionality works properly. For example, we fixed a bug in the e-wallet reload process and then ran regression testing to ensure that the ordering and payment functions were still working. The goal is to make sure that the new code doesn't break existing functionality and is critical for agile projects with frequent updates.

## 4.3 Develop an initial Test Plan

A white sheet of paper with black text

AI-generated content may be incorrect.

Table 5 Test Plan

# 5.0 Deployment Phase

## 5.1 Deployment Strategy

As we evaluated the most appropriate deployment strategy for the APU Food Court Kiosk System, we compared four common deployment strategies: Direct Cutover, Parallel Deployment, Pilot Deployment and Phased Deployment. Each strategy has its own characteristics with varying degrees of risk, cost, and control.

A white table with black text

AI-generated content may be incorrect.

Table 6 Deployment Strategy Comparison

Based on the comparison, pilot deployment is the most suitable for APU because this method ensures quality, user feedback and system stability.

**Step to carried out the APU Food Court Kiosk System with Pilot Deployment:**

**Choose and Set Up the Pilot Site:**

A section within the new campus food court will be marked as a ‘pilot site’ where the APU kiosk system will be integrated with one to two kiosks and/or food stalls. We may invite some students and staff volunteers to provide initial feedback after testing the system.

**System Install Procedures:**

In this phase, the APU Food Court Kiosk System will be deployed at the pilot site, completing all hardware, software, and necessary backend relations at the food counters. During this phase, the APU campus-wide ordering system (“the old system”) will still be in operation in other parts of the campus, providing a fallback.

**Training, Operation, Supervision:**

* **Training:** Users from the food stalls to be located at the pilot site will be trained on basic order and menu controls using the CMS Teaching Module. Briefings will also be given to relevant customized user groups.
* **Operation: Users:** A new kiosk system will operate at the pilot site for 2-4 weeks to gather operational and experiential data.
* **Supervision:** System transaction accuracy and stability will be monitored alongside user and vendor feedback from the pilot site.

**Evaluation and Improvements:**

We will evaluate the identified gaps, system shortcomings, and usability issues using data and feedback from the pilot feedback. Appropriate changes will be implemented afterwards.

**Full Rollout:**

We will deploy the system to the other areas of focus, including the new campus wing food court and specified locations, after completing pilot testing and all optimizations. These areas would most likely be subject to a direct cutover or a phased approach with direct cutovers for each phase, as described in our course readings.

This pilot method allows for new system changes in one location while the old system continues to operate in another. This approach strikes a balance between the risk associated with an immediate direct cutover to the entire new wing and the expense of full parallel operation, blending elements of both approaches and building confidence toward a full system rollout.

## 5.2 Final Sprint Review and Retrospective

The 12-week project adopted Scrum as an Agile methodology. We created sprint planning meetings to break down tasks before deciding which users' stories to prioritize for our sprint backlog. We conducted sprint reviews bi-weekly to show our progress to teammates and mock stakeholders while receiving feedback for improvement. The criteria noted in our sprint retrospective demonstrated some significant areas of focus, such as multi-language support, improvements on kiosk interface, etc. We highlighted some major challenges in our retrospectives such as updating our task status post sprint cycle, no clear task ownership of tasks because they were never truly assigned, and some team members were just putting in fewer hours over the sprint cycle. We learned we should probably set a daily standup meeting, so we knew who was doing what when engaging in our sprint cycle. Otherwise, our team worked well communicating with each other and growing accustomed to the idea of using boards to communicate what was to be done in our check-in and follow-up meetings, etc. In conclusion, the agile and scrum processes allowed our team more flexibility to make tangential adjustments, while soliciting feedback from our instructor and other program members, along with flexibly allowing improvements for a better user experience, Ultimately, our prototype was delivered by our deadline, and we were able to implement our major goals outlined in the parameters of our scope which included the bridge functionality of food ordering, isolated payment integration, and menu management.

# 6.0 User Story Creation and Processing

## 6.1 Yaw Kai Yuan TP080121

**User Story 1**

As a student, I want to reload my card using e-wallet at the kiosk, so that I can use the card to order food.

**Acceptance Criteria**

1. The kiosk allows the student to select an e-wallet option such as Touch 'n Go, Boost, GrabPay after the card is inserted.
2. Users can enter the amount they want to reload or select an amount such as RM10, RM20, RM 50.
3. The kiosk created a QR code for the selected e-wallet payment.
4. Once payment is successful, the card balance is updated.
5. If the payment fails, the kiosk will show that the transaction failed and the balance will not be deducted.

**User Story 2**

As a food stall vendor, I want to receive real-time electronic orders from the kiosk, so that I can start preparing the food immediately and not waste time.

**Acceptance Criteria**

1. When a customer places an order, the order will be sent to the corresponding stall’s system.
2. The vendor can view order details including item name, quantity, special notes, timestamp.
3. Orders are listed in chronological order on the kiosk, with the most recent orders being the most prominent.
4. A new order causes a notification sound or alert.
5. Vendor can mark an order as "In Preparation" or "Ready for Pickup".

## 6.2 Er Chen Liang TP080254

**User Story 1**  
As a student, I want to use TnG e-wallet to do my payment for food at the kiosk. So that I can do the payment without using the AP card.

**Acceptance Criteria**

1. The kiosk should have the payment method of the TnG e-wallet.
2. After selecting the TnG method, the system will display a QR code for the student to scan.
3. The students will receive an e-receipt after completing the payment.
4. The balance of student will be updated.
5. If the payment is unsuccessful the apps will pop out an error message, indicating the transaction has failed, and no balance will be deducted.

**User Story 2**

As a food stall vendor, I want to send a message to the order sender when the food is ready to be served, so that the person knows when to pick up their order.

**Acceptance Criteria**

1. When the food is ready, the vendor should have an option to mark the order as "Ready for Pickup" in the system.
2. Once marked, the system will send a notification message to the customer’s, informing them that their food is ready.
3. The customer will receive a pop-up message on their kiosk screen.
4. The message will stay on the customer’s kiosk screen until the customer acknowledges the notification.
5. Once the food is picked up, the customer should press the “picked up” button at their kiosk screen and the status of the order should be marked as "Completed".

## 6.3 Chong Wai Kit TP080388

**User Story 1**

As a student, I want to save my favourite food orders on the kiosk, so that I can quickly reorder them without selecting everything again.

**Acceptance Criteria:**

1. The student system enables order saving as “Favourites” after placing the order.
2. All saved favourites contain every item detail and quantity information along with custom options.
3. The kiosk main menu contains a “Favourites” button which displays previously saved orders linked to the current cardholder.
4. Students can locate saved favourites under the main menu and execute one-tap ordering for previously selected items.
5. Users possess the ability to rename and delete their saved favourites whenever they want.
6. The person who saved the order maintains exclusive rights to view or access this information.

**User Story 2**

As a food stall vendor, I want to view order trends and item popularity statistics, so that I can better plan inventory and staff preparation.

**Acceptance Criteria:**

1. From the vendor dashboard users can access charts which display the three different time periods of most-ordered items: daily, weekly, and monthly.
2. The system enables vendors to search data through date, time and item category parameters.
3. The system displays peak ordering periods and shows which items generate the lowest performance.
4. The system allows users to download exportable reports in CSV and PDF formats.
5. System data updates automatically whenever new orders get added to the system.
6. Each vendor receives access to analytics data that applies only to their individual stall.

## 6.4 Gan Kai Hong TP080661

**User Story 1**

As a student, I want to customize my order at the kiosk, so that I can collect a meal that fits my requirements.

**Acceptance Criteria**

1. The kiosk system permits users to opt for modifications where applicable (e.g. milder spice, no onions, less oil).
2. Customization options are marked relevant to selection and selection indications and marked with some type of symbol.
3. Chosen customizations are visible in the summary of the order before proceeding to payment.
4. The system accurately describes customizations to the food stall vendor's system.
5. All customizations detailed on the receipt can be used to verify ordered items.
6. Each food item has reconcilable and applicable modification options which are relevant.

**User Story 2**

As an APU administrator, I want to edit and set new prices for menu items on a secure content management system (CMS) interface, so that I can keep the accuracy in the food court system.

**Acceptance Criteria**

1. The CMS offers role specific login for food court managers and stall vendors with appropriate permissions.
2. Information such as Food Categories and Vendor’s details can be changed by administrators where new ones can be added.
3. Audit logs shall be created by the system documenting actions done on menu and price changes with timestamps.
4. After approval, all kiosks update instantly with the new changes made to the menus.
5. The system follows approval workflows, thereby preventing unauthorized price alterations.
6. If necessary, administrators have the capability to hide certain menu items or entire stalls on a temporary basis.

## 6.5 Ting Jac Sheinn TP080719

**User Story 1**

As a student, I want to switch the kiosk interface to my preferred language, so that I can easily navigate the menu and place orders without confusion.

**Acceptance Criteria**

1. As an international student, I want to change the interface at the kiosk, to my preferred language to easily navigate and place orders at the kiosk.
2. The kiosk has a language selection option (e.g. English, Mandarin, Malay) on the main screen.
3. Changing the language will change all text (menu, buttons, instructions) within 3 seconds.
4. The selected language will remain active until the user's session has ended at log out or they place their order.
5. The interface has pictorial icons along with wording to assist users who are unfamiliar with the language. The language switch can be accessed from the main menu within 1 click.

**User Story 2**

As an APU administrator, I want to monitor the kiosk system performance in real time, so that I can quickly detect and address technical problems and keep operations running smoothly.

**Acceptance Criteria**

1. The CMS provides an intuitive display view of the real time state of the kiosks (e.g. online, offline, transaction issues).
2. The CMS dashboard displays many other metrics, such as average transaction time and average error rates, all updated every minute.
3. The administrator will receive alerts sent to the CMS if the kiosk goes offline, or if there is a critical error (e.g., payment failure).
4. The CMS will also allow the administrator to remotely restart a kiosk in case a minor issue prevents a customer from making a transaction.
5. Only authorized administrators using secure login can access the monitoring feature.

## 6.6 Huang Xiao Shun TP077723

**User Story 1**   
As a student, I want to enable a dark mode on the interface so that it’s easier on my eyes in low-light environments.

**Acceptance Criteria:**

1. Kiosk provides a "day/night mode" button that users can switch manually.

2. Night mode uses a dark background, light fonts, and reduces the screen size.

3. After switching, all UI pages will maintain a consistent style until the order is completed.

4. The system uses day mode by default.

5. User switching mode does not affect the selected menu or order content

**User Story 2**As a student, I want to recommend a drink after I add a meal to my cart, so that I don’t forget to order a drink with my food.

**Acceptance Criteria:**

1. When the user adds a main meal, the system automatically pops up drink recommendations (for example, "Would you like a cup of iced lemon tea?")

2. Users can add recommended drinks to their shopping cart with one click or skip them.

3. Recommendations are based on past sales data or dish types (for example: fried chicken rice is often paired with milk tea).

4. Recommendations only appear once and the user can dismiss the prompt.

5. Recommended features can be enabled/disabled by the backend

# 7.0 Agile Principles and Reflection

## 7.1 Yaw Kai Yuan TP080121

### 7.1.1 Agile Principles

**1. Customer satisfaction by early and continuous delivery of valuable software**

This principle means to deliver effective system functions regularly in the weeks after the project starts and then continuously improve the system based on customer feedback rather than waiting until the project is completed. The development team can fix system errors and continuously enhance the system's functions based on this principle.

Delivery of some small, usable features of the APU food court kiosk system in the early stages such as basic reload functions, payment and sending order. In addition, let students and staff test early versions on few kiosks to gather their feedback so that it can be used as a reference to improve the system. Also, receiving customer feedback early on allows the development team to adjust the project and avoid wasting time developing unnecessary features. Development team can prioritize valuable features. For example, students may care more about fast reloading but less about receipt printing, so they can focus on the features that customers value most.

**2. Projects are built around motivated individuals, who should be trusted**

This principle means assigning proactive and capable team members and trusting their decisions. Provide them with the necessary environment to unleash their creativity instead of controlling their every step. If there are other factors that are not conducive to team members, they need to be adjusted.

In this project, the development team can decide on the best technical solution and trusted the UX designer to create a simple and easy-to-use interface without having to adhere to the rigid design of non-technical management. Furthermore, trust the food court stall owners’ feedback on how the order taking system can better fit into the kitchen process. Proactive team members can spot small issues with the system, such as a confusing menu layout, and fix them immediately, without waiting for approval from others, which would slow down the entire project. This reduces bureaucracy and allows the development team to adjust based on the implementation.

### 7.1.2 IS Methodologies

**Scrum Methodologies**

Scrum is fast-paced and requires the participation of the entire team to develop and deliver products in a short period of time, which is called a "sprint". The idea of ​​Scrum is that the team can control their own workflow, develop in an incremental way, and focus on continuous improvement, teamwork, and customer feedback. The following is the Scrum core elements:

**Sprint**

The sprint is a time-controlled, small project that is usually 2 to 4 weeks long and can be used by a team to implement a specific part of a system. It delivers a small part of the project in a certain time frame. Once the scope of a sprint is confirmed, it cannot be changed by anyone.

**Product Backlog**

The Product Backlog is a list of all features, bugs, enhancements, and tasks for the system that is managed by the Product Owner. This list is constantly updated and challenged based on feedback and progress.

**Sprint Planning**

Sprint planning is a process that takes place before each sprint and requires the participation of all Scrum roles. The planning phase is used to determine what the team will build during the sprint. The team selects the highest priority items from the product backlog as the sprint goal.

**Daily Standup**

The Daily standup is a 15-minute meeting with the development team every day to report on progress. This phase allows team members to communicate with each other about what has been done, what is planned, and what obstacles there are. The purpose is to ensure that everyone is on the aligned and to detect problems as early as possible.

**Sprint Review**

The Sprint Review is a phase that is conducted after every sprint. The development team presents the works completed during the sprint to the stakeholders and collects their feedback to improve the system. The purpose is to ensure that user expectations are met.

**Sprint Retrospective**

The sprint retrospective is a private meeting between team members. This phase allows team members to reflect on what went well, what did not go well, and how to improve in the next sprint. Retrospectives are a way to continuously improve the team's processes and communications.

**Scrum Roles**

Product Owner: Prioritize backlog items on behalf of users or stakeholders such as APU Project Manager.

Scrum Mater: Facilitates the SCRUM process, removes blockers, coaches the team.

Development Team: The team responsible for designing, building, and testing a product.

**The pros and cons of using Scrum methodologies**

The pros of using scrum in APU Food Court Kiosk System is fast and incremental delivery. For example, you don’t have to wait until the complete system is delivered, which is very time-consuming. Scrum can deliver usable functions step by step. Next, it is easy to integrate feedback such as students, staff and vendors to test the system early and collect feedback that can improve the system in real time. If priorities change, for example, students want to order food using a QR code, it can be adjusted in the next sprint.

The cons of using scrum in APU food court kiosk system is that it requires a dedicated, experienced team. For example, an inexperienced development team can cause scrum to fail. Next, it requires stakeholder involvement. If the APU admin and vendor are not actively involved, the project progress will be suffered. Also, the user is not clearly defined, the team may waste time developing the wrong features lead to poor project efficiency.

## 7.2 Er Chen Liang TP080254

### 7.2.1 Agile Principles

**1. Sustainable development, able to maintain a constant pace**

This principle aims to promote the team to maintain a stable and balanced workflow in a long term. In this project, we must carefully plan the work and set sprints with a realistic goal to ensure that team members would not be inefficient due to overload. For example, tasks such as designing product service interfaces, developing backend order processing, and testing payment functions are evenly distributed on the project timeline instead of requiring employees to rush the project out at the same time.

In addition, maintaining a steady pace prevents burnout and keeps the team motivated. When problems arise (e.g., a payment is completed with an error), we can identify and resolve the problem efficiently without disrupting the overall progress. This approach can help us keep moving forward while ensuring that the system remains stable and adaptable to changes throughout the development process.

**2. Simplicity-the art of maximizing the amount of work not done-is essential**

This principle aims to reduce the unnecessary design and prevent the project design becoming complicated and hard to use. In this project, we can save the time on the designing feature that was not important to the operation of the food court, thereby improving the project’s work efficiency and enabling the product to be launcher faster. For example, we focus on the core function of the project such as ordering, top-up, and payment.

Designing a simple and clean user interface of the kiosk system can make it easier for vendors or student to use. This not only shorten the development time but also improves the system performance and minimize the complex design of the system.

### 7.2.2 IS Methodologies

**Rapid Application Development (RAD) Methodologies**

RAD is an agile methodology that focuses on rapid prototyping and iterative development to deliver functional software quickly. Unlike the traditional software development methods, RAD focus on rapid design, user feedback, and continuous improvement of the products. RAD is suitable for a project which have a tight timeline and a constantly changing system.

**Core Elements of RAD**

**Prototyping**

Prototyping is the base of RAD. A major working prototype of the system (with some basic functionality) is usually built quickly and presented to stakeholders for immediate feedback. His prototype will be continuously improved based on the feedback provided, and new features will be added over time.

**User Feedback**

The most important element of RAD is continuous user feedback. Students, faculty, and food stall owners interact with the prototype at every stage, giving advise like what needs to be improved and providing valuable insights. This enables developers to enhance the system based on user needs before the final version of system is completed.

**Timeboxing**

Timeboxing is about setting a specific time limits for the development of a prototype or feature. This approach helps to ensure that development work proceeds as planned and that all features are prioritized and delivered within the scheduled deadlines.

**Iterative Development**

RAD uses an iterative cycle that enables teams to focus on small, manageable components of the system and complete them in a short period of time. Each iteration adds more features or improves existing features based on user feedback.

**Component Reusability**

RAD encourage the reuse of existing components to speed up the development process. Instead of building all simple functions from scratch, developers can integrate existing components, thereby reducing development time and cost.

**RAD Roles**

**End Users:** Participate in the development process and giving continuously feedback of the prototypes.

**Project Manager:** Ensure that funding and time budgets are adequate and the stakeholder feedback and suggestions are appropriately incorporated.

**Developer:** Responsible for building and refining the prototypes based on user’s feedback and implementing final functionality.

**The Pros and cons of using RAD methodologies**

The pros of using the Rapid Application Design (RAD) methodologies in the APU Food Court self-service terminal system is that RAD focuses on prototyping and iterative feedback, which enables the rapid production of usable software. For example, the APU Food Court self-service terminal system can quickly launch basic functions such as ordering and recharging and then continue to improve and expand them. In addition, the participation of users throughout the development process can make the system closer to their expectations.

The cons of using the Rapid Application Design (RAD) methodologies are that RAD focuses on rapid development and prototyping, which can lead to limited scalability. So, if not planned properly, it will be more difficult to scale the system to accommodate a larger user base. As we know, the RAD focuses on speed and flexibility, so sometimes quality control may not be ideal to meet deadlines. However, this usually can be mitigated by strong testing and quality assurance practices throughout the development process.

## 7.3 Chong Wai Kit TP080388

### 7.3.1 Agile Principles

**1. Best architectures, requirements, and designs emerge from self-organizing teams**

This principle indicates that effective solutions emerge from teams that organize their work independently instead of needing constant supervision. A self-organizing team takes charge of deciding its work methods instead of requiring guidance from higher authority. Teams that include diverse capabilities from their members produce more useful solutions through their wide-ranging perspectives. These teams work together closely which enables them to determine both the true product requirements and the most suitable building methods effectively.

Such teams demonstrate quick responsiveness to changes because their architecture and design and requirements transform in optimal ways. Self-organizing teams use team collaboration along with constant feedback and strong communication to transform their products during their development process. Agile places its trust in team members to produce high-quality work without needing continuous supervision from outside sources.

**2. Regularly, the team reflects on how to become more effective, and adjusts accordingly**

This principle focused on continuous improvement. It means that teams take time—usually at the end of each sprint—to review how they worked, not just what they delivered. This reflection typically happens during a retrospective, where team members discuss what went well, what didn’t, and what could be improved. The goal is not to blame anyone, but to learn and grow together.

From these discussions, the team identifies specific actions they can take to improve their process, tools, or communication. These changes are then tested in the next sprint, creating a cycle of constant learning and adaptation. By doing this regularly, the team becomes more efficient, more collaborative, and better at delivering value. This principle supports a culture of ownership, transparency, and growth.

### 7.3.2 IS Methodologies

**Spiral Methodologies**

Spiral Methodologies brings together Waterfall model structural elements and prototyping iterations to create a development methodology. Risk assessment together with risk management directs the development process in this approach. The model advances the system development through multiple cycles of planning followed by risk analysis then engineering and evaluation.

The software development process through each spiral delivers growing software versions after customer input and risk mitigation activities. The Spiral Model demonstrates optimal suitability for projects that involve large complexities and high risks since requirements often change during development.

**Core elements of Spiral**

1. **Development Phases in Spirals**

The project consists of multiple repeated cycles named spiral which correspond to development phases starting from planning through risk analysis until engineering and evaluation.

1. **Risk Management**

Risk identification and analysis takes place at each spiral stage of development. Risk reduction strategies are developed for the most critical risks before advancing to the next cycle of development.

1. **Prototyping**

The development of either prototypes or partial implementations exists to validate design approaches and solutions.

1. **Iterative Refinement**

The system develops additional details during each spiral cycle while it reduces unknown elements.

1. **Customer Involvement**

The evaluation process with clients occurs at each spiral’s conclusion for assessment purpose before starting new development.

1. **Documentation and Planning**

The development cycle generates thorough documentation which helps both traceability and future spiral development planning.

**Spiral Roles**

**Project Manager:** The individual manages spiral scheduling along with progress tracking and stakeholder communication activities.

**System Analyst:** The professional aids clients in requirement definition while also identifying areas at risk.

**Risk Analyst:** The team evaluates possible technical and business risks through developing mitigation strategies.

**Developers/Engineers:** Development teams need to implement features and prototypes throughout the engineering period of each spiral.

**Client/Customer Representative:** The team member gives feedback about prototypes while validating designs and assists in requirement prioritization.

**Quality Assurance (QA) Team:** Quality standards must be evaluated for each cycle output.

**Pros and cons of using Spiral Methodologies**

The Spiral Model delivers major benefits which specifically suit large-scale or high-risk projects. Early continuous risk evaluation and mitigation stands as the main strength of this model because it helps organizations avoid costlier mistakes later in development. The model operates through multiple cycles which enable requirements to advance progressively during project development. When customers become involved after each phase the product stays in line with their expectations and enables rapid feedback that leads to early adjustments. The model enables prototyping capability which improves design decision validation before system deployment.

The Spiral Model presents some drawbacks during its application. Skilled teams trained in risk assessment and planning methods must handle this system because it has complex management requirements. The Spiral Model does not work well for small projects or those operating on limited budgets since the time and cost needed for multiple iterations and documentation exceeds potential advantages. A fixed timeline would prevent scope creep when the process remains tightly controlled because the project duration remains undefined. Through each spiral cycle the model requires major documentation as well as thorough review processes that might create delays unless project managers handle these stages effectively.

## 7.4 Gan Kai Hong TP080661

### 7.4.1 Agile Principles

**1. Welcome changing requirements, even late in development**

This principle captures that user expectations and business prerequisites change, and agile takes care of accommodating shifts even toward the end of the development phase. It emphasizes satisfying a customer’s demand over following an inflexible plan.

In regards to APU Food Court Kiosk System, this principle proves useful as ordering preferences for food along with payment options constantly change. For instance, if it turns out that students prefer QR code scanning during the testing, any card-only systems designed can be altered to meet this new expectation. Likewise, if the popularity of a new e-wallet payment option surges during the development stage, its integration can be provided with minimal disruption to the overall system.

This adaptability enables the development team to interact with actual users (students, faculty, and food vendors) and incorporate their suggestions into the subsequent iterations of the system. Users suggest feature enhancements for inventory management to vendors or display nutritional information, and these suggestions can be implemented to improve the system instead of denying them because they were outside of scope.

**2. Face-to-face conversation is the best form of communication (co-location)**

This principle stresses that interaction conducted in a person-to-person basis captures, conveys, and resolves all matters pertaining to issues communication in a project team more effectively than any other means. It understands that immediate interaction is best for addressing sophisticated matters and requirements.

With reference to the APU Food Court Kiosk project, implementing face-to-face communication equates to regularly bringing the developers, UI/UX designers, and APU administrators as well as food stall vendors together in one physical space. For instance, having onsite meetings at the food court enables the developers to see the students’ engagement with the existing ordering systems and discuss ways of improving them with the food vendors. This co-location policy promotes immediate feedback processes whereby any misunderstanding of requirements is rectified instantly.

In regard to the food order customization options, face-to-face conversations between developers and food stall operators portray their perceptions on the proposed customizations on what can actually be done within a busy kitchen. Similarly, engagement with the students concerning their preferred payment methods through direct conversation employs features that would otherwise be left through remote communication or surveys. This principle streamlines processes and reduces the bureaucracy while expediting the decisions made thus fulfilling the needs of all stakeholders with a kiosk system that operates under APU campus.

### 7.4.2 IS Methodologies

**Extreme Programming (XP) Methodology**

This focuses on software development as it incorporates frequent changes and updates from the user. Defined customer requirements necessitate focusing on customer satisfaction and frequent updates.

Core Elements of XP:

**Pair Programming**

Pair programming is when one developer codes, and the other checks the appropriate and optimal use of set paradigms in real time. Quality of the code increases due to review, knowledge is exchanged within the team, and the number of defects is reduced. For setting up the system for a kiosk, pair programming would be crucial in the consideration of basic parameters like payment processing to ensure that testable parameters do not yield insecure or unreliable test results.

**Test Driven Development (TDD)**

Through TDD, programmers can check the functionality of features when they set tests and perform verification and validation tests. For a kiosk system, TDD will verify that claim parameters such as order customization are operable under specific constraints.

**Continuous Integration**

Automated testing integrates new code changes and checks if test failures can occur, which helps in identifying errors in the system. This approach preserves the functionality of the system in a working state.

**Small Releases**

As part of extreme programming, useful system features are released in small increments rather than complete builds. The full kiosk system can be built incrementally, starting with bare-bones ordering and progressing towards advanced customization and loyalty program functionalities.

**Straightforward Design**

XP discourages adding additional features beyond what is required, thereby helping the design overengineering. This principle coincides with maximizing the amount of work not done.

**The Pros and Cons of using XP Methodology**

The XP positives regarding APU Food Court Kiosk System include code quality focus (affirmed by TDD and pair programming) which fosters suitable creation of a payment processing system. XP’s customer involvement focus will help to address issues so that the system is usable and useful for both students and vendors. Additionally, the system's deployable state throughout development makes the Continuous Integration servicing technique beneficial as it removes all integration issue bottlenecks during system development. Short feedback intervals greatly increases responsiveness to evolving regulations, an important trait in a fast-paced university setting.

Discipline and resource cost are cons of XP: XP necessitates all TDD and continuous integrative routines to be strictly adhered to; thus while repetitively paired programming worsens resource spendings, demand for strict adherence maintains discipline. The need for all AV teams to work from the same locality at once to utilize Windows-only XP-supporting hardware creates team fragmentation burdens. All other team members working remotely makes unified teamwork impossible under the XP branding umbrella. Simplicity is essential under XP, creating issues with complexity and robust constructions mandatory within payment handling kiosk systems. Lastly, staff and students being busy persons makes credible full time sponsorship support unavailable, a major issue for Active Customer Presence wants.

## 7.5 Ting Jac Sheinn TP080719

### 7.5.1 Agile Principles

**1. Continuous attention to technical excellence and good design**

For APU Food Court Kiosk System, we use a modular architecture (kiosk UI, backend, CMS) to focus on good design. For the intent of implementing a multi-language switching feature (international student story), we focused on UI response time (< 3 seconds) and intuitive icons to address the user pain point of “complex interface." To allow for real-time monitoring from the CMS perspective (administrator story), we introduced role-based access control and automated errors alerts to increase the systems' reliability. At the end of each Sprint, we lead technical code reviews and required UI usability tests to facilitate methods of maintaining a high level of technical standards, reduce error instances while continuing to enhance system functionality, and iterating quickly on features (e.g., ability to add additional language support).

**2. Working software is the principal measure of progress**

For the APU Food Court Kiosk System, we deliver prototypes every 2 weeks through Scrum Sprints. For example, in week 3, we introduce the multi-language switching interface, addressing the "complex interface" pain point. In week 5, we deliver the CMS monitoring function, allowing the APU staff to monitor and manage problems to improve reliability. Through Sprint Reviews, APU administrators and students are testing prototypes and providing feedback on the smoothness of the UI or the timeliness of monitoring alerts. This user feedback helps guide optimizations such as fine-tuning the language-switching logic or modifying the frequency of alerts, ensuring user needs are being satisfied and allows the problems identified during the Planning Phase, like interface complexity, to be resolved quickly within a fixed 12-week timeline.

### 7.5.2 IS Methodologies

**RUP Methodologies**

The Rational Unified Process (RUP) is a methodology for software development and project management that provides a disciplined and structured approach for delivering high quality software on time and on budget. RUP is prescriptive as compared to lightweight Agile methods, but it nonetheless supports iterative development with incremental delivery by supporting phases and change. RUP is based on a large framework of best practices or process patterns that provide a clear and precise definition of the 'What' and 'When' and ensures that the project deals with 'Why' regarding decisions made in accordance with business goals instead of wasting money, effort, or resources. An organization using RUP is likely to have projects that require heavy analysis, architectural design, and documentation, but supports, and indeed provides an appropriate approach to progressively elaborating a solution through close and continuous collaboration with stakeholders and end-users.

**Core Elements of RUP**

**Phases:** RUP consists of four major phases: Inception: during this phase the scope of the project and relevant requirements are defined and viability is assessed; Elaboration: in this phase the analysis is done in depth, robust architecture is identified and developed on, and risk is reduced on key aspects; Construction: the system is developed in increments and an integrated system is deliverable; and Transition: where the solution is delivered to end-users through testing, training, and so forth.

**Workflows: RUP** prescribes a few workflows that support the development focus including Business Modeling, Requirements, Analysis and Design, Implementation, Testing, Deployment, Configuration and Change Management, Project Management, and Environment.

**Iterative Development:** RUP emphasizes the iterative and incremental nature of development to support the ability to make refinements to the system. This differs from a traditional approach where the entire system is delivered at once, and this allows for the integration of lessons learned and adapting to changing requirements.

**Best Practices:** RUP prescribes certain best practices such as managing requirements, continuously verifying quality, developing iteratively, and using a component-based architecture.

**Roles in RUP**

**Business Analyst:** Collects and clarifies business requirements to match business needs.

**Project Manager:** Runs the project to ensure it stays on time, on cost and on scope. Team Leader: Leads the development teams to ensure efficient collaboration and quality.

**Solution Developer:** Develops the system, in an iterative way producing functional increments.

**Tester:** Validates that each iteration meets the quality of the requirements through continuous quality assurance.

**User Representative:** Represents end-users to provide feedback that ensures the product meets user needs.

**Pros and Cons Using RUP**

RUP uses a structured and disciplined process to help manage large complex software projects. RUP is iterative with no schedule. This enables developers to discover defects and risks early and mitigate negative impacts on their product. RUP has a clear set of phases and workflows, allowing aspects of the project to be managed separately; scope, time and cost are controlled more easily. RUP emphasizes documentation as well as generate traceability, maintainability, and creates better communication between team members and stakeholders. RUP can also be customized to fit specific organizational needs or sizes of projects.

While RUP has many positive attributes, it can be complex and cumbersome, especially where small projects and teams lack the experience with RUP. RUP has many different roles, workflows, and artifacts, which can add up to a lot of overhead and even higher documentation levels than other methodologies, which can hamper engineering progress and decrease agility. Applications of RUP, if careful about goals and baked into the culture and training, can lead to a large commitment of time and money as training, tools and methodology all vie for a say on the project. RUP loses its intended structure when bureaucratic agendas thrive and whole teams lose sight of the work of engineering in a pursuit of packaging engineering.

# 7.6 Huang Xiao Shun TP077723

### 7.6.1 Agile Principles

**Face-to-face communication is the most efficient and effective method of conveying information**

In Agile development, face-to-face communication is appreciated as the most effective method of exchanging information between stakeholders and developers. Written documents, emails, and other digital forms may often cause misunderstanding, incomplete feedback, or delay in clarifications. This principle became very much important in the APU Food Court Kiosk System user requirements gathering and prototype feedback sessions. Instead of relying solely on surveys or requirement forms, informal interviews were conducted, and real-time observations were undertaken with students and vendors of the food court.

This resulted in a series of reiterations centred around user needs, refinement in system interfaces during sprints, and validation of design changes on the grounds of human-centred feedback. Ultimately, a system was developed more intuitive and pragmatic toward real users' expectations in the APU campus environment.

**Working environments should support motivated individuals**

This principle focuses on facilitating and trusting an environment where highly motivated people can freely contribute. Agile teams prosper when every single member has the ownership of features and suggests some enhancements in one way or another. While practicing this particular principle, an inclusive team dynamic was cultivated, where acts of value were considered no matter the origin.

A prime example unfolded when the front-end developer suggested the Night Mode feature after a kiosk usage test was held in a dimly lit room. This feature was not in the initial scope, but the idea was quickly embraced by the team and included in the prototype. In a parallel fashion, the "Language Switch" feature came out of discussions with international students though it was not in the first backlog priorities and was soon designed and tested.

Empowering team members with autonomy in executing small innovations allowed them to go beyond ordinary and think through their design decisions to deliver improved user experiences along with a working prototype incorporating meaningful enhancements. Encouraging this internal motivation fostered rapid problem-solving alongside, and in harmony with, Agile values of flexibility, adaptability, and continuous improvement.

### 7.6.2 IS Methodology

Lean Software Development (LSD) is just a slimmed-down version of the Agile process, springing out of Toyota Production System techniques that aim to reduce wastes, concentrate on the most important features and refactor the product incrementally with user feedback. This kind of process fits best in small student-led projects like the development of a Food Court Kiosk System at APU that have practically limited resources and shorter time.

Following the principal tenets of Lean, we first identified an MVP with only very basic but essential features such as recharge, ordering, and payment. We consciously omitted user profiles or rewards or tracking order history from our early sprints. By restricting the scope creep, every iteration remained focused on delivering a clean functional prototype.

We also observed things in the real world to drive incremental development. For instance, while testing the ordering process, we noticed that users often skipped ordering drinks. We could have gone for a major uplift of the beverage menu, but instead, we chose to implement a small "Recommended Drink" prompt. That generated maximum value from the minimal amount of development hours. This single small and powerful feature is a perfect example of Lean's focus on "delivering value fast".

Furthermore, we always questioned the worthiness of every new feature idea. This helped us to reject unproven suggestions in order to improve existing features instead. By doing so, Lean empowered us to try fast, test early, and intelligently respond to feedback."

## 7.7 Overall Comparison

**A white sheet with black text

AI-generated content may be incorrect.**

Table 7 Methodologies Comparison

According to the above table, we compared the characterize of each method. Finally, we chose the scrum methodologies to apply to this project because it is the most balanced in flexibility, delivery speed, structure and team collaboration. In addition, scrum is also suitable for multi-role collaboration such as administrators, vendors, and users.

# 8.0 Design – System Design

## 8.1 Yaw Kai Yuan TP080121

A diagram of credit reload function

AI-generated content may be incorrect.

Figure 22 Class Diagram for Credit Reload Function

The Kiosk class is the central interaction point in the system that accepts customer card input, tops up, and selects payment method. The MemberCard class displays the credit balance held by the customer and the credit received from a reload transaction. The ReloadTransaction class to associate member cards, payment methods, and kiosks and generate receipts. The ReloadMethod is an inherited class. The following are the CashPayment class and the EwalletPayment class to show how the customer pays. The Receipt class to capture the transaction details and print them.

## 8.2 Er Chen Liang TP080254

**Start:**  When a customer tap on the kiosk system the system starts.

**Customer Browses the Menu:** The kiosk systemdisplays a menu for the customer to view available food and beverage items from different stalls.

**Customer Selects Item:** The customer interacts with the touch screen to select their desired items.

**Confirm Order (Decision): T**he customer can click on the yes and no button to confirm or continue their order when they are checking their order.

**Yes:** Proceed to payment.

**No**: Return to menu browsing.

**Select Payment:** After confirming the order, the customer chooses a payment method. This may include cashless options such as e-wallets, debit/credit cards, or student/staff cards.

**Process Payment:** The system initiates the payment transaction using the selected method. This step involves authentication, deduction, and transaction logging.

**Payment Successful:** The system will check if the payment was successful or not.

**Yes**: Print the receipt for the customer.  **No**: Display an error and redirect the customer back to the **“Select Payment”** step to try again.

**Print Receipt:** When the payment is successful, the system prints a receipt for the customer.

**End:** The transaction is complete. Then the system resets and is ready for the next user.

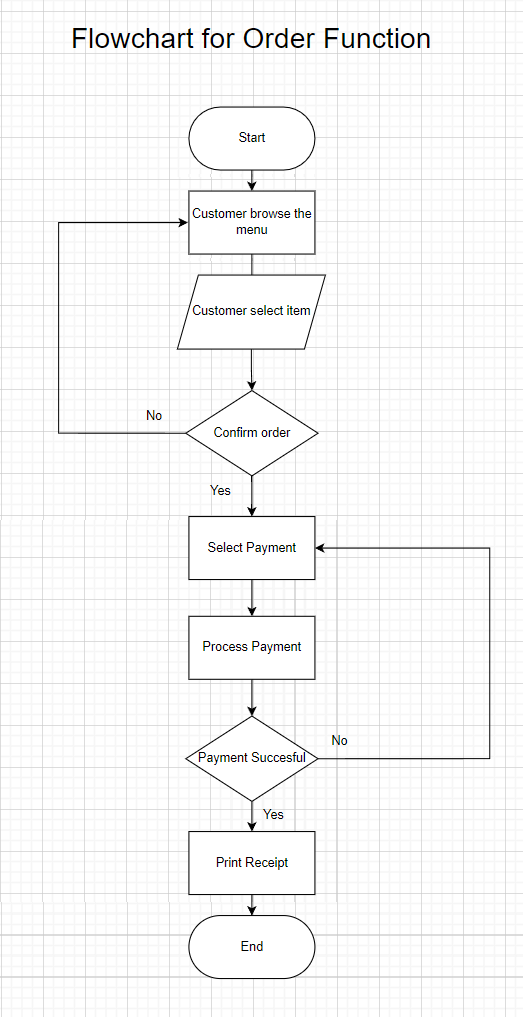


Figure 23 FlowChart for Order Function

## 8.3 Chong Wai Kit TP080388

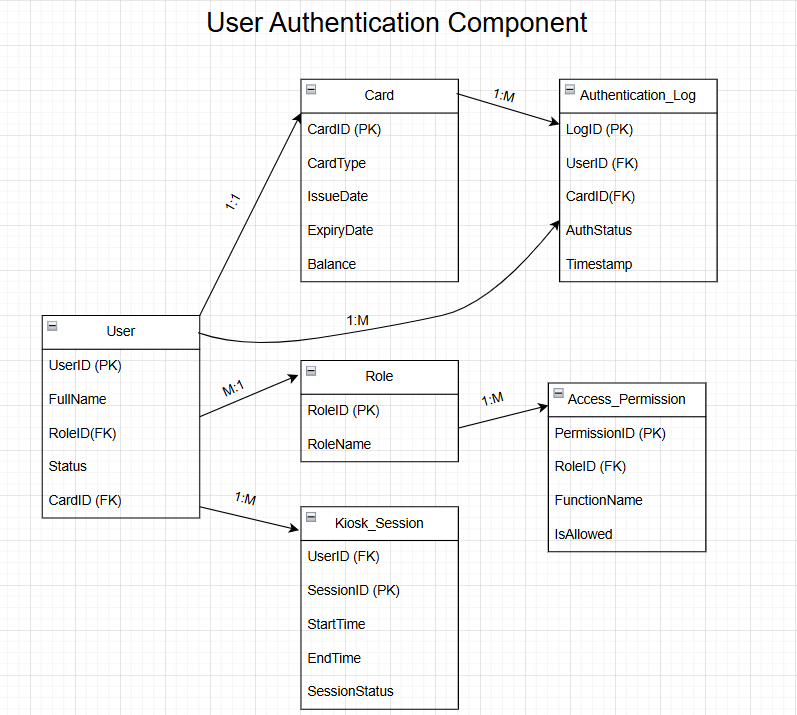


Figure 24 - User Authentication Component for Entity Relationship Diagram

The ERD shows how the user authentication component of Kiosk working. When a student or staff member places their ID card on the kiosk, the system cheques the card against the database. During verification, the system cheques if the card has an active status, belongs to someone who exists and is still valid. If the credentials are correct, the system saves the event and lets the user start using the main kiosk. If the user cannot be verified, the system will reject login session during the audit process. They are connected to access rules that set what tasks the user can carry out. All access to the system is stored in a log for review and session details are also recorded to monitor and secure the site. This ensures that only those who should be using the kiosk can do so and their actions are tracked.

## 8.4 Gan Kai Hong TP080661

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 25 Admin Price Update Activity Diagram

This section describes the procedure for an Administrator to change prices of menu items through the Content Management System (CMS) of the APU Food Court Kiosk System as shown in the Activity Diagram (Figure 14).

The workflow begins with the Administrator logging into CMS and then authentication by CMS System. The Administrator then goes to Menu Management and the CMS presents the pertinent interface. The Administrator chooses the particular Food Stall and Menu Item to be adjusted in price and enters the new price. This price is confirmed by the CMS System. If invalid, an error message is shown to the Administrator. If the price is valid, the Administrator approves the change. Later, the CMS System saves the new price to the database, sends this update to all the Kiosks for immediate correctness, and displays a confirmation message to the Administrator, completing the process.

## 8.5 Ting Jac Sheinn TP080719

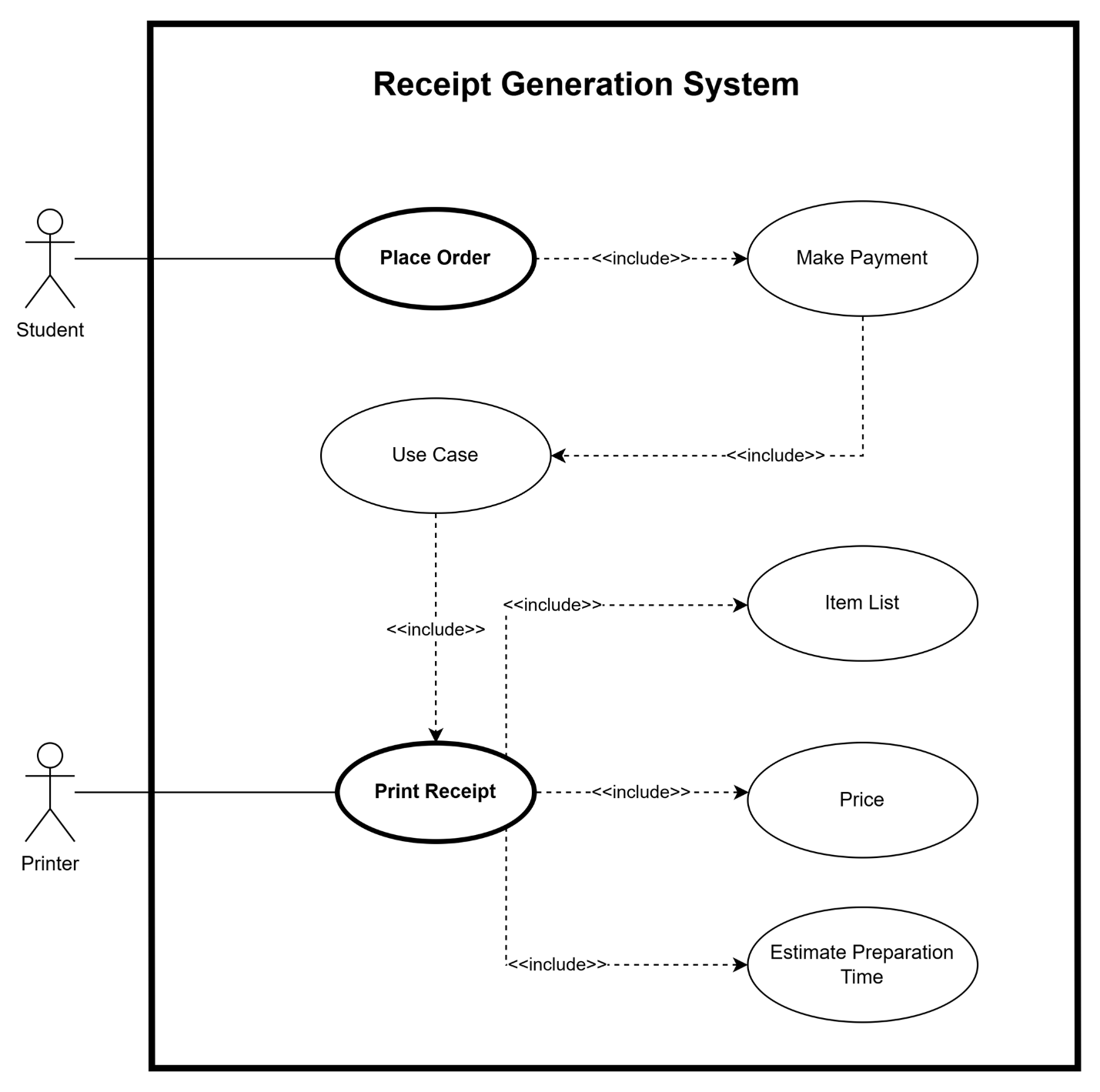


Figure 26 Use Case Diagram for Recceipt Generation System

This use case diagram illustrates the interaction with the receipt generation process in the APU Food Court Kiosk System. The student first places an order and pays for the order. Once the order has been paid for and the payment confirmed, this process of generating and printing the receipt becomes a very basic process with some necessary steps. The printed receipt shows the item list, total price to be paid for the order, and the estimated time it will take to prepare the order. This is as part of the Print Receipt use case. Once the Print Receipt use case is completed, the system sends a command to the printer to print the final receipt.

## 8.6 Huang Xiao Shun TP077723

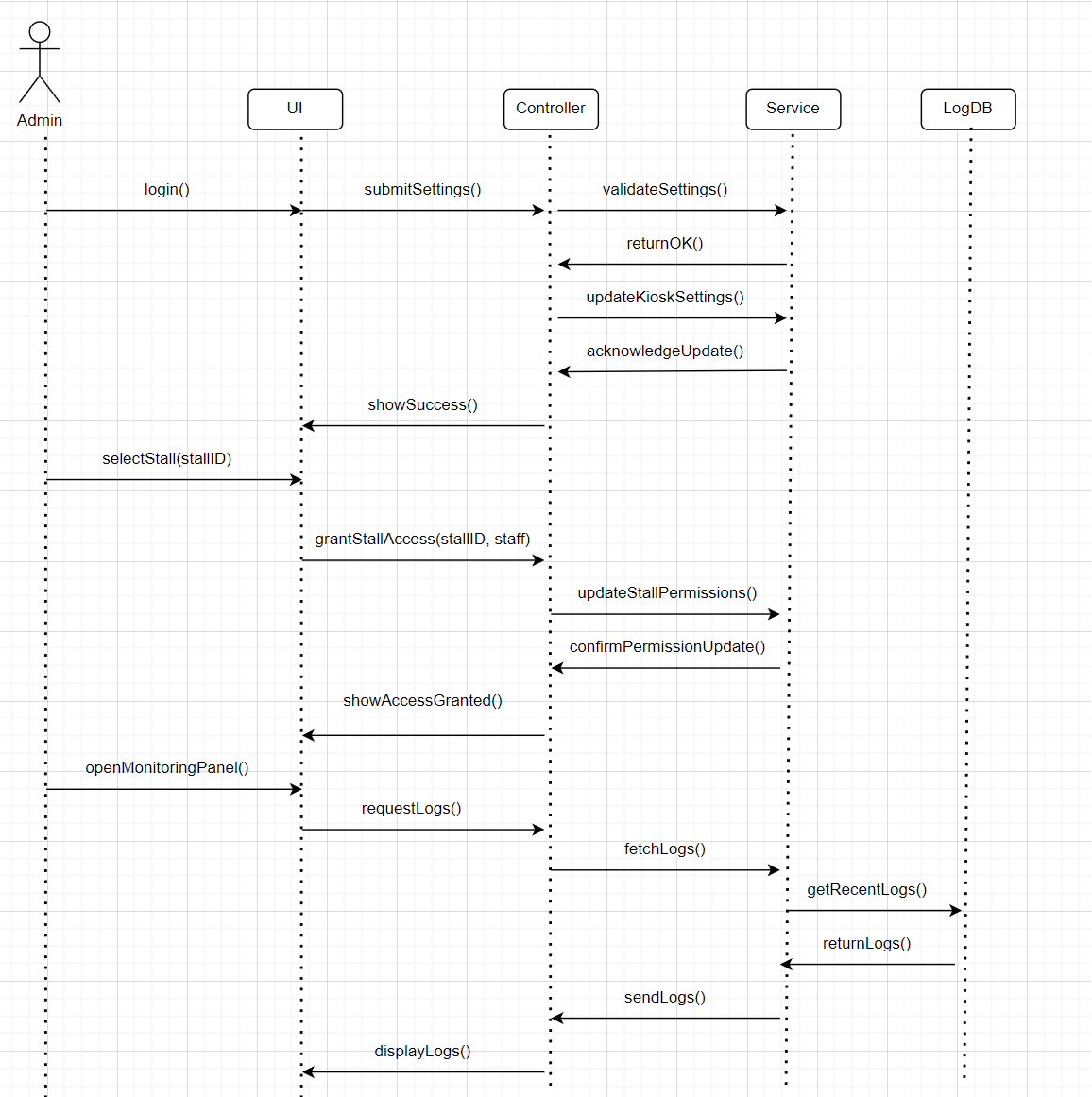


Figure 27 Admin Functional Sequence Diagram

This sequence diagram shows how the APU administrator interacts with the system to manage kiosk settings, assign stop access, and monitor system usage or error logs

# 9. References

Atlassian. (n.d.). *Acceptance criteria: Definition, examples, and best practices*. Atlassian. <https://www.atlassian.com/work-management/project-management/acceptance-criteria>

Amazon Web Services (AWS). (n.d.). *What is Scrum?* <https://aws.amazon.com/what-is/scrum/>

Hartshorne, D. (2024, February 29). *12 Agile principles: What they are and how to use them*. Monday.com. <https://monday.com/blog/rnd/agile-principles/>

AltexSoft Editorial Team. (2021, January 18). Extreme programming: values, principles, and practices. *AltexSoft*. <https://www.altexsoft.com/blog/extreme-programming-values-principles-and-practices/>

GeeksforGeeks. (2024, May 22). *What is RUP(Rational Unified Process) and its Phases?* GeeksforGeeks. <https://www.geeksforgeeks.org/rup-and-its-phases/>

Chien, C. (2020, February 4). *What is Rapid Application Development (RAD)?* Codebots. <https://codebots.com/app-development/what-is-rapid-application-development-rad>

*What is Spiral Model? 11 April 2025*

<https://www.geeksforgeeks.org/software-engineering-spiral-model/>

*Spiral Model: When to Use? Advantages and Disadvantages Lucas Benett 13 Aug 2024*

<https://www.guru99.com/what-is-spiral-model-when-to-use-advantages-disadvantages.html>?

GeeksforGeeks. (2024, September 22). *Rapid Application Development Model (RAD) Software*

*engineering*. GeeksforGeeks.

<https://www.geeksforgeeks.org/software-engineering-rapid-application-development-model-rad/>

GeeksforGeeks. (2024, July 1). *Lean Software Development (LSD)*. <https://www.geeksforgeeks.org/lean-software-development-lsd/>

Visual Paradigm. (n.d.). *What is use case specification?* Visual Paradigm. <https://www.visual-paradigm.com/guide/use-case/what-is-use-case-specification/>

GeeksforGeeks. (2024, July 12). Difference between static and dynamic testing. GeeksforGeeks. <https://www.geeksforgeeks.org/difference-between-static-and-dynamic-testing/>

GeeksforGeeks. (2025, March 17). *Test plan in software testing*. <https://www.geeksforgeeks.org/test-plan-software-testing/>

# 10. Appendix - Workload Matrix

A white sheet with numbers and text

AI-generated content may be incorrect.