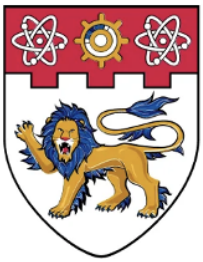


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

**School Of Computer Science & Engineering**

**SC2006 Software Engineering**

**Tutorial Group SDAB**

**Group Name**

SWEg

**Team Members**

| **Name** | **Matriculation Number** |
|---|---|
| Neo Zhi Xuan *(Group Leader)* | U2222293E |
| Kiew Ten Wei | U2221627A |
| Kauthar Ahmed Basharahil | U2240803G |
| Law Wei Lu | U2223511L |
| Mitra Ren Sachithananthan | U2020190D |
| Ng Zhuo Quan | U2222775J |

Contents

---

# 1. Product Description

## 1.1 Purpose

The aim of this project is to explore the potential of using an inventory management mobile application to tackle the issue of food waste in restaurants resulting from inefficient inventory management. In 2022, Singapore accumulated a staggering 813,000 tonnes of food waste, constituting approximately 12% of its overall waste generation (National Environment Agency of Singapore, 2023), 40% of which originates from the Commercial and Industrial (C&I) sector (Ministry of Sustainability and the Environment, 2023). Furthermore, 10% of all carbon emissions come from food waste. Despite the substantial amount of food waste, which has led to significant economic losses and carbon emissions, there has been limited effort to address these inefficiencies.

Team SWEg aims to provide our customers (local fast food chain / restaurant managers, food store owners, etc.) with PanTree, a mobile application that gives them an end-to-end solution to conveniently access and update their live store inventory. This includes having analytic-driven insights such as predictions on the amount of ingredients they require, when to restock their inventory and what menu items to sell based on various varying conditions. Through seamlessly integrating Pantree into their existing workflow, our customers can hope to see a sustainable, digital transformation across their value chain, especially regarding food wastage.

## 1.2 Scope

PanTree is a functional android mobile app that has the following features

- Allows the user to view the items in their current inventory, including their details such as name, quantity and expiration date
- Allows the user to update the inventory when suppliers deliver a new batch of ingredients
- Provides the user with predictions as to how much ingredients are required for the day
- Provide the user with alerts when there is excess or insufficient inventory stock
- Provide insights via a dashboard regarding the historical usage of each ingredient
- Suggest which menu items should be sold, pushed and promoted based on the weather forecast

Our current scope of customers focuses on F&B outlets such as restaurants, food courts, hawker centers, cafes, etc. However, in the future, the scope can be extended to supermarkets, canteens and more. The scope of the analytics is currently limited to simple logic, and suggestions are based on domain knowledge. Future expansion ideas include using ML-driven predictions and insights.

## 1.3 Users and stakeholders

Our users include staff members involved in the procurement of inventory items, the accessing and updating of the F&B outlet's inventory database and those with decision-making abilities regarding restaurant sales. Users must have an internet connection in order to have access to the suggestions feature as it is derived from a weather API that reflects the weather in real time. Other stakeholders include the customers and suppliers of the restaurant using PanTree.

## 1.4 Assumptions and constraints

Some assumptions are that the restaurant grants us access to their inventory data and permissions to connect to their system. Other assumptions are that the 24-hour Weather is consistent throughout the day, and assume that it is accurate.

## 1.5 Initial UI Mockups


Fig 1. Main User Interface - View Inventory


Fig 2. Predictions of Inventory Levels and Sales


Fig 3. Alerts that appear when items are on low stock


Fig 4. Alerts that appear daily at evening, to indicate predictions generated and expiring items identified


Fig 5. Suggestions on menu items to generate based on ingredients

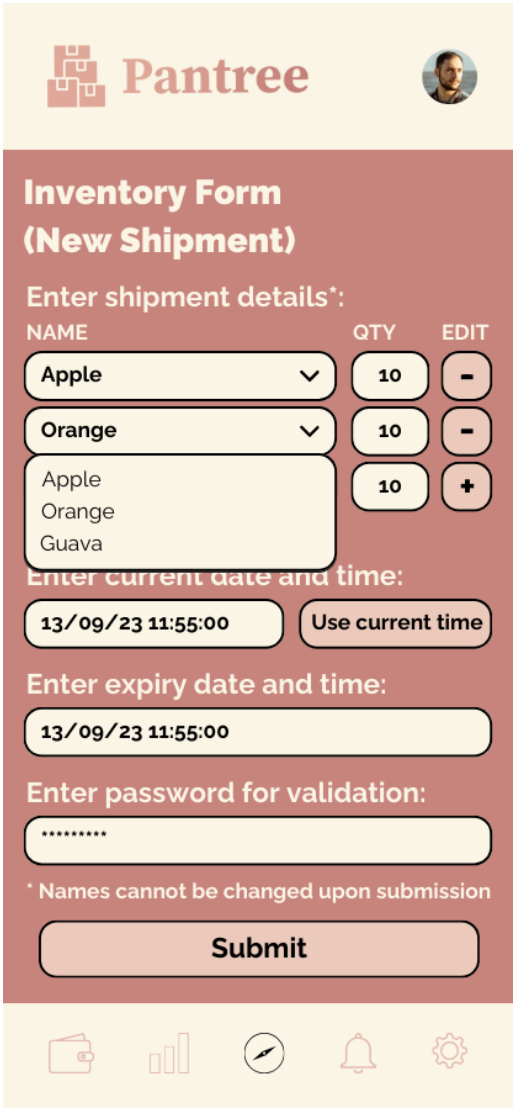



Fig 6. Order Form to fill in when a Shipment arrives

## 2. Functional Requirements

*Updating Inventory upon arrival of new supplier delivery*

**1. The system must allow Users to update the System Inventory Database.**

1.1. The user interface must have multiple buttons, where each button leads to a separate page: including the Order Form page, Suggestions page and Predictions page.

1.2. The user must be able to click on the “Fill in Inventory Form” button, and must be directed to the Order Form page to fill in the Order Form.

1.2.1. The user must be able to enter the details of all the Order Data that have been delivered. This consists of: item name, item quantity, expiry date.

1.2.1.1. The system must be able to fetch the list of valid item names from the database to populate the dropdown bar’s options.

1.2.1.2. The name of the item must be a valid option within the dropdown bar, which consists of the list of all names of items.

1.2.1.3. The quantity of item must be an integer of at least 0, and at most 999.

1.2.1.4. The expiry date must be a date starting from the current date.

1.2.1.5. Every field in the Order Form must be filled in with a value that adheres to the above requirements.

1.2.2. Users must be able to add entry elements from the list of Order Data by pressing the ‘+’ button.

1.2.3. Users must be able to remove entry elements from the list of Order Data by pressing the ‘-’ button.

1.2.4. Users must enter a password to authenticate themselves as approved administrators allowed to submit Order Forms.

1.2.4.1 The password must be text of at least one character and less than 50 characters.

1.2.5. The system must display an error message on the user's screen if an invalid value is entered, or if any of the abovementioned fields are not filled in.

1.2.6. When the users press the 'Submit' button for the Order Form to be submitted, the system must be able to send a request to the backend to update the Order Data of the new Order Form into the System Inventory Database.

*Viewing Current Store Inventory*

**2. Users shall be able navigate to the "View Inventory" page to view the items in the System Inventory Database**

2.1. The system must display the Current Inventory Levels corresponding to the Order Form that the user logged in the Inventory Database.

2.1.1. The system must be able to aggregate the total quantity of each item across all Order Data that is present in the System Inventory Database, and store this data as Current Inventory Levels.

2.1.2. When the users click on the box containing the item's name, the system must display the list of Order Data submitted in previous Order Forms for that item.

2.1.2.1. Each Order Data consists of item quantity and expiry date.

*View Predictions of Inventory Levels*

**3. Users shall be able to receive predictions regarding the quantity of Menu Items to prepare for the day via alerts.**

3.1. The system must send the user an alert every evening at 1800 hours, to notify the user of the required number of food items to order for subsequent Shipments, to meet predicted inventory levels that are generated by algorithmic predictions.

3.1.1. The system must generate abovementioned daily predictions of inventory levels by using historical and current data as inputs to a predictive algorithm.

3.1.1.1 The algorithm must utilise other data available from publicly available APIs as an input to the predictive algorithm, to generate more tailored outputs to the current situation.

3.1.1.2. The system must log and track the predictions generated for audit and analysis purposes into its own collection in the System Inventory Database.

3.1.2. The alert should also contain information about the foods that are expiring soon.

3.1.2.1. The system must be able to query the System Inventory Database for items that have an expiry date within 2 days.

3.2. The user must be able to click on the “View Daily Predictions” button to enter the Predictions page.

3.2.1. When the User enters the Predictions page, the system must display a Predictions Dashboard consisting of inventory trends and predictions.

3.2.1.1. The dashboard must display sales data as a line chart.

3.2.1.2. Users must be able to adjust the time axis of the line chart, where the system must display data based on the window of time selected.

3.2.1.2.1. The time frame must be an option between “Weekly”, “Biweekly”, and “Monthly”.

3.2.1.3. The dashboard must display predicted inventory levels, alongside actual inventory levels, to allow users to compare both statistics and ensure sufficient inventory levels are present.

*Receive alerts when an item stock is low*

**4. Users shall be able to receive alerts about the food items that are over-prepared from the inventory, and relevant suppliers for that item.**

4.1. The system must be able to retrieve information from the database to identify when an item has a stock quantity below 5, and send automated alerts to the user.

4.1.1. Alerts shall be delivered to users through multiple channels, including email, SMS, and within the system's user interface.

4.1.2. When an item is identified as having low stock, the system must display information about the names and quantities of affected items, and information about the suppliers for contact purposes.

4.1.2.1. The system must be able to query the System Inventory Database for a list of suppliers, and filter out the relevant suppliers that can supply the over-prepared items.

4.1.2.2. The information should include the company’s name, and the Person Of Contact’s (POC) phone number.

*Receive suggestions on menu items based on the inventory levels, and post a menu item into the Marketplace*

**5. Users shall be able to receive Suggestions for a list of Menu Items to post to the Marketplace.**

5.1. When the user clicks into the “Suggestions” page, the system must display a suggested list of menu items based on the predictions.

5.1.1. The list of suggested menu items should include item name and a list of ingredient names and quantities required to prepare the item.

5.1.1.1. This list must be sorted, where suggested menu items with the most number of food items that are about to expire will be at the top with highest priority.

5.1.1.2. The system must be able to query publicly available APIs to retrieve information about the weather, and use it in our algorithm to suggest different menu items based on the weather.

5.1.1.3. The quantities mentioned in the list of ingredient names and quantities should be less than the quantity of that ingredient available in our System Inventory Database.

5.2. Users should have the option to set prices and descriptions for the newly created menu items.

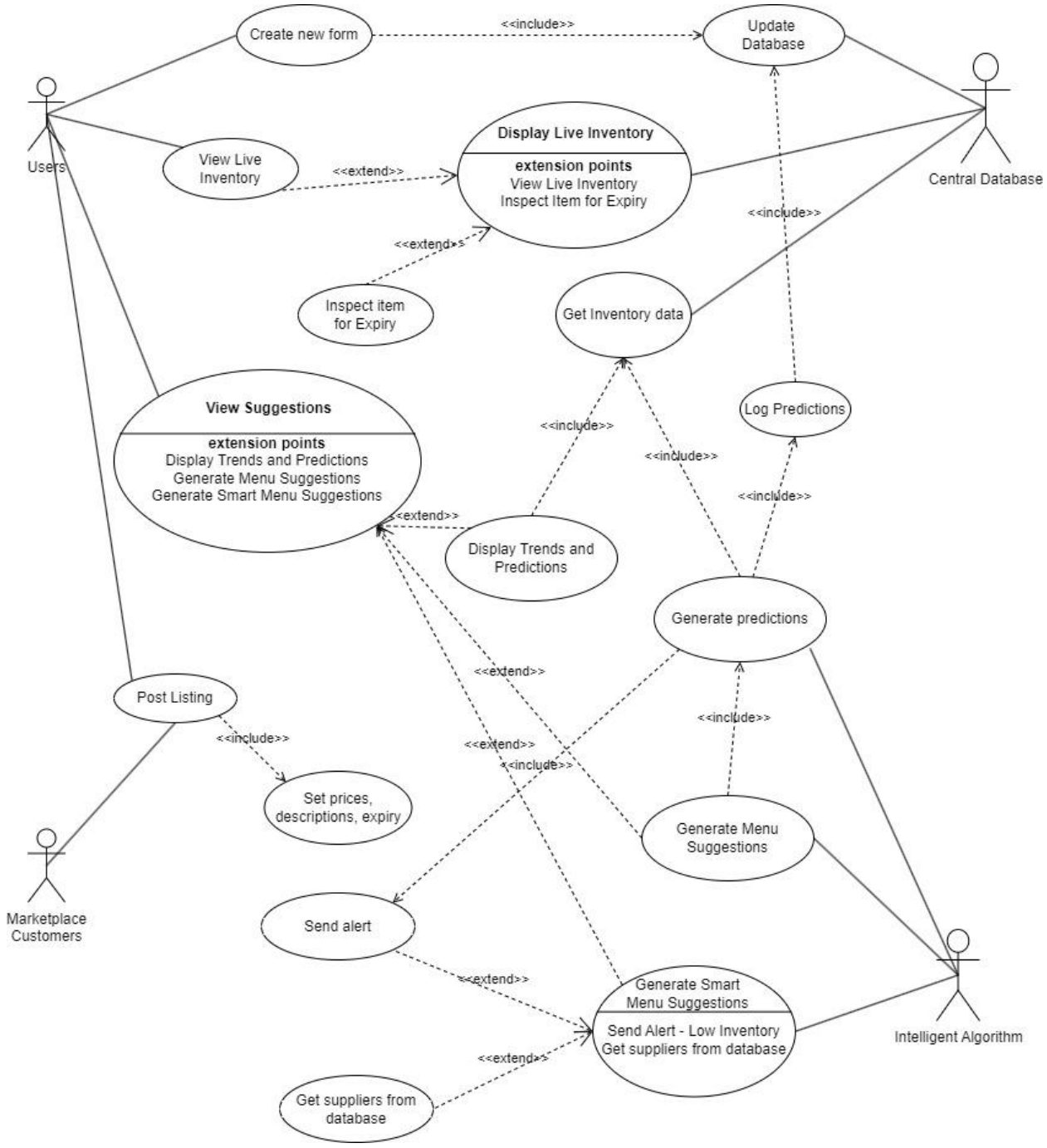
5.2.1. Price must be a positive integer between 0 and 1000.

5.2.2. Description must be text of at least one character and less than 512 characters.

5.2.3. The system must display an error when the inputs do not adhere to the above requirements.

5.3. When the user clicks the “Submit” button, the system must be able to send the data to the System Inventory Database, where the new menu item will be stored in the Marketplace collection for other end-users to peruse.

## 2.1 Use Case Diagrams




## 2.2 Use Case Descriptions

| Use Case ID: | U0001 | | |
|---|---|---|---|
| Use Case Name: | Create new Form | | |
| Created By: | Kiew Ten Wei | Last Updated By: | Zhuo Quan |
| Date Created: | 12 Sep 2023 | Date Last Updated: | 19 Nov 2023 |

| Actor: | Users (initiating) |
|---|---|
| Description: | Allow users to create a new Form |
| Preconditions: | - User is logged into the system. |
| Postconditions: | - A form is displayed for the user to fill. |
| Priority: | - |
| Frequency of Use: | - |
| Flow of Events: | 1. The user navigates to the “Fill in Inventory Form” section in the system’s user interface.<br>2. The system provides a blank inventory form with columns for food item names and quantities.<br>3. The user enters the quantities of each food item as per the shipment.<br>4. The user enters the expiry date of the food items.<br>5. The user enters the administrative password.<br>6. The user presses the “Submit Form” button<br>7. The system updates the database of the new information |
| Alternative Flows: | AF1. S7 - If the user enters leaves the food item ,quantity, expiry date or password field empty,<br>1. The system prompts the user to enter the required fields<br>2. Go back to step 2. |

| | |
|---|---|
| | AF2. S7 - If the user enters enters an invalid quantity or password,<br>1. The system prompts the user to enter a valid input for the required fields.<br>2. Go back to step 2. |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| | | | |
|---|---|---|---|
| Use Case ID: | U0002 | | |
| Use Case Name: | Update Database | | |
| Created By: | Zhuo Quan | Last Updated By: | Zhuo Quan |
| Date Created: | 27 Sep 2023 | Date Last Updated: | 19 Nov 2023 |

| | |
|---|---|
| Actor: | Users (initiating), Central Database (supporting) |
| Description: | The system updates the database on the details of the new Form created. |
| Preconditions: | - The user is logged into the system. |
| Postconditions: | - The database is successfully updated |
| Priority: | - |
| Frequency of Use: | - |
| Flow of Events: | 1. The system sends the information that is updated to the Central Database<br>2. The Central Database updates the quantity of the food item based on the information received<br>3. The Central Database sends a notification to |

| | |
|---|---|
| | the system to indicate completion.<br>4. The system sends a notification to the user to indicate successful updating of the database. |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| | | | |
|---|---|---|---|
| Use Case ID: | U0003 | | |
| Use Case Name: | View Live Inventory | | |
| Created By: | Kiew Ten Wei | Last Updated By: | Zhuo Quan |
| Date Created: | 12 Sep 2023 | Date Last Updated: | 19 Nov 2023 |

| | |
|---|---|
| Actor: | Users (initiating), Central Database (supporting) |
| Description: | Allows users to access and view the live inventory data stored in the central database. Users can see real-time information about the food items, their quantities, and any relevant details pertaining to the inventory. |
| Preconditions: | - The user is logged into the system. |
| Postconditions: | - The user has successfully viewed the live inventory data from the central database. |
| Priority: | - |
| Frequency of Use: | - |

| | |
|---|---|
| Flow of Events: | 1. The user navigates to the “View Live Inventory” section in the system’s user interface.<br>2. The system retrieves live inventory data from the central database.<br>3. The system moves to the Use Case “Display Live Inventory” |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| | | | |
|---|---|---|---|
| Use Case ID: | U0004 | | |
| Use Case Name: | Display Live Inventory | | |
| Created By: | Kiew Ten Wei | Last Updated By: | Zhuo Quan |
| Date Created: | 12 Sep 2023 | Date Last Updated: | 19 Nov 2023 |

| | |
|---|---|
| Actor: | Users (initiating), Central Database (supporting) |
| Description: | The system retrieves data from the database and displays it in a organised table for the user to view in a convenient manner |
| Preconditions: | - The user is logged into the system. |
| Postconditions: | - The user has successfully viewed the live inventory data from the central database. |
| Priority: | - |
| Frequency of Use: | - |

| | |
|---|---|
| Flow of Events: | 1. The system presents the live inventory data to the user in a user-friendly format,such as a list or table. The data includes:<br>a. Names of food items in stock<br>b. Quantities of each food item<br>c. Any additional relevant information(e.g., expiration dates, flow rates)<br>2. The user can scroll through the inventory list to see all the available food items and their respective quantities.<br>3. The user can click on individual items to access more detailed information about a specific item, such as its history, supplier information, and expiration date. |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

.

| | | | |
|---|---|---|---|
| Use Case ID: | U0005 | | |
| Use Case Name: | Get Inventory Data | | |
| Created By: | Zhuo Quan | Last Updated By: | Zhuo Quan |
| Date Created: | 10 Oct 2023 | Date Last Updated: | 19 Nov 2023 |

| | |
|---|---|
| Actor: | Users (initiating), Central Database (supporting) |
| Description: | The system requests specific data from the database and the database returns the required information |
| Preconditions: | - The user is logged into the system. |

| | |
|---|---|
| Postconditions: | - The required information is successfully acquired from the database. |
| Priority: | - |
| Frequency of Use: | - |
| Flow of Events: | 1. The system queries the database for the specific data it requires.<br>2. The database does an internal search to find the data required<br>3. The database returns the data requested by the system. |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| | | | |
|---|---|---|---|
| Use Case ID: | U0006 | | |
| Use Case Name: | View Suggestions | | |
| Created By: | Zhuo Quan | Last Updated By: | Zhuo Quan |
| Date Created: | 10 Oct 2023 | Date Last Updated: | 19 Nov 2023 |

| | |
|---|---|
| Actor: | Users (initiating) |
| Description: | Allows the user to choose between the different suggestions available to view in greater detail. |
| Preconditions: | - The user is logged into the system. |
| Postconditions: | - The user has chosen a specific suggestion to view |
| Priority: | - |
| Frequency of Use: | - |

| | |
|---|---|
| Flow of Events: | 1. The system prompts the user to select the desired action: Display Trends and Predictions or View Smart Menu suggestions.<br>2. If the user selects the activity Display Trends and Predictions, then he uses the included test case Display Trends and Suggestions to view trends and predictions of sales.<br>3. If the user selects the activity View Smart Menu Suggestions, then he uses the included test case Generate Smart Menu Suggestions to view the new menu items generated by the Intelligent Algorithm based on expiring food items and weather data. |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| | | | |
|---|---|---|---|
| Use Case ID: | U0007 | | |
| Use Case Name: | Display Trends and Predictions | | |
| Created By: | Zhuo Quan | Last Updated By: | Zhuo Quan |
| Date Created: | 10 Oct 2023 | Date Last Updated: | 19 Nov 2023 |

| | |
|---|---|
| Actor: | Users (initiating) |
| Description: | Allows the user view the trends and predictions of current and historical sales data |
| Preconditions: | - The user is logged into the system. |
| Postconditions: | - The user has viewed the trends and predictions |

| | |
|---|---|
| Priority: | - |
| Frequency of Use: | - |
| Flow of Events: | 1. The system uses the included use case Get Inventory Data to request past sales data and prediction data from the database.<br>2. Upon receiving the data, the system displays it in a line chart for the user to view.<br>3. The user can adjust the time axis of the graph where the system will display the time frame specified.<br>4. The user can choose a time frame option between “Daily”, “Weekly” and “Monthly”, and the system will generate the graph as specified.<br>5. The system also displays the predicted usage for each menu item in the inventory, and the current amount present. |
| Alternative Flows: | - |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| | | | |
|---|---|---|---|
| Use Case ID: | U0008 | | |
| Use Case Name: | Post Listing | | |
| Created By: | Kiew Ten Wei | Last Updated By: | Zhuo Quan |
| Date Created: | 27 Sep 2023 | Date Last Updated: | 19 Nov 2023 |

| | |
|---|---|
| Actor: | Primary Actor: Users<br>Secondary Actor: Marketplace Customers |

| | |
|---|---|
| Description: | Users can post listing of food items on the app's marketplace. Users have the option to specify various details for the food items, including setting prices and descriptions |
| Preconditions: | - The user has navigated to the "Post Listings"<br>- The user has access to their inventory of food items |
| Postconditions: | - The new listing is added to the app's marketplace and is visible to other users<br>- The food item is available to other users on the app's marketplace |
| Priority: | High |
| Frequency of Use: | Daily |
| Flow of Events: | 1. User selects the option to "Post Listing" within the app<br>2. The app presents the user with a form for creating a new listing<br>3. The user fills in the details for the food item, which include:<br>a. Price<br>b. Description<br>4. The user submits listing to the app<br>5. The app validates the listing to ensure all required information is provided and the data is in the correct format<br>6. If the listing is valid, the app adds the new food item to the marketplace<br>7. The listing is now visible to other users on the app" marketplace |
| Alternative Flows: | AF1: If the user does not provide all required information in the listing form, the app will prompt the user to complete all mandatory fields before submission |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | U00009 | | |
|---|---|---|---|
| Use Case Name: | Generate predictions | | |
| Created By: | Kiew Ten Wei | Last Updated By: | Zhuo Quan |
| Date Created: | 27 Sep 2023 | Date Last Updated: | 19 Nov 2023 |

| Actor: | Primary Actor: User<br>Secondary Actor: Intelligent Algorithm |
|---|---|
| Description: | This use case describes how the system generates predictions for menu item preparation and provides these predictions to users either through alerts or by navigating to the “Suggestions” page. The predictions are based on historical and current data, incorporating factors such as sales trends, customer demand, and seasonal influences |
| Preconditions: | - Historical and current data are available for analysis<br>- The system has access to the Inventory Database |
| Postconditions: | - Users receive alerts with required quantities for food item orders<br>- Users can access a list of menu item suggestions<br>- The system logs and tracks predictions for future reference and analysis<br>- The dashboard displays historical data, predicted inventory levels, and actual inventory levels for comparison |
| Priority: | High |
| Frequency of Use: | Daily |

| | |
|---|---|
| Flow of Events: | 1. The system sends users an alert every evening at 1800 hrs to notify them of the required number of food items to order for subsequent shipments to meet predicted inventory levels<br>2. The system generates daily predictions of inventory levels by utilizing an intelligent algorithm<br>3. The algorithm takes as input historical and current data, including sales trends, customer demand, and seasonal factors like special events, holidays, and promotions.<br>4. The system logs and tracks the predictions generated for audit and analysis purposes into its own collection in the Inventory Database.<br>5. Users can navigate to the “Suggestions” page<br>6. The system displays a suggested list of menu items based on the predictions<br>7. The list is includes item names and quantities used to prepare the items<br>8. The system displays a dashboard of inventory trends and predictions<br>9. The dashboard includes a line chart depicting historical data<br>10. The dashboard also displays predicted inventory levels alongside actual inventory levels to allow users to compare and ensure sufficient inventory<br>11. Users can adjust the time axis of the dashboard, with options for “Daily”, “Weekly,” and “Monthly” time frames. |
| Alternative Flows: | - If no historical or current data is available, the system may not generate predictions<br>- If the system fails to send alerts, users can access the “Suggestions” page to view predictions.<br>- Users can select different time frames on the dashboard to view data for specific periods. |

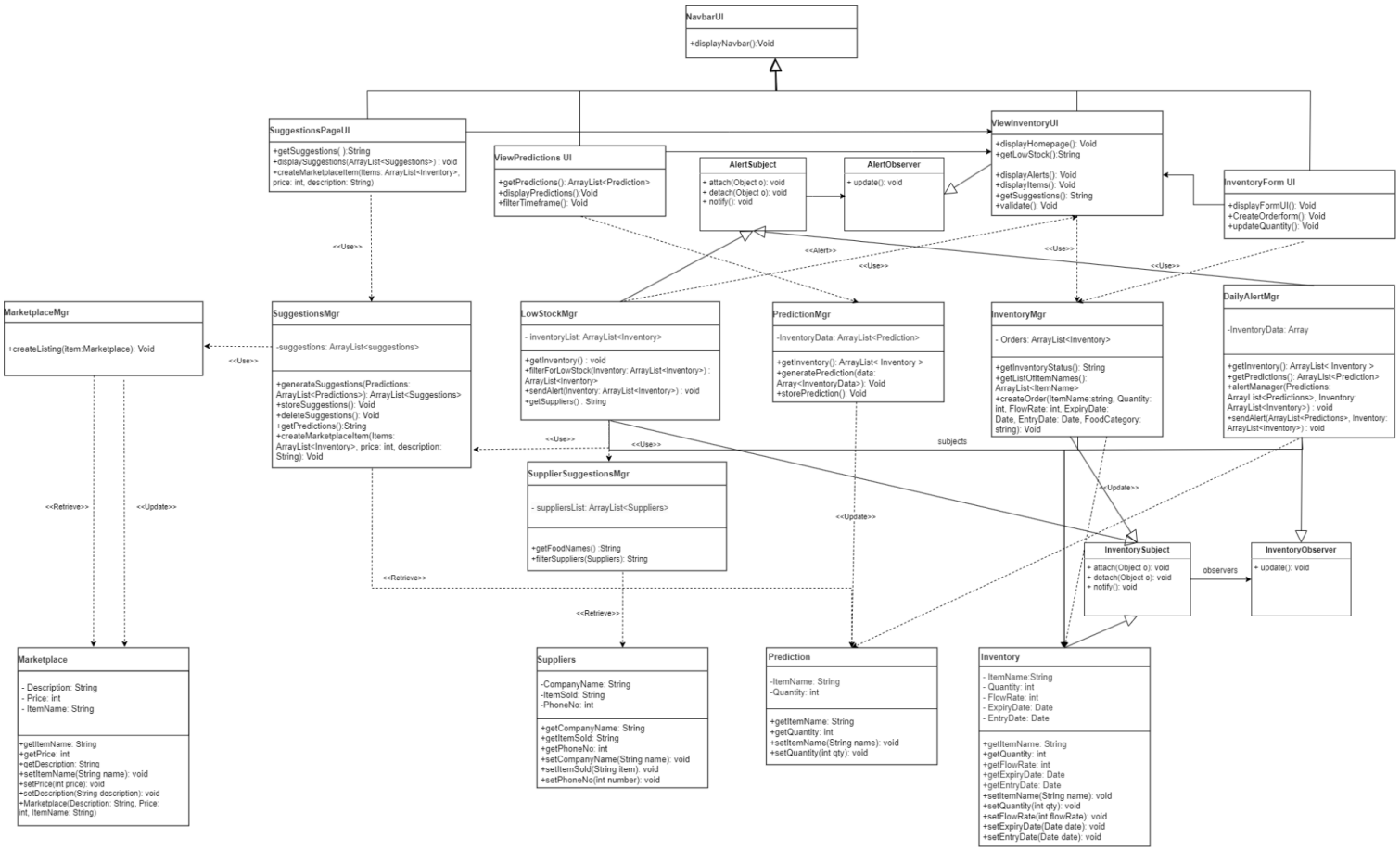
| | |
|---|---|
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

| Use Case ID: | U00010 | | |
|---|---|---|---|
| Use Case Name: | Generate Smart Menu Suggestions | | |
| Created By: | Kiew Ten Wei | Last Updated By: | Zhuo Quan |
| Date Created: | 27 Sep 2023 | Date Last Updated: | 19 Nov 2023 |

| | |
|---|---|
| Actor: | Primary Actor: Intelligent system (initiating)<br>Secondary Actor: Central database |
| Description: | This use case describes how the system analyzes over-prepared food items in the inventory, sends alerts to the user, and generates smart menu suggestions based on these items. The system leverages data from the inventory database and interacts with users and suppliers to make menu suggestions while preventing food wastage. |
| Preconditions: | - The database is updated and provides accurate information.<br>- Supplier information is available in the system |
| Postconditions: | - Users receive alerts regarding over-prepared items and low stock levels<br>- Smart menu suggestions are generated and presented to users.<br>- Inventory levels are adjusted as per user actions |
| Priority: | High |
| Frequency of Use: | Daily |

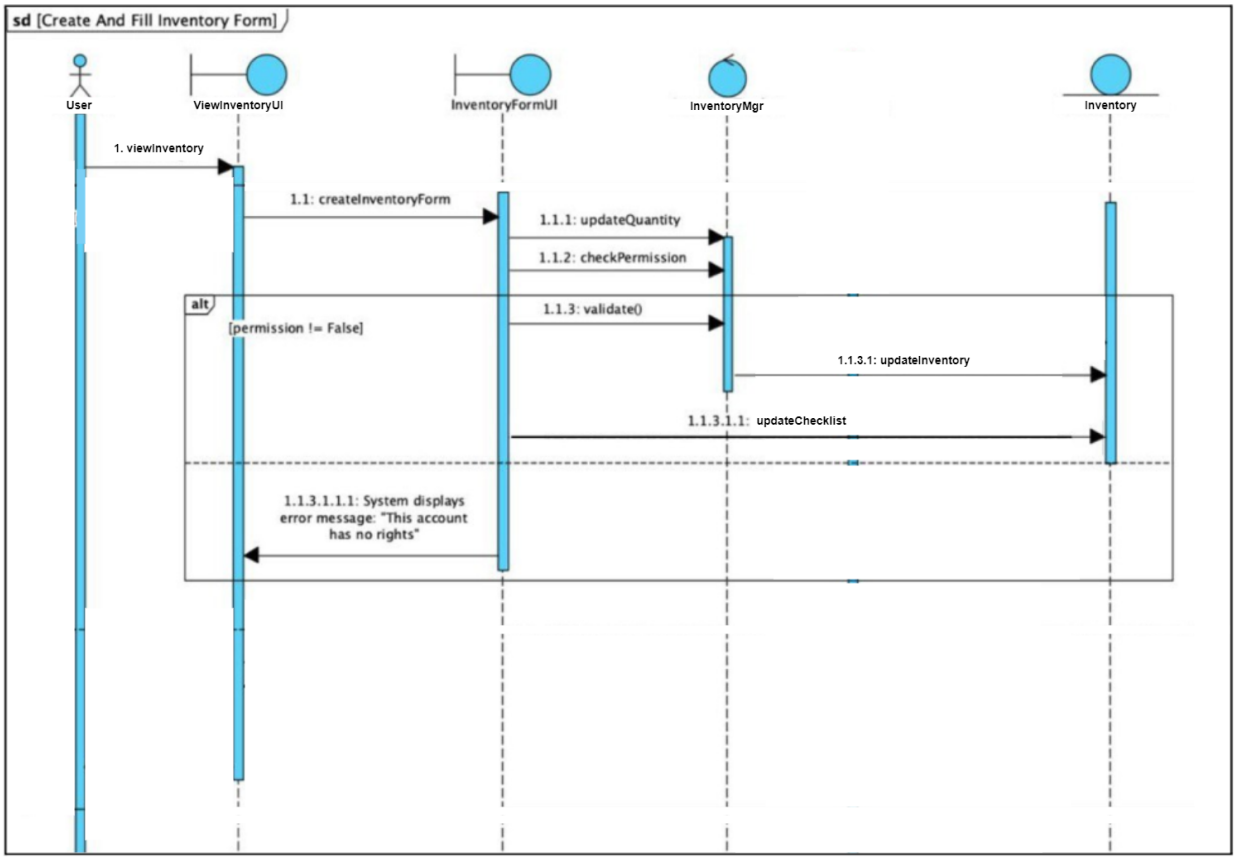
| | |
|---|---|
| Flow of Events: | 1. When the system detects an over-prepared item with low stock, it triggers an alert.<br>2. The system sends alerts to the user through a notification<br>3. The alert contains information about:<br>a. The names and quantities of affected over-prepared items.<br>b. Suggested suppliers for items that need restocking<br>4. The system queries the Inventory Database to retrieve a list of available suppliers and filters them to include only those relevant for supplying the over-prepared items.<br>5. When the user receives an alert, they can access a dedicated “Smart Menu Suggestions” section within the system’s user interface.<br>6. The system utilizes algorithms to analyze the inventory levels and sales data, taking into account over-prepared food items.<br>7. Based on the analysis, the system generates smart menu suggestions that incorporate the over-prepared food items. |
| Alternative Flows: | AF1.S1: If the system fails to identify any over-prepared items or low stock, no alerts are generated.<br><br>AF2.S1: If a user decides not the act upon an alert or the menu suggestions, the system will retain the over-prepared items in inventory |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

## 2.3 Class Diagrams

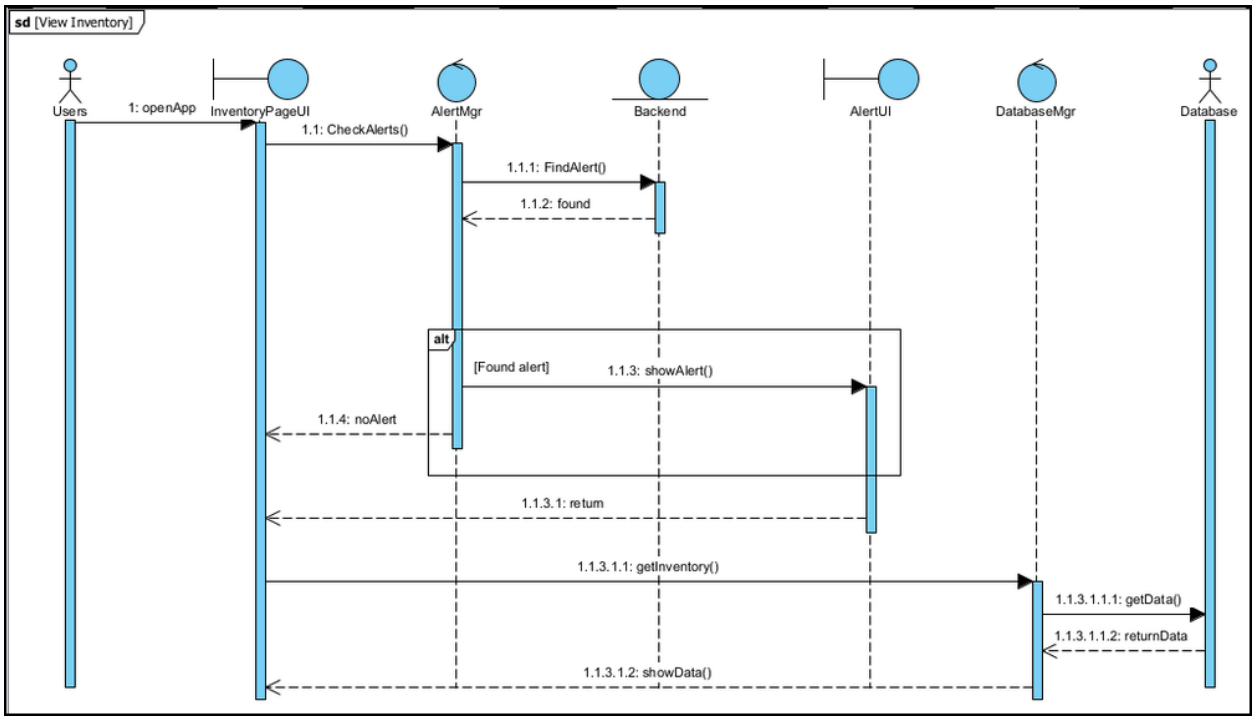


## 2.4 Sequence Diagrams

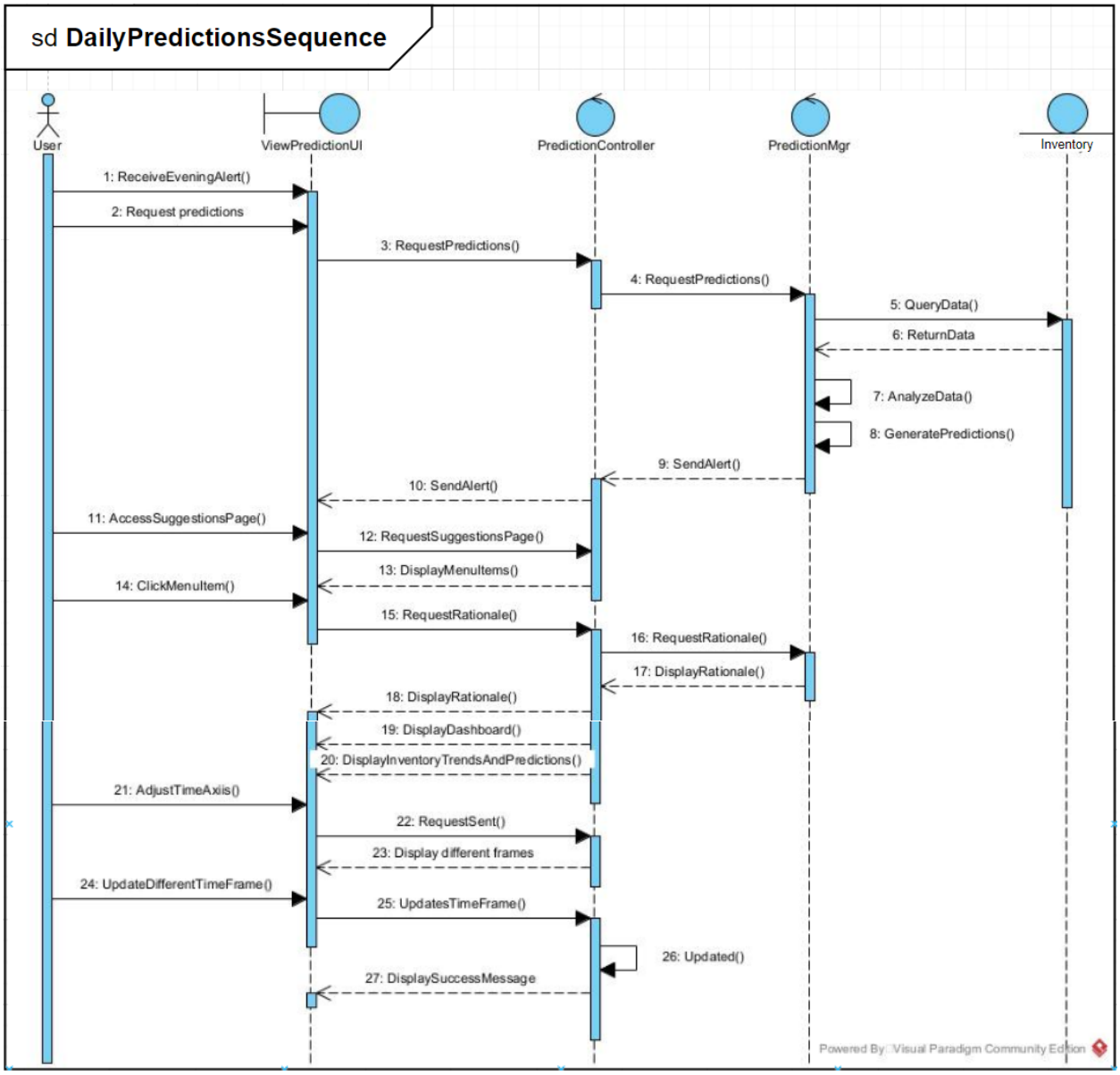
1. Create Inventory Form



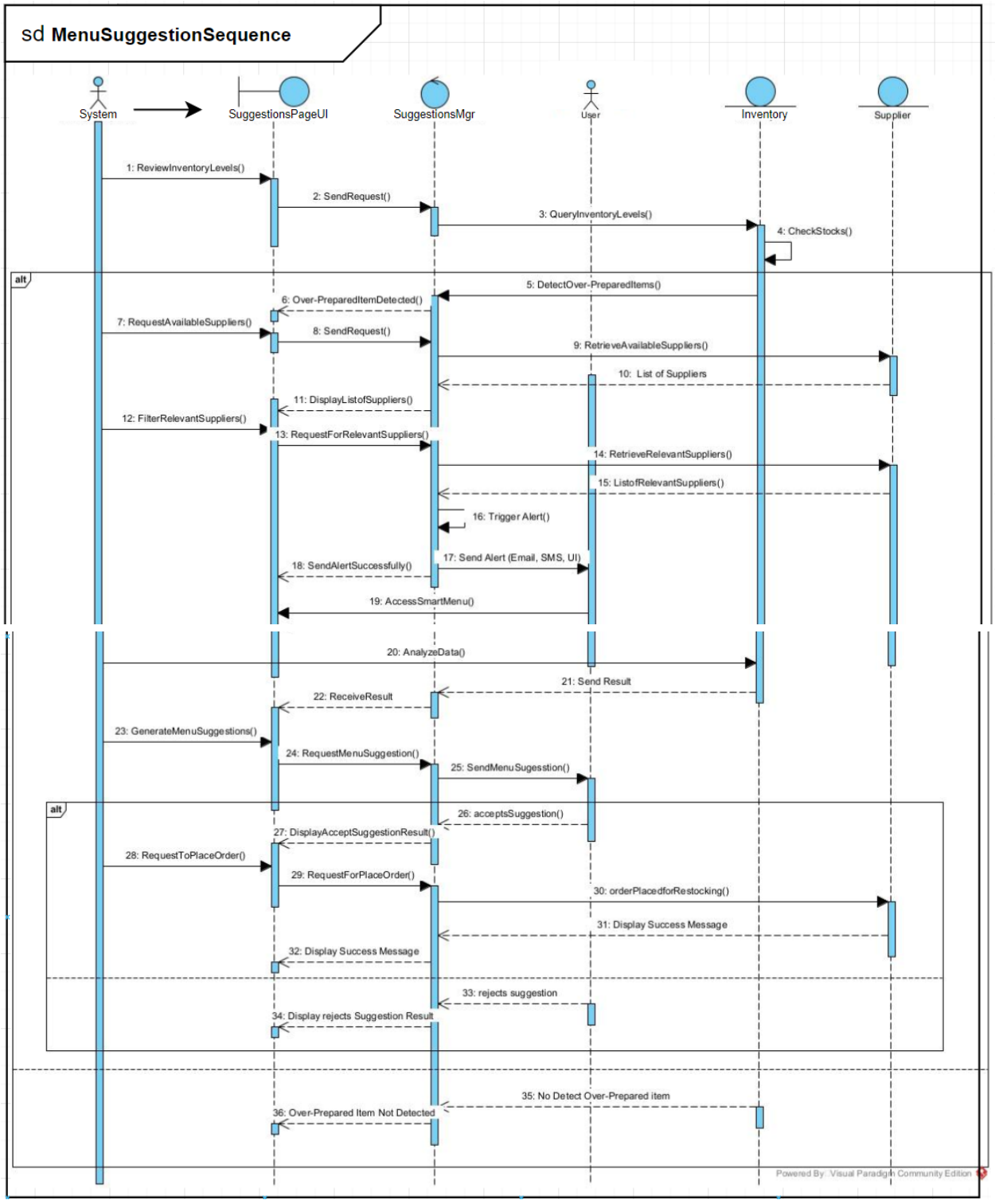
2. Display Inventory Form



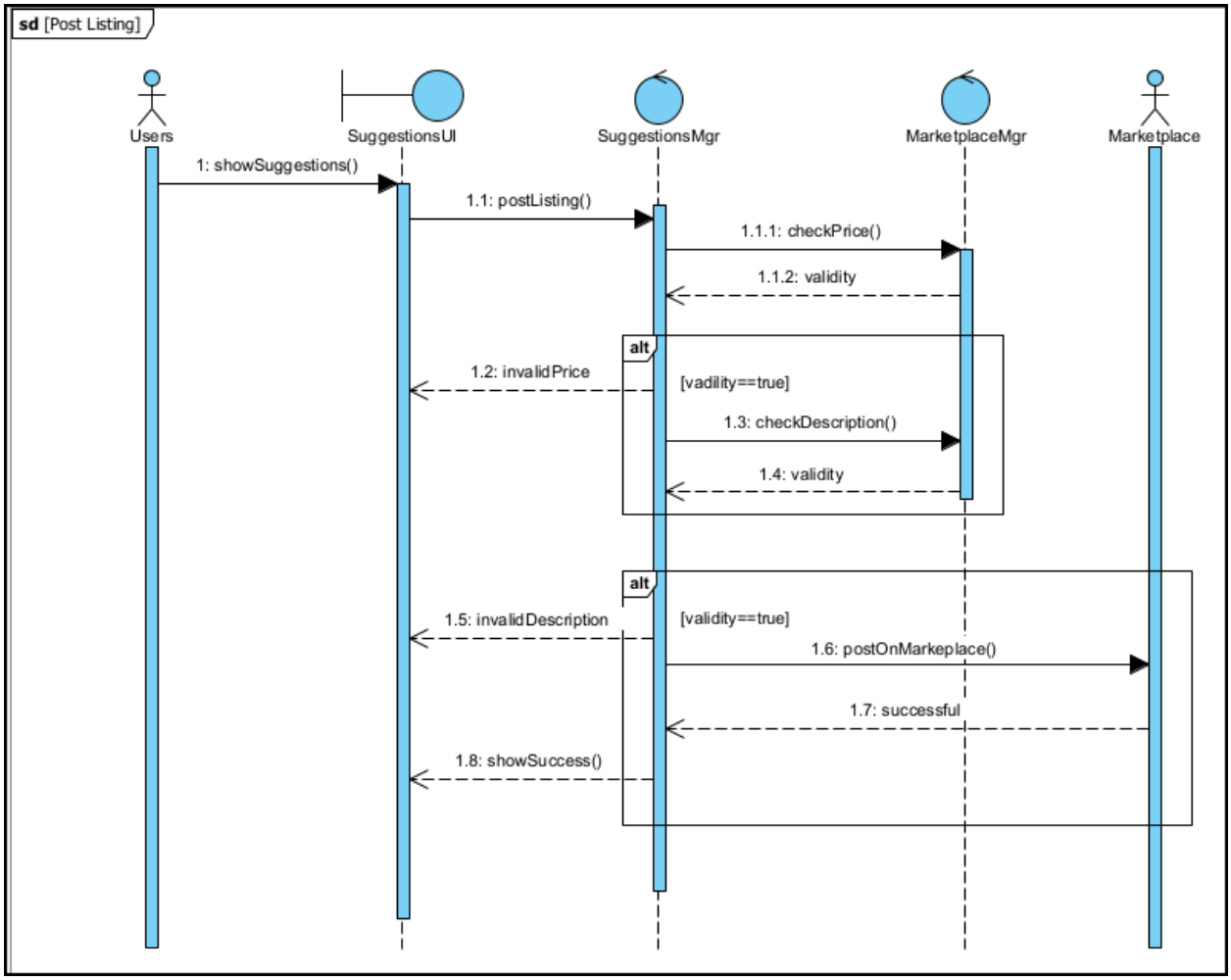
3. Daily Predictions Sequence



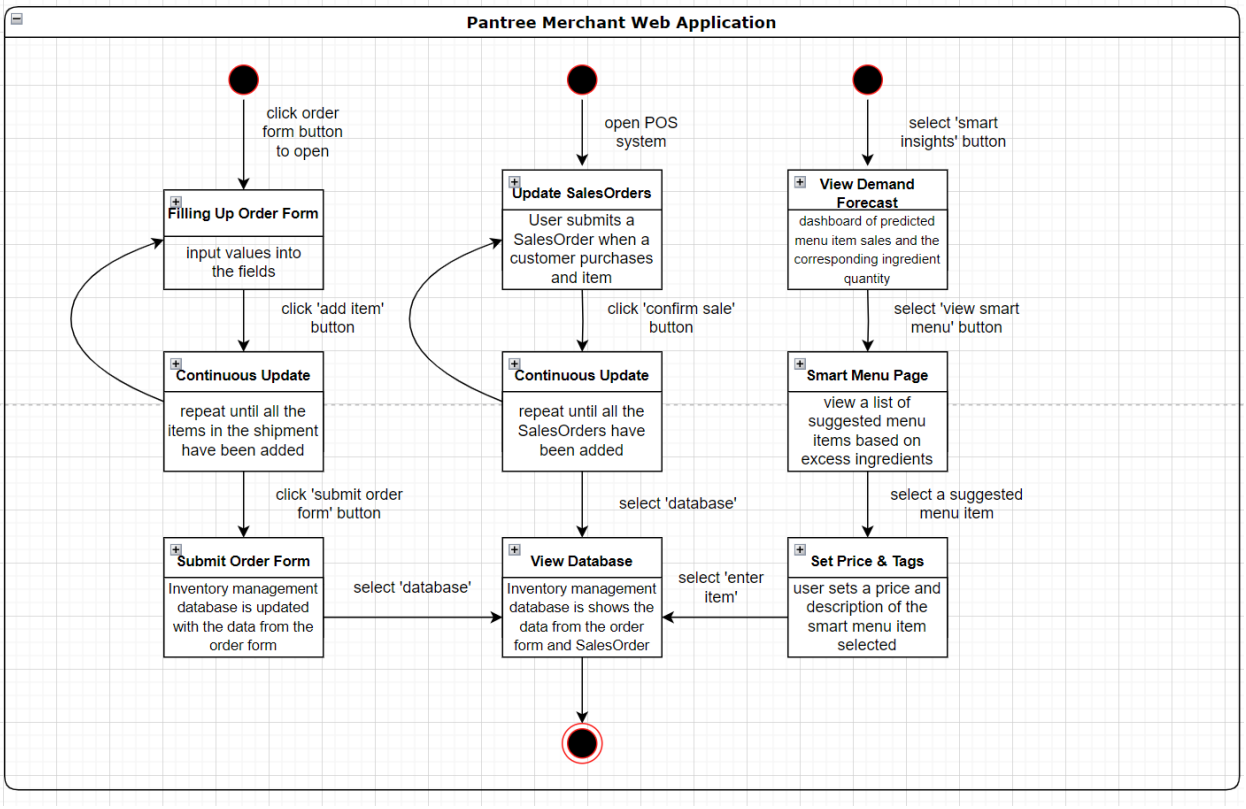
4. Menu Suggestion Sequence




5. Post Listing


sd [Post Listing]
Users
SuggestionsUI
SuggestionsMgr
MarketplaceMgr
Marketplace
1: showSuggestions()
1.1: postListing()
1.1.1: checkPrice()
1.1.2: validity
alt
1.2: invalidPrice
[vadility==true]
1.3: checkDescription()
1.4: validity
alt
1.5: invalidDescription
[validity==true]
1.6: postOnMarkeplace()
1.7: successful
1.8: showSuccess()

## 2.5 Dialog Map



## 3. Non-Functional Requirements

---

**1. Usability**

1.1. A feedback and rating system (out of 5 stars) shall be included at the end of each User interaction cycle.

1.2. Feedbacks involving the UI/UX or software issues must be addressed / fixed within 3 business days.

1.3. Error rate of Users submitting or validating OrderForm and ManualForm should not exceed 5%.

1.4. Predictions should be generated with a reasonably high degree of accuracy and confidence.

**2. Performance**

2.1. Pantree must be able to handle 100 Users on the app concurrently.

2.2. Updates of Sale and Form data should take <=10 seconds.

**3. Scalability**

3.1. Performance Scalability

3.1.1. The system should be able to handle a growing number of inventory items and users without a significant degradation in performance.

3.1.2. Response times for critical operations (e.g., adding items, generating reports) should remain within acceptable limits as the system scales.

3.2. Data Scalability

3.2.1. The system should support a large and increasing volume of data, including inventory records, transaction history, and machine learning training data.

**4. Integration**

4.1. Integration with point-of-sale (POS) systems.

4.2. Export and import capabilities for data transfer with external systems.

4.3. API support for third-party integrations.

# 4. Interface Requirements

## 4.1 User

PanTree is designed for users to have a more streamlined way to manage their inventory of mainly food ingredients. It allows restaurants or food business owners to have an easier way to track their supply and demand.

## 4.2 Hardware

PanTree will need to work on hardware devices with internet connection to be able to access weather data.

## 4.3 Software

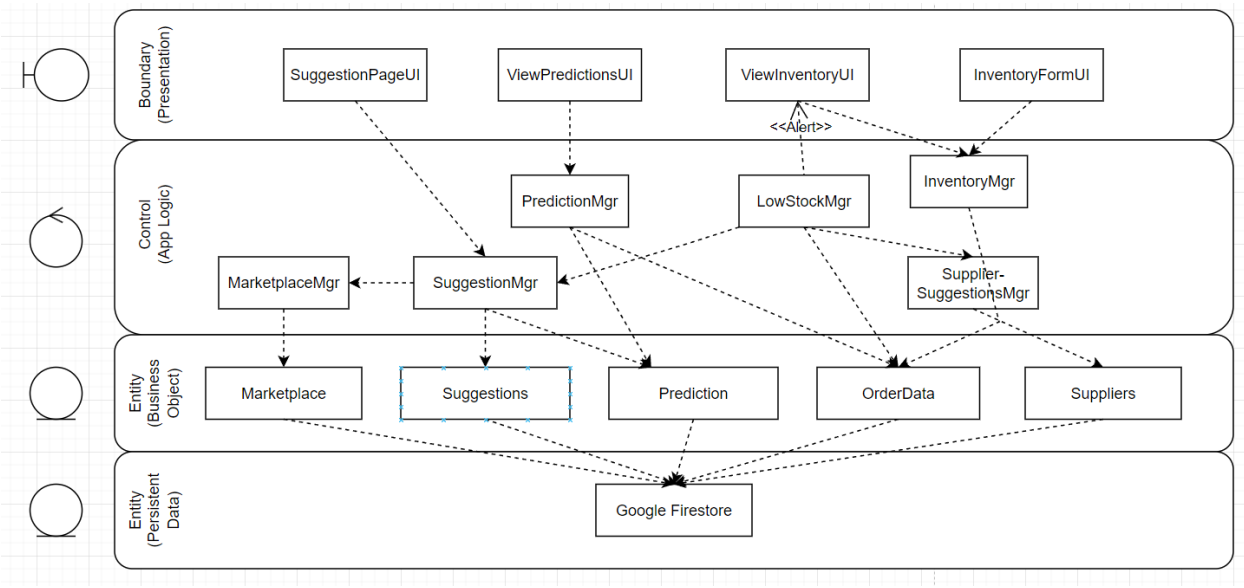
PanTree is designed to work on Android Devices, with further development to IOS, Windows and Mac devices in the future.

## 4.4 Communication

PanTree is accessed on the internet, with a Firebase database stored online. All features will be accessible through the application.

# 5. Architecture Design

## 5.1 System Architecture Diagram



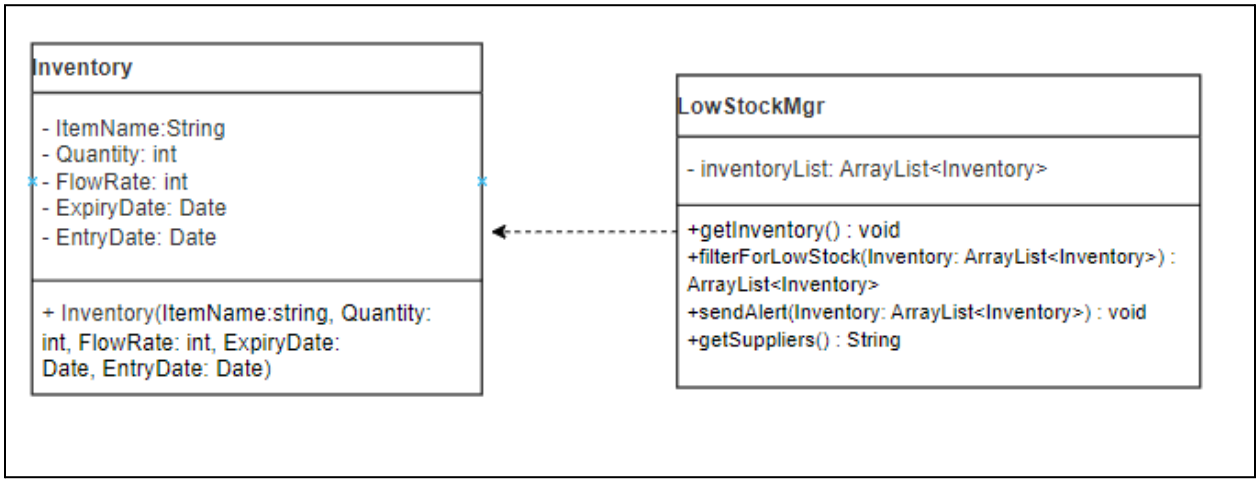
## 5.2 Design Pattern

### 1. Observer Pattern

**Problem 1**

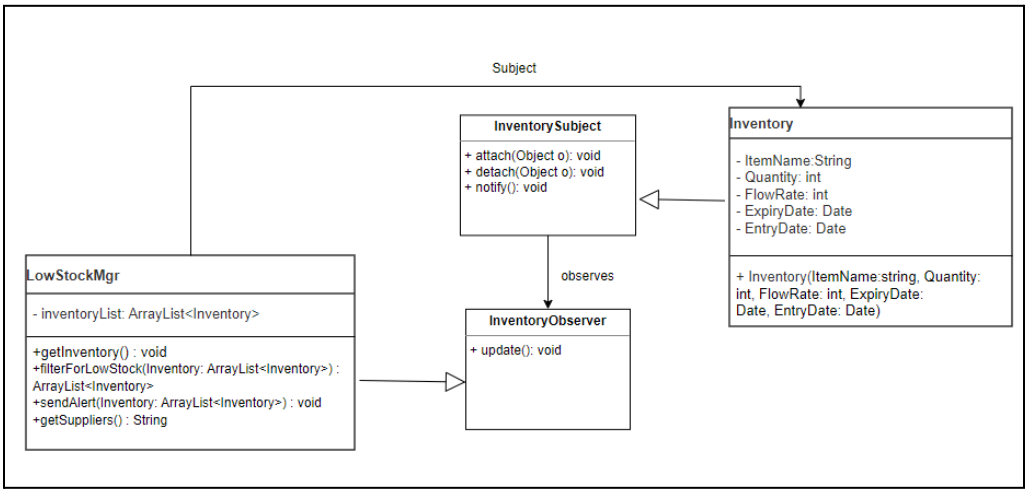
Our application has various controller classes that require consistent real-time updates of the inventory database, in order to identify the moment when the stock for a particular item gets too low (OrderData entity class), and for the user to instantly order new stock to remedy this issue.

At the same time, the OrderData class is highly utilized by multiple components as it holds the core information for our application. If the components are tightly coupled, this would increase the difficulty of adding new components and editing functionalities.




**Solution**

We establish a separation between Subjects and Observers, so that Subjects do not know the Observers' specification and vice versa, leading to increased loose coupling, and improved flexibility to modify classes without affecting other components. Simultaneously, having the Subject-Observer pattern means that we can achieve loosely coupled real-time updates of all Observers.

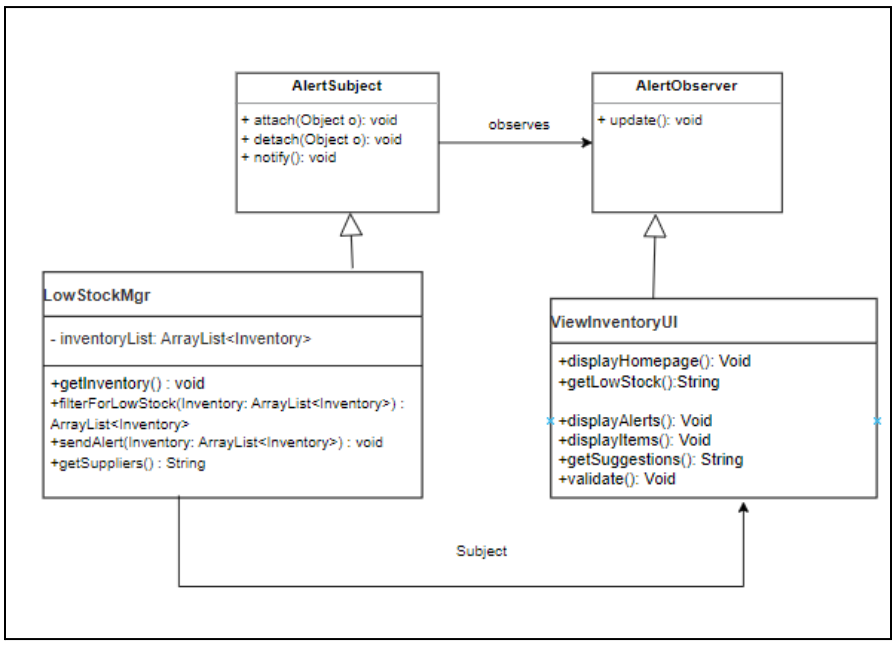


**Problem 2**

Our application has an alerts system where the UI will be alerted during two scenarios; when there are low stocks, and also evening everyday, when the system generates predictions on the livestock demand for the next day. As such, it has to constantly listen for an update, which might be costly and slows down performance.

**Solution**

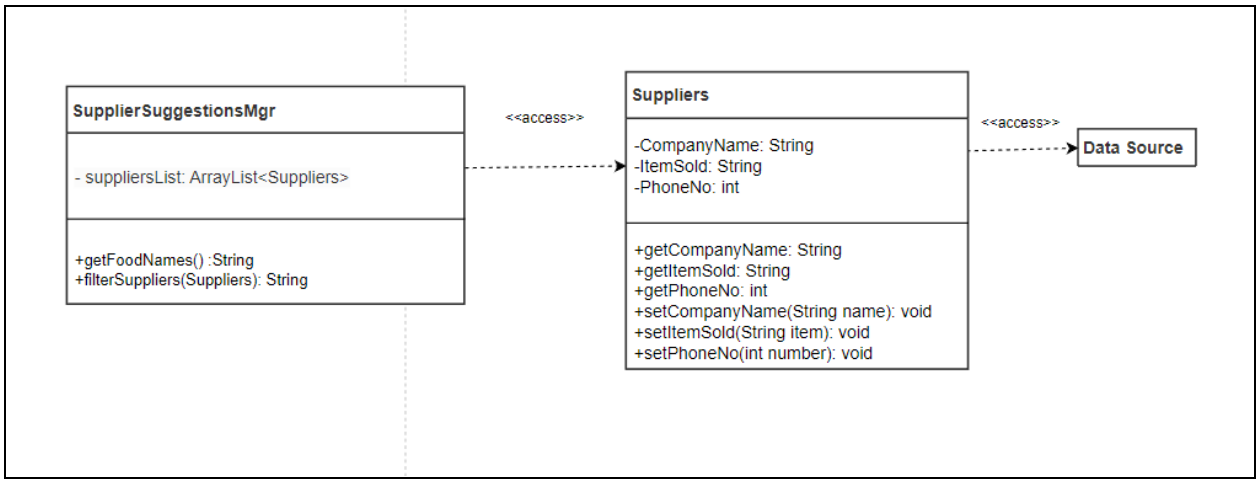
As such, we separate these classes into AlertSubjects which can attach and detach its AlertObservers as needed, and also notify all the relevant Observers. This will ensure timely Alerts from the Subjects, and prevent the Observers from having to constantly listen out for the alert. At the same time, similarly to problem 1, we allow for increased flexibility of adding new observers (eg. User's devices, or other devices that want to be alerted). Lastly, real-time updates are ensured.




2. **Data Access Object Pattern**

**Problem:**

In the restaurant ingredients management system, there may be direct interactions between the business logic (e.g., managing recipes, inventory, ordering) and the database access code. Changes in the database schema, such as adding a new ingredient field or modifying storage procedures, can lead to direct modifications in the business logic, creating a tightly coupled system.




**Solution:**

Create a DAO interface that declares methods for accessing data from the data source, such as retrieving details, updating details, or adding new details. As such, we isolate the database logic from the rest of the application, and business logic components (e.g., supplier manager) interact with the suppliers DAO interface without needing to know the specifics of the database operations. This helps us with flexibility, because if there are changes in the database schema or a decision to switch to a different database system, the modifications are limited to just the DAO implementation.

## 6. Data Dictionary

| Term | Definition |
|---|---|
| User | Human user of System, to view and manage a company's inventory. |
| System | PanTree application system that includes its frontend, backend and database. |
| Shipment | Arrival Shipment. Includes the name and quantity of each item that arrives in the shipment. |
| System Inventory Database | A database that stores and manages all information required by the database. This includes a list of Order Data, Suppliers, Predictions, and Marketplace items. |
| Order Data | The data that is filled in by the user in each Order Form, which corresponds to the quantity of each ingredient available in our system. |
| Order Form | A form that the user can fill up whenever there is a new shipment of stock to the company. |
| Current Inventory Levels | A consolidated list of quantities of all unique items that have been submitted as Order Data from an Order Form (e.g. 5 Chicken Patties, consolidated from multiple Order Forms that submitted Order Data of 1 Chicken Patty and 4 Chicken Patties). |
| Predicted Inventory Levels | A predicted list of quantities of all existing Order Data, based on Current Inventory Levels, and past inventory levels |
| Predictions Dashboard | A visual interface or informational display within the system that provides users with a consolidated view of historical sales information, Current Inventory Levels, and future Predicted Inventory Levels. |
| Menu Item | An item that can be created from a list of ingredients/Order Data that are present in the System Inventory Database. |
| Marketplace Item | An item that can be created from a Menu Item, and have a price and description attached, to be posted into the Marketplace. |
| Marketplace | The platform where the System is able to post Marketplace Items to, and managers of the marketplace can view these Items and sell them to customers with the relevant price and description. |

# 7. Testing

## 7.1 Black Box Testing

---

### 1. Inventory Form

a. Item and quantity

| Test ID | Scenario | Expected Result | Actual Result |
|---|---|---|---|
| 1 | No item chosen and no quantity entered. | The system prompts the user to choose an item and enter a quantity | The system prompts the user to choose an item and enter a quantity |
| 2 | An item is chosen, but no quantity is entered | The system prompts the user to enter a quantity | The system prompts the user to enter a quantity |
| 3 | No item chosen, and an invalid quantity is entered | The system prompts the user to choose an item | The system prompts the user to choose an item |
| 4 | An item is chosen, but an invalid quantity is entered | The system prompts the user to enter a valid quantity | The system prompts the user to enter a valid quantity |
| 5 | No item chosen, but a valid quantity is entered | The system prompts the user to choose an item | The system prompts the user to choose an item |
| 6 | An item is chosen, and a valid quantity is entered | The system displays a successful message provided other fields are valid | The system displays a successful message provided other fields are valid |

b. Expiry Date

| Test ID | Scenario | Expected Result | Actual Result |
|---|---|---|---|
| 1 | No expiry date is chosen | The system prompts the user to choose an expiry date | The system prompts the user to choose an expiry date |
| 2 | An expiry date is chosen | The system displays a successful message provided other fields are valid | The system displays a successful message provided other fields are valid |

c. Password

| Test ID | Scenario | Expected Result | Actual Result |
|---|---|---|---|
| 1 | No password is entered | The system prompts the user to enter a password | The system prompts the user to enter a password |
| 2 | An invalid password is entered | The system prompts the user to enter a valid password | The system prompts the user to enter a valid password |
| 3 | A valid password is entered | The system displays a successful message provided other fields are valid | The system displays a successful message provided other fields are valid |

d. Specific cases (combination)

| Item | Quantity | Expiry Date | Password | Expected Result | Actual Result |
|---|---|---|---|---|---|
| Apple | 10 | 20 Nov 2023 | testpass | Successful submission | Successful submission |
| Empty(“”) | 10 | 20 Nov 2023 | testpass | Please select an Item | Please select an Item |
| Apple | Empty(“”) | 20 Nov 2023 | testpass | Please enter a quantity | Please enter a quantity |
| Apple | 10 | Empty(“”) | testpass | Please choose an expiry date | Please choose an expiry date |
| Apple | 10 | 20 Nov 2023 | Empty(“”) | Please enter a password | Please enter a password |
| Apple | -1 | 20 Nov 2023 | testpass | Please enter a valid quantity | Please enter a valid quantity |
| Apple | 10 | 20 Nov 2023 | wrongpass | Please enter a valid password | Please enter a valid password |

2. **View suggestions**
    a. Price

| Test ID | Scenario | Expected Result | Actual Result |
|---|---|---|---|
| 1 | No price is entered | The system prompts the user to enter a price | The system prompts the user to enter a price |
| 2 | An invalid price is entered | The system prompts the user to enter a valid price | The system prompts the user to enter a valid price |
| 3 | A valid price is entered | The system displays a successful message provided other fields are valid | The system displays a successful message when other fields are valid |

b. Description

| Test ID | Scenario | Expected Result | Actual Result |
|---|---|---|---|
| 1 | No description is entered | The system prompts the user to enter a description | The system prompts the user to enter a description |
| 2 | A valid description is entered | The system displays a successful message provided other fields are valid | The system displays a successful message when other fields are valid |

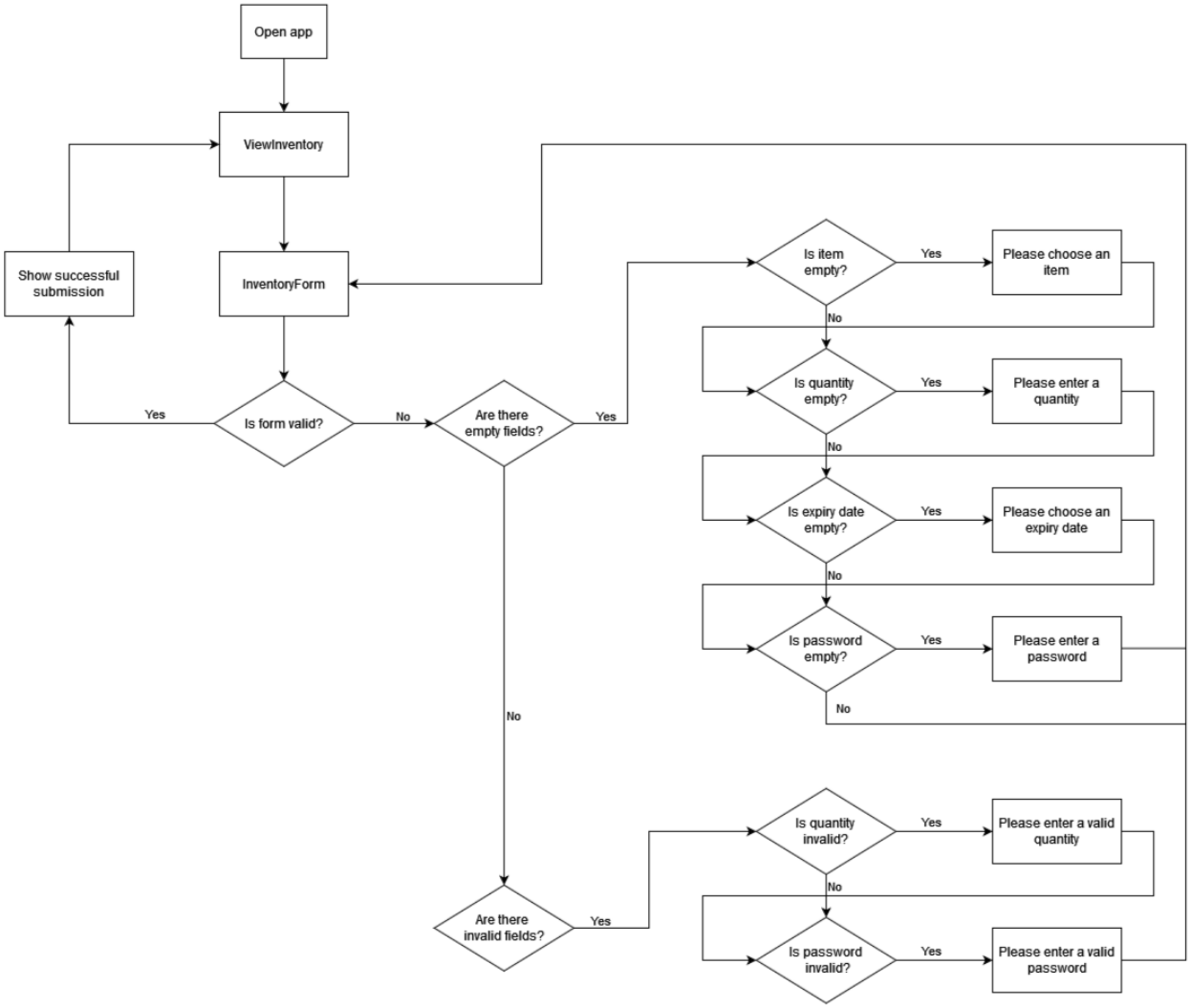
c. Specific cases (combination)

| Price | Description | Expected Result | Actual Result |
|---|---|---|---|

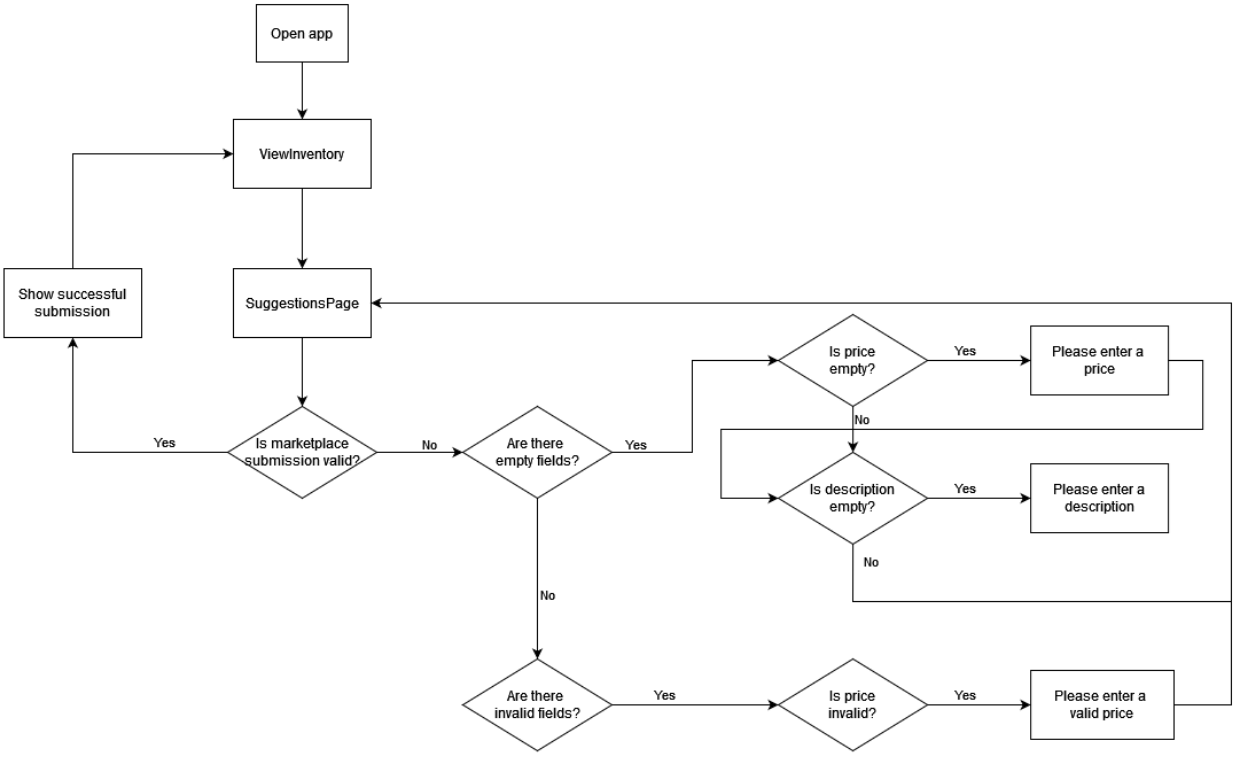
| | | | |
|---|---|---|---|
| 3.50 | Delicious | Successful posting on the marketplace | Successful posting on the marketplace |
| Empty(“”) | Delicious | Please enter a price | Please enter a price |
| 3.50 | Empty(“”) | Please enter a description | Please enter a description |
| 0 | Delicious | Please enter a valid price | Please enter a valid price |
| -1 | Delicious | Please enter a valid price | Please enter a valid price |

## 7.2 White Box Testing

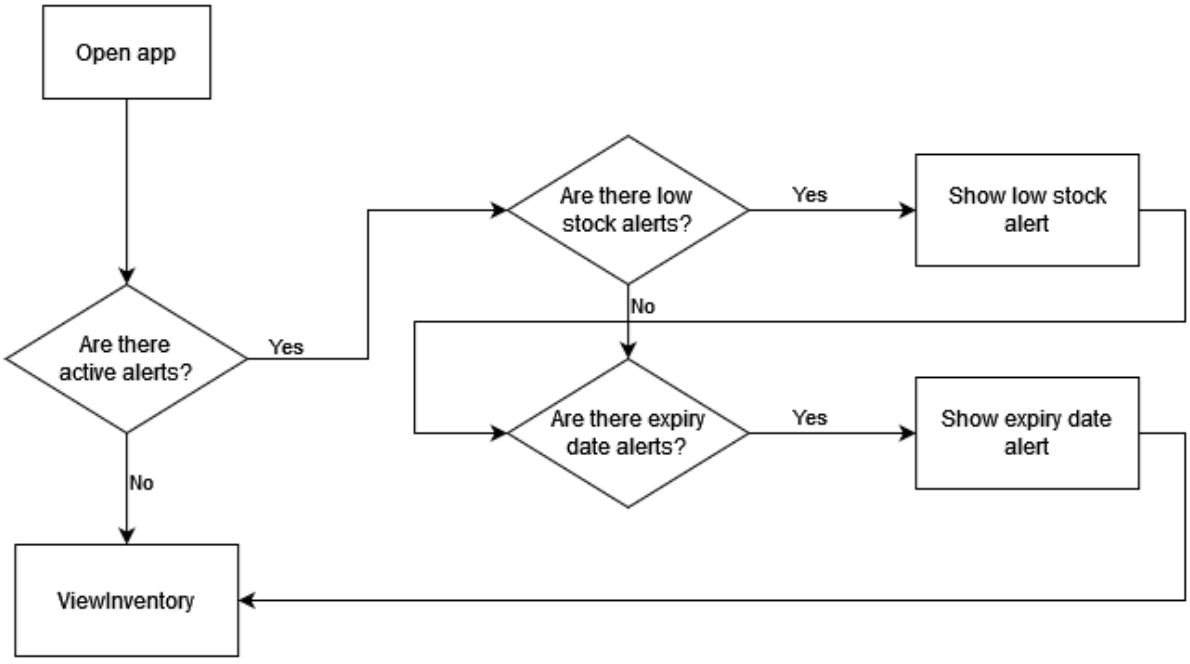
1. Inventory Form



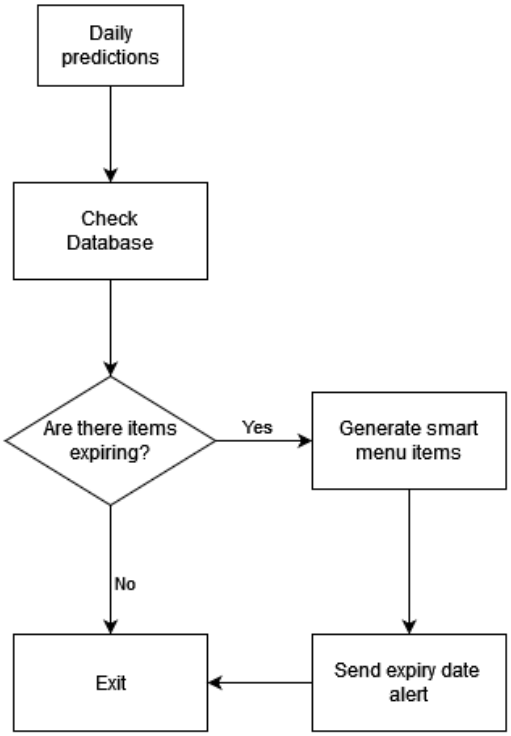

2. Suggestions Page

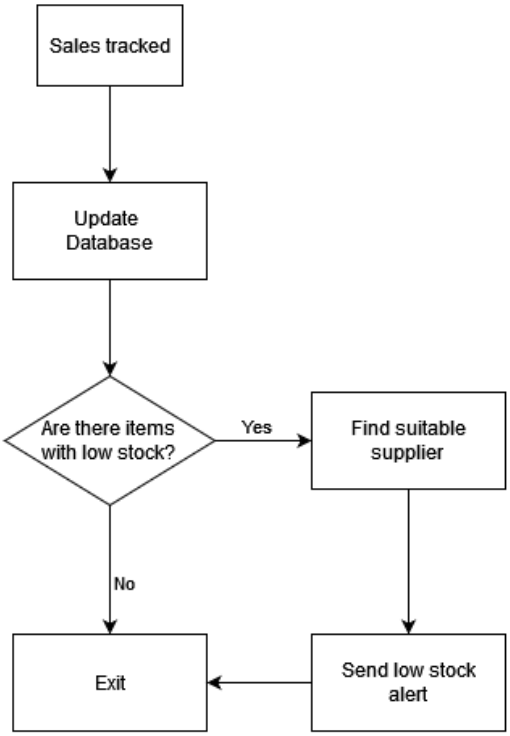



3. Alerts
    a. Frontend




    b. Backend




## 8. Appendix

For more information and a detailed demo on our PanTree app, please refer to the link below.
https://drive.google.com/drive/folders/1W_5ZapaPKtrs3cc2_Q0Mmeix-hCvKpaz?usp=sharing