COVER PAGE

CONTENTS

1. Functional Requirements
2. Non-Functional Requirements
3. Use Case Diagram
4. Use Case Description
5. Data Dictionary

# **1. Functional Requirements**

## **1. Data System**

1.1 The system must extract details of food deal details from the Telethon API, in the following format:

Image of food deal.

Message of food deal.

Time of message sent.

Type of food in food deal.

Location of food deal.

Timeframe of food deal.

Type of deal.

1.2 The system must update the food deal details whenever there is a new food deal message.

## **2. Account**

2.1 The application must allow users to register for an account.

2.1.1 The application must ensure that users fill up a registration form using the following format:

Email Address

Username

Create Password

2.1.2 During registration, the application must validate the account registration.

2.1.3 The email address must not be in use.

2.1.4 The username must be unique.

2.1.5 The password must follow the criteria below.

2.1.5.1 The password should be 8-16 characters long

2.1.5.2 The password should have at least 1 uppercase character

2.1.5.3 The password should have at least 1 lowercase character

2.1.5.4 The password should have at least 1 digit

2.1.2 After registration, the application must allow the user to state their preferences in the format:

Preferred cuisines

Preferred restaurants

Preferred locations

Preferred type of deals

2.2 The application must allow users to sign in to their account.

2.2.1 The application must ensure that users fill up their account details in the following format:

Username

Password

2.2.2 The application must validate the account details.

2.3 The application must record the history of the last 10 deals users click on.

2.4 The application must allow users to view the history of the deals clicked on.

2.5 The application must allow users to bookmark their favourite deals.

## **3. Display deals**

3.1 The application must categorise the deals according to the timeframe.

3.2 The application must display the photo of the deal from the original source.

3.3 The application must provide a summarised description of the deal.

3.3.1 The application must provide the exact location of the deal.

3.3.2 The application must display the expiry of the deal.

3.3.3 The application must state the company of the deal.

3.3.4 The application must display the type of deal.

3.4 The application must remove deals that have expired.

3.5 The application must recommend deals based on the user’s current location.

3.6 The application greys out a deal when it is clicked on by the user.

3.6.1 The greyed out deals are reset every week.

3.7 The application must recommend deals based on the user's preset preferences.

## **4. Filtering**

4.1 The application must be able to filter by location selected by the user.

4.2 The application must be able to filter by company selected by the user.

4.3 The application must be able to filter by a specific time frame selected by the user.

4.4 The application must be able to filter by the type of deals selected by the user.

4.5 The application must be able to filter the deals by earliest expiry.

## **5. Popular Deals**

5.1 The page must feature the top 10 most popular deals based on the user’s location.

5.1.1 If the featured deal has expired, replace the expired featured deal with the next popular deal available.

5.1.2 The user can see the best deals based on a manually inputted location.

## **6. Code Redemption**

6.1 The page allows users to filter the available promotion codes.

6.1.1 The page resets the filter when the user exits the Code Redemption page.

6.2 The page shows users a summarised description of the available promotion codes after selection.

6.2.1 The page must display the deal.

6.2.2 The page must show the terms and conditions that apply for the specific promotion code.

6.2.3 The page must display the promotion code’s timeframe.

6.3 The promotion codes that are viewed must be greyed out.

6.3.1 The greyed out codes must be reset weekly

## **7. Reviews/ Feedback forum**

7.1 The application must allow users to like other users’ reviews.

7.2 The application must allow users to post their reviews.

7.3 The application must allow users to share reviews with their contacts.

7.4 The application must allow users to report reviews that are inappropriate.

7.5 The application must display reviews according to the respective restaurants.

7.6 The application must display the most liked review at the top of the forum.

**8. Communication**

8.1 Users can send deals to friends.

## **9. Booking and Reservation**

9.1 The application must enable users to reserve tables at restaurants.

9.2 The application must allow users to choose the time slots of their reservation.

9.3 The application must allow users to call the restaurant or shop.

## **10. Order Deals**

9.1 The application must allow users to place an order upon viewing the deals.

9.2 The application must allow users to opt for delivery services.

9.2.1 The application must allow orders to be delivered to user’s specified address.

9.2.2 The application must display a map showing the live location of the delivery personnel.

9.2.3 The application must display the estimated time of delivery.

9.2.4 The application must send a notification to inform the user that the order has arrived.

9.3 The application must allow users to opt for self collection.

9.4 The application must allow users to make online payment.

9.4.1 The application must allow payment to be made through a card.

# **2. Non-Functional Requirements**

## **1. Performance Requirements**

**1.1 Data Extraction System**

1.1.1 Update Function - The system must extract from its relevant source, and update the database within 3 seconds, whenever there is new data.

**1.2 Application**

1.2.1 Display Deals Function - The application must display all current deals within 3 seconds.

1.2.2 “Deals Near Me” Function - The application must find the user’s location, and restaurants, with deals, that are at the same location as the user, within 3 seconds.

1.2.3 Filter Function- The application must filter deals, based on the user’s selection of category, within 3 seconds.

## **2. Security Requirements**

**2.1 Data Extraction System**

2.1.1 The system must not extract any data that may breach internet privacy. This includes usernames, profile pictures, comments, and private links.

**2.2 Application**

2.2.1 The application must only be able to access the data in the data extraction system. This is to prevent any potential leakages of private data.

## **3. Software Quality Attributes**

**3.1 Adaptability**

3.1.1 The application should be able to adapt its behaviour according to the user’s information.

3.1.2 The application should show information and deals based on the user’s information and preferences.

**3.2 Flexibility**

3.2.1 The application should be able to able to handle bugs and crashes, rebooting to a previous saved state until the system is operating again.

**3.3 Interoperability**

3.3.1 The application and the data extraction system must be able to exchange information needed to for the application to function.

**3.4 Maintainability**

3.4.1 Third party libraries and frameworks to the most recent latest version no longer than 30 days since their release date.

3.4.2 System and Application must have all infrastructure in code to simplify changes to provisioning and deployment processes.

**3.5 Portability**

3.5.1The application should be compatible with both IOS and Android systems.

**3.6 Reliability**

3.6.1 The data extraction system must be accurate and kept up to date with all information available in its relevant source.

3.6.2 The application must not take longer than 10 seconds to load.

**3.7 Reusability**

3.7.1 The code library used for coding the application should be generic enough to be easily used in other systems with minimal changes to be reused.

**3.8 Robustness**

3.8.1 The application should be able to cope with errors during execution.

3.8.2 Previous states must besaved to reboot when faced with errors with execution.

3.8.3 The application must have test cases to deal with erroneous input from the user.

**3.9 Testability**

3.9.1 The system should be easy to test and find bugs by having 98% unit test coverage for backend and frontend code.

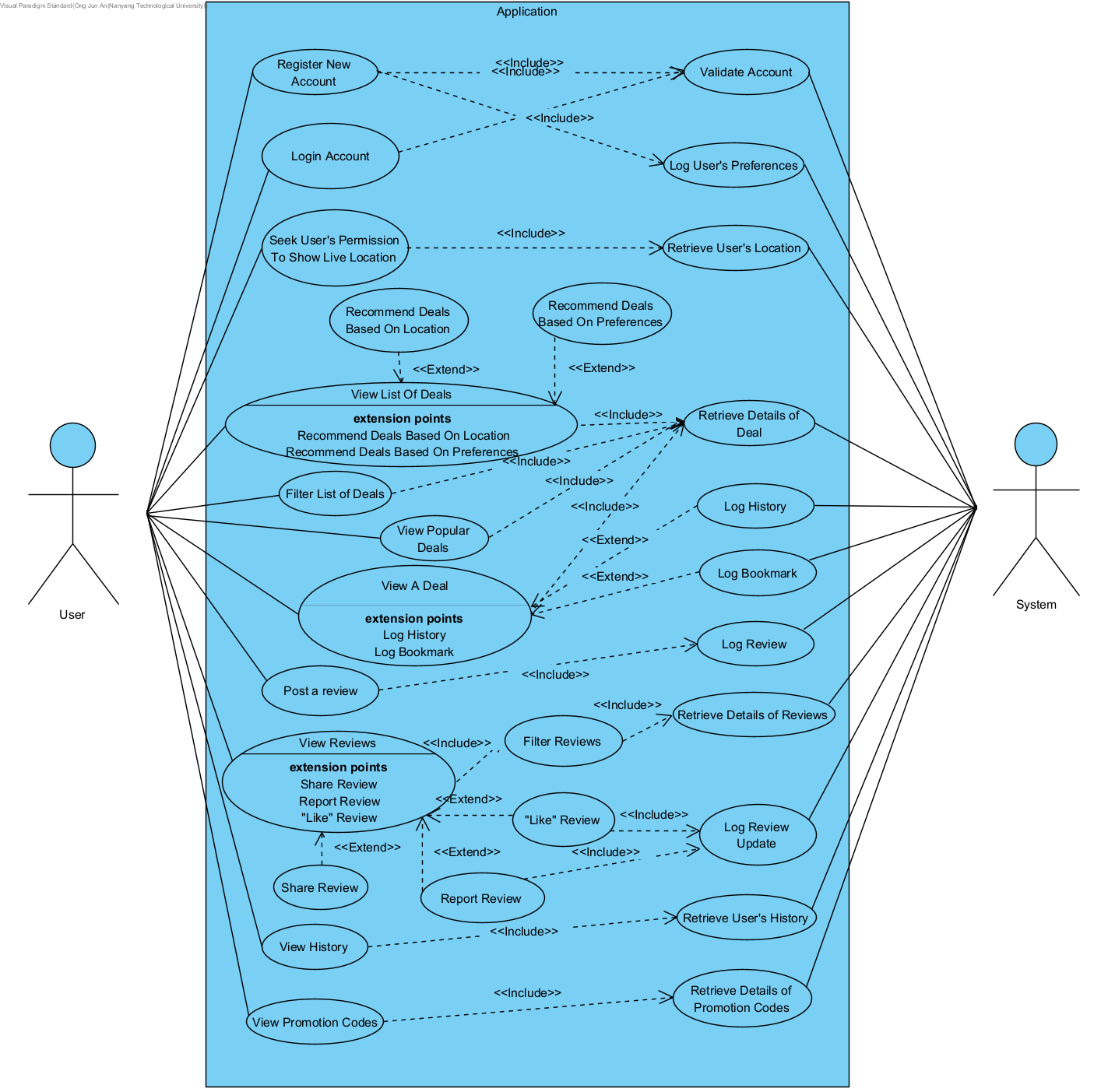
**3.10 Usability**

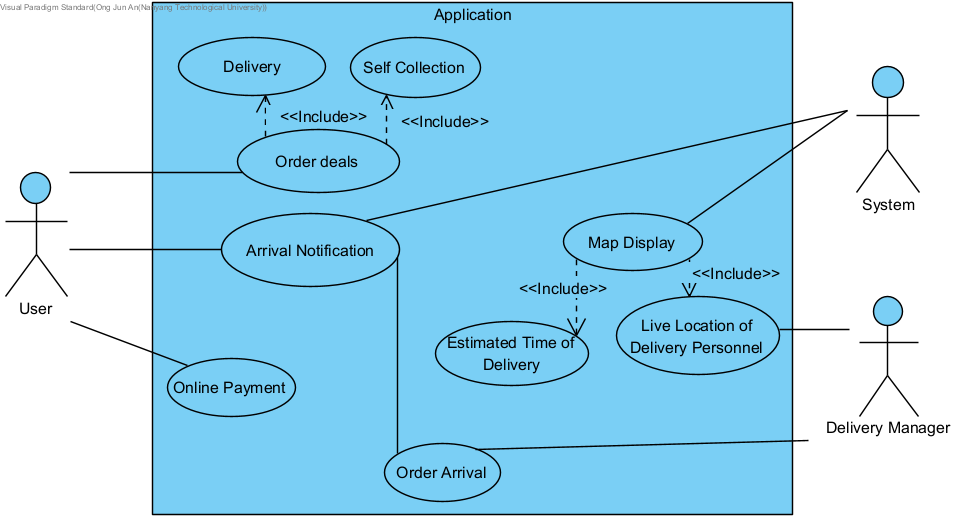
3.10.1 60% of users must find the information on deals displayed within the application to be concise and easy to read.

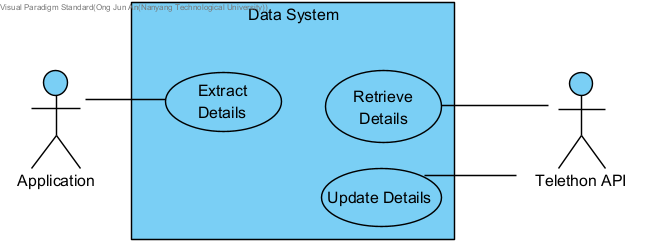
3.10.2 The application must have a “Frequently Asked Question (FAQ)” section to guide users in using it.

3.10.3 60% of users must find the FAQ section to be comprehensive and easily accessible.

# **3. Use Case Diagram**

****

****

****

# **4. Use Case Description**

​​

| Use Case ID: | 1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Extract Details | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | System, Telethon API |
| --- | --- |
| Description: | System extracts data from telethon API. |
| Preconditions: |  |
| Postconditions: |  |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. System extracts the image of food deal. 2. System extracts the message of food deal. 3. System extracts the time of message sent. 4. System extracts the type of food in food deal. 5. System extracts the location of food deal. 6. System extracts the timeframe of food deal. 7. System extracts the type of deal. |
| Alternative Flows: |  |
| Exceptions: |  |

| Use Case ID: | 2 | | |
| --- | --- | --- | --- |
| Use Case Name: | Update Details | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | System |
| --- | --- |
| Description: | System updates the food details from the new message. |
| Preconditions: |  |
| Postconditions: |  |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: |  |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 3 | | |
| --- | --- | --- | --- |
| Use Case Name: | Account Registration | | |
| Created By: | Justin Chua | Last Updated By: |  |
| Date Created: | 5th September 2022 | Date Last Updated: |  |

| Actor: | User, System |
| --- | --- |
| Description: | User registers for a new account. |
| Preconditions: | 1. User account must not exist in the database 2. Device must be connected to the internet. |
| Postconditions: | 1. Instructions for 2-step verification will be sent to the user via email. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User enters desired username, email address, password. 2. User presses the ‘next’ button. 3. System checks that username, email address, and password are valid. 4. System sends a verification code to the indicated email address. |
| Alternative Flows: | AF-S3: System detects that either username, email address, and/or password is invalid.   1. System displays error message in red above the respective input. 2. Return to step 1. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 4 | | |
| --- | --- | --- | --- |
| Use Case Name: | Check Registration | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | System |
| --- | --- |
| Description: | System validates registration details and creates account for user. |
| Preconditions: | 1. User account must not exist in database 2. Device must be connected to the internet |
| Postconditions: | 1. User account is successfully created and stored in database. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. System validates username, password and email address. 2. System validates account availability. 3. System creates account for user in database. |
| Alternative Flows: | AF-S1-1: System detects username exists in database.   1. System displays error message “Username is taken.” 2. User re-enters registration details. 3. Return to step 1.     AF-S1-2: System detects weak password.   1. System displays error message “Password is weak.   - The password should be 8-16 characters long.  - The password should have at least 1 uppercase character.  - The password should have at least 1 lowercase character.  - The password should have at least 1 digit.”   1. User re-enters registration details. 2. Return to step 1.     AF-S1-3: System detects invalid email address.   1. System displays error message “Invalid email address.” 2. User re-enters registration details. 3. Return to step 1.     AF-S2: System detects account already exists in database.   1. System displays error message “Account already exists.” 2. User re-enters registration details. 3. Return to step 1. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 5 | | |
| --- | --- | --- | --- |
| Use Case Name: | Account Login | | |
| Created By: | Justin Chua | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User, System |
| --- | --- |
| Description: | User logs into unique account |
| Preconditions: | 1. User must be connected to the internet 2. User account must exist in database. |
| Postconditions: | 1. User will be logged into their respective accounts. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User enters username and password in the login page. 2. User presses login. 3. System validates the account by checking the username and password against the database. 4. System sends a 6-digit verification code to user’s email address. 5. User enters 6-digit code in the verification page. 6. System validates the verification code entered with the one sent to user’s email address. 7. System authenticates user to login successfully. |
| Alternative Flows: | AF-S2: System detects empty username and/or password fields.   1. System displays error message, and prompts user to fill up empty fields. 2. Return to step 1.     AF-S3: System detects mismatched or non-existent username and password.   1. System displays error message “Invalid username or password.” 2. Return to step 1.     AF-S6: User enters wrong verification code.   1. System displays error message “Incorrect. Try again.” 2. System allows user to attempt 3 times. 3. Upon 3 failed attempts, return to step 1. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 6 | | |
| --- | --- | --- | --- |
| Use Case Name: | Indicate Preferences | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User, System |
| --- | --- |
| Description: | User indicates deal preferences. |
| Preconditions: | 1. User must have an existing account. 2. User must login successfully into account. |
| Postconditions: | 1. Deals will be filtered according to user’s preferences. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User navigates to preferences tab. 2. System displays different cuisines, restaurants, locations, and types of deals available in 4 separate sections. 3. User clicks on the checkboxes beside their preferred cuisines, restaurants, locations and type of deals. |
| Alternative Flows: | AF-S3: User has no preference.   1. User clicks on ‘select all’ checkbox at the bottom of each section. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 7 | | |
| --- | --- | --- | --- |
| Use Case Name: | Bookmark Deals | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User, System |
| --- | --- |
| Description: | User can save their favourite deals for future reference. |
| Preconditions: | 1. User must login successfully to their existing account. 2. User must click on the deal. |
| Postconditions: | 1. Deal will be saved into database. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User navigates to deals page. 2. User clicks on the deal to view details. 3. User clicks the bookmark icon. 4. System records and saves deal into database. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 8 | | |
| --- | --- | --- | --- |
| Use Case Name: | Record History | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | System |
| --- | --- |
| Description: | System records history of deals user clicked on. |
| Preconditions: | 1. User must be logged into an existing accoumt. 2. Device must be connected to the internet. 3. User must click on deals. |
| Postconditions: | 1. System records the deals clicked on into the database. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User navigates to view deals page.  2. User clicks on deal to view details.  3. System records deal into database.  4. User exits back to view deals page.  5. System greys out deal that was clicked on. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 9 | | |
| --- | --- | --- | --- |
| Use Case Name: | View History | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User |
| --- | --- |
| Description: | User can view history of deals they clicked on. |
| Preconditions: | 1. User must be logged into their existing account. 2. Device must be connected to the internet. 3. User must have clicked on deals. |
| Postconditions: | 1. System displays list of the last 10 deals user clicked on. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User navigates to the history tab. 2. System displays the last 10 deals user clicked on. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 10 | | |
| --- | --- | --- | --- |
| Use Case Name: | Display Deals | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | System |
| --- | --- |
| Description: | System displays deals under view deals tab. |
| Preconditions: | 1. Device must be connected to the internet. 2. User must be logged into an existing account. |
| Postconditions: | 1. System displays available deals. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User navigates to view deals tab. 2. System displays all available deals. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 11 | | |
| --- | --- | --- | --- |
| Use Case Name: | Recommend Deals | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | System |
| --- | --- |
| Description: | System will recommend deals near user, according to their indicated preferences. |
| Preconditions: | 1. User must login to an existing account. 2. User must turn on and enable system to access live location. 3. User must indicate preferences. 4. User must be connected to the internet. |
| Postconditions: | 1. System will display recommended deals catered to user’s preferences and convenience. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. System uses user’s live location to filter out deals around the area. 2. System matches filtered deals with user’s preferences. 3. System displays the matching deals as recommendations for user. |
| Alternative Flows: | AF-S1: Live location is not enabled.   1. System displays error message “Live location is not enabled. Proceed to settings to allow live location.” 2. User enables live location. 3. Return to step 1.     AF-S2: No deals available around user’s location.   1. System will recommend deals according to user’s preferences only. 2. Return to step 3. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 12 | | |
| --- | --- | --- | --- |
| Use Case Name: | Filtering the deals available | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User |
| --- | --- |
| Description: | To allow the User to filter the deals available by categories provided by the System. |
| Preconditions: | 1. Mobile device must be connected to an internet connection (Wifi/Cellular Data) |
| Postconditions: | 1. A filtered list of deals will be displayed by the system based the User’s selection. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. System will provide a selection of categories that can be filtered by the System. 2. User can filter the list of deals based on location, company, time frame, type of deals, and expiry date. 3. User will select on the types of filters. 4. System will display the filtered list of deals based on the User’s selection. |
| Alternative Flows: | AF-S4: User provides a change in the types of filters to be used.   1. User can add or remove filters in his/her selection. 2. Return to step 4.   AF-S4: User taps on the ‘Reset filters’ button.   1. User can remove all filters in his/her selection. 2. Return to step 1. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 13 | | |
| --- | --- | --- | --- |
| Use Case Name: | Featuring the most popular deals available | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User |
| --- | --- |
| Description: | System will provide a page to allow the user to view the top 10 most popular deals based on location. |
| Preconditions: | 1. Mobile device must be connected to an internet connection (Wifi/Cellular Data) |
| Postconditions: | 1. User leaves the page. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. System will calculate the popularity among the deals available. 2. System will filter the deals based on location. 3. System will display the top 10 most popular deals based on the User’s location. |
| Alternative Flows: | AF-S2: User manually inputs the location he/she wants.   1. System will provide a location filter. 2. User selects the location. 3. Return to step 3. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| **Use Case ID:** | ­14 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Code Redemption | | |
| **Created By:** | Jewel | **Last Updated By:** | Jewel |
| **Date Created:** | 1/9/22 | **Date Last Updated:** | 1/9/22 |

| **Actor:** | System |
| --- | --- |
| **Description:** | Users can browse through available promotion codes. |
| **Preconditions:** | User must be signed in. |
| **Postconditions:** | Promotion codes that users have interacted with are greyed out. |
| **Priority:** | Medium |
| **Frequency of Use:** | Once a day |
| **Flow of Events:** | 1. User views a list of unfiltered available promotion codes. 2. User clicks on a promotion code. 3. User views a summarised description of the deal. 4. User clicks the Back button. 5. User is redirected back to the list of promotion codes. 6. The previously viewed promotion code is now greyed out. |
| **Alternative Flows:** | AF-S1: User is allowed to filter the available promotion codes. |
| **Exceptions:** | - |
| **Includes:** | Filter |
| **Special Requirements:** | 1. Greying out of the promotion code must be executed in 3 seconds. 2. Reset greyed out promotion codes weekly. 3. Filtered list of promotion codes must be reset in 5 seconds after the user exits the Code Redemption page. |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** | **15** | | |
| --- | --- | --- | --- |
| **Use Case Name:** | **View Reviews** | | |
| **Created By:** | **Sean Lim** | **Last Updated By:** | **Sean Lim** |
| **Date Created:** | **1/9/22** | **Date Last Updated:** | **1/9/22** |

| **Actor:** | **User** |
| --- | --- |
| **Description:** | **View reviews made by other users. Users should be able to interact with the reviews.** |
| **Preconditions:** | **-** |
| **Postconditions:** | **Reviews seen should be filtered out with new reviews** |
| **Priority:** | **High** |
| **Frequency of Use:** | **5 days a week** |
| **Flow of Events:** | 1. **Users will click into the restaurant of their choice** 2. **Users will then be able to view the reviews of that restaurant made by other users** |
| **Alternative Flows:** | **AF-S1: Users can also view reviews at the home page**   1. **Users scrolling down the home page will view trending reviews recently made by other users.** |
| **Exceptions:** | **-** |
| **Includes:** | 1. **Like Review** 2. **Report Review** 3. **Share Review** |
| **Special Requirements:** |  |
| **Assumptions:** | 1. **Reviews are constantly made by users.** |
| **Notes and Issues:** | **-** |

| **Use Case ID:** | **16** | | |
| --- | --- | --- | --- |
| **Use Case Name:** | **Like Review** | | |
| **Created By:** | **Sean Lim** | **Last Updated By:** | **Sean Lim** |
| **Date Created:** | **1/9/2022** | **Date Last Updated:** | **1/9/2022** |

| **Actor:** | **User** |
| --- | --- |
| **Description:** | **User will be able to interact with the reviews made by other users in terms of “likes”. The more “likes” the review has, the algorithm will deem the post trending and it will more likely appear on other user’s home page.** |
| **Preconditions:** | **User must be signed in.**  **User account must be authenticated.** |
| **Postconditions:** | **Number of likes on a review will increase.** |
| **Priority:** | **High** |
| **Frequency of Use:** | **5 days a week** |
| **Flow of Events:** | 1. **Users will be able to view the reviews made by other users when looking at a restaurant and in the homepage** 2. **Users will be able to double tap the review and give the review a like.** 3. **The number of likes for that review will increase.** |
| **Alternative Flows:** | **AF-S1: The user is not signed in.**   1. **User double tap to like a review without being signed in.** 2. **Users will be redirected to the account login/ registration page.** |
| **Exceptions:** | **-** |
| **Includes:** | **-** |
| **Special Requirements:** | **-** |
| **Assumptions:** | **-** |
| **Notes and Issues:** | **-** |

| **Use Case ID:** | **17** | | |
| --- | --- | --- | --- |
| **Use Case Name:** | **Share Reviews** | | |
| **Created By:** | **Sean Lim** | **Last Updated By:** | **Sean Lim** |
| **Date Created:** | **1/9/2022** | **Date Last Updated:** | **1/9/2022** |

| **Actor:** | **User** |
| --- | --- |
| **Description:** | **User will be able to share the reviews with their friends through mainstream messaging applications or by copying the link.** |
| **Preconditions:** | **-** |
| **Postconditions:** | **-** |
| **Priority:** | **Medium** |
| **Frequency of Use:** | **Once a week** |
| **Flow of Events:** | 1. **Users will come across a review they would like to share to someone.** 2. **Users will be able to press a button open up the menu to display the options available for the user to share the review.** 3. **Users will choose and press the method of sharing.** 4. **Users will be redirected to share the review.** 5. **Users will also be able to get and copy the link to the review for sharing.** |
| **Alternative Flows:** | **-** |
| **Exceptions:** | **-** |
| **Includes:** | **-** |
| **Special Requirements:** | **-** |
| **Assumptions:** | **-** |
| **Notes and Issues:** | **-** |

| **Use Case ID:** | **18** | | |
| --- | --- | --- | --- |
| **Use Case Name:** | **Report Review** | | |
| **Created By:** | **Sean Lim** | **Last Updated By:** | **Sean Lim** |
| **Date Created:** | **1/9/2022** | **Date Last Updated:** | **1/9/2022** |

| **Actor:** | **User** |
| --- | --- |
| **Description:** | **User will able to report a post. The post may contain offensive or sensitive content. The post will then be reviewed and may be taken down if deemed offensive or sensitive.** |
| **Preconditions:** | **-** |
| **Postconditions:** | **The post may be taken down** |
| **Priority:** | **High** |
| **Frequency of Use:** | **One day a week** |
| **Flow of Events:** | 1. **User will come across a review that the user thinks has offensive or sensitive content.** 2. **User will press the 3 dots.** 3. **User will see a option to report the post** 4. **Upon tapping that option, the user will be prompted to write a brief description to why the user found the post offensive or sensitive.** 5. **The window will tell the user that the report has been submitted and thank the user.** |
| **Alternative Flows:** | **-** |
| **Exceptions:** | **-** |
| **Includes:** | **-** |
| **Special Requirements:** | **-** |
| **Assumptions:** | **There will be someone to review the reports and deem which reviews are indeed offensive or sensitive.** |
| **Notes and Issues:** | **-** |

| Use Case ID: | 19 | | |
| --- | --- | --- | --- |
| Use Case Name: | Making a booking with a restaurant through the application | | |
| Created By: |  | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User, Restaurant |
| --- | --- |
| Description: | To allow the user to be able to make a booking with a restaurant specified by the user |
| Preconditions: | 1. Mobile device must be connected to an internet connection (Wifi/Cellular Data)   And   1. User must log-in to his/her account. |
| Postconditions: | 1. System has successfully booked a restaurant for the user.   Or   1. System is unable to make a reservation due to the restaurant’s unavailability. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. System will check with the Restaurant on the availability of timeslots. 2. System will provide a list of timeslots available for booking. 3. User will enter a timeslot and the number of seats to be reserved. 4. System will provide the details of the booking for confirmation. 5. User will validate the details of the booking by tapping on the ‘confirm’ button. 6. System will notify the Restaurant on the details of the booking. 7. System will notify the User that the reservation has been made. |
| Alternative Flows: | AF-S5: User requests to make changes to his/her booking.   1. User will request a change in his/her booking by tapping on the ‘back’ button. 2. Return to step 3. |
| Exceptions: | EX1-Restaurant is unavailable for booking.   1. System will notify the User with a message “Restaurant is unavailable for booking.”. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 20 | | |
| --- | --- | --- | --- |
| Use Case Name: | Delivery | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User |
| --- | --- |
| Description: | User can opt for delivery of deals. |
| Preconditions: | 1. Device must be connected to the internet. 2. User must be logged into an existing account. |
| Postconditions: | 1. System records that order is to be delivered. 2. System displays payment page |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User navigates to view deals page. 2. User clicks ‘order’ on deal. 3. System displays menu. 4. User selects items to order from menu. 5. User clicks on ‘delivery’ option. 6. User enters specified address. 7. User clicks ‘next’. 8. User checks address validity. 9. System prompts user to save address. 10. User clicks ‘yes’. 11. System records address into database. 12. System displays payment page. |
| Alternative Flows: | AF-S5-1: User does not want delivery services.   1. User clicks on ‘self collection’. 2. Skip to step 11.     AF-S8: Address entered is invalid.   1. System displays error message “address is invalid”. 2. Return to step 6.     AF-S9: User does not want address saved.   1. User clicks ‘no’. 2. Skip to step 12. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 21 | | |
| --- | --- | --- | --- |
| Use Case Name: | Online Payment | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | User |
| --- | --- |
| Description: | User pays for order using online payment. |
| Preconditions: | 1. Device must be connected to the internet. 2. User must be logged in to an existing account. 3. User must place an order. |
| Postconditions: | 1. System displays transaction page and order is processed. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. System will display a payment page. 2. User clicks ‘proceed to payment’. 3. System displays a form for user to fill up card details. 4. User enters the card number, name on card, cvv, and date of expiry. 5. User clicks ‘next’. 6. System checks validity of card. 7. System prompts user to save card details. 8. User clicks ‘yes’. 9. System records card details into database. 10. System displays transaction page. |
| Alternative Flows: | AF-S6: Card details are invalid.   1. System displays an error message “Card is invalid”. 2. Return to step 3.     AF-S7: User does not want card details saved.   1. User clicks ‘no’ upon prompt. 2. Skip to step 10. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 22 | | |
| --- | --- | --- | --- |
| Use Case Name: | Map Display | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | System |
| --- | --- |
| Description: | System will display a map. |
| Preconditions: | 1. Device must be connected to the internet. 2. User must be logged into an existing account. 3. User must place an order. 4. User must opt for delivery services. |
| Postconditions: | - |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. User navigates to activities tab. 2. User clicks on pending order. 3. System displays map, location of delivery personnel, and estimated time of arrival. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 23 | | |
| --- | --- | --- | --- |
| Use Case Name: | Order Arrival | | |
| Created By: | Justin | Last Updated By: |  |
| Date Created: |  | Date Last Updated: |  |

| Actor: | Delivery Personnel, System, User. |
| --- | --- |
| Description: | Arrival of deal orders. |
| Preconditions: | 1. Device must be connected to the internet. 2. User must be logged into an existing account. 3. User must place an order. |
| Postconditions: | 1. User receives order arrival notification. |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. Delivery personnel arrives at specified location. 2. System detects that delivery personnel is at specified location. 3. System sends arrival notification to User. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

# **5. Data Dictionary**

| **Term** | **Definition** |
| --- | --- |
| User | User refers to a person who uses the application. |
| Data System | Data system refers to a system that retrieves, updates and provides details about food deals, account information, and any other information linked to a user’s account (e.g. Food preferences, reviews, and history) |
| Application | Application refers to the EatNow mobile application. |
| Deals | Deals are discounts, given by food and beverage (f&b) establishments. |
| Image/Photo | Image refers to a picture of a food deal, recorded in JSON format. |
| Message | Message refers to a string of characters, and emojis that depict the details of the food deal. |
| Time | Time is recorded as Singapore Standard Time (SST), which is UTC +8. |
| Location | Location refers to a place where a person can redeem a food deal. |
| Timeframe | Timeframe refers to a food deal’s start to end date. |
| Type of deal | Type of deal refers to a category of food deals, namely 1-for-1, less than 50%, more than 50%, etc. |
| Registration | Registration is a feature that allows users to create an account to bookmark their favourite deals, view their history, post a review, leave a rating, and share their reviews. |
| Account | A user’s account will have an email address, username, and password. |
| Preset preferences | When a user registers for an account, users picking their preferences allows the application to recommend and display deals to provide users with a more personalised experience. |
| History | History is a feature that shows the users’ last 10 deals that they have viewed. |
| Bookmark | Bookmark is a feature that allows the users to favourite their food deal. |
| Display | Display is a feature that shows the list of available food deals. It will show the photo of the deal, and summarised description of the deal. |
| Filtering | Filtering is a feature that allows users to filter the list of food deals according to location, food company, time frame, type of deals, and deals by earliest expiry. |
| Popularity | It is calculated by taking the total number of stars received by reviews divided by the total number of reviews. |
| Popular deals | Popular deals refer to food deals that have received the most popularity. The application will feature the top 10 most popular deals. |
| Review | Review is a feature that allows users to comment on the food deals, and leave a rating. |

# **6. UI Mockup**

# **For Reference**

**1.** **System Features**

***<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>***

**1.1** **Display Deals**

**4.1.1 Description and Priority**

***<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>***

**4.1.2 Stimulus/Response Sequences**

***<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>***

**4.1.3 Functional Requirements**

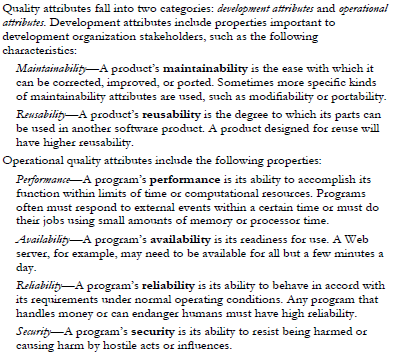
***<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>***

***<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>***

**REQ-1:**

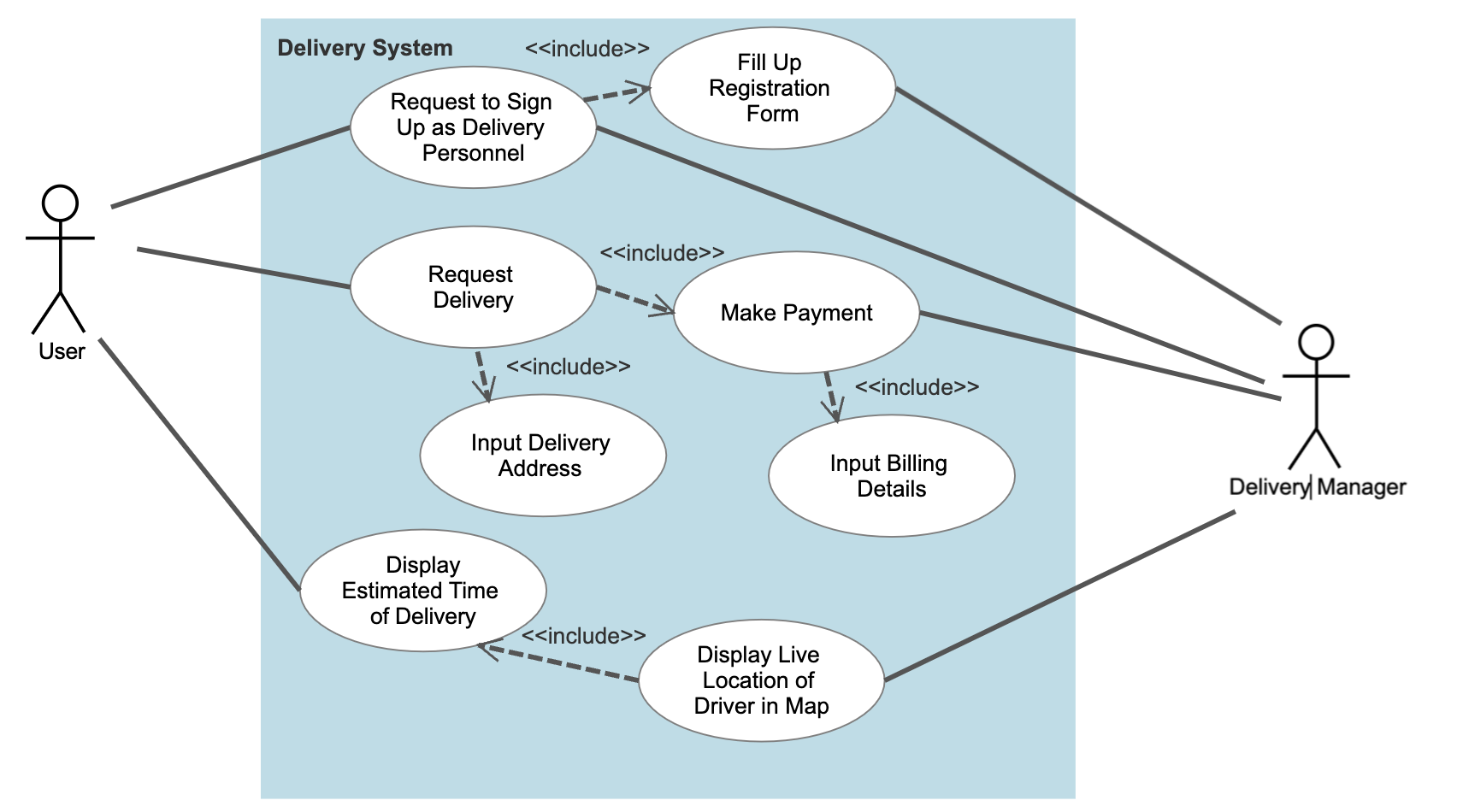
**REQ-2:**

**1/2** **System Feature 2 (and so on)**

****

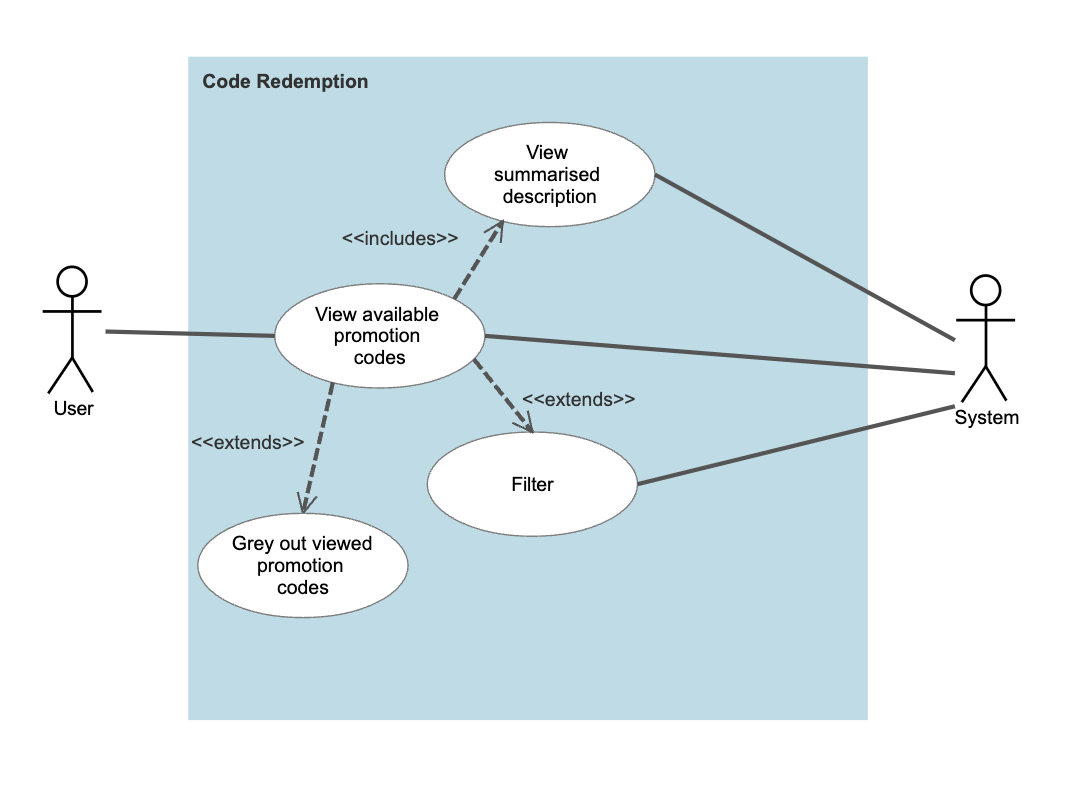
From Introduction to Software Engineering Design, 9.1 Introduction to Architectural Design, pg 257

# **Older Versions:**

****

| **Use Case ID:** | **­**3 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Delivery | | |
| **Created By:** | Jewel | **Last Updated By:** | Jewel |
| **Date Created:** | 1/9/22 | **Date Last Updated:** | 1/9/22 |

| **Actor:** | Delivery Manager |
| --- | --- |
| **Description:** | Immediate use of deals to order food at the comfort of their homes. Request and manage deliveries. |
| **Preconditions:** | User must be signed in.  User’s account must be authenticated. |
| **Postconditions:** | Delivery is made successfully if delivery is requested.  New delivery personnel in the delivery team. |
| **Priority:** | High |
| **Frequency of Use:** | 3 days a week |
| **Flow of Events:** | 1. User selects the request delivery button to use a food deal to order food. 2. User inputs delivery address. 3. Delivery Manager verifies the delivery address by checking if it exists. 4. User selects the online payment button. 5. User inputs billing details. 6. Delivery Manager verifies the billing details. 7. Order is sent through to the respective restaurant. 8. Once delivery has started, the live location of the delivery personnel is displayed on a map. 9. With the live location, estimated delivery time is displayed. |
| **Alternative Flows:** | AF-S1: Users can choose to sign up as a delivery personnel instead of requesting delivery.   1. Registration form is displayed for users to input their information. 2. Users will be redirected back to the delivery page once registration form is submitted.   AF-S3: Incorrect delivery address is provided.   1. Manager will prompt the user to re-enter their delivery address. 2. Return back to step 2.   AF-S6: Incorrect billing details are provided.   1. Manager will prompt the user to re-enter their billing details. 2. Return back to step 5. |
| **Exceptions:** | - |
| **Includes:** | - |
| **Special Requirements:** | 1. Update the live location in intervals of 1 minute. 2. Update the estimated delivery time in intervals of 1 minute. 3. Send notification to the user when delivery personnel are within 0.3km in 3 seconds. |
| **Assumptions:** | 1. Food companies are directly working together with the delivery manager. |
| **Notes and Issues:** | - |

****

| **Use Case ID:** | ­7 | | |
| --- | --- | --- | --- |
| **Use Case Name:** | Code Redemption | | |
| **Created By:** | Jewel | **Last Updated By:** | Jewel |
| **Date Created:** | 1/9/22 | **Date Last Updated:** | 1/9/22 |

| **Actor:** | System |
| --- | --- |
| **Description:** | Users can browse through available promotion codes. |
| **Preconditions:** | User must be signed in. |
| **Postconditions:** | Promotion codes that users have interacted with are greyed out. |
| **Priority:** | Medium |
| **Frequency of Use:** | Once a day |
| **Flow of Events:** | 1. User views a list of unfiltered available promotion codes. 2. User clicks on a promotion code. 3. User views a summarised description of the deal. 4. User clicks the Back button. 5. User is redirected back to the list of promotion codes. 6. The previously viewed promotion code is now greyed out. |
| **Alternative Flows:** | AF-S1: User is allowed to filter the available promotion codes. |
| **Exceptions:** | - |
| **Includes:** | Filter |
| **Special Requirements:** | 1. Greying out of the promotion code must be executed in 3 seconds. 2. Reset greyed out promotion codes weekly. 3. Filtered list of promotion codes must be reset in 5 seconds after the user exits the Code Redemption page. |
| **Assumptions:** | - |
| **Notes and Issues:** | - |

| **Use Case ID:** |  | | |
| --- | --- | --- | --- |
| **Use Case Name:** | **View Reviews** | | |
| **Created By:** | **Sean Lim** | **Last Updated By:** | **Sean Lim** |
| **Date Created:** | **1/9/22** | **Date Last Updated:** | **1/9/22** |

| **Actor:** | **User** |
| --- | --- |
| **Description:** | **View reviews made by other users. Users should be able to interact with the reviews.** |
| **Preconditions:** | **-** |
| **Postconditions:** | **Reviews seen should be filtered out with new reviews** |
| **Priority:** | **High** |
| **Frequency of Use:** | **5 days a week** |
| **Flow of Events:** | 1. **Users will click into the restaurant of their choice** 2. **Users will then be able to view the reviews of that restaurant made by other users** |
| **Alternative Flows:** | **AF-S1: Users can also view reviews at the home page**   1. **Users scrolling down the home page will view trending reviews recently made by other users.** |
| **Exceptions:** | **-** |
| **Includes:** | 1. **Like Review** 2. **Report Review** 3. **Share Review** |
| **Special Requirements:** |  |
| **Assumptions:** | 1. **Reviews are constantly made by users.** |
| **Notes and Issues:** | **-** |

| **Use Case ID:** |  | | |
| --- | --- | --- | --- |
| **Use Case Name:** | **Like Review** | | |
| **Created By:** | **Sean Lim** | **Last Updated By:** | **Sean Lim** |
| **Date Created:** | **1/9/2022** | **Date Last Updated:** | **1/9/2022** |

| **Actor:** | **User** |
| --- | --- |
| **Description:** | **User will be able to interact with the reviews made by other users in terms of “likes”. The more “likes” the review has, the algorithm will deem the post trending and it will more likely appear on other user’s home page.** |
| **Preconditions:** | **User must be signed in.**  **User account must be authenticated.** |
| **Postconditions:** | **Number of likes on a review will increase.** |
| **Priority:** | **High** |
| **Frequency of Use:** | **5 days a week** |
| **Flow of Events:** | 1. **Users will be able to view the reviews made by other users when looking at a restaurant and in the homepage** 2. **Users will be able to double tap the review and give the review a like.** 3. **The number of likes for that review will increase.** |
| **Alternative Flows:** | **AF-S1: The user is not signed in.**   1. **User double tap to like a review without being signed in.** 2. **Users will be redirected to the account login/ registration page.** |
| **Exceptions:** | **-** |
| **Includes:** | **-** |
| **Special Requirements:** | **-** |
| **Assumptions:** | **-** |
| **Notes and Issues:** | **-** |

| **Use Case ID:** |  | | |
| --- | --- | --- | --- |
| **Use Case Name:** | **Share Reviews** | | |
| **Created By:** | **Sean Lim** | **Last Updated By:** | **Sean Lim** |
| **Date Created:** | **1/9/2022** | **Date Last Updated:** | **1/9/2022** |

| **Actor:** | **User** |
| --- | --- |
| **Description:** | **User will be able to share the reviews with their friends through mainstream messaging applications or by copying the link.** |
| **Preconditions:** | **-** |
| **Postconditions:** | **-** |
| **Priority:** | **Medium** |
| **Frequency of Use:** | **Once a week** |
| **Flow of Events:** | 1. **Users will come across a review they would like to share to someone.** 2. **Users will be able to press a button open up the menu to display the options available for the user to share the review.** 3. **Users will choose and press the method of sharing.** 4. **Users will be redirected to share the review.** 5. **Users will also be able to get and copy the link to the review for sharing.** |
| **Alternative Flows:** | **-** |
| **Exceptions:** | **-** |
| **Includes:** | **-** |
| **Special Requirements:** | **-** |
| **Assumptions:** | **-** |
| **Notes and Issues:** | **-** |

| **Use Case ID:** |  | | |
| --- | --- | --- | --- |
| **Use Case Name:** | **Report Review** | | |
| **Created By:** | **Sean Lim** | **Last Updated By:** | **Sean Lim** |
| **Date Created:** | **1/9/2022** | **Date Last Updated:** | **1/9/2022** |

| **Actor:** | **User** |
| --- | --- |
| **Description:** | **User will able to report a post. The post may contain offensive or sensitive content. The post will then be reviewed and may be taken down if deemed offensive or sensitive.** |
| **Preconditions:** | **-** |
| **Postconditions:** | **The post may be taken down** |
| **Priority:** | **High** |
| **Frequency of Use:** | **One day a week** |
| **Flow of Events:** | 1. **User will come across a review that the user thinks has offensive or sensitive content.** 2. **User will press the 3 dots.** 3. **User will see a option to report the post** 4. **Upon tapping that option, the user will be prompted to write a brief description to why the user found the post offensive or sensitive.** 5. **The window will tell the user that the report has been submitted and thank the user.** |
| **Alternative Flows:** | **-** |
| **Exceptions:** | **-** |
| **Includes:** | **-** |
| **Special Requirements:** | **-** |
| **Assumptions:** | **There will be someone to review the reports and deem which reviews are indeed offensive or sensitive.** |
| **Notes and Issues:** | **-** |