

---

# **Software Requirements Specification**

**for**

## **Case Eaters**

**Version 1.0 approved**

**Prepared by Thomas Bentivoglio, Dominic Roberts, Angel Diaz, Sophia  
Kager, Kathryn Borkowski**

**Case Western Reserve University**

**September 12, 2025**

# Table of Contents

<b>1. Introduction</b>	<b>1</b>
1.1 Purpose	1
1.2 Vision Statement	1
1.3 Document Conventions	1
1.4 Intended Audience and Reading Suggestions	1
1.9 Project Scope	1
1.10 References	2
<b>2. Overall Description</b>	<b>2</b>
2.1 Product Perspective	2
2.2 Product Features	2
2.9 User Classes and Characteristics	3
2.10 Operating Environment	3
2.11 Design and Implementation Constraints	4
2.12 User Documentation	4
2.13 Assumptions and Dependencies	4
<b>3. System Features</b>	<b>4</b>
3.1 Report Food	4
3.2 Locate Food	5
3.3 Provide Meal Swipe	6
3.4 Receive Meal Swipe	8
3.5 Flag Post	9
3.6 Account Registration	9
3.7 Email Notifications	10
3.8 Account Settings	11
3.9 Login	12
<b>4. External Interface Requirements</b>	<b>12</b>
4.1 User Interfaces	12
4.2 Hardware Interfaces	14
4.5 Software Interfaces	15
9.1 Communications Interfaces	15
<b>10. Other Nonfunctional Requirements</b>	<b>15</b>
10.1 Performance Requirements	15
10.2 Safety Requirements	16
10.3 Security Requirements	16
10.4 Software Quality Attributes	16
<b>21. Other Requirements</b>	<b>17</b>

## **Revision History**

<b>Name</b>	<b>Date</b>	<b>Reason For Changes</b>	<b>Version</b>

## 1. **Introduction**

### 1.1 **Purpose**

This SRS describes the software functional and non-functional requirements for release 1.0 of Case Eaters. This document is intended to be used by the members of the project team who will implement and verify the correct functioning of the system. Unless otherwise noted, all requirements specified here are high priority and committed for release 1.0.

### 1.2 **Vision Statement**

For the people around the CWRU campus looking for a meal or something to eat, Case Eaters will be a web-based application that will take user reports and meal swipes, displaying them for other users in need. This will be through a combined map and list view that will display current free food or available meal swipes. With this system, we can eliminate food waste around campus and maximize meal plans by sharing unwanted or leftover swipes with other users.

### 1.3 **Document Conventions**

No typographical conventions were applied in this SRS.

If a requirement is designated as high priority, its sub-requirements are also considered to be high priority as every function of a requirement is integral to fulfilling the parent requirement.

### 1.4 **Intended Audience and Reading Suggestions**

1.5 **Developers:** to understand the functional and technical requirements.

1.6 **Project Managers and TAs:** to track scope and ensure timely delivery.

1.7 **Testers/QA Engineers:** to derive test cases and validation strategies.

1.8 **End Users (students):** for reference on planned functionality.

### 1.9 **Project Scope**

A platform for users to report and locate free food. Users can input locations and food types. These can be accessed via both a list of posts or as pins on a live map. Posts have interactive features, and

map locations will be linked to the posts. The food can be filtered by location and type to help users connect with their most desired, closest free food available. Additionally, the app can connect college student users who have extra meal swipes with those who do not, maximizing meal plan value. Users are incentivized to participate with a loyalty system.

## 1.10 References

No references in this version.

# 2. Overall Description

## 2.1 Product Perspective

There is no formalized system at CWRU for broadcasting free food availability or connecting students with extra meal swipes to those not on the meal plan. Communication occurs informally through group chats, word of mouth, or various message forums, which is inefficient, unreliable, and not well-known. *Case Eaters* aim to fix this through:

- Providing a real-time reporting system for food availability.
- Allowing students to offer or claim meal swipes.
- Presenting food events in both map and list views.
- Establishing a secure and scalable web application accessible on both mobile and desktop browsers.

## 2.2 Product Features

At a high level, Case Eaters will support the following features:

2.3 **Food Reporting:** Post food availability with location, description, and type.

2.4 **Food Discovery:** Browse food opportunities via a map (pins) or a sortable list.

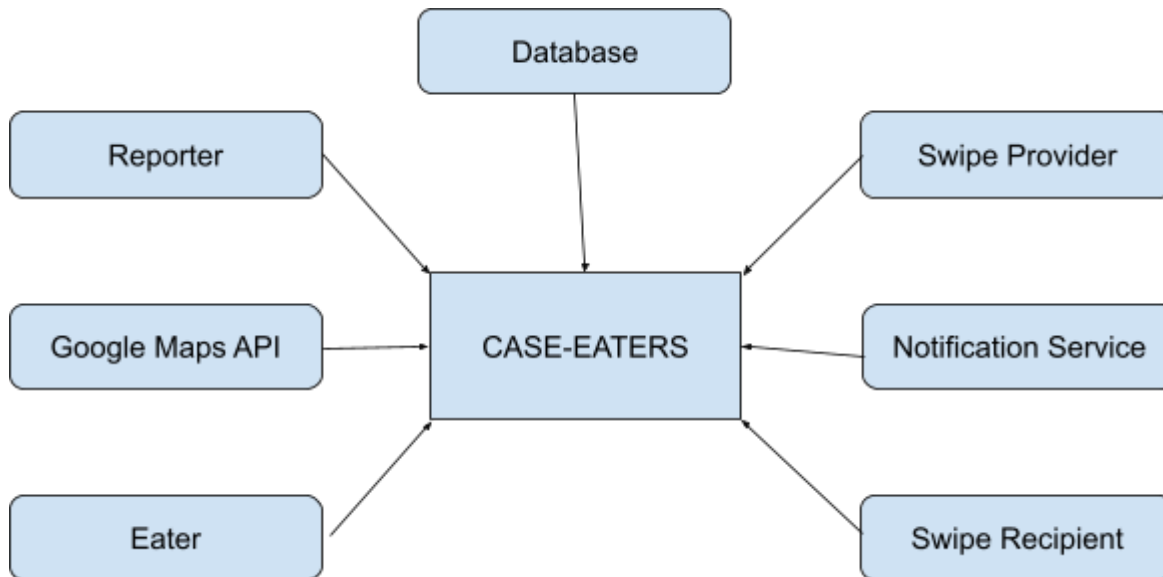
2.5 **Meal Swipe Exchange:** Allow students to offer or claim swipes, with built-in communication support.

2.6 **Reputation and Reporting System:** Users can upvote helpful posts and report inappropriate content.

2.7 **Loyalty System:** Incentives for consistent use, such as badges or points.

**2.8 Profile Management:** Users can toggle between roles (reporter, eater, swipe provider, swipe recipient).

## 2.9 User Classes and Characteristics



There is one user class and 4 roles in Case Eaters: Reporter, Eater, Swipe Provider, and Swipe Recipient. Every user can act in any role. Users can toggle (in their user settings) between being a Swipe provider or not.

Briefly, the roles are as follows:

**Reporter:** User identifies free food (eg, catering leftovers, student gatherings, club meetings, etc). Reporters post these opportunities to Case Eaters with a description of the food and a location.

**Eater:** User uses either a map or a list view to view reported food sightings.

**Swipe Provider:** User creates a post with their location, available swipe, and contact information

**Swipe Recipient:** User uses either a map or a list view to view reported swipes available. Once a swipe is selected, they use provided contact information and location to connect with user external to Case Eaters Platform.

## 2.10 Operating Environment

OE-1: Case Eaters shall operate on the latest versions of Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

OE-2: Case Eaters shall operate on a cloud-hosted server using HTTPS.

OE-3: Case Eaters shall permit user access from both desktop and mobile browsers.

## **2.11 Design and Implementation Constraints**

CO-1: The system will utilize Google Maps API for location and pin services.

CO-2: The system shall abide by CWRU IT security policies for privacy.

CO-3: The system shall use Firebase (or equivalent cloud database) for authentication and data storage.

CO-4: All web code shall conform to HTML5, CSS3, and JavaScript/React standards.

## **2.12 User Documentation**

UD-1: The system shall provide an online tutorial for each role (reporter, eater, swipe provider, swipe recipient).

UD-2: The system shall provide a FAQ/help section accessible from every page.

UD-3: Tutorials shall adapt based on user role selection (e.g., only meal-plan users see swipe provider tutorial).

## **2.13 Assumptions and Dependencies**

AS-1: Users will provide accurate reports and self-moderate by flagging false or malicious posts.

AS-2: Users will have access to Wi-Fi or cellular networks.

DE-1: The operation of Case Eaters depends on third-party APIs (Google Maps, Firebase Authentication).

DE-2: Push notifications and email alerts depend on Gmail API or equivalent.

### 3. System Features

#### 3.1 Report Food

##### 3.1.1 Description and Priority

Users will report publicly available free food. This report will require a location and description of food available. There will be an option to add additional information in the form of free text. This is a high priority feature, as it is essential for the functionality and utility of the application.

##### 3.1.2 Stimulus/Response Sequences

Stimulus: User gains knowledge of free food via flyer/people/clubs/around campus and selects the report button on the application.

Response: System generates a page for the user to fill out details of the food, including location, description, and any additional info.

Stimulus: User submits the required information for the food.

Response: If the system accepts the input, then it gets added to the list of available food and the map of available food. This will be tagged as free food.

##### 3.1.3 Functional Requirements

Report.Add	The system will allow a user who is logged into Case Eaters to add an instance of free food
Report.Add.Description	The system will require the user to enter a description of the type of food available
Report.Add.Location	The system will require the user to enter a location compatible with Google Maps API to geotag the food
Report.Add.Additional	The system will prompt the user to add any additional information as plain text, they can opt out
Report.Add.Date	The system will save the starting date and time of the report
Report.PostList	The system will push the report information to a post as a list
Report.PostMap	The system will push the report information to the map view
Report.Remove	If X time has elapsed or if the user manually removes a report, the report is deleted from the system



## 3.2 Locate Food

### 3.2.1 Description and Priority

Locate Food allows users to see food that has been reported. There will be a map with pins indicating where food is. A toggleable list will be on the side for users to scroll through as well. They will have access to the location of the reports and the description of the reports. This is a high-priority feature, and integral to the purpose of our application as it is the function connecting food and people that want said food.

### 3.2.2 Stimulus/Response Sequences

Stimulus: User hovers over a report on the map.

Response: Report reveals the type of food and location.

Stimulus: User toggles list view on.

Response: List view UI appears next to the map, with

Stimulus: User hovers over a report on the list view.

Response: Report is highlighted on the map.

Stimulus: User clicks on a report.

Response: A new UI element appears, offering more information about the food posting. This includes type of food, location, and description.

### 3.2.3 Functional Requirements

Report.Map.Mouseover	User hovers over a report on the map, providing them with the details of the food
Map.ListToggle	User selects a button which unhides list view
Report.List.Mouseover	User hovers over a report on the list, highlighting it
Report.Select	User clicks on a report, either on map or list view, which opens a new UI window with all details of report. They can use this information to reach the food

### 3.3 Provide Meal Swipe

#### 3.3.1 Description and Priority

Users are prompted at registration to report whether or not they are on the meal plan. If so, Provide Meal Swipe allows them to post a meal swipe for sharing on the map and list view. This will appear with the same interface as Report Food posts, but with a special designation that it is for a meal swipe. This is a high priority feature.

To report free food, users will open a “Provide Swipe” menu, similar to Reporting food, but this will have a predefined list of options for food type based on meal plan options. The location and description will be preset based on the option selected.

#### 3.3.2 Stimulus/Response Sequences

Stimulus: User decides to provide a meal swipe and selects “provide meal swipe” button in app

Response: Application opens a menu with a predefined list of meal swipe options for the user to select from and a plaintext description field.

Stimulus: User selects the meal swipe option they wish to provide

Response: Application generates a map/list post with presets chosen based on the user selected meal swipe option. This post will be tagged as a meal swipe.

#### 3.3.3 Functional Requirements

SwipeReport.Add	The system will allow a user who is logged into Case Eaters to add an instance of a meal swipe. This is a separate (but similar) system to Reporting food
SwipeReport.Add.Location	The system will require the user to select the location from a predefined list of meal swipe options
SwipeReport.Add.Additional	The system will prompt the user to enter optional additional details
SwipeReport.Add.CaseID	The system will record the caseID of the user who posts
SwipeReport.Add.Date	The system will record the date and time the swipe was added
SwipeReport.PostList	The system will push the report information to a post as a list
SwipeReport.PostMap	The system will push the report information

SwipeReport.Add	The system will allow a user who is logged into Case Eaters to add an instance of a meal swipe. This is a separate (but similar) system to Reporting food
	to the map view
SwipeReport.Remove	If X time has elapsed or if the user manually removes a report, the report is deleted from the system

### 3.4 Receive Meal Swipe

#### 3.4.1 Description and Priority

User selects a swipe provider post and is provided information on the type, location, and description of the swipe. The user can then select to get the provider's case email. If they do, the user gets locked from engaging with other swipe posts for 10 minutes. The user can then contact the provider through their email via google chat, email, etc. This is a high-priority feature.

#### 3.4.2 Stimulus/Response Sequences

Stimulus: User clicks a meal swipe on the locate food page.

Response: Meal swipe provides details on type, location, and description of swipe.

Stimulus: User clicks “contact provider”.

Response: The user is directed to the contact information of the swipe provider. The user is locked out of clicking “contact provider” on any other meal swipe for 10 minutes.

#### 3.4.3 Functional Requirements

SwipeReport.Map.Mouseover	User hovers over a provided swipe on the map, providing them with the details of the food
Map.ListToggle	User selects a button which unhides list view
SwipeReport.List.Mouseover	User hovers over a provided swipe on the list, highlighting it
SwipeReport.Select	User clicks on a provided swipe, either on map or list view, which opens a new UI window with all details of provided swipe. They can use

	this information to reach the food.
SwipeReport.Contact	Contact information of the user is revealed on details of swipe UI window.
SwipeReport.Contact.Block	A 10 minute block on a user receiving contact information for other swipes is enabled

### 3.5 Flag Post

#### 3.5.1 Description and Priority

Users can flag a post to identify an issue with the post. This will include no more food, any inaccurate information, and inappropriate posts.

#### 3.5.2 Stimulus/Response Sequences

Stimulus: User clicks on a post, then hits the flag button in the expanded post UI.

Response: An expanded flag UI appears with a dropdown prompting the user to choose a flag(s). The flag options will include: no more food, inaccurate information, inappropriate post.

Stimulus: User clicks a flag type.

Response: A flag is reported. Once 5 flags are reported the post is taken down.

#### 3.5.3 Functional Requirements

Report.Add.Flag	User clicks the flag button.
Report.Add.Flag.IPC	Flag Inappropriate Post
Report.Add.Flag.	Flag
Report.Add.Flag.	Flag
Report.Add.Finished	Button that adds 1 to counter for how many people reported food gone

### 3.6 Account Registration

#### 3.6.1 Description and Priority

Account registration is prompted when the user first enters the application. Required information includes: CASE ID, case email, name, meal plan y/n (toggle-able after registration), and notification settings. This is high priority.

### 3.6.2 Stimulus/Response Sequences

Stimulus: User opens the application.

Response: Application prompts user to enter personal information and preferences.

Stimulus: User enters information and then clicks “create account”.

Response: A new account is created with said information.

### 3.6.3 Functional Requirements

Account.Add	The system will prompt the user to create an account to add to the system
Account.Add.Email	The system requires the user to enter a valid CWRU email.
Account.Add.Password	The system requires the user to enter a valid password.
Account.Add.FirstName	The system requires the user to enter their first name.
Account.Add.LastName	The system requires the user to enter their first name.
Account.Add.MealPlan	The system requires the user to report if they’re currently on the meal plan with a “YES” or “NO” toggle.
Account.Add.Notifications	The system requires the user to report if they want to opt in on email notifications with a “YES” or “NO” toggle.

## 3.7 Email Notifications

### 3.7.1 Description and Priority

Email notifications are sent out at set times during the day (TBD). This allows for users to be reminded that there is free food available, especially around meal times. Low priority feature.

### 3.7.2 Stimulus/Response Sequences

Stimulus: It hits a pre-defined time of day (ex. 1:00pm)

Response: An email is sent out to all users who subscribe to email notifications.

### 3.7.3 Functional Requirements

Reports.Get	Get a list of all active free food reports
SwipeReports.Get	Get a list of all active free swipe reports
Email.Push	Composes an email from a predefined format

## 3.8 Account Settings

### 3.8.1 Description and Priority

Users can change their account settings after they create their account. This includes food preferences, meal plan y/n toggle, and notification settings. This is a high priority.

### 3.8.2 Stimulus/Response Sequences

Stimulus: User enters settings page and changes a setting.

Response: Settings will update and the application will adjust their experience accordingly.  
(ex. User changes meal plan y/n toggle from y to n, the user will no longer be able to provide swipes, and therefore will no longer see a button to do so on their UI)

### 3.8.3 Functional Requirements

Account.Settings.IDInfo	System displays identification info of account (Email, Password, FirstName, LastName)
Account.Settings.ChangePassword	System has a button that allows user to change password. Prompts a small interface to pop up and enter in new password twice.
Account.Settings.MealPlan	System has a toggle of “YES” or “NO” that can be changed.
Account.Settings.Notifications	System has a toggle of “YES” or “NO” that controls if they want notifications.
Account.Settings.Save	System saves new settings once user confirms with a button press.

### 3.9 Login

#### 3.9.1 Description and Priority

Users will be required to go through a login gateway to use Case Eaters. This will require them to provide their Case ID (a unique identifier), and their password.

#### 3.9.2 Stimulus/Response Sequences

Stimulus: User opens the app

Response: System prompts user to input username and password

Stimulus: User inputs correct username and password, then hits enter

Response: The system checks the database to find the UN/PW combination and if found, allows the user into their account

Stimulus: User inputs correct username and incorrect password, then hits enter

Response: The system checks the database to find the UN/PW combination. Since the UN is found, we do not let the user in and add a “forgot password” option

Stimulus: Selects forgot password

Response: System sends an email to the user to reset password.

#### 3.9.3 Functional Requirements

## 4. External Interface Requirements

### 4.1 User Interfaces

**Home Page:** The home page will have a user login prompt with a forgot password and create account button.

**Map/List Page:** The map page has pins that locate each free food location. The list page is a list representation of the map page. Users can use sorting settings to sort by distance and type of food. There are buttons for adding a report and adding a swipe (only if users selected yes for meal plan toggle)

**Expanded Free Food Post:** Users can click on the pins to access more information, such as Description, Location, Type of Food, Reporter, Time of Report.

**Profile/Settings Page:** The profile page allows the users to access and change their user settings.

**Add Free Food Post:** Allows users to input information about the food they are reporting such as title, location, and description.

**Add Swipe:** Allows users to input information about the swipe they are providing such as swipe type, in person (y/n), and description.

Login Page:

The Login Page UI mockup features a central heading "Case Eaters" above a "Login" sub-heading. Below these are two input fields: "Username:" and "Password:". A "Forgot Password" button is positioned below the password field. At the bottom of the form is a "Create Account" link.

Map/List Page:

The Map/List Page UI mockup is divided into two main sections. The left section, titled "Food Postings", lists "Example Post 1" and "Example Post 2", each with "Example location" and "Example Description". The right section is a map view showing blue location pins. At the bottom right of the map view are two circular buttons: one with a plus sign labeled "Add Report" and one with a square icon labeled "Add Swipe".




## Account Settings:

Email:	<input type="text" value="(example@example.com)"/>
Case ID:	<input type="text" value="example123"/>
Password:	<input type="text" value="Change Password"/>
Notifications:	<input type="text" value="Notification Settings"/> ▾

## Expanded Free Food Post:

← Back

**Free Food Post Title**

 Flag Post

## Add Swipe:

← Back

Swipe Type:

In-Person: Yes ☐ No ☐

Description:

## Add Free Food Post:

← Back

Title:

Location:

Description:

## 4.2 Hardware Interfaces

### HI-1: Web Server Interface

**HI-1.1:** The COS shall operate on a dedicated web server.

**HI-1.2:** The web server shall provide connectivity to client devices through HTTPS.

**HI-1.3:** The web server shall support standard TCP/IP networking.

**HI-1.4:** The web server shall log all requests and responses for audit and debugging purposes.

## **HI-2: Patron Devices (Mobile and Desktop)**

**HI-2.1:** The COS shall support interaction with patron devices through standard browsers.

**HI-2.2:** Patron devices shall communicate with the web server via secure wireless (Wi-Fi, LTE/5G) or wired Ethernet.

**HI-2.3:** The COS shall render interfaces that adapt to mobile, tablet, and desktop displays.

## **4.3 Software Interfaces**

**SI-3.1:** The COS shall request location data from the Map API to display campus maps and food locations.

**SI-3.2:** The COS shall transmit coordinates (latitude/longitude) of events to the Map API to place pins.

**SI-3.3:** The COS shall receive rendered map tiles and location metadata from the Map API for display in the user interface.

**SI-3.4:** The COS shall use the Map API to calculate distances between the patron's location and food events.

**SI-3.5:** The COS shall update event locations in real time by polling or subscribing to Map API services.

**SI-3.6:** The COS shall communicate with a third-party Map API through a programmatic interface to support location services.

## **4.4 Communications Interfaces**

**CI-1:** The application requires email and web browser communication interfaces.

**CI-2:** The application will use HTTP protocols.

**CI-3:** The application shall send 3 daily e-mail messages (8am, 1pm, 6pm) to users identifying the locations of free food available.

**CI-4:** The application shall allow eaters to exit the app and message reporters to find meal swipes.

## 5. Other Nonfunctional Requirements

### 5.1 Performance Requirements

**PE-1:** The system should accommodate 400 users during the peak usage time window of 10:00am to 4:00pm.

**PE-2:** The system should be running 95% of the time.

**PE-3:** Responses to queries should take no longer than 7 seconds to load onto the screen after the user submits a query.

### 5.2 Safety Requirements

**SR-1:** Harmful or irrelevant posts will be filtered.

### 5.3 Security Requirements

**SE-1:** Must sign in with case email a valid case email, will check that email entered for registry is valid.

- To prevent unidentified users

**SE-2:** No Admin to prevent injection attacks, as no account has any admin privileges

- Users with repeated flags can be manually removed from the database.

### 5.4 Software Quality Attributes

The following quality attributes are prioritized for the system:

## 6. Usability

- 6.1 The interface must be intuitive and require less than 5 minutes of onboarding for new users.
- 6.2 Ease of use is prioritized over ease of learning.

## 7. Availability

- 7.1 The system must be available 99.5% of the time during the semester (excluding scheduled maintenance).

## 8. Reliability

- 8.1 Failure rate must not exceed 1 in 1,000 user operations.

- 8.2 Recovery from minor faults (e.g., lost network connection) must occur automatically within 30 seconds.

## 9. Performance

- 9.1 Average page load or response time must be under 2 seconds for 95% of requests.

## 10. Scalability

- 10.1 The system must support at least 1,000 concurrent users without performance degradation beyond the above response time target.

## 11. Security

- 11.1 User sessions must expire after 30 minutes of inactivity.
- 11.2 All user data must be encrypted in transit (TLS 1.3 or higher).

## 12. Maintainability

- 12.1 The system code must be modular with less than 20% code duplication.
- 12.2 Updates and bug fixes must be deployable within 1 hour of release.

## 13. Interoperability

- 13.1 The application must integrate with existing university authentication systems (e.g., Single Sign-On).
- 13.2 Data must be exportable in CSV and JSON formats.

## 14. Portability

- 14.1 The system must operate on modern browsers (Chrome, Firefox, Safari, Edge) and mobile devices (iOS and Android).

## 15. Testability

- 15.1 Automated unit and integration tests must cover at least 80% of the codebase.

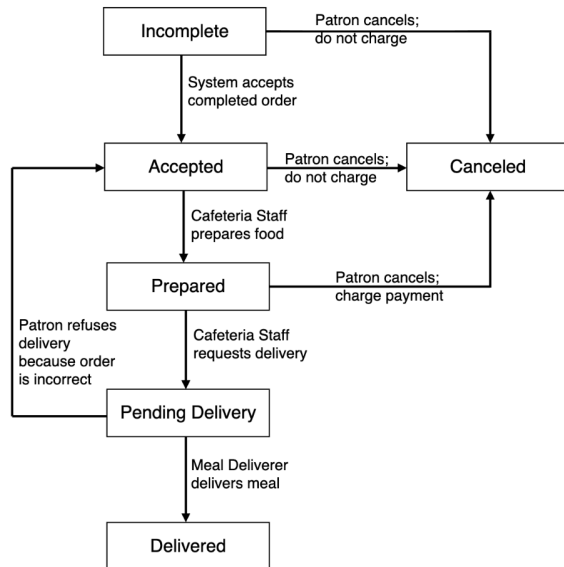
## 16. Other Requirements

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

## Appendix A: Glossary

Food Post	=	food type
	+	description of food
	+	location of food
	+	reporter
	+	report time
Food type	=	*type of food which has been posted (Restaurant, Cuisine, etc)*
Description of Food	=	*description of food beyond location and type*
Location of Food	=	*location of food on campus*
Reporter	=	*user who reports food*
Report time	=	*time at which food was reported xx:xx on MM/DD/YY *
Swipe Post	=	type of swipe
	+	location of swipe
	+	in-person information
	+	description of swipe
Swipe	=	*meal swipe through CWRU*
Type of Swipe	=	*variation of swipe offered through CWRU meal plan*
Location of Swipe	=	*Location at which the swipe receiver will need to pick up food*
In-person or if it Information	=	*information that indicates whether or not the swipe can be ordered digitally must be ordered in-person*
Description of Swipe	=	*description of swipe beyond type, location, and in-person information – can include extra information necessary to receive swipe*
Receiver	=	*user who receives a swipe*
Provider	=	*user who provides a swipe*

## Appendix B: Analysis Models



## Appendix C: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>