Document Name: Release Summary
Product Name: Restaurant Ordering System
Team Name: ROS

Key Completed User Stories and Acceptance Criteria:

Example:
User Story 1.1: As a user, I want to be able to log in to my account so that I can make my posts from my account.
Acceptance Criteria 1.1: I am able to use an email and password, that has been registered, to login past the login page. The login page displays errors, if the user inputs bad information.

User Story 1.2 - As a user, I want to create a new account, so if I do not have an account, I can make one.
Acceptance Criteria 1.2: We are able to set up an account with an email, username, and password. After creating an account, we should be able to log in using the credentials used for the signup.

User story 1.3: “As a customer, I want to add/remove items to/from my cart so that I can create an order.”
Acceptance Criteria 1.3: We are to add or remove items from the cart using the add to cart and remove buttons available in the menu page.

User story 1.4: “As a customer, I want to experience an intuitive user interface.”
Acceptance Criteria 1.4: We have an intuitive interface that is user-friendly.

User story 2.2: "As an admin, I want to view incoming or status of orders so that I can monitor, and confirm orders."
Acceptance Criteria 2.2: We are able to view the status of orders and monitor their status as well as confirm when the order is completed.

User story 3.1: "As a customer, I want to be able to submit orders online, and receive periodic updates.”
Acceptance Criteria 3.1: We are able to submit orders online and receive an update from the kitchen system through the server when the order is complete.

Known Problems:

Connection longevity issues:

- The websocket connection between the server and the client is very unstable.
    - WebSocketConnections are not preserved on refresh (Intended behavior)
    - WebSocketConnections are not preserved on page jumps (Unintended behavior)
        - Example: Going from main to either menu or admin-menu, or vice versa

- If a master system is on the admin-dashboard page and a client sends an order to the server while the server has no previous orders in either waiting orders, working orders, or finished, the admin page will crash due to the websocket not being able to respond fast enough.
    - This is because there was a design flaw in the system. The server loads the order status arrays on load. This means that before an order is placed and there were no previous orders, the order status page grabs the websocketservice order status arrays before they are initialized. This could be solved with a synchronization marshal that would be implemented on the WebSocketService component. There was not enough time.

Shopping cart/menu item state update issues:

- If an item is put into the shopping cart by a customer and the server executes a DELETE/EDIT item order that modifies an item in the shopping cart, the item in the shopping cart is unaffected. This was initially intended design in regards to the shopping cart price. The whole point of this system was to be able to provide live-update prices on menus. This system allowed the user to "lock-in" an item's price, in case it went up. (Realistically speaking however it is apparent that this is not realistic". This soon turned into an issue however when other fields, such as name, description, image, or availability were edited. Therefore an item could be added into the shopping cart, the admin could remove that item from the menu, and the customer could send the item to the server.

"Send order" button in shopping cart button on the landing page acts as a self-destruct button

- The shopping cart on the landing page should not be there, but is. The issue is that the shopping cart component present in the navbar on the landing page is the same navbar that is used in the customer-menu page. This component has dependencies that are contained in the customer menu page, which is not available on the landing page.

Images are implemented using PATH instead of a url or draggable image upload

- Time constraints did not allow us to implement a proper image feature that would allow the admin to upload images. All of the images are stored locally in the repository. Ideally, the images would be hosted on the database, and the server would also pull the images from the database and distribute them to the slave systems on the initial connection.

Perpetual loading when hopping between paths

- A system design flaw is causing these issues, which is now only apparent after the implementation and testing. The WebSocketService attempts a connection to the server when the page is loaded. There are two total web socket services: One on the admin page and one on the customer page. If the webpage is redirected from an admin page to the customer page, or vice versa, the WebSocketService connection from the former page stays life, and the websocketservice connection for the latter page is never spawned. This is an issue because the websocketservice on the admin page connects to the server and requests elevated privileges that allow the system to manage the order statuses as well as the order items. These websocketservices also have different functionalities that are dependent on components present in their respective pages, further leading to undetermined and often incorrect behavior.

Bandwidth issues that scale with Scalability (Not as big of an issue, definitely out of scope)

- The system design is essentially a centralized server, different from the initial plans of creating a multi-master/slave system. This however creates a single chokepoint at the server. The server will start to lag when sending and receiving from multiple websockets simultaneously. Currently, the system has been tested with 5 customer systems and 5 kitchen systems. During testing, a script was written that had all 5 customer systems to rapid-fire about 10 orders within a second. The server would then receive 50 orders all at once. This created concurrency issues, specifically data race. This led to non-deterministic behavior when tested under unrealistic load situations.

NOTE: All issues are fixed with a refresh and not repeating the series of actions that led to the issue :)

Product Backlog:

- **Didn't Complete**
  - Menu Customization
  - Analytics Display
  - Temporary Filter Bar Visibility
  - Sending Updates to Customer Systems
  - Customer Authentication