User Stories

- As a Wismer cook I need to…
    - See what orders have been placed in real time
        - Communication between kiosk and several other displays
        - Each order is displayed with its order id number
        - Only send relevant orders to each station
    - Close orders once filled
        - Each station has its own interface that allows a user to close an order
    - Be confident that the system will not crash
        - Efficient design (Database and command processes)
        - Stability
    - Communicate to customers that certain items are unavailable
        - Schedule certain items/stations to be unavailable at certain times/dates
        - Quickly and easily change an item to Sold Out
            - Able to do this by ingredient, ie at Tres: tortillas, ground beef
- As an Ursinus Administrator I need to…
    - Show off some really cool software to prospective students
    - Receive information on sales
    - Be confident that orders made at the kiosk are deducted from student account balances
        - System consistently makes transactions without issue
        - reports errors if they occur
- As a Wismer Customer, I need to…

- See what food items are available
    - I need to know what is begin offered
    - I need to know if something is sold out/unavailable
        - Restrict shown items to the time of day (ie lunch items not shown before scheduled lunch service)
- Be able to build an order containing multiple items
    - Typically ordering a meal, not just a single item at a time
- Customize items with options
    - After selecting an item, further menus with customization options will appear
    - These options should be logically organized
    - If any option has an additional charge, that is listed with the option
- Pay for orders using credit, debit, or bear bucks/dining
    - If using dining dollars, pay for the remaining balance
    - Card Reader for credit/debit
    - Ursinus Card Swipe for bear bucks/dining/meal swipes
        - Accurately deduct $7.50 per meal swipe
    - Transactions are secure
- Order and pay quickly
    - Reduce wait times around peak hours
    - System should be stable
        - Properly display images on the menu interface, rather than blank
    - System should function promptly

        - Needs to print out receipt covering entire order and order number
    - Cancel an order
- As a Sodexo Official I need to…
    - Trust the system will work consistently
    - Trust the system is secure

ASPIRANT:

- Save a few Favorite orders (at checkout?)
    - Be able to autofill an order to a saved Favorite
- Be reminded about side options like fries or drinks if they are not in my order when checking out
- Make a mobile order on a phone via the app
    - Phone UI
    - Secure communication for transactions
- Kiosk recommends your most frequent order for express checkout
- Meal analytics for Wismer staff
    - Information on popular items and those not selling well
    - Which time of day items are most popular
    - Which time of day have to most/least traffic overall
- Breakfast, lunch, and dinner options given the time of day

Formal Requirements

Functional Requirements

Database: (1)

- Add entry (1.1)
    - 1 Entry per food item
    - Price
    - Availability
    - Image
    - Options/add-ons
    - Location (which station)
        - Category (within location, ie sides, entrees, etc)
- Remove Entry (1.2)
    - Delete items from database that are no longer available
- Modify Entry (1.3)
    - Change price
    - Change availability
- Search/Query (1.4)
    - Search by item categories; price, location, etc.
- Invoice Send(1.5)
    - Send invoice of all orders within the day to be printed
- Microservice to access database (1.6)

Customer UI: (2)

- Categorical Sorting (2.1)
  - All items will be sorted according to specific categories that will make the system efficient for the customer
    - Based on database setup
- Present Items (2.2)
  - Items will be presented through the GUI after selecting category
    - This includes price values and other details about the product
- Items have images of themselves displayed for the user (2.3)
- After selecting an item, presented with customization options if applicable (2.4)
- Options are displayed with their prices if applicable (2.5)
- Confirm adding an item (2.6)

Cook UI: (3)

- Real-time order display (3.1)
  - View only orders for user's station
  - Timestamps? Orders in order
- Remove fulfilled orders (3.2)
  - Remove fulfilled orders from cook UI to reduce clutter

Order Processing: (4)

- Order send (4.1)
  - Complete order is sent to the employee system to be fulfilled
  - Complete order is also sent to the database

- Order association (4.2)
  - Orders will be given an id to keep them in order
  - ID is printed on receipt
  - ID is sent with the order to cooks
- Order categorization (4.3)
  - Orders are categorized by which station they're sent to so that each station receives only orders meant for it
- Order encryption (4.4)
  - Orders have some form of encryption to prevent fraudulent orders from being received as actual ones (signature - kiosk uses a private key, verified on the server by its matching public key)
  - (only relevant if there is not a direct connection from the kiosks to the stations, ie if they use wifi)
- Order decryption (4.5)
  - Ensure received orders are sent from the kiosks
  - (only relevant if there is not a direct connection from the kiosks to the stations)

Transaction Processing: (5)

- Card reader (5.1)
  - Read debit, credit cards
  - Outside hardware attached to the kiosks
- Ursinus card swipe (5.2)
  - Built-in swipe thing for reading Ursinus ID cards
- Processing (5.3)

    - Charge debit, credit cards
    - Charge from student ID bear bucks, dining dollar accounts
    - Charge meal swipe from student ID for $7.50 discount
    - Transactions are secured
    - Information is deleted after transaction
- Transaction math (5.4)
    - System will calculate final total for a finished order
    - System will display final totals before the order is paid for

Non-Functional Requirements

- Better Hardware to run the system efficiently
- Distributed workload

System Architecture

- Database
- Customer UI
- Cook UI
- Payment processing?
    - Assume there is a service in existence for credit/debit cards that we can send to
    - Create a service to interface with dining/bearbucks