WORKSHOP #6 - VENDING MACHINE

# Group 4 Members:

* **Luca Novello - Logic 3 - flowchart**
* **Joshua Civello - Logic1 - flowchart**
* **Seulgi Lee - Logic 2 - Pseudo code**
* **Natsita - Logic 1 - Pseudo code**

# Workshop Overview

Vending machines now almost run without any human maintenance. They employ the“internet of things” (IoT) enabling real-time updates on stock levels, payments, and alerts for machine maintenance. Electronic payments use both swipe and “tap” technology for debit and credit cards and even accept cell phone payments using “near field communications” (NFC). No physical money is stored! Maintenance costs are drastically reduced with these enhancements over the older models as routine inventory and money pickups are eliminated. The only time a service provider needs to physically visit the machine, is for restocking inventory and addressing any general mechanical maintenance(which would be infrequent). But how should this all work?

# HARDWARE STATES

Main Process Product Structure:

• slotID = Unique location slot ID (physical placement in the machine ex: “D8”)

• sku = Unique product identifier

• quantity = Actual quantity available (physically in machine at the given slotID)

• maxQuantity = Maximum machine qty that can be stocked for the slotID

• minQuantity = Re-order when this qty is reached (based on: maxQuantity - quantity)

• price = Vending machine price to charge customer per unit

• description = Product name

Transaction Structure:

• slotID = Unique location slot ID (physical placement in the machine ex: “D8”)

• quantity = Requested quantity

• price = Price per single unit quantity

• description = Product name

SUBPROCESS: IDLE STATE

1. START

2. LCD Screen: Active

3. ENTER button: Disabled

4. COLUMN/ROW buttons: Disabled

5. CORRECTION button: Disabled

6. CANCEL button: Disabled

7. PAYMENT MODULE: Disabled

8. END

SUBPROCESS: ACTIVE STATE

1. START

2. LCD Screen Active

3. COLUMN/ROW buttons - Enabled

4. ENTER button - Enabled

5. CORRECTION button - Enabled

6. CANCEL button - Enabled

7. PAYMENT MODULE: Disabled

8. END

SUBPROCESS: PAYMENT STATE

1. START

2. LCD Screen Active

3. COLUMN/ROW buttons - Disabled

4. ENTER button - Disabled

5. CORRECTION button - Disabled

6. CANCEL button- Enabled

7. PAYMENT MODULE: Enabled

8. END

SUBPROCESS: CANCEL STATE

1. START

2. LCD Screen Active

3. Column/Row Buttons - Disabled

4. ENTER button - Disabled

5. CORRECTION button - Disabled

6. CANCEL button - Enabled

7. PAYMENT MODULE: Disabled

8. END

# 

# PSEUDOCODE

## Overall Pseudocode:

1. Start
2. Vending machine powers on
3. Are there any power interruptions?
   1. Yes: Shutdown Machine: End
   2. No: continue: step 4
4. Enter IDLE STATE
5. Is the user ready to order?
   1. Yes: exit IDLE STATE: continue: step 6
   2. No: continue: step 4
6. Enter ACTIVE STATE
7. Is the user ready to pay?
   1. Yes: exit ACTIVE STATE: continue: step 8
   2. No: continue: step 7
8. Enter PAYMENT STATE
9. Is the user done paying?
   1. Yes: exit PAYMENT STATE: continue: step 4
   2. No: continue: step 9

## 

## 

## Logic 1 - Pseudocode:

1. Start
2. Define vending machine states: Idle, Active, Payment, Cancel
3. Define hardware components: screen, column buttons, payment module, enter button, correct button, cancel button, pay button
4. The current state is Idle?

4.1. Initialize state to Idle

4.2. Set display to welcome message

4.3. Enable screen

4.4. Disable payment module, column buttons, enter button, correct button, cancel button, pay button

4.5. Is there customer interaction?

a. If yes, go to Active state

b. If no, continue waiting for customer interaction

1. The current state is Active?

5.1. Initialize state to Active

5.2. Set display to available products and wait for user input (selected product and quantity)

5.3. Enable screen, column buttons, enter button, correct button, cancel button, pay button

5.4. Disable payment module

5.5. Go to Payment state

1. The current state is Payment?

6.1. Initialize state to Payment

6.2. Set display to payment methods

6.3. Enable screen, payment module, cancel button

6.4. Disable column buttons, enter button, correct button, pay button

6.5. Continue the process?

a. If yes, continue the payment process. After the payment process ends, go to Idle state

b. If no, go to Cancel state

1. The current state is Cancel?

7.1. Initialize state to Cancel

7.2. Enable screen

7.3. Disable column buttons, payment module, enter button, correct button, cancel button, pay button

7.4. Set display to confirmation message (yes or no)

a. If yes, go back to Active state

b. If no, go back to the previous process of the Payment state

1. End

## Logic 2 - Pseudocode:

In the Active state:

1. Create a new session array/collection.
2. Create a new transaction.
3. Wait for the user to select a product.
4. User selects a product slot (e.g., 'A9').
   1. Input : slotID
5. Receive the quantity of the product from the user (between 1 and 9).
   1. Input : quantity
6. If the user makes a mistake and presses the 'Correct' button, ask for the quantity again.
   1. go to: Step-5
7. If the user presses the 'Enter' button, check if the selected product and quantity are in Inventory.
8. If there is no stock, display "Out of Stock", delete the current transaction, and move back to create
   1. a new transaction: go to: Step-2
9. If there is enough stock, add the product to the session.
10. If the user wants to add more products, go back to create a new transaction.
    1. go to: Step-2
11. Display the current session's transaction details on the screen.
    1. Get price from Product: Output : slotID, quantity, price, description (product name)
12. Pressing the 'Pay' button moves to the payment process.
13. Once the payment is completed, the session ends.
14. If the 'Cancel' button is pressed, move to the Cancel state.
    * Cancel the current session:
    * Clear all selections.
    * Display "Session has been canceled" to the user.
    * Move to idle/wait state.
15. END

## 

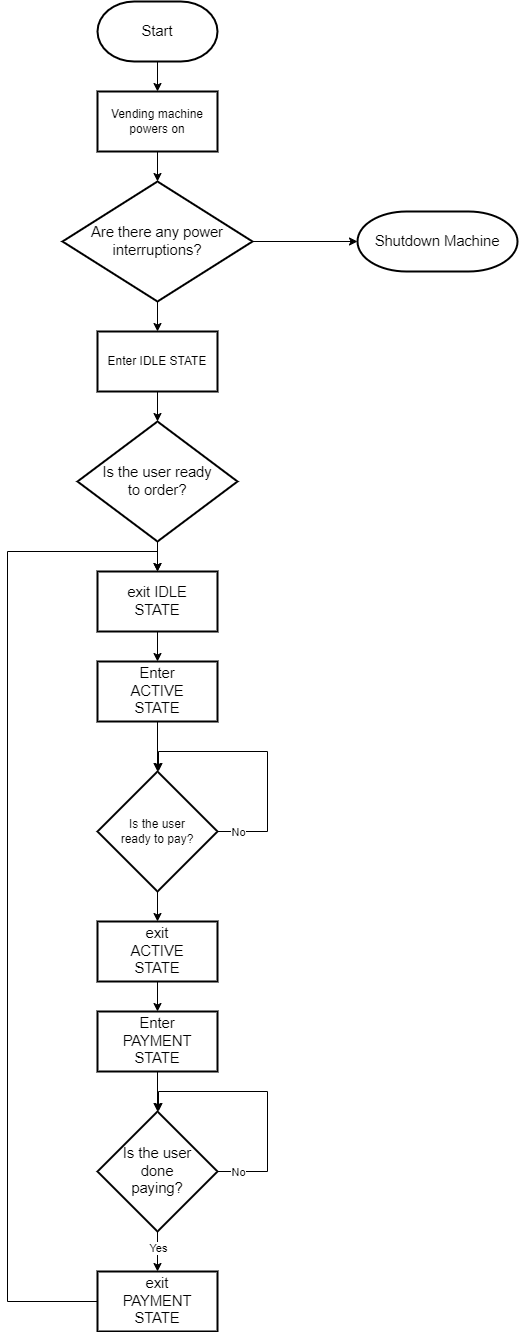
## Logic 3 - Pseudocode:

1. Start/**Active** state
2. Import and store **Session**
3. Display payment options and Session details
4. Does the customer want to **Cancel** the transaction?
   1. *Yes: Clear Session: continue (step: 8)*
   2. *No: continue (step: 5)*
5. Is the customer paying with a **Credit Card**?
   1. *Yes: Process* ***Credit Card*** *payment: continue (step: 8)*
   2. *No: continue (step: 6)*
6. Is the customer paying with a **Bank Card**?
   1. *Yes: Process* ***Bank Card*** *payment: continue (step: 8)*
   2. *No: continue (step: 7)*
7. Is the customer paying with an **NFC Phone**?
   1. *Yes: Process* ***NFC Phone*** *payment: continue (step: 8)*
   2. *No: Process* ***Vending Machine App*** *payment : continue (step: 8)*
8. Is payment successful?
   1. *Yes:continue (step: 9)*
   2. *No: continue (step: 4)*
9. Update **Product.quantity** for session items
10. is **Product.quantity** less than **Product.minQuantity**?
    1. *Yes: Create and send order for Product: continue (step: 11)*
    2. *No: continue (step: 11)*
11. Are there more products to update?
    1. *Yes: continue (step: 9)*
    2. *No: continue (step: 12)*
12. Dispense session items to customer
13. End/Go to **Idle/Ready** state

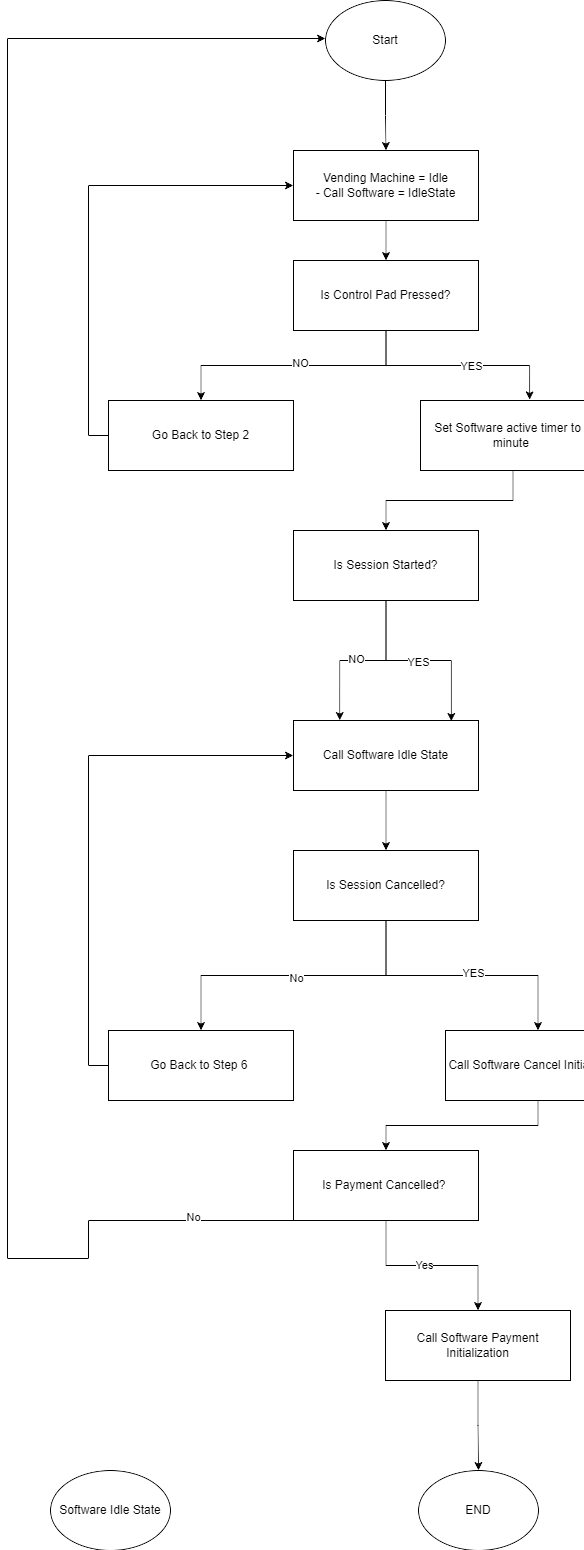
#### 

# FLOWCHARTS:

## Overall Flowchart:



## Logic 1 Flowcharts:



### 

## Logic 2 Flowcharts:

## 

## Logic 3 Flowcharts

# 

# TEST Cases

* Logic 1
* Test Case: Successful Transaction
  + Initial State: Vending machine is powered on and in idle state.
  + Actions:
    - User selects products using keypad.
    - User selects valid product combinations (e.g., A3, C8).
    - User confirms selection and proceeds to payment.
    - User completes payment successfully.
  + Expected Outcome:
    - Total price is displayed correctly.
    - Payment is accepted.
    - Products are dispensed.
    - Inventory is updated.
    - Transaction data is reset.
    - Vending machine returns to idle state.
* Test Case: Cancel Transaction
  + Initial State: Vending machine is powered on and in idle state.
  + Actions:
    - User selects products using keypad.
    - User selects valid product combinations (e.g., A3, C8).
    - User decides to cancel the transaction.
  + Expected Outcome:
    - Transaction data is reset.
    - Vending machine returns to idle state.
* Test Case: Invalid Product Selection
  + Initial State: Vending machine is powered on and in idle state.
  + Actions:
    - User selects products using keypad.
    - User enters invalid product combinations (e.g., B10).
  + Expected Outcome:
    - Error message is displayed indicating invalid selection.
    - User is prompted to re-enter valid product combination.
* Test Case: Insufficient Inventory
  + Initial State: Vending machine is powered on and in idle state.
  + Actions:
    - User selects products using keypad.
    - User selects product with insufficient inventory.
  + Expected Outcome:
    - Error message is displayed indicating insufficient inventory.
    - User is prompted to select another product.
* Test Case: Payment Rejection
  + Initial State: Vending machine is powered on and in idle state.
  + Actions:
    - User selects products using keypad.
    - User proceeds to payment.
    - User attempts payment with invalid payment method.
  + Expected Outcome:
    - Error message is displayed indicating payment rejection.
    - User is prompted to try again with valid payment method.

# FINAL DESCRIPTION

Through this workshop, we were able to communicate the logic parts written by each other in our group to create the final overall pseudocode and flowchart.

Through this workshop, we learned how to design vending machine hardware/software interaction, and we learned about writing detailed steps by analyzing each of the five hardware and four software states into modular components.

First, we learned how vending machines can operate more efficiently by managing inventory in real time, minimizing the human intervention required for maintenance, and optimizing each piece of hardware to do only what it needs to do.

In addition, the separation of hardware states and software logic gave us a deeper understanding of how complex systems are made up of simple logic. The process of defining and implementing the interaction of hardware and software in each state was very educational. Through this process, we realized the importance of meticulousness and accuracy in system design.

Lastly, this workshop made me realize the importance of both individual work and group collaboration. In our new group, everyone played their part to complete the entire project flawlessly.

The logic we learned through this workshop will probably be of great help to us in solving various problems.