
**ECE 378 -EMBEDDED LINUX**

**Project Report**

**On**

# Designing a GUI Model of a Vending Machine in Linux using Zenity GUI

**Submitted By**

[redacted]

**DATE OF SUBMISSION:16/11/2023.**

**School of Electronics and Electrical Engineering.**

## PAGE OF CONTENTS

# 1. INTRODUCTION

A vending machine is an automated machine that dispenses items such as snacks, beverages, cigarettes, and lottery tickets to consumers after cash, a credit card, or other forms of payment are inserted into the machine or otherwise made.Products are loaded into a machine and are available to purchase, generally 24/7. Vending products are commonly; freshly made beverages, bottles, cans, snacks and fresh food.

Vending machines exist in many countries and, in more recent times, specialized vending machines that provide less common products compared to traditional vending machine items have been created.

Here's a brief introduction to vending machines:

1.1 **Purpose and Functionality:**

- Vending machines serve the purpose of providing a quick and convenient way for customers to purchase goods without the need for human assistance.
- They are designed to dispense a variety of items, ranging from snacks, drinks, cigarettes, and even small electronics or personal care products.

1.2 **Types of Vending Machines:**

- Snack and Beverage Machines: These machines offer a selection of snacks, candies, chips, and a variety of beverages like sodas, juices, and water.
- Cigarette Machines: Designed for the sale of tobacco products, these machines are often found in some regions.
- Health and Hygiene Machines: Dispense personal care items like toiletries, condoms, and hygiene products.
- Electronics and Gadgets Machines: Found in some airports and shopping centers, they offer small electronic devices like chargers, headphones, and similar items.

1.3 **Locations and Use Cases:**

- Vending machines can be found in a wide range of locations, including:
- Airports
- Train/bus stations
- Shopping malls
- Schools and universities
- Office buildings
- Hotels
- Public areas like parks or plazas.

## 2. PROJECT OBJECTIVE

**Designing a GUI Model of Vending Machine in Embedded Linux Ubuntu using Zenity GUI.**

The primary objective of this project is to create a user-friendly Graphical User Interface (GUI) model for a vending machine implemented on an embedded Linux platform (Ubuntu) utilizing the Zenity GUI toolkit. The project aims to achieve the following specific goals:

2.1 **User-Friendly Interface:** Develop an intuitive and easily navigable GUI that allows users to interact with the vending machine effortlessly.

2.2 **Imitate Real-World Vending Machine Functionality: Create** a GUI that emulates the behavior and features of a physical vending machine, including product selection, payment options, and item dispensing.

2.3 **Interactive Product Selection:** Enable users to select products from a visually appealing display, simulating the process of choosing items from a real vending machine.

2.4 **Payment Integration:** Implement a simulated payment system within the GUI, allowing users to make virtual transactions using various methods such as coins, bills, or digital payments.

2.5 **Transaction Handling:** Ensure that the GUI accurately processes transactions, deducts the appropriate amount from the user's virtual "wallet," and dispenses the chosen product.

2.6 **Error Handling and Feedback: Provide** informative error messages and feedback to users in case of invalid inputs, insufficient funds, or other potential issues.

2.7 **Multi-Platform Compatibility:** Ensure that the GUI model is compatible with embedded Linux running on the Ubuntu operating system, allowing for seamless integration with target hardware.

2.8 **Resource Optimization:** Optimize the GUI to run efficiently on embedded systems, considering limited resources such as memory and processing power.

2.9 **Security Considerations:** Implement basic security measures to protect against unauthorized access or tampering with the vending machine's functionality.

2.10 **Documentation and User Guide**: Provide comprehensive documentation including installation instructions, user guide, and details about the design and implementation choices.

2.11 **Testing and Validation:** Conduct rigorous testing to verify the functionality and reliability of the GUI model, addressing potential bugs or issues.

2.12 **Scalability and Customization:** Design the GUI in a modular and extensible manner, allowing for future enhancements or customization of features.By successfully accomplishing these objectives, the project aims to deliver a functional GUI model of a vending machine, demonstrating the effective use of Zenity GUI toolkit in an embedded Linux environment. This model can serve as a foundation for further development or as a learning resource for individuals interested in embedded systems and GUI design.

## 3. About Zenity GUI:-

"By using Zenity, a special tool for creating user interfaces, we've made our project even better. Zenity helps us add buttons, menus, and other things that people can click on to use our vending machine. It's like giving our project a nice coat of paint to make it look and work really well. This project shows how Zenity can make things easy and fun for people using our vending machine on a computer with Linux inside."

### 3.1 About Zenity:-

**Zenity** is a rewrite of **gdialog**, the GNOME port of dialog which allows you to create a set of widgets for different operations where users can interact with the script graphically. The widgets are based on the GTK toolkit.

Zenity is an open source program written using **C** programming language. It supports Linux, BSD and Windows. Without further ado, let us see how to display graphical GTK+ dialog boxes from the commandline and shell scripts using Zenity.

### 3.2 How to install Zenity in Linux

First, check if zenity is installed and available to use by running the following commands:

```
$ which zenity
/usr/bin/zenity
$ zenity --version
3.32.0

$ zenity –about
```


*Figure 1 Zenity Gui Interface*

If zenity is not installed in your distribution, run the following commands depending on your distribution to install it.

**Alpine Linux:**

```
$ sudo apk add zenity
```

## 4. Zenity GUI & it's Options :-

### 4.1 Zenity Help & common options

For each widget, there is an associated help option through which you can get a set of supported options.

Run the following command to access the help section.

```
$ zenity --help
```

### 4.2 Message dialog box

A message dialog box will display an error, warning, info, and question dialog box. Depending upon the situation you have to use the appropriate dialog box in the script.

### 4.3 Error dialog box

To access the list of supported options for the error dialog box, run the following command:

```
$ zenity --help-error
```

To display the error dialog box in the script, use the following command. Here, the `--error` flag will create the error dialog box and `--text` flag will print the text message. You can see from the image there is an error icon associated with the dialog box.

```
zenity --error \
    --title "Error Message" \
    --width 500 \
    --height 100 \
   --text "Permission denied. Run with sudo or as root user."
```

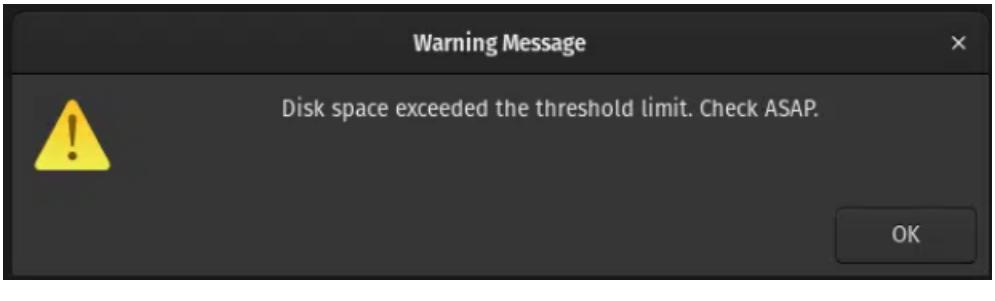



*Figure 2 Message dialog Box*

### 4.4 Warning dialog box

To access the list of supported options for the warning dialog box, run the following command:

```
$ zenity --help-warning
```

Use `--warning` flag in the script to display the warning box and `--text` flag to display the warning message.

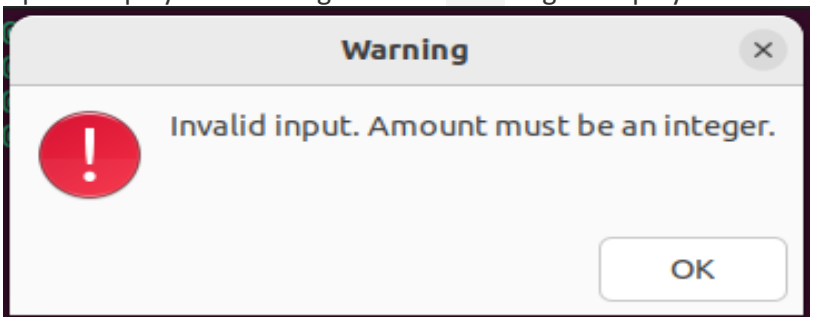



*Figure 3 Warning dialog box*

### 4.5 Info dialog box

To access the list of supported options for the info dialog box, run the following command:

```
$ zenity --help-info
```

Use `--info` flag in the script to display the infobox and `--text` flag to display the info message.

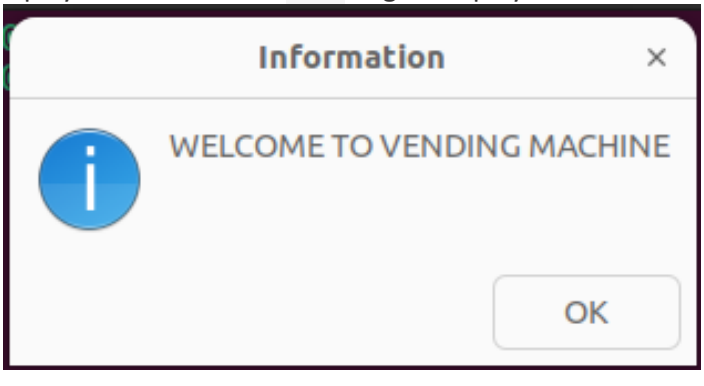



*Figure 4 Information dialog box*

### 4.6 Password dialog box

To access the list of supported options for the password dialog box, run the following command:

```
$ zenity --help-password
```

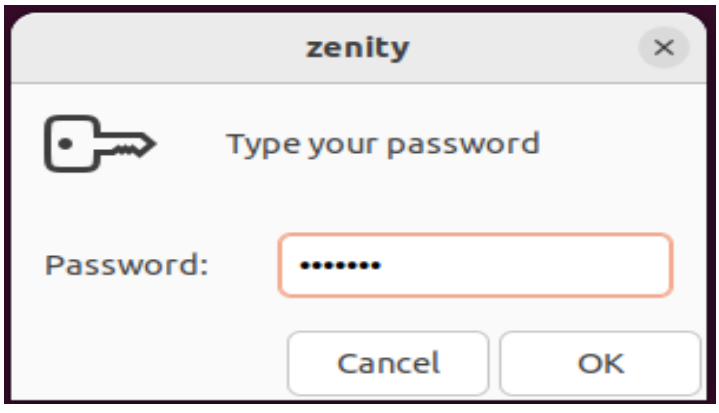



*Figure 5 Password dialog box*

### 4.7 Question dialog box

To access the list of supported options for the question dialog box, run the following command:

```
$ zenity --help-question
```

The question dialog box will prompt a message along with a **Yes** or **No** option. If you press **"Yes"**, the return code will be **"zero"** and for **"No"** the return code will be **"one"**. You have to use the exit codes to write further logic in your scripts.

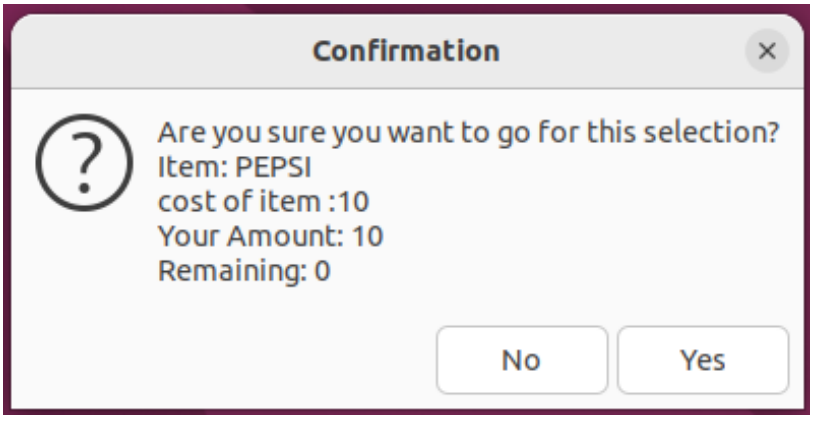



*Figure 6 Question dialog box*

## 5. Admin Privileges:

5.1 **Add Item:** The administrator can easily add new items to the vending machine inventory. This feature streamlines the process of expanding the product offerings based on consumer demand or preferences.

5.2 **Delete Item:** Should the need arise, the admin can swiftly remove items from the machine's inventory. This capability allows for efficient management of the available product selection.

5.3 **Update Item:** The admin has the authority to modify the price of existing items. This feature provides flexibility in responding to market fluctuations or cost adjustments.

## 6. Customer Interaction:

6.1 **Selection Process:** Customers can interact with the vending machine through an intuitive graphical interface. They can view a list of available items and make selections based on their preferences and needs.

6.2 **Payment Options:** The system first asks to Enter the Amount. It only Accepts 10 and 20 Rupee notes. If the customer Enters 20 rupees and purchases 10 rupees worth item, the system displays a message to collect the remaining cash. This ensures a seamless and convenient purchasing experience for customers.

6.3 **Transaction Completion:** Upon successful payment, the selected item is dispensed, and the transaction is finalized.

## 7. Project Source Code

```
#!/bin/bash
# Array to store items
declare -a items
items=("COKE" "PEPSI" "SPRITE")
# Array to store prices
declare -a MRP
MRP=("20" "10" "20")
# Array to store quantities
declare -a Qty
Qty=("5" "2" "3")
# Array to store admin changes
declare -a admin_changes
# Function to add an item
function add_item() {
  local name=$(zenity --entry --width 500 --height 300 --title="Add Item" --text="Enter item name:")
  if [ $? = 1 ]; then
    return  # Terminate the step if Cancel is clicked
  fi
  local valid_price=false
  local valid_qty=false
  local price
  local qty
  while [ "$valid_price" != true ] || [ "$valid_qty" != true ]; do
    price=$(zenity --entry --width 500 --height 300 --title="Add Item" --text="Enter item price:")
    # Check if price is a valid integer
    if [[ "$price" =~ ^[0-9]+$ ]]; then
      valid_price=true
    else
      zenity --warning --width 500 --height 250 --text="Invalid input. Price must be an integer."
      break;
    fi
    # Prompt for quantity
    qty=$(zenity --entry --width 500 --height 300 --title="Add Item" --text="Enter item quantity:")

    # Check if quantity is a valid integer
    if [[ "$qty" =~ ^[0-9]+$ ]]; then
```

```
            valid_qty=true
        else
            zenity --warning --width 500 --height 250 --text="Invalid input. Quantity must be an integer."
        fi
    done
    if [ -n "$name" ] && [ -n "$price" ] && [ -n "$qty" ]; then
        items+=("$name")
        MRP+=("$price")
        Qty+=("$qty")
        admin_changes+=("Added item: $name $price $qty")
        zenity --info --width 500 --height 250 --text="Item added successfully!"
    else
        zenity --warning --width 500 --height 250 --text="Invalid input. Item not added."
    fi
}
# Function to delete an item
function delete_item() {
    local item_list=()
    for i in "${!items[@]}"; do
        local item_name=${items[$i]}
        local item_price=${MRP[$i]}
        local item_qty=${Qty[$i]}
        item_list+=("$i" "$item_name" "$item_price" "$item_qty")
    done
    local item_no=$(zenity --list --width 500 --height 300 --title="Delete Item" --text="Select item to delete" --column="Item No." --
column="Item Name" --column="Price" --column="Quantity" "${item_list[@]}")
    if [ -n "$item_no" ]; then
        unset 'items[$item_no]'
        unset 'MRP[$item_no]'
        unset 'Qty[$item_no]'
        admin_changes+=("Deleted item: ${items[$item_no]} ${MRP[$item_no]} ${Qty[$item_no]}")
        zenity --info --width 500 --height 250 --text="Item deleted successfully!"
    else
        zenity --warning --width 500 --height 250 --text="No item selected. Deletion cancelled."
    fi
}
# Function to update item price and quantity
function update_price() {
    local item_list=()
    for i in "${!items[@]}"; do
        item_list+=("$i" "${items[$i]}" "${MRP[$i]}" "${Qty[$i]}")
    done
    local item_no=$(zenity --list --width 500 --height 300 --title="Update Price" --text="Select item to update price and quantity" --
column="Item No." --column="Item Details" --column="Price" --column="Quantity" "${item_list[@]}")
    local valid_new_price=false
    local valid_new_qty=false
    local new_price
    local new_qty
    while [ "$valid_new_price" != true ] || [ "$valid_new_qty" != true ]; do
        new_price=$(zenity --entry --width 500 --height 300 --title="Update Price" --text="Enter new price:")
        new_qty=$(zenity --entry --width 500 --height 300 --title="Update Price" --text="Enter new quantity:")
        # Check if new_price and new_qty are valid integers
        if [[ "$new_price" =~ ^[0-9]+$ ]] && [[ "$new_qty" =~ ^[0-9]+$ ]]; then
```

```
            valid_new_price=true
            valid_new_qty=true
        else
            zenity --warning --width 500 --height 250 --text="Invalid input. Price and Quantity must be integers."
        fi
    done
    if [ -n "$item_no" ] && [ -n "$new_price" ] && [ -n "$new_qty" ]; then
        local updated_item=${items[$item_no]}
        items[$item_no]="${updated_item% *}"
        MRP[$item_no]=$new_price
        Qty[$item_no]=$new_qty
        admin_changes+=("\nUpdated item price and quantity: ${updated_item% *}")
        zenity --info --width 500 --height 250 --text="Price and quantity updated successfully!"
    else
        zenity --warning --width 500 --height 250 --text="Invalid input. Price and quantity not updated."
    fi
}
# Function to ask for quantity before proceeding
function ask_quantity() {
    local valid_qty=false
    local quantity
    while [ "$valid_qty" != true ]; do
        quantity=$(zenity --entry --width 500 --height 300 --title="Quantity" --text="Enter quantity:")
        # Check if quantity is a valid integer
        if [[ "$quantity" =~ ^[0-9]+$ ]]; then
            valid_qty=true
        else
            zenity --warning --width 500 --height 250 --text="Invalid input. Quantity must be an integer."
        fi
    done
    # Proceed with the selected quantity
    echo "$quantity"
}
# To welcome the vending machine
zenity --info --width 500 --height 250 --text="WELCOME TO VENDING MACHINE"
# Use selection box for admin or customer
activity1=$(zenity --list --width 500 --height 300 --text "Select the option" --radiolist --column "Select" --column "you are" FALSE
"admin" TRUE "customer")
# If it is admin, display the password
if [ "$activity1" = "admin" ]; then
    OUTPUT=$(zenity --password --width 500 --height 250 "Password")

    # Check if password is correct
    if [ "$OUTPUT" = "abc1234" ]; then
        while true; do
            # Display admin menu
            admin_action=$(zenity --list --width 500 --height 400 --text "Admin Menu" --column "Action" "Add Item" "Delete Item"
"Update Price" "Exit")

            case $admin_action in
                "Add Item")
                    add_item ;;
                "Delete Item")
```

```
            delete_item ;;
          "Update Price")
            update_price ;;
          "Exit")
            break;;
          *)
            zenity --warning --width 500 --height 250 --text="Invalid selection" ;;
        esac
      done
    else
      # Throw an error
      zenity --error --width 500 --height 250 --text "Password incorrect. Terminating"
      exit
    fi
fi
# Use selection box for admin or customer
activity1=$(zenity --list --width 500 --height 300 --text "Select the option" --radiolist --column "Select" --column "you are" FALSE
"admin" TRUE "customer")
# If customer enters the amount
valid_entry=false
while [ "$valid_entry" != true ]; do
    ENTRY=$(zenity --entry --width 500 --height 300 --title="Amount" --text="Enter your amount:")
    # Check if ENTRY is a valid integer
    if [[ "$ENTRY" =~ ^[0-9]+$ ]]; then
      valid_entry=true
    else
      zenity --warning --width 500 --height 250 --text="Invalid input. Amount must be an integer."
    fi
done
if [ "$ENTRY" -eq 10 ] || [ "$ENTRY" -eq 20 ] || [ "$ENTRY" -eq 30 ] || [ "$ENTRY" -eq 40 ] || [ "$ENTRY" -eq 50 ]; then
    # Ask for quantity before proceeding
    selected_quantity=1
    # Display item list with quantity
    item_list=()
    for i in "${!items[@]}"; do
      item_list+=("$i" "${items[$i]}" "${MRP[$i]}" "${Qty[$i]}")
      REMAINING=$((ENTRY-(MRP[$i]*selected_quantity)))
    done
    ITEM=$(zenity --list --width 500 --height 400 --title="Choose the item You Wish to purchase" --text="Select items from the list
below" --column="Item No." --column="Item Details" --column="Price" --column="Quantity" "${item_list[@]}")
    selected_quantity=$(ask_quantity)
      if [ "$selected_quantity" -gt "${Qty[$ITEM]}" ]; then
          zenity --warning --width 500 --height 250 --text="Insufficient Quantity. Transaction cancelled."
          exit
      fi
    REMAINING=$((ENTRY-(MRP[$i]*selected_quantity)))
    if [ -n "$ITEM" ]; then
      # Calculate remaining amount
      SELECTED_PRICE=$(echo "$ITEM" | awk '{print $3}')
      #REMAINING=$((ENTRY-(MRP[$i]*selected_quantity)))
       REMAINING=$((ENTRY - (MRP[$ITEM] * selected_quantity)))
      # Ask for confirmation
```

```
    zenity --question --width 500 --height 300 --title="Confirmation" --text "Are you sure you want to go for this selection?\nItem: ${items[$ITEM]}\nCost of item: $((MRP[$ITEM]*selected_quantity))\nActual Quantity: ${Qty[$ITEM]}\nQuantity Chosen: $selected_quantity\nYour Amount: $ENTRY\nRemaining: $REMAINING"
    if [ $? = 0 ]; then
      # Display selected item
      #zenity --info --width 500 --height 300 --text="You selected the item::\n${items[$ITEM]} (Qty: ${Qty[$ITEM]})"
      if [ "$ENTRY" -lt "$((MRP[$ITEM]*selected_quantity))" ]; then
        zenity --warning --width 500 --height 250 --text="Insufficient amount. Transaction cancelled."
        exit
      fi
      zenity --info --width 500 --height 300 --text="Item_No: $ITEM\nAmount: $ENTRY\nRemaining: $REMAINING"
      # Update the quantity of the selected item
      Qty[$ITEM]=$((Qty[$ITEM]-selected_quantity))
      # Update the ITEM variable and item_list array after the purchase
      item_list=()
      for i in "${!items[@]}"; do
        item_list+=("$i" "${items[$i]}" "${MRP[$i]}" "${Qty[$i]}")
      done
    else
      zenity --info --width 500 --height 250 --text="Transaction cancelled."
      exit
    fi
  else
    zenity --warning --width 500 --height 250 --text="Item not selected. Transaction cancelled."
    exit
  fi
else
  # If amount not entered, display warning
  zenity --warning --width 500 --height 250 --text="Invalid Amount entered!!!.\nEnter amount less than 60 rupees!"
  exit
fi
if [ "$REMAINING" -gt 0 ]; then
  zenity --info --width 500 --height 250 --text="Transaction successful!"
  zenity --info --width 500 --height 250 --text="Collect your $((ENTRY-(MRP[$i]*selected_quantity))) rupees cash!!"
else
  zenity --info --width 500 --height 250 --text="Transaction successful!"
fi
ITEM=$(zenity --list --width 500 --height 400 --title="Vending Machine" --text="Items list " --column="Item No." --column="Item Details" --column="Price" --column="Quantity" "${item_list[@]}")
# Thank you message
zenity --info --width 500 --height 250 --text="THANK YOU VISIT AGAIN"
```

## 8. Code Explanation: -

**8.1**

```
#!/bin/bash
    # Array to store items
    declare -a items
    items=("COKE" "PEPSI" "SPRITE")
    # Array to store prices
    declare -a MRP
    MRP=("20" "10" "20")
    # Array to store quantities
```

```
declare -a Qty
Qty=("5" "2" "3")
# Array to store admin changes
declare -a admin_changes
```

- The script starts by declaring three arrays (items, MRP, Qty) to store information about items in a vending machine, including their names, prices, and quantities. Another array (admin_changes) is set up to keep track of administrative changes.

**8.2**
```
# Function to add an item
function add_item() {
    local name=$(zenity --entry --width 500 --height 300 --title="Add Item" --text="Enter item name:")
    if [ $? = 1 ]; then
        return  # Terminate the step if Cancel is clicked
    fi
    if [ -n "$name" ] && [ -n "$price" ] && [ -n "$qty" ]; then
        items+=("$name")
        MRP+=("$price")
        Qty+=("$qty")
        admin_changes+=("Added item: $name $price $qty")
        zenity --info --width 500 --height 250 --text="Item added successfully!"
    } else {
        zenity --warning --width 500 --height 250 --text="Invalid input. Item not added."
    }
}
```

- The add_item function is responsible for adding a new item to the vending machine. It uses zenity to prompt the user for the item name, price, and quantity. The input is validated, and if valid, the item details are added to the respective arrays (items, MRP, Qty), and a log is recorded in admin_changes.

**8.3**
```
# Function to delete an item
function delete_item() {
    local item_list=()
    for i in "${!items[@]}"; do
        local item_name=${items[$i]}
        local item_price=${MRP[$i]}
        local item_qty=${Qty[$i]}
        item_list+=("$i" "$item_name" "$item_price" "$item_qty")
    done

    local item_no=$(zenity --list --width 500 --height 300 --title="Delete Item" --text="Select item to delete" --column="Item No." --column="Item Name" --column="Price" --column="Quantity" "${item_list[@]}")

    if [ -n "$item_no" ]; then
        unset 'items[$item_no]'
        unset 'MRP[$item_no]'
        unset 'Qty[$item_no]'
        admin_changes+=("Deleted item: ${items[$item_no]} ${MRP[$item_no]} ${Qty[$item_no]}")
        zenity --info --width 500 --height 250 --text="Item deleted successfully!"
    } else {
        zenity --warning --width 500 --height 250 --text="No item selected. Deletion cancelled."
```

```
    }
}
```

- The delete_item function displays a list of items to the user using zenity, and the user can choose an item to delete. The selected item is then removed from the arrays (items, MRP, Qty), and the deletion is logged in admin_changes.

**8.4** # Function to update item price and quantity

```
function update_price() {
    local item_list=()
    for i in "${!items[@]}"; do
        item_list+=("$i" "${items[$i]}" "${MRP[$i]}" "${Qty[$i]}")
    done
    local item_no=$(zenity --list --width 500 --height 300 --title="Update Price" --text="Select item to update price and quantity" --column="Item No." --column="Item Details" --column="Price" --column="Quantity" "${item_list[@]}")
    if [ -n "$item_no" ] && [ -n "$new_price" ] && [ -n "$new_qty" ]; then
        local updated_item=${items[$item_no]}
        items[$item_no]="${updated_item% *}"
        MRP[$item_no]=$new_price
        Qty[$item_no]=$new_qty
        admin_changes+=("\nUpdated item price and quantity: ${updated_item% *}")
        zenity --info --width 500 --height 250 --text="Price and quantity updated successfully!"
    } else {
        zenity --warning --width 500 --height 250 --text="Invalid input. Price and quantity not updated."
    }
}
```

- The update_price function allows the user to update the price and quantity of an item. Similar to previous functions, it uses zenity to prompt the user for the item to update and the new price and quantity. If the inputs are valid, the arrays are updated, and the change is logged.

**8.5** # Function to ask for quantity before proceeding

```
function ask_quantity() {
    local valid_qty=false
    local quantity
    while [ "$valid_qty" != true ]; do
        quantity=$(zenity --entry --width 500 --height 300 --title="Quantity" --text="Enter quantity:")

        # Check if quantity is a valid integer
        if [[ "$quantity" =~ ^[0-9]+$ ]]; then
            valid_qty=true
        else
            zenity --warning --width 500 --height 250 --text="Invalid input. Quantity must be an integer."
        fi
    done
    # Proceed with the selected quantity
    echo "$quantity"
}
```

- The ask_quantity function prompts the user to enter a quantity for an item and ensures that the input is a valid integer.

**8.6** # To welcome the vending machine

```
zenity --info --width 500 --height 250 --text="WELCOME TO VENDING MACHINE"

# Use selection box for admin or customer
activity1=$(zenity --list --width 500 --height 300 --text "Select the option" --radiolist --column "Select" --column "you are" FALSE "admin" TRUE "customer")
```

- The script starts by displaying a welcome message and then prompts the user to select whether they are an admin or a customer using zenity.

**8.7** # If it is admin, display the password

```
if [ "$activity1" = "admin" ]; then
    OUTPUT=$(zenity --password --width 500 --height 250 "Password")
    # Check if password is correct
    if [ "$OUTPUT" = "abc1234" ]; then
        while true; do
            # Display admin menu
            admin_action=$(zenity --list --width 500 --height 400 --text "Admin Menu" --column "Action" "Add Item" "Delete Item"
"Update Price" "Exit")
        done
    } else {
        # Throw an error
        zenity --error --width 500 --height 250 --text "Password incorrect. Terminating"
        exit
    }
}
```

- If the user selected "admin," the script prompts for a password using zenity. If the password is correct, an admin menu is displayed where the admin can choose to add an item, delete an item, update prices, or exit.

**8.8** # If customer enters the amount

```
valid_entry=false
while [ "$valid_entry" != true ]; do
    ENTRY=$(zenity --entry --width 500 --height 300 --title="Amount" --text="Enter your amount:")
Done
```

- If the user selected "customer," the script prompts the customer to enter an amount. The amount must be a valid integer, and the script ensures this before proceeding.

**8.9** if [ "$ENTRY" -eq 10 ] || [ "$ENTRY" -eq 20 ] || [ "$ENTRY" -eq 30 ] || [ "$ENTRY" -eq 40 ] || [ "$ENTRY" -eq 50 ]; then

```
    # Ask for quantity before proceeding
    selected_quantity=1
} else {
    # If amount not entered, display warning
    zenity --warning --width 500 --height 250 --text="Invalid Amount entered!!!.\nEnter amount less than 60 rupees!"
    exit
}
```

- The script checks if the entered amount is valid (10, 20, 30, 40, or 50). If valid, it proceeds to ask for the quantity and then displays the item list for the customer to choose from.

**8.10**if [ "$REMAINING" -gt 0 ]; then

```
    zenity --info --width 500 --height 250 --text="Transaction successful!"
    zenity --info --width 500 --height 250 --text="Collect your $((ENTRY-(MRP[$i]*selected_quantity))) rupees cash!!"
} else {
    zenity --info --width 500 --height 250 --text="Transaction successful!"
}
```

- After the customer makes a selection, the script calculates the remaining amount and displays a success message. If there is any remaining amount, it informs the customer to collect the change.

**8.11**ITEM=$(zenity --list --width 500 --height 400 --title="Vending Machine" --text="Items list " --column="Item No." --column="Item Details" --column="Price" --column="Quantity" "${item_list[@]}")

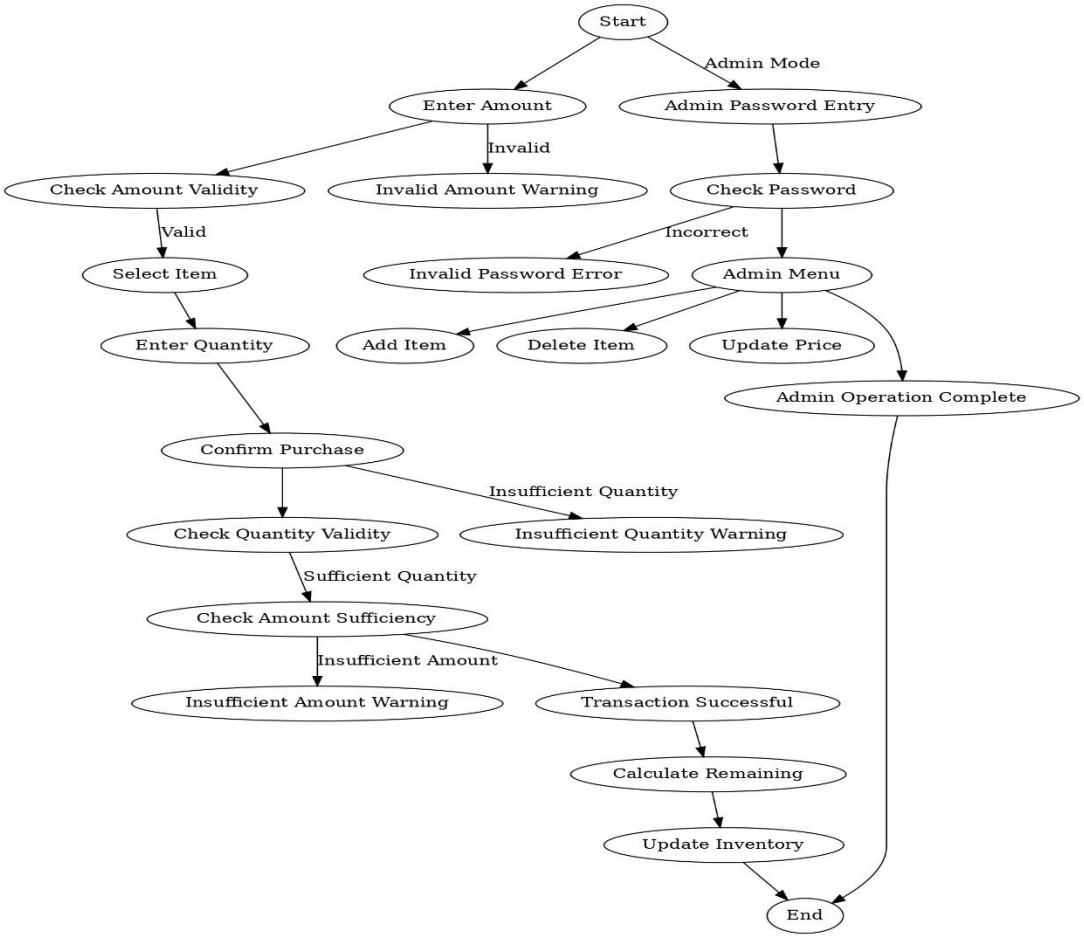
- Finally, the script displays the updated items list using zenity.

**8.12#** Thank you message

zenity --info --width 500 --height 250 --text="THANK YOU VISIT AGAIN"

- The script concludes with a thank you message.

## 9. Implementation & BlockDiagram of the Project:-




## 10. Results and Discussions:-

10.1 Welcome Dialog

This is the welcome dialog box that greets the user when they start the vending machine script. It's a simple informational message indicating the beginning of the interaction with the vending machine.

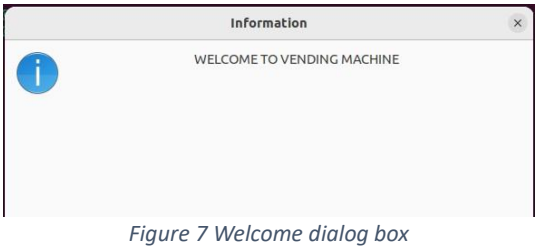



*Figure 7 Welcome dialog box*

10.2 Admin Men

Once the admin logs in, this menu is presented, listing the possible actions the admin can take: adding, deleting, or updating items, or exiting the admin menu.

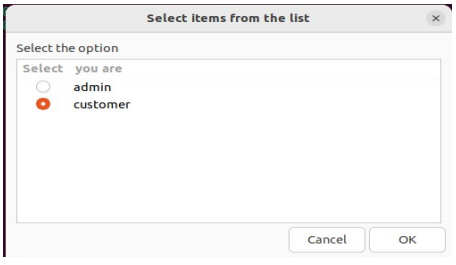



*Figure 8 Main dialog box*

10.3 Password Prompt:

This password prompt is for the admin login. If the user selects "admin" in the previous dialog, they must enter the correct password to proceed to the admin functions.

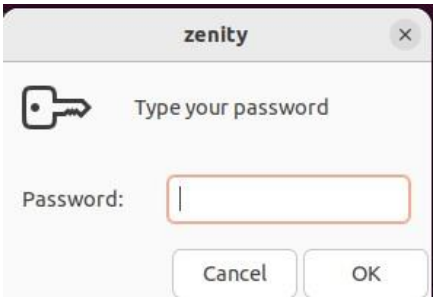



*Figure 9Password dialog box*

10.4 Add Item Dialog - Name Entry:-

This dialog box is where the admin can add a new item to the inventory. The admin is prompted to enter the name of the new item, in this case, "Lays".

Add Item Dialog - Price Entry:

After entering the name of the item, the admin is then asked to input the price for the new item. Here, the price entered is "20".

Add Item Dialog - Quantity Entry:

Following the price entry, the admin is prompted to enter the quantity of the new item being added to the vending machine. The quantity specified is "5".

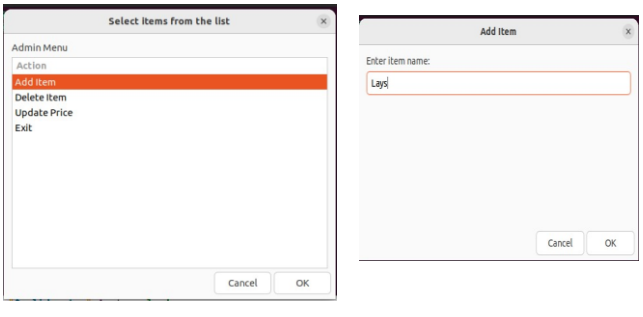



*Figure 11 Options dialog box*

10.5 Delete Item Dialog:

This dialog box presents a list of items currently available for deletion by an admin. The user has selected the last item, "Lays", for deletion.

Item Deletion Confirmation:

After the user confirms the deletion of an item, this message box confirms that the item has been successfully removed from the vending machine's inventory.

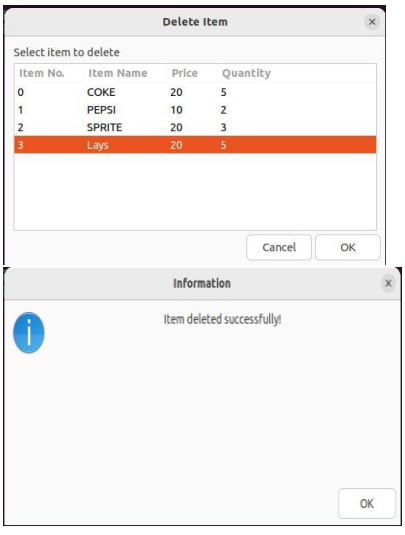



*Figure 12Delete dialog box*

## 10.6 Updated Items List Dialog:

This dialog box shows the updated list of items after the transaction, with the quantity of "PEPSI" reduced according to the purchase.

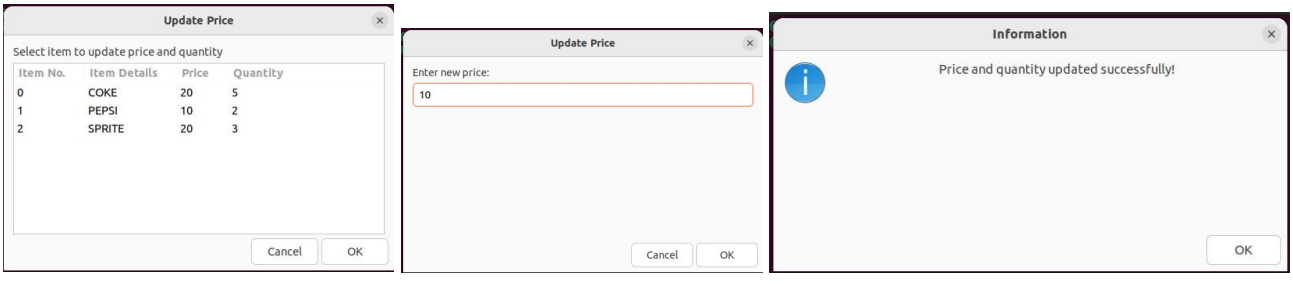



*Figure 13 Update dialog boxes*

## 10.7 User Type Selection:

In this dialog box, the user is prompted to identify as either an "admin" or a "customer". This selection will determine the subsequent options and interfaces presented to the user.

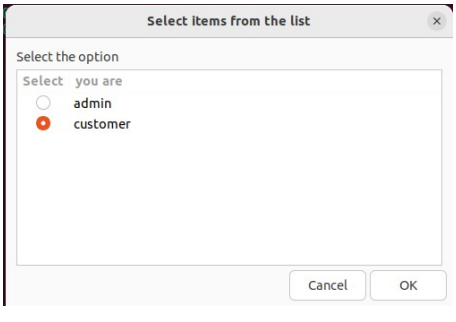



*Figure 14 User Type Selectio*

## 10.8 Amount Input Dialog:

This dialog box prompts the customer to enter the amount of money they have to spend in the vending machine.

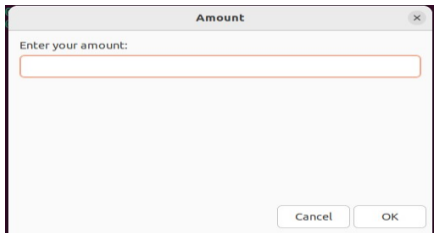



*Figure 15 Amount Input Dialo*

10.9 Item Selection Dialog:

Here, the customer is presented with a list of items available for purchase from the vending machine. The customer has selected "PEPSI" with the intention to purchase it.

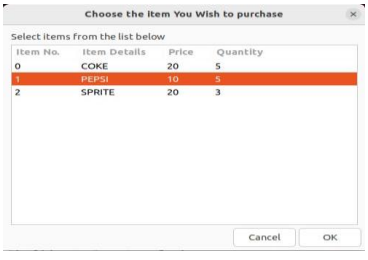



*Figure 16 Item Selection Dialog:*

10.10 Quantity Input Dialog:

After selecting an item, this dialog box asks the customer to specify the quantity of the chosen item they wish to buy. The entered quantity is "3".

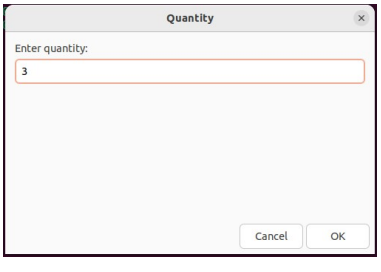



*Figure 17 Quantity Input Dialog:*

10.11 Purchase Confirmation Dialog:

This confirmation dialog box summarizes the purchase details including the item selected, the cost, the actual and chosen quantity, the amount entered by the customer, and the remaining balance after the transaction.

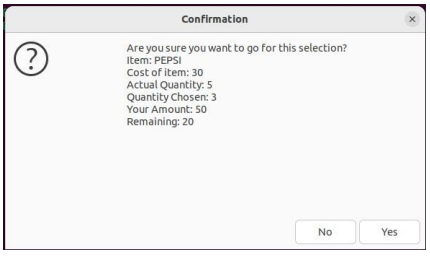



*Figure 18 Purchase Confirmation Dialog:*

10.12 Transaction Success Dialog:

This message box informs the customer that the transaction has been successful.

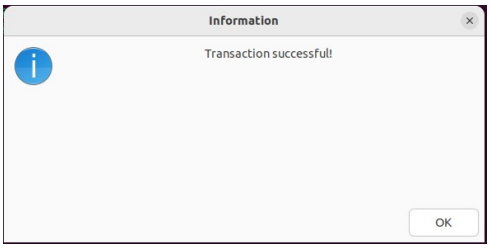



*Figure 19 Transaction Success Dialog:*

10.13 Change Collection Dialog:

An information dialog box tells the customer the amount of change to collect. In this case, it incorrectly shows a negative amount, indicating a possible error in the script or the transaction.

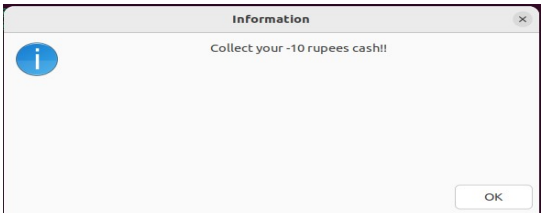



*Figure 20 Change Collection Dialog:*

10.14 Insufficient Funds Warning Dialog:

This warning dialog appears if the customer does not enter a sufficient amount of money to cover the cost of the items they wish to purchase, leading to the cancellation of the transaction.

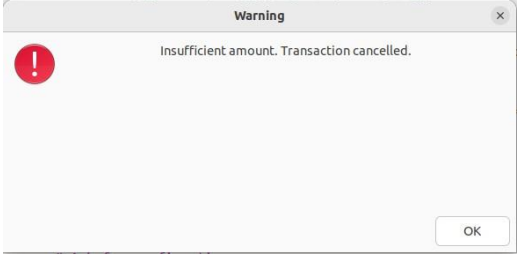



*Figure 21 Insufficient Funds Warning Dialog*

10.15 Thank You Dialog:

A courteous message thanking the customer for using the vending machine, typically shown at the end of the interaction.

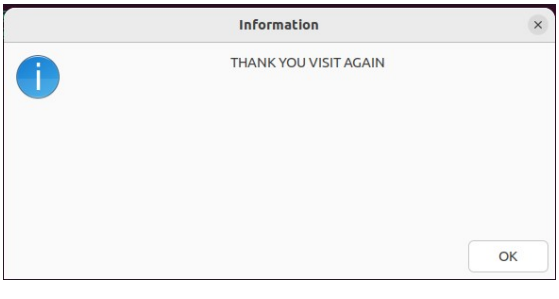



*Figure 22 Thank You Dialog*

## 11. Advantages of Vending Machines:

1. 24/7 Accessibility: Vending machines operate around the clock, providing consumers with access to products even during non-business hours, making them incredibly convenient for late-night or early-morning purchases.
2. Reduced Labor Costs: Unlike traditional stores that require staff for operations, vending machines operate autonomously, eliminating the need for employees and reducing labor expenses for business owners.

3. Convenience in High-Traffic Areas: Vending machines excel in high-traffic locations like airports, train stations, and malls, offering a quick and efficient way for people on the go to purchase items without the need for a full-service store.
4. Instant Gratification: Vending machines offer immediate access to products, satisfying consumer needs and impulses without the wait associated with traditional shopping experiences.
5. Diverse Product Range: Vending machines can dispense a wide array of products, from snacks, beverages, and sandwiches to personal care items, electronics, and even specialty products like health supplements, providing a convenient one-stop solution for various needs.

## 12. Disadvantages of Vending Machines:

1. Limited Freshness of Goods: Perishable items in vending machines may not be as fresh as those in stores, potentially impacting the quality and appeal of products like sandwiches, fruits, or dairy items.
2. Maintenance and Repair Costs: Vending machines require regular maintenance, and technical issues may arise, necessitating repairs, which can incur additional expenses for operators.
3. Lack of Personalized Service: Vending machines lack the human touch and personalized service that a store employee can provide, which can be a drawback for customers seeking advice or assistance with their purchases.
4. Security Vulnerabilities: Vending machines can be targets for theft or vandalism, especially if they are placed in unmonitored or high-risk areas, potentially leading to financial losses for operators.
5. Limited Payment Options: Some vending machines may only accept cash, potentially excluding customers who prefer or exclusively use electronic forms of payment, leading to missed sales opportunities.

## 13. Future Scope :-

The future scope of the vending machine project envisions a seamless integration of online payment systems, ushering in an era of convenience and connectivity. With real-time inventory updates and remote management capabilities, users will experience a personalized journey through a dedicated mobile app. Cashless transactions, facilitated by credit/debit cards, mobile wallets, and NFC technology, will enhance transaction security. The implementation of a robust online payment gateway will further ensure the safety of financial transactions. Advanced features include IoT-driven inventory management, data analytics for customer insights, and machine learning for personalized product recommendations. Additionally, a customer feedback system, energy-efficient practices, multilingual support, and promotional programs will contribute to an inclusive and modern vending machine experience. This evolution aims to meet the diverse needs of consumers while staying at the forefront of technological innovation.

## 14. Conclusion:

The "Vending Machine Interface with Zenity GUI" project exemplifies the convergence of modern technology and convenience. By employing Linux, Zenity GUI concepts, and a sophisticated programmed logic, this project demonstrates the potential for creating highly functional and user-friendly vending solutions. With features tailored for both administrators and customers, this system sets a new standard for vending machine interactions.

## 15. References


[1]. S. S. T. T. M. B. P. M. Sravan Kumar, "Medicine Vending Machine," International Research Journal of Engineering and Technology, vol. 7, no. 6, June 2020.

[2]. T. m. S. m. M. B. Mulugeta Eshetu, "Establishment of Vending Machine Business," 2009.

[3]. G. M. M. M. s. M. y. A. DV Raagu, "“Design and Fabrication of Coin Operated Portable Water Vending Machine.," 2016-2017.

[4]. l. M. K. M. E. M. j. T. Vennan sibanda, "Design of a high-tech vending machine," vol. 30, 2020.

[5]. R. B. B. R. J. G. Susanne Gruber, "The Commodity Vending Machine," FORUM WARE international, February 2005.

[6]. [Online]. Available: http://www.tcnvend.com/main-application-of-vending-machine-182.html.

[7]. A. Alushi, 2008. [Online]. Available: http://umpir.ump.edu.my/id/eprint/193/3/Development%20of%20vending%20machine%20with%20prepaid%20payment%20method%20-%20Chapter%201.pdf.

[8]. M. L. Kasavana, "V-Commerce: vending Machine Technology," vol. 21, no. 1, January 2003.

[9]. "Strategyr influencer driven," [Online]. Available: https://www.strategyr.com/market-report-vending-machines-forecasts-global-industry-analysts-inc.asp#:~:text=Amid%20the%20COVID%2D19%20crisis,the%20analysis%20period%202020%2D2027.

[10]. M. A. DT Wiyanti, "Automated vending machine with Iot infrastructure for smart factory application," in International Conference on Mathematics, Science, and Education, 2019.

[11]. "New Technologies Are Revolutionising The World Of Vending Machines - For The Better," 2020.

[12]. B. P. a. B. Singh., "Design and Development of Vending Machine using AVR ATmega 8515 Microcontroller.," International Journal of Advanced Research in Computer Science, vol. 3, no. 3, May-June 2012.

[13]. M. K. N. K. D. G. S. G. K. D. Amrita Soni, "Arduino based Reverse Vending Machine.," International Research Journal of Engineering and Technology, vol. 7, no. 8, August 2020.

[14]. B. S. Ana Monga, "Finite State Machine based Vending Machine Controller with Auto-Billing Features.," International Journal of VLSI design & Communication Systems, vol. 3, no. 2, April 2012.

[15]. N. A. B. Nilesh C. Dhobale, "PLC based automation of multiple fluid vending machine".

[16]. D. N. T. Dimple Thakwani, "PLC Based Change Dispensing Vending Machine using image processing technique for identifying and verifying currency.," International Research Journal of Engineering and Technology, vol. 3, no. 11, November 2016.

[17]. A. Singh, "Touch Screen Based Automated Medical Vending Machine," –International Journal for Innovative Research in Science & Technology, vol. 1, no. 11, April 2015.

[18]. M. M. M. Sarvesh Pandey, "IOT Based Smart Automatic Juice Vending Machine.," International Research Journal of Engineering and Technology, vol. 6, no. 11, November 2019.

[19]. A. Kar., "RFID - Based Automatic Ration Vending Machine To Avoid Corruption And Malpractices at



Ration Shops," International Research Journal of Engineering and Technology, vol. 5, no. 5, May 2018.

[20]. R. S. D. M. N. S. L. M. Shweta Dour, "Vending Machine using 8051 microcontroller.," International Journal of Advanced Research inScience and Engineering., vol. 6, no. 5, May 2017.