SOFTWARE REQUIRMENTS SPECIFICATION

(SRS)

For the

Vending Machine Controller

(VMC)

Prepared for

CS361 – Software Engineering I

Winter 2020

Eastern Oregon University

Prepared by

Keith Bittner

David Hippe

Table of Contents

1 Executive Overview (David)

1.1 Project Overview

This project is designed to work as a team of two (Keith Bittner; David Hippe); learn and create project documentation; and create a working prototype based on project documentation.

1.2 Purpose and Scope

The prototype designed from this project is intended to simulate a customer's ability to purchase a product from a Vending Machine using both paper and coin currency. Additionally, this prototype will simulate a service technicians' ability to commission/decommission the Vending Machine, provide inventory, make cash collections, and change product prices.

1.3 Product / Service Description

1.3.1 Product Context

This prototype will meet the minimum general requirements of a Vending Machine Controller. Input will be accepted from users and the Vending Machine will produce a product and/or change.

1.3.2 User Characteristics

Users will include customers and service technicians.

1.3.3 Assumptions

This prototype will simulate a Vending Machine Controller's software/firmware.

2 Non-Functional Requirements (Keith)

2.1 Hardware

This section covers the hardware that will communicate within the Vending Machine.

Vending Machine Controller (VMC)	Communicates with all electronic devices contained in the Vending Machine.
Textual Display Screen	Displays messages sent from the electronic devices in the Vending Machine to the user.
Product Selector	Allows a user to select a desired product.
U.S. Coin Acceptor / Coin Repository / Change Dispenser	Accepts valid currency, stores currency, and dispenses currency.
Change Return Button	Returns equal amount of change based on currency inserted.
Coin Return Tray	Allows a user to receive change from purchase, returned invalid currency, and returned currency from cancelled transactions.
U.S. Bill Acceptor	Accepts valid currency and stores currency.
Product Dispenser	Dispenses a user's desired product selection
Numerical Keypad	Allows for numerical input such as a technician's access code or change in a product's price.

2.2 Software APIs

This section outlines the Application Programming Interfaces (APIs) associated with the hardware used to communicate with the VMC.

Component:	Textual Display Screen
Method	**Description**
showDisplay()	Displays a message through the VMC from a hardware device for a user to see visually.

Component:	Product Selector
Method	**Description**
isSelected()	Identifies the specific product selected by a user.

Component:	U.S. Coin Acceptor / Coin Repository / Change Dispenser
Method	**Description**
isValid()	Check's if a user's currency input is valid.
isFull()	Check's to see if the Coin Repository is full.
isLow()	Check's to see if the Coin Repository has > 5 of each denomination.
dispense()	Dispenses the appropriate amount of change to a user.

Component:	Change Return Button
Method	**Description**
isCancelled()	Cancels a transaction and returns change equal to the currency inserted by a user.

Component:	Coin Return Tray Sensor
Method	**Description**
hasDispensed()	Determines if a coin has dispensed.

Component:	U.S. Bill Acceptor
Method	**Description**
isValid()	Checks if a user's currency input is valid.
isFull()	Checks to see if the Bill Acceptor is at maximum capacity.

Component:	Product Dispenser
Method	**Description**
isEmpty()	Checks to see if a product selection is available.
hasDispensed()	Checks to see if a product was dispensed successfully.

Component:	Numerical Keypad
Method	**Description**
isPressed()	Checks which numerical button was pressed.

2.3 Software Design

The software for the VMC was designed in keeping with coding rules and guidelines. It should be assumed that using the Python computer language, that this software is safe and secure allowing the VMC to be programmed with it. The prototype built using this software should work reliably through rigorous testing. By using the Python language, the software itself will have the flexibility to be modified and/or added to as different features may be added and/or removed. The final software designed can be introduced into any VMC.

3 Functional Requirements

3.1 Customer Use Case(s) (Keith)

3.1.1 Description

The Customer Use Case provides the capability to purchase products for the customer.

3.1.2 Characteristic of Activation

The Customer Use Case is initiated by the customer.

3.1.3 Use Case(s)

Use Case 1: Purchase product

Pre-Condition: Vending Machine must not be open and/or in service

Actor (Customer)	System
Select product.	
	Display product price.
Insert currency.	
	Receive and validate currency.
	Monitor, calculate, and display accumulative amount inserted.
Reselect product.	
	Dispenses product.
	Dispenses change.
End of use case.	

Alternate: Customer cancels transaction

Actor (Customer)	System
Presses Change Return.	
	Return inserted currency.
End of use case.	

Exception: Product not available

Actor (Customer)	System

Select product.	
	Check if product available.
	Display “Sold Out”
	Wait for product selection.
Proceed to use case 1.	

Exception: Invalid currency

Actor (Customer)	System
Insert currency.	
	Validate currency.
	Return invalid currency.
Proceed to use case 1.	

Exception: Insufficient change available

Actor (Customer)	System
	Attempt to dispense change.
	Display “Exact Change Only”.
End of use case.	

3.2 Service Technician Use Case(s) (David)

3.2.1 Description

The Service Technician Use Case provides the capability to access the Vending Machine Controller to provide service and maintenance.

3.2.2 Characteristic of Activation

The Service Technician Use Cases are initiated by the service technician.

3.2.3 Use Case(s)

Use Case 1: Enter technician mode

Actor (Service Technician)	System
Enter code to enter technician mode.	
	Display “Entered Technician Mode”
Proceed to use cases 2 – 5.	

Exception: Incorrect code to enter technician mode

Actor (Service Technician)	System
Enter incorrect code to enter technician mode.	
	Display error message.
Proceed to use case 1.	

Use Case 2: Add/Remove inventory

Pre-Condition: Must be in technician mode

Actor (Service Technician)	System
Update inventory.	
	Update inventory info.
Proceed to use cases 2 – 5.	

Use Case 3: Add/Remove cash

Pre-Condition: Must be in technician mode

Actor (Service Technician)	System
Update cash available.	
	Update cash info.
Proceed to use cases 2 – 5.	

Use Case 4: Update prices

Pre-Condition: Must be in technician mode

Actor (Service Technician)	System
Select inventory to update price.	
	Prompt new price for selected inventory.
Input new price for selected inventory.	
	Update price info.
Proceed to use cases 2 – 5.	

Exception: Request inventory item that doesn't exist.

Actor (Service Technician)	System
Enter incorrect inventory selection.	
	Display error message.
Proceed to use case 4.	

Use Case 5: Exit technician mode

Actor (Service Technician)	System
Enter code to exit technician mode.	
	Display "Exited technician mode".

Exception: Incorrect code to exit technician mode

Actor (Service Technician)	System
Enter incorrect code to exit technician mode.	

	Display error message.
Proceed to use case 5.	