Contents

[1. Purpose 3](#_Toc137991535)

[1.1. Intended Audience 3](#_Toc137991536)

[1.2. Intended Use 3](#_Toc137991537)

[1.3. Scope 3](#_Toc137991538)

[1.4. Definitions and Acronyms 3](#_Toc137991539)

[2. Overall System Description 4](#_Toc137991540)

[2.1. Use Case Diagrams 4](#_Toc137991541)

[2.2. System Architecture 5](#_Toc137991542)

[2.3 Functional Requirements 6](#_Toc137991543)

[2.3.1. Startup and Main Menu 6](#_Toc137991544)

[2.3.2. Selection of drinks 7](#_Toc137991545)

[2.3.3 Money counting and Payment 9](#_Toc137991546)

[2.3.4. QR Code Transaction System (online) 11](#_Toc137991547)

[2.3.5 Drink Dispensing + Display 13](#_Toc137991548)

[2.3.6 Burglar Detection System 14](#_Toc137991549)

[2.3.7. Maintenance Services 16](#_Toc137991550)

[2.4 Non-Functional Requirements 18](#_Toc137991551)

[2.4.1 Power management 18](#_Toc137991552)

[3. Software Architecture 19](#_Toc137991553)

[3.1. Static Software Architecture 19](#_Toc137991554)

[4. Additional notes 20](#_Toc137991555)

# Purpose

This SRS document describes the System Requirements and Software Design for an IoT Smart Vending Machine.

## Intended Audience

The target audience of this document shall be for System and Software Engineers working on the development, testing, and deployment of this project.

## Intended Use

The SRS defines the overall System Architecture and Requirements as well as the Software Architecture and Design. This document is also contains the definition of the System Requirements which shall be used as the input for System Test cases and Software Unit Test cases.

## Scope

This document includes the system architecture of the product, use case diagrams, as well as requirements for the smart vending machine. It also includes software architecture, and requirements for perform system, integration, and unit testing.

## Definitions and Acronyms

Stock Array – Remaining Stock of Drinks  
Layout Array – Allocated slots of the drinks

# Overall System Description

## 2.1. Use Case Diagrams

## A picture containing diagram, text, line Description automatically generated 2.2. System Architecture

The System Architecture of the project is as stated in the diagram below.  
“requirements” shall refer to a group/section of requirements and “requirement” shall refer to a single requirement.

A picture containing text, screenshot, font, parallel

Description automatically generated

## 2.3 Functional Requirements

### 2.3.1. Startup and Main Menu

|  |  |
| --- | --- |
| REQ\_ID | Requirement |
| REQ\_1.1 | Console displays message “Select state [r]un , [c]onfig :”, for launch modes. A timeout of 5s will cause the program to default to run.  This is to be in the main shell script file. |
|  | When Config [c] command is inputed,  REQ\_1.2 displays idle message, for config  (Implemented in Shell script)  L1: “ Enterin Config…”  L2 :” [n]etwork , [w]ebsite , [s]tock ”  (for network) inputted  REQ\_1.3 > Show current connection with internet via ifconfig.  (for stock) inputted  REQ\_1.4 > Array showing reference codes of items in their respective details are printed to console. Details include, but may not be limited to, price, slot, stock, redeem codes. Printing in a serialized for is acceptable as this is only viewed by technicians.  REQ\_1.5 is unused.  REQ\_1.6 > Pi returns back to idle config state (REQ\_1.1) upon finishing the previous command  REQ\_1.7 > “/” returns pi back to initial idle message ( REQ\_1.1) |
|  |  |

2.3.2. Selection of drinks  
For physical implementation:

|  |  |
| --- | --- |
| REQ 2.0 | The selection of drinks shall follow the flowchart in Figure 2.1. Requirements are defined in the flowchart as well. |

A screenshot of a computer

Description automatically generated

Figure 2.1. Flowchart for physical implementation (# button is used as a substitute for enter. \* is used for online)  
  
For online implementation:

|  |  |
| --- | --- |
| **REQ\_ID** | **REQUIREMENT** |
| 2.1o | Website fetches and displays stock and details of each different drinks (updated whenever user opens the website) |
| 2.2o | User selects their drink and website moves to payment screen |
| 2.3o | After successful payment, a QR code is generated and the user can scan the QR at the machine to redeem drink.  This is discussed in section 2.3.4. |
| 2.4o  [OPTIONAL] | An orders management system can be implemented on the website to show the user’s current orders |

### 2.3.3 Money counting and Payment

The vending machine shall provide options for payment and counting/validation of money deposited for purchase of a drink. This shall be made available physically at the machine, or remotely via an online website/app.

Payment shall be made after the steps in requirement 2, where the user has selected their preferred drink.

The detail of each implementation is shown below:

#### 2.3.3.1. Physical implementation

|  |  |
| --- | --- |
| **REQ\_ID** | **Requirement** |
| 3.1 | The payment interface shall be made available to the user after selection of their drink, along with activation of the sensors (RFID) |
| 3.2 | The user shall be shown the amount to be paid before payment, and options to pay(Yes) or leave (No) |
| 3.3 | The user shall be able to pay via credit/debit card which shall be read via the RFID card reader |
| 3.4 | The user shall be presented with an option to retry and an option to cancel if payment is unsuccessful (going back to the screen on REQ3.2 is one possible implementation) |
| 3.5 | The flow of payment is to be executed as per Figure 3. |

A picture containing text, diagram, screenshot, design

Description automatically generated

Figure 3.1. Flowchart for payment (physical)

#### 2.3.3.2. Online implementation

|  |  |
| --- | --- |
| **REQ\_ID** | **Requirement** |
| 3.1o | The payment interface shall be made available to the user after selection of their drink on the website/app. |
| 3.2o | The user shall be shown the amount to be paid before payment |
| 3.7 | The user shall be able to pay via credit/debit card by entering the card details |
| 3.4o | The user shall be presented with an option to retry and an option to cancel if payment is unsuccessful |
| 3.8 | If payment is successful, the drink shall be reserved for 24 hours; after which it will be released to existing stock if not collected. |
| 3.9 | If payment is successful, the user shall be redirected to QR generation. |
| 3.10 | The flow of payment is to be executed as per Figure 3.2. |

A diagram of a payment process

Description automatically generated with low confidence

Figure 3.2. Flowchart for payment (online)

### 2.3.4. QR Code Transaction System (online)

#### 2.3.4.1. QR Generation (remote counterpart)

The online/remote counterpart shall include a QR generation capability after the payment of the drink, for the collection at the vending machine.

|  |  |
| --- | --- |
| **REQ\_ID** | **Requirement** |
| 4.1 | The QR code shall be able to convey the purchase information and proof of payment to the vending machine, through any means. |
| 4.2 | The QR code shall be for one time use; any subsequent use of the QR code shall be invalid. |
| 4.3 | The program shall follow the flowchart in Figure 4.1 |
| 4.4o | If an orders management system is added, the QR codes shall be accessible from the orders. |

A picture containing text, diagram, screenshot, design

Description automatically generated

Figure 4.1. The flowchart for QR generation

#### 2.3.4.2. QR Detection (vending machine counterpart)

|  |  |
| --- | --- |
| **REQ\_ID** | **Requirement** |
| 4.4 | The machine shall present an option for the user to scan QR codes. |
| 4.5 | The machine shall redirect the user to collection/dispensing if the payment is verified. |
| 4.6 | The machine shall be able to scan for QR codes with its camera, retrieving its value |
| 4.7 | The machine shall not redirect to collection if payment is not successful. Error messages can be modified if they cannot fit the screen of the LCD. The implementation in production uses a string “Invalid!” which is deemed acceptable by this requirement. |
| 4.8 | The machine shall follow the flowchart in Figure 4.2 |

A picture containing text, diagram, line, origami

Description automatically generated

Figure 4.2. The flowchart for QR detection

### 2.3.5 Drink Dispensing + Display

|  |  |  |
| --- | --- | --- |
| **REQ\_ID** | | **Requirement** |
| REQ\_5.1 | | Database  A database of any format shall hold the drinks information, including, but not limited to:   * Stock(s) * Name * Price * Refcode * Slots * Redemption codes   An example implementation:  ('1','Coca-cola','1.5','1,2,3,4,5,6','3,2,6,4,7,6', "[]"), ('2','Sprite','1.7','7,8,9,10,11','2,5,3,5,6', "[]"), ('3','A&W','1.8','12,13,14,15,16,17','3,2,5,4,2,3,', "[]"), ('4','Fanta\_Grape','1.2','18,19,20,21,22,23','2,5,4,3,2,5', "[]"), ('5','Ice\_Lemon\_Tea','1.8','24,25,26,27,28,29,30','3,2,6,4,3,5,7',"[]"), ('6','Coke\_Zero','1.7','31,32,33,34,35,36,37,38','3,5,2,6,3,4,5,2',"[]"), ('7','7-up','1.4','39,40,41,42,43,44,45','6,6,7,4,3,2,6',"[]"), ("8", "Water", "0.9", "46,47", "1,1", "[]")  The database shall include methods to update and get items from it. |
| REQ\_5.2A | There shall be a dispensing function which can be called from anywhere in the program, including Payment and Online components. | |
| REQ\_5.2B | | When the dispensing functions are called, The display then shows the message "Dispensing. Drink Name [refcode]” |
| REQ\_5.3 | | The gyro will then turn to simulate the dispensing of the selected drink and return to its original position after a few seconds. Gyro refers to the servo motor powering the door. |
| REQ\_5.4 | | Once the dispensing is complete, the display will show the message “Dispensing completed, enjoy your drink!”. |

A diagram of a flowchart

Description automatically generated with low confidence

### 2.3.6 Burglar Detection System

|  |  |
| --- | --- |
| **REQ\_ID** | **Requirement** |
| REQ\_6.1 | The smart vending machine shall be equipped with a potentiometer on the hinge of the door to detect any forceful opening of the vending machine door, ensuring enhanced security measures. |
| REQ\_6.2 | The vending machine will constantly monitor the potentiometer sensor readings in real time to detect changes in the machine's position and detect any forceful movements. |
| REQ\_6.3 | To determine a break-in event, the potentiometer readings will be compared against a predetermined threshold value specifically set to identify forceful break-in attempts.  This threshold value will be carefully calibrated based on the potentiometer 's sensitivity and the expected force required to open the vending machine, ensuring accurate and reliable detection. |
| REQ\_6.4 | Upon surpassing the threshold value, indicating a break-in, the smart vending machine's burglar alarm will be immediately activated.  The burglar alarm will emit a loud and attention-grabbing buzzer sound, designed to deter potential theft or vandalism and alert nearby individuals of the unauthorized access attempt. The camera on the vending machine will also take a picture of the burglar.  This alarm will remain active until the deactivation of it in REQ\_6.6.  The buzzer shall be activated based on the timing diagram below: It will run continuously. |
| REQ\_6.5 | An alert will be sent out to the engineer’s smartphone through email or other means about the break-in. |
| REQ\_6.6 | The buzzer will continue sounding continuously until the engineer’s access code is entered into numeric keypad on the vending machine, causing the buzzer alarm to stop. |

A picture containing text, diagram, plan, screenshot

Description automatically generated

### 2.3.7. Maintenance Services

In order to be able to access the internals of a vending machine, the user must enter a specific code, such in a “key” to be able to access the system of the vending machine.

|  |  |
| --- | --- |
| **REQ\_ID** | **Requirement** |
| REQ\_7.1 | After the user keys in the code, the system will detect if the code is input and registered into the internal database system, the alarm/burglar system will turn off and allow the user access to the internals by displaying the main menu in REQ-1.2.  \*The code must be a specific key for the authentication to work, not the code to access the slots/drinks. |
| REQ\_7.2 | If the system detects an invalid code, the following shall be implemented.  -LCD shall display the following:  Line 1:“Invalid code” |

## 2.4 Non-Functional Requirements

### 

2.4.1 Miscelleneous Extra features which are optional

|  |  |
| --- | --- |
| REQ\_8.1 | The burglar detection system shall also be triggered when the accelerometer senses heavy shaking. |
| REQ\_8.2 | Implement a website status checker in the config screen. |

# 3. Software Architecture

## 3.1. Static Software Architecture

**Application Layer (Physical)**

* hal/ (Hardware Abstraction Layer. <https://github.com/czlucius/raspberry_hal/>)
* .env (Environment variables, added to gitignore)
* .gitignore
* antitheft\_background.py (background program with thread to monitor break-ins)
* breakin.py (program launched when break-in occurs)
* camera.py (camera interfacing functions - all camera functionality is contained here)
* dispense.py (dispensing program, contains functions for dispensing)
* Inventory\_Array.py (SQLite database file, contains functions pertaining data persistence)
* keypad\_interfacing.py (keypad interfacing, orchestration program used to prevent concurrency issues with keypad)
* launch.py (launch file. called with command line arguments from MachV2.sh)
* MachV2.sh (main script. helps to launch Python programs and restarts them when crash/exit with status code)
* mail.py (email program - functions for sending email notifications)
* online.py (online program - launched when user wants to redeem online product)
* payment2.py (payment program (physical))
* requirements.txt (requirements for running the program)
* selection.py (selection program, where users can select drinks)
* statusfile
* test\_mail.py (test for mail)
* utils.py (utility functions)
* Vending.sqlite (vending SQLite database)
* websocket\_client.py (WebSockets client for online)

**Online:**

* **Backend**
  + **Backend is an Express.JS/TS server that**

**Hardware Abstraction Layer**

**(note: only the files that are used are displayed here)**

* hal\_key\_l.py (keypad interfacing file, new)
* hal\_keypad\_new.py (keypad interfacing file, new (2))
* hal\_adc.py (ADC for potentiometer)
* hal\_keypad.py (Keypad for user input)
* hal\_lcd.py (LCD for displaying information)
* hal\_rfid\_reader.py (RFID reader for reading cards)
* hal\_buzzer.py (Buzzer for burglar alarm and dispensing confirmation)
* hal\_keypad\_neo.py (keypad interfacing file, new (3))
* hal\_led.py (LED interfacing program)
* hal\_servo.py (Servo interfacing program)

4. Additional notes



The vending machine shall look like this. There will be multiple slots for the same drink, and the user shall press the relevant refcode for the drink physically. Since the labels are visible, the small LCD will focus on displaying price instead of printing all drink names which will not be feasible. The drinks in the respective slots shall be labelled by the refcode, and the user has to enter the refcode. The physical implementation will also include auto choosing slots.

In the online implementation, the slots will be automatically managed by the system, and a slot with stock will be chosen. The user will just pick a drink from the UI.