SRS Document for Supermarket system

Developed by:

Ming Feng

Jerick

Paven

Chandan

Table of contents

[**Document version 3**](#_d4gy5tz3hxqp)

[**1. Purpose 4**](#_kvoc7j28jg0e)

[1.1. Intended audience 4](#_4itsrgp1i37s)

[1.2. Intended Use 4](#_tcxj2va85f71)

[1.3. Scope 4](#_5lh3admuaf2)

[1.4. Definition and Acronyms 4](#_ert2h58dflck)

[**2. Overall System Description 5**](#_rn3unnq36hj3)

[2.2. System Architecture 6](#_i74spgydsdd9)

[2.3. Functional Requirements 7](#_qrq644uoefs8)

[2.3.1. Database 7](#_c7r97yyeh7cm)

[2.3.2. Kiosk 9](#_82t5rcuuzc8g)

[2.3.3. Website 11](#_tmftejgdvp55)

[2.4. Non-Functional Requirements 12](#_t41befegmzet)

[2.4.1. Ease of Use 12](#_4i20wmvsjwzt)

[**3. Software Architecture 13**](#_rpg38ir0cfau)

[3.1. Static Software Architecture 13](#_jr5gk5r420ti)

[Application Layer 13](#_s5zxt11f9gsr)

# 

# Document version

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Update | Name | Date | Version |
| 1. | Intended audience/use and acronyms | Paven | 9/6/2024 | 0.1 |
| 2. | Use case diagram and software Architecture | Chandan | 9/6/2024 | 0.2 |
| 3. | Database and Kiosk functional requirements + Scope | Jerick | 11/6/2024 | 0.3 |
| 4. | Improved use case diagram | Chandan | 11/6/2024 | 0.4 |
| 5. | Website functional requirements | Jerick | 14/6/2024 | 0.5 |
| 6. | Added flowchart to kiosk functional requirement | Mingfeng | 16/6/2024 | 0.6 |
| 7. | Formatted Document + updated acronym table | Paven | 21/6/2024 | 0.7 |
| 8. | Added flowchart for website functional requirements | Mingfeng | 22/6/2024 | 0.8 |
| 9. | Adjusted requirements for kiosk | Mingfeng | 25/6/2024 | 0.9 |
| 10. | Adjusted use case diagram | Chandan | 26/6/2024 | 0.10 |
| 11. | Backend system  architecture | Paven | 26/6/2024 | 1,0 |
| 11. | SrgUpdated application architecture  arar | Chandan | 15/8/2024 | 1.1 |

# 

# Purpose

## Intended audience

This SRS document describes the system requirements and software design for a Supermarket self-checkout system, with the target audience being system and software engineers working on the development of this project

## Intended Use

The SRS defines the overall system architecture and requirements, as well as the software architecture and design. The document also contains the definition of the system requirements which shall be used ad the input for the system test cases and software unit tests.

## Scope

This document encompasses all specifications relating to the minimum viable product of the Supermarket self-checkout system, and does not include quality-of-life features and other features that will benefit the system but are not necessary to the delivery of the product.

## Definition and Acronyms

|  |  |
| --- | --- |
| Acronym | Description |
| RFID | Radio frequency identification |
| QR code | Quick response code |
| LCD | Liquid crystal display |
| INT | Integer |
| API | Application programming interface |

# 

# Overall System Description

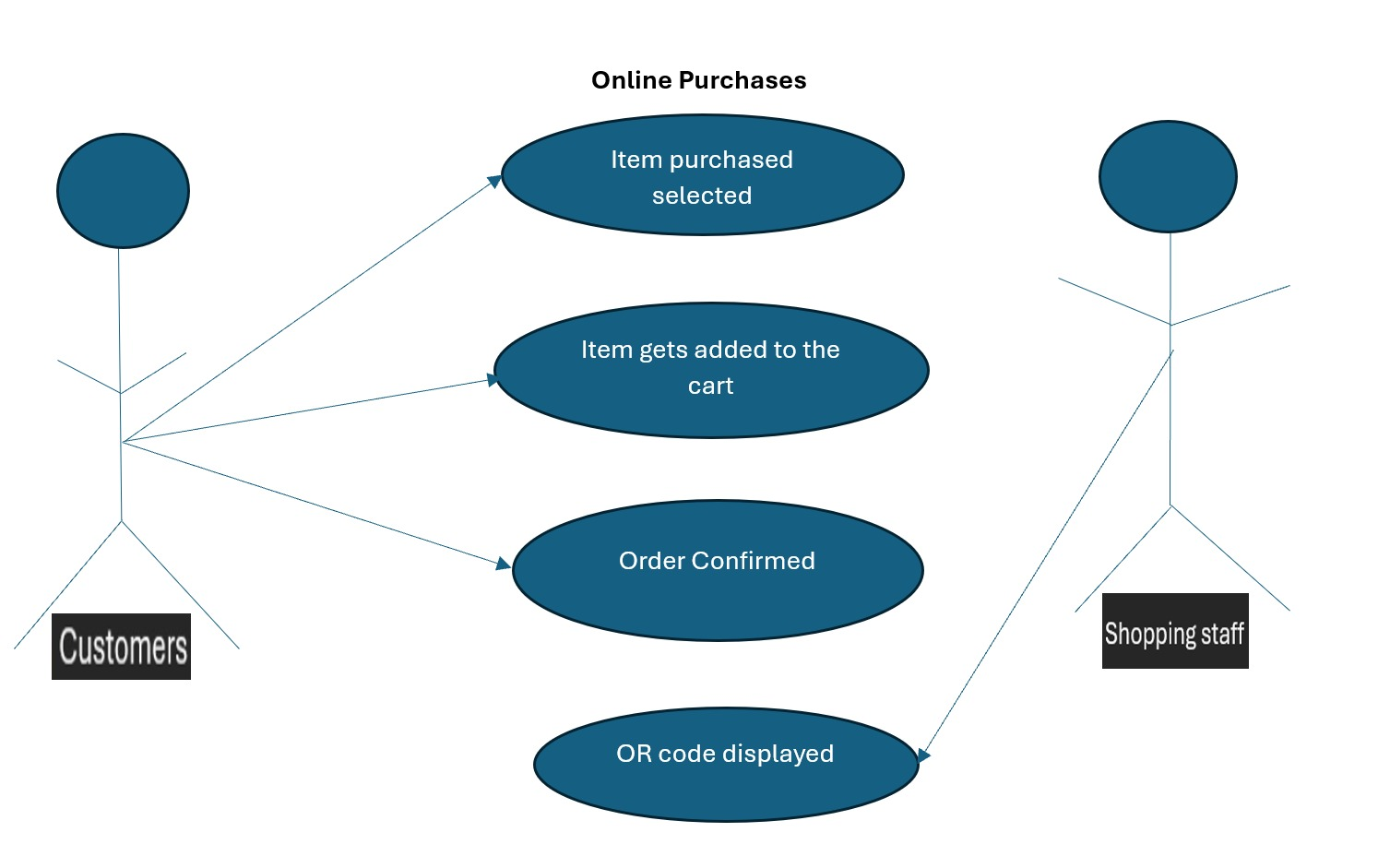
* 1. Use case diagrams****

Figure 2.1.1 Use case diagram of Online Purchases

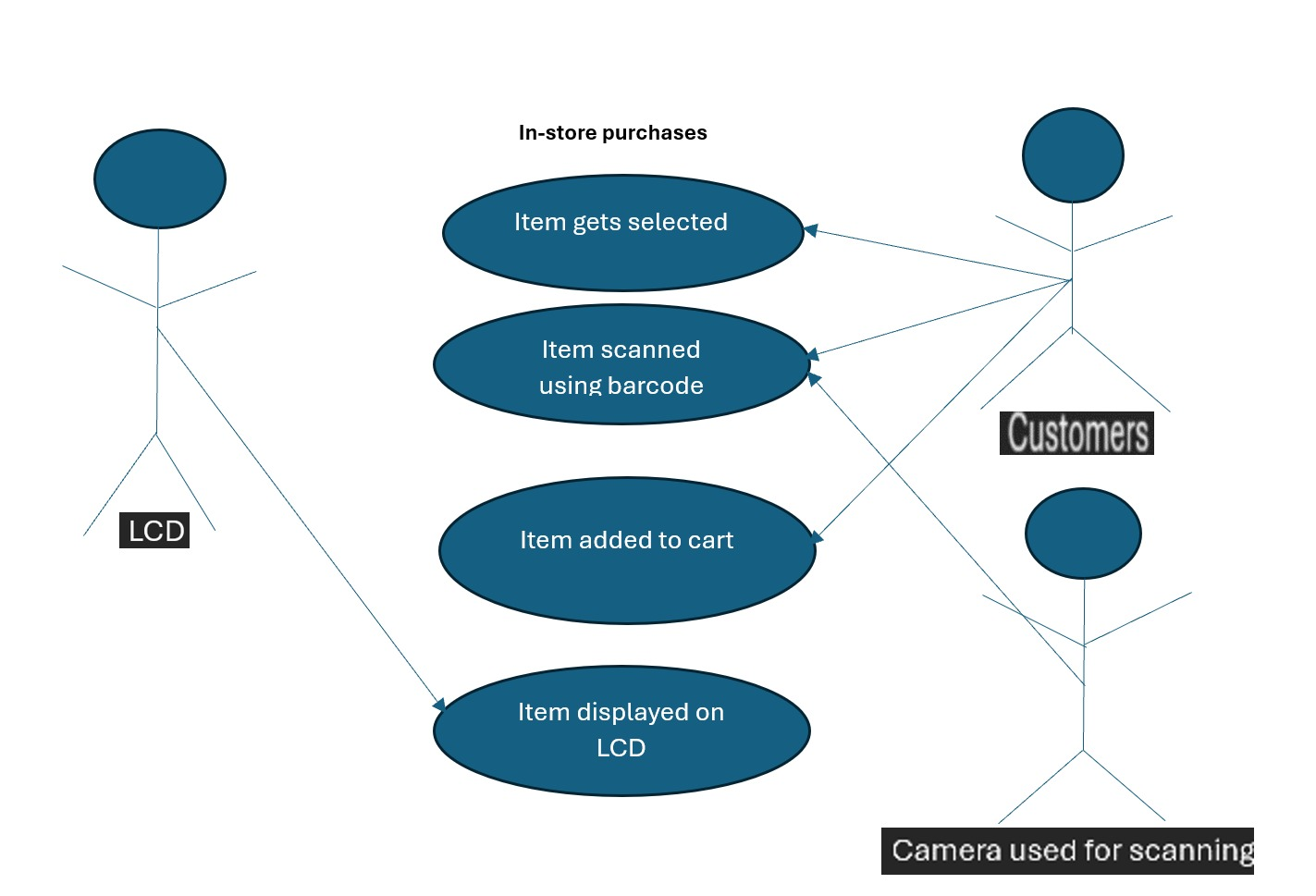
****

Figure 2.1.2 Use case diagram of In-Store purchases

## 

## System Architecture

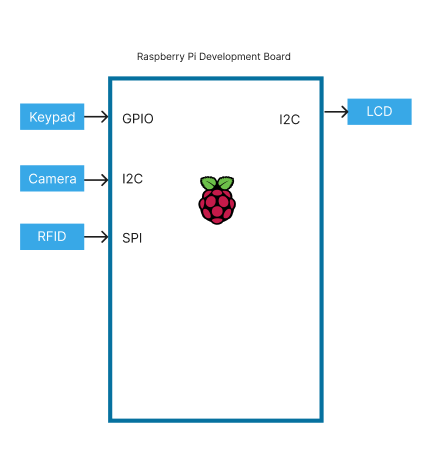


Figure 2.2 System architecture of RPI

## Functional Requirements

### Database

|  |  |
| --- | --- |
| REQ\_ID | Requirement |
| REQ-01 | Create a database called “Products” with the following structure:  {  “id”: (Int),  “name”: (String),  “quantity”: (Float)  } |
| REQ-02 | API for to communicate with the products database defined in REQ-01 they should include the following functionality:   1. Get entire list of products from the database in REQ-01 2. Add product or edit fields of items from database in REQ-01 |
| REQ-03 | Create a database called “Orders” with the following structure:  {  “id”: (int),  “products”:(array of product ID(string) that person has ordered)  “deliver”: (boolean whether it is delivery or collect in-store),  “paid”: (boolean whether it has been paid for)  “collected”: (bool if user has collected item, leave as TRUE)  }  Id is auto generated every time a new order is made |
| REQ-04 | API to communicate with the Orders database as defined in REQ-03 and should include the following functionality:   1. Create order, filling up all fields and appending it to the database 2. Edit “paid” and “collected” fields to reflect if they have paid or collected it. |

### 

### Kiosk

|  |  |
| --- | --- |
| REQ\_ID | Requirement |
| REQ-05 | When the kiosk is powered on, some text should display on the LCD:  ’Welcome!’  Wait for 1s  ’Scan anything to start’ |
| REQ-06 | Import supermarket database and bank database |
| REQ-07 | Able to scan barcode with camera and store it as a variable in code |
| REQ-08 | When Barcode is scanned, Retrieve Product name and price of product using REQ-02. Display product name on the LCD. Then display the item price on line 1 and REQ-08 |
| REQ-09 | If multiple barcodes are scanned, display running total on the line 2 of the LCD. |
| REQ-10 | When “#” is pressed on the keypad, this indicates that the customer has finished scanning all barcodes.  Line 1=’Total cost:’+ total amount |
| REQ-11 | Update supermarket database for products scanned |
| REQ-12 | Display on the LCD:  Line 1=’1: Payment method  Line 2=’1: PayWave, 2: Pin’  Pressing ‘1’ on the keypad would use the “PayWave” method of payment  And pressing ‘2’ on the keypad would use the pin code method of payment |
| REQ-13 | If number pressed is not 1 or 2 loop REQ-12 |
| REQ-14 | Get user to tap their card to see if balance is sufficient to pay |
| REQ-15 | If ‘1’ pressed, display on the LCD:  Proceed with payment and update bank database with new bank balance |
| REQ-16 | If ‘2’ pressed, display on the LCD:  Line1: “Pin Chosen”  Line 2: “Enter Pin”  Wait for the correct pin to enter to authorise payment |
| REQ-17 | After payment has been made show the following message on the LCD:  Line 1: ”Payment successful”  Clear LCD  Line 2: “Thank you”  Line 2: “Have a nice day” |
| REQ-18 | If payment is not successful, display the following message:  Line 1:“Payment unsuccessful”  Line 2=’1: card, 2: atm’  Pressing ‘1’ on the keypad would use the “paywave” method of payment  And pressing ‘2’ on the keypad would use the pin code method of payment |

### 

### Website

|  |  |
| --- | --- |
| REQ\_ID | Requirement |
| REQ-19 | Using REQ-02, display all items available in the supermarket |
| REQ-20 | A search bar is available for easier navigation to products |
| REQ-21 | Users can add items to a cart |
| REQ-22 | Users can remove items from the cart |
| REQ-23 | Users can see subtotal, taxes and overall totals in the cart page |
| REQ-24 | Using REQ-04, users can order items, and select delivery if wanted, else self checkout is by default |
| REQ-25 | If delivery is selected, 3 dollars is added to the total. After that, users will be asked to enter their pin to pay for it. |
| REQ-26 | If self checkout is selected, collected will be set to false, and a qr code will be generated with the order id for the admin to reference and update the collected field when scanned by the admin at the shop. |

# 

## Non-Functional Requirements

### Ease of Use

As the users of the system will be non-technical, the supermarket system should be easy enough to use such that one without technical knowledge should be able to easily navigate through it. This includes both the website and the device itself.

|  |  |
| --- | --- |
| REQ\_ID | Requirement |
| REQ-24 | The website should be clear and organised, with clear labels or icons for all functions |

# 

# Software Architecture

## Static Software Architecture

**The Software architecture defines the various Software components that are developed to realise the implementation of the system requirements.**

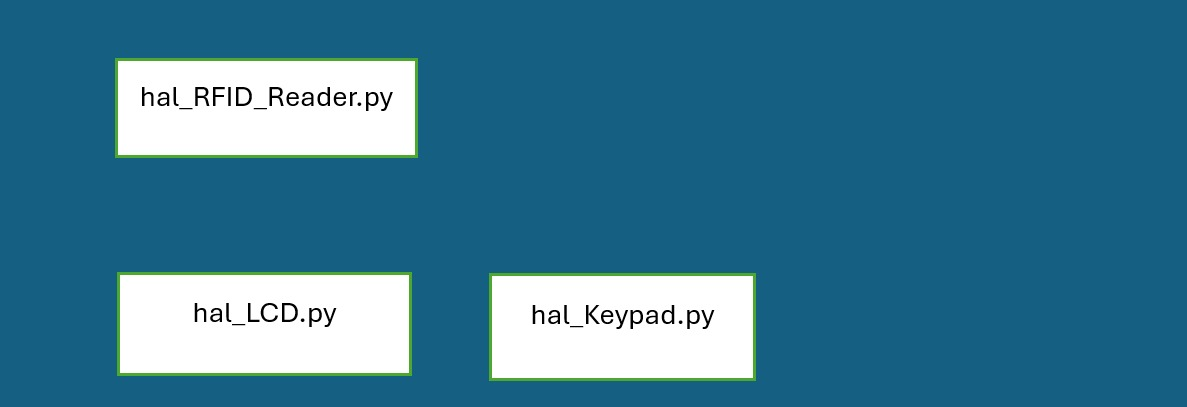
#### Application Layer

**A screenshot of a computer

Description automatically generated**

**Figure 3.1.1: Software architecture of the application layer**

Hardware Abstraction Layer (HAL)

****

**Figure 3.1.2: Software Architecture of the Hardware Abstraction Layer**