# Software Requirements Specification

for

# Pawncore: All-In-One Pawn Management and POS Software (MVP)

**Version 1.0 Approved**

**Prepared by Braden Cariaga,**

**Atef Alhassan,**

**Maverick Hope**

**Syzco Software Solutions**

**<Insert Finish Date Here>**

# Table of Contents

# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| | | | |
| | | | |

# 1. Introduction

## 1.1 Purpose

The product for this document is Pawncore: The all-in-one pawn management and point of sale software. The scope of this project is to create a minimum viable product (MVP) for a pawn management and point of sales system. This document will only cover basic requirements for the modules of customers, pawn/buy, point of sale, and inventory management. As the system grows, so will this document; therefore, future revisions should split the modules into separate documents.

## 1.2 Document Conventions

Some requirements are higher priority than other requirements. Colored font indicating the order of priority from highest to lowest are as follows: red, orange, green. All functional requirements are italicized.

## 1.3 Intended Audience

The intended audience of this document are the developers of the system, project managers, and project advisors. This SRS describes the basic functionality required for a Minimum Viable Product (MVP) for Pawncore: The All-In-One Pawn Management and Point of Sales (POS) System.

## 1.4 Product Scope

*<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>*

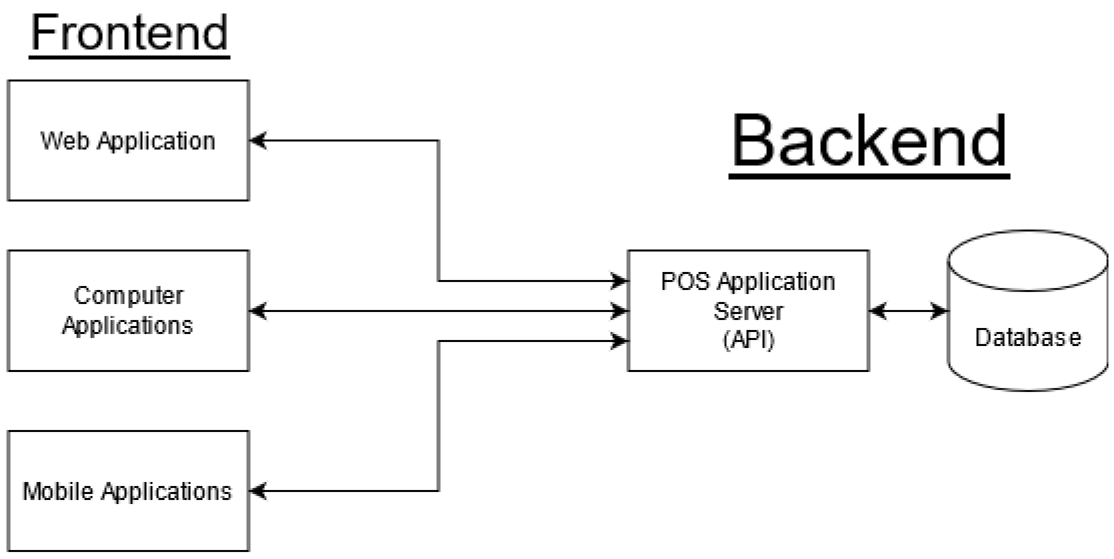
*Come Back to This at End*

## 1.5 References

*There are no applicable references.*

# 2. Overall Description

## 2.1 Product Perspective



This product will be a standalone, self-containing system with no integrations to other systems. The product will utilize the client-server model to provide two separate web-services. The client service will be the main entry point for user interaction with our system.

## 2.2 Product Functions

*<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>*

## 2.3 User Classes and Characteristics

*<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>*

## 2.4 Operating Environment

The processes will be developed using the NodeJS (v14.9.0) runtime environment. The system will utilize the Strapi (v3.6.3) open source content management system as our backend API. The database will be MariaDB for RasberryPi (v10.3.23). The frontend will be TBD (NextJS or ReactJS).

For developmental use, the source code will be hosted on GitHub. The code may be cloned and deployed locally with *.env* defined event variables.

For production use, the system will be hosted on a RaspberryPi running 5.4.51-v7l GNU/Linux. The server will use Nginx for routing/proxying. The backend will be hosted on port 3003. The frontend will be hosted on port 3002.

### 2.5 Design and Implementation Constraints

*<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>*

### 2.6 User Documentation

*No user documentation for the MVP.*

### 2.7 Assumptions and Dependencies

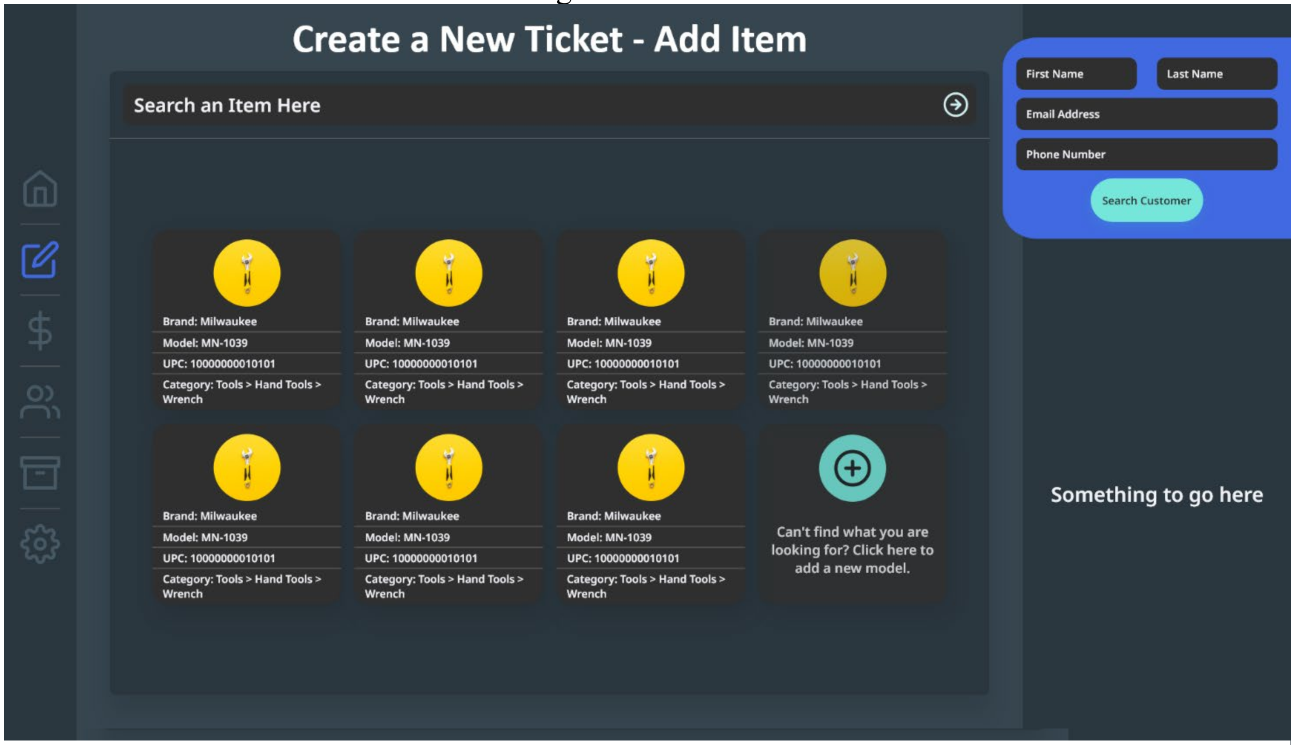
*No assumptions or dependencies for the MVP.*

## 3. External Interface Requirements

### 3.1 User Interfaces

The user interface (UI) should be an easy to learn and practical experience. The UI must follow the established brand guide and color scheme.

The user interface has been mocked with light detail below:



Brand: Milwaukee
Model: MN-1039
UPC: 10000000010101
Category: Tools > Hand Tools > Wrench

Brand: Milwaukee
Model: MN-1039
UPC: 10000000010101
Category: Tools > Hand Tools > Wrench

Brand: Milwaukee
Model: MN-1039
UPC: 10000000010101
Category: Tools > Hand Tools > Wrench

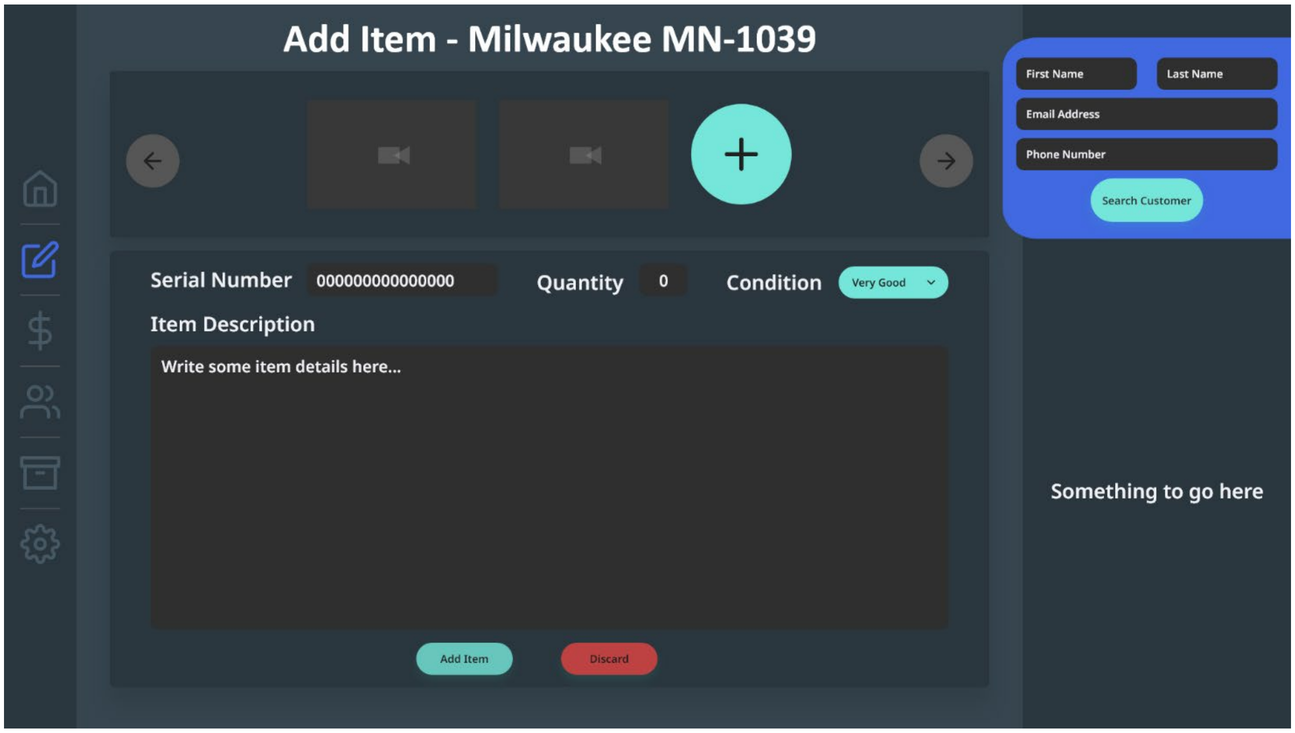
Brand: Milwaukee
Model: MN-1039
UPC: 10000000010101
Category: Tools > Hand Tools > Wrench

Brand: Milwaukee
Model: MN-1039
UPC: 10000000010101
Category: Tools > Hand Tools > Wrench

Brand: Milwaukee
Model: MN-1039
UPC: 10000000010101
Category: Tools > Hand Tools > Wrench

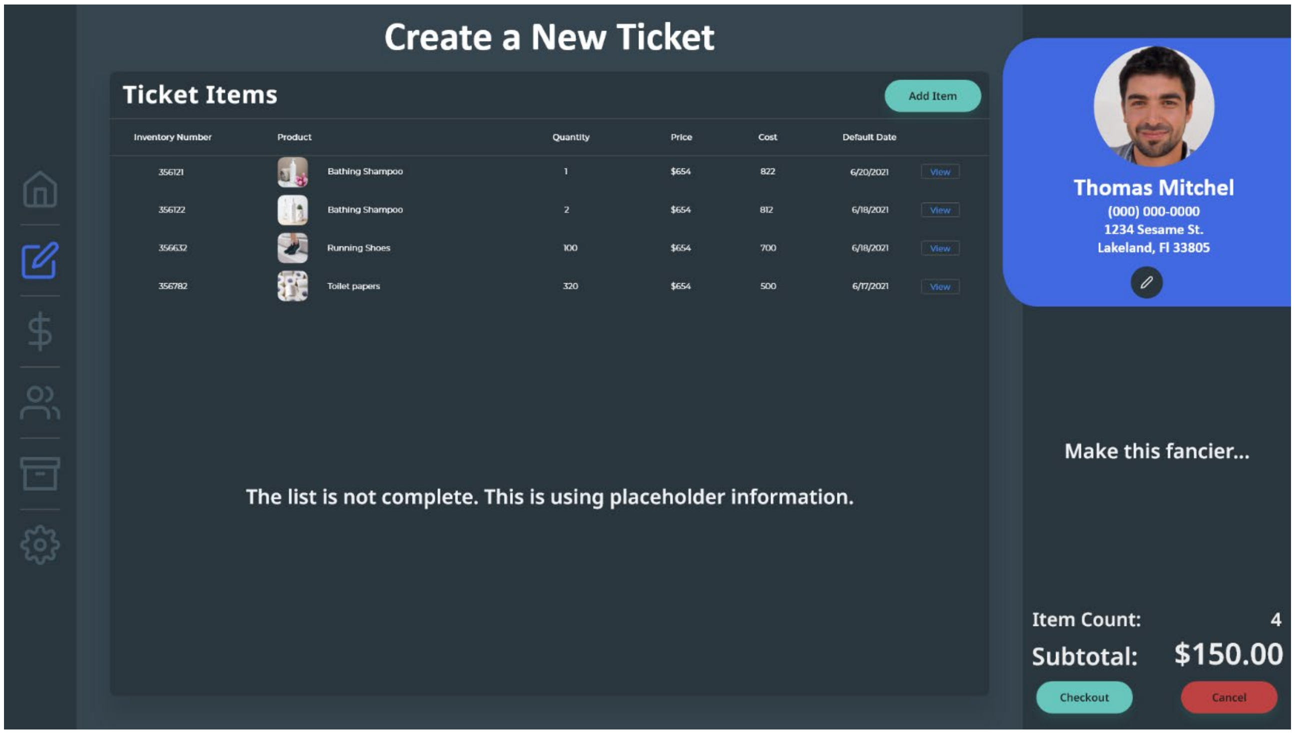
Brand: Milwaukee
Model: MN-1039
UPC: 10000000010101
Category: Tools > Hand Tools > Wrench

*Figure 3 – User Interface Create Ticket Item Search: This is an idea of what the user will see as they search for an item they want to add to a ticket.*



Serial Number 000000000000000
Quantity 0
Condition Very Good
Item Description
Write some item details here...

*Figure 2 – User Interface Add Item Specifics: After you have selected a model, the user can add specific details about the item. (No customer is selected at this time)*


Create a New Ticket

Ticket Items

| Inventory Number | Product | Quantity | Price | Cost | Default Date | |
|---|---|---|---|---|---|---|
| 356121 | Bathing Shampoo | 1 | $654 | 822 | 6/20/2021 | View |
| 356122 | Bathing Shampoo | 2 | $654 | 812 | 6/18/2021 | View |
| 356632 | Running Shoes | 100 | $654 | 700 | 6/18/2021 | View |
| 356782 | Toilet papers | 320 | $654 | 500 | 6/17/2021 | View |

The list is not complete. This is using placeholder information.

Thomas Mitchel
(000) 000-0000
1234 Sesame St.
Lakeland, Fl 33805

Make this fancier...

Item Count: 4
Subtotal: $150.00

*Figure 4 – User Interface Create New Ticket Main: The user will have this view after they have added some items to the ticket. Currently, the view is not complete in its entirety, but gives an idea of what it should look like.*

## 3.2 Hardware Interfaces

There are no hardware interfaces for the MVP. Due to the nature of this software being a web application, any person who has a browser supporting JavaScript will be able to access and utilize our system.

## 3.3 Software Interfaces

*<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>*

*Come back at the end.*

## 3.4 Communications Interfaces

The system will be accessible by any client browser which supports JavaScript. Through the frontend, the user will interface with the backend through HTTP GET, PUT, and POST. There will be security authentication implemented on future versions but is not necessary on the MVP.

# 4. System Features

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

The system features are organized by the modules in which they are contained. The features are colorized by the severity as described in the document conventions (section 1.2).

## 4.1 System Feature 1

*<Don't really say "System Feature 1." State the feature name in just a few words.>*

### 4.1.1 Description and Priority

*<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>*

### 4.1.2 Stimulus/Response Sequences

*<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>*

### 4.1.3 Functional Requirements

*<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>*

*<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>*

REQ-1:
REQ-2:

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

*There are no performance requirements for the MVP.*

## 5.2 Safety Requirements

*There are no safety requirements for the MVP.*

### 5.3 Security Requirements

*There are no security requirements for the MVP.*

### 5.4 Software Quality Attributes

The biggest characteristics that should represent our product is its usability and reliability. It is usability for allowing the user to complete a task effectively and efficiently. The reliability for sure, because want our users to be able trust us and rely on us for both our service and their security of information. This system should also be easily adaptable as there are many features to be added on in the future. Our user interface should have a high focus on ease of use, as opposed to an ease of learning. The API interface should be highly scalable and selectively open for 3rd party use.

### 5.5 Business Rules

No user will be able to perform the delete operation on any table, except:

- Retail Locations
- Ticket Locations

## 6. Other Requirements

*There are no other requirements, currently.*

## Appendix A: Glossary

| Abbreviation | Meaning |
|---|---|
| **MVP** | Minimum Viable Product |
| **TBD** | To Be Determined |
| **API** | Application Programming Interface |
| **HTTP** | Hypertext Transfer Protocol |
| **UI** | User Interface |
| **POS** | Point of Sales |

## Appendix B: Analysis Models

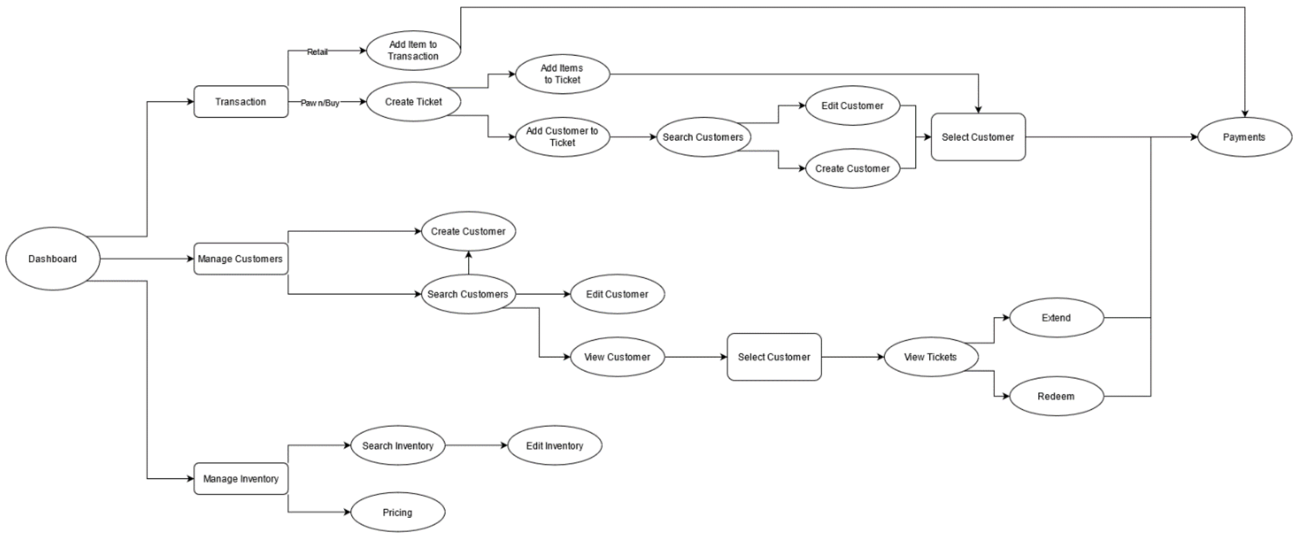


*Figure 5 – User Flow Diagram: This diagram shows how the user would traverse our system and accomplish the basic requirements.*

## Appendix C: To Be Determined List

1. Frontend Framework
2. User Documentation
3. Assumptions and Dependencies
4. Hardware Interfaces
5. Performance Requirements
6. Safety Requirements
7. Security Requirements