Software Requirement Specification

AutoPantry

# Table of Contents

[**Table of Contents**](#_xw59vwjeoylw) **2**

[**1. Introduction**](#_cveg2cts165m) **3**

[1.1 Purpose](#_c8poqh4208jp) 3

[1.2 Document Conventions](#_no8c8u3rk31) 3

[1.3 Intended Audience and Reading Suggestions](#_fuy1pv266ff6) 3

[1.4 Product Scope](#_noj2rxk5kszb) 4

[1.5 References](#_krdqzdlo9avl) 4

[**2. Overall Description**](#_8okg26pide30) **4**

[2.1 Product Perspective](#_4kajyke2mfum) 4

[2.2 Product Functions](#_ir4x4r3cioe0) 4

[2.3 User Classes and Characteristics](#_hnlv2x13cqop) 5

[2.4 Operating Environment](#_dsomea9kzaga) 5

[2.5 Design and Implementation Constraints](#_hf35lug7vjc4) 6

[2.6 User Documentation](#_1chim11loezr) 6

[2.7 Assumptions and Dependencies](#_9di14oxnypkv) 6

[**3. External Interface Requirements**](#_v4asdk6rcjc7) **6**

[3.1 User Interfaces](#_qm84sbbpjki) 6

[3.2 Software Interfaces](#_10hyfphhk48g) 7

[3.3 Communications Interfaces](#_29mk9clvw6ki) 7

[**4. Use Cases**](#_823a9d44u2zk) **7**

[4.1 Online Grocery Shopping [GSHP]](#_e9js9799xthf) 7

[4.2 Login [LOGIN]](#_1r7zyqnd4o7y) 8

[4.3 Recipe Creation[RPCR]](#_7q4tdbbr1c80) 9

[4.4 Pantry Inventory [PINV]](#_p1peg7z1ws58) 11

[**5. Nonfunctional Requirements**](#_n3woc2li9er5) **12**

[5.1 Performance Requirements](#_6ionnf3txy5l) 12

[5.2 Safety Requirements](#_15ap802d7rb9) 12

[5.3 Security Requirements](#_ofv162z42mo) 12

[5.4 Software Quality Attributes](#_cb9do9qngg64) 13

[5.5 Business Rules](#_e806fkpb69xx) 13

[**6. ERD Diagram**](#_we8co3aftqs) **14**

# 1. Introduction

## 1.1 Purpose

This SRS document describes AutoPantry v1.0 in regards to the following components: External Interface, Data Modeling, and Backend Architecture.

## 1.2 Document Conventions

Glossary of Terms

|  |  |
| --- | --- |
| End User | Primary client, the customer seeking a streamlined meal plan |
| User | See End User |
| Vendor | Product Holder, i.e warehouse a Driver obtains Product from |
| Driver | Ordering service Delivery agent |
| Platform | AutoPantry Application and API |
| Product | Food items, possibility for cooking implements (pans, spatulas, etc) |
| Recipe | A collection of Products to be built into a meal and corresponding instructions |
| Menu | A collection of Recipes. Primarily to expedite the ordering process. |
| Pantry | The sum of the products estimated to still be available in a user’s possession. |
| Shopping Cart | A collection of Recipes to be actively ordered, differing from a Menu |
| Stakeholder | Our instructor, Jing Bai |

## 1.3 Intended Audience and Reading Suggestions

Guide is for the developers as both a reference guide and blueprint for implementation, and for the stakeholder as a demonstration that the developers have the project well in hand.

* Section 1 contains an overview of the platform.
* Section 2 provides an overview of the platform implementation.
* Section 3 describes the interfaces provided for accessing this platform.
* Section 4 encapsulates the functional requirements for this platform.
* Section 5 contains the non-functional and miscellaneous requirements of the platform.

## 1.4 Product Scope

This document applies to the AutoPantry v1.0 software application. The Platform facilitates the coordination of recipes and ordering of associated ingredients eliminating the necessity for an End User to evaluate which, and separately purchase, ingredients they require. The Platform offers benefits such as: a comprehensive list of available Recipes, a complete inventory of food Products with competitive prices, secured digital ordering and payment, additional membership features including saved Menus and favorited Recipes for simplified reordering.

The Platform allows users to select Recipes and build a shopping cart based on the selected recipes. The user may also choose to order products directly from the Product Inventory. The user may then choose to log in and save the menu they have created for reuse. Fulfilling a purchase results in a shipment of food products to the End User.

The Platform is provided to the midwestern region within 12 hours shipping from our warehouse in Ann Arbor, Michigan.

## 1.5 References

The references for the above software are as follows:

1. Shipment API <https://www.ups.com/us/en/services/technology-integration/developer-api.page>
2. “Shipt” partial competitor <https://shipt.com/>
3. “Instacart” partial competitor <https://www.instacart.com/>
4. “Postmates” partial competitor <https://postmates.com/>
5. Recipe JSON Schema <https://schema.org/Recipe>

# 2. Overall Description

## 2.1 Product Perspective

The Platform is intended to be a prototype example of an extension to existing services as a stopgap solution, connecting recipes, products, and same-day delivery services. Shipt, Instacart and Postmates currently allow for items to be purchased from Vendors, but the End User needs to source their own meal ideas and plan their order themselves. The Platform intends to fill that gap.

## 2.2 Product Functions

* Create recipes from available products
* Collect recipes and present those collections as weekly Menus
* Facilitate the ordering of Menus
* Browse public Recipes and Menus for expedient ordering
* Some Users permitted to submit Recipes to the public cloud

## 2.3 User Classes and Characteristics

* Busy Users
  + These End Users need the Platform to be able to replace their meal planning as the additional expense is worth the time reclaimed that would otherwise be used shopping at a traditional grocer.
  + These users are our primary demographic and will gain the most use of the Platform
* Curious Users
  + These End Users want to try new things, and want to use the Platform as a means of discovering new foods or recipes
  + These users likely still also shop at traditional grocers
* Creators
  + These End Users are on the platform to share their culinary expertise with the world by uploading their recipes and curated menus for others to use. The creations of these users would be promoted on the Platform for a fee.
  + Ex: Mario Batali’s italian essentials.
* Administrators
  + These are the administrative users for the application, responsible for monitoring and maintaining inventory, recipes, and curated collections.

## 2.4 Operating Environment

The user interaction with the Platform will be device agnostic, using a web browser compliant with the latest version of Google Chrome (Version 77). The web component of the Platform will directly interface with a REST API.

If desired, the Platform REST API could be used to develop sister applications for iOS and Android. The web component of the Platform will be accessible from mobile devices without a native application.

The Platform itself will be hosted via a Docker container, using PHP, hosted via NGINX. The database for the Platform will be stored using MariaDB, a MySQL compliant database server. These will be configured in Alpine Linux containers, although the code will be technically portable to other hosting environments.

The Docker Swarm will be deployed to Amazon Web Services.

## 2.5 Design and Implementation Constraints

The front end will be delivered using Twitter’s Bootstrap as a layout engine and PHP as the server-side compiler. The REST API will be built in PHP as well. The main constraint is building an interface that can be interacted with on both mobile and desktop class devices without the other feeling like a second class experience. Bootstrap thankfully aids in that, allowing interfaces to scale up and down as needed, and ensuring everything is clickable, tappable, and good looking on all platforms.

At scale, a concern with timed operations will arise, with depleting inventories of Vendors. This is not a concern for the v1.0 Release of the Platform, as no external services will be contacted.

Currently, complying with US regulations for data security are the only security concerns, and, with no payment information being stored in the current state of v1.0 of the Platform, this is not a huge issue.

## 2.6 User Documentation

For the initial rollout, the interface design should be suggestive and intuitive to the point of not needing any tutorials or documentation.

A Frequently Asked Questions page will be provided to assist with common issues on the flow of using the Platform.

## 2.7 Assumptions and Dependencies

For the purpose of this prototype deployment, connections to Vendors, Drivers, and Payment Processors will be spoofed to demonstrate the intended use case of the platform rather than placing actual orders. As such, API implementations will be assumed for those platforms and generalized. Further work would be required to make the Platform interface with external services and make food arrive at an address.

# 3. External Interface Requirements

## 3.1 User Interfaces

The User Interface provided will be intuitive and comprehensive. The interface provided will be a web application driven by a REST API and will consist of the following pages and components:

* Landing: Recommended Menus & base navigation
* Browse: The platform’s storefront, for browsing/searching recipes & completed menus.
* Cart: For order confirmation, editing, and Checkout.
* My Menus: A catalog of of all of the menus saved by the User, for convenient ordering.
* Recipe Book: A collection of all of the recipes contained in the User’s menus
* Recipe Designer: A page for Creator Users to design and record recipes.
* Account: Account info, payment, contact, addresses, etc.
* Navigation bar: Provides navigation links to the other pages of the website.
* Login: Login form available from the navigation bar or as a standalone page.
* Help: Contains the FAQ, customer service, and related topics.

As of this initial draft, no particular GUI standard shall be strictly adhered to, beyond QA for general usability.

## 3.2 Software Interfaces

* Alpine Linux 3.8
* ReactJS 5
* CSS3
* Docker Engine 1.13.1
* ECMAScript 5.1
* HTML5
* JavaScript 1.8.5
* MariaDB 10.4.8
* NGINX 1.17.1
* PHP 5.7

## 3.3 Communications Interfaces

* SMS will be used for Order Notifications.
* IMAP will be used for account updates (password resets, etc).
* API and Web traffic will be communicated over HTTPS.

# 4. Use Cases

## 4.1 Online Grocery Shopping [GSHP]

1. Objective: Utilize the platform to fill a cart with food products.
2. Priority: High
3. Source: End User
4. Actors: End User, Platform
5. Flow of Events
   1. Basic Flow
      1. User selects browse from landing page.
      2. User selects products.
      3. User browses to find product by category.
      4. User adds product to shopping cart.
      5. Platform saves shopping cart by user id.
   2. Alternate Flows
      1. Via Recipes
         1. User selects recipes from landing page.
         2. User browses to find an intriguing recipe by category.
         3. User adds recipe to shopping cart.
         4. Platform populates shopping cart with all items in recipe.
         5. Platform saves shopping cart.
      2. Via Search
         1. User searches for a term
         2. Platform matches results in products and recipes.
         3. User selects product or recipe.
         4. Platform populates cart from user selection.
         5. Platform saves shopping cart.
   3. Exception Flows
      1. Out of Stock Exception
         1. User adds items to cart.
         2. Platform returns error that the item is out of stock.
         3. User is unable to complete the recipe.
      2. Unable to find product
         1. User searches for a recipe or product.
         2. User is unable to locate recipe or product in the inventory.
6. Includes
   1. LOGIN
7. Preconditions
   1. Valid internet connection, LOGIN
8. Post Conditions
   1. User will checkout the shopping cart.

## 4.2 Login [LOGIN]

1. Objective: Authenticate with Platform
2. Priority: High
3. Source: End User
4. Actors: End User, Platform
5. Flow of Events
   1. Basic Flow
      1. Unauthenticated User
         1. User navigates to Platform without initially being authenticated
         2. Login Widget presented
         3. User provides authentication details
         4. Verification of credentials
         5. User is granted access to Platform
   2. Alternate Flow
      1. Expired User Authentication Token
         1. User navigates to Platform with expired authentication credentials
         2. Login Widget presented
         3. User provides authentication details
         4. Verification of Credentials
         5. User is granted access to Platform
   3. Exception Flow
      1. Incorrect Credentials Flow
         1. User navigates to Login Widget via “Unauthenticated User” flow or “Expired User Authentication Token” flow
         2. User provides incorrect authentication details
         3. Verification of Credentials with invalid response
         4. User is re-prompted with Login Widget with an error
         5. User provides correct authentication details
         6. Verification of Credentials with Successful response
         7. User is granted access to Platform
6. Includes
   1. N/A
7. Preconditions
   1. Valid Internet Connection
8. Post Conditions
   1. User will authenticate to the Platform and enter another valid flow

## 4.3 Recipe Creation[RPCR]

1. Objective: Creation of Recipes
2. Priority: Medium
3. Source: End User
4. Actors: End User, Platform
5. Flow of Events
   1. Basic Flow
      1. Landing on Recipe designer
         1. User presses ‘New Recipe’ button on the Recipe Book page
         2. User is prompted for a title
         3. User enters title
         4. Recipe designer displayed & session information is logged to the server
         5. User selects Products as ingredients of the recipe
         6. User provides recipe preparation information
            1. See [Json Schema](https://schema.org/Recipe) for examples
         7. User selects the publish option from within the recipe designer
         8. Platform verifies all inputs are valid & necessary fields are filled
         9. The recipe is cataloged in the Database with the public flag set to true
   2. Alternate Flow
      1. Cancel creation
         1. User selects the Cancel option within the recipe designer
         2. User is prompted to confirm
         3. User confirms
         4. Session and edits(entire recipe if it is a new recipe) to the recipe are discarded.
         5. User is sent back to the Recipe Book page.
      2. Pause creation
         1. User selects the save option from within the recipe designer
         2. Recipe is cataloged within the database with the public flag set to false
         3. User is served the recipe book page.
      3. Edit Recipe
         1. User selects an already created recipe
         2. User is presented with recipe designer with the selected recipe’s information preloaded into the fields
         3. User modifies the recipe
         4. User selects the publish option from within the recipe designer
         5. Platform verifies all inputs are valid & necessary fields are filled
         6. Recipe is updated in the database
   3. Exception Flow
      1. Invalid inputs
         1. The user attempts to publish a recipe that fails verification
         2. User is prompted to either fix the issues or save for later
            1. Fix: User is returned to the recipe designer, with problem fields highlighted in red & text explaining the correct inputs
            2. Save: The platform saves the recipe to the database with the public flag set to false
      2. Creation interruption
         1. The user navigates away, closes the tab, attempts to logout, or suffers connection loss while using the recipe designer
         2. User is prompted to Save, Discard, or Cancel logout.
            1. Save: Platform follows the Pause Creation alternate flow
            2. Discard: Platform follows the Cancel Creation alternate flow.
            3. Cancel: The prompt disappears, and the user resumes recipe creation
            4. Default: In case of no response, the Platform follows the Pause Creation alternate flow

1. Includes
   1. LOGIN
2. Preconditions
   1. Creator level membership
3. Post Conditions
   1. Recipe is created and logged in the database with the public flag set to true
   2. Incomplete recipe is logged to the database and public flag is set to false
   3. Recipe appears within the User’s Recipe Book
   4. Published recipes become available for all shoppers.
4. Notes/Issues

## 4.4 Pantry Inventory [PINV]

1. Objective: Platform tracks the products and their quantities the User has ordered in the past in order to prevent the User from double-ordering a product they don’t need more of in order to prepare a recipe.
2. Priority: Medium
3. Source: End User
4. Actors: End User, Platform
5. Flow of Events
   1. Basic Flow
      1. User attempts to add a recipe to their cart
      2. Platform checks User’s PINV against the contents of the recipe
      3. Platform only adds the products the User doesn’t have in their PINV to the cart
      4. Platform adds products ordered by the User to the User’s PINV
   2. Alternate Flow
      1. Upon repeat orders, user confirms whether or not they have prepared the recipes they previously ordered
      2. Platform subtracts all confirmed recipe’s contents from the User’s PINV
6. Includes: LOGIN
7. Preconditions
   1. Logged into User Account
8. Post Conditions: N/A

# 5. Nonfunctional Requirements

## 5.1 Performance Requirements

* Capacity - Website should be available 24/7
* Page should load under 30 seconds in ideal conditions
* Search Queries should return in under 10 seconds to allow page loads to stay within their parameters

## 5.2 Safety Requirements

Due to the nature of being an online grocer, all customers are subject to the possibility of cross contamination between the ingredients we sell. Thus, we must sufficiently notify all customers to the risks therein. Eg: Nuts, meat products, dairy, etc.

Further, do to the transportation of raw foods, the application must be able to determine whether or not the customer’s deliverable address is within range to receive their order within a timely manner, as to not have expired items delivered.

## 5.3 Security Requirements

In holding our user’s payment information within the app, the platform must be up to financial security standards. Further, in holding our customers contact information and addresses, our private policy must be within legal requirements.

The platform will be secured with a minimum 12 character password minimum, with at least 1 capital letter, number, and special character. Password will be hashed using SHA-256 algorithm on the server.

The customer’s full payment information will not be accessible from inside the application; instead, it will only show the last four digits in plain text(For easier identification, in the case of multiple options).

Changing or adding the customer’s payment information, address, login information, or contact information will prompt an email notification, and an optional SMS notification.

Two factor authentication is outside of the scope of the prototype, and will be implemented only if excess time permits.

## 5.4 Software Quality Attributes

Intuitive UI, easy to modify menus, browser agnostic.

As the application will be public facing to the general populace, the user interface must be sufficiently intuitive as to not require any special knowledge. Essentially, it must replicate the design conventions of digital storefronts such as Amazon, ebay, Google Play store, etc.

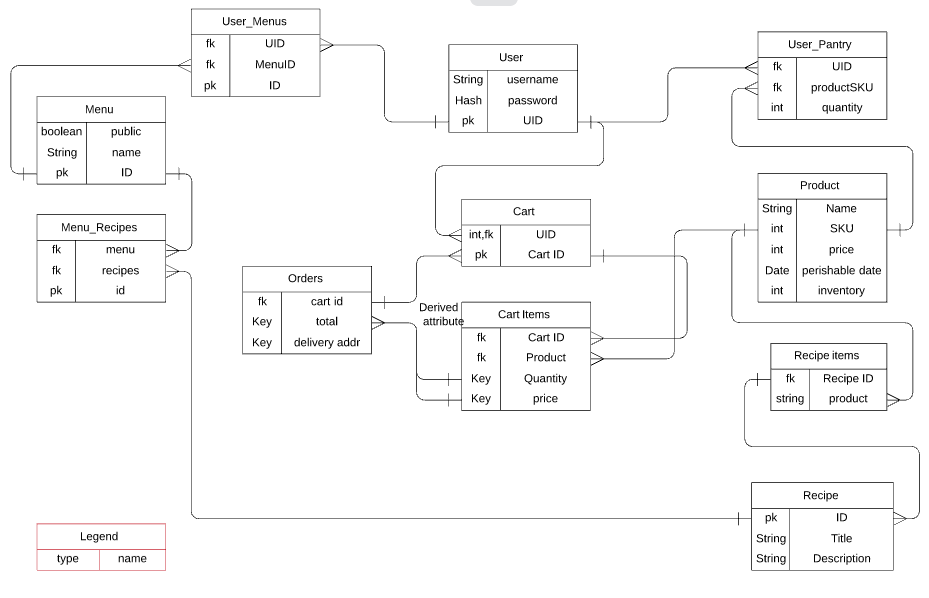
The centerpiece feature of the application, the menus, must be as easy to use as conceivably possible.

All functions will be thoroughly tested.

## 5.5 Business Rules

* User
  + Create Menus
  + Browse Menus
  + Browse Recipes
  + Browse Products
  + Delete Unpublished Menus
  + Publish Menus
  + Create Cart
  + Order Cart
  + Update Account Details
* Creator
  + See “User”
  + Create Recipes
  + Publish Recipes
  + Delete Recipes
  + Delete Menus
* Admin
  + See “User” and “Creator”
  + Modify Inventory
  + Manage Users

# 6. ERD Diagram[[1]](#footnote-0)



1. Subject to change [↑](#footnote-ref-0)