Project Plan

Inventory Management System

IS 663-851 Fall 2022

**Priya Rivera** - pr469@njit.edu

**Sean Kilroy** - sgk4@njit.edu

**Connor Svelling-Pescatore** - cfp4@njit.edu

**John Atkins** - ja727@njit.edu

**Lawrence Yang -** ly23@njit.edu

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 16-10-2022 | 0.1 | Initial Draft | Priya Rivera, Sean Kilroy, Connor Svelling-Pescatore, John Atkins, Lawrence Yang |
| 16-10-2022 | 0.2 | Updated Gantt chart; Proofread | John Atkins |

Table of Contents

[1. Introduction](#_obr71ih1mavt)

[1.1 Foreword](#_vwccuqb2odx3)

[1.2 Background](#_e4xon7gu6sry)

[2. Project Organization](#_rujputrg6ulm)

[2.1 Development Model](#_kda34kl96uhh)

[2.2 Project Team](#_8cx6iigp0md4)

[3. Risk Analysis](#_iiwb08rbfn2s)

[3.1 Risk Analysis](#_9lx9cd8v4kio)

[3.2 Risk Analysis (i)](#_q3n9kuhgk1v1)

[3.3 Risk Analysis Strategy](#_rj03pinzztaw)

[4. Hardware and Software Requirements](#_fct0usunorpl)

[4.1 Software](#_damcquq2spzn)

[Server Software](#_1ixgtfxo5zgu)

[Development Environment](#_wwsqvwxznco5)

[4.2 Hardware](#_tc3dns7bghy1)

[5. Schedule and Work Breakdown Structure](#_re1aqc30kkj1)

# 1. Introduction

## 1.1 Foreword

The purpose of this document is to provide information about the **Inventory Management System (IMS)** project. It will include the project schedules, risks, project team, and the work breakdown structure. This plan will:

* Provide a general product description.
* Identify required resources.
* Identify schedules and activities to be performed.

This product is an inventory-management system for a seller that can be used on desktop and mobile (note, this is not an ecommerce site, per se, but can evolve towards a business-to-business service). It will provide both a direct user interface and API for inventory management.

**Application features:**

* User Interface to enter inventory items and stock: class (accessories, clothing, shoes, jewelry), subclass (jacket, dress, necklace), department (mens, womens), sku, vendor, style, color, size, quantity, lot number, retail price, wholesale price, etc.
* Create an order
* Create invoices and export to PDF
* Move items into order and remove from stock for purchases
* Move items into stock and remove from order for returns
* Provide history by logging changes in stock and associated purchase order and invoice
* Search inventory
* Import and export data via spreadsheets
* Integrate with payment and shipping APIs
* View sales ledger
* Interface for client to view order and shipping information
* Interface for client with payment API

**Stakeholders:** Seller (Small Business Owner), Buyer (Companies/Clients)

## 1.2 Background

Unlike a traditional e-commerce site, this app will cater to a seller who acts as a personal buyer for clients. Therefore the seller acquires inventory, negotiates sales with clients, generates purchase orders, takes payment, and ships. Currently the seller manages inventory in spreadsheets, which is cumbersome to use on mobile devices and makes it difficult to determine the current state of inventory.

There will also be a read-only view for buyers to review order status.

The system needs to provide various interfaces to integrate with the seller’s and buyer’s workflow and processes. For small and medium businesses, this may include manual processes, forms, and spreadsheets.

# 2. Project Organization

## 2.1 Development Model

Agile methodology:

* User Stories to capture requirements
* Work is grouped into Epics, which consist of a set of stories/tasks. Each story/task shall consist of a description of work, an estimate of effort, a definition of done, and method of verification. Method of verification should include an automated test, if applicable.
* Stories/tasks are tracked and prioritized in a project backlog
* Two-week sprints
* Test-driven development
* DevOps philosophy
  + Automated builds
  + Automated deployment
  + Automated testing: unit testing, integration, acceptance testing
  + Continuous Integration/Continuous Deployment

## 2.2 Project Team

The table below lists the members of the Inventory Management System

project team.

| **Project Team** |
| --- |
| Priya Rivera (Product Owner) |
| Lawrence Yang (Systems Engineer and Architect) |
| Connor Svelling-Pescatore (Senior Software Developer) |
| Sean Kilroy (Mid Software Developer) |
| John Atkins (UI Designer) |
| Shannon Pace (QA Manager) |
| Kevin Jones (DevOps Engineer) |

# 3. Risk Analysis

## 3.1 Risk Analysis

| **Risk** | **Affects** | **Description** |
| --- | --- | --- |
| Competing software service | Business, Product | Competing product |
| Integrating with customers workflows and processes | Project, Product | How will the user use the product |
| Staff attrition | Business, Product, Project | How to proceed if key team members leave the organization |
| New software requirements arise | Project, Product | Unknown unknowns |
| Regulatory risk | Business, Product | GDPR and CCPA compliance. These laws define personal data to include name, email address, phone number, purchase history. |
| Degraded service of integrated payment and shipping APIs | Product | Product relies on functioning external APIs to automate the payment and shipping flows. |
| Staff skill mismatch | Business, Product | Mismatch of staff skills and experience with work |

## 3.2 Risk Analysis (i)

| **Risk** | **Probability** | **Effects** |
| --- | --- | --- |
| Competing software service | Moderate | Serious to catastrophic, depending on features and time to market, may have no customers |
| Integrating with potential customers work flows and processes | High | Tolerable. Requirements churn. Can be serious if significant rework is needed. |
| Staff attrition | Very high | Tolerable to serious. If enough people leave at the same time, it could be catastrophic. |
| New software requirements arise | Very high | Tolerable to serious. Rework and schedule delay. |
| Regulatory Risk | Low | Serious. GDPR penalties can be up to 4% of annual revenue. |
| Degraded service of integrated payment and shipping APIs | Very low | Insignificant. Degraded service will introduce minor delays in payment processing and shipping services. |
| Staff skill mismatch | High | Serious. Affects schedule and quality. |

## 3.3 Risk Analysis Strategy

| **Risk** | **Solution / Mitigation** |
| --- | --- |
| Competing software service | Market research to prioritize time-to-market of functionality not present or of functionality where competitors are a poor fit with the current market |
| Integrating with customers work flows and processes | Recruit a variety of actual users to give feedback. Integrate early with prototypes and early releases. |
| Staff attrition | Management one-on-ones. Identify skills and knowledge gaps and arrange for knowledge transfer and training. Knowledge sharing such as mentoring, informal documentation (wiki). Form team with members with overlapping skills. Deferred compensation, bonus, and/or stock options. |
| New software requirements arise | Customer feedback early and often. Prototyping. |
| Regulatory Risk | Process personal information only when there is a business need to do so. Only store personal information for as long as needed. Avoid storing sensitive personal, financial, and/or accounting information. Future features may have tax implications, but currently out of scope. If future functionality does have such implications, segregate into a separate application/product. |
| Degraded service of integrated payment and shipping APIs | Create contracts with external platforms to ensure that an acceptable level of service is always available. If not possible, then ignore risk since the probability is very low and the effect is insignificant. |
| Staff skill mismatch | Identify mismatches while hiring and selecting technologies. Arrange for training, mentoring, and pair development. |

# 4. Hardware and Software Requirements

## 4.1 Software

The selected software should be commonly used for providing software as a service (SaaS) via a web application. Free and Open Source software should be preferred.

### Server Software

The product shall run on server class operating systems. The software shall be containerized and shall be deployed on the chosen cloud server.

The product shall use a database to store, retrieve, and manage data. The database shall work with the chosen programming languages for implementation. The database shall also be compatible with the chosen integration, testing, and deployment software and cloud platform. Common examples are MySQL and PostgreSQL.

### Development Environment

Seat licenses shall be available for each team member for collaboration tools and development management tools. Examples of collaboration tools include Microsoft 365, Slack, Dropbox, Zoom, Confluence, Teams, etc. Examples of development management tools include Jira, Gitlab, Bitbucket, DockerHub, etc.

Source code, containers, and configuration shall be maintained under version control. Version control shall be regularly backed up, off site. The software shall be developed in a version control system(s) that shall integrate with an automated build, testing, integration, and deployment solution(s). This solution(s) should implement DevOps.

The user interface shall use modern web technologies. These shall include modern HTML standards, Javascript, and CSS.

The product should be implemented in a programming language suitable for development web applications, and software as a service, with a preference for Python. Javascript libraries such as React/Vue/Angular may be used for client-side rendering.

User authentication and authorization will be implemented using JWT.

Testing frameworks shall be used to automate testing. Unit testing frameworks shall be suitable for the development language. A suitable framework should be chosen to automate integration and acceptance testing.

## 4.2 Hardware

* 7 Laptops
  + Core i5-i7 12 series or better / AMD Zen 2 CPU or better
  + 16GB RAM or better
  + 512GB SSD or better
  + 2 USB-C Ports
  + Windows 10, Windows 11, Linux, MacOS
* 7 Monitors
  + 1920 x 1080 resolution or better
  + 24 inch diagonal
* 7 Mice
* 7 Keyboards
* 7 USB-C docks
  + Gigabit ethernet
  + 2 HDMI/DisplayPort ports
* Demo devices
  + Android: Pixel
  + iOS: iPhone

At least three servers shall be available for development, integration, and testing. The servers shall be capable of running chosen software solutions, including virtual machines and/or containers. Cloud computing solutions may be considered.

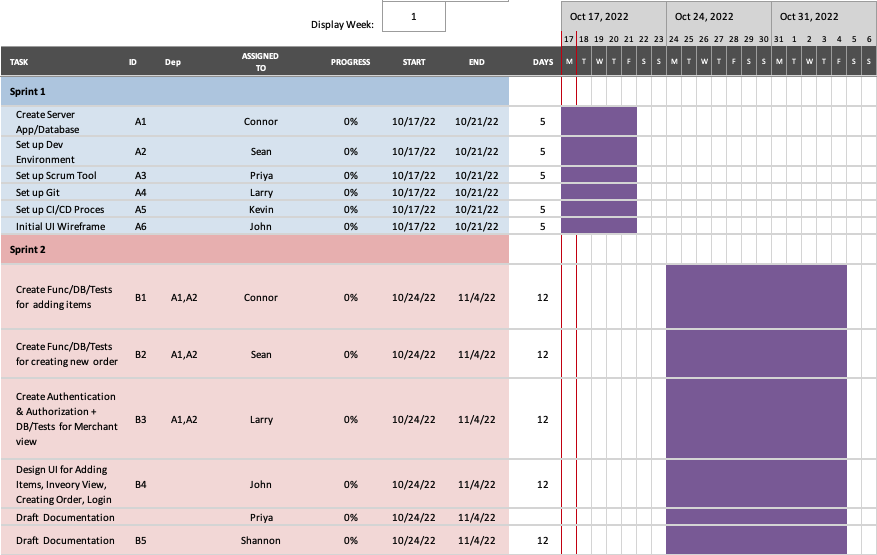
Cloud computing solutions shall be used for deployment, which shall be capable of running chosen software solutions, including virtual machines and/or containers.

# 5. Schedule and Work Breakdown Structure

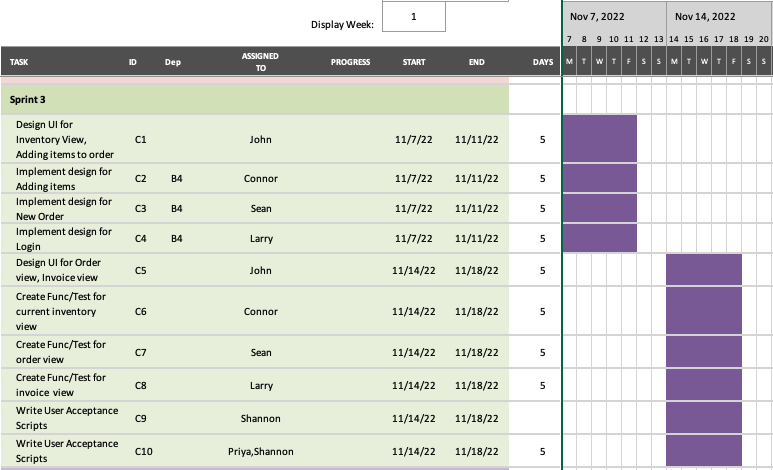
The following schedule provides an estimated development of MVP delivered to the client, including environment configuration, front and back end development, UI design and implementation, database development, deployment to development and production environments, and iterative testing.

The current development estimate is 11 weeks, beginning October 17, 2022, and providing a production-ready build by December 19, 2022.

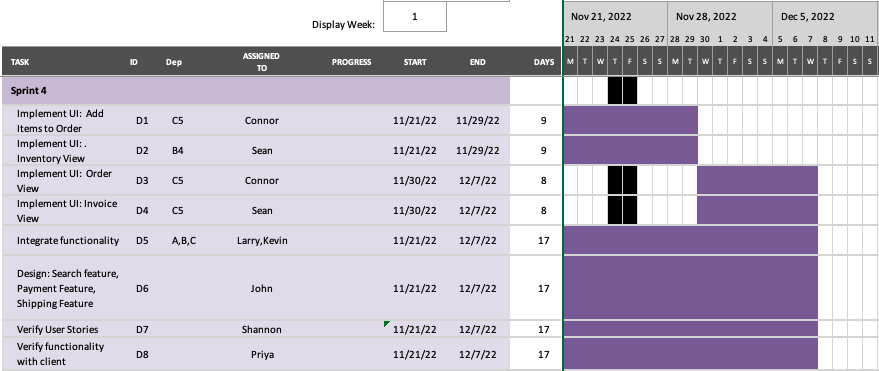
Sprints scheduled past the MVP deployment will be determined based on client feedback, testing results, and backlog of feature enhancements.



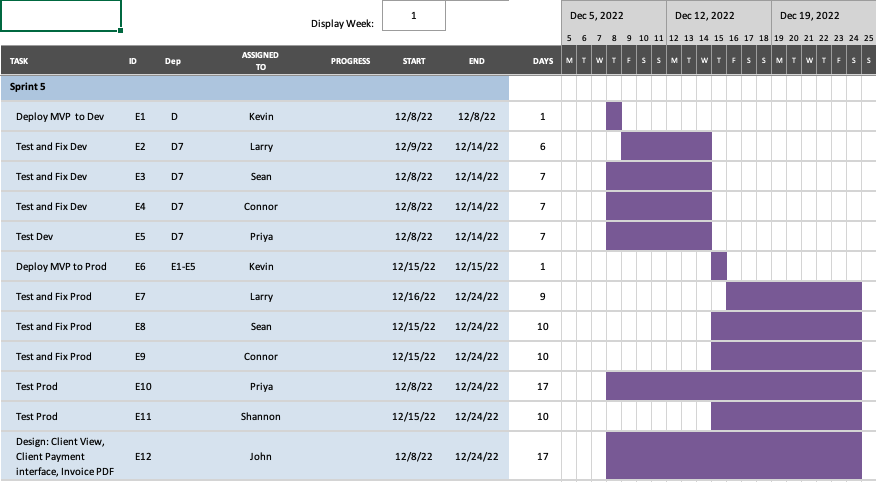
**Figure 1: Gantt Chart, Sprints 1 & 2 (10/17-11/4)**



**Figure 2: Gantt Chart, Sprint 3 (11/7-11/18)**



**Figure 3: Gantt Chart, Sprint 4 (11/21-12/7)**

****

**Figure 4: Gantt Chart, Sprint 5 (12/8-12/24)**