AI Warehouse Management System

Nic Brown, Matthew Gilham, Cody Northcroft

Table of Contents

[System & Design Cycle Overview 0](#_Toc163244917)

[System Engineering Design Life Cycle 0](#_Toc163244918)

[Pre-Phase A: Concept Studies 0](#_Toc163244919)

[Stakeholders: 1](#_Toc163244920)

[Requirements: 1](#_Toc163244921)

[Phase A: Concept and Technology Development 1](#_Toc163244922)

[Phase B: Preliminary Design and Technology Completion 2](#_Toc163244923)

[Phase C: Final Design and Fabrication (not a part of the system design phase) 2](#_Toc163244924)

[Phase D: System Assembly, Integration & Test, Launch & Checkout 2](#_Toc163244925)

[Phase E: Operations and Sustainment 2](#_Toc163244926)

[Phase F: Closeout 2](#_Toc163244927)

[System Details 3](#_Toc163244928)

[AI system components. 3](#_Toc163244929)

[AI Systems 3](#_Toc163244930)

[Full System Summary 4](#_Toc163244931)

[Frontend 4](#_Toc163244932)

[GUI: 4](#_Toc163244933)

[Communicator: 4](#_Toc163244934)

[Backend 4](#_Toc163244935)

[Controller: 4](#_Toc163244936)

[Database 5](#_Toc163244937)

[Stock Level predictor 5](#_Toc163244938)

[StockOrganiser: 6](#_Toc163244939)

[Results 0](#_Toc163244940)

[The plan used for this prototype. 2](#_Toc163244941)

[Internal Team Tools Used 3](#_Toc163244942)

[References 4](#_Toc163244943)

# System & Design Cycle Overview

Assumptions:

* We are to assume the position of a small AI development studio that has been approached by a large pet store supplier that is looking to introduce a more efficient and less hands-on solution to their warehouse stock management solution. Including a solution for predictive stock level requirements and where and if that stock can be stored.
* This large warehouse has an appropriately large sales and inventory dataset(s).
* This supplier already has a warehouse with enough storage to hold all orders.
* The supplier business already has a suitable technology solution in place including a sufficiently powerful server connecting to warehouse manager employees accessible devices.
* All access to any application is password protected and user access restrictions are not to be handled by our application.
* This is to be presented as a real-world business solution.
* The stock manager end user already have a handheld stock computer device similar to the one seen below



[2]

# System Engineering Design Life Cycle

## Pre-Phase A: Concept Studies

In this phase we identify who the stakeholders of this project will be, we also identify what the systems currently in use do and why they do it, we determine the costs related to developing this system, we will begin to develop the requirements of the system and we determine if the system is needed or not.

### Stakeholders:

Developers (*Pink Fluffy Unicorn AI Solutions*): care about ease and speed of production, installation and maintenance.

*Jubilee Pet-Stock Supplier* accountants: care about upfront costs, maintenance costs and ROI.

*Jubilee Pet-Stock Supplier*’s stock managers: care about work disruption, accuracy, performance and UX.

*Jubilee Pet-Stock Supplier‘s* customers: care about accuracy of the system to keep stock on the shelves.

### Requirements:

Develop a solution that is able to predict

First, is AI necessary here:

|  |  |
| --- | --- |
| Pros | Cons |
| * Less human interaction (and labour costs) in predicting future stock levels, sorting warehouse locations and inventory management. * Easily expandable to other solutions. * More control over the purchase data. * More accurate future stock levels. * Creates an easily retrievable location for each item. * More efficient storeroom packing. | * Workers pushback to change. * Difficulty explaining the results. * Blame for mistakes is ambiguous. * Large upfront costs. * Extra step needed when loading/unloading stock |

Yes, AI is necessary as the upfront cost will not exceed the continued cost of skilled labour.

Backend/Front end or single full-stack solution?

A Single application solution would be the easiest to create and set up in a test environment to display capabilities, it might not be suitable for a server-edge device solution. A front/back-end solution allows for better isolation and distribution and adheres to common software solution standards.

## Phase A: Concept and Technology Development

This phase is focused on the further development of the system from the pre-phase, to do this we continued to expand on who the stakeholders are and what specifically they want, the concept is also updated as the needs of the different stakeholders are developed which naturally changes what the system needs, The requirements of the system both functional and non-functional are finalised as we move into the next phase.

## Phase B: Preliminary Design and Technology Completion

In this phase we began to develop the design of the system including the overall design of the system that being the controller, predictor, layout organiser and the GUI as well as the sub-system designs on how each of them works. The prototype was also produced during this phase

## Phase C: Final Design and Fabrication (not a part of the system design phase)

## Phase D: System Assembly, Integration & Test, Launch & Checkout

In this section, we all swapped our work and attempted to integrate into our own environments. This allows for a cohesive working environment and end product as we have to be aware and are made aware of any differences in systems and any unidentified internal dependency.

This individual testing within a wide range of system environments allows for a product that is made to run in many environments.

## Phase E: Operations and Sustainment

During this phase, the capabilities of the existing system need to be assessed and upgraded if needed and integration testing between the system components, server and end devices.

## Phase F: Closeout

This prototype system will be presented to a board and client to secure further funding and aspect changes.

# System Details

Our system *Shelf Power-Intelligence* is a stock management tool that is built for *Jubilee Pet-Stock Supplier* to better provide suggestions for future stock requirements and better warehouse stock storage. *Jubilee Pet-Stock Supplier* is a small pet store warehouse and stock distributor that works between manufacturers and local pet merchandise retail stores. They have been having issues with insufficient and overstocking issues when ordering stock causing losses in revenue and an overcrowded warehouse.

Their current temporary solution is using the current warehouse managers to manually comb through previous stock data and produce predictions for when certain items are more in demand, especially for seasonal items such as heated cat beds or doggy rain jackets. Although this has proven to have helped, their previous duties have been neglected leading to a poorly managed warehouse causing safety concerns. Hiring a specific data scientist and/or stock-level manager to do these jobs is not a wanted solution due to the high ongoing costs of skilled labour.

We wish to produce an automated stock predictor and manager program that will help foresee and produce future stock sales from past data. We intend to have this system make stock order suggestions to estimate future customer purchasing patterns that a stock manager can use as a base line. This will also help the warehouse hold fewer unsold products. Another issue we hope to solve in this warehouse is the optimisation and retrieval of stored stock within the warehouse to increase warehouse safety and worker efficiency.

This will innovate on the current solution as it frees up skilled worker time whilst providing a superior and much more time-efficient result. This innovation also opens the business up to further opportunities, especially in the world of IoT.

# AI system components.

## AI Systems

The stock predictor is to be used to predict the next months expected invoices based on previous data, it is designed in such a way that the end user should be able to press a button to predict the next 30 days of invoices. The training of the data will either be done at the end of every month or when the predict button is pressed. The re-training needs to be done before predicting, this is a restriction brought in by the model that we have decided to use for our predictions as the SARIMAX model predicts the next period of days it doesn’t predict from an inputted date so the trained model needs to be up to date

## Full System Summary

To emulate how a system like this would be produced in a professional environment, a separate front and back-end solution will be developed. This is to allow for better isolation in situations if a business were to have the system's main computations in a centralised server subsystem which then connects to a graphical interface for the end user.

Although we might not be able to produce this ideal system in full to industry standards in time, we hope to emulate some portions of this.

The front end will be able to request and view an on-demand result for the stock level predictions as well as the suggested storing solution for all products in inventory. The training of the solution will be an offline training solution that will be scheduled for the end of each month.

## Frontend

### GUI:

Provides the user interface for all interactions between the user and the program. This should perform next to no processing and is exclusively to provide a reformatted design for receiving orders and presenting results in order to achieve a good user interface and good user experience.

### Communicator:

Creates, maintains and facilitates the communication of data between the front and back ends. Although a part of the same program, this portion will be identical between all different device builds [multiple device builds are not implemented in this assignment, but the system design is intended to allow for easy future implementation].

## Backend

### Controller:

Provides receives and fulfils communication requests between the front end by calling appropriate programs and handling the correct dataflow. This portion will receive orders from the front end and will provide the relevant function (usually re-training, testing or generating from the StockPredictor and StockOrganiser). Once the function has been executed, the controller is then to communicate any results back to the front end for the user.

### Database

In this assignment, we will assume that there is a separate fully formed relational database and system that can produce a simple CSV of relevant data from a preexisting professional solution.

### Stock Level predictor

#### Preprocessing Steps

* Turn csv into a data frame.
* Drop unnecessary features.
* Convert date to index and drop date feature.
* Convert index to datetime.
* Extract itemIDs.
* Split data frame by itemIDs.
* Extract only the itemQTY from each data frame
* For each data frame get index range
* Create new data frame using index range
* Add corresponding itemQTY with matching date any not filled are left as 0

#### 

#### Model

This subsystem needs to make predictions for the numbers that each stock item will be purchased in the next month. A SARIMAX model has been selected due to its expertise in identifying trends in time series data and forming accurate forecasts. SARIMAX is also quite interpretable in its coefficients for both the autoregressive and exogenous variables and has useful diagnostic tools such as residual analysis, AIC/BIC comparison, and model selection criteria.[7]

A linear regression model was also considered for this problem because of its simplicity and explainability however it was deemed to be too simple in this case as we wanted predictions that could be influenced by seasonal trends.

#### Postprocessing

The total stock for each item in the next month is totalled to match the warehouse’s order schedule and compiled into an array of the item’s identification number and quantity (key-value pairs) for easy piping.

#### Things to be added in the final version.

* adding new items handled correctly (currently not handled so training and predicting won’t work with items with no sale history)
* averaging data from month over days in the month (this should hopefully make the model more accurate but may cause issues in prediction)
* normalising (may improve accuracy but may also impact the predictions)
* parameter tuning (increase accuracy of the model)

#### Current Issues

* daily predictions are way out.
* current adding of days has many 0 qty and jumps to high tens when there was data present in the original data frame (throwing out the model)
* testing currently not working in .py format (but working in original .ipyn file)

### StockOrganiser:

#### Preprocessing

The data is pre-sorted based on the frequency of an item so that the most frequently purchased items are placed into bays closer to our front of the warehouse for quicker access.

#### Model

Initially, we were looking at a greedy algorithm as it is a fast and simple model that takes very little into consideration with quick processing time. This base algorithm is a bit simple and may not capture many nuances, so the final version is a modified greedy algorithm that incorporates a weighting and priority based on frequency of use for each item. Incorporating an A\* algorithm was also considered but might have been too time-consuming and a less time-efficient algorithm so was abandoned

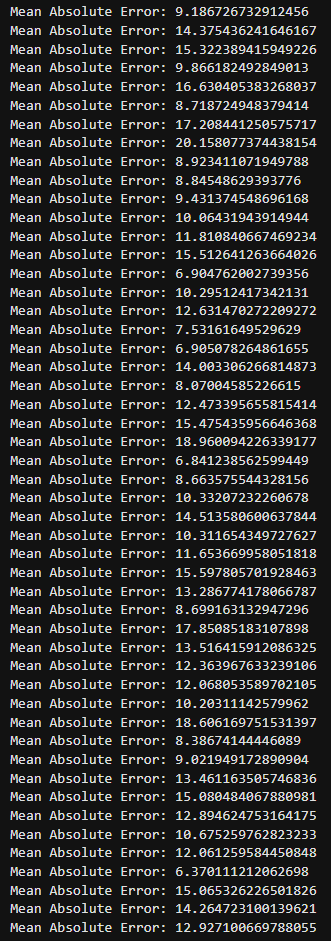
# Results

Previously, it took the main factory manager 2 working days a month to complete this task and was allowed to make an order only after 1 year but was only able to make a somewhat accurate prediction after 3 years of experience. This software, on the contrary, is able to produce a considerably more accurate result, only taking 1-2 minutes to train and seconds to make a prediction.

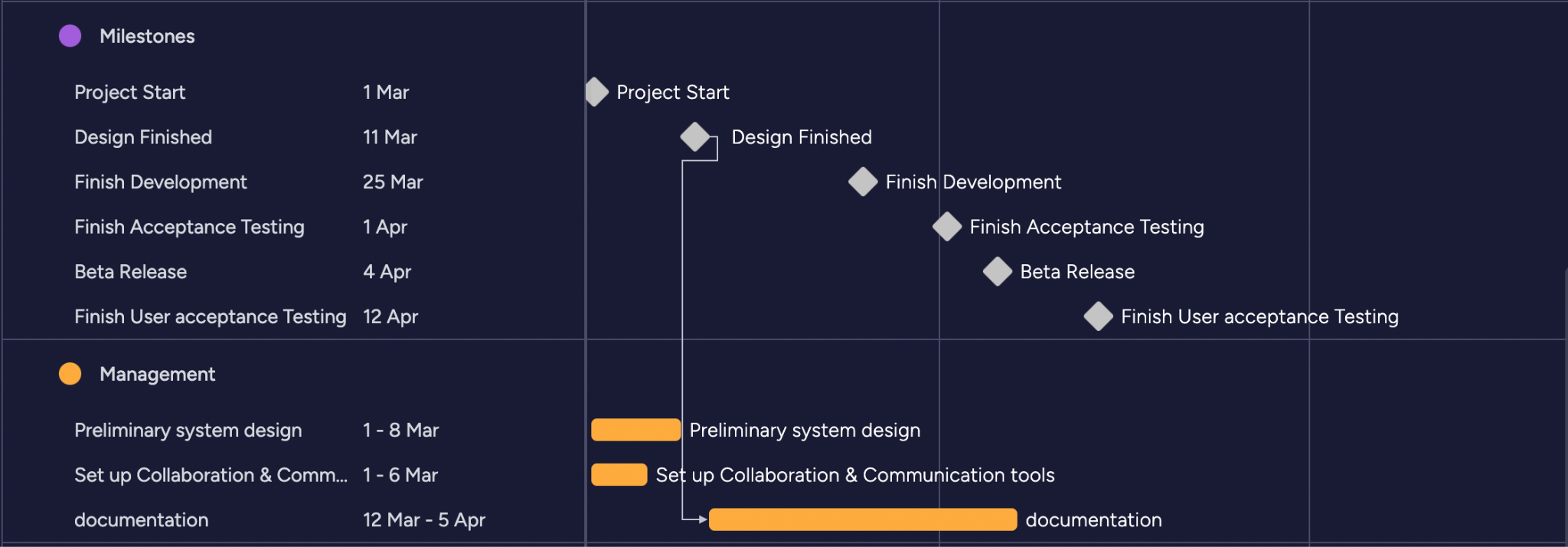
Using the average hourly wage for this position at $55-$70 costs the company 1k a month in near wasted time and labour plus the majorly disrupting learning period. As the company already had a running server/data entry system, this system can be installed without much disruption and quick results.

This time made free by this system to the warehouse manager will allow them to fully tend to their true duties and not have to cover for these two days.

Predictor Mean Absolute Error for each model:



# The plan used for this prototype.

[1]

# Internal Team Tools Used

Project management:

We are using Monday.com’s Gantt chart to visualise and schedule our project. [1]

Version Control:

We use Git for the version control and distribution of all Python scripting in our project. [3]

We use Plastic SCM for the version control of the Unity front-end project. [4]

Collaboration:

We use Google docs to collaborate on documents. [5]

We use git to collaborate on scripting. [3]

We use Plastic SCM to collaborate in unity. [4]

We use Discord to schedule and communicate this collaboration. [6]

Communication:

To communicate effectively at any time, we all are part of a discord group server where text messages, files and voice calls can be used to communicate.

We all make sure to attend and stay back for the weekly tutorial where we discuss our weekly progress. [6]

# References

[1] Monday. "Your go-to work platform." monday. <https://monday.com> (accessed

04/04/24, 2024).

[2] ZEBRA. “TC7X series Mobile Computer”.

<https://www.zebra.com/ap/en/products/mobile-computers/handheld/tc7x-series.html> (accessed 05/04/2024).

[3] github. "Let’s build from here." github. [https://github.com](https://github.com/) (accessed 05/04/24, 2024).

[4] Unity. "LOOKING FOR PLASTIC SCM?" Unity. [https://www.plasticscm.com](https://www.plasticscm.com/) (accessed

05/04/24, 2024).

[5] Google. "Build your best ideas together, in Google Docs." google.

<https://www.google.com.au/docs/about/> (accessed 05/04/24, 2024).

[6] Discord. "IMAGINE A PLACE..." discord. [https://discord.com](https://discord.com/) (accessed 05/04/24, 2024).

[7] Time Series forecasting using SARIMAX. Soubhik Khankary (June 2022).

<https://medium.com/@soubhikkhankary28/time-series-forecasting-using-sarimax-ca98dd7238a0> (accessed 28/03/2024)