Consumer Aware Warehouse Management

Design Document

SDMay20-25

Client: Jimmy Paul

Advisor: Goce Trajcevski

Team Member	Roles
Lindsey Sleeth lssleeth@iastate.edu	Meeting Scribe Project Manager Software Developer
Sam Stifter stifter@iastate.edu	Test Engineer Software Architect Software Developer
Omair Ijaz oijaz@iastate.edu	Quality Assurance Engineer Meeting Facilitator Software Developer
Jameel Kelley jamkelley22@gmail.com	Report Manager Software Architect Software Developer
Andrew Smith arsmith3@iastate.edu	Database Administrator Quality Assurance Engineer Software Developer
Elijah Buscho elijah@iastate.edu	Test Engineer Project Manager Software Developer

Team Email: sdmay20-25@iastate.edu

Team Website: http://sdmay20-25.sd.ece.iastate.edu/

Revised: November 3, 2019, Version 2

Executive Summary

This document exists to detail the overall planning and design of the Consumer Aware Warehouse Management system. This section gives a very brief overview of the document and what will be covered. Development standards, practices, and applicable courses, along with skills, are briefly listed.

Development Standards & Practices Used

Here the system architecture, requirements, project management, and engineering standards are briefly listed.

Software Architecture

UML Diagrams

- Class Diagram
- Use Case Diagram
- Activity Diagram
- Sequence Diagram

Requirements Analysis

- Volere Requirements Process
- Iterative Requirements Process
- Iterative Requirements Strategy
- EARS (Easy Approach to Requirements Syntax)
- Design Thinking

Team Workflow / Project Management

- Gantt Chart
- Agile Methodology

Engineering Standards

- Security
- Ethics
- Minimum Versions of Software
    - Postgres ver 11

Summary of Requirements

The system-to-be will satisfy the following requirements. For more information on each requirement and rationale behind each view section 1.3.

- Take data from the Crafty database
- Make predictions about optimal ordering to maintain product warehouse stock
- Consider future product ordering
- Have a visual component to interface with the generation of orders
- Be able to handle approximately 1200 SKUs a day
- Generate a report on demand

Applicable Courses from Iowa State University Curriculum

- SE/COM S 409, Software Requirements Engineering
- SE 319, Software Construction and User Interfaces
- SE 329, Software Project Management
- SE 339, Software Architecture and DesignA
- ComS 227, Intro to Object-Oriented Programming in Java
- ComS 228, Intro to Data Structures in Java
- ComS 311, Intro to Algorithm Design and Analysis
- ComS 309, Software Development Practices
- ComS 363, Intro to Database Management Systems
- DS 201, Intro To Data Science
- ENG 314, Technical Communication

New Skills / Knowledge Acquired Not Taught in Courses

The following is a short list of skills that the members of SDMay20-25 feel have not been converted in classes taken thus far in the required curriculum. They represent the knowledge that needs to be acquired through research spikes and self-learning.

- Machine Learning Algorithms
- Front End Development
- DevOps
- Time-Series Forecasting
- Amazon Web Services

Table of Contents

List of Figures/Tables

Definitions

Product Warehouse - The warehouses where Crafty keeps its products before shipping them out to clients

Forecasting - The work of determining the optimal ordering schedule for distributors based upon inputs from the Crafty system

Distributor - A company that Crafty purchases its goods from to be later sent to Crafty’s clients

SKU - A combination of a Product and its ordering size. For instance, a six-pack of LaCroix would be different from a twelve-pack of LaCroix.

1. Introduction

The purpose of this introduction is to provide the reader with an understanding of the SDMay20-25 project, Consumer Aware Warehouse Management. This will come from a detailed description of the project which is supported by a definition of the problem and a proposed solution and approach which has been crafted in response to the needs of SDMay20-25's client Crafty.

1.1 Acknowledgement

SDMay20-25 acknowledges Iowa State University's Departments Software and Computer Engineering and the Department of Computer Science for providing a strong foundational knowledge through education, professional insights and experience, and resources to complete the project.

SDMay20-25 gives special acknowledgment to Dr. Goce Trajcevski for providing technical expertise related to the project subject and mentorship to guide SDMay20-25 to meet course outcomes and successful project completion.

SDMay20-25 also acknowledges Crafty LLC, headquartered in Chicago, IL, for the opportunity to collaborate with industry to solve a real-world problem that provides current industry technological challenges and opportunities for learning. Crafty has granted access to real data used to design, build, and test a solution with, as well as mentorship in industry best-practices.

SDMay20-25 gives special acknowledgment to Crafty CTO and co-founder, Jimmy Paul, for the time, feedback, and resources provided to SDMay20-25 during project development.

1.2 Problem and Project Statement

The purpose of the problem and project statement is to identify the problem that Crafty has requested to be addressed, the motivation for a solution, and a proposed solution in response to the problem. The proposed solution will be detailed and supported by the problem motivation and Crafty's current software solution. Additionally, current industry practices will be used to support the development of SDMay20-25's software solution.

1.2.1 Problem Statement

SDMay20-25's client Crafty desires a software solution to predict the need for products to be stocked in Crafty product warehouses. The main motivation for this project comes from Crafty's need to

maximize their revenue by reducing (1) waste of expired product, (2) missed sales due to insufficient inventory, and (3) current labor efforts for manual calculation that can prove to be erroneous and time-consuming.

Currently, Crafty maintains a software solution to auto-generate a report of products that Crafty must order from each distributor that delivers to them. The report generated must allow Crafty to place orders to their distributors which enables them to maintain the appropriate product thresholds to serve both the needs of their current customers, as well as, the needs of new customers and any seasonality or sudden known changes outside of regular needs.

As it stands, the software solution in place takes into account four points of data to determine a distributor purchase order. These input variables are as follows: client order aggregate history, distributor schedule and lead time, missed sales, and inventory thresholds. This solution is insufficient as it does not currently consider the following known data points: (1) the needs of new customers for each product, (2) the trends of historical client inventory levels for each product, (3) seasonality or other sudden known changes that determine product demand, (4) product expiration dates, (5) current space available in the product warehouse, (6) trends for pricing of the same product across multiple distributors, and (7) accuracy of distributor delivery windows. Because Crafty does not currently consider the mentioned data points, Crafty is limiting opportunities for revenue. In response to this, Crafty seeks a software solution that will build on current inputs to the software solution, as well as, incorporate the missing inputs, in order to maintain an accurate and robust system to predict the products which must be reordered from Crafty distributors and allow Crafty to be optimally operated.

A specific example of Crafty’s current operation is illustrated in figure Figure 1.2.1.1. In this figure, we see the overall use case of the company, Crafty. While this is undoubtedly simplified, it serves to define the context of which the Consumer Aware Warehouse Order Management system will be operating in. We see the system-to-be will be internal to the Crafty team yet will function alongside the numerous adjacent systems already in place, such as the stock inventory management system and order shipment system.

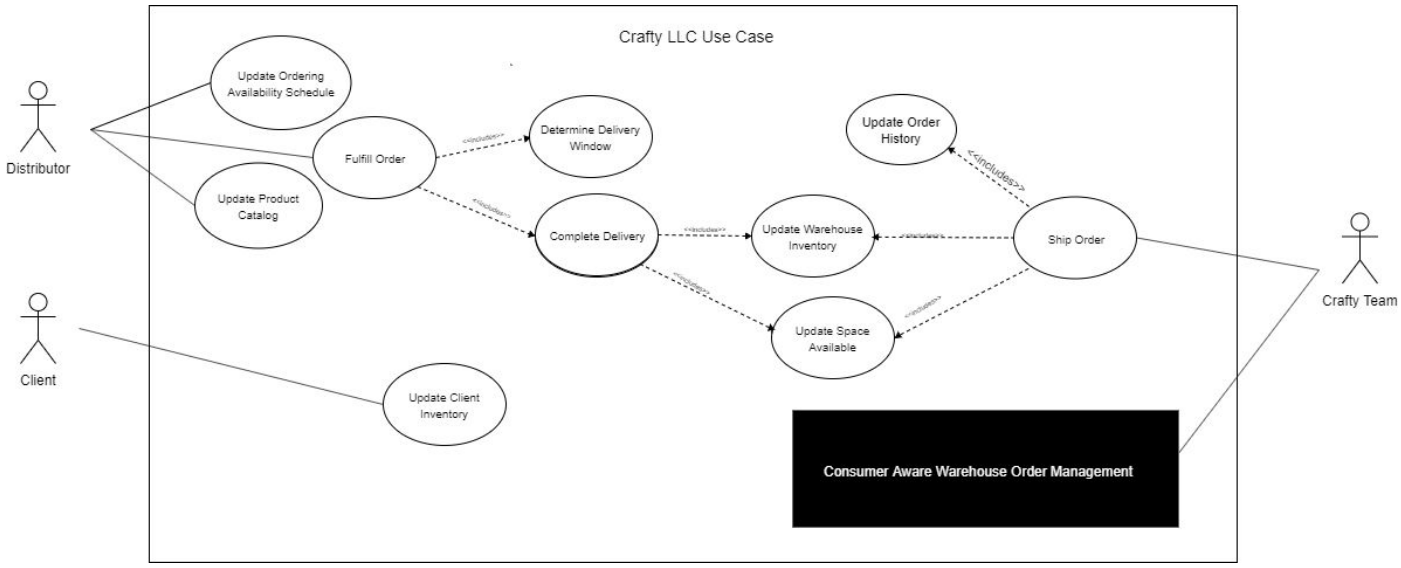


Figure 1.2.1.1 - Crafty LLC Use Case Diagram

The SDMay20-25's software solution is referred to as the "Work of Forecasting" in figure 1.2.1.2. Here a context diagram of adjacent systems is displayed, which shows the specific inputs and outputs that the software solution will be receiving and expected to output.

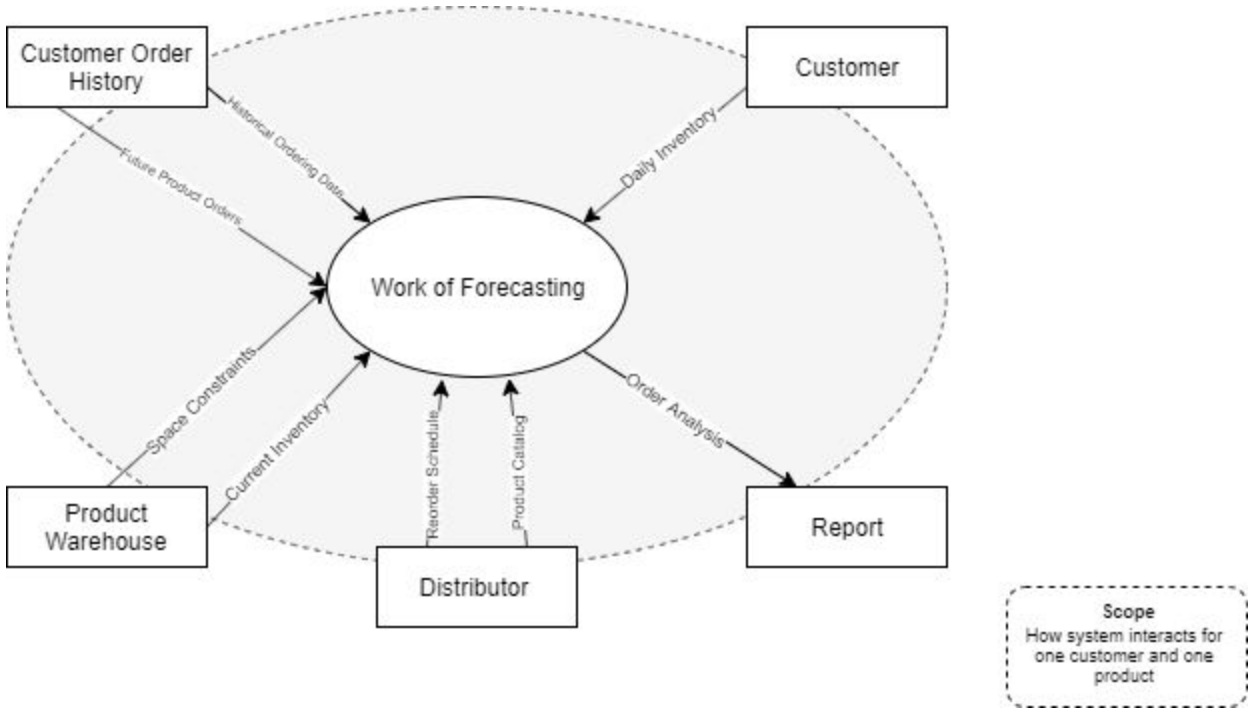


Figure 1.2.1.2 - Work of Forecasting Context Diagram

The specifics of the expected product use cases for the software solution can be seen in figure 1.2.1.3. Here the main use case of viewing the distributor purchase order can be seen and how other actions interact with it.

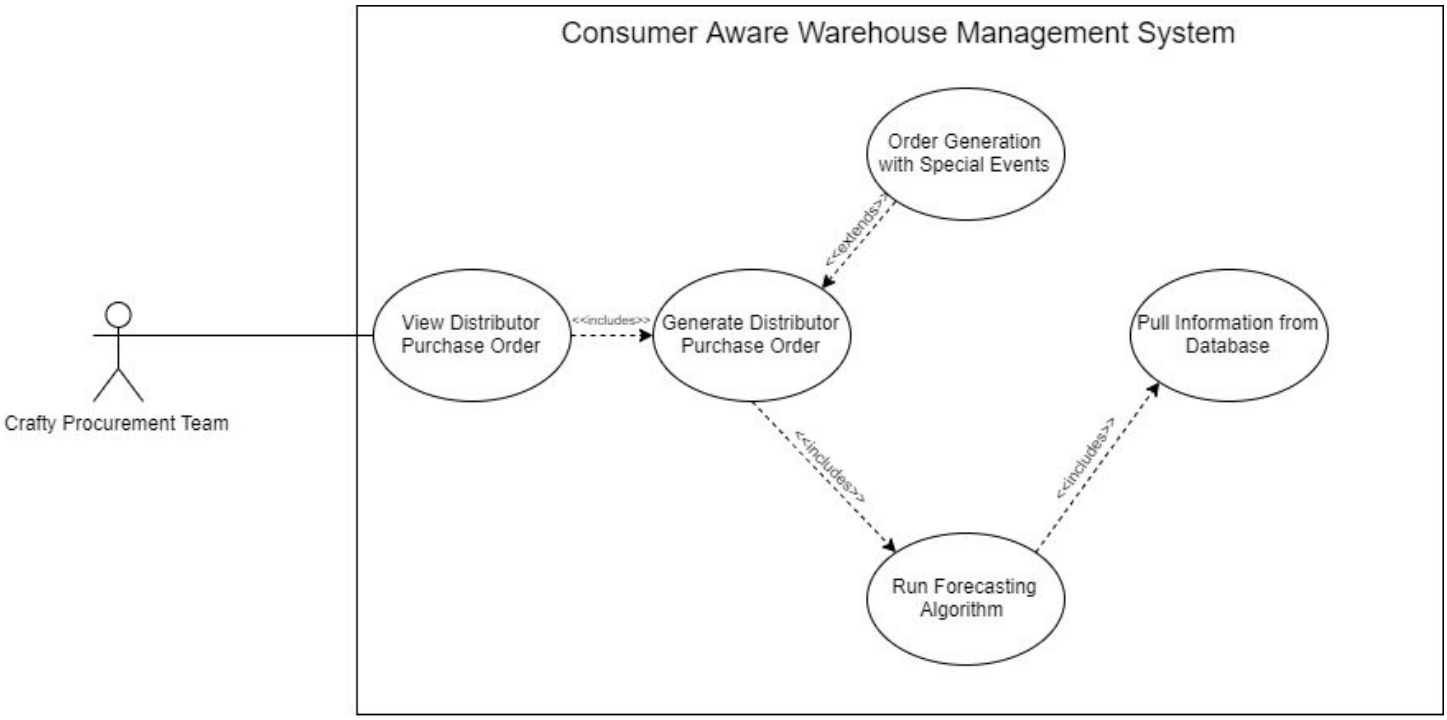


Figure 1.2.1.3 - Product Use Case Diagram

1.2.2 Project Statement

SDMay20-25 has crafted a proposed solution in response to the problem detailed in Section 1.2.1, which addresses the needs shown in figure 1.2.1.3. The use case of the software solution here has been clearly defined to allow the Crafty Procurement Team to view a product order table for a selected distributor. This will contain data populated by the Consumer Aware Warehouse Forecasting System and enable the Crafty Procurement team to place orders that are correct and to maintain appropriate thresholds in the product warehouse.

The proposed software solution will use the available data points to model current product consumption and forecast the future orders of those products. The input to the solution will make use of data points Crafty currently has available and also uses data points Crafty currently does not utilize. The priority of these data points for incorporation into a solution will be determined by our client, Crafty.

The proposed solution will serve as a proof of concept to (1) reduce expired product waste, due to a reduced number of unneeded items sitting in inventory; (2) allow for increased sales, by enabling Crafty to maintain the correct product thresholds in order to capitalize on potential future sales and reduce missed sales opportunities; and (3) reduce misdirected taskforce by eliminating the tedium of checking when products should be reordered.

The following artifacts will be generated in response to the development of the proposed solution:

1. Team Meeting Notes
2. Bi-Weekly Status Report Updates
3. Architecture Plan Documentation
4. Requirements Specification Documentation
5. Design Document (final and revisions)
6. Senior Design Website
7. A Functional and Improved Software Solution
8. A User Interface to Display Features and Usage of Software Solution
9. Legal Documentation for Crafty
10. Lightning Talks for Senior Design

1.3 Requirements

The purpose of this section is to describe the requirements (functional and non-functional) within the project context.

1.3.1 Functional Requirements

1. The system SHALL take data from the Crafty database as input in creating a distributor purchase order

   This requirement makes it such that SDMay20-25 software solution will be tied to complying with the format of the current data stored in Crafty’s database. This can be seen in figure 1.2.1.3 as the “Pull information from Database” use case.

2. For each distributor, the system SHALL make predictions about optimal ordering to maintain product warehouse stock for individual goods provided by that distributor based on previous product orders, current product warehouse stock, current product warehouse space.

   As seen in figure 1.2.1.2, the adjacent systems being worked with allow the input of the previous product orders, current product warehouse stock, current product warehouse space.

These system inputs will allow the software solution to generate a distributor purchase order output.

3. The system SHALL consider future product ordering, which includes new onboarding of Crafty clients and current Crafty clients with needs outside of contract as input when generating orders to be placed to distributors.

   Crafty needs a tool that will allow them to account for more than just happy path use cases. Thus the software system needs to have an inbuilt input for special or exception events that would otherwise not be accounted for. This can be seen in the figure 1.2.1.3 as “Order Generation with Special Events.”

4. The system SHALL have a visual component to allow Crafty to interface with the generation of orders to be placed.

   The Crafty procurement team needs a way to interface with the system tool to make manual adjustments and view the distributor product ordering report. This is documented in figure 1.2.1.3 as the “View Product Order Table” that is an inclusion use case of the Crafty procurement team.

1.3.2 Non-Functional Requirements

1. The system SHALL be able to handle approximately 1200 SKUs a day

   The system in place that Crafty is using is able to handle this number of SKUsin a day. Due to this, the SDMay20-25 software solution must be able to handle this many skus a day at a minimum.

2. The system SHALL generate a report on demand that will take less than two minute 90% of the time and will take less than five minutes the remainder of the time in order to satisfy the need for on-demand ordering analysis.

   The current Crafty system in place for doing ordering analysis was observed to take less than a minute to generate its results and display it to the view. It is important that the SDMay20-25 solution be no worse than the current system in place in terms of performance. In figure 1.2.1.3, this product use case can be seen as the “View Product Order Table” use case.

1.4 Intended Users and Uses

The software solution proposed in section 1.2.2 is designed and intended for usage by Crafty as a solution to the problem described in section 1.2.1. This solution will enable Crafty to maximize their

product margins by using a larger subset of data currently used to forecast which products Crafty must order from their distributor. The primary use case is shown in figure 1.2.1.3 as viewing the distributor purchase orders by the Crafty procurement team. Specifically, the software solution proposed requires a user interface for Crafty to evaluate the solution and for the Crafty team to interact with the forecasted products to be reordered. SDMay20-25 will take into consideration Crafty's needs for security and performance, into consideration for design and implementation and these will be discussed in depth later in section 5.4.

1.5 Assumptions and Limitations

(Refer back to the use case diagram and talk about the solution and why we are building our solution as a result of Crafty's current system. We are not implementing the current Crafty solution. We are revising their solution and minimizing their current pain points in our solution. Refer to the use case diagram and talk about the uses for how our system minimizes those pain points)

Upon project completion, SDMay20-25 will provide the software solution and associated artifacts to Crafty to serve as a proof of concept for improvement of the current solution that Crafty operates with. SDMay20-25 will take into design consideration and use the current solution that Crafty operates with, when possible, but only to the extent that technology allows us to do so, and with the intent that SDMay20-25's solution will be an extension of the current Crafty solution and may not use all of the same technologies for implementation in order to reduce the expense of development and produce a more accurate solution. These deliverables will be well documented in section 1.6 in order to allow Crafty to determine what components SDMay20-25 solution they will incorporate and how they will incorporate them.

1.5.1 Assumptions

- The system will maintain communication with an instance of the Crafty database.

  We see this is necessary in the figure 1.2.1.2 context diagram. In order for the software solution to forecast the products that must be ordered, we must have access to required inputs existing in the Crafty database. These include the historical ordering data, future product orders, daily inventory, product warehouse space constraints, product warehouse current inventory, distributer reordering schedule, and distributor product catalog.

- The software solution will be developed, used, and maintained by English speakers.

  The software solution will be developed and implemented in English. This is because the project will be implemented and documented in English, therefore, it will not be translated into another language.

- The system will only be used inside of the US.

  This is because the project will be created and comply to the laws and regulations of the US.

- The system will only be used to manage the inventory of a single product warehouse.

  Since this is a proof of concept, showing that the algorithm works for one product warehouse.

- The system will only predict products to be reordered by one distributor at a time.

  This is because ordering from multiple distributors would degrade the performance of the system.

1.5.2 Limitations

- Accuracy of system predictions is completely reliant on accuracy of data system is given as inputs.

  The SDMay20-25 software solution is only planned to be a single part of the whole of Crafty's software based solution. Thus the software to be must work with current data being stored and provided by the Crafty team. If this data is incorrect for any reason when being used as input into the system-to-be, the result cannot be ensured correct. The scope of work can be seen in figure 1.2.1.1 and this can be shown to not include any other potential failure points.

- Implementation will match the Crafty system architecture wherever reasonably possible.

  The system-to-be will need to match the current Crafty system for this product. This is due to the nature of integration being easier when things are similar. This is the requirement that drives tool standards such as the minimum version of PosGreSQL required to function. In figure 1.2.1.1 it is displayed that the system-to-be will be interfacing with the Crafty team and its architecture.

1.6 Expected End Product and Deliverables

The SDMay20-25 solution to the proposed problem by Crafty will consist of multiple parts to be delivered in part to Crafty and to Iowa State University's senior design program. In section 1.2.2 the artifacts to be delivered at the end of the year were listed. The main deliverables to the company Crafty will consist of this design document and the software solution implementation.

To better explain the software solution implementation, below in figure 1.6.1 is the proposed System Architecture Diagram. This diagram shows the technology stack that is being used for development as well as the flow of communication between systems.

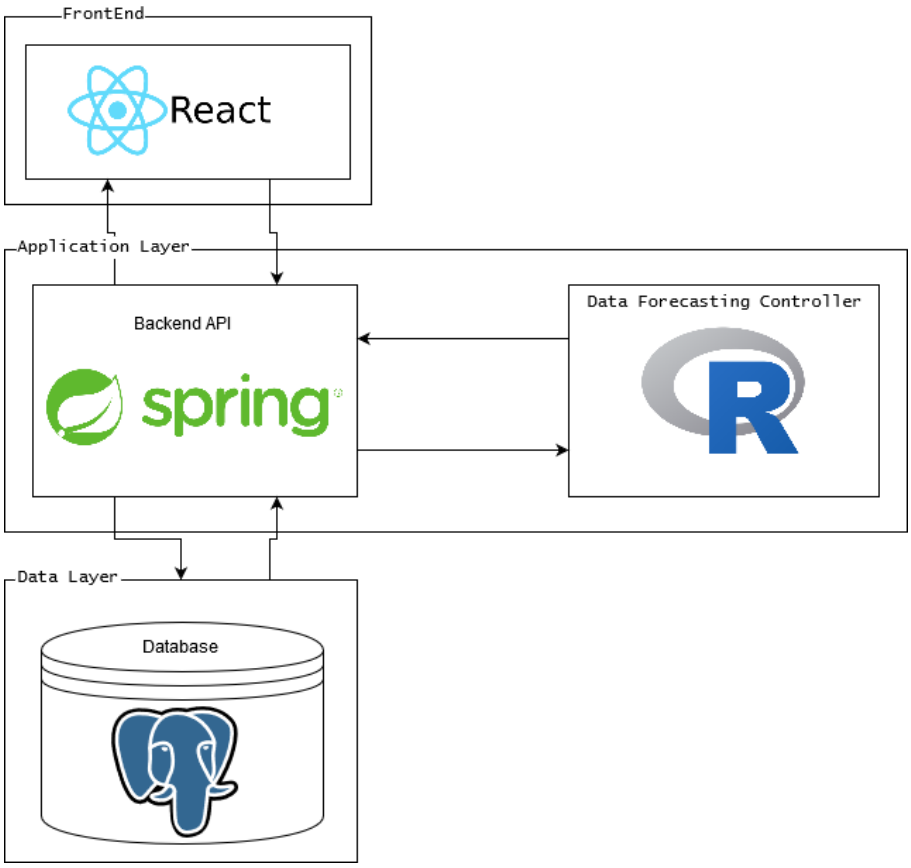


Figure 1.6.1: System Architecture Diagram

2. Specifications and Analysis

This section describes the design and implementation of the software solution proposed in 1.2.2.

2.1 Proposed Design

SDMay20-25 has defined a set of requirements and finalized a tech stack that will be used for development of the software solution.

SDMay20-25 currently seeks to work in an agile manner by defining project goals to be revised on a bi-weekly basis. The agile methodology is a development practice used in industry that is proven to be effective for the development of short projects with a high level of adaptability due to changing variables. The first part of the project will be composed almost entirely of requirements analysis and documentation to ensure that SDMay20-25 adheres to the requirements that Crafty has defined for the project, minimize the number of missed requirements, and reduce development efforts that are not meaningful. During requirements analysis, SDMay20-25 has been working on setting up the server to host the database and initial website. SDMay20-25 will iteratively adjust the requirements as necessary and complete the initial website and backend framework by the end of the first semester.

One of SDMay20-25's main objectives is to obtain anonymized data from Crafty as SDMay-20-25 will need access to the data on demand. SDMay20-25 will use a PostgreSQL database which will contain the data from Crafty. Because all members of the SDMay20-25 have experience with SQL, the team should be able to adapt quickly to PostgreSQL. The completed database will result in SDMay20-25's first software artifact and will enable SDMay20-25 to produce an initial website which will allow Crafty to view data outputs and interact with the software solution. SDMay20-25 will also produce documentation artifacts such as research and design decisions, weekly reports, lightning talks, and UML diagrams which contribute to the understanding of the design of the proposed solution.

The development and completion of the main functional requirement of forecasting product orders will be completed in the Spring semester. SDMay20-25 is currently researching Our main functional requirement is a predicting minimum required order quantity and frequency of a given product in order to fulfill the demands of the clients. There are a couple of potential solutions to this. One solution is a machine (deep) learning approach, and the other solution is linear regression. In order to test/demo the system, the team will need to develop a front-end application with the following functionality: a way to trigger the computation of the prediction algorithm and a way to display the results of the prediction algorithm (how much to order). We will solve these problems using an MVC design pattern, where the Model will be a PostgreSQL database, the controller will be an AWS server running Java Spring, and the view will likely be written using ReactJS. The system architecture diagram detailing this can be seen in figure 1.6.1.

See Appendix A for a diagram of Crafty's existing database schema and relation.

2.2 Design Analysis

SDMay20-25 has decided to use PostgreSQL as the database to host Crafty data and interface with the software solution. This decision was made for a variety of factors, with major emphasis on the fact that this is the database that Crafty currently uses. Other contributing factors include an open source framework, and team familiarity with SQL. Another database considered was MySQL, as all members of SDMay20-25 were familiar with it. Ultimately, PostgreSQL was chosen since it is similar to SQL, but expands upon functionality and will make delivery and integration of the final product easier for Crafty.

For the backend server, SDMay20-25 has decided to use Java Spring, as all members of the team are familiar with the Java software suite and Spring. Additionally, Java has widespread support in the university, is predominantly used in industry, and maintains a large online community of support and resources. Spring is also very well documented and has many online resources both in the forums and directly from the developers.

SDMay20-25 has considered a variety of options for forecasting the products that Crafty must reorder. One technology taken into consideration is pandas, a Python library; pandas is a data analysis library that includes tools for time series analysis. In addition, the programming language R has been considered. R encompasses many different statistical use cases, and it includes a time series forecasting package called "forecast". SDMay20-25 will continue to evaluate this technology further depending on how the early mathematical models work and Crafty's needs.

In order to accurately forecast the products that Crafty will need to reorder, SDMay20-25 will be using mathematical regression or machine learning. SDMay20-25 will continue to research which solution has the best potential for success and will document the decisions made. SDMay20-25 may attempt solutions for both mathematical regression and machine learning as a prototype. However, at this point in time the first prototype will use regression first as it is a simpler method for prediction; unlike Machine Learning, regression doesn't require us to train a dataset. Additionally, there may not be enough data for an effective model for deep learning.

2.3 Development Process

SDMay20-25 will adhere to an agile development process. This method of development was selected due to the iterative nature of the work and potential for variable requirements. SDMay20-25 will host a meeting with Crafty bi-weekly to demonstrate components of the solution and ensure the requirements should remain unchanged. SDMay20-25 will break into development groups and assign

tasks for completion between each sprint meeting. SDMay20-25 has broken down the solution into tasks outlined in section 3.3. The list has been broken into small tasks that can be implemented in bi-weekly sprints. SDMay20-25 chose the agile methodology over the Waterfall and Iterative methodology because of the level of client interaction from project start to finish, and the fact that the team will not build one component individually at a time. Instead, the project should be completed in short development sprints with regular customer interaction and feedback.

An example sprint entails the following:

1. In the beginning of the sprint, SDMay20-25 has defined upcoming tasks, estimated time for completion, and selected tasks to be completed for the upcoming sprint. SDMay20-25 identified that:
    a. The server and database needs setup
    b. The database and website prototype should be hosted on the server
    c. The database should be instantiated with Crafty's data
    d. Data endpoints should be defined
    e. A prototype of website should be completed
    f. The website should interact with the database to display data
2. Tasks were evaluated based on the time taken for completion. Not all tasks were selected for the first iteration. For example, in the first iteration
3. Tasks were assigned to each individual team member based on the available number of work hours.
4. Throughout the week, each member worked on assigned tasks and updated the task board.
5. At each team meeting throughout a given week, SDMay20-25 discussed what was done, what needs to be done next, when the tasks should be completed by, and any impediments with current task completion.

2.4 Design Plan

(Talk about each bubble in the use case diagram and how we are going to complete them. We can relate them back to -- task decomposition (3.3))

SDMay20-25 will develop the software solution with the following use cases in mind:

1. View Distributor Purchase Order
2. Generate Distributor Purchase Order
3. Order Generation with Special Events
4. Run Forecasting Algorithm
5. Pull Information from Database

These use cases are represented in figure 1.2.1.3 with the product use case diagram.

The requirements for the software solution depend entirely on the needs specified as requirements by Crafty. The requirements and intended users and use cases have been specified in sections 2.3 and 2.4.

2.4.1 Architectural Overview

Our software solution will consist of three main systems with a number of subsystems in each. These will be the database, the server(backend), and the frontend. These systems will communicate through database queries and REST calls. This integrates with our use cases by allowing the software solution to store and retrieve data in non-volatile memory, run an intensive forecasting algorithm on a dedicated server, and communicate with an interactive frontend to display results.

Each part of the system allows the SDMay20-25 team to implement a solution to the list of 5 use cases generated in figure 1.2.1.3. The frontend view will display the purchase orders that were created from the algorithm. The server will run the algorithm that will allow generation of the purchase orders. It will also query the database for inputs in running that algorithm. Finally, the frontend will allow for input of special event cases which were not taken into account by the backend system.

3. Statement of Work

This section specifies current available software solutions which are similar to the solution proposed by SDMay20-25. Additionally, this section specifies what technologies SDMay20-25 will use to implement the software solution with the corresponding rationale. Finally, this section will discuss tasks to be implemented with different milestone dates, any risks associated with those tasks, and validation of the proposed solution.

3.1 Previous Work And Literature

There are a few examples of other companies doing similar forecasting work in the industry such as Walmart, which will be described in the next paragraph. These solutions range from machine learning/deep learning, which considers every possible aspect of supply chain, to single regression from an Excel sheet.

The discussion regarding the article, artificial intelligence in supply chain planning is changing retail and manufacturing by Bruno Delahaye, will be limited to the scope of the Consumer Aware Warehouse project. The project uses machine learning and AI to predict the demand of a product based on several factors such as: weather, seasons, holidays, promotions, and past sales data. The article gave an example where if the weather is sunny and cloudy the sale of steaks is higher that hamburgers. If the weather is warm and sunny, like during the summer, the sale of hamburgers if higher. The algorithm takes into consideration the limitation and conditions of how to get the product to the product warehouse and store. It looks at shipping time. It does this by looking into the distance, road conditions, weather, and the capacity of the shipping container [1].

The advantage of this system is providing products with a better accuracy to the demand. It does this efficiently and cost effective as there is a lot less costly and error prone human interaction. The disadvantages to this system are the size and amount of resources needed to maintain. This relates to SDMay20-25's project because the above solution takes into consideration past sales data and the time it takes to get the product to the product warehouse. It is different from SDMay20-25's solution because the above solution takes in more factors such as weather and promotions.

A contrast to the machine learning approach is an example of a regression based model as outlined in the article single regression: approaches to forecasting from the North Carolina State University. The regression based model is able to predict the demand based on past sales and the seasons. Using one variable linear equations it is able to forecast the demand for a given day during a season [2]. The primary advantage of this approach is the small number of resources needed to build and maintain because it doesn't need much data to operate.

Some disadvantages of this method include not taking into account spikes in demand due to factors that aren't based on past sales, such as weather. Also, it doesn't take into consideration the shipping time of the product. Instead, it might determine that the demand for a product the next day is high, so an order is placed but may take a week to arrive, thus, leading to missed sales during that week. This relates to SDMay20-25's solution because the above solution because it takes into consideration past data. It differs from SDMay20-25's solution because the above solution doesn't take into consideration the shipping time to get the product to the product warehouse.

3.2 Technology Considerations

The technology SDMay20-25 is using PostgreSQL for the database. The strengths of this software are that its queries are based around MySQL. It is open source so there is no licensing and free. It also is the software our client uses so it will be easy to import their data and integrate with their system. It also has high reliability because it doesn't allow users to bypass the data checks to make sure it is valid data unlike MySQL where a user can prior to version 5.0. Since it is not a NoSQL language it will be easier to interpret the data for what we need.

For the server our team is using the Java Spring Framework and Hibernate. The strengths are our team has experience using Java, Spring, and Hibernate. There also is a big community with all three so if there is a problem, answers can be found. Some weaknesses include the learning curve and it can be difficult to set up.

For the front-end our team will be using JavaScript with a React framework. There are several benefits received when choosing React as the front end framework to write in. These include the experience the members of SDMay20-25 have with writing React. The majority of members have done web development with React and thus its use will speed up development time. Additionally, React provides a flexible framework which will allow the fronted to adjust to changes by the backend. One of the strengths of React is writing and reusing declarative components. If requirements change and components are modular and reusable then not all of the code will have to be rewritten. Finally, React allows for specific rendering of components. Rather than running a program and reloading a page when the data is retrieved, the component can be shown as loading until that data is populated, then only that specific component is re-rendered. While performance on the frontend is not a large concern this will help improve the user experience of using the end product.

We will also be using a statistic-specific algorithm portion. We will be using R for the forecasting and will be having it interface with the Spring backend. There is a connector to allow R programs to run and communicate back to Java application. One drawback to this is that there is another interface that will have to be considered and tested to ensure the data is transferred without loss and accurately.

3.3 Task Decomposition

The tasks are separated into two categories: backend and frontend.

1. Backend
    1.1. Local Database Setup. Having a local instance of the database running on SDMay20-25 development machines will be beneficial. The local database should have the provided dataset from Crafty imported as well
    1.2. Initial Server Setup. SDMay20-25 will get an AWS EC2 Server instance running for the web app hosting as well as the backend code running. SDMay20-25 will also need a server database, which will be hosted on AWS RDS instance. The EC2 instance and the RDS instance will need to be able to talk to each other.
    1.3. Initial API Setup. SDMay20-25 will set up a basic API that proves the backend server can get information from the database and return it through the API. It should also be able to receive information from the API and put it into the database.
    1.4. Prediction Environment Communication Integration. SDMay20-25 will test the framework for getting the predictions and make sure that it can get data from the database and that it can return the results back to the backend server so the data can be handled appropriately. There does not necessarily need to be a working prediction at this point; it should prove that we can communicate with the backend server and the prediction algorithm.
    1.5. Prediction Framework Integration with the Database. SDMay20-25 will want to make sure we can get the data from the database into the Prediction Environment so that we can manipulate the data more easily there and start to run some predictive algorithms.
    1.6. Test Dataset Development: SDMay20-25 will need to develop a test dataset or choose a window of test data from the provided database file where the desired prediction is known. This will allow SDMay20-25 the ability to test the algorithms and have an idea of what the algorithm should predict. Using this, SDMay20-25 will have an idea how different versions of the algorithm are running.
    1.7. Time Series Development: SDMay20-25 will need to get an initial version of a TimeSeries predictive algorithm working. This will involve getting the data, manipulating it into a workable form, running the Series Prediction and then reporting back the predicted result.
    1.8. Refining Inputs: The algorithm will be tested on different subsets of the input data to see what will give the most accurate prediction. This will help SDMay20-25 work to figure out which data points are most important to the order prediction.
    1.9. Testing. SDMay20-25 will have a few test sets of data where the expected prediction is known. We will run each iteration through with the test datasets and we will be able to identify how well that revision of the algorithm is working.

2. Frontend
    2.1. Create starter website in React: The first task of the frontend is to create the base website on which all development will be done. This is important as it gives all other SDMay20-25 team members the ability to work from the same starting point.
    2.2. Host starter website on server: Next, we must host a basic website on the server that will allow for pipeline creation and outside viewing of the project progress.
    2.3. Mock-up UI: After that, wireframing and UI mockup will allow pinning down the exact look and feel of the website. Additionally, this will allow for much faster implementation later. By doing this prior to implementation multiple people can work on multiple aspects of the frontend without constant collaboration.
    2.4. Define endpoints: We will need a well defined system of communication with the backend to the frontend and vice versa. We will accomplish this by creating an API document that defines what endpoints will be on the backend, what data they will give, and what data they will receive in what format.
    2.5. Fake JSON: An aspect of frontend development is quick iteration of data to check for visualls and errors. An easy way to do that is to have fake, differing, data on hand that would normally be sent by the backend. Additionally, by creating fake JSON, the backend team can better understand the required format.
    2.6. Static table components: Finally to implementation, here the table components will be defined and created. They will hold the fake JSON data received but not do dynamic updating.
    2.7. Make fields editable: Next, the editable fields of the table will be made as such. That data will also be sent off to the backend through the defined endpoint.
    2.8. Round trip edited data: This task simply ensures that all data sent from the frontend is saved on the backend and the frontend refreshes the data stays updated.
    2.9. Testing of data: Finally, testing will be done to ensure that the correct values populate each field from the fake data. Oracle will need to be written to ensure this is the case.

In terms of task interdependence, frontend and backend tasks can be worked on in parallel. Frontend can complete their tasks using fake JSON data. Backend can deliver endpoint data "on the fly"; frontend team will notify the backend team whenever a new endpoint needs to be created, and then the backend team will compile a list of these endpoints and create them on a weekly basis.

3.4 Possible Risks And Risk Management

SDMay20-25 has looked into several risks. The largest risk is not getting the software to work. For example, not being able to get the frontend and backend to communicate, or having an algorithm whose output is incorrect. This can be avoided by integrating early and integrating often. SDMay20-25 has carefully evaluated technology options that have influenced design decisions by thoroughly researching and communicating with the client Crafty and SDMay20-25's mentor Dr. Goce Trajcevski.

Another risk SDMay20-25 will have to deal with is the lack of understanding regarding the implementation of the solution. Few members have worked on machine learning or time-series forecasting. This risk can be mitigated by collecting various resources and researching relevant topics. The wiki on GitLab will be used to post findings from individual and group research.

A big risk in the project is that the initial model will not meet the needs of the client or it will not be accurate enough for the client's needs. In this case, we will have a couple of options to get a more accurate model. We may need to change the inputs that the formula uses. We may need to change the machine learning model that is used to generate the predictions. This could potentially mean that we would be forced to choose a different prediction model. The time investment could be large if a new machine learning model may be required. It would be trying different variables on the machine learning model to tweak things hoping for a better outcome.

3.5 Project Proposed Milestones and Evaluation Criteria

1. September 30th - Finalized Project Plans and Architecture

   This will include the first iteration of the intended tech stack that will be used. The most immediate design decision is the database; Crafty will send a sample database of a product warehouse to SDMay20-25. An instance of a server and backend will follow. This will establish the round trip connection. Adding a frontend component will be useful to visualize our data.

2. October 31st -Constraints and the impact on the design decisions . This deliverable report will list the project constraints at both software and system level. In addition, this report will include requirements that will influence our future implementation decisions. Constraints listed will allow for traceability in all of our design decisions. This will be a living document as we may be getting more requirements from our client as the project goes on.
3. November 30th - Methodology selection and tools . By this time, we will have selected our tools and justified why they will be the best for the task. We will then be able to start with building the finalized architecture and the finalized data flow for the project.
4. January 31st - Testing framework. During late January we will be providing the deliverable of a testing framework. This framework will allow for Test Driven Development as we complete

development tasks. Unit tests and automated tests will be detailed in this document. This document will also likely be a living document as our testing will likely evolve as our understanding of the project increases over time.

5. February 29th - Alpha version of the software. The initial version of the software will be the deliverable due in February. At this time we will have a working version of the system that shall satisfy the major requirements put in place by the client. This will be presented to the client, Crafty, for review and analysis of satisfaction of requirements. The alpha version will have a full round trip connection as a proof of concept.
6. March 31st - Unit testing and validation. This deliverable will consist of a report on the overall state of the unit testing and what it covers. The scenero previously created as well as many others will be created and tested on each component of the system. See Section 5 on testing for more details.
7. April 30th - Integration testing and report. The final deliverable will be a full integration test suite that will verify that all the systems work together and handle errors correctly. See Section 5 on testing for more details.

3.6 Project Tracking Procedures

SDMay20-25 will use GitLab issues are the primary method to track project progress. Each task found in section 3.3 will be posted onto GitLab as an issue, and each issue will include a due date, assignees, and a corresponding milestone. Tasks are generated by SDMay20-25 using work-breakdown system (WBS); tasks were broken down from our original proposal based on task interdependence and whether they were backend related or frontend related. These tasks are to be completed on a weekly basis; completed tasks usually produce software or written deliverables. It is important that each task is well written and thoroughly documented. Issues on GitLab has many features that will be useful to use including: labels, comments, due dates, markdown support, a board view (see Figure 3.6.1), milestones, and many other managing tools.

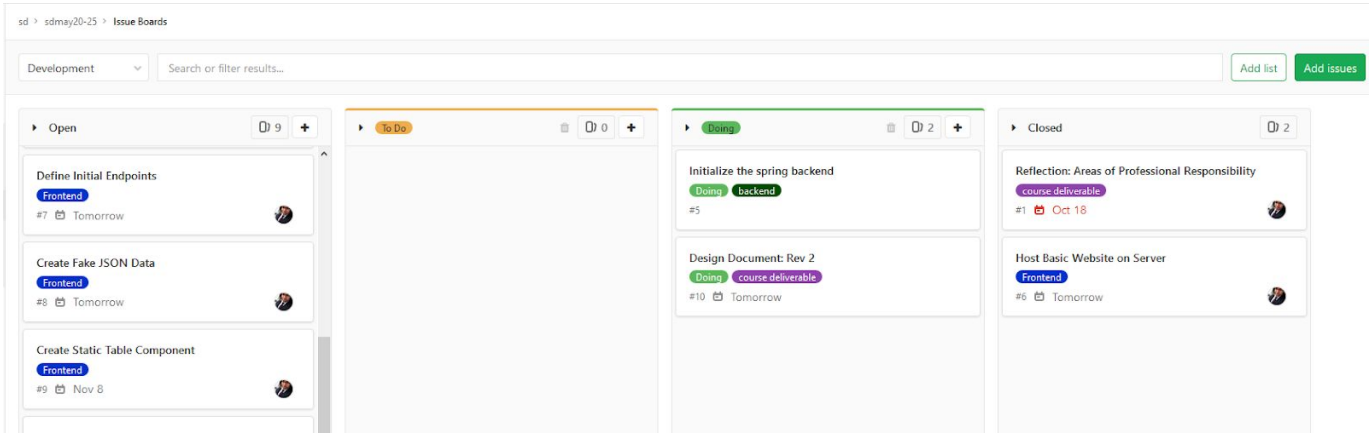


Figure 3.6.1: The Gitlab Board View used to track issues

Informal communication is handled using Slack. Any relevant information will be copied to the appropriate GitLab issue.

3.7 Expected Results and Validation

Crafty should expect a proof-of-concept that works, has been tested, and is proven to be a better solution than their current implementation. The Consumer-Aware Warehouse Management proof-of-concept will accurately predict the demand of the goods for the current ordering window while also trying to maximize profit. See the project statement in Section 1.2.2. This product will contain: a database, a backend, frontend, and will be hosted on a server. See Section 1.6 for our expected end product. We will know our solution is correct by the time it is ready to be delivered from the results of the tests we run. This is outlined in Section 5.

4. Project Timeline, Estimated Resources, and Challenges

In this section, the schedule of the project is discussed. The major points include: task breakdown, timeline, feasibility and effort hours required.

4.1 Project Timeline

The project timeline will be split into two semesters. In the fall semester, we will focus on project design development and core architecture implementation. In the spring semester, the focus will shift to development and testing of the core functionality, which is our prediction algorithm, and its integration within our architecture.

Week	Tasks
9/2-9/8	Setting Team Roles and Expectations
9/9-9/15	
9/16-9/22	Initial Requirement Analysis
9/23-9/29	Project Tech Stack Decisions and Research
9/30-10/6	Initial Project Scheduling and Design Doc Rev 1
10/7-10/13	Evaluating the project context in the existing client system
10/14-10/20	Requirement, Use Case, and Scenario Elicitations
10/21-10/27	Initial Backend Framework API
10/28-11/3	Design Doc Revision 2
11/4-11/10	Backend: Setting up the Queries for the frontend Solution
11/11-11/17	Backend: Starting an Initial Forecasting Model
11/18-11/24	Backend: Starting an Initial Forecasting Model
11/25-12/1	Thanksgiving - No Tasks
12/2-12/8	Presentation preparation and Final Revision of the Design Document.

12/9-12/15	Dead Week: Presentations

The purpose of the Fall semester is to lay the groundwork in preparation for the Spring semester, which will consist mostly of implementation.The prediction algorithm is the most complex part of this project, and will require the most resources in terms of research, development, and testing. In order for the implementation of the algorithm to go smoothly, we need to solidify the framework upon which this algorithm stands.

Tasks are to be completed weekly or bi-weekly with client input in between tasks. This follows the Agile software development process. See Section 2.3 Development Process for an example of a task being completed.

The design component of the framework includes constraints, requirements, and details about Crafty's operation and needs. These constraints and details are important for the development of a test scenario upon which we will base the development of our algorithm. These constraints will be collected through October 31st as seen in number 2 under Section 3.5.

Week	Tasks
1/13-1/19	Develop the testing Scenario
1/20-1/26	Develop the testing Scenario
1/27-2/2	Complete testing framework and re-evaluate design decisions
2/10-2/16	Integration of Forecasting Algorithm with Backend Server
2/17-2/23	Initial Revision of Forecasting Algorithm
2/24-3/1	Alpha Version Complete
3/2-3/8	Tuning Inputs to the Algorithm
3/9-3/15	Testing the Algorithm
3/16-3/22	Secondary Iteration of the Forecasting Algorithm
3/23-3/29	Tuning the inputs to the Algorithm; Unit Testing Complete
3/30-4/5	Testing the Second Iteration of the Algorithm.
4/6-4/12	Final Integration Testing
4/13-4/19	Final Tuning of the Algorithm

4/20-4/26	Documentation, Integration, and Final Testing
4/27-5/3	Final Presentation

The Spring semester is all about the prediction algorithm. There are three main components that need to be addressed in this phase of the project: The test scenario, the prediction algorithm itself, and the testing and analysis of the algorithm.

The test scenario will be developed using information received from discussions with Crafty, and is necessary to drive our development of the prediction algorithms.

Once we have out test scenario we will begin researching and developing the prediction algorithms. Our initial implementation will use linear regression. Once we have a working prototype with linear regression, we will branch out to machine learning techniques.

Finally, we will iteratively implement, test, analyze, and optimize these algorithms to get the best algorithm to suit Crafty's needs.

4.2 Feasibility Assessment

Our project will primarily be an experimentation with prediction methods for Crafty's product warehouse stock ordering. We will implement the prediction algorithms in a basic software system as the proof of concept of their use in Crafty's system. The main challenges with this project exist in the algorithm development and testing. As a team we have limited knowledge in machine learning. To account for this weakness, we have a portion of the work set aside for research into areas we are unfamiliar with. Developing a robust test scenario will also be a challenge in this project. Weekly discussions with our client will be utilized to get the most accurate and well rounded test scenario.

Another potential problem that could arise is the initial model will not meet the needs of the client or it will not be accurate enough. In this case, we will have a couple of options to get a more accurate model. The first option will be to modify which inputs and data points the model is using. We may need to adjust a number of things with regard to how much data should be given to the model. Another option would be to change the model used altogether. This would be a riskier approach since we would have to start the solution over, and the time investment would likely be much greater.

4.3 Personnel Effort Requirements

All tasks are referenced from the task decomposition in Section 3.3.

Task	Effort (Time in Hours)

1.1	5
1.2	5
1.3	15
1.4	15
1.5	20
1.6	40
1.7	40
1.8	35
1.9	20
2.1	2
2.2	8
2.3	10
2.4	10
2.5	10
2.6	20
2.7	20
2.8	10
2.9	20
	Total: 305

Table 4.3.1

4.4 Financial Requirements

At this point in time, we will use the free tiers of all software when possible. If the need for paid software arises, we will negotiate financial resources needed with Crafty.

5. Testing and Implementation

This section contains the testing plan for SDMay20-25. It discusses the methods, tools, and requirements testing environment.

5.1 Interface Specifications

The project does not have a hardware component of the project, so the only interfacing we will have to test is the communication between the backend, database, and the frontend. However, there are many software interfaces in the project. Referring to the Software Architecture Diagram (Figure 1.6.1), there are many different points of component interaction.

SDMay20-25 has a database which will interface with the Spring backend API. There is also the Data Forecasting Controller which is where the data forecast will run. This will communicate directly with the backend. The frontend display app will communicate with the Spring Backend App directly for all of its data requests. All of these interactions will have to be tested.

5.2 Hardware and Software Required

There will be little hardware required for the testing of the project since this is a software implementation. We will only require a computer to run the software we develop. This could be done on the server hardware we will be using, or if the computation overhead is not too large, testing can be completed on our own development machines.

SDMay20-25 has selected some software tools to aid the testing, and SDMay20-25 will be using some automated testing where we can. For the server component, JUnit testing for basic unit testing of the web server will be used. These should be run by the developers as the developer to ensure the basic functionality is always maintained. Gitlab CI/CD will be used for integration testing and deployment. All commits will be will have tests run to make sure the backend and frontend build. Unit tests will also run for backend code. The code will deploy to the server so the server when it successfully builds and passes tests so the code will always be running the latest software. The API testing can be automated with the assistance of some software as well. Postman will let us automate API calls and check if the results are as expected.

5.3 Functional Testing

On the backend unit testing will be used on all classes to ensure the class works properly. Integration testing will also be implemented with the interaction between the classes and ensure everything on

the backend is working properly. 85% branch and line coverage of the model will be expected with our test cases.

To test the backend API, SDMay20-25 will also be testing every endpoint to ensure that the data is formatted correctly. Postman can be used to automate the testing of the endpoints. Test calls using selected data with an expected result will be set up so it can be determined if the algorithm is giving accurate predictions.

To test the frontend, manual testing will be used. The frontend developers will be responsible for generating the test cases for the functions they create. They will run through the different functionality on the frontend to make sure that it is complete as they develop new features.

Also, test cases will be developed using the provided data to generate test data sets. The data will include a test case for when a product should be ordered for the product warehouse when the stock is low, and another when the stock is projected to be low before the next order can be placed. Another test case would be where the product should not be ordered and make sure the algorithm does not falsely suggest that product should be ordered.

5.4 Non-Functional Testing

SDMay20-25 will be testing for performance, security, usability and compatibility in addition to the functional tests above. This section will outline the test plan for these metrics.

Performance testing will be a priority for the project. The client will provide feedback for this metric. The software should run with performance that the client sees as acceptable. The software will be demoed to the client and they will tell us if the performance is sufficient or not. The client will tell us if our operations are running in an acceptable amount of time.

Security will be considered in the design. Measures will be taken to ensure data in transit is using appropriate security protocols. Security will not be an important factor for the data at rest. The client will again tell us if the practices are not up to their company’s security standards. SDMay20-25 will ensure access to the servers is properly secured so no unauthorized parties can gain access.

Usability testing will also be important to the client. The client will determine how usable our solution is for their procurement team. The client will evaluate how usable they see our system, and compare it to the existing system to see if it is more approachable for the team to understand the decisions that the prediction algorithm is making.

Compatibility testing will help the client determine if our implementation will work well with their existing system. SDMay20-25 will check that the version of software used is compatible with the client’s version of software. This should ensure that the solution will work with the client’s software.

The software will also be tested with an anonymized version of the client's database so that we can see how the software would perform with real data from the client.

5.5 Process

Coming in the final revision of the document.

5.6 Results

We have not completed any tests at this point in time.

6. Closing Material

In this section, there is a summary of the project, References from the various sections of the document and the included appendices.

6.1 Conclusion

The project aims to maximize profit for Crafty by ensuring the product warehouse is adequately stocked for new customer accounts and existing customer demands. It will also reduce waste by ordering just enough until the next order can be placed, so nothing will expire, be thrown out, or waste space in the product warehouse so the maximum amount of product possible can be held in the product warehouse.

Our solution will use Data Forecasting to use the inputs from the past orders, incoming orders and current stock to recommend the ideal amount of product to order from a given distributor at a given time. This will allow us to make a relatively accurate model with the relatively small dataset we have. It will also allow for external inputs, like new customer onboarding where the initial consumption will be higher than the expected weekly consumption.

6.2 References

[1] B. Delahaye. “How Artificial Intelligence in Supply Chain Planning Is Changing Retail and Manufacturing.” NeuroChain. https://www.neurochaintech.io/artificial-intelligence-supply-chain-planning/ (accessed Oct. 6, 2019)

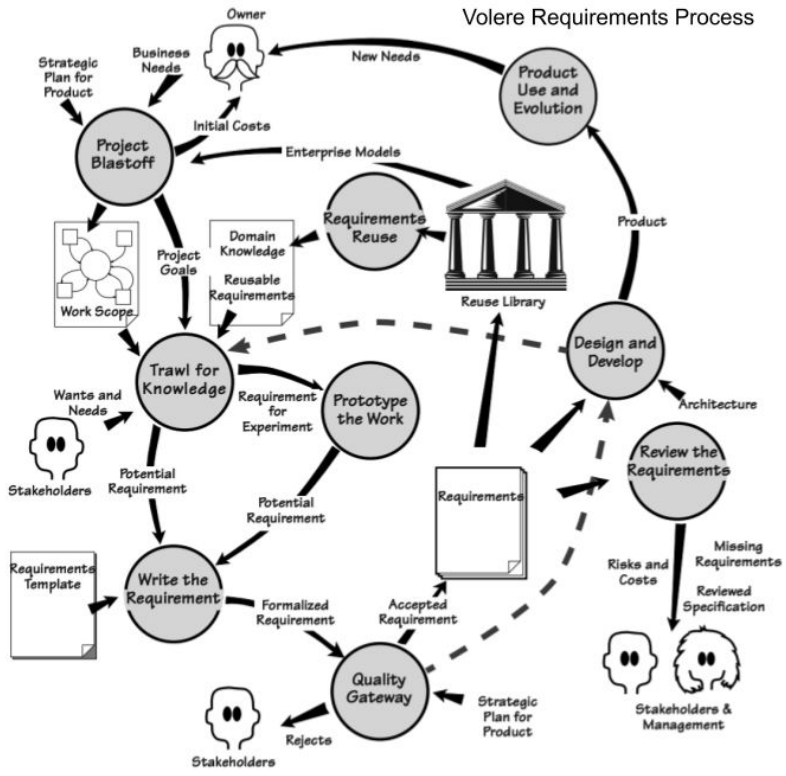
[2] “Supply Chain Resource Cooperative Single Regression Approaches to Forecasting A Tutorial.” North Carolina State University. https://scm.ncsu.edu/scm-articles/article/single-regression-approaches-to-forecasting-a-tutorial (accessed Oct. 6, 2019)

[3] “All the Risk Assessment Matrix Templates You Need” smartsheet.com. https://www.smartsheet.com/all-risk-assessment-matrix-templates-you-need (accessed Oct. 6, 2019)

[4] Roberson, Suzanne. “Volere Requirements Specification Template.” Mastering the Requirements Process, Getting Requirements Right, edited by James Roberson, 3rd ed., Addison-Wesley, 2013, pp. 393–471.

[5] A. Mavin, P. Wilkinson, A. Harwood and M. Novak, "Easy Approach to Requirements Syntax (EARS)," 2009 17th IEEE International Requirements Engineering Conference, Atlanta, GA, 2009, pp. 317-322.

[6]



6.3 Appendices

Appendix 6.3.1 Database Schema Diagram