Consumer Aware Warehouse Management

Design Document

SDMay20-25

Client: Jimmy Paul

Advisor: Goce Trajcevski

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

Team Email: sdmay20-25@iastate.edu

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

Revised: April 9, 2020, Version 4

Executive Summary

SDMay20-25 is working with Crafty LLC, a company that provides offices with food, beverages, and event management. It is estimated by Crafty that $600,000 worth of sales are missed annually due to insufficient stock, and $100,630 worth of annual profit is lost due to expired products. In addition, three full-time employees dedicate 50% of their time to evaluating product stock that should be ordered. Crafty is seeking a product forecasting algorithm to predict how much of each product should be stocked in Crafty's product warehouses to account for their client's needs. Crafty is experiencing issues with their current solution because it takes into account only four input variables when they actually have many more available. SDMay20-25's will develop a forecasting algorithm that will strive to reduce (1) waste of expired products, (2) missed sales due to insufficient inventory, and (3) erroneous and time-consuming labor efforts. SDMay20-25 has analyzed the data provided by Crafty and has determined the limits of the current solution and identified additional input variables.

SDMay20-25 will develop a forecasting algorithm, Consumer Aware Warehouse Management System, to determine how much of each product should be stocked in Crafty's product warehouses. The forecasting algorithm will take four input variables that Crafty is currently using and incorporate up to an additional seven input variables. The output of the forecasting algorithm will generate a report detailing which products should be ordered. These results will be displayed in a table, where Crafty will be able to see the reasoning behind the decisions that the algorithm made, as well as, make adjustments to the input variables.

Below, SDMay20-25 has briefly listed the requirements, development standards, practices, and applicable courses for the development of the project.

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

Engineering Standards and Design Practices

Software requirements and design process artifacts, team workflow, and project management, and engineering standards are briefly listed below.

Software Requirements and Design Process Artifacts

- Architecture Diagram
- Class Diagram
- Context Diagram
- Use Case Diagram
- Model View Controller Design Pattern
- Code Review

Team Workflow and Project Management

- Agile Methodology
- Gantt Chart
- Iterative Methodology
- Risk Assessment Matrix
- Work Breakdown Structure
- Xtreme Programming

Applicable Courses from Iowa State University Curriculum

The following list emphasizes the curriculum at Iowa State University whose contents directly apply to the main technical components in this project.

- Com S 227, Intro to Object-Oriented Programming in Java
- Com S 228, Intro to Data Structures in Java
- Com S 252, Linux Operating Essentials
- Com S 309, Software Development Practices
- Com S 311, Intro to Algorithm Design and Analysis
- Com S 362, Object-Oriented Analysis and Design
- Com S 363, Intro to Database Management Systems
- Com S 409, Software Requirements Engineering
- Com S 417, Software Testing
- SE 319, Software Construction and User Interfaces
- SE 329, Software Project Management
- SE 339, Software Architecture and Design

- DS 201, Intro To Data Science
- ENG 314, Technical Communication

New Skills / Knowledge Acquired Not Taught in Courses

The following list emphasizes the main technical components in our project that have not been captured through Iowa State University coursework. Knowledge of these topics will be acquired through research, experimentation, and implementation.

- Machine Learning-Based Algorithms
- Frontend Development Frameworks
- Regression-Based Algorithms

Table of Contents

List of Figures and Tables

Definitions

1. Product Warehouse

   A location where products are stored before they are shipped to customers.

2. Product Forecasting

   The work of predicting the future need of a product given input variables such as historical consumption of a product and knowledge of any future events that might impact the forecasts such as weather or seasonal products.

3. Distributor

   A wholesaler of goods that Crafty orders from to supply its warehouses with stock to later distribute to their clients.

4. Stock Keeping Unit (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.

5. Product Order Table

   A table generated by Crafty to display what needs to be ordered from distributors to maintain the proper stock in the Crafty warehouses.

1. Introduction

SDMay20-25 is working with Crafty LLC, a company that provides offices with food, beverages, and event management. It is estimated by Crafty that $600,000 worth of sales are missed annually due to insufficient stock, and $100,630 worth of profit annually is lost due to expired products. In addition, three full-time employees dedicate 50% of their time to evaluating product stock that should be ordered.

Crafty is seeking a product forecasting algorithm that improves upon the existing algorithm to predict how much of each product should be stocked in Crafty's product warehouses. SDMay20-25 has analyzed the data provided by Crafty and has determined the limits of the current solution and identified additional input variables needed to make successful improvements.

1.1 Acknowledgement

SDMay20-25 would like to acknowledge and thank the following:

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.

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.

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.

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

SDMay20-25's client, Crafty, desires a product forecasting algorithm to predict how much of each product should be stocked in Crafty's product warehouses.

Crafty is experiencing issues with their current solution because it is only taking into consideration four input variables when they actually have many more. The fact that the solution is not incorporating more of the available variables results in:

- Waste of expired product
- Missed sales due to insufficient inventory
- Erroneous and time-consuming labor efforts

The software solution SDMay20-25 plans to develop will incorporate up to seven additional variables, which will enable Crafty to accommodate product needs from (1) current customers and (2) any seasonal or sudden, known changes outside of regular needs. The priority of these variables to be incorporated will be determined by Crafty.

The software solution, that Crafty currently maintains, generates a distributor purchase report. The report gives Crafty an itemized list of products to order from each distributor. As mentioned previously, when generating a distributor purchase report, the current software solution only takes into account the following four input variables:

- Client order history
- Distributor schedule and lead time
- Missed sales
- Client inventory reorder thresholds

The current solution leaves a lot of room for improvement. Specifically, the following input variables are known but not utilized for each product:

- Consumption trends for each product
- Seasonality, or other sudden known changes, that determines product demand
- Accuracy of distributor delivery windows

Incorporating the mentioned input variables would help Crafty improve their solution and give them more opportunities for revenue. It is estimated by Crafty that $600,000 worth of sales are missed annually due to insufficient stock, and $100,630 worth of annual profits are lost due to expired products. In addition, three full-time employees dedicate 50% of their time to evaluating product stock that should be ordered. In response to this problem, SDMay20-25 will build a software solution that will incorporate the same input variables that Crafty is currently using, as well as, additional input variables. Incorporating additional input variables will help to maintain an accurate and robust system to predict the products that Crafty must reorder.

The overall use case for Crafty's current operation is shown below in Figure 1.1. While undoubtedly simplified, it serves to define the context of which the Consumer Aware Warehouse Order Management system will be operating in.

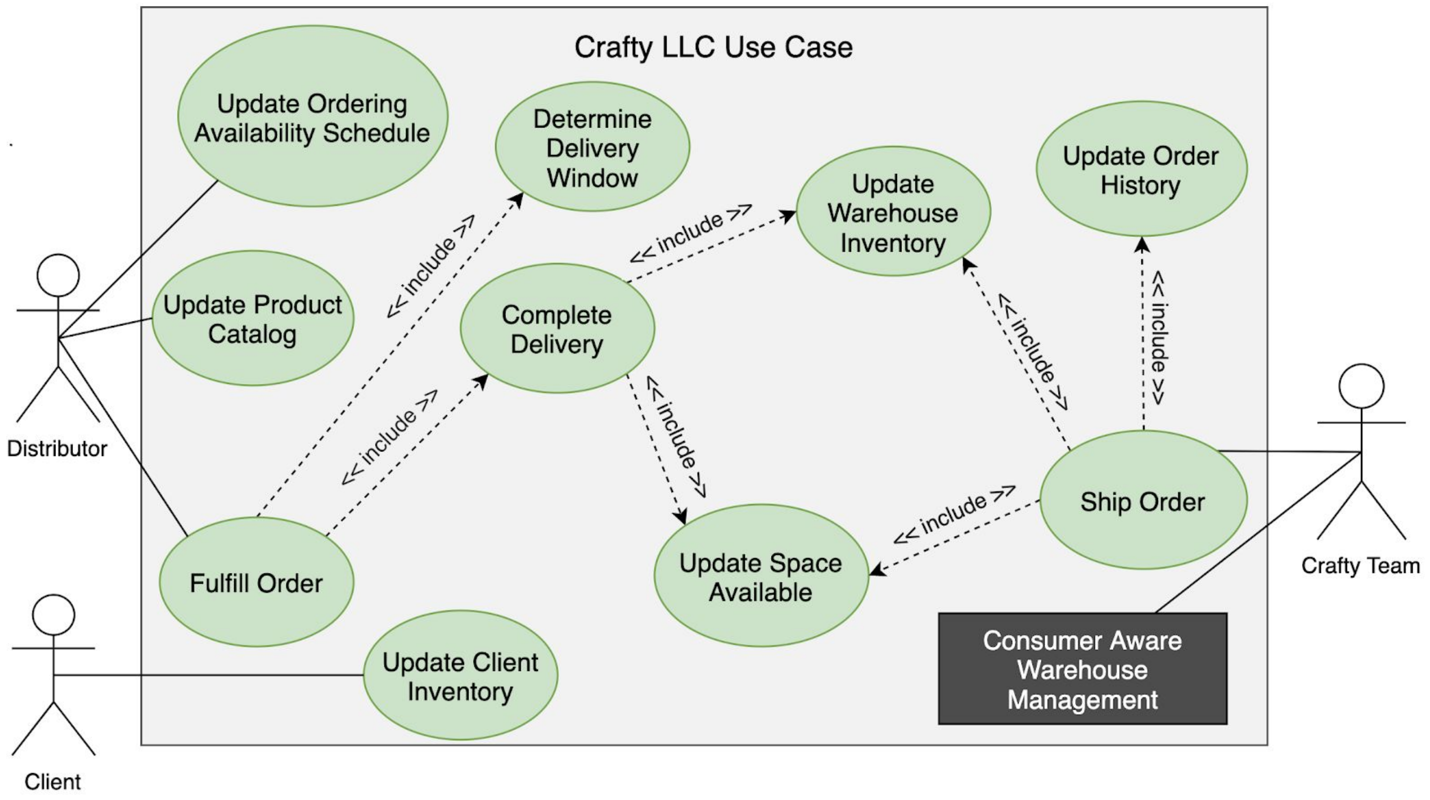


Figure 1.1 - Crafty LLC Use Case Diagram

We see the system-to-be to be internal to the Crafty team, yet it will function alongside the numerous adjacent systems already in place shown in Figure 1.2. Thus, the Consumer Aware Warehouse Management system will not be a core component of Crafty's operation but will function as an advisory tool.

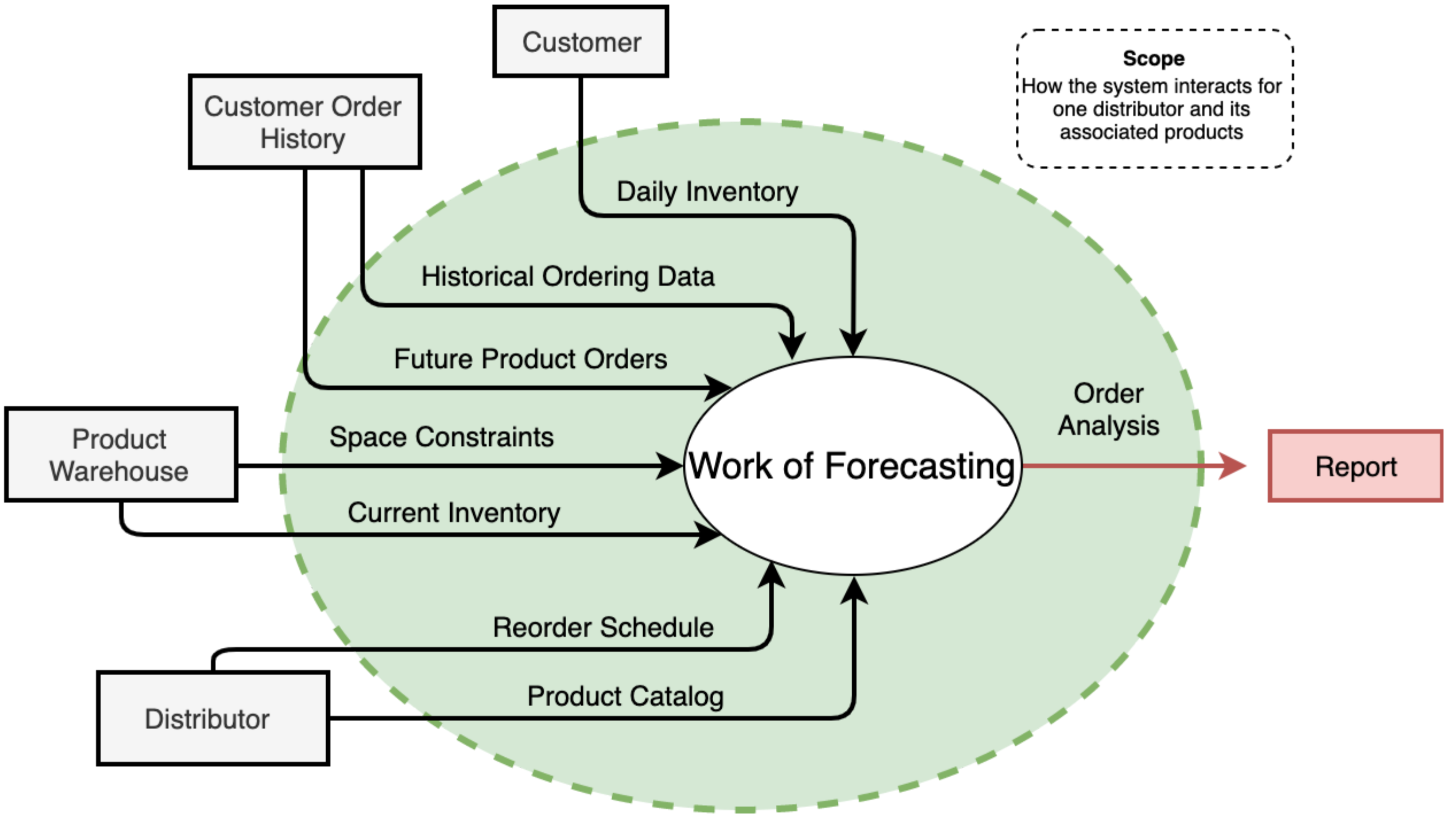


Figure 1.2 - Work of Forecasting Context Diagram

Figure 1.2 provides scope for the project and shows how the data flows to and from the adjacent systems, which are the categories of input variables that directly influence the work of forecasting. The scope of this diagram is how the system interacts with all available products. It can be seen that the forecasting algorithm, shown as the work of forecasting, will take into consideration input variables from customer order history which includes future product orders and historical ordering data; product warehouse data such as space constraints and current warehouse inventory; daily inventory of each customer; and distributor information such as the products available by that distributor, which accounts for seasonal items, pricing of items, and the reorder schedule which provides turnaround time from product order to delivery. All of these input variables are pulled from the Crafty database to be input into the forecasting algorithm which will generate a report of what Crafty should order from that distributor.

The product use case diagram shown in Figure 1.3 shows the main use case of viewing the distributor purchase order. The user Crafty selects a distributor to view a purchase order for. The forecasting algorithm is run, when prompted, and outputs products that must be reordered and stores the predictions in the database. The frontend then pulls this prediction information from the database. These results will be displayed in a product order table which will provide the total number to reorder for each product and the reasoning behind why that prediction was made. This diagram would be placed where the black box labeled “Consumer Aware Warehouse Management” is in Figure 1.1. It is placed in a separate diagram for allowing for more detail and separation of the work of forecasting from the overall Crafty system.

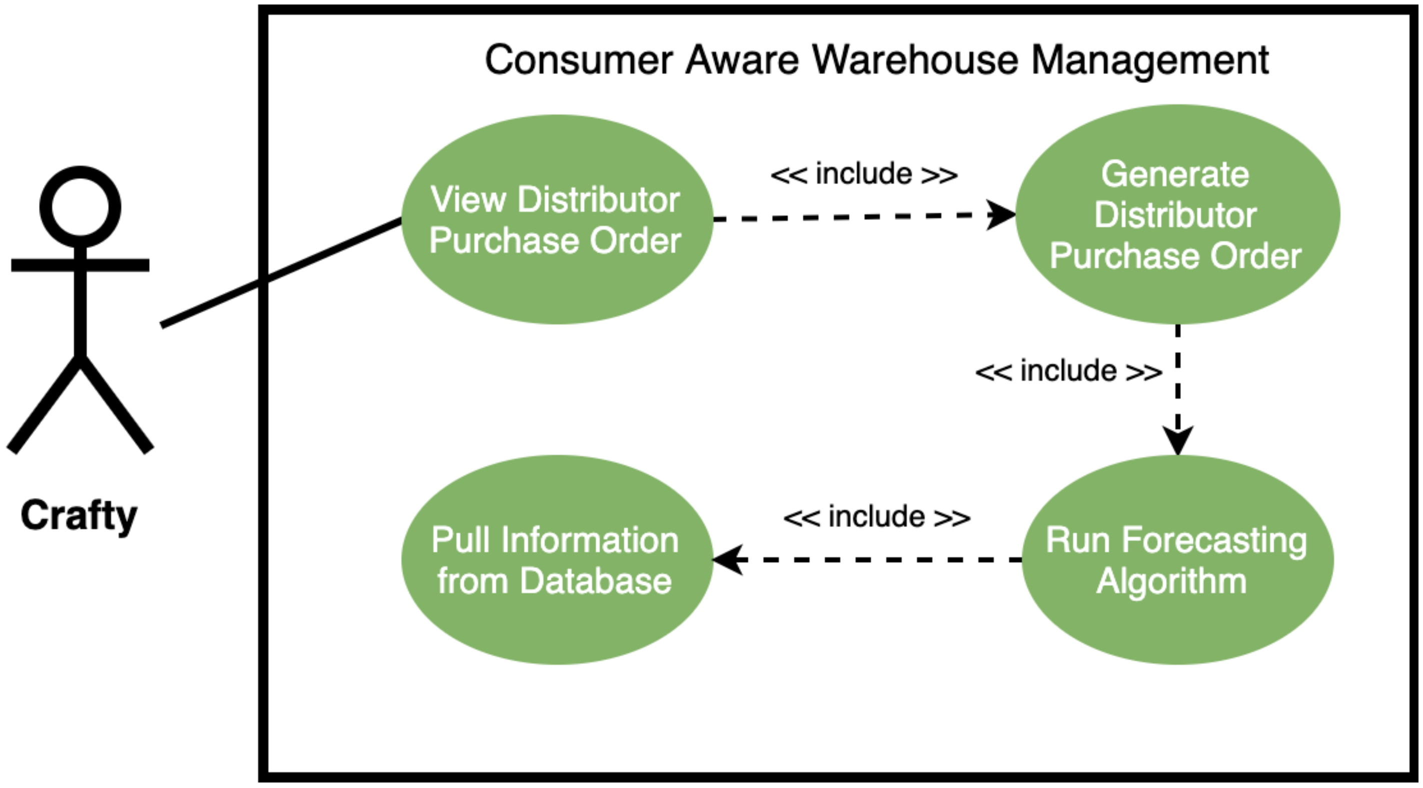

Consumer Aware Warehouse Management
Crafty
View Distributor Purchase Order
<< include >>
Generate Distributor Purchase Order
<< include >>
Run Forecasting Algorithm
<< include >>
Pull Information from Database

Figure 1.3 - Product Use Case Diagram

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

- Team Meeting Notes
- Bi-Weekly Status Report Updates
- Architecture and Design Plan Documentation
- Design Document (final and revisions)
- Senior Design Website
- A Functional and Improved Product Forecasting Algorithm
- A User Interface to Display Product Order Table and Modify Input Variables

1.3 Requirements

The following functional and non-functional requirements are defined within the project context to be demonstrated for successful project completion.

1.3.1 Functional Requirements

1. The system shall take data Crafty's database as input for creating a distributor purchase report.

   The software solution must take data from Crafty's database as input for the product forecasting algorithm so that it can accurately predict quantities of products Crafty must reorder from each distributor. This requirement is seen in Figure 1.1 as the "Pull Information from Database" use case.

2. The system shall optimize ordering for each product.

   This requirement is shown by the adjacent systems in Figure 1.2 as input variables for the work of forecasting. These input variables will be used to generate a distributor purchase report as output.

3. The system shall take existing input variables.

   In order for the product forecasting algorithm to perform better than the existing algorithm, it must take into consideration the current input variables, as well as, additional input variables. This is discussed in Section 1.2.

4. The system shall have a user interface to display the rationale for output

   The Crafty procurement team needs a way to interface with the system tool to view the distributor product ordering report. This is documented in Figure 1.3 as the "View Product Order Table" which 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 Stock Keeping Units a day

   The system in place that Crafty is using is able to handle this number of SKUs in 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 minutes 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 within a reasonable time frame. The client has expressed that a longer execution time can be acceptable if it means saving considerable employee effort. In Figure 1.3, this product use case can be seen as the “View Product Order Table” use case.

Please note that in addition to the listed requirements, SDMay20-25 recognizes that there are additional non-functional requirements such as security and privacy. This project will not be used in a production environment, and for that purpose, those requirements are out of scope and will be handled accordingly by SDMay20-25’s client Crafty.

1.4 Intended Users and Uses

The software solution proposed in Section 1.2 is designed and intended for usage by Crafty as a solution to the problem. The solution will use a larger set of data than what is currently being used, and will enable Crafty to maximize reduce the number of missed sales, expired products, and redirect their workforce. The Crafty procurement team will use the system to generate and modify a product order table by interacting with the user interface. As shown in the use case diagram Figure 1.3, the product order report will then be used to generate a distributor purchase order.

1.5 Assumptions and Limitations

Assumptions and limitations for the project are detailed below. It is intended 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. Design decisions as such will be well documented in Section 2.1.

1.5.1 Assumptions

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

  The context diagram shown in Figure 1.2 implies that in order for the software solution to forecast the products that must be ordered, the system must maintain communication and must have access to input variables existing in the Crafty database.

- 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 to manage the inventory of a single product warehouse.

  This system will not be used in a production environment and the scope has been limited by showing that the algorithm will work for one product warehouse.

- The system will only predict products to be reordered, not what distributor to order them from.

  This is because ordering from multiple distributors may degrade the performance of the system and orders are placed individually by a distributor.

1.5.2 Limitations

- The accuracy of system predictions is completely reliant on the accuracy of input variables given by the database.

  The context diagram shown in Figure 1.2 shows the output of the software solution is based entirely on the input variables pulled from the Crafty database. If the data in the database is incorrect for any reason, the result cannot be guaranteed to be correct.
- The database will match Crafty's implementation.

  The software solution is largely based on the input variables provided by the Crafty database. Crafty requires that a minimum version of PostgreSQL version 11 is used for easy integration of the solution with their existing solution and to avoid any data loss.

1.6 Expected End Product and Deliverables

The software solution will consist of several deliverables to Crafty and Iowa State University's Senior Design program. These deliverables are discussed in Section 1.2 and will be delivered upon project completion. The primary deliverables to Crafty will be a Design Document and the software solution implementation.

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.

2. Specifications and Analysis

SDMay20-25 has developed the following design implementation plan for the software solution. The design and implementation are developed in response to the problem statement in Section 1.2 and is inclusive of functional and non-functional requirements defined in Section 1.3.

2.1 Proposed Design

The following development technologies have been chosen for the development of the software solution in response to the requirements defined in Section 1.3. The System Architecture Diagram shown below in Figure 2.1 lists the technologies used for development, as well as, the flow of communication between systems.

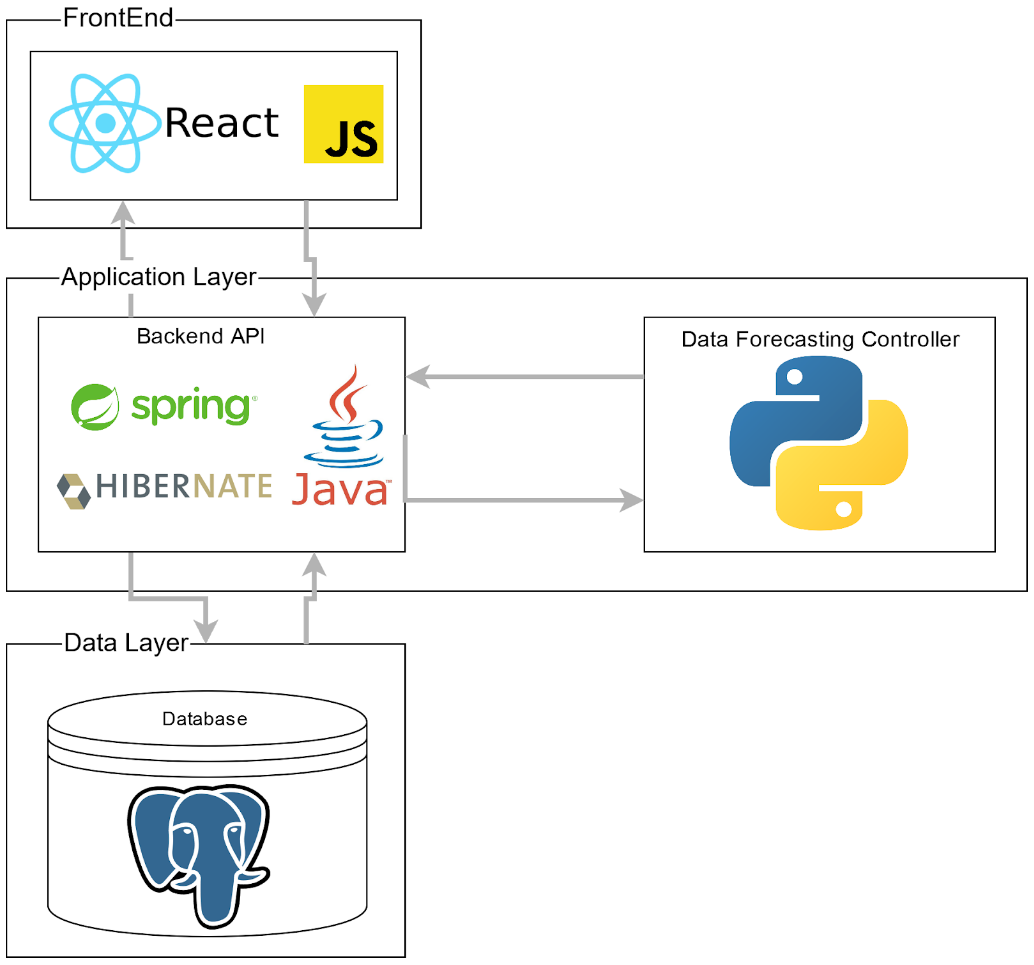


Figure 2.1: System Architecture Diagram

In accordance with functional requirement 1, SDMay20-25 must obtain anonymized data from Crafty as SDMay-20-25 will need access to the data on demand. In order to meet this requirement, SDMay20-25 will use a PostgreSQL database which will contain the data from Crafty. See Appendix A for a diagram of Crafty's existing database schema and relation. The completed database will enable SDMay20-25 to produce an initial website which will allow Crafty to view the output from the algorithm.

There are several potential solutions to address the main functional requirements 1.3.1.2. It is required that the forecasting algorithm be able to predict the minimum quantity of a product to order, given the same input variables Crafty is currently using, as well as, additional variables mentioned in Section 1.2. SDMay20-25's approach would be a time series based approach that would take several input variables and calculate the demand for a product. In order to test and demonstrate the system, the team will need to develop a frontend application with the functionality to display the results of the algorithm

We will solve these problems using an MVC design pattern, where the Model will be a PostgreSQL database, the controller will be a server running Java Spring Hibernate, the data forecasting controller will be using Python Pandas, and the view will be written using ReactJS. The system architecture diagram detailing this can be seen in Figure 2.1.

2.2 Design Analysis

As mentioned in Section 2.1, SDMay20-25 will use PostgreSQL in the data layer to host Crafty's data and to interface with the application layer. This decision was made for a variety of factors, with major emphasis being that it is the database system that Crafty currently uses. Other contributing factors include an open-source framework and team familiarity with MySQL scripting language. PostgreSQL was chosen since it is similar to MySQL, but expands upon functionality and will make delivery and integration of the final product easier for Crafty.

In the application layer, SDMay20-25 has decided to use Java with a Spring-hibernate framework. 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. Additionally, all members of the team are familiar with the Java software suite and Spring.

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 is an alternative to pandas. 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 a time series prediction algorithm. 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 time series as it is a simpler method for prediction. Additionally, there may not be enough data for an effective model for deep learning.

For the frontend design decision, SDMay20-25 decided to utilize the React framework. React is a component-based javascript library that allows each aspect of the application to control its own state. This means that pages can load dynamically in a single page application. This is important for larger applications because it enables user interaction before full page load. React was also selected for its familiarity among team part of the team thus starting development streamlined with fewer roadblocks.

2.3 Development Process

SDMay20-25 will adhere to Iterative development processes. This method of development was selected over the Waterfall development process due to the iterative nature of the work and the potential for variable requirements. Iterative was selected over Agile because the requirements are defined at the beginning of the project. Requirements will likely not be added as the project progresses; however, as the solution is developed, there will be several rounds of analysis on the solution to determine the accuracy. Each change will represent an iteration and the previous iteration will be used to make the next iteration better. The Iterative methodology is frequently used in industry and is effective for the development of projects in which a high level of adaptability is induced by changing variables.

The project will be completed in development cycles with regular customer interaction and feedback. Regular customer interaction and feedback will be demonstrated by hosting bi-weekly meetings with Crafty to demonstrate components of the solution and iteratively adjust the requirements based on their needs to ensure adherence to the requirements that Crafty has defined for the project, minimize the number of missed requirements, and reduce development efforts that are not meaningful.

SDMay20-25 has broken into development groups and the groups were assigned tasks for completion between each cycle. These tasks are outlined in Table 3.1. Each task has been broken into subtasks that can be implemented in bi-weekly cycles. An example cycle entails the following:

1. At the beginning of the cycle, SDMay20-25 has defined upcoming tasks, estimated time for completion, and selected tasks to be completed for the upcoming cycle. SDMay20-25 identified that:
    a. The server and database needs to be 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 the 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.

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

The following design plan is based on the functional and non-functional requirements described in Section 1.3, users and use cases in Section 1.4, and assumptions and limitations in Section 1.5.

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. Run Forecasting Algorithm
4. Pull Information from Database

These use cases are represented in Figure 1.3 in the product use case diagram.

SDMay20-25's software solution will consist of three main systems, each with a number of subsystems. The layers of SDMay20-25's system are the database, the server (backend), and the frontend. These systems will communicate through database queries and REST calls. This integrates with the use cases shown in Figure 1.3 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.

A time-series prediction framework will be used for the initial approach to solving the problem. Time-series predictions are especially useful for datasets where the goal is to make a prediction over a period of time. The time series can also be useful for identifying seasonal trends, which Crafty has identified as a priority in their desired prediction algorithm. As development continues, the solution will be evaluated (more details are explained in Section 5.3) If the results are not satisfactory, changes will be made to inputs or other prediction parameters. In a worst-case scenario, another model may be evaluated and explored.

Each part of the system allows the SDMay20-25 team to implement a solution to the use cases shown in Figure 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 the 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

SDMay20-25 has completed market research to understand the products and technologies currently available. Market research helped shape the appropriate technologies selected and develop a set of tasks for developing the forecasting algorithms. Tasks were derived from the provided milestones and evaluation criteria. Once the tasks were defined, personnel assignments were made, project risks were evaluated, and expected results were defined.

3.1 Previous Work And Literature

The following paragraphs will discuss two different approaches that are within the scope of SDMay20-25's solution. These different approaches include a machine learning approach that is being used by Walmart, and a regression-based approach.

The machine learning approach, as outlined in Artificial Intelligence in Supply Chain Planning is Changing Retail and Manufacturing by Bruno Delahaye, uses machine learning and AI to predict the demand of a product based on several factors such as: (1) weather, (2) seasons, (3) holidays, (4) promotions, and (5) past sales data. The article gives an example of showing how weather trends impact the sale of steaks and hamburgers. If the weather is warm and sunny, the sale of hamburgers is higher. Otherwise, if the weather is cloudy, the sale of stakes is higher. This approach also takes into consideration the shipping time it takes for the product to reach the warehouse and store. It does this by looking into the distance, road conditions, weather, and the capacity of the shipping container [1].

The advantage of this machine learning approach is that it provides the ordering of products with better accuracy to the demand. Additionally, the machine learning approach reduces cost and error-prone human interaction, which results in efficiency and cost-effectiveness. The disadvantages of this approach are the size and amount of resources needed to maintain. This relates to SDMay20-25's project because the machine learning approach 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 approach takes in more factors such as weather, promotions, and road conditions.

The other 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. This regression-based approach is able to predict the demand based on past sales and the seasons. This approach takes the input variables and is able to calculate the demand for a given day during a season using variable linear equations [2].

The advantage of this approach is the small number of resources needed to build and maintain it because it doesn't need much data to operate. One disadvantage of this approach is that it does not take into account spikes in demand due to factors that aren't based on past sales, such as weather. Another disadvantage is that 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 this approach takes into consideration past data. It differs from SDMay20-25's solution because this approach doesn't take into consideration the shipping time to get the product to the product warehouse.

3.2 Technology Considerations

SDMay20-25 is using PostgreSQL for the database. SDMay20-25 chose this because the queries are similar to MySQL which our team has experience with and are comfortable using relational databases. PostgreSQL also is the software our client uses so it will be easy to import their data and integrate it with their system. PostgreSQL also has high reliability because it doesn't allow users to bypass the data checks, which makes sure it is valid data. Whereas, with MySQL, a user could bypass data checks, prior to version 5.0. PostgreSQL is open source so there is no licensing and free.

For the server, SDMay20-25 is using Java with Spring and Hibernate Frameworks. Java, Spring, and Hibernate are taught at Iowa State University and will allow the team to develop quickly due to the support and familiarity that can be provided. Additionally, there is a large online community where answers can be found. A drawback to this software is the learning curve as it can be difficult to set up.

For the frontend, SDMay20-25 will be using TypeScript (JavaScript superset) with a React framework. The react framework provides benefits such as flexibility, modularity, rendering efficiency, and team experience with the framework. TypeScript will allow us to write a flexible interface that can adjust to changes by the backend. Additionally, one of the strengths of React is writing and reusing modular components that can be reused by passing different properties to it. If requirements change and components are modular and reusable then not all of the code will have to be rewritten. Also, 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. Finally, SDMay20-25's frontend team has done web development with React and thus, will speed up the development time and decrease the learning curve.

SDMay20-25 will be using a statistic-specific algorithm. SDMay20-25 will be using Python Keras and tensorflow for the forecasting and will be having it interface with the Spring backend. There are connectors that will allow Python programs to run and communicate back to a Java application. Specifically, we used API calls that allowed the Python prediction algorithm to store its predictions once they are made. One drawback to this is that it is another interface that will have to be tested to ensure the data is transferred accurately and without loss.

3.3 Task Decomposition

The tasks are decomposed into three categories: requirements, backend, and frontend.

The frontend and backend teams will work synchronously which allows for a minimal amount of task interdependence. This means that each task for the frontend and backend should be completed around the same time. Tasks are distributed to each team such that they correspond to the current milestone.

Team Members	
Elijah Buscho (**EB**)	Omair Ijaz (**OI**)
Jameel Kelley (**JK**)	Lindsey Sleeth (**LS**)
Andrew Smith (**AS**)	Sam Stifter (**SS**)
Devin Uner (**DU**)	

Task		Description	Dependencies	Team Member (s)	Effort Hours
1 Requirements Elicitation & Project Planning					
1.1		Define Team Collaboration Policies		JK, LS, SS, OI, EB, AS	6
	1.1.1	Role Assignments		JK, LS, SS, OI, EB, AS	6
	1.1.2	Development Workflow		JK, LS, SS, OI, EB, AS	6
	1.1.3	Communication Policy		JK, LS, SS, OI,	6

					EB, AS	
1.2			Client Background		JK, LS, SS, OI, EB, AS	12
1.3			Meet With Client to Understand Problem		JK, LS, SS, OI, EB, AS	24
	1.3.1		Create Problem Statement		JK, LS, SS, OI, EB, AS	6
		1.3.1.1	Context Diagram		JK, LS	4
		1.3.1.2	Product Use Case Diagram		LS, EB	4
	1.3.2		Obtain Supporting Information About Problem From Client		JK, LS, SS, OI, EB, AS	12
		1.3.2.1	Define Assumptions and Limitations		JK, LS	2
		1.3.2.2	Financial Requirements		JK	1
		1.3.2.3	Risk Matrix		OI, EB	4
1.4			Define Functional and Non-Functional Requirements	1.3	JK, LS	12
1.5			Define Milestones		OI, SS	2
	1.5.1		Project Tracking Procedures		OI, SS	10
	1.5.2		Evaluation Criteria		OI, SS	4
1.6			Proposed Design	1.3, 1.4	JK, LS, SS, OI, EB, AS	8
	1.6.1		Research Technology		JK, LS, SS, OI, EB, AS	30

	1.6.2		Market Research		AS	4
	1.6.3		Design Plan	1.6.1, 1.6.2	JK, LS, SS, OI, EB, AS	30
		1.6.3.1	System Architecture Diagram		SS, JK	4
		1.6.3.2	Class Diagram		EB, LS, OI	9
	1.6.4		Define Development Process		SS, OI	10
1.7			Design Analysis	1.6	AS, LS	6
1.8			Task Decomposition		JK, LS, SS, OI, EB, AS	6
	1.8.1		Work Breakdown Structure		LS, OI, SS, JK	24
	1.8.2		Personnel Effort Requirement		JK, LS, SS, OI, EB, AS	6
	1.8.3		Gantt Chart		OI, EB	4
	1.8.4		Project Timeline		OI, EB	4
	1.8.5		Team Task Assignments		JK, LS, SS, OI, EB, AS	10
1.9			Define Test Plan	1.6, 1.8	JK, LS, SS, OI, EB, AS	6
	1.9.1		Interface Specification		SS	2
	1.9.2		Hardware and Software Required		SS	1
	1.9.3		Functional Testing		SS, JK	6
	1.9.4		Non-Functional Testing		SS, LS	6
	1.9.5		Overall Process		JK, LS, SS, OI, EB, AS	12

	1.9.6	Expected Results		JK, LS, SS, OI, EB, AS	24
Total Requirements & Project Planning					333
2 Backend					
2.1		Setup Development Environment	1.6	AS, OI, SS	6
2.2		Local Database Setup	1.6, 2.1	AS, OI, SS	6
	2.2.1	Importing Client Data		AS, SS	5
2.3		Initial Server Setup	1.6, 2.1	AS, SS	10
	2.3.1	Install Required Packages		SS	2
	2.3.2	CI/CD		SS, OI	6
	2.3.3	Access Control		SS	2
2.4		API Setup	1.6, 2.1, 2.2, 2.3	AS, SS,OI	10
	2.4.1	Create Initial Endpoints		AS, SS	10
	2.4.2	Iterate On Endpoint Development		AS, SS, OI	6
2.5		Provide Algorithm With Required Data	1.6, 2.1, 2.2, 2.3	AS, SS, OI	9
	2.5.1	Establish Round Trip Communication With Spring API Controller		AS, SS, OI	9
2.6		Test Dataset Development	2.2, 2.3	OI, SS, LS	44
	2.6.1	Identify Key Dates of Special Events In History		SS, LS	10

	2.6.2		Split Dataset Into Training and Evaluation		SS, OI	6
2.7			Algorithm Development	2.6	EB, DU	76
	2.7.1		Evaluate Algorithms		EB, DU	10
		2.7.1.1	Analyze Usefulness of Existing Input Variable		EB, DU	6
		2.7.1.2	Speculate Usefulness of Additional Input Variables		EB, DU	8
	2.7.2		Acceptance Testing		Everyone	20
Total Backend						261
3 Frontend						
3.1			Setup Development Environment	1.6.1	LS, EB, JK	6
3.2			Mockup UI	1.4, 1.6	JK, LS	4
3.3			Create Website Using React	1.4, 1.6, 3.1	JK, LS, EB	2
	3.3.1		Table Components		JK, EB, LS	6
		3.3.1.1	Static Components		JK, EB	20
		3.3.1.2	Dynamic Components	3.3.1.1	JK	8
		3.3.1.3	Editable Fields	3.3.1.2	JK, LS	4
	3.3.2		Dropdown Selectors		JK	4
	3.3.3		Round Trip Data Communication	3.3.1, 3.3.2	JK, LS	2
		3.3.3.1	Populate Data in Table From Database		JK	4

		3.3.3.2	Store Data in Database From Editable Fields		JK, LS	6
3.4			Host Website On Server	3.3	JK	4
3.5			Define Endpoints	3.6	JK, LS, AS	6
	3.5.1		Determine Endpoints Needed		JK, LS, AS	6
	3.5.2		API Documentation		JK, LS, AS	6
	3.5.3		Endpoint Documentation		JK, LS, AS	6
3.6			Fake JSON	2.1	JK	3
3.7			Test Components	3.3.1, 3.3.2, 3.3.3	SS, JK, LS	12
Total Frontend						109
Grand Total						703
Weekly Hours per Individual						7.3

Table 3.1: SDMay20-25's Work Breakdown Structure

3.4 Possible Risks And Risk Management

The following risk matrix (Table 3.2) was created to assess the risk of the project. The risk matrix takes into consideration the likelihood that the risk will occur and the impact that it would have on the project. A higher likelihood or a greater impact increases the severity of a risk. Likelihood ranges from very likely to very unlikely. A risk that is very likely is defined as almost guaranteed to happen in the project life cycle, and a risk defined as very unlikely is almost impossible. Impact ranges from minor to severe; a risk defined as minor would cause a delay of a day at most, meanwhile a risk categorized as severe would cause massive project delays in the order of days or weeks.

	Impact			
Likelihood		Minor	Moderate	Severe
	Very Likely	Medium	High	High
	Likely	Low	High	High
	Possible	Low	Medium	High
	Unlikely	Low	Medium	Medium
	Very Unlikely	Low	Medium	Medium

Table 3.2: The Risk Matrix Used to Assess The Risk Severity

There are three risks that SDMay20-25 has categorized as high; inaccurate results, results not clearly understood, and our project having a steep learning curve.

The first of the highest risks is SDMay20-25s' proposed solution gives inaccurate results. Crafty has an implementation in place, and this risk happens if SDMay20-25's proposed solution doesn't perform better than the already-in-place solution. The long term goal of SDMay20-25's software solution is to minimize loss. In this case, there are a couple of options to get a more accurate model. The testing plan, covered in Section 5, states that the inputs to the algorithm are refined. Another option is to create a new algorithm entirely using a different framework. SDMay20-25 will initially start their prototype with a time series forecasting algorithm, however, SDMay20-25 has also considered working with linear regression and machine learning approaches.

The second high-level risk SDMay20-25 will face is the results of the report output are not clearly understood by the Crafty Procurement Team. It is important that the output report has a thought train that clearly outlines the reasoning for each order.

The final high-level risk is the steep learning curve associated with this project. This is a risk because it can lead to major delays in the project. Only one member of the team has experience working with machine learning approaches.

SDMay20-25's risk is mitigated by following these guidelines: integrating early and integrating often, individual research to back up technology considerations, and communicating with the client Crafty and SDMay20-25's mentor Dr. Goce Trajcevski. Risks that involve finances are not relevant to our project, as stated in Section 4.4.

3.5 Project Proposed Milestones and Evaluation Criteria

1. September 30th - Finalized Project Plans

   By the end of September, SDMay20-25 will have completed the following tasks: define team collaborations policies, obtain client background, and have met with the client to understand the problem at hand. SDMay20-25 will have a greater understanding of Crafty's problem and how to approach it with the completion of this milestone.

2. October 31st - Finalized Project Architecture and Initial Tech Stack

   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 one of their product warehouses to SDMay20-25. An instance of a server and backend will follow. This will establish a round trip connection. Adding a frontend component will be useful to visualize data. This milestone also covers the system design diagram.

3. November 30th - Methodology Selection and Tools

   SDMay20-25 will have selected technologies and justified why they will be the best for the task. SDMay20-25 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 a testing framework will be provided as a deliverable. This framework will allow for testing as development tasks are completed. 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 Solution

   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 the 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 scenario 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. If the

7. April 30th - Integration Testing, Final Version of Software Solution, 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. This milestone also includes a complete working software solution (See Section 5 on testing for more details).

3.6 Project Tracking Procedures

SDMay20-25 uses GitLab issues as the primary method to track project progress. Each task found in Table 3.1 is posted in GitLab as an issue. Each issue includes a due date, assignees, and a corresponding milestone as defined in Section 3.5. Tasks are generated by SDMay20-25 using a work-breakdown structure (Table 3.1). Tasks were broken down into three categories: project requirements or project planning, frontend, or backend. These tasks are to be completed on a weekly basis; completed tasks usually produce software or written deliverables. It is important that each issue in GitLab is well written and thoroughly documented. This is accomplished by adding a deadline to each issue, assigning it to the appropriate milestone, and a detailed description of the issue.

Issues on GitLab have many features that will be useful to use including: labels, comments, due dates, markdown support, a board view (see Figure 3.3), milestones, and many other managing tools. Gitlab will be used to maintain an internal wiki. This wiki includes SDMay20-25's meeting notes, guides for installing various technologies, and individual research.

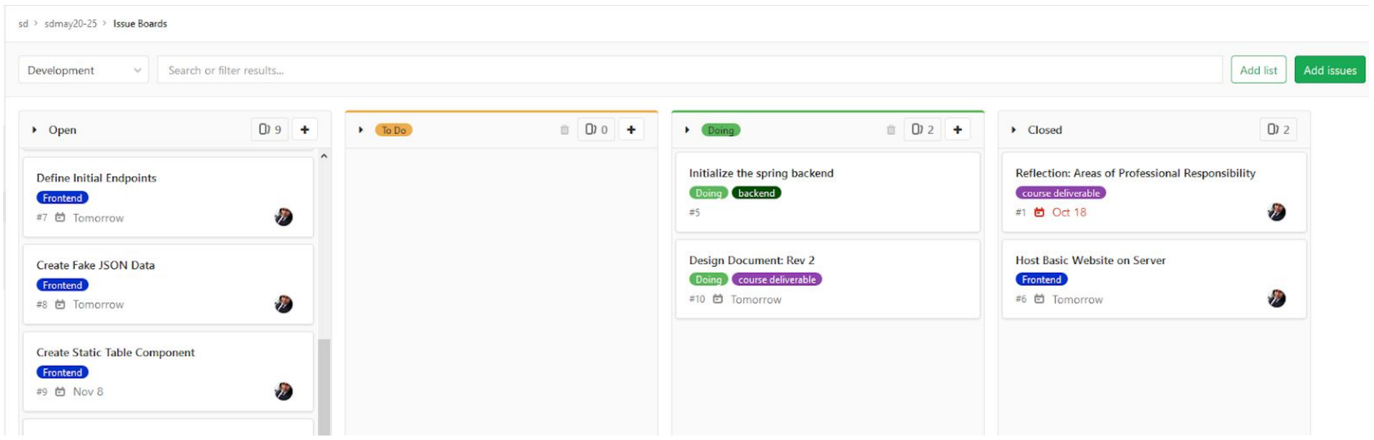


Figure 3.3: The Gitlab Board View Used to Track Issues

Informal communication is handled using Slack, a work-oriented instant messaging program. SDMay20-25 has created a relevant chat room, or channels, that includes all group members. Any relevant information will be copied to the appropriate GitLab issue.

3.7 Expected Results and Validation

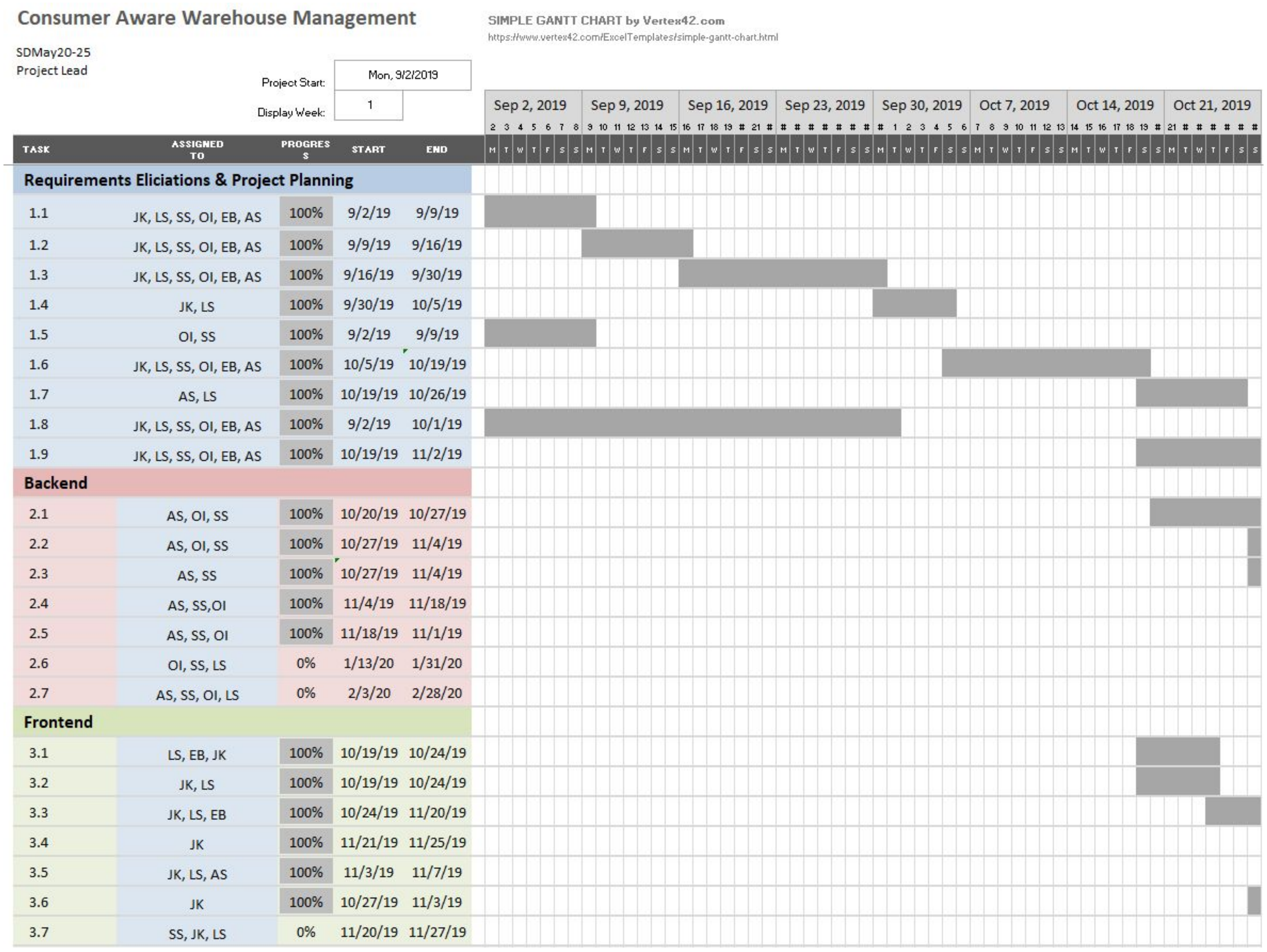
Crafty should expect a proof-of-concept that works, has been tested, and has a promising accuracy with historical levels of warehouse stock. The Consumer-Aware Warehouse Management proof-of-concept will accurately predict the demand for the goods for the current ordering window while also trying to maximize profit. Accuracy of the software solution is determined by testing it against our test dataset, this is further defined in Section 5.3. This proof-of-concept will contain: a database, a backend, frontend component, and will be hosted on a server; see Section 1.6 for the expected end product.

4. Project Timeline, Estimated Resources, and Challenges

This section highlights SDMay20-25's project timeline and the feasibility of completing the project successfully. A project timeline is provided and includes weekly tasks based on Figure 3.1 which covers both semesters. It is important to determine whether SDMay20-25's software solution can be completed successfully and the feasibility report will cover the likelihood of that occurring given all relevant factors to the project. Finally, any outstanding resources not covered earlier will be described here as well.

4.1 Project Timeline

The project timeline will be split into two semesters. In the Fall semester, efforts will be focused on project design development and core architecture implementation, which includes setting up a frontend and server and ensuring the connection between them and the Crafty database. 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.



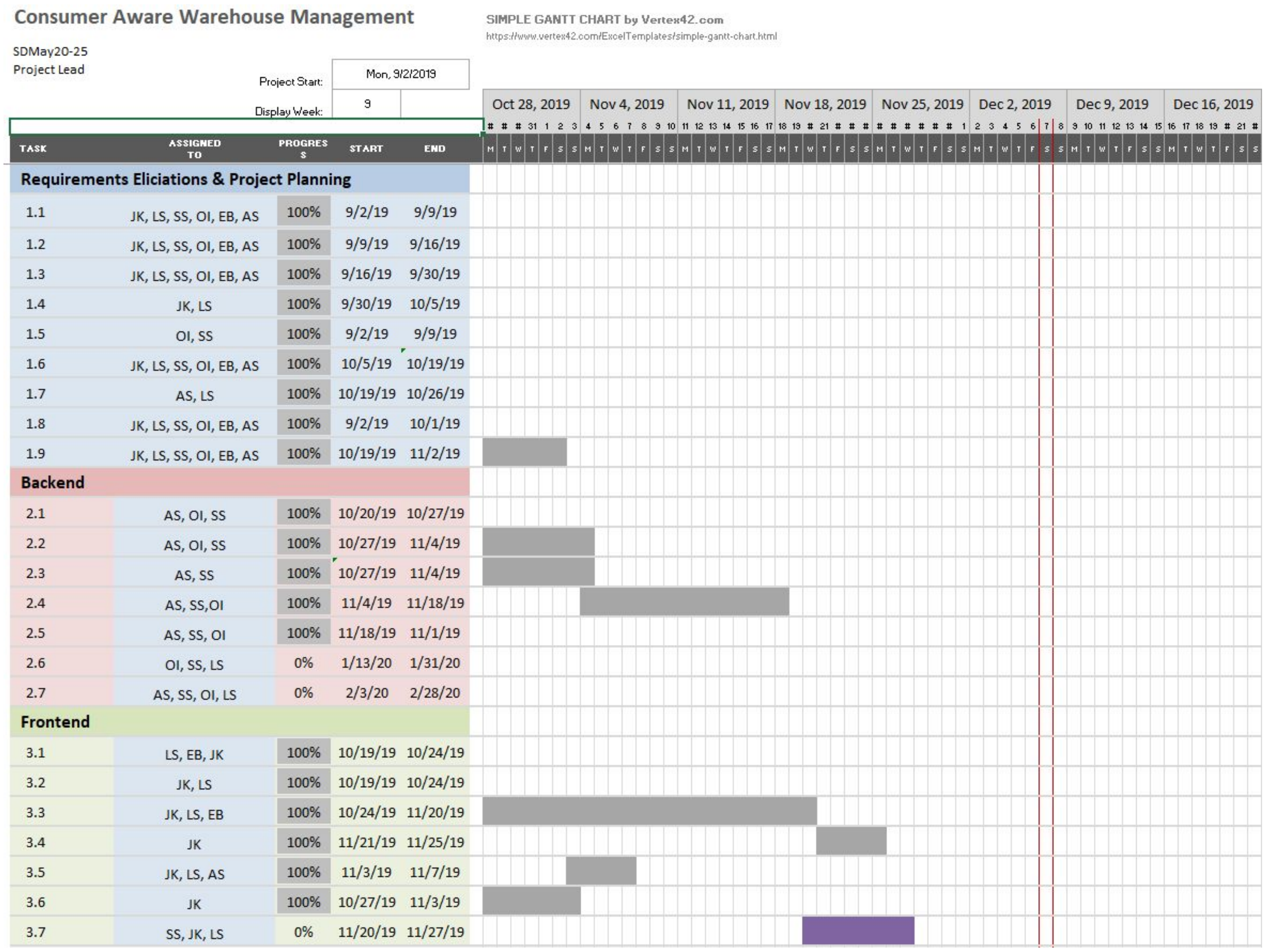
Consumer Aware Warehouse Management

SDMay20-25
Project Lead

Project Start: Mon, 9/2/2019
Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Requirements Eliciations & Project Planning				
1.1	JK, LS, SS, OI, EB, AS	100%	9/2/19	9/9/19
1.2	JK, LS, SS, OI, EB, AS	100%	9/9/19	9/16/19
1.3	JK, LS, SS, OI, EB, AS	100%	9/16/19	9/30/19
1.4	JK, LS	100%	9/30/19	10/5/19
1.5	OI, SS	100%	9/2/19	9/9/19
1.6	JK, LS, SS, OI, EB, AS	100%	10/5/19	10/19/19
1.7	AS, LS	100%	10/19/19	10/26/19
1.8	JK, LS, SS, OI, EB, AS	100%	9/2/19	10/1/19
1.9	JK, LS, SS, OI, EB, AS	100%	10/19/19	11/2/19
Backend				
2.1	AS, OI, SS	100%	10/20/19	10/27/19
2.2	AS, OI, SS	100%	10/27/19	11/4/19
2.3	AS, SS	100%	10/27/19	11/4/19
2.4	AS, SS,OI	100%	11/4/19	11/18/19
2.5	AS, SS, OI	100%	11/18/19	11/1/19
2.6	OI, SS, LS	0%	1/13/20	1/31/20
2.7	AS, SS, OI, LS	0%	2/3/20	2/28/20
Frontend				
3.1	LS, EB, JK	100%	10/19/19	10/24/19
3.2	JK, LS	100%	10/19/19	10/24/19
3.3	JK, LS, EB	100%	10/24/19	11/20/19
3.4	JK	100%	11/21/19	11/25/19
3.5	JK, LS, AS	100%	11/3/19	11/7/19
3.6	JK	100%	10/27/19	11/3/19
3.7	SS, JK, LS	0%	11/20/19	11/27/19

Figure 4.1 Gantt Chart for Fall 2019 (Weeks 1-8)



Consumer Aware Warehouse Management

SDMay20-25
Project Lead

Project Start: Mon, 9/2/2019
Display Week: 9

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

TASK	ASSIGNED TO	PROGRESS	START	END
Requirements Eliciations & Project Planning				
1.1	JK, LS, SS, OI, EB, AS	100%	9/2/19	9/9/19
1.2	JK, LS, SS, OI, EB, AS	100%	9/9/19	9/16/19
1.3	JK, LS, SS, OI, EB, AS	100%	9/16/19	9/30/19
1.4	JK, LS	100%	9/30/19	10/5/19
1.5	OI, SS	100%	9/2/19	9/9/19
1.6	JK, LS, SS, OI, EB, AS	100%	10/5/19	10/19/19
1.7	AS, LS	100%	10/19/19	10/26/19
1.8	JK, LS, SS, OI, EB, AS	100%	9/2/19	10/1/19
1.9	JK, LS, SS, OI, EB, AS	100%	10/19/19	11/2/19
Backend				
2.1	AS, OI, SS	100%	10/20/19	10/27/19
2.2	AS, OI, SS	100%	10/27/19	11/4/19
2.3	AS, SS	100%	10/27/19	11/4/19
2.4	AS, SS,OI	100%	11/4/19	11/18/19
2.5	AS, SS, OI	100%	11/18/19	11/1/19
2.6	OI, SS, LS	0%	1/13/20	1/31/20
2.7	AS, SS, OI, LS	0%	2/3/20	2/28/20
Frontend				
3.1	LS, EB, JK	100%	10/19/19	10/24/19
3.2	JK, LS	100%	10/19/19	10/24/19
3.3	JK, LS, EB	100%	10/24/19	11/20/19
3.4	JK	100%	11/21/19	11/25/19
3.5	JK, LS, AS	100%	11/3/19	11/7/19
3.6	JK	100%	10/27/19	11/3/19
3.7	SS, JK, LS	0%	11/20/19	11/27/19

Figure 4.2 Gantt Chart for Fall 2019 (Weeks 9-16)

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 the framework upon which this algorithm stands needs to be solidified. A good framework includes solid requirements, a plan, and a core system architecture. This framework will be developed in the Fall semester. Tasks are presented in Figures 4.1 and 4.2. Requirements Elicitation and Project Planning spans the first 9 weeks. The rest of the semester involves the implementation of the core software architecture.

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

Table 4.3 Weekly Task Breakdown for Spring 2020

The Spring semester is all about the prediction algorithm. Our tasks are outlined in Table 4.3. 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 the information received from discussions with Crafty and is necessary to drive the development of the prediction algorithms.

Once the test scenario is defined, development will begin on the prediction algorithms. The initial implementation will use a time series forecasting. The algorithm will then be iteratively implemented, tested, analyzed, and optimized to tailor the solution to Crafty's needs outlined in Section 1.3. For more information, the development process is outlined in Section 2.3.

4.2 Feasibility Assessment

SDMay20-25's project will primarily be an experimentation with prediction methods for Crafty's product warehouse stock ordering. SDMay20-25 will implement the prediction algorithms as the proof of concept of their usefulness for product ordering in Crafty's system. There are 2 main challenges with this project. (1) SDMay20-25 has limited experience in relevant areas including machine learning and forecasting, and (2) developing a robust test scenario will also be a challenge in this project. To account for limited knowledge on machine learning and forecasting algorithms, a portion of the work has been set aside for research into these areas. More information on testing can be found in Section 5.

4.3 Personnel Effort Requirements

All task time estimates are found under the Effort Hours column in Figure 3.1. To generate the effort estimates, the team met and made estimates as work was completed in the project. Estimates for the tasks going forward were made based on the information about the time it took to complete past tasks.

4.4 Financial Requirements

Software development and management tools will be used during the development of this project. Free tiers of all software will be used when possible, and at this point in time, they will suffice. If the need for paid software arises, SDMay20-25 will negotiate financial resources with Crafty. There are no other financial requirements.

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, therefore the only interfacing to test is the communication between the backend, database, and the frontend. However, there are many software interfaces in the project. As shown in the Software Architecture Diagram, Figure 2.1, there are many different interactions between the different points. There is a database that will interface with the Spring backend API. From there, the Data Forecasting Controller is where the data forecast will run. The data forecasting controller 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. The solution 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 the development machines.

SDMay20-25 has selected software tools to aid the testing outlined in Table 5.1.

Test Type	Software Used
Continuous Integration and Deployment	Gitlab CI/CD
API Call Testing	Postman

Table 5.1: Software Tools for Testing

5.3 Functional Testing

Integration testing will be used to implement the interaction between the components to ensure interactions between classes are functional and data is being communicated properly.

To test the backend API, SDMay20-25 will test every endpoint to ensure that the data is formatted correctly. Postman will be used to manually test endpoints. The API will be checked to ensure the data is being sent in the correct format. There will also be some selected tests to make sure the right data is being sent for the endpoints.

The React-Testing-Library package will be used to test the frontend. The frontend developers will be responsible for generating the test cases for the functions they create, as well as generating tests to detect defects of rendered components. These test suites will be automatically run in the pipeline along with the backend test suite. If the tests fail then the product will not be deployed.

Acceptance testing will be a large component of the overall testing in this solution. The solution was reviewed with the client on a bi-weekly basis to show our progress. This will identify changes the client desires early so they can be corrected with less effort. SDMay20-25 will also keep an open line of communication with the client between meetings to ensure the requests of the client are well understood.

Next, testing will be done by simulating the current day as a day in the past. The dataset will be split into two subsets: training and evaluation. The most recent 3 months of data will be reserved as evaluation data and the rest will be allocated for training. By telling the system that the current date is at the end of the training algorithm, the evaluation data can be used to evaluate the performance of the prediction. The consumption in the evaluation data will give an indication of the order predicted by the solution would have resulted in missed sales or expired products.

5.4 Non-Functional Testing

In addition to the functional tests provided above, SDMay20-25 has outlined testing for performance, security, usability and compatibility requirements.

Performance testing is a main priority for the project and the software should run with the performance that the client sees as acceptable. The client has defined an initial benchmark as keeping the loading time for a purchase order will be kept under 2 minutes for 90% of orders generated and under 5 minutes for the remaining 10%. The client has noted that this metric can be flexible and can be revisited throughout the development cycle.

SDMay20-25 and crafty have identified security as a concern, but have determined that it is not a large concern in this solution. The solution is meant to be a proof of concept for the client. The solution will not be directly used in production. If the solution is adopted by the client, they will be adapting it to work within their existing system which already takes security considerations into account.

Usability testing is a high priority in the solution. The client will determine how usable the solution is for their procurement team. The client has indicated that their current implementation has room for improvement mainly because the team does not trust the predictions. Therefore, usability will be tested by ensuring that the algorithm outputs the correct reasoning behind a prediction. The solution should be approachable and understandable to any member of the Crafty team. This means that the Crafty team should not need knowledge of the implementation to understand the reasoning for the predictions that are given. In addition to approachability, the client will compare the new solution to the existing solution to determine if it is more approachable for the team to understand the decisions that the solution is making.

Compatibility testing comes into consideration in the database for the project because the solution focuses on retrieving data from the database and will write the results to the database. It is important to ensure the database version we use is compatible with the version of the database in use at Crafty. The version used on the Crafty Server is reported to be compatible with the version we use on our server, and a conversation with the client reported that the version in use on SDMay20-25's server is in use on Crafty development machines and is used without issue.

5.5 Process

The procedures for testing are outlined in Sections 5.3 and 5.4 above. Acceptance testing by the client is the most important testing that we have in this project.

5.6 Results

In the end, our result focused on data from historical warehouse stock levels. In the future, if the algorithm needed to be refined, more of the input variables enumerated in Section 1.2 could be added in to focus more on modelling consumption trends instead of inventory trends.

At this point in time, the development of the testing framework has begun. For the initial prototype of the solution, manual acceptance tests have been employed. For the experiments, less stringent testing is used. Once the development moves on to steps toward the final implementation, the full test process will be employed.

The algorithm was around 80-85% accurate for the SKUs we tested with (it was around 80-85% accurate at predicting the stock levels of a SKU at any given day). Of course, this is not necessarily an entirely valid metric, as it was not tested on all SKUs. This is because not every SKU **can** be tested, or even trained, with the algorithm. Thus, our scope was limited to only products that had enough data to be trained on, and a random, small sample of that. This limitation would apply to any algorithm though, and does not represent a limitation in our algorithms specific design.

6. Closing Material

The problem Crafty is facing is defined by the following points:

- Waste of expired product
- Missed sales due to insufficient inventory
- Erroneous and time-consuming labor efforts

The system will fulfill the following requirements. These were further explained in 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

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 reduce waste by ordering just enough until the next order can be placed. 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.

The 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 for 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.

Crafty already had an algorithm in place to aid with orders, but it was not trusted by the procurement staff. Therefore, the proposed solution will have to be better than the existing solution and provide the users with confidence in the decisions it makes. The proposed solution takes the needs into account provided in the requirements.

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)

6.3 Appendices

Appendix 6.3.1 Database Schema Diagram - https://dbdiagram.io/d/5d9d4bb3ff5115114db50d55

6.4 Appendix 1: Operation Manual

Gathering Required Code and Data

- Download code from http://sdmay20-25.sd.ece.iastate.edu/files/sdmay20-25-master.zip
- Download data from http://sdmay20-25.sd.ece.iastate.edu/files/Database.zip

Creating a local Postgres DB

Prerequisites

- Postgres is installed (v11 was used)
- Postgres server is running (I think it runs on startup with Windows, but not extensively tested)

Steps

1. Connection Details
   - Server: localhost
   - Database: postgres
   - Port: 5432
   - Username: postgres (default user)
   - Password: [This is the password you set up when installing Postgres Server. I don't know if you can reset it.]
2. Create a new database for the Crafty dump with the command: CREATE DATABASE crafty;. In this example, my database would be called crafty. I will use this example name throughout the tutorial.
3. Switch to the new database: \c crafty or maybe more intuitively \connect crafty.
4. Now, run the import command for craftyisu.psql: \i 'C:/Users/<samst>/Downloads/craftyisu.psql'
   - **IMPORTANT**: Even on windows, you must use / instead of \ in your file path.
   - The above uses my file path. Find your psql file in the Windows explorer and copy the path to it. **By default** Windows will use \ in the file path if you copy from file explorer, so double check.
   - This will take a few minutes (about 5 maybe?). So let it run and take a walk or something.

5. Run the import again for the following in the Database/additional-tables folder: 1-create-insert-inbounds-modified.psql, 2-create-insert-inbound-items-modified.psql. Then run the following SQL:
   ALTER TABLE public.shopping_cart_items ADD COLUMN inbound_item_id integer;
   ALTER TABLE public.shopping_cart_items ADD CONSTRAINT inbound_constraint FOREIGN KEY (inbound_item_id) REFERENCES inbound_items (id);
   Then, import the add_predictions_table.psql and the modify_predictions_table.psql
6. Now, the data should be imported. We can see all the tables with this command: SELECT table_schema,table_name FROM information_schema.tables WHERE table_schema = 'public'; (Side note: you can remove the where clause to see all tables, which includes those that are internal for Postgres)

Backend Server Setup

Prerequisites

Java 1.8

Python 3.x

Postgres Setup

Tested in Bash

Steps

1. Make application.properties
   a. Look at application.properties.template (backend/src/main/resources/). Make a copy and fill in the blanks with the appropriate information
2. In the backend folder of the project, make sure your gradlew is executable (for a bash shell, chmod u+x gradlew). Then run ./gradlew bootRun

Algorithm Setup

1. Install python3, pip3
2. Install Keras, tensorflow, and sklearn (instructions more specific than this cannot be provided very easily, it is highly system dependant)
3. Run non_api_version.py with python3, provide a SKU as a parameter
4. Repeat step 1 for however many SKUs you need predictions for, run them on random days and times throughout one month the first time, later on, run it once each day for every SKU

Fronted Setup

Prerequisites

- Node.js v13.11.0+
- NPM v6.13.7+
- Yarn v1.22.4+ (This is optional, if using npm first delete yarn.lock before running npm install)

Steps

Develop

1. Change directories to sdmay20-25/client-app
2. Run “npm install” or “yarn install”
3. Set Environment Variables or Connect to VPN
   a. Depending on the development environment the environment variable “REACT_APP_BASE_API_URL” will have to be set. By default it is “http://sdmay20-25.ece.iastate.edu:8080” which will require the user to be connected to the ISU VPN with the Cisco AnyConnect application.
      i. Note: changing this environment variable will require a development server restart with “npm start”.
      ii. Note: the sdmay20-25 server at ISU will likely be wiped soon after semester end (Spring 2020) and connecting to the VPN to access the server will no longer be an option.
   b. More info here on getting connected (https://it.engineering.iastate.edu/how-to/install-and-connect-to-vpn-pc/).
   c. More info on setting React environment variables can be found here (https://create-react-app.dev/docs/adding-custom-environment-variables/)
4. Run “npm start”
   a. To specify an API endpoint URL other than the default “http://sdmay20-25.ece.iastate.edu:8080”
      i. Run “REACT_APP_BASE_API_URL=<YOUR_URL_HERE> npm start”
      ii. Or create a “.env” file at the root of the React project (sdmay20-25/client-app/) and add “REACT_APP_BASE_API_URL=<YOUR_URL_HERE>”
5. Development server will begin running on localhost:3000 (or will ask to port crawl if that port is in use)

Build

1. Change directories to sdmay20-25/client-app

2. Run “npm install” or “yarn install”
3. Run “npm run build”
4. Output files will be located in sdmay20-25/client-app/build

6.5 Appendix 2: Initial Planned Design

The initial plan of the project was to incorporate different input variables such as warehouse space, expiration dates, and daily customer inventory level. As development went on it was clear there wasn’t any data to incorporate these types of input variables. In the future, when more data is generated, adding these input variables could prove beneficial to accurately predicting future needs and optimizing a distributor order based on cost.

6.6 Appendix 3: Lessons Learned

Backend

The biggest lesson learned was working with existing databases. Crafty’s database was difficult to learn at first. Our biggest hurdle was finding relevant data at any point. Spring was also not easy at first, since we had such a large database. This, in turn, leads to some complex queries.

The Backend team also learned that working between groups that required our data requires careful planning. An API design had to put into place to support a flexible frontend. Planning and task distribution was monumental as it led to timely endpoint deliveries to the frontend team.

Frontend

A number of challenges occurred on the frontend team working with React and a number of npm packages which we found ways to overcome. The first and most daunting was teaching a new framework to the other members. Not everyone on the team knew React and thus the process of development started and continued with a lot of pair programming. This is a large takeaway of the importance of investing time outside the project to help educate your team on what the technology is, why its being used, and most importantly: how to use it.

Additionally, the issue of endpoint integration came up as the project progressed and the teams grew more independent. We did not keep an API versioning system so as endpoints changed it broke things on the frontend since they were tightly coupled. In order to resolve this, the frontend team invested time into decoupling the API from the components by deserializing the data into more robust and usable Typescript classes. In this way, the data can be consumed uniformly by the components and the API changes only required changes in the deserialization of the data to classes.

Algorithm

The biggest thing we learned was how LSTM neural networks work with keras and tensorflow. We learned how to create the models, which types of models are good (the right numbers of nodes to use at each layer), how to scale the data to be used by the networks, and how to train the networks. We also learned a bit about how to make a valid architecture for ML related projects.

We also learned the importance of the data when it comes to predicting. We discovered that for many of the SKUs Crafty maintains warehouse stock for there was very little data. This made predictions using LSTM difficult, because there wasn’t adequate data to train the algorithm with. We only found this information out until very late in the project, because it took us a while to get necessary database queries. This taught us the importance of early database exploration, because knowledge of the data is critical in choosing the right algorithm for the task.