Consumer Aware Warehouse Management

Design Document

sdmay20-25

Jimmy Paul

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: October 6, 2019, Version 1

Executive Summary

Development Standards & Practices Used

List all software practices used in this project. List all the Engineering standards that apply to this project that were considered.

Software Practices

- UML Diagrams
- Design Thinking
- EARS (Easy Approach to Requirements Syntax)
- Gantt Chart
- Agile Methodology

Engineering Standards

- Security
- Ethics

Summary of Requirements

- List all requirements as bullet points in brief.

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 that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- Machine Learning Algorithms
- Front End Development
- DevOps
- Data Forecasting

Table of Contents

List of figures/tables

1. Introduction

1.1 Acknowledgement

Our team, Consumer-Aware Warehouse Management, would like to acknowledge Iowa State University's Department of Software Engineering for assisting this team by providing professional experience, necessary resources, and expert guidance. Special thanks are to be given to Dr. Goce Trajcevski for providing technical and managerial supervision on the project to guide the team to the course outcomes and to a successful end product. He is our subject matter expert on almost all technical topics of the project and has directed our efforts to the best use of our attention and research. The company Crafty, headquartered in Chicago, IL, will provide many of the requirements and constraints that will allow us to create the system being requested. Additionally, they will provide access to data used to forecast and test our system with. Finally, upon integration into Crafty systems, the hardware already implemented will be providing our software solution with the required data to make specific decisions. Additionally, we would like to acknowledge our contact at Crafty, CTO and co-founder, Jimmy Paul, for the time and feedback he has given to our team during the project development.

1.2 Problem and Project Statement

Problem Statement

The company Crafty desires a software solution to assist in managing stock of its warehouses for distribution of food stuffs to its area clients. Currently, the determination of when to order more goods and restock is done manually and can take hours that can be better spent. It is a tedious task that has the potential for human failure to take into account all factors in the future needs of a certain stock of goods. Thus, Crafty occasionally misses out on sales to potential and current clients, decreasing their profit margin.

Proposed Solution Approach

The company, Crafty, has proposed a potential solution in the form of a prediction and forecasting software that will take in all the available data points and determine what goods need restocking and when. It has been suggested that the implementation use a form of machine learning to determine the optimal conditions of ordering more stock. This software solution will eliminate the tedium of constantly checking when goods need to be restocked, thereby decreasing tedious workload on employees, allowing for prediction of increased sales, and eliminating situations where sales might have been missed due to human errors.

Project Need

The need for forecasting software to predict the optional restocking schedule for a warehouse boils down to reduction in waste and missed sales. Without this product, Crafty may allow goods to go unused and spoil in warehouses if the incorrect amount of goods are ordered and go unshipped. Alternatively, if too few goods are ordered, when a client calls the lack of goods represents a missed sales opportunity for crafty and thus a reduction in the overall profit margin. These basic needs represent the basics of business: customer satisfaction and maximal sales.

Project Description

To fulfill the solution for the pain point, Crafty has reached out to Iowa State's Senior Design course to partner with senior students to create a proof of concept software for prediction of automatic optimal goods ordering for its warehouses.

Project Artifacts and Outputs

This project will have a number of artifacts and outputs at its conclusion that will be distributed over both Crafty's needs and SE 491 needs.

- Meeting Notes
- Weekly Report Updates
- Architecture Plan Documentation
- Design Document (final and revisions)
- Senior Design Website
- Prediction Software Proof of Concept
- Frontend to display software capabilities

1.3 Operational Environment

The end product of this project will be entirely software based, thus, no consideration of hardware robustness will be given. The hardware that it will run on will be left entirely left up to Crafty to ensure that it is reliable and sufficient.

However, design considerations will be considered to reduce the performance cost of the system to reduce the expense of the required equipment. Additionally, considerations will be given to the current systems Crafty is using. Where possible and prudent we will attempt to mimic systems that Crafty has been using to make the transition from proof of concept to implementation smoother.

1.4 Requirements

List all requirements for your project – functional requirements within your project context, economic/market requirements, environmental requirements, UI requirements, and any others relevant to your project.

1. The system SHALL take anonymized data from Crafty as input
2. The system SHALL make predictions about optimal ordering of input goods
3. The system SHALL be able to take in a variable number of input goods
4. The system SHALL have a visual component for interfacing with the backend
5. The system database SHALL be implemented using PostGreSQL
6. The system server SHALL run on an EC2 instance
7. The system server code SHALL be implemented in Java Spring
8. The system frontend SHALL be implemented using React

1.5 Intended Users and Uses

The software system being created will be integrated into the existing Crafty systems if successful, thus there are no clear end users for our parallel system we will be creating. However, we still will be taking design considerations to ensure maintainability, reliability, security, performance, modularity, and scalability. One

concession that was made was to later in the project, create a user interface that will allow the team to demo the project to the Senior Design course. This system will only have the end user of alpha testers.

1.6 Assumptions and Limitations

Assumptions

- The system will have constant access to the Crafty database
- The system will have access to the internet
- The system will be used and maintained by English speakers
- The system will only be used inside of the US
- The system will only be used to manage the inventory of a single warehouse

Limitations

- Accuracy of system predictions is completely reliant on accuracy of data system is given as inputs
- Implementation will match the Crafty system architecture wherever reasonably possible

1.7 Expected End Product and Deliverables

The end product will consist of a prototype of the software to be implemented on Crafty systems. It will be a full stack application (frontend, backend, and database).

1. Finalized project plans and architecture (September 30th)
   This will include the first iteration of the intended tech stack that we will be using. The most immediate technology will be our database, since we will be receiving data from our client for one of the warehouses. We plan to add our server and backend shortly afterwards. This will establish our round trip connection. Adding a frontend component will not be required, but our team plans on implementing it as a means to demo and visual our data in a concise.
2. Constraints and the impact on the design decisions (October 31st)

   This deliverable report will list what our project constraints are both at software and system level. We will cover what 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. Methodology selection and tools (November 30th)

   In this deliverable we will be detailing how and why we selected our tools and methods to complete the project. This will cover decisions such as our development environment, server choice, development plan, and code verification methods. This will be distinct from our constraints as these will be self enforced constraints. By solidifying these tools and methods, we will be better able to communicate and work together as a team.
4. Testing framework (January 31)

   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. Alpha version of the software (February 29th)

   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. Unit testing and validation (March 31st)

   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 system of the system. Tests will likely include statement, decision, branch, condition, and finite state machine coverage (where each type of test is applicable). Test results will be documented and placed in a report. Additionally, the software will be validated with the client that we have fulfilled the requirements adequately for the alpha version.

7. Integration testing and report (April 30th)

   The final deliverable will be a full integration test suite that will verify that all the systems work together and handle errors correctly. Additionally we will be submitting a final report on the process and technical description of what we built at this time. The product will have undergone multiple rounds of tests and will be made available for Crafty to analyze and integrate into their own systems in the manor they best see fit.

2. Specifications and Analysis

2.1 Proposed Design

So far we have finalized the first iteration of our tech stack. Our first checkpoint towards our goal is to obtain anonymized data from the client. Once we have this data we will need to import into our own database. The database we have chosen is PostgreSQL. Our first software artifact will be setting up the database in an organized way and be able to query it. Since all of our group members have experience with SQL, we are hoping to pick up on Postgres quickly. Most of the work early on in the semester will be documentation and analysis; however, we would like to have our frontend, backend, and server up and running before our alpha version, which is due February. This way we can focus on our machine learning approach to the problem. In terms of documentation, we have completed our weekly reports, lightning talks, and we have construction various component diagrams.

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, we also need to develop a front-end application with the following functionality: a way to trigger the computation of the prediction algorithm, a way to display the results of the prediction algorithm (how much to order), and any other functionality we determine important. 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.

Insert Crafty's Database Design Here

2.2 Design Analysis

We have decided to use PostgreSQL for the database behind this project. This is mostly because that is the format that crafty is already using. It is also an open source framework. We considered MySQL since all the developers were familiar with it, but moved to PostgreSQL since it is similar to SQL and expands upon the functionality and it will make handing off the final product to the customer much easier at the end.

For the backend server, we have decided to use Java Spring. All developers are familiar with the software suite. Java also has widespread support in the university, industry as well as online resources. Spring specifically is very well documented and has many online resources both in the forums and directly from the developers. We have also considered Python for some machine learning related tasks since it is known for that, but we will evaluate that further depending on how the early mathematical models work and customer needs.

In terms of the predictions, we will be using mathematical regression or machine learning. We will attempt the regression first as it is a simpler method for prediction and if we can get it to work, it could be more effective. If we don't need to go to a more complex algorithm such as deep learning, it would be better. The stakeholder also told us there may not be enough data to get an effective model for deep learning.

2.3 Development Process

For the development process we are going with the agile model. We are going with this method because we are going to be doing biweekly sprints where after the two weeks we are having a sprint meeting with our client. We are going to be assigning tasks to be completed for a different portions of the project due between each sprint meeting. We aren't going with Waterfall because we will have client interaction from beginning to end. We are also not going with iterative because we aren't building each piece one at a time and we are going to have customer interaction throughout the process.

2.4 Design Plan

There are many principles of design that we will be following for this product to fulfill the use cases of the requirements.

- Compatibility

  The system we will be creating will need to be compatible with the systems already running at Crafty. Both the hardware interfaces that will be feeding signals to our system, and the interfaces to parallel systems that ours will be interacting with will need to be easily integratable.

- Extensibility

  The software we will be creating will need to be open to extension as there may be more data that Crafty has access to in the future that may lead to better predictions if run through an updated model. Thus we will have to avoid hard coding any values to how our algorithm learns when to order more of a specific stock.

- Modularity

  The system being created will need to be able to slot into the current Craft solution. Thus any code written must be module as to make interrogation possible.

- Fault-Tolerance

  As with any business, the product that they are providing should not be easily susceptible to breaking. Thus the implementation will need to be fault-tolerant of all data coming into the system and always give a logical answer as a prediction.

- Maintainability

  While the proof of concept will not need to be maintained, the manner in which the architecture is constructed should be able to be maintained in the future for any updated that might need to be done on the system after the completion of the project.

- Reliability

  The system will need to be reliable enough to demo, however, this implementation will not be serving the client. Thus, we will not require a high degree of up time nor perfect performance in high stress testing. This allows us to use cheaper options to create this as a proof of concept.

- Reusability

  New warehouses should be able to reuse the system we develop. This means that the inputs should be configurable depending on what is required on site.

- Robustness

Again, the robustness of this system is not mission critical as this is a proof of concept. However, the more variance we can account for in our proof of concept, the easier re-implementation will be for our client.

- Security

  The security of the system will be dictated by the requirements our client gives us. The risk of a data breach is determined by the sensitivity of the data we will be working with. It is unlikely that we will be working with sensitive data for this project on our end and thus we will likely acknowledge security vulnerabilities but not make them our first priority.

- Usability

  The system will be implemented as an automatic system solution. The client has expressed that any user interface we make will not be implemented in the final version. This means that we will be catering the usability of the system solely for show and tell purposes.

- Performance

  Performance will be a large concern if we implement a real time system. Additionally, the upkeep costs of the system will be dictated by the requirements of performance we create through the algorithm we implement.

- Portability

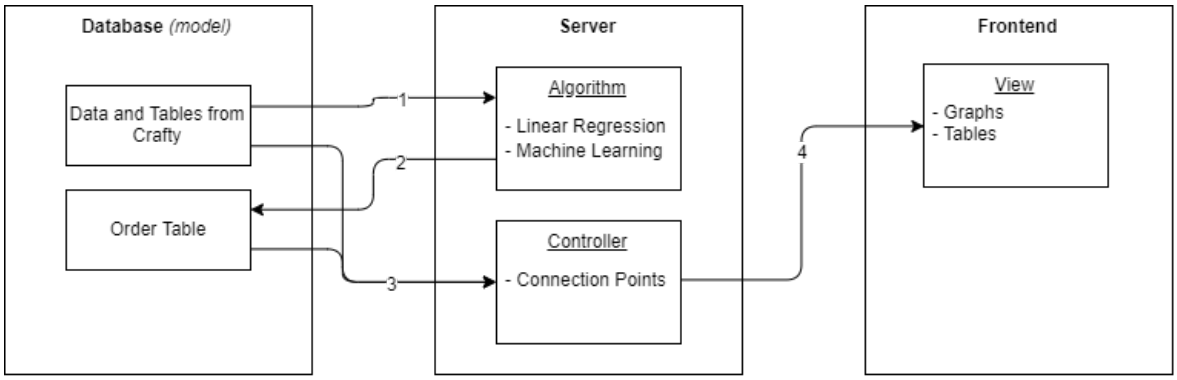
  This system will focus less on portability as it is being implemented for a specific client on a specific system. However, it will need to be able to be written in a way the analysis and re-implementation will be feasible.

- Scalability

  The system will need to scale to handle the needs of a single warehouse for this proof of concept. However we will need to keep in mind that in the future the system may have greater demands on it and it cannot scale in requirements on equipment nor time at an exponential rate with each new data source added to a reasonable amount.

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.



Database
- PostGreSQL

Server
- Java
- Spring
- JPA/Hibernate

Frontend
- JavaScript
- React

Figure 2.1

3. Statement of Work

3.1 Previous Work And Literature

There are several products that are in use in the industry. From full on machine learning, that considers every possible aspect of supply chain, to single regression from an Excel sheet and spectrum between the two ends. Our product will lie within this spectrum. I will describe the machine learning product but only the aspects that apply to our projects' scope. It 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. It takes into consideration the limitation and conditions of how to get the product to the 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 advantages to this system are that it will provide products with a better accuracy to the demand. It also does this extremely efficient 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. On the other end of the spectrum exists a single regression. It is able to predict the demand based on past sales and the seasons. Using simple math it is able to forecast the demand for a given day or season [2]. The advantages to this system is the little amount of resources needed to build and maintain. The disadvantages are that it doesn't account for spikes in demand due to factors that aren't based on past sales such as weather and it doesn't take into consideration the shipping time of the product. It might say that the demand for a product tomorrow is high so places an order but it takes a week to get stock, missing out on sales during that week.

3.2 Technology Considerations.

The technology our team 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. Disadvantages are that it is slower than MySQL. The trade-off of this software is that is slower than MySQL but more reliable and easier to integrate. 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 Java with Spring 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. Weaknesses are that there is a learning curve and can be difficult to setup. The trade-off for using these is speed as there are more efficient and faster software. For the front-end our team will be using JavaScript with a React framework. The strengths of using JavaScript and React are the community and the experience that our team has using both.

3.3 Task Decomposition

The tasks are separated into three categories, database, server, front-end.

Database:
-setup
--import data
--setup user/privileges
--host on server

-design
--create tables/views needed
--create queries needed

Server:
-base
--connections to the database and front-end
--spring framework setup
--host on AWS or other hosting service
-algorithms
--linear regression
--ordering
--retrieving and requesting information from/to the database

Front-end:
-base
--connection to the server
-design
--graphs and tables of consumption
--table of what needs to be ordered
--anything else needed for visual consumption

3.4 Possible Risks And Risk Management

There are a few risks that our team has looked into. The biggest risk would be not getting our software up and running. We are avoiding this risk by integrating early and integrating often. We plan to have our database setup within the first few weeks of obtaining our data. Once that is completed we would like to set up our backend and our server. Once these components are up and running, then it is a matter of connecting them. We have taken careful consideration in the technology we want to use; we have talked to our client and our mentor closely about what technology we want to use.

Another risk deals with our own knowledge of the solution we plan to implement. Very few members of our team have worked with machine learning approaches. This risk can be mitigated by having various resources that our team can use. These resources include: our mentor, each other, and any online resources.

3.5 Project Proposed Milestones and Evaluation Criteria

The following are our proposed Milestones. The task decomposition for each milestone are further discussed in section 3.3 Task Decomposition.

- Milestone 1: Have our data in our database in their proper view.
- Milestone 2: Host the server
- Milestone 3: Have our backend set up with some simple endpoints
- Milestone 4: Create a Frontend that will display useful information to the client

There are four major milestones for our project. Each milestone will be completed every one to two months. The first milestone is due October 31st and the next three milestones are due on the following dates:

Milestone 2 - December 13th

Milestone 3 - February 28th

Milestone 4 - April 30th

3.6 Project Tracking Procedures

GitLab issues will be our primary way to track project progress throughout the semester. Each task will be posted onto GitLab as an issue. A task will be determined and created by the team and will we written in detail. 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, milestones, and many other managing tools. To manage our communication we are using both Slack and GitLab issues. The former will be used informally and the latter lets us communicate task specific details at a much more thorough level.

3.7 Expected Results and Validation

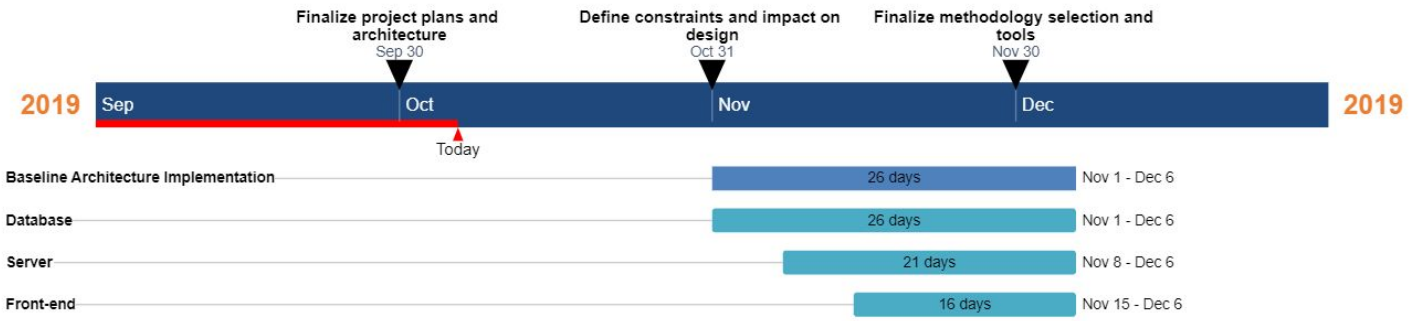
At the end of the project, Crafty should expect a product that works, has been tested, and will be easy to integrate with their current databases and warehouses. The Consumer-Aware Warehouse Management Product will be able to accurately predict the goods that a warehouse needs to order based on previous data. This product will contain: a database, a backend, frontend, and will be hosted on a server. We will know our solution is correct by the time it is ready to be delivered because of our testing plan.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

Our 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 we will focus on the development and testing of the core functionality (prediction algorithm), and its integration within our architecture.

Fall Timeline

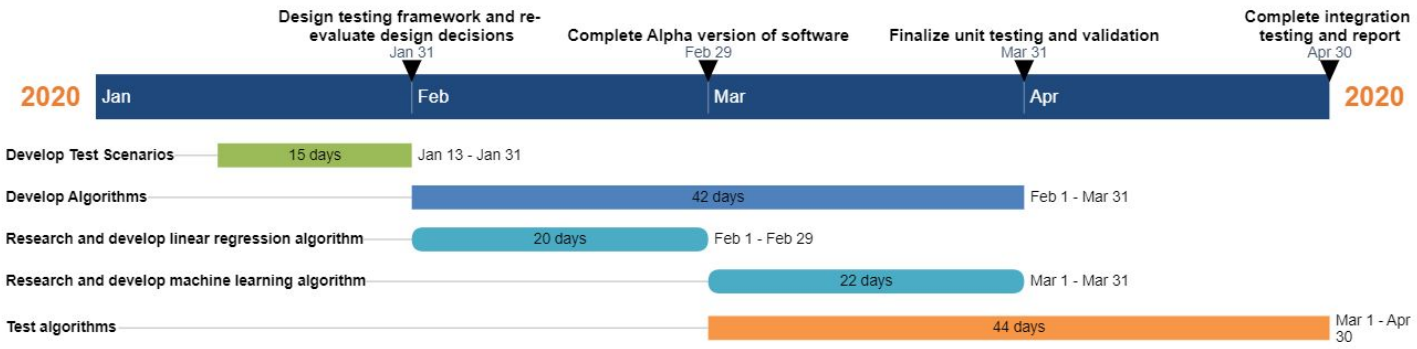


The purpose of the Fall semester phase of this project is to lay the groundwork in preparation for the Spring semester, so that we can develop something great with as little disruptions as possible. The meat of this project is the prediction algorithm. The prediction algorithm is the most complex part of this project, and it will require the most resources in terms of research, development, and testing. In order to let the Spring semester be a success, we need to solidify the framework upon which this algorithm stands. This framework has a design component and a software component.

The design component of the framework includes all sorts of constraints and other 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. We will collect these constraints and details through October 31st.

The software component of the project framework is a baseline architecture. This architecture includes the implementation of database, server, and front end component, and their integration with one another. These components will by no means fleshed out by the end of this semester, so their development will continue in the Spring. So that we can focus on the algorithm in the Spring without worrying about architectural issues that may alter our design, it is important that we prove the architecture early. The Database is the most important component as it what the algorithm relies on to function, so it will get the most time devoted to its implementation. The baseline architecture will be developed starting in November, and ending in early December.

Spring Timeline



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 Craftys needs.

4.2 Feasibility Assessment

Our project will primarily be an experimentation with prediction methods for Crafty's 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.

4.3 Personnel Effort Requirements

Task	Description	Effort (Time in Hours)
Baseline system architecture implementation	Implement a functional software architecture composed of a frontend application, a database and a server component. Ensure	120

	communication between all components.	
Develop Test Scenarios	Develop a robust test scenario in which we define metrics for a successful algorithm.	80
Develop Prediction Algorithm(s)	Research and implement a prediction algorithm using linear regression, and time permitting, research and implement another algorithm using machine learning.	150
Test Algorithm(s)	Apply the test scenario we developed to the algorithm(s) we developed.	150

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

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case 5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested 8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

5.1 Interface Specifications

– Discuss any hardware/software interfacing that you are working on for testing your project

We do 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.

5.2 Hardware and software

– Indicate any hardware and/or software used in the testing phase

– Provide brief, simple introductions for each to explain the usefulness of each

Hardware

In terms of the hardware required for testing, since this is a software implementation, we will only require the computer hardware needed to run the test software. 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.

Software

In terms of software, we will be using some automated testing where we can. For the server component, we can use JUnit testing for basic unit testing of the web server. We can also automate some testing of the API. Postman will let us automate API calls and check if the results are as we expect them to be. For the frontend website, manual testing will be employed.

Also, we will be developing some test cases using the provided data to generate test cases. We will want test cases for when a product should be ordered for the warehouse when the stock is low, when the stock is projected to be low before the next order can be placed. We will also want to make test cases where the

product should not be ordered and make sure the algorithm does not falsely suggest that product should be ordered.

5.3 Functional Testing

Examples include unit, integration, system, acceptance testing

On the backend we will be using unit tests on all classes to ensure the class works properly. We will also test the interaction between the classes and ensure everything on the backend is working properly. We will aim for 85% branch and line coverage with our test cases.

To test the backend API, we will also be testing every endpoint to ensure that the data is formatted correctly at the very least. Using postman, we can automate the testing of these api tests. We can also set up some test calls using the datasets with the predetermined outcomes and make sure the algorithm returns the expected value.

5.4 Non-Functional Testing

Testing for performance, security, usability, compatibility

Using the other test cases, we can look at runtime and make sure the programs and test cases are run in a reasonable amount of time. For usability, we will make sure the program is easy to use by the client by showing the production version to the client and seeing how easy it is for them to use. We will also use all parts of the software on multiple environments (personal development machines with Windows, MacOS and Linux, and the server).

5.5 Process

– Explain how each method indicated in Section 2 was tested

– Flow diagram of the process if applicable (should be for most projects)

5.6 Results

– List and explain any and all results obtained so far during the testing phase

- – Include failures and successes
- – Explain what you learned and how you are planning to change it as you progress with your project
- – If you are including figures, please include captions and cite it in the text
- This part will likely need to be refined in your 492 semester where the majority of the implementation and testing work will take place

-**Modeling and Simulation**: This could be logic analyzation, waveform outputs, block testing. 3D model renders, modeling graphs.

-List the **implementation Issues and Challenges**.

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

6. Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

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)