Project Deliverable E

Project Schedule and Cost

Marie Levin, Leila Smaili, Rafiullah Hamdard, Hanna Paik, and Yifei Li

February 19, 2023

Abstract

This document contains Project Deliverable E. The purpose of this document is to provide an estimation of the costs and the materials that will be required for our project. The costs of the materials were analysed by the team and will be approved by TA. This cost estimation was examined based on types of materials and their quantities that are required for the project.

Table of Contents

1.0 Introduction

Robert Ritchie has tasked our design team to create a product that displays the specific conveyor speeds that optimise the yield of beer from his manufacturing line. Following the initial client meeting, our team created the following problem statement.

“A need exists for Robert Ritchie and his fellow supervisors to find the optimal speeds of their beer packaging process to ensure it is “always at top efficiency (Project Background, Brightspace)”. The solution must have an attractive and straightforward interface that uses a flexible algorithm based on the V-Curve Theory to report optimised speeds of each unit and make recommendations on how to achieve them.”

From this problem we developed a design criteria and metrics that will be used to measure our design’s ability to solve our client’s problem. With research, benchmarking, and brainstorming, the first conceptual design was developed and it was presented to the second client meeting.

After we presented our presentation to the client in our second meeting, he displayed interest in our conceptual design. The client promised that he will provide us with the following raw data:

1: Speed of the filler station (cans per minute).

2: The speed of the conveyors (cans per minute.)

3: The preset speeds of different units

We presented the client with two possible designs:

1: Windows software 2: Mobile application

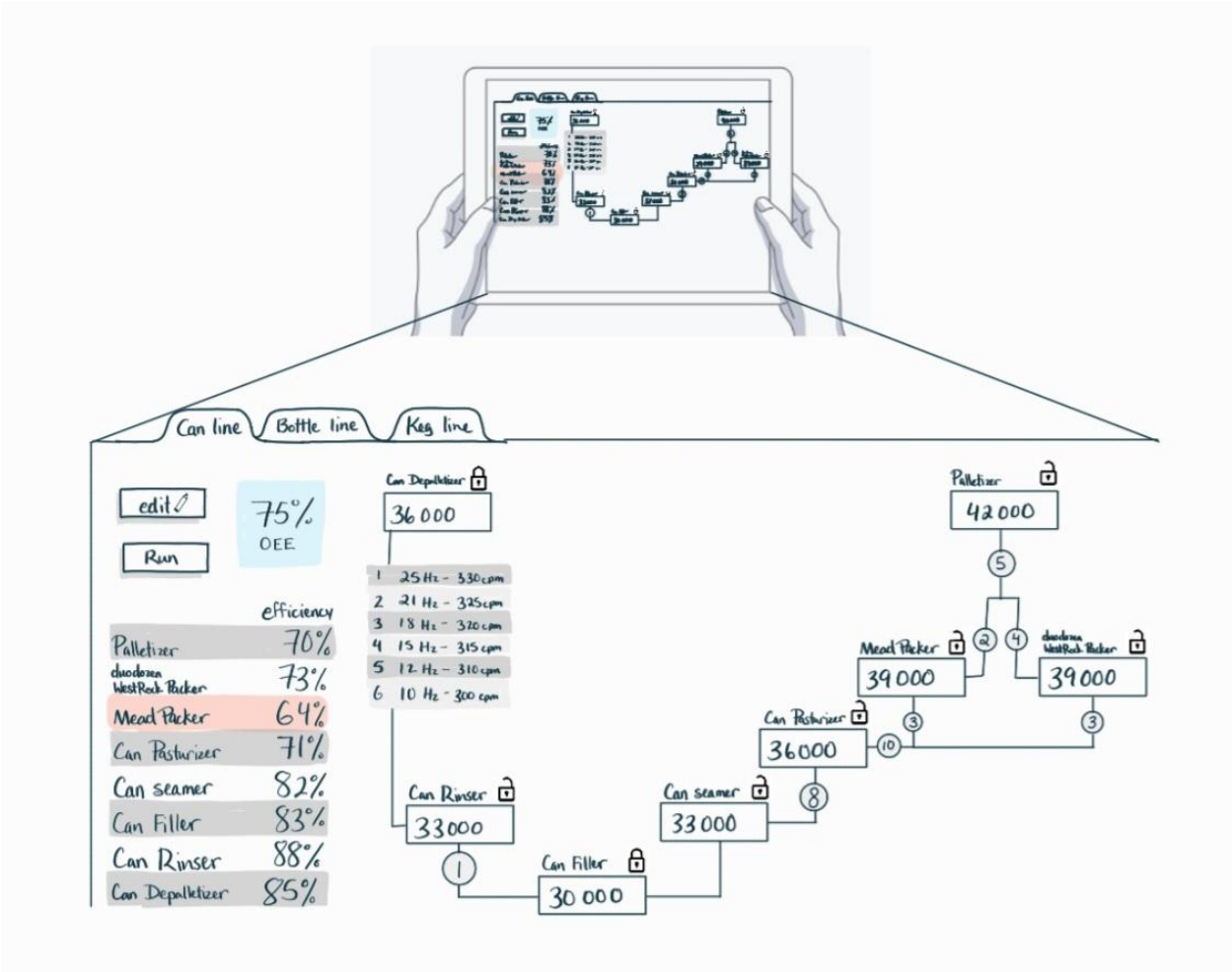
The client was more interested in mobile applications than windows software. Because he and his team currently use iPads in their office, it was clear that an iOS app was of much more interest to him as opposed to an android device. Moreover, the client insisted that he wants to be able to carry the device around the production line. Then, he can change the can filler speeds and type the desired input speed into the device, thus he should be able to get the best output and optimise his production line.

We will need to find a mechanism to convert hertz to cans per minute, because they use a PLC (an industrial computer control system), and it shows the drive in the hertz that is currently running. Once we do this, we will be able to optimise the conveyors based on the given raw data.

This document will go in depth into our final design concept. Furthermore, it will list all the required materials and equipment that we estimated. These materials will be used to construct and test prototypes, and to create a final design. This document will outline a comprehensive overview of prototypes and tests that will be incrementally done to ensure the completion of all aspects of the design. In addition, we will have our BOM (bill of materials), that will summarise the estimated budget of the project.

We must be careful with our limited budget and make a list of all the materials that we need, alongside their associated costs.

2.0 Final Conceptual Design



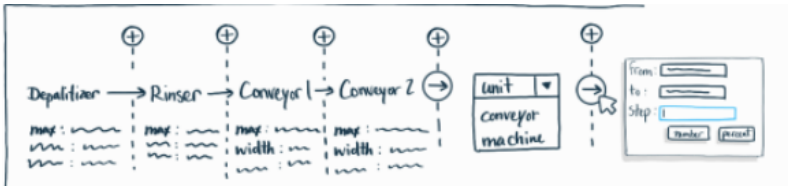
2.1 Elements of UI

→ **Line windows**

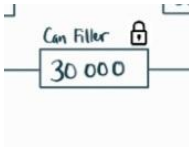
Our application will have the ability to have up to 5 separate line simulations. Upon selecting a tab, the simulation for the chosen line can be used.

→Edit line function

The production line can be edited with the click of the ‘edit line’ button. This button will lead to a separate window that displays the production line in a linear view:

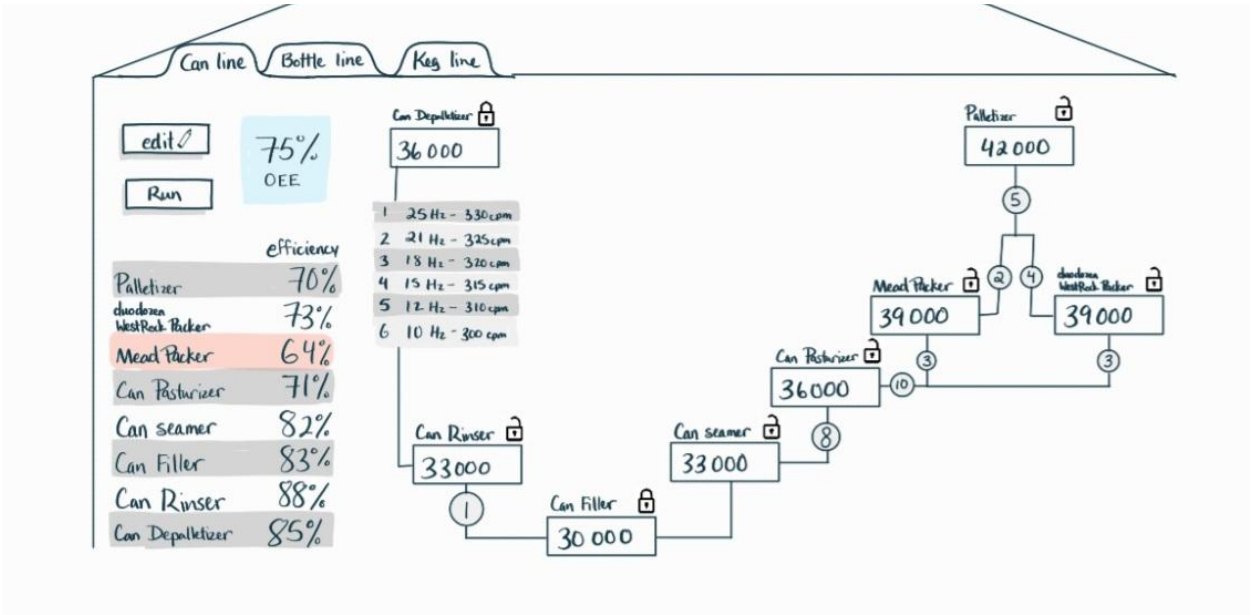


This window contains all the preset information of the production line which can be edited and new units and conveyors can be added to the line. There will be a finite amount of units and conveyors that can be added to a single production line. Inevitably, there will be too many units to display at once so the list can be scrolled horizontally.



→ **Unit edit boxes**

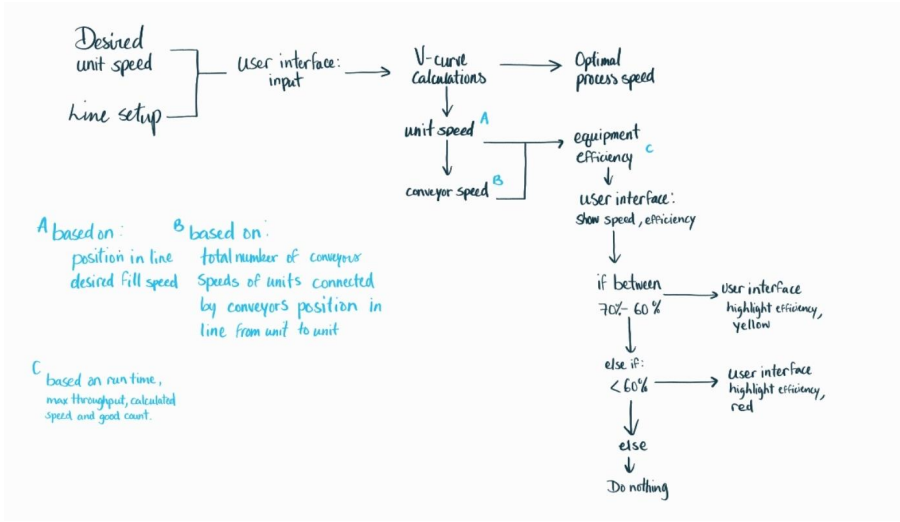
The unit edit boxes will be able to set the desired speed for the unit that the production line depends on. Although this will mainly be the filler line, other units can be the dependent unit by selecting the locks at the side of the unit editor of the selected unit. This makes those values constraints. Then to begin the calculations with the desired information the user will tap “Run”.



→ **Production line display**

The production line display will visually show the speeds of each part of the production line. The units will be displayed at different heights depending on their speed in reference to the slowest speed to visually Show the V-Curve. Additionally, all information relevant to the conveyors will be present. conveyors are shown as a circle with a number (the number of conveyors) when collapsed. When expanded they are shown in a table format with both frequency and cans per minute. Clicking on a unit will display their max speeds, and the conveyors should show both their cans/per minute speed and frequency (Hz).

2.2 Algorithm Breakdown



The Algorithm will be used to calculate the speeds and efficiencies of each unit with reference to the desired speed of any machine. The algorithm will be given the desired speed and the set up of the line. it will then calculate the speeds of all equipment in the line based on a v-curve with the filler as the bottleneck step. The units around the filler unit will need to be 10% faster and increment to be ten percent faster based on their distance from the filler speed.

For example if there was a production line set up like so:

Depalletizer - Rinser - Filler - Pasteurizer - Packer

Then the speeds of the rinser and the pasteurizer will be 10% faster then the filler, and the depalletizer and packer will be 20% faster than the filler. The conveyors between the units will have speeds that incrementally build up to the speed of the next unit. The graduation of conveyor speeds from one to the next will be determined by how many conveyors are between each line and the change in the conveyor widths. It will also calculate the efficiency of each machine and the overall equipment effectiveness (OEE) based on the inputted information and the calculated data. Finally, each machine efficiency and the OEE will be marked if they are outside an acceptable range (to be determined).

2.3 Hardware Specifications

Our software will be a native iOS application that will run on an iPad. The iPad was the client's preferred mobile device, as he wanted to bring the device into the production line area, and the company already has apple devices in use. To properly integrate into pre-existing company products and listen to the client's preferences, we had to switch from our original windows application concept to an iOS application.

To program and construct the software application, Xcode will be used. It is an IDE that can only be used on an Apple computer (Mac).

3.0 List of Equipment

In the table below, we have outlined the software and technology that we will be using to create our final product.

No	Item name	Description	Type	Prototype name (refer to section 7.0)	Source
1	Xcode	IDE for native iOS application	Software	All prototypes except for ExcelA	https://developer.apple.com/xcode/
2	Xcode Object Libraries	UI libraries for personalising and customising UI	Software	- UI shell - R1 - R2 - R3	https://developer.apple.com/documentation/uikit https://developer.apple.com/xcode/swiftui/
3	Excel	Application used to create algorithm and compute test cases	Software	- ExcelA	https://www.microsoft.com/en-us/microsoft-365/excel
4	TestFlight	iOS application testing software	Software	Prototypes that require Ipad testing simulation - UI shell - R1 - R2 - R3	https://developer.apple.com/testflight/
5	Ipad	Device used to run application software	Hardware	Prototypes that require Ipad - Xcode tut - R1 - R2 - R3	https://www.apple.com/ca/ipad-air/
6	Mac	Device used to run Xcode	Hardware	All Prototypes except ExcelA	https://www.uottawa.ca/faculty-arts/facilities-resources

4.0 Bill Of Materials (BOM)

In the table below, we have outlined the required materials and costs for creating our final product.

No	Item name		Description	Units of measure	Quantity	Unit Cost ($)	Extended Cost ($)	Link
1	Apple developer account (refer to section 6.0 on free alternative ways of deploying application)		To deploy the application for the company	Subscription	1	99	99$ per year the app is used (possibility that subscription can be cancelled without losing the application)	https://developer.apple.com/programs/enroll/
2	Xcode		IDE for native ios application	Software application	1	0	0	https://developer.apple.com/xcode/
3	Xcode Object Libraries	Swift UI	UI libraries for personalising and customising UI	Additional software download	1	99$ one time cost 29$ yearly cost	0 (we only need 1 year subscription)	https://developer.apple.com/xcode/swiftui/
		UIkit			1	0	0	https://developer.apple.com/documentation/uikit
4	Excel		Application used to create algorithm and compute test cases	Software application	1	0	0	https://www.microsoft.com/en-us/microsoft-365/excel

5	TestFlight	iOS application testing software	Software application	1	0	0	https://developer.apple.com/testflight/
6	Ipad	Device used to run application software	Equipment	1	0	0	https://www.apple.com/ca/ipad-air/ - personal devices for testing - Client also has devices available for deploying
7	Mac	Device used to use Xcode	Equipment	5	0	0	https://www.uottawa.ca/faculty-arts/facilities-resources
Total product cost (without taxes or shipping)						128$	
Total product cost (including taxes and shipping)						144.64$	

5.0 Project Risk

Teams must outline a list of the significant project risks and their associated contingency plans to mitigate the critical risks that are reasonably likely, in addition to the task plan update.

The list below highlights the potential risks that come with building our application. Under each risk, we have indicated the current solution to the potential issue and a series of contingency plans in-case our current solution is not feasible.

5.1 Deploying the iOS application to an iPad

The usual way an application is deployed onto an iPad device is through using a developer account. In our case, it can be released privately to a specific business. The main problem with this type of application is that access to a developer account requires a $99 service, renewed annually. It is unclear whether the application will be unusable if the account is deleted after we complete the project. Because this will take up most of our budget, it would be beneficial to avoid this cost.

Highlighted in the link below, the solution is to use a workaround method of deploying the application: https://ionic.io/blog/deploying-to-a-device-without-an-apple-developer-account This link explains that Xcode can directly deploy an application to a device.

a. Plan A: The apple developer account will be purchased and used to deploy the application, and the subscription will be deleted after the application is deployed on the clients device. Supposedly the app will work but can no longer be redownloaded after the developer account ends the subscription.
b. Plan B: Since the client has workplace iPads, it's possible they have a free developer account or enterprise account that could be used to deploy the application.
c. Plan C: Android Studio will be used instead and the application will be on google play store. An Android device could be purchased for the client.

5.2 Application Security

With the development of any software or app, there is always a potential risk of security breaches. Whether it's improper authentication of users, or vulnerabilities in the app code, there must be proper protective devices set in place. Because we are creating an iOS app, there is an increase in risk as, according to a figure outlined by Helen Vakhenko (agile.com, 2023), the "percentage of errors in app security mechanism" is 74% for iOS, about 17% higher than Android.

The current plan is to not release the application on the Apple store because it is possible to create a standalone application that can be placed on a single device. The instructions are highlighted in the following link:
https://ionic.io/blog/deploying-to-a-device-without-an-apple-developer-account
This should be secure as the app can only be used on a single device, that being the workplace iPad. This iPad should only be accessible by those with the proper authorization.

d. Plan A: Make the app only accessible after unlocking a password
e. Plan B: Implement a user authorization method that requires client identification to access application

5.3 Mac availability to use Xcode

Apple computers are needed to use Xcode. Only 2/5 members in our team own macs to use Xcode. With half our team unable to use Xcode, the process of coding our application would be too slow of a process to create a feasible design.

The current solution is to gain access to the arts faculty Atelier coFab lab in LRR 218, as they have a mac computer lab. We will want to get key card access as most of us would need to work in the lab after the 17:30 closing time. In addition, the Macs don't have Xcode installed, so we would need to request that the faculty installs Xcode on the computers.

f. Plan A: Convince the university to give us enough money to rent macs for a month using a service such as, https://meetingtomorrow.com/ottawa/computer-rentals/
g. Plan B: Find other IDE that use the same language as Xcode and those without macs can use them and transfer code to those that have macs for debugging and testing. Below, we have listed a couple examples:
    i. Swift:https://www.swift.org/blog/swift-on-windows/
    ii. Objective C: https://stackoverflow.com/questions/56708/objective-c-for-windows

6.0 Prototype Plans

Below, we have outlined the current plans we have in place for prototyping.

6.1 Tutorial of Xcode Application [Xcode tut]

This will be used for the deploy test.

This prototype will consist of a completed tutorial application using Xcode. It will have basic UI features and a basic algorithm that allows the user to manipulate various variables and output desired results.

6.2 Excel Algorithm [ExcelA]

This will be used for the algorithm test, as well as to create test cases for the integration test, when/if successful.

This prototype will consist of an excel spreadsheet that computes the V-curve theory algorithm and shows all the values, elements and functions needed to produce desired outputs in our application.

6.3 Pure UI [UI Shell]

This will be used for client UI aesthetics test

This prototype will consist of only the UI portion of the application done with UI libraries and Xcode. It will not perform any calculation or simulations.

6.4 Integration of UI and Algorithm [R1]

First rendition of application

This prototype will be a first rendition of our application and implement the excel algorithm into the previous Xcode UI application using Xcode.

6.5 Invalid Input Catching Feature [R2]

Rendition of application that catches invalid inputs

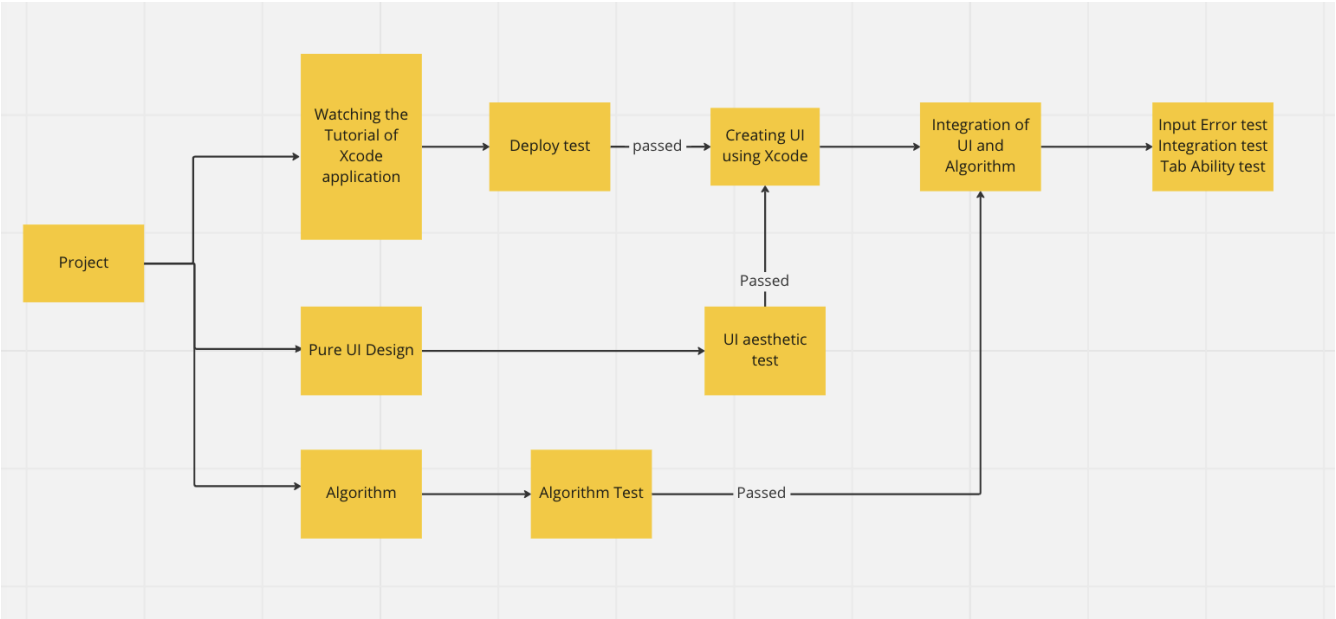
This prototype will be the first rendition of the application with the added feature of catching invalid inputs using Xcode.

6.6 Multiple Line Simulations Feature [R3]

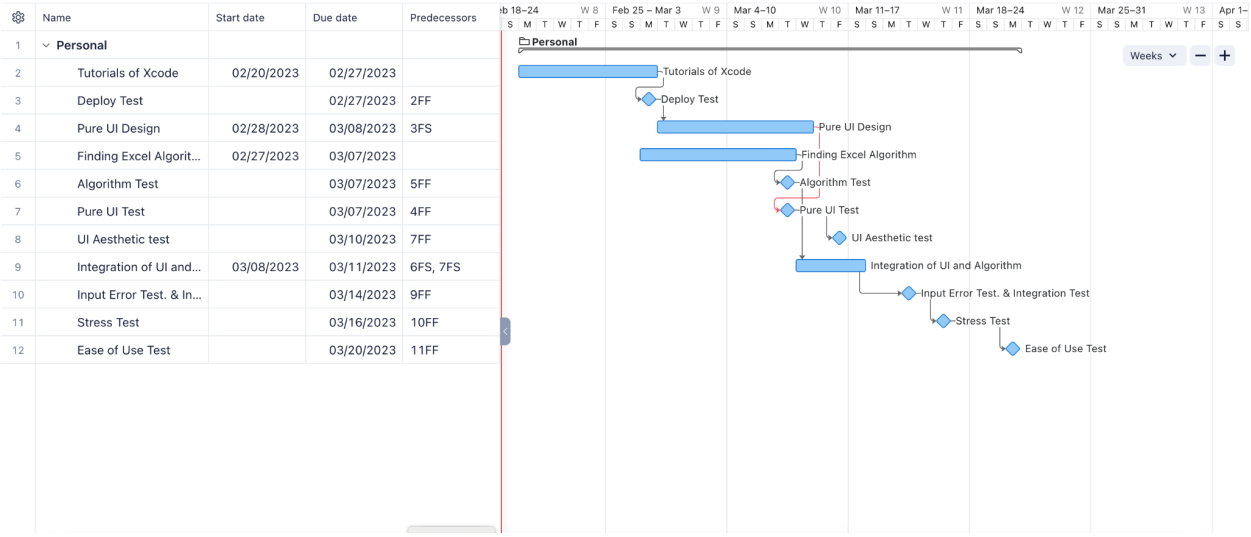
Rendition of application that can run multiple simulation in different windows (Full feature rendition)

This will be used for the stress test and the random user test.

This will be the prototype of the application with all of its features, especially the ability to create and save multiple production lines.



[Figure 1] Flowchart of Basic Prototyping Test Plan



	Name	Start date	Due date	Predecessors
1	Personal			
2	Tutorials of Xcode	02/20/2023	02/27/2023	
3	Deploy Test		02/27/2023	2FF
4	Pure UI Design	02/28/2023	03/08/2023	3FS
5	Finding Excel Algorit...	02/27/2023	03/07/2023	
6	Algorithm Test		03/07/2023	5FF
7	Pure UI Test		03/07/2023	4FF
8	UI Aesthetic test		03/10/2023	7FF
9	Integration of UI and...	03/08/2023	03/11/2023	6FS, 7FS
10	Input Error Test. & In...		03/14/2023	9FF
11	Stress Test		03/16/2023	10FF
12	Ease of Use Test		03/20/2023	11FF

[Figure 2] Full Gantt Chart of Prototype Test Plan
https://www.wrike.com/frontend/ganttchart/index.html?snapshotId=bsq464HdScN5voRvAB3Omv53WJ3De6SF%7CIE2DSNZVHA2DELSTGIYA

Team Member:

- Excel Algorithm group: Hanna Paik, Rafiullah Hamdard, Leila Smaili

- UI Design & Xcode group: Marie Levin, Yifei Li

7.0 Prototyping Test Plan

Test ID	***Test Objective*** ***(Why)***	***Description of Prototype used and of Basic Test Method*** ***(What)***	***Description of Results to be Recorded and how these results will be used (How)***	***Estimated Test duration and planned start date*** ***(When)***
1	[Deploy Test] The objective is to test if we need to buy a developer account to deploy an application for our client. And test the feasibility of using Xcode.	Prototype Xcode tut will be used. An application that will most likely be a tutorial of how to use/ getting started with Xcode will be attempted to be downloaded on an Ipad without a developer account.	**Record:** Whether the application was successfully deployed or not is recorded.(Pass/Fail) **Response:** If the test fails then a developer account will be used to deploy the real thing.	26-27th Feb
2	[Pure UI Test] The objective of the test is to make sure the program can be interacted with without issue before focusing on its ability to perform its task.	Test the functionality and ease of use of the UI. Prototype UI Shell will be used. A prototype of the UI that does not perform any real calculations or simulations.	**Record:** Score out of ten of its ability to stay formatted and display the expected response or change in UI. (#/10) Score > 8/10 : pass Score <8/10 : fail **Response:** Reprogram UI and Iterate until pass	4 - 7th March
3	[UI aesthetics Test] The objective is to test the aesthetics of the UI with the user.	Test the aesthetics of the UI with client feedback. Prototype UI Shell will be used. A picture of the UI will be sent to the client for feedback.	**Record:** A number from one to ten will be recorded on the client's likability of the UI colour scheme, Formatting and Aesthetics. Score > 8/10 : pass Score <8/10 : fail **Response:** Ask for feedback and fix the UI. Iterate until it passes.	6-10th March Need time for the Client to respond.

4	[Algorithm Test] The objective of this Test is to prove our ability to calculate the optimal speeds before implementing it into the application.	Test of the algorithm that predicts the optimal speeds. Prototype ExcelA will be used. An excel spreadsheet, that displays all the elements and functions of the algorithm and when given the historical data it is able to replicate the optimal speeds within a small percentage of error.	**Record:** The average margin of error that the algorithm's calculated optimised speeds are from the historical data. Margin of error must be within 15% for Pass. **Response:** Edit algorithm and Iterate until pass	4-7th March
5	[Integration Test] The objective of this test is to make sure that the display of the input and output is well formatted on the UI.	Test of the algorithm's integration into the UI.Prototype R1 will be used. A first rendition of the application will be created that includes the UI and the algorithm working together, the focus will be on the formatting of the input and output numbers and other variables on the UI and the ability of the user to change aspects of the production line and the algorithm still calculates things accordingly to test cases made from Excel prototype.	**Record:** The Pass or Fail of the applications ability to correctly format the UI and output correct speeds of three test cases derived from the Excel prototype. 3 Passes = Pass < 3 Passes = Fail **Response:** Reference working Excel algorithms and find out what's different or causing problems. Iterate until 3 passes.	9-10th March

6	[Input Error Test] The objective of testing wrong inputs is to make sure that the inputs cannot exceed the max speeds of the conveyors.	Test of the UI ability to catch error inputs. Prototype R2 will be used. On the second rendition of the application where invalid errors are caught, a comprehensive collection of bad inputs will be tested on each possible place where the user can do something wrong.	**Record:** Number of invalid inputs that were stopped Number of invalid inputs that were accepted The ratio between invalid inputs caught and accepted must be 90% caught to pass. **Response**: catch invalid inputs that were accepted and iterate until pass	13-14th March
7	[Multiple Tab Test] The objective of this test would be to make sure that the program is able to handle multiple sets of inputs and display each accordingly.	Test of the applications ability to simulate multiple production lines and save and load the users edits to each production line. Prototype R3 will be used. On the third rendition of the application where multiple lines can be added. Multiple production lines will be added to the application. They will be edited, saved and loaded at selected points throughout a simulated use of the application.	**Record:** Based on the information of different production lines. Three production lines will be tested to run on multiple simulations at once. If they can be created, saved and loaded in the right places during regular usage of the application then the test is passed. **Response:** Figure out what went wrong and fix the code and iterate the test until it passes.	13-14th March

8	[Stress Test] The objective of the test would be to figure out and test the limits of the application so measures can be put in place to make sure that the user knows the limit of the application	A stress test on the amount of data can be added to the program before it loses functionality or UI formatting. Prototype R3 will be used. On the Third rendition of the application. Production lines will continuously be added to the application until it crashes. Conveyors and Units will continuously be added to a single line until the UI formatting becomes illegible or the program crashes. Extremely large numbers will be added to the filler speed or other inputs until the application crashes or the UI formatting becomes illegible.	**Record:** The limit to the number of production lines, conveyors and units and numbers that can be inputted into the system before the UI gets messed up or the application crashes. If the limits < the theoretical values that would be needed to run the application under normal conditions (determined from the information we gathered for multiple tab tests) then it fails. **Response:** Find the source of the data storage limit and work around or buy more storage. Fix how data is saved and loaded if there is a formatting issue. Iterate until pass.	15-16th March
9	[Random Person Ease of Use Test] The objective would be to make sure that the program and final user manual is comprehensive for the client.	A test on the ease of use of random person ability to follow an instruction manual and the application. Prototype R3 will be used. The application and a user manual will be sent to a person with no knowledge of beer production. They will be required to complete a set of instructions from the user manual. They will be asked to give feedback on the ease of use.	**Record:** The feedback and the percentage of tasks that the person was able to complete will be recorded. **Response:** Use feedback to improve the user manual and UI if failed. Iterate until pass.	March 18-20th.

Wrike Snapshot:

For Delivery E:
https://www.wrike.com/frontend/ganttchart/index.html?snapshotId=eo7MlRnKuQJ9ATdtvx1LNuX50kTdlS6v%7CIE2DSNZVHA2DELSTGIYA

For Project Plan:
https://www.wrike.com/frontend/ganttchart/index.html?snapshotId=bsq464HdScN5voRvAB3Omv53WJ3De6SF%7CIE2DSNZVHA2DELSTGIYA

8.0 References

Abdulrasool, S. (2020, September 22). *Introducing Swift on Windows*. Swift.org. Retrieved February 19, 2023, from https://www.swift.org/blog/swift-on-windows/

Before You Enroll. (n.d.). Apple Developer. Retrieved February 19, 2023, from https://developer.apple.com/programs/enroll/

Beta Testing Made Simple with TestFlight. (n.d.). Apple Developer. Retrieved February 19, 2023, from https://developer.apple.com/testflight/

CEED. (2022, August 10). ☕. YouTube. Retrieved February 19, 2023, from https://en.wiki.makerepo.com/wiki/Professional_development/Design_thinking/Detailed_designs

Facilities and resources | Faculty of Arts. (n.d.). University of Ottawa. Retrieved February 19, 2023, from https://www.uottawa.ca/faculty-arts/facilities-resources

iPad Air. (n.d.). Apple. Retrieved February 19, 2023, from https://www.apple.com/ca/ipad-air/

Microsoft Excel Spreadsheet Software. (n.d.). Microsoft. Retrieved February 19, 2023, from https://www.microsoft.com/en-us/microsoft-365/excel

Muramoto, A. (2016, August 2). *Deploying to a Device without an Apple Developer Account*. Ionic.io. Retrieved February 19, 2023, from https://ionic.io/blog/deploying-to-a-device-without-an-apple-developer-account

Objective-C for Windows. (n.d.). Stack Overflow. Retrieved February 19, 2023, from https://stackoverflow.com/questions/56708/objective-c-for-windows

Ottawa Computer Rentals | Laptops, MacBooks, Chromebooks. (n.d.). Meeting Tomorrow. Retrieved February 19, 2023, from https://meetingtomorrow.com/ottawa/computer-rentals/

Vakhnenko, H. (2023). *Mobile App Security Risks And Their Impact On Your Business*. Agilie. Retrieved February 19, 2023, from https://agilie.com/blog/mobile-app-security-risks-and-their-impact-on-your-business

Xcode 14 Overview. (n.d.). Apple Developer. Retrieved February 19, 2023, from https://developer.apple.com/xcode/