Team 12

CAD Import For Robotic Dispensing System

By: Francis Dator, Adrian Manrique, Jarrett Koran, Gerald DeVera

Project Report

Version: (5.0) **Date: (5/02/2022)**

1 Abstract

Nordson uses precision fluid dispensing systems that utilize automation that is programmed to help assist in the designing of electronics. To specify, Nordson ASYMTEK's machinery utilizes PCBs (Printed Circuit Board) created by manufacturers to complete tasks during its final manufacturing process. When dealing with circuit boards specifically, the machinery takes in Raw Gerber files or ScanCAD renderings of CAD drawings which contain information on a PCB's design. Although these are the formats currently supported, other additional formats are to be determined. The team is tasked to translate and refactor pre-existing Python to C# in order to integrate the software needed to receive information imported by the user. These files serve as important information to help the machinery understand dispensing elements.

One of the most important parts of the PCB blueprints is the locations of fiducials and dots. Both fiducials and dots will help to dictate the cartesian coordinates in order to create the desired product. To be more specific, the project at hand is meant to take CAD drawings of PCBs and translate dispensing elements such as Dots and Fiducials into canvas recipes, however, there may be instances where fiducials are not specified within a user's CAD drawing. After extracting the information, the software would then be able to grant a user an XML file which the machine is able to translate into instructions. Through the use of the CAD Import API, the team will be able to properly convert files by utilizing ScanCAD tools needed in reverse engineering pre-existing CAD drawings. Similarly, the utilization of Gerber files as used by PCB manufacturers will define each layer for PCBs required for canvas recipe instructions. Along with these tools, the team will also create a user-friendly GUI aimed to assist users by alerting them to failures and displaying a file's progress.

In consideration of both the benefits and impact of the system, it will help to implement new machinery used at a Nordson facility in charge of creating PCBs. The program created by the team will help to translate files to a language understood and used by both machinery and both current and future employees at Nordson. The local impact of this project is that Nordson will be able to use and further develop the file translation to be used with their simulation software. From a wider perspective, it is possible that the team, along with other companies, can use this software when reorganizing or converting file types. At this point in time, the system is only used for the development of circuit boards at Nordson, but with further interest and development, it is possible that the system can be used in a multitude of software systems.

This report will cover the Problem Statement including the Business Background, Project Needs, and the Objectives for the system to be. Next, the document will include the User and System Requirements explaining the objectives for the system including the exploratory studies conducted by the team when researching a multitude of frameworks and techniques. Other facts explained in the documentation include the team's System Design, System Implementation, and System Testing conducted throughout the entirety of the project. Finally, the documentation will incorporate any Challenges and Issues the team has faced, and the experience gained as a result.

2 Report Revision History

2.1 Changes in Version 1.5

Report changes in version 1.5 consist of a name change for the project title. Other changes include reformatting of Project objectives along with additional grammatical errors. Most importantly any information that is needed regarding the system implementation, including version control, coding conventions, and programming tools.

2.2 Changes in Version 2.0

Report Version 2.0 report consists of changes from the feedback given by the team through its industry mentor. These changes include the correction of semantics within the documentation, the revision of requirements to allow for consistency with the project description given by Nordson, and the reassignment of functional requirements to their proper user requirements. Aside from these changes, the Version 2.0 Report also includes the implementation of sequence diagrams and class diagrams covering factory/strategy design patterns and MVVM detailing the design of both the systems frontend and backend development. Finally, report Version 2.0 elaborates on the problems the team experienced during the second sprint along with the current status of the team's issues.

2.3 Changes in Version 2.5

Report Version 2.5 consists of feedback provided by the faculty advisor. Changes include further detail regarding section 8 for System Testing, section 11 for the conclusion, and the addition of a project timeline also located in section 11.

2.4 Changes in Version 3.0

Report Version 3.0 consists of changes made to the use cases and functional requirements. To be more specific, the existing requirements and use cases did not detail the depth of the system resulting in a more specific explanation for the systems development and its requirements. These specific changes can be seen within section 4 Requirements and include the removal of SFC (System Functional Requirement) 05 detailing the unit of measurement and the revision of SFC 04 detailing the output file based on XML Recipe.

2.5 Changes in Version 3.5

Report Version 3.5 includes all grammatical errors, and an updated timeline to reflect sprints 7, 8, and 9, along with additional test cases created by the team as seen in Section 8.0. Section 8 also includes created test cases on the User Interface and the testing for the CAD Converter class.

2.6 Changes in Version 4.0

Report Version 4.0 includes the removal of SF-A-02 under the User Requirement U0-01. With this in mind, the document also represents the updated table shown in Figure 2. Other additions include an updated Section 6.3, Section 7.4, and Section 7.5, Analysis and Key Algorithms detailing information regarding File Conversion. The team has also implemented Section 8, Unit Testing. This Section includes both File Importer Exporter Unit Tests and View Model Unit Tests. Finally, the team has updated Section 10.1 which details setting up the development environment.

2.7 Changes in Version 5.0

Report Version 5.0 includes final changes to the Abstract, Section 1.0. These changes include the organization based on business context, main features implemented, broader impacts of the system, and an outline for the report.

3 Problem Statement

3.1 Business Background

The machinery used for Asymtek machinery utilizes Canvas Recipe files for PCB blueprints. The current program that is designed to translate both Gerber and ScanCAD files is designed in C# as a standard language used by Nordson. The existing program that serves as a base for the system, currently only accepts a single specific ScanCAD document and is programmed in Python. In order to implement the software and accept multiple file types, the team is tasked to convert any existing code and refactor based on inconsistency and improvements that can be made within the program including the ability to accept additional file types. The program itself involves taking a user's import from PCB drawings into a Canvas Recipe. These recipes hold information about the dispensing elements that are needed for electronic components to be attached to the PCB. To be more specific, if a user's drawing proceeds through a file authentication where PCB drawings will be converted into Canvas Recipe files. XML.

3.2 Needs

The project will convert both ScanCAD and Gerber file formats. These files, if accepted, will be converted into Canvas Recipe Files(.XML) after the file has passed a file authentication. With this in mind, however, the project will need to work directly with ScanCAD files as these will be formatted, directly tested, and expanded on additional files as used by PCB manufacturers. The system aims to obtain units of measurements as given through a user's CAD drawings.

3.3 Objectives

- Understand different file formats
- Develop a system GUI
- Utilize an existing Scancad API
- Establish Software Development Environments
- Create a program capable of translating Gerber and ScanCAD files
- Understand the machinery's use of Canvas Recipe for PCB blueprints
- Develop a user-friendly interface for the user import status
- Develop a system that efficiently transfers files flexible to the user's desired output.

4 Requirements

4.1 User Requirements

4.1.1 Glossary of Relevant Domain Terminology

Scancad API - Application Programming Interface that will handle user file conversion, and verify file authenticity.

GUI (Graphical User Interface) - System interface that will be developed by the team to allow for a user-friendly interface. Handles tasks such as user requests and system notifications.

PCB (Printed Circuit Board) - Created using Canvas Recipe files through the use of intelligent and precise machinery. Blueprints for each board utilize user imports converted XML files to determine instructions.

XML (Extensible Markup Language) - File type that is required for the machinery to understand blueprints. This file will be created after the user imports a file to be converted.

Gerber file - An acceptable ASCII vector format file a user can select to be imported to be converted into an XML. These files will contain blueprint information for PCB design including vector coordinates.

ScanCAD file - An acceptable file type that a user can select to be imported to be converted into an XML. These files include blueprint information such as vector graphics and many others.

Fiducials - Marker information located within a user's imported file. These markers will be calculated within a recipe and serve to give the machinery both a field of view and point of measurement.

Fluid dispensing system - A system used by the machinery to precisely dispense fluid requested by the blueprints. This system through the instructions provided by a Canvas Recipe File will be able to calculate the amount, location, and many other factors.

UML (Unified Modeling Language) - Consists of diagrams that provide visual representations of the system specifications, construction, and documentation.

Python - Language used in existing Software originally designed for importing and converting ScanCAD and Gerber files to Canvas Recipe Files (XML). This code will later be translated into C#.

C# - Language that already exists and is used through Nordson. Will be the language utilized to both import and convert user imported files into Canvas Recipe files.

Cartesian Coordinate System - A system that is utilized to locate a specific point based on the coordinates provided. One of the methods used to calculate fiducials is designed within a user's blueprints.

Nordson Asymtek - Company that provides services for automating fluid dispensing. Nordson also designs and manufactures equipment such as PCBs as designed by the customer.

Canvas Recipe File - Blueprints converted from ScanCAD or Gerber format and converted into an XML file.

SDE (Software Development Environment) - Consists of programming tools used to create the system. The system will utilize tools such as Scancad API, TKinter libraries, and many others to develop the system and the GUI.

MVVM (Model-View-ViewModel) - A structural design pattern that separates objects into three distinct groups the model, view, and view model.

4.1.2 User Groups

The system is designed to accommodate one type of user. That user has access to all functionality of the program. There is no need to add additional user groups. This is because there is no hierarchical structure or security restrictions present within the system.

4.1.3 Functional Requirements

4.1.3.1 Project Scope

Within the scope of this project, the team has been able to come up with a few key user requirements. The user is first expected to be able to import an appropriate Scancad file into the software system and be able to request and download a Canvas Recipe File in the end.

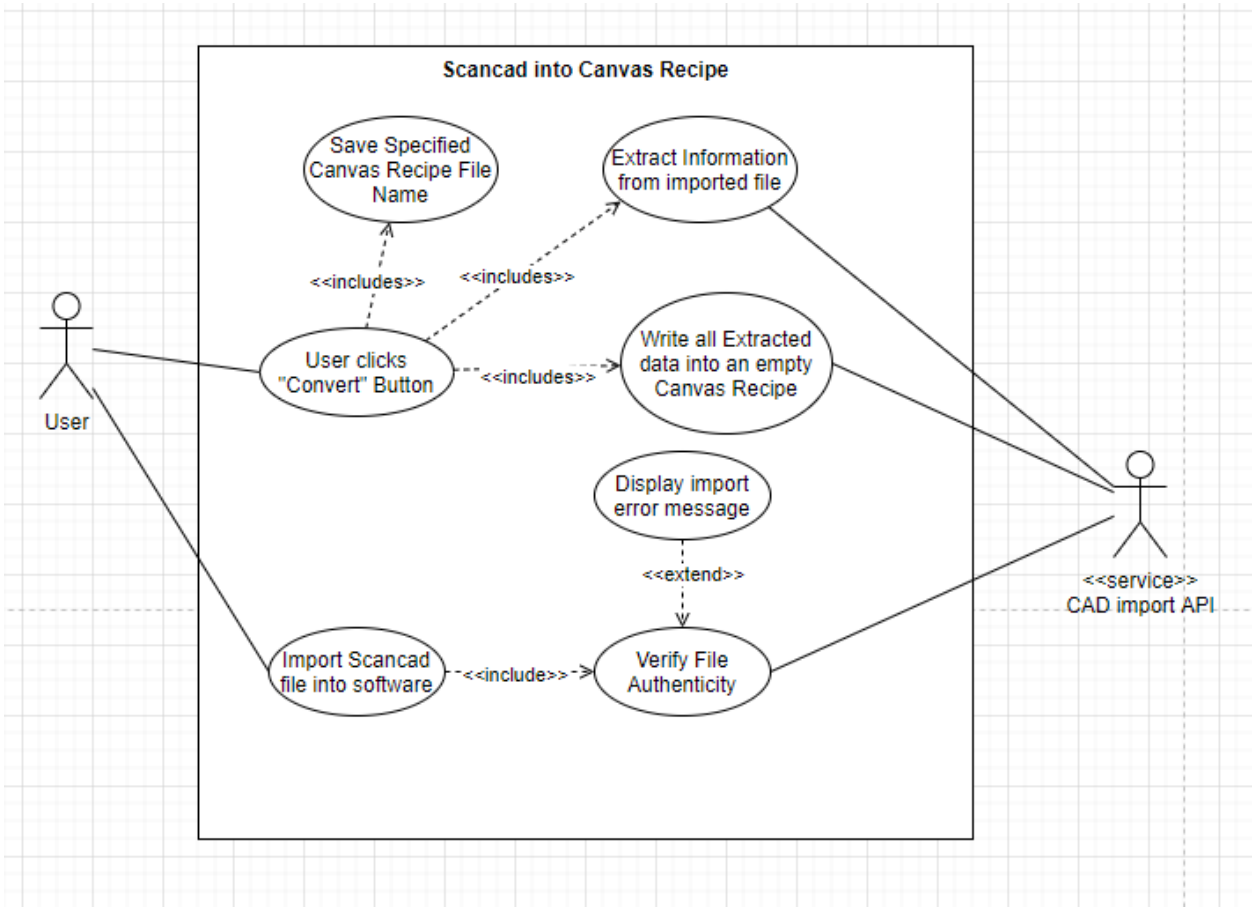

Figure 1

4.1.3.2 User Scenarios

To gain a better understanding of the requirements, a use case table was created along with the diagram. A few of the important use cases showcased in the table (located in Appendix U of this document) involve the main goals of the software.

- UC-001; Import Scancad file into Software
    - This scenario involves the user importing a specified file into the software of the system. The main step the user is required to do is select a file and place it into a GUI element in the software. The system then goes through multiple steps of detecting previously imported files, verifying the currently imported file, and displaying any errors in the process to the user.
- UC-003; Request Canvas Recipe File
    - This scenario involves the user receiving an output file. The user is required to make an initial request which involves the interaction of a button. The system then handles extracting information from the current imported file and then writes the extracted information into a Canvas Recipe. The user finally saves the output file under a user inputted name.

4.1.3.3 User Functional Requirements

The user functional requirements encompass the base functionality required for the program to operate. It should include basic data input, output, and other information necessary for successful operation.

- UF-A
  - The user shall be able to import a file.
  - This functional requirement encompasses the main data input for the program.
  - This user requirement is refined into SF-A-01
- UF-B
  - The user shall receive a written Canvas Recipe.
  - This functional requirement encompasses the main data output for the program.
  - This user requirement is refined into SF-B-01, SF-B-02, SF-B-03, and SF-B-04

4.1.4 NonFunctional Requirements

4.1.4.1 Product: Usability Requirements

The nonfunctional usability requirements outline the specifications for the program ensuring easy use of access by any user.

- UP-01
  - The user shall be able to easily use the system.
  - This requirement encompasses the entirety of nonfunctional requirements established for the system.
  - This requirement is refined into SP-01-01, SP-01-02, SP-01-03, SP-01-04

4.2 System Requirements

4.2.1 Functional Requirements

4.2.1.1 System Functional Requirements

The system functional requirements encompass the system requirements refined from user functional requirements and detail the necessary functionality in order for the system to operate successfully.

- SF-A-01
  - The system shall accept a ScanCAD file.
  - This system requirement is refined from UF-A
- SF-B-01
  - The system will write the extracted information into a selected empty Canvas recipe.
  - This system requirement is refined from UF-B

- SF-B-02
    - The system shall send the converted file to the user
    - This system requirement is refined from UF-B
- SF-B-03
    - The system shall extract information from imported ScanCAD.
    - This system requirement is refined from UF-B
- SF-B-04
    - The system should set up an output file based on an XML recipe.
    - This system requirement is refined from UF-B

4.2.1.2 Data Requirements

The data requirements for this system are as follows. Currently, the user is mainly expected to input files into the system. The user is also required to input measurement units and a name for the output file. The initial file type accepted is a ScanCAD or Gerber file. There are plans to include Gerber files in the near future. The system should be designed so that adding additional file types is easy moving forward. The only measurement input currently allowed is millimeters, which is also the default. Future unit measurement inputs will be added as specified by the mentor. As for name inputs, this should be limited to letters(both uppercase and lowercase), numbers, and "_" or underscores in place of a space character. The main data output should be Canvas Recipe. These Canvas Recipes are in. XML file extension format. There is no need for a functioning database as no information is required to be stored long term.

4.2.2 NonFunctional Requirements

4.2.2.1 Product: Usability Requirements

- SP-01-01
    - The system will provide a button to start processing the imported file.
    - This system requirement is refined from UP-01
- SP-01-02
    - The system shall prompt the user to save to a specified Canvas Recipe File Name through the GUI once the CAD Import has finished. through the GUI once the CAD import has finished.
    - This system requirement is refined from UP-01
- SP-01-03
    - The system shall display CAD import status(success/failure).
    - This system requirement is refined from UP-01
- SP-01-04
    - The system shall explain the reason for process failure.
    - This system requirement is refined from UP-01

4.3 Requirements Trace Table

Project Name: CAD Import for Robotic Dispensing System			
User Requirements		**System Requirements**	
Req ID	**Description**	**Req ID**	**Description**
UF-A	The user shall be able to import a file.	SF-A-01	The system shall accept a ScanCAD file.
UF-B	The user shall receive a written Canvas Recipe.	SF-B-01	The system will write the extracted information into selected empty Canvas recipe.
		SF-B-02	The system shall send the converted file to the user.
		SF-B-03	The system shall extract information from imported ScanCAD.
		SF-B-04	The system should set up an output file based on an XML recipe.
UP-01	The user shall be able to easily use the system.	SP-01-01	The system will provide a button to start processing the imported file.
		SP-01-02	The system shall prompt the user to save to a specified Canvas Recipe File Name through the GUI once the CAD Import has finished.
		SP-01-03	The system shall display CAD Import status (success/failure)
		SP-01-04	The system shall explain the reason for process failure.
Acknowledgment: Generated from the CapStone process management system ©2022			

Figure 2

5 Exploratory Studies

5.1 Relevant Development Frameworks

The framework design patterns suggested by the team's industry mentor were the strategy design and factory design. The strategy design pattern is a pattern that allows a user to choose which algorithm to use at runtime instead of having just a single algorithm implemented, where the factory is for creating objects without having to make it a part of a specific class. Both of these are very modular designs. The strategy pattern is suitable for this project because it allows modular changes which can be applied to file types for the input. The strategy design is the best for the beginning operation of the program. With the strategy pattern, the software system selects the algorithm at runtime instead of implementing a single algorithm. For the system the team strives to achieve, selecting which file type to import and convert will play right into the design that comes with using the strategy design pattern.

The factory design is the one this team is planning on using when the project is more developed because it can support multiple classes and objects. This works well with our project because we will be creating a single object with different possible file types for input and output. The industry mentors have requested that Object-Oriented Programming be used for the system.

Both the strategy and the factory methods are fitting for the requested system. The team thought of some other options in case of running into issues later in development. Some examples of other methods are the abstract factory method, possible implementation of an object pool, and possibly the facade method. The abstract factory method would be a method that allows the system to create multiple objects based on certain templates, and the user could choose which template object they would like to receive. The facade is another object-oriented design method where a facade object serves as the mask in front of the more complex backend structural code.

5.2 Relevant Solution Techniques

Nordson uses ScanCAD as a 3rd party product to assist with the development of their technologies. ScanCAD International Inc. is an American corporation founded in 1990 with a multitude of services and products. The team will be utilizing ScanCAD to reference, look at required data, and use the ScanCAD file as a baseline for the operation of the software system. The industry mentors at Nordson are working on providing relevant hardware and software to the team for the development and testing of the new software system. Included in this will be a copy of the ScanCAD software utilized in the current system Nordson uses.

As of now, XAML is the only 3rd party open source library being utilized in the software system. XAML will be used to develop and design the GUI of the application. This is a GUI development tool similar to the one currently being used, Tkinter. Because of the shift from Python code to C#, the use of XAML is needed to redesign the GUI.

For the team's version control, Git will be used through the service GitHub.

5.3 Broader Impacts

This project will help to implement new machinery used at a Nordson facility in charge of creating PCBs as well as allow for the consumer to submit their designs. The program created by the team will help to translate Gerber files to a language understood and used by both machinery and both current and future employees at Nordson. The local impact of this project is that Nordson will be able to use and further develop the file translation to be used with the simulation software.

Looking at this project with a wider view, it is possible that the team, along with other companies, can use this software or similar software to create more systems to reorganize or convert file types into the needed type for the hardware system being used by that company. As of now it only relates to the development and creation of circuit boards and relevant technology

at Nordson, but with further interest and development, it is possible to go even further and be used in a multitude of software systems.

Going back to a more narrow viewpoint, our team will gain valuable industry experience and learn how to utilize new and unique hardware elements in software systems. The information and new technologies and techniques the team will learn from working with Nordson will be something all members of the team will carry into our future professions.

6 System Design

6.1 Architectural Design

The architecture of the system utilizes a mix of the factory and strategy design patterns to easily allow abstract classes to be implemented and utilized within the system. The GUI executable package will contain the GUI that the user will be interacting with to access the rest of the system. The CadImportDLL contains the factory method of the system which will also contain the CadImportExporter which will allow for information to be extracted from the files that have been imported to the system from the user.

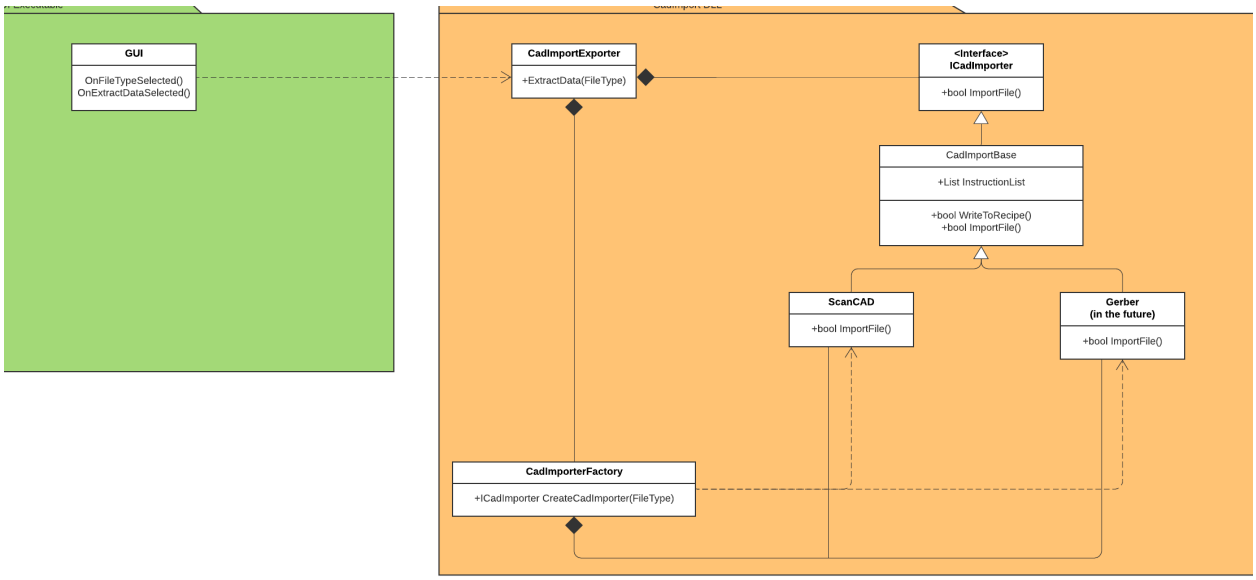

Figure 3

6.2 Structural Design

For a User Interface Design, the system will be utilizing a Model-View-ViewModel (MVVM) pattern. The view and ViewModel will consist of the frontend, or the user interface of the system, which will send information back to the model portion, or the backend logic.

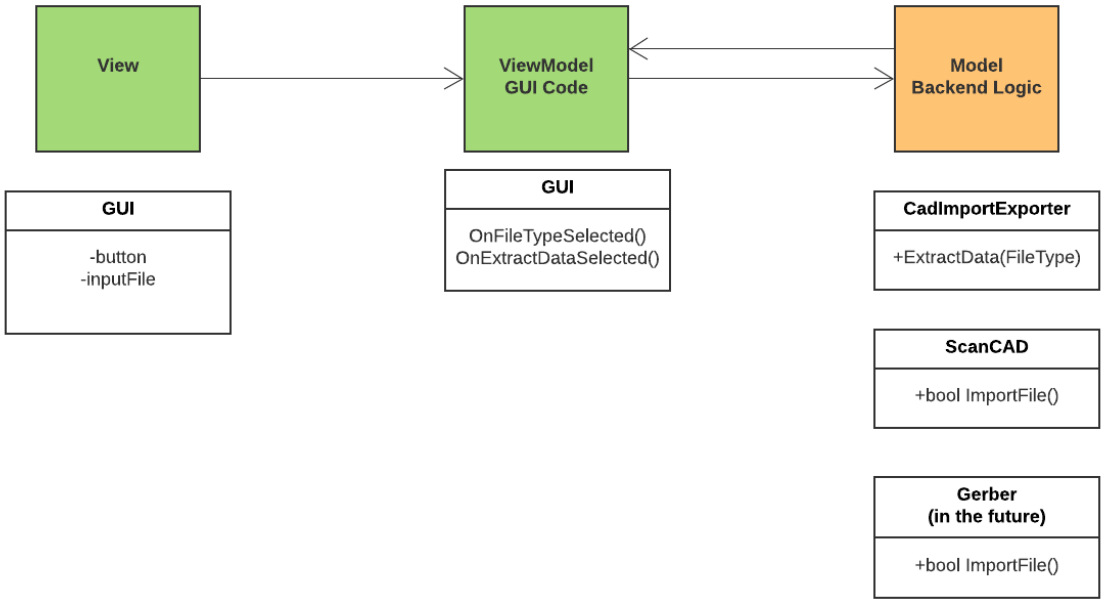

Figure 4

6.3 User Interface Design

The user interface features buttons that will be available to the user depending on which stage of the process he/she is in. These features include importing a file and requesting a written canvas recipe. This UI will also provide messages to the user regarding the status of the process. This includes file verification, import status, writing completion, and any other necessary information.

6.4 Behavioral Design

For the behavioral design, a sequence diagram is utilized to help showcase how the system is going to interact through the different levels. In the current system iteration, there exist two major actions that start through the user input. Both of which are import ScanCad, and Request Canvas Recipe user requirements, as stated back in section 4.1.3.2.

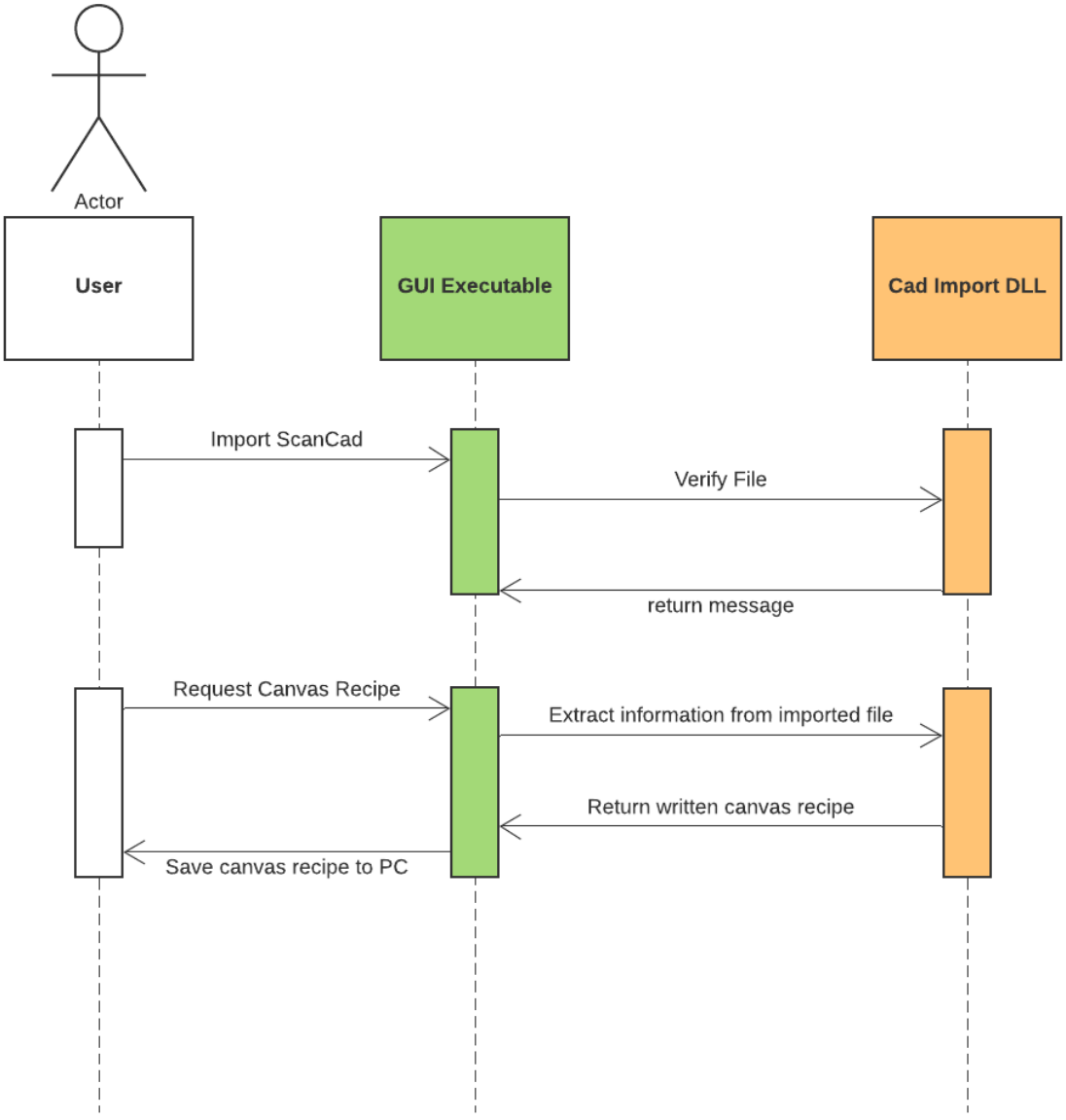

Figure 5

6.5 Design Alternatives & Decision Rationale

The team decided to use a mix of factory and strategy for the architecture. The reason behind this is that the team believes that this is a better alternative due to the compact nature of the system. We found this to be a great fit, and while looking at alternative designs such as the facade, decided that the initial decisions were already sufficient to meet the requirements of the system.

7 System Implementation

7.1 Programming Languages & Tools

The coding language being used is C# at the request of the team's industry mentor. The team believes that C# was chosen because it is often used in application development as well as being easy to maintain and highly versatile. C# is also a very popular language for Windows-based applications. Another language the team will work with is Python, having to translate and adapt Python code as a part of the software development. The GUI will be designed using Visual Studio and XAML, these are based on C# and will be used for the superficial workings of the user interface. For the backend of the GUI, we will be using the Windows File Explorer API to access the files on the computer that is using the application.

7.2 Coding Conventions

The team plans to use Object Oriented Programming heavily in the coding of the software system. This is at the request of the industry mentor, as they wish for the team to create versatile and scalable software. The team briefly considered working with other coding conventions such as Functional Programming, however, the industry mentors requested the use of Object-Oriented Programming, so there was no need to look further.

7.3 Code Version Control

Code version control will be managed by using the software Git through GitHub. The team decided to use Git because all members of the team are familiar with its use and have worked on projects using it in the past. Using a familiar version control software makes the team's task easier by eliminating the worry of learning new software.

7.4 Implementation Alternatives & Decision Rationale

The team did not have any implementation alternatives. All decisions made initially were not changed as the system requirements were properly met. These decisions include using a combination of Factory and Strategy design patterns. No other alternatives were suggested.

7.5 Analysis of Key Algorithms

For the conversion of ScanCAD files into an XML recipe, a complex algorithm detailing the coordinate positions of fiducials and lines is required. Overall the process is split into two distinct phases. The initial phase extracts coordinates from the imported file and systematically stores these coordinates as instructions in a list. The second phase involves loading an XML Recipe file and then updating the instructions inside the Recipe with the coordinate data located inside the Instructions list. the process would be the same for any future file types. Although, the implementation for extracting data will differ due to the difference in how certain file types organize information. This can include delimiters such as commas, semicolons, or any other unique forms of organization.

8 System Testing

8.1 Test Automation Framework

For an automated testing framework, the team is utilizing Unit testing with MSTest which is a testing framework built into Visual Studio. MSTest is used for unit testing with C# and since it has built-in integration with Visual Studio, it was able to be added to the system with relative ease.

8.1.1 Steps for Installing Test Framework

The test framework is installed automatically with Visual Studio. The only requirement from the user is to add it to their project solution, which they can do by simply navigating to adding a new project and selecting the test project which utilizes the .NET framework. If the framework is not installed automatically, Visual Studio utilizes a package manager, where the user may be able to install MSTest. TestFramework and MSTest.TestAdapter for the testing solution.

8.1.2 Steps for Running Test Cases

After writing an appropriate test case, the user may be able to run their test by simply navigating to the test tab located at the top of the Visual Studio Window, and clicking “Run All Tests”.

8.2 Test Case Design

The team has one preliminary test case provided to us by the company. This is an example ScanCAD file which includes the fiducial, line, and dot information that needs translating into XML format. The team was also provided with an expected output XML file to validate this test case. Outside of validating whether the system properly translates the proper information.

Testing related to the UI is designed to make sure that the correct files are being received by the system and the correct files are being outputted by the system after the conversion has been completed. At this time it is difficult to add more tests due to the limited resources available to the team; we are hoping to receive further resources in order to more thoroughly test the system and see if the application has the potential to be applied to existing systems present within Nordson Asymtek.

8.2.1 Test Suites

The first suite tested in the system is the FileConverter(backend) logic. This test suit ensures the file conversion process is working properly. The classes covered include the FileImporterExporter, FileImportBase, and CadFileType.

The testing suite for the ViewModel currently ensures that the logic pertaining to storing file path information is working properly, as well as making sure that the conversion progress is being tracked and notifying the user of the status of the program. This suite also tests to see if the responses from the FileConverter logic are being returned correctly and that the ViewModel is handling expected outputs from the FileConverter.

The ViewModel testing suite does not cover commands, as those commands are called in response to activity on the View, and their logic only calls logic that is covered in the testing suite.

8.2.2 Unit Test Cases

File Importer Exporter Unit Tests:

TestExtractDataWithCorrectFiles
- Input: Existing correct files
- Expected Output: Successful conversion

TestExtractDataWithEmptyImport
- Input: Empty ScanCAD
- Expected Output: Format Exception

TestExtractDataWithWrongImport
- Input: Invalid import file type
- Expected Output: Argument Exception

TestExtractDataWithEmptyStrings
- Input: Empty ScanCAD file path
- Expected Output: File Not Found Exception

TestExtractDataWithNoRecipe
- Input: Empty Recipe file path
- Expected Output: File Not Found Exception

View Model Unit Tests:

setFileTestASY
- Input: A .ASY File
- Expected Output: Import Path file path is filled appropriately

setFileTestXML
- Input: A .xml File
- Expected Output: Recipe Path file path is filled appropriately

setFileTestUnsupported
- Input: A non-supported file type
- Expected Output: No file paths are filled out

checkConversionReadyTest
- Input: None
- Expected Output: The conversion process is not ready

checkConversionReadyTestTrue
- Input: .ASY and .xml file paths are filled out
- Expected Output: The conversion process is ready

changeStatusMessageTestReady
- Input: .ASY and .xml file paths are filled out
- Expected Output: The status message is changed to ready

changeStatusMessageTestNotReady
- Input: None
- Expected Output: The status message is changed to not ready

convertFileTest
- Input: An appropriate.ASY and .xml file
- Expected Output: The status message displays a success

convertFileTestFailure
- Input: An invalid.ASY and .xml file
- Expected Output: The status displays an error message

8.3 Test Case Execution Report

The Test Case Execution Report covers the test statuses and what code is covered through the team's testing.

8.3.1 Unit Testing Report

ViewModel Testing Report

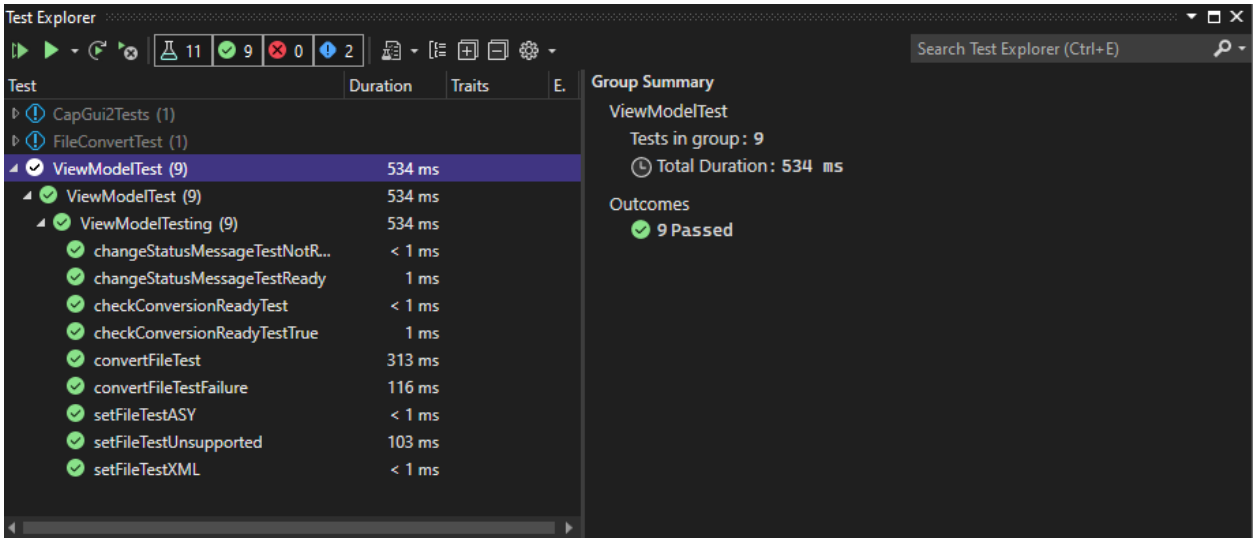

Figure 6

FileConverter Testing Report

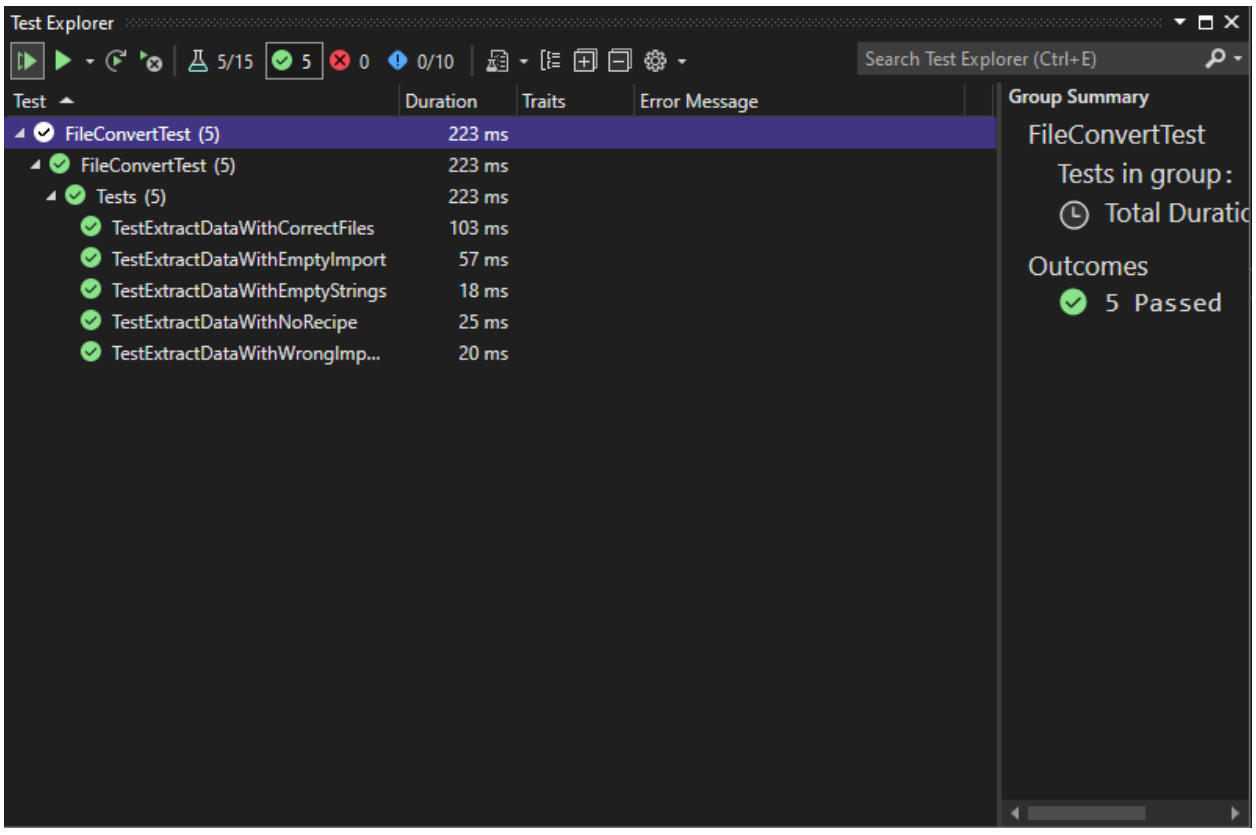

Figure 7

9 Challenges & Open Issues

9.1 Challenges Faced in Requirements Engineering

The largest challenge faced by the team was the lack of a project and the availability of industry mentors. Initially, the industry mentors were unable to respond to the team's email. Trying to solve the issue the team reached out to Dr. Fan in order to get the project moving forward. With help from Dr. Fan, the team was able to contact the industry mentors and begin the setup of the project. In the early weeks after this, the team couldn't accomplish many tasks because the industry mentors were still busy planning the project to be worked on. To fill time and learn what was available, the team had a meeting in person at the Nordson facility and were introduced to the technologies and the possible tasks related to the future. Once the project was created and given to the team, the biggest challenge was the starting point, but due to the help of industry mentors, this challenge is being overcome.

9.2 Challenges Faced in System Development

There are a few challenges we see in the development of the system. The largest workable challenge is going to be the use and development using the XAML system. The team is decently familiar with both C# and Python coding, however, is no experience with the XAML coding practices for creating the GUI. The simplicity and usability of the GUI are some of the most important parts of the finished software system. We have access to the full documentation libraries of the XAML service. We are lacking the ability to use and test the final project with the ScanCAD systems because Nordson was unable to provide us with a laptop to access the software. The largest agile issue was that the team had to wait to be able to start on the actual implementation and coding of the system due to the lack of tools to complete and test the system. Possible test cases would be very polarizing.

9.3 Open Issues & Ideas for Solutions

The issues with XAML are easily overcome, because of the team's combined coding experience. The entire documentation for XAML is available for free on the internet, so this will be used in the team's learning and use of XAML with C#. The biggest unfixed issue is the lack of software tools for the completion of the software system. Due to not having the hardware and software needed for accessing ScanCAD and testing the system, the team is going to have a hard time getting to a point where we are able to say that the system is complete and that it is completely functional and bug-free. Nordson was able to provide the team with a single test case to use once the system is in its final stages. The team will be able to use this singular test case to verify that the system works.

10 System Manuals

10.1 Instructions for System Development

10.1.1 How to set up the development environment

For the development environment, we used Visual Studio 2022 with a WPF project. XAML is based in WPF so with the innate setup of the Visual Studio project, it was very simple and straightforward to get the environment set up. With XAML, there is a lot of built-in support for setting up buttons and text boxes for input and output. This was all for the View of the project.

The ViewModel was created by creating a Windows Forms project in Visual Studio. This was to utilize file explorer logic as well as getting folders from Windows Systems.

To set up the project for future development, the developer simply needs to download the project and open the solution found in CapGui2 using Visual Studio. If there is an error with the dependencies, the developer simply needs to build the FileConverter, and add the .dll file to the dependencies of the ViewModel. From there, a build of the ViewModel is necessary and adding the .dll of the ViewModel to the CapGui2.

10.1.2 Notes on system further extension

The team was tasked with currently accepting only ScanCAD files, however, the system allows for future file types. At this stage the next step would be the implementation of Gerber files, however, future file types can be incorporated in the future.

11 Conclusion

11.1 Achievement

The team has accomplished many things for the System. These achievements include the designing of Use Case diagrams, the creation of User and Functional Requirements as well as others contributing to the design of the system. Aside from this, prior to their development, the team has also conducted successful research detailing the project's functionality and the development of the system itself. The research includes system manuals such as the development environment including any frameworks the system will utilize. As the team moves towards the creation of the system, the team's next steps include coding the backend of the system detailing the backend logic of data extraction, the front end of the system including our User Interface, and

finally System testing which will cover system functionality, creation of builds and any other refinements for the system.

11.2 Lessons Learned

As the team works with Nordson Asymtek, there are many skills and lessons taught by both the Industry Mentor along with our Academic Advisor. These lessons include industry standards, best practices, and many other skills that can be applied within a professional environment. Moving into specifics, the team has recently outlined successfully the system's usage of MVVM and strategy/factory method which helps to contribute to the team's overall understanding of how these design patterns will be integrated into the code.

11.3 Acknowledgment

At this moment, the only contributors to the system and its development include Ali Ahmadinia a professor at CSUSM along with Gregory Butler a supervisor at Nordson Asymtek. Both have contributed to the feedback seen within reports and continue to support the systems development. To be more specific, Ali Ahmadinia through the weekly conducted meetings provides necessary feedback and additional advice for moving the project forward. Gregory Butler throughout the week gives the teams feedback on system design, and system implementation, and helps to manage the project's success.

11.4 Timeline

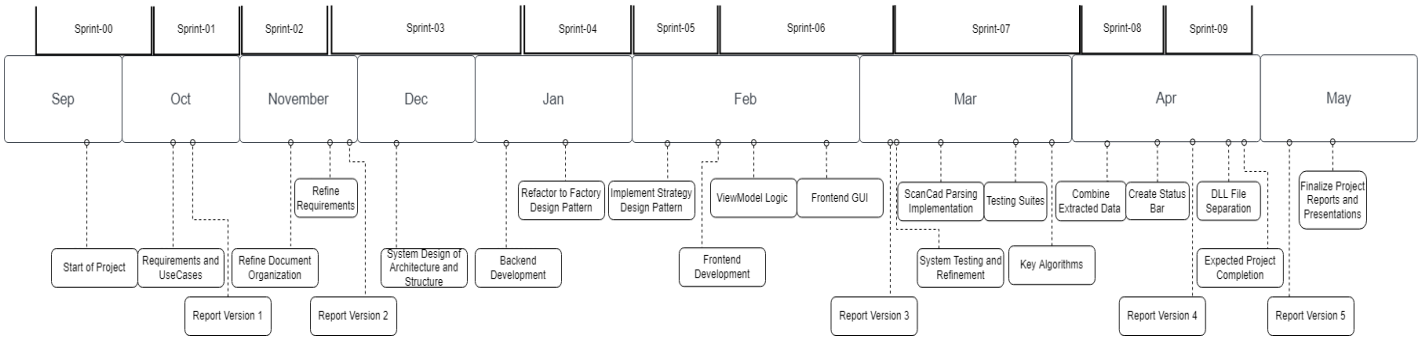

Figure 8

12 References

Adegeo. “XAML Syntax in Detail - WPF .NET Framework.” *WPF .NET Framework | Microsoft Docs*, https://docs.microsoft.com/en-us/dotnet/desktop/wpf/advanced/xaml-syntax-in-detail?view=netframeworkdesktop-4.8.

BillWagner. “Language-Integrated Query (LINQ) (C#).” *Language-Integrated Query (LINQ) (C#) | Microsoft Docs*, https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/.

Factory Design Pattern - Youtube. https://www.youtube.com/watch?v=ub0DXaeV6hA.

“Fluid Dispensing: PCB Conformal Coating: Nordson Asymtek.” *Fluid Dispensing | PCB Conformal Coating | Nordson ASYMTEK*, https://www.nordson.com/en/divisions/asymtek.

“Gitignore Documentation.” *Git*, https://git-scm.com/docs/gitignore.

J-Martens. “Visual Studio Documentation.” *Microsoft Docs*, https://docs.microsoft.com/en-us/visualstudio/windows/?view=vs-2022.

Ncarandini. “Unit Testing C# with MSTest and .Net - .NET.” *.NET | Microsoft Docs*, https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-mstest.

Strategy Design Pattern - Youtube. https://www.youtube.com/watch?v=-NCgRD9-C6o.