John Minney, Riley Peterson, Mohamed Aden
Shannon Daly, Katie Williams, Widmaier Saint-Julien

SAD - Final Submission – G-Code Generator

Team Roles and Responsibilities

Below are the team members, their roles, and the responsibilities that correspond to their roles. Team members may have been given multiple roles and might need to work on responsibilities outside of their role's realm throughout the life cycle of this project.

Team Member Name	Role
Katie Williams	Security Engineer
Mohamed Aden	Business Analyst
Riley Peterson	Project Manager
Shannon Daly	Network Architect
Widmaier Saint-Julien	Web Developer
John Minney	Software Developer & Database Architect

Role	Responsibilities
Project Manager / Coordinator	- Coordinate meetings - Ensure the schedule is kept - Keep track of tasks
Database Architect	- Design the database scheme - Select the correct technology stack
Business Analyst	- Analyze business requirements - Develop technical requirements
Software Developer	- Determine the correct architecture and technology of the system - Write code
Web Developer	- Determine the right technology for web-based projects - Write code
UI/UX Designer	- Design the user experience for the solution
Network Architect	- Design and implement the network requirements
Security Engineer	- Design and implement security requirements for the solution

System Request

Project Sponsor: Owner of Frontier Machining, Gary Davies

Business Need:

- Quickly and easily create tool paths for the CNC Router/Mills to move projects forward by introducing shareable aspects like a tool library and parametric templates.
- Allow a novice user an easy interface to get started on programming, without all the busy UI and complicated options other systems offer – meaningfully reduce the options so anyone can use the system.
- Have access to a G-Code editor on any computer.

Business Requirements:

- Design and develop a good web application to generate g-code from a simple drawing, so then users can quickly generate simple tool paths without needing a heavy, complicated system.
- Implement a simple shape drag-and-drop sketchpad, so then users can generate common paths without needing to go step by step.
- Generate a downloadable file of the generated g-code, so the g-code can be used on physical machines in production.
- Take input from the user to store vital information about the desired cut, so the system can consider any special parameters for the g-code.
- Take into consideration the specific settings based on user input, so the g-code does not cause errors on the physical machine and cause damage.

Business Value:

- Reduce the dependence on expensive software which is overkill for a lot of applications and allow businesses or individuals to have an inexpensive alternative for their simpler parts.
- Allow users to implement easy to use pre-defined templates in order to accelerate their programming, reducing time in the seat and maximizing time at the machine.
- Eliminate the use of advanced and complex systems for simple operations. This will reduce the learning curve for programming CNC's for novice users and allow almost anyone to get started making parts.
- Eliminate the guesswork of settings for the desired bit or materials, have a shareable tool library to keep everyone on the same page and make programming and setup even faster.

Constraints:

- There are a lot of different settings to take into consideration. They can change depending on many factors from the material being cut to the age of the machine.

- The system will only allow the user to do simple drag-and-drop shapes and minor edits to the paths.
- The system will be aimed at beginners and won't be enough for more advanced users or complex parts.

User Stories & Level of Effort:

User Story	Level of Effort
As a user, I would like to sign into the app so then I can store my custom tools and files securely.	5
As a user, I would like to use predefined tools so then I don't have to worry about getting specific details correct.	2
As a user, I would like to create new tools so then I can make paths specific to my machine.	2
As a user, I would like to see a human-readable version of my generated G-Code so I can learn how it works.	5
As a user, I would like to be able to access the application whenever I have internet access so than I can draw paths when I need them.	1
As a user, I would like to be able to download the G-Code file so then I can use it for my Router/Mill.	1
As a user, I would like to save my G-Code file so then I can access it from any computer.	3
As a user, I would like an easy-to-use UI so then I can draw paths quickly and efficiently.	5
As a user, I would like the G-Code generation to consider my tool so then I get accurate cuts.	5
As a user, I would like to see a live view of the paths that I'm creating so then I can make sure my paths align as I expect them to.	5
As a user, I would like to have a few different options of premade paths so then I can use the system for simple projects.	3

Feasibility Analysis

This web application will require a few technical pieces that can be challenging to implement.

The first technical hurdle that we need to pass is allowing the user to draw on the screen. Thanks to frameworks that are already in place and JavaScript it is a relatively painless process to add this functionality. HTML and JavaScript already have processes in place to use the canvas element to find positions on the screen and allow user interaction.

The second technical challenge will involve converting the user drawing to the G-Code. Once again, thanks to the canvas element we can find pixel coordinates on the screen. With the help of a calibration square and constraints, we will be able to use conversion factors to find real-world coordinates of the machine.

Being a web application, it will need to be hosted and managed. Thanks to the extensive list of hosting services available, we will have a wide range of options to choose from.

Economic Feasibility

This web application is designed to increase efficiency and production for CAM shops by making the creation of simple toolpaths easier, less time consuming, intuitive, and creating a viable alternative to industry-standard expensive CAM software packages. That being said, it is not a complete replacement for these more robust systems. It is meant to make simple tasks less time-consuming, but is not meant to replace the complicated ones, as that is beyond our scope. It may also be used as a beginner's app, for those delving into CNC mills and CAM on their own. However, our target consumer is generally people that already have mills and we are just trying to develop a web application to make their lives easier and improve efficiency.

We aim to make this software free and open source. This means that once it leaves our hands, anyone can use our system or take the code for themselves. However, our final version will be hosted online using PythonAnywhere. Because we aim to make the system free to end-users, the costs associated with it occur on the development side of things. Notably, costs come from server hosting, and theoretical labor costs (even though we aren't being paid, I have factored in some lost wages to add some realism to our costs) costs are based on initial development labor costs for year 1, and then server hosting at 5-12 dollars a month on PythonAnywhere. However, development and labor costs drop off after the initial year 0, so from then on, it is just our server hosting costs. We have not discussed the inclusion of ads on our website, which could net us revenue based on traffic. For the end-user, this application has endless ROI as it is meant to be free to them.

	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5	Total
Total Benefits		3,750	3,825	3,902	3,980	4,059	19,515
Total Cost	5968	144	144	144	144	144	749
Net Benefits	-5986	3,894	3,972	4,051	4,132	4,215	14,297
Cumulative Net Cash Flow	-5986	-2,074	1,898	5,949	10,082	14,297	

$$\textbf{ROI} = \frac{(19{,}515-749)}{749} = \frac{18{,}766}{749} = \textbf{25.05\%}$$

Organization Feasibility

When fabrication shops bring on new employees, they will be able to use this application to give them a "crash-course" on G-Code and how it is generated.

The application has a much simpler UI than most G-Code editors so it will be easy for inexperienced users to learn, reducing potential training time.

This could also reduce potential miscommunications between pattern designers and coders due to the graphical interface.

Project Plan

Approach: We have the benefit of time, clear user requirements, use cases, and a project sponsor. Due to this, the waterfall approach is the most feasible and likely the most efficient.

Tasks & Timelines

Task ID	Task Name	Duration	Status
1	Planning Phase	6 Weeks	Closed
1.1	Determine Business Need & Requirements		
1.1.1	Determine Business value and Constraints		
1.1.2	Level of Effort		
1.2	Feasibility Analysis		
1.2.1	Technical Feasibility		
1.2.2	Economic Feasibility		
1.2.21	Cost-Benefit analysis and ROI		
1.2.3	Organizational Feasibility		
1.3	Establish Approach		
1.4	Establish work plan and Identify tasks		
2	Analysis Phase	~6 Weeks	Open
2.1	Establish Functional and Nonfunctional requirements of software		
2.2	Use Case analysis		
2.3	Process modeling		
2.4	Data Modeling		
3	Design Phase	6-10 weeks	Open
3.1	Architecture Design		
3.1.1	Hardware and software specification		
3.2	UX Design		
3.3	Program design		
3.4	Data storage design		
4	Implementation	4 weeks	Open

5	Maintenance	Ongoing	

Interview Q&A

- What are the weak points of current systems?
    - The current programs we have in place to generate tool paths are complex to learn and almost always require a wireframe or solid model to get started. You can't just tell the program you need to mill a line, you need to upload a model, convert it to a wireframe, then select a tool path, isolate it, and tweak it to do exactly what you need.
- How often do you use your current system?
    - Multiple times a day, at least.
- How often do you use your current system for simple paths?
    - To be honest, probably too much. Operations like cutting the back off a part or doing an "OP 0" is super simple yet we still get stuck using the time-consuming software we have, otherwise, we'd be stuck hand-coding a lot of things.
- What do you like about your current system?
    - You can do really complex tool paths and things like 3D milling with it, that conversational doesn't allow. And it has great support, probably because it costs so much.
- What about your current systems makes doing simple tasks annoying?
    - You need a lot of pre-requisites to do something simple. You can't just draw a line and go.
- Does the complex UI in current systems cause struggles for users new to these systems?
    - Completely. At a glance, the program we use (MasterCAM) is really confusing on the surface. There are many menus you need to know and remember to change, or really bad things are prone to happening. Inexperienced users obviously would struggle with something like this because even navigating the menus can be a pain at times.
- How would you use a system such as ours?
    - I would use the system to do simple operations, or even simple parts. There are times where it's just not justified to use something like a complete cam package if the part is super simple and doesn't require complex moves.
- What are the most common used path patterns you use?
    - Contour, Pocket, Face, Bore, Drill, Chamfer, Slotting

- What are some technical and user errors that you run into with your current system?
    - Well, with its inherent complexity it makes the system really prone to some error if the user isn't completely familiar with what they're doing and has some quirks with boundaries when using a solid model. Additionally, with most commercial CAM packages, the cost is pretty high, we have one seat because of this which means only one person can be programming a new toolpath at once, even though we have multiple computers. We could obviously buy more licenses to circumvent that issue, but the cost is extremely high.

Requirements Definition Statement

Project

Create an online hosted program which will allow users to create simple toolpaths for their CNC machine tools which will empower users to create sharable tool libraries, user defined templates and work anywhere there is an internet connection. Users will "stitch" together parametric templates / shapes in order to create a template which the system can generate G-Code from, the universal programming language machine tools read.

Aim

This project aims to create a system that is easier to use and understand than its more complex counterparts. This system will allow users to create paths quickly and easily for both CNC routers and mills. The aim of this system isn't to be a match to the more advanced systems, but to be a placeholder when the complex systems are overkill. The secondary aim of this system is to allow users that are new to programming machine tools have a simpler system in place where they can start from scratch or use our parametric templates in order to get started on their parts and creations.

Functional Requirements

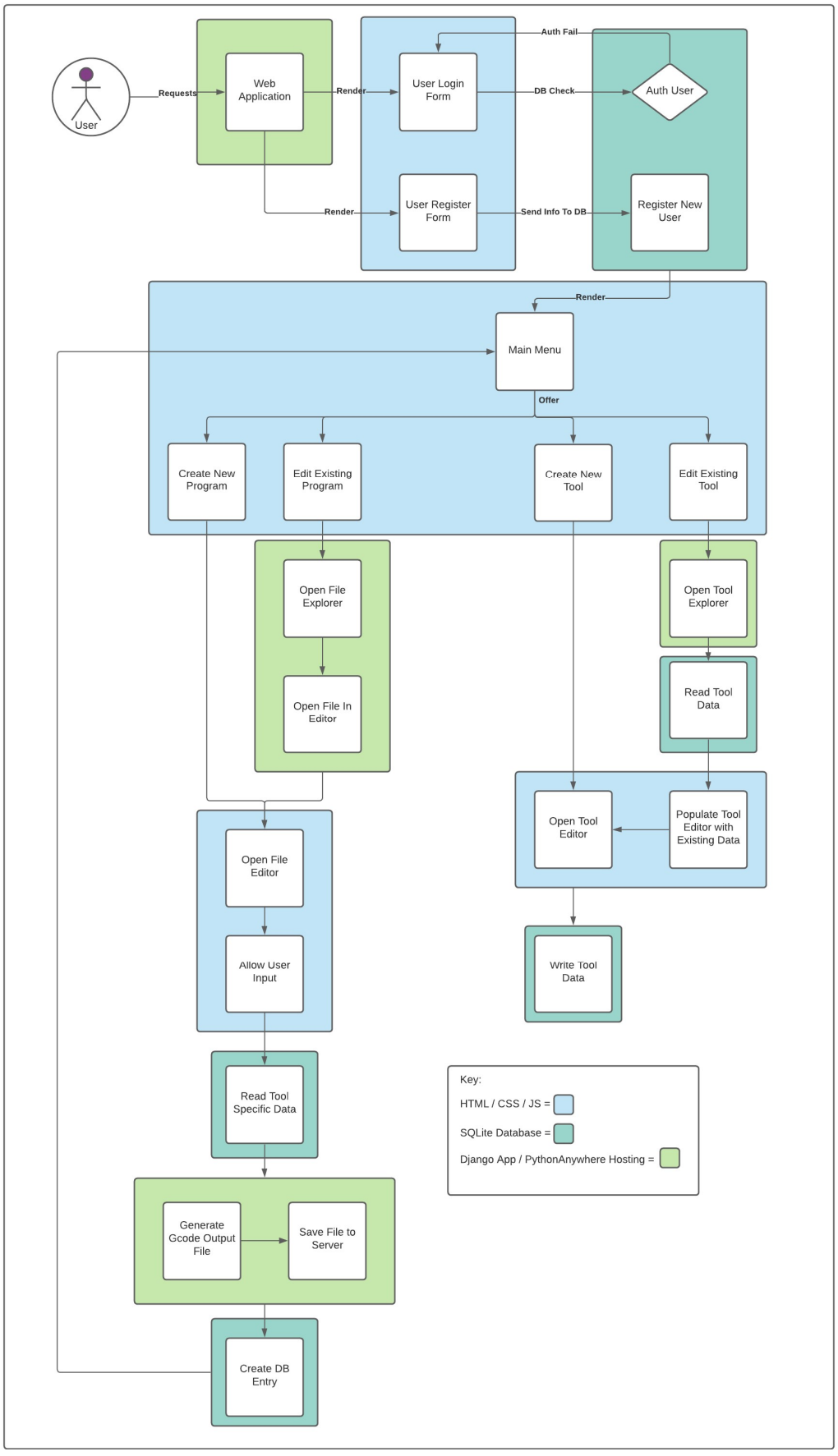
- User Inputs for drawing – The system should allow the user to input positional variables to draw lines and other paths.
- Real-time canvas to show drawn paths – The system should show the user the paths they are drawing in real-time. Each time a new path is added, the canvas should be updated.
- G-code conversion – The system should take the user inputs for tool path constraints and convert them to g-code.

- Human-readable g-code – The system should show the user their created g-code.
- Multiple Path Options – The system should allow the user to choose from the most used tool paths in industry.
- The system should include a sharable tool library so people of the same domain can all have access only to the same standardized tooling.

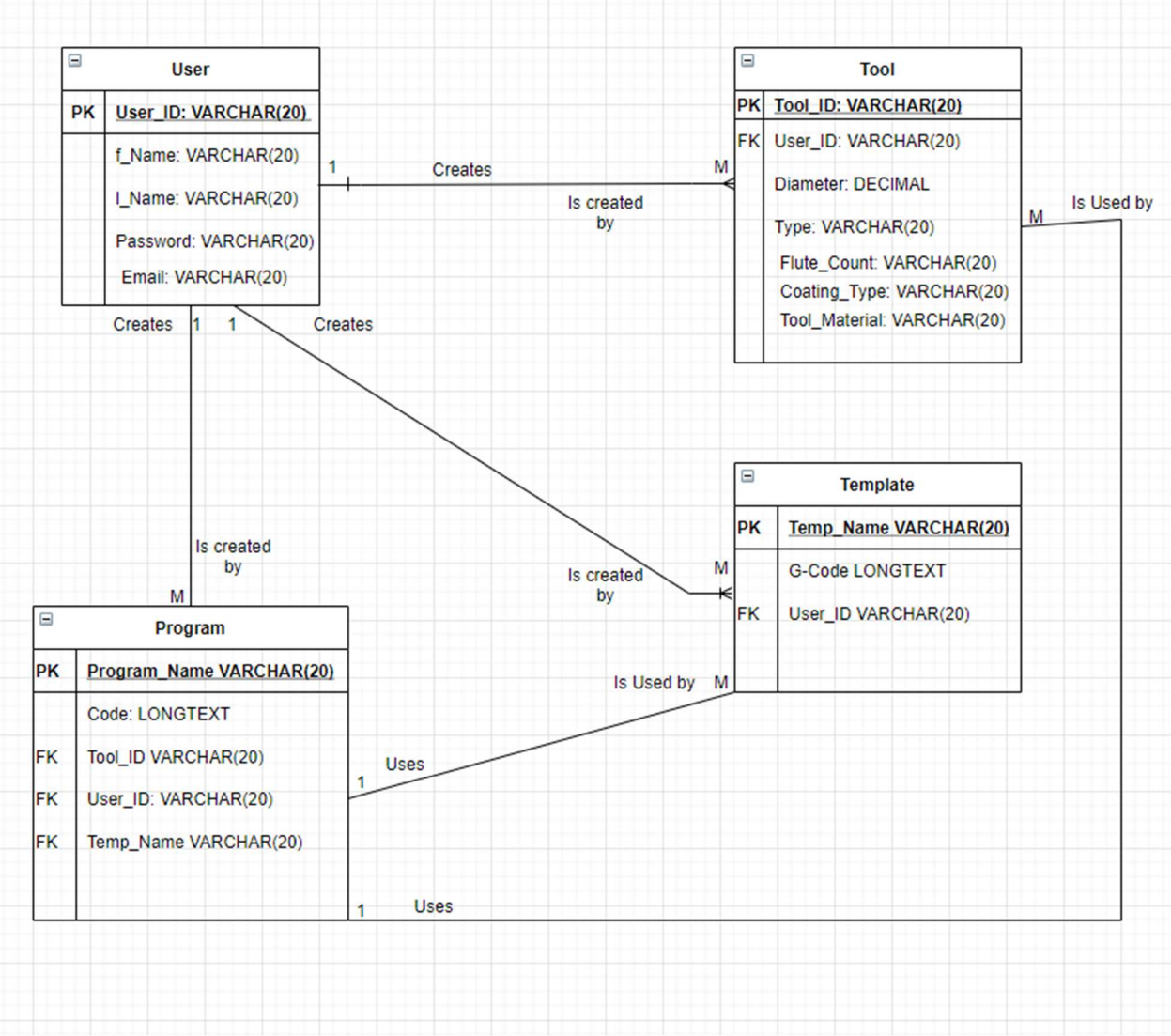
Non-functional Requirements

- Easy interface – The system should have a simple UI.
- Scalability – The system should be built in a way that new paths can be added later as the system matures.
- Open-Source – The system should be open-source and free to all users. The system should also have a public repository that users can open pull requests to add, update, and change the system if they wish.
- Understandable – The system and its instructions should be easy to understand
- A ‘help’ screen should be present to help users when they don’t understand how to do something, or they get lost.
- Quick Conversion – The generation of g-code should be as fast as possible
- Quick Use – The user should be able to use the system quickly for simple projects.

Process Models (on next page)


User
Requests
Web Application
Render
User Login Form
Auth Fail
DB Check
Auth User
Render
User Register Form
Send Info To DB
Register New User
Render
Main Menu
Offer
Create New Program
Edit Existing Program
Create New Tool
Edit Existing Tool
Open File Explorer
Open File In Editor
Open Tool Explorer
Read Tool Data
Open File Editor
Allow User Input
Open Tool Editor
Populate Tool Editor with Existing Data
Write Tool Data
Read Tool Specific Data
Generate Gcode Output File
Save File to Server
Create DB Entry
Key:
HTML / CSS / JS =
SQLite Database =
Django App / PythonAnywhere Hosting =

Data Models



Design Narrative

In modern-day manufacturing, companies all over the world rely on complicated-to-use, expensive, limited user CAM (Computer Aided Manufacturing) software packages to create programs their machine tools can use. Even the most mundane of tasks can take experienced programmers thirty minutes or more. Our goal is to create a free-to-use, open-source solution which makes programming machine tools easy, effective, and efficient.

The solution will be hosted as a web application built from the Django Framework. This will allow the application to be built on a beginner-friendly programming language while still being powerful enough to complete its tasks. Django

also has an extensive Object Relational Model that will make user authentication, tool settings, and CRUD operations quick and easy. Because the solution is going to be open source, using Django will allow us to easily allow users to create pull requests and make additions without having to be super well versed in a more complex programming language or framework.

Because Python is a popular language for open-source projects there are plenty of libraries and packages that we can use to make the process go a little bit smoother and quicker since we will have a shorter time limit. Because Django allows us to use Python and JavaScript together to create the application, we will be able to use HTML and JavaScript's Canvas elements and functions to draw out paths onto the screen.

Django comes pre-packaged with SQLite. Thanks to Django's ORM we will be able to do most of our database operations writing little to no SQL scripts and keeps a layer of abstraction over our data. This will also allow us to use Django's prebuilt authentication system and user management to keep sensitive information away from those that shouldn't have access. This will save us a lot of time and allow us to focus more on path creation and g-code generation without having to re-invent the wheel.

We will keep our project in sync between developers, and the community using GitHub. Not only will this allow us to keep a working production version of the project, but it will also allow the users to open pull requests to make additions and changes to the application. Using GitHub will also allow us to keep markdown files to explain various parts of the system and store some of our documentation so everyone can contribute and view our work.

There are many different hosting platforms we can choose from, but a safe option will be pythonanywhere.com. Python anywhere is built specifically for python web applications and has an easy user interface for those that haven't worked with web hosting before. Because we will allow users to store some of their g-code on the server, we will have to be careful not to exceed the storage limit for their free hosting platform. We might have to implement constraints in the number of files and create a script to "clean" the server of files that have remained inactive for a certain amount of time. PythonAnywhere allows us to do this by creating scheduled tasks.

Technologies

- Frameworks & Libraries
  - Django
  - Bootstrap Style Framework
- Storage
  - Django ORM
  - SQLite
  - Django FileSystemStorage
- Programming Languages
  - Python
  - HTML / CSS
  - JavaScript
  - G-Code

- Software Development Practices
  - Object-Oriented Programming Practices
  - Git Version Control
  - GitHub Version Control Hosting
- Extra Technologies
  - PythonAnywhere Hosting Platform
  - PythonAnywhere Scheduled Tasks
  - Visual Studio Code IDE

Source Control

GitHub Repo Link: https://github.com/john-minney-iii/GCode-Senior-Project

Work Assignments

Since Riley Peterson has the most experience with this type of software and is our project manager. He will help us delegate the work. To keep track of tasks we can use the issues functionality that is already implemented into GitHub, though these are supposed to be for bugs they work really well for tasks.

Since we are going to use the issues, this will allow us to link the task directly to our GitHub account so we can keep a TODO list and update everyone on progress using the comments.

Testing

The Django framework comes prepackaged with a unit testing library. This will give us the ability to test CRUD operations easily. We will also test the system by hand as we develop it.

Documentation

Each developer will have responsibilities for the documentation. We might assign basic documentation to the project manager, but the developer that writes the code will have the easiest time writing the documentation for a specific feature.

The system documentation will be written at the end of the project once we know that we have implemented everything that is needed. Once we get to this point we will all sit down together and make sure the system is well documented and we don’t miss anything.