i281 Processor Simulation and Visualization Software

Design Document

Team Number: sd-may21-38
Client: Alexander Stoytchev
Advisers: Alexander Stoytchev

Team Members
Aiman Priester - Team Facilitator
Eric Marcanio - Meeting Scribe
Colby McKinley - Team Report Manager
Brady Kolosik - Documentation Manager
Bryce Snell - Chief Design Engineer
Jacob Betsworth - Test Engineer

sdmay21-38@iastate.edu
https://sdmay21-38.sd.ece.iastate.edu/

Revised: 10/2/2020 (Ver 1.0)

Executive Summary

Development Standards & Practices Used

Extreme Programming, an Agile based methodology which better involves client feedback.

ECMAScript standard, HTTP protocol

Summary of Requirements

- Create a lightweight Javascript Web client which simulates the i281 Processor
- Develop an assembler to output machine code to the CPU
- Create a Verilog model of the i281 processor

Applicable Courses from Iowa State University Curriculum

- CPR E 281: Digital Logic
  - Understanding of logic gates
- CPR E 381: Computer Organization and Assembly Level Programming
  - Understanding of processors and assembly language
- CPR E 581: Computer Systems Architecture
  - Understanding of processor simulators
- S E 319: Construction of User Interfaces
  - Understanding of front end technologies and implementation
  - Miniature Javascript project
- S E 329: Software Project Management
  - Understanding of creating software project proposals and the study of software development in industry
- COM S 309: Software Development Practices
  - Collaboration on a semester long software project
  - Understanding of front end technologies and implementation
- MATH 201: Introduction to Proofs
  - Aids understanding of the correctness of the program, beyond enumeration of all cases
- MATH 317: Theory of Linear Algebra
  - Understanding vectors and matrices, important to binary operations and the implementation of the register file and ALU.
- MATH 314: Graph Theory

    - Understanding pathways through a systems (enhances testing by creating meaningful and concise tests)

New Skills/Knowledge acquired that was not taught in courses

- Design and Implementation of practical graphical user interfaces and libraries
- Developing large pure Javascript projects
- Fundamentals of Compiling and Assemblers

Table of Contents

List of figures/tables/symbols/definitions

As of the current version of the document,

1 Introduction

1.1 Acknowledgement

We would like to state our appreciation towards Dr. Alexander Stoytchev for guiding us through this project. Dr. Stoytchev has committed himself to lending a helping hand and advising us along the way. We would also like to extend our appreciation towards the Electrical and Computer Engineering department for allowing us to perform hands-on work to further our knowledge in this subject area. We are certain that with the knowledge gained from this project, we are able to advance our careers and become innovators in our respective fields.

1.2 Problem and Project Statement

This project is being created to help students learn how a CPU works in CPRE 281, Iowa State's Digital Logic course. The course builds from boolean algebra into designing complex circuits. Since the course is often a student's first course in the Computer Engineering curriculum, it is challenging to understand at first. At the end of the semester the course introduces assembly code and the fundamentals of a processor, which are very abstract concepts to understand. This is why we are creating a simulation to explain the concept in an in depth and responsive manner.

The project is a web app CPU simulation. The user will be able to input assembly language and have it compile on the mock CPU. Each component of the CPU will have functionality and show the user the information being stored inside. Each line of assembly will propagate through the CPU with colored and detailed lines.

Through this project we want to accomplish a working understandable CPU. The main goal is to have the CPU be easy to understand and also simple enough that any user can learn from the app as they work through it.

1.3 Operational Environment

Our project will be used in the classroom setting as a teaching tool. Thus there are some considerations that must be taken into account.

- Ease of use. Our client wants this to be simple for him and students to use.
- Clarity. This should clearly show how the processor functions.
- Responsiveness. A slow application does not benefit our client or his students.

1.4 Requirements

Our project needs the following elements:

- A dynamic model of the i281 processor.
- An assembler using the i281 ISA.
- A Verilog version of the i281 processor.

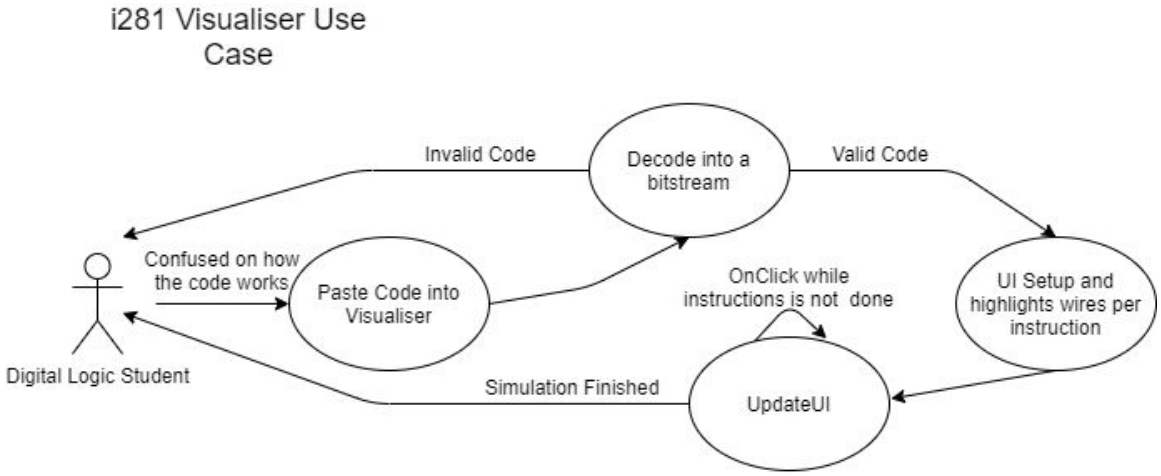
More specifically we are trying to accomplish the following:

- Run the application on a website.
- Show inner workings of processor components.

- Show the conversion from assembly to machine code.
- Align the graphical modules of the processor to Verilog.

1.5 Intended Users and Uses

This project will be used by Dr. Stoytchev as a teaching tool in his lectures in CPRE 281. It could also be used by students in that class to dive deeper into architecture.



1.6 Assumptions and Limitations

Assumptions

- Can only be used if there is an internet connection
- Students have a preliminary understanding of the i281 CPU
- Signal lines will be colored to show the dataflow of the bits
- The visualiser should somewhat scale to the screen size of the user
- User will be able to either type in the program or upload a text file to run programs

Limitations

- End product will not include timing analysis for the CPU
- Cost to produce the project should be close to zero
- Visualiser must be lightweight to be able to run on a browser

1.7 Expected End Product and Deliverables

There shall be three deliverables by the end of the work period. Those are, the i281 Visualiser, Verilog conversion of the i281 and an Assembler for the i281.

HTML and Javascript code that when run on a website will show the different elements of the CPU architecture as well as the data that will run through them. There will also be elements for user input, such as switches, that will control input bits and allow for the user to input a text file of assembly code to simulate. The i281 Visualiser will be hosted by ISU on the clients' website. The Visualiser will be kept there for use in the Fall of 2021 in CPRE281. Students shall be able to access it at any time. Non-students, inclusive students that are not from ISU are also permitted to use the Visualiser to enhance their Digital Logic learning.

A Verilog version of the i281 CPU that will simulate the specifics of the i281 CPU design. It will be debugged using an FPGA to make sure it is as similar in functionality as possible to the CPU. The Verilog version of the i281 shall also be available on the clients' website. This is also to be used in the Fall of 2021. Students will not necessarily need this for the class, however, the client has expressed preference towards having it in case of a class overhaul.

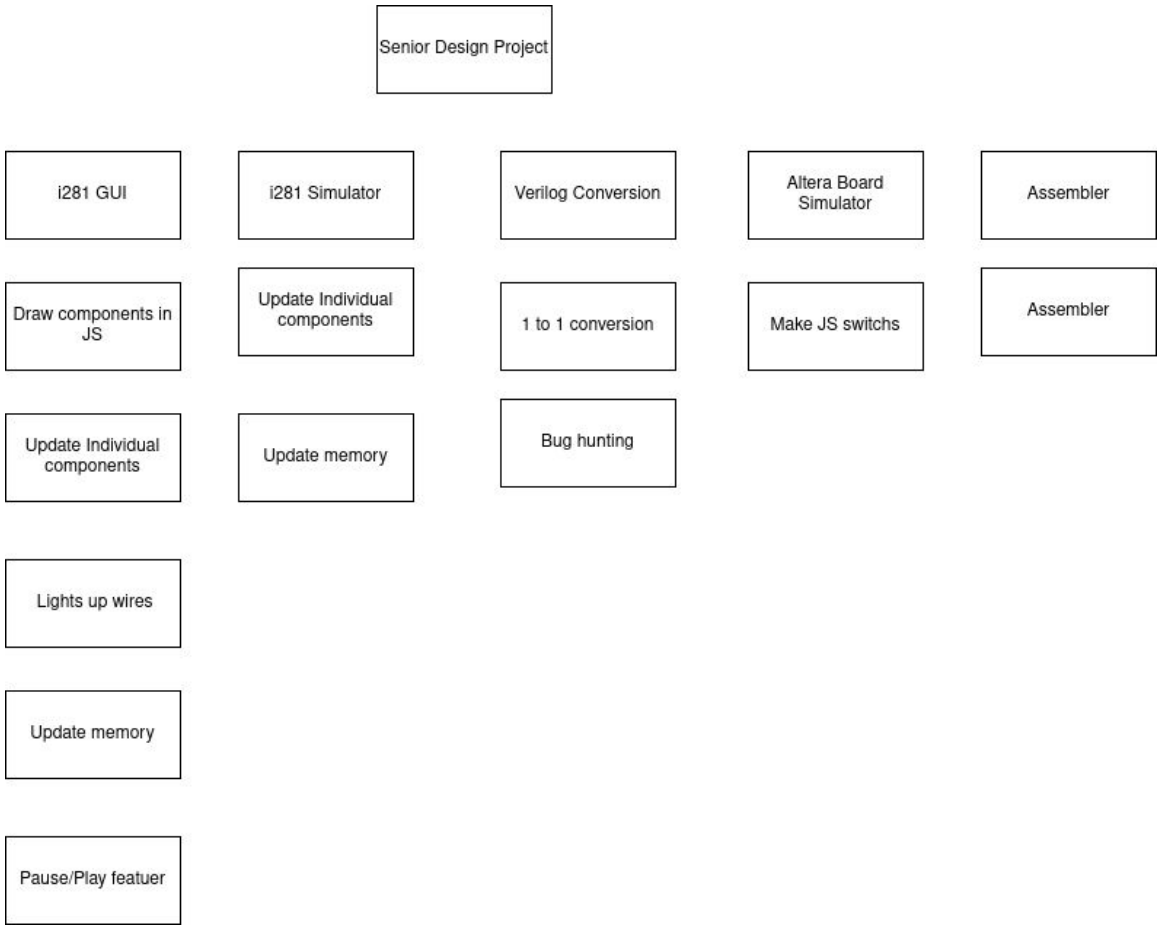
Lastly, the final deliverable is the Assembler. The Assembler will allow students to confirm their programs are exactly what they expect to write. This should allow for error checking and shall be available on the clients' website.

All three deliverables are expected to be ready by the end of the work term in May 2021.

3 Project Plan

2.1 Task Decomposition

- Assembler in Javascript to output machine code
- i281 Visualiser
    - Learn and compile information about drawing in Javascript
    - Learn to change individual components within the drawn CPU
    - Each wire lights up when data is being transferred through it
    - Memory is updated when it is being ran
    - Pause and play feature to understand what is happening
- Conversion to Verilog
    - 1 to 1 conversion from current version of the i281
    - Bug hunting and patching



2.2 Risks And Risk Management/Mitigation

Developing verilog version of i281 CPU. The FPGA board may overheat and subsequently start a fire. A comparable event is unlikely to happen (0.01). This task cannot be eliminated since it is deliverable in the final product. We can buy a fan for additional cooling support.

When developing the software for other parts there may be data loss in the project. This is a very high risk (.9) and we are going to use Gitlab to minimize the loss of data in the project.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Key milestones:

- Completion of Verilog i281 CPU
- JS assembler outputting files
- JS assembler with acceptable UI
- Visualizer UI completed
- Completed i281 CPU similar
- Ability to play pong

2.4 Project Timeline/Schedule

As it is still very early in the project, the provided gantt chart offers the goals that have been set for this semester and the next as well. As previously stated, our end goal is to have all three deliverables completed by the end of the spring semester in 2021, and that is reflected in the spring 2020 semester area of the chart below.

In order to successfully complete the task at hand, the group has divided up tasks to achieve parallelism in the development of the project, this semester will be spent further solidifying the division of work, and we have already split up the work as you can see in the chart below.

The fall semester will be spent completing the assembler such that a student will be able to translate from assembly to machine code, with the front end developers mainly exploring javascript and the various frameworks that can be used to draw, and animate the simulation that is required by the deliverable defined as “i281 visualizer”.

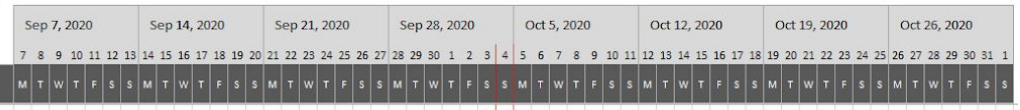
The spring semester will likely be a continuation of the fall semester, with some tasks having a lot of carry-over. At that point, the documentation will be out of the way leaving the team to fully focus on the development of the project. It’s expected that around half-way or so through the semester a semi-working prototype will be developed so that the phase of debugging and polishing can begin.

Note that this is also a rough schedule, and hard deadlines have the potential to be moved around.

i281 CPU Visualizer

Project Start: Thu, 9/3/2020

Display Week: 2



TASK	ASSIGNED TO	PROGRESS	START	END
Fall 2020 Semester				
JS Assembler	Eric	20%	9/3/20	11/24/20
JS Drawing	Brady, Colby, Jacob	20%	9/3/20	11/19/20
JS Architecture	Bryce	0%	9/3/20	11/24/20
JS Simulation	Brady, Colby, Jacob	0%	n/a	n/a
VHDL Conversion	Aiman	20%	9/3/20	11/24/20
Spring 2020 Semester				
JS Simulation	Brady, Colby, Jacob, Eric, Bryce	0%	1/18/21	5/6/21
VHDL Conversion	Aiman	0%	1/18/21	5/6/21
VHDL Bug hunting	Aiman (others if needed)	0%	1/18/21	5/6/21
UI	Jacob	0%	1/18/21	5/6/21
Simulation debugg	all	0%	1/18/21	5/6/21

2.5 Project Tracking Procedures

We are using multiple tools to keep our project moving forward. First, we are using Discord for inner team communications and discussions. We are keeping a list of weekly minutes for the discussion with Dr. Stoychev on our senior design website. Additionally we are using Gitlab as our code repository. Gitlab also allows us to create and organize tasks for our sprints.

2.6 Personnel Effort Requirements

Name	Task	Description	Reference
Eric Marcanio	Javascript assembler	Create an assembler that will take in i281 Assembly language and output machine language to the CPU simulator	https://simple.wikipedia.org/wiki/Assembler
Brady Kolosik	i281 Visualizer	Explore frameworks and methods to create a visualized version of the i281 CPU executing assembly code written for its architecture	
Colby	i281 Visualizer	Create a visualization of a simulated CPU while executing arbitrary assembly code.	https://developer.mozilla.org/en-US/docs/Web/JavaScript
Jacob Betsworth	User Interface of the Website	Creating the elements the user will interact with for input and to see output.	
Bryce Snell	i281 Visualizer	Develop a tool to simulate the i281 processor's output in a webpage	
Aiman Priester	Verilog Conversion	Converting each module into Verilog with the intention of squashing bugs in the i281 design	https://www.ece.iastate.edu/~alexs/classes/2020_Fall_281/

Table 2.6.1: Personnel Effort Requirements

2.7 Other Resource Requirements

Most of the project is a web application so we will not need any physical resources aside from a server to host on.

2.8 Financial Requirements

Funding for the FPGA Board was provided by the team member who required its use. ISU was not able to provide funding for this due to unspecified purchasing rules by ISU Purchasing Office.

3 Design

3.1 Previous Work And Literature

Initially the framework to this project was created by Dr. Stoytchev and a former student, Kyung-Tae Kim. The completed processor was implemented with basic logic gates from the ground up. This was to ensure that the Digital Logic students could understand the underlying design of a processor.

Since this processor was made in-house at ISU, this processor is unique. However, it similarly models the MIPS processor that is introduced in CprE 381. However, introducing the MIPS processor to CprE281 students will be overwhelming. Hence the i281 processor was born.

To be specific to this project, the team did find a project that has been done with a similar idea. This external project was a JQuery assembler like project. We ended up using this to validate the achievability of creating a minimalistic Javascript project.

3.2 Design Thinking

The initial project objective was to create a CPU simulator for students to interact with. The motivation for the project is to help Digital Logic students grasp a high level understanding of computer organization and the motivation behind the class. Due to the many moving parts of a CPU, the normal power point slides would not be sufficient material. In hopes of having a long product lifespan, we chose to develop a Web application.

3.3 Proposed Design

There were several approaches to the solution we researched, using a particular framework (AngularJS or React), the use of leveraging languages together (Javascript and HTML, integrating Web Assembly C/C++), and several graphical design approaches. Based on parliamentary discussions with our advisor, we are creating a minimalistic Javascript application. There are no frameworks integrated, and will only be integrating HTML alongside Javascript.

So far we have created a NodeJS application which can properly send back HTML code with any tags. There are no third party libraries in use in the project; this may change depending on the graphics route. The project currently correctly simulates a functioning ALU and has basic graphics to go along with it. The project will eventually have each component being a separate class.

After several iterations of development, we have successfully created an SVG image which can be manipulated by Javascript interactions. We have created all major components in our display of the i281 processor and are able to dynamically interact with them

.

Additionally we have started trying to create code which simulates a CPU. The majority of the processor components have been made and a full simulator has been started. Since Javascript is a weakly typed language, we will be using arrays to do bit manipulations. There are ongoing discussions on how to handle Endianness and the differences between hardware representations and software representations.

The assembler has a simple interface. It can take a given use file, either text input or file upload, and completes the first of through iterations of completing. It is successfully compiling some assembly files.

The project will adhere to ECMAScript standards. It is meant to ensure the interoperability of Web pages across different Web browsers, and is common industry practice to use in NodeJS projects.

3.4 Technology Considerations

There are two main programming used in web development today, PHP and Javascript. Historically PHP was more common than in today's web market, however has been seen increasing popularity due to new changes. Javascript is currently more popular in web development, and has a vast amount of third party frameworks and libraries. Since PHP is more oriented towards server side development rather than Javascript, which is used more for client side development, we choose to go with Javascript.

We choose NodeJS since it uses non-blocking, event-driven I/O to remain lightweight and efficient real-time applications that run across distributed devices. NodeJS is based on the open web stack (HTML, CSS and JS), which perfectly aligns with our other design choices. It is difficult to compare NodeJS to another technology in its classification, however a comparable technology is ASP.Net. ASP.Net provides support for web applications, complex APIs and recently microservices. In our project most of these support features would go used.

3.5 Design Analysis

This design has a large amount of potential. We are using very common technologies for web applications that perform similar functions. Javascript seems well suited for the task of controlling dynamic web content. Our client also seems happy with this approach

Our current minimalistic approach to the project adds overhead into the implementation of the ideas. Since we have no framework to handle common actions, such as serving html pages, it is left to use to build the underlying infrastructure. There is little help offered online beyond library documentation and intuitive debugging. Building a runnable project has been time consuming and frustrating at points, however it hopefully increases our understanding of Javascript and will help us troubleshoot bugs later on in development.

The minimalist approach aligns with some current industry professionals opinions on the future of web development.

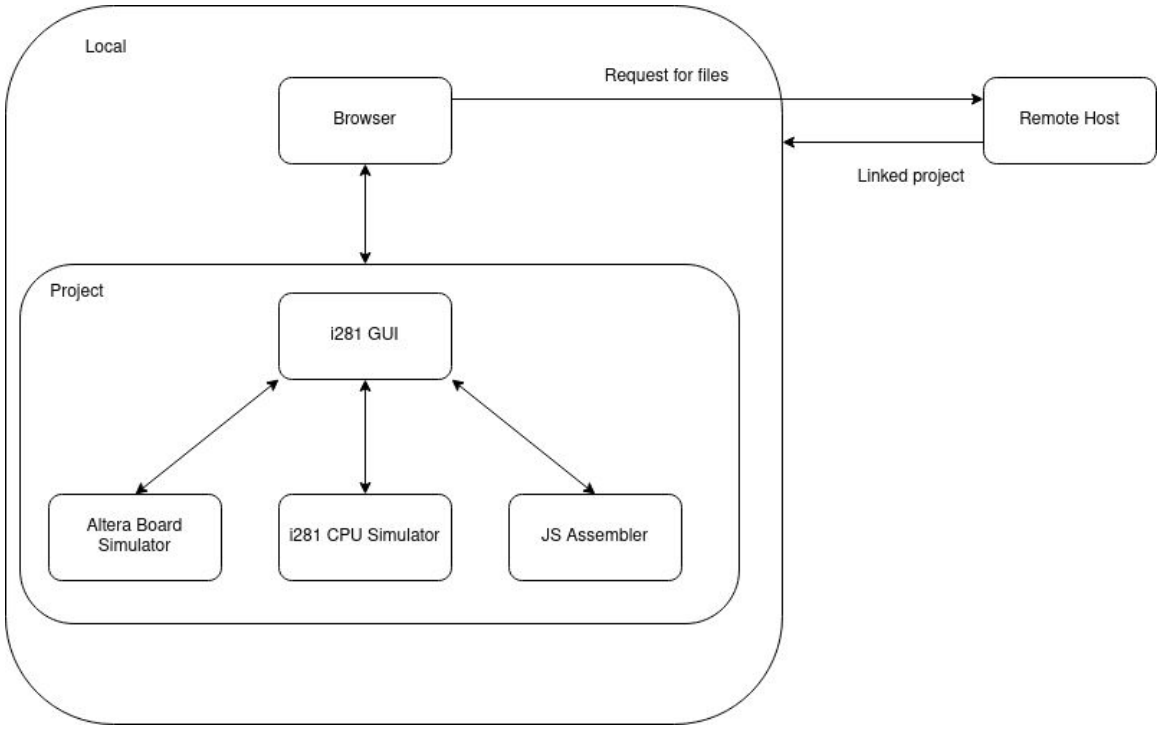
3.6 Development Process

We are developing this project with a modified version of Agile, called extreme programming. Extreme uses the core ideas from Agile but places extreme emphasis on client feedback.

3.7 Design Plan

The user will make a request to a remote host for the project via their browser. The browser will be the interface between the user and the project. Within the project the i281 GUI will interface with each of the modules.

Modules: i281 GUI, i281 CPU simulator, Altera board simulator, JS assembler



4 Testing

Verilog Conversion

The modules created on a per component basis must be rigorously tested. In the early stages, testbenches via ModelSim will be made to verify the designs of the components. However, later on in the development, hardware testing must be performed to ensure that the conversion is in fact accurate. This is a concern due to the variability of hardware performance.

i281 Visualiser

In large part testing a GUI is simplistic. We need only to check the elements we created on the document i.e. checking if the fetched id returns null or not. A further step is the check to see if the desired listeners on these objects are defined. In testing the wrapper classes we will have more traditional unit tests to verify the functionality. Currently in industry testing the GUI is often handwaved due to the scaling complexity in verification and ease of verification. That is trusting the object is created correctly if it is created. This also falls into the testing philosophy of not testing code that is not yours.

i281 Simulator

The simulator testing is likely to be the most straightforward. We understand how the processor should work (through both the diagrams provided as well as the work done in the Verilog Conversion), we simply need to test that the simulated portions match the expected results. Pieces will be tested individually, then as larger functional units.

4.1 Unit Testing

We are using a common Javascript testing framework, Jest. We have created some simple test cases to demonstrate how a test file is created. The tests are unit tests for the CPU simulator.

4.2 Interface Testing

Interfaces will be tested with drivers that will be developed alongside the modules. Some of the needed drivers are:

- Memory driver to test memory and register units
- ALU driver
- Control driver
- OpCode driver

These drivers will send commands through the final interface between modules and check for proper responses from the output interface. This will allow us to test edge cases and confirm that the final control works as expected.

Small, incomplete versions of these unit tests have been started for testing the simulator logic. These will likely adapt and grow over time as we focus more on verification of our project.

4.3 Acceptance Testing

Our goal is to include our client after each cycle of iteration. This will give him a good view of progress and allow him to make changes as he sees fit. We plan to show that our tool meets his requirements by doing live demonstrations as well as reports of unit testing. This will show him the overall functionality as well as confirm the quality of the underlying code.

4.4 Results

Currently we have developed some small snippets of code to test languages and frameworks. Since this testing phase was intentionally open minded and less structured (to allow exploration and experimentation) we did not develop any formalized tests for the online tools.

One of our goals for the next phase of development is to develop a testing procedure and tests for it. Specifically, we will be implementing unit testing and hopefully a pipeline of known good tests to prevent project regression.

5 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3.

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

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software

manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.