Circuit Drawing and Simulation Website

DESIGN DOCUMENT

Team Number: SDMAY19-44
Client: Professor Andrew Bolstad
Adviser: Professor Andrew Bolstad

Team Members/Roles:
Joseph Veal: Back-End Code Leader
Alexandra Sutton: Meeting Facilitator & Scribe
Lucas Maring: Report Manager &
Keegan McCarthy: Team Leader
Cassandra Plata: Front-End Code Leader
Tyler Schurk: Back-End Code Leader

Team Email: sdmay19-44@iastate.edu
Team Website: https://sdmay19-44.sd.ece.iastate.edu

Revised: December 2nd, 2018 / Version: 2.0

Table of Contents

List of Figures

List of Tables

List of Definitions

CSS: Cascading Style Sheets; a style language used for changing the style of HTML files

Django: a framework within Python

GUI: Graphical User Interface; the visual software component that users view

HTML: Hypertext Markup Language; is the standard markup language for creating web pages and applications

PNG: Portable Network Graphics; the file format that the circuit schematics will be saved in

Python: an interpreted language that is nice for scripting because it is an open source language that has a wide range of modules for developers to pick from. Python can be used as a server-side language as well; and because it interacts with servers so well, it can be used to manage the backend of a web dev application

1 Introduction

1.1 Acknowledgement

We would like to thank our client, Professor Andrew Bolstad, for providing us with this project. We all hope to create a website application that will be an educational tool for students and develop our own programming skills in doing so.

1.2 Problem and Project Statement

Professor Andrew Bolstad teaches Electrical Engineering courses for Mechanical Engineers; however, he is unhappy with his lecture notes concerning circuit diagrams. He is currently using PowerPoint to create circuit diagrams for his lectures, but he found that it is too time consuming and difficult to create the circuits that he wants. Therefore, Professor Bolstad is looking for a website that is intuitive and easy to use so he will be able to make visually appealing circuit diagrams for his class lectures. He wants to save these circuit diagrams for later use, efficiently create any circuit he needs in a timely manner, and possibly simulate the circuits for educational practices.

Although there are currently circuit drawing websites available that Professor Bolstad can use, no one website has everything he needs. Therefore, we will create a website application that will do just that – satisfy every need of Professor Bolstad and his students. We will create visually appealing circuit drawings using Python and JavaScript in order to create a website that can be updated and altered to tailor Professor Bolstad's needs. This website will include easy-to-use rotate and delete features, it will have a "components" tab that will be an educational tool for the user to learn about various components, it will have a "help" tab that explains the program, and the user will have the ability to move the circuit around while maintaining wired connections.

This website application will not only be a tool for students inside Professor Bolstad's class to use, but it will be a tool for any students willing to learn about circuit theory. We hope to create both an efficient circuit drawing website and an educational resource that will teach students the fundamentals of circuit theory and application.

1.3 Operational Environment

The operating environment for this project is the server/database that we plan to use. This database will securely store all of the data needed for this project, and so it will not be exposed to harsh conditions or security threats.

Furthermore, this website will primarily be used by our client, Professor Bolstad, but it will also have the potential to be used by any students either inside or outside of the class who are interested in learning more about circuit design and simulation. Any outside user will not be able to alter any of the code associated with the project, and so the user will not be able to harm the website in any way.

1.4 Intended Users and Uses

There are many possible uses for this project. The primary function of the project is to create model circuits and easily printable schematics. There are many reasons an engineer would use the tool that will be created. The main use case the team took into consideration when designing the tool was the need for elegant schematics to be used in educational presentations. The primary use of the tool is to create a schematic that can be easily integrated into PowerPoint. However, the tool has other potential uses as well, such as circuit modeling and potentially simulation. It will have all of the basic components in circuit design. The uses cases, drawing, and simulation, allow the tool to be used for many engineering applications.

This project was created with educators in mind. However, there are many possible uses for this project. Students can use this tool to model and test circuits. Field engineers can save time by making simulations using our tool. Anyone interested in simulating a circuit without designing it is a potential user of this tool. The tool has such a wide range of users because it will be web-based and available for everyone.

1.5 Assumptions and Limitations

Assumptions:

- Django and Python will be relatively easy to learn and use, making for a simplistic end user result
- Server space will be provided by ISU ECpE for user login
- Users should be able to access the web app from anywhere (not just ISU students)

Limitations:

- Learning a brand new coding language for each group member, slows down completion
- The ability of the group implementing features that can be used in the simulation app like the wires staying connected
- The ability to complete each task within the timeframes set out

Note: We will discover more limitations as the project moves along, and have more assumptions as well

1.6 Expected End Product and Deliverables

The expected end product and other deliverables will be the following:

Prototype I Completion Date: December 15th, 2018

Prototype I is intended to be a test to show how things will be displayed and overall functionality. Prototype I will consist of a functional GUI with the following capabilities.

- Drag and Drop component features
- Functional website layout (HTML/CSS side of the project)
- Front End displayable GUI (Just on the computer not on Website yet)

Prototype II Completion Date: February 16th, 2019

Prototype II will be an add-on to the first prototype. Prototype I will be a finished final product for the circuit drawing. Prototype I will include all functionality except the circuit simulation. The following capabilities will be added on after prototype I:

- Running a basic website for testing purposes
- Easy to use circuit drawing
- Picture Exporting into a PNG file
- Hotkeys
- Wires move with components

Final Product Completion Date: April 20th, 2019

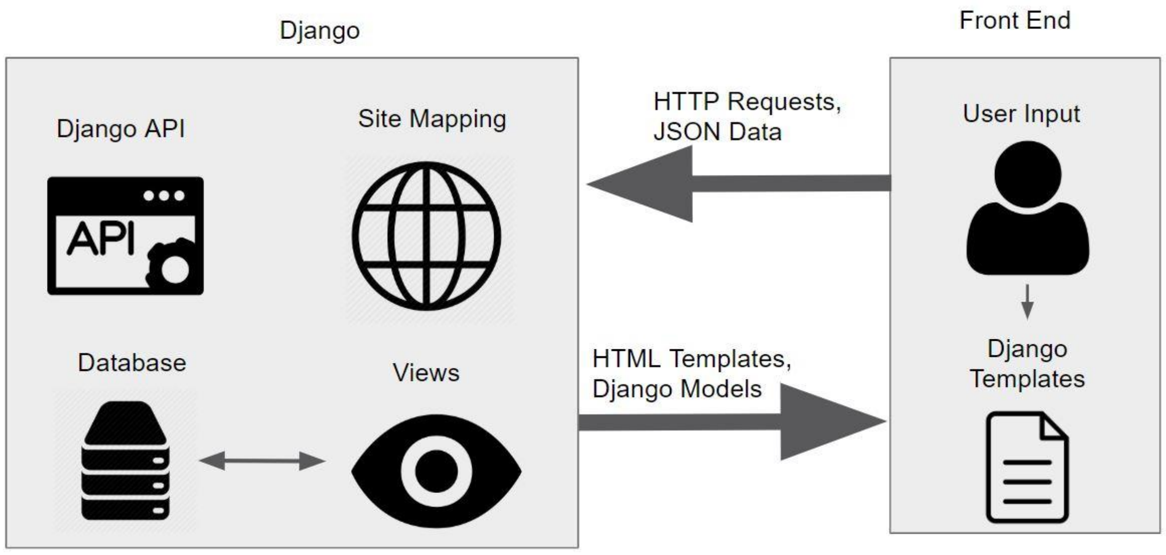
The final product will include an immersive circuit drawing website with basic simulation capabilities as a stretch goal. It will run seamlessly with the following:

- User circuit saving interface
- Pre-drawn basic circuits
- Tutorial interface

2. Specifications and Analysis

2.1 Proposed Design

The figure below reveals the process flow of the back-end working with the front-end of the project:



3 Figure 1: Process Flow Diagram

The project consists of implementing two major components: the ability to drag and drop circuit components to form a nice-looking picture and, later on, the ability to simulate the circuits that have been drawn. Implementing these two components will consist of a backend structure that will be managed by the framework Django, which will allow us to use Python. Python and Django will give us a test server to run our application on so that we can test the feedback given to the server and compare it with what we expect to see on the website. The front-end will be in HTML, CSS, and JavaScript, allowing us to design our web application in a basic but user-friendly manner.

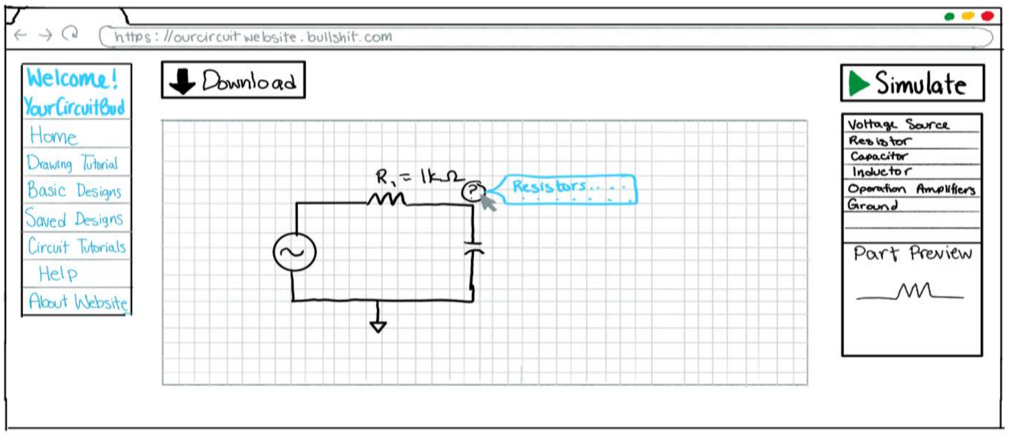


Figure 2: Drawn Prototype of Web Application

The figure above is a drawn prototype of what we expect our web application to look like. The left column will consist of buttons and tabs consisting of tutorials, examples, and a help tab. In the middle is the "drawing grid" which is where all of the components will be drag and dropped into to create the picture our client needs. Whenever a component is dropped into the grid, there will be an option to give that component a name and a value like shown above. A possible goal we also may include will be the question mark next to the component in the grid, giving information on that component. On top of the drawing grid is the download button that allows the user to save the image of the drawing space to wherever the user requests. Lastly, we have the simulation button and the components on the right. The simulation will be a stretch goal towards the end of the second semester where it will simulate the circuit and give output values the user may want. The components will be listed on the right in a fashion to be decided later. A preview of the component selected will be shown as it is above.

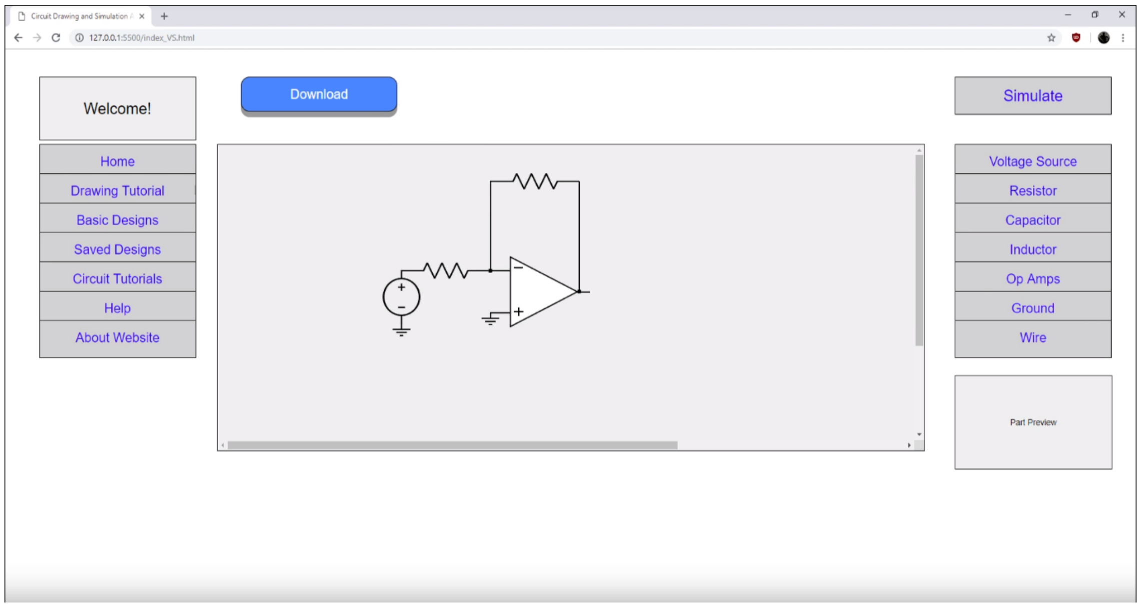


Figure 3: Prototype of Web Application

Currently, we have completed what is shown above in Figure 3. The HTML side of the project has been completed including laying out the buttons, tabs, and drawing space. The JavaScript code is currently being written and we are working out the kinks of our drag and drop feature, which has been implemented with either exclusively dragging or exclusively dropping features. At this point, we have not gotten both features to work at the same time yet. We also have a Python/Django server up and running to test our prototype of the application for drawing.

We, as a team, are accomplishing these tasks by splitting up into three smaller teams of two people. These smaller teams tackle separate parts of the project so we can complete more tasks in a shorter period of time. The current teams include the HTML and CSS side of the web application, the Python and Django local server end, and the final team working on the JavaScript function to complete the integration of everything.

2.2 Design Analysis

Due to our limited experience with the creation of web applications, our initial work had consisted of research and brainstorming. One of our first tasks was to take a look at competitors, both paid and free, to see what the positive and negative features were of each. We analyzed the applications that we have had to use inside and outside of our classes such as PSPICE, Multisim, and Falstad. PSpice is a tool used for the simulation of circuits while Multisim functions as both a drawing and simulation tool. These are heavy, expensive tools that are quite a bit beyond our end-goal for the drawing tool but they offered insight into how accessible we should make our website. Falstad is a free web application that some of our teammates have used in the past that functions similarly to Multisim. Its ease of use was an important takeaway; however, the circuit drawings are not visually appealing to place into documents.

Step two in moving forward with our project had us decide what programming language we wanted to use. Python and JavaScript were the two primary choices due to prior experience. They both feature extensive packages that would be beneficial for use on our website. With our research, Java seemed appropriate because of the drawing packages available. However, Python is easier to learn due to prior experience with C coding while only one member is familiar with Java. Our group saw the advantages with both so we decided to meet in the middle and use JavaScript for the front-end functions and Python for the back-end functions.

The creation of the website can be split up into four categories: HTML/CSS appearances, JavaScript front-end functions, Python back-end functions, and the Django framework.

The HTML/CSS code provides the basic visuals for the site. Figure 3 shows our current test server. The left side of the website features buttons that link to different parts of the website which will be added to as time goes on. In the middle is an iframe element which embeds a separate HTML file called "drawSpace" within the main HTML file. The JavaScript will be incorporated into this embedded file. On the right is the list of buttons that, when pressed, should add images (components) to the drawing space. This feature has not been implemented as mentioned in the JavaScript portion.

JavaScript is used to create the functions for the interactive portion of the website. These functions allow for components to placed, dragged, and snappable within the drawing space. There were a couple of open source drawing libraries we tested that allowed for this type of functionality. In our first attempt, we used a popular library called jQuery. This provided a customizable dragging feature where components can be moved around in a grid-like fashion. However, users did not have the ability to add more to the circuit as we only managed to get this working for a premade circuit in the drawing space. Attempts were made to use a function called ".appendTo" which attaches an HTML divider, typically text, on the press of a button but this did not work when using an image file. We were then able to enable the dropping feature in which we can click on a component on the right side of our website, and drop the chosen component into the drawing space. The appropriate image then appears, but the component is no longer able to be dragged around the drawing space. Our current plan is to try using a different library that combines proper placement, dragging, and resizing functionality.

3. Testing and Implementation

Our testing process is done on a local server in which we can live edit our code, and immediately see the success of our code's intended function. The code should then be shared with the rest of the team, and each team member should intuitively be able to execute the intended function on their own devices as well. Lastly, we plan to present our progress to our client and advisor, Dr. Bolstad for approval and feedback. Thus far, the appearance of our website has been created with HTML and CSS and shown to our client for a confirmed approval of the graphical presentation.

We are currently in the middle of testing the drawing capabilities of our drawing space. Using jQuery, we were able to implement the dragging feature of pre-populated components within our drawing space. This has been tested and demoed to our client for approval. We then tackled the dropping feature, and this was done with native HTML5. The feature could successfully be dragged from the components tabs into the drawing space, but could not be dragged around the drawing space from there. Thus, we have a failed testing result for the "drag and drop" feature, as both functions are only working independently from one another. We learned that if this feature is to be successfully implemented, we must move forward using a single library to do both functions.

In the following sections, we will be outlining the interface specifications and relationships of our design's implementation. What is also covered is the software we are using, the tests that we still intend to carry out, and the procedure in which we will follow to declare a successful test.

3.1 Interface Specifications

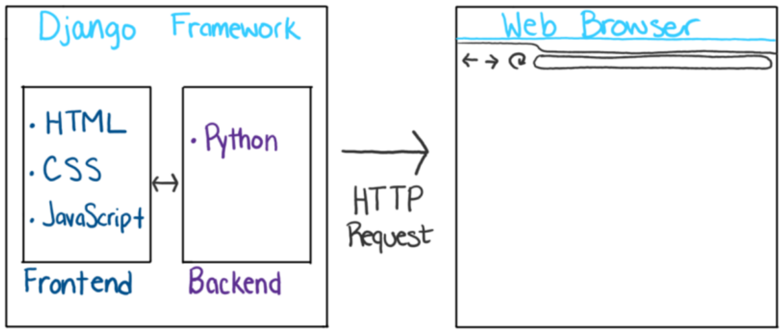


Figure 4: Relationship Between Django and Supporting Languages

In this project, the only hardware required is a computer to write, test, and upload the code with. The software we will be using includes Django for the framework platform to hold our front and backend of the application's functionality. Every team member will use their preferred code editors and compilers outside of Django if they wish. We plan to request server space from the university to host our website on when we're ready to pipeline. Until that point, we can host our website application locally using Django.

The Django framework, as explained above contains both our front end and back end code. The front end code includes all aspects of what the user interacts with directly. This includes the appearance of the website, which is dictated by HTML and CSS, as well as the interaction handlers, which is what the JavaScript is used for. This functionality includes the drag and drop feature for the circuit components. From there, the Python will handle the backend code, which will include all of the circuit simulating actions. This means that the JavaScript will define the placement and movement of the components, while the Python will handle the behavior of how those components simulate.

3.2 Hardware and software

Similar to the above section, the testing phase will not have any kind of hardware required to complete our task. The software will still be using the Django web framework as our testing environment. In the software, we will be unit testing each key feature individually to ensure functionality. This will be the most beneficial because it means we can all make local changes and check features individually and in pairs before pushing it to the master code. Django will have a debugger, as well as the capability to create unit tests with a built-in module in the Python standard library. Once a unit test has successfully passed, we will turn to systems for others in the group to review a pairing or individual's code, and if they can approve functionality, the feature can be pushed to the master code, as well as updated on the website. This testing platform is a system that our group agreed to be the most useful for our project.

The JavaScript code has the functionality of dropping a component from button on the right side of the website to the drawing space, then being able to move them around within the drawing grid and connecting wires between component parts. This means the JavaScript tracks the position of the components within the drawing grid.

The Python code will handle the backend user authentication, downloading the drawing space and saving it as a file, and all of the simulating functionality. When the JavaScript would handle the positioning of the components, the python will handle what happens when the user hits the simulation button, and if the circuit can be simulated, the python will determine this and handle all of the cases.

3.3 Functional Testing

The following tests have been separated into three categories. Those categories are unit tests, box tests, and system tests. Each test case includes a procedure and status. These cases are designed for basic layout and design.

Unit Tests

Number	**Test**	**Desired Outcome**	**Status**
1	Page Links	Clicking links yield HTTP 200 Status Code	Passed
2	GUI Drag and Drop	Dragging a component places it on GUI	In progress
3	GUI Wires	Wires must move with the object when the object is dragged	Not Attempted
4	GUI Background	Background must be clients desired color	Passed
5	GUI Component Menu	GUI component menu can place parts into GUI and all parts are easily visible	Passed

6	GUI Hotkey Placement	Pressing one of the Hotkeys displays a component on the GUI	Not Attempted
7	Download Button	Download button results in a file that matches user's design	Not Attempted
8	Save Button	A record of the current circuit can be found in the database	Not Attempted
9	Help Menu	Help menu appears when clicked, HTTP 200 Status code returned	Not Attempted
10	Sign in	Sign in allows for authentication, rejection, and user creation	Not Attempted
11	New Page Rendering	Django server displays the result of last HTTP Request	Passed
12	Database	Database features can be saved and accessed in HTML templates	Not Attempted

Table 1: Unit Test

Procedures

Test Case 1

1. Boot up Django testing environment
2. Click on links
3. Make a determination on the functionality

Test Case 2

1. Render the page using Django testing environment, or open template with chrome
2. Attempt to drag a component to the Schematic
3. Test the JavaScript code status using the chrome JavaScript debugger.
4. Make a determination on the functionality

Test Case 3

1. This test can only be executed if test 2 passes
2. Attempt to place a wire.
3. Attempt to connect components with the wires.
4. Move the wire and components and interpret the results

Test Case 4

1. Render the page in the Django testing environment or with google chrome
2. Examine the GUI and evaluate the background

Test Case 5

1. Start up the Django testing environment
2. Attempt to drag a component onto the GUI
3. Scroll through the components to make sure all are easily found
4. Observe the behavior of the GUI

Test Case 6

1. Start up the Django test environment
2. Make use of the chrome JavaScript debugger
3. Press one of the Hotkeys
4. Observe Behavior

Test Case 7

1. This test can only be run if tests 1, 2, and 3 are working
2. Click the download button
3. View the process with a debugger
4. Open the downloaded schematic and make a determination

Test Case 8

1. The required testing for this step is unclear at this time

Test Case 9

1. Start the Django Testing Environment
2. Navigate to the help menu.
3. Click on the help menu and make a determination.

Test Case 10

1. Start the Django testing environment
2. Click the sign in button
3. Attempt to sign in incorrectly
4. Attempt to sign in correctly
5. Attempt to create a new user
6. Make a determination on the functionality of the sign in

Test Case 11

1. Start the Django testing environment
2. Complete functionality necessary to move to a new page (To Be Determined)
3. Make a determination

Test Case 12

1. The required testing for this step is unclear at this time

Box Tests

Number	**Test**	**Desired Outcome**	**Status**
1	GUI	Schematics can be created with all parts labeled and mobile	In Progress
2	Authentication	Correct Username and Password allow for viewing of old schematic	Not Attempted
3	Homepage	Homepage allows for GUI creation, site navigation, and download capabilities	Not Fully Implemented
4	Backend	The backend is able to process any data necessary and control website views	Not Attempted

Table 2: Box Test

Test Case 1

1. Start the Django testing server
2. Send a GET request to the server and view the home page
3. Create a schematic using all of the GUI components from the unit tests
4. Save the design

Test Case 2

1. Start the Django testing environment.
2. View the effects of HTTP requests on authentication box
3. Test the User creation and sign in parts
4. Make a determination on the quality of the authentication process

Test Case 3

1. Start the Django testing server.
2. Ensure all buttons on the home page are working together - ensure unit test cases have no effect on each other
3. Document the results of different functionalities
4. Make a determination on the quality of the homepage

Test Case 4

1. The testing process for this case is yet to be determined

System Test

Number	**Name**	**Desired Outcome**	**Status**
1	System Integration Test	Fully Functional System	Not Attempted

Table 3: System Test

3.5 Non-Functional Testing

Our non-functional requirements encompass the visual aspects of our website. This includes:

NFR.1: Switchable white/grid background

After implementing a checkbox to cycle between a white and grid background, we will test if the appearance changes on a test server.

NFR.2: Help Menu

Test the button's ability to bring up the components features and equations with all components implemented in the drawing tool.

NFR.3: All Basic Components Available

We will look at the most common components used in circuits and implement their images into the drawing tool

NFR.4: Easily seen within presentations

After the imaging tool is created, we will copy it into popular presentation tools and verify its legibility

NFR.5: Educational tab with example circuits

Our educational tab will be filled with common circuits that do not require a user to recreate them. We will test whether the circuits appear in their own drawing space on this separate tab.

3.4 Process

Below is a figure that explains the basic process for our project:

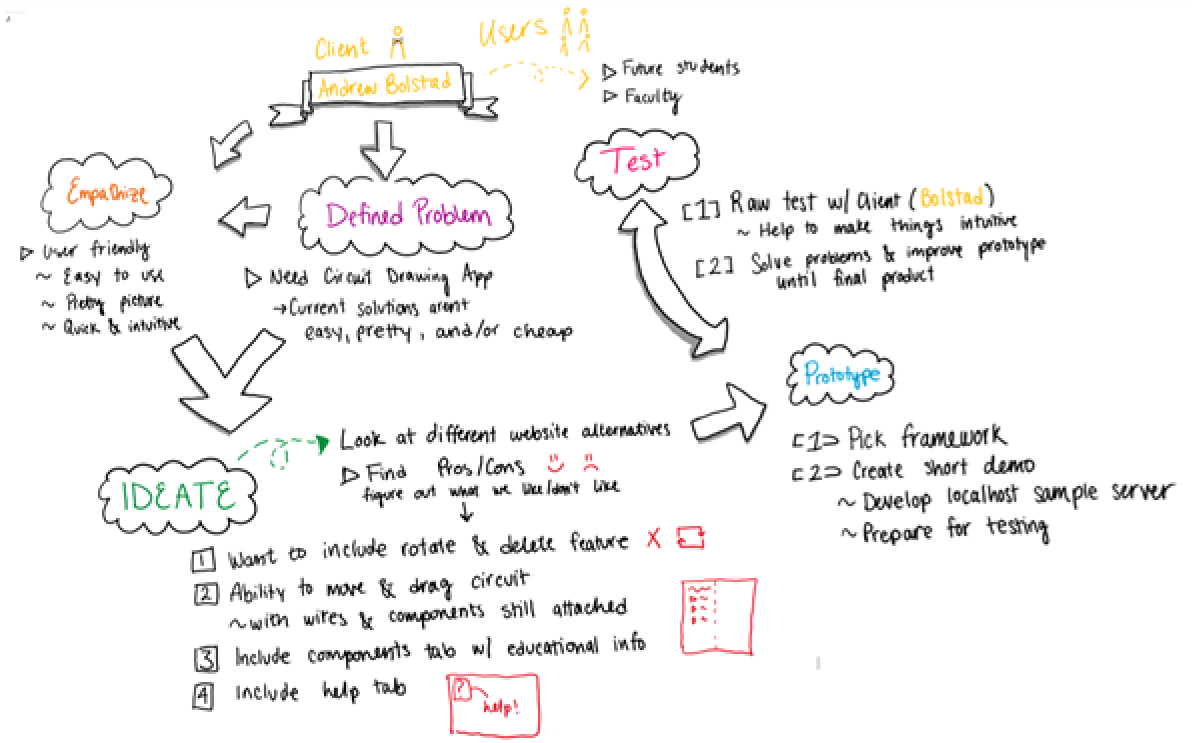

Client
Andrew Bolstad
Users
▷ Future students
▷ Faculty
Empathize
▷ User friendly
~ Easy to use
~ Pretty picture
~ Quick & intuitive
Defined Problem
▷ Need Circuit Drawing App
→ Current solutions aren't easy, pretty, and/or cheap
Test
[1] Raw test w/ Client (Bolstad)
~ Help to make things intuitive
[2] Solve problems & improve prototype until final product
Prototype
[1] Pick framework
[2] Create short demo
~ Develop localhost sample server
~ Prepare for testing
IDEATE
Look at different website alternatives
▷ Find Pros/Cons
figure out what we like/don't like
1 Want to include rotate & delete feature
2 Ability to move & drag circuit
~ with wires & components still attached
3 Include components tab w/ educational info
4 Include help tab
help!

Figure 5: Process Flow Relationships

We started the project by meeting with Professor Bolstad in order to understand exactly what he needs. We empathize with our client and determined that he is unhappy with his lecture notes that contain circuit diagrams. Professor Bolstad is currently using PowerPoint to create circuit diagrams for his lectures, but he found that it is too time-consuming and difficult to create the circuits that he wants. Rather than using pre-made circuit components for his notes, he is using the shapes found in PowerPoint to build his circuits and these components do not offer the functionality that he needs. Furthermore, our client is unable to save the circuits that he creates, in order to use the circuit diagram for other class notes or for homework; and so he is continuously building new circuit diagrams. This has been a very time consuming and labor-intensive task for creating his class lectures, so Professor Bolstad is looking for a solution - a website that is intuitive and easy to use to make visually appealing circuit diagrams. Therefore we defined the problem and determined that Professor Bolstad wants a circuit diagram that is neat, easy to read, easy to comprehend, and has labels for each circuit component. In addition, he wants to save these circuit diagrams for later use, efficiently create any circuit he needs in a timely manner, and possibly simulate the circuits for educational practices. We then began to ideate new features that we could add into the website application such as rotate and delete functionality, the ability to move and drag the entire circuit, a components tab that will educate users on various circuit components, and a help tab that will explain how to use the website. As we implement each feature into the website, we will test its functionality and compatibility in order to ensure that we are creating a user-friendly website for our client.

Once all of the features have been incorporated and tested, we will review the prototype of the website with Professor Bolstad in order to determine what he likes and dislikes about the website. This final stage of the project will allow us to make certain that we have satisfied all of Professor Bolstad's needs with the website and we can make any changes based on his preferences.

Furthermore, while the first stages of the project (empathizing with our client and defining the problem) took about two weeks for us to complete, the final stages of the project (ideating new features for the project, building prototypes of each new addition to the website, and testing the functionality of each new feature) will be an iterative process as we alter the project according to our client. In addition, because we have divided the team into smaller teams so each team can work on different features within the project, each team will be in varying final stages of the project. For example, once one sub-component of the project has been developed, tested, and reviewed by our client, one team will begin developing the next sub-component until we have implemented everything that Professor Bolstad wants.

3.5 Results

Our project currently consists of a Django server and a variety of web pages it supports. We have performed some of our unit tests on the server to verify the functionality of specific components. We have performed tests in which the status was determined by the HTTP status displayed by the server. These tests were able to pass. We have tried working with the Django ORM to interact with the database and have successfully saved and accessed data from the SQLite database. We have not finished the implementation of specific data points we will need because we are still in the development phase of the GUI portion. From a systems level, we are only able to try some box tests at this point. We have performed a box test on the functionality of the server and that yielded a passing result; however, we were not able to test any of the GUI functionality on the box level.

Our group has also encountered many challenges throughout the course of our project. Some of the large obstacles we are currently facing are caused by our need to move data. We need to move the user input to the components to be moved to the database. Django allows us to do this with predefined forms. However, we are not entirely sure what fields will be necessary at this time because we are trying different ways to move the data. One way we have tried to move data is to put data into a JSON file and save that file to the SQLite database. We have also tried using predefined fields and using JavaScript to assign values to our fields. The best way to achieve this goal is still under testing.

Our group has also tried to implement the GUI in several ways. Since we are still working on changes to the GUI libraries, we have not been able to perform any testing. This problem is closely related to our issue with moving data because we need the GUI to be able to hold data before it is moved. One of the GUI options we have tried is the drawSpace library. We are still attempting to store data while using this library.

Nearing the end of completion of Prototype I, we currently have a running HTML/CSS prototype of our web application. This includes a homepage with functional tabs that will be on the final version of the application and a drawing space for the circuit components. From a JavaScript standpoint, we have a draggable component in the drawing space working properly with the HTML/CSS prototype. After testing the prototype where it stands, it is clear that there is a lot of work to do for drawing wires seamlessly in the drawing space.

4. Closing Material

4.1 Conclusion

There exists many different circuit applications, but they are often weighed down with heavy processor requirements and expensive licensing costs. In addition, some of these applications are not free and intuitive to use. Therefore, we are creating an educational, easy to use, and free web-based application that will eliminate these issues. Our website will feature an appealing circuit drawing tool that can be used to teach students who are interested in circuit design and theory. We plan to add important formulas for circuit components and a simulation feature for the circuit so students will be able to understand the function of any circuit.

In order to make this project possible, we will divide the work accordingly so each member will learn how to utilize both Python and JavaScript. This will give each member thorough experience with software design, and it will allow everyone to follow the progress of the project. Furthermore, we will have weekly meetings in order to discuss the team's accomplishments, schedule, and issues concerning the various components to the project. This constant communication will ensure an efficient design process and it will allow us to stay on top of any issues in order to resolve them as soon as possible. Although we are all electrical engineering students, our planning and organization skills are exactly what this project needs to solve this software design problem.

4.2 References

1. Falstad Circuit Drawing Website: https://www.falstad.com/circuit/
2. Partsim (SPICE) Informative Website: http://www.pspice.com/
3. Circuit Diagram Website: https://www.circuit-diagram.org/editor/
4. Circuit Lab Website: https://www.circuitlab.com/editor/#?id=7pq5wm
5. Django Documentation https://www.djangoproject.com/
6. Python Documentation https://docs.python.org/3/