Debugger and Visualizer for a Shared Sense of Time on Batteryless Sensor Networks

Design Document

Team 15 (May 2021)

Client/Advisor

Professor Henry Duwe

Team Members

Adam Ford - Report Manager

Allan Juarez - Scribe

Maksym Nakonechnyy - Design Lead

Anthony Rosenhamer - Facilitator

Quentin Urbanowicz - Test Engineer

Riley Thoma - Project Manager

Email: sdmay21-15@iastate.edu

Website: http://sdmay21-15.sd.ece.iastate.edu

Revised: October 4, 2020 (v1)

Executive Summary

Development Standards & Practices Used

The proposed solution is to create a web-application that will allow the user to visualize and debug shared sense of time on batteryless sensor networks. The system will also provide functionality to simulate a sensor network, and visualize and debug these simulated networks.

The backend application will be developed following the microservices architectural style to allow for code maintainability and easy additions of new functionality. Microservices will provide RESTful APIs that will be used to provide access points for the frontend application and for communication between microservices. This communication will take place using the HTTP requests. We decided to use HTTP instead of HTTPS because the client does not have any requirements regarding security of the data.

We agreed to use standard coding conventions for the programming languages to make the code more readable and documentable. Where possible, we will use tools to automate documentation (e.g. Javadoc).

We decided to use GitLab for both source control and task tracking. The plan is to also integrate a CICD pipeline for our project in GitLab.

We decided to follow agile methodology for our project aiming to deliver functionality incrementally. We will communicate with the client after each iteration to make sure that the product satisfies the requirements and client's needs. For risk management, we will use Jackson's principle of commensurate care.

Summary of Requirements

- The system shall display the live status of time on the sensors.
- The system shall store past data.
- The system shall simulate the data in the same format as real data.
- The system shall accept a seed value for pseudo-random simulation.
- The system shall "replay" past data
- The system shall display up to 15 sensor nodes on screen

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

TODO in next design document

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum to complete this project.

TODO in next design document

Table of Contents

List of figures

List of tables

List of symbols

List of definitions

Backend	Application for storing and processing time data to be exported or sent to the Frontend
Frontend	Web application for viewing the sensor network’s time data
GUI	Graphical User Interface
Simulator	Application that produces time data that simulates the time data that would be coming from a sensor network

1 Introduction

1.1 Acknowledgement

We would like to acknowledge our faculty advisor and client, Dr. Henry Duwe, for his guidance and expertise throughout the course of this project. We would also like to thank Dr. Duwe's graduate research partner, Vishal Deep, for their research on distributed batteryless timekeeping systems, which forms the need for our project and without which our work would not be possible. In addition, we would like to extend our gratitude to Iowa State University for providing access to software and hardware resources for testing and development, and for providing our team this opportunity to contribute to the ongoing development of batteryless intermittent computing technologies.

1.2 Problem and Project Statement

Problem Statement

In distributed embedded computing systems, reliable timekeeping is essential. Typical methods of keeping track of time require continuous power to function, usually from a battery. However, the use of batteries in certain applications, like distributed sensing, may pose significant challenges, particularly in circumstances where replacing batteries is infeasible or impossible.

For such use cases, batteryless devices, which draw the energy required for their operation solely from ambient sources, present numerous advantages; the lack of a consistent power source, however, necessitates a new method of keeping track of time and ensuring that sense of time is shared between all nodes in a system.

A graduate research group at Iowa State University, led by Dr. Duwe, has demonstrated designs for real-time embedded clocks which are capable of keeping time without the need for a continuous source of power. However, due to variations in manufacturing, susceptibility to noise, and the added complexity of keeping a sense of time synchronized across a distributed system, visualizing and debugging these clock systems is exceedingly difficult.

In order to optimize designs and detect unknown bugs, a simulation tool capable of modeling the interactions within a distributed network of clocks and determining each node's resultant sense of time is required. A debugging system capable of probing and visualizing the state of the network and examining the dependencies between each node's sense of time is also needed.

Proposed Solution

Our team aims to develop a pair of software tools for debugging the shared sense of time across a network of distributed clocks: a simulator, which models and records time estimates across a network over time, and a visualization dashboard, which utilizes data from simulations or a yet-to-be-developed hardware sniffer, to visualize these changes and analyze the interconnections between individual nodes. Our simulator will utilize parallelization where possible to reduce computation time and enable scalability across larger numbers of nodes. The dashboard will utilize a locally-hosted web application to ensure compatibility across all operating systems and enable use by multiple users simultaneously. With these deliverables, we hope to create a set of utilities which are powerful enough to achieve the levels of accuracy and precision required for academic research, yet flexible enough to adapt to design changes and enable collaboration with minimal effort. The entire system is shown below in Figure 1.1

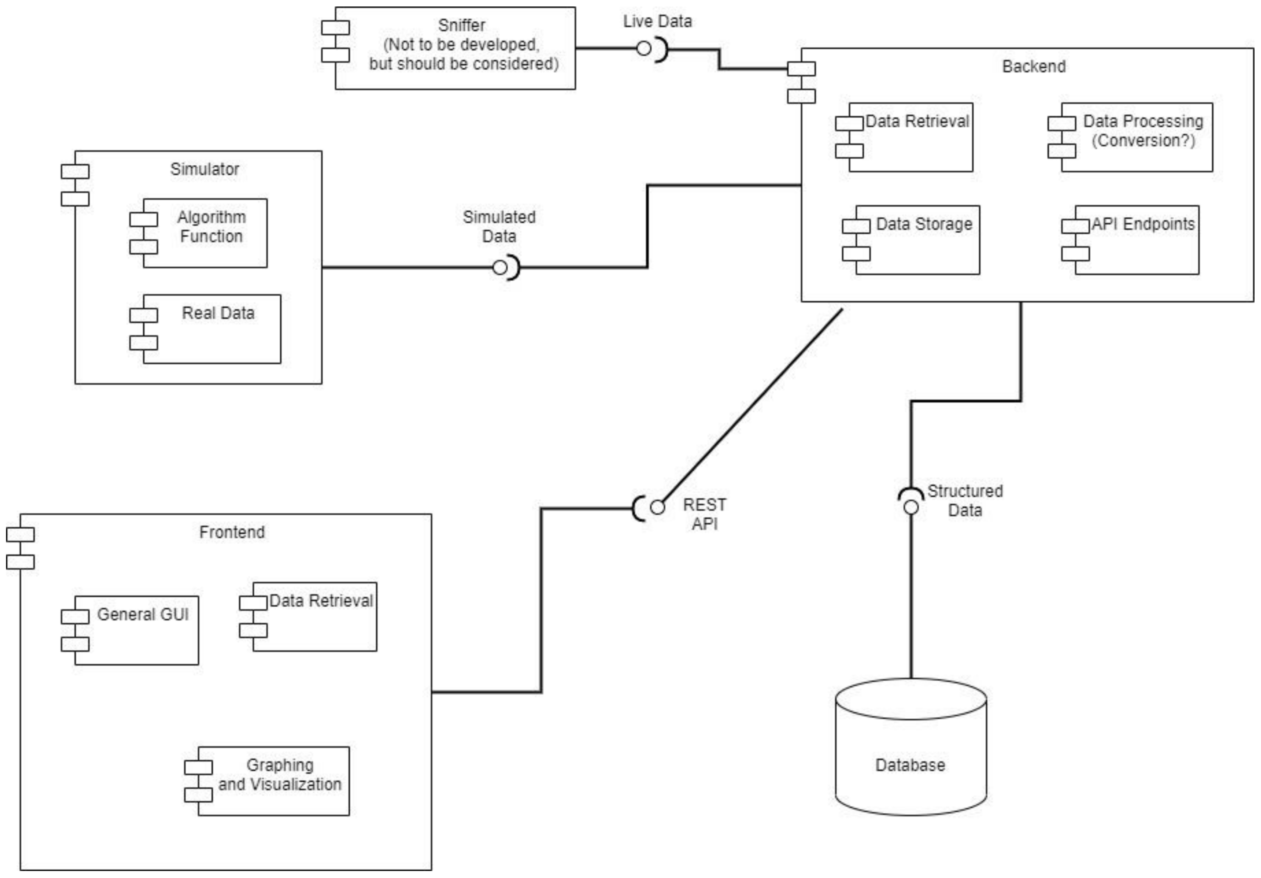

Figure 1.1: High-Level System Diagram

1.3 Operational Environment

Due to the necessity of generating, storing, and analyzing large sets of data, our team will be developing our software tools with high-performance computing hardware in mind. The simulator will be tailored to run natively in a Linux environment and will be operated primarily through a command-line interface. The visualization dashboard will run as a locally-hosted web application and thus will require a web browser with support for modern web standards such as HTML5, CSS 3, JavaScript, and WebGL. Since the dashboard web app will be locally hosted, and since interaction with sensitive information is not required, there are no specific concerns for data security or privacy compliance.

1.4 Requirements

Functional requirements:

- The visualizer shall display the live status of time on the sensors.
- The system shall store past data.
- The simulator shall generate the data in the same format as real data.
- The simulator shall accept a seed value for pseudo-random simulation.
- The visualizer shall "replay" past data.
- The visualizer shall display up to 15 sensor nodes on screen.
- The visualizer shall display previously saved sample data.
- The visualizer shall visualize the statistics of system communication.

Non-functional requirements:

- The visualizer shall update node status every second.
- The simulator shall maintain sub-second accuracy of timing.
- The visualizer shall be implemented as a web-application.
- The system shall be modular to allow for maintainability.
- The system shall not lose any sensor readings.
- The simulator shall produce on-time/off-time data from a user-provided function.
- The system shall be accessible from any device or OS.

Economic requirements:

- The system shall utilize hardware provided by the client.
- The system shall be built using open source software to minimize costs.

Environmental requirements:

- The system shall run in a controlled environment, so there aren't any environmental requirements.

1.5 Intended Users and Uses

The system shall support research group members. Their team requires this system for their research into timekeeping for batteryless sensor networks. The system shall provide the following functionality to users:

- View the state of sensors in real time.
- Add a sensor to be tracked.
- Remove a sensor.
- Simulate a network.
- "Replay" past data for the network as a whole.
- "Replay" past data for an individual node.
- See the statistics of system communication.
- Provide a seed value for pseudo-random simulation.
- Customize the dashboard.
- See the propagation of error from one node to another.
- Log the errors.
- Load previously saved logs to be displayed.

- Export past and live data.
- Import past data.

1.6 Assumptions and Limitations

Assumptions

- We will be provided data from the sniffer to emulate it.
- The minimum number of nodes to be simulated is three.
- The project will only be used by Professor Duwe and his graduate students.

Limitations

- With no budget right now, we will have to find some ways to host our project and find a database for the project without spending anything.
- With the pandemic going on, we are unable to go into the lab and test our project, and get fresh data.
- There is no actual sniffer, so it will be hard to test the full system.

1.7 Expected End Product and Deliverables

Design Deliverables

By the end of the first semester we are expected to complete and present our design document. The design document will have all our procedures and plans to complete our full system for the following semester.

Development Deliverables

The expected end product for this project is to have a simulator and a dashboard. The simulator shall provide data similar to an actual sniffer node from the hardware project. These goals will be met at the end of the spring semester. For the dashboard we will be expected to at a minimum display three nodes, but our professor expects us to display fifteen nodes at a time. The dashboard will be accessible from any computer that Professor Duwe and his Graduate Students have. In the dashboard we will add panels that give information about all the nodes, including the error each node currently has and what time it is displaying.

This development process will also result in a deliverable of the code documentation. This will detail how the code works and how it should be used in specific. Additionally, our project will have test cases and other deliverables related to testing the product.

2 Project Plan

2.1 Task Decomposition

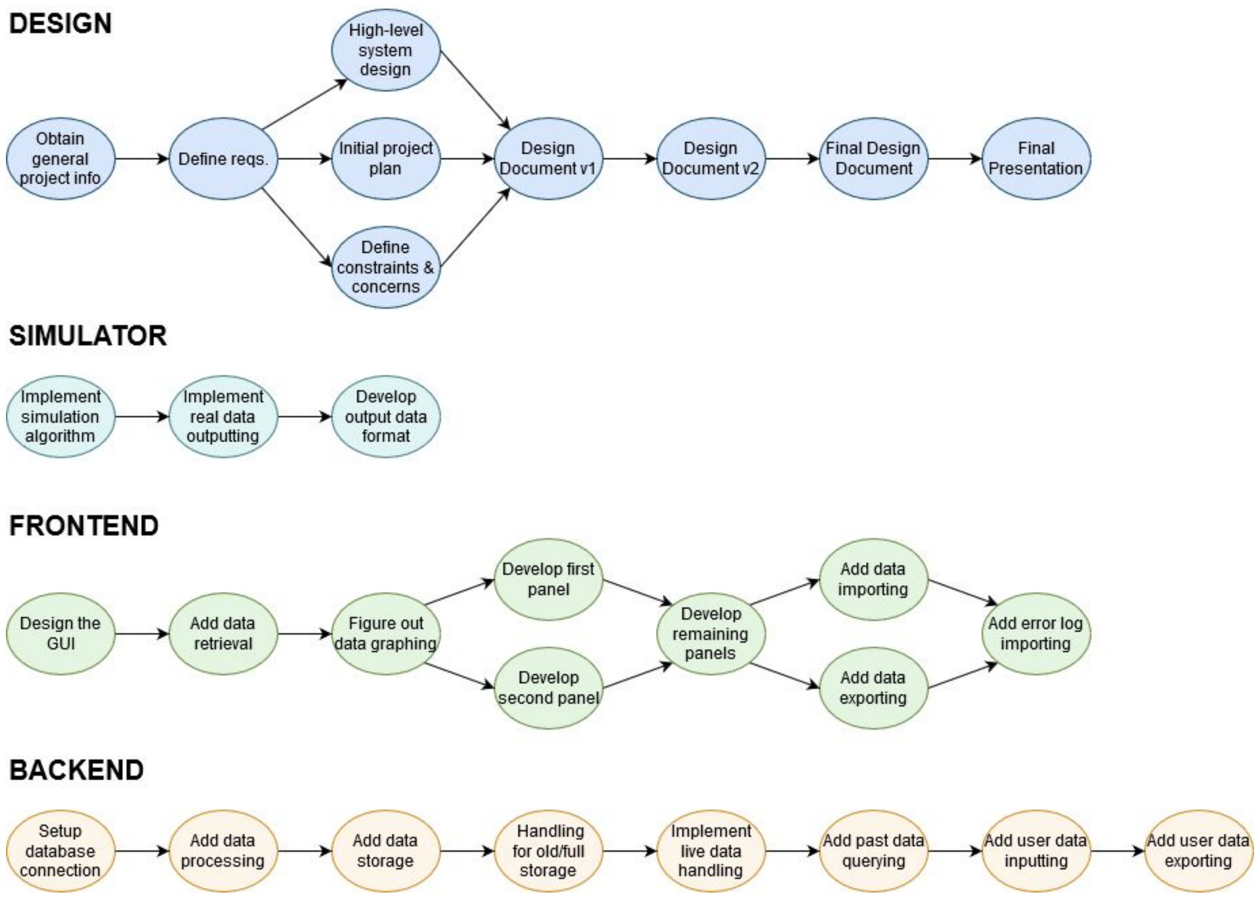

Figure 2.1: Task Graphs

The task graphs in Figure 2.1 show the dependencies between each of the tasks for the four sections of the project (Design, Simulator, Frontend, Backend). The Design and Frontend tasks have stages that allow for concurrent work on tasks that are co-dependencies for the following task.

Component	Task	Allocated Time (person-weeks)	Due By	Risk
Design		Fall semester		0
	Design Doc v1	2	Oct. 4	0
	High-level system design	2	Oct. 4	0
	Initial project plan	2	Oct. 4	0
	General project information	2	Oct. 4	0
	Define requirements	2	Oct. 4	0
	Define constraints and concerns	2	Oct. 4	0
	Design Doc v2	3	Oct. 25	0
	Design Doc Final	3	Nov. 15	0
Simulator		8 person weeks		
	Implement simulation algorithm for test data	3		0.5
	Implement real data output functionality	2		0.25
	Develop output data format to mimic sniffer	3		0.25
Frontend		14 person weeks		
	Design a GUI	2		0
	Retrieve and Process data from backend	1		0.5
	Learn how to graph and visualize the data	1		0.5
	Develop the First Panel	2		0.25
	Develop the Second Panel	1		0.25
	Develop the rest of the necessary panels	4		0.25
	Import data to the database	1		0.25
	Export Data from the database	1		0.25
	Import a log of the error in nodes	1		0.25
Backend		12 person weeks		
	Make and setup database connection	1		0.5
	Process real or simulated data into storable format	1		0.25
	Store data in database from backend	1		0
	Drop data when database is full, or it is too old	2		0.25
	API Endpoints:	7		
	Provide live current data	1		0.75
	Query and return past data to “replay”	2		0.75
	Accept input data from user	1		0.25
	Query and return past data to user as export	3		0.25

Table 2.1: Task Decomposition

2.2 Risks and Risk Management/Mitigation

The risk level for each task is given in Table 2.1 of the Task Decomposition section. The higher risk items (> 0.5) are analyzed below.

Provide live current data - risk: 0.75

The inherent risk in this task comes from the need for live data to be generated, processed, and displayed. This will be the crux of our project and will be risky in that our performance may not be where we want it when we first get it working. If our performance is not where we need it other tasks may be affected as the code may need to be cleaned or changed to be more efficient. To mitigate this risk we will be conscious of efficiency when coding other parts of the projects.

Query and return past data to "replay" - risk: 0.75

This task is risky because of the difficulty in trying to retrieve old data to then replay it again. We will mitigate this risk by prototyping the replay feature and making sure our design will support it once we reach this task. We will future proof our application to make sure data is saved and easily retrieved for replaying.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Major Milestones:

- Design Document Final Draft Complete
- Minimum Viable Product (MVP) built
- Simulator Complete
- Frontend Complete
- Backend Complete
- Integration Complete (Final Release)

Minor Milestones:

- Simulator algorithm working and producing good data
- Frontend panels are created and working (milestone for each)
- GUI is fully designed
- Importing and Exporting functionality is complete
- Database can store simulator data and provide it to the frontend
- Database can store past data for replayability
- Database setup and working

2.4 Project Timeline/Schedule

The following Gantt chart shows the timeline for our project. The tasks are divided into four sections: Design, Simulator, Frontend, and Backend.

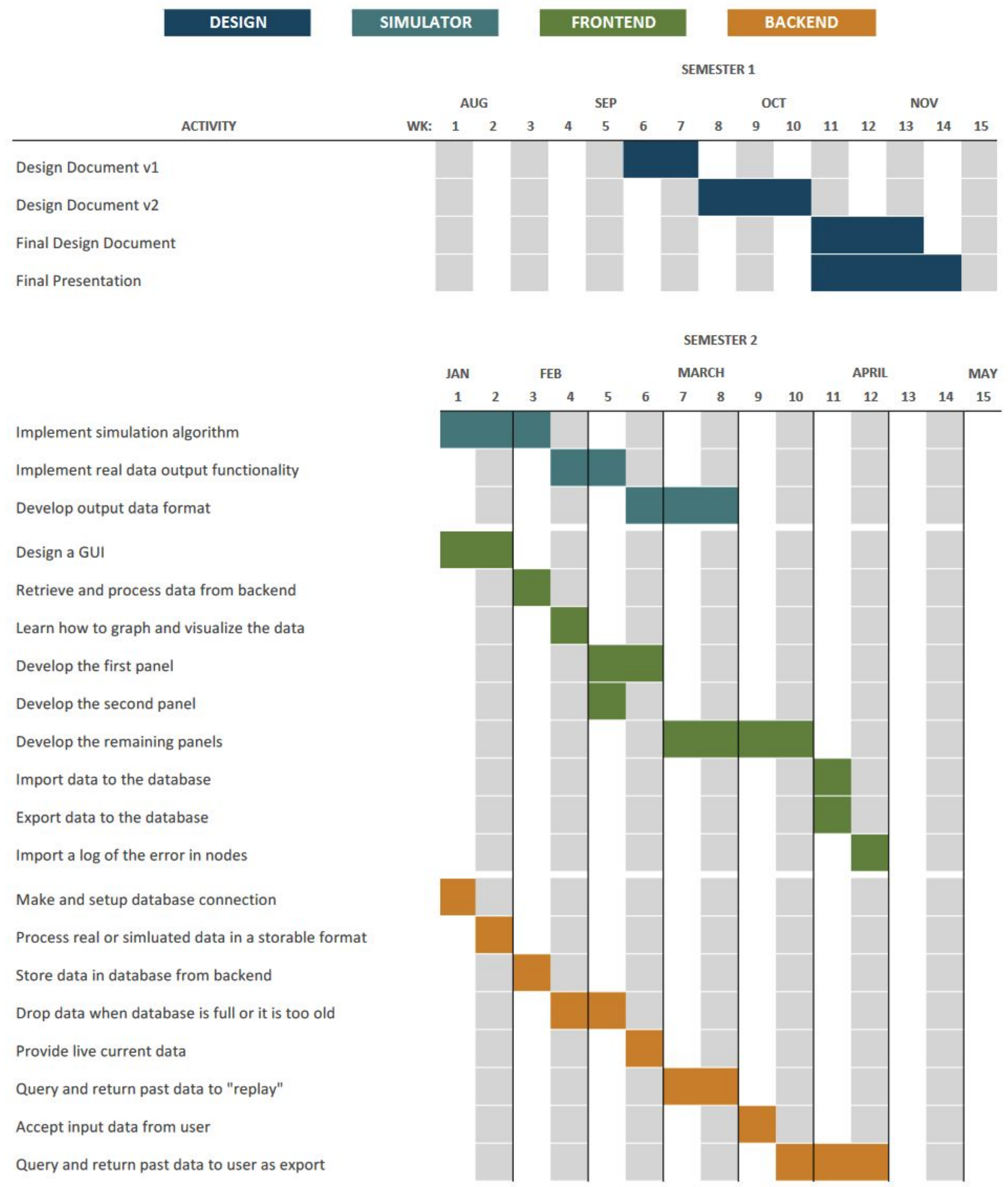

Figure 2.2: Gantt Chart

All of the Design tasks will be completed in the first semester. The work on consecutive design documents will likely begin while the previous document is being completed. The Simulator tasks include implementing the simulation algorithm, outputting real data, and developing the output data format, and these should be completed around the middle of the semester. The next section of the Gantt chart shows the Frontend tasks, which has a few tasks that allow for concurrent work.

The Frontend tasks are broken into GUI design, data retrieval, data visualization, GUI panel development, data importing and exporting, and error log importing. The final section shows the Backend tasks; it is broken into tasks for setting up the database, processing and storing data, handling old or excess data, providing live data, query past data, accepting user data inputs, and providing data exporting. The divisions in the second semester represent the two-week sprints that we will use since we are practicing the Agile approach to software development.

2.5 Project Tracking Procedures

Our group will use a combination of online tools to track progress on our project. We have a Slack workspace that we are using for quick communication and schedule coordination. We also have a shared Google Drive folder that we use to store collaborative files, such as Status Reports, Design Documents, Lightning Talks, etc. We will use GitLab for managing code, tracking issues, and monitoring overall progress. We will use GitLab's built-in sprint management tools, such as the Kanban board, to track our Agile development.

2.6 Personnel Effort Requirements

The effort estimates are shown in Table 2.1. However, each team member's focuses are shown below. These will guide which tasks each of team members primarily works on, however adjustments can be made when the project pivots.

Simulator (8 person weeks) - Anthony, Quentin
Frontend (14 person weeks) - Riley, Maksym
Backend (12 person weeks) - Allan, Adam

2.7 Other Resource Requirements

Linux Machines (VMs) - Our simulator is planned to run on a Linux Machine for production, so we will need the ability to develop and test on similar machines.

Software Packages - We are attempting to utilize open source software in as many places as possible, however it is possible that we may need a license to a package if the project specifics adjust to it. The most likely case of this is for live graphing, more research will need to be done to determine packages that can serve our needs.

2.8 Financial Requirements

There is currently no financial need with the project. Some unknowns that may lead to financial need are hosting, Linux Virtual Machines and package licenses. The assumption is that the university can provide hosting, virtual machines and licenses for our uses. However, if they cannot, we will address those budget concerns when they occur.

3 Design - TODO for the second design document

3.1 Previous Work and Literature

Include relevant background/literature review for the project

- If similar products exist in the market, describe what has already been done
- If you are following previous work, cite that and discuss the **advantages/shortcomings**
- Note that while you are not expected to "compete" with other existing products / research groups, you should be able to differentiate your project from what is available

Detail any similar products or research done on this topic previously. Please cite your sources and include them in your references. All figures must be captioned and referenced in your text.

3.2 Design Thinking

Detail any design thinking driven design "define" aspects that shape your design. Enumerate some of the other design choices that came up in your design thinking "ideate" phase.

3.3 Proposed Design

Include any/all possible methods of approach to solving the problem:

- Discuss what you have done so far – what have you tried/implemented/tested?
- Some discussion of how this design satisfies the **functional and non-functional requirements** of the project.
- If any **standards** are relevant to your project (e.g. IEEE standards, NIST standards) discuss the applicability of those standards here
- This design description should be in **sufficient detail** that another team of engineers can look through it and implement it.

3.4 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.
Discuss possible solutions and design alternatives

3.5 Design Analysis

Did your proposed design from 3.3 work? Why or why not?
What are your observations, thoughts, and ideas to modify or iterate over the design?

3.6 Development Process

Discuss what development process you are following with a rationale for it – Waterfall, TDD, Agile. Note that this is not necessarily only for software projects. Development processes are applicable for all design projects.

3.7 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

4 Testing - TODO for the second design document

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or software.

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study or acceptance testing for functional and non-functional requirements).
2. Define/identify the individual items/units and interfaces to be tested.
3. Define, design, and develop the actual test cases.
4. Determine the anticipated test results for each test case
5. Perform the actual tests.
6. Evaluate the actual test results.
7. Make the necessary changes to the product being tested
8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you have determined.

4.1 Unit Testing

Discuss any hardware/software units being tested in isolation

4.2 Interface Testing

Discuss how the composition of two or more units (interfaces) are to be tested. Enumerate all the relevant interfaces in your design.

4.3 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

4.4 Results

List and explain all results obtained so far during the testing phase

- Include failures and successes
- Explain what you learned and how you are planning to change the design iteratively as you progress with your project
- If you are including figures, please include captions and cite it in the text

5 Implementation - TODO for the final design document

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

6 Closing Material - TODO for the final design document

6.1 Conclusion

Summarize the work you have done so far. Briefly reiterate your goals. Then, reiterate 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, PCB testing issues, software bugs, etc.