Microgrid Data Platform Enhancements

DESIGN DOCUMENT

sdmay23-37

Client: Mat Wymore
Alexander Haack - Software Engineer
Carson Love - Cybersecurity Engineer
David Harmon - Software Engineer
Harvey Forchu - Computer Engineer
Kenyon Fergen - Cybersecurity/Software Engineer
sdmay23-37@iastate.edu
https://sdmay23-37.sd.ece.iastate.edu/

Revised: 4/29/2023/Version 2.0

Executive Summary

Development Standards & Practices Used

NIST SP 800-53:

This standard is a catalog of different Cyber Security Controls and their use cases. It is a rather flexible framework for establishing what an organization's needs are, and ensuring sufficient controls are in place. This framework will be useful when we assess the Solar Crate's current security posture, as well as for facilitating conversations with the client to determine areas for improvement.

IEEE STD 12207:

This standard provides a framework for the life cycle of creating software. This may be helpful as we progress from planning to implementation of our design.

IEEE STD 829:

This standard is concerned with the documentation of testing, including both software, hardware, and the interfaces connecting them. This will be helpful when we begin testing, as our design covers software and hardware, and we will need to ensure communication succeeds between them.

IEEE STD 830:

This standard describes the proper specification of software requirements. Since our project is heavily influenced by a client, this standard would be helpful to confirm that our design meets the given needs.

Summary of Requirements

Functional Requirements:

- The web app should display historical data from microgrids.
- Ability to view data from following perspectives: 24 hrs, 7 days, month, year
- Ability to download data from graphs
- The system should restrict microgrid viewing access to authorized users.
- The system should securely transmit data from the microgrid to the database and webserver.
- The system should collect and store data from microgrid sensors.
- The system should allow new components on the microgrid to send data to the database.
- The system should allow new microgrids to be added to the system.
- The system should query data from the solar crate every 1 minute.
- The web app should have functionality parity with the existing mobile app

Resource Requirements:

- The system shall use existing ETG computing resources.

User Experience Requirements:

- The web app should display the graphs from the micro grid in an easy and digestible format appropriate to its audience.
- The web app should display simplified data for public users.
- The web app should provide a more advanced breakdown for researchers.
- The system should be easy to navigate between different graphs and different components of the microgrid.

Economic Requirements:

- No budget allotted; Granted ETG computing resources
- The system shall utilize the existing Solar Crate.

UI Requirements:

- Our web app should be reachable from the Electric Power Research Center's website
- Needs an administrator page as well as public view

Applicable Courses from Iowa State University Curriculum

- COM S 228 (Data Structures)
- COM S 327 / CPR E 288 (Advanced Programming / Embedded Systems I)
- SE 309 (Software Development Practices)
- COM S 363 (Database Management Systems)
- COM S 352 / CPR E 308 (Operating Systems)
- SE 329 / SE 317 (Software Project Management / Software Testing)
- CPR E 388 (Embedded Systems II)
- CYB E 230 / CYB E 231 (Cyber Security Fundamentals / Cyber Security Concepts and Tools)

New Skills/Knowledge acquired that was not taught in courses

- React
- Plotly
- Python
- NoSQL Databases

Table of Contents

1 Team

1.1 Team Members

Alexander Haack

Kenyon Fergen

Carson Love

David Harmon

Harvey Forchu

1.2 Required Skill Sets for Your Project

Backend Server Development - We will be updating and modifying the backend code to suit any API changes we need to make for the updated security measures.

Frontend Client Development - We will be updating the frontend of the microgrid from a mobile application to a web application.

Cyber Security Knowledge - We will be upgrading the security measures of the microgrid data retrieval system in order to protect user data and access rights.

Database Querying - We will be upgrading our database scripts to accommodate different microgrids.

1.3 Skill Sets covered by the Team

Backend Server Development - Alexander, David, Harvey, Kenyon

Frontend Client Development - Alexander, David. Harvey, Kenyon

Cyber Security Knowledge - Carson, Kenyon

Database Querying - Alexander, David, Harvey, Kenyon

1.4 Project Management Style Adopted by the team

We will be utilizing Waterfall + Agile as our project management style. We will use waterfall for the overall project, and agile for the cyber security portions of the project.

Waterfall works well for the overall project due to the preset requirements and expectations for the system. Additional frontend features for the web application and backend APIs to the current system are relatively well known and predefined.

Agile works well for the cyber security portion of the project due to the open ended nature of the requirement. It makes more sense to use an iterative approach to gradually increase the security of the project based on findings and changing priorities.

1.5 Initial Project Management Roles

Alexander: Software

Kenyon: Software/Cyber Security

Carson: Cyber Security

David: Software

Harvey: Computer

2 Introduction

2.1 Problem Statement

Our project is focused on gathering data from self-contained power stations known as microgrids, storing the data, and then presenting it for research, demonstration, and publicity on a web application. We are focused primarily on a specific microgrid called a solar crate owned by Iowa State University, which fits inside of a small shipping container. The data collected will be made available to microgrid administrators, researchers, and the general public. The groundwork for this project has already been developed by a prior senior design team (Senior Design Team 21 in Fall 2021 [sddec21-21]), and we will be building upon this foundation by upgrading the frontend from a mobile application to a web application, improving the security measures of the system, and enabling collection of data from more microgrids besides the solar crate owned by Iowa State University.

2.2 Intended Users and Uses

This product will be used by researchers, renewable energy enthusiasts, and microgrid administrators. They would be able to use our product to understand and further study the data collected by the microgrid, assessing if it works as intended, and finding out new data trends from it.

Microgrid Administrators: Microgrid Administrators will be more technical and interested in finer details than the other users. They will need to be able to make sure it is working correctly and be able to change settings. They will benefit from this by being able to fulfill this role remotely.

Researchers: Researchers are actively working on the solar crate. They need to see how efficiently the solar crate is converting energy and what devices are using more energy than others in order to make changes to the interior. This will allow for them to view information from the crate remotely so they can monitor energy levels more often.

Renewable energy enthusiasts: These individuals do not have any immediate cause of the need for the data, but are interested in understanding how the grid works, how it conserves energy, and how it is used to benefit the community. They would like to see how much energy is being processed and how efficiently the grid is working. The platform we create would be able to easily depict this information and provide them with other material they may be interested in.

2.3 Requirements & Constraints

Functional Requirements:

- The web app should display historical data from microgrids.
- Ability to view data from following perspectives: 24 hrs, 7 days, month, year
- Ability to download data from graphs
- The system should restrict microgrid viewing access to authorized users.

- The system should securely transmit data from the microgrid to the database and webserver.
- The system should collect and store data from microgrid sensors.
- The system should allow new components on the microgrid to send data to the database.
- The system should allow new microgrids to be added to the system.
- The system should query data from the solar crate every 1 minute.
- The web app should have functionality parity with the existing mobile app

Resource Requirements:

- The system shall use existing ETG computing resources.

User Experience Requirements:

- The web app should display the graphs from the micro grid in an easy and digestible format appropriate to its audience.
- The web app should display simplified data for public users.
- The web app should provide a more advanced breakdown for researchers.
- The system should be easy to navigate between different graphs and different components of the microgrid.

Economic Requirements:

- No budget allotted; Granted ETG computing resources
- The system shall utilize the existing Solar Crate.

UI Requirements:

- Our web app should be reachable from the Electric Power Research Center's website
- Needs an administrator page as well as public view

2.4 Engineering Standards

NIST SP 800-53:

This standard is a catalog of different Cyber Security Controls and their use cases. It is a rather flexible framework for establishing what an organization's needs are, and ensuring sufficient controls are in place. This framework will be useful when we assess the Solar Crate's current security posture, as well as for facilitating conversations with the client to determine areas for improvement.

IEEE STD 12207:

This standard provides a framework for the life cycle of creating software. This may be helpful as we progress from planning to implementation of our design.

IEEE STD 829:

This standard is concerned with the documentation of testing, including both software, hardware, and the interfaces connecting them. This will be helpful when we begin testing, as our design covers software and hardware, and we will need to ensure communication succeeds between them.

IEEE STD 830:

This standard describes the proper specification of software requirements. Since our project is heavily influenced by a client, this standard would be helpful to confirm that our design meets the given needs.

3 Project Plan

3.1 Project Management/Tracking Procedures

We will be utilizing Waterfall + Agile as our project management style. We will use waterfall for the overall project, and agile for the cyber security portions of the project.

Waterfall works well for the overall project due to the preset requirements and expectations for the system. Additional frontend features for the web application and backend APIs to the current system are relatively well known and predefined.

Agile works well for the cyber security portion of the project due to the open ended nature of the requirement. It makes more sense to use an iterative approach to gradually increase the security of the project based on findings and changing priorities.

Our team will be utilizing Git to version control our software development and to enable easy collaboration, and we will be using Discord as our primary form of communication to discuss important issues and plan out meetings.

3.2 Task Decomposition

- Study existing Cassandra and Spring
- Change/implement Cassandra and Spring to suit project needs
- Design web application
- Design data collection scripts
- Prototype web application
- Prototype data collection scripts
- Edit APIs in existing backend to suit project needs
- Create frontend of web application
- Integrate graphs into frontend
- Identify and evaluate existing security posture
- Expand on existing security implementations and expand on further implementations.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestones:

- Web app prototype
- All mobile app functionality transferred to the web app.
- Report of current security posture completed.
- All added/improved security and access control integrated into the system.

3.4 Project Timeline/Schedule

Task	Timeline
Web app design	October 30, 2022
Data-collection script design	October 30, 2022
Study existing Cassandra and Spring	January 30, 2023
Identify and evaluate existing security posture	January 30, 2023
Web app prototype	January 30, 2023
Data-collection script prototype	January 30, 2023
Change/implement Cassandra and Spring to suit project needs	February 30, 2023
Edit APIs in existing backend to suit project needs	February 30, 2023
Create frontend of web-app	March 15, 2023
Integrate graphs into frontend	April 15, 2023
New Security Controls implemented	April 15, 2023

3.5 Risks And Risk Management/Mitigation

Risks

	Probability	Severity	Total Risk
Solar Grid goes offline (breaks)	0-10%	Moderate	Low
Solar Grid goes offline (moved/scheduled maintenance)	40-60%	Moderate	Moderate
Security Breach - corrupting/manipulating data	0-10%	High	Low
Code lost to laptop crash	80-100%	Low	Moderate
ETG VMs go down	20-40%	High	Moderate

Mitigations

Risk	Mitigation
Solar Grid goes offline (breaks)	Use existing data for testing the project.
Solar Grid goes offline (moved/scheduled maintenance)	Be aware of planned downtime by the grid and plan around it.
Security Breach - corrupting/manipulating data	Backup our data to restore any lost/corrupted information.
Code lost to laptop crash	Push code regularly. Every time a feature is finished, push it. Work on separate branches to store progress without breaking other people's code.
ETG VMs go down	Restart VMs, be able to contact ETG for support if needed.

3.6 Personnel Effort Requirements

Task	Personnel Effort
Web app design	3 hours
Data-collection script design	3 hours
Study existing Cassandra and Spring	5 hours
Identify and evaluate existing security posture	6 hours

Web app prototype	6 hours
Data-collection script prototype	6 hours
Change/implement Cassandra and Spring to suit project needs	8 hours
Edit APIs in existing backend to suit project needs	12 hours
Create frontend of web-app	40 hours
Integrate graphs into frontend	6 hours
New Security Controls implemented	40 hours
Total	135 hours

3.7 Other Resource Requirements

- ETG hosted VMs for backend
- Solar Crate
- Web frameworks/database frameworks

4 Design

4.1 Design Context

4.1.1 Broader Context

Public Health/Safety/Welfare - The Solar Crate can be used to promote Public Health when deployed in the event of an emergency. Getting electricity back online after a disaster is key to ensuring things like proper food preparation and adequate lighting and heating.
Global/Cultural/Social - As a society we are moving towards green energy sources. The shipping container microgrid could have been powered by a diesel generator, but instead it is solar to further investment in the area of green energy. - As a society we choose to set aside resources to prepare for emergencies. This crate is only possible because of this prioritization.
Environmental - This device utilizes solar panels, which have less environmental impact when running. Micro grids also have the potential to reduce resource investment in power lines if the generation and consumption are closer together.
Economic - When deployed in the event of a disaster, this crate has the potential to reduce economic damage and speed up recovery. - The majority of the crate's life will be spent not deployed in a disaster. With that in mind, the publicity and research use of the crate allows it to provide value outside of this rare circumstance.

4.1.2 Prior Work/Solutions

A previous senior design team, sddec21-21, created a mobile application with the same purpose as our project. The problem with having a mobile application was the inconvenience of only being able to access the site through one platform. With a web application, it can be reached from any device with internet access. The previous project was able to narrow down the options of a backend and database, choosing to use a non-relational, NoSQL database for better scalability in the long run.

One con of our project being a web application as opposed to a mobile application will be portable accessibility. This means that it will be somewhat more difficult for our users to access the data they want from our system, as they will need to access it from the web instead of simply opening it up as an app on their phone.

The prior project also had limited security considerations in its design. Our design is focused on improving upon the existing security within the prior project and making the connections more secure.

4.1.3 Technical Complexity

Components/Subsystems:

- Microgrid container
- Python data collectors
- Cassandra Database
- SpringBoot Backend server
- Web-based Frontend

We have full-stack interactions between each of these components and subsystems, where the Microgrid container has a variety of different sensors and devices (e.g. Tesla Wall, Solar Panels, Sensors) attached to it which we must interface with. We interface with these subsystems within the Microgrid container via data collectors written as Python scripts. Connecting between the frontend and the backend components requires networking principles.

In addition to the full-stack interactions between these components, we have to apply authentication and modern cyber security practices to these layers to ensure that none of the data can be manipulated or stolen. This ensures that the data is accessed only by authorized personnel.

Our solution must ensure that we have secure data transmission over two different connections. First, from the various microgrids (e.g. SolarCrate) to the backend server, and then again to the front end web clients. This increases our potential attack surface and requires heightened security as a result.

The final portion of our project requires us to graph the output of the microgrids onto the web application for others to see and understand. This will require processing the data and displaying it in an easy and accessible format.

4.2 Design Exploration

4.2.1 Design Decisions

1. What web framework to use?
2. What database to use?
3. What graphing framework to use?

4.2.2 Ideation

React		
HTML/Javascript	Web Framework	
Wordpress/Wix/Other online framework	Angular	Vue

- React - JavaScript based frontend framework created by Facebook with simple hooks and well documented
- Angular - JavaScript based frontend framework created by Google with well defined class structure and well documented
- Vue - Independent JavaScript based frontend framework
- HTML/JavaScript - Create the web app using JavaScript and HTML but without a bespoke framework
- Wordpress/Wix/Other online framework - Create the web app with minimal code by using third party website builder

4.2.3 Decision-Making and Trade-Off

Web Framework Decision Matrix

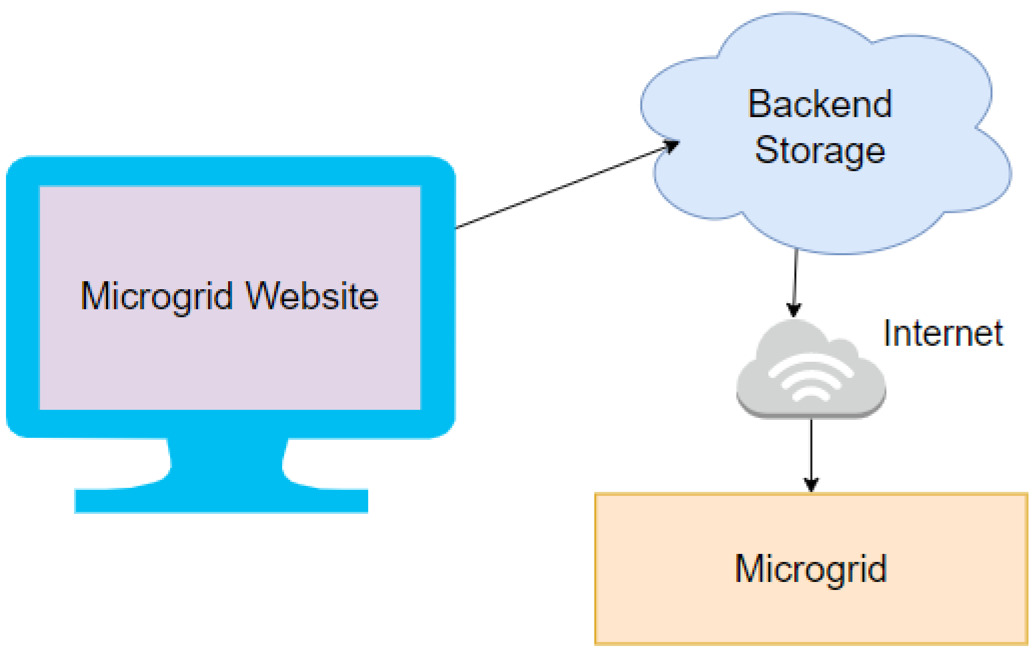
Criteria	Weight	React	Angular	Vue	Pure HTML/JS	Wordpress/ Other website-builder
Ease of Implementation	0.3	4	3	3	1	3
Flexibility/Customizability	0.3	5	5	5	5	1

Prior Experience/Knowledge	0.4	4	4	1	3	1
Totals	1.0	4.3	4	2.8	3	1.6

We chose to use a decision matrix to determine the best design options because of the ability to make a choice based on what we believed were the most important attributes. We chose features that we wanted our ideal framework to include and weighted them based on the impact each would have on the outcome of our project. We wanted to stick with a framework we were familiar with, but also wanted to make sure another option wouldn't be better, so this was able to show us a ranking of each option we considered based on what we valued.

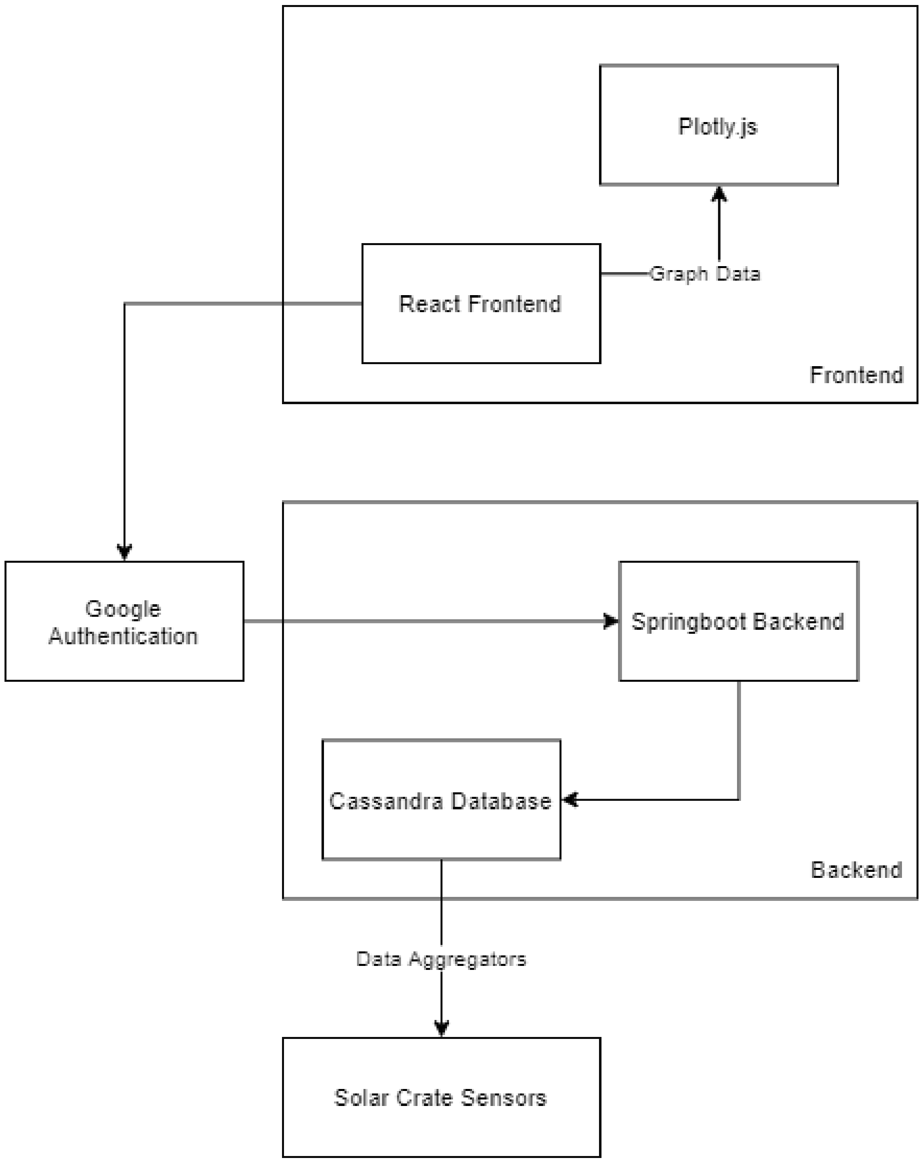
4.3 Proposed Design

4.3.1 Overview



Our web application's design consists of a frontend and backend, along with the physical ISU EPRC Solar Crate. The backend communicates with sensors on the Solar Crate to store information about efficiency and power usage on the cloud, which is then displayed by the frontend of the website for users to see and interact with.

4.3.2 Detailed Design and Visual(s)



We will have a React-based frontend that interacts with a Spring Boot backend. Our data is stored within a Cassandra database, which grabs data from the microgrid via Python script data aggregators.

The React-based frontend will rely on using Plotly.js to display the requested data in graph format. Users on our site will be able to browse as the general public without needing to sign in, or they will be able to sign in as a researcher or an operator. Signing in as these roles will allow them to view restricted information not available to the general public. Upon user request, the frontend will query the backend for information about a particular microgrid and then display that information in graph form using Plotly.js.

The Spring Boot backend will interface between the frontend and the Cassandra database, processing API calls from the frontend to produce a database query that will produce the requested data. The backend will then translate that retrieved raw data into more refined values, depending on what the user requested specifically, and then send the response back to the frontend to be displayed. During this process, the backend will authenticate the user to verify that the requested data is either viewable by the general public, or that the user is an authenticated researcher/operator recognized by our system.

The Cassandra database will store the different types of microgrids connected to the system and their respective data that gets produced. Cassandra is a non-relational, NoSQL database that allows dynamic and flexible schema definitions, so the backend can query for any available data that the microgrid has available and store it in the Cassandra database, regardless of what other data we are/are not storing from other microgrids.

Finally, the Python data aggregators will periodically query for data from the microgrids themselves and then insert the data into the Cassandra database to their respective tables.

4.3.3 Functionality

There are two major user groups, and with that two use cases:

1) General Public

One of the purposes of the solar crate is to raise public awareness of solar microgrids. The public will reach our site either by navigating to it from the Electric Power Research Center's site, or when interacting with the crate while on display. The home page of our site will provide basic information with regards to the crate. There will also be navigable tabs for users to learn more about specifics like live performance data. Information that is accessible to unauthorized users will be kept simple and high level.

2) Researchers/Operators

Another purpose of our site is to assist researchers and operators that regularly interact with the crate. Once authenticated, our site will provide more verbose information collected from the crate. Researchers/Operators with quick questions can use the different dynamic graphs to focus on interesting data. These graphs can also be used for reporting purposes. Authenticated users will also be able to download data to be used offline with other analytical tools.

4.3.4 Areas of Concern and Development

The current design is set up to provide our client with a web frontend they want in order to visualize the data from the microgrid. Our design is ensuring data can be pulled from the grid into a database which our backend will pull from and send to the frontend to create a visualization for our client.

One aspect that concerns us is whether the current backend and database set up is able and sufficient enough to handle all the necessary data and sort them into the necessary categories. We plan to test the backend and database to ensure it works as intended, and write python scripts to address APIS not meeting requirements.

A question we have for the client is what kind of data are they interested in viewing from the microgrid? Which of the information getting pulled form the microgrid would be classified as vital, nice-to-have, and unnecessary?

4.4 Technology Considerations

For the frontend framework our group has chosen to use React. React's strengths include its ease of use and reduced workload compared to a html/javascript solution, its increased flexibility compared to a website building service such as wordpress or wix, and its native support for more complex functionality. One trade-off for this is it requires a bit more experience to use and can be a little more difficult for initial setup, but this is offset by our group already having experience with React.

Our group plans to use Okta for login and authentication. Okta is a secure method for authentication and is easier to use for a project than creating our own login system. It also has the advantage of our app potentially being integrated with Iowa State University's Okta login for easier use by users. One downside to this is our app would need to be approved by ISU for use with their Okta and this is not guaranteed and can't be decided until they can review the code for our application. If this is the case we may need to use an alternative such as building our own login system or using our own instance of Okta.

For the backend framework our group is planning to continue to use and modify as needed the existing Java Spring backend. Java Spring is a flexible and robust framework for providing APIs to the frontend. Working with the existing code also has the advantage of allowing time to be focused on other systems.

Our group is planning to continue using Cassandra for our database as well. Cassandra is a NoSQL database whose strengths include scalability, accessibility, and easier querying, especially with less organized data. Its weaknesses are that its data can be less organized and atomicized, and that it is harder to migrate its data across databases. These tradeoffs make sense for our project since we will have large amounts of data from potentially several microgrids, which may have different types of data to share, and are offset by our use of only a centralized database to increase data atomicity.

Graphing in our project is going to be handled by Plotly.js. Strengths of Plotly include a library for Plotly to work with React and a high degree of customizability in order to support different types of microgrids. One weakness of Plotly is that it is not the fastest graphing library available, however this is offset by its increased customizability and not needing to render multiple graphs simultaneously, or a large number of datapoints.

4.5 Design Analysis

For future implementations, we will be testing the current APIs built in the backend and write/adjust them to meet client expectations. While working on the backend, we will begin working on a general frontend and graph outlay which will be fully integrated with the backend once it is complete.

5 Testing

5.1 Unit Testing

- Aggregators
  - We will utilize unittest, a Python unit testing framework for testing our Python data aggregators. unittest will allow us to mock and fake calls to the microgrid in order to test functionality of the aggregators themselves and how they are prepared to be stored into the database.
- Backend API
  - We will be unit testing our backend APIs that our frontend will interact with. We will want to test underlying functionality of the microgrid data processing without interacting with the database level or the frontend calls. We will utilize JUnit and Mockito for mocking and testing the retrieved microgrid data and transforming it into readable and presentable data based on specific search parameters and filters.
- Frontend/React
  - We will be unit testing our React components using a combination of React Testing Library and Jest. Jest will allow us to mock React components by rendering them without their child components. React Testing Library will allow us to test the functionality of the components themselves. This will be useful when mocking out and testing our graphing components so they have all the correct information being included and displayed when provided certain properties related to filtering and authorized account type.

5.2 Interface Testing

- Frontend/React to Backend API
  - To test that the frontend of the web application is able to interact with the backend API we can use Postman to simulate the frontend to query the backend to test the response from the backend API. We can also use postman to stand in as a backend for the frontend web application to query in order to test that the queries are being formulated correctly.

5.3 Integration Testing

- Frontend to Backend API
  - We will use JUnit and Spring Boot's MockMVC to test the backend API calls from the frontend. MockMVC will allow us to simulate the calls to ensure that the frontend is properly connected to the backend and displays the appropriate data.
- Backend API to Database
  - We will use JUnit and Mockito to test the backend web server's interaction with the database. The database calls will be mocked with Mockito, which will ensure the backend can access and interact with the database.
- Aggregators to Database

- We will use the Python testing framework unittest to test that the aggregators can access and interact with the database. We will verify that data can be properly input into the database following grabbing data from the microgrids.

5.4 System Testing

- Aggregation System Testing
    - Our System testing will involve testing the frequency of the aggregators accessing the microgrid and then storing it into the database. This will primarily involve the unit tests and integration tests specifically related to the Python data aggregators in conjunction with the database.
- Full-stack System Testing
    - Our system testing will also involve testing the interaction of grabbing various specific data related to different components of the microgrid stored within our database, processing it in the backend, and then displaying the desired information on our frontend. Our system testing will not specifically test the visual aspects of the data but will verify that the expected data has made its way to the frontend from the database.

5.5 Regression Testing

For regression testing we plan on configuring the CI/CD pipeline within GitLab to run the existing unit tests and ensure that they continue to function as expected. Our project doesn't have any critical features however having components such as the backend API offline could create problems for other components so this regression testing will allow us to only build versions of the software that are working.

5.6 Acceptance Testing

We will hold meetings with our advisor every other week to showcase our finished implementation of the design and functionality as it appears on the frontend. This time will be used to demo new features or functionality that we have added to the application in terms of processing microgrid data, applying certain filtering, applying various permissions based on public users versus authorized users, or displaying specific graphs for different types of microgrid data. In addition, this time will be used to verify the design and styling of the application is pleasing and satisfactory to our client.

5.7 Security Testing

With security being a big part of the initial ask for this project, performing proper security testing is a high priority. We are tasked with assessing the current security posture of the Solar Crate and based on our findings refining the security. As such, we will split security testing into two groups: Initial and Ongoing.

5.7.1 Initial Security Testing

In order to assess the initial security of the Solar Crate, we will use a combination of provided documentation and open source tools. The provided documentation will inform us on what assets exist so we ensure full coverage of the tests. We will then perform the actual testing using an Operating System called Kali. Kali has a bunch of open source tools that are useful for tasks like scanning networks and testing web applications. Results from the initial security test will be compiled into a report for the client.

5.7.2 Ongoing Security Testing

Security is an ongoing process, and as such it is necessary to consider security as we build out new features. Ideally security flaws will be caught by individual expertise and open communication while a product is being created. However, this is not always the case. In order to ensure a secure final product, our group will make use of a peer review system. Before a new feature is pushed out to production, at minimum one other group member must review it for both functionality and security.

5.8 Results

Given that a significant portion of our project involved guaranteeing feature parity with the mobile app, regression testing was a significant aspect of our testing plans. The primary feature we had to guarantee parity for was the graphing of the live data. Given the graphing by nature is a visual aspect, there was no simple way to automatically test this. As such, our goal was to build upon the graphing capabilities present in the mobile application and improve them using plotly.js in our web application. In our Plotly graph, we labeled out different time frames that you could graph, extending from a few minutes ago to viewing all-time data, as well as an export button to download the data. Although there are performance improvements that can be made in the future, this is matching and improving upon the mobile application's feature of viewing live data (within the past several minutes) and exporting a month of data at a time.

In addition, throughout the development process this semester we continually updated one of the database tables that had been created by the prior team. Throughout each feature we verified that the endpoints and services provided by the prior team had not been disrupted or broken by any of our changes.

Integration testing our application took the form of verifying the various backend API endpoints were accessible from Postman and returned the correct results from the database. This also involved verifying that any users had proper verification when accessing the backend, so that only users who had the proper privileges were able to access certain data from the server. This involved storing a Google Authentication Token as an environment variable in Postman, sending it in an Authorization header in each API request, and verifying the token through Google on the backend. These processes tested our backend to frontend integration as well as our backend integration with Google Authentication.

Security testing was successful on both the initial and ongoing fronts. There were three major findings and four recommendations from our initial testing efforts (See Appendix II). We actually used these findings to feed into our ongoing testing efforts. When resolving the issues found, we had the expert in that given area get hands-on experience with the solution. This helped prevent siloing of cybersecurity information, as well as helping prevent recurrences of the particular issue.

Every other week when we met with our advisor Mat we presented the changes we had implemented that week. Our advisor, who was also representing our client, provided us with acceptance testing to verify that the features we were improving and implementing were aligned with what he and the client had in mind.

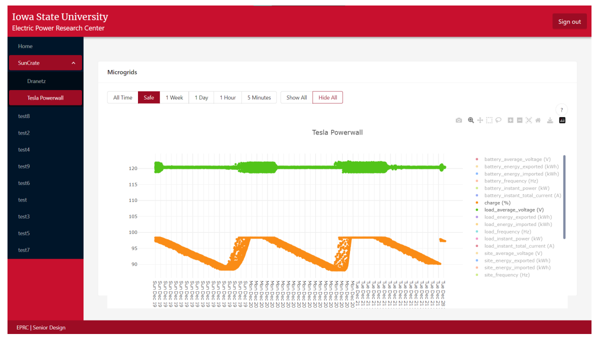
Utilizing these various testing processes, we have confirmed that we have satisfied our client's requests and requirements for this project.

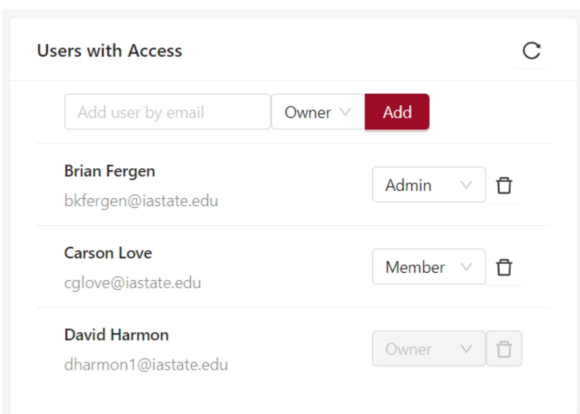
6 Implementation

Our implementation can be broken down into the following three major objectives: design a web application, evaluate and strengthen the cybersecurity posture, and generalize the data collection to allow for more data sources. The sections below outline the high level outcomes of implementing each section of this project.

6.1 Web Application

One key feature of this project was moving the application over from a mobile application to a web application. We utilized React as our framework for building our application, and built out the graphing capabilities with a third party library called plotly.js. Most of the process surrounding this involved guaranteeing feature parity between the sites and improving functionality.





Given that one of our other goals was to improve security, we added in support for user authentication and permission roles to accessing various parts of the site. We implemented a member role, an admin role, and an owner role to correspond with what level of permission a user had with a site. These permissions were granted on a site-by-site basis, so that any given user could be an owner of one site and also a member of another site.

Members only had read-access to sites. This meant that if a user was added a member to a site, they were able to view all of the graphs and associated data with the site, as well as any other users who have access to the site and their respective roles. Admins had one level of permissions above them, where admins were allowed to add new members to a site. Owners had unrestricted access over the site, and as such were able to add any user of any role and change the roles of any other user. This meant that if an owner wanted to promote another user to owner status, they could. In addition, any owner can demote or remove any other owners from the site.

Finally, we also implemented the ability to make sites public or private. If they were public, than any user who logged in to a Google account was able to bookmark that site and view the graphs and data associated with the site. If the site was private, then an admin or owner would have to manually add any users if they wanted to provide them with viewing access.

6.2 Cybersecurity

As outlined in the Security Testing section, one of the first major cybersecurity objectives was to gain some visibility on the current security posture. As our investigation included testing live systems on Iowa State's network, we had to get the proper permissions in the form of a Statement of Work (SoW). From there we tested the VM and solar crate using techniques outlined in 5.7.1. Furthermore, we dug into the documentation left by the prior senior design team along with their git repo. Based on this work, we had 3 major findings and 4 major recommendations to make to the client (See Appendix II).

After presenting our findings to the client, we got the go ahead to implement some of the recommendations made. We started by cleaning up secrets we found on the old and new git repo. To prevent these from ending up on the github again, these items were moved to a configuration file which is ignored by git. A template configuration file was also created to allow for new developers to set up their environment easily.

Implementing a form of backend API verification was another important aspect of improving the security posture, as the backend API endpoints implemented by the prior team had no form of authentication or verification which prevented bad actors from accessing private information on sites. To remedy this, we utilized Google Authentication to pass a credential token to the backend API to verify the access permissions of each role. Not only does this work in conjunction with the access roles discussed in section 6.1, but it also prevents attackers from accessing private data from the database.

Some of the recommended changes were not possible due to the Solar Crate being undeployed for the semester. For these changes, we created documents that step through what is needed to make these changes. These documents have been made available to the client, and will be passed along to any future senior design groups.

6.3 Generalized Data Collection

There are a changing number of sensors on the microgrid, so it is important that we are aware of what sensors are available and how each of them work. The active sensors include Tesla powerwall, Tesla Gateway, Dranetz, and temperature, humidity, and irradiance sensors from a HOBO data logger. The main sensors our project focused on were the Tesla sensors and the Dranetz. Data from these sensors were already stored on the existing backend. We did research to better understand how the sensor API's work, and used these to write our API's to better pull data from the sensors and increase simplicity when storing data to the backend. We were intentional in sending data to the backend using specific elements on a json file, so the backend only needed one API to pull data from the backend. Due to the varying data being collected from each sensor, and the varying credentials that are needed to allow access to different sensors, it made it difficult to write one generic script to pull data from every sensor. Hence, for any new sensors added to the microgrid, a new script would need to be written, but the data should be stored into the backend using the same json file elements as the previous API's so no further work needs to be done on the backend. The Tesla Gateway and Powerwall sensors are wired together for serial communication between them so only one API was needed to collect data. We ran tests to ensure the data was shared between both sensors and we found that to be true.

7 Professional Responsibility

This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 Areas of Responsibility

Table 1. The seven areas of professional responsibility in the assessment instrument

Area of responsibility	Definition	NSPE Canon
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts.
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.
Communication Honesty	Report work truthfully, without deception, and understandable to stakeholders.	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.
Health, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.
Sustainability	Protect environment and natural resources locally and globally.	
Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.

Area of Responsibility	Code of Ethic: "Seek, accept, and offer honest criticism"
Work Competence	This code will be useful in ensuring that we are putting forth our best work, and getting feedback on areas we can improve on
Financial Responsibility	Offering our honest opinion is a guide to make sure we are not exploiting our stakeholders
Communication Honesty	This code ensure that we are honest in our communication, and make others feel safe sharing their opinions
Health, Safety, Well-Being	Providing honest critics includes giving heads up and warnings where necessary if we notice something might become a health or safety hazard
Property Management	This code guides us to call out and be honest about any acts that might lead to destruction of property
Sustainability	Seeking and accepting criticism can help us

	find and implement more sustainable methods in our project
Social Responsibility	This code calls us to be honest and seek feedback, which can entail learning how to account better to our social surroundings

7.2 Project Specific Professional Responsibility Areas

Area of Responsibility	Importance Level	Current Performance
Work Competence	High. It is important for us to work to our strengths to achieve our goals	High. We've been working well together to each other's strengths to get our different tasks done
Financial Responsibility	Low. Our project does not involve any financial commitments	Low. We are not expected to handle any finances
Communication Honesty	High. We need to communicate properly and be on the same page if we want to succeed	High. We talk a well with each other and communicate what is expected of one another
Health, Safety, Well-Being	Low. Our project does not create any health and safety hazards	Low. We haven't done anything so far that might cause a safety hazard
Property Ownership	Medium. We are responsible for the microgrid, and the data stored on it	Low. We haven't done anything that could jeopardize the grid or its data
Sustainability	Medium. We will need to use the microgrid carefully to preserve it for future use	Low. We haven't done anything with the microgrid yet that could jeopardize it
Social Responsibility	Medium. We will need to ensure that our product has good societal implications	Medium. We haven't worked on anything yet to hinder or improve our products societal impact

7.3 Most Applicable Professional Responsibility Area

Work Competence is our most applicable area. We would like to complete the project we have in a timely manner, so focusing on our strengths will be important to maximize our work performance.

8 Closing Material

8.1 Discussion

We met the three main requirements for this project, along with the stretch goal of allowing for multiple microgrids to be added to the system. The new web application achieved feature parity with the existing mobile application, along with the addition of user types and permissions. An assessment of the cybersecurity posture was made, and recommendations were implemented to improve it.

8.2 Conclusion

Our goals with this project were to create a web application to show real time and historical information for the main microgrid, allow administrators of the microgrids to change settings, and to increase the cybersecurity of the existing systems and the web application. A stretch goal was to allow for multiple microgrids to be added to the system and have their data visible. Everything we planned to do for this project was accomplished and we were able to complete the stretch goal as well. We had a thorough design from last semester which prepared us for the implementation process and allowed us to finish everything on time.

8.3 References

"Important Engineering Software/Hardware Design Standards." Software/Hardware Design Standards, 2012, http://users.encs.concordia.ca/~ecewebdv/EDS/Software/std_list.htm.

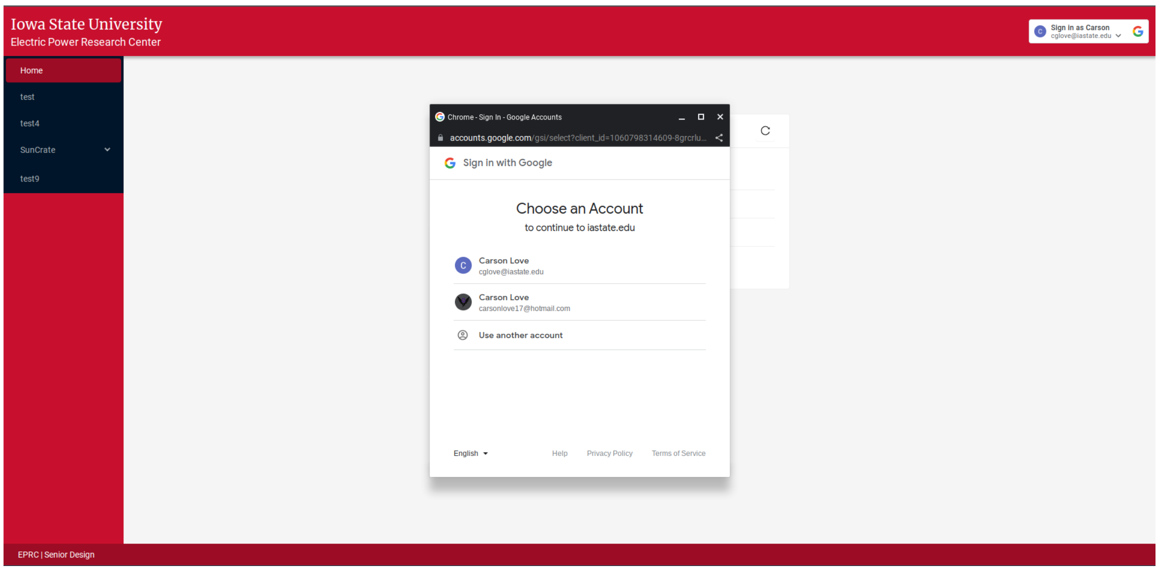
"App For Microgrid Demonstration Project." sddec21, 2021, https://sddec21-21.sd.ece.iastate.edu/.

"Digital Standards." *Iowa State University*, 20 Nov. 2018, https://www.brandmarketing.iastate.edu/digital-standards/.

Appendix I - Operator's Manual

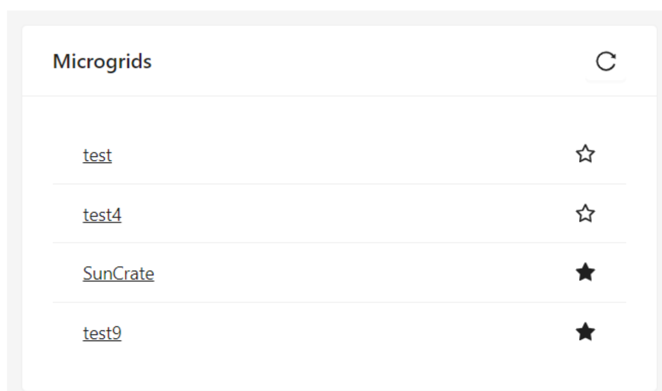
A1.1 - How to Log In

The first thing a user has to do to access the site is to log in using Google Authentication. Clicking the sign in button on the top right of the screen will bring up the following screen that allows the user to sign in using their email address.



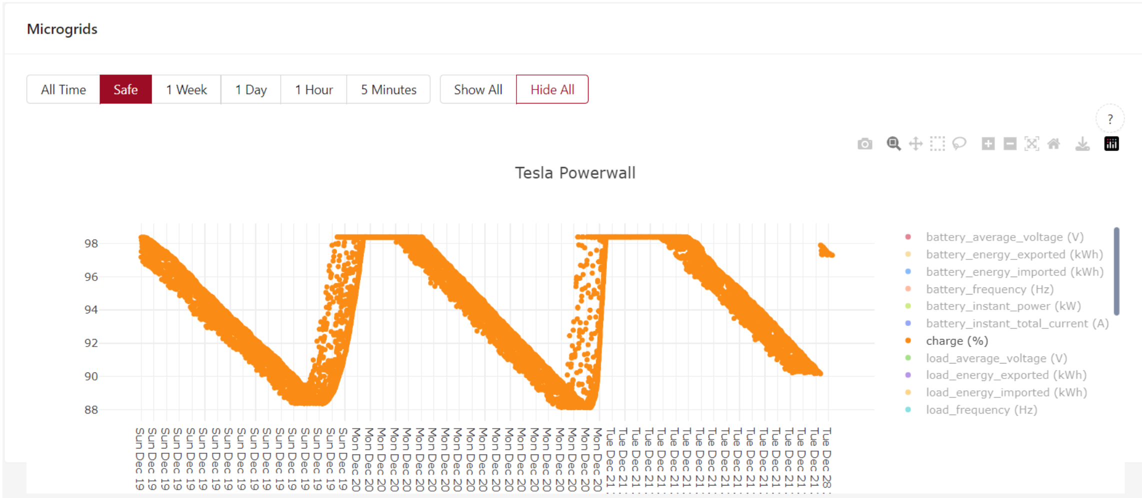
A1.2 - Bookmarks & View Sites

The opening screen to the application displays a list of public microgrid sites. Users are able to bookmark sites they are interested in by selecting the star icon, which will save them on the sidebar for the user to visit.



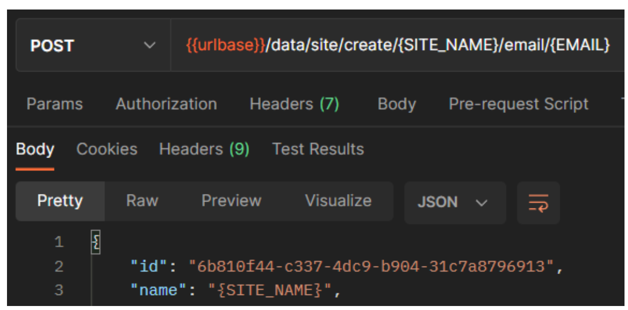
A1.3 - Navigate Graph and Export

To view the data for a given site, simply select a site from the list on the left hand menu. If there are no sites listed, please make sure to bookmark a site (as described above) or have a site administrator add you to their site. Then, select one of the sensors from the dropdown that appears below the site name. If there are no sensors listed, follow the steps in A1.4 and A1.5 to add a site and add sensors to the site.



Now that the sensor graph is shown, select any of the time ranges from the menu at the top, and then select which measurements you would like to view. Click the download button to download the displayed data as a csv. There are a variety of other functions you can use to view the data, including click and dragging a selection-box on the graph to zoom into the data.

A1.4 - Add New Sites

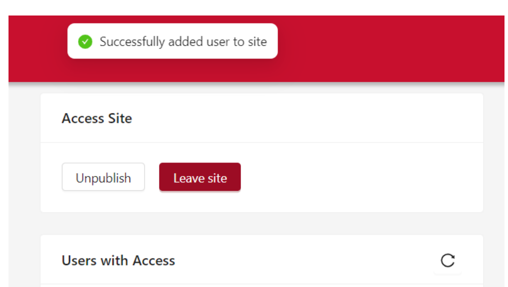


Adding new sites is currently an administrative feature, as we do not want users to be able to create any site they wish, since they will have to work with Iowa State University to get the sensors added to the site (see A1.5). System administrators can add new sites for users by accessing the endpoint "`/data/site/create/{SITE_NAME}/email/{EMAIL}`" with an https request. This endpoint will create a new site named `{SITE_NAME}` and assign the user with the email address of `{EMAIL}` as the owner. The recommended way of accomplishing this is by using Postman (pictured above).

It is worth noting that the email must be attached to a user who has logged into the application at least once before.

A1.5 - Add Members to Site

The individual site homepage displays a list of users the site is shared with along with their roles. There are three roles a user can take on for each site: Owner, Admin, and Member. An Owner has full control of the site and can promote Members to Admins. Admins can add users to the site and Members can only view the data.



A user can be added to the site by entering their email address into the field, selecting a user role, and clicking the Add button. Any user that wants a site shared with them has to have signed into the website previously in order for their email to be saved in the database.

Once added, Owners can change the roles of shared users by selecting the desired role from the dropdown next to each name.

A1.6 - Deploy to VM

To initially set up the application, begin by cloning the app. Copy the certificate to a folder in `Backend/microgrid_app/` (we recommend `Backend/microgrid_app/src/main/resources/keystore/`). Rename the `Backend/microgrid_app/src/main/resources/application.properties.example` to `Backend/microgrid_app/src/main/resources/application.properties` and add the path to the keystore where your SSL certificate is stored for Spring Boot to it. Next, copy the certificate and key into a folder within the `Frontend/microgrid-app/` folder (we recommend storing them in `Frontend/microgrid-app/.cert/`,) and configure the React application to point to your certificate and key in the `Frontend/microgrid-app/package.json` on the `scripts.start` line.

Once you have set up the application, you can deploy it (or redeploy it after pulling changes) using these steps. Start by CDing into `Backend/microgrid_app/` and run `./mvnw clean install`. Then CD into `Backend/docker/` and run `docker compose up --build --detach`.

A1.7 - Collecting data from Sensors to Backend

When writing scripts to pull data from the sensors and store in the backend, a json file format is used with specific elements for simplicity.

`‘siteId’,` is used to pass the site ID for the sensor

`‘datasourceName’` is used to pass information about the sensor like the address, username and password

`‘timestamp’` is used to note the time the data was sent to the backend

`‘data’` is used to send the different values of the information gotten from the sensor to the backend.

The siteID and datasource information should be gotten from the admins in charge of the microgrid.

Appendix II - Cybersecurity Findings & Recommendations Report

Beginning on the next page is the Findings and Recommendations report presented to the client on March 23, 2023.

Findings and Recommendations

sdmay23-37 - March 23, 2023

Summary

Finding	Recommendation(s)
F01 - Githubs Contain Sensitive Data	- Scrub sensitive data from both repos - Store sensitive data in ignored config files
F02 - Solar Crate Assets Open to Whole Internet	- Tighten firewall rules to only allow connections originating from ISU network
F03 - Limited Visibility Over Solar Crate Assets	- Set up log collection and feed to an IDS (Intrusion Detection System)

Findings

Finding	Description	Risk
F01 - Githubs Contain Sensitive Data	Both the current and prior senior design teams utilize an ISU managed gitlab to store the project's code. Code stored within both repos contains sensitive information such as API keys, emails, and user/password combos. It is trivial for individuals to query the gitlab and find this information. Luckily, these repos are only accessible on the ISU network, by authenticated users. Gitlab Links: https://git.ece.iastate.edu/sd/sddec21-21 https://git.ece.iastate.edu/sd/sdmay23-37	Medium - This is information that should not be publicly available, especially since some of it is still up to date. This is not a high risk only because of the partially limited access to the repos.
F02 - Solar Crate Assets Open to Whole Internet	The Solar Crate devices are accessible to the public internet. These are exposed web and ssh services, which accept connections from anywhere. This provides an attack surface for even trivial attacks like password guessing.	Low - These services must be exposed in some manner to be usable. Theoretically they are hardened by the vendors that supplied them. Even still it is good practice to only expose the minimum attack surface.
F03 - Limited Visibility Over Solar Crate Assets	The Solar Crate is an off-premise group of devices. Currently, if these devices were compromised, we would have no way of knowing. Furthermore, we have no idea if/what attempted attacks the Solar Crate is seeing. Being accessible to the public internet, it is expected they are seeing at least some rudimentary attacks.	Medium - Lack of visibility is always a bad thing when it comes to managing risk. This is made worse by the fact these devices are accessible to the public internet.

Recommendations

Recommendation	Description	Risks
R01.1 - Scrub Sensitive Information from Githubs	Since some of the sensitive information exposed is still in use, it would be wise to censor it. First we will need to identify all information to be scrubbed. Then we will use github's tools to remove the target lines/files from the repo history.	**Low:** - Loss of code due to aggressive scrubbing - Loss of functionality due to missing/moved credentials
R01.2 - Store Sensitive Information in Ignored Configuration File	Sensitive information will be placed in configuration files, which we will tell github to ignore. Example configuration files will be uploaded to github for users to modify when deploying.	**Low:** - Loss of functionality due to missing/moved credentials
R02 - Tighten Solar Crate Firewall to Only Accept ISU Network Connections	Use network and application firewalls on the Solar Crate to only accept connections that originate from ISU's network.	**High:** - Loss of access to Solar Crate due to misconfigured firewall - Updates from vendor fail due to aggressive firewall rules - Uncertainty about business needs of Solar Crate access
R03 - Log Collection and Intrusion Detection System	Logs from sensors, network devices, etc will be collected on the ETG VM. Said logs will then be fed into an IDS. From there events can be examined manually, and email alerts can be set up if requested.	**Low:** - Logs must cross the public internet (must be encrypted) - ISU IT might not like that traffic

Appendix III - Team Contract

Team Members:

1) Alexander Haack
2) Carson Love
3) David Harmon
4) Harvey Forchu
5) Kenyon Fergen

Team Procedures

1.Day, time, and location (face-to-face or virtual) for regular team meetings:

Wednesdays at 4:15PM in Coover (or asynchronously)

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

Discord + Email

3. Decision-making policy (e.g., consensus, majority vote):

Consensus

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Individual notes when talking with the advisor/client.

We will take turns tracking meeting minutes.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

Attendance and punctuality are expected unless notified otherwise ahead of time.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Everyone is expected to put in their best effort.

3. Expected level of communication with other team members:

Communicate regularly with the rest of the team when needing help. Allow 24 hours response time. Respond within 24 hours.

4. Expected level of commitment to team decisions and tasks:

Everyone is expected to put in their best effort.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

David: Software

Xander: Software

Kenyon: Software/Cyber Security

Carson: Cyber Security

Harvey: Computer

2. Strategies for supporting and guiding the work of all team members:

Discuss what needs to be done, and what steps need to be taken during team meetings, sharing the workload and assisting each other where need be.

3. Strategies for recognizing the contributions of all team members:

Discuss individual progress made during team meetings.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

David: React and Fullstack experience

Kenyon: React and Fullstack experience

Xander: Fullstack and embedded

Carson: Threat Assessment and Security Controls

Harvey: React, backend experience

2. Strategies for encouraging and support contributions and ideas from all team members:

Go around and ask for input from each team member for their opinion/ideas on a problem/solution for a problem.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

Bring up issues as soon as they arise so the team can adapt.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

A grade, gain experience working with other majors. Get a better understanding of the microgrid and how it holds/transmits data.

2. Strategies for planning and assigning individual and team work:

Work together in meetings to plan out and distribute the work evenly and based on each individual's interests/expertise.

3. Strategies for keeping on task:

Check in with each other during meetings to ensure that deadlines are met.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

Talk to the team member and reason out why said infraction is happening.

2. What will your team do if the infractions continue?

Bring it to the notice of the project advisor, and then the professor if the individual is not willing to change.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) ______________Harvey Forchu_______________________ DATE _____04/28/2023____

2) ___________Alexander Haack________________________ DATE _____04/28/2023____

3) ___________Kenyon Fergen_________________________ DATE ______04/28/2023____

4) ___________Carson Love___________________________ DATE ______04/28/2023____

5) ___________David Harmon_________________________ DATE ______04/28/2023____

Appendix IV - Images

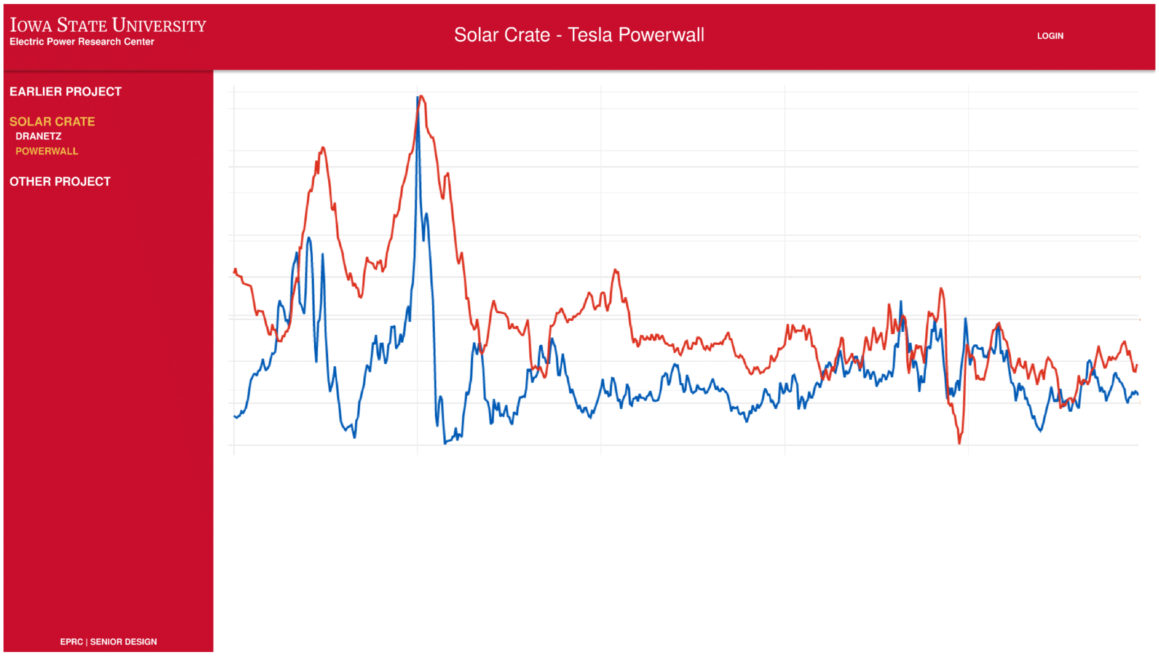


Figure 1 - Figma Prototype

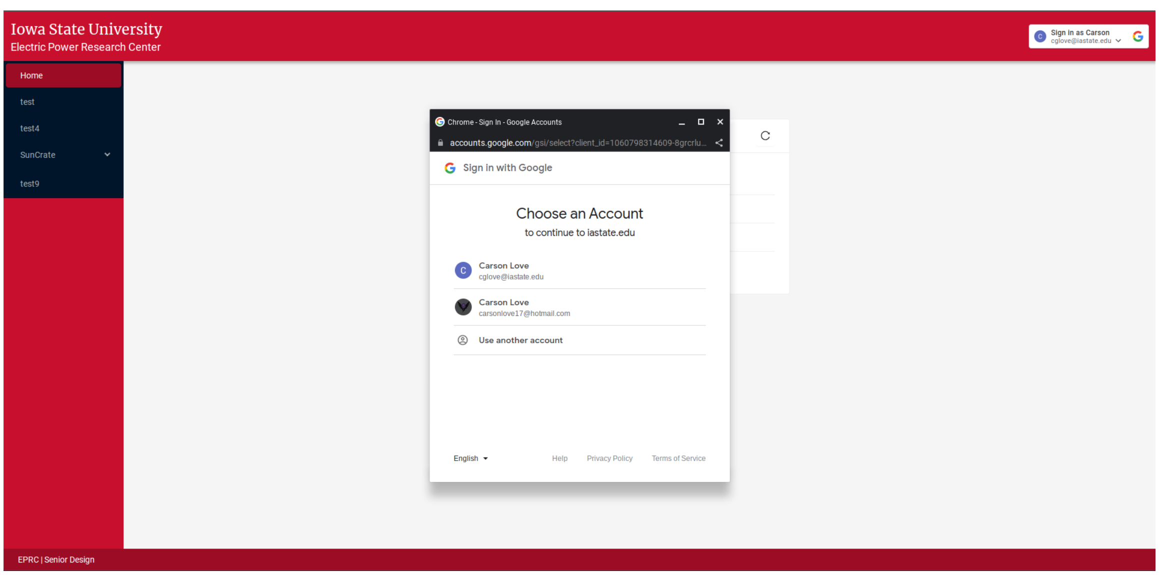


Figure 2 - Login Page

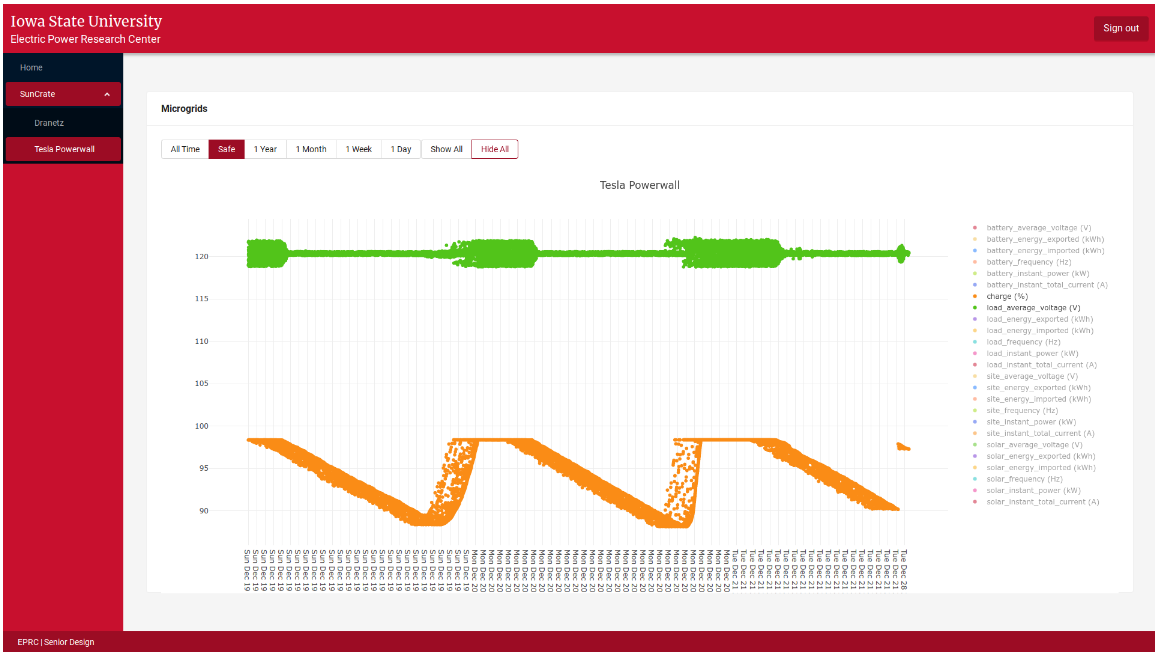


Figure 3 - Sensor Graph Page