NAU Energy Dashboard:

Design Document Version 1.0

Team: Save Watt

February 25, 2019

Sponsored by:

Jonathan Heitzinger & Dr. Truong Nghiem

Faculty Mentor:

Isaac Shaffer

Team members:

Madison Boman, Hyungi Choi,

Ian Dale, Brandon Thomas

This page was intentionally left blank.

Table of Contents

1. Introduction

1.1 Project Motivation

In the advent of alarming reports about climate change, there has been an ever-growing effort in favor of energy conservation. Climate change is a genuine concern for the future of our planet. Northern Arizona University (NAU) has always done its part to reduce waste, and promote a greener future. NAU is looking to take further action to improve sustainability by leveraging data collection and analysis.

For the last several years, Northern Arizona University has been collecting a trove of raw data related to the operation and energy consumption of its buildings. As of now, there is not a comprehensive way to analyze NAU's energy data. Current methods are cumbersome and require a great deal of technical expertise. There are four major issues with the existing process that our sponsors would like to solve:

- Manually retrieving data from sensors and the current database is time-consuming. Automatic retrieval of the desired data sets is preferred.
- Current tools do not allow for interactive graphs which show trends over customizable time frames.
- Broad statistics must be run using external tools. Some automatic statistical analysis is preferred.
- Exporting data to mutable file types is very difficult.

Our sponsors are looking to make the most of NAU's energy data. This project is sponsored by Jonathan Heitzinger and Dr. Truong Nghiem. Together they are looking to develop a web-based system which analyzes and elegantly presents NAU's building operation data. They hope to create a product that impacts campus-wide sustainability. To facilitate the collection and analyzation of NAU's energy data, we are developing the NAU Energy Dashboard.

1.2 Solution Overview

The NAU Energy Dashboard is a comprehensive web-based application which can retrieve, graphically display, export, and run basic analytics on NAU's collected data. We plan to simplify the existing process with a tool that abstracts away the complexities that stifle current research efforts. Our users would like the NAU Energy Dashboard to:

- Automatically handle building operation data retrieval.
- Graphically display data in a way that is interactive.
- Run applicable broad statistics on a wide range of data points.
- Cleanly export mutable CSV files for additional analysis.

This solution eliminates the need for sifting through endless data, presenting data using external tools, and using specialized tools for fundamental statistical analyses.

1.3 Key Requirements

To make our sponsors' vision come to life, we conducted an in-depth process of acquiring domain knowledge and requirements. We concluded with the following list of functional and performance requirements:

Functional Requirements

- Historical Data Retrieval
- Live Data Retrieval
- Statistical Analysis
- Graphical User Interface
- Data Presentation
- Data Export
- Administrative Tools

Performance Requirements

- Maintainability
- Security
- Performance
- Usability

These requirements will guide our design and implementation plans as we continue to develop the NAU Energy Dashboard. Along with these requirements, we must consider one key constraint. The NAU Energy Dashboard must run on NAU's network. This means that our system must be able to run on a Linux (REHL7) environment and all technologies must be found in a YUM repository. This is essential for our system to continue to function properly when blanket system updates occur.

1.4 Summary

We are using the requirements and constraints mentioned above to guide our development to accomplish our sponsors' goals. In the following sections, we outline our plans for implementation and describe our architecture in detail.

2. Implementation Overview

2.1 Implementation Tools

To implement this project, we have selected several tools that facilitate our design. Since we are building a web-based application, we have chosen to use the Django 2.1.5 framework. Django is a free, open source, high-level Python-based web framework that influences each of our design decisions and architectural structure. Django provides a basis for our package structure and guides our development toward an object-oriented design pattern. Django allows for simple integration with Python, HTML, CSS, and JavaScript. This makes developing a dynamic web application intuitive. Finally, Django integrates well with several database management systems. (MYSQL, MSSQL, SQLite, etc.) This simplifies the process of connecting to an existing database and maintaining a back-end database.

Python is the primary language used to implement the NAU Energy Dashboard. Python is a powerful multi-paradigm programming language that facilitates object-oriented programming. Using python, we can access robust math packages (Numpy and

PANDAS). These packages allow us to manipulate data using high-level data structures and mathematical formulas. Django's dialect of Python allows us to build dynamic HTML-based templates. This makes it possible to dynamically generate pages for each building, which reduces redundancy.

HTML, CSS, and JavaScript are the primary tools used to create the pages that make up the NAU Energy Dashboard. Using JavaScript, we can create dynamic charts using Charts.js. Charts.js integrates easily with Django, facilitating quick data transfer using JSON files.

To connect to building controllers we will be using tools compatible with the current system. NAU uses a system a system of controllers called Jace and an interface called Alerton to collect building data and create trend logs. This system uses BACnet, MSTP, and NiagaraAX to transfer data between sensors and to a database. In order for our system to connect to this system, we will be using third party tools such as DGLux alongside the NiagaraAX language to query the system.

2.2 Summary

To implement this project, we are using Django, Python, HTML, CSS, and JavaScript. These tools provide the robust functionality and intuitive integration needed for development. With these tools in mind, we begin to discuss our architecture as a whole.

3. Architectural Overview

3.1 Architectural Diagram

Figure 1 shows a diagram of the general architecture of the system. The diagram follows the guidelines for a level 2 C4 model. This means that it shows each of the containers that make up the NAU Energy Dashboard. In this section, we use 'module,' 'container,' and 'component' interchangeably to mean a part of the system. The main components are:

- The GUI Module
- The Data Presentation Module
- The Data Export Module
- The Statistical Analysis Module
- The Data Cleaning Module
- The Data Retrieval Module
- The Back-End Database

Our architecture embodies the component-based development style. This allows us to separate functionality by utilizing loosely coupled modules. These modules work together to perform the functional requirements of the system. Now that we have laid out the high-level structure of our system, we begin to discuss each component.

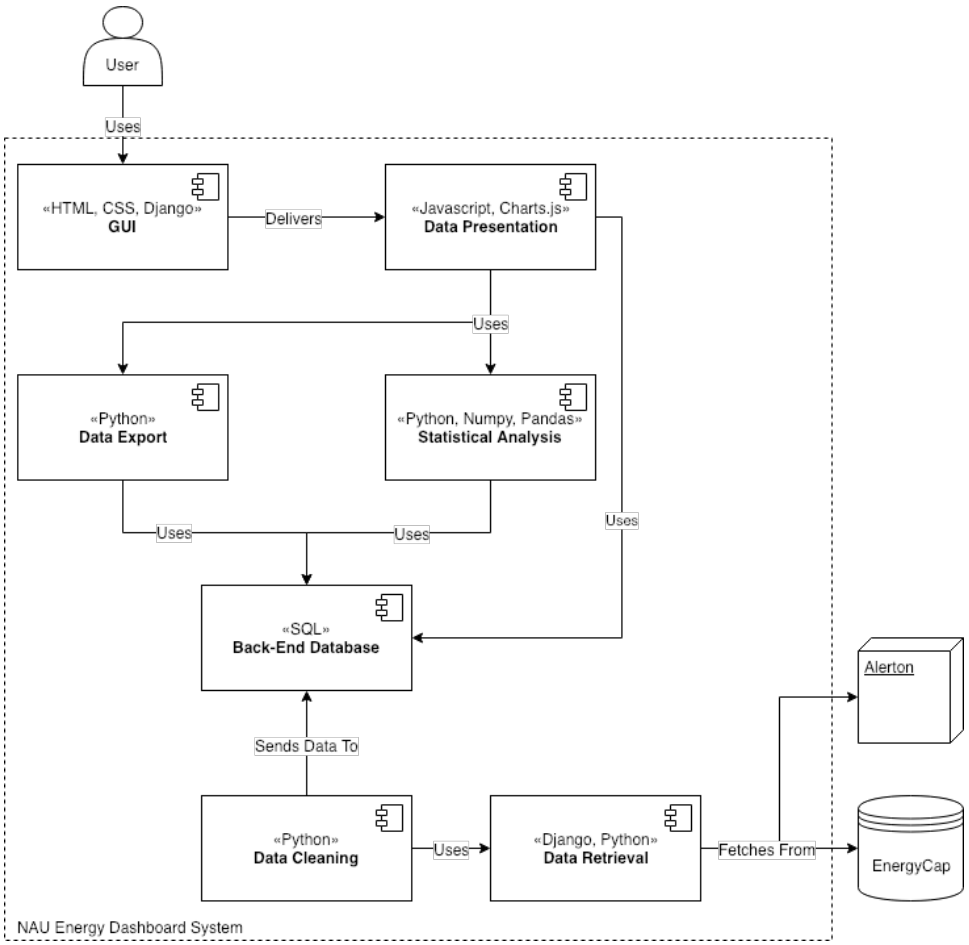


Figure 1. Architectural Diagram. In the style of a C4 model container diagram.

3.2 Component Overview

3.2.1 The GUI Module

The GUI module is responsible for presenting each page to the user. This component provides the user access to all the functionality of our system. The graphical user interface consists of all static pages written with HTML, CSS, and Django. All of these pages are generated on the server side. The GUI module is highly integrated with the data presentation module. These modules work together to create what the user sees and manipulates. The fundamental difference between this and the data presentation module lies in where each is processed.

3.2.2 The Data Presentation Module

The data presentation module is delivered to the user via the GUI module. This means that any dynamic data (charts, statistics, live data) is sent to the user and processed by the user's browser. This component is made up of scripts and API calls which create charts and present live data. This module uses the data export module, statistical analysis module, and back-end database components to generate CSV files, present statistics, and graph historical data.

3.2.3 The Data Export Module
The data export module is called upon by the data presentation module to generate CSV files. This module reads from the back-end database to find the requested tables. It then formats the tables and aligns data points to create a coherent CSV file. The file is sent to the user to download.

3.2.4 The Statistical Analysis Module
The statistical analysis module is called upon by the data presentation module to run relevant statistical tests. These tests include 5-point summaries, heating/cooling degrees, averages, regressions, correlations, and conversions. This component reads from the back-end database to return aggregate statistics.

3.2.5 The Data Retrieval Module
The data retrieval module periodically queries NAU's existing database (EnergyCap) and makes API calls to NAU's building sensors (Alerton). Once the data is fetched, it is converted to an easily manipulated format (JSON) in preparation for data cleaning and error checking. This module feeds our system the data it needs to present. This module keeps our Back-End Database updated.

3.2.6 The Data Cleaning Module
The data cleaning module takes the raw data retrieved from EnergyCap and Alerton and looks for anomalies. It detects corrupted, missing, and outlying data and flags it before storing it in our database. This module ensures that the data in our database is viable or that it is properly dealt with if flagged. This is the only component that can populate the back-end database as any entry must be error checked.

3.2.7 The Back-End Database
The back-end database is responsible for storing all historical data used by our system. This allows for quick local queries that ensure our system runs efficiently. This database also stores user data. User data is used when managing permissions and system settings. Aside from historical data and user data, the back-end database stores building data, controller data, and sensor data that serve as identifiers. This allows us to organize the data coming from EnergyCap in a way that is intuitive and efficient for queries.

3.3 Summary
The main components of the NAU Energy Dashboard's architecture are the GUI module, the data presentation module, the data export module, the statistical analysis module, the data cleaning module, the data retrieval module, and the back-end database. These components form the basis of the system's functionality. In the following section, we describe each module's underlying structure in detail.

4. Module and Interface Descriptions

4.1 Introduction

After discussing a brief overview of each module, we begin to outline the implementation details of our components. In this section, we use UML diagrams to discuss the intricate functionality of each of our modules. Figure 2 shows a UML diagram of the system as a whole.

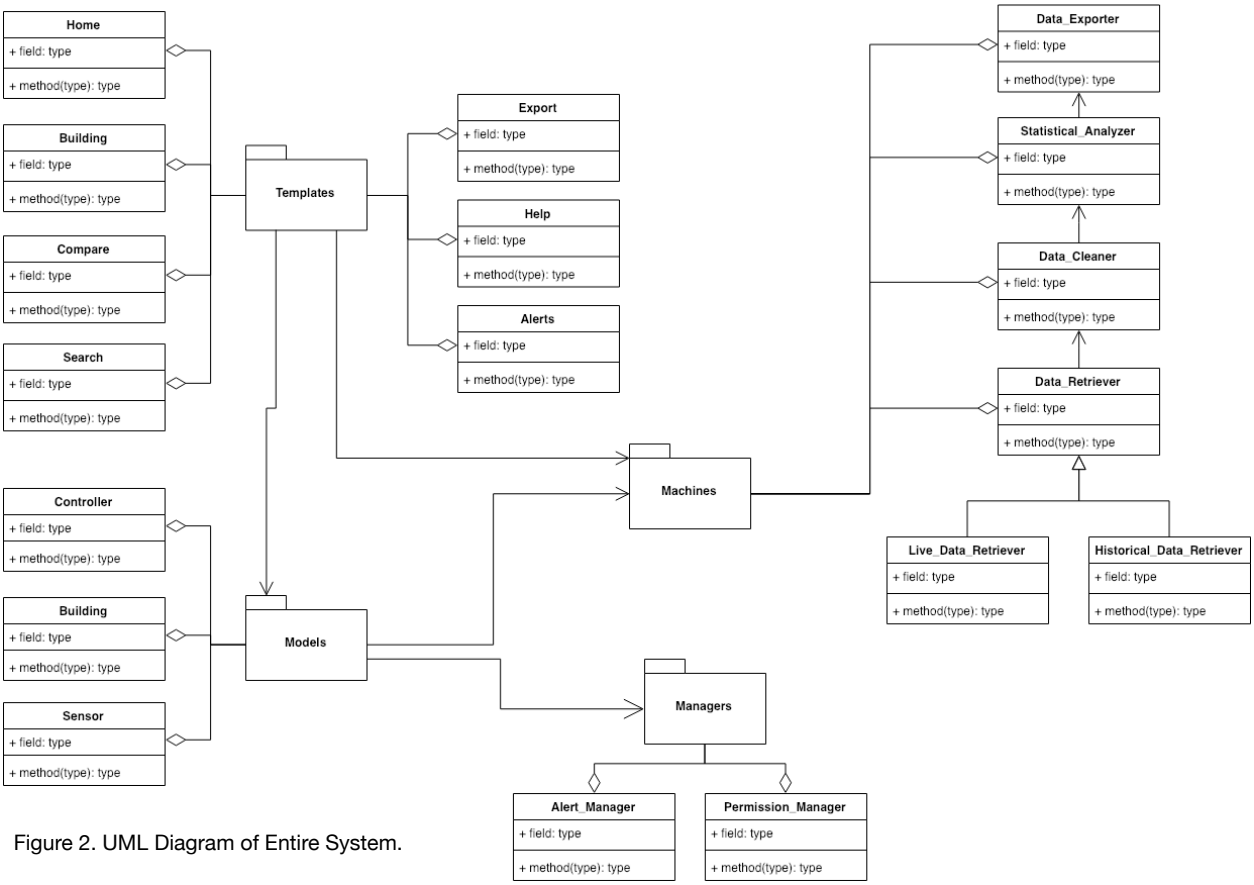


Figure 2. UML Diagram of Entire System.

4.2 The GUI Module

In Django, the template is used as an outline for HTML where Django injects code into the HTML templates that will be displayed on the website.

Each page that is displayed on the website will be a template that Django will put python code into the page. The home page will display HTML and Django, it will show HTML where it is needed and when it shows the Django in the template, it usually in a bracket label block code and end block telling Django where to put the function is used.

The homepage will show Northern Arizona University utility use of energy use throughout its campus in either a pie graph or a graph. Then there will be a menu on the top that will direct you to compare data between a building that are selected, an export function that

shows data that can be export into a CSV file, an alert function that will show if any meters are offline or some weird energy usage. On the right side menu will have a selection of buildings that will go the building page showing it energy usage. The energy usage is electricity, gas, water, and steam.

On the compare page will show a selection of building that can compare information between two building. It will be showing the difference between the two selected buildings.

On the Export page, will show a list of information on a table that will have an export button. Clicking on the button will export the information of the building which will show up in the download of the browser which can be open in Excel.

In the building page, it will show a list of the building. Users will be able to select the building and it will show a graph and stoical analysis of the building's energy usage. The Information will also show an energy conversion.

The search page will show a search bar that users will input a building name, it will list a list of possible building that the user is trying to search for.

The alert page will show possible problems in the data. If the data is a sudden drop or rise from abnormality will show up on the alert page as a possible problem.

Finally on the help page will have a contact form which will email the admin. It will have a text box that users can report the problem and have a text field for name and emails.

4.3 The Back-End Database

To store data in the back-end we will be using a structure of entities as depicted by figure 3. In this entity relationship diagram we can see that a sensor relates to a building and a building relates to a controller. This is the basis for how our entities must interact to ensure a logical way of finding data related to a building. In order to restrict access to data, we have related our users to certain permissions. Users will have permission to view, certain sensors. This will ensure that only internal NAU administrators can see in depth data while the general public views aggregate data.

In order to connect Django to the database, the name of the database engine in the setting should be changed to the one that wants to use. The Django supports several engines. But the connection between database and Django requires additional installation. The installation creates a link between a Django system and a database engine.

The class that inherits the model objects in the model means a table in the database. Adding variables within a class is the task of creating table columns within a table. First, set up the required data foundation work within the model in the project, and create tables and columns within the database through a migrate operation.

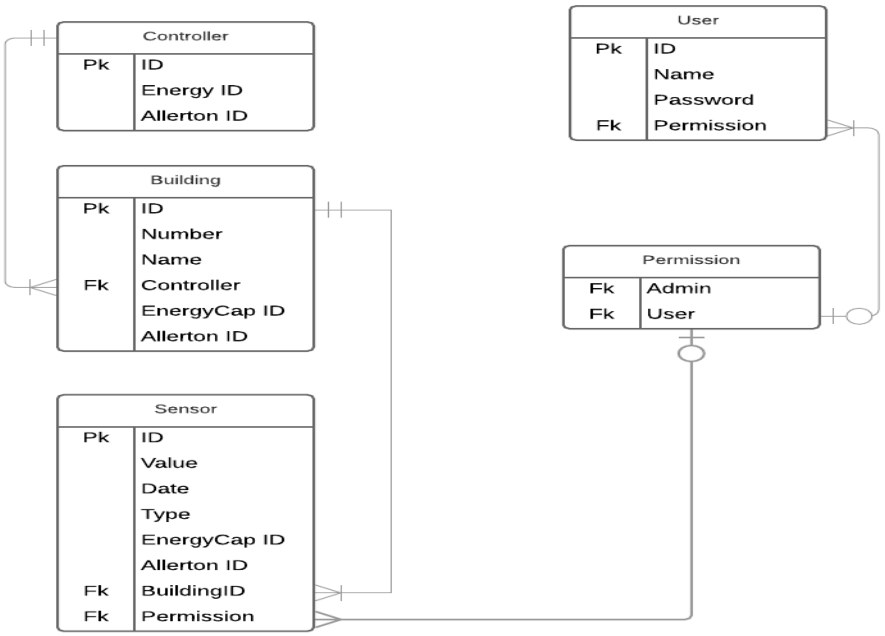


Figure 3. Back-End Entity Relationship Diagram

A new table needs to import the existing data in order to present visually comparable graphs and analyzed information. It is shown as visual data through the template view using the data recorded in the new table.

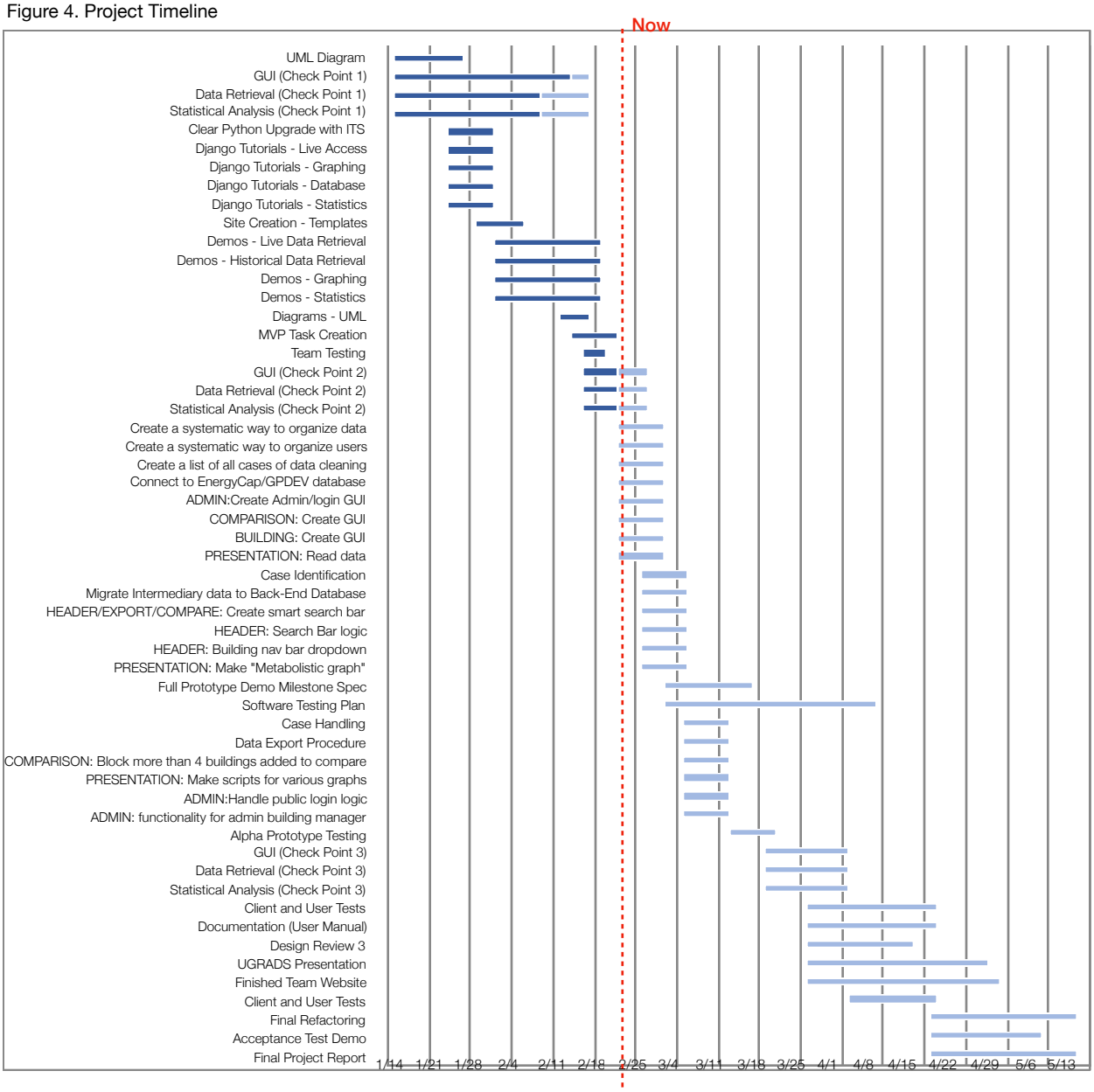
5. Implementation Plan

5.1 Project Timeline

Figure 4 shows the timeline for the development of the NAU Energy Dashboard. We are currently developing the minimum viable product. This minimum viable product will serve as our alpha prototype and will allow us to further test and refine our system. The system will be built in 2 phases.

- Phase 1: Development of Minimum Viable Product
- Phase 2: Testing and Refactoring

Figure 4. Project Timeline



5.2 Phase 1: Development of Minimum Viable Product [01/14/19 - 03/15/19]

This phase of the project is partially complete. Here we have implemented the basic architecture for the system. We have built templates for each of our pages and outlined the duties of each of our modules. We have outlined a set of tasks from 02/25/19 to 03/11/19 which will bring us to completion of the necessary functionality of the system. Once this is complete, we will have a prototype which is able to read data from the historical database, clean any anomalies, display it to the user, and allow the user to export a CSV file. Using this basis, we will begin testing. Testing at this time will consist of internally checking for major bugs, creating an error checking system, and gathering feedback from our clients.

5.3 Phase 2: Testing and Refactoring [03/15/19 - 05/12/19]

During this phase of development we will build upon our prototype. This phase consists of cyclic testing and refactoring phases. During this phase we will run several client and user tests. The goal of these tests is to inform our final implementation cycle. In March, we will begin to work toward checkpoint 3. This cycle will take into account feedback from clients' and users. We will implement any lingering features, bug fixes, and aesthetic changes. During this time we will also refine our documentation so that it is accurate in describing our evolved system. Checkpoint 3 will be followed by user tests and documentation review. Following these tests and documentation reviews, we will fix any bugs and update our documentation as necessary. After checkpoint 3, we will not implement new features, we will simply ensure the system has satisfied all requirements and is ready for launch. After final refactoring, we will deliver our final product.

6. Conclusion

This document serves as a description of the structural makeup of the NAU Energy Dashboard. In this document, we list the tools used for development. These tools are Django, Python, HTML, CSS, JavaScript, and SQL. This document outlines each component of our system and how they relate to one another. The modules described are the GUI module, the data presentation module, the data export module, the statistical analysis module, the data cleaning module, the data retrieval module, and the back-end database. Finally, we detail our plan for implementation using a Gantt chart. The Gantt chart shows when each module is developed and which team member is responsible for the module's implementation. As we continue with our implementation plan, this document will serve as our guide. Using it, we will be able to work toward one vision.

Sustainability is a growing global concern. Any step we can take to manage our consumption of resources more efficiently is a step in the right direction. Our mission is to create an interactive energy dashboard for NAU's facility services. As we work with our sponsors, Jonathan Heitzinger and Truong Nghiem, to create the NAU Energy Dashboard, we are ensuring that NAU positively impacts the future of our planet and is a leader among the Universities of the world. With the NAU Energy Dashboard, we are thinking for tomorrow; we are thinking of sustainability.