AI-VVO: Cloud-Based Machine/Deep Learning for Volt-VAR Control and Optimization

DESIGN DOCUMENT

Sdmay21-24

Client/Advisor: Gelli Ravikumar

Abdul-Salam Andedoja
Ian Kegley
Jacob Gleason
Rene Chavez
Tyler Norris

Sdmay21-24@iastate.edu
https://sdmay21-24.sd.ece.iastate.edu
Revised: 10/25/2020/Final Version

Executive Summary

Development Standards & Practices Used

Software Practices

- Machine/Deep Learning
- Volt-VAR Control/Optimization
- Google Cloud Platform
- Tensorflow
- Docker
- Front-end/back-end components

Engineering Standards

- ISO Standard Collection
- Software Engineering Standards Committee (SESC) Strategic Program Model

Summary of Requirements

- Take input from sensor readings
- Use sensor readings to formulate and design Machine Learning / Deep learning algorithms
- Store sensor reading data on Google Cloud Platform
- Take desired output from Machine Learning / Deep learning algorithms to display information on our Main Page
- Design a webpage that displays various sensor data through scripting

Applicable Courses from Iowa State University Curriculum

- Com S 227
- Com S 228
- Com S 309
- Com S 319
- Com S 311
- Math 207
- Math 165/166

New Skills/Knowledge acquired that was not taught in the course

- Design and implementation of Machine Learning / Deep Learning algorithms
- Using Google Cloud Platform, TensorFlow, and Docker resources
- Python and ReactJS programming languages

Table of Contents

List of figures/tables/symbols/definitions:

1 Introduction

1.1 Acknowledgement

We would like to give special thanks to Dr. Gelli Ravikumar and Professor Akhilesh Tyagi for giving us this opportunity to work on the project about AI-VVO: Volt/VAR Optimization, providing materials which helped us in developing our team coordination, and guiding us during our research for this project, respectively. This project could not have come to fruition without their assistance.

We would also like to thank everyone that worked on this project for putting in the time to make this project a success.

1.2 Problem and Project Statement

Distribution and regulation of energy is an essential issue in today's society. Historically, devices such as in-load tap changers, shunt capacitors, and in-line voltage regulators have been used to manage the voltage and reactive power of the target distribution grid. However, in general, these devices are slow and operate in discrete steps [1]. Volt-VAR control (VVC) determines the strategies that these devices use. As distribution grids grow, they have to transmit large amounts of power over vast distances. In these situations reducing voltage fluctuations becomes key to reducing energy loss. Since current VVC devices are slow, they consume large amounts of power when voltage changes become more frequent [1]. Recently, however, Smart inverters have been emerging as popular devices to assist with VVC. Researchers have begun exploring how machine learning could be used with these smart inverters for Volt-VAR optimization (VVO) to minimize energy loss, voltage deviation, and peak load.

This project aims to design and implement cloud-based machine learning or deep learning algorithms for VVC and VVO for distributed energy resources integrated distribution grid to increase voltage stability and reduce energy loss.

1.3 Operational Environment

This project operates using Google Cloud Platform Environment as well as a ReactJS front-end interface. We will be using Django to establish our back-end connection plus utilizing TensorFlow to create our Machine Learning/Deep Learning algorithms. We also intend to use Docker Images to establish our project's key parts with some ideal similarities for our interfaces.

We have been provided with Virtual Machines using the PowerCyber Testbed environment given by our faculty advisor. Here we can do all of our development for our project with ease and allow us to monitor each of our work to ensure productivity.

1.4 Requirements

- Collecting data streams and publish in the data pipelines
    - This is an essential part of our project because without collecting our data, we would not progress and implement our Machine Learning/Deep Learning algorithm. Simply put, with this project, we will not have to implement this portion due to receiving "mock" data from our advisor.
- Communications - HTTP / MQTT
    - We first have to establish a connection between our back-end environment and between our front-end environment. With this, we will be establishing our

connection using the Django framework for our application and have our server be running over our Google Cloud Platform.

- Design and implementation of ML/DL algorithms for VVC and VVO application
    - We will be using Tensorflow to be able to design our algorithms to be able to implement our goal in mind successfully. We need to establish a way to receive our data from the distribution grid and successfully save power overall and have a better output voltage. Establishing an excellent working algorithm is essential for our project.
- Dashboard using client-side (front-end) scripting to visualize the data, plots, and analytics,
    - We first need to create our front end for our application in order for our client to navigate through the application. We will be using ReactJS in order to create our front end. Once completed, we will have to establish a connection with our back-end database to be able to successfully pull our data in order to display our data using scripting.
- Test and validate the application with available distribution grid simulators such as OpenDSS and GridAPPS-D
    - OpenDSS and GripAPPS-D are frameworks where users can simulate their data and visualize what expected output would arise. Here we can test and formulate our algorithm to ensure our application is tested and working correctly.

1.5 Intended Users and Uses

Volt/VAR control (VVC) refers to the process of managing voltage levels and reactive power (VAR) throughout the Distributed Energy Resources (DER) grids. Using VVC, we can improve voltage profiles for all end-use customers and achieve multiple objectives, such as real power losses and voltage deviation. With this in mind, it is essential to ensure our product's quality as this program affects many end users. Formally this product is intended to be used by workers managing the DERs and designed to improve these grids' quality. Overall this project will also be used for Iowa State Lab Students and utility operators.

1.6 Assumptions and Limitations

Assumptions

- This project will not get data directly from a real distribution grid; it will instead receive simulated data in the form of a .json file. Due to this, we are not able to effectively test our product on a real-life system.

- This project will be established using a centralized VVC approach in order to create our application. We can establish a centralized controller to receive all information from our DERs and decide on what is the best option continuing forward. From there on, we can provide system-wide optimization and handle various challenges presented by DERs to VVC from that system-wide perspective.

Limitations

- The budget consists of $300 Trial Credits to be used for our Google Cloud Platform workspace in order to be able to progress on our project.

- Since we are receiving simulated data from a distribution grid simulator, we are given the limitation of testing our product in a simulated environment. With this, we can use GridApps-D, OpenDSS, and Opal-RT to test our simulated data. Our distribution grid model will be simulated,

and measurements will be delivered to us in a JSON format. Therefore, the limitation here is that we will not be working within a real-life setting.

- Due to receiving mock data from our faculty advisor, we are only able to test our product in a simulated environment. It could potentially be an obstacle to overcome when utilized in a real-world application.

1.7 Expected End Product and Deliverables

The first main deliverable of this project will be the machine learning algorithm our team will design to manage the distribution grid. The algorithm will be created using Tensorflow and trained using simulated data provided by our client. It is expected in April 2021.

The second deliverable will be a dashboard designed by our group. The dashboard will display data from the entire grid as well as individual nodes in the grid. Data for the grid will be updated for the user to see in real-time. The dashboard will be constructed using ReactJS. It is expected in April 2021.

The third deliverable will be a back-end framework capable of transferring data between our machine learning algorithm and our dashboard. The framework will handle HTTP requests to communicate with our front-end, and it will use a database to receive data from the power grid simulation. It will be constructed using Django and is expected by March 2021.

2 Project Plan

2.1 Task Decomposition

- Frontend
    - Build and organize user display
    - Establish communication with the back-end server
    - Build components to receive and display data out from the machine learning algorithm
    - Create a docker container for the design for deployment to the Google Cloud Platform
    - Test components for accuracy
- Backend
    - Deploy and configure database and webserver
    - Establish connection with OpenDSS
    - Implement machine learning algorithms
    - Train machine learning algorithms
    - Test machine learning algorithms for accuracy

2.2 Risks And Risk Management/Mitigation

Possible risks in this project include;

- Our team has little experience with several of the technologies we are using on this project. This could slow down progress on the project and disrupt our timeline. It could also limit our design's functionality if we are unable to learn the technologies fast enough. We have mitigated this risk by

researching every technology we are using to ensure it is best for our project. We have also created skeleton projects to learn the basics of the technologies.

- A large portion of our project requires virtual machines on the PowerCyber Testbed provided by Iowa State University. If a virtual machine is corrupted or taken offline, the project process could be interrupted or lost. We will mitigate this through frequent use of Git. Pushing code to Git often will reduce the amount of progress lost in the case of an interruption to our virtual machines.

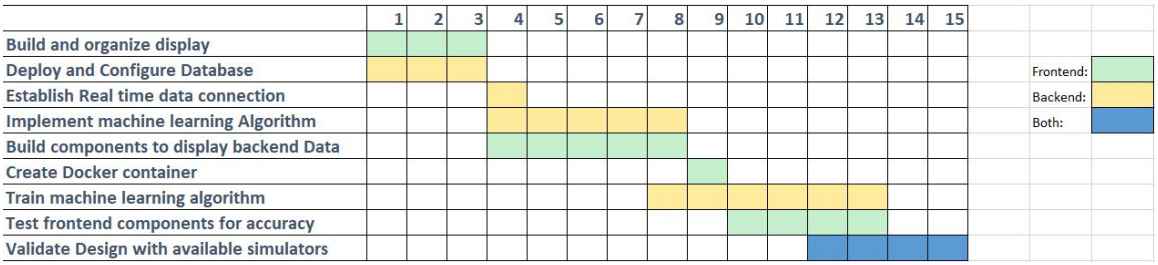
- Our team will use docker to mitigate disparities between different machines. By containerizing our project, it is isolated from its environment. This reduces the risk of our system being unable to run our design.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- *Collect data sets from distribution grid simulators such as OpenDSS, GridApps-d, and Opal-RT* - **Receive data sets; this data collected will be from simulated energy resources.**
    - *Retrieve Data Sources (csv, json, etc)* - **Receive json file from faculty advisor**
- *Establish direct access via MQTT/HTTP from the distribution grid simulator to our AI-VVO software* - **Establish a connection to our Google Cloud Platform with Django implementation**
- *Design Frontend Dashboard*
    - *Display datasets from our Distribution Energy Resources* - **The required data from our Distribution Energy Resource are displayed on our dashboard**
    - *Frontend scripting to visualize the data, plots, and analytics* - ***See desired data on front-end client-side dashboard***
- *Design Backend*
    - ***Give input data and receive output data from core applications***
    - ***Handle HTTP requests from the front-end***
- *Design and implement our core applications for Volt-VAR Control and Volt-VAR Optimization* - **Design algorithm to be able to optimize our results for our distributed energy resources integrated distribution grid**
    - *Begin training our system in our environment* - ***Train system to optimize our data streams to at least a 75% efficiency rate increase***
    - *Begin testing our system in our environment* - **By testing our system, we want to ensure that our algorithm is functioning correctly and delivered our expected result**
- *Test and validate the application with available Distribution Grid simulators* - **Test application to ensure all functions are running smoothly and efficiently**

2.4 Project Timeline/Schedule

Figure 1: Project timeline Gantt Chart



The immediate implementation of our project will begin with the Iowa State University spring semester on January 25; the first week of the semester is defined as week 1 in the Gantt chart in figure 1 above. We expect our first two main deliverables, the machine learning algorithm for VVC/VVO and a dashboard for data output, to be ready by the end of week 13 in mid-April 2021. Our project's third deliverable, a back-end framework capable of handling the real-time data transfer between our algorithm and front-end, is expected to be finished by week 8 in mid-March 2021.

2.5 Project Tracking Procedures

Our team will be using the kanban board available in Gitlab to keep track of tasks and subtasks. Tasks will be assessed and assigned in our weekly meetings with our client/advisor. Merge requests will need to be approved by other team members working in the same area. For example, a merge request for our front-end will be reviewed by the other front-end team members. Our team will also use zoom to attend weekly meetings with our client/advisor. Discord will be used to coordinate meetings among team members as well as discuss project progress and issues.

2.6 Personnel Effort Requirements

Table 1 - Task time effort estimates

Task	**Explanation**	**Person-hours**
Build and Organize user display	Create a platform in ReactJS and create our components' base to present a basic version of the dashboard.	25
Deploy and configure database and webserver	Deploy the project database and set up a Django web server. Includes URL mapping	20
Establish connections to the real-time data stream	Establish a connection to openDSS for real-time data input of a distribution grid. Our client/advisor will heavily assist this task.	5
Implement Volt/VAR	Create machine/deep learning algorithms to	50

Control/Optimization machine learning Algorithm	manage Distribution Grid from a source file to an output data file.	
Build components to receive and display data out from the machine learning algorithm	Build components up to continuously receive and display data that is output from the machine learning algorithm. Data about individual nodes and the grid as a whole will be displayed.	50
Create a Docker container for design to be deployer to Google Cloud Platform	Create a docker container that can build our project and deploy it continuously on the Google Cloud Platform.	15
Train Machine Learning Algorithm	Display the datasets produced by our simulation on our front end dashboard.	50
Test front-end Components for accuracy	Create and run tests to ensure our components are updating and displaying data correctly	40
Test and validate application from available Distribution Grid simulators	Simulate our project for real-life use	40

2.7 Other Resource Requirements

Other resources aside from financial required to complete the project are the PowerCyber workbench and virtual machines. These resources will be provided by our client/advisor and Iowa State University.

2.8 Financial Requirements

$300 - Google Cloud Platform trial credits provided by our client/advisor and Iowa State University College of Engineering.

3 Design

3.1 Previous Work And Literature

An article titled “Volt/VAR control for Power Grids with Connections to Large-Scale Wind Farms: A Review” discusses the challenges currently being faced with implementing VVC. The issues mentioned include; increasing uncertainty of power flow and voltage, intensifying conflicts among VVCs of interconnected power grids, reducing voltage stability, increasing operation of VVC devices, and complications in the reactive power market. A couple of these issues, complications in the power market and conflicts among VVCs, are outside this project’s scope. The main issue our design can help solve is the increasing operation of VVCs. Current VVC devices are slow and

consume large amounts of power when dealing with fluctuating amounts of transmitted power. An extreme example is a rectifier's reactive power being calculated to be 30%-50% of the transmitted power. This situation was calculated concerning a large scale wind farm (LSWF), which is isolated and deals with large amounts of fluctuating power. While our design could be applied to an LSWF, it will not always be. That is not to say the issues will exist, but they will be smaller in magnitude. The paper also goes on to talk about areas of ongoing and future research. Such as applying big data techniques to VVC.

The article titled "Volt-VAR Control in Power Distribution Systems With Deep Reinforcement Learning" mentions the approach of working with a Deep Reinforcement Learning algorithm. Here the main problem addressed is the issue concerning VVC and ensuring voltage levels are optimal for efficiency. Both over-voltage and under-voltage conditions can reduce energy efficiency, cause equipment malfunction, as well as damage consumer's electrical appliances. Here using the reinforcement approach, we can make control decisions online based on off-line trained models. Q-learning based algorithms are for measuring input voltage levels and to receive a final output voltage. In this article, it states that all reinforcement learning based-algorithms are developed for the VVC problem as an action-value method. Here the algorithm learns the values of actions it can take and then selects actions based on estimated action values.

An article titled "Voltage/VAR Control and Optimization: AI Approach," a team used a hybrid system for their VVC/VVO design. The design consisted of two main components. First, a centralized decision tree-based system for intelligent VVC/VVO in a power system to contain voltage stability. The decision tree was developed from a model of online decision trees called Proximity Driven Streaming Random Forest (PDSRF). Properties of the PDSRF combined with voltage stability L-index indicators as target vectors, made the machine learning approach possible. The tree was pruned using a replace-the-looser approach combined with the adaptive ensemble trees aggregation rule. The second component was a decentralized multi-agent control system for preventing voltage collapse. This component was used for fixing the moment of critical overload and switching to the load shedding procedure. Overall this approach was successful in identifying the global L-index with 97.24% accuracy, 10% higher than other contemporary algorithms. This design is similar to ours in the aspect of using a centralized VVC. However, our design will use a reinforcement learning algorithm instead of a decision tree.

3.2 Design Thinking

Users need easy to digest data in order to see how their algorithms perform with the given simulation data in order to make adjustments to improve their algorithm. When it comes to working with VVO applications, many sections need to be implemented in order to be able to fulfill the project objectives successfully.

We want to use Docker images so that other back-end algorithms can be swapped in with the ones we design. They will provide greater flexibility for users to test multiple approaches to VVC/VVO. Using Docker allows for more effortless transitioning of our first design and implementing them with other aspects of the project. By solely creating a based docker image, we can use that image and smoothly transition it to other aspects of the project. This decision to use Docker was on our faculty advisor as they have experience working with more technologies that he believed would be suitable and effective for our project.

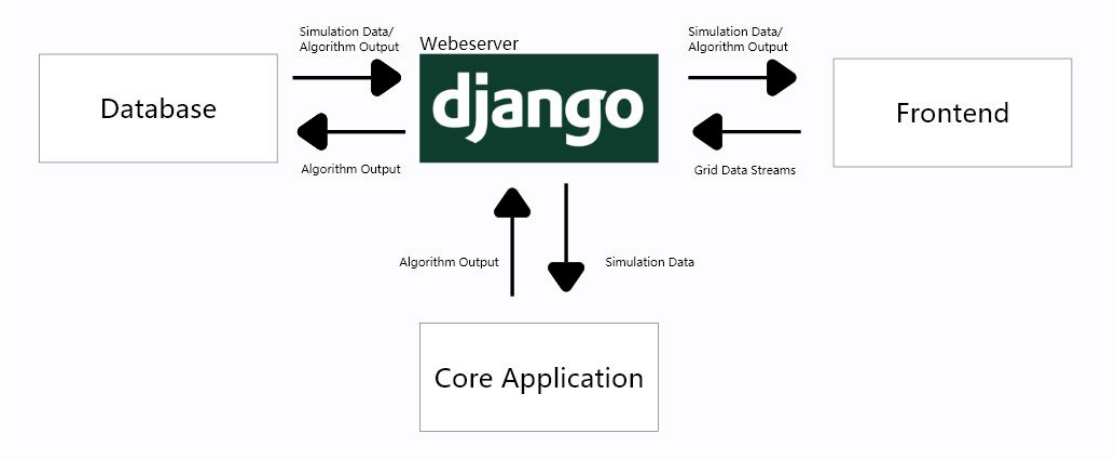
We plan to use ReactJS for our front-end interface due to some experience in our group with it. We believe that ReactJS has sufficient capability to provide us with the tools to implement our front-end interface successfully. We were discussing the possibility of being able to use AngularJS over ReactJS, but with the experience in our group, we leaned towards ReactJS. We can rely on our group's experience and, with their provided expertise, implement our front-end interface effectively this way.

3.3 Proposed Design

Our design will have three central modules; a back-end built using Django and PostgreSQL, a core application built using TensorFlow, and a front-end built using ReactJS.

The web server built using Django will be responsible for receiving simulation and user data and sharing data output from the machine learning algorithm with the dashboard. A diagram of the back-end's communications can be seen below in figure 2. The Django web server will pull simulation data from the database and feed it to the machine learning algorithm. It will also update the database with output from the algorithm. HTTP requests from the dashboard will be handled by the Django web server as well.

Figure 2 - Backend Communication diagram



Our core application will be built using TensorFlow. The machine learning algorithm will be managing a centralized VVC/VVO system, meaning that it will receive all the information of power system states. By receiving simulated data we will be able to produce, implement, and test our machine learning algorithm. The overall design that we plan to implement for our algorithm will be using a reinforcement learning approach. The reinforcement learning approach is an agent that is essentially trying to resolve a problem within it has given the environment by receiving feedback and learning off the given feedback. For our current design our agent would be our voltage levels from our given nodes from the Distribution grid. The environment for our project would be the given testgrid provided with each given input and output, respectively. Here our goal is to manage our voltage levels and be able to monitor our levels to our expected output level. We will use our given data and be able to provide our agent the feedback that is necessary for it to learn the process of managing the voltage levels.

The front-end dashboard will be built using ReactJS and is expected to consist of three main panels. The first panel will be a home and configuration panel. The panel will allow the user to decide which data streams will be enabled once the simulation begins. If a second algorithm is available, the user will be able to select either or both algorithms to run. The second panel will be a distribution grid data panel. This panel will provide the user with an interactive visualization of the distribution grid. The user will be able to interact with individual nodes in order to see relevant data from it, such as current-voltage and energy consumption. Data for the entire grid, such as load voltage, will also be accessible on this panel. The last panel will be the machine learning output page. This page will output the decisions of the algorithm. This panel will also allow the user to enable and disable input streams to test different scenarios.

Figure 3 - User interface skeletons for the (from left to right) distribution grid data panel, Output Panel, and Configuration panel

3.4 Technology Considerations

Django

We plan on using Django for our back-end framework. Django's REST framework will make transferring data back and forth with our front-end easier. Django is also based in Python, an easy to read and influential language for software development. However, Django can become bulky due to the amount of code needed to run it. This problem is most prevalent with small and low-end projects. Our project will be sufficiently large enough that the initial bulk will be negligible when the project is completed.

PostgreSQL

For our database, we will be using PostgreSQL. A main advantage of PostgreSQL is that Django officially supports it, so we will not have to add any other libraries to use it. PostgreSQL is also an object-relational database. PostgreSQL is extremely similar when compared to MySQL, another popular database. The deciding factor when choosing between them was in our research, PostgreSQL seemed to be the most popular choice when working with Django.

TensorFlow

TensorFlow is one of the most comprehensively used machine learning platforms. It has built-in library management, debugging, and graphs. For more comfortable use, the Keras API is available. Tensorflow is also currently the most widely used machine learning platform for production. However, when compared to similar libraries such as PyTorch, TensorFlow's computational speed is relatively slow. Tensorflow's widespread popularity and support overshadow this deficiency.

React

React main benefits come from its component-based architecture. React components can be reused and used to build other components. With those functionalities, we can deconstruct our user interface into small, flexible components. However, we have to be careful not to go overboard with components, as too many can lead to confusing and bulky code. React also supports JSX as a syntax extension. JSX can make creating user interface elements easier and make code easier to read as well. Representing visual data with React can be supplemented with libraries such as React-vis, which allows graphs and charts to be built using React components. The main disadvantage of React will be its learning curve. Our team members have relatively little React experience, so React's learning curve could slow implementation, especially early on.

3.5 Design Analysis

While our immediate implementation has not begun yet, we have worked on setting up a skeleton project to ensure that we have communication between our front-end and back-end. This skeleton consisted of a Hello World project in ReactJS and Skeleton Django web server. We created an example model that could be sent to the React project in JSON format. We also created a Docker container for this skeleton project. The container was capable of building the Django image without issue but ran into problems building the React image. It is believed that these issues were caused by an outdated design library used in the project.

3.6 Development Process

We are using Agile for our development process. As we progress through our project and add features, we can get feedback from our client/advisor and change our design and plans based on that feedback. Gitlab's kanban board will allow us to take advantage of Agile's kanban philosophy. Kanban philosophy stresses continuous delivery and development. We will still have set deadlines for when certain milestones must be reached, but kanban allows us to be more flexible in accomplishing those goals. Another Agile technique we will use is pair programming. Pair programming will hopefully work to counteract our team's limited experience with some of our chosen technologies, such as Django and Tensorflow. Pair programming will facilitate sharing knowledge between team members, create clearer code, and reduce confusion.

3.7 Design Plan

Our project has three main modules, the front-end, the back-end, and the core applications, which will interact with each other in various ways.

The front-end is required to display all the nodes of the simulated distribution grid relative to each other and display input and output from the core application. The front-end must also relay information to the back-end about which Grid data streams are enabled. This module is dependent on receiving both data about the simulated distribution grid and the output of our core application simultaneously and continuously.

The back-end must handle HTTP requests from the front-end. It also must receive information from the connected distribution grid simulators and feed the enabled data streams to our core application. It must also receive the output of our core application and relay that data to our front-end. It is dependent on the front end for receiving which data streams are enabled. It is also dependent on the core applications for receiving the machine learning output that will be displayed

on the front end. This module is also dependent on maintaining a connection with the distribution grid simulators to receive real-time data.

The core application will be our machine learning algorithm for managing the simulated distribution grid. The core application must receive information about the grid and will process it before outputting its decisions. The core application is dependent on receiving distribution grid data from the back-end.

4 Testing

4.1 Unit Testing

The central core application that we will have to test in isolation would be our Machine Learning/Deep Learning algorithm for our project. We have to test our algorithm with the simulated data that we will be receiving in order to ensure that our function is getting the most optimal results. We will use GridApps-D, OpenDSS, and Opal-RT to test our simulated data in a controlled environment. Each of these frameworks allows us to run simulations and see what our expected results are.

Another piece of software that will need to be tested is our connection to our back-end Google Platform Database. In order to ensure that our data is being securely saved we will have to establish a base connection to our database and ensure that we won't lose any data when uploading or receiving any data from it. Establishing this connection with Django is essential, as this is a core part of our project.

4.2 Interface Testing

Key interfaces in our project will include our back-end PostgreSQL database to store our data and results, which will need to be connected to our front-end interface. Our front-end interface will essentially display our information and show the results of our application. We have to ensure that both of these interfaces work individually to have both of them working together. By ensuring that our front-end and back-end interfaces have established connections together, we can guarantee that our functionality will operate as so. We can test our back-end by ensuring that our sent data (JSON) is being processed and stored correctly.

4.3 Acceptance Testing

We first have to ensure that our operations are working correctly and completing their set tasks. By having our essential functions working correctly, we can start implementing and incorporating our client for acceptance testing. By ensuring that each piece is working correctly and meeting our client's standards, we can optimize and change any needed pieces accordingly. We will have to demonstrate some critical functions in our project to show that our results are correct and optimized as much as possible. We want to ensure that our functions are working correctly to ensure that we meet our client's expectations; otherwise, we can continue to revamp our projects accordingly.

4.4 Results

For our results, some of the primary things that we have begun testing will be establishing our key docker images in our project. We want to ensure that we have a correct based docker image to begin then implementing our docker tree of images correctly. We are currently in the implementation phase of that and have not truly established a set docker image for our application. Some key things that we are learning are getting that first-hand exposure by working with Docker, which is relatively new. We are getting our base image set but have come with some issues. We are working with our faculty advisor to ensure that we are doing things correctly and having our docker image set. We are having trouble getting the set docker image to display our test data to ensure that we are doing it correctly. It is currently a work in progress.

As for other aspects of our project, currently, we are still in the design phase and honestly have not gotten the chance to test anything else.

5 Implementation

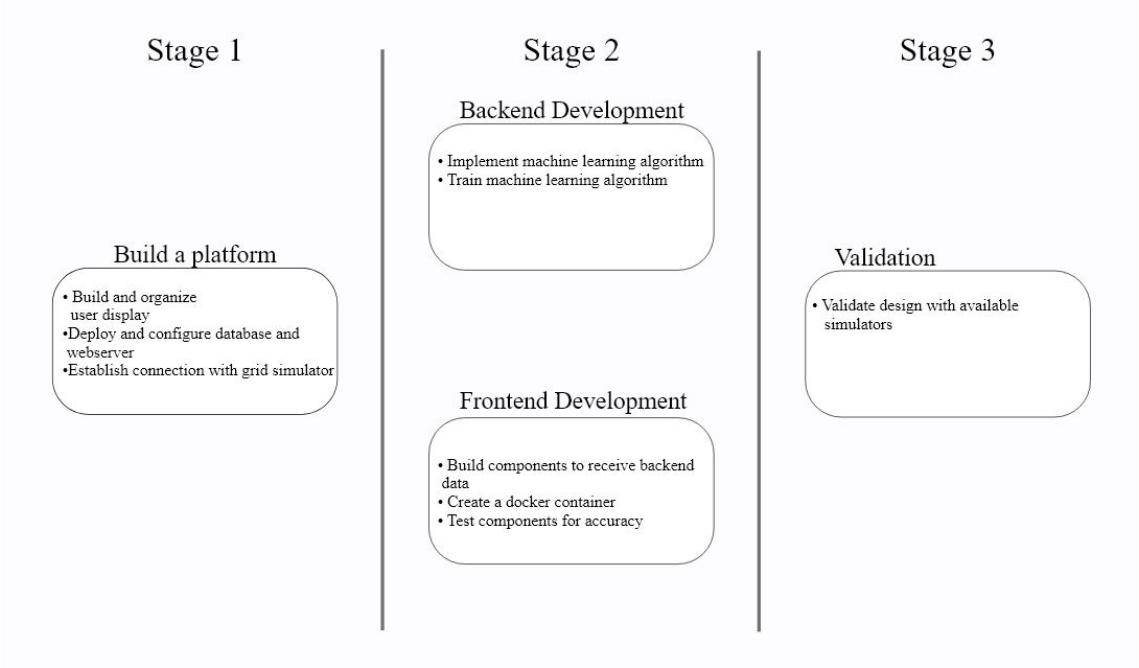
After discussions with our client/advisor, we have decided that implementation will occur in three main stages.

The first stage will be building the foundation for our design. At the end of this stage, we will have a React template populated with components that have limited functionality. These components should resemble the look of the product that will be seen at the end of stage 2. We will also deploy A Django web server and configure the database we will use. The web server should be capable of handling HTTP requests from the front-end. At the end of this stage, our client/advisor will assist us in establishing a connection to a distribution grid simulator such as GridApps-d, OpenDSS, or Opal-RT.

The second stage will add the main functionality of the project. Our back-end team will implement and train the core applications for the design. The back end should also be capable of receiving and transmitting data between the simulator, core applications, and front-end without issue in the first week or two of this stage. Our front-end team will ensure that components placed in stage 1 can display data output from the core application quickly and accurately by the end of this stage. Our front-end team will also create a Docker container to assist with the integration of our design and prepare it to be deployed to the Google Cloud Platform.

The last stage of implementation will be validation. We will use available simulators such as GridApps-d to ensure our core applications are functioning correctly. The design will be deployed to the Google Cloud Platform.

Figure 4 - Stages of Implementation



6 Closing Material

6.1 Conclusion

In summary, the AI-VVO software will utilize machine learning algorithms to manage a grid-based on simulated data from existing distribution grid simulators. Our core application will take a reinforcement learning approach in order to increase voltage stability. Our back-end will receive simulated data from a distribution grid and relay that information to our core application. The back end will then relay the output of the core application to our front-end. Our front-end will visualize the distribution grid for the user and provide information about each of the nodes in the distribution grid. The front-end will also display the input and output of our core application for the user. Finally, the front-end will also allow the user to configure which grid data streams will be enabled.

6.2 References

[1] Li, Qinhao, et al., "Volt/VAR Control for Power Grids With Connections of Large-Scale Wind Farms: A Review," IEEE Access, Vol. 6, pp. 26675-92, June 2018.

[2] Zhang, Ying, et al. "Deep Reinforcement Learning Based Volt-VAR Optimization in Smart Distribution Systems." IEEE Xplore, 2020

[3] Tomin, N. et al. "Voltage/VAR Control and Optimization: AI Approach," IFAC, 2018. Available: www.ScienceDirect.com.