AI-VVO:

Cloud-Based Machine Learning for Volt-VAR Control and Optimization

Meet the team!

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

Client/Advisor: Gelli Ravikumar

Project Information:

sdmay21-24@iastate.edu
https://sdmay21-24.sd.ece.iastate.edu

Problem:

Distribution and regulation of energy is an important issue. Noting this the issue current devices such as shunt capacitors and in-line voltage regulators typically manage voltage and the reactive power of a distribution grid. These are to put simply, slow. Researchers began looking for a solution by exploring machine learning to assist a new technology (smart inverters for Volt-VAR Optimization.

Solution:

Our team created a machine learning algorithm for VVC and VVO for distributed energy resources integrated distribution grid to increase voltage stability and reduce energy losses.

Functional Requirements:

- Collecting data streams and publish in the data pipelines
- Communications - HTTP
  - Establish connection between our back-end environment and our front-end environment
- Design and implement of ML/DL algorithm for VVC and VVO
- Dashboard using client-side scripting to visualize data, plots, and analytics.
- Test and validate the applications using available distribution grid simulators

Non-Functional Requirements:

- Function Portability using Docker Containers
- Web Server/Communication Security
- Algorithm Accuracy & Efficiency
- Dashboard Usability
- Visualization Performance

Intended Users and Uses:

Users:

- Iowa State Lab Students
- Utility Operators

Uses:

Managing and voltage levels and reactive power throughout Distributed Energy Resource (DER) grids. This process allows the improvement for voltage profiles and achieve objectives such as, real power losses and voltage deviation.

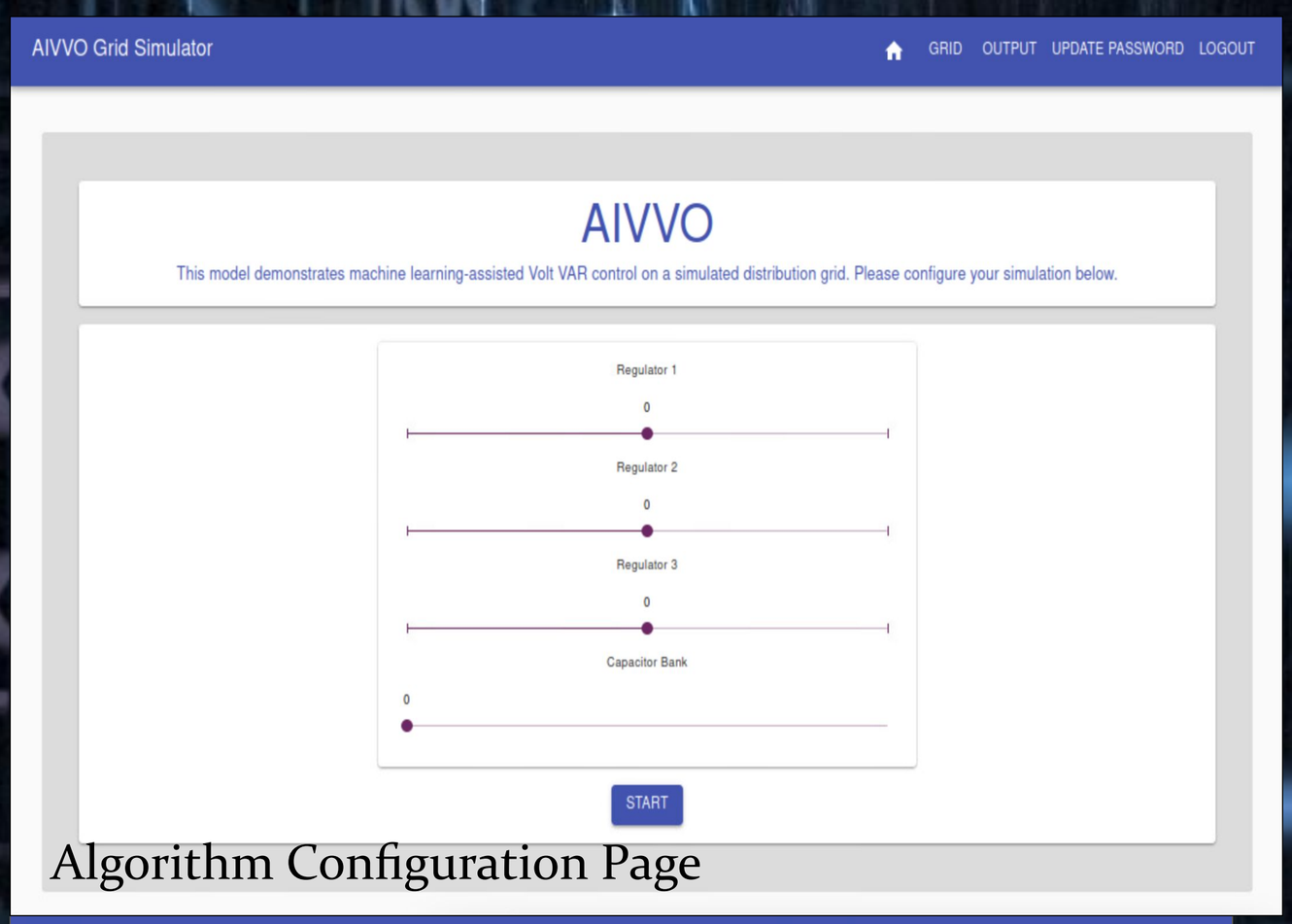
Design Approach:

Our approach has three central modules; a back-end built using Django and PostgreSQL, a core application built using TensorFlow, and a front-end built using ReactJS

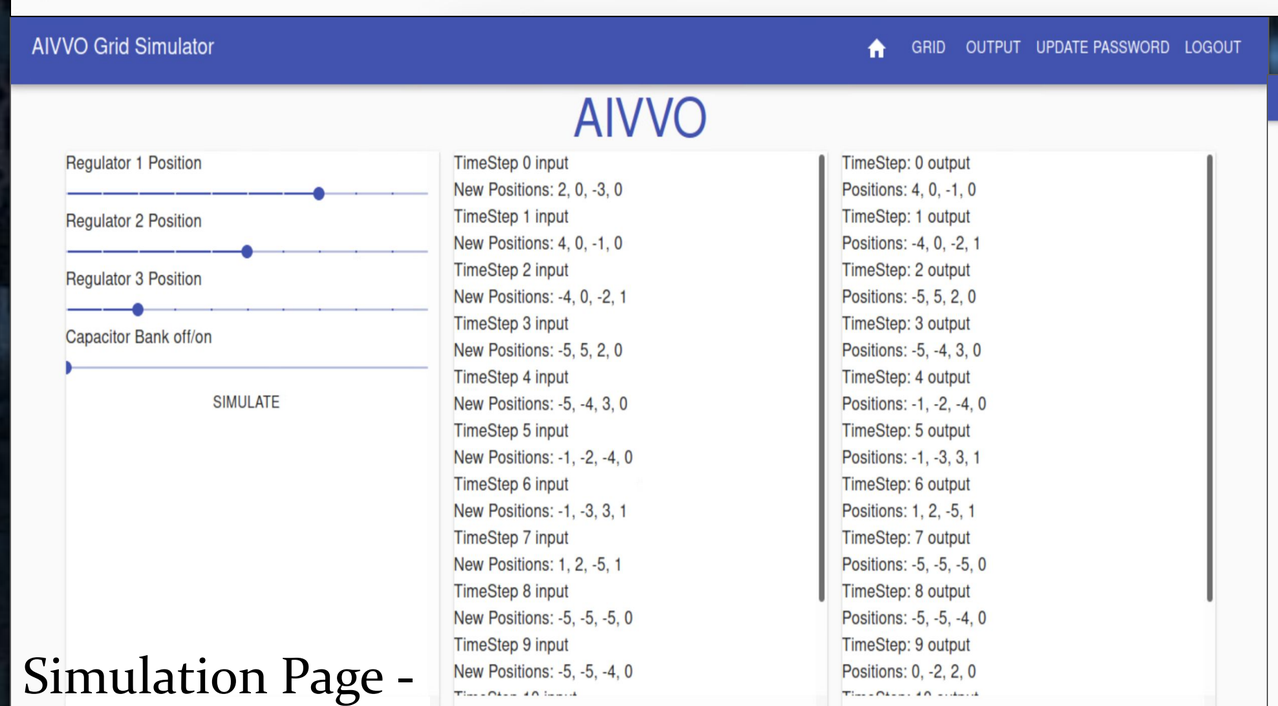
The web server was built using Django and is responsible for receiving simulation and user data and also sharing data output from the algorithm with the dashboard

The front-end dashboard was built using ReactJS and contains three main panels. The home and configuration panel being the first. Second will be a distribution grid data panel which will allow the user to interact with the distribution grid. The last panel is the machine learning output page.

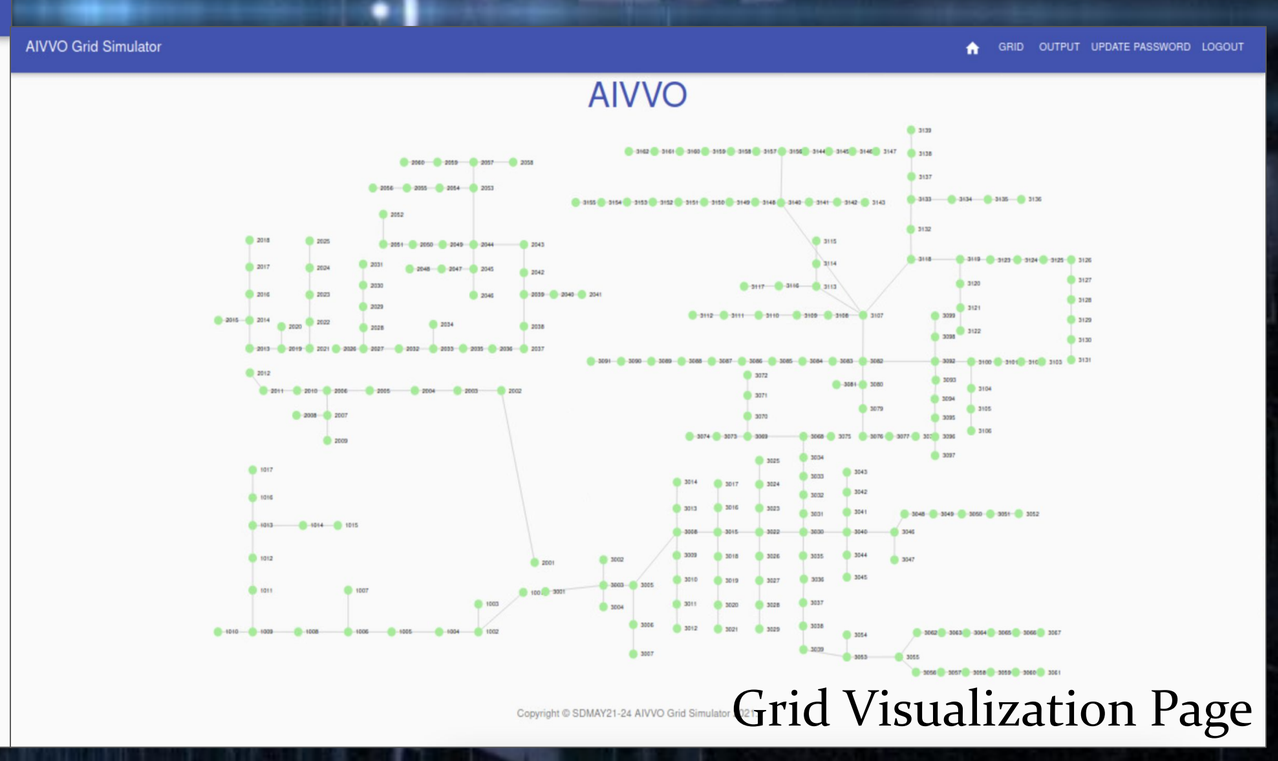
The core application was built using TensorFlow where the machine learning algorithm operates. The algorithm receives data and manages voltage levels to monitor and ensure our expected output level.



Algorithm Configuration Page



Simulation Page - Algorithm Input (middle) and Output (right)



Grid Visualization Page

Technical Details:

What we utilized:

- Django
- PostgreSQL
- ReactJS

Languages:

- Python
- Javascript

Testing:

Unit Testing:

Using GridApps-D, OpenDSS and Opal-RT we tested our data in the controlled environments. Each frameworks allowed us to run simulations and see our expected results. Also needed to test that our data is being securely saved.

Interface Testing:

The two key interfaces include our back-end database to store data and results. Then we also have our front-end interface will display our information and show the results. This testing ensures that they function separately.

Acceptance Testing:

Ensures that each piece is working correctly to meet expectation.