GridAI: Cloud-Based Machine/Deep Learning For Power Grid Data Analytics

sdmay21-23

Faculty Advisor & Client:

Dr. Gelli Ravikumar

Team Members:

Abir Mojumder

Karthik Prakash

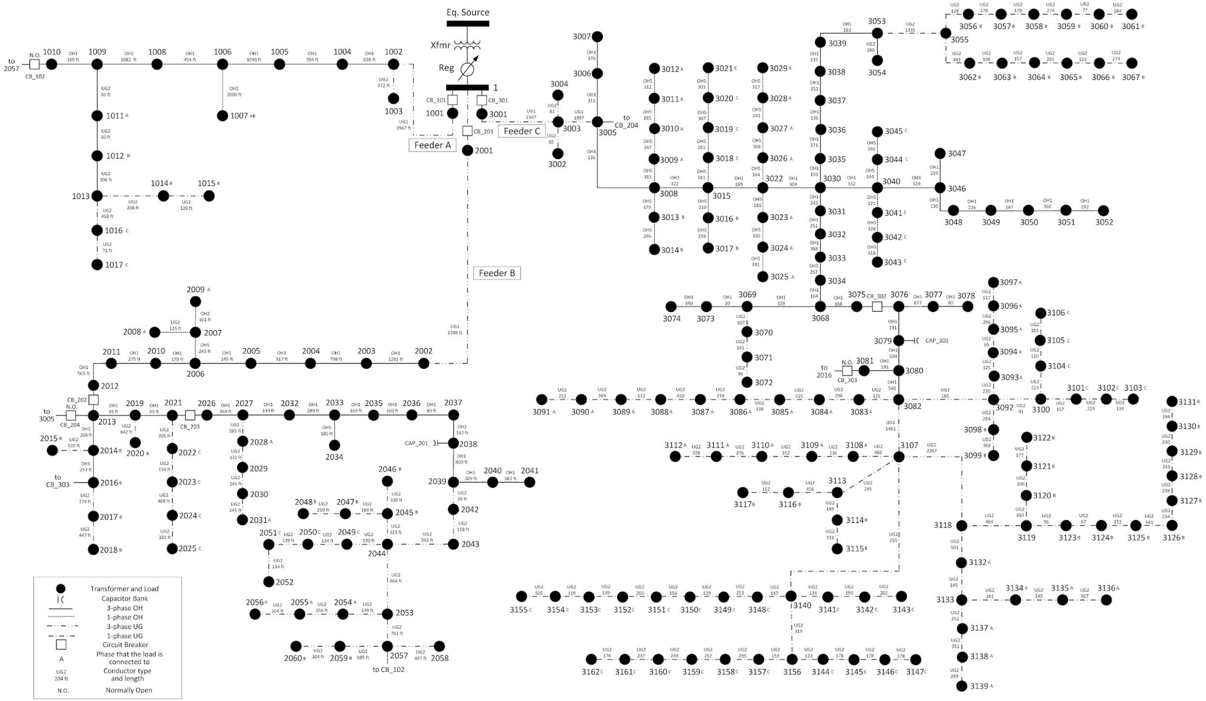
Justin Merkel

Patrick Wenzel

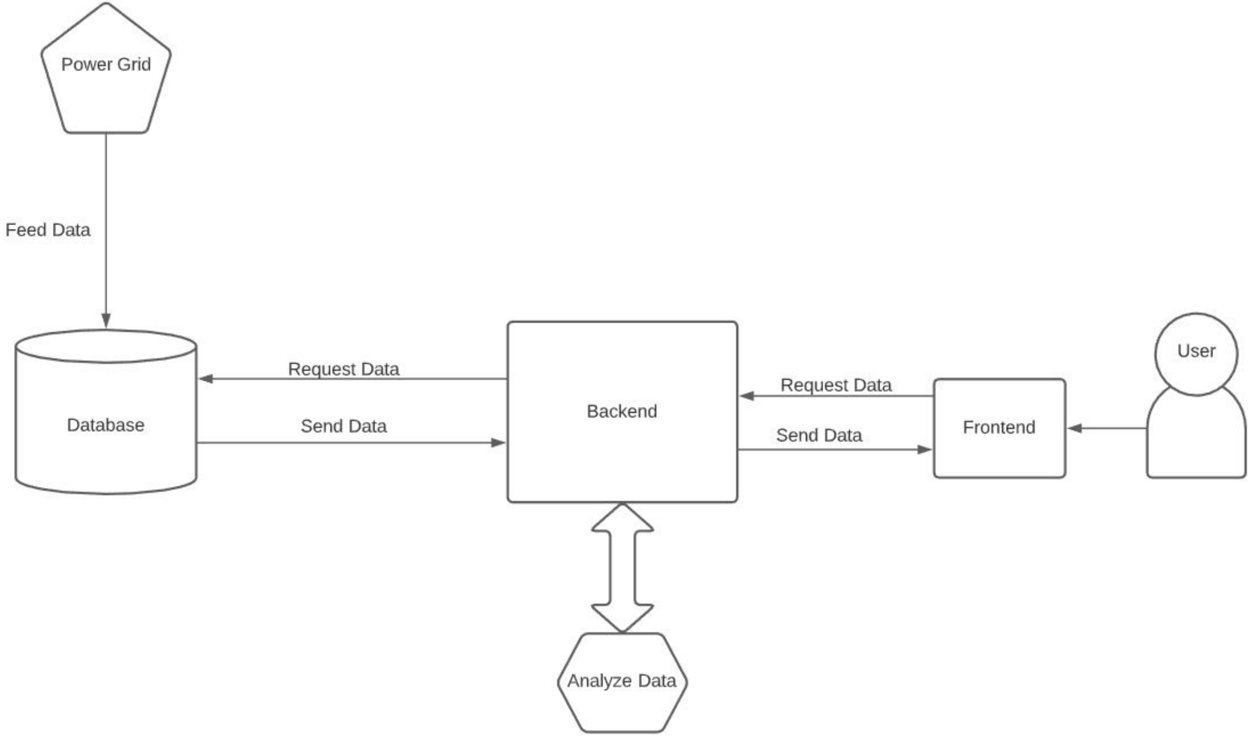
Abhilash Tripathy

Project Vision

- The Grid AI project seeks to use Deep Machine Learning to develop insights and analytics on a power grid in real time.



Conceptual Design Diagram



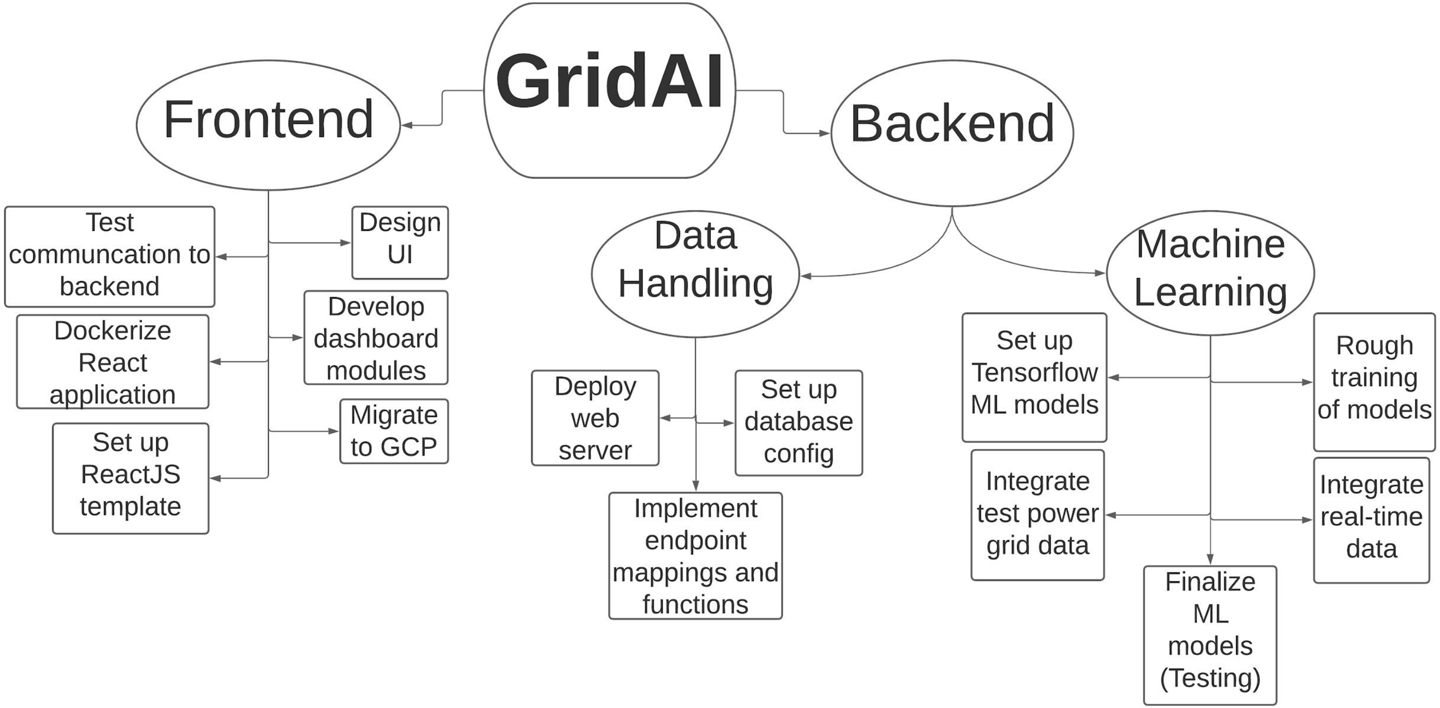
Requirements: Functional

- Analyze grid data in real-time
    - Neo4j
    - Flask
- Run on remote system
    - Google Cloud Platform
- Accurately predict and classify anomalies
    - Tensorflow/Keras
- Visualize data and analysis in comprehensible format
    - React (Material UI)

Requirements: Non-Functional

- Scalability
    - Docker
    - Modular Design
- Maintainable
    - Loosely coupled
- Performance
    - Lightweight backend framework

Planned Tasks Overview


GridAI
Frontend
Backend
Test communcation to backend
Design UI
Develop dashboard modules
Dockerize React application
Migrate to GCP
Set up ReactJS template
Data Handling
Deploy web server
Set up database config
Implement endpoint mappings and functions
Machine Learning
Set up Tensorflow ML models
Rough training of models
Integrate test power grid data
Integrate real-time data
Finalize ML models (Testing)

Risks and Mitigation

Risks:

Machine Learning:

- Real-world power grids have several variables to consider
  - Size of Power Grid
  - Population of Consumers affecting power draw
  - Demand in different seasons
- The accuracy of the machine learning algorithm to detect anomalies will only be trained on OPAL-RT

Cloud Integration:

- $300 GCP access to be utilized later.
- Deployment issues on server will be difficult to resolve with time constraint.

Risks and Mitigation

Mitigation:

Machine Learning:

- Larger pool of data; try different ML algorithms to test with the DNN.

GCP setup

- Use trial benefits to understand the platform and requirements.

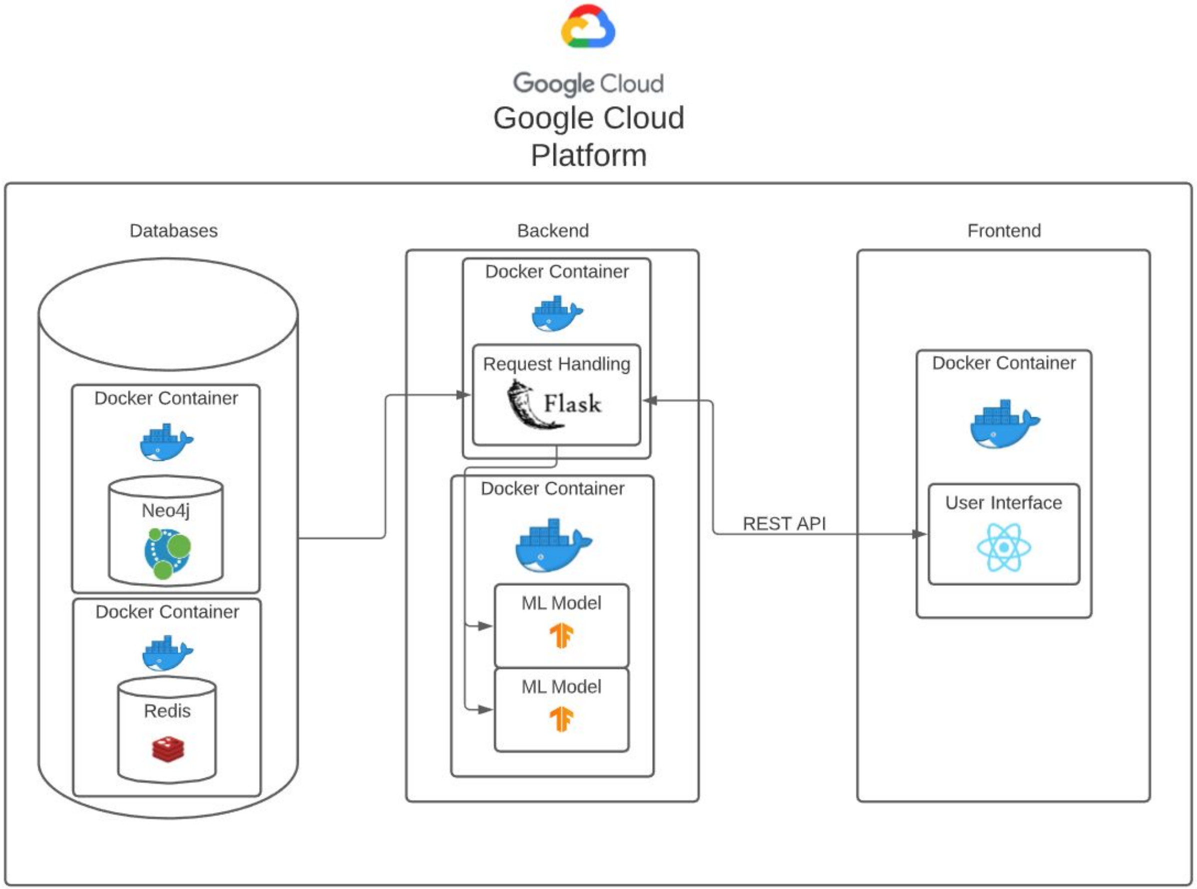
Task related issues - Priority Rating

- Resolve higher priority issues first.

Test Components according to IEEE standards:

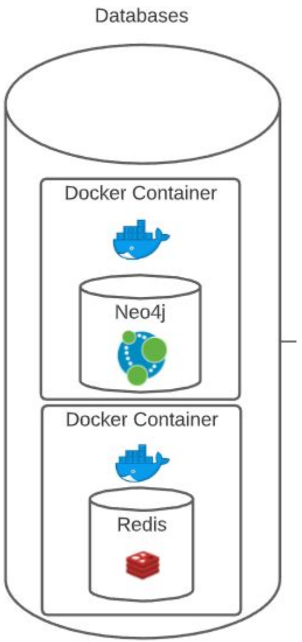
- Evaluate component functionality and reliability with others (check modularity).

System Design: Architecture



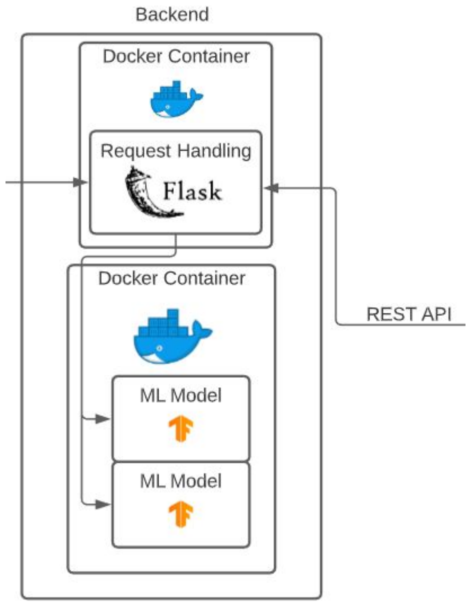
System Design: Databases

- Neo4j
  - Main Database
  - NoSQL node architecture
  - Store grid data
- Redis
  - Cache Memory Database
  - NoSQL architecture
  - Potentially store ML analysis for short time



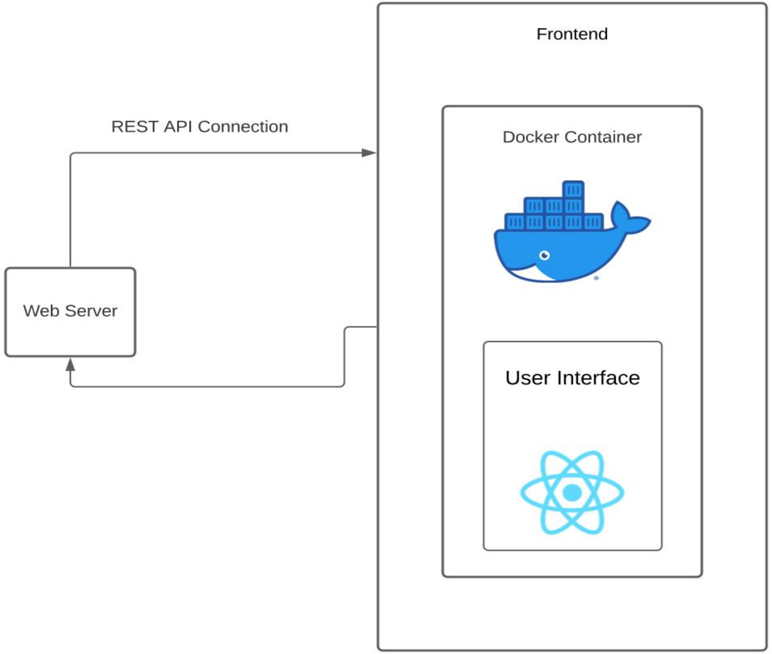
System Design: Backend

- Request Handling
  - Flask
    - Lightweight
  - REST API
  - JSON data structure
- Machine Learning Models
  - Tensorflow & Keras
  - Predictor and Classifier
  - Create more as necessary



System Design: Frontend

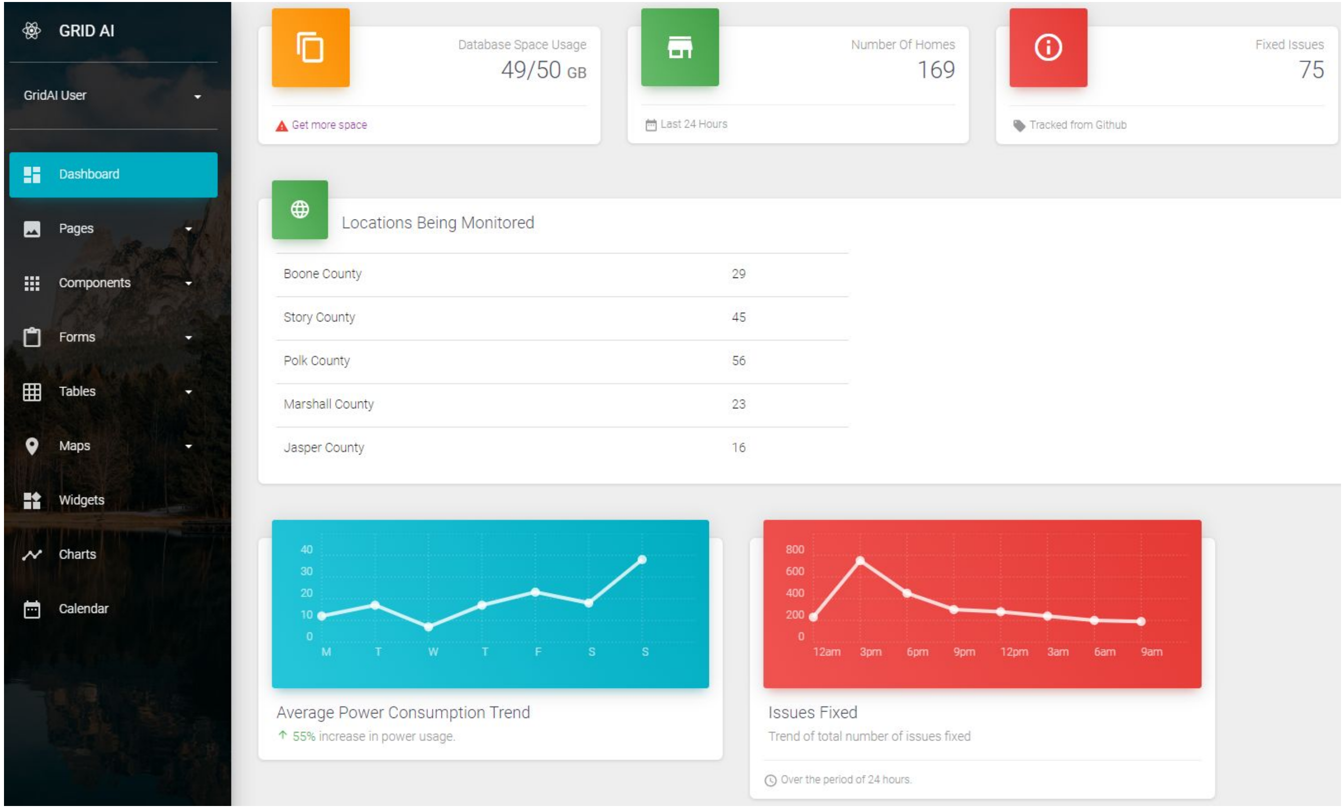
- User Interface
  - Web Application
  - React
    - Material UI



System Design: Deployment

- Docker
  - Containerize individual components
  - Allows for efficient deployment
  - Do not have to worry about host system configuration
- ISU PowerCyber Testbed
  - Developmental environment
- Google Cloud Platform
  - Cheap and accessible
  - Reduce resources needed from our end and user end

Frontend Prototype



Database Space Usage
49/50 GB

Get more space

Number Of Homes
169

Last 24 Hours

Fixed Issues
75

Tracked from Github

Locations Being Monitored

Boone County	29
Story County	45
Polk County	56
Marshall County	23
Jasper County	16

Average Power Consumption Trend

↑ 55% increase in power usage.

Issues Fixed

Trend of total number of issues fixed

Over the period of 24 hours.

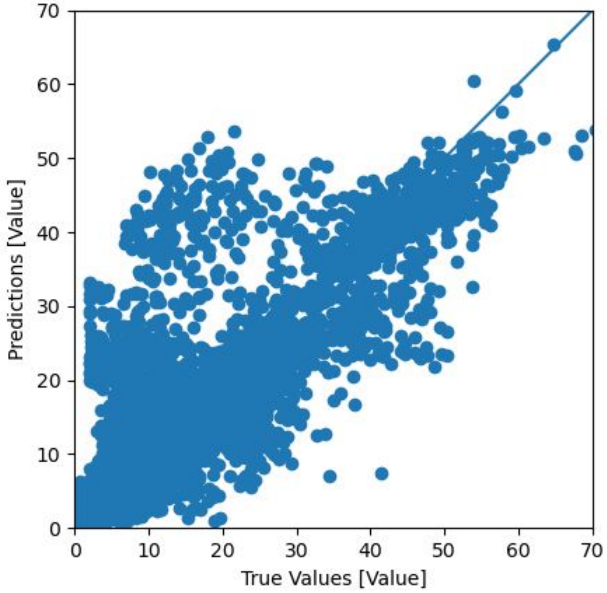
Machine Learning Prototype

- Initial naive implementation for estimating node outputs

- Currently takes static node information and timestamp as input into DNN

- Does not include output from previous nodes in chain

Plot of Prediction vs Actual Value For Feeder A

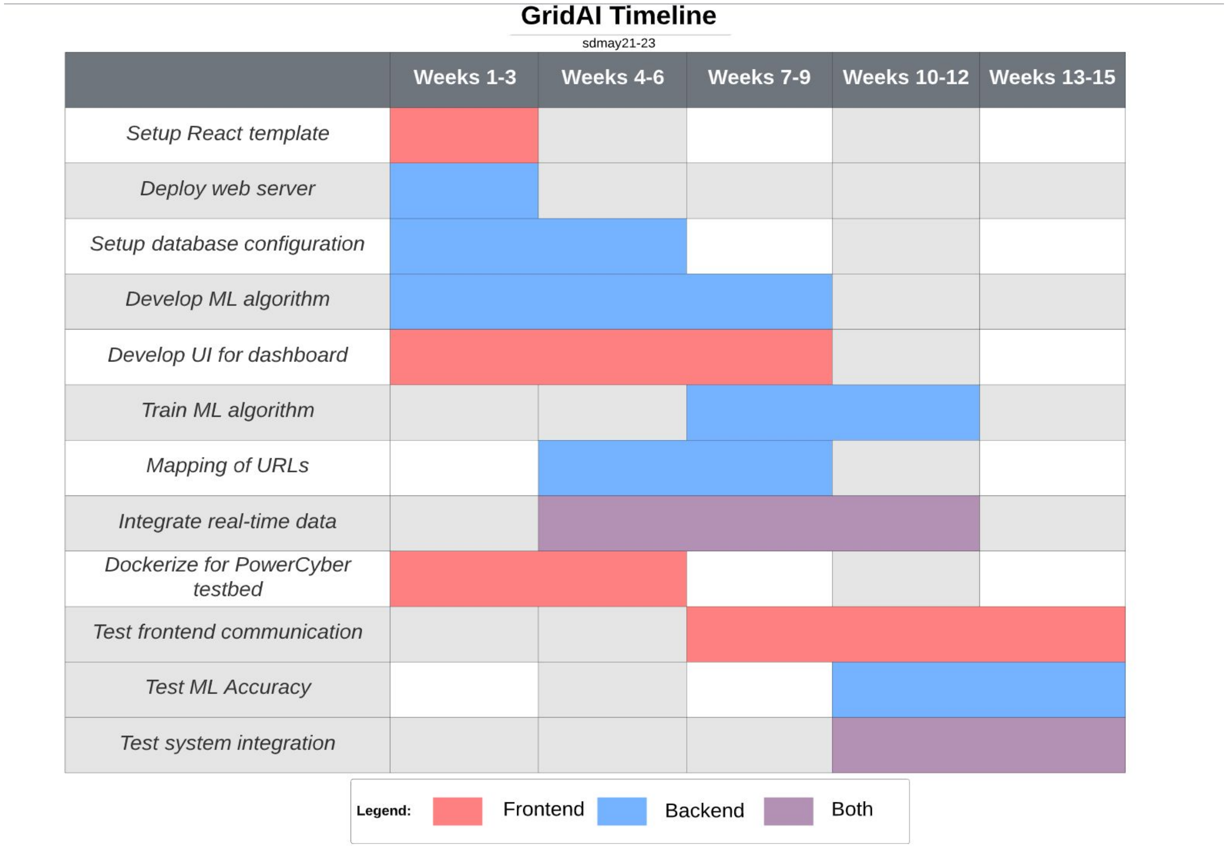


Project Plan: Milestones

Milestone Duration and Progression Metrics:

- Difficulty of tasks based on team-member experience
- Client evaluation
- Meeting functional and non-functional criterias
- Weightage based on estimated sub tasks

Project Plan: Milestones



GridAI Timeline

sdmay21-23

	Weeks 1-3	Weeks 4-6	Weeks 7-9	Weeks 10-12	Weeks 13-15
Setup React template	🟥				
Deploy web server	🟦				
Setup database configuration	🟦	🟦			
Develop ML algorithm	🟦	🟦	🟦		
Develop UI for dashboard	🟥	🟥	🟥		
Train ML algorithm			🟦	🟦	
Mapping of URLs		🟦	🟦		
Integrate real-time data		🟪	🟪	🟪	
Dockerize for PowerCyber testbed	🟥	🟥			
Test frontend communication			🟥	🟥	🟥
Test ML Accuracy				🟦	🟦
Test system integration				🟪	🟪

Legend: 🟥 Frontend 🟦 Backend 🟪 Both

Test Plan: ML Models

- 3-stage Process
    - Predicting output using the training data set
    - Predicting output using the OpenDSS simulator on the same parameters as the training set
    - Predicting output of a generic power grid using Opal-RT
- At each stage the expected and actual values will be compared to be acceptable by the client.

Test Plan: Backend Functions

- Utilize Postman
  - Test individual endpoints with dummy data
  - Test database integration
  - Test ML model integration

Test Plan: Frontend Functions & Interfaces

- Manual Testing
  - Making sure data is getting processed in JavaScript functions correctly
  - Validate data showing in graphs
  - Verify components that need perform functions perform them and perform them correctly
- Unit Testing
  - Jest
- Acceptance Testing
  - Verify no components overlap, run off screen, or aren’t showing up
  - Have faculty advisor/client also validate our interface design

Conclusion

- Still in the early stages of project implementation
    - Machine Learning Model (Justin Merkel, Karthik Prakash, Abir Mojumder)
        - Docker Container Configured
        - Development in progress
    - Backend (Abir Mojumder, Karthik Prakash, Justin Merkel)
        - Database Setup in progress
        - REST API endpoint mappings in progress
    - Frontend (Patrick Wenzel, Abhilash Tripathy)
        - UI template implemented

Next Semester Plans

Backend

- Set up Google Cloud Platform (GCP) account
- Set up database instances in GCP
- Develop and train machine and deep learning models
- Set up data pipeline

Frontend

- Develop frontend interface
- Connect frontend to GCP database
- Developing ability to set up queries on the frontend to get data from the backend
  - Verify data on the frontend side
  - Visualize the data

Questions?