App for Microgrid Demonstration

sddec21-21

Team Members: William Bronson, Michael Doyle, Gabriel Rueger, Patrick Shirazi, Micheal Thai
Client: Anne Kimber
Faculty Advisors: Mathew Wymore, Nicholas David, Steven Nystrom

Introduction

Problem Statement: The ISU Electric Power Research Center operates a solar crate with microgrid data from multiple data sources, but there isn't a effective way of accessing the data.

Solution: Our team created a mobile application to retrieve and present the data collected by the solar crate to public users and site maintainers.

Design Requirements

Functional Requirements:
- Ability to add additional crates
- Ability to add additional data sources
- Query and search different subsets of the data
- Configurable data collection interval
- Support automatic archiving of data.

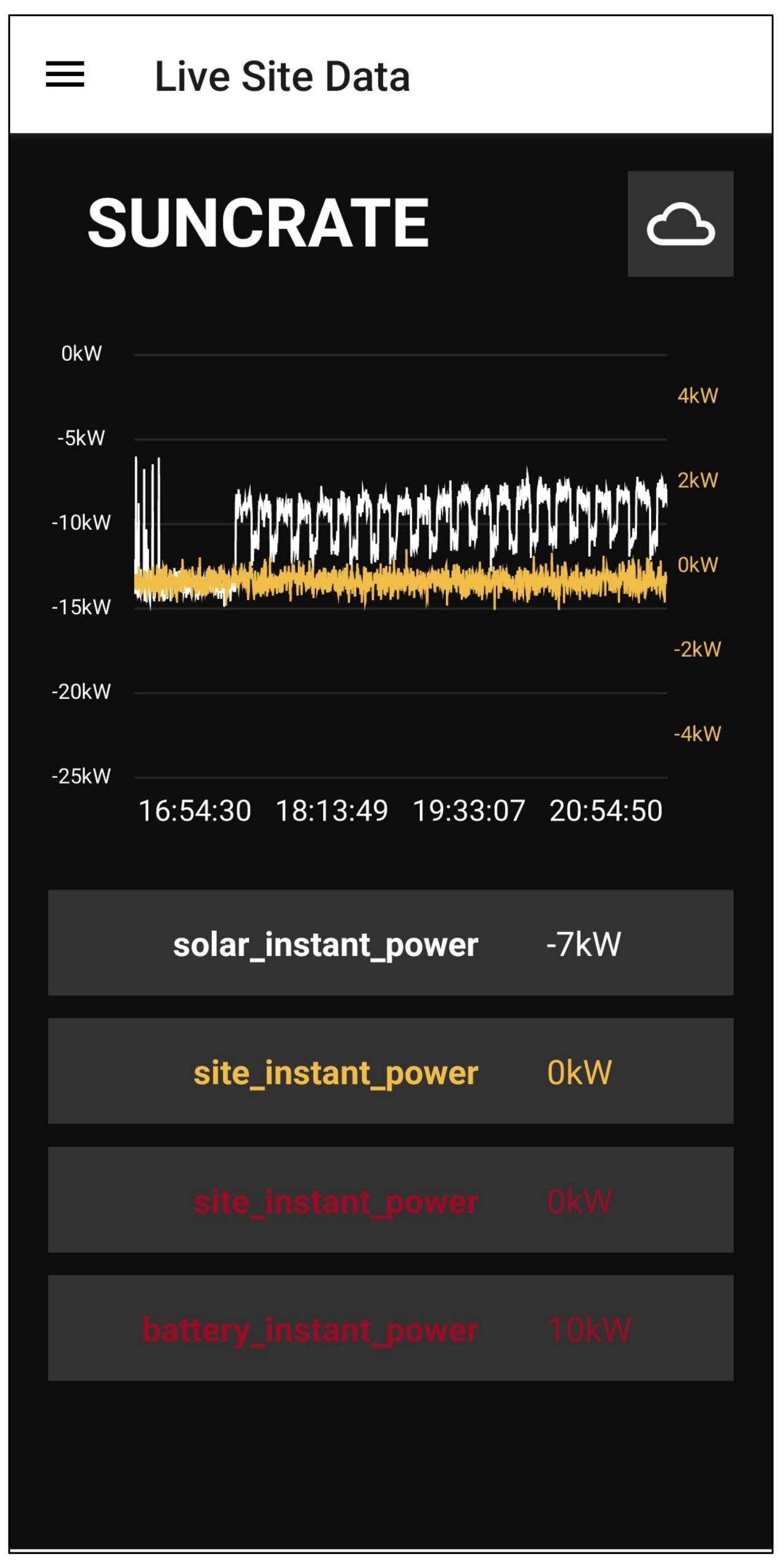
Non-Functional Requirements:
- Data must be displayed within a minute of collection
- Database size scales with data sources and time
- Libraries must be well-supported and maintained
- Must use open and well-supported communication standards
- All design decisions must be well documented

Engineering Constraints:
- Scalability - scale well for large number of sites
- Performance - want timely data
- Adaptability - new data sources and sites easily added
- Usability - user friendly

Operating Environments:
- iOS/Android for mobile app
- Linux Server for backend

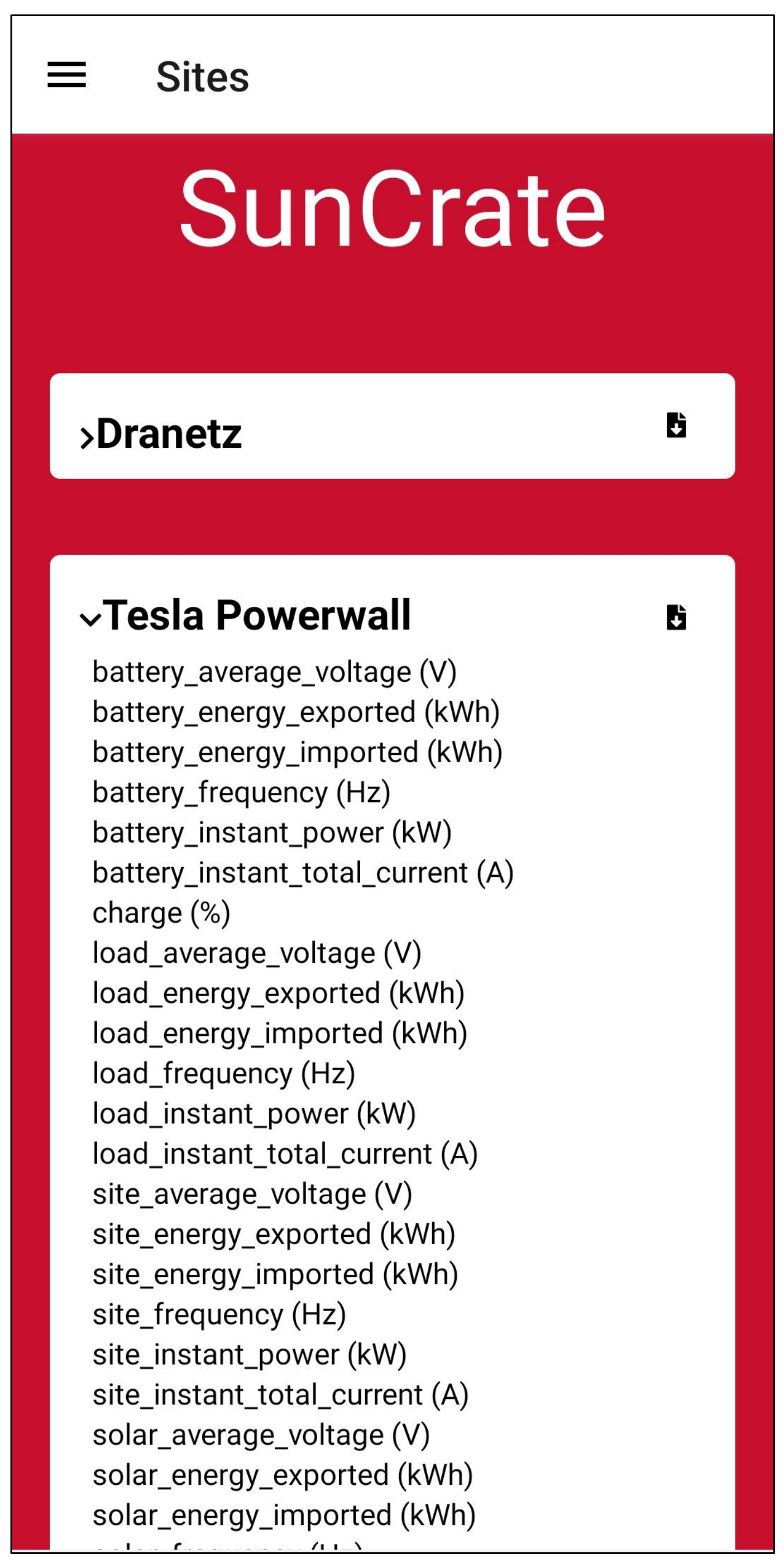
Intended Users and Uses

This mobile application is intended for educational/informational purposes and publicity and will display each site's overall performance. Additionally, researchers should be able to access the voltage, current, and frequency data readings.



solar_instant_power -7kW
site_instant_power 0kW
site_instant_power 0kW
battery_instant_power 10kW

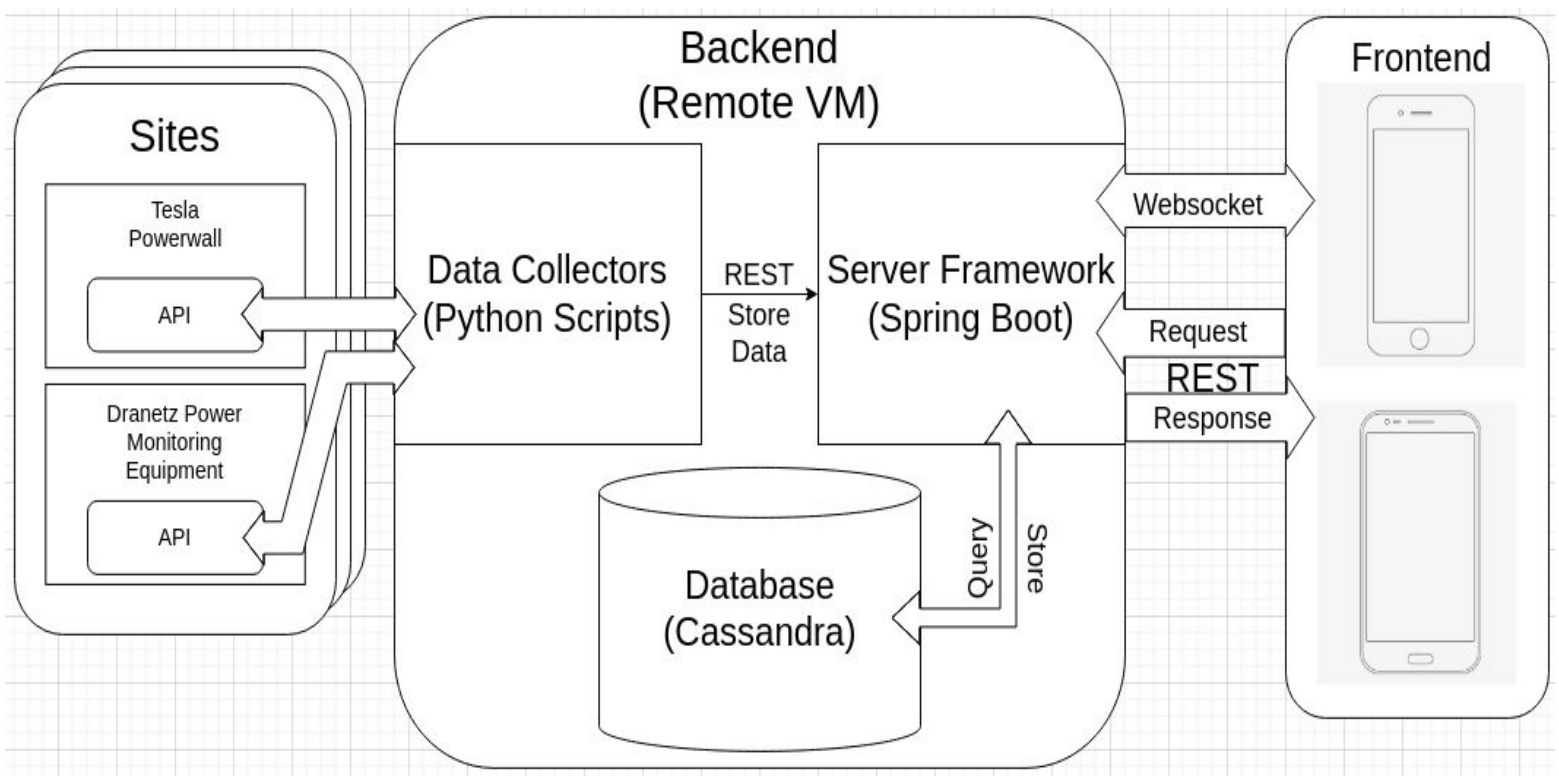
Page for viewing live data from a site



battery_average_voltage (V)
battery_energy_exported (kWh)
battery_energy_imported (kWh)
battery_frequency (Hz)
battery_instant_power (kW)
battery_instant_total_current (A)
charge (%)
load_average_voltage (V)
load_energy_exported (kWh)
load_energy_imported (kWh)
load_frequency (Hz)
load_instant_power (kW)
load_instant_total_current (A)
site_average_voltage (V)
site_energy_exported (kWh)
site_energy_imported (kWh)
site_frequency (Hz)
site_instant_power (kW)
site_instant_total_current (A)
solar_average_voltage (V)
solar_energy_exported (kWh)
solar_energy_imported (kWh)

Page for exporting data from a site

Design Approach



Testing

Testing Tools:
- Jest and Enzyme for Frontend
- JUnit, Mockito and Postman for Backend

Testing Strategies:
- Unit tests - Tests for functions used in frontend components and backend services
- Integration Tests - Testing screen components with different states and actions, as well as endpoints
- Acceptance Testing - Weekly demos to advisors on features completed during the week to gather feedback or approval

Technical Details

The Backend was comprised of a Spring Boot server, Apache Cassandra database and data collector scripts written in Python. Each of these services was containerized and deployed using docker and docker-compose.
The Frontend was a cross platform mobile application for iOS and Android written in React Native. Key libraries we utilized were react-native-svg-charts to handle the graphing capabilities and sockjs to handle websocket communication.
Important considerations during the design and development of the application included handling the large amount of data consumed by the system which affected all parts of the app, including database design and tool choice, socket communication and graph displays.