1. The webserver used to serve the application was Flask. This is a small micro-framework which has very little overhead in terms of processing. In an environment of limited availability, this becomes a very important step. The Flask application used here is modularized to reduce co-dependency of the separate components of the application. This framework acts as a basis for integrating all the other components on top of the boilerplate. Flask also handle the different HTTP methods and provides a simple API to retrieve the sent form data as show in Fig 1, 5 and 6, accompanying files, authentication, headers and many more aspects of the HTTP packet. This enables easy retrieval of data with minimal overhead.
2. Celery is an asynchronous wrapper that allows the execution of the functions without an overhead of network bandwidth. It is a highly multi-threaded service which inputs tasks from a message queue that holds tasks that are queued by the Flask Web Server. In this app, Celery is responsible for handling the Calculation of parameters which are derived from the independent parameters as shown in Fig 2, insertion into the database and performing simple ML to better predict the behavior of battery over an extended period. Celery runs as a background service and can be scaled infinitely to support multiple threads, cores and even multiple virtual machines.

3. RabbitMQ is the most popular open source message broker. RabbitMQ is lightweight and easy to deploy on premises and in the cloud. It supports multiple messaging protocols. RabbitMQ can be deployed in distributed and federated configurations to meet high-scale, high-availability requirements. RabbitMQ runs on many operating systems and cloud environments, and provides a wide range of developer tools for most popular languages. Here, RabbitMQ is used for storing the jobs that need to be executed as a result of the Flask-server's computation as shown in Fig 6. This feeds the data directly to celery, which is a worker to execute these tasks.

4. NGINX is a free, open-source, high-performance HTTP server and reverse proxy. NGINX doesn't rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, predictable amounts of memory under load. Even if you don't expect to