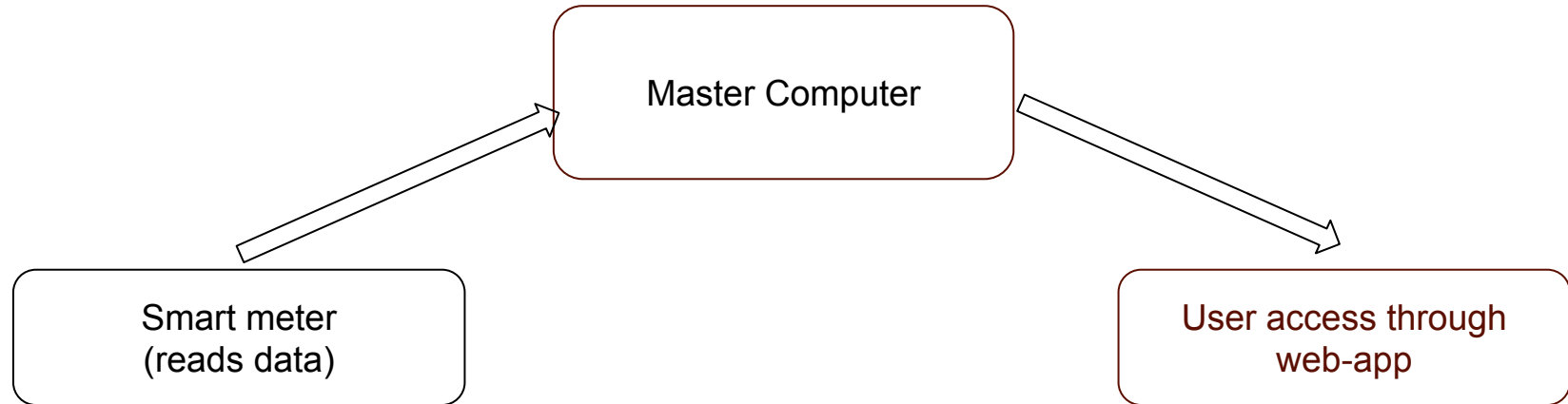


B. Tech Project: Android App for Powerplants

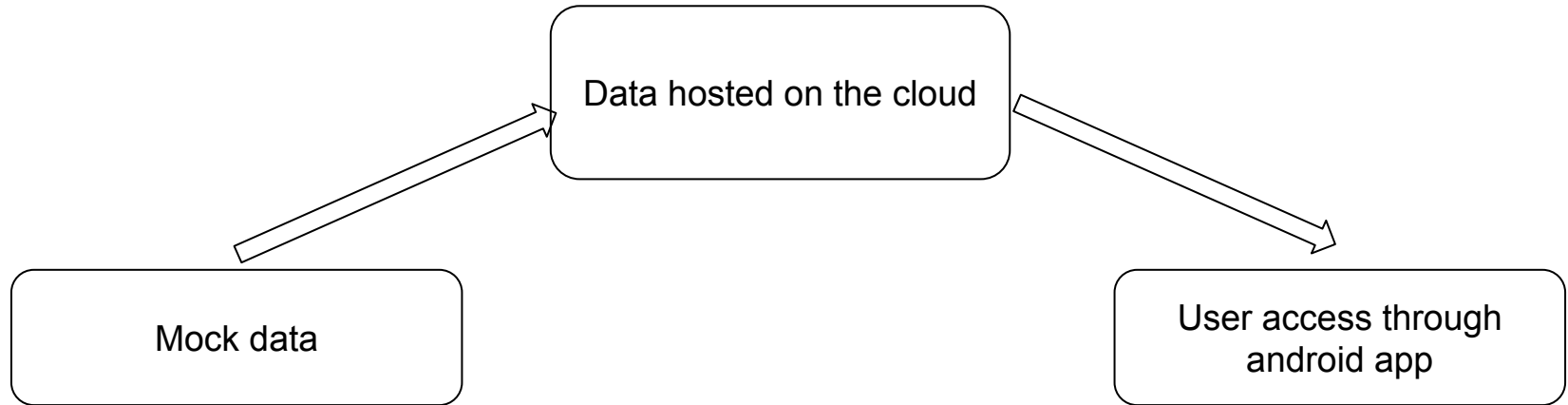
Prahlad Chaudhary (17085054)
Sarthak Choudhary (17085066)
Shashank Kumar Singh (17085068)
Vishal Indora (17085076)

Introduction

Background: Most power-plants employ smart-meters, placed around the powerplant, to detect and record various parameters. These then upload these recorded values to a master computer (all are on a common network, say WiFi), located in the control-room of the powerplant. When the value of these is needed (by the manager), he needs to be physically present in the control-room. The required parameters are then displayed to him through a web-app which he runs on the said computer.

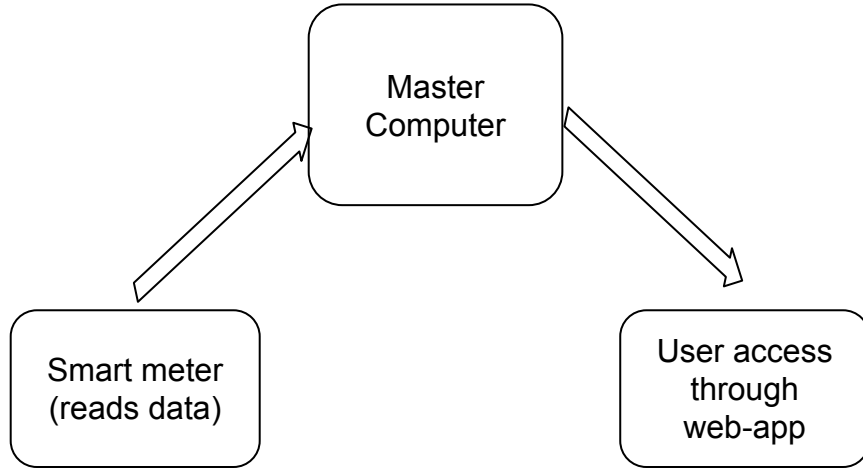


Goal: to develop an app that would mimic the present system, and on use, would allow the manager to access these parameters anywhere within the powerplant through his phone, effectively saving man-hours spent in traveling to the control-room.



Comparison: Web App vs Android App

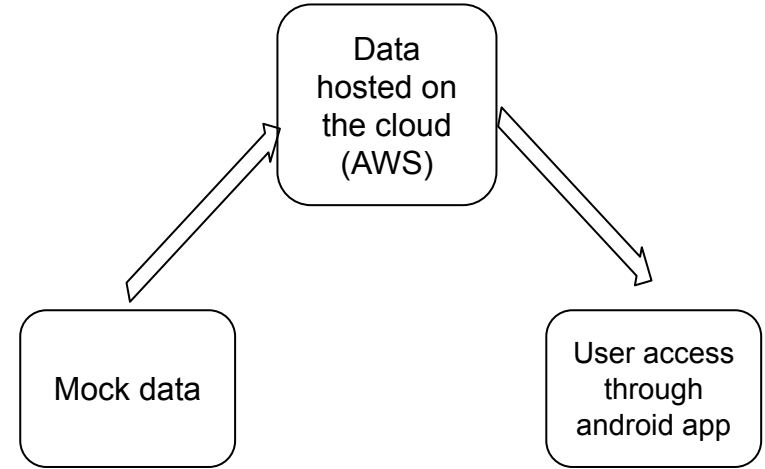
Web App



User access only through local computer

Data lost if computer damaged

Android App



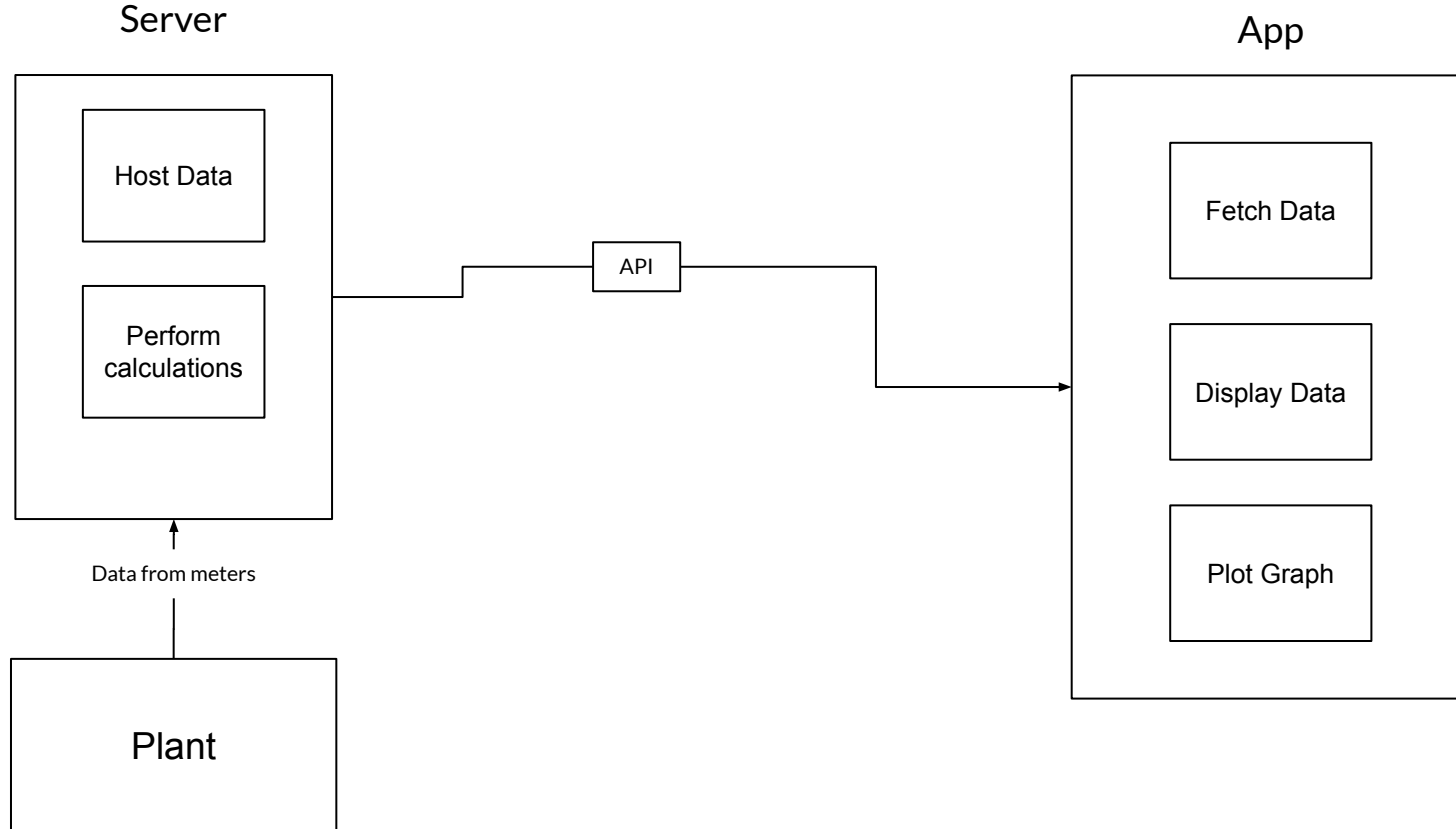
User access anywhere

Data safe

Working of the App

- The installed smart meters read the relevant data and uploads or stores it on a central server in the powerplant. This server and the smart meters are connected on a common network. The meters uploads the data and the server.
- The server has an app which features a bunch of REST APIs which are used for exchange of data between android app and server. Any calculations that are required are performed on the server itself.
- The android app fetches the data through REST API hosted on the AWS server. The data is being manipulated in a see-saw manner, i.e, every second the data is being pushed on the AWS server, and at the same moment, it is being pulled by the Retrofit API by the android app. Retrofit API helps in fetching the data through the “GET” request.

Working of the App (FLOWCHART)



Server Setup

- Since the application is completely dependent on server side REST APIs for data so we have managed to replicate the present power-plant system.
- A RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data.
- We've hosted the data on an EC2 instance on AWS (Amazon Web Services), An EC2 instance is nothing but a virtual server in Amazon Web services terminology. It stands for Elastic Compute Cloud (EC2).
- We used Python and Django REST framework to build the server side application which will have a bunch of REST APIs which will be responsible for healthy interaction of data between server and Android App. Django REST framework is a powerful and flexible toolkit for building web APIs.
- Currently, the AWS server is accessed by the android app by contacting the REST APIs. While running the app at the powerplant, it can be modified to access data from the power-plant's server APIs. It would function in the same way.

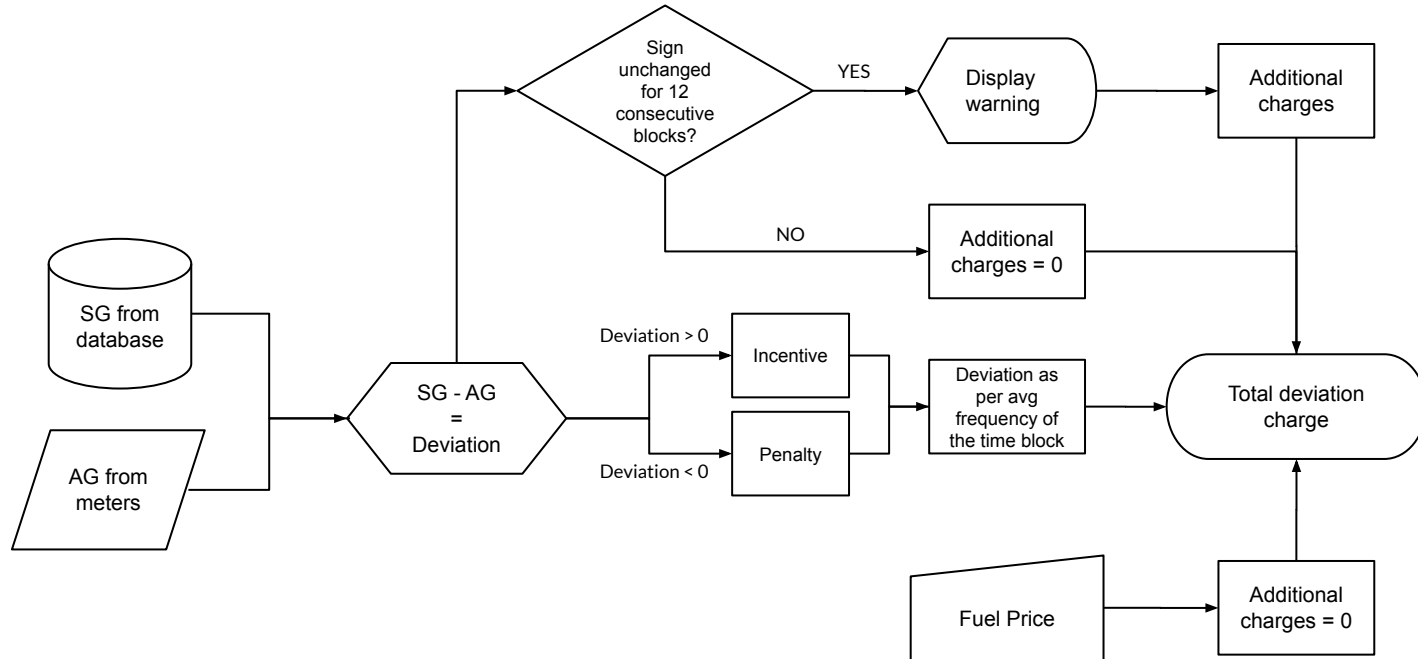
The development of the app can be divided into three main parts:

1. Preparing the mock data and performing calculations.
2. Hosting the required data on the cloud
3. Building the android app to fetch it.

Working: Mock Data and Calculations

The mock data could be divided into the following three categories: updated every second, updated every time-block (15 minutes), and updated every day. We knew the ranges of every data parameter, and we used random number generators to generate the required data.

For each time block, the total charge was calculated as per the following flowchart:



Additional Features

The following additional features are also implemented in the app:

Alarm: an alert will be shown if there is sustained deviation in one direction for more than 11 time blocks.

Prediction model: a machine learning model to predict the actual generation is also implemented.



The screenshot shows a mobile application interface with a dark background. At the top, there is a status bar with the time 18:50 and battery level 73%. Below this is a table with two columns. The first column lists various metrics, and the second column shows their values. Below the table are three blue buttons labeled 'PREVIOUS BLOCK DATA', 'CURRENT BLOCK DATA', and 'NEXT BLOCKS 30 VALUES'. At the bottom, there is a white warning message box that reads 'Warning: Sustained Positive Violation for 11 Blocks'. The word 'Alarm' is written in white text at the very bottom of the screen.

BLK. Time	13:15-13:30
Time Rem (mm:ss)	09:25
Time Elapsed (mm:ss)	05:35
Inst. BLK Hz	50.61
Avg. BLK. Hz	50.01
Inst. Ex Bus (MW)	0
Avg. Ex Bus (MW)	0
FUEL RATE	5.12
B.E.F.	49.94

PREVIOUS BLOCK DATA CURRENT BLOCK DATA NEXT BLOCKS 30 VALUES

Warning: Sustained Positive Violation for 11 Blocks

Alarm

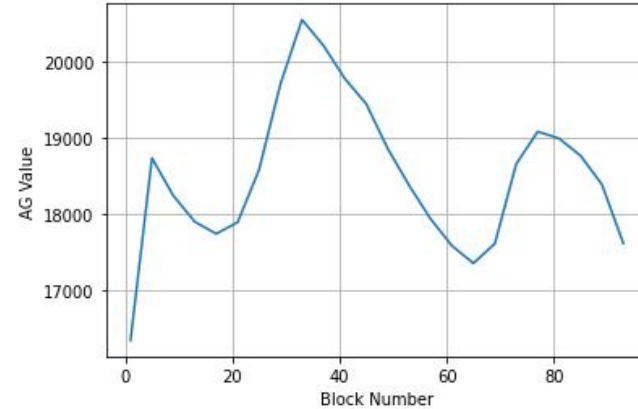
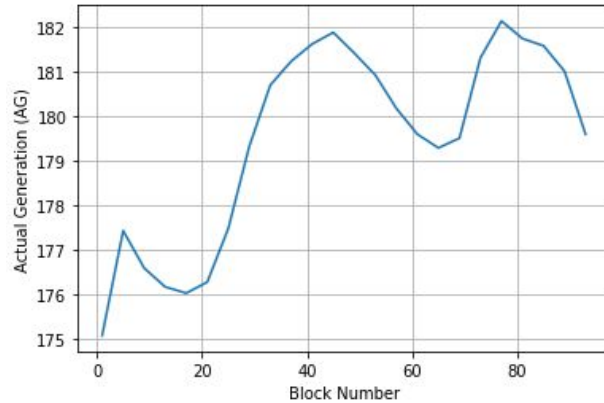
Prediction Model

Data: in the form of comma separated date-time and energy consumption (MW) values.

Datetime	AEP_MW
2004-12-31 1:00:00	13478
2004-12-31 2:00:00	12865
2004-12-31 3:00:00	12577
2004-12-31 4:00:00	12517
2004-12-31 5:00:00	12670
2004-12-31 6:00:00	13038
2004-12-31 7:00:00	13692
2004-12-31 8:00:00	14297
2004-12-31 9:00:00	14719
2004-12-31 10:00:00	14941
2004-12-31 11:00:00	15184
2004-12-31 12:00:00	15009
2004-12-31 13:00:00	14808
2004-12-31 14:00:00	14522
2004-12-31 15:00:00	14349

Prediction Model (Slide 2)

Observations: The data resembles a polynomial function with peaks around the 40th and the 80th time-block. Consequently, we decided to implement a polynomial regression model for prediction




Prediction Model: Polynomial Regression

We have data in the form of x_i (input data) and y_i (output data). We are required to predict a relationship between them. Suppose we have n such points (x_i, y_i) .

To predict a first-degree polynomial using these points, we'll consider the predicted polynomial as follows:

$$\bar{y} = (\beta_0 + \beta_1 x_i)$$

Here, β_0, β_1 are the parameters to be determined. To determine these parameters, we first get the cost function J as follows:

$$J = (\bar{y} - y_i)^2 = \sum_{i=1}^n (\beta_0 + \beta_1 x_i)^2$$


The cost function simply takes the squared error of our predicted value from the actual values.

Polynomial Regression (2)

In order to have good prediction, it is required that the value of the cost function is minimum. For this, we partially differentiate the cost function with respect to each of the unknowns as follows:

$$\frac{\delta J}{\delta \beta_0} = 0 \quad \text{And} \quad \frac{\delta J}{\delta \beta_1} = 0 \quad \text{which gives,}$$

$$\beta_0 n + \beta_1 \sum x_i = \sum y_i$$

$$\beta_0 \sum x_i + \beta_1 \sum x_i^2 = \sum y_i x_i$$

Combining the above two in matrix form, we get

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \end{bmatrix} \quad \text{Or} \quad \boxed{\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \sum y_i x_i \end{bmatrix}}$$

Polynomial Regression (3)

Similarly, if we want to predict using a quadratic polynomial of the following form:

$$\bar{y} = (\beta_0 + \beta_1 x + \beta_2 x^2)$$

We will partially differentiate with respect to β_0 , β_1 and β_2 . We'll obtain three equations which will then be combined into the following matrix form:

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{bmatrix} \quad \text{Or}$$

$$\boxed{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{bmatrix}}$$

Based on the previous discussion, we can form the following generalization for a k-degree polynomial prediction function:

1. The cost function is obtained by taking the square of the difference of the predicted value from the actual value.
2. The cost function is differentiated with respect to $\beta_0, \beta_1 \dots \beta_k$ to obtain (k+1) equations.
3. These equations are then combined into a single matrix equation and value of the β matrix is found by solving the equation by taking inverse.

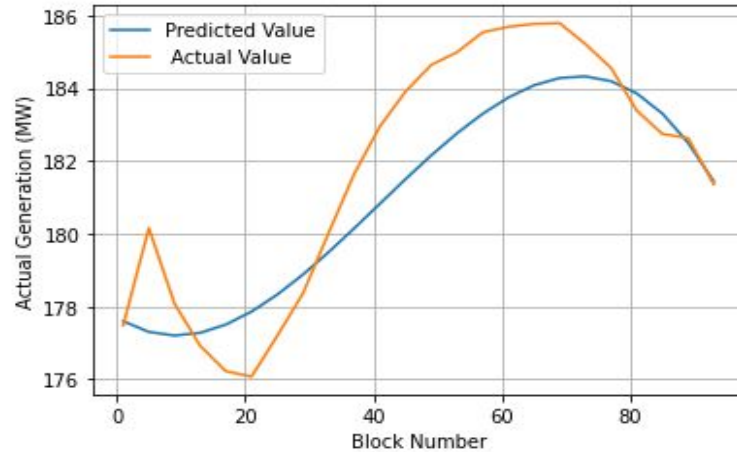
In general, we have the following:

$$\begin{bmatrix} \beta_0 \\ \dots \\ \beta_k \end{bmatrix} = \begin{bmatrix} n & \dots & \sum x_i^k \\ \dots & \dots & \dots \\ \sum x_i^k & \dots & \sum x_i^{2k} \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \dots \\ \sum x_i^k y_i \end{bmatrix}$$

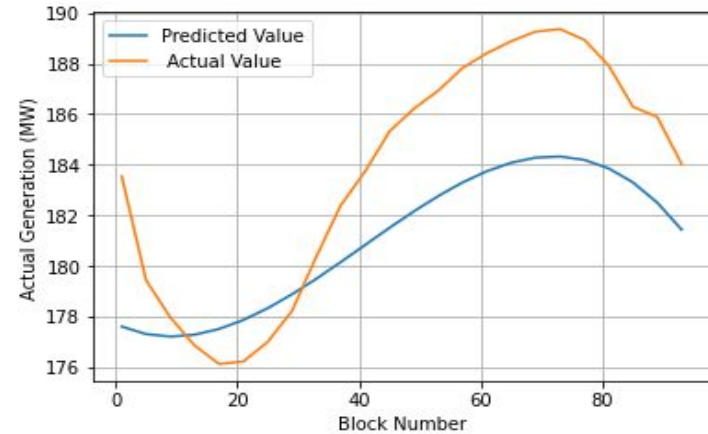
Prediction

The data was split into training and testing sets. The function for polynomial regression was trained using the training set. A split of 40% training data to 60% testing data gave the most desirable results. On further increasing the size of training data, the function lost the ability to generalize. Finally, the prediction was done on the testing set.

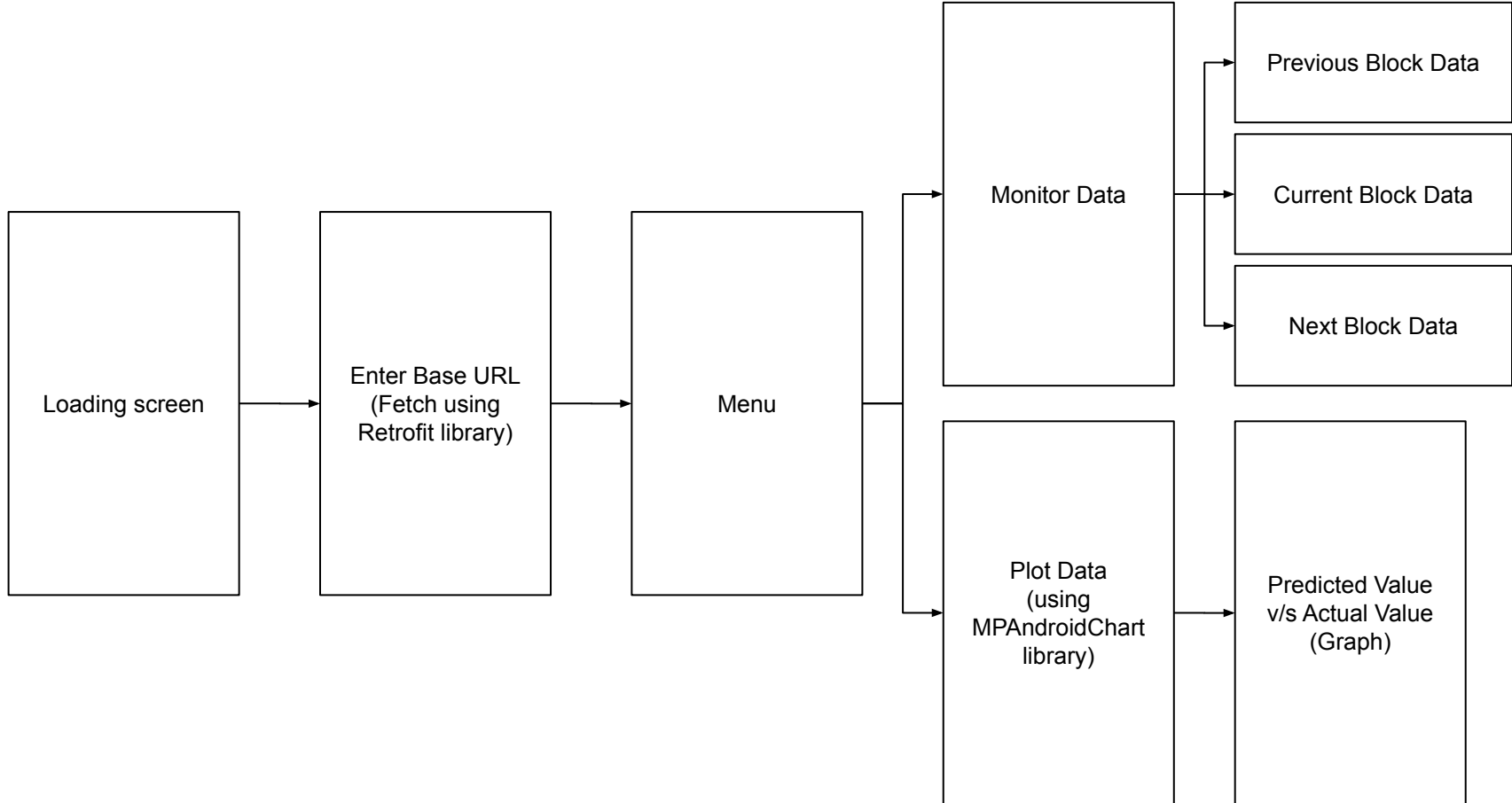
Day 1 Prediction



Day 2 Prediction



DEMO



DEMO

Result

1. The required parameters are successfully displayed in the app.
2. Prediction of the scheduled generation was achieved with sufficient accuracy.

Further Scope

- Login functionality can be added.
- An integration with a payments app (like Google Pay or Paytm) can be added so that the penalty or incentive that is payable or receivable respectively could be dealt with directly in the app.

1. Constraints and protocols for calculations of various parameters

- Letter to Generators SCED (Dated 18 April 2019) - POSOCO
- DSM 5th Amendment (Dated 28 May 2019) - Central Electricity Regulatory Commission
- Notification 132 (Dated 6 January 2014) - Central Electricity Regulatory Commission

2. Polynomial Regression Model

- Eva Ostertagová, Modelling using Polynomial Regression, Procedia Engineering, Volume 48, 12012, Pages 500-506, ISSN 1877-7058, <https://doi.org/10.1016/j.proeng.2012.09.545>.

THANK YOU