**Added Predicted Data Table and Predictionattime Python script in the Django Application**

**Introduction**

This report documents the addition of the PredictedData table to the existing Django application database and include the predictionattime script in main module for predicting the data at different points and save the result into the database. The PredictedData table is intended to store predicted sensor data. The report outlines the modifications made to the Django model (models.py) for the integration of the PredictedData table and adding the prediction script into the folder (main > management > commands).

**Model Creation**

In the existing Django application, a new model named "PredictedData" was created. This model represents a table in the database that will store predicted sensor data.

The PredictedData model includes the following fields:

* **id**: Auto-incrementing primary key.
* **date**: Date of the predicted data (max length: 15 characters).
* **time**: Time of the predicted data (max length: 15 characters).
* **light**: Light intensity (max length: 15 characters).
* **nitrogen\_data**: Nitrogen data (small integer).
* **phosphorus\_data**: Phosphorus data (small integer).
* **potassium\_data**: Potassium data (small integer).
* **relative\_humidity**: Relative humidity (max length: 5 characters).
* **temp\_c**: Temperature in Celsius (max length: 5 characters).
* **temp\_f**: Temperature in Fahrenheit (max length: 5 characters).
* **soil\_moisture**: Soil moisture (max length: 5 characters).
* **prediction**: Field for storing prediction information (max length: 25 characters).

**After Model Creation:**

* Commands need to execute in the terminal after creating the new table in the existing database. Below are the commands:
  + **python manage.py makemigrations**
  + **python manage.py migrate**
  + **python manage.py runserver # this will start the python Django server**
* In this way our new table will be registered into the Django application.

**Integrating Prediction Script into Django Application**

This section is about providing details about the integration of a prediction script into the existing Django application. The prediction script utilizes sensor data and predicts values based on the provided input.

**Prediction Script Structure**

**Command Class**

The **Command** class is a Django custom management command responsible for running predictions and saving the results. It extends **BaseCommand** and defines the **handle** method, which orchestrates the prediction process.

**Helper Functions**

1. **get\_current\_time()**: Retrieves the current time in a specified format.
2. **make\_prediction(sensor\_data)**: Makes a prediction based on given sensor data using an API endpoint and returns the result.
3. **predict\_at\_time(date, interval\_start, interval\_end)**: Predicts and saves data for a given date and time interval by fetching sensor data and making predictions based on the provided interval.
4. **predict\_daily\_average()**: Predicts and saves daily average sensor readings.

**Main Function**

The **main()** function is the entry point of the prediction script. It defines the time intervals for predictions and iterates over each interval to make predictions using **predict\_at\_time()**. Additionally, it predicts daily average sensor data using **predict\_daily\_average()**.

**Integration into Django Application**

The prediction script is integrated into the Django application through a custom management command. This allows for convenient execution and management within the Django environment. The script uses Django models to fetch sensor and daily average data, and it saves predictions into the **PredictedData** model.

**After Script Integration:**

* Commands that need to execute after successful integration of custom predictions script
  + First make sure we have our Django server up and running with command: python manage.py runserver
  + Now open the new terminal and run **python manage.py prediction\_at\_time**
* In this way the script will predict the values against the given time and predict the value for the whole day
* It is the best practice to run the script at midnight before starting new day so that it will predict the whole day result and store it in the database.

**Conclusion**

The combined efforts of adding the PredictedData table to the Django application and integrating the prediction script significantly enhance the application's capabilities. The PredictedData table allows for the organized storage of predicted sensor data, facilitating further analysis and utilization within the application. Simultaneously, the prediction script provides a structured approach to predicting sensor data, enabling real-time and daily average predictions. The seamless integration of the prediction script into the Django application further streamlines prediction processes, enhancing overall functionality and user experience.