**IOT Log Parser Dashboard Documentation**

## **1. Project Overview**

The **IoT Log Parser Dashboard** is a data visualization and analytics application designed to parse log files, extract meaningful insights, and represent the data through interactive visualizations. It is built using **Flask** for backend processing and **Matplotlib**/Seaborn for data visualization, combined with responsive front-end elements for enhanced user interaction.

## **2. File Structure**

The project files are organized as follows:

IoT-Log-Parser-Dashboard/  
│  
├── assignment\_prod.log   
├── templates/  
│ ├── dataplots.html   
├── static/   
├── app.py

README.md   
└── Iot log parser.docx

## **3. Technologies Used**

### **Backend**

* **Flask**: Lightweight Python framework for building web applications.
* **Pandas**: For data manipulation and preprocessing.
* **Matplotlib** & **Seaborn**: For creating data visualizations.
* **Python re**: For parsing log files using regular expressions.

### **Frontend**

* **HTML5 & CSS3**: For responsive and aesthetic design.

## **4. Features**

1. **Log Parsing**:
   1. Extracts log details such as timestamps, log messages, users, and action types from raw log files.
   2. Handles errors like invalid timestamps with data cleansing.
2. **Dynamic Chart Generation**:
   1. Visualizes patterns, trends, and relationships in the log data.
3. **Responsive UI**:
   1. The dashboard adapts across devices with grid-based charts and popup functionalities.
4. **Interactive Visuals**:
   1. Charts are interactive, providing users with zoomed-in views for better insights.

## **5. Data Visualizations**

### **Charts Generated**

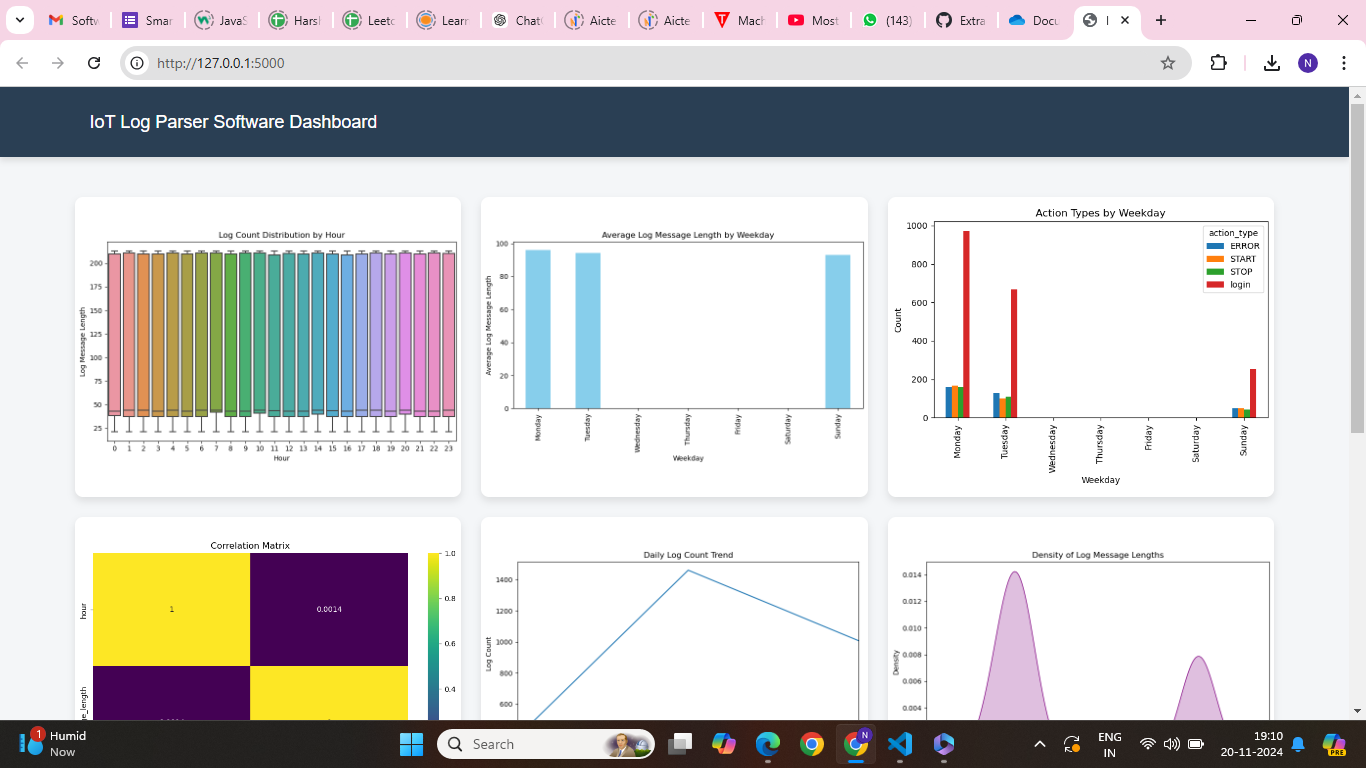
1. **Log Count Distribution**:
   1. Boxplot representing the distribution of log message lengths across different hours of the day.
2. **Average Log Message Length by Weekday**:
   1. Bar chart showing the average log message length for each day of the week.
3. **Action Types by Weekday**:
   1. Grouped bar plot displaying counts of various action types logged each day.
4. **Correlation Matrix**:
   1. Heatmap showing the correlation between hour and log\_message\_length.
5. **Daily Log Count Trend**:
   1. Line chart representing the trend of log counts over days.
6. **Density Plot of Log Message Lengths**:
   1. KDE plot visualizing the density distribution of log message lengths.
7. **Violin Plot of Message Length by Time Period**:
   1. Insights into the variation in message lengths across different times of the day.

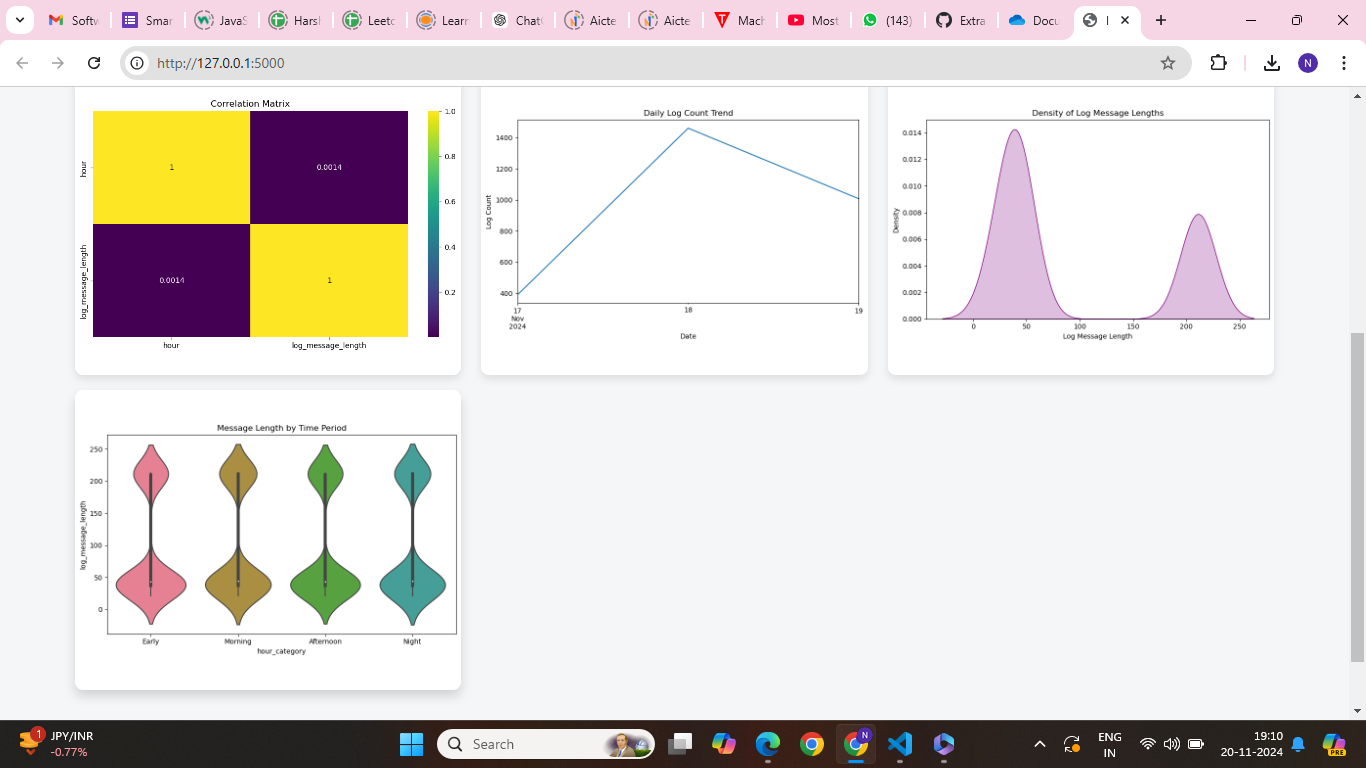
## **6. Code Walkthrough**

### **Key Components**

* **Log Parsing Logic**: The parse\_log\_file function uses regular expressions to extract timestamps, messages, and other metadata. It leverages Pandas for data preprocessing.
* **Plot Creation**: The create\_plot function dynamically generates visualizations using Seaborn or Matplotlib and returns base64-encoded images for embedding in the HTML.
* **Flask Routes**:
  + /: The main route reads the log file, generates charts, and passes them to the template.
* **Frontend Integration**: The HTML template uses Jinja2 to dynamically embed charts and JavaScript for popups.

## **7. Output**





## **8. Conclusion**

The **IoT Log Parser Dashboard** simplifies the process of extracting and visualizing data from raw IoT logs. Its robust backend ensures accurate parsing and meaningful insights, while the responsive frontend enhances user interaction. This project demonstrates how Python's data processing and visualization libraries can be effectively combined with Flask to build impactful analytics tools.