Status Report:

Roles and Members:

Vamshi Krishna Perabathula – Project lead and Data Engineering

- I'll be managing the entire setup, including creating the Docker environment, handling data ingestion and processing, building the machine learning model, and making sure everything works together to predict bike demand.

Sai Niharika Hari – Data Engineering and Model Development

- I'll be working alongside Vamshi to set up and manage Docker containers, assisting with data ingestion, processing, and storage, and collaborating on model development, testing, and deployment for accurate bike demand forecasting.

Project Goals and Data system overview:

So, I was thinking to choose prediction of bike sharing system analysis.

The main aim of this project is to create a data system that can help bike-sharing companies keep their bikes in the right places at the right times. We're using predictive analytics to figure out where and when bikes will be needed most, which will allow the companies to make sure people always have a bike when they need one, especially during peak hours.

Problem Definition: Bike-sharing programs often struggle to have enough bikes available at busy locations, while other stations have too many. This problem can lead to frustration for customers who can't find a bike when they need it, or for companies that can't efficiently manage their fleet. By predicting future bike demand, we can help bike-sharing operators keep things balanced and make users happier.

Data Overview:

- **Dataset**: We're using the **New York City Citi Bike dataset** from Kaggle, which contains lots of useful information—like when and where bikes were rented, their start and end times, and how long each trip lasted. So, i want to use this dataset
- **Real-Time Simulation**: Since we want to simulate a real-world system, I've created a Python script that will mimic real-time rental activity, streaming the bike data into the system as if it's coming in live.

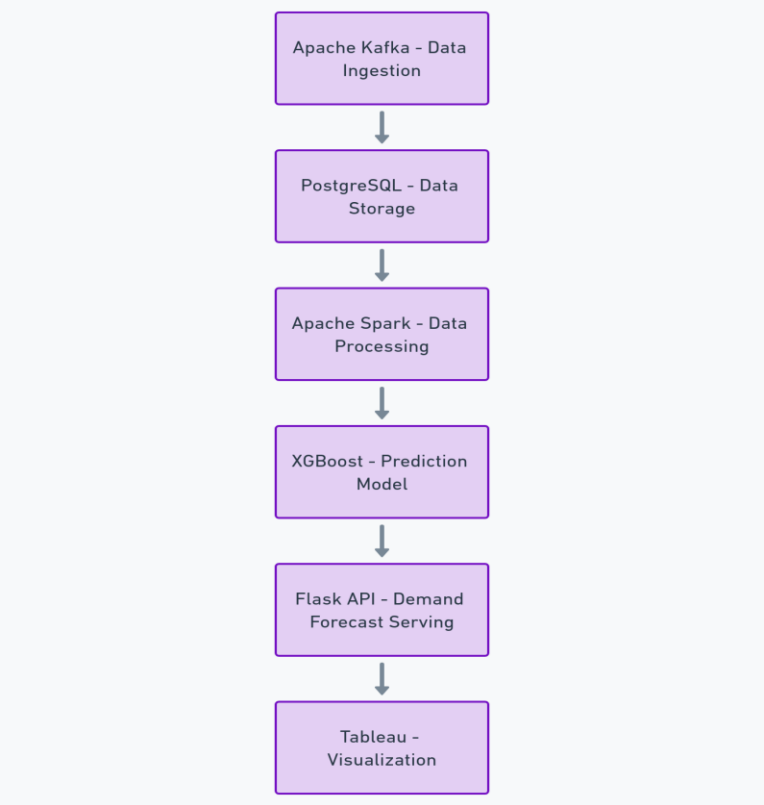
Data System Overview:

I want to integrate my data system with docker desktop with other architecture with other components for scalability, modularity and deployment for perfect outcome

1.For Data ingestion i want to use Apache Kafka which can used to to deal with real time data from each bike station is streamed into Kafka topics for further processing.
2. For Data storage purposes i want to use PostgreSQL which i used this for healthcare course for storing the data in server and we can retrive the data which allows efficient querying the data for analysis.
3. For Data Processing we are using Apache Spark which is used to clean and analyzing purpose
4. For machine learning model for evaluation and predictions i want to use XGBoost, a powerful machine learning model that can predict future bike demand based on historical patterns This means we can tell which stations will need more bikes in the next hour and ensure they're available.
5. At last, we are going to use API and Visualization where Flask API will allow bike sharing companies to check demand and plan accordingly. For this I'm going to use docker for easy management
6. Lastly for creating dashboards with Tableau which is used to visualize everything. This makes it easy for decision-makers to see the data in a visual format.

Data Flow Diagram

Kafka (data ingestion) → **PostgreSQL** (data storage) → **Apache Spark** (data cleaning/processing) → **XGBoost** (prediction) → **Flask API** (serving demand forecasts) → **Tableau** (visualization)



Challenges and Issues Foreseen

- **Demand Spikes**: One big challenge is predicting when there will be a sudden surge in bike demand. Things like holidays, events, or unexpected weather changes can drastically alter bike needs, and our model might have difficulty keeping up with these spikes. We may need to add extra data sources, such as local events or weather forecasts, to improve our accuracy.
- **Data Quality**:

    - **Missing or Incorrect Data**: Since the dataset includes many records, there are bound to be missing or incorrect values—like incomplete trip details or inaccurate times. We need to preprocess the data to handle these gaps. **Apache Spark** will be doing the heavy lifting here, making sure the data is clean before analysis.
    - **Real-Time Synchronization**: Simulating real-time bike data and getting it streamed into Kafka can be tricky. The data from multiple stations might not all arrive at the same time, so we need to ensure our system can synchronize everything smoothly.
- **Scalability**: Although Docker makes deploying these components straightforward, coordinating multiple containers (Kafka, Spark, PostgreSQL, Flask, and Tableau) can be challenging. As the project grows and the dataset gets bigger, managing communication between these services effectively will be crucial.

Next Steps:

- **Completing Docker Setup**: Finalizing the setup of Docker containers for **Kafka**, **PostgreSQL**, **Spark**, **Flask**, and **Tableau**.
- **Real-Time Simulation**: Beginning simulating the data flow from bike rentals to Kafka using the Python script.
- **Data Processing**: Starting processing the data with **Apache Spark** to clean it and extract useful features for predictions.
- **Model Training**: Training the **XGBoost** model with historical data to ensure our predictions are accurate.
- **Visualization and API Deployment**: Deploying the **Flask API** to serve demand forecasts and connect **Tableau** to visualize bike availability and predictions.

References:

- **Dataset**: New York City Citi Bike dataset from Kaggle.
- https://www.kaggle.com/datasets/akkithetechie/new-york-city-bike-share-dataset.
- **Technologies**: Docker Desktop, Apache Kafka, PostgreSQL, Apache Spark, Flask, Tableau