Project Progress Report

1. Which tasks have been completed?

We've successfully implemented querier.py to interact with our Flask web service, enabling us to feed and query data. This forms the foundation of our data ingestion process.

The Flask app outlined in view.py is operational and houses our video play count cache. This script facilitates the caching and retrieval of data.

We have developed mock_user_traffic.py to generate synthetic user traffic. This script creates concurrent access to video data, simulating real-world conditions and allowing us to test our system's response to load.

We have created map_reduce_worker.py script to lay the groundwork for our map-reduce operations. It is designed to process and analyze large volumes of data efficiently.

2. Which tasks are pending?

The map-reduce functionality needs to be fully fleshed out. This includes optimize the mapper and reducer functions to accurately process the incoming data and generate a list of the top 10 most viewed videos.

The rendering of results, which will display the top videos, requires further development. This involves both backend logic to sort and retrieve the top videos and frontend development for presenting these results to the end-user.

We need to test the system's scalability, ensuring it can handle increased loads without performance degradation. This will likely involve iterative testing and optimization.

Implementing comprehensive evaluation metrics to measure efficiency, accuracy, and scalability is pending. Quality assurance processes must be established to ensure the system's reliability and correctness.

3. Are you facing any challenges?

We've encountered challenges in optimizing the map-reduce operations to handle the simulated traffic loads efficiently. This requires some adjustments to the threading model and data handling processes.

The management of video play counts within a cache has presented difficulties, particularly concerning data consistency and retrieval speed. Solutions involving better designed caching strategies are being considered.

Ensuring thread safety and managing the concurrent execution in mock_user_traffic.py has proven complex.