Customer Usage Analytics System - Project Documentation

1. Introduction

The system captures and analyzes telecom customer usage data such as calls, SMS, and internet data sessions. It enables telecom operators to generate insights on customer behavior, usage patterns, revenue streams, and network utilization. The system uses a Python-based backend with a MySQL database for storing usage logs and a console interface to display formatted analytics reports. This design provides a foundation for efficient data collection, aggregation, and reporting to aid decision-making in marketing, billing, and customer retention.

2. Module Overview

The system consists of the following key modules:

2.1 Data Collection Module

- Logs and stores customer usage details (calls made/received, SMS sent, internet data sessions) with timestamps.
- Validates and inserts data into the MySQL database.

2.2 Database Management Module

- Manages the MySQL database connection, schema setup (tables for customers, calls, SMS, and data sessions), and CRUD operations.
- Implements optimized SQL queries for data aggregation and retrieval.

2.3 Usage Analytics Module

- Computes metrics like total calls, total SMS, total data consumed per customer.
- Identifies peak usage hours from time-stamped data.

- Supports per-customer and overall network analytics.

2.4 Reporting Module

- Uses the PrettyTable Python library to generate well-formatted console reports.
- Presents customer-wise summaries and aggregated network statistics.

2.5 User Interface Module (Console Based)

- Provides a menu-driven console interface for data input and analytics report selection.
- Handles interaction flow, input validation, and navigation.

2.6 Utilities and Helper Functions

- Facilitates time conversions, data formatting, input validation, and error handling.

3. Architecture Overview

3.1 Architectural Style

- Backend: Python application logic interacting with a MySQL relational database.
- Interface: Console-based menu-driven interface for user operations and reporting.
- Data Storage: MySQL storing structured usage logs and customer information.

3.2 Component Interaction

1. **Data Collection Module** captures usage logs and inserts into the database via the **Database Management Module**.
2. **Usage Analytics Module** queries aggregated data from the database.
3. The **Reporting Module** formats this analyzed data using PrettyTable and displays it in the console.
4. The **User Interface Module** drives interaction, accepting commands and showing reports.
5. **Utilities Module** supports data conversion, validation, and error messaging throughout.

3.3 Technology Stack

- Programming Language: Python (for backend logic and reporting)
- Database: MySQL (for data persistence and query support)
- Interface: Console (menu-driven)
- Libraries: PrettyTable (for tabular reporting)
- Development Environment: Local setup for development and testing

4. Module-wise Design

4.1 Data Collection Module

- Features: Input logging for calls, SMS, and data sessions with timestamps.
- Data Flow: Receives input → validates data → inserts into MySQL tables → confirms success/failure.

4.2 Database Management Module

- Features: MySQL connection pooling, schema creation, CRUD operations.
- Data Flow: Provides interfaces for inserting new logs and querying aggregated data via optimized SQL queries.

4.3 Usage Analytics Module

- Features: Aggregation of usage data for each customer and overall network statistics.
- Data Flow: Receives requests for metrics → runs SQL aggregation queries → returns results for reporting.

4.4 Reporting Module

- Features: Creates customer-wise and network summary reports using PrettyTable.
- Data Flow: Accepts analytics data → formats into tables → outputs to console.

4.5 User Interface Module

- Features: Menu-driven interaction, options to log data, generate reports, and exit.
- Data Flow: Displays menu → accepts user input → triggers respective module functions → repeats or exits.

4.6 Utilities and Helpers

- Features: Functions for time conversions (e.g. peak hour determination), data validation, and error handling.

5. Non-Functional Requirements

- Performance: The system should handle simultaneous logging and reporting for multiple customers with prompt output.
- Usability: Menu interface should be intuitive with clear prompts and error messages.
- Scalability: Modular design supports future extension to GUI or web interfaces and advanced analytics.
- Security: Basic validation for input data; further work needed for authentication and encryption.