Project: Network Outage Impact Analyzer

1. Introduction

The **Network Outage Impact Analyzer** is a Python-based console application designed to assess the business impact of telecommunication network outages. While traditional monitoring tools focus on fault detection, this system translates outages into business metrics such as customer impact, revenue loss, and SLA penalties.

2. Module Overview

The project consists of the following modules:

2.1 Data Collection

- Log outage events (region, tower ID, start time, duration).
- Store subscriber count and ARPU (Average Revenue Per User) for each region.

2.2 Impact Analysis

- Calculate number of customers affected.
- Estimate revenue loss per hour.
- Compute SLA penalty costs.
- Aggregate total outage duration per region.

2.3 Reporting

- Generate region-wise outage reports.
- Display total customers affected and overall revenue loss.
- Present data in a tabular console view.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Console-based interface
- **Backend**: Python application
- **Database**: CSV or SQLite (for local storage)

3.2 Component Interaction

1. User inputs outage data via console.
2. Backend processes data and performs calculations.
3. Results are displayed in tabular format in the console.

4. Module-Wise Design

4.1 Data Collection Module

Features:

- Add outage logs.
- Store regional subscriber and ARPU data.

Data Flow:

1. User enters outage details.
2. System stores data in local storage.

Entities:

- **OutageLog**: Region, TowerID, StartTime, Duration
- **RegionData**: Region, SubscriberCount, ARPU

4.2 Impact Analysis Module

Features:

- Calculate affected customers.
- Estimate revenue loss and SLA penalties.

Data Flow:

1. System retrieves outage and region data.
2. Performs calculations.
3. Stores results for reporting.

Entities:

- **ImpactReport**: Region, CustomersAffected, RevenueLoss, SLAPenalty

4.3 Reporting Module

Features:

- Generate summary reports.
- Display data in tabular format.

Data Flow:

1. User requests report.
2. System compiles and displays data.

Entities:

- **ReportTable**: Region-wise summary of impact metrics

5. Deployment Strategy

5.1 Local Deployment

- Application runs locally using Python.
- No external dependencies required.

5.1.1 Console Interface

- Python CLI for user interaction.

5.1.2 Backend Deployment

- Python scripts executed locally.

5.1.3 Database

- CSV or SQLite used for data persistence.

6. Database Design

6.1 Tables and Relationships

- **OutageLog**: Primary Key – LogID
- **RegionData**: Primary Key – Region
- **ImpactReport**: Foreign Key – Region

7. User Interface Design

7.1 Console Views

1. Outage Entry Form
2. Region Data Entry
3. Impact Summary Table
4. Region-wise Report Display

8. Non-Functional Requirements

8.1 Performance

- Handles up to 50 outage logs efficiently.

8.2 Scalability

- Can be extended to use web interface or cloud storage.

8.3 Security

- Local data access only; no external exposure.

8.4 Usability

- Simple CLI prompts and tabular output for clarity.

9. Assumptions and Constraints

9.1 Assumptions

- All data is entered manually or simulated.
- ARPU and subscriber data are accurate.

9.2 Constraints

- No real-time data integration.
- Limited to local execution.