MedTechInnovators: Medical Facility Digital Twin

Overview

This project demonstrates a basic digital twin system for simulating and visualizing patient flow through different departments in a medical facility. It incorporates real-time data generation, quality checks, and a simple visualization to track the movement of patients across departments.

The goal is to showcase foundational data management, real-time simulation, and visualization capabilities as a part of understanding and implementing a digital twin system.

Features

- Real-Time Data Collection: Simulates patient arrivals and movements using Python.
- Data Quality Checks: Validates patient data for accuracy and flags any inconsistencies.
- Visualization: Provides a graphical representation of patient flow in different departments.

System Architecture

1. Input:
   - Simulated patient data (randomly generated in real time).
2. Processing:
   - Validates the data for missing fields or incorrect department names.
   - Stores validated data in a .csv file for further analysis.
3. Output:
   - Real-time bar charts showing the number of patients in each department.

System Diagram

```
[ Data Simulation ] → [ Data Validation ] → [ Visualization ]
```

Setup Instructions

1. Install Python 3.8 or higher.
2. Install the required libraries

```
pip install -r requirements.txt
```

How to Run

1. Clone this repository: `git clone <https://github.com/SumeshSurendran12/MedTechInnovators>`
2. Navigate to the project directory: `cd MedTechInnovators`
3. Run the simulation: `python src/generate_data.py`
4. Visualize patient flow: `python src/visualize_data.py`

Sample Output

- A real-time bar chart showing the distribution of patients across departments.

Project Structure

```
MedTechInnovators
    src
        generate_data.py
        validate_data.py
        visualize_data.py
    data
        generated_data.csv
    tests
        test_generate_data.py
        test_validate_data.py
  Dockerfile
  README.md
  requirements.txt
  LICENSE
```

Key Algorithms

1. Data Generation:
    - Randomly generates patient data with unique IDs, timestamps, and assigned departments.
    - Simulates real-time behavior using Python's time.sleep().
2. Data Validation:
    - Checks for missing or incorrect patient records.
    - Validates department names and logs invalid entries.
    - Outputs clean data for visualization.
3. Data Visualization:

- Displays patient distribution using bar charts.
- Implements real-time updates for a dynamic view of patient flow.

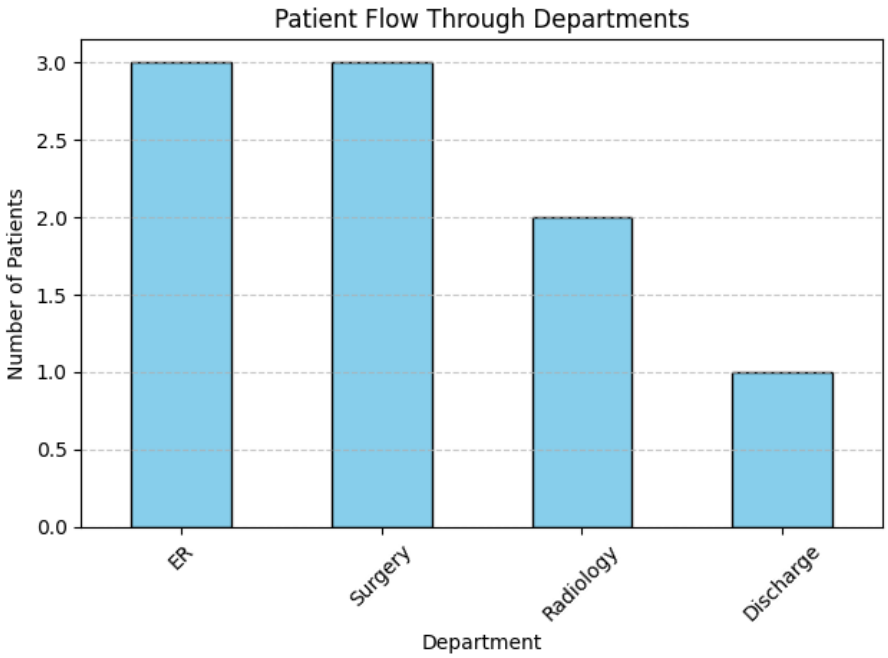

Figure 1: Figure_1

Team Contributions

Sumesh Surendran: System Architecture and Project Setup

- Designed the overall system architecture.
- Created the project repository and folder structure.
- Drafted the initial README.md file and set up dependencies.

Muskaan Shahzad: Data Simulation

- Developed the script for generating patient flow data.
- Simulated real-time patient arrivals and department transitions.
- Created sample data files for testing.

Benjamin Bui-Dang: Data Quality Checks

- Implemented validation rules for missing and inconsistent data.
- Wrote logic to handle invalid department entries.
- Ensured clean and reliable data for downstream use.

Autry Marshall: Visualization and Output

- Created real-time visualizations of patient flow.
- Used bar charts to represent patient distribution across departments.
- Handled dynamic updates for real-time simulations.

Shanecia Holden: Documentation and Presentation

- Coordinated the creation of the pre-recorded video presentation.
- Compiled technical documentation, including challenges and solutions.
- Finalized all deliverables for submission.

Challenges and Solutions

Challenges:

1. Real-Time Simulation:
   - Generating realistic data in real-time was initially challenging.
2. Data Validation:
   - Handling invalid or missing fields required robust checks.
3. Dynamic Visualization:
   - Updating visualizations in real time while ensuring accuracy.

Solutions:

1. Used `time.sleep()` to simulate real-time behavior effectively.
2. Wrote modular validation functions to handle data inconsistencies.
3. Leveraged Python libraries like matplotlib for real-time chart updates.

License