Project Report: From Unstructured Text to Interactive Application

Table of Contents

1. Executive Summary

This report documents the design, development, and implementation of a data processing pipeline that transforms raw, unstructured text containing bus schedule information into a fully interactive web application. The project successfully addresses the challenge of parsing complex text strings to extract meaningful data points, such as timings, price, and availability. The implementation was carried out in three distinct phases: initial data extraction into a structured format using Python's re and pandas libraries; persistent storage of the structured data in an SQLite database; and finally, the development of a user-friendly graphical interface using Streamlit that allows for dynamic filtering and data exploration. The final result is a functional prototype of a bus booking dashboard, demonstrating a complete and practical workflow from raw data to a valuable end-user application.

2. Introduction

2.1. Problem Statement

The source data for this project was provided as a multi-line string of text, where each line represented a different bus service. An example line is:
21:45 06:45 9h 29 Seats (8 Single) 1,289 Onwards IntrCity SmartBus A/C Seater / Sleeper (2+1)
This format, while readable by humans, is unsuitable for automated processing, analysis, or querying. The lack of a consistent structure, a mix of data types, and extraneous text make it

impossible to perform tasks such as filtering by price, sorting by departure time, or aggregating data by operator without a preliminary parsing step. The core problem was to bridge the gap between this unstructured textual data and a structured, machine-readable format.

2.2. Project Objectives

The primary objectives of this project were:

1. **To parse and extract** key data fields (Departure Time, Arrival Time, Duration, Seats Available, Price, Operator) from the raw text.
2. **To structure** the extracted information into a clean, tabular format for ease of use.
3. **To implement a persistent storage** solution to store the structured data, enabling efficient querying and retrieval.
4. **To develop an interactive user interface** that allows end-users to dynamically filter and explore the bus data without needing technical expertise.

3. System Architecture and Technologies

3.1. System Architecture

The project follows a logical three-tier data flow:

1. **Data Extraction Layer:** Raw text is processed using Python scripts. Regular expressions identify and extract relevant data points.
2. **Data Storage Layer:** The extracted and cleaned data is stored in a structured SQLite database table, creating a reliable single source of truth.
3. **Presentation Layer:** A Streamlit web application queries the database (or an in-memory representation) and presents the data to the user through an interactive dashboard with various filtering controls.

3.2. Technologies Used

- **Python:** The core programming language used for all scripting and development.
- **re (Regular Expressions):** The primary tool for pattern matching within the text strings. It was essential for accurately identifying and isolating specific data fields. For instance, \b\d{2}:\d{2}\b was used to find timestamps, and (\d+) Seats was used to extract the number of available seats.
- **pandas:** This library was used to create and manage DataFrames, which are two-dimensional, labeled data structures. The DataFrame served as an intermediary in-memory container for the cleaned data before it was loaded into the database or used by the application.
- **sqlite3:** A lightweight, serverless, self-contained SQL database engine. It was chosen for its simplicity and ease of integration into a Python application for creating a persistent local database.
- **streamlit:** A Python framework for rapidly building and sharing web applications for data science and machine learning. It was used to create the interactive front-end, including

widgets like sliders, multi-select boxes, and data tables.

4. Implementation Details: A Phased Approach

The project was developed incrementally across three major phases, each building upon the last.

4.1. Phase 1: Initial Data Extraction and Structuring

The first version of the code focused solely on parsing the raw text and organizing it.

- **Process:** The script began by splitting the raw text block into a list of individual lines. It then iterated through each line.
- **Extraction Logic:** Within the loop, a series of regular expression searches (re.search and re.findall) were performed to locate and capture each required data point. The code included checks to handle cases where a pattern might not be found, assigning None to prevent errors.
- **Structuring:** The extracted values for each field were appended to their respective lists (departure_time, arrival_time, etc.). Finally, these lists were passed into a dictionary to create a pandas DataFrame, achieving the initial goal of converting unstructured text into a structured table.

4.2. Phase 2: Database Integration for Persistent Storage

This phase introduced a database to store the cleaned data, making it persistent and more robust for querying.

- **Database and Table Creation:** The sqlite3 module was used to connect to a database file (busdata.db). A SQL CREATE TABLE IF NOT EXISTS statement was executed to define a buses table with appropriate columns and data types (e.g., Bus_ID INTEGER PRIMARY KEY, Price INTEGER, Operator TEXT).
- **Data Insertion:** The data extraction logic from Phase 1 was adapted. Instead of just creating a DataFrame, the script now generated a list of tuples, where each tuple represented a row to be inserted into the database. The cursor.executemany() method was used for an efficient bulk insert of all bus records.
- **Querying Demonstration:** To validate the database integration, a simple SQL query (SELECT * FROM buses WHERE Price < 1600) was executed, and the results were loaded into a DataFrame, demonstrating the ability to filter data directly from the database.

4.3. Phase 3: Building an Interactive Web Application

The final phase brought the project to life by creating a user-facing application with Streamlit.

- **UI Design:** The application was given a title (st.title) and a sidebar (st.sidebar.header) to house the filter controls.
- **Interactive Controls:** The sidebar was populated with various Streamlit widgets:
  - st.sidebar.multiselect for selecting one or more bus operators.
  - st.sidebar.select_slider for choosing a departure time range.

    - st.sidebar.slider for setting a price range and an available seats range.
- **Dynamic Filtering Logic:** The core of the app's interactivity lies in its filtering mechanism. The main DataFrame was filtered using boolean indexing based on the current values of the sidebar widgets. This ensures that the displayed data is always in sync with the user's selections.
- **Data Presentation:** The filtered DataFrame was rendered on the main page using st.dataframe. An additional st.selectbox was added to allow users to pick a specific bus from the filtered list and view its full details in a neatly formatted block.

5. Results and Discussion

5.1. Final Product

The final outcome is a fully functional, interactive dashboard for exploring bus data. Users can intuitively filter a list of buses based on multiple criteria simultaneously, such as finding all buses from "IntrCity SmartBus" that depart after 21:00 with a price under ₹1,600. The application is responsive, easy to use, and effectively hides the complex data processing happening in the background.

5.2. Significance and Key Achievements

- **Data Accessibility:** The project successfully made a complex, raw dataset accessible and explorable for non-technical users.
- **Complete Pipeline:** It demonstrates a complete, end-to-end data pipeline: from ingestion and cleaning to storage and interactive visualization.
- **Rapid Prototyping:** The use of Streamlit showcases the power of modern tools for quickly building powerful data applications with minimal code.
- **Foundation for Analytics:** The structured database and clean data serve as a solid foundation for future, more advanced analysis, such as price prediction modeling, demand forecasting, or route optimization.

6. Conclusion and Future Work

This project effectively transformed a static block of unstructured text into a dynamic and valuable data tool. It highlights the importance of data cleaning and structuring as a critical first step in any data-driven project. The final Streamlit application serves as a powerful proof-of-concept for how raw information can be made insightful and interactive.

Future work could include:

- Connecting the application to a live, real-time data source instead of a static text file.
- Adding more advanced features like sorting by columns, user ratings, and bus amenities.
- Deploying the application to a cloud service to make it publicly accessible.
- Using the cleaned dataset to train a machine learning model for tasks like price prediction.