The Journal of Open Source Software

Automating GTFS Feed Generation for Public Transit: A Case Study of Islamabad's Orange Line

Muawia Irfan [1¶], **Irfan-ul-Haq Qureshi** [1], **Rabeeh Ayaz Abbasi** [1], **Hifza Irfan** [2], **Imran Sabir** [2], **and Muhammad Zaman** [2]

1 Department of Computer Science, Quaid-i-Azam University, Islamabad **2** School of Sociology, Quaid-i-Azam University, Islamabad ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 07 August 2025
Published: unpublished

Summary

Public transportation is the backbone of urban mobility, and its effectiveness relies heavily on accessibility. The General Transit Feed Specification (GTFS) offers a standardized format for representing transit schedules and spatial data, and enables easy integration with digital platforms like Google Maps. This paper outlines the preparation and deployment of GTFS data for Islamabad's Orange Metro Bus Line. Instead of manual transcription or complex automated systems, we propose a semi-automated scripting-based approach that is efficient, user-friendly, and adaptable for transit agencies. To encourage broader GTFS adoption, especially in resource-limited settings, we have made our code publicly available.

Statement of Need

In recent years, Islamabad has significantly expanded its public transport network with the addition of the Red, Green, Blue, and Orange metro bus lines, along with the launch of an electric vehicle (EV) service. However, a major yet overlooked challenge remains: The lack of accessible digital transit information for daily commuters. To address this gap, we implemented GTFS for the Orange Metro Bus Line and developed semi-automated Python scripts to efficiently generate standardized transit feeds.

Methodology

This section explains how the GTFS data for the Orange Line Metro Bus in Islamabad was developed and it covers the process of collecting relevant data, organizing it into GTFS format, and using Python scripts to automate different steps.

Data Collection

A key component of our dataset was the stop coordinates and bus schedules for the Orange Line Metro. Although the transit authority provided basic station locations and timings, the data was incomplete. To address this, we manually collected additional entry and exit point coordinates using Google Maps, ensuring precise latitude and longitude values for each station. This enhanced the dataset's accuracy and reliability for both analysis and mapping.

GTFS Components

While preparing the GTFS dataset, we created several standardized text files to define various components of the transit system, following the GTFS specification (*GTFS Reference Documentation*, 2023). - The agency.txt file contains basic information about the transit authority, such as its name, website, time zone, and a unique agency ID. - routes.txt stores route-level data, including route names, IDs, types (e.g., bus or metro), and associated agency. - trips.txt

links each route to specific trips, while stop_times.txt defines the schedule recording arrival and departure times at each stop. - stops.txt lists all stop locations with their names and precise coordinates. - calendar.txt specifies the days on which services operate (e.g., weekdays, weekends, or specific dates). - fare_attributes.txt defines fare rules, including ticket prices, payment methods, and transfer policies. - fare_rules.txt links specific fares to routes, where applicable.

Together, these files form a complete GTFS feed that enables effective organization, mapping, and trip planning. Figure 1 illustrates the GTFS schema, highlighting the relationships between stops, routes, trips, and schedules, and how these files interconnect

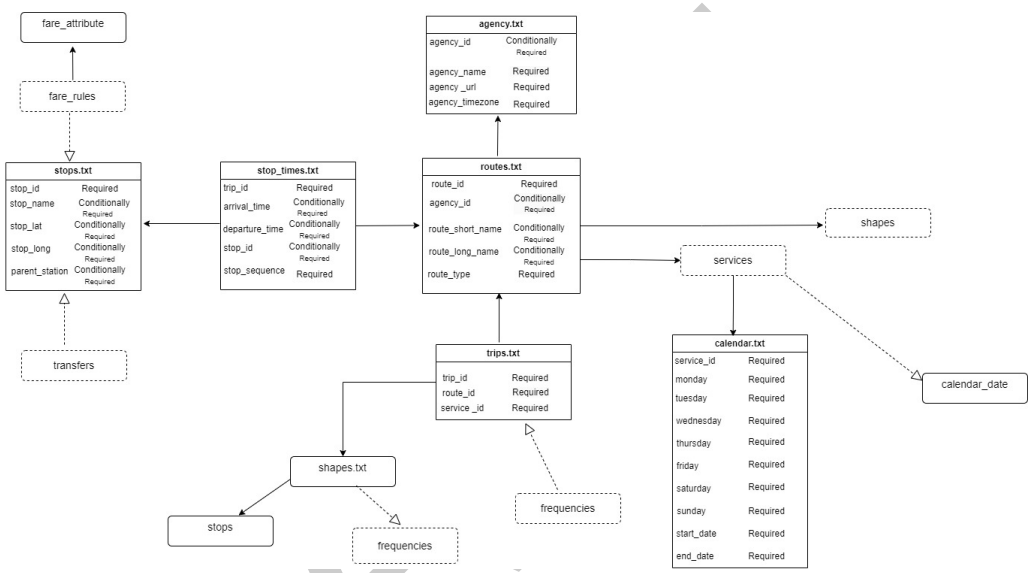

Figure 1: GTFS schema showing relationships between stops, routes, trips, and schedules.

Automation Using Python

Some GTFS files, such as trips.txt and stop_times.txt, require large and complex datasets. However, most other GTFS files are relatively simple and can be created manually with ease. To streamline the process for the more data-intensive files, we developed two Python scripts to automate their generation. Below is the pseudocode for these scripts, which create the stop_times.txt and trips.txt files.

Script 1: Generating stop_times.txt

To generate stop_times.txt in the GTFS feed automatically, we suggest the following approach. This script automates the assignment of stop sequences, arrival times, and departure times, minimizing manual effort and ensuring efficient scheduling.

Input Parameters

- **T_s**: Start time of first trip (e.g., `"06:00:00"`)
- **T_e**: Start time of last trip (e.g., `"23:00:00"`)
- **G**: Gap time between trips (in minutes)
- **D**: Dwell time at each stop (in seconds)
- **S**: Ordered list of stop IDs
  $S = [s_1, s_2, \ldots, s_n]$ where $n \geq 2$
- **τ**: Travel times between stops
  $\tau = [\tau_1, \tau_2, \ldots, \tau_{n-1}]$ where $\tau_i \geq 1$
- **TripID$_0$**: Initial trip ID, ending in a numeric suffix (e.g., `"M-01"`)

Output

- GTFS-compliant `stop_times.txt` file
- Each row:
  `(trip_id, arrival_time, departure_time, stop_id, stop_sequence)`

Pseudocode

```
Initialize:
    t ← TripID□
    T ← T_s
    □ ← ∅

While T ≤ T_e do:
    cumulative_time ← 0

    For i from 1 to n do:
        a_i ← T + cumulative_time
        d_i ← a_i + D
        □ ← □ ∪ {(t, a_i, d_i, s□, i)}

        If i < n then:
            cumulative_time ← cumulative_time + τ□ + D

    t ← IncrementTripID(t)
    T ← T + G
End While

Output □ to stop_times.txt
```

```
Function IncrementTripID(t):
    Extract numeric suffix x from t
    Convert x to integer and increment by 1
    Pad x with leading zeros to match original length
    Return prefix(t) + x
```

Script 2: Generating trips.txt

The following algorithm is designed to generate trips.txt in the GTFS feed automatically. This script systematically creates trip identifiers, assigns route and service details, and structures the dataset for integration into public transportation systems.

Input Parameters

- R: Set of routes
  Each route (r R) has:
  - `route_id□`: Unique route identifier
  - `service_id□`: Service calendar ID
  - `headsigns□ = [h□, h□, ..., h□]`: Trip headsigns per direction
  - `num_trips□ = [n□, n□, ..., n□]`: Number of trips for each headsign
  - `block_ids□ = [b□, b□, ..., b□]` (optional)
  - `shape_ids□ = [s□, s□, ..., s□]` (optional)
  - `trip_id□□`: Starting trip ID (e.g., `"M-001"`)

Output

A set T = { (trip_id, route_id, service_id, headsign, direction_id, block_id, shape_id) } is written to `trips.txt` following the GTFS specification.

Pseudocode

```
Initialize: □ ← all input routes
            □ ← ∅

For each route r ∈ □:
    Extract base_trip_id and numeric suffix from trip_id□□
    Let trip_counter ← starting numeric suffix

    For each direction index d from 1 to k:
        Let h ← lowercase(headsigns□[d])
        Let b ← block_ids□[d] (if exists, else "0")
        Let s ← shape_ids□[d] (if exists, else "0")

        For i ← 1 to num_trips□[d]:
            trip_id ← base_trip_id + trip_counter (zero-padded)
            □ ← □ ∪ {
                (trip_id, route_id□, service_id□, h, d, b, s)
            }
            trip_counter ← trip_counter + 1

Write □ to `trips.txt` as CSV
```

```
Given: trip_id□ = PREFIX + NUMERIC_SUFFIX
Extract:
    base ← PREFIX
    counter ← int(NUMERIC_SUFFIX)
    width ← length(NUMERIC_SUFFIX)

Then:
    trip_id ← base + str(counter).zfill(width)
```

Automating the creation of trips.txt and stop_times.txt significantly simplifies the process of building and managing a GTFS feed. This approach enables transit agencies to generate properly structured transit data without requiring advanced technical expertise. It also reduces manual effort, making GTFS adoption more practical for cities with limited time or resources. The complete GTFS feed was validated using the GTFS Validator by MobilityData (*GTFS Schedule Validator by MobilityData*, 2024), ensuring structural accuracy and full compliance with the GTFS specification.

Integration with Google Transit

After successful validation, the GTFS feed was uploaded to the Google Transit Data Feed portal. Once processed by Google, the Orange Line Metro routes and schedules became available on Google Maps (*Google Maps Transit Developer Guide*, 2024). Figure 2 displays the live Orange Line Metro schedule as seen on Google Maps, verifying successful integration.The deployment also demonstrated that our semi-automated approach reliably produces accurate and structured GTFS feeds, while allowing transit agencies to manually input static data where automation is unnecessary.

The Journal of Open Source Software

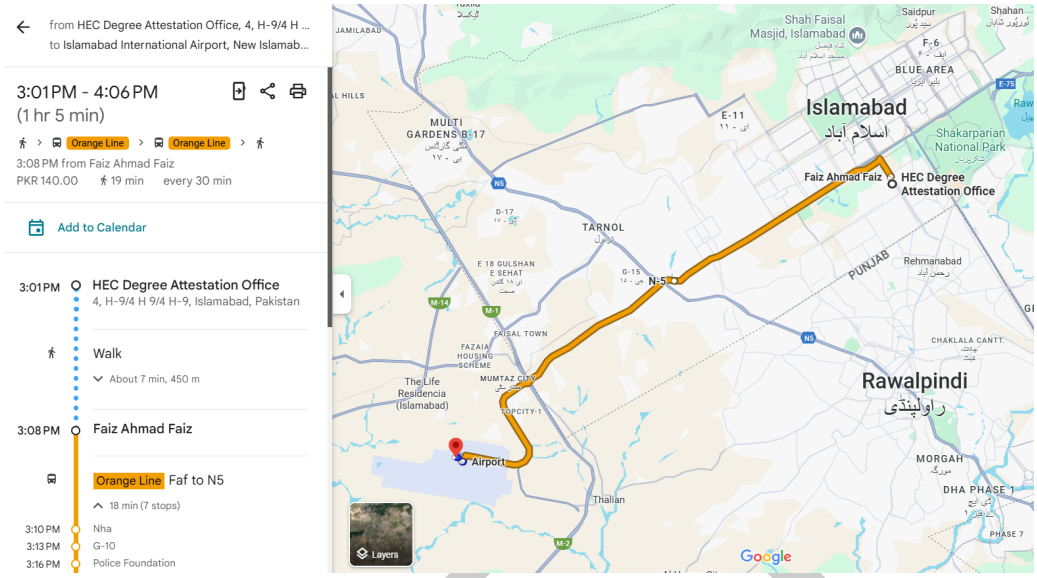

Figure 1: GTFS feed successfully integrated into Google Maps

Figure 2: GTFS feed successfully integrated into Google Maps, displaying Orange Line Metro transit data.

Conclusion

This work presents a practical, semi-automated approach for generating GTFS feeds, demonstrated through the case of Islamabad's Orange Line Metro Bus. By automating the creation of complex files such as trips.txt and stop_times.txt, the method minimizes manual effort and reduces the risk of errors. Designed for ease of use, it enables transit authorities with limited technical resources to efficiently digitize and publish transit data. The successful integration with Google Maps confirms the feed's accuracy and reliability. Overall, this work promotes wider GTFS adoption in developing regions and advances the goal of making structured transit information more accessible to the public.

Future Work

While our current work automates the generation of two key GTFS files (stop_times.txt and trips.txt), future enhancements will focus on automating additional components such as stops.txt, routes.txt, and calendar.txt. Another key direction is the integration of real-time transit data to enable live updates and vehicle tracking. The existing codebase can also be adapted for use in other cities, offering a scalable solution for diverse transit authorities. Looking ahead, we plan to develop simple, user-friendly interfaces to allow non-technical users to easily create and manage GTFS feeds. This will improve accessibility, promote community involvement, and support open data sharing and collaborative improvements.

Software Availability

The GTFS automation scripts and dataset for Islamabad's Orange Line Metro Bus are publicly available at GitHub repository code folder. This software is released under the MIT License and version 1.0.0.

Acknowledgments

This work was conducted under the project GCF-744: "Optimum Use of Existing Resources–A Prototype Model of Road Safety," funded by the Higher Education Commission (HEC) of Pakistan(gcf744 Roadsafety, Quaid-i-Azam University, 2023). We acknowledge the Transit Authority of Islamabad CDA (Capital Development Authority) for their collaboration and for providing access to transit data, which was instrumental in developing and validating the GTFS automation system.

Reference

gcf744 Roadsafety, Quaid-i-Azam University. (2023). *Project GCF-744: Optimum use of existing resources – a prototype model of road safety*. http://rs.qau.edu.pk.

Google maps transit developer guide. (2024). https://developers.google.com/transit.

GTFS reference documentation. (2023). https://gtfs.org/documentation/schedule/reference/.

GTFS schedule validator by MobilityData. (2024). https://gtfs-validator.mobilitydata.org.