CERTIFICATE

The thesis titled **'Routed – Dynamic Bus Scheduling System'** submitted by,

1.	**Rushabh Kapadia**	**141080014**
2.	**Dharmit Prajapati**	**141080038**
3.	**Kevin Daftary**	**141080052**

is found to be satisfactory and is approved for the Degree of Bachelor of Technology in Information Technology.

Mahesh Shirole
Examiner

Associate Professor,

C.E. and I.T. Department, VJTI.

Project Guide

Prof. H. K. Chavan
External Examiner

Date: 16 / 5 /2018

Place: V.J.T.I., Mumbai

CERTIFICATE

This is to certify that **Rushabh Kapadia (141080014)**, **Dharmit Prajapati (141080038), Kevin Daftary (141080052),** students of **Bachelor of Technology in Information Technology,** as a team have completed the report entitled, **" Routed Dynamic Bus Scheduling System "** to our satisfaction.

Prof. Mahesh Shirole
Assoc. Professor,
CE & IT Department, VJTI
Project Guide

Dr.V.B.Nikam
Head of CE And IT Department, VJTI

Dr. Dhiren Patel
Director, VJTI

CERTIFICATE

The report **" Routed Dynamic Bus Scheduling System "** submitted by **Rushabh Kapadia, Dharmit Prajapati and Kevin Daftary** is found to be satisfactory and is approved for the Degree of **Bachelor in Technology in Information Technology**.

Prof. Mahesh Shirole
Associate Professor,
CE & IT Department, VJTI
Project Guide

Prof. H. K. Chavan
External Examiner

Date: 16 / 05 / 2018

Place: VJTI, Mumbai

Acknowledgements

We would like to take this opportunity to express our sincere gratitude towards our mentor and project guide, Associate Professor Mahesh R. Shirole for his constant support and encouragement. We would like to thank him for devoting his time and effort to this project. His words of encouragement and wisdom have kept us on track throughout this project and he has helped us achieve our results.

We would like to express gratitude towards Vikhroli Bus Depot and the entire staff with a special mention for Mr. Shankar Jagtap for helping us understand the existing solution and its nuances and also for providing us with the required data.

We would like to thank the staff at Electric House, Colaba for providing us with the permission to obtain data for our project. A special mention to Mr. Noronha for guiding us with official works to be done for getting the required data.

We would also like to take this opportunity to thank our parents for bearing with us and taking care of us while we set about this journey. They have been the pillars of support that helped us work on this project with dedication and spirit.

Finally, we would like to express our gratitude towards our alma mater, Veermata Jijabai Technological Institute, for the extremely conducive environment it has provided and the infrastructure it has made available to students like us to pursue our projects.

Declaration of the Student

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources.

We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission.

We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature of the students

Rushabh Kapadia
(141080014)

Dharmit Prajapati
(141080038)

Kevin Daftary
(141080052)

Date: 16 / 05 / 2018

List of Figures

List of Tables

Abstract

With buses being a primary mode of transport for the common man, it is imperative that the bus transportation system works with optimal efficiency with respect to its resources. In Mumbai, there are 3600 buses running over 422 routes, managed by Brihanmumbai Electric Supply and Transport (BEST) and carrying millions of citizens every day. The fleet distribution is currently updated every four months manually by BEST officers. Currently, scheduling and allocating solution of BEST is time consuming, error prone and inconsistent. The current literature on this problem considers factors like traffic simulations, bus capacity and dynamic routes. The existing approaches vary from finding peak time of passenger frequency, to finding number of buses that should be allocated based on self-defined cost functions. The shortcomings of these approaches include the highly dynamic nature of traffic, consideration of infinite bus capacity and redefining routes completely. As the variation in the data over the short term weekly cycles is difficult to fit using linear and polynomial regression models, we find it unsuitable for our problem. Other regression models suffer from the lack of availability of attribute values for the next day to generate trip forecasts. Hence, we propose a solution to dynamically allocate buses to routes and generate their running schedules. We considered ARIMA and SARIMAX models that allow forecasting trips that are implemented using real-world data. Based on the forecasted data, fleet size allocation is done. The result and statistics obtained clearly depict the reduction of manual labour for Bus Scheduling and fleet size allocation. It distinctly shows better accountability of trends achieved due to automation.

Keywords—Bus Scheduling, ARIMA, SARIMAX, Resource Optimization

Contents

Chapter 1.

Introduction

Public modes of transportation in Mumbai include trains, buses and ferry services. With over 88% commuters preferring public transportation, there is an increased emphasis on optimally planning these transit systems. The greater connectivity via buses makes them the preferred choice of public transport by commuters for their daily travels. This raises the issue of using the fleet size of 3500 efficiently and scheduling them to make the efficient use of these resources.

Bus transportation is the primary mode of transport by road for most people. There are on an average 8.5 lakh people commuting by buses spread across 135 bus routes in Island city, Mumbai. While most people have to travel in jam-packed conditions to get their day started, there are empty buses running on the routes they take as well as on the routes not often travelled by commuters, which is a waste of resource that is needed on some different routes. Overcrowded and infrequent buses have become commonplace and as a result cause a lot of

discomfort to commuters. This is the result of inefficient scheduling and allocation of resources i.e., fleet size.

This is mainly because of the inefficient bus routines and high inter-arrival time between buses on the same route that is currently observed as a by-product of the static mundane bus schedules, because of which people cannot afford to miss their daily favoured bus that gets them to their workplace on time. 75% of the Island city is connected by high frequency buses and yet, the bus services suffer from overcrowding.

The growth in computer technology can come to the rescue by helping the service identify the number of buses and their schedules on the specific routes allotted to them. Such a proactive system would help ease the hassle-filled experience of daily commuters, which would help build a sense of trust and responsibility towards the public transport services among the masses. Moreover, by ensuring dynamic schedules tailored for the specific routes based on the frequency of people commuting along the route would ensure better utilization of the buses on the route as well as efficient fuel consumption and utility, ensuring better experiences for commuters as well as profitability for the service providers.

1.1 The Problem

Because a public transit agency is an unusual hybrid of private business and government, transit planning is a difficult profession. Should transit focus on providing transportation for those with no other choice, or should it strive to become a competitive alternative for the car? Unfortunately, it is difficult to simultaneously serve both alternatives. This difficulty is frequently aggravated by political interference in the transit planning process, which often forces transit agencies to operate inefficient bus routes and to construct sub-optimum rapid transit projects.

As a result, the existing BEST bus services use a static and inadequate system of routing buses on various routes which is quite inefficient from the economic point of view and overutilization of resources. The current Bus schedules are static and fixed which do not vary the frequency of buses along the routes according to the frequency of travellers at specific times of the day, which leads to overcrowding of buses. The current Bus schedules also have a fixed number of buses running on each route which leads to many buses running along routes without being utilized by passengers at all due to the time at which they are travelling along the routes and because the buses are being driven on routes that already have enough buses to cater the needs along the route.

A project to help the Bus service to better utilize the buses has been developed but it only considers routes in one direction as well as just helps to predict how many buses are needed along each route. Another shortcoming of this solution is that it considers unlimited number of buses being available at the disposal of BEST.

1.2 Existing Solution

The existing solution used by the public transport agency relies primarily on human schedulers who decide the routes travelled and the stops to be kept on the given route depending on the traffic analysis and by considering the demographics of regions such as commercial areas, residential areas, areas where traffic is high. The existing solution decides and finalizes the schedules by manually travelling along specific routes and finding out the frequency of people travelling along the given route at different times of the day. This approach is quite tedious since it requires a large human workforce to exhaustively cover all the routes and travel along these routes at all possible times of the day.

This solution also requires a very drawn out and lengthy period of research before the bus schedules are even created and finalized. As a result, this approach usually takes 2 to 3 years

of research before even considering whether the bus schedules need to be changed. This imposes an inherent inability on the existing system that ensures that the bus schedules cannot be dynamic and redesigned on monthly basis, let alone weekly basis.

However, the benefits of this approach are the fact that it ensures that there is no ambiguity or a factor of unpredictability of bus schedules since the data considered has been obtained through thorough observation. It also ensures that factors such as construction of new roads or construction of new skywalks, etc. along the existing routes are considered beforehand and can be accounted for in the designing of bus schedules and distribution of buses across different areas and routes.

1.3 The Proposal

The solution proposed here, mainly focuses on improving and better utilisation of Buses along various routes with dynamic schedules being generated to prevent overcrowding o buses during peak hours. The solution has mainly been divided into two modules for decoupling of two separate processes being carried out –

1. Trip Prediction: Predicting the optimum number of trips required along each route by analysing the dataset from the previous week using clustering algorithms and regression models for prediction.

2. Allocate Buses and Schedule Generation: Once we have the optimum number of trips required along each route, the next process is to allocate and schedule these buses optimally along the routes such that it ensures better overall experience for passengers and prevents overcrowding of buses during peak hours. This will also ensure that during peak hours, the amount of waiting time for passengers for a particular bus would be restricted to a minimal value.

This solution aims to create dynamic schedules for BEST bus services to ensure better utilization of resources (buses, fuel, etc.) and cutting down unnecessary losses due to incumbent static schedules currently being used.

The solution will take into consideration constraints like:

- Fixed number of buses to be optimally distributed across various routes
- Reusable buses i.e. same bus can travel along the given route multiple times in both directions
- Passenger centric schedules, i.e. considering the frequency of passengers at different time slots and accordingly varying the frequency of buses running along the route in that timeslot
- Only a single type of bus with an arbitrary capacity has been considered rather than many different types of buses that are available.
- The route is bi-directional i.e. the buses can travel on the same route in both directions.

The solution will not be considering following cases/instances like:

- Overlapping routes i.e. routes which have some of their part overlapping with each other
- Dynamic allocation of buses along the routes within a day based on timeslots i.e. once the buses have been distributed along the routes for the given day, they will not be re-allocated to some other route during that day
- Economic factors such as trying to reduce the cost of travel for passengers by running the same bus on a combination of two or more routes.
- It will only consider pre-existing routes from the start stop to the end stop i.e. the buses once scheduled will travel the whole route from start stop to end stop and not just a specific subset of the route.

1.4 Report Organization

This report is organized as:

Chapter 2 covers the literature survey conducted for bus (resource) allocation and bus scheduling along with data cleaning. In section 2.1, we present a brief summary of the survey on data cleaning. In section 2.2, we present the survey on resource allocation techniques drawing the comparison with buses as resources. In section 2.3, we put forth the survey on bus scheduling and timetable generation. We conclude with a summarizing table for all literature reviewed and the research gaps and hence the need for the proposed work.

Chapter 3 discusses the proposed solution and the system analysis for the proposed solution. In section 3.1, the system analysis is presented through architectural diagrams (subsection 3.1.1) and through use case analysis (subsection 3.1.2). Section 3.2 discusses the structure of the proposed system and expands on the components of the system. Subsection 3.2.1 covers the data collection process for the system. Subsection 3.2.2 talks about the pre-processing tasks done on the data collected to convert it into a suitable format. Subsection 3.2.3 describes the methods used to forecast number of trips to be made by the buses along particular routes. In subsection 3.2.4, we present the method used for bus allocation and generating a schedule for the day for every timeslot. Section 3.3 extends on section 3.2 and presents the experimental setup for the proposed methods.

Chapter 4 presents the results of the proposed models and goes on to discuss the meaning behind the output values obtained.

Chapter 5 summarizes the work done in the report (section 5.1) followed by a conclusion for the work done (section 5.2). Section 5.3 presents the possibilities of future enhancements of the proposed system.

Chapter 2.

Literature Survey

In our literature survey regarding the relevant domains and research papers to be focused on for the proposed solution, we mainly focused on research work in the fields of Cluster Algorithms, Bus Scheduling algorithms, optimal resource allocation, etc. Our prime focus during the literature survey has been to find out research papers from various sources that can be realized or cited in our solution.

The primary hurdles while performing the literature survey has been finding relevant research in alignment with our project objectives and that can be modelled or adapted to a real-time bus allocation and timetabling environment. The approaches considered within these research papers vary widely from statistical methods to data mining models and linear programming for Bus scheduling. Another type of research work considered has been identifying the trade-offs between different attributes/features that can affect the creation of optimal Bus Scheduling. Papers also considering the prediction of continuous values have been surveyed to identify predictive algorithms that can be adapted for the given problem of Bus schedule prediction.

With respect to Bus/resource allocation approaches, our primary aim has been to identify different types of research papers, such as papers proposing mathematical models for Bus Allocation, papers defining various angles to approach the problem of Bus Allocation and papers focusing on the identification of different attributes/features in the data that can be exploited for resource allocation have been surveyed in detail.

Also, as an additional reference point, some of the already existing bus scheduling and allocation models proposed previously have been studied to identify and concisely define the problem that we wish to target. A few Data-cleaning approaches and their implementation considerations have also been surveyed to support the data cleansing pre-processing that might be required.

The research papers considered are in no way the most exhaustive ones but they mainly cover all the domain areas that we wish to cover and implement in our solution. Also, another important factor is the fact that these papers have been drawn from a restricted pool of free research papers that are easily accessible on the internet, thus, excluding the paid research papers. All in all, the literature survey provides us with a basic foundation and an understanding of the problems that we might encounter during implementation and the difficulties associated with each approach.

Our main take away from this survey is to understand in depth the algorithms and approaches we might consider but in no way are bound to implement. The research papers have mainly been studied as being a guiding direction and not as something that is to be implemented and compared with the theoretical results. We have attached the summary of the research papers that we have surveyed below.

2.1 Data Cleaning

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. The raw data, usually, cannot be used directly as it contains a lot of noise. Data cleaning task primitive helps re-format and re-structure data so that it can then be utilized by the system that is going to process the data.

Xu Chu et al. listed three main questions that every data cleaning technique needs to answer namely: A) What to detect, wherein errors are classified into different types like functional dependencies, integrity constraints, and so on. B) How to detect, where the approaches are categorised based on human involvement and C) Where to detect, which elaborated the possible locations of error in the entire system. It also focused on how to repair the errors encountered.

2.2 Bus Allocation

The idea of optimal resource allocation to minimize costs is the driving force behind considering bus allocation. Bus allocation refers to identifying the total number of buses that need to be run on a route every day in both directions, up (source-to-destination) and down (destination-to-source). Too many buses (resources) would greatly increase the cost of the system and too few would result in the system underperforming.

Eshetie Berhan et al. [4] dealt with bus assignment per route for four different shifts of work in a day: two peak and two off-peak shifts. It considers features such as route performances, number of passenger served, total trips made, revenue collected, operating cost and total distance covered. In their approach, a Linear programming model was considered with

parameters like heterogeneous fleet size, demand proportion to predict number of trips made for each bus type. The problem with their model was that there were fixed number of buses scheduled per route which resulted in some buses running empty while others are overcrowded.

Carbno Collin [6] came up with mathematical models that weigh numerous factors on delivery schedules, in determining the allocation of resources to different projects. The paper helps understand optimal resourcing and Manfred's distribution and design intelligible linear models given the multitude of factors affecting such decisions.

2.3 Bus Scheduling

In a public transit system like BEST, it is imperative that the timetable for buses be such that there is no significant wait time for passengers. A timetable that forces passengers to wait for long durations can never service the passengers/customers efficiently. Also, the timetable formulated should be in keeping the peak operation times.

Makrand Wagale et al. [3] developed a demand and travel time responsive (DTR) model to find optimal bus frequency and to optimize bus schedules. It considers social costs like passenger waiting and riding times and simultaneously considers bus operation costs, bus headways and so on at every bus stop. The paper used evenly spaced time intervals while formulating the timetable.

Avishai Ceder [5] proposed three methods for timetable development by trying to minimize the passenger waiting time. Even-Headway timetables (scheduling buses after fixed intervals based on bus capacity and passenger load), Even-Load timetables (scheduling buses at varying intervals based on passenger load variation) and the combination of the two are the three methods suggested. It formulates vehicle scheduling as a cost-flow network problem and uses step function termed deficit function, as it represents the deficit number of vehicles required at

a particular terminal in a multi-terminal transit. It uses deadheading (empty trips) to ensure achieving multiple trips from different terminal by a single vehicle.

Anirudha Nanda et al. [7] proposes a heuristic model for the timetabling problem from the teachers' perspective that uses three different data structures, namely, Output, Clash and Day_Clash. This paper aided us in understanding the problem of timetable scheduling with resource availability constraints. The only drawback is that it assumed no repetition of subjects while generating the time table. By drawing a parallel between buses and teachers, we can make use of the idea presented in this paper. The resource in their proposition is the teacher's time while for our model it is the buses that are to be allocated to routes.

Liping Fu et al. [8] proposed a dynamic scheduling model with a skipping control for every alternate bus. The normal buses halt at all stops while the express buses skip over some stops. The model includes passenger waiting times, even in the event of a bus skipping their stop, headways and total in vehicle travel times and total bus trip time. To decide the stops to be skipped, four different methods have been proposed – zone scheduling wherein routes are divided into zones and accordingly stops out of zone are skipped, short turning where the express buses do not run full length trips thereby reducing required buses, dead heading which allow buses to run empty through some stops to reduce headways later, dynamic stop skipping allows vehicles behind schedule to skip low demand stops.

Output:

The result of this literature survey is to help us get a direction in our efforts by allowing us to learn about different algorithms and approaches and their application to problems that are similar to ours in some respect.

Lacuna:

None of the literature surveyed considered only ticket records as their data set. Most of them have different self-defined attributes that are used for clustering, prediction and scheduling buses. Also, the relative order of importance of those attributes isn't defined explicitly. The papers do not consider the problem of allocating appropriate number of buses and scheduling them on a route as a combination, rather they attempt at resolving either one of them at a time. The literature did not cover the idea of using timestamps to identify timeslots when buses are supposed to travel on their assigned routes.

Need of New System:

At present, there is no system in place that consider just the ticket records i.e., data from previous weeks to dynamically alter bus travel routes and timings. The current implementations focus on growing the service by changing routes completely or modifying the types of buses for better service instead of focusing on using the existing resources in an efficient manner.

Chapter 3.

Proposed Solution and Analysis

In this section, we mainly analyse the problem statement, raw data and goals to generate the system requirements and components required for predicting optimal bus schedules and allocation. Here, our main consideration is to ensure that we generate results that perform at the same level if not better than the current manual methods used for bus scheduling and allocation. The proposed solution also needs to consider whether there are certain optimizations that can be used for the manual counterparts currently in use. Some of the assumptions considered for the solution are related to the position of the solution in the current pipeline of service used by BEST for efficient planning and operations. Another assumption consider is the granularity of approaching the problem. We consider generating new schedules daily according to the passenger load and efficiency of current schedules as a solution that qualifies as dynamic. We use the conventional software development life cycle approach for the

development of the solution with a recursive approach to obtain feedback and optimize/modify the solution as per the comments.

3.1 SYSTEM ANALYSIS AND DESIGN

System analysis and design mainly refers to "the process of studying a procedure or business in order to identify its goals and purposes, to create systems and procedures that will achieve them in an efficient way"

After identifying the goals and various requirements for the system, we now perform a thorough analysis of these requirements to generate a high level model of the solution. For the proposed solution, we first identify different use cases and components of the system. The primary focus here is to identify and analyse the different use-cases in the system, in order to ensure that the solution meets the demands and constraints of the real – world. We then provide a detailed overview of the solution from the perspective of system design by analysing and formulating the different flows within the system.

3.1.1 ARCHITECTURAL DIAGRAMS

In simplest terms, the system, named "Routed" (Refer Fig 3.1) is a bus scheduler to optimize the transport system currently in place. With the current scenario of crowded buses and the delay and discomfort caused by the crowd as well as the traffic, this system hopes to alleviate the aforementioned problems.

The schedule manager has to input the data records for buses i.e., the tickets sold for the previous weeks into the "Routed" system which mainly refers to the number of passengers travelling on a specific route in the previous weeks, and has to simply wait to receive the output - an optimized bus schedule for the next day and the number of buses allocated per route – as

the system goes about working on the input. The system essentially groups the data according to various grouping criteria and then uses the input data to forecast trips for each route so as to come up with an "intelligent" solution i.e., the optimized schedule which is given to the schedule manager and a count of the number of buses along each route given to the bus regulator who can then decide which bus should make trips on which route.

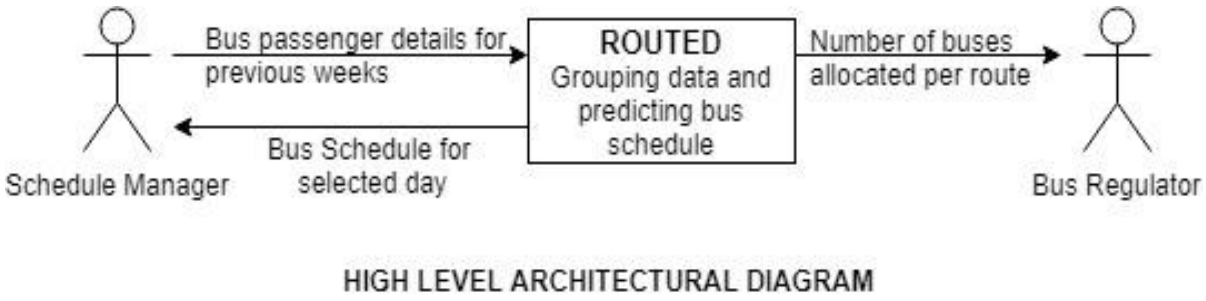

Figure 3.1 – Routed high level Architectural Diagram

The "Routed" System is aimed at optimizing bus schedules on specific routes so as to alleviate traffic problems and the delay caused because of the same. The entire system can be broken down into multiple modules with each module responsible for certain tasks. The system is divided into the following modules (Refer Fig 3.2):

1) Data Pre-processing:

Input: Bus passenger records from previous weeks (Fed by Schedule Manager)

Output: Cleaned and validated dataset

Function: To check the input data records for attributes, data range and format of data.

2) Forecasting Trips:

Input: Data records on each route with attribute list

Output: Hourly trip forecasts for each route (Sent to Bus Regulator)

Function: To determine the number of trips required on a route based on the frequency of passengers on that route, day details and date details. This information can be used by the Bus Regulator to decide which buses would run on which route.

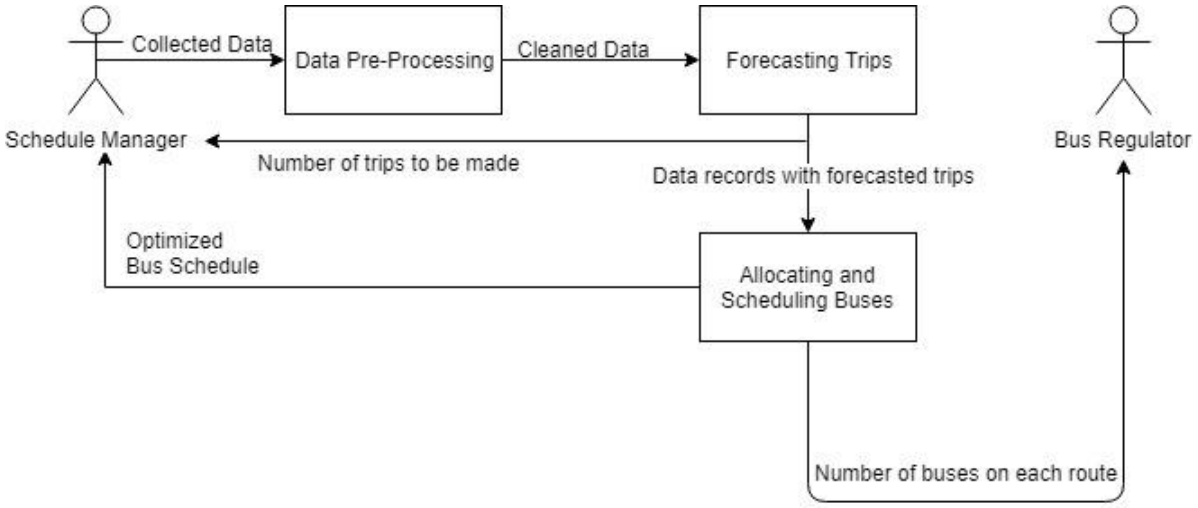

Figure 3.2 – Routed Low Level Architectural Diagram

3) Allocating and Scheduling Buses:

Input: Trip forecasts from Forecasting Trips module for each route

Output: Optimized Bus Schedule for each route and bus allocation for each route (Sent to Schedule Manager)

Function: To prepare an optimized bus schedule based on the forecasted trips obtained from module 2 and optimally allocate buses according to the schedule.

3.1.2. USE CASE ANALYSIS:

Use case analysis is a technique used to identify the requirements of a system (normally associated with software/process design) and the information used to both define processes used and classes (which are a collection of actors and processes) which will be used both in the use case diagram and the overall use case in the development or redesign of a software system or program. The use case analysis is the foundation upon which the system will be built. The Use case diagram for the whole system is as follows:

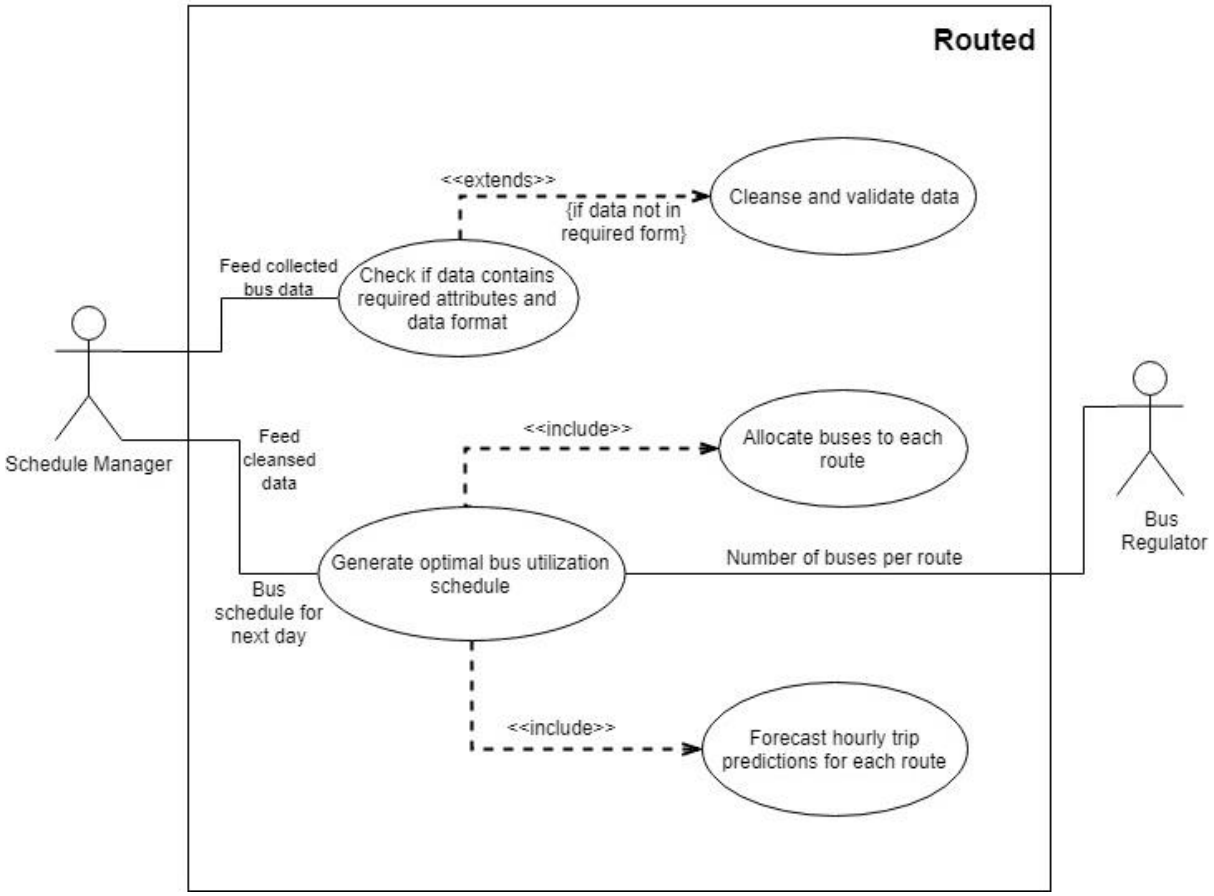

Figure 3.3 - Use Case Diagram for Routed System

3.1.2.1 Use Case Specification and Realisation

Use case 1: Cleanse and Validate data

• Goal in context:

The goal of this use case is to validate and ensure data entered into the system is cleansed and validated according to the requirements of the system if the data can be converted into valid form by cleansing.

• Primary Actors:

Routed System (Primary)

• Pre-conditions:

1. Routed System should be configured with right attribute name list and data formats.
2. Check should be performed to decide whether data is in required form or not. (Use case: Check if data contains required attributes and data format)

• Post-conditions:

1. System converts Data into valid format for further use.

• Basic Flow of Events:

1. Data is forwarded to the module from Use case: Check if data contains required attributes and data format.
2. Data is validated against the attribute name list and data formats.
3. The data is checked for all attributes to see whether it has values within the permissible range.
4. If Data passes step 2 and step 3, the data is sent back to Use case: Check if data contains required attributes and data format.

• Alternative Flows:

2a. Data is not validated.

2a1. Display error message and notify about the data formats violated and attributes missing.

3a. Data values are not within permissible range.

3a1. System normalizes data to fit within permissible range or displays error message if data format does not match with allowed formats.

4a. Data does not pass step 2 and/or step 3

4a1. Exit the system.

The Fig 3.4 shows the sequence diagram corresponding to the Use Case 1.

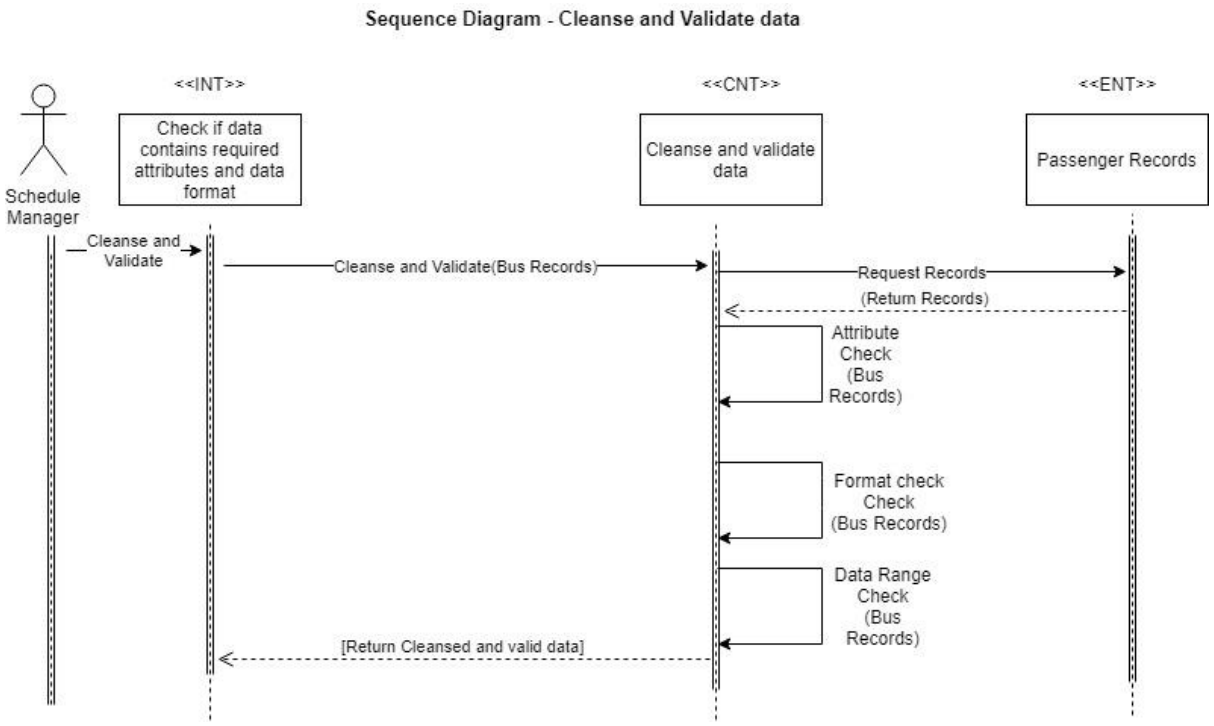

Figure 3.4 - Sequence diagram for Cleanse and Validate data

Use case 2: Check if data contains required attributes and data format.

• Goal in context:

The goal of this use case is to perform data validation and ensure data is converted in the format required by the system if possible.

• Primary Actors:

Schedule Manager (Primary), Routed System (Secondary)

•Pre-conditions:

1. Routed System should be fed the data by the Schedule Manager.

2. System should be configured with the required metadata and policies.

• Post-conditions:

1. The data returned by the system should be in the form required by the system for further processing (i.e. in useful form)

• Basic Flow of Events:

1. Schedule Manager enters the Bus usage statistics data into the system as a file input.

2. System checks the data whether it passes all the data policies and formats required by the system for further processing.

3. If data passes step 2, output data to Schedule Manager as a file.

• Alternative Flows:

2a. Data does not pass the data policies and formats required by the system.

2a1. Perform Use Case: Cleanse and validate data.

2a2.If Use Case: Cleanse and validate data returns data, resume from step 3.

2a3. If Use Case: Cleanse and validate data does not return data, notify Schedule manager to submit appropriate data.

3a. If data does not pass the required formats.

3a1. Notify Schedule Manager about the error and exit the system.

The Fig 3.5 shows the sequence diagram corresponding to the Use Case 2.

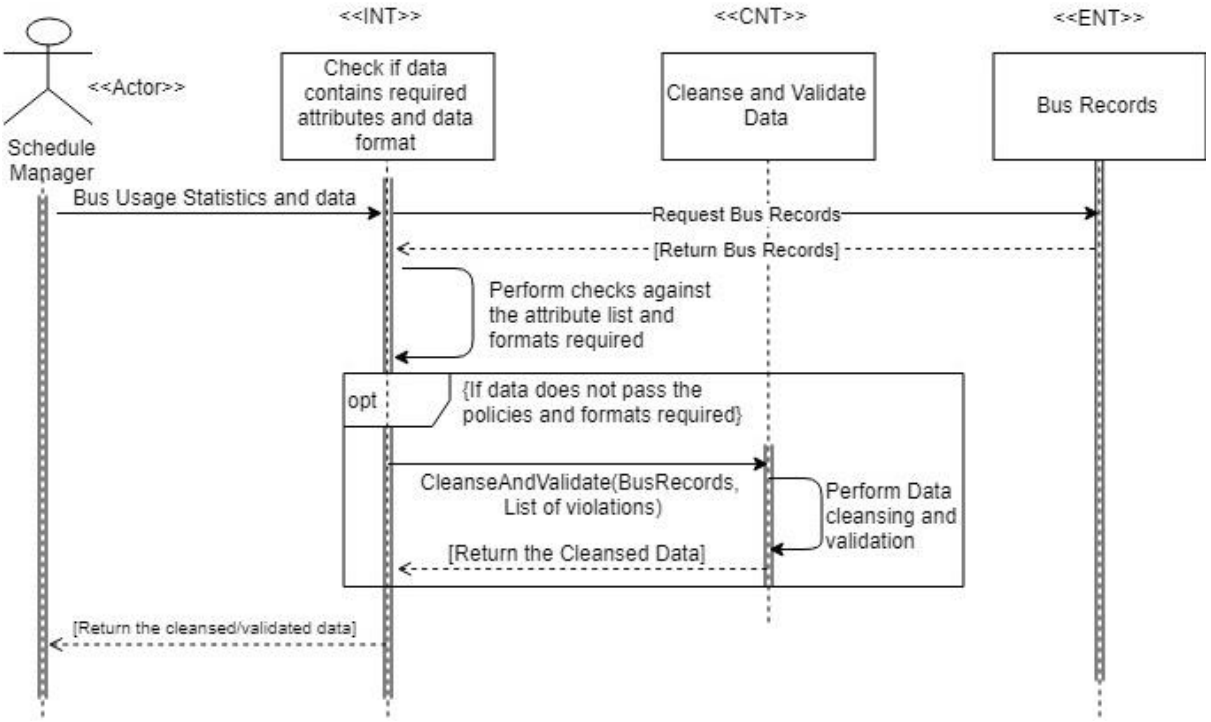

Figure 3.5 - Sequence diagram for Check if data contains required attributes and data format.

Use case 3: Allocate buses to each route

• Goal in context:

The goal of this use case is to use a favourable predictive/fleet size allocation algorithm to allocate optimal number of buses to each route.

• Primary Actors:

Routed System (Primary)

• Pre-conditions:

1. Routed System should be fed data already cleansed and clustered into groups. (Use case: Check if data contains required attributes and data format)

• Post-conditions:

1. System generates optimal Bus allocations for each route.

• Basic Flow of Events:

1. Data is forwarded to the module from Use case: Generate optimal bus utilization schedule.
2. Apply the bus allocation algorithm by using the forwarded data as input.
3. Send the bus allocations for each route to the Use case: Generate optimal bus utilization schedule.

• Alternative Flows:

2a. Bus allocation algorithm encounters an error.

2a1. Retry the bus allocation algorithm 3 times.

2a2. Display error message and exit the system if retries fail.

3a. Data could not be sent back to the Use case: Generate optimal bus utilization schedule.

3a1. Retry to send data 3 times.

3a2. Display error message and exit the system if retries fail.

The Fig 3.6 shows the activity diagram with the basic flow as well as the alternate flow for Use Case 3 – Allocate buses to each route. Refer Use Case 3 for specification.

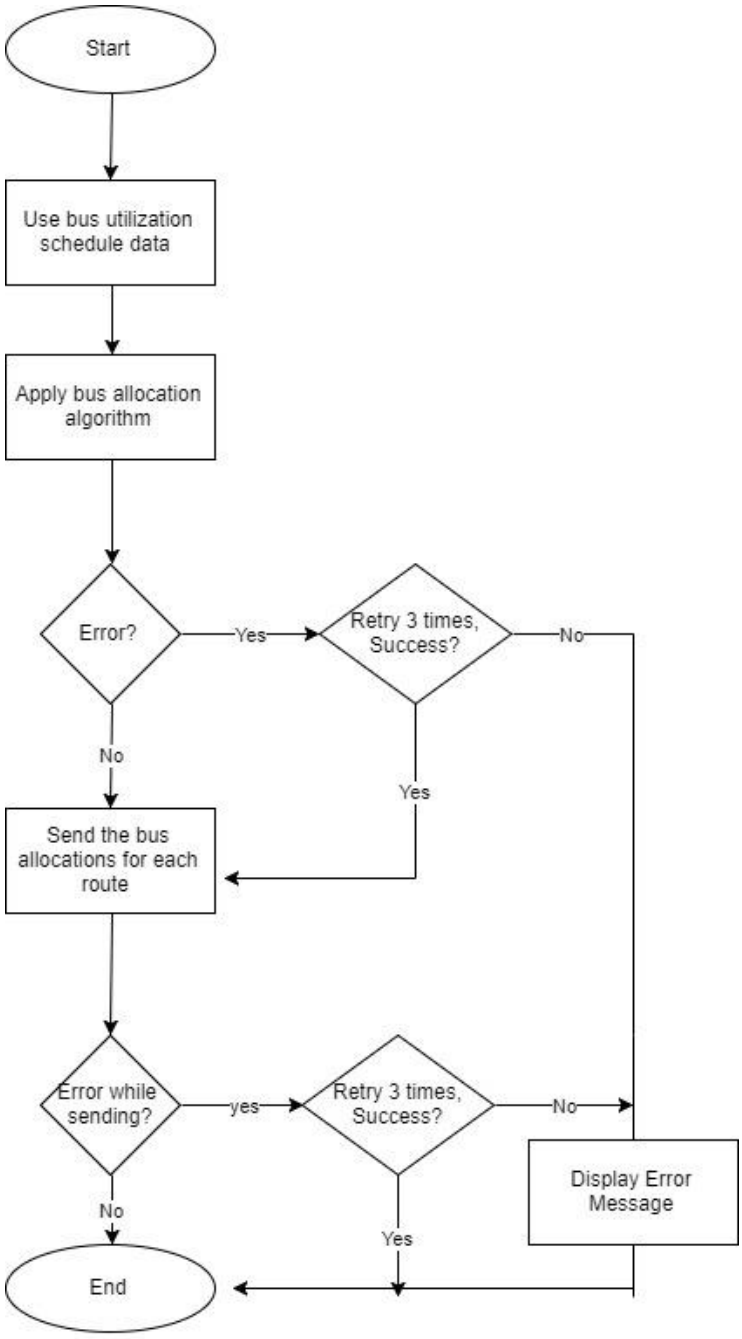

Figure 3.6 - Activity diagram for Allocate buses to each route

The Fig 3.7 shows the sequence diagram corresponding to the activity diagram (Fig 3.6) and Use Case 3.

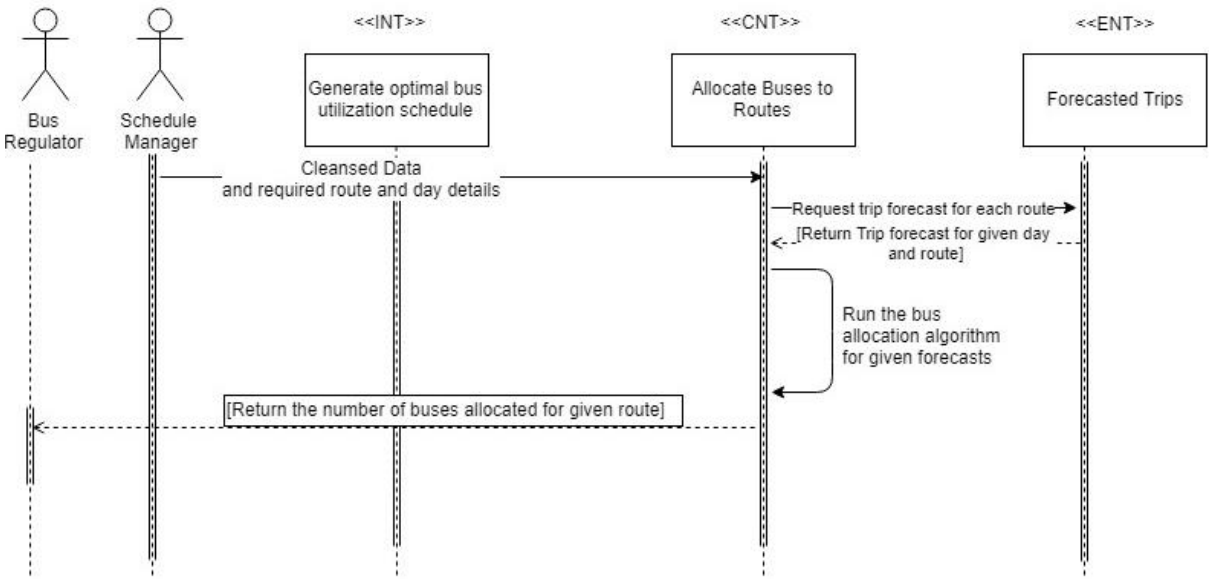

Figure 3.7 - Sequence diagram for Allocate buses to each route.

Use case 4 - Forecast hourly trip predictions for each route

• Goal in context:

The goal of this use case is to use a favourable predictive algorithm to generate bus schedules for each route.

• Primary Actors:

Routed System (Primary)

• Pre-conditions:

1. Routed System should be fed data already cleansed and clustered into groups. (Use case: Check if data contains required attributes and data format)
2. Routed System should have information about the number of buses allocated per route. (Use case: Allocate buses to routes)

• Post-conditions:

1. System generates optimal Bus schedules for each route.

• Basic Flow of Events:

1. Cleansed and clustered data file and the bus allocation information is forwarded to the module from Use case: Generate optimal bus utilization schedule.
2. Apply the predictive algorithm to generate schedule for each route by using the forwarded data as input.
3. Send the bus schedules for each route generated in step 2 to the Use case: Generate optimal bus utilization schedule.

• Alternative Flows:

2a. Predictive algorithm encounters an error.

2a1. Retry the algorithm 3 times.

2a2. Display error message and exit the system if retries fail.

3a. Data could not be sent back to the Use case: Generate optimal bus utilization schedule.

3a1. Retry to send data 3 times.

3a2. Display error message and exit the system if retries fail.

The Fig 3.8 shows the sequence diagram corresponding to the Use Case 4.

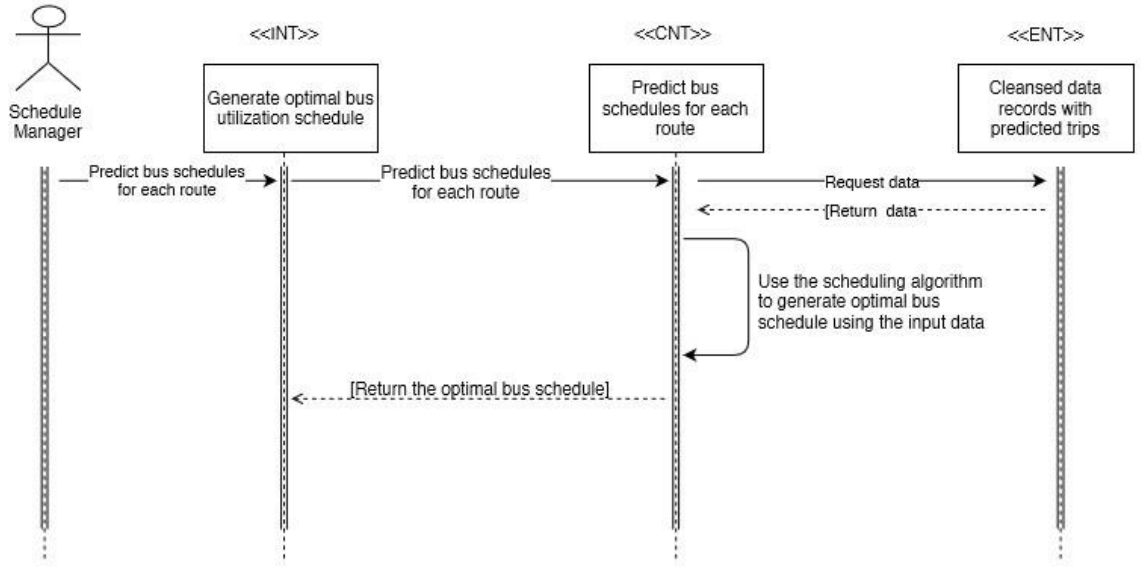

Figure 3.8 - Sequence diagram for Forecast hourly trip predictions for each route.

Use case 5: Generate optimal bus utilization schedule

• Goal in context:

The goal of this use case is act as an interface which co-ordinates between general allocation and predictive algorithms, and generate the bus utilization and scheduling tables.

•Primary Actors:

Schedule Manager (Primary), Bus Regulator (Primary), Routed System (Secondary)

• Pre-conditions:

1. Routed System should be fed data already cleansed and clustered into groups. (Use case: Check if data contains required attributes and data format)

• Post-conditions:

1. System generates bus utilization and scheduling tables for each route.

• Basic Flow of Events:

1. Cleansed and clustered data file is fed to the module by the Schedule Manager in the form of a file.

2. To generate bus allocation per route, do Use case: Allocate buses to routes and send the data file as input.

3. If Use case: Allocate buses to routes returns data, then output the bus allocation tables ie. Number of buses per route to the Bus regulator as a file and go to step 4.

4. To generate bus schedules for each route, do Use case: Predict bus schedules for each route and send the clustered data along with the bus allocation information obtained in step 2 as input.

5. If step 4 returns data, then send the bus schedules for each route generated in step 4 to the Schedule Manager as a file.

• Alternative Flows:

3a.Use case: Allocate buses to routes does not return data.

3a1. Retry Use case: Allocate buses to routes 3 times.

3a2. if step 3a1 returns data, go to step 4 else go to step 3a3.

3a3. Notify Schedule Manager and Bus regulator about the error and exit the system.

5a.Use case: Predict bus schedules for each route does not return data.

5a1. Retry Use case: Predict bus schedules for each route 3 times.

5a2.if step 5a1 returns data, go to step 5 else go to step 5a3.

5a3. Notify Schedule Manager about the error and exit the system.

The Fig 3.9 shows the sequence diagram corresponding to the Use Case 5.

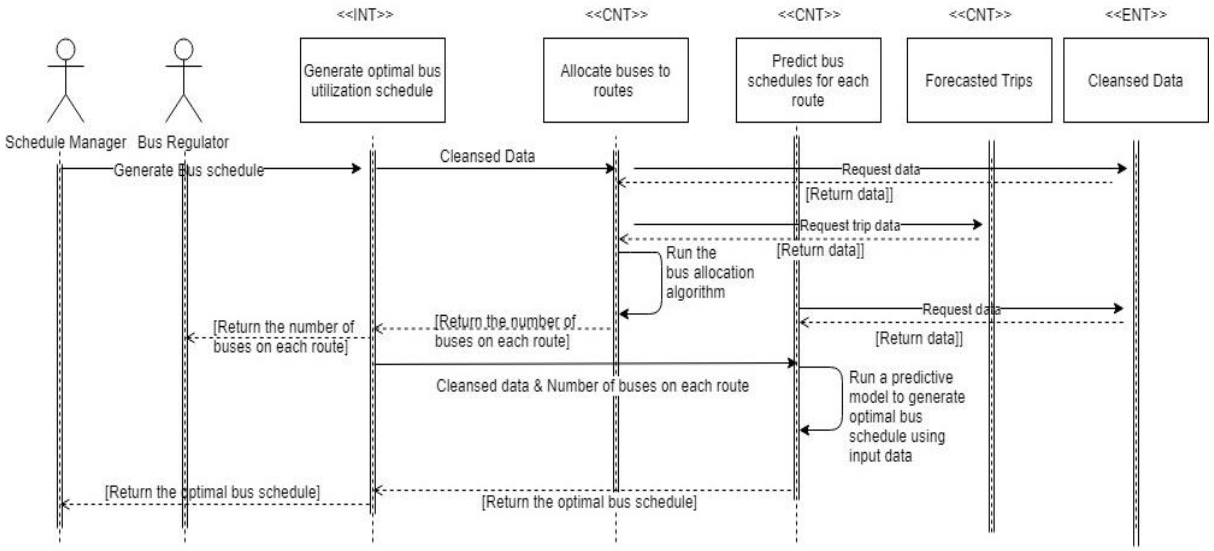

Figure 3.9 - Sequence diagram for Generate optimal bus utilization schedule

3.2. SYSTEM STRUCTURE AND ARCHITECTURE

Based on the system analysis and requirements of the problem statement, we thus formulate a system divided into three independent services which function isolated from each other. These three services are based on the three core tasks to be performed for generation of optimal bus schedules and allocation. We further sub-divide the data preparation service into data collection which identifies the required attributes in the raw data and data pre-processing which identifies missing values, replaces NULL values and converting attributes within correct format and range. The other two core services are forecasting trips on which the second service, namely bus scheduling and allocation depends on for optimal bus utilization.

The proposed architecture of the system based on the system analysis and design is a single threaded application with serial execution of each service. The system can be broken down into four different sub-sections:

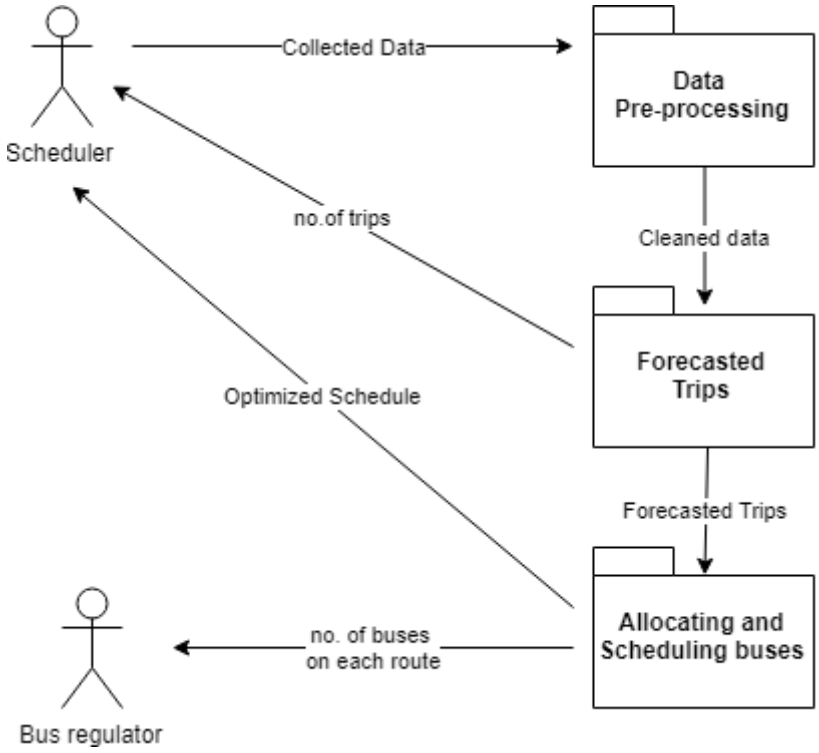

Figure 3.10 – Package Diagram for Routed System

3.2.1 Data collection, which describes input data.

3.2.2 Data pre-processing describes various methods applied to data to make it suitable for further processing.

3.2.3 Forecasting trips, outlines the different methods used to forecast trips

3.2.4 Allocating and scheduling buses, describes the algorithm used to allocate and schedule the buses.

Other miscellaneous/supplementary services are error handling and robust system which can handle peak loads of CPU-intensive processing required for recurrent networks and statistical models.

3.2.1. Data Collection

The input data for our project has been obtained from Vikhroli Bust Depot, Mumbai. The collected data contains information about 16 different routes in up-direction (source-to-destination) and down-direction (destination-to-source). The summary of the data fields collected from the bus depot isshown in Table 1. The data for number of passengers ranged from as low as 50 to as high as 600 while the number of trips were in the range 0 to 10.
To consider the variations in data that arises due to public holidays, we consider a list of all holidays to alter the behaviour of the system accordingly. Holiday name, Date, Day and Month are the attributes in the file.

TABLE 3.1 - SUMMARY OF THE DATA FIELDS COLLECTED FROM THE BUS DEPOT

Input Data Columns	**Description**
Route	Contains the route number
Date	Contains the date on which the data was collected
Day	Contains the day on which data was collected
Time Slab	Timeslots in which the 24 hours of the day are divided. The format is HH:MM:SS
Up Trips	Number and percentage of trips made during the day distributed according to the timeslots for up direction
Up Passenger	Number and percentage of passengers travelling during the day distributed according to the timeslots for up direction
Up Revenue	Total and percentage revenue generated during the day distributed according to the timeslots for up direction
Up Run KM	Total and percentage kilometres covered during the day distributed according to the timeslots for up direction
Down Trips	Number and percentage of trips made during the day distributed according to the timeslots for down direction

Down Passenger	Number and percentage of passengers travelling during the day distributed according to the timeslots for down direction
Down Revenue	Total and percentage revenue generated during the day distributed according to the timeslots for down direction
Down Run KM	Total and percentage kilometres covered during the day distributed according to the timeslots for down direction
Total Trips	Number and percentage of trips made during the day distributed according to the timeslots for both directions combined
Total Passenger	Number and percentage of passengers travelling during the day distributed according to the timeslots for both directions
Total Revenue	Total and percentage revenue generated during the day distributed according to the timeslots for both directions
Total Run KM	Total and percentage kilometres covered during the day distributed according to the timeslots for both directions

3.2.2. Data Pre-Processing

Real-world data is often incomplete, inconsistent, lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. Hence, it is an essential process to ensure that the data is compatible with the proposed method. To predict trips in up-direction and down-direction, data is split into two separate files. Converting the .xls files to .csv files which is more favourable and widely used in python environment. Missing or empty values in the data are replaced with the constant zero. Days of the week are enumerated for ease of use with Monday being numbered 1 and Sunday being numbered 7.

Data pre-processing is mainly required in our solution to ensure that the trip forecast as well as bus scheduling and allocation algorithms do not deal with incomplete data or inconsistent data. The bus scheduling and allocation algorithm is not robust enough to handle NULL values as trip predictions. Similarly, trip forecasting algorithm is highly sensitive to incorrect format and

error values since it may lead to incorrect predictions. Thus, ensuring data is clean and workable is an important task.

3.2.3 Methods to Forecast Trips

Based on the variations and trends in the data, we mainly select three different methods based in the domains of Statistics and Neural networks. These methods are mainly chosen due to their ability to inherently identify trends as well as assign different weights to different data records based on their correlation factor such as PCF and ACF. Other than this, we mainly choose a recurrent neural network method that can remember short-term as well as long-term trends in the data which is quite apparent in the passenger data containing seasonal as well as weekly cycles of variations.

The other methods that were considered for our solution are based in the fields of machine learning and regression. These methods mainly had limitations related to the representation of cyclic trends and poor accountability of the contribution of each record to the generated model, that is, it was much more suitable for cumulative results but generated poor results for next day forecasts which represent higher dependence on previous week values as compared to previous month values.

Thus, the three representative models which have been considered and compared inour solutions are:

3.2.3.1. ARIMAX Model

3.2.3.2. SARIMAX Model

3.2.3.3 RNN using LSTM units

3.2.3.1. ARIMAX model for trip prediction:

The ARIMA (Auto-regressive integrated moving average) model [9, 10] is a time-series analysis model. It considers future values of a variable (time series) to be dependent on its previous values. When an ARIMA model includes other time series (other attributes on our case) as input variables, the model is referred to as an ARIMAX model. We use this method due to its inherent ability to handle noise and identify trends as a function of time. A non-seasonal ARIMA model is classified as an "ARIMA (p,d,q)" model, where

- p is the number of autoregressive terms,
- d is the number of non-seasonal differences needed for stationarity, and
- q is the number of lagged forecast errors in the prediction equation.

The forecasting equation is constructed as follows. First, let y denote the d^{th} difference of Y (the time series under consideration), which means:

- If d=0: $y_t = Y_t$
- If d=1: $y_t = Y_t - Y_{t-1}$
- If d=2: $y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2}$

In terms of y, the general ARIMAX forecasting equation is as shown in Equation A below [11]:

$$\bar{Y}_t = \mu + \beta x_t + \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} - \theta_1 e_{t-1} - \ldots - \theta_q e_{t-q} \quad (1)$$

Where,

$\bar{Y}_t$ is the forecasted value of series Y at time t

μ is the mean of the series

X_t is a covariate at time t

β is the co-efficient of xt

Φ_i is the ith auto-regressive (AR) co-efficient of the series

θ_i is the jth moving average (MA) co-efficient of the series (their signs are negative in the equation, following the convention introduced by Box and Jenkins)

e_t is normally distributed error term with a mean of 0 at time t

Another popular equation of the ARIMAX model is given as shown below: [12]

$$(1-\sum_{i=1}^{p}\Phi_i L^i)(1-L)^d(X_t - m_t) = (1+\sum_{i=1}^{q}\theta_i L^i)\varepsilon_t \quad (2)$$

$$L^k X_t = X_{t-k} \quad (3)$$

$$m_t = c + \sum_{i=0}^{b}\eta_i y_{t,i} \quad (4)$$

In general, the data is analysed by evaluating weekly (or monthly/yearly) trends over time using the Box-Jenkins procedure [13] applied to each input variable in order to control for autocorrelation in the corresponding time series. We then fit an autoregressive integrated moving average model with exogenous covariates (ARIMAX) to each data time series, Xt, where p, d, and q, are the respective autoregressive, differencing, and moving average orders of the model (equation (B)). The ϕi and θi are the autoregressive and moving average parameters, respectively, εt is a normally distributed error term with a mean of 0, L is a lag operator defined as in equation (C), and mt is defined as in equation (D), where yt is a series of predictors, the ηi are a series of predictor weights, and b is the total number of predictor time series.

We do not consider the moving average terms (q=0) in our solution since the data used in our solution is captured at real-time without using any approximations or forecasts. Thus, having a factor that

relates the current period value to the random error in previous periods is of little significance. Also, we use the first order difference of cleaned data (instead of the original data) to make the data stationary.

3.2.3.2. SARIMAX model for trip prediction:

SARIMAX (Seasonal ARIMAX) is a time series analysis model that takes seasonality into consideration along with ARIMAX [14, 15]. In a seasonal ARIMA model, seasonal AR and MA terms predict xt using data values and errors at times with lags that are multiples of S (the span of the seasonality).The additional seasonal terms are simply multiplied with the non-seasonal terms. In practice, we examine the differenced data when we have seasonality. Seasonality usually causes the series to be non-stationary because the average values at some particular times within the seasonal span (months, for example) may be different than the average values at other times. Since the seasonal pattern is both strong and stable over time, we use a seasonal difference regardless of whether a non-seasonal difference is used, since this will prevent the seasonal pattern from "dying out" in the long-term forecasts.

We use the SARIMAX model since there is a seasonality trend in our data (the trip prediction values are almost same for the same day of the week). Since we use daily data for our time series (different time slots within a day and different days within a week) and there is too much variation in the data to determine the trends, we look at the rolling mean, rolling standard deviation, seasonality (seasonal difference and different order of difference of these metrics), and residuals before tuning the model. The residual values essentially take out the trend and seasonality of the data, making the values independent of time.

The seasonal ARIMA model incorporates both non-seasonal and seasonal factors in a multiplicative model. One shorthand notation for the model is [16]

$$\text{ARIMA}\ (p, d, q) \times (P, D, Q)\ S,$$

With,

p = non-seasonal AR order, d = non-seasonal differencing, q = non-seasonal MA order, P = seasonal AR order, D = seasonal differencing, Q = seasonal MA order and S = time span of repeating seasonal pattern.

The mathematical formula for a SARIMAX model is [17]

$$(1 - \sum_{i=1}^{p} \varphi iBi)(1 - \sum_{i=1}^{P} \Phi iBi(S))(1-B^{d})(1-B^{S})Y_t + \beta X_t$$
$$= (1 + \sum_{i=1}^{p} \Theta iBi)(1+\sum_{i=1}^{Q} \theta iBi(S))e_t$$

Where,

B is the Lag operator ($B_1Y_t = Y_{t-1}$)

$(1 - \sum_{i=1}^{p} \varphi iBi)$ is the Non-seasonal Auto-regressive (AR) term of order p

$(1 - \sum_{i=1}^{P} \Phi iBi(S))$ is the Seasonal Auto-regressive (AR) term of order P and the seasonality cycle of length S

$(1-B^{d})$ is the Non-seasonal difference (I) term of order d

$(1-B^{S})$ is the seasonal differencing (I) term of order S

Y_t is the value to be forecasted at time t

$(1 + \sum_{i=1}^{p} \Theta iBi)$ is the Non-seasonal Moving average (MA) term of order q

$(1+\sum_{i=1}^{Q} \theta iBi(S))$ is the Seasonal Moving average (MA) term of order Q and the seasonality cycle of length S

e_t is the random error term with zero mean

X_t is the covariate at time t and β is its co-efficient

In our model, we do not consider the non-seasonal moving average (MA) and seasonal auto-regressive (AR) terms since the autocorrelation values and the ACF plots show little

significance of these factors in the forecasting and formulation of the model. We use a weekly seasonality cycle (S=7) since we are dealing with weekly trends. This model has the inherent capability to factor in the seasonality and inherent randomness of values that are observed (and captured) in real-time in the cleaned data.

3.2.3.3. Long Short Term Memory (LSTM) Recurrent Neural Networks for trip prediction:

LSTMs [18] act as the basic building blocks in the layers of RNNs (a single neuron is an LSTM neuron). As the name suggests, the LSTM units can store information and trends in the memory associated with it and thus, can remember inputs, associations, etc. over a long period of time. This memory is generally in the form of gated cells composed of sigmoid logic units and/or point wise multiplication operation to save and alter cell state to learn long term dependencies. The cell decides whether or not to store or delete information based on the importance it assigns to the information. The assigning of importance happens through weights, which are also learned by the algorithm. To minimize LSTM's total error over a training set, iterative gradient descent such as back-propagation through time is preferred to change each weight in proportion to its derivative with respect to the error. This simply means that it learns over time which information is significant for future predictions. The fact that they are analog enables LSTMs to perform back-propagation. It is an extension to the typical RNNs (which suffer from vanishing and exploding gradients) and it solves the problem of vanishing gradients by self-adjusting the gradients to be steep enough. This makes the training process complex to understand but relatively shorter with higher accuracy [19].

The data flow through an LSTM unit [20] is shown in Fig 3.11.

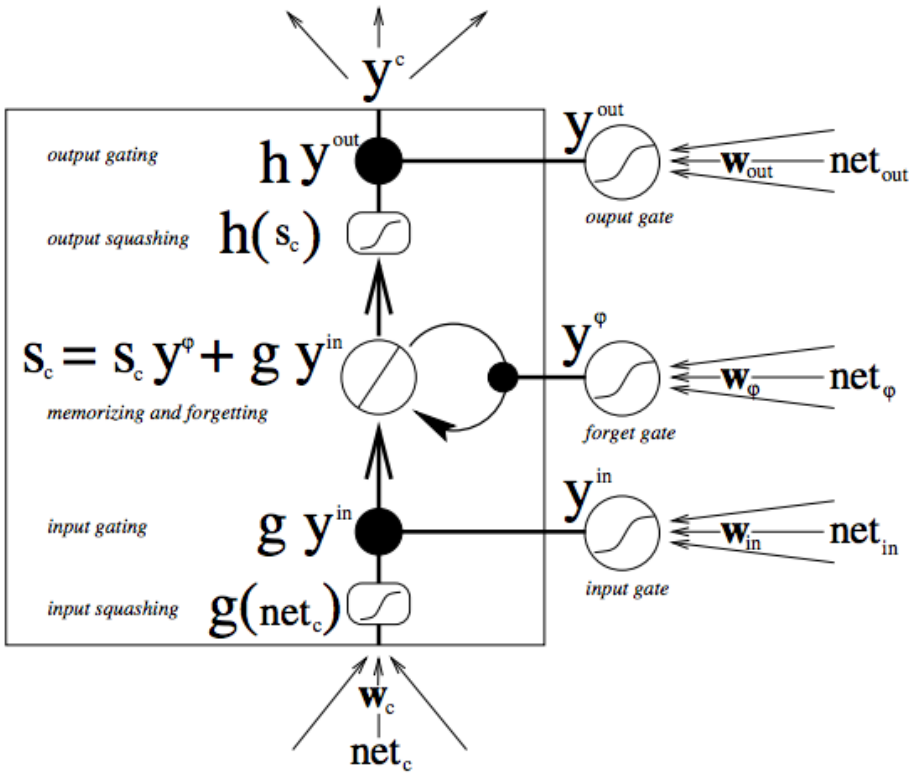

Figure 3.11 - Internal structure of an LSTM unit

We prefer LSTMs recurrent neural networks since they are able to seamlessly model multivariate time series forecasting problems. Due to their chain-like structure and associated memory, they are much more suitable for problems involving sequences and list (in our case, time series). Since the LSTM RNNs are inherently capable of learning simple as well as non-linear order dependence between records in a sequence, they are the ideal model for making trip predictions. In our model, we consider a multilayer RNN fed with inputs scaled to analog values between 0 and 1 for fine tuning and better results.

3.2.4 Allocating and Scheduling Buses

Bus scheduling (timetabling) and optimal allocation of buses is an important task in operations planning in public transit service. This is because, the resources expended contribute to the operational costs (in the form of fuel required for buses, cost for renting buses, cost for renting additional drivers, etc) of the service which makes it difficult for BEST service to generate

profits or break even. Thus, minimizing the number of buses required to cater to the requirements of the bus schedule and serve the general public is significant to reduce the expenses. We deal with the problem of bus scheduling and allocation as a combined problem.

We have derived an algorithm by combining the existing method with newer concepts of empty trips. We do not consider this problem from the perspective of minimizing passenger waiting time but rather minimizing the number of buses required and extensive reuse of already allocated resources. This approach inherently ensures minimization of passenger waiting time as the trip prediction methods implicitly consider passenger frequency per time slot.

Using these forecasted trips, we then generate a bus schedule (timetable) for each route using the Algorithm 1. This algorithm uses the trips for each slot to schedule the buses in each time slot with the aim of reducing the number of buses allocated for each route. The algorithm uses the concept of empty trips (sending an empty trip from in one direction to reuse the bus in other direction if required) and allocating new buses when a given trip cannot be scheduled by using the allocated buses.

The variables used are:

• List of forecasted trips for every timeslot for both directions: up_trips and down_trips

• Data structure to hold schedules in both directions: up_schedule and down_schedule

• Data structure to track buses arriving for rescheduling: up_arrival and down_arrival

• Bus identification number and number of buses allocated currently: Bus_tag

• The current hour for which the scheduling is to be done: curr_hour

• The next hour for which scheduling needs to be done: next_hour

• The current minute in the timeslot till where scheduling is done: curr_min

• The average travel time for buses in the given route: travel_time

• The number of trips yet to be assigned a bus for both directions: up_count and down_count

Algorithm 3.2.4.1: Bus Scheduling and Allocation in both directions for a bus route

Input: List of trips in both direction *up_trips and down_trips;* Bus travel time for the route *travel_time* in minutes

Output: Bus Schedule for the given route

1: Initialize *Bus_tag*← *0, curr_hour* ← *4, next_hour* ← *5, curr_min* ← *0*

2: Compute *up_count*= $\sum_{i=0}^{Number\ of\ Time\ Slots} up_trips[i]$and *down_count*= $\sum_{i=0}^{Number\ of\ Time\ Slots} down\_trips[i]$

3: while j ≤ Number of Time Slots do

4: If *up_trips[j]>1* then generate the schedule for *curr_hour* and append to *up_schedule* with headway between trips = Number of minutes in 1 hour / *up_trips[j]*

5: Else if *up_trips[j] == 1* then schedule the trip for *curr_hour* at 30 minutes past *curr_hour* and append to *down_schedule*

6: If *down_trips[j]>1* then generate the schedule for *curr_hour* and append to *down_schedule* with headway between trips = Number of minutes in 1 hour / *down_trips[j]*

7: Else if *down_trips[j] == 1* then schedule the trip for *curr_hour* at 30 minutes past *curr_hour* and append to *down_schedule*

8: j ← j+1, *curr_hour = curr_hour + 1*, *next_hour* ←*next_hour + 1*, *curr_min* ← *0*

9: end while

10: Create a sorted combined list of *up_schedule* and *down_schedule* named combined with *dir* ← *'U'* appended to *up_schedule* records and *dir* ←*'D'* appended to *down_schedule* records

11: *up_schedule_index* ← *0*, *down_schedule_index* ← *0*

12: while j ≤ len (*combined*) do

13: If *combined[j].dir== 'U'* then *down_count* ← *down_count – 1*

14: If *down_count> 0* then

15: If *len (up_arrival)>0* then sort *up_arrival* by time in ascending order

16: If *up_arrival [0].time ≤combined[j].time* then

17: *up_schedule [up_schedule_index].Bus_Tag* ← *up_arrival [0].Bus_tag*

18: *down_arrival.append (up_arrival [0].time + travel_time,up_arrival [0].Bus_tag)*

```
19:                 up_arrival [0].drop()
20:                 up_schedule_index ← up_schedule_index + 1
21:             Else sort down_arrival by time in ascending order
22:                 If down_arrival[0].time + travel_time ≤ combined[j].time then
23:                     up_schedule [up_schedule_index].Bus_Tag ← down_arrival
                        [0].Bus_tag
24:                     down_arrival.append (down_arrival [0].time +
                        2*travel_time,down_arrival[0].Bus_tag)
25:                     down_arrival [0].drop()
26:                     up_schedule_index ← up_schedule_index + 1
27:                 Else create new bus for the trip
28:                     Bus_tag ← Bus_tag + 1
29:                     up_schedule [up_schedule_index].Bus_Tag ← Bus_tag
30:                     down_arrival.append (combined[j].time +travel_time,Bus_tag)
31:                     up_schedule_index ← up_schedule_index + 1
32:             Else goto step 20
33:         Else goto step 20
34:     Else up_count ← up_count – 1
35:         If up_count> 0 then
36:             If len (down_arrival)>0 then sort down_arrival by time in ascending order
37:                 If down_arrival [0].time ≤ combined[j].time then
38:                     down_schedule [down_schedule_index].Bus_Tag ← down_arrival
                        [0].Bus_tag
39:                     up_arrival.append (down_arrival [0].time +
                        travel_time,down_arrival[0].Bus_tag)
40:                     down_arrival [0].drop()
41:                     down_schedule_index ← down_schedule_index + 1
42:                 Else sort up_arrival by time in ascending order
43:                     If up_arrival[0].time + travel_time ≤ combined[j].time then
44:                         down_schedule [down_schedule_index].Bus_Tag ← up_arrival
                            [0].Bus_tag
45:                         up_arrival.append (up_arrival [0].time + 2*travel_time,up_arrival
                            [0].Bus_tag)
```

```
46:                     up_arrival [0].drop()
47:                     down_schedule_index ← down_schedule_index + 1
48:                 Else create new bus for the trip
49:                     Bus_tag ← Bus_tag + 1
50:                     down_schedule [down_schedule_index].Bus_Tag ← Bus_tag
51:                     up_arrival.append (combined[j].time + travel_time,Bus_tag)
52:                     down_schedule_index ← down_schedule_index + 1
53:             Else goto step 40
54:         Else goto step 40
55:     j ← j+1
56: end while
57: Total_buses ← Bus_tag
58: Return up_schedule, down_schedule, Total_buses
```

3.3. EXPERIMENTAL SETUP

This section details the implementation setup for the proposed solution. The implementation works with constraints such as reusable buses that can be rescheduled, no overlapping routes, single type of buses with arbitrary capacity, no rescheduling during a day, different buses for different routes and end to end travel of buses. The three methods implemented are: ARIMA, SARIMAX and LSTM.

Generally, the ARIMAX model is preferred when the data is stationary or can be made stationary. In the ARIMAX model proposed, we first inspect and plot the available past trip data to identify the trends, seasonality and stationarity of data. Based on the various metrics (such as rolling mean, rolling standard deviation, residuals and p-test on the differenced and original data), we arrived at the ARIMAX (1, 1, 0) model. We do not directly use the I component defined in the standard ARIMAX library but rather define our own difference

function. This provides us better flexibility and control over the removal of non-stationary trend in data. The MA component has been omitted since we consider the available data without any random error. A sample data trend for a particular timeslot with respect to time steps (for our model, days) is as shown in Fig 3.12

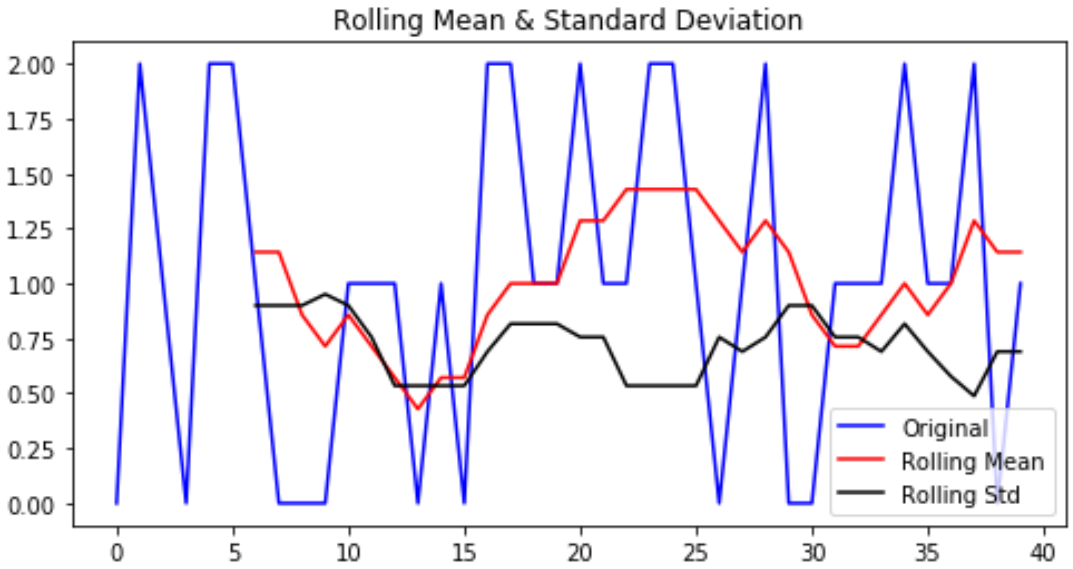

Figure 3.12 - Number of trips for the time slot 10:00:00 – 11:00:00 for route 0071 v/s time steps (as days)

In our implementation, we first predict the number of passengers per time slot for each route using an ARIMA model. We use the ARIMA (1, 1, 0) model for these forecasts as well, with the only exception being the absence of exogenous variables. Once the forecasts of passengers have been calculated, by using the past data and this predicted passenger value, we forecast the number of trips required for each time slot in a day. This is done for both directions and for each route. We smooth out the noise and outliers in the data by using the difference (Integrative component) to generate stationary data suitable for ARIMAX [11].

To cope with the seasonality trends in our data (the trip prediction values are have low variance for same days of the week), we use the SARIMAX model. Since we use daily data for our time series and there is high variation in the consecutive days' data to determine the trends, we look at the rolling mean, rolling standard deviation, seasonality (seasonal difference and different order of difference of these metrics) and residuals before tuning the model as shown in Fig 3.13 and Fig 3.14. The parameters chosen in our implementation stabilize the data by minimizing the seasonality factor for short term forecasts and ensure better ease of processing. The residual

values essentially take out the trend and seasonality of the data, making the values (and their variations) independent of time (Seasonality cycle). A SARIMAX (1,1,0) x $(0, 1, 1)_7$ model for our weekly data (S=7) can be written as in Equation (E) [14]. The additional seasonal terms are simply multiplied with the non-seasonal terms for better processing and a compact representation.

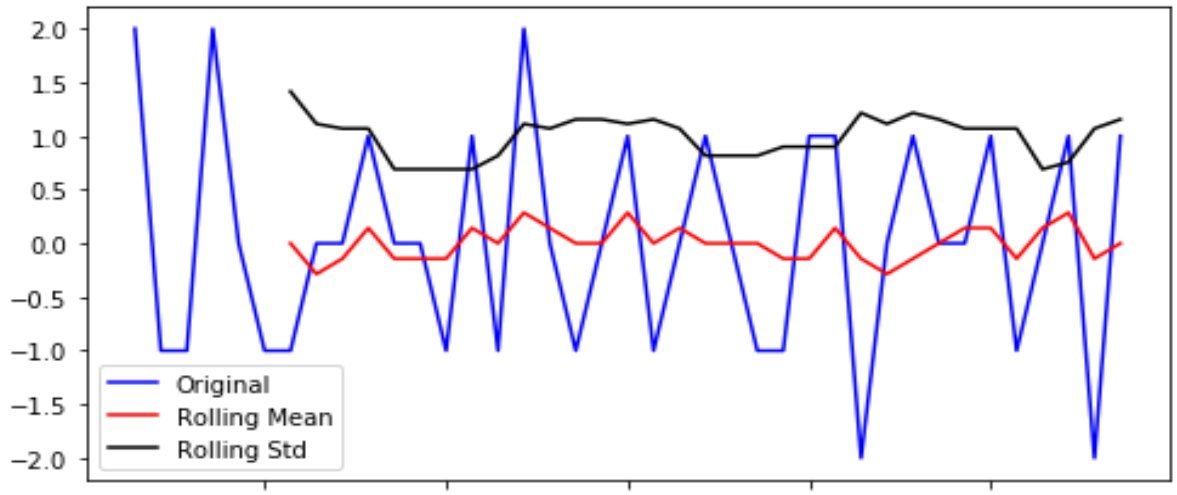

Figure 3.13 - Rolling mean and standard deviation of First difference of trips for 45 day from 1st Jan -14th Feb, 2018

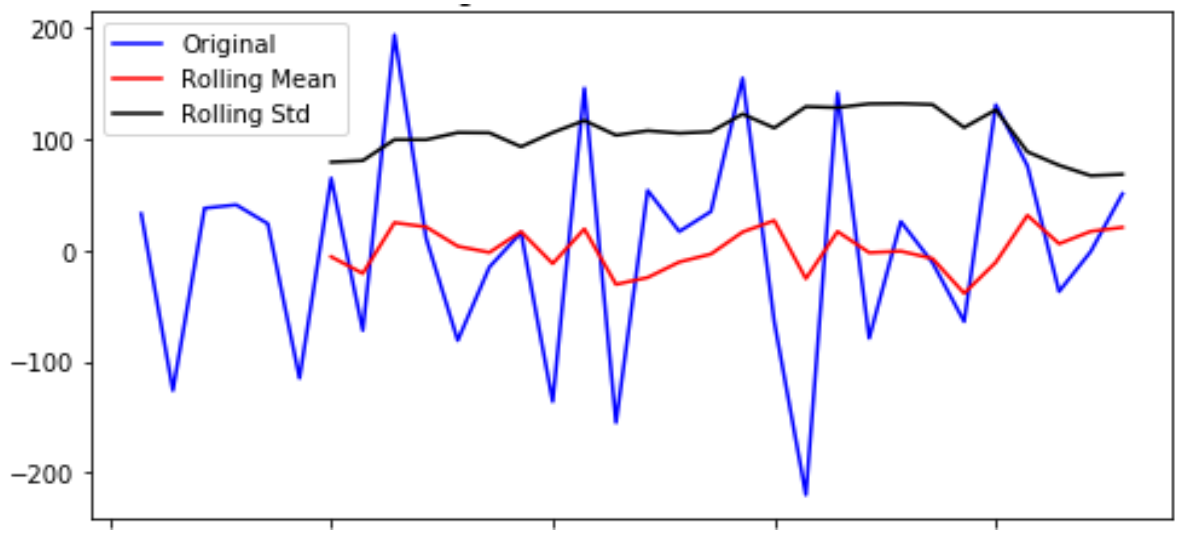

Figure 3.14 - Rolling mean and standard deviation of Seasonal First difference of passenger data from 1st Jan -14th Feb, 2018

In our implementation of the SARIMAX model, we first forecast the number of passengers per time slot for the given route by using a Seasonal ARIMA model. The configuration of the

model used is SARIMA(1,0,0)x(0,1,1)$_7$. The above model does not consider the non-seasonal MA and I components since the passenger data can be better modelled using seasonal MA and I components. Using the forecasted passenger data and other exogenous variables like the day of the week and whether it is a holiday on that day, we forecast the number of trips required for every timeslot for the given route using the SARIMAX(1,1,0) x (0, 1, 1)$_7$ model. From our inspection of different metrics and after performing the Dickey-Fuller test [15], we decided to use the non-seasonal first difference and the seasonal first difference of trips data (past data) in our model for better trip forecasts. The available passenger data (zero difference) and the seasonal first difference of passenger data is used for passenger forecasting.

In the implementation of the LSTM RNN model, we directly forecast the number of trips required per time slot for the given route instead of forecasting passengers (performed in the other two methods). Based on the visual inspection of data and the best formulations of LSTM networks on the internet, we finalized on a two-layer six nodes structure for the network with five hidden LSTM nodes. This configuration yielded the best results in comparison to the single node single layer and the two layers, 32 hidden nodes structure. The 32 hidden node structure provided similar results (about 90-95% same predictions) as the finalized configuration but required almost thrice the amount of time for training. We use the Adam optimizer [21, 22] for training the system, which is an extension to the gradient descent training algorithm. Other optimizers considered were classical stochastic gradient descent and RMS propagation for training, but they yielded inefficient results and required significant amount of time for training. Based on various executions of this model under different configurations of epochs and batch size, we found optimal results with 20 epochs, each having a batch size of three. The results obtained from this model had better variations for different time slots, weekdays and weekends. It provided smoother results with very few outliers during the testing of system on small test sets of 40 samples.

Once the trip forecasts for the next day are available, we begin the bus allocation and scheduling process for each route based on the trip forecasts. The algorithm 1 defined in the proposed solution section has been formulated after understanding the manual bus scheduling process. We have considered optimizations such as using empty trips in both directions for better re-use and maintaining the average headway within the specified limits of the transit

service. This method has been formulated with the focus on minimizing the number of buses required and thus it succeeds in doing so for the considered routes when compared with the current allocations for each route. This ensures the total number of buses required is less than the current allocations. On further calculation of statistics based on the results obtained, we found that the sum of the number of buses allocated to all the routes is at least 5% less than the manual allocation method currently used.

This experimental setup has been mainly taken care of in the form of an application having a GUI to interact and upload new data. This application also helps us to create forecasts, schedules and allocations for specific days in the future. Also, providing a GUI ensures for better localization of errors and easy debugging. The execution flow and snapshot of the user interface design of the GUI application for our solution is given in **Appendix B**.

Chapter 4.

Results and Discussions

Once the trip forecasts for the next day are available, we begin the bus allocation and scheduling process for each route based on the trip forecasts. The algorithm 1 defined in the proposed solution section has been formulated after understanding the manual bus scheduling process. We have considered optimizations such as using empty trips in both directions for better re-use and maintaining the average headway within the specified limits of the transit service. This method has been formulated with the focus on minimizing the number of buses required and thus it succeeds in doing so for the considered routes.

The results of our comparative study between our proposed solution and SVM and Polynomial Regression are shown in Table 4.1.

Table 4.1 - Comparative study of different methods for trip forecasting over all routes

Date	**SVM**	**ARIMAX**	**SARIMAX**	**Poly Regression**	**LSTM**
8/2/18	5	5	4	6	5
9/2/18	5	7	5	5	8
10/2/18	3	5	5	6	6
11/2/18	2	3	4	5	4
12/2/18	5	3	3	5	5
13/2/18	7	3	5	7	7
14/2/18	5	3	5	5	7
Average	4.57	4.14	4.42	5.57	6

We have compared the predicted trips of all the timeslots for a week starting from 8th February, 2018 against the data obtained from Vikhroli depot. Here, the values in each cell for each method mainly depicts the number of times the forecasts were incorrect with respect to the number of passengers and the current number of trips being used. The values suggest that ARIMAX and SARIMAX models proposed in this paper perform better than the SVM and Polynomial Regression models as higher the average error value, indicates higher are the deviations in the forecasts in opposite direction from the seen trend. This can be further explained with an informative example. When we observe that the number of passengers on an average is higher as compared to the number of current trips per hour (we consider bus capacity as maximum 80 passengers per bus), we expect the forecasts to generate more number of trips per hour and vice versa, if the number of current trips per hour are excess as compared to passenger frequency, then we expect the trip forecasts to reduce the number of trips per hour. If the forecasts are opposite to this rule, we consider the forecasts to be erroneous and thus calculate the number of times the forecasts have deviated. The values for individual days, as shown in Table 4.1, are considered for 320 data points. Polynomial Regression performs the worst due to its inability to capture rapid changes and the presence of outliers. LSTM and SVM show comparative results but do not show significant improvement over Polynomial Regression. ARIMAX and SARIMAX perform the best as they track the seasonality trends and rapid change patterns in our data and also keep a track of the outliers much better.

Another consideration for determining the legitimacy of the methods is comparing the headways. Headway is maximum permissible time after which the bus must be dispatched on the route. The maximum and the minimum columns in the table indicate the maximum and minimum headways for each route For example, if the headway for a route is 30 minutes then maximum permissible time between dispatches of two buses must be 30 minutes. In our proposed solution we have calculated the headway as:

Headway = ∑difference between the scheduled times of consecutive schedules/no. of trips

Table 4.2 – Comparison of Headways calculated for five routes using different methods

Routes	Minimum	Maximum	ARIMAX	Poly Regression	SARIMAX	SVM	LSTM
3860	10	26	23	24	22	24	25
4100	20	31	18	28	16	18	23
4781	20	23	25	26	26	28	28
6020	5	10	8	11	9	9	10
6040	12	18	20	21	18	19	20

Values in each of the last five cells of the table 4.2 above demonstrates the headways calculated for each route by each method (in minutes). The headway calculated for a particular day is then compared with predefined headways that are available for Vikhroli depot. From the table 4.2, we see that the values calculated in each cell are in the permissible range ensuring there is no degradation of service compared to the current method and permissible waiting time limits for passengers across all routes. On comparing the different methods, we see that the polynomial regression produces average headway values above the permissible values about 60% of times whereas LSTM performs similar to polynomial regression but generates headways within the permissible limit more frequently than polynomial regression. The best performance in terms of headway has been observed for SARIMAX and then for ARIMAX followed by SVM. The headway values for SARIMAX are normally always within the permissible limits, almost 80% of the times. ARIMAX performs better than other methods but

mainly suffers due to its inability to identify and remove cyclic/seasonal trends in the data. SVM provides a result which is representative of the normal mean behaviour, having the average headway values within the limits around 50-60% of the time.

Table 4.3 – Comparison of Bus allocations for given routes using trip forecasts from different methods

Routes	ARIMAX	Poly Regression	SARIMAX	SVM	LSTM	Original
3860	14	12	11	11	10	8
4100	19	15	16	27	9	20
4781	17	17	16	18	7	27
6020	25	28	26	24	20	28
6040	11	11	20	18	8	9
TOTAL	86	83	89	98	54	92
% CHANGE	6.52	9.78	3.26	-6.52	41.3	

Table 4.3 depicts the number of buses allocated by different methods for a subset of all the routes specified (The subset contains a combination of high and low passenger frequency routes which is representative of all the routes from Vikhroli Depot). The algorithm 3.2.4.1 is used to schedule and allocate buses on the specified routes. The last column 'Original' has the data obtained from the Vikhroli depot i.e., it specifies the number of buses being currently utilized on the corresponding routes.

Although the number of buses allocated on certain routes is greater than the original number of buses, the total number of buses over all the routes sees a general reduction trend from the existing solution's total number of buses being utilized. As the table shows, SVM is the only method that increases the number of buses that are required to be used – signified by the negative value (-6.52%) in the '% Change' row - whereas ARIMAX (6.52%), SARIMAX (3.26%), Polynomial Regression (9.78%) and LSTM (41.3%) all reduce the total number of buses thereby potentially saving costs. The polynomial regression method ensures relatively

high savings as compared to ARIMAX and SARIMAX, but the trip forecasts as well as headway estimations contain high deviations from the current values as well as in comparison with other methods considered. The most significant observation is the significantly high % change for LSTM, namely 41.3%. We can observe that the % saving for LSTM is higher than the cumulative sum of all the other methods which helps us consider it as a valuable solution when the constraint of cost-cutting is of high importance.

From table 4.2 and table 4.3, we can infer that along with maintaining the predefined headway, the methods also minimize the number of buses required. This helps us gain better understanding of the correlation between average headways and % saving, where we see a slight negative correlation between the two in the case of LSTM and SARIMAX as we see that higher the number of times the headway is outside the permissible range, higher is the amount of reduction in number of buses allocated for all the routes together.

Thus, from the results and the different comparison measures discussed above, we find that over all the different constraints of headways, minimal bus allocation, deviations from intuitive manual trends, we see that ARIMAX and SARIMAX provide comparable results, with best results obtained in the case of ARIMAX. These two methods perform much better than the other methods overall. LSTM mainly performs exceptionally well when we consider minimizing bus allocation and provides good results when the constraint of headway is considered. It mainly performs comparably mediocre in case of the first constraint, which makes its performance poor than ARIMAX and SARIMAX overall. The other two methods mainly provide random results, with few good forecasts for certain routes, mainly for less frequency routes, while sub-par performance for medium to high frequency routes over all the three constraints considered.

Chapter 5.

Summary and Conclusions

5.1 Summary

The proposed solution mainly has been divided into two core components of predicting trips based on past data using statistical and machine learning models, and bus allocation and scheduling using an algorithm that allocates resources based on a variation of job shop scheduling problem. There are other auxiliary components as well to support the integration of these two core components and for data cleansing and handling storage. Here, we take a different approach to transit services operation planning from the conventional one, in that we consider trip prediction to be much more indicative of the bus utilization levels for better planning and utilization of resources. This strongly differs from the headway approach of operation planning (scheduling buses according to fixed headways manually) which considers the availability aspects while sacrificing efficient utilization of resources. The ARIMAX and SARIMAX models ensure that the seasonality of weeks and months is better represented in the trip prediction and subsequent bus allocation and scheduling. The trip prediction algorithms also take into account peak time slots of a day and consequently predict trips based on the timeslot and passenger frequency. This allows us to optimally allocate resources. The proposed scheduling algorithm ensures better bus utilization while also ensuring that the average headway for a route is within permissible limits to avoid long waiting times for passengers. The bus allocation and scheduling algorithm simply allocates optimal number of buses to each route for daily operation according to the predicted number of trips for each hour and the start travel time of buses along the route in both directions, up and down. The analogy with the job shop scheduling relates trips to jobs to be performed and buses to resources required to perform the job. The variation in this job shop scheduling formulation is due to the reusability of resources and bi-directional graph like structure of job dependency.

5.2 Conclusions

Based on the results obtained for the proposed solution as compared to the current solution, we find that the new system

1. Ensures all considerations of the current solution (availability of buses, reasonable waiting time for passengers and spare buses for unforeseen situations)
2. Ensures optimal number of buses is used for each route to cut down operational costs while maintaining an acceptable level of efficiency (in terms of headway)
3. Helps reduce manual labour being used in generating schedules and limits the user interference in the process
4. Helps overcome inconsistencies that arise depending on the human tasked with working on schedule generation
5. Generates schedule dynamically as opposed to the current implementation where the timetable is static for four months

We also find that out of the five methods considered, ARIMAX and SARIMAX perform exceptionally well as compared to the other three. LSTM however, still provides us with an intriguing alternative that is highly appealing for trip forecasting when the main aim is to minimize bus allocation across all routes without compromising substantially over headway and service quality. Also, from our observations, we find that by using any of the three methods mentioned above, we can ensure a reduction of around ~4% in the number of buses allocated while improving on the current manual solution in terms of number of hourly trips and average headways.

5.3 Future Enhancements

The proposed system is based on a few assumptions which open the door for future improvements.

First, the future system could consider overlapping routes that would bring in additional complexities in terms on routes the buses travel. It could also allow for buses to change routes at intersection points and allow for better utilization.

Second, the future implementations could look at the possibility of altering the number of buses along various routes depending on the passenger traffic on those routes as the day progresses with the help of live tracking. This could help prevent the dead headways (empty trips) by utilizing those buses on routes that could improve passenger satisfaction with the help of more resources.

Third, the buses could be scheduled to run only a subset of the end-to-end routes and subsequently even change routes. This could be beneficial in cases when the later stops are known to be visited infrequently by passengers and hence, not all buses need to travel the entire distance.

Fourth, in the future, the option of having different types of buses (express buses, buses with differing capacities, etc) could be used to exploit patterns found in passenger frequencies on different routes.

Thus, the proposed solution has a lot of scope for extension and improvement in the future.

References

[1] https://www.alphabet.com/en-ww/article/public-transport-system-knife-edge-mumbai (Last Visited : 15th May, 2018)

[2] https://ridlr.in/blog/list-of-best-depots-in-mumbai-with-routes-stations-numbers/ (Last Visited : 15th May, 2018)

[3] Makrand Wagale, Ajit Pratap Singh, Ashoke K Sarkar and Srinivas Arkatkar, "Real-time Optimal Bus Scheduling for a City Using A DTR Model," in *IEEE Electron DeviceLett.*, vol. 20, pp. 569–571, Nov. 1999.

[4] Eshetie Berhan, Dejene Mengistu, Berhanu Beshah and Daniel Kitaw, "Modelling and Analysis of Bus Scheduling Systems of Urban Public Bus Transport," (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[5] Avishai (Avi) Ceder, "Optimal Multi-Vehicle Type Transit Timetabling and Vehicle Scheduling," M. Shell. (2002) IEEEtran homepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

[6] Carbno Collin, "Optimal Resource Allocation for Projects,"*FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.

[7] Anirudha Nanda, Manisha P. Pai, and Abhijeet Gole, "An Algorithm To Automatically Generate Schedule For School Lectures Using A Heuristic Approach,"

[8] Liping Fu, Qing Liu, and Paul Calamai, "Real-Time Optimization Model For Dynamic Scheduling Of Transit Operations,"

[9] https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/ (Last Visited : 15th May, 2018)

[10] http://www.forecastingsolutions.com/arima.html (Last Visited : 15th May, 2018)

[11] https://people.duke.edu/~rnau/411arim.htm (Last Visited : 15th May, 2018)

[12] https://www.researchgate.net/publication/299491997_Using_Social_Media_to_Perform_Local_Influenza_Surveillance_in_an_Inner-City_Hospital_A_Retrospective_Observational_Study

[13] 21. Box GEP, Jenkins GM, Reinsel GC. Time series analysis: forecasting and control. Hoboken, NJ: John Wiley; 2008

[14] http://www.seanabu.com/2016/03/22/time-series-seasonal-ARIMA-model-in-python/ (Last Visited : 15th May, 2018)

[15] https://people.duke.edu/~rnau/seasarim.htm (Last Visited : 15th May, 2018)

[16] https://onlinecourses.science.psu.edu/stat510/node/67 (Last Visited : 15th May, 2018)

[17] https://www.otexts.org/fpp/8/9 (Last Visited : 15th May, 2018)

[18] http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (Last Visited : 15th May, 2018)

[19] https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5 (Last Visited : 15th May, 2018)

[20] https://deeplearning4j.org/lstm.html#long (Last Visited : 15th May, 2018)

[21] http://ruder.io/optimizing-gradient-descent/index.html#adam (Last Visited : 15th May, 2018)

[22] https://arxiv.org/abs/1412.6980 (Last Visited : 15th May, 2018)

Appendices

Appendix A

Development and Execution Environment Installation Details

This document explains how to compile and install Anaconda IDE and Keras library in Anaconda.

The following commands were tested on Windows 10 and Windows 8.1, and should be modified accordingly for other distributions.

1. OBTAINING AND INSTALLING ANACONDA

You can download and install the IDE by following the given instructions:

I. Download the installer for Anaconda IDE from:

- https://www.anaconda.com/download/#windows

II. Double-click the .exe file to install.

III. Follow the instructions on the screen.

IV. If you are unsure about any setting, accept the defaults. You can change them later.

V. When installation is finished, from the **Start** menu, open the Anaconda Prompt and test your installation.

An alternative is to check out the new version of Anaconda IDE on their website and follow the documentation there for installation.

2. OBTAINING AND INSTALLING KERAS DEPENDENCY

- To install the Keras library on Anaconda IDE, simply type the following instruction on the conda command prompt
  conda install -c conda-forge keras
- After execution of the above instruction, test your Keras dependence.

3. RUNNING KERAS AND JUPYTER NOTEBOOKS ON ANACONDA

- To use Keras in Jupyter Notebooks, first start Jupyter Notebooks using Anaconda IDE and try importing Keras. It should work flawlessly.
- Alternative approach is to use command Jupyter Notebook in Anaconda prompt to open Jupyter Notebooks and then import Keras. It should work in the same way as before.

4. Installing PyQt4 on Windows for GUI development

- Downloading SIP
  1. SIP must be installed before building and using PyQt4. You can get the latest release of the SIP source code from http://www.riverbankcomputing.com/software/sip/download.
  2. The SIP documentation can be found at http://pyqt.sourceforge.net/Docs/sip4/.
- Downloading PyQt4
  1. You can get the latest release of the GPL version of the PyQt4 source code from http://www.riverbankcomputing.com/software/pyqt/download.
  2. If you are using the commercial version of PyQt4 then you should use the download instructions which were sent to you when you made your purchase. You must also download your license file.

- Configuring PyQt4
  1. After unpacking the source package (either a .tar.gz or a .zip file depending on your platform) you should then check for any README files that relate to your platform.
  2. If you are using the commercial version of PyQt4 then you must copy your license file to the sip directory, or to the directory specified by the --license-dir option of configure-ng.py.
  3. You need to make sure your environment variables are set properly for your development environment.
  4. In order to configure the build of PyQt4 you need to run either the configure-ng.py or the configure.py script.
  5. configure.py is the original configuration script that uses the build system of SIP v4 (i.e. the sip.sipconfig module). It will be supported for the life of PyQt4. configure-ng.py is the new configuration script that uses Qt's qmake program to do all the heavy lifting.
  6. The configure-ng.py script is used to configure PyQt4 as follows:
     python configure-ng.py
  7. This assumes that the Python interpreter is on your path. Something like the following may be appropriate on Windows: *c:\python34\python configure-ng.py*
  8. If you have multiple versions of Python installed then make sure you use the interpreter for which you wish to build PyQt4 for.

To check whether the PyQt4 framework has been correctly installed, we import PyQt4 modules. If there are no errors, then PyQt4 can now be used for GUI development.

Appendix B

Routed System User Interface and Operations details

This document depicts the various implementation and user interface design properties of the solution. We also depict the format of the various files that are generated by the system.

1. Main layout of the GUI for the Routed System

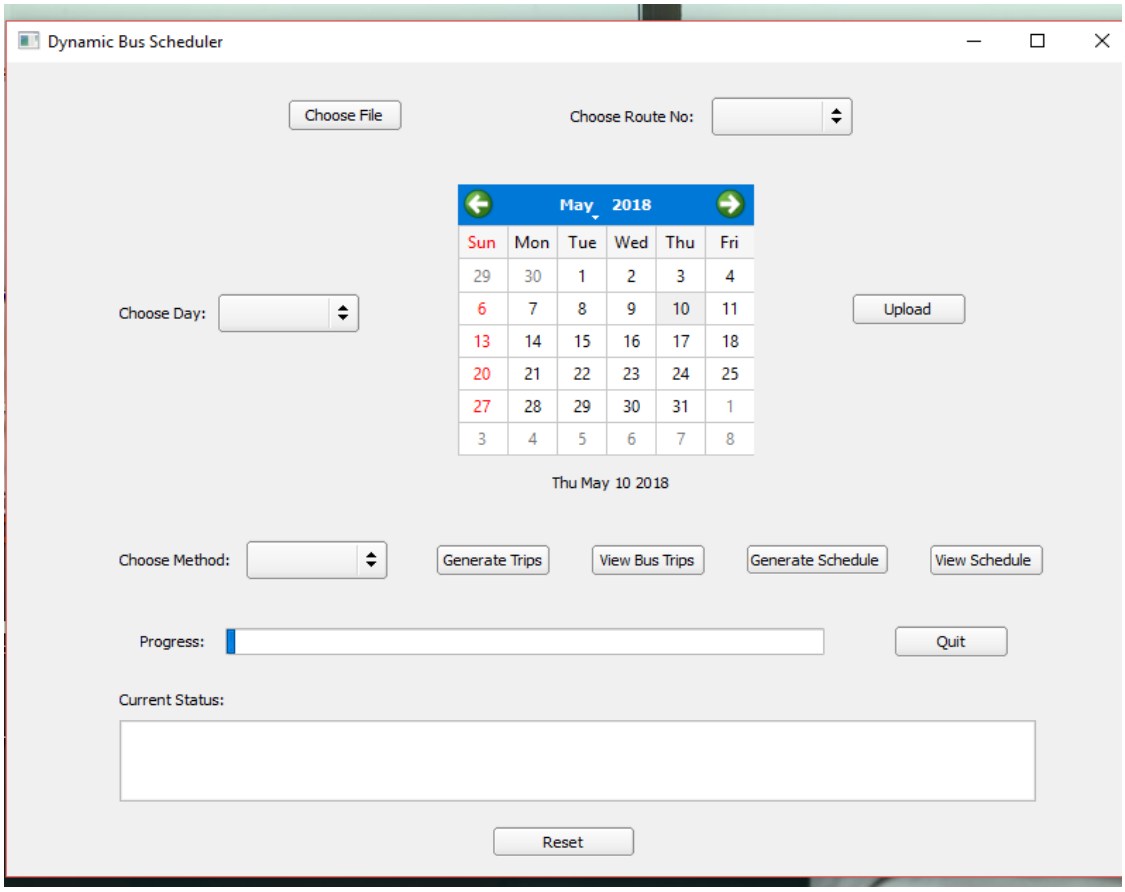

Figure B.1 – User Interface Layout of Routed System

The layout mainly depicts the different interaction components such as,

- File picker to upload the passenger data file for the current day.
- Route selector for selecting the correct route and to tune the internal parameters accordingly.
- Day selector for selecting the day of the week to input day type.
- Date picker for selecting the date we want to create the forecast for.
- Upload button to commit the input and selection data to the system.
- Method selector to select the forecast method to use.
- Generate trip button to forecast the trips for selected date.
- View trips button for displaying the .csv file created containing all the trip forecasts.
- Generate schedule button to create bus schedule and bus allocations for given route, day and date using the algorithm from Algorithm 3.2.4.1 in **chapter 3**.
- View Schedule button to open the .csv file containing the bus schedule and allocation predictions for given route, day and date.
- Progress bar to provide visual feedback about the amount of processing left.
- Status box which provides information about internal state of system and success/failure status of different operations.
- Reset button to reset the state of the system in case the system fails or becomes unresponsive.
- Quit button to close the application.

2. System operations

The Routed system can in simplest terms, pre-process and store passenger data (Data collection and Data pre-processing modules), forecast trips using the selected method (Forecasting trips module), schedule and allocate buses for given route, day and date (Allocating and scheduling buses module).

- File selection and upload

  We select the input file using choose file button and fill in the details about the day for which the forecast is to be generated using other selection tools.

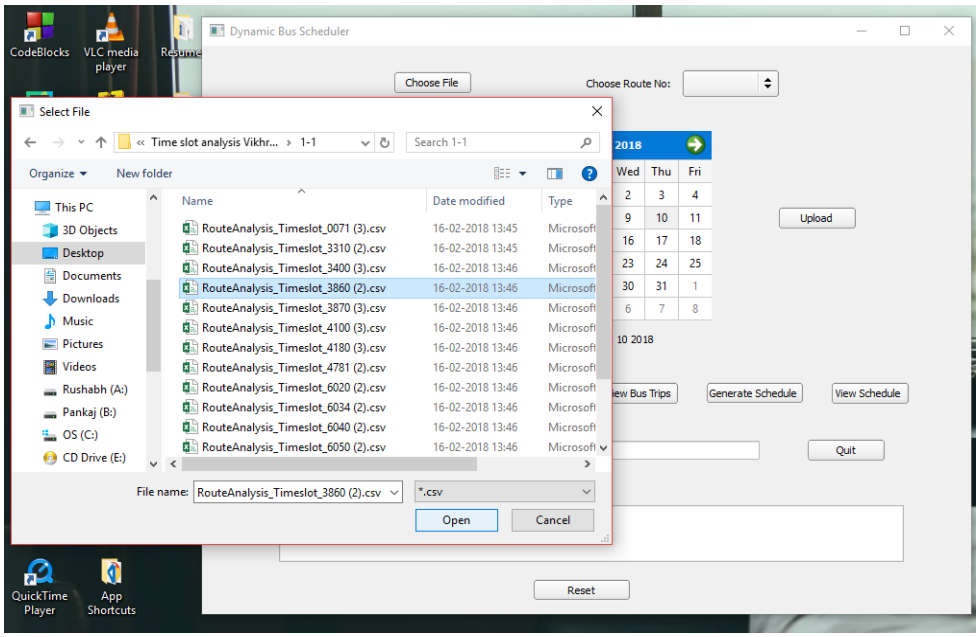

Figure B.2 – File selection and upload

- File Cleaning and storage

  The system cleans the file after uploading of file and creates a new cleaned file at the location shown in status box.

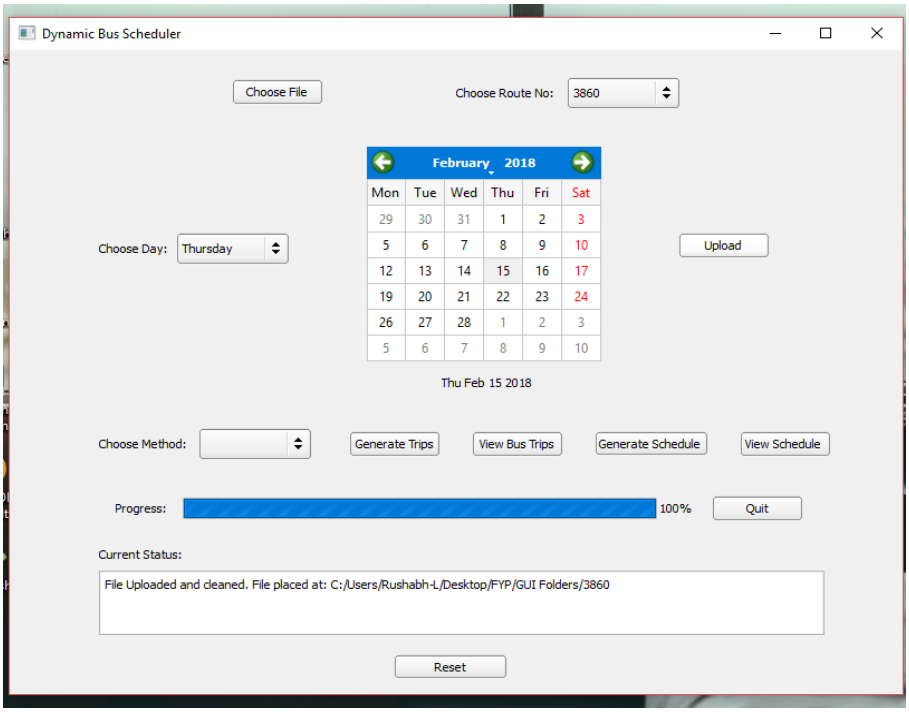

Figure B.3 – File cleaning and storage

- Forecasting trip method selection and processing

  Here, we mainly show an example execution using Arimax method for trip forecasting in both directions and the created file has been displayed using view trips button.

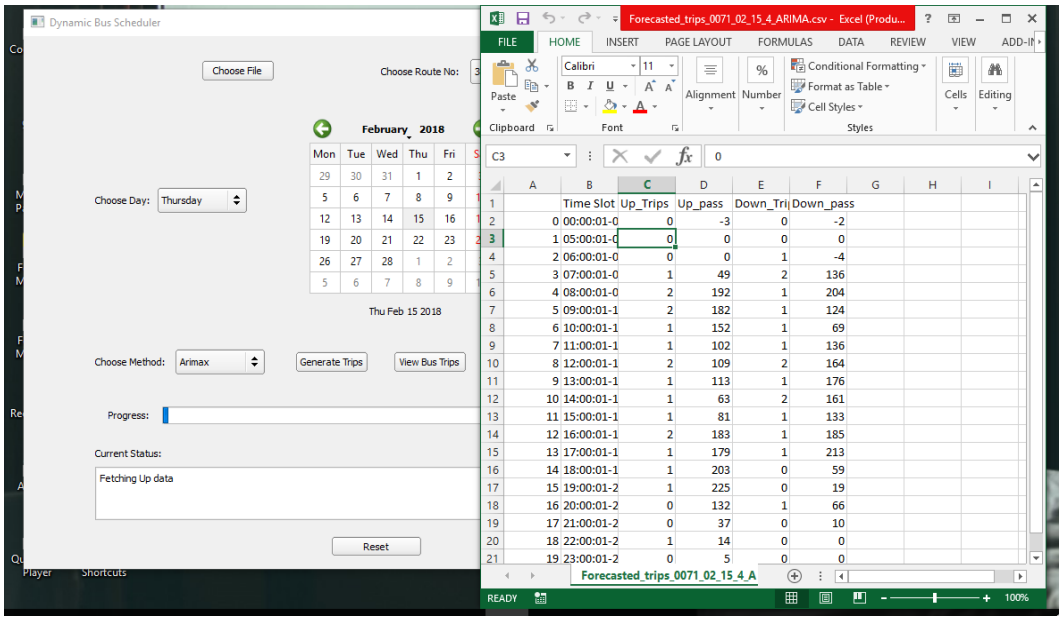

Figure B.4 – Forecasting trip method selection and processing

- Generating Bus Schedule and Allocation

  By using the generate schedule button from **figure B.1**, we can generate bus schedules and allocation of buses and store the output file. To open the file, we use view schedule bus button.

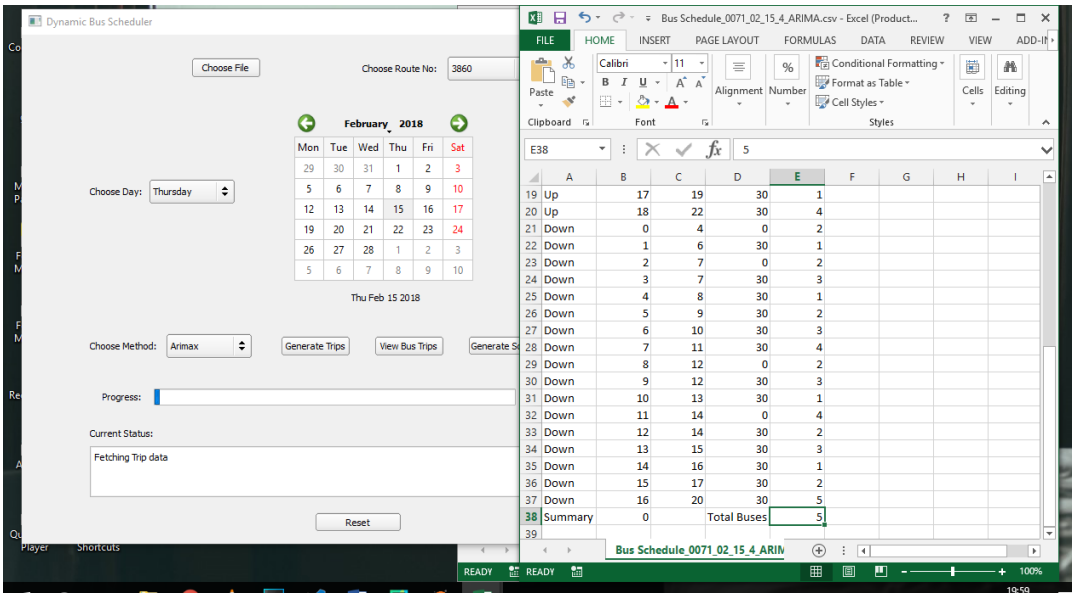

Figure B.5 – Generating Bus Schedule and Allocation

- Reset System

  In case of error/failure or if the system becomes unresponsive, we use the reset button to reset the state of the system.

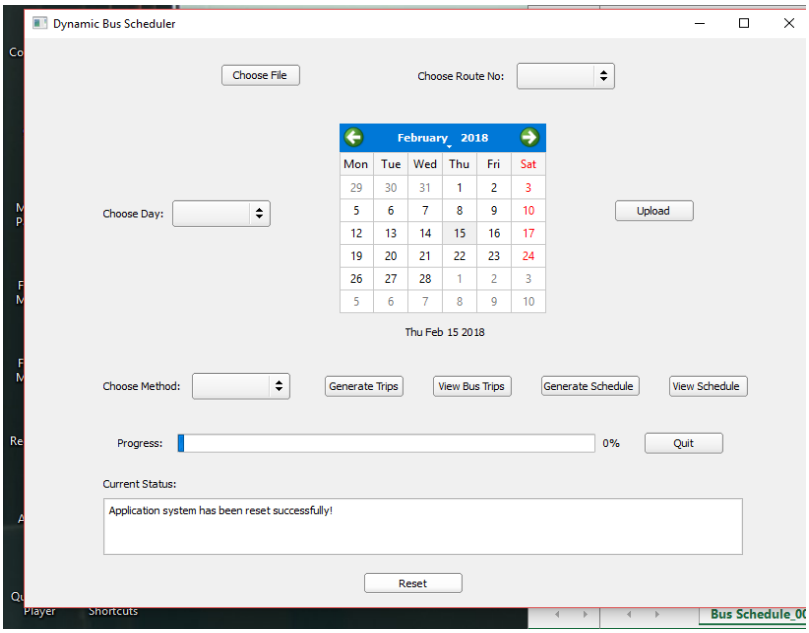

Figure B.6 – System Reset

The system thus provides a simple interface that is easy to use.