Design of Passenger Information System and Smart Passenger Crowdedness Avoidance: Case of AALRT

Yetages Rorissa Koricha

A Thesis Submitted to

The School of Electrical and Computer Engineering

Presented in Fulfilment of the Requirements for the Degree of Master of Science (Electrical and Computer Engineering)

Addis Ababa University

Addis Ababa, Ethiopia

June 2017

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

This is to certify that the thesis prepared by Yetages Rorissa, entitled: *Design of Passenger Information System and Smart Passenger Crowdedness Avoidance: Case of AALR* and submitted in partial fulfillment of the requirements for the degree of Master of Sciences (Electrical and Computer Engineering) complies with the regulations of the University and meets the accepted standards with respect to originality and. quality.

Signed by the Examining Committee:

Examiner _____ Signature _____ Date _____

Examiner _____ Signature _____ Date _____

Advisor _____ Signature _____ Date _____

Advisor _____ Signature _____ Date _____

Declaration

I, the undersigned, declare that this thesis is my original work, has not been presented for a degree in this or any other university, and all sources of materials used for the thesis have been fully acknowledged.

Yetages Rorissa _____

Name Signature

Place: Addis Ababa

Date of Submission: _____

This thesis has been submitted for examination with my approval as a university advisor.

Abebe Teklu _____

Advisor Signature

Abstract

Addis Ababa Light Rail Transit is playing a prominent role in alleviating the city's transportation problem. Currently, it only has an onboard passenger information system, and there is no wayside passenger information system. Consequently, passengers waiting at the station are unable to know if a train is coming, the estimated arrival time of the train, the crowdedness level on the train and whether the train is single or coupled. The other challenge is that on pick hours there exist a high passenger crowdedness on the trains which makes the service unintentionally unfair as only who can survive the high congestion on the train use the service.

The primary objective of this thesis is to avoid the existing passenger information system problems of Addis Ababa Light Rail Transit by devising a reliable passenger information system and come up with a cost-effective solution for the current passenger over crowdedness problem. Hence, a passenger information system was designed by developing a website, an Android application and station displays which will all show the available space on the next train, the number of passengers expected to get off on that station, and its estimated remaining time of arrival. The simulation result showed that we achieved the stated objective as it can give detail information about the train for each passengers waiting at the stations. In addition to this, we introduced a passenger overcrowding solution. It is a passenger counting system that will put a limit on passengers getting on a train when the train load exceeds its carrying capacity. Android based point of sale (POS) ticketing device is proposed to be used as a passenger counter and ticketing at the same time. It is expected that the current manual ticketing problems encountered by Addis Ababa Light Rail Transit can be solved by this device.

Keywords: passenger information system (PIS), Over Crowdedness avoidance, Android-based POS ticketing device, AALRT

Acknowledgement

I always regret the times that I did not thank God for letting me accomplish what I accomplished so far. And I don't want to make the same mistake in this great work. So first and foremost, praises and thanks to the God, the Almighty, for his mercy and showers of blessings.

I would like to express my deep and sincere gratitude to my research advisor, Mr. Abebe Teklu, for his dedication and invaluable guidance throughout this research. His dynamism, vision, sincerity, perfection, and motivation have deeply inspired me. He made me think the research in a wider way.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my better future. I am the man I am today, because of the man who raised me. Rest in peace my beloved father, I will always remember you.

Finally, I would like to thank Alpha University (Main Campus) security men who let me use the University's fast internet on the daily basis, for the last 12 months. Alpha University librarians were also very cooperative in providing me with the resources I needed in completing this research. If it wasn't for their cooperation, I couldn't have finished this work at all. I am also grateful to my lovely friends too.

Table of Contents

List of Figures

List of Tables

Abbreviations and Acronyms

AALRT Addis Ababa Light Rail Transit

AS Available space

ATD Android Ticketing Device

APC Automated passenger counter

ASA Automatic stop annunciators

AVL Automatic vehicle location

CMD Command prompt

CPN Current number of passenger on train

$\mathbf{DT_n}$ Dwell time at station

DRT Demand responsive transit

ETA Estimated time of arrival

ERC Ethiopian Railway Corporation

FS Free space

GIS geographic information systems

GPRS general packet radio service

GPS Global Positioning System

HTML Hypertext markup language

IP Internet protocol

ITS Intelligent transportation systems

JS Javascript

LAN Local area network

LGFNP Last get off number of passengers

LGONP Last get on number of passengers

LRT Light Rail Transit

NTS Number of ticket sold

PAO The area passanger occupied on tain

PIDs Public information displays

PIS Passenger Information System

PNP Previous number of passengers on train

POS Point of sale

PIXC passengers in excess of capacity

RF/ID Radio Frequency /ID

SCADA Supervisory control and data acquisitions

SIM Subscriber identity module

TA Total area

UK United Kingdom

URL Uniform resource locator

USA United States of America

Wi-Fi Wireles fidelity

$\mathbf{X_{didnt\_get\_on}}$ → Number of passengers who bought a ticket but didn't get on a train

3G Third generation

Chapter 1

Introduction

Passenger information, entertainment, and security systems are indispensable in Light Rail Transit and other mass transit transportation modes. They respond to the changes underway in the railways and mass transit global environments, such as government debt reduction, demands of the aging population, integration of disabled people in society, private–public partnerships, utilizing information technology to lower costs, improved customer services, and enhanced commuter safety and security. Several major cities (New York; Montreal, Quebec; Hong Kong; Santiago, Chile) around the world have successfully introduced passenger information, entertainment, and security technologies that also allow for the generation of advertising revenues [1].

For too long, railway passengers have suffered from poor information about services, particularly at times of disruption. Reliable and timely information is essential for planning travel, and when things go wrong, for assessing how best to complete journeys. Addis Ababa light rail transit, in spite of its great role in easing the transportation problem, it came into operation with some problems that have to be dealt with. As it was stated before, two of the biggest problems that were observed and have become a base for this thesis are the lack of PIS for passengers waiting at the stations and the high congestion of passengers on the trains which is common to see especially during rush hours.

It is know that the estimated transportation capacity of AALRT is around 600,000 passengers per day and this transportation capacity is incapable of supporting the current transportation demand of the city. Consequently, there is always too much crowdedness on the trains in East-West (EW) and North-South (NS) corridors, especially during the rush hours. With the current situations, many weak, teenagers and old peoples are facing the difficulty of accessing the transportation services. To tackle this problem, the obvious solution will be increasing the number of trains which is normally expensive and sometimes not feasible as the carrying capacity of the line is limited to a fixed number. Thus this thesis has presented the devised system to deal with the stated problems.

1.1 Problem Statement

On October 21, 2016, in the afternoon there was a heavy rain following which train service was halted for more than an hour. Around Saris station, the flood covered the rails that until the flood subsided the train service had to be stopped. And it took more than an hour for the trains to get back to service. But the passengers who were waiting for the trains in other stations did not know what was going on Saris station. Hence, they were made to wait for hours. If there was a wayside PIS the passengers could have been informed with the developing scenario so that they could have resolved for other means of transportation. This incident clearly shows that the lack of wayside passenger information system is troubling the passengers who are waiting for a train. On top of that, there is a huge discrepancy between the stated arrival time of the trains and the actual time the trains take to arrive at a station. The stated arrival time is 10 minutes, which means that in every station a train will arrive within 10 minutes of time. But the real situation shows that the trains take more than 30 minutes to arrive at a station. In a nutshell, the first problem that has inducted this thesis is the lack of wayside passenger information system.

The second problem is that the LRT is facing a high passenger congestion on a daily basis especially on pick hours. With the current situations many weak people, teenagers, primary school students and old peoples are facing difficulty in accessing the transportation services. Thus the service is becoming unintentionally unfair as only people who could survive the high congestion on the trains get the service. To tackle this problem the obvious solution will be to increase the number of trains which is costly and sometimes not feasible as the train carrying capacity of the lines is limited to a fixed number. Thus a cost effective, easy and fast to implement passenger overcrowding avoiding system is a needed.

The author personally has faced these two problems. Sometimes when there is an electric power outage the trains will be forced to stop on their way. When this happens the passengers who are waiting for their trains don't know it because there is no way of information passing to them. On top of that, the crowdedness on the trains is beyond tolerable, especially during rush hours. One can't even carry his laptop or any other fragile things if he is going to use this train service. This problem has made a comfortable environment for thieves. Every day we hear people losing their wallets, phones, money and many other things. Sometimes when the trains get overcrowded with passengers it is very difficult to get off from the trains. Many peoples

were seen getting off the train after they have passed one or two stations away from the station where they were supposed to get off. This over crowdedness has also resulted many other problems.

1.2 Research Objective

1.2.1 General Objectives:

The general objectives of this study are mainly two. The first is to avoid the existing PIS problem at the stations of AALRT by devising a dependable PIS that will assist passengers on the waiting station. The second is to come up with a cost-effective solution for the existing passenger over crowdedness problem on AALRT trains.

1.2.2 Specific Objectives:

The specific objectives of this thesis clearly stipulate the expected outcome of this work. The specific objectives of this thesis are listed out below.

I. Establishing a cost effective passenger counting system that can be used for managing passengers and which will also help future studies relating to passengers and their travel behavior of train service.
II. Solving the problems of manual ticketing by introducing Android-based digital ticketing device.
III. To enable any passenger waiting on station to have an information about the next train using station display, Android app or website.
IV. To establish a system that will put a limit on the carrying capacity of the trains so that the overcrowding problem on AALRT trains can be avoided.

1.3 Research Methodology

The method used in this thesis mostly involves PIS designing and coding using different programming languages. The first step in dealing with the stated problems was to refer research papers that were made on PIS and overcrowding. Based on the knowledge adopted from months of reading and researching, a cost effective PIS was designed which is unique in so many ways compared to the PIS being used in the rest of the world. Few of the pillars that the designed PIS rest upon includes; cost-effectiveness, easy installation, scalable and easy to

troubleshoot. To deal with the overcrowding problem a very simple system was designed based on an Android ticketing device. Once the overall system design was completed, all those designs were converted into a code i.e. Android app and website was developed and central controller that will do the lion share of the job was coded. And finally, the designed PIS was simulated to see if the promised features are met. The methodology is summarized by the flow chart shown in Figure 1.

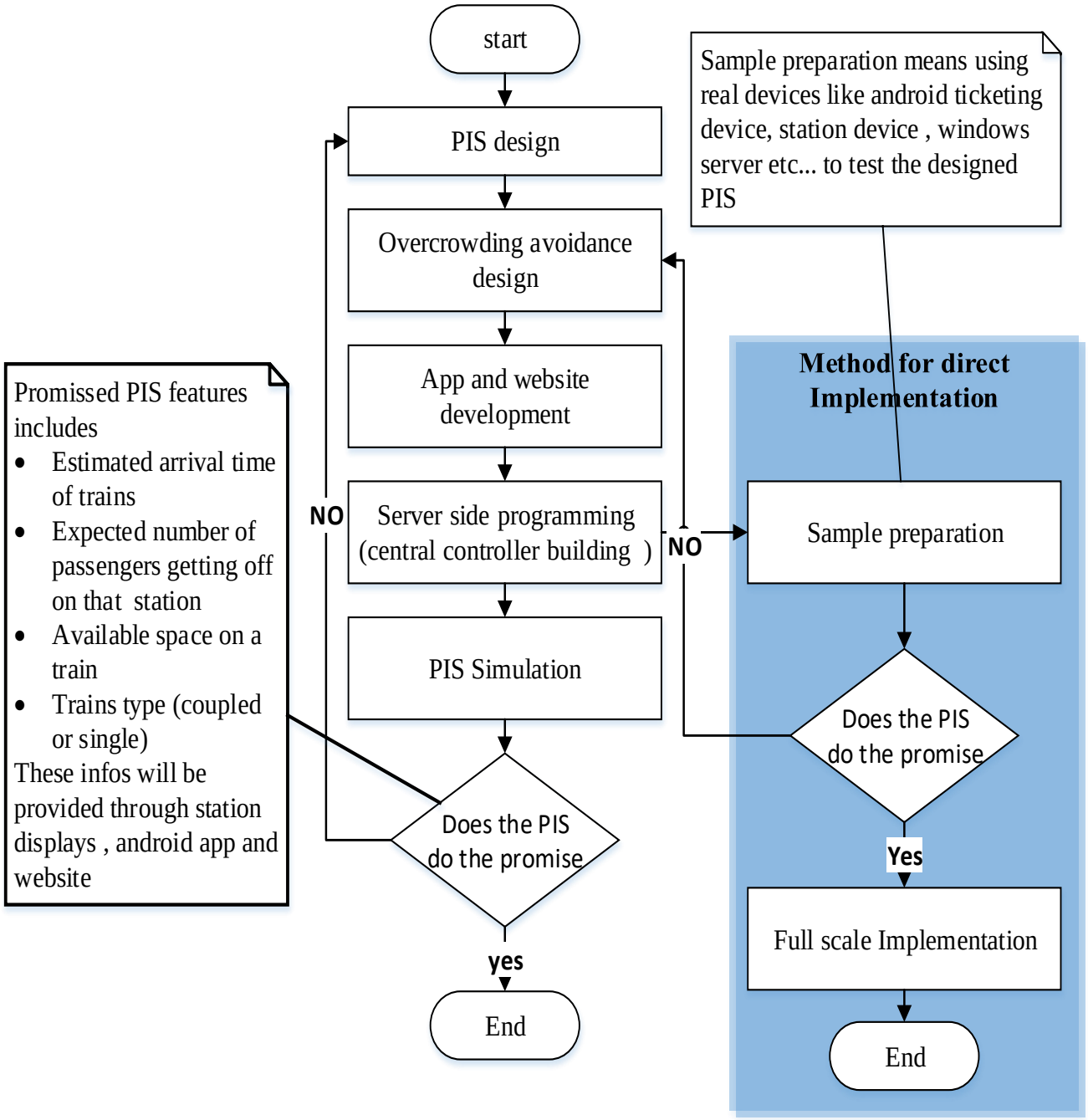


Figure 1: Flow chart of the methodology used in this thesis

1.4 Contributions

Of course, PIS on railway transportation is one of the most researched areas of study. So it would be like adding a spoon of water in the ocean if this thesis were meant to contribute

something on the already matured PIS on railway systems. Instead, this thesis contributed the concept and framework of tailoring PIS for other uses such as overcrowding avoidance. The philosophy behind this way of passenger overcrowding avoidance is that if we provide a crowding level information to the waiting passengers, to the ticket selling people, and to any user in need such as:

- Number of passenger waiting at the station
- Number of passenger on the train
- Available space on the train
- Expected number of passenger getting off at the next station

Then it is expected that overcrowding on the train can be greatly reduced. Thus this concept was changed into a system that was incorporated in PIS design. In a nut shell, this thesis has shown that PIS can also be used to deal with overcrowding.

1.5 Limitation

Due to the limited time, manpower and financial resources of a master thesis, this work was restricted to developing PIS systems for those who have access to internet connection and who can see. The designed PIS did not consider those passengers who are visually impaired. And some of the applications that were developed were developed based on majority users. For example, most of the smartphone used in Addis Ababa run on an Android operating system. Thus Android applications were developed for PIS use. This means Application for iPhone, Windows and other kinds of smartphones were not included in this work.

The other constraint in this work is that the proposed passenger information system does not give a 100% correct passenger count because it counts passenger by the number of ticket sale and there are some uncontrolled situations like illegal passengers who don't get a travel ticket to get the service. Thus these problems make the passenger counting system susceptible to wrong passenger counts.

1.6 Document Outline

This section describes how the thesis is organized. Short introductions for all chapters are presented as follows;

Chapter two presents the theoretical background of passenger information system and smart overcrowding avoidance. Passenger information system used in different countries are revised in this chapter. The methods employed for passenger counting system are also discussed. A short introduction to crowding on trains has been presented to help the reader understand what crowding means and how it is measured. Finally, various researches which have been consulted to straight out this work have been discussed in the literature review section of this chapter.

In chapter three the designed PIS with all the system components are presented. The designed PIS has many system components that can be broadly divided into two parts; Hardware system components and software system components. Each and every system component, which is either under the hardware or software system components, is listed and exhaustively discussed. One of the objectives of the designed PIS is to provide passengers with the estimated arrival time of the coming train. Thus in this chapter EAT calculation with the corresponding PHP code for implementation is devised. The strategy that is going to be used to avoid overcrowding has been clearly discussed. And USA's train crowd measurement technique, which was used to measure the crowding on AALRT trains quantitatively, is presented. Finally how the AALRT info app, the web map, and the central controller are programmed and what tools were used to develop them were explained under user app and web map development section of the chapter.

The designed PIS was simulated and the result of the simulation was presented in chapter four. Since real GPS devices were not used to track the real AALRT trains, virtual trains were needed to be created first. Thus how the virtual trains were generated is discussed in here. The result of the web map simulation is discussed with the help of different screen captures of the web map. The same discussion was made for the AALRT info App simulation result.

The final chapter closes the thesis with a conclusion and future recommendation. It explains that this work has shown an affordable, easy to implement PIS design for AALRT and all the concepts and frameworks associated with it are exhaustively discussed. And it is clearly stated that in the near future this work is planned to be proposed to ERC for direct implementation.

Chapter 2

Theoretical Background

2.1 Passenger Information System

A passenger information system or Passengers Inforamtion display systems (PIS or PIDS) is an electronic based information redering system which provides real-time train information to passengers or anybody who make use of this information. It may include both predictions about arrival, passengers count and departure times, as well as information about the nature and causes of disruptions or any other developing scenario. It may be used both physically within a transportation hub and remotely using a web browser or mobile device.

Passenger information, specifically information related to the transit service, takes many forms, including information needed before making a trip, during the trip and at the termination of a trip. The main sources of information need to be provided as follows [2]:

- **Through appropriate media that can be accessed before making a trip**. These help with trip planning and the ultimate decision to use transit.

- **At stations or other points prior to boarding a transit vehicle.** Relating primarily to vehicle arrival, these help instill rider confidence and comfort and can contribute to overall travel time competitiveness (e.g., a rider can do a quick errand if a vehicle arrival is several minutes away).
- **On the vehicle itself.** Next-stop information, traffic updates and service disruption alerts can also instill confidence and comfort by helping passengers reach destinations and transfer connections most efficiently.
- **At a termination point.** Destination and transfer information can help manage rider expectations during and after the trip.

Through ITS, the most valuable advance in quality transit service information has been real-time information, where pre-set and theoretical schedule times have been replaced by real time, which takes into account delays and other variations that affect the actual arrival and departure times. Real-time information is generated by algorithms that can accurately predict the time a

transit vehicle will arrive at a specific location thanks to data from technologies such as satellite-based GPS or roadside sensors. This is then combined with automatic signage that transmits this information to users of the service [2].

Many BRT services, and transit systems in general, are now providing real-time transit service information through a fully integrated ITS system that brings together many or all of the elements described in this section. Passenger information services that are part of these systems include the following [2]:

- **"Next-train" arrival displays at stations:** These displays are provided trough variable message signs (VMS), either LED or similar, programmable for a single route or multiple routes (flashing or scrolling mode), with sign units mounted into the shelter or station infrastructure.
- **"Next-stop" displays and automatic stop annunciators (ASA) on vehicles:** This is also provided through VMS, typically mounted behind the driver, used in conjunction with the onboard audio system using automated voice generation (important for those who are visually impaired). An automated audio system ensures accuracy and sound clarity and removes the need for drivers to announce next stops.
- **Online information sources:** This including schedule pages on transit Web sites and personal communication devices.
- **Interactive voice response (IVR) system:** This interactive telephone service uses prompts to provide schedules and other information.
- **Interactive automated trip planners:** These can be accessible via online services, IVR, mobile devices and kiosks at stations.

2.2 Automatic Passenger Counting Systems

APC systems are used to count passengers in public transport. There exist different APC technologies and they can be divided into two categories: static and on-board systems. The static systems are installed in stations and the on-board systems are embedded in vehicles. In technical terms the on-board APC system is a module installed inside a vehicle and a central computer in an office used for storage of the APC data. The on-board module consists of sensors and an on-board computer that converts and stores the information registered by the

sensors into passenger counts. An on-board APC system is shown in Figure 2 [3].

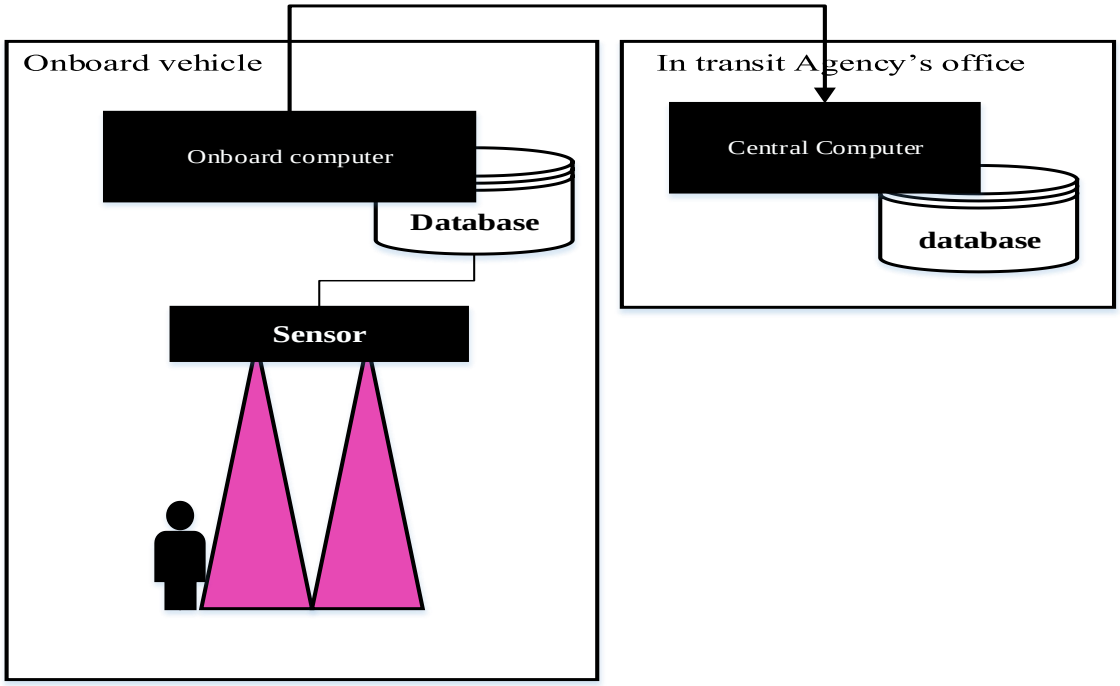


Figure 2: An onboard APC system components and how they are connected. [3]

The most common technique for counting in APC systems is infra-red (IR) sensing. Other techniques used are: pressure-sensitive mats, horizontal beams and cameras. The IR sensors are mounted above each door and are only active to register passengers boarding and alighting when the doors are open. The stored counts are transferred from the on-board computer to the central computer at a regular time for storage. Most analysis of APC data requires a moderate data sample, which can be met with APC systems installed on 10-15 % of a transit agency's total vehicle fleet [4].

2.1.1 Integration with an Automatic Vehicle Location System

Some of the usual ITS and technologies within public transportation are: Automatic Vehicle Location (AVL) systems, Geographical Information Systems (GIS), Passenger Information Systems (PIS), and Automated Fare Payment systems [5]. An APC system can be integrated with other ITS, which makes the data outcomes richer. The most common is to integrate an APC system with an AVL system [4].

An AVL system determines the vehicles position at a certain time by the use of either the Global Positioning System (GPS) or an odometer [6]. Depending on which level of intensity of position data that is needed, there are different techniques to collect data. The most common

is to get polling records or stop records. Polling records is when a central computer continuously requests a vehicle's position, which mainly is used for getting a high frequency of data points like real-time information [3].

Stop records provides the position of a vehicle and the time when it arrives at certain stops, this is the type of record that is used when an AVL system is integrated with an APC system [4]. An example of a data record from an APC system integrated with the time and stop location from an AVL system is shown in Table 1. The table shows the number of passengers boarding (Ins) and alighting (Outs) the vehicle during the time period from 7:06:47 until 7:36:59. This vehicle makes 10 stops during this time with a total of 48 passengers boarding and 48 alighting. The vehicle arrives from a previous trip at the terminus 7:06:47, starts a new trip 7:07:12 and arrives at the end terminus of that trip at 7:36:46. A new trip starts 7:36:59 [3].

Arr Time	Dep Time	Stop	Ins	Outs	Route Number
7:06:47	7:07:12	1	28	0	55
7:13:48	7:14:36	2	4	0	55
7:14:52	7:15:05	3	4	4	55
7:19:13	7:19:22	4	1	1	55
7:20:20	7:20:34	5	0	15	55
7:25:17	7:25:28	6	6	4	55
7:27:07	7:27:16	7	0	2	55
7:32:49	7:35:16	8	3	6	55
7:35:51	7:36:17	9	2	8	55
7:36:46	7:36:59	10	0	8	55
Total			48	48	

Table 1: A table showing an APC/AVL data sample from central computer [3]

2.3 Overcrowding on Trains

As the demand for rail travel continues to grow amongst the travelling public, along with fuel price increases, traffic congestion, and employment and population growth, overcrowding of rail services is fast becoming a pressing concern worldwide [6], including in Ethiopia.

The notion that passengers are being crammed into trains like cattle or cramped like sardines in a small tin on the trains is also not uncommon elsewhere . For this reason, train overcrowding is being recognized increasingly as a potential threat to physical health, personal safety, psychological wellbeing, and even the quality of life [7].

Research into the nature and effects of rail passenger crowding on commuters is relatively limited. At present, our knowledge is constructed not only from studies of rail travel but also from those involving other forms of transport. Most rail-related crowding research, however, focuses on the direct effects of crowding, such as psychological and physiological stress as well as commuting satisfaction, but with an emphasis on physical environmental variables, such as passenger density or train capacity, as the determinants of these outcomes [7].

2.3.1 Measures of Crowding: Passenger Train

Compared to bus, much more diverse crowding measures are defined in the passenger rail industry. For passenger rail, different specifications for measuring crowding are found across countries and even within a country, which are summarized below [8].

2.3.1.1 Rail Crowding Measures in the UK

The passengers in excess of capacity (PIXC) is a crowding measure that applies to all London and South East operators' weekday train services arriving at a London terminus during the morning peak from 07:00 to 09:59, and those departing during the afternoon peak from 16:00 to 18:59 The overall PIXC figure, derived by combining the PIXC of both peaks, considers the planned standard class capacity of each train service, as well as the actual number of standard class passengers on the service at the critical point (i.e., the location on a train's journey with highest passenger load). PIXC is the number of standard class passengers that surpass the planned capacity for the service. The PIXC is given in percentages, calculated as the difference between the number of actual passengers and the capacity of the train divided by the actual passenger number. It is zero if the number of passengers is within the capacity [8].

The PIXC can be converted into a common measure for crowding, i.e., the number of standing passengers per square meter (standing passengers per m^2). For example, a PIXC of 40 percent is equivalent to five standing passengers per m^2. The standing passenger density is used by many rail industries around the world. According to Office of Rail Regulation (2011), the benchmark for train crowding is 2.22 passengers per m^2 for most train operators in the UK [8].

2.3.1.2 Rail Crowding Measures in Australia

There are five major metropolitan rail systems in Australia: CityRail, Metro Trains, Trans Perth, Adelaide Metro, and Queensland Rail, located in Sydney, Melbourne, Perth, Adelaide and Brisbane, respectively, and each has its own measure of crowding. Therefore, the measures for rail crowding in Australia are not consistent. For example, Sydney's CityRail, operated by

Ralcorp, uses the number of standing passengers per square meter to measure crowding, and its benchmark is 1.9 standees per m². An alternative crowding measure for CityRail is load factor (passengers per seat), and the corresponding target set by the Minister in the Rail Services Contract6 is no more than 5 percent higher than 135 percent of seat capacity during the peak hours. Melbourne uses the rolling hour average loads to measure crowding in its Metro trains, and if the number of average passengers per train during a given hour, as counted at the Melbourne city cordon, exceeds 798, a railway line is considered overcrowded [8].

2.4 Literature Review

"Smart Bus Station-Passenger Information System": In this study, a smart bus stop-passenger information system was developed in order to enable administers effectively monitor the public transportation system and also enable the people who utilize this system simultaneously observe the information about the location and status of those vehicles. In the designed system, the embedded mini-computer based systems and digital monitors were used in order to instantly present the information related to the travel and transportation in the public transportation vehicles. The instant movement information of the vehicle was transferred to the central server through a GPS module which functions integrated to the embedded computer systems and web services. Moreover, the embedded mini-computer based systems and digital monitors were installed to the bus stops in order to present the information related to the movements of the public transportation vehicle and their approach to the related bus-stop. The mini-computers embedded on the bus stops provide communication with the central server through web services and the bus stops, public transportation vehicles and central server formed information network of the transportation. The software developed to manage the system provided the authorities the advantages of instant status observation, remote-informing and updating related to the management of the status and travel of the public transportation vehicles. Through this developed system, moreover, it was ensured that the position and travel information of the vehicles through the monitors both inside the public transportation vehicles and at the bus stops, increase the life qualities of the people who use the public transport vehicles and facilitate their urban life cycles [9].

"Smart Onboard Public Information System using GPS & GSM Integration for Public Transport": This paper describes challenges of urban transportation and one of the cost effective approach to intelligently manage the public transpiration in the city. The GPS based

urban transportation management system in which the fleet tracking using GPS & GSM/GPRS technology and public information system unit mounted at bus. The real-time coordinates help to guide the onboard commuter to destination. The unit mounted on bus sends the data using GSM/GPRS module to central monitoring system & displays it on City Map. This application is easy to deploy and provide effective management tool to urban transportation authorities for optimal utilization of present resources for improvements in term of load management, optimal route designing & real-time monitoring and control of the fleet [10].

"A System for Monitoring and Reporting Excessive Passengers in Public Buses Case Study Tanzania": The purpose of this work has been to develop a system for monitoring and reporting excessive passengers in public transport in Tanzania. It is based on Passive Infrared Sensor (PIR), Global Positioning System (GPS), and Global System for Mobile Communication (GSM) modem. The system is connected with SQL Server database which receives and stores all information. The system monitors the number of passengers boarding in public buses and reports to the respective authorities when a certain preset number is exceeded [11].

"Design of an Integrated Transit Monitoring System based on Radio Frequency Identification": The prototype described in this paper represents the integration of automatic passenger counting with automatic vehicle location to allow the automatic collection of transit system performance. The system involves the incorporation of the RF/ID tags into bus passes for the purpose of improving the method of data collection regarding transit users. Passengers carrying the tags can then be uniquely identified. In addition, RF/ID tags are placed at the bus stops to track the movement of buses along the route. This paper details the system hardware and software architecture, the functionality of the system, passenger and cost considerations, and the application of the system for the possible use in route control strategies [12].

"Real-Time People Counting system using Video Camera:" The aim of this thesis was to make a prototype of a real-time counting people system video based by using the Matlab-Simulink programming tool. And then try to measure its accuracy and compare to another system based on laser beam sensors. And finally, conclude in which situations this system is more reliable and therefore find its advantages and drawbacks [13].

2.4.2 Researches Made on Overcrowding

To the best of the author's knowledge there is not much research that has been made on overcrowding. One of the few notable works on overcrowding is revised below.

"Quality of rail passenger experience: the direct and spillover effects of crowding on individual well-being and organizational behavior." To the best of the author's knowledge this is the greatest research made on this area. This thesis describes a research work aimed at (1) investigating the relationships among the different psychological components of crowding and their effects on commuters' experience of stress and feelings of exhaustion, and (2) exploring how the effects of rail passenger crowding can spill over to the individual's broader work and life. To achieve these aims, an operational model is built that is consistent with the framework of Cox et al.'s (2006) model of crowding, stress, health, and safety, and is tested in a two-phase study. While Phase One of the research qualitatively explored the perceptions of rail passenger crowding and other associated issues among key stakeholder institutions Phase Two quantitatively examined the effects of rail passenger crowding on commuters' individual well-being and their organizational behaviors [8].

The results of Phase One demonstrate that passenger crowding is perceived only as a minor problem compared to capacity, infrastructure, and service quality issues among the key stakeholders. On the other hand, the results of Phase Two reveal that crowding is indeed stressful for the commuters and has the potential to spill over to other aspects of their life and work. Using structural equation modelling techniques, the results show first the relationships among passengers' evaluation of the psychosocial aspects of the crowded situation and of its ambient environment as well as their affective reactions to it, and the relationships among these psychological components of crowding and passenger density. Second, they demonstrate that the different psychological components of crowding together with rated passenger density are combinatorially predictive of commuters' stress and feelings of exhaustion. Third, while the effects of crowding on feelings of exhaustion disappeared after controlling for demographic factors and individual differences in commuting experience, its effects on the experience of stress remained significant, further highlighting the negative consequences of rail passenger crowding. Fourth, the results reveal different patterns of spillover effects for passenger stress, particularly on commuters' reports of somatic symptoms of ill health, their propensity for lateness and absenteeism at work, and intention to quit, but not in terms of their job or life

satisfaction. The implications of these findings are discussed in terms of the existing literature and the operational framework set out at the beginning of the research work, which could lend support for future crowding research and management [8].

Chapter 3

PIS Design

The urgent need for wayside PIS for AALRT is quite vivid, knowing the fact that there is a huge discrepancy between the stated arrival time of the trains and the actual time the trains take to arrive to a station. The stated arrival time is 10 minutes, which means that in every station a train will arrive within a 10 minutes of time. But the real situation shows that there are time that trains take more than 30 minutes to arrive to a station. That is why this thesis focus on dealing with this existing problem. The PIS is designed to provide a detailed information about the next train including the expected arrival time, number of expected passengers to get off at the station and the space available on the train for additional passengers to get on. And this information is expected to avoid the inconvenience created by a train delay as the detailed information about the next train provided by the PIS will help the passenger to decide whether to wait for the next train or look for other means of transportation.

The design of the PIS implemented for this thesis, takes on ideas from a wide cross section of the state of the art technologies and reassembles them in a way that is quite unique. The system is designed to take a real-time GPS location feed and from an Android ticketing device (Figure 3)**.** The PIS is specifically designed to AALRT and it will provide information which are not common on railway transportation. This uncommon information are; the available space on the coming train, number of passengers getting off on a station and the train type (single train or coupled one).The need for this additional information arises from the existing problems of AALRT which were stated in section 1.1**.** Information like the number of available space on the next train is found to be very important in AALRT because many passengers need this information. This type of uncommon information is not quite important for the passengers in well-established railway transportations like the case in USA, UK and Japan as they can support the demand for the rail transportation efficiently compared to AALRT. Even though there are no researches made on the demand for the AALRT, The daily experience on the AALRT transportation service clearly shows that the demand for AALRT and what it can actually support are quite wide apart. This unmated demand for the train service has to be comforted with important and unusual information like; expected number of passengers getting

of the station, available space on trains and the like. This information has to be passed out to the user so that the users can use them to schedule their transportation accordingly.

In the contemporary world railway transportation without PIS are almost hard to find. This is because PIS plays the important role by providing the necessary information for the passenger so that they can compare and decide their means of transportation.

The designed PIS can provide the necessary information to passengers waiting at the station with a station display that can be installed on each stations. The display will be showing the following,

1) On what platform the passengers are currently located
2) The estimated arrival time of the next train .
3) Emergency report (if there are delays made on the expected trains due to several reasons then this will be displayed for the passengers waiting for that train)
4) Information about the available space on the train
5) Expected number of peoples getting off at the station

A public website and Android application is also used to render this information so that passengers can check if any train is coming or not from any location in Addis. There are two ways to track the trains. The first is by using the existing SCADA train information. But this way of tracking trains will create a security risk. This is because SCADA system is employed in the train control center to control, track and monitor almost every operation related with train's movement. Thus an access to the SCADA system means a total control on every activity of the trains. Thus integrating public information rendering devices with SCADA system can leave a loophole for cyber-attack. Thus the PIS need to be isolated from the existing SCADA system. The second way is the use of GPS tracking devices on the trains this way we can get the real time location of the trains. Hence, the proposed PIS system uses GPS train tracking.

The other way of addressing the customer for information is through their Android smart phones using a custom developed AALRT info service app and for non-smart phones and other kind of smart phones users which runs on IOS, Windows etc.… with internet access can get this information through a dedicated AALRT website.

In addition to the above common information that are usually displayed for many of the railway transportation in all over the world, the AALRT display is designed to show the maximum number of peoples that are going to be taken by the coming train. This is to avoid passenger over crowdedness on a train which is being encountered by the AA-LRT on the daily basis. This is of course will be facilitated by station coordinators. The station coordinators will see the displayed number of available room of space for passengers and then passengers according to their arrival time will get on board until the allowed number of passenger on a train is reached. The train capacity (maximum passenger carrying capacity) is known to be 286 per single train. Thus this capacity will be used to avoid over crowdedness and maintain the number of passengers below that.

The methodology for this particular work mostly involves software development, interfacing and networking. For dealing with the PIS problems, which were explained in the problem statement section of the proposal, an assistive wayside PIS is developed. And to circumvent the over crowdedness problem passenger counting and managing system is designed.

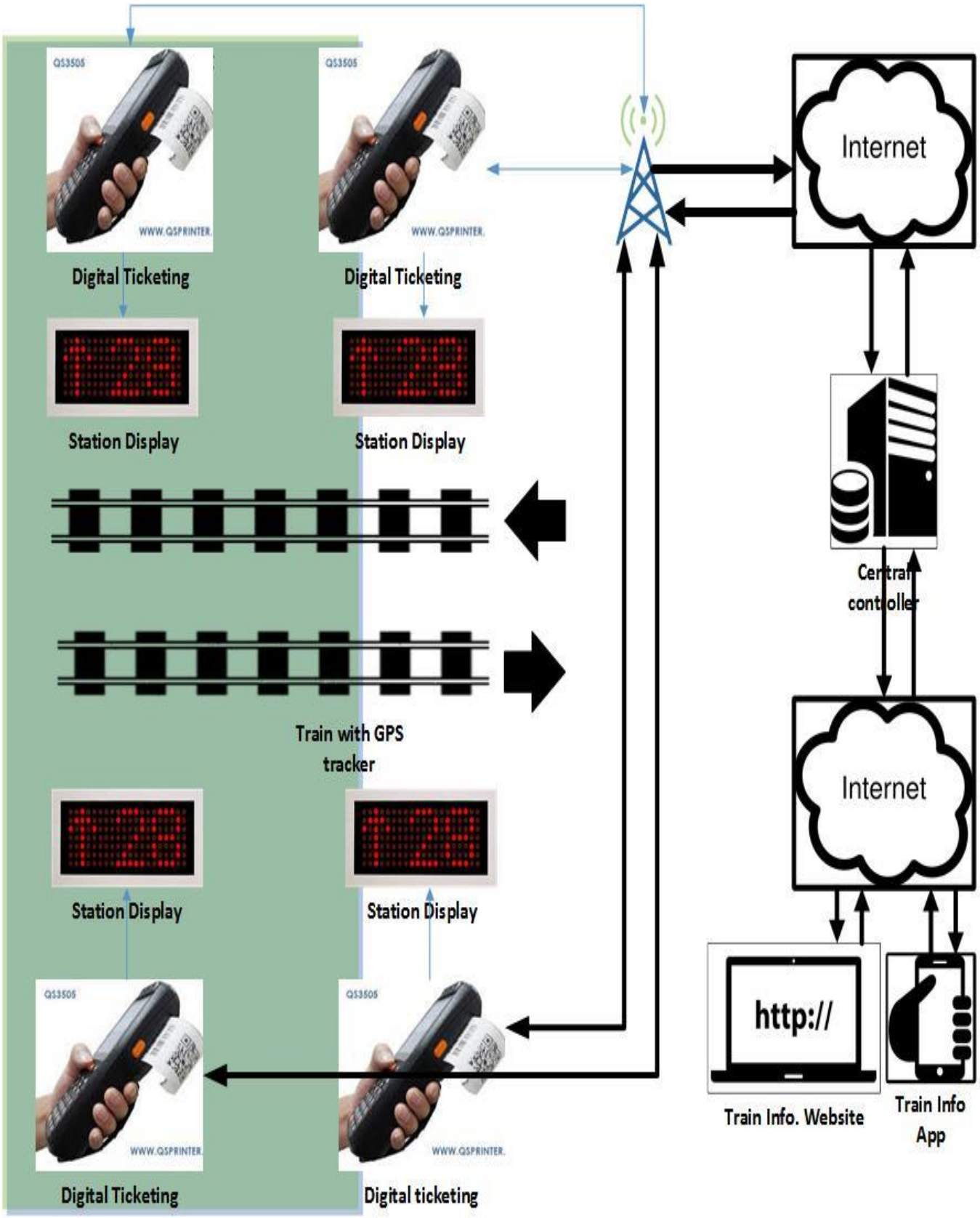


Figure 3: AALRT PIS schematic view

3.1 Hardware PIS System Components:

The passenger information system implemented for this thesis – which from now on will be called the AALRT Info Service - consists of four main components:

I) **The Ticketing Device**: Currently AALRT train ticket is being sold manually. And it is proposed to change it with an Android based POS machine that will print tickets up on sale. This device will be networked with the central controller so that every time a ticket is sold using this device an information about this ticket will be sent to the central controller. Some of the information that will be sent includes the origin of the ticket i.e. the station from which the tickets is sold, the destination of the ticket and the train ID on which this ticket will be used. And the central controller (which is basically a server) will process all the information sent from all the ticketing shop and it will generate the necessary information and it will send it back to the ticketing device and also it will be made available to the user.

Unlike the manual tickets which are already printed for a single day use and which lacks timestamps, expiry date, and train IDs this ticketing device will bear time stamps and train IDs (Figure 4). An interview with the ticket checking peoples of the company revealed that currently, ERC is losing a great deal of money due to fraud committed on a daily basis. The inherent problem on the tickets being sold is that it doesn't have an expiry time printed on it. Thus it is exposed to fraud use the whole day. But the proposed ticket will have estimated expiry time printed on it that it is impossible to be used more than once.

Figure 4: The proposed AALRT train Ticket

The ticketing device will also be used as a communication channel between the station displays and the central controller. Thus the station displays will be feed with real time data by the ticketing device that will be sent from the central controller. AALRT have 39 stations on both the North-South and East-West corridor. Thus there will be an overall of 39x2 station ticketing shop. Assuming two ticketing POS device per a single ticking shop we will have a total of 39x4 device that will send and receive information to/from the central controller.

II) Central Controller: The heart of the designed PIS is the central controller which is responsible for many of the important calculations, user login credentials, clients' information, agent information and tracking data. It also enforces data integrity by making sure that data is gathered and presented using a consistent format. For the system to be usable, it must retrieve data efficiently. The need for efficiency has led to use of complex data structure to represent data in the database. Some of the jobs that is going to be handled by the central controller are

1) Calculating the remaining time for arrival of the trains to a certain station using information acquired from GPS on the trains.
2) Determining the number of passengers on any train and also calculating the expected number of passengers getting off the next station using information acquired from the ticket sale.
3) Sending contents through nearby ticketing devices to be displayed by station displays. Thus the central controller indirectly controls the content of the station displays.
4) Serving Android application users with real-time information of a train of interest
5) Serving website users with real-time information of trains by feeding data to the dedicated PIS public website.

The database architecture consists of the following layers:

1) Presentation layer: This is the topmost level of application. The presentation layer displays information related services. The presentation layer- communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network.

2) Business Logic Layer, Data Access Layer (or middle layer): The logical layer is pulled out from the presentation layer and, as its own layer; it controls an application's functionality by performing detailed processing. Another in-between layer added to make benefit of the reusable set of functions performing database operations, this is the DB Worker Layer.
3) Data layer: This layer consists of database servers. Here the information is stored and retrieved. This keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance.

III) Tracking Service: AALRT have around 41 operational trains and to track each train's location the PIS system will use GPS tracking device which will be installed on the trains. The GPS device will use a 3G SIM card to communicate with the central controller. Getting it from satellites the device will send the real-time location of the train to the central controller every pre-specified duration of time. If it is required the device not only provides the latitude and longitude values of the train but it can also tell the altitude and speed of the train. Thus this values can be used by the central controller to estimate the remaining time for arrival of each train to the next station.

IV) Station Displays: any passenger waiting on the train stations will be provide with details of the train coming to the station on station display. The station displays are networked with the central controller via the station ticketing device.

Thus the central controller updates the stations displays to display the latest information about the coming train. Any developing scenarios, such as emergency, will also be displayed on this display. As shown in Figure 5, estimated arrival time, train type, the crowdedness level are some of the information that will be displayed on a station display.

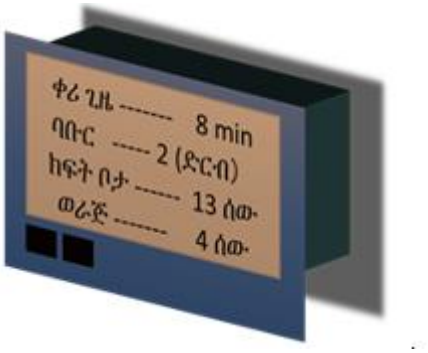


Figure 5: Station display

3.2 Software System Componets

3.2.1 Android App

There are many smart phone users in Addis. And most of the smart phones run on Android operating system. Thus by developing an Android application that will give the whereabouts of AALRT trains we can reach many smart phone users. Thus Android Studio was used to develop the Android application shown in Figure 6. And it has the following features

1) The app will show an offline Addis Ababa map with all AALRT waiting stations depicted with markers
2) One can see trains of interest on the map and touch on the train icon to see the details about that train (like the estimated time of arrival)
3) The current location of the user will also be shown in the app.

The Android app development is effectively completed. The Tile View java library was used to develop the app. The app have two option; one for text search and the other for map view. Open street map's map tiles were used for making the offline map (Figure 6). Unlike other maps this map don't need internet connection to download maps because it uses an inbuilt tiles of maps at various zoom levels (from 11 to 16 zoom level). This makes the operational cost of the app to be very small which means it don't use much data for rendering the necessary train information. This will make it ideal for the Addis Ababa user.

The target user for the app are primarily those who can read Amharic. But in addition to that English languages also added on the app as supplement for other non-Amharic users.

- How does the app work? The application need internet connection and an access to user current location for getting the real time location information of the trains to be displayed on the map.
- Which Android versions are supported? API level 16 and above are supported (>4.0 Android version)
- What is the size of the app? It will take 30 MB of memory space upon installation
- Does it take much data for operation? Because it uses an offline map data, it will not consume much data for displaying trains information

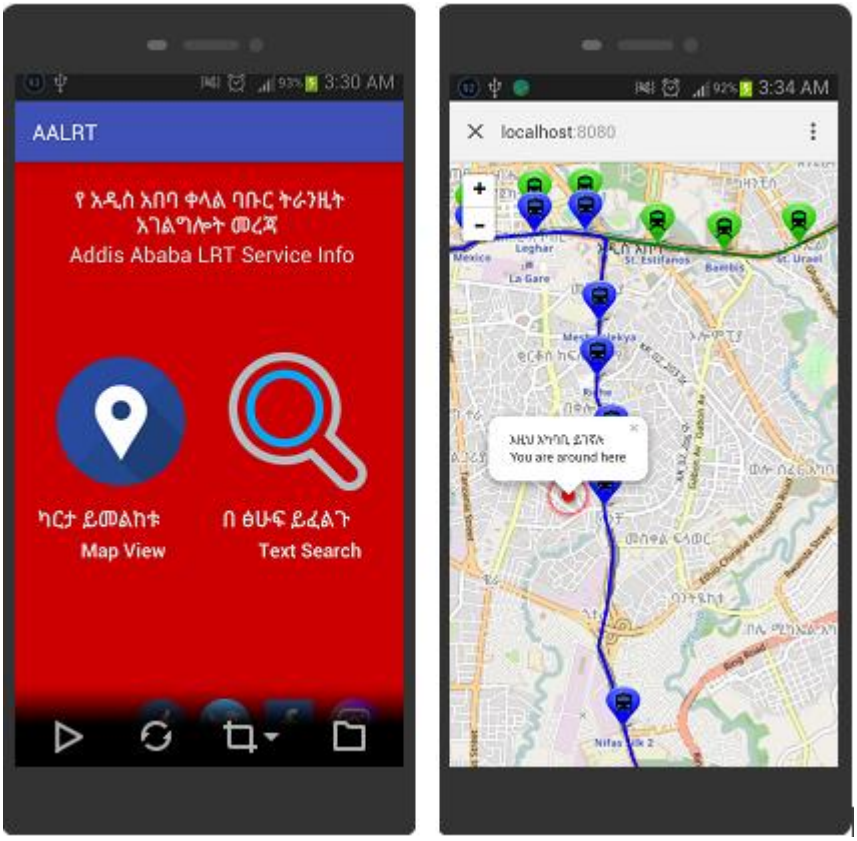


Figure 6: Android PI app screen shot

3.2.2 Website Development

Using Leaflet JS a web map with six zoom levels (zoom level 11 to 16) were developed as shown in the Figure 7 below. The map is mad of small tile images, unlike other maps which uses vector tiles instead of rasterized image tiles. The web map only shows the Addis Ababa city only to the places where the AALRT railway covers. There are 44 icons on the map showing all stations residing along either side of the railroads in the East-West and North-South direction. All icons are responsive and upon touching one can read the station names in both Amharic and English. The web map can also show the current location of the user on the map with a pulsing red circular object but the user need to let the web file to access the location information from the device first. The green icon shows the station located in East-West corridor. The blue icon shows the station in South-North corridor. The blue-green icon shows the station where both the East–West, and North-South railways commonly share.

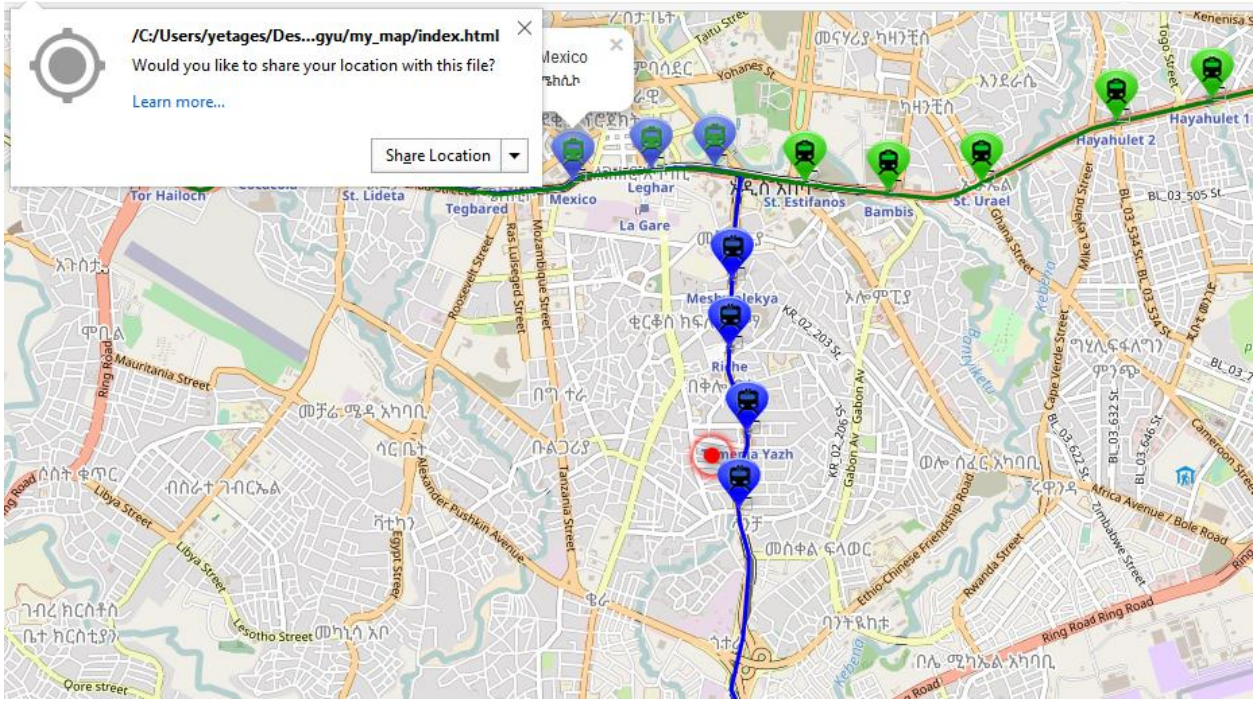


Figure 7: Screen shot of the developed Leaflet JS web map

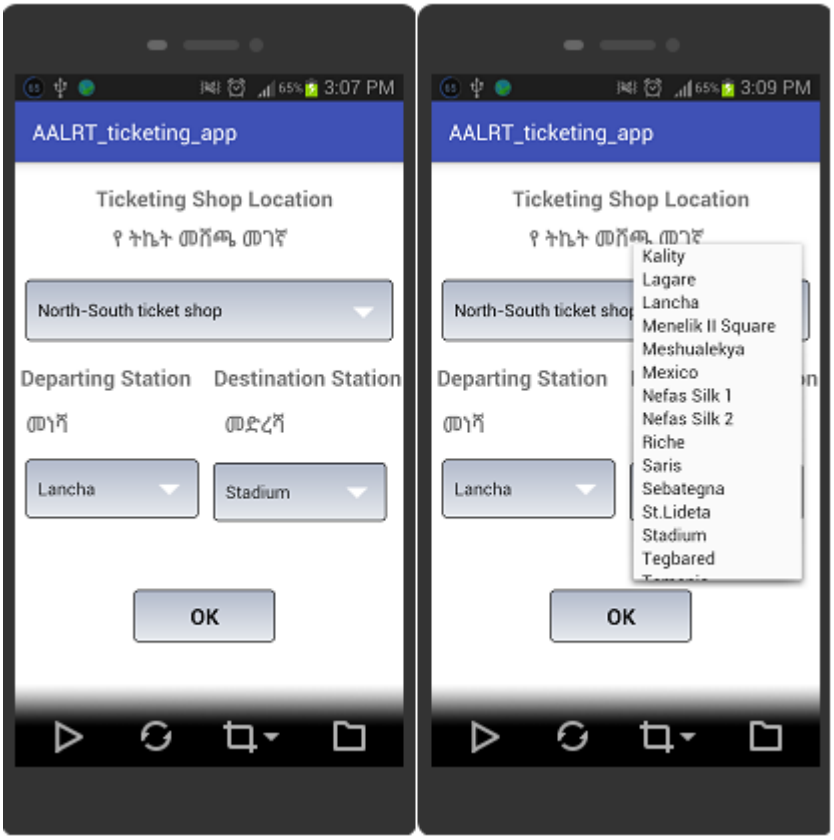


Figure 8: Android ticketing app

For non-Android phone type users, provided that their phone have an internet access, a website that can support mobile devices is developed so that they can go to this website and get the necessary information about the coming train. The JavaScript code for building the web page is already completed as it was explained above. Thus the second step in completing a fully functional web page is to acquiring a domain name. For the time being the website will be hosted on a private computer using XAMPP server.

3.2.3 Android Ticketing App

The proposed ticketing device runs on Android operating system thus an Android app that will do the job of ticketing and communication with central controller over 3G internet connection will be used to facilitate passenger counting and ticketing. The Android app, Figure 8, will also be used as a channel to control the content of the station displays by the central controller.

3.3 Estimated Time of Arrival Calculation

The estimated time of arrival of trains (ETA) can be calculated if we know three values. These values are:

1) Distance between a station for which the ETA is calculated and the current real-time location of the next train
2) The speed of the train
3) The dwell time of the train on stations before it arrives to a station

The following PHP code calculates the distance between two geo-coordinates on a map. It takes latitude and longitude (in degrees) as an input variables and returns distance between the two points in meters.

```
function distance (lat1, lng1, lat2, lng2) {

     var radlat1 = Math.PI * lat1/180

     var radlat2 = Math.PI * lat2/180

     var theta = lng1-lng2

     var radtheta = Math.PI * theta/180

     var dist = Math.sin(radlat1) * Math.sin(radlat2) +
Math.cos(radlat1) * Math.cos(radlat2) * Math.cos(radtheta);
```

```
dist = Math.acos(dist)

dist = dist * 180/Math.PI

dist = dist * 60 * 1.1515*1000;

return dist }
```

Once the distance is calculated to determine the time, the following simple physics law can be used; Time = (distance / speed (in m/s)). But how one can get the speed of the train. The answer is very simple, from GPS device. Many GPS tracking devices not only provide latitude and longitudes they can also provide speed of the train using an in built software.

The other question is how one can determine the dwell time of AALRT trains. Dwell time of a particular train can be estimated from a research. For example, one can record the dwell time of a train in a particular station for many days and take the average dwell time of the train on that station and this average can be made the dwell time of a train for that station.

Thus the ETA (in seconds) is calculated as

$$\text{ETA} = \frac{\text{distance}}{speed} + \sum_0^n DT_n \tag{3.1}$$

Where: DT_n represents dwell time at station n

3.4 Passenger Counting Using Android Ticketing Device

Passenger counting system is important for marketing research (passenger crowding management, passenger flow estimation) or in security application (in the case of an evacuation, it is essential to know how many people are inside the train at any given time) [13] .The primary purpose of the passenger counting system in this thesis is for determining the number of passengers on a train so that based on this knowledge passenger over crowdedness can be easily managed.

A key benefit associated with an APC system is the ability to provide a continuous record of ridership onboard a transit route, and the ability of quickly summarizing the data. Conducting ridership surveys historically has been labor extensive and has provided only a snapshot of ridership conditions to transit agencies, given that only a sampling of bus routes generally can be surveyed [2].

APC units can be tied into an overall vehicle ITS monitoring system, including integration with the AVL and TSP systems. If integrated with TSP, conditional priority to trains is given based on a minimum number of passengers onboard a train. In some cases, APC has been implemented as a standalone system with its own GPS separate from the AVL system, either because the APC preceded the AVL or because the APC was deployed after the AVL but with a different vendor. These situations have created incomplete and mismatched data and higher maintenance costs, with added post-data-collection processing needed to match the APC with AVL data [2]. But in this thesis a different approach has been employed for passenger counting. The proposed counting system indirectly counts passengers using sold tickets information provided by station ticketing devices.

There are different was of passenger counting methods, for example, some sensors are used to count people in train station with different advantages and drawbacks and different accuracy such as laser beam, infra-red sensor, thermal sensor and recently video camera. But in this thesis an indirect way of passenger counting using ticketing device is employed.

As it was explained in section 3.1 every ticketing shop will be equipped with the Android ticketing device. The ticketing device works like a POS machine and have a thermal printer built on it which is used to print tickets up on sale. An Android application has been developed to assist the ticket sale. The following information will be printed on a ticket when it is sold

1) The station from which the ticket is sold
2) The destination of the ticket
3) The time of sale and predicted expiry time
4) The train ID on which the ticket will be applied (It is very important that the ticket has to be used only on the pre specified train or else the passenger has to have it changed).

Every ticket that is sold using ATD will be associated with only a single train. It cannot be used in other train because the tickets are used to count passengers getting on a train. The ticket will also be used to determine the expected number of passengers that will get off from the train.

When a ticket is sold from a station an information about that ticket will be sent to the central controller. The central controller will accept all the ticket information that has been sent to it from all ATD. The first thing it will do after having this information is to identify the source of the tickets i.e. the station from which the tickets were sold and this information will be used

to update the number of passengers waiting in the corresponding station. The ticket destination will be used to update the number of expected passengers that will get off in each station. As shown in the flow chart Figure 9, the system will continuously track the location of the train and when a train reaches a station, passengers will get off and waiting passengers, holding a ticket applicable to the current train, will get on the train. Thus the system automatically update the number of passengers on the train as soon as the train departs from that station using the relation below.

$$CPN = PNP - LGFNP + LGONP \quad (3.2)$$

$$LGONP = NTS - X_{didnt\_get\_on} \quad (3.3)$$

$$\text{Thus,} \quad CPN = PNP - LGFNP + NTS - X_{didnt\_get\_on} \quad (3.4)$$

Where CPN → Current number of passenger on the train
PNP→ Previous number of passengers on the train
LGFNP→ Last got off number of passengers
LGONP→ Last got on number of passengers
NTS → Number of tickets sold on a station
$X_{didnt\_get\_on}$ → Number of passengers who bought a ticket but didn't get on a train

Equation 3.4 is used by the central controller to calculate the number of passengers every time the train passes a station. To determine the number of passengers who bought a ticket but didn't get on a train is very challenging. But it expected that this value is near to zero, this is because from the author experience many passengers of AALRT cannot afford to buy a ticket and not use it unless the train service is stopped. Specially, after the introduction this designed PIS to AALRT, the value of $X_{didnt\_get\_on}$ is expected to go to zero. This is so because one reason for not using the tickets bought is the lack of information about the coming train. But if the detailed information about the coming train is provided to them before they buy a ticket then this will help them to decide to buy the ticket or look for other means of transportation. Thus the centeral controller assumes that the value of $X_{didnt\_get\_on}$ is zero.

The need for CPN calculation in the fist place is for avoiding over crowdedness on AALRT trains. The vary in the value of $X_{didnt\_get\_on}$ doesn't exacerbate the over crowdedness problem

on a train in anyway. Infact, the increase in this value will creat a less crowding environment on the train. Every time the AALRT train passes a station the central will use Equation 3.4 to calculate the CPN and then the LGONP will be set to zero.

As it was described in the previous sections the proposed ATD is a hybrid of Android smart phone and POS machine, Figure 10, that it can be interfaced with local area network or the internet via 3G Data SIM which can be inserted on the ATD. This makes it possible to wirelessly connect the central controller and all the ticketing devices through a fast 3G data network.

There are several advantages associated with this way of passenger counting system. For example the manual way of ticket selling have an inherent problems like the tickets which are already printed for a single day use, lacks time stamps, expiry date and train IDs but the proposed ATD will print out a ticket which bears time stamps and train IDs. An interview with the ticket checking peoples of the company revealed that currently ERC is losing a great deal of money due to fraud committed on a daily bases.. But the proposed ticket will have expiry time printed on it that it will make it impossible to be used more than once.

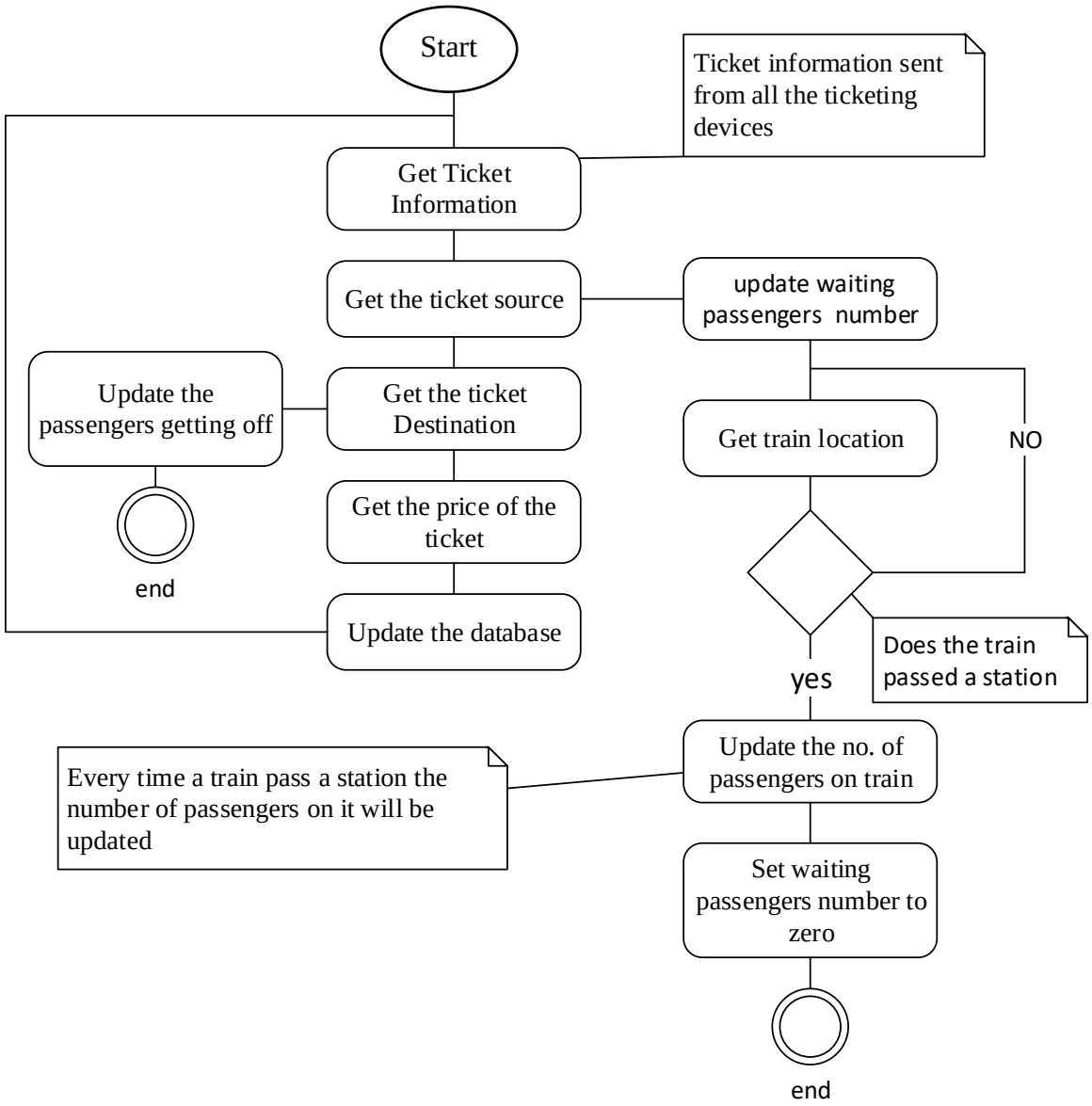


Figure 9: Flow chart for passenger counting system

The use of ATD for passenger counting has limitations for the reason that it is not directly used to count passengers. The system assumes that if a ticket is sold at a station then the passenger holding that ticket will get on the train with the train id that is stated on the ticket. In this way the system will count the passengers. But the system cannot exactly tell that the passenger holding that ticket will get on the train. This could be one source of error in passenger counting. The other is that many teenagers were seen getting on a train without holding a ticket. This could be the other expected source of error in the train count.

Figure 10: Typical Android Ticketing device

3.5 Over Crowdedness Avoidance

Crowding in public transport is becoming a growing concern as demand grows at a rate that is outstripping available capacity. To capture the user benefits associated with reduced crowding from improved public transport, it is necessary to identify the relevant dimensions of crowding that are meaningful measures of what crowding means to travelers [8]**.** But the overcrowding on AALRT trains does not seem to need a meaningful measurement because many people agree that it is a high, air tight overcrowding. And to circumvent this problem it is suggested that if some information about the crowding is passed out to the passenger prior to ticket selling, then some of the passengers may look for other means of transportation depending on the information gathered about the coming train. Because if AALRT cannot avoid the overcrowding, it at least have to provide an information about the crowding level of the coming train. In this thesis to avoid, if not possible, or reduce the overcrowding on the trains, sticking to the carrying capacity of the trains is suggested. The question is how we do that? How we make sure that the carrying capacity is not exceeded. And how can we calculate the carrying capacity of the train?

A key measure used by many U.S. transit authorities to evaluate in-vehicle crowding is load factor (passengers per seat), which is calculated as the number of passengers divided by the number of seats. A load factor of 1.0 indicates that all seats are occupied. With regard to load factor, different benchmarks are defined according to the nature of the service—for example, 1.0 for long-distance commute trips and high-speed mixed-traffic operations, 2.0 for inner-city rail service, and in between for other services, according to the current Transit Capacity and

Quality of Service Manual**.** This paper defined the thresholds for the level of service (LOS) with respect to in-transit crowding, shown in Table 2 [8].

LOS	Load Factor	Standing Passenger Area		Comments
	(passengers/seat)	(ft^2/passenger)	(m^2/passenger)	
A	0.0–0.50	>10.8^	>1.0^	No passenger need sit next to another
B	0.51–0.75	8.2–10.8^	0.76–1.0^	Passengers can choose where to sit
C	0.76–1.0	5.5–8.1^	0.51–0.75^	All passengers can sit
D	1.01–1.25*	3.9–5.4	0.36–0.50	Comfortable standee load for design
E	1.26–1.50*	2.2–3.8	0.20–0.35	Maximum schedule load
F	>1.50*	<2.2	<0.20	Crush load

*Approximate value for comparison, for vehicles designed to have most passengers seated. LOS is based on area. ^ Used for vehicles designed to have most passengers standing.

Table 2: LOS Thresholds for Crowding [8]

In addition to load factor, another crowding measure used in the U.S. is standing passenger area (space [m^2] per standing passenger), which can be easily converted into the number of standing passengers per square meter (standing passengers per m^2) (see Table 2). As an example, for the crowding level of maximum schedule load (which is the defined crowding threshold), the load factor range is 1.26–1.50, while the corresponding measure of standing passenger area is 0.20–0.35 square meter per standing passenger (or 2.86–5 standing passengers per m^2) [8].

3.5.1 Passenger Standing Area Calculation

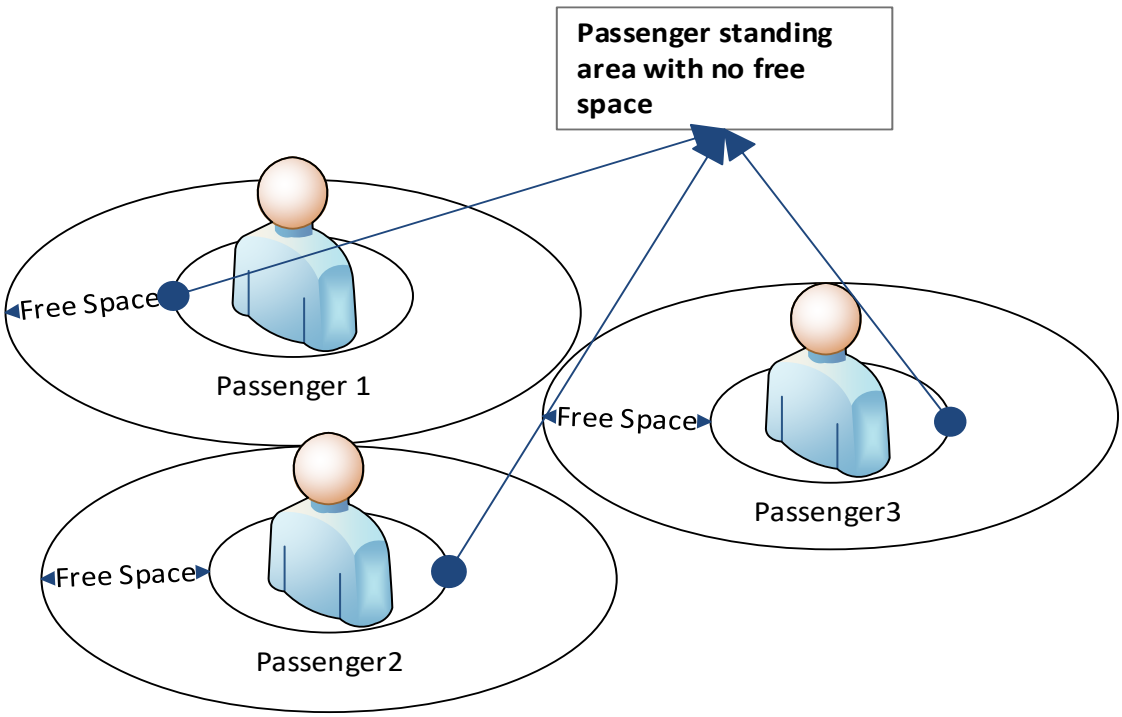


Figure 11: Passenger standing area representation in a public transport

The area that a passenger take in any public transport is represented on Figure 11. The total area taken by the passenger (TA) is the sum total of the free space around him (FS) and the area that the passenger actually occupied (PAO). To calculate the actual area occupied by a passenger we will take the top view of the passenger and approximate it with a circular area. For example, the wider area from the top view can be easily shown that it is around the chest including arms. Thus we can take an arm inclusive chest measurement and approximate it with a circle to approximate the actual area occupied by passenger.

$$TA = PAO + FS \tag{3.5}$$

A small survey was made on AALRT to find out the average area take by a single passenger on the rush hours. The survey was made on passengers getting off on Lancha, Gojam Bernda and Riche stations. Photos like the one shown in Figure 12 were taken on passengers getting on a train. Using a photo tool the average circumference of the standing area of a single passenger was found to be 38 inch and this circumference were approximated to a circular area (so that the maximum possible area can be found). Thus it was easy to calculate the corresponding actual standing area of the passenger to be 0.074 m^2 using simple mathematics

formulas. Note: on the rush hours FS is zero, which means no free space as can be seen from the photo.

The simple calculation reveals that on the rush hours the crowding level on AALRT train is around 0.074 m^2 per single passenger which is way below the crush load of 0.2 m^2 per single standing passenger stated on the table. Thus this simple crowding level calculation quantitatively describes the severity of the crowding level on AALRT train.

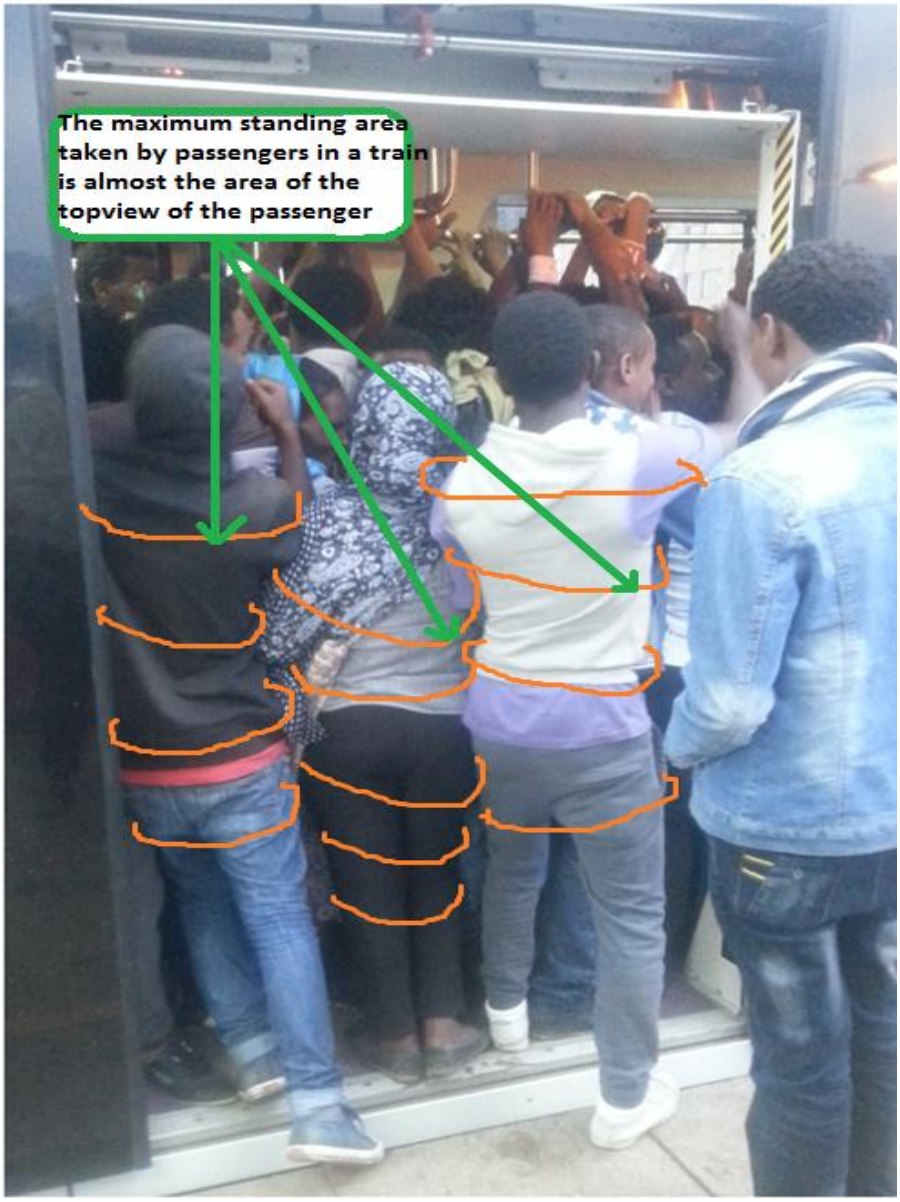


Figure 12: A photo showing how passengers standing area is calculated [photo was taken on Riche station, June 2017]

3.5.2 Strategy for Overcrowding Avoidance

The carrying capacity of a single AALRT train were specified to be 286. Using the passenger counting system we can put a limit to the number of tickets that is being sold to passengers based on the information acquired from the central controller. For example, let say a single train is coming to a station and it is carrying 260 passengers from which 5 of them are expected to get off in the immediate station. This means that there is a 31 people of space available on the next train thus tickets can be sold for a maximum of 31 passengers. Even though this approche was criticized, saying that avoiding crowding by restricting passengers is no solution, the best overcrowding solution that ERC can get with a cheap cost is the one that was stated in this work. Of course, this is not a permanent solution rather it is a temporary solution that will employ ticket sale restriction to deal with the existing overcrowding problem. Thus until a dependable and permanent solution is introduced, this low cost solution could be used to avoid passenger overcrowding on the trains.

The advantage of the system is that all passenger will have equal access to the train, irrespective of the station from where the passenger will get on board. Which means the system is first come first served. Let say AALRT train is coming from Kality heading to Piyazza. And let us assume that there is only one ticket available and there are no pass*engers getting off on any nearby station. If someone, let say located around stadium, come early to stadium ticketing shop to get a ticket than any passenger who went late to Lancha, Saris or Abo ticketing shops. Then the ticket will be sold to the passenger located in stadium all because he comes first. But currently the situation is not the same, passengers who are located near to the end stations have a great advantage of getting a place on a train (even a sit) than users located in the midway stations.

Lately, ERC come up with a solution for the locational advantage taken by peoples located in the end stations. On the rush hours, few of the coupled trains will deliberately pass the nearby station and start to take passengers from midway stations or so. But this solution is greatly creating a delay on the transportation service and also inconvenience on the passengers. A typical scenario is explained here.

On Wednesday, May 1, 2017, around 17 Hrs.(Figure 14 & Figure 16) in the afternoon on Gojam Berenda station more than 4 coupled trains were seen to pass this station to their destination (Menlik II station) but it took more than an hour for them to head back to Kality.

After an hour of frustration, two of the coupled trains passed the Gojam Bernda station carrying no passenger. After 10 minutes the third coupled train come and stopped to Gojam Bernda station. People then start to fight to get on board the train. The station coordinator said that the delay is deliberately introduced to take passengers at once. But passengers on this station reacted in vigorous way that some of them demanded to get their money back some other left the station offended by the delay and the fight to get on board. Thus this particular incident explains why this system does not work and need some better system. The Android based ticketing system will effectively provide a solution for the location advantage taken by the stations which close to the end stations.

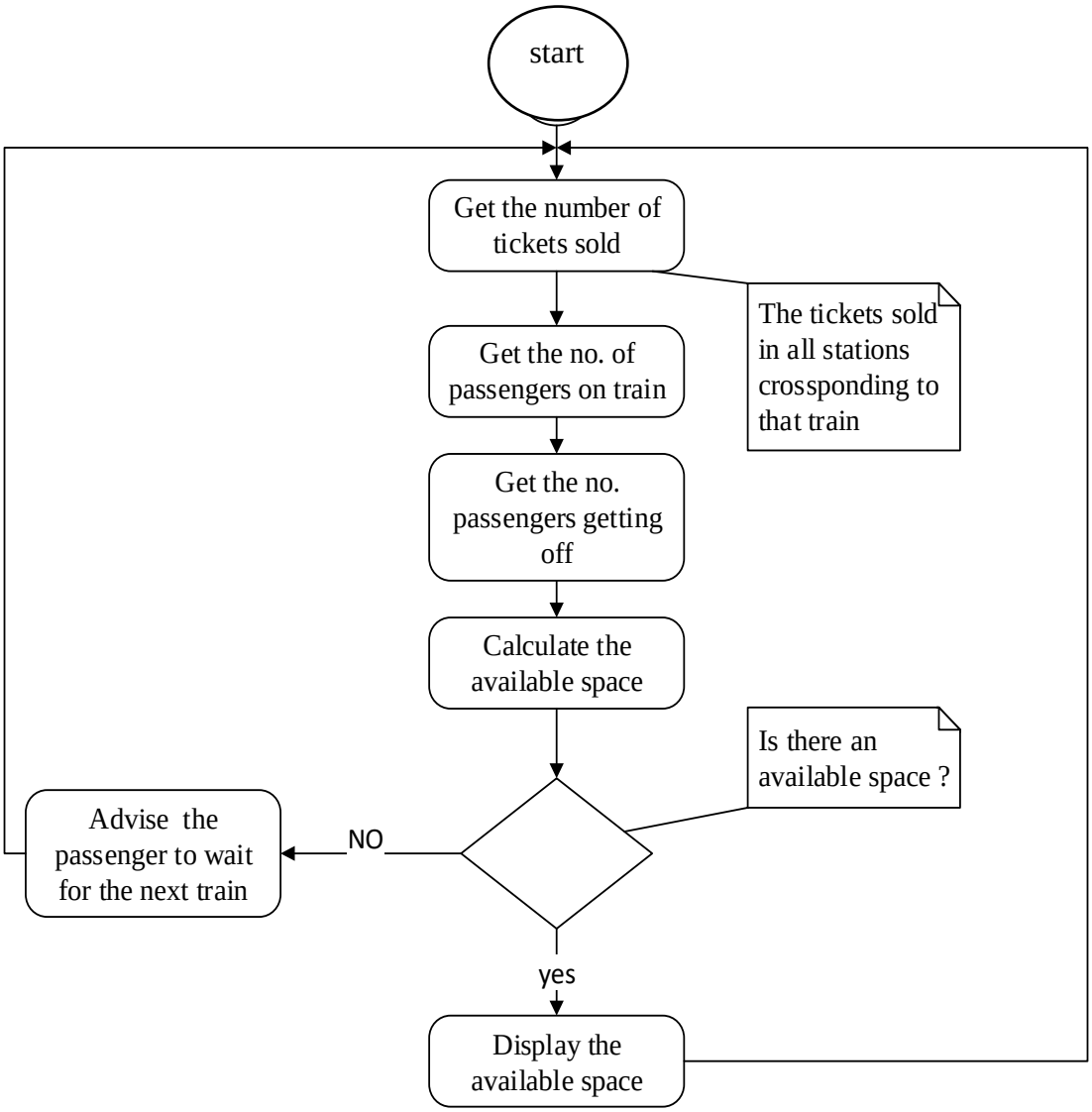


Figure 13: Simplified overcrowding avoidance flow chart

As shown in the flow chart, Figure 13, the system starts by getting the number of tickets sold from all the station ahead of the train of interest. And then the number of passengers on that train and the expected number of passengers that will get off on that station together with the number of already sold tickets will be used to calculate the available space using the relation 3.6. The formula finds out the available space in terms of passengers. And the reader has to note that the number of tickets sold would mean the number of passengers who are waiting on that station.

$$AS = 286 - (LGFNP + CNP + NTS - X_{didnt\_get\_on}) \quad (3.6)$$

Where CNP → Current number of passenger on train

AS→ Available space

LGFNP→ last got off number of passengers

NTS→ Number of ticket sold

NB: $X_{didnt\_get\_on}$ = zero (see section 3.4)

3.5.3 Passengers Management on Station

Currently how the passengers wait trains on station is depicted on Figure 14. The passengers are standing erratically some of them sitting, some of them standing, and some other wandering here and there. But when the train arrives the station, they all run to the train doors, with no hesitation they will push hard until they get on the train. This is becoming a common trend on train. Many people are getting tired of using trains because of the lots of inconvenience they have to face by the transportation service.

Tickets sold using Android ticketing device bear time of sale. So it won't be difficult to make a queue of passengers holding the tickets. If passengers can get on the train keeping their queue then the unnecessary fighting for getting on the train will no longer be a problem. Figure 15 shows a schematic diagram of queues made by passengers holding a ticket. Passengers who came early will get on the front and passenger who come last will be in the last row of the queue. The queue is made for all train doors which are on the side of the station. A queue made on a single train door will have 2 to 3 columns and varying rows. Of course, there have to be station coordinators who will do this coordinating job.

Figure 14: Wednesday, May 31, 2017 Photo taken from Gojam Berenda station: passengers waiting for a train standing erratically

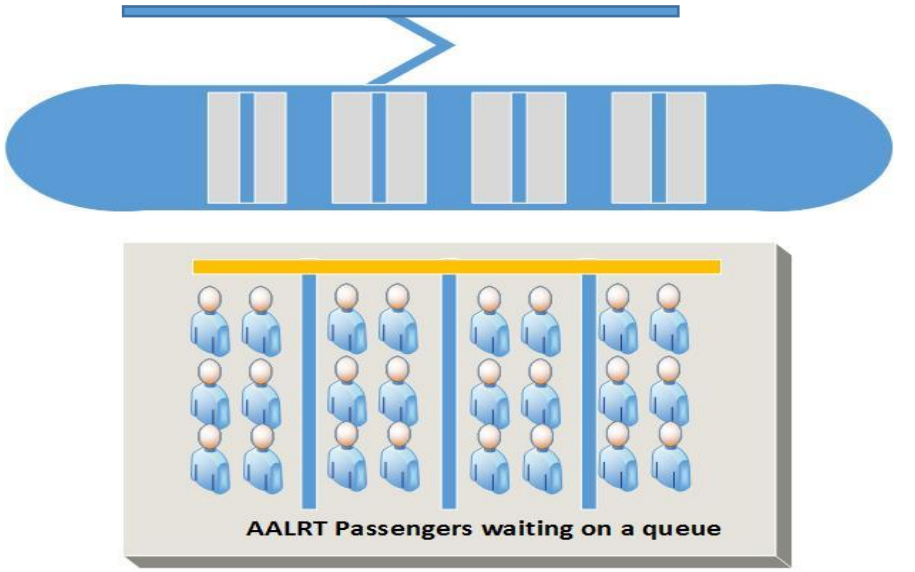


*Figure 15**:** Passengers waiting strategy*

Figure 16: Wednesday, May 31, 2017, Photo taken from Gojam Berenda station: Passengers fighting to get on a train

3.6 User App and Web Map Development

Of the many PIS components used to transfer information to the passenger, Android apps and websites will come next to the station displays. Android apps are very handy when it comes to information rendering and they can be used at any place where there is an internet connection. For example, one can check whether there is a next train or not at his home using the AALRT info app. He can also get the detailed information about the coming trains, like estimated arrival time, train crowding level etc.…And for non-Android smartphones users, there is a dedicated AALRT website to get the information about the needed train. The developed website is so highly responsive that it can even be opened by small keypad phones. The first step in developing the Android app and the web map was to get the offline map. The map was taken

from free map rendering site, open street map (www.openstreetmap.org). The offline map has six zoom levels starting from the zoom level of 11 up to 16. On each zoom level image tiles of size, 256X256 Figure 17 will be called and merge sidewise to make the full image. Thus the map that is displayed on the app and web map are raster image tiles rather than vector tiles like in the case of maps rendered by Google map, Map Box, and open street maps.

Figure 17: Four Image tiles of size 256X256

3.6.1 Map Tiles Using QGIS

Geographical data provides the foundation on which the train's movement will be displayed on. Thus, the first stage in the implementation of the offline map system was downloading a rasterized map file of Addis Ababa from open street map free world map providing site using QGIS as shown in Figure 18. QMBTiles which is a plugin for QGIS was used to create the mbtiles formatted map of Addis Ababa. And then using Mbutil tiles images of 256x256 size was created with 6 zoom levels. And then the generated map tiles were used to make an offline map which is embedded in the app.

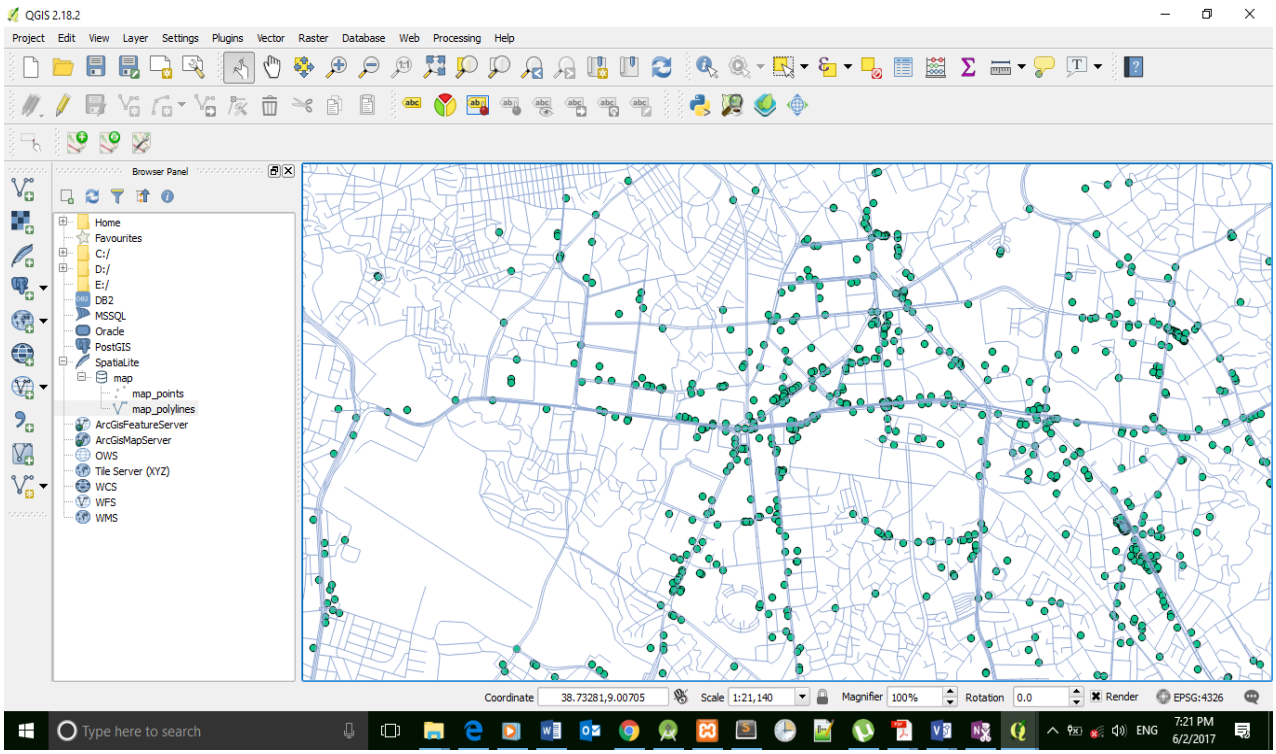


Figure 18: Screen shot of QGIS 2.18.2 being used for Addis Ababa map tile creation

The building block of both software (the app and the web map) are very similar, what makes them different is the language in which they are written. As it was stated before, Android Java was used to develop the Android app and Leaflet javascript was used to develop the web map. XAMPP server will be used for serving the necessary train information. Since GPS is not used in the implementation of the PIS due to budget constraints, virtual trains were created using windows task manager and a PHP file. The windows task manager will run a PHP file to generate a train's geographical location (latitude and longitude). The generated latitude and longitude values of the virtual trains will be stored on MySQL server. Then the virtual trains will be simulated using XAMPP server. The flow chart shown in Figure 19 shows the sequence of operation done by the app and the web map to render information to the user.

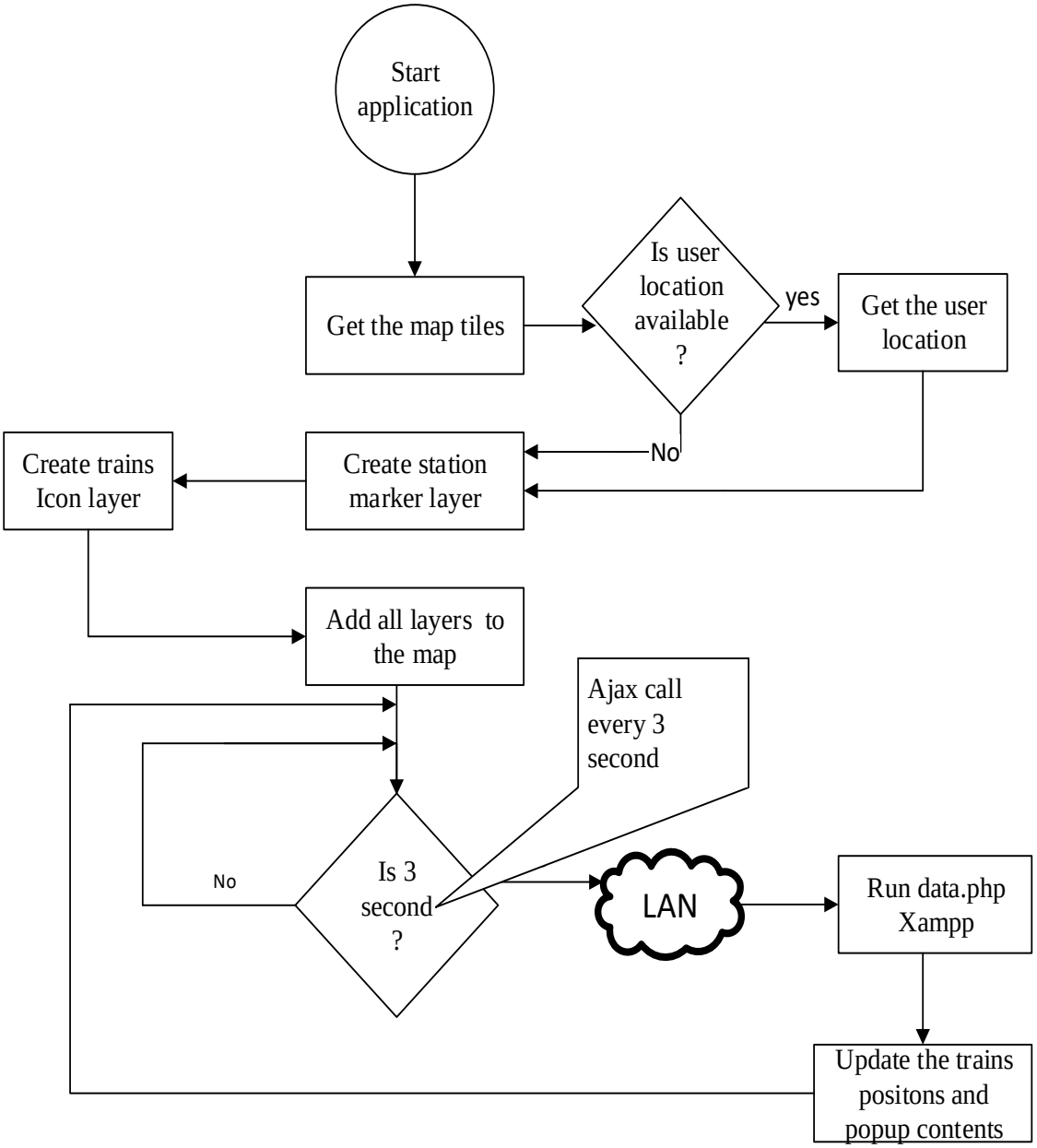


Figure 19: Flow chart showing the sequence of operation done by the web map & the Android app

3.6.2 AALRT Info App Development on Android Studio

Android is everywhere. In mid-2013, Android ran on 53 percent of all smartphones in the United States and on 80 percent of all smartphones worldwide. In a study that spans the Americas, Europe, Asia, and the Middle East, GlobalWebIndex reports that Android tablets outnumber iPads by 34 million. More than a million apps are available for download at the

Google Play store (double the number of apps that were available in May 2012). And more than 9 million developers write code using Java, the language that powers Android devices. [14].

Android Studio was used to develop the Android app, Figure 20. The developed app has two possible views; the first is map view and second is text search. The app uses offline inbuilt image tiles so that one can see trains movement on the map and touch the pulsing signal to get information about that train. What makes it ideal for the user is that it does not consume match data upon connection to the internet. It downloads only very small Kbytes of train's information unlike Google maps or online maps which take Mbytes of data in few seconds. The other advantage is that one can go through the map without needing an internet connection. The main activity java codes used in the Android app development is included in Appendix B.

There are many smartphone users in Addis. And most of the smartphones run on the Android operating system. Thus by developing an Android application that will give the whereabouts of AALRT trains we can reach many smartphone users. The app will show an offline Addis Ababa map with all AALRT waiting stations depicted with markers. One can see trains of interest on the map and touch on the train icon to see the details about that train (like the estimated time of arrival). The current location of the user will also be shown in the app.

There were some challenges in connecting the app to the server. To deal with this problem Nanohttpd server was used to make the app work like a server so that it can effectively communicate with remote XAMPP server.

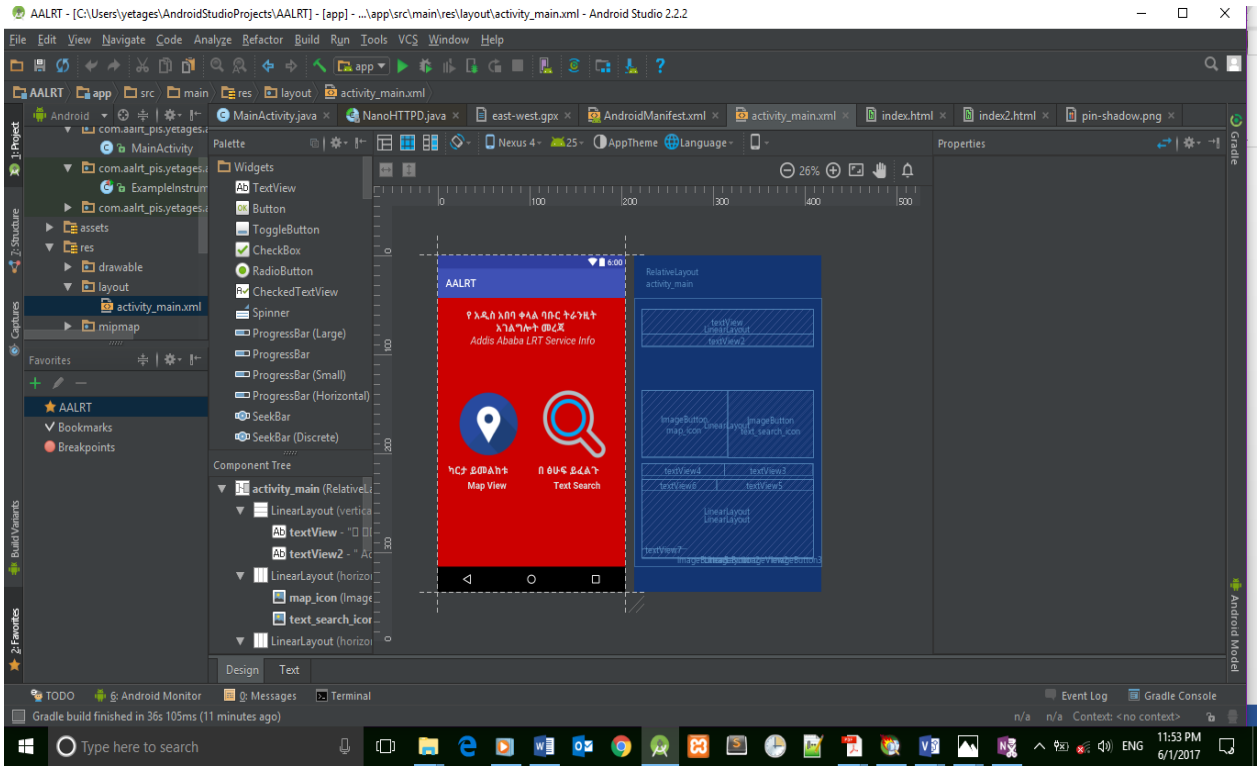


Figure 20: Android Studio; being used for AALRT info app development

3.6.3 Android Ticketing App Development

The flow chart shown in Figure 21 shows how the ticketing app works. Upon starting, the app displays three drop down boxes, Figure 8. The drop down box on top has two items in it; East-West ticket shop and North-South ticket shop. If the East-West ticket shop is selected then the down below two drops down boxes will be filled with all station names that are found along the East-West rail line. The drop down box on the left is named departing station. The one on the right it is named destination station. Pressing on one of this drop down boxes will result in a list of station names.

If the departing station and the destination are set and the ok button is pressed, then the app check first if the next train is full or not. If it is not full then it will get the train ID and prints the ticket. At the same time, it will send ticket information to the server. The ticket is printed only if sending data to the server is successful. If it is full then it will display that the next train is full and it will suggest getting the next train ticket. If the next train ticket is required then the same procedure will be used to print the ticket.

Figure 21: Sequence of operation done by the ticketing app

3.6.4 Leaflet JS Web Map Development

Leaflet JS API is a well established javascript library that can be used for a responsive web map development. Once the library is included in the web map directory the usage is very simple. The following example shows a simple map usage. The central class of the leaflet API is a map and it is used to create a map on a page and manipulate it.

```
// initialize the map on the "map" div with a given center and zoom
var map = L.map('map', {
    center: [51.505, -0.09],
    zoom: 13
});
```

The index.html file that make up the developed web map is included in Appendix A.

Sublime Text 3 (Figure 22) was used to write up the java scripts and also the server-side programs. The index file is written in HTML with Java scripts embedded in it. Using Leaflet JS a web map with six zoom levels (zoom level 11 to 16) were developed as shown in the Figure26. The map is mad of small tile images, unlike other maps which use vector tiles instead of rasterized image tiles. The web map only shows the Addis Ababa city only to the places where the AALRT railway covers.

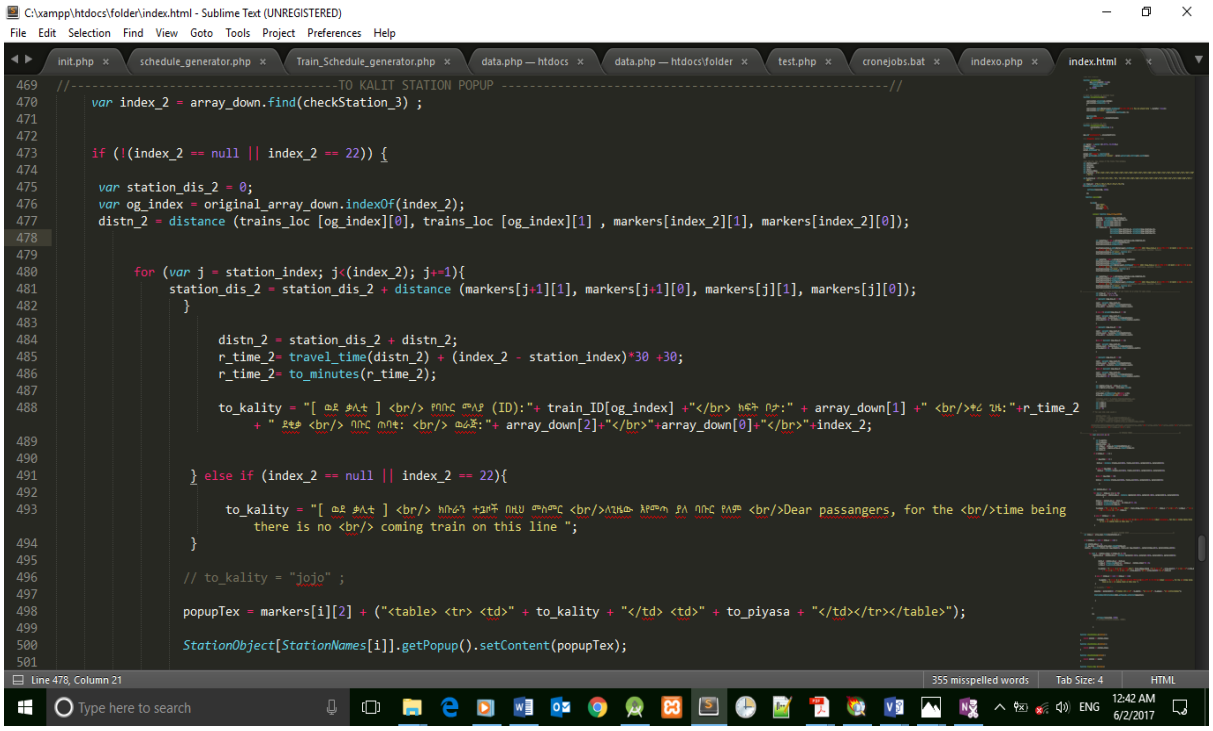


Figure 22: Sublime Text 3 windows; being used for AALRT web map development

There are 44 icons on the map showing all stations residing along either side of the rail roads in the East-West and North-South direction. All icons are responsive and upon touching one can read the station names in both Amharic and English including the details of information coming to that station. The web map can also show the current location of the user on the map with a pulsing red circular object but the user needs to let the web file to access the location information from the device first. The green icon shows the station located in East-West corridor. The blue icon shows the station in South-North corridor. For the sake of convenience, the five common stations were deliberately duplicated as shown in Figure 23 so that the green train information and blue train information can be shown in the popups separately.

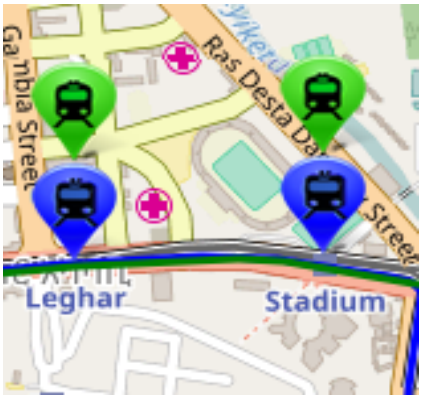


Figure 23: Duplicated station markers at Leghar and Stadium Station

Some web pages on the internet are not highly responsive that their contents cannot easily be opened on mobile devices. But web maps developed with Leaflet JS are highly responsive that the pages can be opened on many mobile device types without their contents being misplaced or distorted. That was the reason why Leaflet JS was used to develop the web map.

For non-Android phone type users, provided that their phone has an internet access, a website that can support mobile devices is developed so that they can go to this website and get the necessary information about the coming train.

3.6.5 Server Side Programming with XAMPP

Of course, there's not much point to being able to change HTML output dynamically unless one have a means to track the changes that users make as they use a website. In the early days of the Web, many sites used "flat" text files to store data such as usernames and passwords. But this approach could cause problems if the file wasn't correctly locked against corruption from multiple simultaneous accesses. Also, a flat file can get only so big before it becomes unwieldy to manage—not to mention the difficulty of trying to merge files and perform complex searches in any kind of reasonable time [15].

That's where relational databases with structured querying become essential. And MySQL, being free to use and installed on vast numbers of Internet web servers, rises superbly to the occasion. It is a robust and exceptionally fast database management system that uses English-like commands [15].

The highest level of MySQL structure is a database, within which you can have one or more tables that contain your data. For example, let's suppose you are working on a table called

users, within which you have created columns for the surname, first name, and email, and you now wish to add another user. One command that you might use to do this is [15]:

```
INSERT INTO users VALUES ('Smith', 'John', 'jsmith@mysite.com');
```

XAMPP server, with Apache, MariaDB, PHP, and Perl installed on it, is used for the server side programming. To log in, add and retrieve data from the database PHP is used. The reason for using PHP as an interface to MySQL is to format the results of SQL queries in a form visible in a web page. As long as one can log into your MySQL installation using a username and password, it is also possible to do so from PHP. However, instead of using MySQL's command line to enter instructions and view output, one will have to create query strings that are passed to MySQL. When MySQL returns its response, it will come as a data structure that PHP can recognize instead of the formatted output one see when it is worked on the command line. Further PHP commands can retrieve the data and format it for the web page. On the XAMPP three databases are created. These are:

1) **All_train _information;** this database stores the information like the number of passengers at a specified time, train speed in km/hr. and geo-coordinates of all trains (Figure 24)
2) **Train_route;** this database stores the geographical path followed by the trains. In train simulation, this database will provide the path that will be followed by the generated trains.
3) **Ticket_data;** All data related to ticket sale will be stored in this database. In the database for each ticketing shops, there is a single a table with many columns.

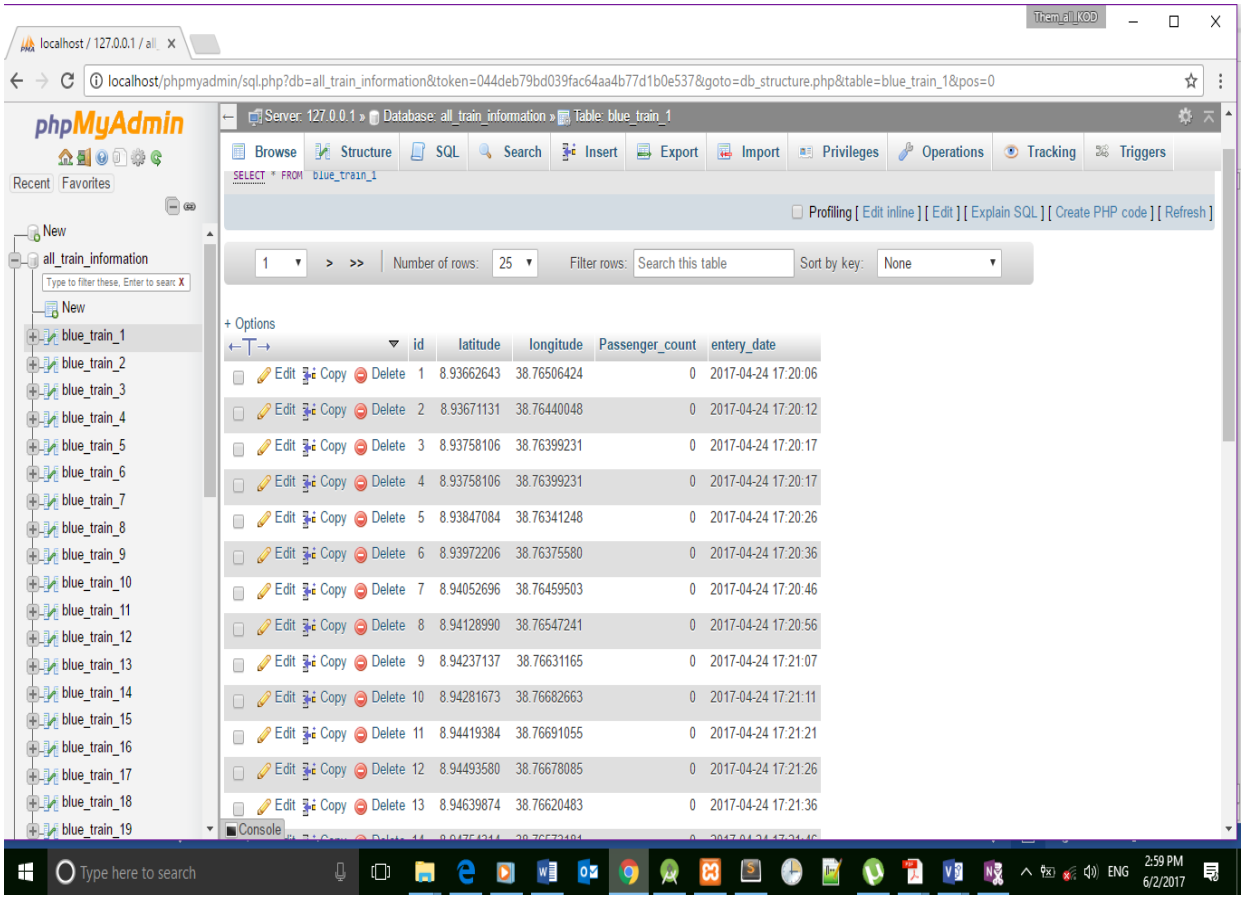


Figure 24: Screen shot of XAMPP server showing all_train_inforamtion database

The server side PHP files are included in Appendix C.

Chapter 4

PIS Simulation

A PIS is a large system consisting of many system components. This thesis does not attempt to implement a full real-world passenger information system. The goal of this thesis was to develop an affordable PIS that can be implemented in short period of time and that would validate the design in Chapter 3. Some of the aspects of the real-world system were not included in this proof-of-concept prototype. Firstly, since the GPS tracking system could not be used due to budget constraint, there was a need to build a train simulation to provide an artificial geo-position (latitude, longitude) feed. However, in order to simulate the train moving on virtual rails and stopping at virtual stations, geographical data is needed. Therefore the first system to be implemented was the GIS service. After that was the simulator, then the tracking service and finally the HTTP Server and corresponding views.

Two views were implemented; HTML view and Android app view. The developed web map files were added to htdocs folder of the XAMPP server installation directory. Figure 25 shows the locally hosted AALRT web map.

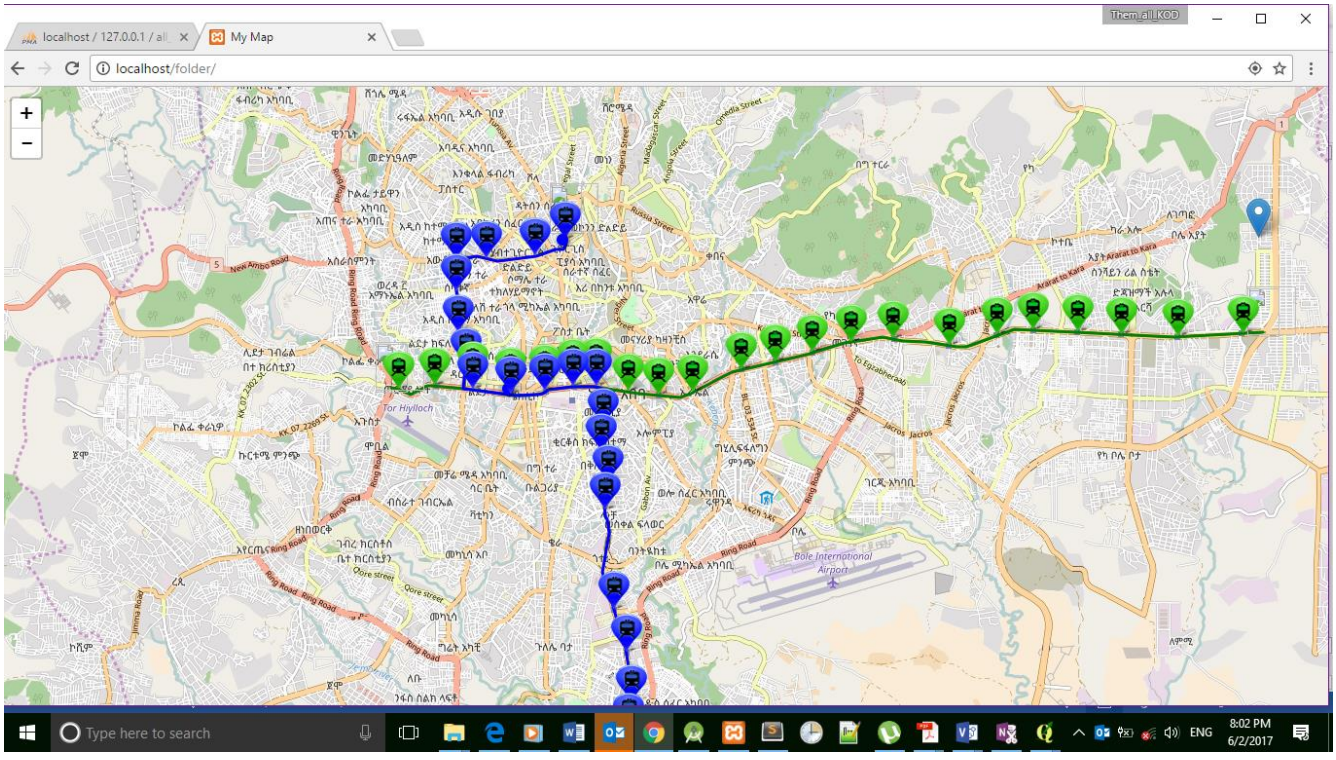


Figure 25: Screen shot of locally hosted web map

4.1 Simulating Train Movement

As it was stated before in this simulation we didn't use a real GPS to track the train rather the train movement was generated and artificial geolocation was feed to the database. For this purpose PHP files that would generate the latitude and longitude position of virtual trains were created. And the generated values were inserted into a database automatically. The simulation was made only for three trains. Windows task scheduler was employed to create a crone job, Figure 26. Crone jobs are scheduled tasks that will be carried out by the operating system when the pre specified parameters are meet. The parameter is mostly time. A simple crone job example is a scheduled task to run a file one time at 6:00 AM in the morning. Thus windows task scheduler was used to create three tasks that will run three PHP files every 3 second to generate artificial train location. While creating a task in the security options one need to make sure that "use the following account" option is set to 'SYSTEM'.

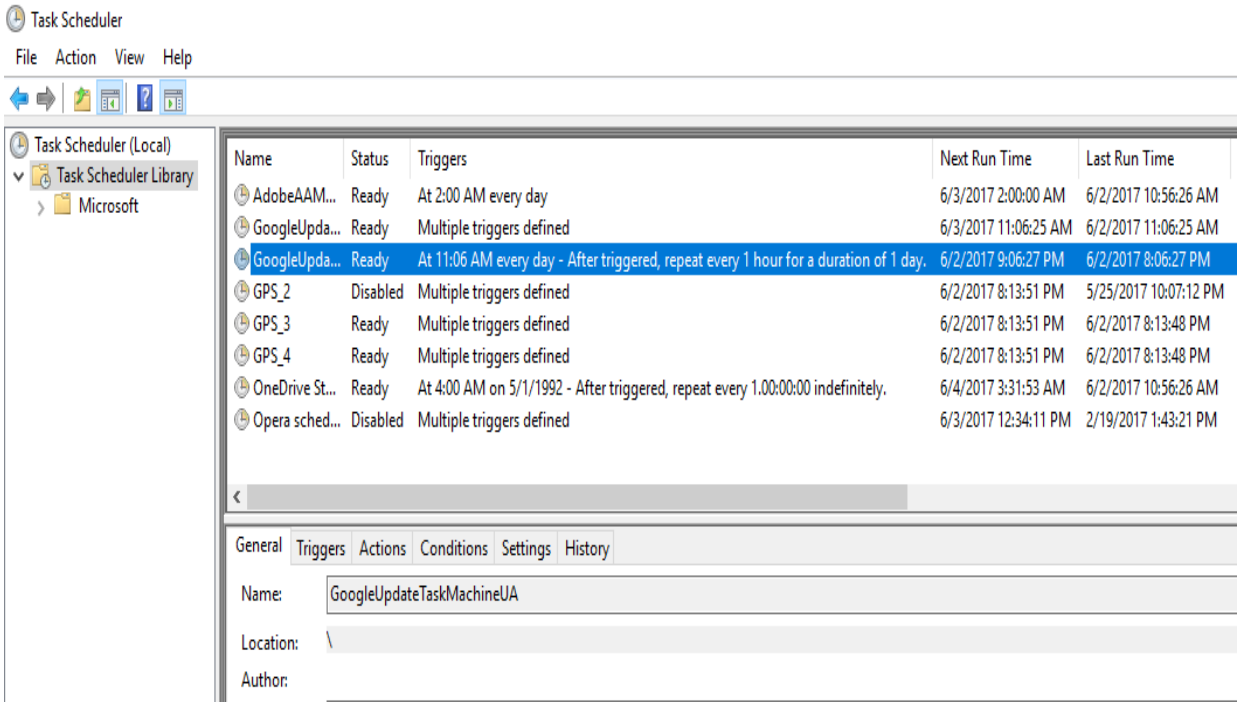


Figure 26: Screen shot of windows task scheduler with two tasks running and one disabled

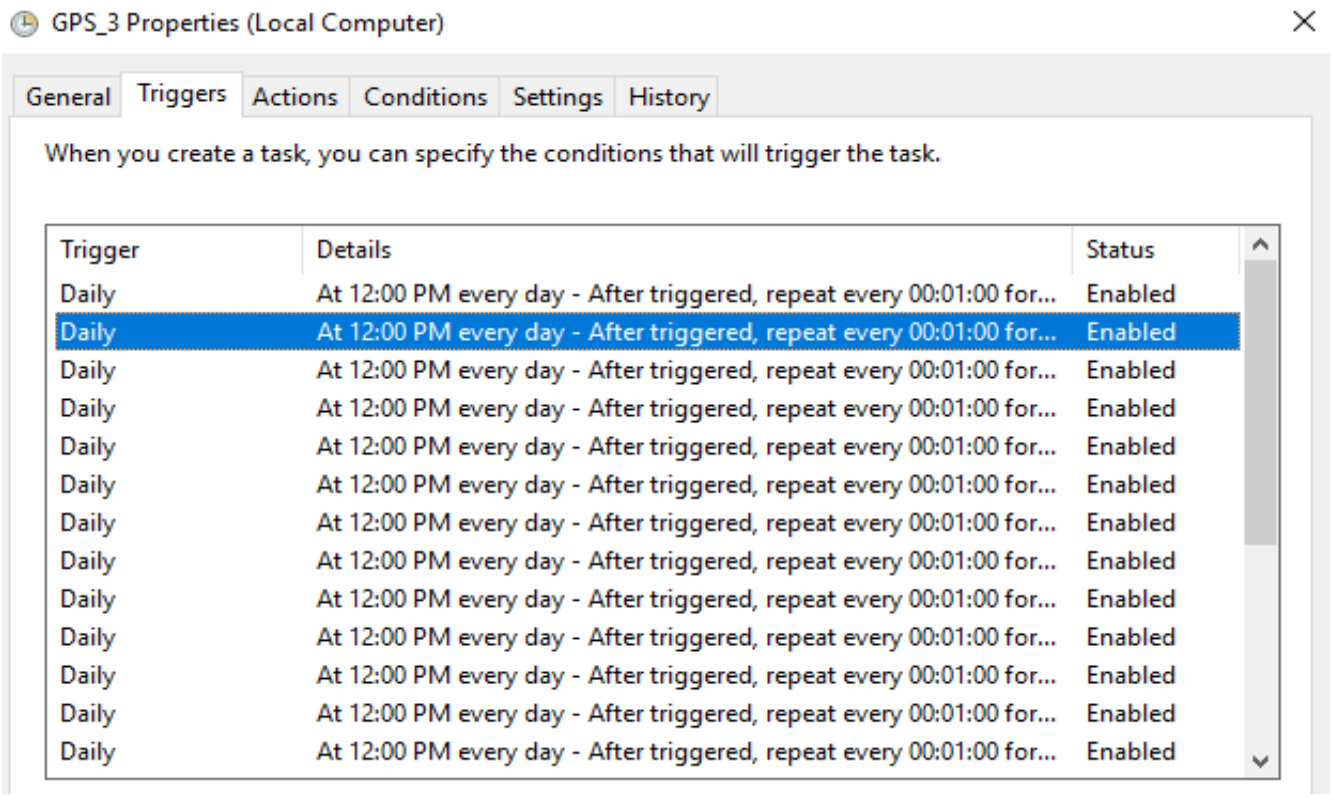


Figure 27: Screen shot of twenty triggers which are 3 seconds out to each other made on a single task

One problem encountered with the windows task scheduler was that the minimum period between two task run times is 1 minute. But we need to update the train positions every three second or so. The easy solution for this was to define multi triggers in single task scheduled every 3 second Figure 27.

4.2 Web Map Simulation Result

Opening the locally hosted web map resulted in three pulsing blue icons representing the blue trains in the north south corridor. For the sake of simulation we only be looked at the North-South train movement simulation. This was because the East-West line train movement simulation would not be in any way different from this corridor.

When the pulsing icon was clicked a popup appeared showing the train Id, speed of the train, number of passengers on train and the train type (coupled or single) Figure 28. The small blue dot circled with a light blue circle shown in the map is a train icon. On the figure, the train was moving to the south direction (heading to Kality) and it was located near Abnet station. Its

moving speed was 45kmhr. There were 274 passengers on it. It was a single train with a train ID Blue_train_3.

Basically in real implementation the speed of the train will be calculated by the GPS and will be sent to the server. But in this thesis it was assumed that the train speed is constant, 45Kmhr. Thus to calculate the estimated time of arrival of the train this speed was used as one input.

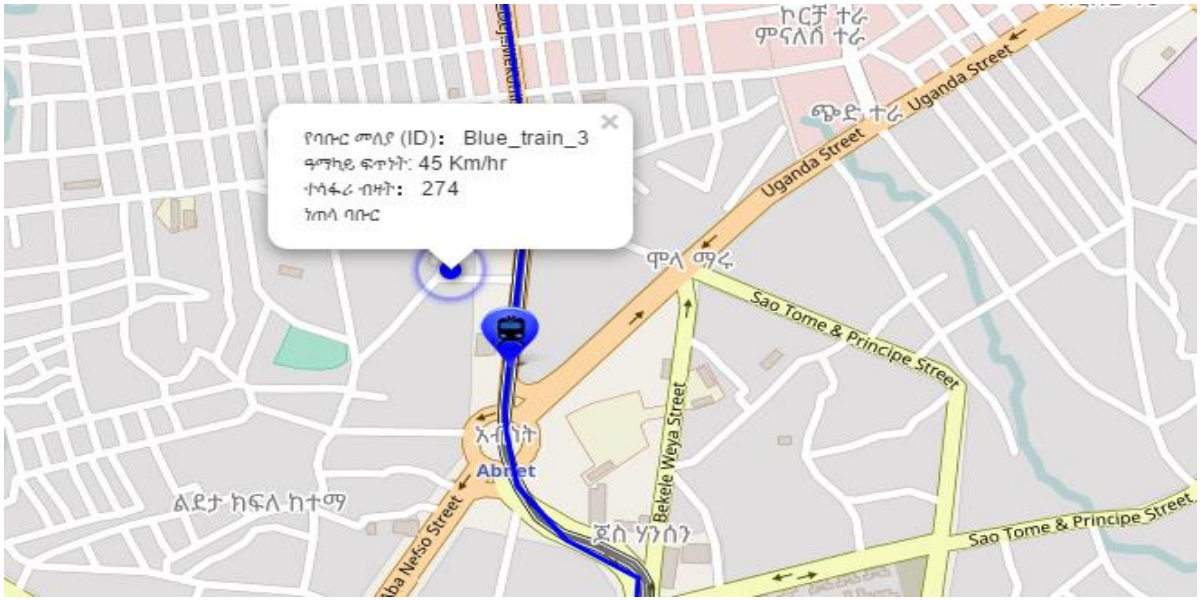


Figure 28: A blue pulsing icon popup showing detailed info about a train

When one clicks on one of the station icon on the map a popup will appear. For example, if the station is a blue station (located on North- South rail line) the popup will display two columns titled [ወደ ፒያሳ] and [ወደ ቃሊቲ], Figure 29. The one on the left details about the trains moving to Kality, that is why it is titled [ወደ ቃሊቲ]. And the one on the right details about trains moving to Menelik II and it is titled [ወደ ፒያሳ]. The station name is shown intop, left side of the popup.

The train on Figure 29 arrived at Riche station and that was why the ETA on the popup showed zero minutes. The expected number of passengers getting off at this station was shown to be 18. And there was an 8 people of space available on the train before the passengers get off. The train ID was BT_3 and it was heading to Kality. But on the line moving to Menelik II there were no train coming.

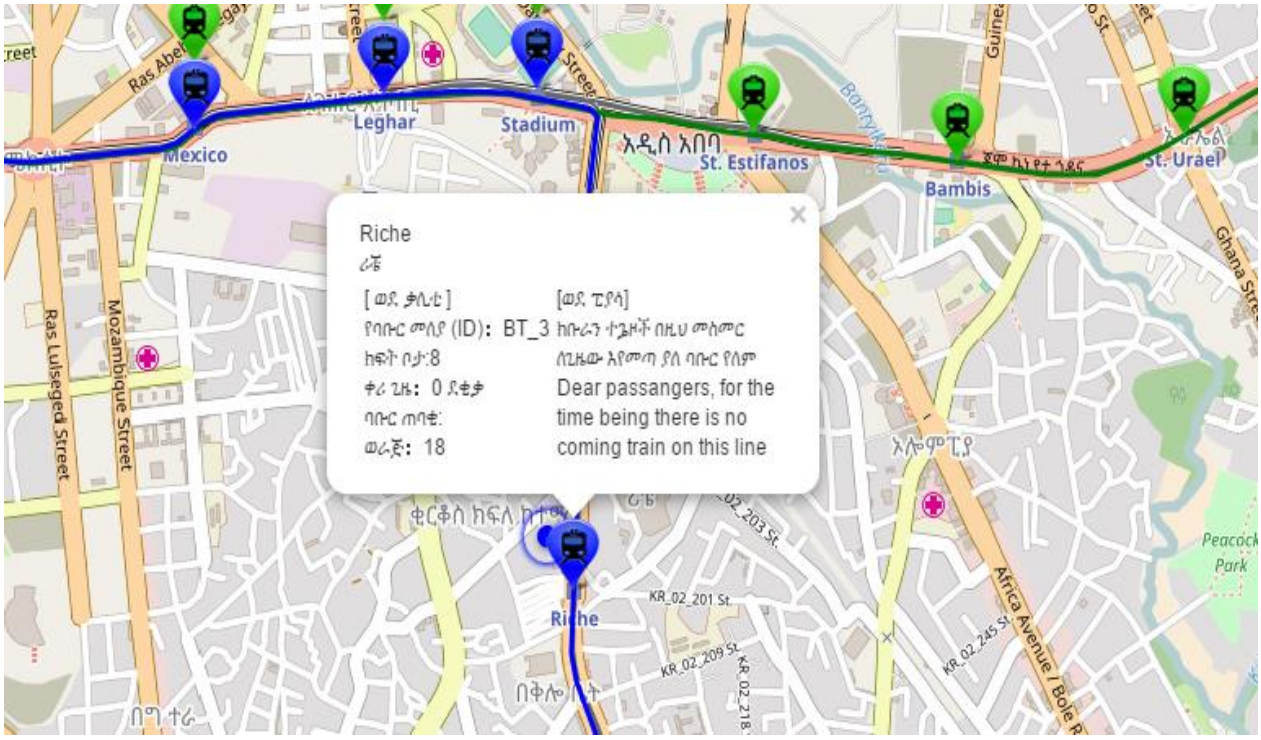


Figure 29: Riche station icon popup detailing on the next train

The train, BT_2, on Figure 30 which was heading to Menelik II arrived around Legare station as shown in the map and the ETA of the train to get to Tegbared station was 3 minutes. There was an 11 people of space available on the next train and 11 passengers were expected to get off at the station. One can see also that there was no next train on the line heading to Kality.

The 3 minutes ETA was determined knowing that on Mexico station the estimated dwell time of the train was 1 minutes. Moving with 45 Km/hr the time needed to move from one station to the next station is around 1 minutes except in some station like between Lancha station and Nefasilk 2 which takes around 2 minutes.

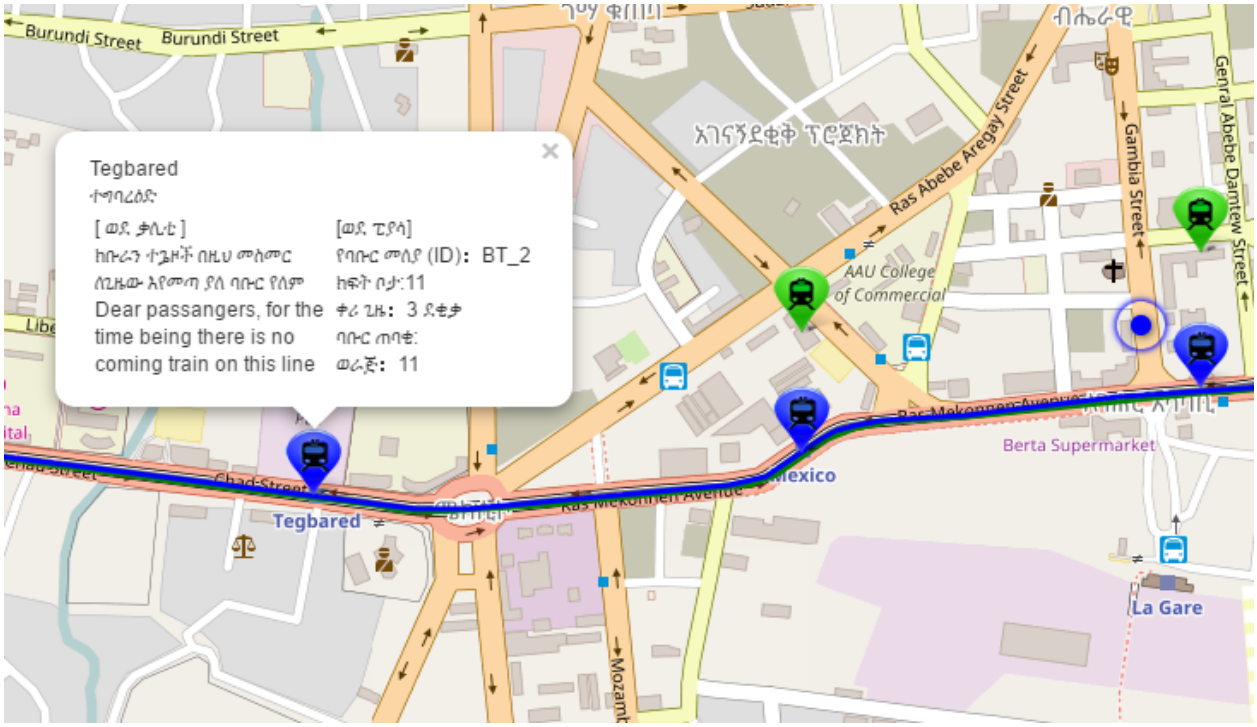


Figure 30: Tegbared station icon popup detailing on the next train

4.3 Simulation Result of AALRT Info App

As it was stated earlier the XAMPP server is running on personal computer. And to access the database on the XAMPP server establishing a direct Wi-Fi connection between the laptop and the Android phone was needed. A LAN could also be used to connect the two devices. On this simulation, a direct Wi-Fi was used to establish connection between the devices. Below is the procedure used to connect the laptop and the android phone with direct Wi-Fi. On the phone the wireless and network setting was opened and the tethering and mobile hotspot option was selected to create the hotspot

Then the created Wi-Fi appeared on the laptop network and then the laptop was connected to the mobile hotspot. Now to access the locally hosted files one needs to know the IP of the machine. Running ipconfig on CMD, the IP of the machine was determined. And then to access the local host http:// IP_of_the machine/folder was run on the Android browser. This URL was used to run the file and fetch the result remotely. The result of the simulation is shown in Figure 31.

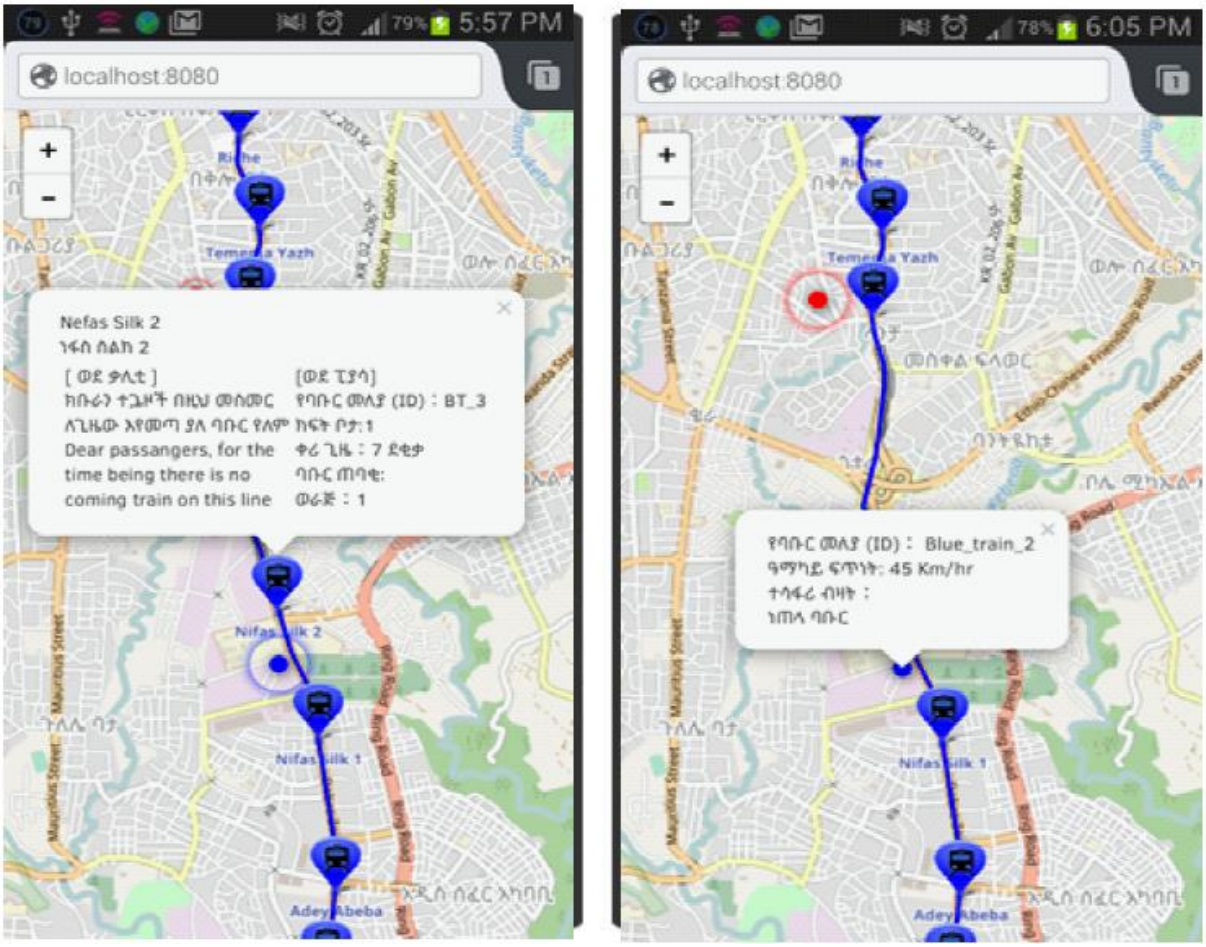


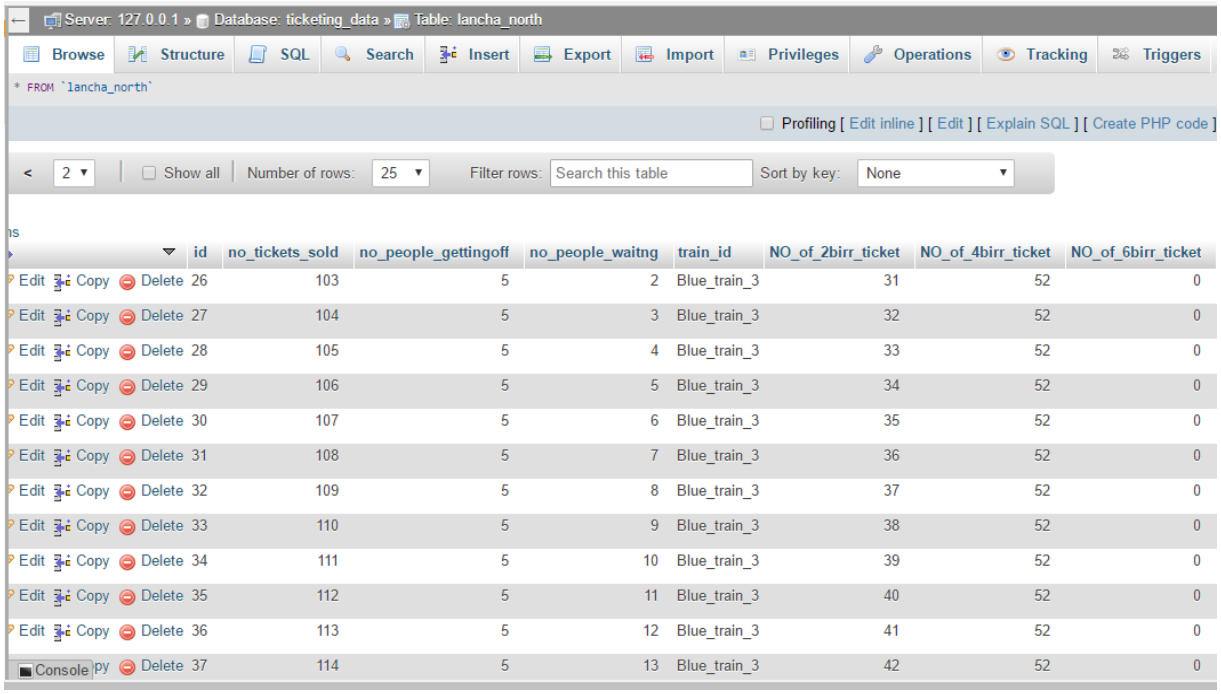
Figure 31: Screen capture of AALRT info app showing trains information

The screen capture on the right ,Figure 31, shows an information abot a train coming to Nifas Silk 1 station. And the one on the left side shows a popup of Nifas Silk 2 station. And The popup details about the coming trains to this station.

4.4 Over Crowding Avoidance Simulation

The database shown in Figure 32 shows a table of ticketing information collacted from Lancah station left side ticket shop. The table showes that the coming train's ID was Blue_train_3 and five people were expected to get off from this train at Lancha station. At this station, it is also shown in the table that there were 13 peoples waiting for this train.

Before the last ticket was sold on the lancha station there were 12 peoples waiting for the train. The passenger who bought the last ticket destined for the Stadium as shown in Figure 33.



Server: 127.0.0.1 » Database: ticketing_data » Table: lancha_north

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

```
* FROM `lancha_north`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]

< 2 | Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

	id	no_tickets_sold	no_people_gettingoff	no_people_waitng	train_id	NO_of_2birr_ticket	NO_of_4birr_ticket	NO_of_6birr_ticket
Edit Copy Delete	26	103	5	2	Blue_train_3	31	52	0
Edit Copy Delete	27	104	5	3	Blue_train_3	32	52	0
Edit Copy Delete	28	105	5	4	Blue_train_3	33	52	0
Edit Copy Delete	29	106	5	5	Blue_train_3	34	52	0
Edit Copy Delete	30	107	5	6	Blue_train_3	35	52	0
Edit Copy Delete	31	108	5	7	Blue_train_3	36	52	0
Edit Copy Delete	32	109	5	8	Blue_train_3	37	52	0
Edit Copy Delete	33	110	5	9	Blue_train_3	38	52	0
Edit Copy Delete	34	111	5	10	Blue_train_3	39	52	0
Edit Copy Delete	35	112	5	11	Blue_train_3	40	52	0
Edit Copy Delete	36	113	5	12	Blue_train_3	41	52	0
Console py Delete	37	114	5	13	Blue_train_3	42	52	0

Figure 32: Ticketing database showing Lancha left side ticket shop data

Figure 33: Ticketing app being used to print a ticket destined for Stadium

When this ticket was sold the database was updated and the number of passengers waiting for the train increased by one and became 13. The last line on the database table showed the most recent information about the station.

On the web map, when the Lancha station icon was clicked a popup showing information about the trains going to the north and south was displayed, Figure 34. As it can be seen from the figure, the train was 11 minutes away from this station. There were 13 peoples who were waiting for the coming train. And there were a 25 people of space available on it. The same as the data read from the database, 5 peoples were expected to get off at Lancha station.

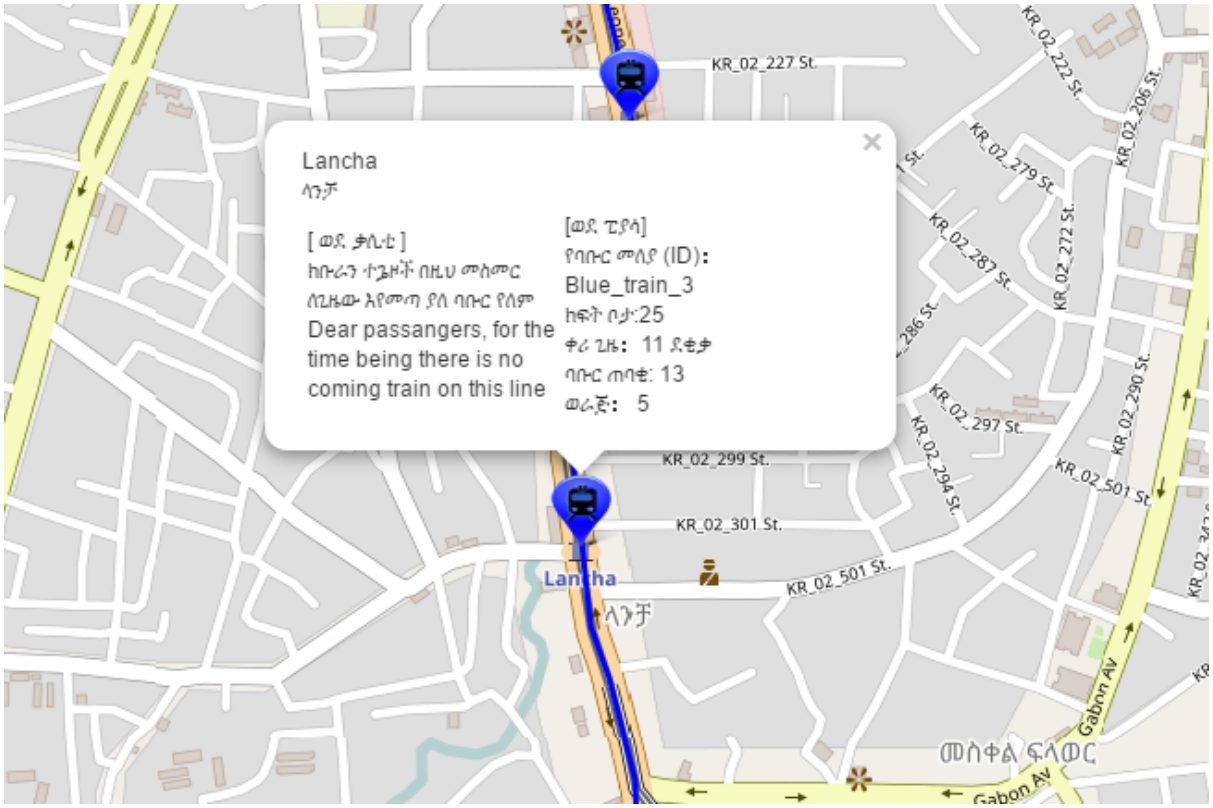


Figure 25: Lancha station popup showing an info about the next train

Train icon popup showing information about Blue_train_3 is shown in Figure 35. The train speed was 36 km/hr. And the number of passengers on the train was 261, which means there were a 25 people of space available on it as it was shown by the Lancha station icon popup above. The popup also showed that the train was a single train. But when the train arrived at Autobus Tera it reached its carrying capacity which is 286 passengers.

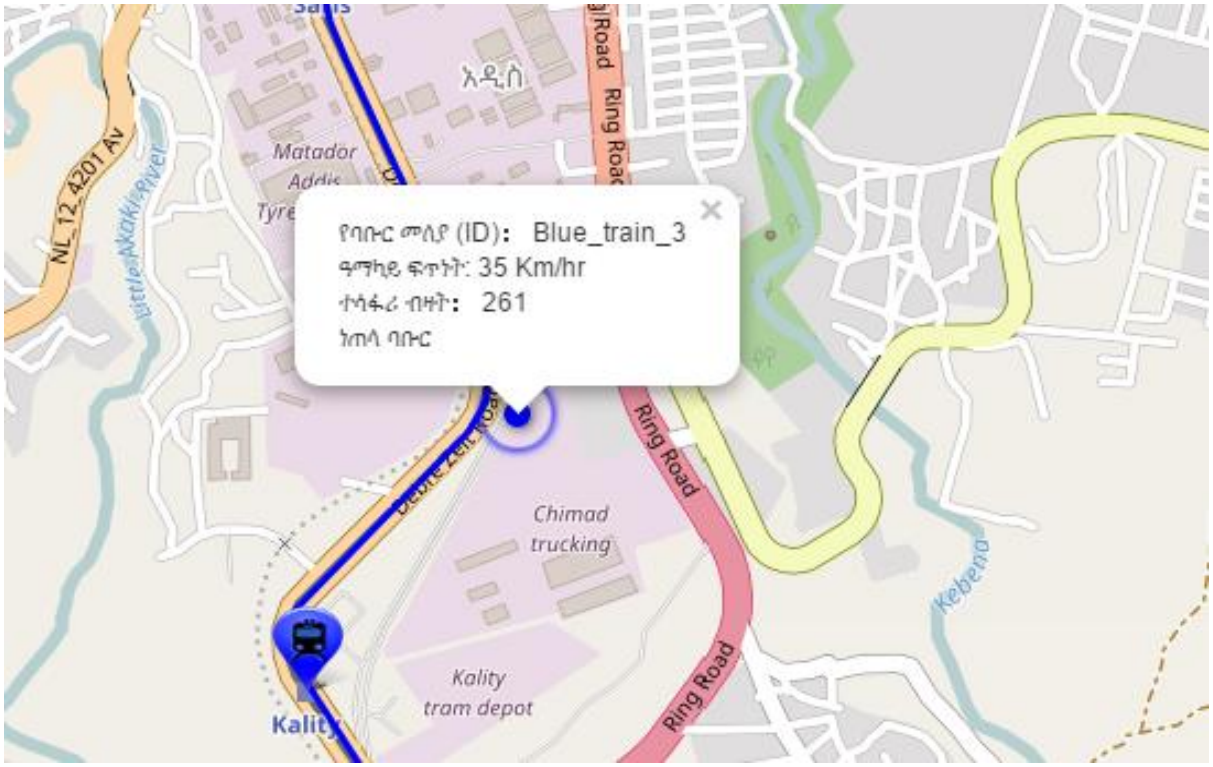


Figure 26: A train icon popup showing an info about a train

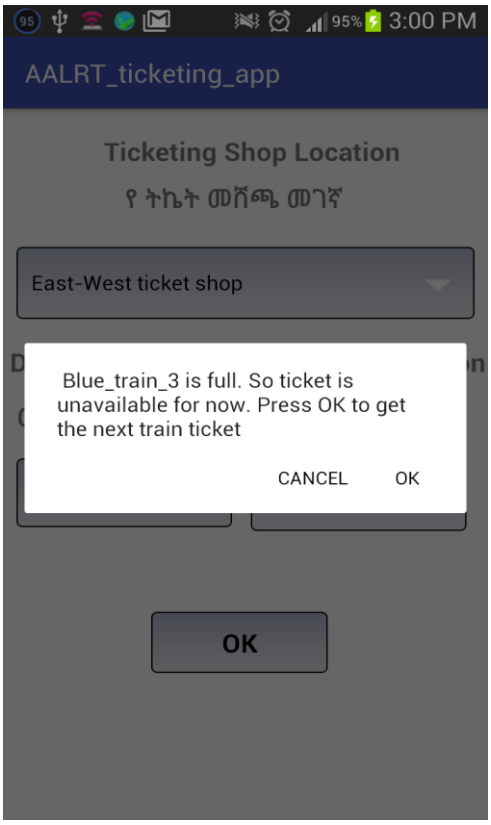


Figure 27: Ticketing App train full Alert

Thus the ticketing app started to alert the ticket seller that the next train was full, Figure 36. It gave an option for getting the next train ticket. Thus if the passenger wants to get the next train ticket OK button can be pressed. This alert continued to be displayed until the train passed that station. Figure 37 and Figure 38 shows that the next train is full.

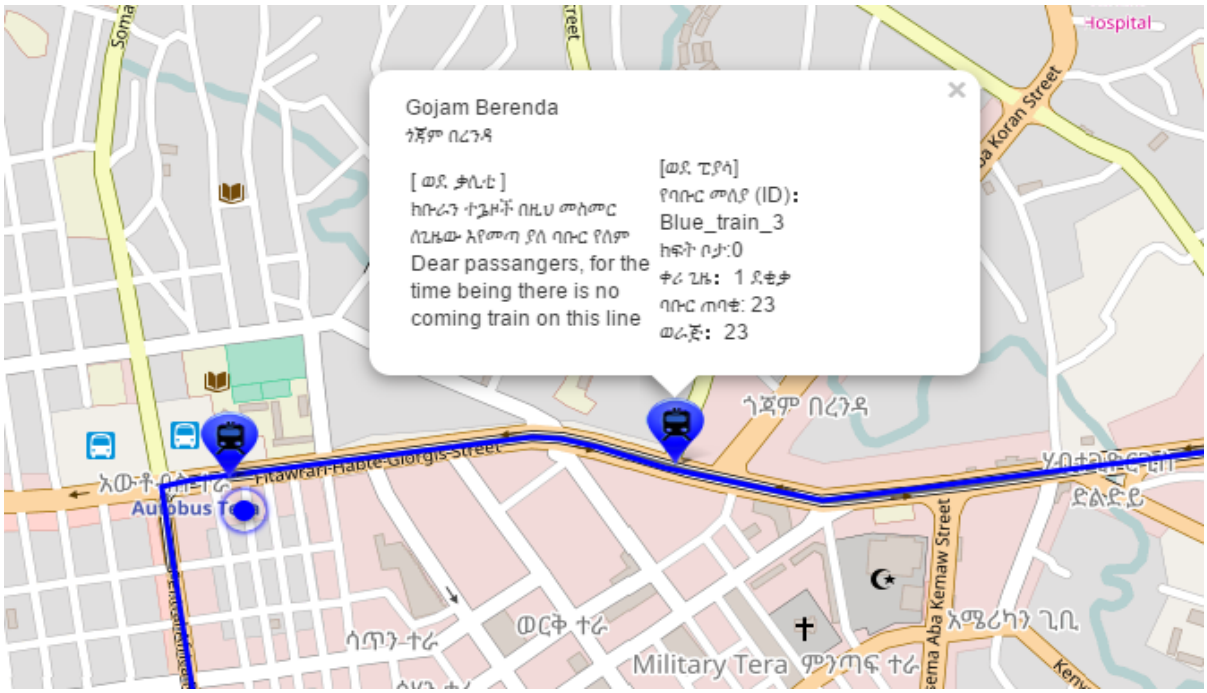


Figure 28: Gojam Berenda icon popup showing no space available on the next train

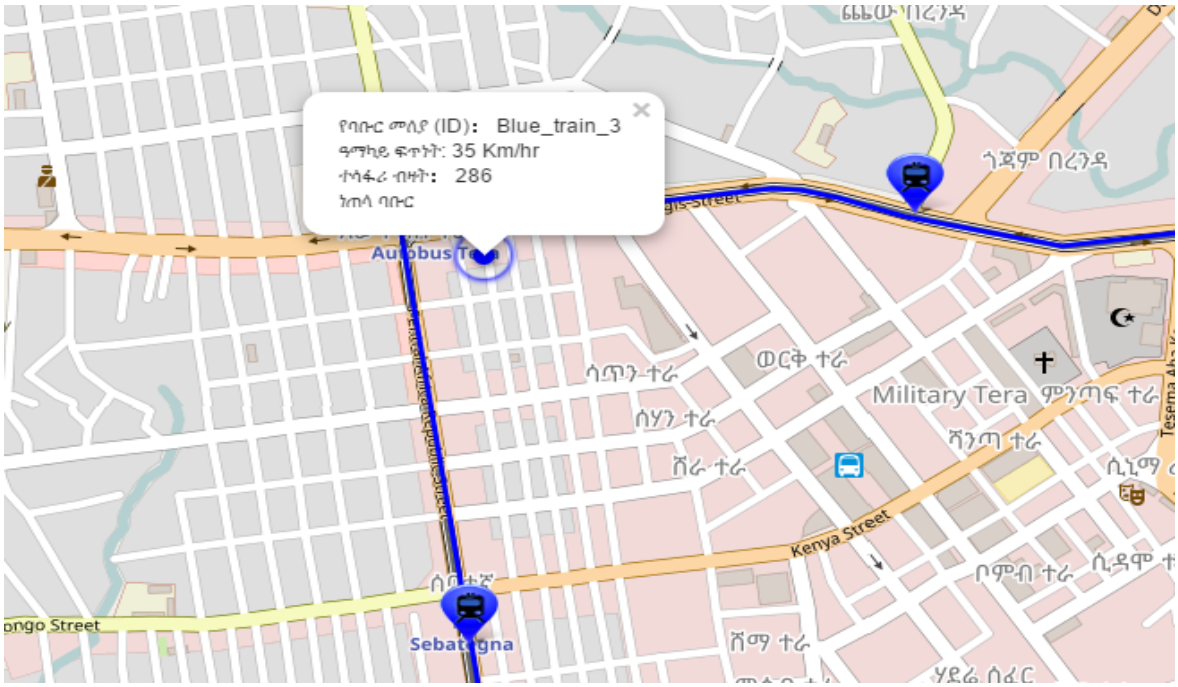


Figure 29: Train popup showing that the train is full

Chapter 5

Conclusion and Recommendation

5.1 How Well the Stated Objectives Were Achieved?

The first objective of the thesis was the establishment of a cost-effective passenger counting system that can be used for managing passengers and which will also help future studies relating to passengers and their travel behavior of train service. As it was stated in previous sections it was shown that, without requiring a video camera based passenger counting system or any other sophisticated passenger counting system, Android ticketing device could be employed for indirect passenger counting. Every time a ticket is sold by this machine information about this ticket is sent to the central controller (or server). This data is collected from all the ticketing shops' Android-based ticketing POS machines and processed to find out the number of passengers on the trains, number of passengers waiting at the station etc.…Once this information is sent to the server, it will be stored on the server so that the data can be used for future studies. And the system was simulated and the result showed that the objective is achieved.

The second objective was solving the problems of manual ticketing by introducing Android ticketing device. The travel tickets that are being used by AALRT don't have a time stamp on it. Thus a single ticket is being used illegally many times in a single day. But the proposed ticket which will be generated by the Android ticketing device has expiry time and time of sale printed on it that it cannot be used more than once. Android ticketing app was developed for the device to print the tickets. The ticketing app was simulated together with other system components and it was shown that the ticketing app could be used to sell a ticket. Thus if this device is used for selling tickets then ticket fraud can be greatly reduced. Thus the stated objective was achieved by developing an Android ticketing app.

The third objective was to enable any passenger waiting on station to have an information about the next train using station display, Android app or website. And an Android app with the name AALRT info app was developed for providing the user with the detailed information about the coming train. Train information will also be provided through a dedicated website. The designed PIS simulation has shown that both the Android app and the website can provide a

detailed information about the coming train. The station display was not part of the simulation because of budget constrained but the fact that the designed PIS worked with the Android app and the website can tell us that it can also work with station display. Thus the stated objective is achieved.

The final objective of this work was to establish a system that will put a limit on the carrying capacity of the trains so that the overcrowding problem on AALRT trains can be avoided. And the carrying capacity of a single train is 286 peoples and it is assumed that this maximum passenger count is the threshold for overcrowding (overcrowding is very difficult to measure and the research [7] showed that what is overcrowded for some passengers in New York might be less crowded for Addis Ababa passengers. Thus based on the design capacity of the AALRT trains a 286 passenger count is assumed to be less crowded). Thus the proposed solution for overcrowding problem is to stick to this carrying capacity of the train. To do so a PIS system that will keep the number of passengers below a maximum count of 286 was established. The designed PIS is capable of determining the number of passengers, the available space and the number of passengers getting off at any station from any train. And in the simulation chapter (Figure 36, Figure 37, and Figure 38) it was shown that when the number of onboard passengers added with the number of passengers, who are holding the ticket for this train, waiting at the stations ahead becomes 286, ticket sale for this particular train will be stopped. And if this system is implemented it is expected that the overcrowding problems on AALRT will be avoided. This overcrowding avoidance system was simulated, and the result showed that the established system do its intended job.

5.2 Conclusion

This thesis was aimed at doing away with or alleviating the two main problems of AALRT; these are overcrowding on trains and lack of wayside information about the coming train. In doing so a cost-effective, easy to implement PIS was designed. The PIS basically renders information to the user in three ways; through dot matrix station display, Android app named AALRT info app and dedicated AALRT Website. More importantly, the station display will help the passengers waiting at the station by providing the necessary information about the coming train. The information that will be displayed at the station display includes; estimated arrival time of the coming train, the number of passengers expected to get off at the station and the train type (coupled or single). This information is very important both to AALRT and the

passengers. The passengers could use it for decision purpose. For example, if they are told that all the coming trains are fully loaded then they may resolve for other means of transportation rather than waiting for a less crowded train that might not come. AALRT peoples could use this information for operational decision making i.e. whether to send a coupled train or single train depending on the number of passengers waiting at various stations.

The PIS design also included indirect passenger counting using Android ticketing devices, which are proposed to be used by AALRT ticketing shops. The passenger count will help to determine the current number of passengers on any AALRT train. Not only that, the expected number of passengers getting off from any train can also be known. Using this information it was shown that it is possible to determine the available space on the coming train. And this information will be used by the ticketing peoples to decide when to stop selling tickets for the coming train. The proposed Android ticketing device is also expected to solve the existing inherent problems of manual ticketing in the station's ticketing shops. The travel tickets that are being used by AALRT don't have the following information; time of ticket sale, estimated expire time, and train ids. The lack of this information on the tickets made them vulnerable for fraud use. An interview with ERC people revealed that ERC is losing a great deal of money due to ticket fraud. But if the proposed ticketing device is used for selling tickets that bear time of ticket sale, estimated expire time and train ID's then the stated ticket problems can be easily avoided.

5.3 Recommendation

This thesis is a fast and timely response for the existing PIS and overcrowding problems encountered in AALRT. The designed PIS is recommended to be implemented by AALRT as a soon as possible so that the stated problems can be solved before they get worse than they already are.

From what is experienced upon doing this thesis the author believes that there is a plenty of room for improvement. The designed PIS could have been designed in a broad way that could have provided a train information to the user through voice and SMS. In addition to that, an application for iPhones, Windows, and other smartphones could have been developed. But due to time, budget, and lack of experience constraints, the PIS could not be designed wider than the one design on this thesis. The ticketing system can also be integrated with mobile banking

that user can buy tickets online without having to go to the ticketing office. And to increase the reliability of the passenger counting system redundant passenger counter using laser sensor or another sensor can also be introduced.

To the best of the author's knowledge, the concept of using passenger information system to avoid overcrowding on public transport was not used before. Thus it can be said that it is a new research area. On top of that the main problem that AALRT is facing at the time of this paper preparation is the overcrowding on trains. Thus many research can be made regarding overcrowding and railway systems. Some research ideas are listed below.

- Devising an overcrowding forecasting system that can be used by AALRT operational peoples so that overcrowding can be predicted and avoided before it happens.
- Train Scheduling for overcrowding avoidance. (Determining the optimal operating speed of the train so that overcrowding can be reduced, coming up with a dynamic train schedule that can reduce overcrowding etc.)

Reference

[1] J. Schumann *et al.*, "Experience, Economics, and Evolution From Starter Lines to Growing Systems," 9th Nat. LRT Conf., Portland, Oregon, USA, 2003, pp.528

[2] J. Barr *et al.*, "Implementing BRT Intelligent Transportation Systems", American Public Transpo. Assoc., Washington DC, 2010, pp.13-15

[3] John Fihn & Johan Finndahl, "A Framework for How to Make Use of an Automatic Passenger Counting System", Uppsala Univ., Uppsala, Sweden, 2013

[4] P.G. Hemily *et al.*, "TCRP Report 113: Using Archived AVL-APC Data to Improve Transit Performance and Management", Transpo. Research Board, Washington, United States of America, 2006, pp.6 - 21.

[5] Ubaka and Glotzbach "ITS/APTS Architecture Guide: To Help Florida Transit Systems Comply with the National ITS Architecture Consistency Policy", Florida Dept. of Transpo., Tallahassee, United States of America, 2006, pp.2-4

[6] M. Saavedra,"An Automated Quality Assurance Procedure for Archived Transit Data from APC and AVL Systems", PhD thesis, Univ. of Waterloo, Waterloo, Ontario, Canada, 2010.

[7] N. M. Mahudin, "Quality of rail passenger experience: the direct and spillover effects of crowding on Mahudin well-being and organizational behavior". Ph.D. dissertation, Univ. of Nottingham, Nottinghamshire, England, 2012

[8] Z. Li and D. A. Hensher, "Crowding in Public Transport: A Review of Objective and Subjective Measures" The Univ. of Sydney, Sydney, Austrialia, 2014

[9] C. Sungur *et al.*, "Smart Bus Station-Passenger Information System", 2nd ICISCE Int. Conf. Shanghai, China , 2015

[10] D. Patingel *et al.*, "Smart Onboard Public Information System using GPS & GSM Integration for Public Transport", Int. J. of Advanced Research in Comput. and Commun. Eng., Vol. 1, Issue V, 2012

[11] K. Hassan *et al.* "A System for Monitoring and Reporting Excessive passengers in Public Buses Case Study: Tanzania", Int. J. of Eng. and Comput. Sci. ISSN:2319-7242 Vol. 2 Issue 8 August, 2013 Page No. 2342-2347

[12] M. D. Rossetti and T. Turitto, " Design of an Integrated Transit Monitoring System based on Radio Frequency Identification", Dept. syst. Eng., Univ. of Virginia, 2013

[13] D. Lefloch, "Real-Time People Counting system using Video Camera" M.S thesis , Dept. of Comp. Sci., Gjøvik Univ.College, Gjøvik, Norway, 2007

[14] B. Baud, *Java programming for Android Programmers for Dummies*., 1st ed., New Jersey: Wiley, 2014

[15] R. Nixon, *Learning PHP, MySQL, JavaScript, CSS & HTML5* , 3rd ed., O'Reilly Media, 2014

Appendix

A. AALRT's Web Map Index.html File

```
  <!DOCTYPE html >
   <html >
   <head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <meta http-equiv="X-UA-Compatible" content ="IE=edge">
   <title > My Map </title>
   <meta name='viewport' content='initial-scale=1,maximum-
scale=1,user-scalable=no' />
   <link rel="stylesheet" href = "scripts/leaflet/leaflet.css"/>
   <link rel="stylesheet" href="scripts/leaflet/L.Icon.Pulse.css" />
   <link rel="stylesheet" href="scripts/leaflet/font-
awesome.min.css" />
   <script src="scripts/leaflet/leaflet.js"> </script>
   <script src="scripts/leaflet/gpx.js"></script>
   <script src="scripts/leaflet/L.Icon.Pulse.js"></script>
   <script type=" text/javascript" src ="scripts/leaflet/jquery.js">
</script>
   <script type = "application/json" src =


   <style>
   body {
   margin: 0px;
   font-family: "Open Sans";
   }
   #map{
   position: absolute; top:0px; bottom:0; width:100%; background-
color: f8f7f0;
   }
   .leaflet-label{
   line-height: 16px;
   }
   .disabled{
   cursor: default;
   pointer-events: none;
   opacity: .4;
   }

   </style>


   </head>
   <body style ="height: 100%;">    <div id="map"></div>
   <script >



    // map tile created here
   var mapTile = L.tileLayer ('map/{z}/{x}/{y}.png');
```

```
   standardLayerGroup = L.layerGroup([mapTile]);

   //Bound for the map created here
   var  southWest = L.latLng(8.75412, 39.02275),
   northEast = L.latLng (9.10074, 38.49678),
   bounds = L.latLngBounds(southWest, northEast);



   var stationMarker = L.Icon.extend({
   options: {
   iconUrl: 'data/ew_stations_icon.png',
   shadowUrl: 'data/shadow.png',
   iconSize:     [50, 40], // size of the icon
   shadowSize:   [10, 10], // size of the shadow
   iconAnchor:   [25, 40], // point of the icon which will
correspond to marker's location
   shadowAnchor: [0, 10],  // the same for the shadow
   popupAnchor:  [0, -40] // point from which the popup should open
relative to the iconAnchor
            }

            });


   var blueMarker = new stationMarker
({iconUrl:'data/sn_stations_icon.png'}),
   greenMarker = new stationMarker ({iconUrl:
'data/ew_stations_icon.png'}),
   commonStationMarker = new stationMarker({iconUrl:
'data/common_stations_icon.png'});

   //Array of sations name


   var markers =[
   // COMMON STATIONS i=0 to i=4


   // BlUE STATIONS i=5 to 21

   [38.763032,8.937983, "Kality<br>ቃሊቲ"],//i=21 Kality
   [38.766241,8.944995, "Abo Junction<br>አቦ መገንጠያ"],//i=20 Saris Abo
    [38.763379,8.951932, "Saris<br>ሳሪስ"],//i=19 Saris
    [38.763848,8.956766, "Adey Ababa<br>አደይ አበባ"],//i=18 Adey Abeba
    [38.762952,8.965691, "Nefas Silk 1<br>ነፋስ ስልክ 1"],//i=17 Nifas
Silk 1
    [38.760749,8.973094,"Nefas Silk 2<br>ነፋስ ስልክ 2"],//i=16 Nifas
Silk 2
    [38.759209,8.990339, "Lancha<br>ላንቻ"],//i=15 Lancha
   [38.759761,8.995205, "Temenja Yazh<br>ጠመንጃ ያዥ"],//i=14 Temenja
Yazh
   [38.75864,9.000533 , "Riche<br>ሪቼ"],//i=13 Riche
   [38.758827,9.004876, "Meshualekya<br>መሸለኪያ"],//i=12  Meshwalekya
```

```
    [38.757685,9.011996, "Stadium<br>ስቴዲየም"],//i=4 Stadium
    [38.753669,9.011863, "Lagare<br>ለገሀር"],//i=3 Lagare
    [38.748676,9.011074, "Mexico<br>ሜክሲኮ"],//i=2 Mexico
    [38.742552,9.010530, "Tegbared<br>ተግባረዕድ"],//i=1 Tegbared
    [38.735947,9.011398, "St. Lideta<br>ቅድስት ልደታ" ],//i=0 St. Lideta
   [38.734696, 9.015413, "Darmar<br>ዳርማር"],//i= 11 Darmar
   [38.733399, 9.021280, "Abnet<br>አብነት"],//i=10 Abnet
   [38.733030,9.028309, "Sebategna<br>ሰባተኛ"],//i=9 Sebategna
   [38.733031,9.034023, "Autobus Tera<br />አውቶቡስ ተራ" ],//i=8
Autobus Tera
   [38.738371,9.034190, "Gojam Berenda<br />ጎጃም በረንዳ"],//i=7 Gojam
Berenda
    [38.746908,9.034582,"Atkilt Tera<br />አትክልት ተራ" ],//i=6 Atkilt
Tera
   [38.752255,9.037427,"Menelik II Square<br />ሚንሊክ 2ኛ አደባባይ"],//i=5
Menelik II Square38.749655 9.035427

   //GREAN STATIONS i= 22 to markers.length

   [38.871784,9.021129, "አያት <br> Ayat"],//i=22 Ayat
   [38.860366,9.020585, "መሪ <br> Meri"],//i=23 Meri
   [38.850455,9.020964, "ሲ.ኤም.ሲ <br> CMC" ], //i=24 CMC
   [38.842595,9.021353,"St. Michael<br>ቅዱስ ሚካኤል"],//i=25 St.
Michael
   [38.834779,9.021678,"Civil Service College<br>ሲቪል ሰርቪስ ኮሌጅ"],//i=
26 Civil Service Collage
   [38.828364,9.020835, "Management Institute<br>ማኔጅመንት
ኢንስቲትዩት"],//i=27 Management Institute
   [38.819942,9.019058,"Gurd Shola 1<br>ጉርድ ሾላ 1"],//i=28 Gurd Shola
1
   [38.810144,9.020199,"Gurd Shola 2<br>ጉርድ ሾላ 2"],//i=29 Gurd Shola
2
   [38.802716,9.019610,"Megenagna<br>መገናኛ"],//i=30 Megenagna
   [38.795890,9.017881,"Lem Hotel<br>ለም ሆቴል"], //i=31 Lem Hotel
   [38.789336,9.016206,"Hayahulet 1<br>ሃያሁለት 1"],//i=32 Hayahulet 1
   [38.783242,9.014853,"Hayahulet 2<br>ሃያሁለት 2"],//i=33 Hayahulet 2
   [38.774723,9.010951,"St. Urael<br>ቅዱስ ኡራኤል"],//i=34 St. Ureal
   [38.768766,9.010356,"Bambis<br>ባምቢስ"],//i=35 Bambis
   [38.763430,9.010997, "St. Estifanos<br>ቅዱስ እስጢፋኖስ"],//i=36 St.
Estifano
   [38.757685,9.013696, "Stadium<br>ስቴዲየም"],//i=4 Stadium
   [38.753669,9.013563, "Lagare<br>ለገሀር"],//i=3 Lagare
   [38.748676,9.012574, "Mexico<br>ሜክሲኮ"],//i=2 Mexico
   [38.742552,9.012030, "Tegbared<br>ተግባረዕድ"],//i=1 Tegbared
   [38.735947,9.013098, "St. Lideta<br>ቅድስት ልደታ" ],//i=0
   [38.722877,9.011370,"Tor Hailoch<br/>ጦር ሃይሎች" ],//i= 37 Tor
Hailoch
   [38.729251,9.012023, "Coca Cola<br/>ኮካ ኮላ"],//i = 38 Coca Cola


   ];


   // function for choosing the right marker color
```

```
   function stationColorSet (i)
   {
   /*if (i>8 && i<14) { return commonStationMarker; }
      else if (i<9 ||  i>13 && i<22){ return blueMarker ;}
         else{ return greenMarker;}*/
         if (i<22){ return blueMarker;}
           else { return greenMarker;}
   }


   var StationObject = {}; // Array object for storing all station
markers
   var StationNames = [

"BG_0","BG_1","BG_2","BG_3","BG_4","BG_5","BG_6","BG_7","BG_8","BG_9
",

"B_5","B_6","B_7","B_8","B_9","B_10","B_11","B_12","B_13","B_14","B_
15","B_16","B_17","B_18","B_19","B_20","B_21",

"G_22","G_23","G_24","G_25","G_26","G_27","G_28","G_29","G_30","G_31
","G_32","G_33","G_34","G_35","G_36","G_37","G_38"

   ];


   var MarkerArray = [];

   for (var i=0;i<markers.length;i+=1)
    {

         var lon = markers[i][0];
         var lat = markers[i][1];
         var popupText = markers[i][2] + ("</br>ወደ ቃሊቲ
_____ ወደ ፒያሳ <br/> ክፍት ቦታ:"+latitude+" ____ክፍት ቦታ :  "+
latitude +"<br/>ቀሪ ጊዜ : " + latitude +"_____ቀሪ ጊዜ : "+latitude+" <br/>
ባቡር ጠባቂ:"+ latitude +"____ባቡር ጠባቂ:" + latitude + "<br/>
ወራጅ : "+latitude+" _____ወራጅ : " + latitude );
         var markerLocation = new L.LatLng(lat, lon);
         StationObject[StationNames[i]] = new
L.Marker(markerLocation, {icon: stationColorSet(i)});
         StationObject[StationNames[i]].bindPopup(popupText);


     }



   // change the object array into normal arrayes for layer group
use

   for (var i=0;i<markers.length;i+=1){
```

```
        MarkerArray[i] = StationObject[StationNames[i]];

    }

        // create Marker Layer
      var ew_gpx = 'east-west.gpx';
      var ns_gpx = 'north-south.gpx';

      var ew_LineLayer = new L.GPX( ew_gpx, {async: true
,polyline_options: {color: 'green', opacity: 1}});

     var ns_LineLayer = new L.GPX( ns_gpx, {async: true
,polyline_options: {color: 'blue', opacity: 1}});

     MarkerLayer = L.layerGroup(MarkerArray)//(MarkerArray)
                                          .addLayer(ns_LineLayer)
                                          .addLayer(ew_LineLayer);


    // creat Map

    var map = L.map('map', {
    center: [8.99665, 38.81573],
    zoom: 12,
    maxZoom: 16,
    minZoom: 11,
    maxBounds: bounds,
    attributionControl: false,
    layers: [standardLayerGroup, MarkerLayer]

    });


  ns_LineLayer.on('loaded', function(e)
{map.fitBounds(e.target.getBounds());})
  ew_LineLayer.on('loaded', function(e)
{map.fitBounds(e.target.getBounds());});



    // user and train icon creat
    var UserPulsingIcon = L.icon.pulse({
          iconSize:[12, 12],
          color:'red'
       });

    var BlueTrainIcon = L.icon.pulse({
          iconSize:[12, 12],
          color:'blue'
       });
    var GreenTrainIcon = L.icon.pulse({
          iconSize:[12, 12],
          color:'green'
       });
```

```
   var userLocation = L.marker([0, 0],{icon: UserPulsingIcon});
   var blueTrainLocation = L.marker([-100, -40],{icon:
BlueTrainIcon});
   var blueTrainLocation_2 = L.marker([-100, -40],{icon:
BlueTrainIcon});
   var blueTrainLocation_3 = L.marker([-100, -40],{icon:
BlueTrainIcon});
   var greenTrainLocation = L.marker([0, 0],{icon: GreenTrainIcon});

   //get user location

   function locateUser(){
         map.locate({watch: true});
         setTimeout( function(){
            locateUser();
         }, 2000);
      }


   // mark user location on location found
   function onLocationFound(e) {

        userLocation.setLatLng(e.latlng);
      userLocation.setOpacity( 1 );
        }

        userLocation.addTo(MarkerLayer).bindPopup("እዚህ አካባቢ ይገኛሉ <br>
You are around here ", {autoPan: false});
      userLocation.on('click', function () {
                               userLocation.openPopup(); });

        locateUser();
        map.on('locationfound', onLocationFound);


    // error in location not find
    function onLocationError(e) {
             userLocation.setOpacity( 0 );
              }


    map.on('locationerror', onLocationError);

   // A draggable marker icon


   var marker = L.marker ([9.03777, 38.87448],{
   draggable: true,
   }).addTo(map);
   marker.bindPopup('');

   marker.on('dragend', function(e){
   marker.getPopup().setContent('clicked' +
```

```
marker.getLatLng().toString()).openOn(map);
   });

    //get the lat. log. values of the trains from database
    var station_count ;
    var latitude;
    var longitude;
    var count;
    var station_index;
    var B_station =
["99","118","136","146","164","183","218","230","243","255","279","2
86","292","301","311","318","331","342","355","367","390","413"];

    var B_station_2 = ["1","13","23","50", "59",
"69","85","94","109","120","133","141","157","168","179","194","223"
,"237","250","260","273","288"] ;

    var train_ID = ["BT_1","BT_2","BT_3","BT_4","BT_5"];

   $(document).ready(function() {

       setTimeout(executeQ, 1000);

      });

      function executeQ(){


          $.ajax({
                type:"GET",
                url:"data.php",
               dataType:"json",


            success: function (msg,string,jqXHR){

               latitude = parseFloat(msg.latitude_1);
               longitude = parseFloat(msg.longitude_1);
               count_1 = parseInt(msg.count_1);
               count_2 = parseInt(msg.count_2);
               count_3 = parseInt(msg.count_3);
                    var trains_loc = [
                                        [parseFloat(msg.latitude_1),
parseFloat(msg.longitude_1)],
                                        [parseFloat(msg.latitude_2),
parseFloat(msg.longitude_2)],
                                        [parseFloat(msg.latitude_3),
parseFloat(msg.longitude_3)],

                                        ];

                   var newlatlng_2 = new
L.LatLng(msg.latitude_2,msg.longitude_2);
               blueTrainLocation_2.setLatLng(newlatlng_2);
```

```
                    blueTrainLocation_2.setOpacity(1);


blueTrainLocation_2.addTo(MarkerLayer).bindPopup('የባቡር መለያ (ID) :
Blue_train_2 </br> ዓማካይ ፍጥነት: 45 Km/hr <br/>ተሳፋሪ ብዛት : <br/> ነጠላ ባቡር
</br>');//+blueTrainLocation.getLatLng().toString(), {autoPan:
false});
                    blueTrainLocation_2.on('click', function () {
                    blueTrainLocation_2.openPopup(); });


                    var newlatlng = new L.LatLng(latitude,
longitude);
                blueTrainLocation.setLatLng(newlatlng);
                   blueTrainLocation.setOpacity(1);

blueTrainLocation.addTo(MarkerLayer).bindPopup('የባቡር መለያ (ID) :
Blue_train_1 </br> ዓማካይ ፍጥነት: 45 Km/hr <br/>ተሳፋሪ ብዛት : <br/> ነጠላ ባቡር
</br>');//+blueTrainLocation.getLatLng().toString(), {autoPan:
false});
                    blueTrainLocation.on('click', function () {
                    blueTrainLocation.openPopup(); });


                    var newlatlng_3 = new
L.LatLng(msg.latitude_3,msg.longitude_3);
                blueTrainLocation_3.setLatLng(newlatlng_3);
                   blueTrainLocation_3.setOpacity(1);

blueTrainLocation_3.addTo(MarkerLayer).bindPopup('የባቡር መለያ (ID) :
Blue_train_3 </br> ዓማካይ ፍጥነት: 45 Km/hr <br/>ተሳፋሪ ብዛት : <br/> ነጠላ ባቡር
</br>');//+blueTrainLocation.getLatLng().toString(), {autoPan:
false});
                    blueTrainLocation_3.on('click', function () {
                    blueTrainLocation_3.openPopup(); });

// -------------------------To put The location of all trains in an
array for easy access ----------------------------------
                    var array_up = [-1,-1,-1];
                    var array_down = [-1,-1,-1];

                    if (parseInt (msg.blue_1) == 0){

                    count = parseInt (msg.count_1);
                    station_count = B_station.find(checkStation);
                    array_up[0] = B_station.indexOf(station_count);


                    } else if( parseInt(msg.blue_1) == 1){

                    count = parseInt (msg.count_1) ;
                    station_count = B_station_2.find(checkStation);
                    array_down[0]= 21-
```

```
(B_station_2.indexOf(station_count));

                    }

                    if (parseInt (msg.blue_2) == 0){

                    count = parseInt (msg.count_2) ;
                    station_count = B_station.find(checkStation);
                    array_up[1] = B_station.indexOf(station_count);


                    }else if (parseInt (msg.blue_2) == 1){

                    count = parseInt (msg.count_2);
                    station_count = B_station_2.find(checkStation);
                    array_down[1]= 21 -
(B_station_2.indexOf(station_count));

                    }


                    if (parseInt (msg.blue_3) == 0){

                    count = parseInt (msg.count_3);
                    station_count = B_station.find(checkStation);
                    array_up[2] = B_station.indexOf(station_count);

                    }else if (parseInt (msg.blue_3) == 1){

                    count = parseInt (msg.count_3);
                    station_count = B_station_2.find(checkStation);
                    array_down[2]= 21-
(B_station_2.indexOf(station_count));


                    }

                    var orginal_array_up = array_up.slice();
                    var original_array_down = array_down.slice();

                    // sort them here
                    array_down.sort(function(a,b){return a-b});
                    array_up.sort(function(a,b){return b-a});

                    //station_count =
B_station.find(checkStation_up);
                    //var index = B_station.indexOf(station_count);
                    var r_time_1;
                    var r_time_2;
                    var r_time_3;
                    var j = 0;

                /* for (var i=0; i<22 ;i+=1) {
```

```
                    station_index = i;
                   var index_1 = array_up.find(checkStation_2) ;
                   var og_index = orginal_array_up.indexOf(index_1);
                  // var lat = trains_loc
[orginal_array_up.indexOf(index_1)][0];


StationObject[StationNames[i]].getPopup().setContent(array_up[0]+"<b
r/>"+array_up[1]+"<br/>"+array_up[2]+"<br/>"+array_down[0]+"<br/>"+a
rray_down[1]+"<br/>"+index_1+"<br/>"+ lat);

                } */
//-------------------------------------------------TO PIYASSA POPUP--
-----------------------------------------------------------//
            for (var i=0; i<22 ;i+=1)

                {
                  var to_kality;
                  var to_piyasa;
                    station_index = i;
                    var index_1 = array_up.find(checkStation_2) ;
                    var og_index =
orginal_array_up.indexOf(index_1);
                    var distn_1;

                 if (!(index_1== -1 )) {


                   if (og_index == 2) {

                     distn_1 = distance (trains_loc[2][0],
trains_loc[2][1], markers[i][1], markers[i][0]);


                   } else if (og_index == 1){
                     distn_1 = distance (trains_loc[1][0],
trains_loc[1][1], markers[i][1], markers[i][0]);


                    }else if (og_index == 0){

                   distn_1 = distance (trains_loc[0][0],
trains_loc[0][1], markers[i][1], markers[i][0]);

                       }


                var station_dis_1 = 0;

                for (var j = index_1; j<i; j+=1){
                   station_dis_1 = station_dis_1 + distance
(markers[j+1][1], markers[j+1][0], markers[j][1], markers[j][0]);
                            }
```

```
                    distn_1 = station_dis_1 + distn_1;
                    r_time_1= travel_time(distn_1) + (i-index_1)*30
+30;
                    r_time_1= to_minutes(r_time_1);

                    to_piyasa = "[ወደ ፒያሳ] <br/> የባቡር መለያ (ID) : "+
train_ID[og_index]+"</br> ከፍት ቦታ:" + index_1 +" <br/>ቀሪ ጊዜ : "+
r_time_1 + " ደቂቃ <br/> ባቡር ጠባቂ: <br/> ወራጅ : "+ index_1;


                } else if (index_1 == -1){

                   to_piyasa = "[ወደ ፒያሳ] <br/>ከቡራን ተጓዞች በዚህ መስመር
<br/>ለጊዜው እየመጣ ያለ ባቡር የለም <br/>Dear passangers, for the <br/>time
being there is no <br/> next train  on this line " ;

                }



//---------------------------------------TO KALIT STATION POPUP -----
-------------------------------------------------//
     var index_2 = array_down.find(checkStation_3) ;


     if (!(index_2 == null || index_2 == 22)) {

      var station_dis_2 = 0;
     var og_index = original_array_down.indexOf(index_2);
     distn_2 = distance (trains_loc [og_index][0], trains_loc
[og_index][1] , markers[index_2][1], markers[index_2][0]);


          for (var j = station_index; j<(index_2); j+=1){
               station_dis_2 = station_dis_2 + distance
(markers[j+1][1], markers[j+1][0], markers[j][1], markers[j][0]);
               }

                    distn_2 = station_dis_2 + distn_2;
                   r_time_2= travel_time(distn_2) + (index_2 -
station_index)*30 +30;
                    r_time_2= to_minutes(r_time_2);

                    to_kality = "[ ወደ ቃሊቲ ] <br/> የባቡር መለያ
(ID) : "+ train_ID[og_index] +"</br> ከፍት ቦታ:" + array_down[1] +"
<br/>ቀሪ ጊዜ : "+r_time_2 + " ደቂቃ <br/> ባቡር ጠባቂ: <br/> ወራጅ : "+
array_down[2]+"</br>"+array_down[0]+"</br>"+index_2;


                } else if (index_2 == null || index_2 == 22){

                    to_kality = "[ ወደ ቃሊቲ ] <br/> ከቡራን ተጓዞች በዚህ
መስመር <br/>ለጊዜው እየመጣ ያለ ባቡር የለም <br/>Dear passangers, for the
```

```
<br/>time being there is no <br/> next train  on this line ";
                    }

                    // to_kality = "jojo" ;

                    popupTex = markers[i][2] + ("<table> <tr> <td>" +
to_kality + "</td> <td>" + to_piyasa + "</td></tr></table>");


StationObject[StationNames[i]].getPopup().setContent(popupTex);


                    }

             }


              });

                 setTimeout(executeQ, 2000);
               // setTimeout(station_display, 10000);



             }


function checkStation_2(station) {

   return station <= station_index;
}

function checkStation_3(station) {

   return station >= station_index;
}


function checkStation(station) {

   return station >= count;
}


function travel_time (distance)

{

var time = distance / 12.5 ;
return time;

}
```

```
function to_minutes (time)
{
var time_between_two_points = ~~((time % 3600)/ 60); // ($distance /
12.5);
return time_between_two_points ;

}


// This a distance calculator function

function distance (lat1, lon1, lat2, lon2) {

   var radlat1 = Math.PI * lat1/180
   var radlat2 = Math.PI * lat2/180
   var theta = lon1-lon2
   var radtheta = Math.PI * theta/180
   var dist = Math.sin(radlat1) * Math.sin(radlat2) +
Math.cos(radlat1) * Math.cos(radlat2) * Math.cos(radtheta);
   dist = Math.acos(dist)
   dist = dist * 180/Math.PI
   dist = dist * 60 * 1.1515*1000;
   return dist
}


   </script>

   </body>
   </html>
```

B. AALRT Info App Main Activity Android Java Code

```
package com.aalrt_pis.yetages.aalrt;
import android.content.Context;
import android.util.Log;
import fi.iki.elonen.NanoHTTPD;
import java.io.IOException;
import java.io.InputStream;

import java.net.HttpURLConnection;
import java.net.URL;
import android.os.Bundle;
import android.net.Uri;
import android.support.customtabs.CustomTabsIntent;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ImageButton;


public class MainActivity extends AppCompatActivity {
ImageButton  button ;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = (ImageButton) findViewById(R.id.map_icon);
        MyHTTPD server = null;
        try {
            server = new MyHTTPD(getApplicationContext());
            try
            {
                server.start();
            }
            catch( IOException ioe )
            {
                System.err.println( "Couldn't start server:\n" + ioe
);
                System.exit( -1 );
            }
            System.out.println( "Listening on port 8080. Hit Enter
to stop.\n" );
            try { System.in.read(); } catch( Throwable t ) {
                System.out.println("read error");
            };
        } catch (IOException e1) {
            // TODO Auto-generated catch block
```

```
            e1.printStackTrace();
        }
        /*
        try {
            (new WebServer()).start();
        } catch(IOException ioe) {
            Log.w("Httpd", "The server could not start.");
        }
        Log.w("Httpd", "Web server initialized."); */
    }


    public void onTouch (View view)
    {
        String url =
"http://localhost:8080/index.html";//"http://192.168.43.54/folder/in
dex.html"; //
        CustomTabsIntent.Builder builder = new
CustomTabsIntent.Builder();
        CustomTabsIntent customTabsIntent = builder.build();
        customTabsIntent.launchUrl(this, Uri.parse(url));


    }

    private final static int PORT = 8080;
    private final static String TAG="application";



    public Context ctx = null;
    /**
     * Constructs an HTTP server on given port.
     */
   public class MyHTTPD extends NanoHTTPD {

        public MyHTTPD(Context ctx) throws IOException {
            super(PORT);

        }
        public static final String MIME_JAVASCRIPT =
"text/javascript";
        public static final String MIME_CSS = "text/css";
        public static final String MIME_PNG = "image/png";
        public static final String MIME_JSON = "application/json";

       @Override
```

```
        public Response serve(IHTTPSession session){
            String mime_type = NanoHTTPD.MIME_HTML;
            Method method = session.getMethod();
            String uri = session.getUri();
            System.out.println(method + " '" + uri + "' ");
            InputStream descriptor = null;
            if(method.toString().equalsIgnoreCase("GET")){
                String path;
                if(uri.equals("/")){
                    path="/index.html";
                }else{
                    path = uri;
                    try{
                        if(path.endsWith(".js")){
                            mime_type = MIME_JAVASCRIPT;
                        }else if(path.endsWith(".css")){
                            mime_type = MIME_CSS;
                        }else if(path.endsWith(".html")){
                            mime_type = MIME_HTML;
                        }else if(path.endsWith(".png")){
                            mime_type = MIME_PNG;
                        }else if(path.endsWith(".php")){
                            mime_type = MIME_JSON;
                        }
                    }catch(Exception e){

                    }
                }
                try {
                    // Open file from SD Card
                    if (mime_type == MIME_JSON){


                       try {
                            HttpURLConnection urlConnection =
(HttpURLConnection) new
URL("http://192.168.43.54/folder/data.php").openConnection();
                            urlConnection.setRequestMethod("POST");
                            urlConnection.setDoInput(true);
                            urlConnection.connect();
                            descriptor
=urlConnection.getInputStream();

                        } catch (IOException e) {
                            e.printStackTrace();
                        }
```

```
                    } else {

                                descriptor =
getAssets().open("www"+path);}

                } catch(IOException ioe) {
                    Log.w("Httpd", ioe.toString());
                }
            }
             NanoHTTPD.Response res = new NanoHTTPD.Response(
Response.Status.OK,mime_type,descriptor);
             return res;
        }
    }

}
```

C. Server Side Files

C.1 PHP File for linking The Client with The Server

```
<?php

require_once 'init.php';

$blue_1 = 1;

$blue_2 = 1;

$blue_3 = 1;

$query = "SELECT * FROM all_train_information.blue_train_1 WHERE id
= (SELECT max(id) FROM all_train_information.blue_train_1)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed: ". mysql_error());

      $row_1 = mysql_fetch_row($result);

            $lat_1 = $row_1[1];

            $lon_1 = $row_1[2];

$query = "SELECT * FROM train_route.northsouth_route2 WHERE ID =
(SELECT id FROM train_route.northsouth_route2 WHERE latitude =
$lat_1 AND longitude = $lon_1)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed 4: ". mysql_error());

    $row_t1 = mysql_fetch_row($result);

   if ($row_t1[0] == null) {

$query = "SELECT * FROM train_route.northsouth_route WHERE ID =
(SELECT id FROM train_route.northsouth_route WHERE latitude = $lat_1
AND longitude = $lon_1)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed 4: ". mysql_error());

    $row_t1 = mysql_fetch_row($result);

    $blue_1 = 0;

   }

////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
```

```
$query = "SELECT * FROM all_train_information.blue_train_2 WHERE id
= (SELECT max(id) FROM all_train_information.blue_train_2)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed: ". mysql_error());

    $row_2 = mysql_fetch_row($result);

          $lat1_1 = $row_2[1];

          $lon1_1 = $row_2[2];

$query = "SELECT * FROM train_route.northsouth_route2 WHERE ID =
(SELECT id FROM train_route.northsouth_route2 WHERE latitude =
$lat1_1 AND longitude = $lon1_1)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed 4: ". mysql_error());

    $row_t2= mysql_fetch_row($result);

        if ($row_t2[1] == null) {


$query = "SELECT * FROM train_route.northsouth_route WHERE ID =
(SELECT id FROM train_route.northsouth_route WHERE latitude =
$lat1_1 AND longitude = $lon1_1)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed 4: ". mysql_error());

    $row_t2 = mysql_fetch_row($result);

    $blue_2 = 0;

   }

//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////


$query = "SELECT * FROM all_train_information.blue_train_3 WHERE id
= (SELECT max(id) FROM all_train_information.blue_train_3)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed: ". mysql_error());

    $row_3 = mysql_fetch_row($result);

          $lat3 = $row_3[1];

          $lon3 = $row_3[2];
```

```
$query = "SELECT * FROM train_route.northsouth_route2 WHERE ID =
(SELECT id FROM train_route.northsouth_route2 WHERE latitude = $lat3
AND longitude = $lon3)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed 4: ". mysql_error());

    $row_t3= mysql_fetch_row($result);


if ($row_t3[1] == null) {


$query = "SELECT * FROM train_route.northsouth_route WHERE ID =
(SELECT id FROM train_route.northsouth_route WHERE latitude = $lat3
AND longitude = $lon3)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed 4: ". mysql_error());

    $row_t3 = mysql_fetch_row($result);

    $blue_3 = 0;

   }


$test = array('latitude_1'=> $row_t1[1], 'longitude_1' =>
$row_t1[2], 'count_1'=> $row_t1[0], 'latitude_2'=>
$row_t2[1],'longitude_2' => $row_t2[2], 'count_2'=> $row_t2[0],
'latitude_3'=> $row_t3[1],'longitude_3' => $row_t3[2], 'count_3'=>
$row_t3[0] ,'blue_1' => $blue_1,'blue_2' => $blue_2 ,'blue_3' =>
$blue_3 );

echo json_encode($test );

?>
```

C.2 PHP File Used in Windows Task Manager to Simulate Trains

```
<?php

require_once 'init.php';

date_default_timezone_set("Africa/Addis_Ababa");

$lat_1 = 8.93632984;

$lon_1 = 38.76506424;

$B_station = array
("99","118","136","146","164","183","218","230","243","255","279","2
86","292","301","311","318","331","342","355","367","390","413");

$G_station = array ("1","15","30","41","51","60", "72","83","92",
"101","110","118","132","141",
"149","157","162","171","181","192","203","217");

$B_station_2 = array ("1","13","23","50","60",
"69","85","97","110","122","133","141","157","168","179","194","223"
,"237","250","260","273","288") ;

// for blue train

$query = "SELECT * FROM all_train_information.blue_train_2 WHERE id
= (SELECT max(id) FROM all_train_information.blue_train_2)";

$result = mysql_query($query);

if(!$result) die ("database access failed 3: ". mysql_error());

$row = mysql_fetch_row($result);

        $latt_1 = $row[1];

        $lonn_1 = $row[2];

       $current_time = strtotime (date('Y/m/d H:i:s'));

        $last_logged_time =strtotime(date_format(new
DateTime($row[4]), 'Y/m/d H:i:s'));

        $time_diff_check = $current_time - $last_logged_time ;


$query = "SELECT * FROM train_route.northsouth_route WHERE ID =
(SELECT id FROM train_route.northsouth_route WHERE latitude =
$latt_1 AND longitude = $lonn_1)";

$result = mysql_query($query);

if(!$result) die ("database access failed 4: ". mysql_error());
```

```
$row_up = mysql_fetch_row($result);

$query = "SELECT * FROM train_route.northsouth_route2 WHERE ID =
(SELECT id FROM train_route.northsouth_route2 WHERE latitude =
$latt_1 AND longitude = $lonn_1)";

$result = mysql_query($query);

if(!$result) die ("database access failed 4: ". mysql_error());

$row_down = mysql_fetch_row($result);

if ($latt_1 == 9.03729916 && $lonn_1 ==38.75315475 &&
$time_diff_check >120) {

    $lat_2 = 9.03753281;

    $lon_2 = 38.75169754;

    $query = "INSERT INTO all_train_information.blue_train_2 VALUES
(NULL, $lat_2, $lon_2, 0, now())";

    $result = mysql_query($query);

    if (!$result) die ("Database access failed 2: " .
mysql_error());

}

if ((($latt_1 == 8.93556690 && $lonn_1 ==38.76395035) || $row_up ==
null )&& $time_diff_check > 120) {

   $lat_2 = 8.93662643;

   $lon_2 = 38.76506424;

   $query = "INSERT INTO all_train_information.blue_train_2 VALUES
(NULL, $lat_2, $lon_2, 0, now())";

   $result = mysql_query($query);

   if (!$result) die ("Database access failed 2: " . mysql_error());

}


//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////

if (!($row_up == null) AND !($latt_1 == 9.03729916 && $lonn_1 ==
38.75315475 ))

{

$query = "SELECT * FROM all_train_information.blue_train_2 WHERE id
= (SELECT max(id) FROM all_train_information.blue_train_2)";
```

```
$result = mysql_query($query);

if(!$result) die ("database access failed 3: ". mysql_error());

$row = mysql_fetch_row($result);

        $lat_1 = $row[1];

        $lon_1 = $row[2];

$last_logged_time =strtotime(date_format(new DateTime($row[4]),
'Y/m/d H:i:s'));

        //echo $lat_1;

if (($lat_1==null) ){


  $lat_2 = 8.93662643;

  $lon_2 = 38.76506424;

  $query = "INSERT INTO all_train_information.blue_train_2 VALUES
(NULL, $lat_2, $lon_2, 0, now())";

  $result = mysql_query($query);

    if (!$result) die ("Database access failed 2: " .
mysql_error());


} else {

    $query = "SELECT * FROM train_route.northsouth_route WHERE ID =
1 + (SELECT id FROM train_route.northsouth_route WHERE latitude =
$lat_1 AND longitude = $lon_1)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed 4: ". mysql_error());

    $row = mysql_fetch_row($result);

    $lat_2 = $row[1];

    $lon_2 = $row[2];

    $travel_distance = distance($lat_1,$lon_1,$lat_2,$lon_2);

    $time_between_two_points = travel_time($travel_distance);

    $current_time = strtotime (date('Y/m/d H:i:s'));

    $time_diff = $current_time - $last_logged_time ;

 echo $time_diff.'<br/>';
```

```
 echo $time_between_two_points. '<br/>';



if ($time_diff > $time_between_two_points) {
      $count = $row[0];
if(!in_array($count, $B_station))  {
while ($time_diff > $time_between_two_points && $count < 412)
     {
       $query = "SELECT * FROM train_route.northsouth_route WHERE id
=$count";
       $result = mysql_query($query);
       if(!$result) die ("database access failed 7: ".
mysql_error());
       $row = mysql_fetch_row($result);
       $lat_2 = $row[1];
       $lon_2 = $row[2];
       echo $count;
       if (in_array($count, $B_station))
    {
       $query = "SELECT * FROM train_route.northsouth_route WHERE id
=$count-1";
       $result = mysql_query($query);
       if(!$result) die ("database access failed 7: ".
mysql_error());
       $row = mysql_fetch_row($result);
       $lat_2 = $row[1];
       $lon_2 = $row[2];
       break; }
       $travel_distance = distance($lat_1,$lon_1,$lat_2,$lon_2);
       $time_between_two_points = travel_time ($travel_distance);
       $count++;
          }
```

```
$query = "INSERT INTO all_train_information.blue_train_2
VALUES(NULL, $lat_2, $lon_2, 0, now())";

         $result = mysql_query($query);

         if (!$result) die ("Database access failed 2: " .
mysql_error());


} else if ( in_array($count, $B_station)){

 if($time_diff >=30 ){

    $query = "INSERT INTO all_train_information.blue_train_2
VALUES(NULL, $lat_2, $lon_2, 0, now())";

    $result = mysql_query($query);

    if (!$result) die ("Database access failed 2: " .
mysql_error());

                }

            }

         } //if time is greater than

      }//else


}

////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

elseif ( !($row_down == null) AND !($latt_1 == 8.93556690 && $lonn_1
==38.76395035)) {

$query = "SELECT * FROM all_train_information.blue_train_2 WHERE id
= (SELECT max(id) FROM all_train_information.blue_train_2)";

$result = mysql_query($query);

if(!$result) die ("database access failed 3: ". mysql_error());

$row = mysql_fetch_row($result);

        $lat_1 = $row[1];

        $lon_1 = $row[2];

$last_logged_time =strtotime(date_format(new DateTime($row[4]),
'Y/m/d H:i:s'));
```

```
$query = "SELECT * FROM train_route.northsouth_route2 WHERE ID = 1 +
(SELECT id FROM train_route.northsouth_route2 WHERE latitude =
$lat_1 AND longitude = $lon_1)";

    $result = mysql_query($query);

    if(!$result) die ("database access failed 4: ". mysql_error());

    $row = mysql_fetch_row($result);

    $lat_2 = $row[1];

    $lon_2 = $row[2];

    $travel_distance = distance($lat_1,$lon_1,$lat_2,$lon_2);

    $time_between_two_points = travel_time($travel_distance);

    $current_time = strtotime (date('Y/m/d H:i:s'));

    $time_diff = $current_time - $last_logged_time ;

    echo $time_diff.'<br/>';

    echo $time_between_two_points. '<br/>';

if ($time_diff > $time_between_two_points) {

      $count = $row[0];

if(!in_array($count, $B_station_2))  {

while ($time_diff > $time_between_two_points && $count < 294)

     {

       $query = "SELECT * FROM train_route.northsouth_route2 WHERE
id =$count";

       $result = mysql_query($query);

       if(!$result) die ("database access failed 7: ".
mysql_error());

       $row = mysql_fetch_row($result);

       $lat_2 = $row[1];

       $lon_2 = $row[2];

       //$count = $row[0];

       echo $count. "<br/>";

       if (in_array($count, $B_station_2))

        {
```

```
        $query = "SELECT * FROM train_route.northsouth_route2 WHERE
id =$count-1";

        $result = mysql_query($query);

        if(!$result) die ("database access failed 7: ".
mysql_error());

        $row = mysql_fetch_row($result);

        $lat_2 = $row[1];

       $lon_2 = $row[2];

        break; }


        $travel_distance = distance($lat_1,$lon_1,$lat_2,$lon_2);

        $time_between_two_points = travel_time ($travel_distance);

        $count++;

        }

$query = "INSERT INTO all_train_information.blue_train_2
VALUES(NULL, $lat_2, $lon_2, 0, now())";

          $result = mysql_query($query);

          if (!$result) die ("Database access failed 2: " .
mysql_error());


} else if ( in_array($count, $B_station_2)){

if($time_diff >=30 ){

    $query = "INSERT INTO all_train_information.blue_train_2
VALUES(NULL, $lat_2, $lon_2, 0, now())";

    $result = mysql_query($query);

    if (!$result) die ("Database access failed 2: " .
mysql_error());

                }

            }

        } //if time is greater than

}
```

```
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////
function travel_time ($distance)
{
$time_between_two_points = round ($distance / 12.5);
return $time_between_two_points ;
}
// This a distance caculator function


function distance ($lat1, $lon1, $lat2, $lon2) {
$theta = $lon1-$lon2;
$dist = sin(deg2rad($lat1))*sin(deg2rad($lat2)) +
cos(deg2rad($lat1))*cos(deg2rad($lat2))*cos(deg2rad($theta));
$dist = acos($dist);
$dist = rad2deg($dist);
$dist_KM = $dist*60*1.1515*1.609344*1000;
return $dist_KM;
}
//  train movement function
function estimated_time_for_arrival (){
}
?>
```

C.3 Windows Task Manger's Task File Used to Simulate a Train

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2"
xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
    <RegistrationInfo>
        <Date>2017-04-24T20:05:44.4424595</Date>
        <Author>DESKTOP-SVD3TSB\yetages</Author>
        <URI>\GPS_1</URI>
    </RegistrationInfo>
    <Triggers>
        <CalendarTrigger>
            <Repetition>
                <Interval>PT1M</Interval>
                <Duration>P1D</Duration>
                <StopAtDurationEnd>false</StopAtDurationEnd>
            </Repetition>
            <StartBoundary>2017-05-12T17:20:00</StartBoundary>
            <Enabled>true</Enabled>
            <ScheduleByDay>
                <DaysInterval>1</DaysInterval>
            </ScheduleByDay>
        </CalendarTrigger>
        <CalendarTrigger>
            <Repetition>
                <Interval>PT1M</Interval>
                <Duration>P1D</Duration>
                <StopAtDurationEnd>false</StopAtDurationEnd>
            </Repetition>
            <StartBoundary>2017-05-12T17:20:03</StartBoundary>
            <Enabled>true</Enabled>
            <ScheduleByDay>
                <DaysInterval>1</DaysInterval>
            </ScheduleByDay>
        </CalendarTrigger>
        <CalendarTrigger>
            <Repetition>
                <Interval>PT1M</Interval>
                <Duration>P1D</Duration>
                <StopAtDurationEnd>false</StopAtDurationEnd>
            </Repetition>
            <StartBoundary>2017-05-12T17:20:06</StartBoundary>
            <Enabled>true</Enabled>
            <ScheduleByDay>
                <DaysInterval>1</DaysInterval>
            </ScheduleByDay>
        </CalendarTrigger>
```

```
<CalendarTrigger>
    <Repetition>
        <Interval>PT1M</Interval>
        <Duration>P1D</Duration>
        <StopAtDurationEnd>false</StopAtDurationEnd>
    </Repetition>
    <StartBoundary>2017-05-12T17:20:09</StartBoundary>
    <Enabled>true</Enabled>
    <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
    </ScheduleByDay>
</CalendarTrigger>
<CalendarTrigger>
    <Repetition>
        <Interval>PT1M</Interval>
        <Duration>P1D</Duration>
        <StopAtDurationEnd>false</StopAtDurationEnd>
    </Repetition>
    <StartBoundary>2017-05-12T17:20:12</StartBoundary>
    <Enabled>true</Enabled>
    <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
    </ScheduleByDay>
</CalendarTrigger>
<CalendarTrigger>
    <Repetition>
        <Interval>PT1M</Interval>
        <Duration>P1D</Duration>
        <StopAtDurationEnd>false</StopAtDurationEnd>
    </Repetition>
    <StartBoundary>2017-05-12T17:20:15</StartBoundary>
    <Enabled>true</Enabled>
    <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
    </ScheduleByDay>
</CalendarTrigger>
<CalendarTrigger>
    <Repetition>
        <Interval>PT1M</Interval>
        <Duration>P1D</Duration>
        <StopAtDurationEnd>false</StopAtDurationEnd>
    </Repetition>
    <StartBoundary>2017-05-12T17:20:18</StartBoundary>
    <Enabled>true</Enabled>
    <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
    </ScheduleByDay>
```

```
</CalendarTrigger>
<CalendarTrigger>
    <Repetition>
        <Interval>PT1M</Interval>
        <Duration>P1D</Duration>
        <StopAtDurationEnd>false</StopAtDurationEnd>
    </Repetition>
    <StartBoundary>2017-05-12T17:20:21</StartBoundary>
    <Enabled>true</Enabled>
    <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
    </ScheduleByDay>
</CalendarTrigger>
<CalendarTrigger>
    <Repetition>
        <Interval>PT1M</Interval>
        <Duration>P1D</Duration>
        <StopAtDurationEnd>false</StopAtDurationEnd>
    </Repetition>
    <StartBoundary>2017-05-12T17:20:24</StartBoundary>
    <Enabled>true</Enabled>
    <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
    </ScheduleByDay>
</CalendarTrigger>
<CalendarTrigger>
    <Repetition>
        <Interval>PT1M</Interval>
        <Duration>P1D</Duration>
        <StopAtDurationEnd>false</StopAtDurationEnd>
    </Repetition>
    <StartBoundary>2017-05-12T17:20:27</StartBoundary>
    <Enabled>true</Enabled>
    <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
    </ScheduleByDay>
</CalendarTrigger>
<CalendarTrigger>
    <Repetition>
        <Interval>PT1M</Interval>
        <Duration>P1D</Duration>
        <StopAtDurationEnd>true</StopAtDurationEnd>
    </Repetition>
    <StartBoundary>2017-05-12T17:20:30</StartBoundary>
    <Enabled>true</Enabled>
    <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
```

```
        </ScheduleByDay>
    </CalendarTrigger>
    <CalendarTrigger>
        <Repetition>
            <Interval>PT1M</Interval>
            <Duration>P1D</Duration>
            <StopAtDurationEnd>false</StopAtDurationEnd>
        </Repetition>
        <StartBoundary>2017-05-12T17:20:33</StartBoundary>
        <Enabled>true</Enabled>
        <ScheduleByDay>
            <DaysInterval>1</DaysInterval>
        </ScheduleByDay>
    </CalendarTrigger>
    <CalendarTrigger>
        <Repetition>
            <Interval>PT1M</Interval>
            <Duration>P1D</Duration>
            <StopAtDurationEnd>false</StopAtDurationEnd>
        </Repetition>
        <StartBoundary>2017-05-12T17:20:33</StartBoundary>
        <Enabled>true</Enabled>
        <ScheduleByDay>
            <DaysInterval>1</DaysInterval>
        </ScheduleByDay>
    </CalendarTrigger>
    <CalendarTrigger>
        <Repetition>
            <Interval>PT1M</Interval>
            <Duration>P1D</Duration>
            <StopAtDurationEnd>false</StopAtDurationEnd>
        </Repetition>
        <StartBoundary>2017-05-12T17:20:36</StartBoundary>
        <Enabled>true</Enabled>
        <ScheduleByDay>
            <DaysInterval>1</DaysInterval>
        </ScheduleByDay>
    </CalendarTrigger>
    <CalendarTrigger>
        <Repetition>
            <Interval>PT1M</Interval>
            <Duration>P1D</Duration>
            <StopAtDurationEnd>false</StopAtDurationEnd>
        </Repetition>
        <StartBoundary>2017-05-12T17:20:36</StartBoundary>
        <Enabled>true</Enabled>
        <ScheduleByDay>
```

```
            <DaysInterval>1</DaysInterval>
        </ScheduleByDay>
    </CalendarTrigger>
    <CalendarTrigger>
        <Repetition>
            <Interval>PT1M</Interval>
            <Duration>P1D</Duration>
            <StopAtDurationEnd>false</StopAtDurationEnd>
        </Repetition>
        <StartBoundary>2017-05-12T17:20:39</StartBoundary>
        <Enabled>true</Enabled>
        <ScheduleByDay>
            <DaysInterval>1</DaysInterval>
        </ScheduleByDay>
    </CalendarTrigger>
    <CalendarTrigger>
        <Repetition>
            <Interval>PT1M</Interval>
            <Duration>P1D</Duration>
            <StopAtDurationEnd>false</StopAtDurationEnd>
        </Repetition>
        <StartBoundary>2017-05-12T17:20:42</StartBoundary>
        <Enabled>true</Enabled>
        <ScheduleByDay>
            <DaysInterval>1</DaysInterval>
        </ScheduleByDay>
    </CalendarTrigger>
    <CalendarTrigger>
        <Repetition>
            <Interval>PT1M</Interval>
            <Duration>P1D</Duration>
            <StopAtDurationEnd>false</StopAtDurationEnd>
        </Repetition>
        <StartBoundary>2017-05-12T17:20:45</StartBoundary>
        <Enabled>true</Enabled>
        <ScheduleByDay>
            <DaysInterval>1</DaysInterval>
        </ScheduleByDay>
    </CalendarTrigger>
    <CalendarTrigger>
        <Repetition>
            <Interval>PT1M</Interval>
            <Duration>P1D</Duration>
            <StopAtDurationEnd>false</StopAtDurationEnd>
        </Repetition>
        <StartBoundary>2017-05-12T17:20:48</StartBoundary>
        <Enabled>true</Enabled>
```

```
            <ScheduleByDay>
                <DaysInterval>1</DaysInterval>
            </ScheduleByDay>
        </CalendarTrigger>
        <CalendarTrigger>
            <Repetition>
                <Interval>PT1M</Interval>
                <Duration>P1D</Duration>
                <StopAtDurationEnd>false</StopAtDurationEnd>
            </Repetition>
            <StartBoundary>2017-05-12T17:20:51</StartBoundary>
            <Enabled>true</Enabled>
            <ScheduleByDay>
                <DaysInterval>1</DaysInterval>
            </ScheduleByDay>
        </CalendarTrigger>
        <CalendarTrigger>
            <Repetition>
                <Interval>PT1M</Interval>
                <Duration>P1D</Duration>
                <StopAtDurationEnd>false</StopAtDurationEnd>
            </Repetition>
            <StartBoundary>2017-05-12T17:20:54</StartBoundary>
            <Enabled>true</Enabled>
            <ScheduleByDay>
                <DaysInterval>1</DaysInterval>
            </ScheduleByDay>
        </CalendarTrigger>
        <CalendarTrigger>
            <Repetition>
                <Interval>PT1M</Interval>
                <Duration>P1D</Duration>
                <StopAtDurationEnd>false</StopAtDurationEnd>
            </Repetition>
            <StartBoundary>2017-05-12T17:20:57</StartBoundary>
            <Enabled>true</Enabled>
            <ScheduleByDay>
                <DaysInterval>1</DaysInterval>
            </ScheduleByDay>
        </CalendarTrigger>
    </Triggers>
    <Principals>
        <Principal id="Author">
            <UserId>S-1-5-18</UserId>
            <RunLevel>LeastPrivilege</RunLevel>
        </Principal>
    </Principals>
```

```
    <Settings>
        <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>

<DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
        <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
        <AllowHardTerminate>true</AllowHardTerminate>
        <StartWhenAvailable>false</StartWhenAvailable>
        <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
        <IdleSettings>
            <StopOnIdleEnd>true</StopOnIdleEnd>
            <RestartOnIdle>false</RestartOnIdle>
        </IdleSettings>
        <AllowStartOnDemand>true</AllowStartOnDemand>
        <Enabled>true</Enabled>
        <Hidden>false</Hidden>
        <RunOnlyIfIdle>false</RunOnlyIfIdle>
        <WakeToRun>false</WakeToRun>
        <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
        <Priority>7</Priority>
    </Settings>
    <Actions Context="Author">
        <Exec>
            <Command>"C:\scripts\cronejobs - Copy.bat"</Command>
        </Exec>
    </Actions>
</Task>
```