

**ENHANCEMENT OF ONLINE TOLL TAX
DATABASE MANAGEMENT MODEL USING
PUSH DOWN AUTOMATA APPROACH**

A Thesis Submitted

In Partial Fulfillment of Requirements

for the Degree of

MASTER OF TECHNOLOGY

in

SOFTWARE ENGINEERING

by

RAVI PRATAP SINGH

(1140449011)

Under the Supervision of

Dr. V. K. Singh

Professor, Dept. of IT, BBDNITM

to the

SCHOOL OF ENGINEERING

BABU BANARASI DAS UNIVERSITY

LUCKNOW

May, 2016

CERTIFICATE

It is certified that the work contained in the thesis entitled “**Enhancement of Online Toll Tax Database Management Model using Push Down Automata approach**”, that is being submitted by **Mr. Ravi Pratap Singh** (Roll No. 1140449011), in partial fulfillment for the award of **Master of Technology** in Computer Science (Specialization: Software Engineering) from Babu Banarasi Das University has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Name: Dr. V. K. Singh
Professor
Department of IT,
BBDNITM, Lucknow.
UP, India.

Date:

ABSTRACT

We highlights the use of Push-Down Automata in maintaining the vehicle records and provide the clustered view of them to make comparative analysis easier and faster. We focus to provide a communicative framework that can record the vehicles coming from a particular state. The Object Constraint Language (OCL) is being applied on Object Oriented Language so that the communication framework can be represented into Object Oriented Language.

Tolling as a method of financing the transportation system is becoming more common day by day. Neither the traveling public nor State Departments of Transportation wants the vehicles to slow down or stopped to pay to use a toll facility. In this project we highlight the use of Push-Down Automata (PDA) in sorting and maintaining the vehicle records coming for which state, calculate the total revenue generated and showing this information into cluster view of vehicle logs. Our objective is to develop a more interactive and communicative framework that can maintain a record of the vehicles coming from a particular state. Since Object Constraint Language is being applied on object-oriented language we will be representing the entire communication work into the Object-Oriented Language.

In order to achieve this goal we have to identify these basic steps, the first step deals with the development of a ubiquitous computing environment as the vehicle coming from. Secondly, development of push down automata model for vehicle logs database management system. The proposed work of this research work is to highlight the use of PDA in sorting and maintaining record and focus is to provide a communicative framework that can record vehicles log. For this, we will be developing a ubiquitous computing environment and development of PDA model for database management system to verify the model and do the performance analysis.

ACKNOWLEDGMENT

I would like to express my thanks and gratitude to Dr. V. K. Singh, Information Technology Department, BBDNITM, Lucknow for offering much-appreciated advice, support and thought-provoking ideas throughout in carrying out the project. It is solely his motivation that has driven me in efforts. I would like to express my sincere thanks to all our colleagues whose periodic inputs and informal discussions have given me a better perspective of my progress.

Student Name: Ravi Pratap Singh

University Roll No: 1140449011

TABLE OF CONTENTS

	Page No.
Certificate	ii
Abstract	iii
Acknowledgement	iv
List of Tables	ix
List of Figures	x
List of Symbols and Abbreviations	xii
CHAPTER 1: INTRODUCTION	1-3
1.1 INTRODUCTION	1
1.2 MOTIVATION	1
1.3 FOCUS	2
1.4 THESIS ORGANIZATION	3
CHAPTER 2: PRESENT SCENARIO OF OTTMS	4-16
2.1 TOLLING PRACTICES	4
2.1.1 Objectives of Tolling	4
2.1.2 Evolution of Tolling	4
2.2 CLASSIFICATION OF TOLLING PRACTICES	6
2.2.1 Manual Tolling	6
2.2.2 Electronic Toll Collection	7
2.2.2.1 Components of ETC	8
2.2.2.2 Enhancements	8
2.2.2.3 Benefits and Cost of ETC	9
2.3 UPCOMING TOLLING TECHNOLOGIES	11
2.3.1 Odometer Tolling	12

2.3.2	Cell Phone Tolling	14
2.3.3	Satellite Tolling	14
CHAPTER 3:	OBJECTIVE OF RESEARCH WORK	17-19
3.1	PROBLEM STATEMENT	17
3.2	CONSTRAINT	17
3.3	OBJECTIVE	18
CHAPTER 4:	LITERATURE REVIEW	20-33
4.1	RESEARCH AND BACKGROUND	20
4.2	COMPARSION CHART	22
4.3	STUDY OF COMPARSION CHART	23
CHAPTER 5:	METHODOLOGY USED	34-48
5.1	TWO-STACK PUSH DOWN AUTOMATA	34
5.1.1	Problem in normal PDA and Variations	36
5.1.2	Need of Two-Stack PDA	37
5.1.3	Components of Two-Stack PDA	37
5.2	REAL TIME CONSTRAINT NOTATION	40
5.2.1	Introduction of RTCN	40
5.2.2	Object Oriented Real Time Modeling	42
5.2.3	Characteristics	43
5.3	OBJECT CONSTRAINT LANGUAGE	45
5.3.1	Introduction of OCL	45
5.3.2	UML and OCL	46
5.3.3	Characteristics	47
CHAPTER 6:	PROPOSED WORK	49-56
6.1	FLOW CHART OF PROPOSED PDA MODEL	49

6.2	ALGORITHM	51
6.3	STEPS INVOLVED	52
6.4	PDA TRANSITIONS FOR VEHICLE RECORD UPDATE	53
6.5	ID FOR VEHICLE RECORD UPDATE	55
6.6	PDA TRANSITIONS TO GENERATE CLUSTER VIEW	55
CHAPTER 7: BENEFITS OF PROPOSED MODEL		57-58
CHAPTER 8: IMPLEMENTATION WORK		59-86
8.1	DATABASE	59
8.1.1	Tables	60
8.1.1.1	Main Table	60
8.1.1.2	Dummy Table	61
8.1.2	Trigger	62
8.1.2.1	Definition	62
8.1.2.2	Types of Trigger	62
8.1.2.3	Advantages of Trigger	64
8.1.2.4	Creation of Trigger	65
8.2	JAVA APIs	67
8.2.1	JAVA Introduction	67
8.2.2	Stack	68
8.2.3	JDBC	68
8.2.4	Swing	72
8.2.5	JFrame	72
8.2.6	Util	73
8.2.7	Events and Event Handling	74
8.2.8	Graph (JFreeChart Library)	77

8.3	IMPLEMENTATION RESULT	80
8.3.1	Number of Passed Vehicles	81
8.3.1.1	Show Graph of Type-1 Vehicles	81
8.3.1.2	Show Graph of Type-2 Vehicles	81
8.3.2	Revenue Generated	82
8.3.2.1	Revenue Earned by Type-1 Vehicles	82
8.3.2.2	Revenue Earned by Type-2 Vehicles	82
8.3.3	Total	83
8.3.3.1	Total Number of Passed Vehicles	83
8.3.3.2	Total Revenue Generated	83
8.3.4	Stack	84
8.3.4.1	Pop Type-1 Stack	84
8.3.4.1	Pop Type-2 Stack	84
8.3.5	About	85
8.4	INTEGRATED DEVELOPMENT ENVIRONMENT	85
CHAPTER 9: CONCLUSION AND FUTURE SCOPE		87
REFERENCES		88-90
PLAGIARISM REPORT		
LIST OF PUBLICATION		
CURRICULUM VITAE		

LIST OF TABLES

		Page No.
Table 2.1	Stages of Tolling	5
Table 2.1	Toll Tag System in U.S. and Interoperability	9
Table 4.1	Comparison Study Chart	22
Table 8.1	Event Classes and their Listener Interface	74

LIST OF FIGURES

	Page No.	
Figure 2.1	On-board mileage-counting equipment in the Oregon Pilot Project	13
Figure 2.2	The Galileo Satellite System for Global Positioning	15
Figure 3.1	Clustered View of Proposed Model	16
Figure 5.1	General Push Down Automata Model	35
Figure 5.2	Two-Stack Push Down Automata Model	38
Figure 5.3	Use of Stack in Two-Stack PDA	40
Figure 6.1	Flow Chart for working of Two-Stack PDA Model	50
Figure 6.2	Incoming vehicles record updation process	53
Figure 7.1	Output Clustered View of Proposed Model	57
Figure 8.1	Screenshot of MainTable	61
Figure 8.2	Screenshot of Dummy Table	61
Figure 8.3	Screenshot of created Trigger	66
Figure 8.4	Two-tier Architecture for Data Access	70
Figure 8.5	Three-tier Architecture for Data Access	70
Figure 8.6	Screenshot of Main Window	80
Figure 8.7	Screenshot of Type-1 Vehicles passed	81
Figure 8.8	Screenshot of Type-2 Vehicles passed	81
Figure 8.9	Screenshot of Revenue Generated by Type-1 Vehicles	82
Figure 8.10	Screenshot of Revenue Generated by Type-2 Vehicles	82
Figure 8.11	Screenshot of Total number of passed vehicles state wise	83
Figure 8.12	Screenshot of Total revenue generated state wise	83
Figure 8.13	Screenshot of Stack-1 Data items	84
Figure 8.14	Screenshot of Stack-2 Data items	84

Figure 8.15	Screenshot of Help menu items	85
Figure 8.16	Screenshot of NetBeans IDE	86

LIST OF SYMBOLS AND ABBREVIATIONS

ANPR	Automatic Number Plate Recognition
AVC	Automatic Vehicle Classification
AVI	Automatic Vehicle Identification
CFG	Context Free Grammar
CFL	Context Free Language
DSRC	Dedicated Short Range Communication
ETC	Electronic Toll Collection
GNSS	Global Navigation Satellite System
GSM	Global System for Mobile communication
ID	Instantaneous Description
IDE	Integrated Development Environment
OCL	Object Constraint language
OMG	Object Management Group
ORT	Open Road Tolling
OTTDBMS	Online Toll Tax Data Base Management System
OTTMS	Online Toll Tax Management System
PDA	Push Down Automata
RFID	Radio Frequency Identification
RTCN	Real Time Constraint Notation
TSPS	Toll Snapping and Processing System
UML	Unified Modeling Language

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Tolling as a method of financing the transportation system is becoming more common in the country. Neither the traveling public nor State Departments of Transportation want vehicle to stop or slow down to pay to use a toll facility. In this project we highlight the use of Push-Down Automata (PDA) in sorting and maintaining the vehicle records coming for which state, calculate the total revenue generated and show this information into cluster view of vehicle logs.

1.2 MOTIVATION

In the last two decades, Online Toll Tax Management System has been improved a lot. However, there still remain some problems which have not been answered satisfactorily. Currently, the toll rates are typically brought down to 40% of the ongoing toll rate after the end of a concession period. The Minister for Road Transport and Highways Mr. Nitin Gadkari's promise of limiting toll collection to recover project cost is facing resistance on grounds that it will add to the fiscal burden of the government. Officials in the road ministry and National Highways Authority of India (NHAI) have pitched to make a presentation to the minister to explain why the proposal is not feasible. [27]

Usually these are good stretches and cost of maintenance is high and there aren't enough funds with the government to ensure maintenance of these stretches without the toll money. Currently, the toll rates are typically brought down to 40% of the ongoing toll rate after the end of a concession period. Concession period is the duration for which a developer is given the contract for tolling a road project for recovering the cost of construction and maintaining it, and typically ranges between 20 and 30 years. The tolling policy is very vague. It needs to be revisited for many reasons like defining what constitutes the cost of construction and why some stretches are tolled and some not.

However, the toll that is collected after the concession period is used for maintaining and cross-subsiding other national highway stretches that cannot be tolled. If this is removed where

will the money for maintaining the national highway network come from? Addressing a conference on 100 days of the National Democratic Alliance (NDA) government on 15 September, the Ministry of Road Transport and Highways, a branch of the Government of India, reiterated his keenness to go ahead with the proposal. The remarks come in the backdrop of several instances of protests against toll collection. [27]

Road ministry officials are exploring alternative, acceptable solutions for resource mobilization that could at least address the resentment of the local residents. One of the suggestions is to toll only the commercial traffic after the cost of the project is recovered as anyway 80% of the toll collection comes from the commercial traffic. The National Highways Fee (Determination of Rates and Collection) Rules formulated in 2008 are in force currently. The tolling strategy needs to be relooked at, keeping in mind the issue of the local users. More importantly the issue of collecting toll until perpetuity for stretches without providing an alternative free route to users who may not wish to pay also needs to be addressed. In this work, we have identified such problems and tried to provide an effective solution to one of that problem.

1.3 FOCUS

We focus to provide a communicative framework that can record the vehicle coming from a particular state. In order to achieve this goal we have identified three basic steps, the first step deals with the development of Ubiquitous computing environment as the vehicle coming from. Secondly, development of push down automata model for vehicle logs database management system. Since Object Constraint Language (OCL) is being applied on Object Oriented Language we will be representing the entire communication framework into Object Oriented Language. In the research work we are mainly focused on the following two points:

- Highlight the use of PDA in sorting and maintaining record.
- Focus is to provide a communicative framework that can record vehicles log.

For this, there will be three basic steps:

- Development of Ubiquitous computing environment.
- Development of PDA model for database management system.
- To verify the model and do the performance analysis.

1.4 THESIS ORGANIZATION

This will describe how this documentation organization done. Thus the rest of the thesis is organized in the following chapters as follows:

Chapter 1: This chapter introduces the research topic, motivation, what is tolling system, the online toll tax management system, what I am going to state in this research work and the organization of the chapters.

Chapter 2: In this chapter the present scenario of online toll tax management system is described. This chapter includes the various tolling practices, their objectives and evolution of tolling. A glimpse of upcoming tolling technologies included here.

Chapter 3: In this chapter we describes the Scope of the work, it consists of the problem statement of the system and constraints generated in the system. Here the rectified objective of this research work is illustrated related to the problem statement and constraints.

Chapter 4: In this chapter a wide range literature review of the key concepts and outstanding works in the area of the toll tax, push down automata, online toll tax database management system and study of those concepts are mentioned. A comparison chart of those techniques with their pros and cons is also mentioned in this chapter.

Chapter 5: In this chapter we describe the applied approach of the proposed system i.e. methodologies used. It basically describes all the detailed methodology of the system, their features, components, characteristics and process used in the applied system.

Chapter 6: In this chapter the proposed work i.e. the proposed PDA model, algorithm and the flow chart is described. The steps involved for PDA transitions of vehicle record updating process and to generate cluster view of it is described.

Chapter 7: This chapter describes the performance measure of the system based on the different parameters and the benefits of proposed model.

Chapter 8: In this chapter we describe the implementation parts i.e. the whole system works as a unit and the implementation results with the screenshots of the designed system are given.

Chapter 9: This chapter summarizes the thesis work and draws the conclusions. Some of the new directions for further work are alleged in the final chapter.

CHAPTER 2

PRESENT SCENARIO OF OTTMS

2.1 TOLLING PRACTICES

In this section the motivations for tolling, expected evolution of tolling systems, and the benefits and costs of tolling are reviewed.

2.1.1 Objectives of Tolling

There are three main reasons why tolling, or road pricing, is implemented:

Finance/Revenue Generation: To recoup the costs of building, operating and maintaining the facility. Road pricing is becoming a more appealing means of funding transportation, since revenues from federal and state gas taxes have not kept up with growth in demand for infrastructure. Moreover, toll financing allows projects to be built sooner instead of waiting for tax revenues to accumulate.

Demand Management: To moderate the growth in demand on the transportation system and to encourage more use of public transportation and carpooling. For example, vehicles are charged to enter inner London, England, as a way of regulating the demand in the region.

Congestion Management: To place a price on limited roadway space in proportion to demand. In this application the toll increases with the level of congestion. In the absence of such pricing, drivers do not appreciate the costs they impose on others as a result of the congestion they cause.

2.1.2 Evolution of Tolling

Roadway tolling is expected to become more pervasive over time. Four stages are envisioned as shown in Table 2.1, beginning with corridor tolling and cordon tolling, then area wide or vehicle-miles-traveled (VMT) tolling, and ultimately an integrated system management strategy. Each stage improves system efficiency over the previous one, but also has higher complexity. Each stage also requires certain conditions before implementation. Only the first two strategies, corridor tolling and cordon tolling, have been widely implemented, with ETC

being a necessity to move to the next two stages. The third stage is now being pilot tested in a few areas, while the final stage, an integrated system and lies in the future.

Table 2.1 Stages of Tolling [26]

Tolling strategy over time	Objectives	Complexity/Efficiency	Required conditions
Corridor Tolling	Repayment for facility	Low	Road must be exclusive to those who pay
Cordon Tolling	Demand management	Medium	Public trust that benefits will outweigh costs
Area-mileage Tolling	Revenue generation	High	Uniformity/interoperability
Integrated system management	Demand/congestion management	Very high	Flexibility across modes: access to information

Corridor Tolling: This is the most common form of tolling, in which a driver pays a fee to use a specific stretch of roadway or bridge. High Occupancy Toll (HOT) lanes, lanes designated for multi-passengers but which single-occupant vehicles can use if they pay a toll, are also included on this category. The primary objective of the toll is to repay the cost of building and operating the facility. Complexity can be as low as having the driver stop and pay cash on entry, although most systems are implementing ORT.

Cordon Tolling: This is a charge for entering a specific area. The primary objective is to reduce the number of vehicles entering. Every entry point must be equipped with means of identifying vehicles and ensuring that they pay, have paid, or will pay. To be an effective strategy, the public must be convinced that benefits (improved mobility, lower pollution, etc.) will be realized fairly quickly. An efficient public transportation system is essential for this strategy to be effective.

Area-wide Mileage Tolling: This is a mechanism whereby vehicles are charged based on VMT - a road user fee. An example of this system is the German truck toll, in which all trucks are required to pay tolls based on the distance traveled inside Germany. In some respects this strategy is analogous to the U.S. gas tax, in that, theoretically, each vehicle pays based on miles

driven. The primary objective is to generate revenue for the transportation system and, to a lesser degree, to regulate the amount of driving. The complexity of distance-based tolling is relatively high and requires uniform application area-wide, as well as interoperability across borders.

Integrated System Management: In this visionary concept, demand for transportation would be managed through information: users would have a choice of modes and routes and an array of ways to pay for a trip. The charge would incentivize the most efficient transport choice and the market would drive the provision of capacity. Highly complex systems, such as roadside vehicle-traveler communications would be required, but system usage is expected to be highly efficient. Required conditions include market flexibility and access to information.

2.2 CLASSIFICATION OF TOLLING PRACTICES

Tolling as a method of financing the transportation system is becoming more common day by day. Neither the traveling public nor State Departments of Transportation want vehicles to stop or slow down to pay to use a toll facility. Most commonly used Tolling Practices:

- Manual Tolling
- Electronic Toll Collection(ETC)

2.2.1 Manual Tolling

The most traditional approach for collecting charging or toll collection is the manual one. The manual toll is the conventional toll collection system which is made by an operator in the cabin that carries out the payment of the fee for each vehicle, using multiple methods of payment: cash (coins or banknotes), credit cards (magnetic bands or chip standard EMV), discount cards, etc. Currently, manual collection is usually combined with other technologies, forming the well-known mixed lanes. However, there are countries that continue to operate only with manual systems.

The most significant components of a MTC system are:

- Toll Plaza / Booths
- Road Side Equipment (RSE)
- Toll Collectors & Staff

- Cash Handling System
- Back Office System

According to the manual toll collection methodology, a driver has to stop at a charging booth and pay the required fee directly to a collector. The amount to be paid by each vehicle is determined by its characteristics or classification.

2.2.2 Electronic Toll Collection

Electronic Toll Collection (ETC) is a system that automatically identifies a vehicle equipped with a valid encoded data tag or transponder as it moves through a toll lane or checkpoint. ETC aims to eliminate the delay on toll roads by collecting tolls electronically. ETC determines whether the cars passing are enrolled in the program, alerts enforcers for those that are not and electronically debits the accounts of registered car owners without requiring them to stop. In 1959, Nobel Economics Prize winner William Vickrey was the first to propose a system of electronic tolling for the Washington Metropolitan Area. [8]

The ETC system then posts a debit or charge to a patron's account, without the patron having to stop to pay the toll. ETC increases the lane throughput because vehicles need not stop to pay the toll. The motorized society which developed in tandem with the establishment of toll roads has dynamically expanded our range of activities while enriching and coloring our daily lives. However, traffic jams caused by the concentration of cars on convenient toll roads are increasing year by year. Traffic jams are caused by saturated roads, which mean that jams can be resolved by increasing the flow of traffic. Up until now, there have been active efforts to widen roads in order to alleviate traffic jams but these initiatives soon reached their limit, bringing about the need for a fundamental system reform of toll roads. This marked the beginning of the ETC project.

Implementation and Operation Challenges:

- Insufficient knowledge of ETC technology by consumers who fear their movements will be 'tracked'.
- Political disinclination, also mainly because of ignorance about ETC technology, as well as a desire to avoid antagonizing voters who have a misguided notion of ETC.
- Interoperability issues between different systems which raise costs.

- Reconstruction of highways to include ORT lanes builds gantries or dismantles existing manual toll collection booths.
- Non-paying users—because of minor shortcomings of ETC technology, some users may slip through the system without paying.

2.2.2.1 Components of ETC

Electronic toll collection systems rely on four major components: [11]

- a) **Automated Vehicle Identification:** Automated vehicle identification (AVI) is the process of determining the identity of a vehicle subject to tolls. The majority of toll facilities record the passage of vehicles through a limited number of toll gates. At such facilities, the task is then to identify the vehicle in the gate area.
- b) **Automated Vehicle Classification:** Automated vehicle classification is closely related to automated vehicle identification (AVI). Most toll facilities charge different rates for different types of vehicles, making it necessary to distinguish the vehicles passing through the toll facility. The simplest method is to store the vehicle class in the customer record, and use the AVI data to look up the vehicle class.
- c) **Transaction Processing:** Transaction processing deals with maintaining customer accounts, posting toll transactions and customer payments to the accounts, and handling customer inquiries. The transaction processing component of some systems is referred to as a “customer service center”.
- d) **Violation Enforcement:** A violation enforcement system (VES) is useful in reducing unpaid tolls, as an unmanned toll gate otherwise represents a tempting target for toll evasion. Several methods can be used to deter toll violators. Police patrols at toll gates can be highly effective.

2.2.2.2 Enhancements

Interoperability

A major concern regarding electronic tags is the degree to which they are interoperable with tags from other regions. If there is to be an area-wide tolling program, a single tag must work in all jurisdictions. This is not the case with most systems. Table 2.2 shows tag operations in the U.S. currently and those that provide interoperability.

Table 2.2 Toll Tag Systems in the U.S. and Interoperability [26]

Tag System	Jurisdiction	Interoperable with
C-Pass	Key Biscayne, Florida	
Cruise Card	Atlanta, Georgia	
E-PASS	Orlando, Florida	SunPass
EXpressToll	Colorado	
E-ZPass	U.S. Northeast	
Fast Lane	Massachusetts	E-ZPass
Fastrak	California	
I-Pass	Illinois	E-ZPass
K-Tag	Kansas	
LeeWay	Lee Country, Florida	SunPass
MnPass	MinneSota	
O-PASS	Osceola County, Florida	SunPass
PalmettoPass	South Carolina	
Pikepass	Oklahoma	
Smart Tag	Virginia	E-ZPass
SunPass	Florida	SunPass
TollTag	Texas	TxTAG
EZ TAG	Texas	TxTAG
TxTAG	Texas	EZTAG, TollTag

The idea is that the customers would have information stored in the database. The traffic would then share that data with toll authorities across the customers that will be able to drive through any ETC lane without the use of a transponder. Instead of registering them as violators, the electronic toll system would scan the customer's information, look up the customer on the database and charge the owner's credit card.

Rental Tags

The rental car industry is considering renting out toll tags to its customers. They have found that rental car customers do not object in principle to paying tolls, but do object to having

to wait in line to pay them because the car does not have a tag. Budget Rent a Car gives the option of renting a transponder for 99 cents a day, tolls not included.

2.2.2.3 Benefits and Cost of ETC

Benefits of ETC

Among the benefits of ETC are illustrated below:

- ETC lanes improve the speed and efficiency of traffic flow and save drivers time. Manual toll collection lanes handle only about 350 vehicles per hour (vph) and automated coin lanes handle about 500 vph. An ETC lane can process 1200 vph, with ORT lanes allowing up to 1800 vehicles per hour (Tri-State Transportation Campaign, 2004).
- As a result of better flow, congestion is reduced, fuel economy is improved, and pollution is reduced.
- Increased revenue: time savings, faster throughput, and better service attract more customers, thus increasing revenue.
- Reduced accident rates/ improved safety because of less slow-and-go driving.
- Increased efficiency of roads because of better distribution between tolled and non-tolled routes.

Two benefits of open-road tolling are especially noteworthy (Tri-State Transportation Campaign, 2004):

- Safety benefits: Generally, ORT facilities are nearly accident-free. ORT allows vehicles to travel at normal highway speeds, avoiding dangerous stop-go traffic and sudden merges, and eliminating the danger of drivers jockeying for lane position. ORT can also cut down on the distractions toll payers face while driving, such as fumbling for change or having to slow down or stop to pay the toll.
- Economic Benefits: Delays cause losses to both the driver and the overall economy. Drivers suffer direct costs of increased fuel consumption and vehicle wear and tear owing to idling and stop-and-go movement, as well as indirect costs of stress. Valuable time is spent in traffic instead of productive work. Delays also drive up the cost of shipping goods - a cost usually passed on to the consumer. ORT reduces delays and thus provides economic benefits.

Costs of ETC: There are several costs in implementing an ETC system. Among the major costs are:

- Toll Agency Costs: According to a 2002 study by the California Center for Innovative Transportation, the cost per transaction of an ETC system is between \$0.05 to \$0.10 (Smith, ITS Decision, 2002). A manual toll cost per transaction is \$0.35 to \$0.45. Not only are the costs per transaction usually lower in an ETC system, the number of transactions is far higher than in a manual system. Additionally, the number of people required to operate an ETC system is far fewer than required for a manual toll collection system. Overall costs per transaction, therefore, shrink significantly. Oklahoma Turnpike, one of the first U.S. highways to use high-speed toll plazas, saw a 90 percent reduction in collection costs on ETC lanes.
- Costs to the User: Most systems which have implemented ETC require motorists to buy or rent the equipment. In addition to the cost of the system, the motorist is also required to pay a security deposit, keep a minimum balance in his account, and, in some cases pay a monthly fee for the ETC equipment. Some systems also require motorists to keep a credit card balance.
- Initial Sunk Costs: The initial costs of implementing ETC or converting a manual toll facility into an ETC can be quite high. There are also significant operational and maintenance costs to an ETC system that are difficult to predict or to figure into present worth calculations.

2.3 UPCOMING TOLLING TECHNOLOGIES

There are also some ETC technologies that are currently being considered as possible alternatives to transponders and video tolling and that may find applications in area-wide mileage tolling programs and in integrated system management, including:

- a. Odometer Tolling
- b. Vehicle Positioning Systems
 - I. Cell Phone Tolling
 - II. Satellite Tolling

2.3.1 Odometer Tolling

The Oregon Department of Transportation (ODOT) is conducting a pilot project on odometer tolling (Oregon Department of Transportation, 2005). In 2001, the Oregon State Legislature created the Road User Fee Task Force (RUFTF) to look at means to raise revenue as a replacement for Oregon's gas tax. RUFTF looked at twenty-eight different options and focused on a distance-based charge on the number of miles traveled in Oregon. The Road User Fee Pilot Program was created to examine the technical and administrative feasibility of implementing a per-mile fee. The program uses on-board mileage-counting equipment to keep track of the number of miles traveled. Based on the results of the pilot test, ODOT will draft legislation to be put before the state legislature in 2009.

During the fall of 2005, a pre-pilot program of twenty volunteers started the program to work out any unexpected issues that could occur. Volunteers' cars were equipped with on-board mileage counting equipment (Figure 2.1). In spring 2006, 280 volunteers in Portland had the equipment added to their vehicles. For a period of one year, volunteers will pay a fee equal to 1.2 cents a mile and will not pay the gas tax. There will be two service stations in the Portland area equipped with mileage reader devices and pilot participants will be asked to fill their vehicles at these participating service stations when convenient. When refueling, the on-board mileage counter will communicate with the mileage readers placed at the pumps. When the purchase is totaled, the gas tax will be deducted automatically and the road user fee will be added automatically.

A federal requirement of the Pilot Program is to test the ability to count separately the miles traveled during rush hour within a congested area. Some of the pilot volunteers will be in a rush hour pricing group to test this concept. Because the pilot is a test, many policy options remain for decision-makers, such as charging a lower rate-per-mile for vehicles that achieve a certain fuel efficiency, for motorists that avoid rush hour zones, or for those participating in other environmentally-friendly activities. The road user fee program does not track, store or collect private information. There is a switching device that counts the number of miles the vehicle has traveled. The device cannot record the location of the vehicle except when the vehicle passes through certain designated rush-hour zones. The device counts only the number of miles traveled within the zone, not the time of day, location in the zone, or even the day.



Figure 2.1 On-board mileage-counting equipment in the Oregon Pilot Project [26]

There is also a GPS receiver in the cars that simply tells the electronic odometer whether to count the miles as in state or out of state. This is to prevent Oregonians from being charged for miles driven outside the state. No location data is transmitted anywhere or stored in the device or elsewhere; since vehicle location data is not collected, it cannot be accessed. The only data collected and transmitted is the mileage, which is sent to the gas pump reader through a radio frequency that can only travel about 8 to 10 feet. As the driver fuels up, the VMT is calculated and the gas tax is deducted.

The Oregon Road User Fee concept recommends that only new vehicles be equipped with the on-board technology. All of the technologies being used in the pilot program are already being manufactured in cars today. Some automobile manufacturers have already announced that key components will be standard equipment on all models within the next few years. The Federal Highway Administration (FHWA) and transportation standards organizations are working to adopt universal standards for the same technologies being used in the pilot program. In the near future, therefore, it is very likely that a state adopting a GPS-based mileage fee would not need to require additional hardware to be installed in vehicles. Some sort of software upgrade seems more likely.

With the Road User Fee Pilot Program, Oregon is not looking to raise revenue but to look at options for the inevitable future road revenue decline. The ODOT is obliged to test congestion pricing in the pilot program (as a requirement of ODOT's FHWA Value Pricing Pilot Program grant). It is not an indication of a specific policy directive adopted by the Oregon DOT or the state legislature. Any future policy decision Oregon may make on the mileage fee does not

necessarily translate into application of congestion pricing, as these two policy decisions are separate. The pilot program will simply test whether or not an electronically collected mileage fee could technologically include congestion pricing, should policymakers ever decide to go in that direction.

2.3.2 Cell Phone Tolling

Cell phone tolling is a concept that this has potential for mileage-tolling. Essentially, a chip similar to a cell phone chip would be installed in a vehicle, and frequent communication between cellular towers and the chip would determine how far the car has moved and would assess a toll. Given the near total coverage of cell phone signals in urban (and congested) areas of the U.S. and the deployment of GPS capabilities in cell phones for 911 phone locating, this technology appears to be technically feasible. It is likely to be less expensive than satellite-based systems because the infrastructure needed (cell phone towers) already exists. In addition, installing a cell phone chip in a car will likely be less expensive than installing a GPS unit capable of picking up satellite signals.

The proof of viability of the concept is in a recent marketing campaign by telecommunications firm Sprint to help parents keep track of their children. The service lets parents look at maps on their cell phones to locate their children, who also carry cell phones. Sprint's service shows data such as street addresses to which a child is in close proximity and the estimated accuracy of the reading, which could range from a radius of 2 yards around the child to hundreds of yards.

2.3.3 Satellite Tolling

Satellite tolling uses a satellite-based vehicle-tracking system to determine exact vehicle location while using mobile communication technology to compute toll charges. Each vehicle has an onboard unit which records the vehicle's movements by periodically downloading satellite time stamped location coordinates. Satellite tolling is considered the most promising technology for ETC because it allows for accurate, distance-based tolling. It is also flexible, allowing for time and location-variable tolls. Satellite tolling is touted to become the preferred method of ETC, especially in Europe.

Satellite technology is improving rapidly. With the launch of the European Galileo system (beginning in 2008), the technology will improve further. Galileo is the next generation of satellites and will overcome most of the shortcomings of the current GPS system, being more accurate and reliable. It will also be interoperable with existing systems, allowing for greater access and backup ability. The project is being managed by the European Commission and European Space Agency.

The Galileo system will work as shown in Figure 2.2 (BBC: Europe's Galileo Project):

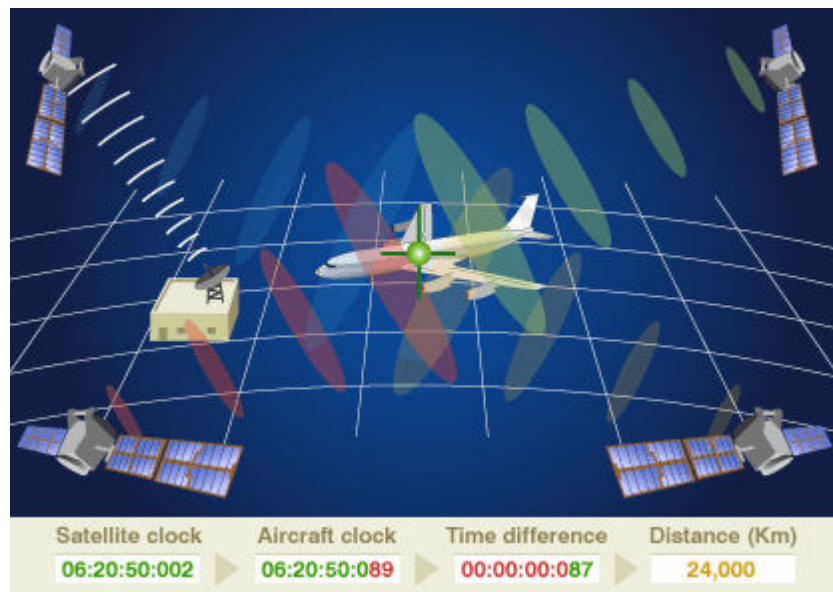


Figure 2.2 The Galileo Satellite System for Global Positioning [26]

- Satellite navigation systems determine a position by measuring the distances to at least three known locations the Galileo satellites.
- The distance to one satellite defines a sphere of possible solutions; the distance to three defines a single, common area.
- The accuracy of the distance measurements determines how small the common area is and thus the accuracy of the final location.
- In practice, a receiver captures atomic-clock time signals sent from the satellites and converts them into the respective distances.
- Time measurement is improved by including the signal from a fourth satellite. Galileo time is monitored from the ground.

Advantages of Satellite Tolling:

- Faster, hassle free and less paperwork: A GPS system will involve less paperwork and lower transaction costs than other forms of tolling. All a driver has to do is drive through a toll station and his driving distance and toll information can be uploaded into the system automatically through wireless connection. If the driver has a prepaid account, the toll charge can be deducted automatically.
- Ancillary services could be provided through transponders and GPS: A GPS system will not only allow collection of tolls, but other information and services can be passed along to the driver as well. For example, the driver will be able to receive real time weather and traffic information. In case of an emergency, his position and situation can be accurately monitored.
- Negates the necessity of investing in expensive roadside infrastructure: Once the infrastructure is in place, there are few costs involved in the operation of a GPS-based tolling system.

Disadvantages of Satellite Tolling:

- Phase-in period: At present, an OBU has to be installed for a vehicle to use GPS tolling. It is expected that it will be another 10–15 years before OBUs become standard equipment on cars. In the meantime, deployment is proceeding in the trucking industry because of the desirability of tracking shipments.
- Interference in certain situations: At present, GPS is not entirely reliable because there can be situations where satellite signals are lost (such as in urban canyons, or heavily forested roads, and during lightning storms). The technology, however, is getting more accurate and these problems are expected to be, for the large part, resolved with Galileo.
- Public reluctance of being tracked: GPS is, in fact, a passive system and cannot track individuals themselves. Just because someone carries an active receiver does not mean his every move can be followed.
- Start-up costs: Some of the start-up costs such as distribution of OBUs and installation of payment booths may be expensive. The German Toll Collect system, for example, went well over budget. Once installed, however, it is not as expensive to maintain as other forms of toll collection methods.

CHAPTER 3

OBJECTIVE OF RESEARCH WORK

3.1 PROBLEM STATEMENT

This research work focuses on enhancement of the current toll tax database management system to view vehicles coming from which territory in clustered form and show this into cluster view of vehicle logs. This clustered view will help in comparative analysis. In this project we develop an interactive and communicative framework by using the propose PDA model that can maintain a record of the vehicles coming from various states. It focuses on the development of a feature to view vehicles coming from which territory in cluster view. We can calculate the total revenue generated territory wise and show that in graphical view as well.

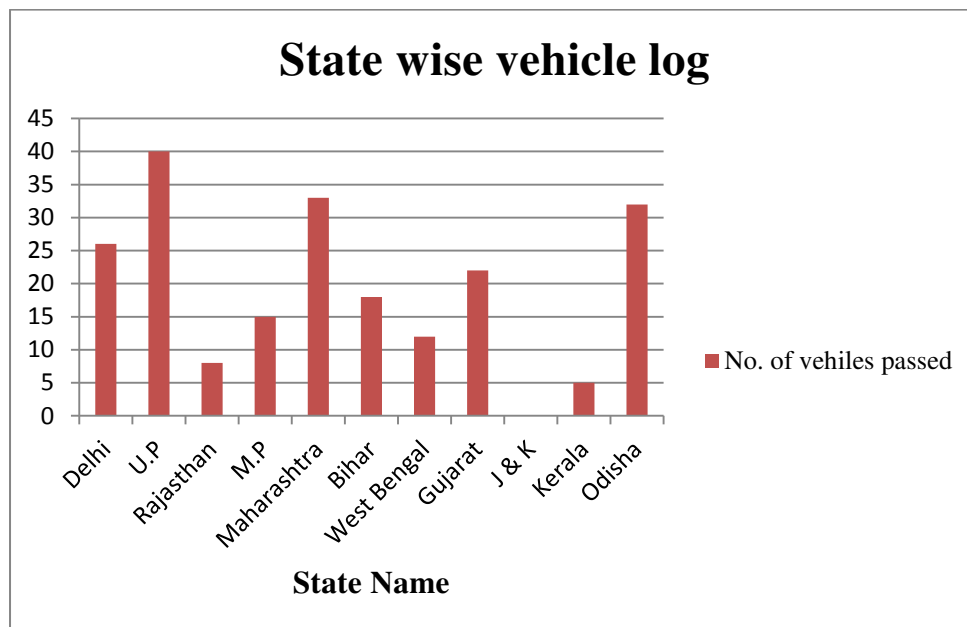


Figure 3.1 Clustered View of proposed model

3.2 CONSTRAINT

While proposing this research work there are some constraints generated at the stage of development which also creates an issues for the system designing which discussed below:

- The user's subjective intentions: Different user may have different perception of same software, which refers to the subjectivity of user's perceptions. The research of how to reflect it in toll plaza data presentation is rather few.
- Communication Gap: Communication gap refers to the difference in the way a user perceives toll plaza management system and the way a system perceives it on the basis of its certain characteristics. A major constraint in toll plaza data management system is information flow and data representation.
- Integration of various features: Multi features outperform the single feature in this system. The proposed system user can view the number of incoming vehicles from one direction and number of incoming vehicles from the opposite direction as well in the clustered view separately. In the same way, the revenue generated by both types of vehicles can be observed in classified manner state wise. Moreover, the user can view state wise the total number of vehicles passed and revenue generated by those in graphical manner for comparative analysis.
- Customization of the toll data and information interpretation: Customization of the toll data and information interpretation is the big challenge to achieve because a high dimensionality of the system is required to incorporate versatile technique. This customization was also a big constraint in the matter of the Toll Tax Database Management Model.
- Data confidentiality and security: The toll data which is used as an input to the system must be designed appropriately and according to the desire of the system's objective what the user wants to be search, it must reflect the properties of the data confidentiality and security.

3.3 OBJECTIVE

Our objective is to develop a more interactive and communicative framework that can maintain a record of the vehicles coming from a particular state. In order to achieve this objective the real time constraint notation being applied to the push-down automata for formal verification of the model. Since Object Constraint Language is being applied on object-oriented language we will be representing the entire communication framework into Object-Oriented Language. The proposed work of this research work can be described as following points:

- We highlight the use of PDA in sorting and maintaining the records: We are proposing a Two-Stack Push-Down Automata model for the Toll Plaza record maintenance. Our model is very much capable of storing and maintaining the vehicle log records and as per the requirement it will generate the state wise total number of vehicles passed and revenue generated in the clustered view.
- Our focus is to provide a communicative framework that can record vehicles log: We are going to develop an interactive and communicative framework that will allow the user to interact with the Toll Plaza Management System in easier way as before.

For this, there will be 2 basic steps:

- Development of Ubiquitous computing environment.
- Development of PDA model for database management system.

CHAPTER 4

LITERATURE REVIEW

4.1 RESEARCH AND BACKGROUND

Online Toll Tax Database Management System for general-purpose databases is a highly challenging problem because of the large size of the database, the difficulty of understanding and the continuously increasing data. The difficulty of formulating a query and the issue of evaluating results properly. A number of general-purpose Online Toll Tax Database Management System has been developed. Toll Tax Management System is an application that can provide all the information related to toll plazas and the passenger checks in either online and pays the amount, then he/she will be provided by a receipt. With this receipt he/she can leave the toll booth without waiting for any verification call.

The information would also cover registration of contractor, toll plaza collection, toll plaza collection entry for individual sector , date wise report entry, toll tax rates for different type of vehicle, Immediate Superior details and HOD details, cheque entry based on receipt date, cheque, department and other details. Based on the user login, privileges are provided to access, such as when the contractor login using his username, he can access only his details and not the other contractors details. Besides the toll plaza data, the system will have information relating to the Annual Confidential Report of contractors.

In the present day scenario, power is a major need for human life. There is a need to develop non- conventional sources for power generation due to the reason that our conventional sources of power are getting scarcer by the day. The main aim is to run the toll system using piezoelectric sensor. In turn we are saving the power needed for running the toll gate. Now a days, there is a huge rush in the toll plazas in order to pay the toll tax. Therefore in order to reduce the traffic jam and to save time and also to reduce the money loss of 300 cores / year. Delay at tollbooths is nowadays quite common in almost all the major highways. This leads to increases in congestion, inconvenience and fuel consumption. Automatic Toll Collection System brings a new idea to eliminate the delay at tollbooths. Here a micro controller based automatic system using infrared sensors is being utilized. IR sensors have been used as these are

economical, have perfect line of sight, hence less interference. Automatic toll collection system aims to collect toll from the vehicles without making the vehicle stop at the tollbooth. Each user is provided with a module that is to be mounted on the windshield of the vehicle which consists of transmitter, receiver, microcontroller and a LCD unit. The toll booth is equipped with another module for collecting toll from each user. The computer at the toll booth holds the database consisting of details regarding each user along with a unique code. The Automatic Toll Collection System include benefits to both toll authorities and facility users, in terms of time and cost saving, improved security, increased capacity and greater convenience.

Moving to other aspect of this thesis work, pushdown automata are a widely used model both in language theory and program verification. Recently, several models have been introduced that extend pushdown automata with clocks and real-time constraints. In the mean time, several works have extended the model of timed automata with prices (weights). Weighted timed automata are used in the modeling of embedded systems, where the behavior of the system is usually constrained by the availability of different types of resources.

There is a demand for reliable real-time systems. A typical real-time application consists of several processes, each of which has to be completed within a given time frame. Up until recently there has not been enough emphasis on the tools which can be used to produce real-time software with performance and reliability guarantees. By introducing real time euclid, a language designed specifically to address reliability and guaranteed schedulability issues in real-time systems. Real time euclid employs exception handlers and import/export lists to provide comprehensive error detection, isolation, and recovery. The philosophy of the language is that every exception detectable by the hardware or the software must have an exception handler clause associated with it. Moreover, the language definition forces every construct in the language to be time and space bounded. Consequently, Real-Time Euclid programs can always be analyzed for guaranteed schedulability of their processes. Thus, we feel that real time euclid is well-suited for writing reliable real-time software.

4.2 COMPARISON CHART

Table 4.1 Comparison Study Chart

Year	Paper Name	Focus	Technology	Pros and Cons
2015	Toll Snapping and Processing System	Automation in toll tax payment using Online Registration form and RFID is designed. Proposed TSPS is design of paying toll through online transaction System.	Combination of microcontroller and RFID technology	Eliminates the manually perform ticket payments and toll fee collections. Low cost, security, far communication distance and high efficiency etc. Problem for out-station vehicles.
	Computer Vision Based Vehicle Detection for Toll Collection System Using Embedded Linux	To design and develop a new efficient toll collection system which will be a good low cost alternative among all other systems	Computer Vision vehicle detection using OpenCV library. System is designed using Embedded Linux development kit(Raspberry pi)	There is hardly any effect of skipping of frames on the output. Tests on algorithm suggest that the threshold of variance between foreground and background is crucial parameter to look for.
2014	Automatic Toll Gate System Using Advanced RFID and GSM Technology	Automation in Toll Plaza which will reduce the complete processing time.	RFID, GSM, DSRC Technology	Automatically detect and identifies the vehicles High implementation cost
	The Charge Collector System	Proposed a new system to collect tolls on Open Road Tolling (ORT) infrastructures.	DSRC, GSNN, NFC	Flexible payment options, recording of incurred tolls and make them available to the end user and management entities. C2S is a work in progress project. It is required to test this solution in real scenarios, across several countries in different systems.

	Gateless Electronic Toll Collection using RFID	An effective and efficient utilization of communication link between RF Modems over a wireless channel to facilitate monitoring, authentication and automated toll collection of vehicles.	RFID Tag, RFID Reader, Serial Communication Port, VB .Net	The vehicles are automatically identified by the system which can be used to track a vehicle and classified as a 2-wheeler, 4-wheeler or a heavy vehicle The vehicles are tracked on road in real time
2013	Image Processing Based Automatic Toll Booth in Indian Conditions	Examine the image and respective information to make the toll plaza system efficient.	ANPR Technology	Reduce the human effort at toll plaza and doesn't required any tag. Camera is used for capturing the image, which can be easily affected.
	Dynamic approach towards Toll Tax Collection & vehicle tracking with the help of RFID	Use of RFID technology for ETC system. The primary requirement is to eliminate the need for motorists and toll authorities to manually perform ticket payments and toll fee collections, respectively.	RFID Technology, Remote Database Connection	The problem of traffic congestion and human errors in the system is effectively rectified and provides efficient toll collection facility for the consumers. The overall cost of implementing the system is high.
2012	Development of a Model for Electronic Toll-Collection System	Proposed a new method for ETC system and to match car number plate detection system. Here a system is developed to collect tolls according to the weight of the vehicle.	Microcontroller, Image Processing, MATLAB	Low complexity and reduced the processing time, ensures safety and better traffic management.

4.3 STUDY OF COMPARISON CHART

I go through various research papers for literature review and till now. I am taking some paper as literature review for my research, those are:

➤ Toll Snapping and Processing System

In this paper automation in toll tax payment using Online Registration form and RFID is designed. Automation of toll plaza is made using the combination of microcontroller, RFID. The TSPS is design of paying toll through online transaction System, designed for real time toll collection that uses tags that are mounted on the windshields of vehicles, through which information embedded on the tags are read by RFID readers. The proposed system eliminates the manually perform ticket payments and toll fee collections, respectively need for motorists and toll authorities. In the present day scenario, in order to pay the toll tax we find huge rush in the toll plazas. Therefore in order to reduce the traffic jam and to save time and also to reduce the money loss of 300 cores / year. The proposed RFID system uses tags that are mounted on the windshields of vehicles, through which information embedded on the tags are read by RFID readers. The toll authorities and motorists can easily exchange data information, thereby enabling a more efficient toll collection by reducing traffic and eliminating possible human errors.

Any structure, system or building needs maintenance and rehabilitation, which are of course costly. Highways and roads are also not an exception. From the very past, the construction, extension, maintenance and operating costs of highways, roads, bridges and tunnels were collected directly or indirectly. In the old indirect method, the expenses are compensated either by the tax payment for fuel or by budget allocation of the national income. The shortcoming of this method is that a number of taxpayers, who do not use any of the roads and carriageways, have to pay extra money. However, in the other system, called direct method, the tolls are taken directly from the drivers passing that road or street. The advances in the technologies related to wireless communication has led to the emergence of several engineering designs to aid the human requirements. Today on one side the importance for secured access is growing in several fields and on the other side with technology advancements the RFID cards and readers are becoming low cost. Both these aspects are the primary reasons for rapidly growing RFID based authentication system. Today, several Wireless technologies are used for building wireless networks. Among them the 2.4GHz wireless network is most widely deployed and used. The wide usage of 2.4 GHz wireless communication indicates that this infrastructure can give near real time responses and makes suitable for crucial, industrial systems. It has been observed that a

lot of electrical energy is wasted at toll gates just to operate gates, computer systems and for lighting. [3]

➤ **Computer Vision Based Vehicle Detection for Toll Collection System Using Embedded Linux**

This paper presents a brief review of toll collection systems present in India, their advantages and disadvantages and also aims to design a new efficient toll collection system which will be a good low cost alternative among all other systems. The system is based on Computer Vision vehicle detection using OpenCV library in embedded linux platform. The system is designed using Embedded Linux development kit (Raspberry pi).

Many highway toll collection systems have already been developed and are widely used in India. Some of these include Manual toll collection, RF tags, Barcodes, Number plate recognition. All these systems have disadvantages that lead to some errors in the corresponding system. In this system, a camera captures images of vehicles passing through toll booth thus a vehicle is detected through camera. Depending on the area occupied by the vehicle, classification of vehicles as light and heavy is done. Further this information is passed to the Raspberry pi which is having web server set up on it. When raspberry pi comes to know the vehicle, then it access the web server information and according to the type of the vehicle, appropriate toll is charged. This system can also make to count moving vehicles from pre-recorded videos or stored videos by using the same algorithm and procedure that is followed in this paper.

India is a country where we get to observe most extensive National highways. Government plans various phases to complete the projects under construction. The government signs agreement with the private companies who build the infrastructure like road, port and other stuff for a particular span of time generally in years. The invested amount is charged from the vehicles passing on that newly built highway. This charged amount is called as toll tax. People have no choice to pay for toll tax for using the infrastructure. The private agency involved in the manufacturing of the infrastructure is free to charge citizens. For some places, it is observed that toll tax is still being collected even after completion of contract period. Initially there were toll collection systems such as manual toll collection without generating computer receipts. This method is really very inefficient. This method of payment was used to stop the vehicles at toll station and wait for relatively long time for their turn to come. This was causing congestion of

traffic. The states of congestion and inefficiency prompted government to plan and implement Electronic Toll Collection (ETC) system which can remove out these problems and facilitate convenience for all who involved in the process of toll collection directly or indirectly.

ETC systems are designed and developed to cooperate in the operations of toll management through the use of technology. These systems gather data on the basis of traffic and then they will classify the vehicles and collect the expected amount of fare. Electronic/automated toll collection systems are very popular these days. They do not require manual intervention for their working. There are various methods of ETC in which toll is collected and also various toll booths on which these toll collection systems are implanted. There are many toll collection systems which are present for very long duration still they are collecting toll from people. There is no transparency provided by these systems. Transparent systems play an important role in toll collection such that there will be no corruption regarding toll. The proposed system in this paper is transparent to appropriate toll collection. [2]

➤ **Automatic Toll Gate System Using Advanced RFID and GSM Technology**

The concept proposed is of automatic toll tax payment system and the amount transaction information sends to the cell phone of the motorists through the GSM modem technology. It is an innovative technology for expressway network automatic toll collection solution. In this paper, the frame composing and working flow of the system is described and data information is also easily exchanged between the motorists and toll authorities, thereby enabling a more efficient toll collection by reducing traffic an eliminating possible human errors.

Most Electronic Toll Collection (ETC) systems around the world are implemented by DSRC (Dedicated Short Range Communication) technology. Any structure, building or system needs maintenance and rehabilitation, which are of course costly. Highways and roads are also not an exception. From the very past, the construction, extension, maintenance and operating costs of highways, roads, bridges and tunnels were collected directly or indirectly. In the old indirect method, the expenses are compensated either by the tax payment for fuel or by budget allocation of the national income. The shortcoming of this method is that a number of taxpayers, who do not use any of the roads and carriageways, have to pay extra money. However, in the other system, called direct method, the tolls are taken directly from the drivers passing that road or street. The other three main reasons why tolling, or road pricing, is implemented are the

advances in the technologies related to wireless communication has led to the emergence of several engineering designs to aid the human requirements. Global system for mobile communication is that it is an international standard. If you travel in parts of the world, GSM is the only type of cellular service available.

Implementing mobile communication based health monitoring via short message service (SMS). Simple wireless control device to achieve the targets, or use the GSM network technology to achieve. Nevertheless, the functions of these devices are too simple to prevent the vehicle theft crimes from happening, furthermore, their burglarproof methods are not only character. There are millions of drivers passing through Toll Gate Stations every day. The conventional or the traditional way of collecting the toll from the vehicle owners or the drivers is to stop the car by the Toll Gate Stations and then pay the amount to the toll collector, standing (or perhaps sitting) by the side of the toll booth, after which the gate is opened either mechanically or electronically for the driver to get through the toll station. So in order to stop all these problems and inconvenience, this paper introduces an automated or a more convenient way of collecting the toll and traffic management. It's called Electronic Toll Gate Stations using RFID Technology. [14]

➤ **The Charge Collector System**

This paper proposes a new system to collect tolls on Open Road Tolling (ORT) infrastructures. The actual ETC systems do not fulfill fundamental user requirements, such as interoperability and portability between systems and road operators (in the same or different countries), as well as advanced toll logging and reporting (capabilities ensuring user privacy, an interesting feature in car rental or sharing use cases).

The Charge Collector System (C2S) is a work in progress project that will provide some features, such as flexible payment options, recording of incurred tolls and make them available to the end user and management entities, exploring a synergy of technologies in ETC scenario, namely Dedicated Short Range Communication (DSRC), Global Navigation Satellite System (GNSS), Near Field Communication (NFC) and smart phone based mobile applications. This system is also an approach to interoperable European ETC solutions, in a way that uses DSRC and GNSS-based solutions together.

Since the 60's, the Electronic Toll Collection (ETC) is used around the world and, on last decade, it is becoming more and more pervasive. The benefits of ETC associated to free flow systems are well known and the actual trends point to the creation of Open Road Tolling (ORT) infrastructures i.e. roads where tolls are entirely electronic with little or no impact on traffic flow. The two main technologies used on ORT are based on the Dedicated Short Range Communication (DSRC) transponders and Global Navigation Satellite System (GNSS) with Global System for Mobile Communications (GSM). Besides its advantages, the current ORT systems do not address conveniently the following issues:

- The flexibility of the bank account that can be used for toll payment, i.e. currently, users have always to pay by the same bank account, which can be an issue in some cases.
- The payment information/receipts or toll logs are not readily available to the user or management entities. In some cases, such as rent-a-car or car sharing companies, there is a need to know if costumers used tolls and payment has been done. In some systems, the information or toll receipt takes 48 hours to be available.
- The lack of an On-Board Unit (OBU) user interface that could provide road information and support for implementing Dynamic Road User Charging (DRUC) systems. On current systems it is impossible to apply different prices on several users in real-time because there is no user interface to say how much he/she has to pay.
- The problems related to interoperability between different systems. If a user travels abroad, she/he cannot use ETC automatically without buying the local OBU.
- In case of GNSS-GSM based infrastructures, the toll companies have to use GSM so they are dependent of mobile operators. [10]

➤ **Gateless Electronic Toll Collection using RFID**

An effective and efficient utilization of communication link between RF Modems over a wireless channel to facilitate monitoring, authentication and automated toll collection of vehicles on the highways is proposed in the paper. The system is implemented to automatically register vehicles getting on or off a motorway or highway, shortening the amount of time for paying toll in large queues.

This paper also introduces the implementation and design of an active RFID tag based system for automatically identifying running vehicles on roads and collecting their data. The

architecture and the basic principles of design of the system are presented, including reading equipment (readers and antennas) and active electronic tag. Finally, the effectiveness and efficiency of the system is analyzed as a whole. Electronic toll collection (ETC) is a technology enabling the electronic collection of toll payments. It has been applied over many highways and expressways for faster toll collection and reduced traffic congestion. This ETC system is capable of determining if the car is registered or not, and then informing the authorities of toll payment violations, debits, and participating accounts. The most significant advantage of this technology is that it eliminates congestion near toll booths. It is also a method by which vehicles at tollbooths can be tracked. Other than this obvious advantage, implementation of the ETC could also benefit the toll operators. The benefits or advantages for the users include:

- 1) Minimized queues at toll plazas by increasing toll booth service turn-around rates.
- 2) Faster and more efficient service.
- 3) The ability to make payments anytime anywhere on the card itself or by loading a registered credit card.
- 4) The well timed notification to users via the push notification that informs them about their current account status.

An ETC system extensively utilizes radio frequency identification (RFID) technology. RFID is a generic term used to identify technologies utilizing radio waves to identify people or objects. RFID technology was first introduced in 1948 when Harry Stockman wrote a paper exploring RFID technology entitled, “Communication by Means of Reflected Power”. RFID technology has evolved since then, and has been implemented in various fields, such as in, library system warehouse management, theft prevention attendance system and so on. Generally, RFID is used for, tracing, tracking, and identifying objects. The whole RFID system consists of a transponder (tag), reader/writer, antenna, and computer host. The transponders/tags are a microchip amalgamated with an antenna system in a compact Toolkit. The system contains a microchip which contains memory and logic circuits to receive and send data back to the reader. These tags are classified as either active or passive tags. The batteries in the Active tags provides a longer read range, on the other hand the passive tags are powered by the signal of the reader and hence have shorter read range.

The reader contains two components a decoder along with the RF module and an antenna to send and receive data from the tag. It can be mounted or built as a mobile portable device. The

desktop host acts as the interface for IT platform for transferring information from RFID system to the end-user. This host then transforms the information received from the RFID tag into usable resource for the end-user. [13]

➤ **Image Processing Based Automatic Toll Booth in Indian Conditions**

In this research paper we examine the image and the respective information will be processing based toll collection system and how to make more efficient and perfect. On any toll both the vehicle has to stop for paying the toll. In this paper the authors are trying to develop a system that would pay the toll automatically and reduce the queue at the toll booth. In this system camera is used for capturing the image of the vehicle number plate. The captured image would be converted into the text using ANPR and the toll would be cut from the customer's account and then open the gate. Moreover in this system if a vehicle is stolen and an entry is being made in the central database by the police then if the vehicle passes through the toll both then silent alarm would buzz which would indicate the operator at the toll booth that the vehicle is a stolen vehicle. For the identification of the vehicles, the information of the vehicles is already stored on the central database. So captured number will be sent to the server received at the toll.

The purpose of this paper is collecting the toll according to vehicles and builds the real time application which recognizes vehicles licenses number plate at entry gate. Automatic toll collection is considered as one of the intelligent transport systems. It is aimed at making toll taxation more efficient, reliable, and safe and environment friendly. In the past, customer would have to wait at the toll booth to pay the collector, creating traffic congestion, pollution and of course of a lot of frustration. Today Automatic toll collection successfully removes unnecessary traffic delays keep an eye on any car that might not be correctly registered. Automated toll collection is fast becoming a globally accepted of toll collection. IPTB system is used as a system for fast and efficient collection of toll at the toll plazas. This is possible as the vehicles passing through the toll plaza do not need to stop to pay toll and the payment automatically takes place from the account of the user. This automatic system used the technology of ANPR.

Hence this system works very fast with the best results. This new toll system depends on four components.

- AVI (automatic vehicle identification): Automatic vehicle identification systems are used for the purpose of effective control. License plate recognition (LPR) is a form of automatic vehicle identification. It is an image processing technology used to identify vehicles by only their license plates. Real time LPR plays a major role in automatic monitoring of traffic rules and maintaining law enforcement on public roads. Since every vehicle carries a unique license plate, no external cards, tags or transmitters are required.
- AVC (Automatic Vehicle Classification): In IPAT (image processing automatic toll) system, AVC automatically verifies the classifications of vehicles. Because the vehicle is already classified at the time of registration.
- Traffic Controller System: Traffic controller system is a computer system which manages the traffic in a single row or line by using Traffic signals and sensors.
- Central Server: For more security and maintain records of each toll and customers Central server is required. A central server stores the data which comes from different toll plaza. A local computer of every toll plaza is connected to a central server through Internet. The consumer / owner have to register in a central server and deposit money in their account. AVI and AVC totally depend on the vehicle license number plate.

The benefits for the motorists include:

- Less or shorter queues at toll plazas by increasing toll booth service Turnaround rates.
- Faster and more efficient service (no exchanging toll fees by hand).
- The ability to make payments by keeping a balance on the register account.
- The use of prepaid toll statements (no need to request for receipts). [15]

➤ **Dynamic approach towards Toll Tax Collection & vehicle tracking with the help of RFID**

This paper focuses on use of radio frequency identification (RFID) technology for ETC system. The primary requirement is to eliminate the need for motorists and toll authorities to manually perform ticket payments and toll fee collections respectively. The proposed RFID system uses tags that are mounted on the front glass of vehicles, with the help of which information on the tags are read by RFID readers.

This system has been around since 1992, during which RFID tags began to be widely used in vehicles to perform toll collection process automatically. Data information exchanged

between the owner of vehicle and toll authorities is done efficiently. Due to which the problem of traffic congestion and human errors in the system is effectively rectified. Million of drivers/consumer passes through toll booths paying toll tax. The past toll payment system was manual and drivers are using manual system using coin or cash by hand to cross the toll plaza gate. Manual process is too much time consuming and drivers have to wait in row for long time for crossing the toll plaza. In waiting time fuel of vehicle is also consuming fuel. Now days this manual toll deduction system is changed to automated system. Where driver no wait for pay cash or get token to cross the toll plaza. This automatic system used the technology of RFID. This new automated system works very fast with the help of RFID. RFID based automated Toll Collection system (RATS) is a fairly mature technology that allows for electronic payment for motorways and expressways. An Electronic Toll Collection system is able to determine if a car is registered in a toll payment program, alerts enforcers of toll payment violations, and debits the participating account. Electronic toll collection is fast becoming a globally accepted method of toll collection, a trend greatly aided by the advancement in the field of interoperable Electronic Toll Collection technologies.

Radio Frequency Identification (RFID) is an auto identification technology which uses Radio Frequencies (between 30 kHz and 2.5GHz) to identify objects remotely. The system does the job of detecting and accounting for vehicles as they pass through a tollgate using RFID as the identification technology. The system is a great asset in the transport industry. It reduces the common problems in accounting for the transportation of goods from point to point. This can be further developed to support the satellite surveillance systems once all toll gates are networked. An RFID tag is loaded with information in the form of an Electronic Product Code, which can be read over a considerable distance, with the help of which we can identify the vehicle and enhance a transaction to be undertaken with respect to the specific tag data, taking advantage of radio frequencies and ability to travel longer ranges with better data capacities and high speed attained with maximum accuracy. [16]

➤ **Development of a Model for Electronic Toll-Collection System**

The paper presents a new method for ETC system and to match car number plate detection system. The electronic toll collection system (ETS) has been fabricated based on microcontroller. Here a system is developed to collect tolls according to the weight of the vehicle. The car

number plate detection method utilizes template matching technique to approximate the location of car number plate.

Then, using this output from the template matching method, color information is used to eliminate the unwanted color areas from the approximate number plate region without affecting the correct color regions. Hence, the number plate region can be determined more accurately. This work can easily be done by image processing system using MATLAB. The method has low complexity and reduced the processing time magnificently. This automated system also shows a better performance in highway traffic management. This paper shows the gateway to fabricate a highly automated toll-plaza.

The purpose of this paper is to collect toll according to the weight of the vehicles and to build a real time application which recognizes license plates from cars at a gate, for example at the entrance of a parking area. The endeavor of our work was to develop a highly automated toll-collection system. The automated system classifies a vehicle based on its load range. In recent world, while collecting tolls in the bridges/ tunnels there may arise various problems such as: lengthy process for money transactions, which leads long traffic congestion as well.

Moreover, it is time consuming and not accurate. The overloaded vehicles give the same amount of toll like the unloaded one though the overloaded vehicles damage the bridge more than that of the unloaded one. In some cases the vehicles carry a huge amount of load which crosses the legal limit. Moreover, corruption is also a major problem here. This is why; such a toll collection system is fabricated here. This toll-collection system is designed for achieving a fraud free and completely auditable operation by accurately detecting and registering vehicle movements and toll collector's activities. A central control system is introduced here for online monitoring of the toll-plaza. The objectives of image processing based electronic toll collection are many. It gives an accurate data as well as saves the license plate of the vehicles to recognize any unwanted vehicle. [21]

CHAPTER 5

METHODOLOGY USED

5.1 TWO-STACK PUSH DOWN AUTOMATA

In computer science, a pushdown automaton (PDA) is a type of automaton that employs a stack. Pushdown automata are used in theories about what can be computed by machines. They are more capable than finite-state machines but less capable than Turing machines. A Push-Down Automata is a finite automata with auxiliary storage devices called stack. A pushdown automaton may be pictured as a finite automaton with the stack or pushdown store onto which symbols may be 'pushed' or from which they may be 'popped'. A normal PDA has one stack which controls the parsing of the input string in input tape and on each input the input tape will move one cell left. [1]

The fact is clearly known that the compilers design for the compilation purpose has great use of push down automata, a part of automata machine. In particular, we show a procedure to design the parsers or syntax analyzer using the concept of pushdown automata and it is very important part of the compilation. Mainly Pushdown automata have three components, out of these stack is most important one. The stack may vary according to the requirement of the input and automata machine. Stack is a data structure works on the basis of LIFO (Last In First Out). Where top of the stack increases as the value is inserted in the stack called PUSH and decreases when the value is deleted from the stack called Popped. The communication between stacks is only allowed by applying a synch. The pushdown machines have special case 'Visibly Pushdown Automata' (VPA) where the stack operations are driven by the input. Push Down Automata can be described with a formal grammar and it is more capable than a finite-state machine but less than Turing machine. Increase in number of stacks in PDA increases the problem solving capability and efficiency of PDA.

An automata machine designed for CFG (Context Free Grammar) is the Pushdown Automata (PDA). The language generated by the CFG is CFL (Context Free Language) through which the PDA is designed. Mehlhorn was the first to play with the input-driven pushdown automaton in 1980, where input alphabet split into three classes and the type of the current

symbol determines whether the automaton must push onto the stack, pop from the stack, or ignore the stack and the final complexity is $O(\log^2/\log \log n)$ [17]. A Pushdown Automata contains three parts:

1. An input tape
2. A finite control
3. A stack (data structure)

Input tape contains the input values which can be moved one cell at a time to the left. The control unit contains both tape head and the stack head, and finds itself at any moment in a particular state. The stack is a sequential data structure. The input tape is read and the value is inserted in the stack (called push operation). The value deleted from the stack is called pop.

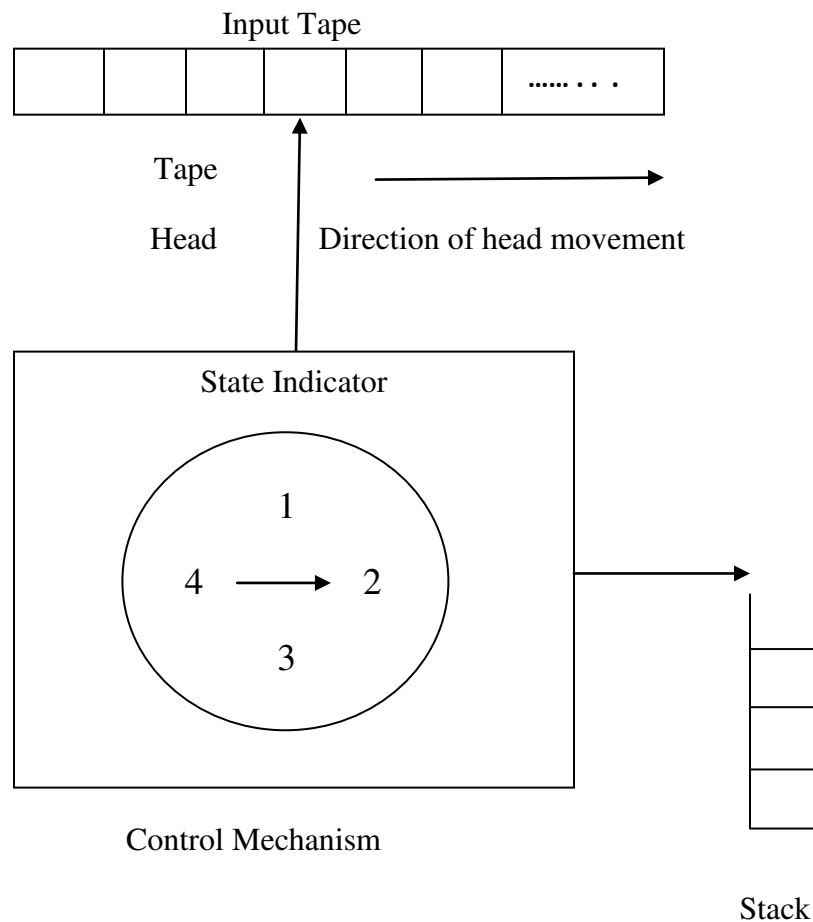


Figure 5.1 General Push Down Automata Model

Definition of Normal PDA:

A general PDA machine can be represented as:

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where

- Q is a finite set of states.
- Σ is a finite set which is called the input alphabet.
- Γ is a finite set which is called the stack alphabet.
- δ is a finite subset of $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$, the transition relation.
- $q_0 \in Q$ is the start state.
- $Z_0 \in \Gamma$ is the initial stack symbol of stack1.
- $F \subseteq Q$ is the set of accepting states.

5.1.1 Problem in Normal PDA and Variation of Stack

Normal PDA has one stack which controls the parsing of the input string in input tape and on each input the input tape will move one cell left. It is highly impossible to solve and support the all languages of context free grammar with normal PDA. Such as for the complex context free languages like $L = \{a^n b^n c^n ; n \geq 1\}$ the normal PDA will not work and hence we need another option, Which might be complex and more time taking than PDA. These problems can be solved by PDA by increasing the number of stacks into it.

The number of stacks may vary according to the requirements of input tape (input alphabets) and the Parser machine. It can be increased in number of stacks in PDA if needed to solve complex problems of CFL (Context Free language). Two-stack PDA is an important variation of normal PDA. Push down Automata can be designed for acceptance of CFL by considering two aspects:

- Acceptance by Final state: The PDA accepts its inputs by consuming it and finally it enters in the final state.
- Acceptance by empty stack: IN the reading of input string from initial configuration for some PDA, the stack of PDA becomes empty.

5.1.2 Need of Two-Stack PDA

It is highly impossible to solve and support the all languages of context free grammar with normal PDA. Such as for the complex context free languages like $L = \{a^n b^n c^n ; n \geq 1\}$ the normal PDA will not work and hence we need another option, which might be complex and more time taking than PDA. These problems can be solved by PDA by increasing the number of stacks into it. The number of stacks may vary according to the requirements of input tape (input alphabets) and the Parser machine. With increased in the number of stacks the PDA can be designed for more complex CFLs. Since the Time Complexity of the stack operations (i.e. push and pop) is $O(1)$. Hence if we increase the number of stacks on a PDA, it won't affect more in the efficiency of the PDA. The push and pop operation can be performed in one iteration and hence the complexity of the PDA will not high.

5.1.3 Components of Two-Stack PDA

Two-stack PDA is a stack variation of normal PDA, the only difference is the number of stacks present in the automata machine PDA. The two stacks PDA machine can be represented as: $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, Z_1, F)$ where

- Q is a finite set of states
- Σ is a finite set which is called the input alphabet
- Γ is a finite set which is called the stack alphabet
- δ is a finite subset of $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$, the transition relation.
- $q_0 \in Q$ is the start state
- $Z_0 \in \Gamma$ is the initial stack symbol of stack1
- $Z_1 \in \Gamma$ is the initial stack symbol of stack2
- $F \subseteq Q$ is the set of accepting states

Example: Let the problem $L = \{a^n b^n c^n ; n \geq 0\}$. It is not solvable by normal PDA but we can solve this by two-stack PDA; in which:-

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b, c\}$$

$$\Gamma = \{Z_0, Z_1, Z_a, Z_b\}$$

$$q_0 = \text{Initial state}$$

Z_0 = Initial symbol of first stack

Z_1 = Initial symbol of the second stack

q_3 = Final State

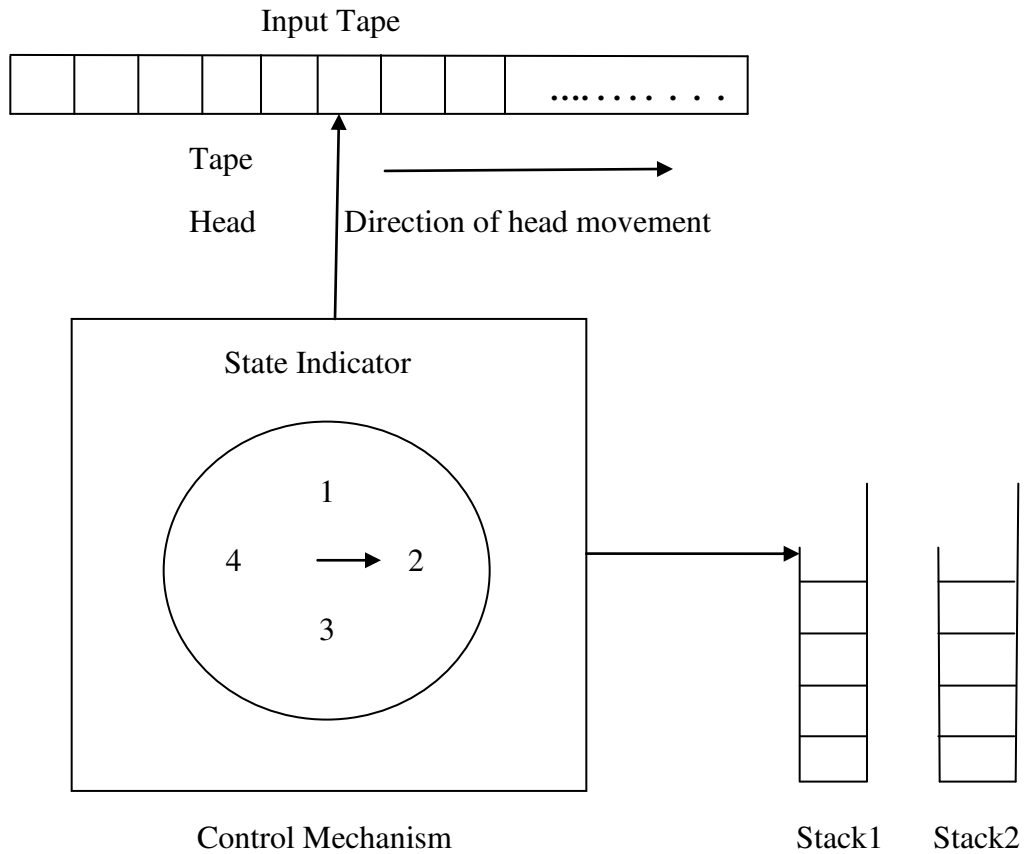


Figure 5.2 Two-stack Push Down Automata Model

Transitions are as follows:-

$$(q_0, a, Z_0, Z_1) = (q_0, Z_a Z_0, Z_1)$$

$$(q_0, a, Z_a, Z_1) = (q_0, Z_a Z_a, Z_1)$$

$$(q_0, b, Z_a, Z_1) = (q_1, Z_a, Z_b Z_1)$$

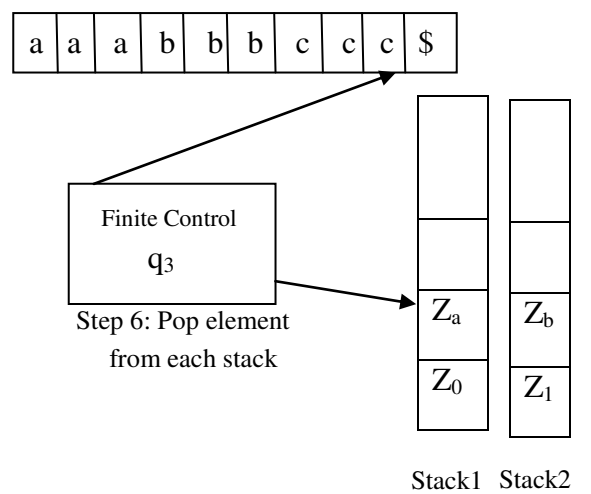
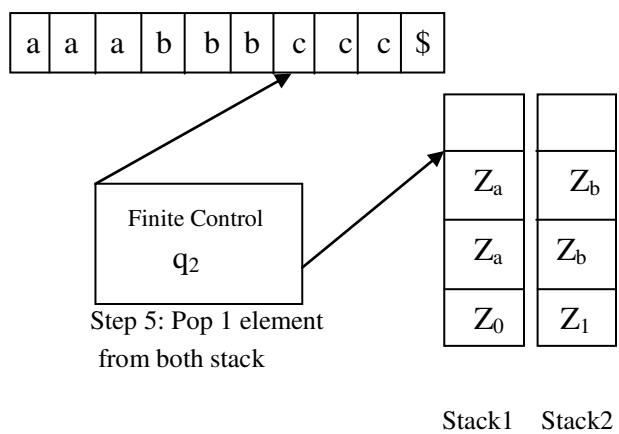
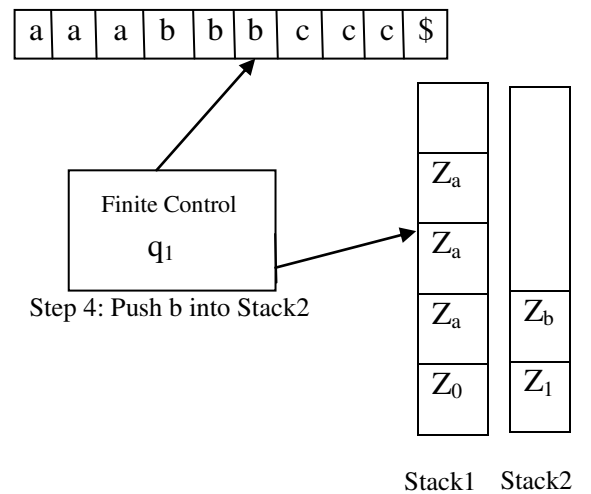
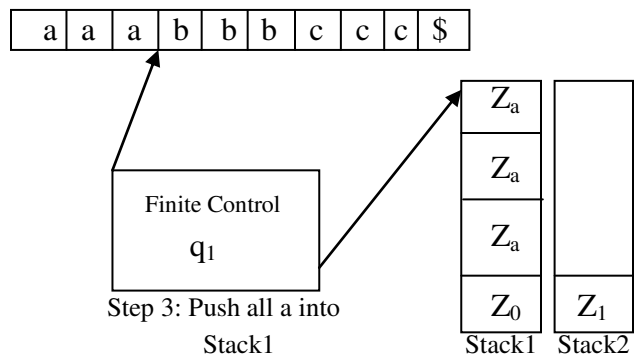
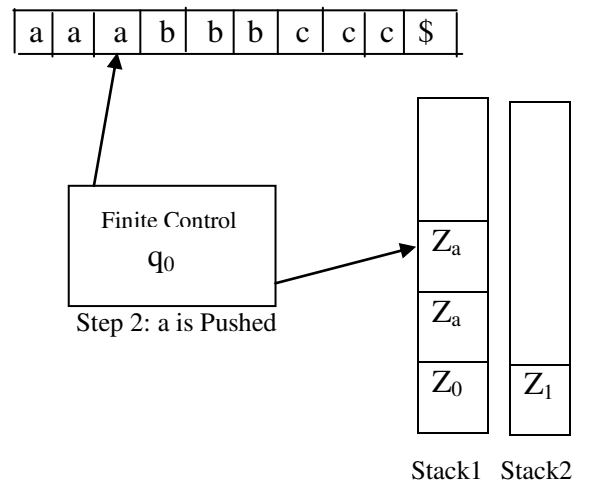
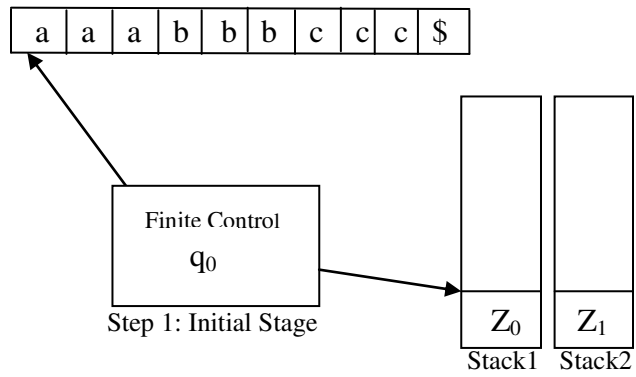
$$(q_1, b, Z_a, Z_b) = (q_1, Z_a, Z_b Z_b)$$

$$(q_1, c, Z_a, Z_b) = (q_2, \epsilon, \epsilon)$$

$$(q_2, c, Z_a, Z_b) = (q_2, \epsilon, \epsilon)$$

$$(q_2, \epsilon, Z_0, Z_1) = (q_3, \epsilon, \epsilon)$$

For the input string $w = a^3 b^3 c^3$; the two stack PDA is represented as:



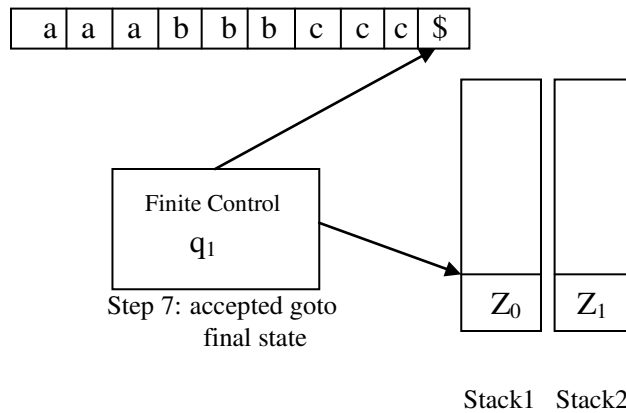


Figure 5.3 Use of Stack in Two-Stack PDA

5.2 REAL TIME CONSTRAINT NOTATION

A Real-time constraint notation are used in object-oriented language that provide sufficient real-time specifications as one may expect from a real-time language, while integrating these specifications within the object-oriented tapestry. Real Time Constraint Notation (RTCN) is used for representing and modelling the constraints in real time modelling.

Languages like Object-Z, OCL etc. are equipped with such refinement machinery. There is a high applicability of formal software verification in the development of security related software in the previous 10 years. Various quality levels are defined by ISO in ISO 15408 standard different levels of quality for software testing and verification. Common criteria project has been representing this standard with the members of security organization in the whole world. Various formal specifications and verifications tools have been introduced for quality and standard software development process. The area of concentration includes projects of security from verification of hardware circuit to verification of software driver. Specially, model tool checking has been seen as a successful tool.

5.2.1 Introduction of RTCN

We can perform formal verification of real time systems by RTCN. The functioning of the Real Time System is carried out by formal verification for ensuring efficient performance and functioning. The two-stack PDA Model verification by Sequence Diagrams in the form of a Formal Language And toll plaza database management system verification by Sequence Diagrams in the form of a formal language. Formal Description Languages support language

standardization program verification and software reliability. There are three distinct approaches to specify a programming language:

- Operational Semantics
- Axiomatic semantics
- Devotional semantics [1]

A language in the form of operation of an abstract machine is called operation semantics, which gives an abstract interpreter of the language. A language in terms of assertions and inference rules for reasoning about the programs is known as axiomatic semantics. The Axiomatic specification of a programming language will be mathematical theory for that language. A Devotional semantics is most useful devotional semantics of the well known programming language ADA, PASCAL and LISP can be found. The formal specification language is often used as a Meta - language for describing programming language semantics.

For model checking we need to have description of system and a specification of properties. Generally these properties are represented in Computational Tree Logic (CTL). For model specification we can have OCL based approach wherein we can have OCL specifications instead of CTL specifications. OCL extensions have been introduced by concepts based on time bounded variant of CTL. Hence, real time and model checking specifications are carried out by OCL. Once the constraints are specified for UML State Chart Diagrams (state space) they can be used for past oriented temporal logics and Activity Diagrams without any modifications. The key role has been played by description techniques with a modeling method which defines a various views of the systems. Few description techniques from UML, have been used which are specialized for enabling definition of semantics. They are as follows:

- Informal Text and Diagrams (ITD)
- Programming Language (PL)
- State Transition Diagram (STD)
- Specification Language (SL)
- Message Sequence Chart (MSE)
- Object Model (OM)

The above discussed documents are providing with mathematical system model based semantics. Using this semantics, a precise semantics for documents has achieved along with an

integrated one. This permits to represents transformation among documents along with the rigorous context conditions among various description techniques. The system development has been represented by the development graph which consist the documents in the form of nodes and dependencies among the direct arcs. Each document will contain all the information of its predecessor so the document state is captured to analyze whether it is redundant or important. This information is necessary for development process so that it can analyze requirements, design decisions and to permit changes in the requirements in a prescribed way.

5.2.2 Object Oriented Real Time Modelling

The significant aspects in object oriented real-time modelling can be identified as:

- The use of inheritance and redefinition of real time constraint through the inheritance hierarchy and extension of the inheritance of the state and behaviour of a class to include the definition of temporal constraints.
- The reuse of the temporal constraints specifications through the inheritance mechanism and across class boundaries.
- Time-abstraction seems like a natural approach to specifying timing constraints, where the constraint is defined at the class definition and then associated with the class implementation.

Software engineering and specifically real-time embedded software is still an emerging area. It is applied to enhance the complex systems and its modelling approaches are neither perfect nor reliable. Software models have a specific and remarkable advantage over the other engineering approaches like they could be used to automatically produce executable programs for specific platforms. Mathematical explanation of the modelling language semantics gives the effort to explain properly about system's characteristics and to forecast its behaviour and functioning. Adequate real-time software models could be developed, yet there is another problem of getting it automatically from a model and finally its implementation is a way that produces code which behaves appropriately as the model. An expressive, well founded transformation mechanism for automatic and proprietary accepting code generation design methodology has been developed. The development of the system must be based on the Platform Independent Model. This is

finally derived from Platform Specific Model (PSM). At the meta level queries, views and transformations are areas that would be important for wellness of MDA.

Formal verification in model refinement: The model refinement using formal verification can be best explored if models are created using the language with following features:

- Language has formal refinement mechanism.
- Language can evaluate all refinements possible from a given model.
- Language can prove that a particular model is outcome of refinements on a given model.

5.2.3 Characteristics

One of the choices of designers for developing real time System is UML, for enhancing UML notation in modeling of real time application several approaches have been developed. For applying additional constraints over UML model, the model developers use OCL. Presently, temporal constraints cannot be fully expressed using UML's real time extension and OCL. Also dynamic behavior involved in UML models (like state evolutions and transitions) cannot be specified significantly using OCL. But, real time systems need to ensure system behavior correctly and thus need these constraints, to be specified correctly. For this, an OCL extension is proposed which is capable of expressing time bounded constraints. UML provides a technique to present design decisions and requirements at various stages in the software development process. For modeling the behavior of system the level of abstraction should be used that gives adequate information for generating precise diagrams. Sequence Diagram (SD) and State Charts (SC). Sequence Diagram (SD) is much interpretable for humans to produce and explain, while State chart describes system behavior in depth.

Real Time Constraint Notation Model Terminology: Constraints in real time systems are modelled using Real Time Constraint Notations (RTCN).

- Sequence Diagram and Message Sequence Charts:

In UML, scenarios can be represented using two main kind of representing scenarios: Sequence diagram and Collaboration diagrams. Sequence Diagram focuses on order of

events with respect to time and the structure of interactions among objects are represented by collaboration diagrams.

➤ Scenario Composition:

It plays a significant role in description of Real Time System models. Program's structure must be reflected by the sequence diagrams.

➤ Defining Finite State Machine and State Charts:

FSM (Finite State Automata) defines a system by defining all the possible states and all the transitions that are possible among these states. But, for a complex system, we will have a large FSM which would be beyond the comprehension of people. FSM suffers with state explosion problems which can be controlled using State charts.

➤ Explaining State Chart Synthesis:

To ensure the completeness of information for precise cohesive diagram, the real time system model has been utilized. In software development process, first of all Sequence Diagrams are developed and hence, we compose the state chart from sequence diagram.

➤ Representing Sequence Diagram Composition:

UML sequence diagrams do not depict composition information. Message sequence charts use hierarchical graphs to represent the composition information.

➤ Deterministic Grammar:

The major issue in state chart composition utilizing just adequate information from couple of sequence diagrams for developing precise state charts. These state charts should not reflect ambiguous data. For a given sequence diagram, corresponding Context Free Grammar (CFG) will constitute a set of message – response pairs. Unique response/responses must be generated by an object corresponding to every message (or set of messages).

$$SD \rightarrow \text{message response } SD \mid \epsilon$$

Here ϵ depicts that no message or response is present. Objective here is to develop a State Diagram using context free grammar (CFG) of the form:

$$\text{message response} \rightarrow \alpha \text{ResponseA} \mid \beta \text{ResponseB} \mid x \text{ResponseC}$$

where α , β and x are specific sequences of messages.

➤ Representation of State Information, Data and Timing Information:

For generating a deterministic state chart, state information will be required. It may be possible that execution depends on some data values which are stored but does not represent any state or transition in the model. This additional information is depicted using pre and post conditions. The timing information, like duration of simulation should create a response of a real time system. The major goal of models is to support engineers with the interpretable concepts of a system, before receiving the expense and problem of actually producing it. Software engineering has a well established modeling approach and their application of models is identified as a useful and effective approach.

5.3 OBJECT CONSTRAINT LANGUAGE

The Object Constraint Language is capable of defining constraints over object-oriented system and thus is an expression language. In an object oriented system, the constraints are identified as restrictions imposed on the values. Object Management Group (OMG) having about 700 companies aims at providing a framework for development of applications (with the help of object oriented programming methodologies) and standards for object oriented design and analysis. Now-a-days software professionals are rapidly getting habitual with OCL. At times, UML diagrams fail at representing constraints (like on ions, pre and post conditions) and at this point OCL plays a significant role. Expressions which involve accessing of attributes, invoking operations etc can be constructed easily using OCL. The characters used in OCL lie in normal alphanumeric set and hence is regarded as a simple language. OCL has proved to be significant in large number of application thus exploring many under specified domains (involving UML and OCL). [1]

5.3.1 Introduction of OCL

The Object Constraint Language (OCL) is an expression language that enables one to describe constraints on object-oriented models and other object modeling artifacts. A constraint is a restriction on one or more values of (part of) of an object oriented model or system. OCL is the part of the Unified Modeling Language (UML), the OMG (Object Management Group, a consortium with a membership of more than 700 companies. The organization's goal is to provide a common framework for developing applications using object-oriented programming techniques) standard for object – oriented analysis and design. OCL has been used in a wide

variety of domains, and this has led to the identification of some under – specified areas in the relationship between OCL and UML.

Recently, Object Constraint Language (OCL) is being used for formalization of Object Oriented Language semantics. Apart from great acceptability of Unified Modeling Checker (UMC), Object Constraint Language (OCL) has got an excellent identification of applicability. OCL is mainly used for specifying constraints in UML diagrams (mainly on classes) and in behavioral diagrams (on guards). But still OCL cannot provide sufficient specification for the constraints over dynamic behavior of diagrams i.e. OCL cannot sufficiently specify constraints over state configurations of states, their evolution and transitions with time. Therefore, real time specifications are not possible using OCL. Equivalence and model checking method in formal verification have been used for few applications. OCL can be used for a number of different purposes:

- To specify invariants on classes and types in the class model.
- To specify type invariants for Stereotypes.
- To describe pre- and post- conditions on Operations and Methods.
- To describe guard.
- As a navigation language.
- To specify constraint on operations.

5.3.2 UML and OCL

The Object Constraint Language is a modeling language with which we can build software models. It is defined as a standard “add-on” to the Unified Modeling Language (UML). Every expression written in OCL relies on the type (i.e. the classes, interfaces and so on) that are defined in the UML diagrams. OCL expressions can be used anywhere in the model to indicate a value. An outstanding characteristic of OCL is its mathematical foundation. It is based on mathematical set theory and predicate logic and it has a formal mathematical semantics.

In OCL, UML operation semantics can be expressed using pre and post condition constraints. The pre condition says what must be true for the operation to meaningfully execute. The post condition expresses what is guaranteed to be true after execution completes about

- the return value
- any state changes (e.g. instance variables)

The development of software models are utilizing the OCL, hence regarded as a modeling language. OCL when used along with UML supports a lot more additional features. OMG is a standard for analyzing and designing in object oriented manner. Each expression which is represented in OCL is based on the types (whether classes or interfaces or any other) which are explained in the UML diagrams. Therefore UML will definitely be used in the OCL based applications. Models serve as building block for the development of software, a significant feature of MDA. A combined effort of OCL and UML produces consistent models.

5.3.3 Characteristics of OCL

A model has more than one class, required OCL statements to be a consistent model. With the help of UML diagrams many inconsistencies would persist. Therefore, OCL is a necessary language for developing efficient models.

- **Query and Constraint Language Description:** Earlier, OCL was assumed to be just limited to constraint implementation where constraints define the conditions when system data values and objects would be valid. In UML 2, constraints and statements in the UML designs can be defined using OCL. Taking an example, say we have a statement $2+7$. It is a genuine OCL statement of integer type, depicting integer value 9. An OCL expression can be used as constraint, if it is of type Boolean. OCL expressions represent a value. Which can be a simple value, (like an integer), a collection of values, may be a reference to an object, or a collection of references to objects. A Boolean value which represents a message in an interaction diagram or a constraint in a state chart can also be represented using an OCL expression. OCL has same capabilities as SQL. We can use a single OCL expression representing the whole body of a query operation. But, SQL is not a constraint language whereas OCL is a both a query and a constraint and language at the same time.
- **Foundation with Mathematical Logic:** OCL is based on predicate logic and set theory, making it significant. One who is familiar with mathematical notation can easily use to represent accurate statements but not everyone can interpret it. Hence, a mathematical notation cannot be considered as a standard language. For a modelling language we need to have precision using mathematics, as well as it should be as simple as natural language. Both the above requirements are conflicting, so we need to look for a balance

between these requirements. OCL provides this balance using mathematical concept and simple ASCII words. Therefore, we have an unambiguous language which is easy for both, users of object technology and their customers. One can also define syntax of OCL for himself provided that syntax should map to the language structures as defined in the standard.

- **Strongly Typed Language:** As OCL expressions specify the type of data hence it's a typed language. It is not necessary that every model is directly executed and hence there would w many OCL expression written for which corresponding systems will don't have executable versions. Even then, one can test an OCL expression. These expressions can be checked before execution (i.e. while modelling phase) and hence errors can be removed at modelling stage only.
- **Declarative Language:** OCL expressions describe what is to be accomplished rather than how to do it hence it comes under declarative language category. When an OCL expression is evaluated, it does not affect the system's state. The model developers do not go deep into how computations should be done, rather decisions are made at high level abstraction. Something should be accomplished depends on the implementation approach. One can mention the associations in the coding on the implementation approach. [24]

OCL is mainly focusing on the application of constraints in the UML diagrams. Object Oriented Languages and their semantics are well modelled by using Object Constraint Language (OCL). These OCL integrated UML is being utilized for the modelling of real time systems. For the formal verification of the developed models, such types of approaches are highly applicable. OCL has been utilized in various forms:

- Specification of constraint on operations
- To define guard
- As a navigation language
- To define pre and post- conditions on Operations and Methods
- Specification of type invariants for Stereotypes
- Specification of invariants on classes and types in the class model

CHAPTER 6

PROPOSED WORK

6.1 FLOW CHART OF PROPOSED PDA MODEL

A flowchart is a type of diagram that represents an algorithm workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. Flowcharts are used in designing and documenting simple processes or programs. Like other types of diagrams, they help visualize what is going on and thereby help understand a process, and perhaps also find flaws, bottlenecks, and other less-obvious features within it. This technique allows the author to locate the responsibility for performing an action or making a decision correctly, showing the responsibility of each organizational unit for different parts of a single process. Flowcharts depict certain aspects of processes and they are usually complemented by other types of diagram.

Types of Flowchart: Flowcharts can be modeled from the perspective of different user groups and that there are 4 general types:

- Document flowcharts, showing controls over a document-flow through a system
- Data flowcharts, showing controls over a data-flow in a system
- System flowcharts, showing controls at a physical or resource level
- Program flowchart, showing the controls in a program within a system

Notice that every type of flowchart focuses on some kind of control, rather than on the particular flow itself. In addition, many diagram techniques exist that are similar to flowcharts but carry a different name, such as UML activity diagrams.

Working of two-stack PDA model for maintaining record of the vehicles passing through a Toll Plaza in clustered way can be represented by the flow chart as shown in figure 6.1 below:

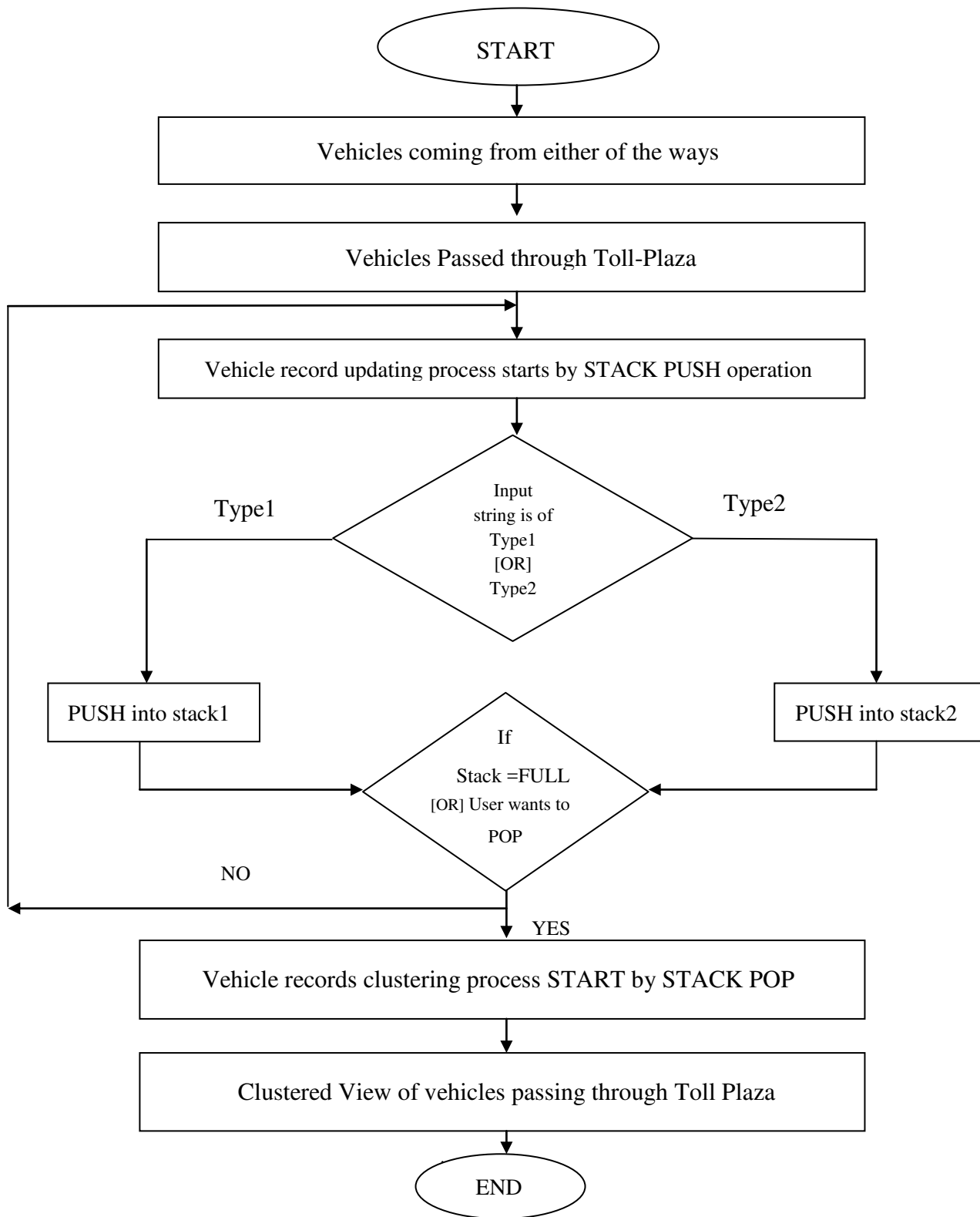


Figure 6.1 Flow chart for working of two-stack PDA model for vehicles record passing through a Toll Plaza

6.2 ALGORITHM

In mathematics and computer science, an algorithm is a self-contained step-by-step set of operations to be performed. Algorithms perform calculation, data processing and/or automated reasoning tasks. An algorithm is an effective method that can be expressed within a finite amount of space and time and in a well-defined formal language for calculating a function. Starting from an initial state and initial input (perhaps empty) the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing “output” and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic some algorithms, known as randomized algorithms incorporate random input. An algorithm is a procedure or formula for solving a problem. Simple as the definition of the notion of algorithm is, the concept of what it attempts to convey is a matter of debate and scientific research. An algorithm is a sequence of unambiguous instructions for solving a problem, i.e. for obtaining a required output for any legitimate input in a finite amount of time.

It means an algorithm is the step by step procedure designed to perform an operation, and which (like a map or flowchart) will lead to the sought result if followed correctly. Algorithms have a definite beginning and a definite end and a finite number of steps. An algorithm produces the same output information given the same input information and several short algorithms can be combined to perform complex tasks such as writing a computer program. A cookbook recipe, a diagnosis, a problem solving routine is some common examples of simple algorithms. Suitable for solving structured problems algorithms are, however, unsuitable for problems where value judgments are required. The proposed algorithm for our Two-Stack PDA model is given below:

```
While (input != empty) {  
    Read input[i]  
    If (input[i] == type1) {  
        If (stack2 != empty)  
            Change state  
        Push (input[i]) in stack1
```

```

        Count1 ++ }
Else if(input[i] == type2) {
    If (is first iteration)
        Change state
    Push (input[i]) in stack2
    Count2 ++ }
Else {
    If (is first iteration)
        Change state
    If (stack1 == empty || stack2 == empty)
        Break
    Else {
        Pop (stack1)
        Pop (stack2) }
    } }
If (input tape == empty && stack1 == empty && stack2 == empty)
    Input string accepted
Else if (input tape == empty && count1 == count2) {
    Free (stack1)
    Free (stack2)
    Input string accepted }
Else
    Input string is rejected
STOP.

```

6.3 STEPS INVOLVED

The following steps those are mentioned below can be modeled with the help of two-stack PDA with the help of the implementation of the above algorithm. The steps are following:

1. Read input tape
2. Check its type
 - 2.1. If input type = type1 then push into stack1
 - 2.2. Change state if stack 2 is NOT empty
 - 2.3. Increment count1 by one
3. if input type is of type2
 - 3.1. Change state if it is first iteration of type2
 - 3.2. Push input value into stack2
 - 3.3. Increment count2 by one
4. Else change state
 - 4.1. If it is first iteration of other type
 - 4.2. Check the stacks if stacks are empty then break the operation
 - 4.3. Else POP(stack1) and POP(stack2)
5. Follow the above steps till input tape is empty
6. If input tape is ended and both stacks are empty then accept input
7. If input tape is empty and count1 = count2 then accept input
8. If above operations fail then reject it.

6.4 PDA TRANSITIONS FOR VEHICLE RECORD UPDATE

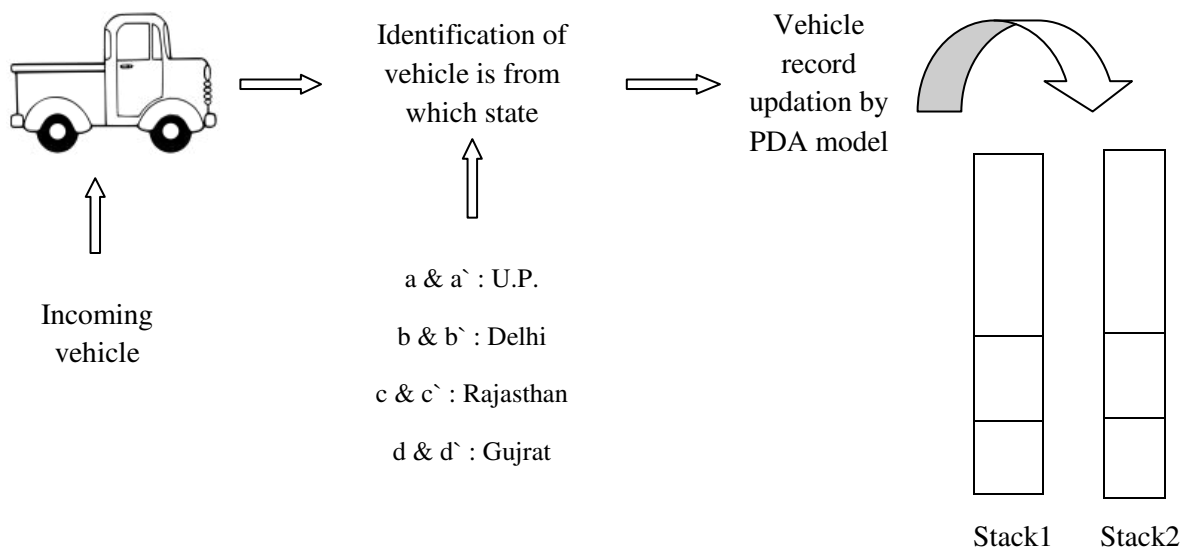


Figure 6.2 Incoming vehicles record updation process

Let us assume the input string is $abcd a' b' c' d'$. Now, to achieve this goal, we design the two-stack PDA as:

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, Z_1, \phi)$ where

➤ Q : is the set of states for maintaining the vehicles record passing through toll plaza i.e.

$$Q = \{q_0, q_{PUSH}, q_{POP}\}$$

➤ Σ : is the finite set of input alphabets for vehicles from different states i.e.

$$\Sigma = \{a, b, c, d, a', b', c', d'\} \text{ where:}$$

- a and a' represents: vehicles from U.P.
- b and b' represents: vehicles from Delhi
- c and c' represents: vehicles from Rajasthan
- d and d' represents: vehicles from Gujrat

➤ Γ : is finite set of stack alphabets for vehicles from different states i.e.

$$\Gamma = \{Z_0, Z_1, Z_a, Z_b, Z_c, Z_d\} \text{ where:}$$

- Z_0 : is the initial stack alphabet of stack1
- Z_1 : is the initial stack alphabet of stack2
- Z_a : is the stack alphabet PUSHED into or POPPED from stacks when vehicle is from U.P.
- Z_b : is the stack alphabet PUSHED into or POPPED from stacks when vehicle is from Delhi
- Z_c : is the stack alphabet PUSHED into or POPPED from stacks when vehicle is from Rajasthan
- Z_d : is the stack alphabet PUSHED into or POPPED from stacks when vehicle is from Gujrat

The PDA transitions for input string as mention above is:

- i. $\delta(q_0, a, Z_0, Z_1) = (q_{PUSH}, Z_a Z_0, Z_1)$
- ii. $\delta(q_{PUSH}, b, Z_a, Z_1) = (q_{PUSH}, Z_b Z_a, Z_1)$
- iii. $\delta(q_{PUSH}, c, Z_b, Z_1) = (q_{PUSH}, Z_c Z_b, Z_1)$
- iv. $\delta(q_{PUSH}, d, Z_c, Z_1) = (q_{PUSH}, Z_d Z_c, Z_1)$
- v. $\delta(q_{PUSH}, a', Z_d, Z_1) = (q_{PUSH}, Z_d, Z_a Z_1)$

- vi. $\delta(q_{PUSH}, b, Z_d, Z_a) = (q_{PUSH}, Z_d, Z_bZ_a)$
- vii. $\delta(q_{PUSH}, c, Z_d, Z_b) = (q_{PUSH}, Z_d, Z_cZ_b)$
- viii. $\delta(q_{PUSH}, d, Z_d, Z_c) = (q_{PUSH}, Z_d, Z_dZ_c)$

6.5 ID FOR VEHICLE RECORD UPDATE PROCESS

An instantaneous description (ID) for a PDA is a triple of the form (state, unconsumed input, stack contents). The instantaneous description (ID) of a PDA is represented by a triplet (q, w, s) where:

- q is the state
- w is unconsumed input
- s is the stack contents

Turnstile Notation: The “turnstile” notation is used for connecting pairs of ID’s that represent one or many moves of a PDA. The process of transition is denoted by the turnstile symbol “ \vdash ”. Consider a PDA (Q, Σ , Γ , δ , q_0 , Z_0 , F). A transition can be mathematically represented by the following turnstile notation: $(p, aw, T\beta) \vdash (q, w, \alpha b)$

This implies that while taking a transition from state p to state q, the input symbol ‘a’ is consumed and the top of the stack ‘T’ is replaced by a new string ‘ α ’. If we want zero or more moves of a PDA, we have to use the symbol (\vdash^*) for it. As per the rules, the ID for the input string as mention above is given as below:

$$\begin{aligned}
 & q_0, abcda\bar{b}\bar{c}\bar{d}, Z_0, Z_1 \vdash q_{PUSH}, bcda\bar{b}\bar{c}\bar{d}, aZ_0Z_1 \\
 & \vdash q_{PUSH}, cda\bar{b}\bar{c}\bar{d}, baZ_0, Z_1 \vdash q_{PUSH}, da\bar{b}\bar{c}\bar{d}, cbaZ_0, Z_1 \\
 & \vdash q_{PUSH}, a\bar{b}\bar{c}\bar{d}, dcbaZ_0, Z_1 \vdash q_{PUSH}, \bar{b}\bar{c}\bar{d}, dcbaZ_0, a\bar{Z}_1 \\
 & \vdash q_{PUSH}, \bar{c}\bar{d}, dcbaZ_0, b\bar{a}\bar{Z}_1 \vdash q_{PUSH}, \bar{d}, dcbaZ_0, c\bar{b}\bar{a}\bar{Z}_1 \\
 & \vdash q_{PUSH}, \epsilon, dcbaZ_0, d\bar{c}\bar{b}\bar{a}\bar{Z}_1
 \end{aligned}$$

6.6 PDA TRANSITION TO GENERATE CLUSTER VIEW

The vehicle log record update process, as shown in Figure 6.2, will continue till stack is full or user itself wants to pop. When either of the two pop conditions occurred, then transitions for cluster view generation start according to the following steps:

Step (I): When the pop condition came then \$ symbol is generated on the input tape and q_{PUSH} is converted to q_{POP}

- i. $\delta(q_{PUSH}, \$, d, d') = (q_{POP}, d, d')$
- ii. $\delta(q_{PUSH}, \$, c, c') = (q_{POP}, c, c')$
- iii. $\delta(q_{PUSH}, \$, b, b') = (q_{POP}, b, b')$
- iv. $\delta(q_{PUSH}, \$, a, a') = (q_{POP}, a, a')$

Step (II): The following transitions occurred for the generation of cluster view with the help of STACK POP operations as follows:

- i. $\delta(q_{POP}, \$, a, a') = (q_{POP}, \epsilon, \epsilon)$
- ii. $\delta(q_{POP}, \$, b, b') = (q_{POP}, \epsilon, \epsilon)$
- iii. $\delta(q_{POP}, \$, c, c') = (q_{POP}, \epsilon, \epsilon)$
- iv. $\delta(q_{POP}, \$, d, d') = (q_{POP}, \epsilon, \epsilon)$

Step (III): When the top of stack1 = Z_0 and stack2 = Z_1 , it implies that all the stack alphabets corresponding to different states are popped out and top of the stack1 is Z_0 and stack2 is Z_1 then state q_{POP} is transit to q_{PUSH} and the PDA model is ready to update the new vehicle log record into stacks. Step III condition in two-stack PDA model can be completed with the help of transition:

- i. $\delta(q_{POP}, \$, Z_0, Z_1) = (q_{PUSH}, Z_0, Z_1)$

CHAPTER 7

BENEFITS OF PROPOSED MODEL

In this project we develop an interactive and communicative framework that can maintain a record of the vehicles coming from various states. It focuses on development of a feature to view vehicles coming from which territory in cluster view. We can calculate the total revenue generated territory wise and generate a cluster view of vehicle logs.

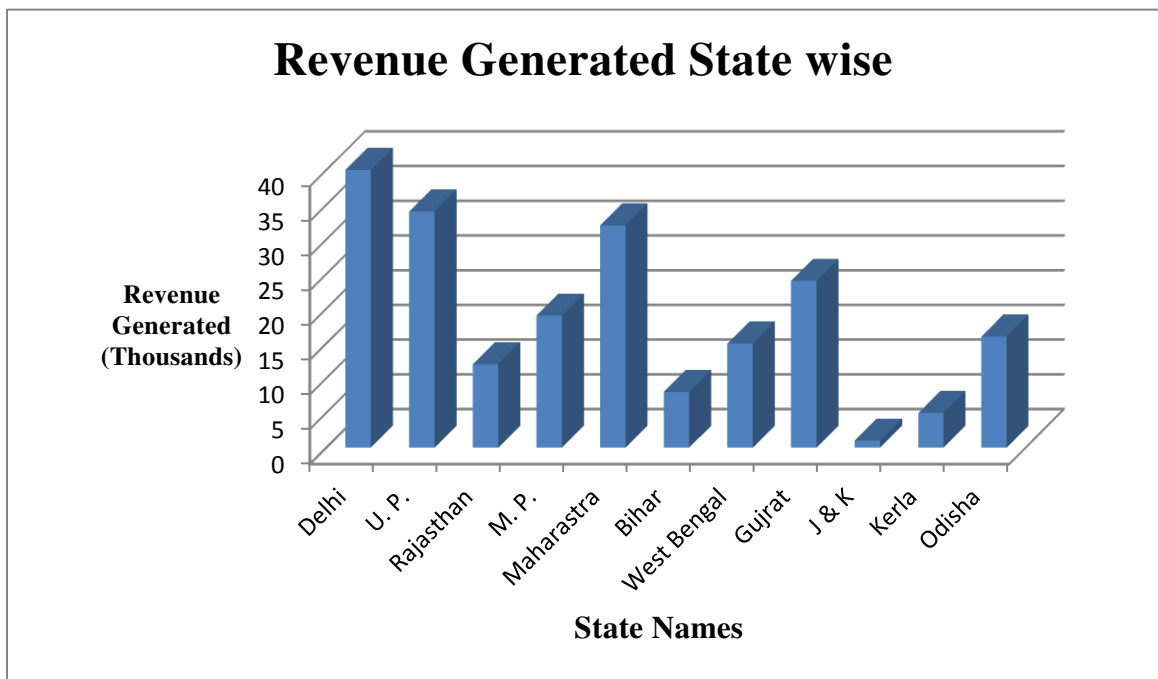


Figure 7.1 Output Clustered View of proposed model

It helps in comparative analysis, having the following importance:

- Financial leakage control.
- Vehicle tracking.
- Congestion Management.
- Low cost and easy to implement.

Human Visualization:

The working of Push-down Automata model for vehicle passing through the particular highway records storage in cluster way and can be represented by the graphical way as well.

Better Interaction and Analysis:

The interaction between the user and the framework can help in achieving better retrieval results and interaction ranges from simply allowing the user to calculate the total tax generated from a specific state.

Efficient Result:

In general, the automata theory always provides the more efficient result comparatively. By model checking we will analysis and prove that the developed framework giving result more efficiently in comparison to other existing models.

CHAPTER 8

IMPLEMENTATION WORK

8.1 DATABASE

A database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies. In this project, we are using Oracle 11g as the backend database. As per the each and every Online Toll Tax Database Management model there would be a database which would be working as backend of the system. Systematically organized or structured repository of indexed information (usually as a group of linked data files) that allows easy retrieval, updating, analysis, and output of data. Stored usually in a computer, this data could be in the form of graphics, reports, scripts, tables, text etc representing almost every kind of information. Most computer applications are databases at their core.

The Oracle DBMS can store and execute stored procedures and functions within itself. PL/SQL (Oracle Corporation's proprietary procedural extension to SQL), or the object-oriented language Java can invoke such code objects and/or provide the programming structures for writing them. Oracle database management tracks its computer data storage with the help of information stored in the SYSTEM tablespace. The SYSTEM table space contains the data dictionary and often (by default) indexes and clusters. A data dictionary consists of a special collection of tables that contains information about all user-objects in the database. Since version 8i, the Oracle RDBMS also supports "locally managed" tablespaces that store space management information in bitmaps in their own headers rather than in the SYSTEM tablespace (as happens with the default "dictionary-managed" tablespaces). Version 10g and later introduced the SYSAUX tablespace, which contains some of the tables formerly stored in the SYSTEM tablespace, along with objects for other tools such as OEM, which previously required its own tablespace.

8.1.1 Tables

Tables are the basic unit of data storage in an Oracle Database. Data is stored in rows and columns. We define a table with a table name, such as employees and a set of columns. We can give each column a column name, such as employee_id, last_name and job_id; a data type, such as VARCHAR2, DATE or NUMBER and a width. The width can be predetermined by the data type as in DATE. If columns are of the NUMBER data type, define precision and scale instead of width. A row is a collection of column information corresponding to a single record. We can specify rules for each column of a table. These rules are called integrity constraints. One example is a NOT NULL integrity constraint. This constraint forces the column to contain a value in every row. We can also invoke transparent data encryption to encrypt data before storing it in the data file. Then, if users attempt to circumvent the database access control mechanisms by looking inside data files directly with operating system tools, encryption prevents these users from viewing sensitive data.

After we create a table, insert rows of data using SQL statements. Table data can then be queried deleted or updated using SQL. Before creating a table, we should also determine whether to use integrity constraints. Integrity constraints can be defined on the columns of a table to enforce the business rules of your database automatically. In this project we are working with two tables which are discussed below.

8.1.1.1 Main Table

This is the main table of the Toll Plaza System, all the passing vehicles information are stored into this. By accessing this main table anyone can retrieve the required information as per their requirement. A snapshot of this table is as below:

8.1.2 Triggers

8.1.2.1 Definition

A database trigger is a compiled stored program unit, written in either PL/SQL or Java, that Oracle Database invokes (“fires”) automatically whenever one of the following operations occurs:

- DML statements on a particular table or view, issued by any user
- DML statements modify data in schema objects. For example, inserting and deleting rows are DML operations.
- DDL statements issued either by a particular user or any user
- DDL statements define schema objects. For example, creating a table and adding a column are DDL operations.
- Database events

Oracle allows us to define procedures that are implicitly executed when an INSERT, UPDATE, or DELETE statement is issued against the associated table. These procedures are called database triggers. Triggers are similar to stored procedures, “Procedures and Packages”. A trigger can include SQL and PL/SQL statements to execute as a unit and can invoke stored procedures. However, procedures and triggers differ in the way that they are invoked. While a procedure is explicitly executed by a user, application, or trigger, one or more triggers are implicitly fired (executed) by Oracle when a triggering INSERT, UPDATE, or DELETE statement is issued, no matter which user is connected or which application is being used. Triggers are schema objects that are similar to subprograms but differ in the way they are invoked.

8.1.2.2 Types of Triggers

Triggers can be categorized according to their means of invocation and the type of actions they perform. Oracle Database supports the following types of triggers:

- **Row Triggers:** A row trigger fires each time the table is affected by the triggering statement. For example, if a statement updates multiple rows, then a row trigger fires

once for each row affected by the UPDATE. If a triggering statement affects no rows, then a row trigger is not run. Row triggers are useful if the code in the trigger action depends on data provided by the triggering statement or rows that are affected.

- **Statement Triggers:** A statement trigger is fired once on behalf of the triggering statement, regardless of the number of rows affected by the triggering statement. For example, if a statement deletes 100 rows from a table, a statement-level DELETE trigger is fired only once. Statement triggers are useful if the code in the trigger action does not depend on the data provided by the triggering statement or the rows affected.
- **Instead of Triggers:** An instead of trigger is fired by Oracle Database instead of executing the triggering statement. These triggers are useful for transparently modifying views that cannot be modified directly through DML statements.
- **Event Triggers:** We can use triggers to publish information about database events to subscribers. Event triggers are divided into the following categories:
 - A system event trigger can be caused by events such as database instance start up and shutdown or error messages.
 - A user event trigger is fired because of events related to user logon and logoff, DDL statements, and DML statements.

BEFORE vs. AFTER Triggers

When defining a trigger, we can specify the trigger timing. That is, we can specify whether the trigger action is to be executed before or after the triggering statement. BEFORE and AFTER apply to both statement and row triggers.

BEFORE Triggers: BEFORE triggers execute the trigger action before the triggering statement.

This type of trigger is commonly used in the following situations:

- BEFORE triggers are used when the trigger action should determine whether the triggering statement should be allowed to complete. By using a BEFORE trigger for this purpose, you can eliminate unnecessary processing of the triggering statement and its eventual rollback in cases where an exception is raised in the trigger action.
- BEFORE triggers are used to derive specific column values before completing a triggering INSERT or UPDATE statement.

AFTER Triggers: AFTER triggers execute the trigger action after the triggering statement is executed. AFTER triggers are used in the following situations:

- AFTER triggers are used when you want the triggering statement to complete before executing the trigger action.
- If a BEFORE trigger is already present, an AFTER trigger can perform different actions on the same triggering statement.

Timing for Triggers: We can define the trigger timing i.e. whether the trigger action is to be run before or after the triggering statement. A simple trigger is a single trigger on a table that enables you to specify actions for exactly one of the following timing points:

- Before the firing statement.
- Before each row affected by the firing statement.
- After each row affected by the firing statement.
- After the firing statement

For statement and row triggers, a BEFORE trigger can enhance security and enable business rules before making changes to the database. The AFTER trigger is ideal for logging actions. A compound trigger can fire at multiple timing points. Compound triggers help program an approach in which the actions that you implement for various timing points share common data.

8.1.2.3 Advantages of Triggers

The correct use of triggers enables us to build and deploy applications that are more robust and that use the database more effectively. We can use triggers to:

- Automatically generate derived column values.
- Prevent invalid transactions.
- Provide auditing and event logging.
- Record information about table access

We can use triggers to enforce low-level business rules common for all client applications. For example, several applications may access the employees table. If a trigger on this table

ensures the format of inserted data, then this business logic does not need to be reproduced in every client. Because the trigger cannot be circumvented by the application, the business logic in the trigger is used automatically. Excessive use of triggers can result in complex interdependencies that can be difficult to maintain in a large application. Oracle automatically manages the dependencies of a trigger on the schema objects referenced in its trigger action. The dependency issues for triggers are the same as dependency issues for stored procedures. In releases earlier than 7.3, triggers were kept in memory. In release 7.3, triggers are treated like stored procedures; they are inserted in the data dictionary. Like procedures, triggers are dependent on referenced objects. Oracle automatically manages dependencies among objects.

8.1.2.4 Creation of Triggers

The CREATE TRIGGER statement creates or replaces a database trigger. A PL/SQL trigger has the following general syntactic form:

```
CREATE TRIGGER trigger_name
    Triggering_statement
    [trigger_restriction]
BEGIN
    Triggered_action
END;
/
```

A PL/SQL trigger has the following basic components:

- Trigger name: The name must be unique with respect to other triggers in the same schema.
- The trigger event or statement: A triggering event or statement is the SQL statement, database event, or user event that causes a trigger to be invoked.
- Trigger restriction: A trigger restriction specifies a boolean expression that must be true for the trigger to fire.

- Triggered action: A triggered action is the procedure that contains the SQL statements and code to be run when a triggering statement is issued and the trigger restriction evaluates to true.

Execution of Triggers: Oracle Database executes a trigger internally using the same steps as for subprogram execution. The only subtle difference is that a user has the right to fire a trigger if he or she has the privilege to run the triggering statement. With this exception, the database validates and runs triggers the same way as stored subprograms.

Storage of Triggers: Oracle Database stores PL/SQL triggers in compiled form in a database schema, just like PL/SQL stored procedures. When a CREATE TRIGGER statement commits, the compiled PL/SQL code is stored in the database and the source code of the PL/SQL trigger is removed from the shared pool.

In this project, a BEFORE INSERT trigger is applied on the MainTable which fires automatically when ever any new entry made to the MainTable. A snapshot of created trigger is as below:

```
1 create or replace trigger MainTable_Dummy
2 before insert
3 on MainTable for each row
4 declare
5     x MainTable.Vehicle_Number%type;
6     y char(2);
7 begin
8     x := :new.Vehicle_Number;
9     y := substr(x, 1, 2);
10    insert into Dummy values(y, :new.Vehicle_Type);
11 end;
12 /
13
14
```

Figure 8.3 Screenshot of created Trigger

8.2 JAVA APIs

An application programming interface (API), in the context of Java, is a collection of prewritten packages, classes, and interfaces with their respective methods, fields and constructors. In Java, most basic programming tasks are performed by the API's classes and packages, which are helpful in minimizing the number of lines written within pieces of code. The Java API, included with the JDK, describes the function of each of its components. In Java programming, many of these components are pre-created and commonly used. Thus, the programme is able to apply prewritten code via the Java API.

Java API is a list of all classes that are part of the Java development kit (JDK). It includes all Java packages, classes, and interfaces, along with their methods, fields, and constructors. These prewritten classes provide a tremendous amount of functionality to a programmer. A complete listing of all classes in Java API can be found at Oracle's website: <http://docs.oracle.com/javase/7/docs/api/>. Java Development Kit (JDK) is comprised of three basic components, as follows:

- Java compiler
- Java Virtual Machine (JVM)
- Java Application Programming Interface (API)

8.2.1 JAVA Introduction

Java is a computer programming language that is concurrent, class-based, object-oriented and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers “write once, run anywhere” (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2014, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform.

8.2.2 Stack

Stack is a subclass of Vector that implements a standard last-in, first-out stack. Stack only defines the default constructor, which creates an empty stack. Stack includes all the methods defined by Vector and adds several of its own. The Stack class represents a last-in-first-out (LIFO) stack of objects. The usual push and pop operations are provided, as well as a method to peek at the top item on the stack, a method to test for whether the stack is empty, and a method to search the stack for an item and discover how far it is from the top. In this project we will be creating two classes for stack since our proposed model is two stack PDA. The first stack will perform PUSH operation as any vehicle came into, similarly the second stack will also do PUSH operation when another vehicle came from opposite direction as earlier. When we want to see the graphical view of vehicles passed through toll plaza then POP operation will take place.

8.2.3 JDBC

The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database. JDBC helps you to write Java applications that manage these three programming activities:

1. Connect to a data source, like a database
2. Send queries and update statements to the database
3. Retrieve and process the results received from the database in answer to your query

The following simple code fragment gives a simple example of the above three steps:

```
public void connectToAndQueryDatabase(String username, String password) {
    Connection con = DriverManager.getConnection(
        "jdbc:myDriver:myDatabase", username, password);
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("some SQL query");
    while (rs.next())
    {
        // some code goes here
    }
}
```


JDBC Product Components: JDBC includes four components which are illustrated below:

1. **The JDBC API:** The JDBC API provides programmatic access to relational data from the Java programming language. Using the JDBC API, applications can execute SQL statements, retrieve results, and propagate changes back to an underlying data source. The JDBC API can also interact with multiple data sources in a distributed, heterogeneous environment. The JDBC 4.0 API is divided into two packages: `java.sql` and `javax.sql`. Both packages are included in the Java SE and Java EE platforms.
2. **JDBC Driver Manager:** The `JDBC DriverManager` class defines objects which can connect Java applications to a JDBC driver. `DriverManager` has traditionally been the backbone of the JDBC architecture. The Standard extension packages `javax.naming` and `javax.sql` let you use a `DataSource` object registered with a Java Naming and Directory Interface (JNDI) naming service to establish a connection with a data source. You can use either connecting mechanism, but using a `DataSource` object is recommended whenever possible.
3. **JDBC Test Suite:** The JDBC driver test suite helps you to determine that JDBC drivers will run your program. These tests are not comprehensive or exhaustive, but they do exercise many of the important features in the JDBC API.
4. **JDBC-ODBC Bridge:** The Java Software bridge provides JDBC access via ODBC drivers. Note that you need to load ODBC binary code onto each client machine that uses this driver. As a result, the ODBC driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in three-tier architecture.

JDBC Architecture: The JDBC API supports both two-tier and three-tier processing models for database access.

In the two-tier model, a Java applet or application talks directly to the data source. This requires a JDBC driver that can communicate with the particular data source being accessed.

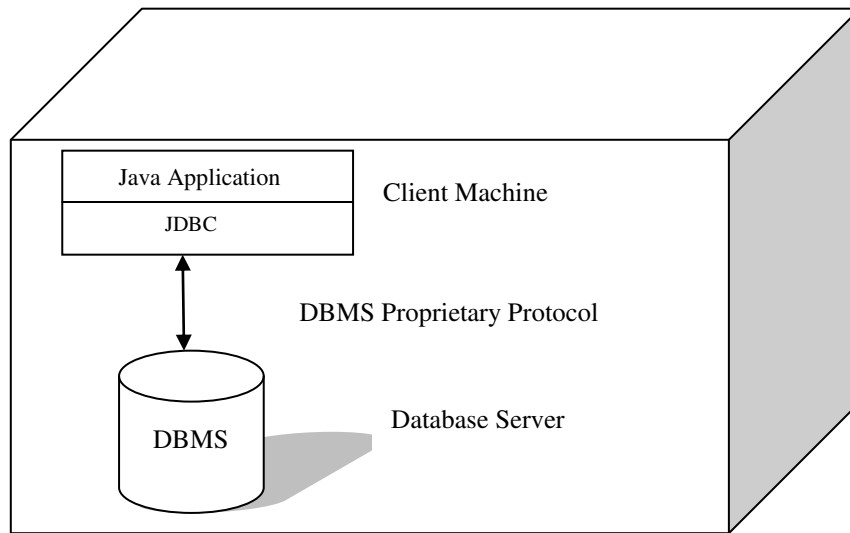


Figure 8.4 Two-tier Architecture for Data Access

In the three-tier model, commands are sent to a “middle tier” of services, which then sends the commands to the data source. The data source processes the commands and sends the results back to the middle tier, which then sends them to the user.

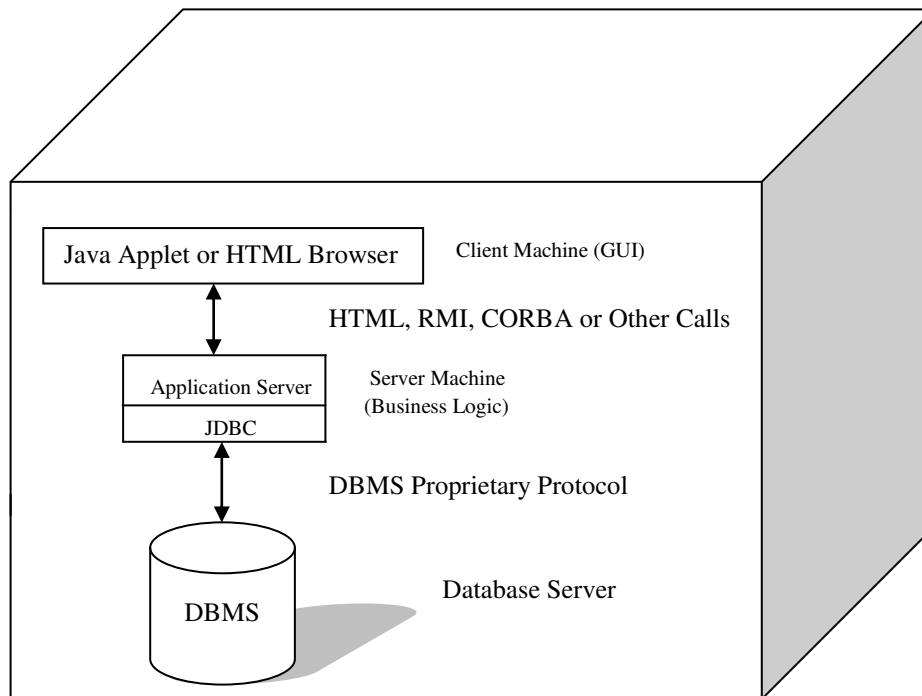


Figure 8.5 Three-tier Architecture for Data Access.

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections. JDBC connections support creating and executing statements. These

may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

- Statement - the statement is sent to the database server each and every time.
- PreparedStatement - the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.
- CallableStatement - used for executing stored procedures on the database.

DML statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information. Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types. There is an extension to the basic JDBC API in the javax.sql. JDBC connections are often managed via a connection pool rather than obtained directly from the driver.

JDBC Drivers: JDBC driver JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. There are commercial and free drivers available for most relational database servers. These drivers fall into one of the following types:

- Type 1 that calls native code of the locally available ODBC driver.
- Type 2 that calls database vendor native library on a client side. This code then talks to database over network.
- Type 3, the pure-java driver that talks with the server-side middleware those then talks to database.
- Type 4, the pure-java driver that uses database native protocol.

8.2.4 Swing

Swing is the primary JAVA GUI widget toolkit. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists. Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element. The Internet Foundation Classes (IFC) was a graphics library for Java originally developed by Netscape Communications Corporation and first released on December 16, 1996. On April 2, 1997, Sun Microsystems and Netscape Communications Corporation announced their intention to incorporate IFC with other technologies to form the Java Foundation Classes. The "Java Foundation Classes" were later renamed "Swing".

Swing is a highly modular-based architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces. Users can provide their own custom implementation(s) of these components to override the default implementations using Java's inheritance mechanism. Swing is a component-based framework, whose components are all ultimately derived from the `javax.swing.JComponent` class.

8.2.5 JFrame

`JFrame` is a JAVA swing public class which is a top level container class just like `JApplet` and `JDialog` which implies that it is a top level container, `JFrame` component is the root of the containment hierarchy and has a content pane that contains the visible components, directly or indirectly, in that top-level container's GUI. `BorderLayout` is the default `ContentPane` set for the `JFrame` component. When the user closes the window, `JFrame` component is hidden. In order to let the `JFrame` behave similar to AWT `Frame` instance, the following code fragment is used: `setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);`

Any children of the JFrame have the contentPane as its parent. Following are the constructors used in creating JFrame components:

- JFrame(): which creates a new frame that is initially invisible.
- JFrame(GraphicsConfiguration gc): which creates a Frame in the specified GraphicsConfiguration of a screen device with a blank title.
- JFrame(String title): which creates a new, initially invisible Frame with the specified title.
- JFrame(String title, GraphicsConfiguration gc): which creates a JFrame with the specified title and the specified GraphicsConfiguration of a screen device.

8.2.6 Util

Java.util package contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes. It contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array). The package java.util contains a number of useful classes and interfaces. Although the name of the package might imply that these are utility classes, they are really more important than that. In fact, Java depends directly on several of the classes in this package, and many programs will find these classes indispensable. The classes and interfaces in java.util include:

- The Vector class, which supports variable-length arrays.
- The StringTokenizer class for parsing strings into distinct tokens separated by delimiter characters.
- The EventObject class and the EventListener interface, which form the basis of the new AWT event model in Java 1.1.
- The Calendar and TimeZone classes in Java. These classes interpret the value of a Date object in the context of a particular calendar system.
- The class Date provides a convenient way to represent and manipulate time and date information. Dates may be constructed from a year, month, day of month, hour, minute, and second, and those six components, as well as the day of the week, may be extracted from a date. Time zones and daylight saving time are properly accounted for.

8.2.7 Events and Event Handling

Any program that uses GUI (graphical user interface) such as Java application written for windows, is event driven. Event describes the change of state of any object. Example: Pressing a button, Entering a character in Textbox. Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen. Some of the important even classes ant their listener interfaces are listed below:

Table 8.1 Event class and their Listener Interface

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

Components of Event Handling: Event handling has three main components that are given below:

- Events: An event is a change of state of an object.
- Events Source: Event source is an object that generates an event.
- Listeners: A listener is an object that listens to the event. A listener gets notified when an event occurs.

Types of Event: The events can be broadly classified into two categories:

- **Foreground Events:** Those events which require the direct interaction of user. They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page etc.
- **Background Events:** Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

Event Handling

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism has the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

- **Source:** The source is an object on which event occurs. Source is responsible for providing information of the occurred event to its handler. Java provide as with classes for source object.
- **Listener:** It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received, the listener processes the event and then returns.

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to the separate piece of code. In this model, listener needs to be registered with the source object so that the listener can receive the event notification. This is an efficient way of handling

the event because the event notifications are sent only to those listeners that want to receive them.

Steps involved in event handling:

- The User clicks the button and the event is generated.
- Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
- Event object is forwarded to the method of registered listener class.
- The method is now got executed and returns.
- Steps to perform Event Handling.

Following steps are required to perform event handling:

- Implement the Listener interface and overrides its methods.
- Register the component with the Listener.

For registering the component with the Listener, many classes provide the registration methods. For example:

- Button

```
public void addActionListener(ActionListener a){...}
```
- MenuItem

```
public void addActionListener(ActionListener a){...}
```
- TextField

```
public void addActionListener(ActionListener a){...}
public void addTextListener(TextListener a){...}
```
- TextArea

```
public void addTextListener(TextListener a){...}
```
- Checkbox

```
public void addItemListener(ItemListener a){...}
```
- Choice

```
public void addItemListener(ItemListener a){...}
```
- List


```
public void addActionListener(ActionListener a){...}  
public void addItemListener(ItemListener a){...}
```

Any number of event listener objects can listen for all kinds of events from any number of event source objects. For example, a program might create one listener per event source or a program might have a single listener for all events from all sources. A program can even have more than one listener for a single kind of event from a single event source. Multiple listeners can register to be notified of events of a particular type from a particular source. Also, the same listener can listen to notifications from different objects. Each event is represented by an object that gives information about the event and identifies the event source. Event sources are often components or models, but other kinds of objects can also be event sources.

8.2.8 Graph (JFreeChart Library)

JFreeChart is a free 100% Java chart library created by David Gilbert. JFreeChart makes it easy for developers to display professional quality charts in their applications. JFreeChart requires the JCommon class library. JFreeChart is a free chart library for Java that can generate a wide variety of charts for use in both client (Swing and JavaFX) and server (web) applications. JFreeChart supports pie charts (2D and 3D), bar charts (horizontal and vertical, regular and stacked), line charts, scatter plots, time series charts, high-low-open-close charts, candlestick plots, Gantt charts, combined plots, thermometers, dials and more. JFreeChart can be used in client-side and server-side applications.

JFreeChart Features: The major features of JFreeChart library are listed below -

- It is open source and 100% free. It is distributed under GNU Lesser General Public License (LGPL), which permits its usage in commercial applications without any cost.
- It comes with well documented API which makes it quite easy to use.
- It supports a wide range of chart types like Pie Chart, Line Chart, Bar Chart, Area Chart etc.
- JFreeChart is easy to extend and can be used in both client-side and server-side applications.
- It supports multiple output formats like PNG, JPEG, PDF, SVG etc.
- It allows extensive customizations of charts.

Prerequisites for Use

- Java: JFreeChart is written entirely in Java and uses Java 2D API for drawing. The current version of JFreeChart would work with JRE 1.4.2 or later.
- JFreeChart requires JCommon class library. It has common classes used by JFreeChart to provide global utility functions.

Getting Started: Creating charts with JFreeChart is a simple three steps process illustrated below:

- First create a dataset containing the data to be displayed in chart. The type of dataset varies with type of chart to be created.
- Next step is to create a JFreeChart object for the particular type of chart we wanted to create. We need to pass the dataset along with other parameters while creating this object.
- The last step is to initiate the process of drawing this chart on the target output like panel, web page etc.

JFreeChart's extensive feature set includes:

- A consistent and well-documented API, supporting a wide range of chart types.
- A flexible design that is easy to extend, and targets both server-side and client-side applications.
- It support for many output types, including Swing and JavaFX components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG).
- JFreeChart is open source or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

Create Dataset: The dataset used by bar chart is of type `org.jfree.data.category.CategoryDataset`. Similar to default implementation for pie chart dataset, there is default implementation class for this dataset as well. The class is named as `org.jfree.data.category.DefaultCategoryDataset`. This can be used as follows:

```
DefaultCategoryDataset objDataset = new DefaultCategoryDataset();  
objDataset.setValue(pass the required arguments);
```

Create Bar Chart: Similar to the pie chart creation method, ChartFactory has a method to create bar chart as well. The method name is createBarChart(). This method has few additional properties. These are used to customize the bar chart. The following code snippet is used for that:

```
JFreeChart objChart = ChartFactory.createBarChart(  
    "Chart Title",  
    "Domain axis label",  
    "Range axis label",  
    chartData, // Chart Data  
    PlotOrientation.VERTICAL, // orientation  
    true // include legend  
    true // include tooltips  
    false // include URLs);
```

Displaying Chart: Since we are going to display the chart in frame as well hence the display part of code would remain the same. We just have to pass the bar chart instance reference instead of pie chart. The following code snippet is used for that:

```
ChartFrame frame = new ChartFrame("Demo", objChart);  
frame.pack();  
frame.setVisible(true);
```

8.3 IMPLEMENTATION RESULT

After running the project the following window will appear to the user, which consist various options as per the requirement:

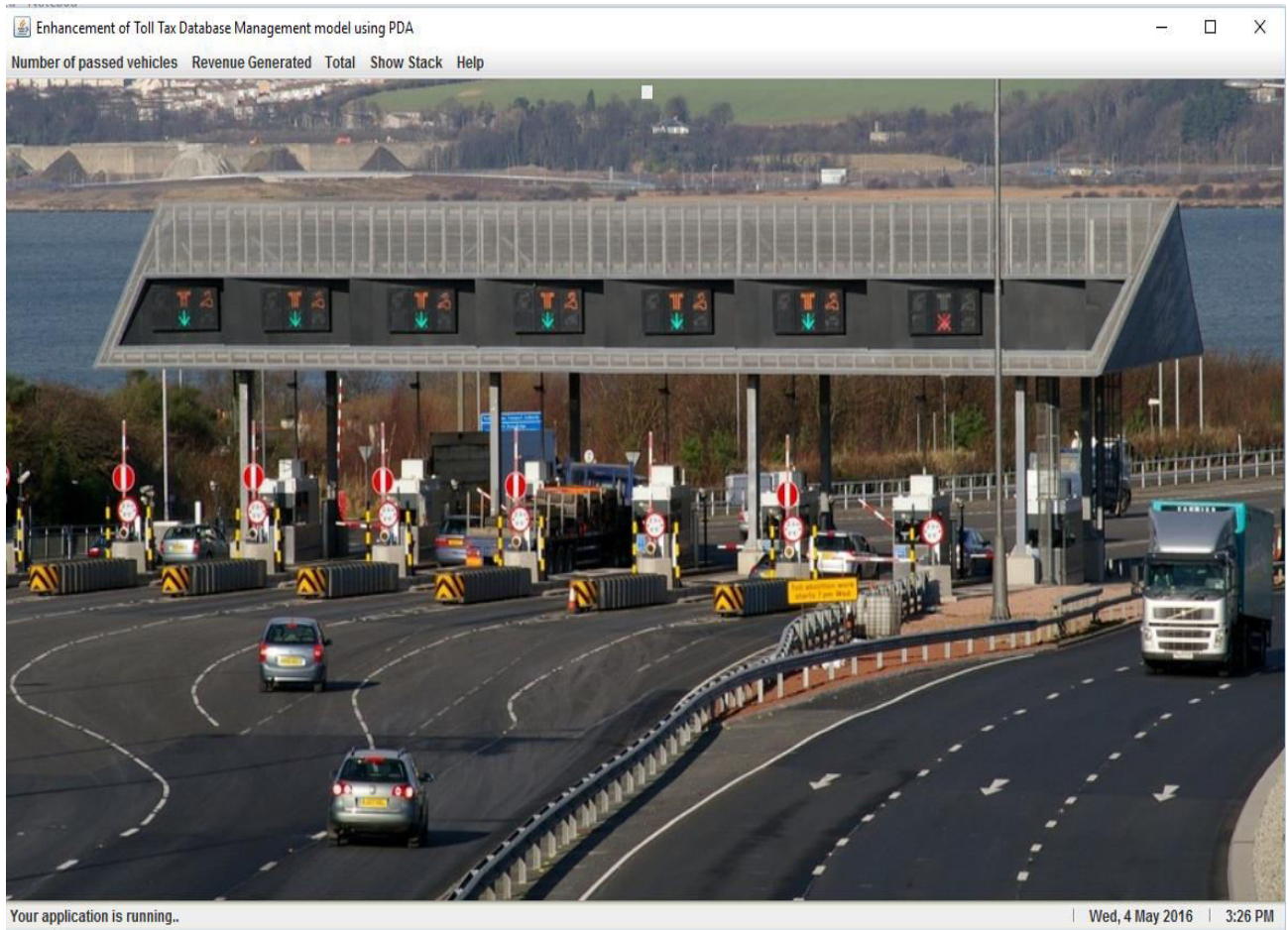


Figure 8.6 Screenshot of Main Window

8.3.1 Number of Passed Vehicles

8.3.1.1 Show Graph of Type 1 Vehicles

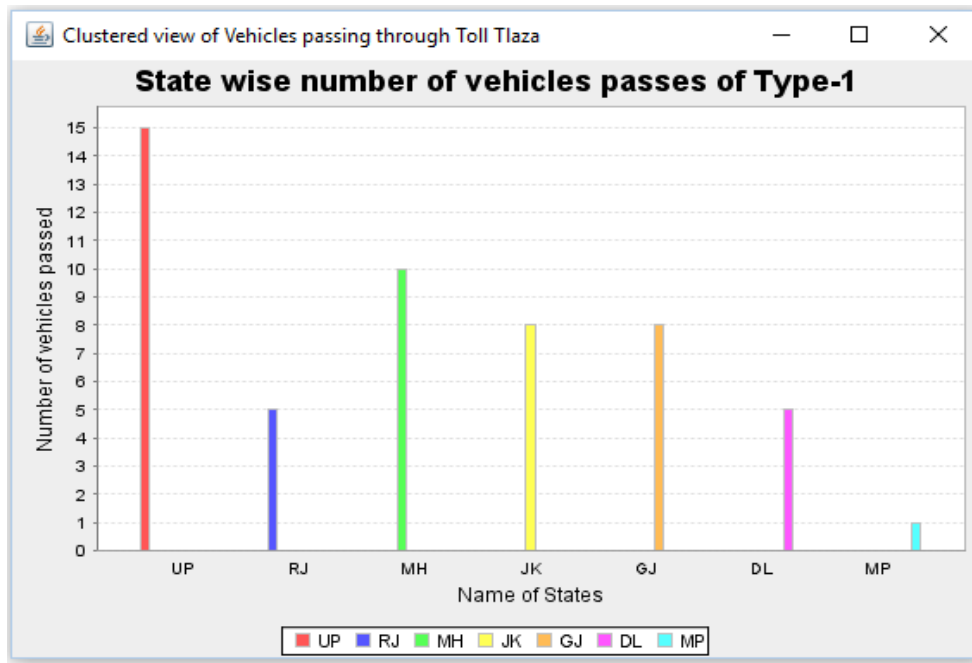


Figure 8.7 Screenshot of Type-1 vehicles passed

8.3.1.2 Show Graph of Type 2 Vehicles

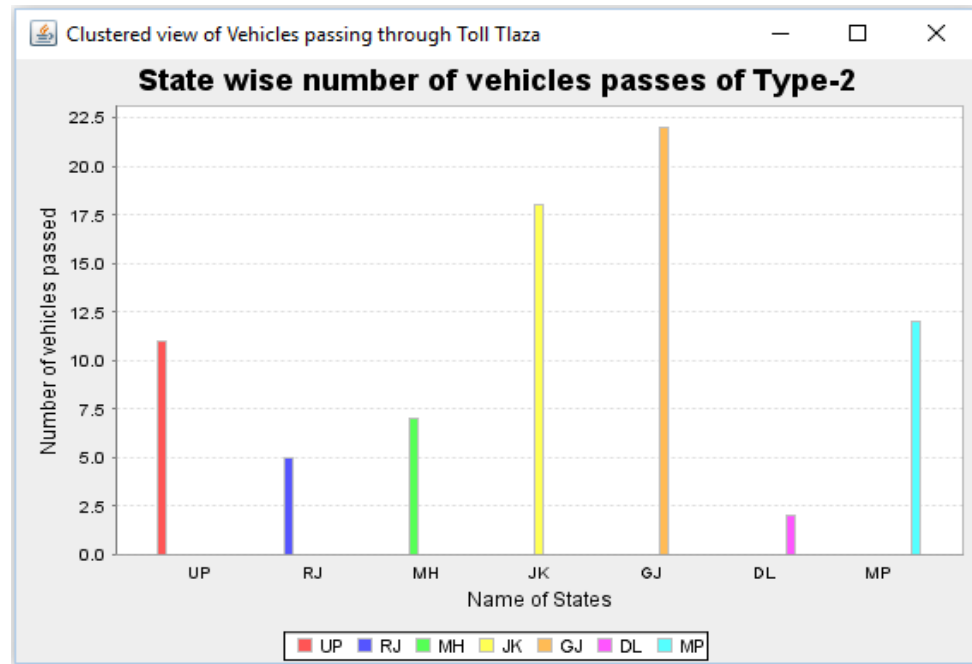


Figure 8.8 Screenshot of Type-2 vehicles passed

8.3.2 Revenue Generated

8.3.2.1 Revenue Earned by Type-1 Vehicles

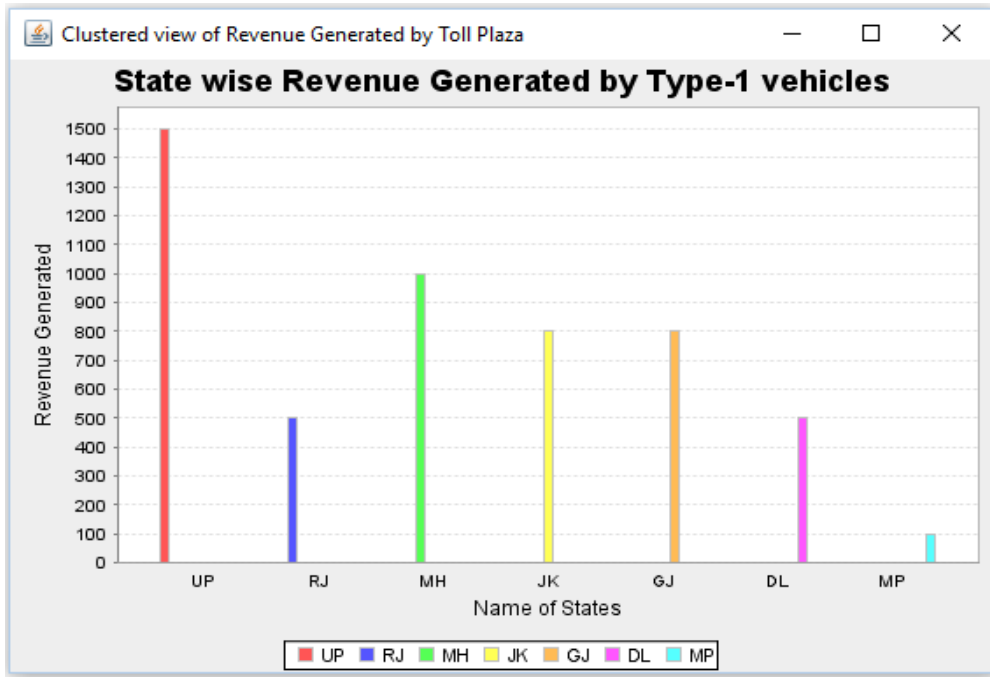


Figure 8.9 Screenshot of revenue generated by Type-1 vehicles

8.3.2.2 Revenue Earned by Type-2 Vehicles

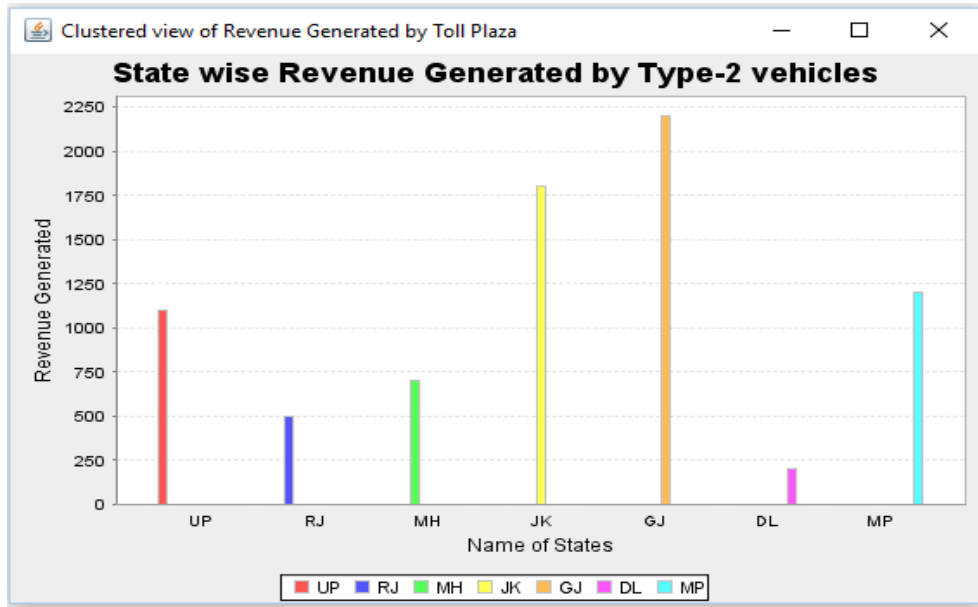


Figure 8.10 Screenshot of revenue generated by Type-2 vehicles

8.3.3 Total

8.3.3.1 Total Number of Passed Vehicles

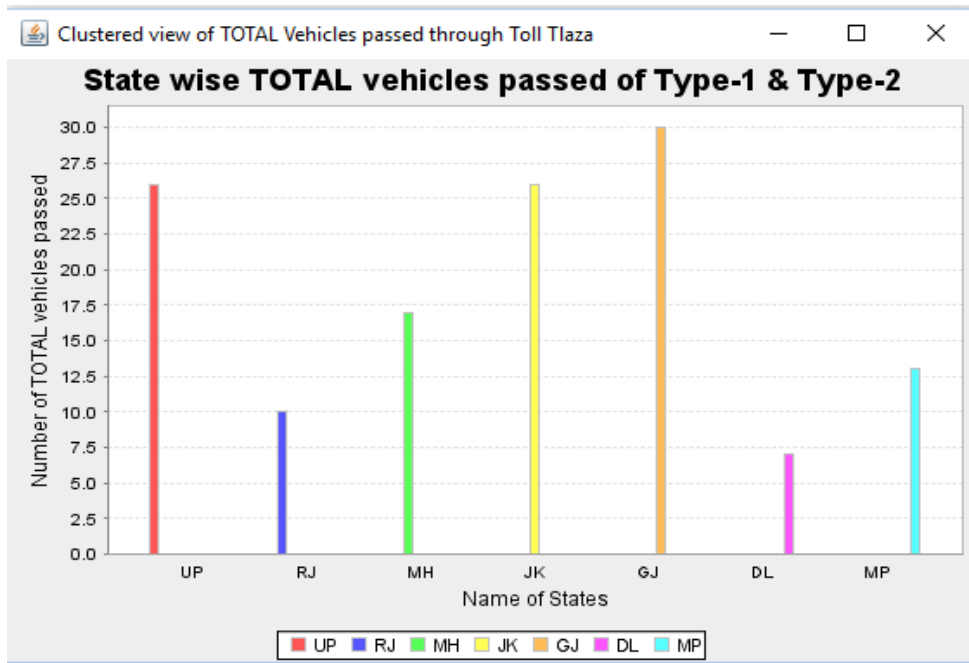


Figure 8.11 Screenshot of Total Number of Passed Vehicles State wise

8.3.3.2 Total Revenue Generated

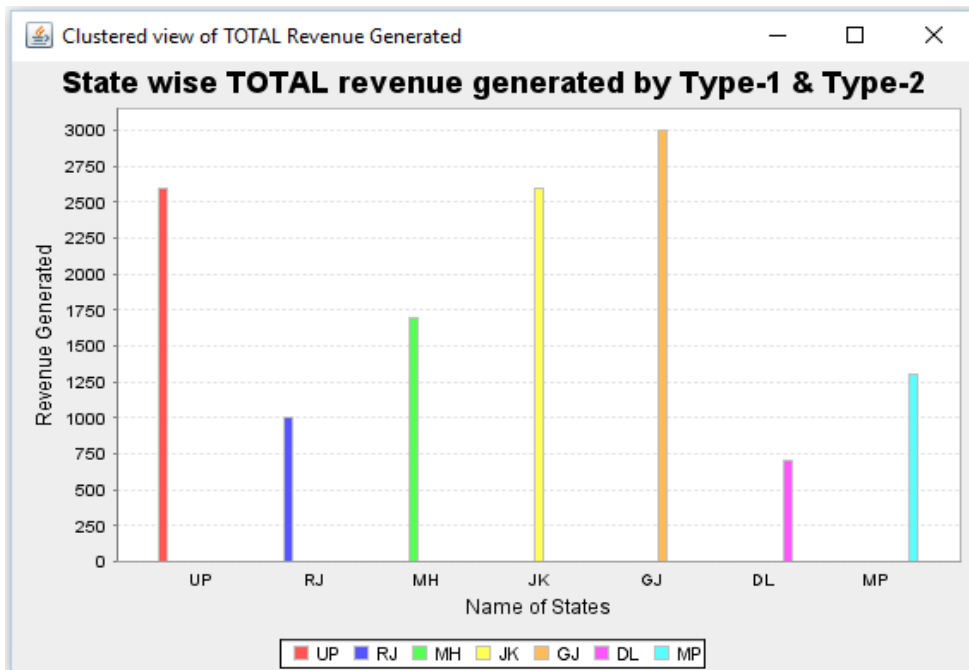
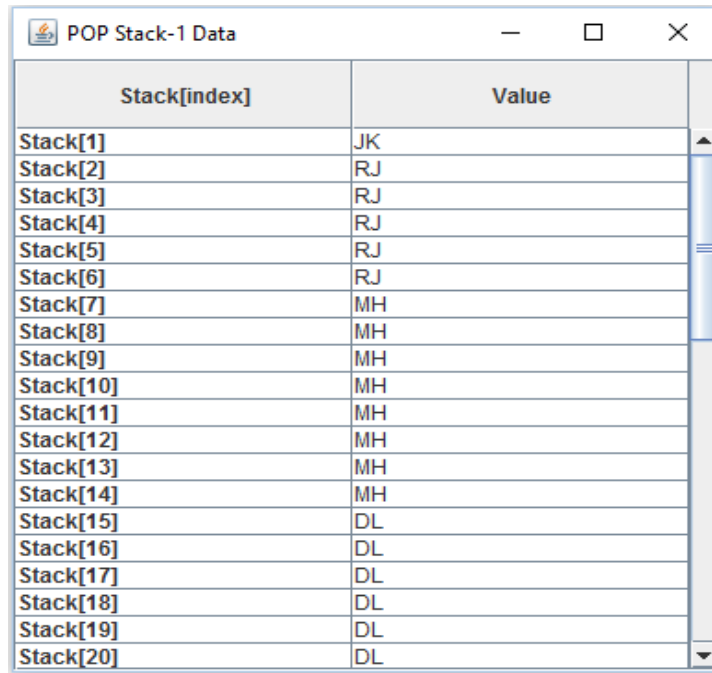


Figure 8.12 Screenshot of Total Generated Revenue State wise

8.3.4 Stack

8.3.4.1 POP Type-1 Stack

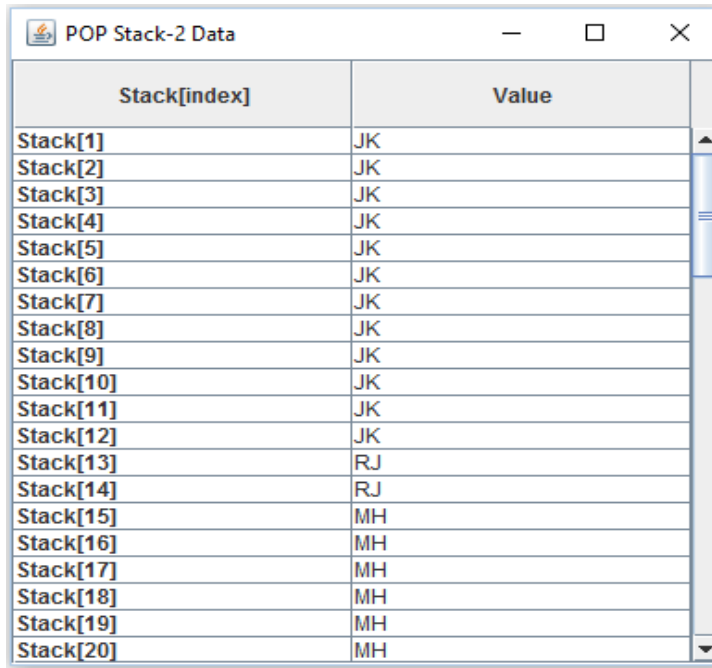


The screenshot shows a window titled "POP Stack-1 Data" containing a table with two columns: "Stack[index]" and "Value". The table lists 20 stack entries from index 1 to 20. The values are: JK, RJ, RJ, RJ, RJ, RJ, MH, MH, MH, MH, MH, MH, MH, MH, DL, DL, DL, DL, DL.

Stack[index]	Value
Stack[1]	JK
Stack[2]	RJ
Stack[3]	RJ
Stack[4]	RJ
Stack[5]	RJ
Stack[6]	RJ
Stack[7]	MH
Stack[8]	MH
Stack[9]	MH
Stack[10]	MH
Stack[11]	MH
Stack[12]	MH
Stack[13]	MH
Stack[14]	MH
Stack[15]	DL
Stack[16]	DL
Stack[17]	DL
Stack[18]	DL
Stack[19]	DL
Stack[20]	DL

Figure 8.13 Screenshot of Stack-1 data items

8.3.4.2 POP Type-2 Stack



The screenshot shows a window titled "POP Stack-2 Data" containing a table with two columns: "Stack[index]" and "Value". The table lists 20 stack entries from index 1 to 20. The values are: JK, JK, JK, JK, JK, JK, JK, JK, JK, JK, JK, JK, RJ, RJ, MH, MH, MH, MH, MH.

Stack[index]	Value
Stack[1]	JK
Stack[2]	JK
Stack[3]	JK
Stack[4]	JK
Stack[5]	JK
Stack[6]	JK
Stack[7]	JK
Stack[8]	JK
Stack[9]	JK
Stack[10]	JK
Stack[11]	JK
Stack[12]	JK
Stack[13]	RJ
Stack[14]	RJ
Stack[15]	MH
Stack[16]	MH
Stack[17]	MH
Stack[18]	MH
Stack[19]	MH
Stack[20]	MH

Figure 8.14 Screenshot of Stack-2 data items

8.3.5 About

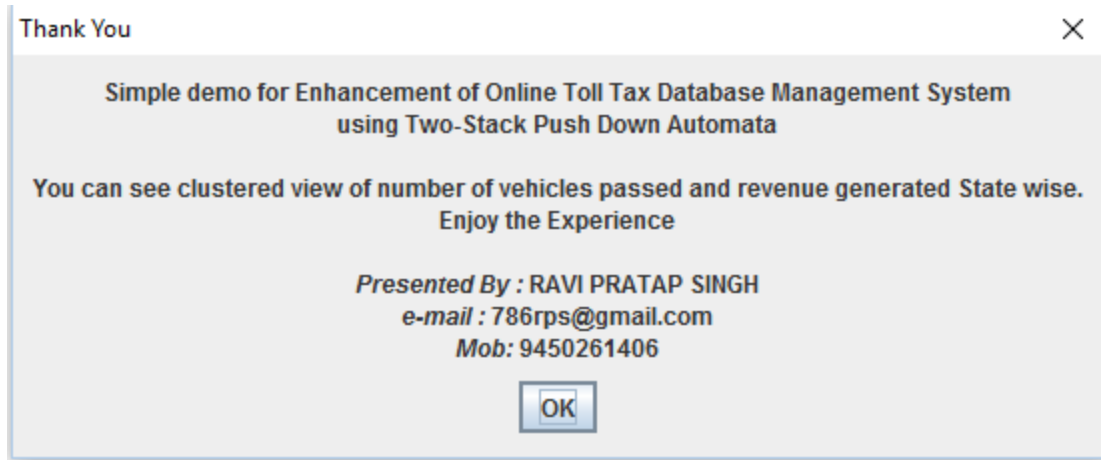


Figure 8.15 Screenshot of Help Menu Item

8.4 INTEGRATED DEVELOPMENT ENVIRONMENT

An Integrated Development Environment (IDE) is an application that facilitates application development. In general, an IDE is a graphical user interface (GUI)-based workbench designed to aid a developer in building software applications with an integrated environment combined with all the required tools at hand. An IDE is a programming environment that has been packaged as an application program, typically consisting of a code editor, a compiler, a debugger, and a GUI builder. Most common features, such as debugging, version control and data structure browsing, help a developer quickly execute actions without switching to other applications. Thus, it helps maximize productivity by providing similar user interfaces (UI) for related components and reduces the time taken to learn the language. An IDE supports single or multiple languages.

The IDE may be a standalone application or may be included as part of one or more existing and compatible applications. IDEs provide a user-friendly framework for many modern programming languages, such as Visual Basic, Java, and PowerBuilder. An IDE is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Some IDEs contain a compiler, interpreter, or both, such as NetBeans and Eclipse; others do not, such as SharpDevelop and Lazarus.

We are using NetBeans IDE 8.1 for implementing our thesis, which is the current version of NetBeans IDE and was released on November 4, 2015. NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform, including the NetBeans integrated development environment (IDE), can be extended by third party developers. The NetBeans IDE is primarily intended for development in Java, but also supports other languages, in particular PHP, C/C++ and HTML5. NetBeans is cross-platform and runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM. In 1997, Roman Stanek formed a company around the project and produced commercial versions of the NetBeans IDE until it was bought by Sun Microsystems in 1999. Sun open-sourced the NetBeans IDE in June of the following year. Since then, the NetBeans community has continued to grow. In 2010, Sun (and thus NetBeans) was acquired by Oracle. A snapshot of the NetBeans IDE is as below:

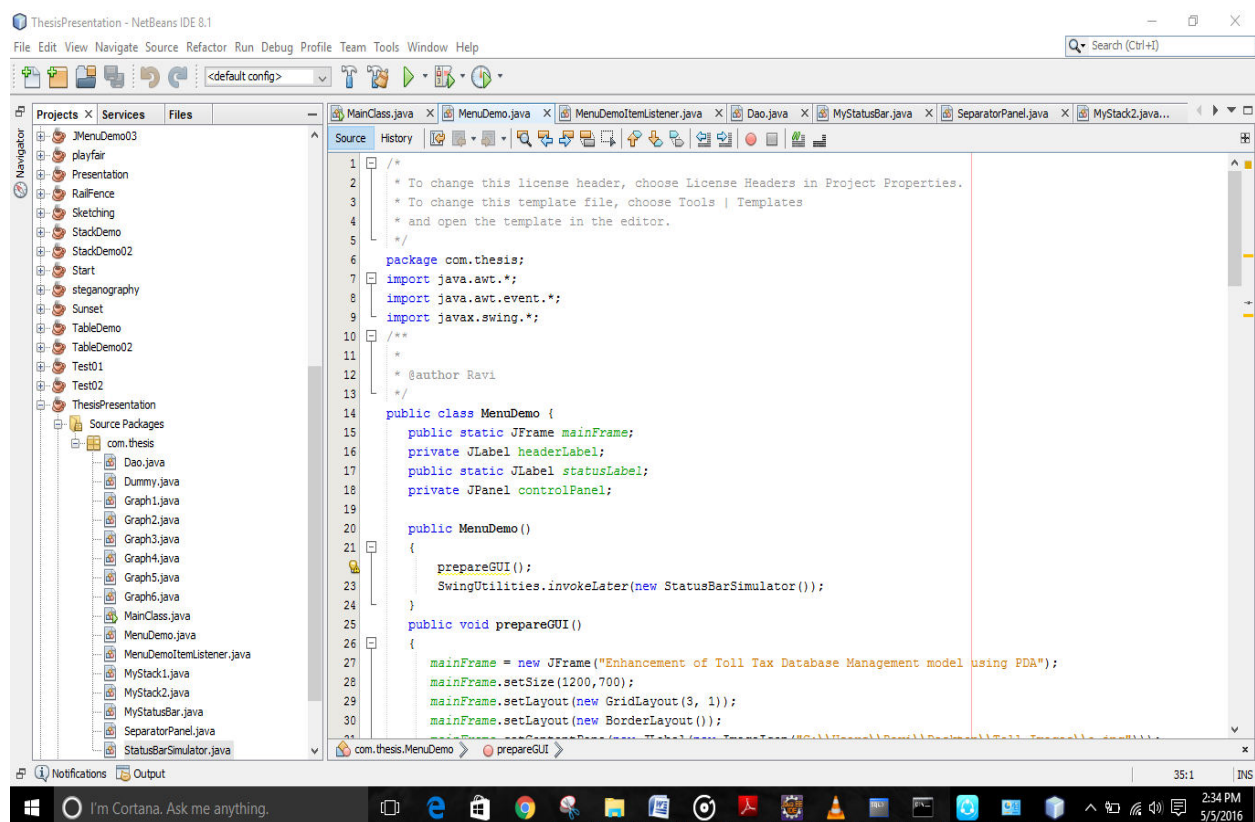


Figure 8.16 Screenshot of NetBeans IDE

CHAPTER 9

CONCLUSION AND FUTURE WORK

This research work focuses on development of a feature to view vehicles coming from which territory in cluster view and show this into clustered view of vehicles log. This clustered view will help in comparative analysis. It is low in cost and easy to implement that will support into financial leakage control. In future, this kind of interactive framework can also be maintained for Airlines Database Management model for domestic and international flights as well. Further, this model can be enhanced on the Ship Database Management for maintaining the log of different ships for the purpose of comparative analysis.

Online Toll Tax Database Management is a rapidly changing and vast research area and has many open questions and challenges. Designing a Toll Plaza system involves choosing particular feature representation techniques, optimal dimensionality and reliable similarity functions in order to achieve best results. The ultimate aim is to reduce the gap between semantic information exchange between servers and making the Toll Plaza functionality faster. The real time constraint notation is used for representing and modeling the constraints in real time modeling. When these real time specifications will be integrated with in the object-oriented tapestry then that provide sufficient real time specifications as one may expect from real time language. So, as the future aspect of the proposed PDA model the RTCN being applied to the PDA for formal verification of the proposed model.

The developed Online Toll Tax Database Management System using the PDA approach can be extended to include stronger features and additional learning capabilities. This will provide higher accuracy values thus facilitating the investigation of results. More database objects and constraints can also be used to increase confidence in the results obtained and improve the security threats. Investigations of the experimental results are further required for additional insights into sample size issues. Moreover, this report is a starting point for enhancing a possible set of features which could be included for appropriate and consistent performance evaluation. This is crucial to enable comparison analysis of revenue obtained from various states on similar grounds.

REFERENCES

- [1] Vivek Kr. Singh, S. P. Tripathi, J. B. Singh and R. P. Agarwal, “*Enhancement of User’s Call Logging facilities using Push Down Automata with Real Time Constraint Notation*”, *International Journal of Computer Science Issues(IJCSI)*, vol. 9, issue 3, 2012, pp. 216-220 .
- [2] Mr.Abhijeet Suryatali and Mr. V. B. Dharmadhikari, “*Computer Vision Based Vehicle Detection for Toll Collection System Using Embedded Linux*”, *International Conference on Circuit, Power and Computing Technologies [ICCPCT], IEEE*, 2015, pp. 978-984.
- [3] Padmavathi Shyadambi, Saikumari S, Jyoti Chitti, Deepika Paragoudar, Raghavendra Havin and Praveenkumar Hadapad, “*Toll Snapping And Processing System*”, *International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)*, vol. 14, issue 2, 2015, pp. 63-66.
- [4] Abel Shajan, Abin Jose and Kalung Tari, “*Automatic Toll Collection System*”, *International Journal of Emerging Trends in Science and Technology (IJETST)*, vol. 02, issue 04, 2015, pp. 2242-2247.
- [5] Devadasu Saraswathi And P. Suresh Kumar, “*Electronic Toll Collection System and GSM Trace for Smart Vehicles*”, *International Journal of Scientific Engineering and Technology Research*, vol. 04, issue 28, 2015, pp.5419-5424.
- [6] P. Siva Kumar and R. S. Pratap Singh, “*Implementing ALPR System using Vehicle LP Recognition Information*”, *International Journal of Scientific Engineering and Technology Research*, vol. 04, issue 15, 2015, pp. 2744-2746.
- [7] Hong Yang, Kaan Ozbay, Bekir Bartin and Ozgur Ozturk, “*Assessing the Safety Effects of Removing Highway Mainline Barrier Toll Plazas*”, *Journal of Transportation Engineering*, 2014, pp. 943-950.
- [8] Marco Amorim, Antonio Lobo, Carlos Rodrigues and Antonio Couto, “*Optimal location of electronic toll gantries: The case of a Portuguese freeway,*” *ScienceDirect*, 2014, pp. 880-889.
- [9] Nale Zhao, Tongyan Qi, Lei Yu, Juwen Zhang and Pengpeng Jiang, “*A Practical Method for Estimating Traffic Flow Characteristic Parameters of Tolloed Expressway Using Toll Data*”, *ScienceDirect, The 9th International Conference on Traffic & Transportation Studies (ICTTS’2014)*, 2014, pp. 632-640.
- [10] Joao Dias, Joao Nuno Matos and Arnaldo S. R. Oliveira, “*The Charge Collector System*”, *ScienceDirect, Conference on Electronics, Telecommunications and Computers – CETC*, 2014, pp. 130-137.
- [11] Saurabh Vats, Gaurav Vats, Rahul Vaish and Varun Kumar, “*Selection of optimal electronic toll collection system for India: A subjective-fuzzy decision making approach*”, *Elsevier, Applied Soft Computing 21*, 2014, pp. 444-452.

- [12] Yogesh Kamble, Ajinkya Abhyankar, Tanmay Pradhan and Aditya Thorat, "Check post and Toll Tax Collection using RFID", *International Journal of Innovative Science (IJISSET), Engineering & Technology*, vol. 1, issue 2, 2014, pp. 47-51.
- [13] Anish Dhurat, Parag Magal, Manish Chheda and Darshan Ingle, "Gateless Electronic Toll Collection using RFID", *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 16, issue 2, ver. VI, 2014, pp. 73-80.
- [14] S.Nandhini and P. Premkumar, "Automatic Toll Gate System Using Advanced RFID and GSM Technology", *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 3, issue 11, 2014, pp. 13002-13007.
- [15] Priyanka Chhoriya, Govinda Paliwal and Poonam Badhan, "Image Processing Based Automatic Toll Booth in Indian Conditions", *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, issue 4, April 2013, pp. 2250-2459.
- [16] R. M. Hushangabade and S.V. Dhopte, "Dynamic Approach towards Toll Tax Collection and Vehicle Tracking with the Help of RFID", *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 3, issue 1, 2013, pp. 368-371.
- [17] Navin Kumar Chaudhary, Rabin Karmacharya, Binaya Ghimire, Dr.N. Srinivasu, "Stack Variation In Push Down Automata", *International Journal of Engineering Trends and Technology (IJETT)*, vol. 04, issue 5, 2013, pp. 1535-1539.
- [18] Bihari T., Gopinath P. and Schwan K., "Object-Oriented Design of Real Time Software", *IEEE Software Trans.*, pp. 194-201.
- [19] Dr. S. P. Tripathi, Prof. J. B. Singh, Vivek Kr. Singh, "Design Mobile Data Logging Framework using Ubiquitous Computing Environment", *International Journal of Hybrid Information Technology*, vol. 2, issue 4, October 2009.
- [20] Pereira C., "Putting OO work: Results from Applying the Object-Oriented Paradigm during the Development of Real Time Applications", *Fifth Euromicro Workshop on Real-time Systems Proceedings*, pp. 166-170.
- [21] Md. Farhad Ismail and M.A.R. Sarkar, "Development of a Model for Electronic Toll-Collection System", *I.J. Intelligent Systems and Applications*, vol. 01, 2012, pp. 39-45.
- [22] Kligerman E. and Stoyenko A., "Real Time Euclid: a Language for Reliable Real Time Systems", *IEEE Trans. on Software Engineering*, vol. SE-12, No. 9.
- [23] Manuj Darbari, Rishi Asthana and Vivek Kr. Singh, "Integrating Fuzzy MDE-AT Framework for Urban Traffic Simulation", *International Journal of Software Engineering (IJSE)*, vol. 1, issue 2, pp. 24-31.

[24] Anneke G. Kleppe and Jos Warmer, “*The Object Constraint Language: Precise Modeling with UML (Addison-Wesley Object Technology Series)*”.

[25] Bruce K. “*A Pattern Language for Object RDBMS Integration, Knowledge System Group*”.

[26] Dr. Khali Persad, Dr. C. Michael Walton and Shahriyar Hussain, “*Toll Collection Technology and Best Practices*”, Project 0-5217: Vehicle/License Plate Identification for Toll Collection Applications, 2007.

[27] <http://www.indiantollways.com/category/toll-management-system/>

PLAGIARISM REPORT

(Checked By SmallSEOTool)

Page Number	Unique Content %
iii	97
1-2	92
3-4	93
5-6	91
7-8	90
9-10	86
11-12	89
13-14	91
15-16	90
17-18	94
19-20	92
21-22	93
23-24	91
25-26	89
27-28	87
29-30	90
31-32	86
33-34	91
35-36	92
37-38	98
39-40	87
41-42	90
43-44	94
45-46	92

47-48	94
49-50	86
51-52	94
53-54	95
55-56	98
57-58	90
59-60	85
61-62	89
63-64	87
65-66	88
67-68	88
69-70	86
71-72	89
63-74	90
75-76	93
77-78	91
79	86
80-84	Skipped
85-86	88
87	100

Overall % unique content = 88% i.e.

12% Plagiarism

PUBLICATIONS

[1] Ravi Pratap Singh and Dr. V. K. Singh, “*Online Toll Tax Database Management Model through Push-Down Automata*”, *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)*, vol. 4, issue 2, 2016, pp. 1323-1333.

Ravi Pratap Singh

Contact : 9450261406

E-Mail : 786rps@gmail.com

I would like to utilize my experience and energetic attitude in teaching student with great enthusiasm. I am looking for a challenging position as Lecturer to utilize my technical skills for the growth of the Institution and students.

SYNOPSIS

- ⇒ Pursuing M. Tech (Software Engineering) from BBD University, Lucknow (till 1st year 80%)
- ⇒ PG DAC from C-DAC, Pune.
- ⇒ B. Tech (Information Technology).
- ⇒ Qualifying the “**Sun Certified Programmer for the Java Platform (SE 5.0)**” examination and got the SCJP Certificate.
- ⇒ Topper in C-DAC during DAC course on **J2EE** module and got the certificate.
- ⇒ An effective communicator with excellent relationship, management skills and strong analytical, problem solving and organizational abilities.

PROJECT EXPERIENCES

M. Tech THESIS:

Topic : Enhancement of Online Toll-Tax Database Management model using PDA.

Description : In this, we highlights the use of 2-Stack Push-Down Automata in maintaining the vehicle records and provide the clustered view of them to make comparative analysis easier and faster. We focus to provide a communicative framework that can record the vehicles coming from a particular state.

B. Tech PROJECT:

Name : Secure Data Transfer.

Tools Used : Java and Oracle 10g.

DAC PROJECT:

Name : Airlines Reservation System.

Tools Used : Microsoft .Net 4.0, ASP.Net, C#.Net, MS SQL Server 2008.

COMPUTER SKILLS

- **Languages** : C, C++, Java SE, Java EE, JSP, C#.Net, ASP.Net
- **DBMS** : PL/SQL, Oracle, SQL Server, MySql.

SUMMER TRAINING

- Name of Company : HP(Hewlett-Packard), IIIT-Allahabad
- Technology : J2EE
- Duration : 10th June 2010 to 25th July 2010.

ACADEMIA

- **PG DAC**(Diploma in Advanced Computing) from SunBeam Institute of Information Technology, Pune in 2012.
- **B.Tech** (Information Technology) from UCER, Allahabad, with 67.84% in 2011.
- **H.S.C.** (U.P. Board) from SVN Inter College with 79.20% in 2005.
- **S.S.C.** (U.P. Board) from SVN Inter College with 74.33% in 2003.

AREA OF INTEREST

- Database Management Systems
- Object Oriented Programming Languages
- Operating Systems
- Automata
- Data Structure

PG DAC (Diploma in Advanced Computing)

It is the flagship programme of **ACTS CDAC** during which I polished my skills in:

- C, C++, OS Concepts, SE, Web Programming, DBT, Core Java, J2EE, MS .Net.

ACHIEVEMENTS & ACTIVITIES

- Participated in the Science Festival on “National Science Day” organized under the auspices of CST U.P. Govt. and got the certificate.
- Winner of Inter Group Aptitude Competition and Inter Group HR Interview Competition during PG DAC.
- Participated in the KABADDI game as the caption of team and got the certificate of excellence.
- Participated in a quiz contest in school and got the certificate (within top 5).

PERSONAL DETAILS

Date of Birth : Oct 02, 1989
Address : 10 C/A Nai Bazar, Naini - Allahabad
U. P. (211008)

DECLARATION

I hereby declare that all the information mentioned above is true and correct to best of my knowledge and belief.

Date :

Place : Lucknow

Ravi Pratap Singh

BABU BANARASI DAS UNIVERSITY, LUCKNOW

CERTIFICATE OF THESIS SUBMISSION FOR EVALUATION

(Submit in Duplicate)

1. Name:

2. Enrollment No. :

3. Thesis title:

.....

.....

4. Degree for which the thesis is submitted:

5. Faculty of the University to which the thesis is submitted

.....

6. Thesis Preparation Guide was referred to for preparing the thesis. YES NO

7. Specifications regarding thesis format have been closely followed. YES NO

8. The contents of the thesis have been organized based on the guidelines. YES NO

9. The thesis has been prepared without resorting to plagiarism. YES NO

10. All sources used have been cited appropriately. YES NO

11. The thesis has not been submitted elsewhere for a degree. YES NO

12. Submitted 2 spiral bound copies plus one CD. YES NO

(Signature of the Candidate)

Name:

Roll No

Enrollment No:

BABU BANARASI DAS UNIVERSITY, LUCKNOW

CERTIFICATE OF FINAL THESIS SUBMISSION

(To be Submit in Duplicate)

1. Name:

2. Enrollment No. :

3. Thesis title:

.....

.....

4. Degree for which the thesis is submitted:

5. School (of the University to which the thesis is submitted)

.....

6. Thesis Preparation Guide was referred to for preparing the thesis. YES NO

7. Specifications regarding thesis format have been closely followed. YES NO

8. The contents of the thesis have been organized based on the guidelines. YES NO

9. The thesis has been prepared without resorting to plagiarism. YES NO

10. All sources used have been cited appropriately. YES NO

11. The thesis has not been submitted elsewhere for a degree. YES NO

12. All the corrections have been incorporated YES NO

13. Submitted 4 hard bound copies plus one CD. YES NO

(Signature of the Supervisor)

Name:

(Signature of the Candidate)

Name:

Roll No

Enrollment No: