Minimizing EDA Cost Without Impacting R&D Effectiveness

A Thesis

submitted in partial fulfillment of the requirements for the award of the degree of

Master of Engineering

in

Department of Computer Science

by

Niyati Trivedi
(Reg no: 801532037)

Thapar University
Patiala-147004, Punjab, India
June 2017

Candidate's Declaration

I hereby certify that the work, which is being presented in the thesis, titled **Minimizing EDA Cost Without Impacting R&D Effectiveness**, in partial fulfillment of the requirements for the award of the degree of **Master of Engineering** and submitted to Thapar University is an authentic record of my own work carried out under the supervision of **Dr. R. K. Sharma**. I have also cited the source(s) of text(s)/figure(s)/table(s).

The matter presented in this thesis has not been submitted elsewhere for the award of any other degree or diploma from any institution.

Niyati

Niyati Trivedi

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

RK Sharma

(**Dr. R. K. Sharma**)
Professor
CSED

Abstract

Present logic synthesis, streamlining calculations and optimization algorithms in Electronic Design Automation (EDA) are heuristic. They don't ensure ideal arrangements, notwithstanding they are supposed to deliver arrangements of adequate quality. The EDA industry offers a wide variety of software licensing models. EDA software tool for integrated circuit (IC) design are most of the time the second biggest financial budget cost for fabless IC design organizations (after compensations). Fabless assembling is the outline and offer of equipment gadgets and semiconductor chips while outsourcing the creation or "fab" of the gadgets to a specific producer called a semiconductor foundry. As a result, numerous organizations have concentrated on dynamic administration of this financial plan to secure a focused advantage. The steadily expanding many-sided quality of IC plans has been empowered by an always growing scope of progressively effective programming tools.

In the world of networking everything is connected and idea of connectivity is to exchange the information in such scenario licensing takes the best advantage when it is shared and used to the best utilization, in the process of optimization of licenses there are many criterias are there to look into and make the best use of the same. Licensing mainly involves giving an authority to perform something productive. For license management, we are having a one in all dashboard which shows hourly based license peak usage for various EDA tools and also indicates the license utilization, tracks Costcenter wise usage, manages limits, captures queues and all. Based on the utilization we can make out whether licenses are under-utilized or is there in any shortage of the same.

This venture is the product bundle to monitor summarized or average peak license usage and a rundown table of this observed information is used to evaluate required permit pools for remixes and for renegotiating EDA rental understandings. This project is made at "Infineon Technologies India Pvt. Ltd.", Bengaluru which was set up in 1997 as Siemens Semiconductors. Today it is a noteworthy improvement focus of Infineons R&D organize, and assumes a key part in programming advancement and equipment outline.

Keywords: EDA, fabless IC, costcenter, Software Package, licenses, Infineon Technologies India Pvt. Ltd.

Acknowledgements

Foremost, I would like to express my deep gratitude to my supervisor **Prof. Dr. R.K. Sharma** for his unfailing support and belief in me at every step of my Masters program. Without his invaluable advice and encouragement, this thesis would not have been possible. His contribution to this thesis goes well beyond his role as an academic supervisor and includes constant support on a personal level without which this journey may never have been completed. And for this, I am truly grateful.

I have been extremely lucky to get the internship offer from one of the top semiconductor companies, Infineon Technologies Pvt. Ltd. I am truly thankful to my manager **Ahsan Jafri**, my mentor **Shantagouda Goudar** and the team **Global License Management**, for their constant support and motivation throughout my internship programme.

I am also thankful to **Dr. Ashutosh Mishra**, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work. He has always been supportive and provided us required information on regular intervals.

I would like to acknowledge **Dr. Maninder Singh**, Head of Computer Science Department for setting good standards for his students and providing all the help and facilities that were essential throughout the journey. Your encouragement time and again has helped students to achieve the set goals.

Above all, deepest thanks to the Almighty, my family and friends for always being there for me and staying calm at times required.

Niyati

Niyati Trivedi

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CPAN	Comprehensive Perl Archive Network
ECAD	Electronic Computer-Aided Design
EDA	Electronic Design Automation
FIFO	First In First Out
ISV	Independent Software Vendor
lmstsat	License Management Statistics
LSF	Load Sharing Facility
LV	License View
NNU	Named Users or Clients
RTE	Real-Time Enterprises
SDLC	Software Design Life Cycle
WL	Waiting Lounge
XGBoost	eXtreme Gradient Boosting

Chapter 1

Introduction

Electronic Design Automation (EDA), sometimes referred as Electronic Computer-Aided Design (ECAD), is a process for designing electronic systems. EDA is a key technology enabler of Real-Time Enterprises (RTE). It complements existing enterprise technologies by providing the infrastructure needed to support the event-driven solutions that analyze complex relationships between business events to identify business opportunities, threats, and anomalies. Business users need timely information about significant business activities to improve the quality of the decisions they make and to close the insight-to-action gap. Why EDA software licensing and administration of EDA software licenses is an important point in semiconductor industry by any means ? The reason is that present advancements for EDA programming permitting and administration of EDA software licenses have been composed under the suspicion that the license server (purpose behind the approval of the use of a license secured application) and the EDA tools are situated in the same regulatory and system area, and EDA budget causes the second highest budget after employee salaries and also there is a need for effective management of the EDA tool licenses. In this manner, these licenses are given on the premise of named clients or customers (NNU), hostnames (IP-addresses), or consistently as a land site permit for the administrative domain of the firm. In the event that we need to utilize this EDA tool programming in a conveyed benefit situated setup, utilizing resources that are spread crosswise over different administrative domains, the network license server setup can be seen in figure 1.1.

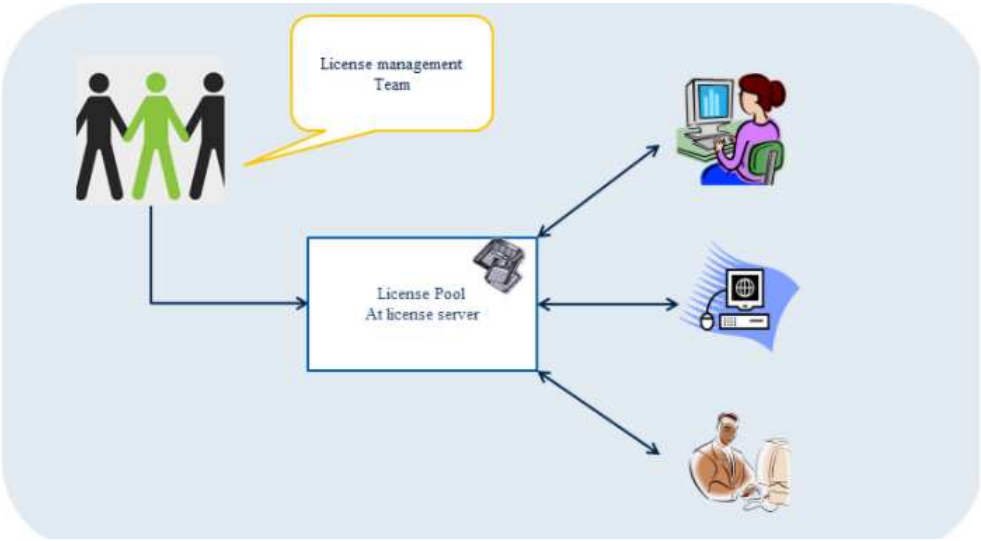

Figure 1.1: License Server setup

1.1 Background and Motivation

1.1.1 Early Days

Up until the point that the 1980s, there were hardly any licensing techniques, despite basically offering a software program and trusting that it would not be replicated. Authorizing was not an issue for some early applications, in light of the fact that the software was transported in a difficult to duplicate format, for example, restrictive amusement tapes or on sound tapes, which were hard to copy in the days prior to mass market tape recorders. License management has two aspects:

i Dealing with the licenses, e.g. distributing them in the organization, purchasing increasingly if necessary, returning unneeded licenses, and so forth.

ii Keeping the unlicensed utilization of software or charging the client for such utilization.

The principal angle is considerably more mind boggling than it shows up at first look; one can see that by concentrating on the present framework from, say, Macrovision. One can interpret the second angle as sealing the license administration framework. Early software sealing techniques were extremely powerless.

Earlier, all the license management tasks like license remixes, maintaining the real-time usage track of different softwares provided by different vendors, servicing queued license requests in a fast and better way, were manually handled. Some automated and centralized system is required which can help a license manager in doing effective license management. Below are some points which plays an important role in managing software permits.

1.1.1.1 Technology Drivers for EDA

At whatever point a business in the Electronic Design Automation (EDA) industry contributes a lot of time, cash or push to build up some type of competitive software or equipment advantage for the EDA commercial center, it is imperative to ensure such exclusive premium lawfully. The technology drivers for EDA are listed below.

i Loosely Coupled Architecture,

ii Ease of Integration,

iii High volume, low latency processing,

iv Business Process and Service interaction,

v Partner integration,

vi Real-time monitoring,

vii Pattern matching and analysis.

1.1.1.2 FlexNet License Setup

A FlexNet license grants access to software product for multiple users and multiple computers. These are available only for some Keil products. FlexNet License Server is compatible with v16.8 (and later) client applications and it runs on the following OS:

i Windows 32 / 64 bit,

ii Linux Standard Base 32 / 64 bit,

iii Mac OS X.

A FlexNet license overrides all other licensing methods for a product group. For example, if you have installed a Single-User MDK-ARM Professional License and you install a FlexNet MDK-ARM Standard License, then the Single-User MDK-ARM Professional License is disabled and the FlexNet MDK-ARM Standard License enabled.

1.1.1.3 Licenses and License Management Software

For the licenses acquired from the ISV by the end-client's home association, it is valuable to help new plans of action as e.g. pay per use. This would require an adjustment of the independent software vendor licenses, which now are purchased at a changed esteem autonomous from the successful utilization and with yearly help costs. Likewise, as said over a Cloud supplier could buy the license from an ISV and host the licenses in a license server inside the Cloud.

The Cloud provider could then offer access to the license secured software on a compensation for each use base to the Cloud customers. This model would not oblige changes of the ISV licenses. **There are four major actors involved:**

i The independent software vendor (ISV), which created and possesses the application, pitches the privilege to utilize the application to its clients and gives a license that rules the terms of use.

ii The customer who will use the application to fulfill an objective.

iii The customer's home affiliation that buys the benefit of utilizing the application from the ISV and receives a license for authorization of the application use oversaw by a license server.

iv The specialist co-op, which is contracted by the clients home association to give an administration (the application) in an outer framework condition.

1.1.2 License Administration Tools

1.1.2.1 Waiting Lounge

Waiting Lounge is one of the applications of this software package for administrators and managers, and also for end users. This is developed mainly for the purpose of tracking

queues and in the final step minimizing the waiting times for licenses for the end users. In this, we are processing a huge license statistics text file and capturing all the information for the checked-out licenses, like, users id, host from which the license is being checked out, job name, process id, start time of the queue and all such information. Sometimes when the license limit is reached, i.e., all the licenses from the available pool of licenses are in use, then some licenses are queued. We are also capturing the queued licenses information. Through which queued license users for different tools are tracked on an interval of 2 minutes. This application was initially started with two main vendors - **Vendor1** and **Vendor2** and shows average, minimum and maximum waiting time for individual queued users for a particular tool. And now third major vendor is added to the application - **Vendor3**.

1.1.2.2 LiSA

LiSA is an internal tool which targets:

i Analysing the license usage for more efficiency and license - cost reduction

ii Infineon wide standard for license server setup

iii Global - Web Interface and Database located in Villach

iv Online license remix for better ordering workflow

The duration of license checkouts for jobs submitted by different users and other details related to jobs are stored in LiSA, which was introduced quite ago by the Infineon employees. We are using combined data from LiSA and Heatmap (developed by us) for our analytics part in predicting wait time for queued license requests.

1.1.3 Research Motivation

The Motivation behind the License Usage Monitoring System and Wait-Time Prediction is to have sharp perception on the license usage and to license ensured applications on the grounds that the licenses more often than not are bound to hardware within the area of the client and do not permit access from outside, e.g. because of firewalls, thus, authorizing neighborhood utilization of the protected applications as it were. Preceding examining the prerequisites we introduce and talk about the diverse on-screen characters in licensing.

EDA licensing models are of two fundamental types: perpetual or time-limited. A perpetual license, the more conventional model, is an EDA tool license with a term that is long to the point that it is successfully the same as an altogether buy. The length of an interminable license is regularly 99 years, which is any longer than the sensible lifespan of any EDA tool. Income is received all in advance, when the permit is paid for. Upkeep which incorporates specialized bolster, bug fixes, and item upgrades - may be obtained independently at a yearly value that is normally 10-15 percent of the ceaseless

license cost. A time based license is basically a lease of software for a shorter time of time (regularly 2-3 years) that for the most part includes support.

The time-based license is a later development that is for the most part credited to Gerry Hsu, CEO of Avant! Partnership His inspiration was to make a progressing stream of repeating income to backing the high cost of new item advancement - a cost that was not being met by the one-time offers of unending licenses through the generally humble income stream of yearly upkeep contracts.

1.2 Organization of Thesis

This thesis is structured into total 7 chapters. A brief review of each chapter is provided below:

- **Chapter 1:** This chapter discusses about basic introduction to Electronic Design Automation (EDA), Background of the area, FlexNet License Setup and motivation for doing this project. This section also provides a brief summary of all the chapters and acts as a guide for the reader as he/she can know briefly about each chapter and directly move to the chapter which is of interest.
- **Chapter 2:** This chapter describes the literature review of the problem. In this, how to use a License Server with License Files and Overview of Trusted Storage Flexlm are discussed along with the papers I have read before starting work on this project.
- **Chapter 3:** This chapter focuses on the description of problem that is being tackled by the Thesis. It includes the Problem Statement, Research direction and the key research area to be focused. This chapter talks about the objectives formulated to carry out this research in a systematic way.
- **Chapter 4:** This chapter describes the system design and implementation, discussing the overall design of the Heatmap and Waiting Lounge, architectures, flow charts and implementation methods.
- **Chapter 5:** This chapter describes the testing and results for GUI Interface for Heatmap tool, analysed results for Method I - Queuing Theory and Method II-Average Base for waiting lounge and wait-times prediction, additionally; all the key results along with required graphs are discussed.
- **Chapter 6:** This chapter contains the conclusion and the future scope. There are possibilities that can be explored further to get some more specic and better results, they are explained here.

Chapter 2

Literature Review

A literature survey is an assortment of content that plans to survey the basic purposes of current learning including substantive discoveries and in addition hypothetical and methodological commitment to specific point, and for the following project there was a requirement to study and understand what is EDA tool licensing[1], Flexlm[2] license management software which is solely responsible for distributing licenses to the end via network.

2.1 Getting Familiar with Software License Setup

2.1.1 Utilization of License Server and License Files

The vendor gives a layout of the process for installing a license server and it is also described how to use licenses from license files [3]. The vendor also provides information about the steps to be followed in order to install the license server. These steps are described below.

i **Choose the machine(s) on which the license server(s) will be installed.**

- Determine the number of licenses and machines on which FlexEnabled applications will be installed.
- Consider what technique, assuming any, you need to use to guarantee that, at whatever point possible, licenses are accessible to your end clients.

ii **Installation of license server components.** The distributor will supply a duplicate of their vendor daemon and directions describing the installation process. The license server supervisor, lmadmin or lmgrd, might be provided by the distributor or we can also download a duplicate from the site of Flexera Software. It is suggested that you install the most recent version of the license server supervisor.

iii **Obtain details of the license server machine(s) and send them to the publisher.** Normally publishers supply concurrent licenses that are locked to a specific license server. When licenses are held in license files, they are locked to the license server using an identity obtained from the achine. This identity is called a hostid and is platform specific. There are several different hostids available for each platform. The publisher will provide instructions on what host they are using, for your licenses and platforms. They may supply an application that you can run to obtain the host identity or ask you to use the FlexNet Publisher utility, lmhostid, which you can download from the site of Flexera Software. If you are

using lmadmin, it shows the standard host-identity for the machine on which it is running in System Information. Depending on the usage of the license model, the publisher may need some other information of your license server like the machine address on which it is running and details of your network also.

iv **Installation of licenses on the license server.** The publisher may give a particular region for installing the license files on the license server.

v **Installation of the FlexEnabled application on end user machines.** The publisher will supply installation directions to install the FlexEnabled application.

vi **Set up end user machines to access the license server.** There are a few strategies for designing the end user machine to get to a solitary license server or multiple license servers. These rely upon the contents of the license files provided by the publisher and your license server(s) configuration.

vii **Optionally, create an options file.** In the case that you need to restrain license utilization, configure logging, or kill the programmed reread of licenses, create an options file, which has all the access mentioned and install it in the same directory as the vendor daemon.

viii **Configure and start up the license server manager.** There is a key contrast between the configuration of lmadmin and lmgrd so the procedures required for each are independently plotted here.

2.1.2 License-Enabling Models

The distributor can make license-enabled products that use node-locked licenses or network licenses [4].

2.1.2.1 Node locked license-enabled products (non-runtime-based)

In the event that the distributor picks node-locked license with non-runtime-based enabling, the item does not make utilization of License Usage Management Runtime on the workstation where it runs. Following the distributor's guidelines for installation, you may be required to store the key for such an item in a vendor-selected node-lock registry. When you begin the application, it checks the node-lock catalog to guarantee that you have a legitimate permit. Data about utilization of the item is not logged. You cannot use the essential license tool to see the data or to get the reports about the item and its utilization.

2.1.2.2 Network license-enabled products (non-runtime-based)

If the vendor picks a network license with non-runtime-based enabling, the item does not make utilization of License Usage Management Runtime on the workstation where it runs and requires manual configuration on that workstation. The licenses are stored on

one or more network license servers. When the user at a client machine starts a licensed program, the license server determines whether or not a license is available.

2.1.2.3 Network license-enabled products (runtime based)

If the vendor picks a network license with runtime-based enabling, the item makes use of License Usage Management Runtime on the workstation where the product runs and requires some limited configuration on that workstation to run a licensed program.

```
Users of server:  (Total of 5 licenses issued;  Total of 4 licenses in use)

"server10" v10.0, vendor: klocwork
individual license

tyoung host01 /dev/tty (v10.0) (flex1.klocwork.com/27000 57756), start Mon 3/21 9:06 (linger: 1209600)
jfall host02 /dev/tty (v10.0) (flex1.klocwork.com/27000 17731), start Fri 3/18 12:54 (linger: 1209600)
jfall host03 /dev/pts/1 (v10.0) (flex1.klocwork.com/27000 29438), start Thu 3/24 9:33 (linger: 1209600)
jfall host04 /dev/tty (v10.0) (flex1.klocwork.com/27000 12791), start Thu 3/24 13:39 (linger: 1209600)

Users of client:  (Total of 10 licenses issued;  Total of 5 licenses in use)

"client" v10.0, vendor: klocwork
individual license

ajones ajones01 ajones01 (v10.0) (flex1.klocwork.com/27000 55789), start Fri 3/18 16:08 (linger: 604800)
ayeats ayeats01 ayeats01 (v10.0) (flex1.klocwork.com/27000 10557), start Thu 3/24 15:57 (linger: 604800)
bsmith bsmith01 /dev/tty (v10.0) (flex1.klocwork.com/27000 47368), start Fri 3/18 13:36 (linger: 604800)
pwall pwall2 /dev/tty (v10.0) (flex1.klocwork.com/27000 34499), start Mon 3/21 14:33 (linger: 604800)
tyoung tyoung02 /dev/tty (v10.0) (flex1.klocwork.com/27000 32739), start Mon 3/21 9:06 (linger: 604800)
```

Figure 2.1: Sample of lmstat output

The figure 2.1 shows the example output of lmstat which tells about the presence of feature and its installed count and in use license information and also the user who has checked out the license, from which machine user has got the license and how many parallel licenses been checked out.

2.1.3 License System

Jobs submitted to the compute farm by EDA users request for license to use particular EDA tools or features needed for the job. These license requests reach the license system and are served by the license servers. If all the available licenses get exhausted, new license request gets queued.

The work of the license queueing system is to queue new requests whenever the number of used licenses reach the set limit, until existing busy licenses become available again. Figure 2.2 represents the license system .

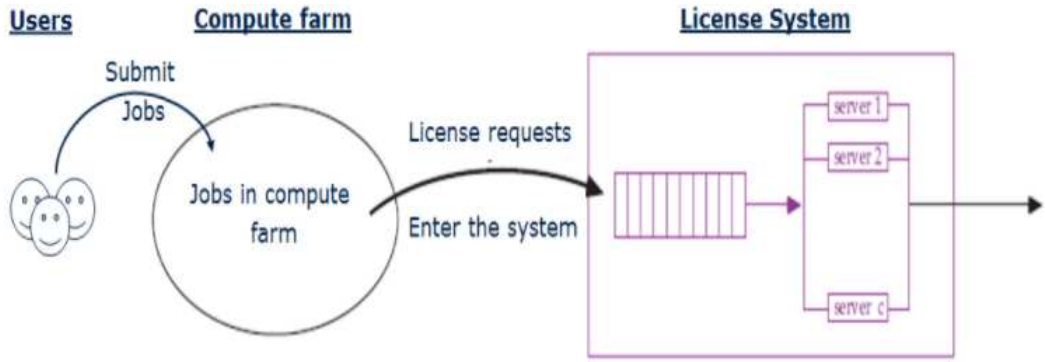

Figure 2.2: License System

2.1.4 FlexNet License Server (Flexera Software)

The FlexNet Cloud Licensing service [2] simplifies the enterprise experience by enabling software license enforcement and usage-based licensing for both on-premises and cloud-based applications in a managed environment hosted by Flexera Software. The basic components of a FlexNet Publisher license server are as shown in the figure 2.3.

i **License server manager:** These are lmadmin or lmgrd provided by your product provider or accessible from Flexera Software.

ii **License file:** This is made by your product provider. In this report the provider of a FlexEnabled application is alluded to as the publisher.

iii **Vendor daemon:** This is made by the publisher. Every publisher has their own vendor daemon. On the off chance that you have FlexEnabled applications from different publishers, you should install multiple vendor daemons.

iv **Debug log:** This is written by the license server manager. The components mentioned in next points may be present on a license server.

v **Options file:** These are used to limit license utilization; for example, to allocate a particular number of licenses to a user or group of users.

vi **Report log:** This is a file that can be used by the FlexNet Manager, Flexera Softwares license management product. You enable report logging using the options file.

vii **Trusted storage:** Some publishers use trusted storage to store licenses. When trusted storage is used, the publisher provides additional components that create trusted storage and add licenses to it. Trusted Storage is further discussed in the coming section.

2.1.5 Design Ecosystem

The license ecosystem contains the tool, license server and the compute farm. Traditional cost optimization efforts have concentrated only on license server and compute farm

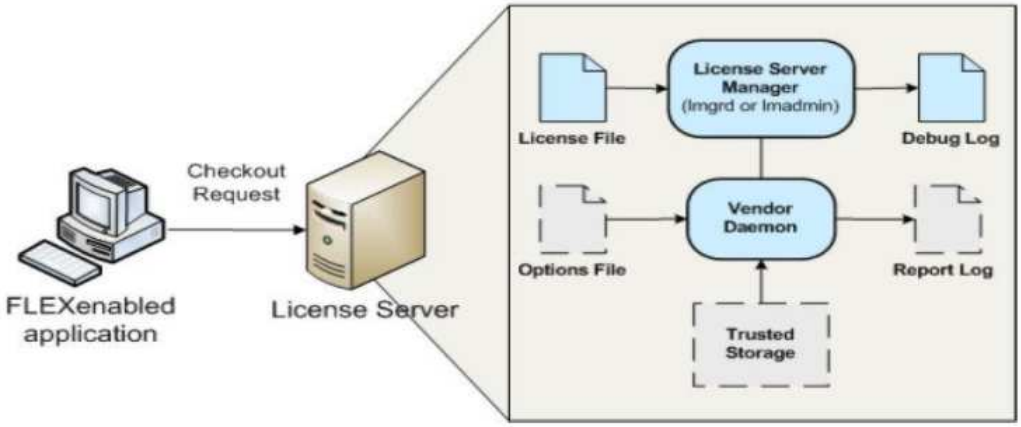

Figure 2.3: FlexNet Based Server configurations

information stored in databases, without the design information generated by the tool. In such setups one cannot derive any relationship between tool usage and the design process runs.

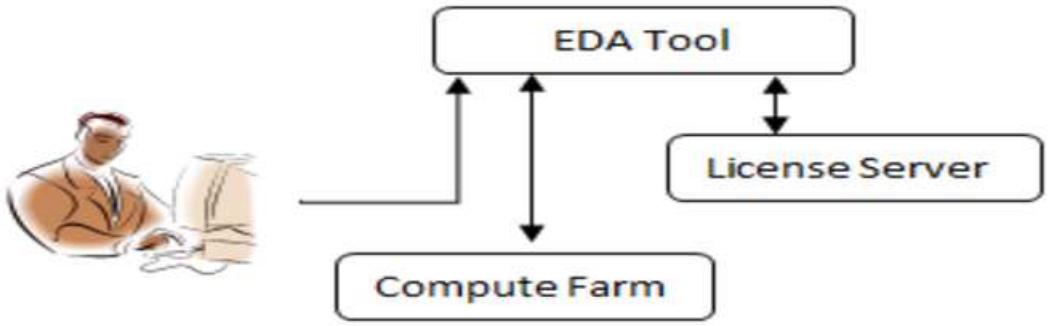

Figure 2.4: License Ecosystem

Understanding the purposes and the usage behavior with the design context are essential for gathering better actionable insights. So a mechanism of gathering design/tool data, storing and analyzing the information is essential for the next generation of EDA cost optimization efforts.

2.1.5.1 License Server

Jobs submitted to the compute farm by EDA users request for license to use particular EDA tools or features needed for the job. These license requests reach the license system and are served by the license servers.

2.1.5.2 Compute farm and LSF

Compute farm is an accumulation of clients, used to run high asset hungry occupations put together by the EDA designers. Infineon utilizes Load Sharing Facility provided by IBM. IBM platform LSF is an intense workload administration stage which gives a complete arrangement of insightful, strategy driven scheduling highlights that empower you to use the greater part of your register foundation assets and guarantee ideal application execution.

2.1.5.3 Electronics Design Tools

Electronics Design Tools (EDA) are a class of software tools for designing integrated circuits, these tools work together in a design flow that chip designers use to design and analyze entire semiconductor chips. These tools are mainly used to work on the Design, Simulation, Analysis and Verification and Manufacturing Preparation.

2.1.6 Trusted Storage

Trusted storage is a protected area that is bolted to the machine on which it is located utilizing a mix of machine personalities. The substance of trusted storage are encoded and must be gotten to by FlexEnabled parts. This technique for putting away licenses empowers your publisher to give additional license models and automate some licensing forms.

2.1.6.1 Trusted Storage Components on a License Server

The basic components of a FlexNet Publisher license server that uses licenses held in trusted storage are as illustrated in figure 2.5.

i License server manager binaries lmadmin or lmgrd provided by the distributor or accessible from Flexera Software.

ii Bootstrap license file is made by the distributor. This is required for starting the license server manager when the license server is using trusted storage to store all of its licenses.

iii Vendor daemon is also made by the distributor. This must be the distributor vendor daemon that can get to the trusted storage. Make sure that you always use the right vendor daemon provided by the distributor: a prior form that is just ready to use the license files will not have the ability to utilize licenses held in trusted storage.

iv Trusted storage - This contains licenses in satisfaction records.

v Server activation utility - This is a FlexEnabled part that deals with the exchanges with the distributor server and makes and deals with the contents of trusted storage.

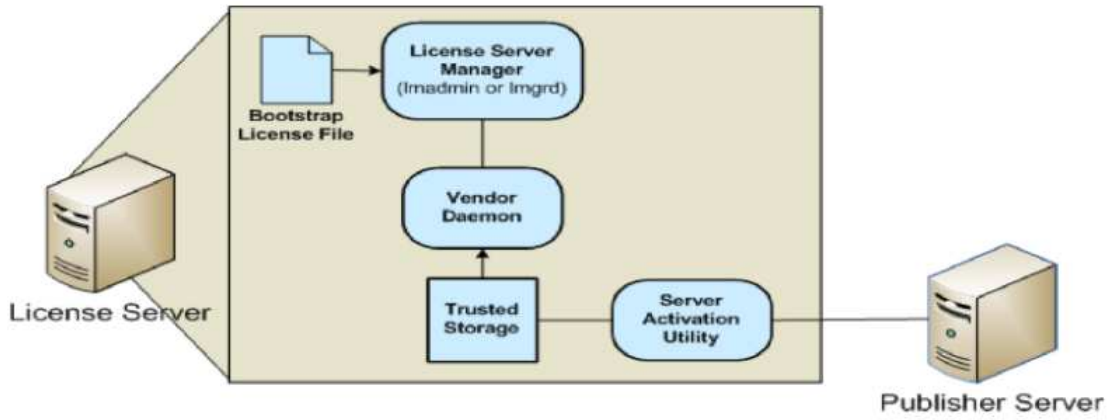

Figure 2.5: Distributions of Node-Locked Licenses to Networked Machines

Node locked systems are the one who get the special privilege to access license all the time and this access is provided by the license manager where .opt file will have reservations lines which are responsible for the same and same will be kept on the trusted storage, as soon as user provides license server path while invoking EDA tool, Tool will contact the license server and server will validate the user id then it will provide access to the EDA tool.

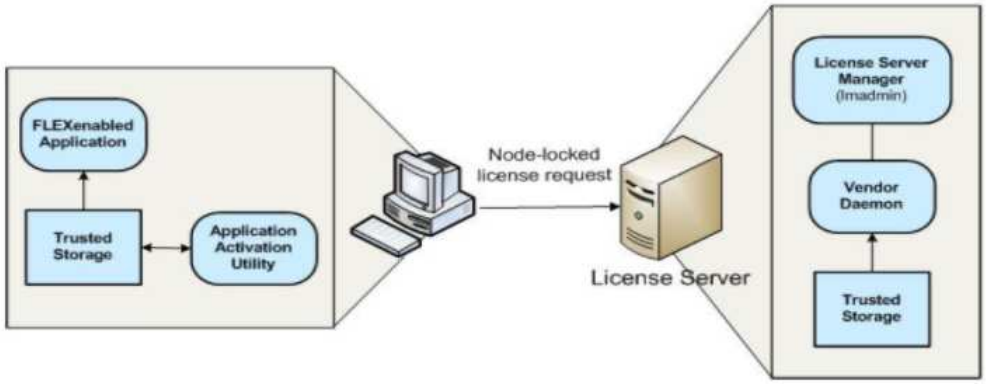

Figure 2.6: Node Locked License request to License Server

The FlexEnabled components which are required in implementing the distribution of node-locked licenses to networked licenses, machines using trusted storage are:

i License server manager - Use lmadmin as the license server manager as it shows the details of licenses held in trusted storage which can be activated.

ii Vendor daemon is also made by the distributor. This must be the distributor vendor daemon that can get to the trusted storage. Make sure that you always use the right vendor daemon provided by the distributor: a prior form that is just ready to use the license files will not have the ability to utilize licenses held in trusted storage.

iii Trusted storage on the license server - This contains licenses in satisfaction records which can be transported to a networked machine.

iv Server activation utility - This is a FlexEnabled part that deals with the exchanges with the distributor server and makes and deals with the contents of trusted storage. This utility also manages the contents of the servers trusted storage through return, repair, and modify requests.

v Application activation utility - This is FlexEnabled part that requests a license from the enterprise license server or the distributor's activation server and makes and deals with the contents of trusted storage. The distributor can integrate this functionality into a component that supports other functions, for example, the utility could be integrated into the FlexEnabled application installer.

vi Trusted storage on the network machine - This contains licenses locked to the machine.

2.1.6.2 License Usage Management Vendor Perspective

License Usage Management advantages software merchants by empowering them to do the following:

1. Ensure that all clients use EDA software tools licenses inside entitled limits,
2. Base software product costs on real utilization
3. Protect scholarly property from unapproved utilization
4. Increase general income as clients get all the licenses they need.

2.1.6.3 Basic Concepts of License Usage Management

A license, with regards to License Usage Management, is consent to use an instance of an authorized software item or administration, as indicated by the premise on which the distributor charges for the item or administration. The term license in this setting does not allude to the license agreement that represents utilization of, and rights to, an item.

2.1.7 Overview of Trusted Storage Flexlm

Trusted capacity is a protected area that is bolted to the machine on which it is found utilizing a blend of machine personalities. The substance of trusted stockpiling is encoded and must be gotten to by Flex Enabled parts. This strategy for putting away licenses empowers your publisher to give extra permit models and mechanize some permitting courses of action.

2.2 Queuing Theory

Queuing theory is the mathematical study of waiting lines, or queues. Queuing theory is used to understand the behaviour of queuing systems. The main components of a queuing system include queues (waiting line), customers (in need of service) and servers (who serve the customers). The basis of queuing theory lies with the understanding of arrival and service rates of the system. A model is examined so that queue lengths and waiting times can be predicted for numerous licensing applications.

Lindley [5] presumed that where there is a solitary queue and a solitary server taking care of it; the hypothesis of different and multiple queues or numerous servers appears, with the exception of under simplifying suspicions which do not generally compare to reality, to be an issue of extensive trouble. Past work regarding the matter has basically been kept to an extraordinary situation where the clients touch base aimlessly, as, in a current paper by Kendall [6]; the present hypothesis makes no such assumption and enables the customer to join the queue in different ways, however the hypothesis streamlines when the more restrictive presumption is made. The focal point of enthusiasm for the present hypothesis is the waiting times of the clients.

Table 2.1, describes the terms used in queuing theory in our licensing context. The arrival rate is calculated from the difference between arrival times of two consecutive requests. The average of all the arrival rates will be the total arrival rate of the system.

Table 2.1: Queuing theory Definitions

Attribute	**Definition**	**Licence System**
Arrival Rate	The rate at which customers arrive	The rate at which license requests arrive
Service Rate	The rate at which servers serve the incoming customers	The rate at which the license requests are served by the license servers

2.2.1 Model Attributes

In order to identify the type of queuing model to use for our analysis we investigate the arrival process, service time distribution, service discipline, queue type and the number of servers.

i **Arrival Time**

The license requests arrival distribution was found to be exponentially distributed, as shown in figure **??** below. Each arrival is independent of the previous arrival.

ii **Service Time**

The service time which is the amount of time taken to serve a license request along

with the corresponding job execution was also found to be exponentially distributed, shown in figure **??**. The service time of one request is independent of other requests, because each job is unique and has independent runtimes.

iii **Service Discipline**

The license system considered was FlexNet license server with FIFO (First in First Out) service discipline.

iv **Number of Queued Request**

There is no fixed queue limit in the license system; there can be any number of license requests in the license system, representing an infinite queue system.

v **Number of Servers**

In our case, the number of available licenses is the number of servers. The number of available license is variable and at any point in time depends on a number of factors.

2.2.2 Queue Formation in System

In view of the above outcomes, M/M/c show is picked. MMC - In queuing hypothesis, a discipline within the numerical hypothesis of likelihood, the M/M/c queue (or ErlangC model) is a multi-server queuing model. In Kendall's notation [6] it depicts a framework where entries shape a solitary queue and are administered by a Poisson procedure, there are c servers and job service times are exponentially appropriated. It is a speculation of the M/M/1 queue which considers just a solitary server. The model with interminably numerous servers is the M/M/ queue. Entries happen at rate as indicated by a Poisson procedure and move the process from state i to i+1. Service times have an exponential circulation with parameter in the M/M/c queue. There are c servers, which serve from the front of the queue. On the off chance that there are not as much as c jobs, a portion of the servers will be sit still. In the event that there are more than c jobs, the jobs queue in a cradle. The cradle is of unbounded size, so there is no restriction on the quantity of solicitations it can contain.

2.3 Usage of License Queues (Wait-times)

Following could be the applications, possible with the understanding of license queue wait times and tool runtimes through the discussed methods.

2.3.1 Data analysis

We can look back into the historical data, and analyse how the results would have varied from the original ones by changing the situation parameters. Also this type of analysis is helpful in designing and configuring future licensing systems. This is possible with

the queuing model discussed, which helps to conduct a variety of what if analysis. For example, we can get answers to; what would have been the queue wait times if the license number was modified?

2.3.2 Drive licensing operational decisions

Analytical models discussed in this thesis can help drive the operational decisions of the license management teams. Important decisions can be made based on fact based quantitative approach. For example we can get answers to questions like, by how much should we reduce the license numbers to increase the utilization? By how much should we increase the license numbers to reduce wait times? Should we increase the license numbers? Below are the steps that can be followed for answering some of these questions.

2.4 Tracking Design Approach

The steps used in the solution can be split into the following steps:

i Track all the new tool logs in real-time through our custom built file watcher (Watcher daemon).

ii Create an instance of a custom built parser for each log file and then parse and package the required information into a JSON format (Perl Parser).

iii Push them to Elastic Search engine via Redis.

iv Further data retrieval and processing of text analytics and process analytics are done through custom built R scripts (R). Following diagram depicts the architecture used for tracking design flow.

2.4.1 Data Archiving and Extraction

The count and size of log files generated by EDA tools are very high. So to mine data from these log files one has to move from traditional storage and extraction methods (relational-sql databases) to latest high performance technologies. To overcome this technology challenge we moved to Elastic search which is a very versatile full-content search and 10 analytics engine. It enables one to store, look, search, break down, analyze huge volumes of data quickly and in close continuous.

2.4.1.1 Elastic Search

It is largely utilized as the hidden engine/innovation that forces applications which have complex search elements and pre-requisites. Some of the important features:

i RealTime Data Processing

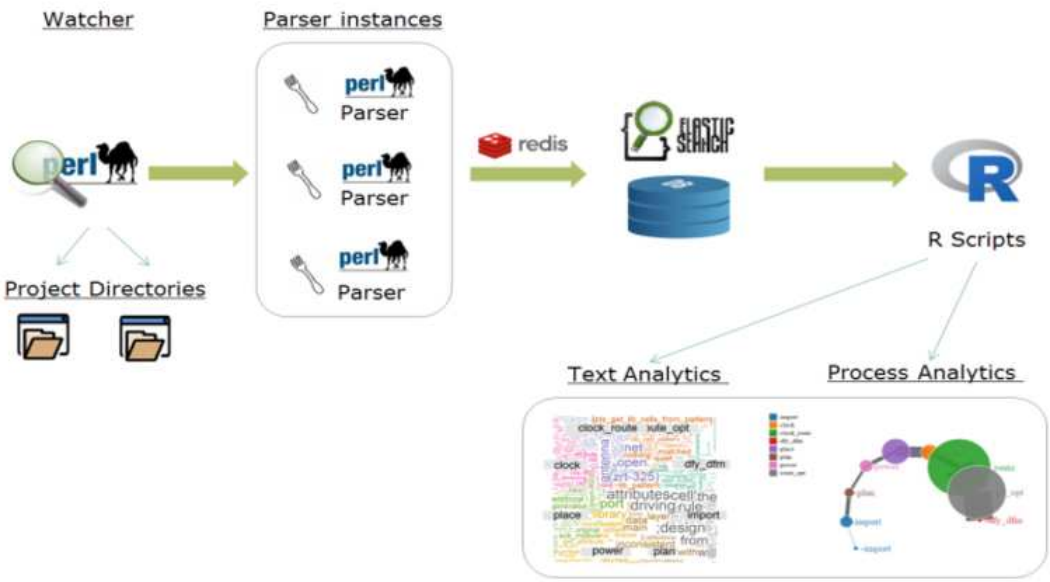

Figure 2.7: Tracking Design Architecture

ii Massively Distributed

iii High Availability

iv Full-Text Search Engine and Schema-Free

2.4.1.2 Redis

Redis is an open source, advanced key-value store and a genuine answer for building superior, versatile web applications. Redis has two features that set it apart from quite a bit of its competitions

i Redis holds its data store entirely in memory.

ii Redis can replicate data to any number of slaves.

2.4.2 Analytics Engine

Our business problem called for a fast paced analytics engine with high performance visualizations for our text analytics and process analytics needs. We use R which has quickly raised as the market leader in data analytics and enables easier data analysis and loads of visualization methods to represent our complex data structures.

2.4.2.1 R

R and its libraries implement a wide assortment of statistical and graphical arrangements, including straight and nonlinear demonstrating, established factual tests, time-series ex-

amination, characterization, clustering and others. R is effortlessly extensible through functions and expansions, and the R community is noted for its dynamic commitments regarding packages. R mines data and tackle issues through information perceptions , data visualizations.

Once the data scientist has finished the frequently tedious procedure of cleaning and setting up the data for analyzing. R is a well known programming package for really crunching the numbers and visualizing the outcomes. An open-source statistical demonstrating dialect, R has customarily been prominent in the scholarly group, which implies that bunches of data scientist will be comfortable with it.

R has actually a great many augmentation packages that enable analysts to embrace particular undertakings, including content analysis, speech recognition, and apparatuses for genomic sciences. The focal point of a flourishing open-source environment, R has turned out to be progressively well known as developers have made additional add-on packages for taking care of enormous datasets and parallel processing methods that have come to command factual modelling today.

2.4.2.2 XGBoost Method

XGBoost (eXtreme Gradient Boosting)is an advanced implementation of gradient boosting algorithm. XGBoost is widely used for Kaggle competitions. This algorithm has become the ultimate weapon of many data scientist. It is ahighly sophisticated algorithm, sufficiently intense to manage a wide range of anomalies of data. It is easy to build a model using XGBoost. But, the difficult part is to improve the model using XGBoost (it made us struggle a lot to get to the set target). This algorithm uses multiple parameters, for our application, we are using 26 parameters for the prediction. To improve the model, parameters are tuned and the data is cleaned first for the better results. Gradient boosting has empirically proven itself to be highly effective for a vast array of classification and regression problems. One arena where this becomes particularly apparent is the competitive machine learning scene.

There are different high-level interfaces. Presently, there are interfaces of XGBoost in C++, R, Python, Julia, Java and Scala. The main functionalities in XGBoost are implemented in C++, in this manner it is quite simple to share models among various interfaces. We are for the most part going to concentrate on the R packageXGBoost, which has an easy to use interface and furthermore thorough documentation.

2.4.2.2.1 Model Features The implementation of the XGBoost model supports the characteristics of the scikit-learn and R usage, with new increments like regularization. Three principle types of gradient boosting are upheld:

i **Gradient Boosting** algorithm also called gradient boosting machine including the learning rate.

ii **Stochastic Gradient Boosting** with sub-sampling at the row, column and column per split levels.

iii **Regularized Gradient Boosting** with both L1 and L2 regularization.

2.4.2.2.2 Algorithm Features The execution of the algorithm was designed for effectiveness of compute time and memory assets. A plan objective was to make the best utilization of accessible assets to train the model. Some key algorithm implementation highlights include:

i **Sparse Aware** implementation with automatic handling of missing data values.

ii **Block Structure** to support the parallelization of tree construction.

iii **Continued Training** so that you can further boost an already fitted model on new data.

2.4.2.2.3 Advantages of XGBoost

i Execution Speed

ii Model Performance

iii Regularization

iv Parallel Processing

v High Flexibility

vi Handling Missing Values

vii Tree Pruning

viii Built-in Cross-Validation

ix Continue on Existing Model

2.5 Adaption of Waterfall Method for the System

The following adaption of waterfall methods for used the projects works.

2.5.1 Planning and Requirement Analysis

Global License management team decided to have a monitoring dash board for effectively monitoring the license peak usage of various EDA vendors and there features, and it should be a web based application so that it can be used over the intranet and can shared to any user in the Infineon office across the world wide.

2.5.2 Defining Requirements

Requirement includes a web server preferably apache with MySQL DB for storing the license information and it should produce web GUI interface where GLM can observe heatmaps of various vendor/ feature usage which typically displays hourly based license peak usage and on available license counts legend will be defined and then based on the usage counts colour coding will be applied in heatmap.

2.5.3 Designing the item structural engineering

For designing the tool we have used various technology to make web based tool and have effective algorithm which is scalable and robust in size, the technologies which are used in this project are Perl CGI, HTML, MySQL and JavaScript etc.

2.5.4 Building or Developing the Product

In this phase of SDLC the genuine improvement begins and the item is fabricated. The programming code is created, We have taken coding rules characterized by their association and programming tools such as compilers, debuggers and so forth are used to produce the code.

2.5.5 Testing the Product

Testing was done on the topics of Unit testing, Integration testing and System testing and based on the negative results we have acted and modified the code to meet the requirements and abide by the software standards.

2.5.6 Deployment in the production and Support

Once the software tool is tried and prepared to be sent is discharged formally in the fitting production sector. And later end user have access to the software which has been developed and if the user finds any bugs and seeks any help on how to then support phase will help them to use the tool at maximum usage.

Chapter 3

Description of the problem

Software license and maintenance costs accounts fornearly one-third of IT budgets. So were not exaggerating, when we say that your companys software is already a big investment. If you break the terms of software licenses, you could face retroactive charges, fines, or even criminal charges. Managing high-value software assets can be difficult for two primary reasons, the increasing complexity of software licensing agreements and the lack of software license management best practices. This chapter focuses on the problem domain in literature survey and objectives that are formulated.

3.1 Problem Statement

Effective License Management has been a serious problem facing in most of the IT as well as Semiconductor industry for the recent decade. By some estimation we can say EDA budget is the second largest budget for the fabless semiconductor industry/company and an effort to make effective management of the same gives the competitive edge in business. A few strategies have been proposed all together to face such issue. May be the most simplest approach to take care of issue is to have real time monitoring dash board which showcase the hourly based peak usage and which also convey the percentage of utilization of the same license usage and dashboard will be an web application. This project is all about the building such system to help in doing effective license management.

3.2 Research direction

As an employee of the company, this has been found that earlier the team was doing all the manual efforts for managing the licenses world-wide. There was a real need of a centralized system where we can find all the required license information and we do not have to switch here and there for the same, which is also a reliable and trustworthy system. Also, it was already listed in to do list of the team to track the queues, so that we can get an idea of how much the user is suffering because of unavailability of the licenses at times. And we could serve them better.

3.3 Existing Situation of EDA

Challenges faced in the existing process of effectively managing and providing EDA licenses are:

- Manually checking usage patterns of license checkouts
- Manually looking at alerts/data and checking queueing patterns of licenses
- Inferring user demand
- No way to track the queued requests automatically
- Users do not have any idea about what is the best time to schedule their jobs
- If user is queued, how much will he has to wait

3.4 Objectives

Objectives for proposed project are as follows:

i A centralized dashboard to serve all the license requirements

ii To monitor the real-time license usage and utilization factor.

iii Security- only authorized users must have access to the system so that others should not take the disadvantage of the same.

iv Data should be always available and also consistent through the time line, any data inconsistency is not

v Ensure that users use software licenses within specified limits.

vi Increase overall revenue as users use all the licenses they need.

vii Track the queued requests also, so as to serve them in a better way

viii Prediction of wait times for the queued users for different tools

3.5 Project Modules

i Heatmap for Infineon worldwide (Showing license usage on hourly basis).

ii License view and Watchdog.

iii Costcenter Interface and Option Files Automation

iv Costcenter Heatmaps

v Waiting Lounge and Suggest Limits

vi Prediction on wait times for queued license requests

vii License Ticker (A linux application for end users having license information on single GUI)

Chapter 4

System Design and Implementation

Design process is the representation of making of a system, or a process of producing a model, which will be used to develop a system. The design process for the product framework regularly has two levels, at the first level the focus is on deciding which modules are needed for the framework, the details of these modules and how the modules can be interconnected.

4.1 System Overview

This includes below three points-

i The representation of a system, which includes the mapping of usefulness onto equipment and programming parts, mapping of the product structural planning onto the equipment building design, and human association with these segments.

ii An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.

iii A review involves the most imperative, pervasive, top-level, vital innovations, choices, and their related methods of reasoning about the general structure (i.e., fundamental components and their connections) and related qualities and conduct.

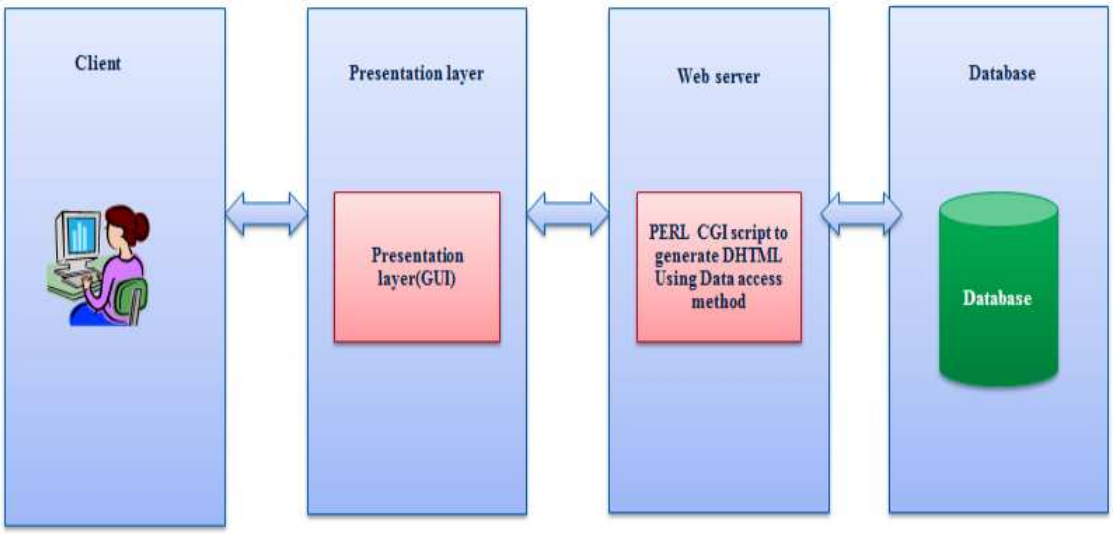

Figure 4.1: System Overview of Heatmap

4.2 System Requirements Specification

A System Requirements Specification is a complete portrayal of the conduct of the framework to be created. The result of the SRS stage is the framework prerequisite particular archives, which portrays the complete outside conduct of the proposed framework. It incorporates the practical and non-utilitarian necessity for the product to be created. Prerequisites must be quantifiable, testable identified with recognized needs or opportunities, and characterized to a level of subtle element adequate for framework plan.

4.2.1 Non-Functional Requirement

The procedure of securing the administrations the framework ought to give and the requirements under which it must work is called prerequisites building. Particulars of necessity building procedure are necessities definition, prerequisites particular and programming determination. Prerequisites definition is an announcement, in a characteristic dialect in addition to charts of what administrations the framework is required to give and the limitations under which it should work. It is produced utilizing client supplied data. Prerequisites particular is an organized report which sets out the framework benefits in point of interest. This report ought to be exact and here and there called as practical determination.

4.2.2 Availability

Application accessibility is the degree to which an application is operational, useful and usable for completing or fulfilling a customer's or business' prerequisites. This measure is utilized to examine an application's general execution and focus its operational insights in connection to its capacity to execute as needed.

4.2.3 Reliability

Software Reliability is the likelihood of disappointment free programming operation for a predefined time of time in a foreordained domain. Programming Reliability is in like manner a discriminating component influencing framework trustworthiness. It changes from hardware steady quality in that it mirrors the configuration flawlessness, instead of assembling faultlessness. The high flightiness of programming is the significant contributing variable of Software Dependability issues. Programming Reliability is not a component of time despite the fact that scientists have concocted models relating the two. The demonstrating system for Software Reliability is coming to its flourishing, yet before utilizing the method, we should deliberately choose the fitting model that can best suit our case. Estimation in writing computer programs is still in its earliest stages. Awful quantitative techniques have been produced to speak to Software Reliability without excessive hindrances.

4.2.4 Performance

By Performance we mean response time for end clients or throughput as far as business exchanges every second. Ventures put resources into ability to meet these execution targets and afterward continue observing the base and include more limit as time progresses. What is truly needed is to respect execution as a coordinated part of the product usage or support life cycle.

4.2.5 System Security

The protection of PC based assets that include hardware, software, data, procedures and people against unauthorized use or natural Disaster is known as System Security. System Security can be divided as follows:

- Security
- Integrity
- Privacy
- Confidentiality

4.3 Software Requirements

- Perl and CGI scripting platform
- MySQL
- Apache Webserver
- Languages used: HTML/CSS, PHP, Python, JavaScript and JQuery
- Any Web browser on any Operating system

4.4 Hardware and System Requirements

i System : Pentium IV 2.4 GHz and above

ii Dedicated linux CentOS 7 server

iii Installed and configured http server (including ssl and Kerberos/SSO)

iv Dedicated Server for running MySQL DBs - 4 CPUs and 20G RAM

v Dedicated R-Server

vi Hard Disk : 100 GB

4.5 System Architecture

System architecture is a conceptual model which explains structural behaviour of the system and architecture can comprise system components and for software architecture is the high level of structure of a software system. Figure 4.2 shows the Heatmap tool architecture.

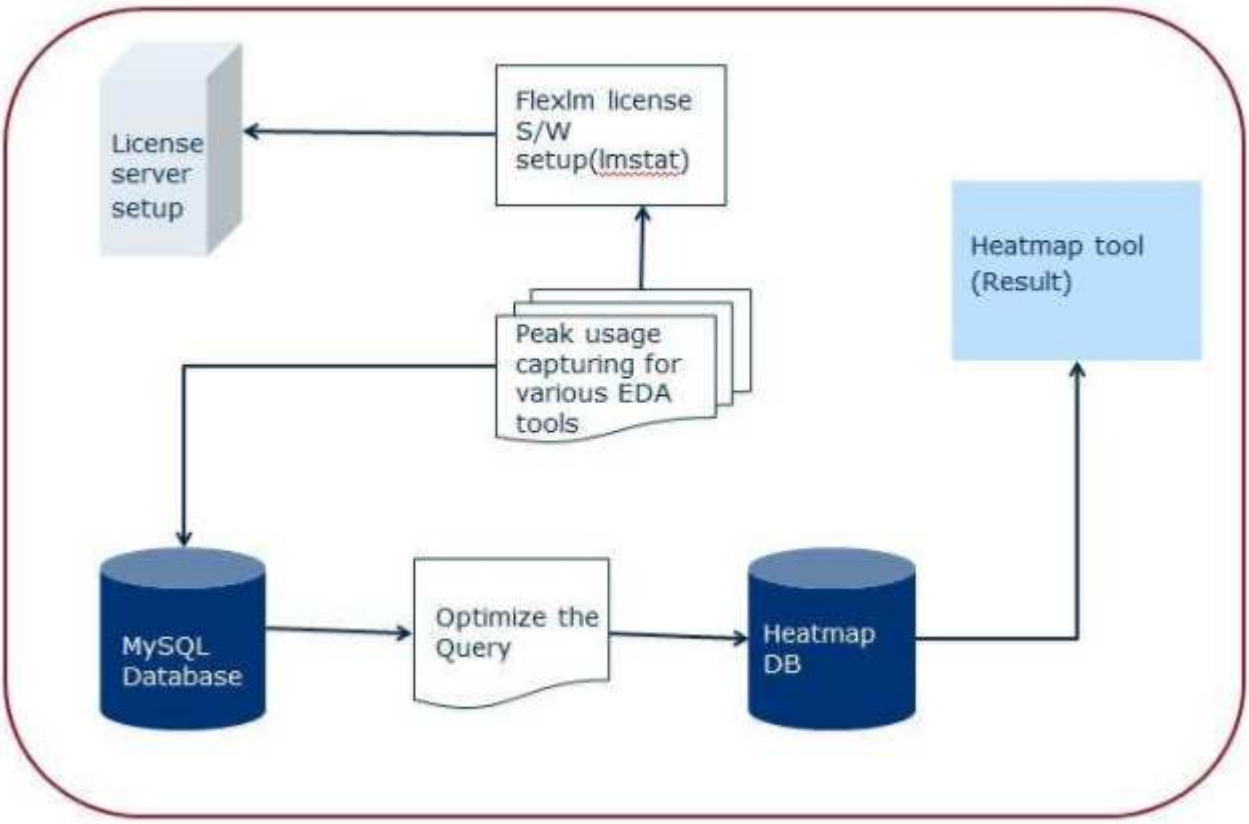

Figure 4.2: System Architecture of Heatmap

4.5.1 Overall Design of the Heatmap and Waiting Lounge

Software design is a procedure to change client prerequisites into some suitable structure, which helps the software engineer in programming coding and execution. For evaluating client prerequisites, a SRS (Software Requirement Specification) record is made though for coding and execution, there is a need of more particular and nitty gritty necessities in programming terms. The yield of this procedure can specifically be utilized into usage as a part of programming dialects.

Programming outline is the initial phase in SDLC (Software Design Life Cycle), which moves the focus from issue area to arrangement space. It tries to determine how to satisfy the prerequisites said in SRS.

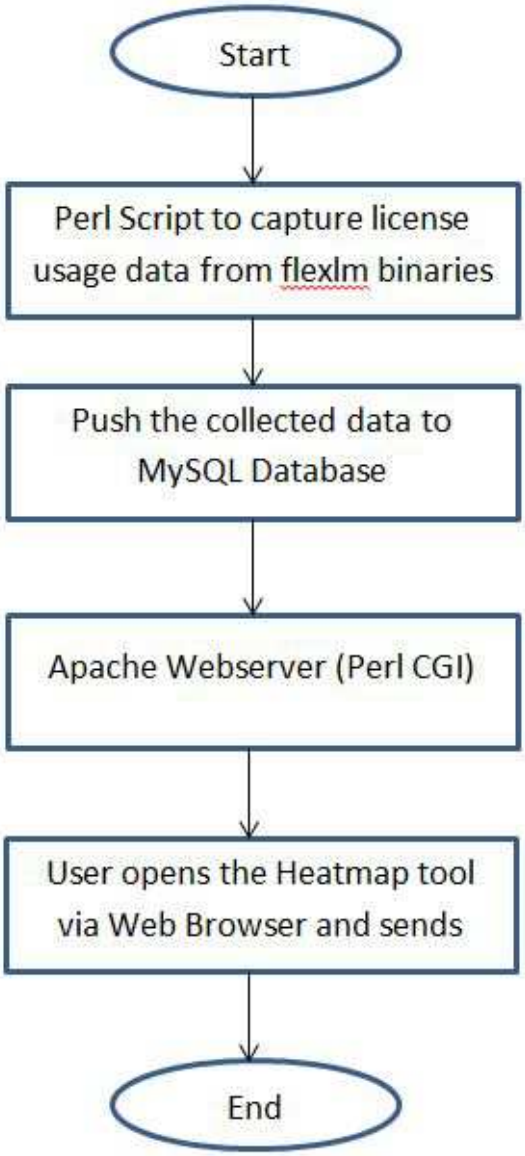

Figure 4.3: Overall Design of the Heatmap

4.5.2 License usage Data Gathering design

License usage data will be in the format shown in figure 4.4 and command to get such information is **lmutil lmstat a c port@server**. Basically it reaches out license server with mentioned port number and executes and return the output of the command. Execution will be processed by perl scripts to capture data and captured data will be pushed into the database. The flowchart of the process is shown in figure 4.5

```
Users of redhawk_tpm:  (Total of 1200 licenses issued;  Total of 1196 licenses in use)

  "redhawk_tpm" v9999.99, vendor: apacheda, expiry: 30-sep-2017
  floating license

    mausssa vihlc1688 /dev/pts/0 (v2002.7) (ulicserv1.muc.infineon.com/2028 1701), start Fri 7/21 10:55
    feuerbam vihlc1021 /dev/pts/0 (v2002.7) (ulicserv1.muc.infineon.com/2028 2101), start Fri 7/21 10:56
    pummerer vihlc1180 /dev/pts/38 (v2002.7) (ulicserv1.muc.infineon.com/2028 702), start Mon 7/24 8:16
    feuerbam vihlc1104 /dev/pts/0 (v2002.7) (ulicserv1.muc.infineon.com/2028 4407), start Tue 7/25 7:26
    molihua vihlc1305 /dev/pts/2 (v2002.7) (ulicserv1.muc.infineon.com/2028 4603), start Tue 7/25 8:28
    molihua vihlc1690 /dev/pts/0 (v2002.7) (ulicserv1.muc.infineon.com/2028 4211), start Tue 7/25 8:40
    molihua vihlc476 /dev/pts/0 (v2002.7) (ulicserv1.muc.infineon.com/2028 903), start Tue 7/25 8:41
    feuerbam vihlc042 /dev/pts/0 (v2002.7) (ulicserv1.muc.infineon.com/2028 2007), start Tue 7/25 9:06
    mausssa vihlc1727 /dev/pts/0 (v2002.7) (ulicserv1.muc.infineon.com/2028 4907), start Tue 7/25 16:05
    molihua vihlc1285 /dev/pts/1 (v2002.7) (ulicserv1.muc.infineon.com/2028 4080), start Wed 7/26 2:54
    molihua vihlc1708 /dev/pts/2 (v2002.7) (ulicserv1.muc.infineon.com/2028 2519), start Wed 7/26 5:49
        1185 RESERVATIONs for USER lmadmin (ulicserv1.muc.infineon.com/2028)

Users of resistance_calculation:  (Total of 200 licenses issued;  Total of 0 licenses in use)
```

Figure 4.4: lmstat File Structure

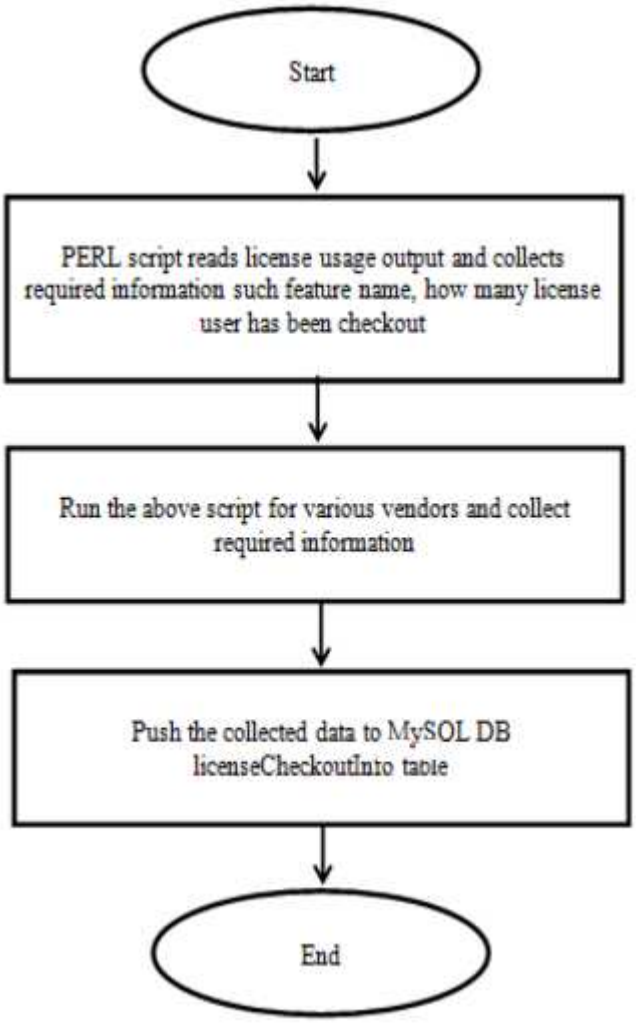

Figure 4.5: Gathering of license usage information

4.5.3 Database Table structure

There are four major tables as vendors table, features table, junction table and license checkouts table. Figure 4.6 shows the relational view between the tables and information about the fields.

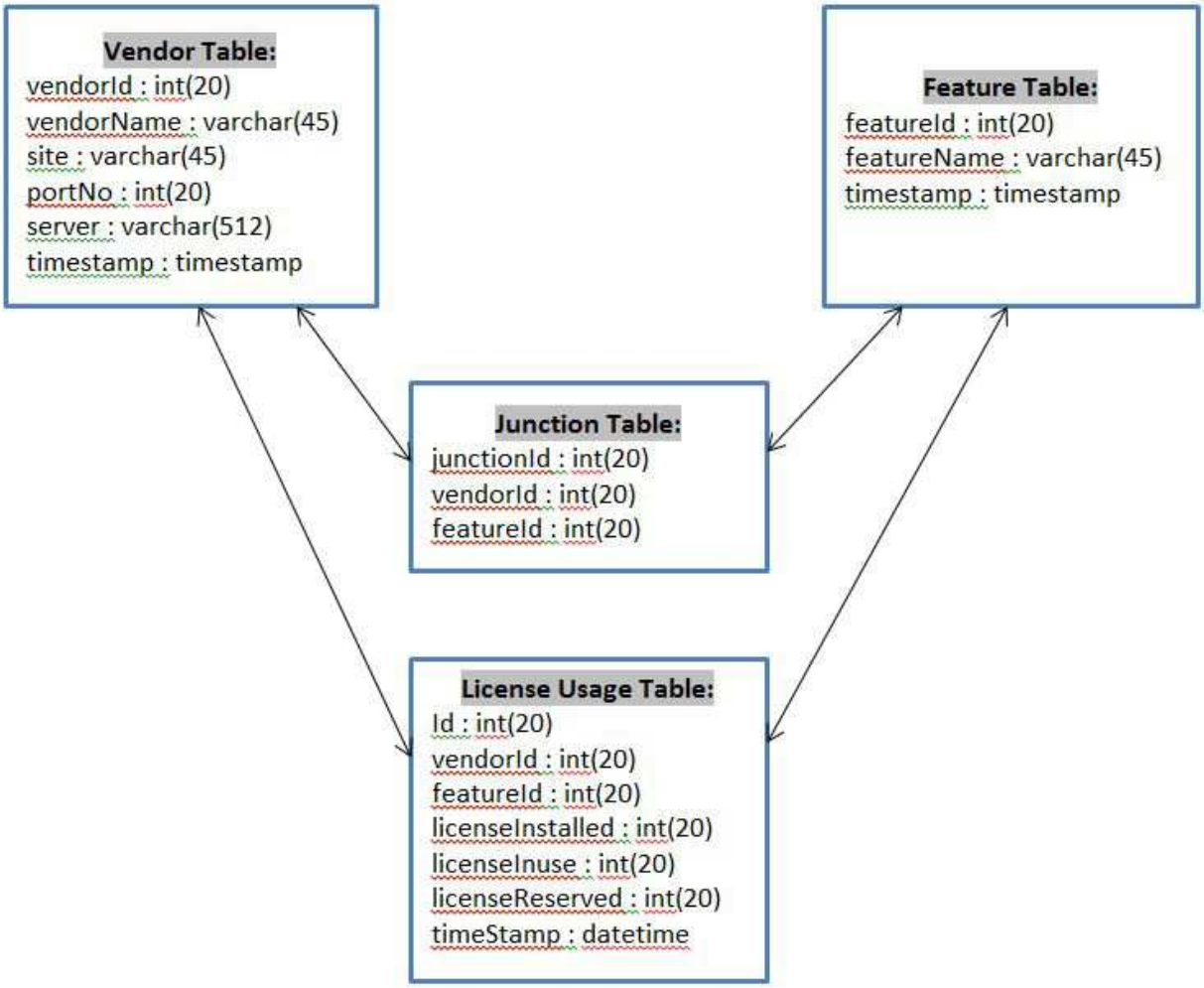

Figure 4.6: Relational View of the Database Schema

4.5.4 Flow Chart for CGI script to generate DHTML

Steps included to get to the end output of the Heatmap:

i First CGI script tries to authenticate wether the user is authorized to access the data or not.

ii If yes, then the script reads all teh required fields such as region, vendor, feature and month or else ends the processing.

iii After that it will query the database with the required information and prints out the data in the HTML format on the client's browser.

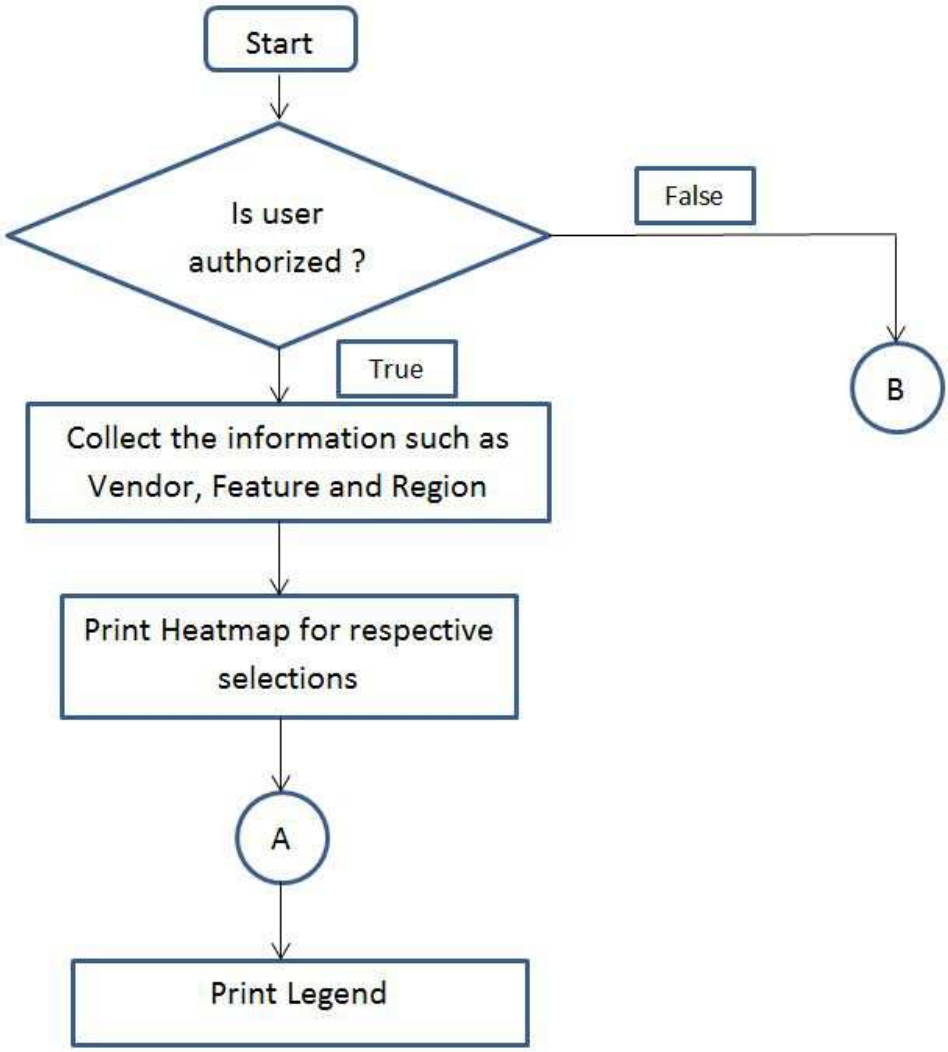

Figure 4.7: Flow chart for generating DHTML

4.6 System Implementation

Execution of any product went constant before by imperative choices in regards with determination of stages, the dialect utilized, and so on. These choices are frequently impacted by a few variables, for example, genuine environment in which the framework lives up to the expectations, the velocity that is needed, the security concerns, and other execution particular points of interest. There are two noteworthy execution choices that must be made before undertaking the implementation of the system. These are:

i Selection of platform (operating system).

ii Selection of programming languages for the development of application.

4.7 Selection of platform

Windows has acquainted a few new highlights with the windows line, including quicker start-up. Another highlight is windows interface is ostensibly more easy to use including the system for creating topics for the desktop environment. The Quicker client exchanging, which permits a client to spare the current state and open utilization of their desktop and permit another client to sign on without losing the data. Windows investigates the execution effect of visual impacts and uses this to figure out if to empower them, in order to keep the new usefulness from destroying extra handling overhead. Clients can further modify these settings.

4.8 Selection of Programming language for the development of applications

For the execution of our task, we require adaptable framework usage dialect. Accumulation ought to be generally straight forward compiler, give low-level access to memory, build the guide productively for guiding the machine, and require insignificant runtime support. System ought to be incorporated for a mixture of PC stages, also working frameworks with negligible changes to its source code. For Graphical User Interface, programming languages picked must be easy to use, secured, construction modelling impartial and versatile to achieve all requirements. In our project, we have used Perl/CGI, HTML/CSS, PHP, Python, R, JavaScript.

4.8.1 Perl

Perlis a family ofhigh-level, universally useful, translated, element programming dialects. The dialects in this family incorporate Perl 5 and Perl 6. Despite the fact that Perl is not formally an acronym, there are different acronyms being used, for example, Practical Extraction and Reporting Language. Perl was initially grown by Larry Wall in 1987 as a universally useful Unix scripting dialect to make report. From that time later, it has experienced numerous progressions and amendments. The most recent significant stable update of Perl 5 is 5.18, released in May 2013. Perl 6, which was started as an upgrade of Perl 5 in 2000, in the long run, advanced into a different dialect. Both the languages keep on being produced freely by distinctive advancement groups and generously acquire thoughts from one another. The Perl language obtain highlights from other programming languages including C, shell scripting (sh), Python.

4.8.1.1 Features of Perl

i Perl runs fundamental CGI and is the dialect that made CGI omnipresent on the web.

ii It takes the best features from other languages, such as C, awk, sed, sh, and BASIC, among others.

iii It gives a magnificent interface to about every accessible database, alongside a deliberation layer that permits you to switch databases without re-composing the majority of the code.

iv Re-usable code construction modeling (modules, object-oriented programming, and so on). Perl is architectured to permit and support re-utilization. The center square of re-utilization, the module, makes it simple to influence business rationale cross-wise over stages in web applications, clump scripts, and a wide range of coordination segments.

v CPAN, the Comprehensive Perl Archive Network, is one of the biggest archives of free code on the planet. On the off chance that you require a specific sort of usefulness, there are a few alternatives on the CPAN, and there are no expenses for utilizing them.

vi Perl can be utilized to create Web applications, clump transforming, information examination and content control, order line utilities and applications, GUI applications.

vii The Perl interpreter can be embedded into other systems.

Considering all the mentioned features of Perl really made us to use this in our project for capturing license usage information for all the applications in the system.

4.8.2 MySQL

MySQL is a relational database administration framework (RDBMS), and boats with no GUI apparatuses. To oversee information contained inside the information bases, MySQL is a famous decision of database for utilization in web applications, and is a focal part of the generally utilized LAMP open source web application programming stack (and other AMP stacks). LAMP is an acronym for **Linux, Apache, MySQL, Perl/PHP/Python**. MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages. MySQL is customizable.

4.8.3 Dynamic HTML

Dynamic HTML, or DHTML, is an umbrella term for a gathering of innovations utilized together to make intelligent and live web sites by utilizing a mix of a static mark-up dialect, for example, HTML, a customer side scripting dialect, for example, JavaScript, a presentation definition dialect, for example, CSS, and the Document Object Model. The use of DHTML was presented by Microsoft with the arrival of Internet Explorer 4 in 1997. DHTML permits scripting dialects to change variables in a site pages definition language, which thus influences the look and capacity of static HTML page content, after the page

```
mysql> desc vendorTable;
+---------------+-------------+------+-----+-------------------+-----------------------------+
| Field         | Type        | Null | Key | Default           | Extra                       |
+---------------+-------------+------+-----+-------------------+-----------------------------+
| vendorId      | int(20)     | NO   | PRI | NULL              | auto_increment              |
| vendorName    | varchar(45) | NO   |     | NULL              |                             |
| site          | varchar(45) | NO   |     | NULL              |                             |
| portNo        | int(20)     | NO   |     | NULL              |                             |
| server        | varchar(512)| NO   |     | NULL              |                             |
| vendorTimestamp | timestamp | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+---------------+-------------+------+-----+-------------------+-----------------------------+
6 rows in set (0.01 sec)
```

Figure 4.8: Vendor Table

```
mysql> desc featureTable;
+-----------------+------------+------+-----+-------------------+-----------------------------+
| Field           | Type       | Null | Key | Default           | Extra                       |
+-----------------+------------+------+-----+-------------------+-----------------------------+
| featureId       | int(20)    | NO   | PRI | NULL              | auto_increment              |
| featureName     | varchar(45)| NO   |     | NULL              |                             |
| featureTimestamp | timestamp | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-----------------+------------+------+-----+-------------------+-----------------------------+
3 rows in set (0.00 sec)
```

Figure 4.9: Feature Table

```
mysql> desc junctionTable;
+-----------+---------+------+-----+---------+----------------+
| Field     | Type    | Null | Key | Default | Extra          |
+-----------+---------+------+-----+---------+----------------+
| jId       | int(20) | NO   | PRI | NULL    | auto_increment |
| vendorId  | int(20) | NO   | MUL | NULL    |                |
| featureId | int(20) | NO   | MUL | NULL    |                |
+-----------+---------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Figure 4.10: Junction Table

has been completely stacked and amid the review process. Therefore the dynamic normal for DHTML is the way it capacities while a page is seen, not in its capacity to produce a novel page with every page load. Perl CGI is responsible for generating HTML web pages.

```
mysql> desc licenseUsageInfo;
+------------------+----------+------+-----+---------+----------------+
| Field            | Type     | Null | Key | Default | Extra          |
+------------------+----------+------+-----+---------+----------------+
| checkoutId       | int(20)  | NO   | PRI | NULL    | auto_increment |
| vendorId         | int(20)  | NO   | MUL | NULL    |                |
| featureId        | int(20)  | NO   | MUL | NULL    |                |
| licenseIssued    | int(20)  | NO   |     | NULL    |                |
| licenseInuse     | int(20)  | YES  |     | NULL    |                |
| licenseReserved  | int(20)  | YES  |     | NULL    |                |
| checkoutTime     | datetime | NO   |     | NULL    |                |
+------------------+----------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Figure 4.11: License Usage Info Table

4.9 Use Case for the Waiting Lounge and Wait-Time Prediction

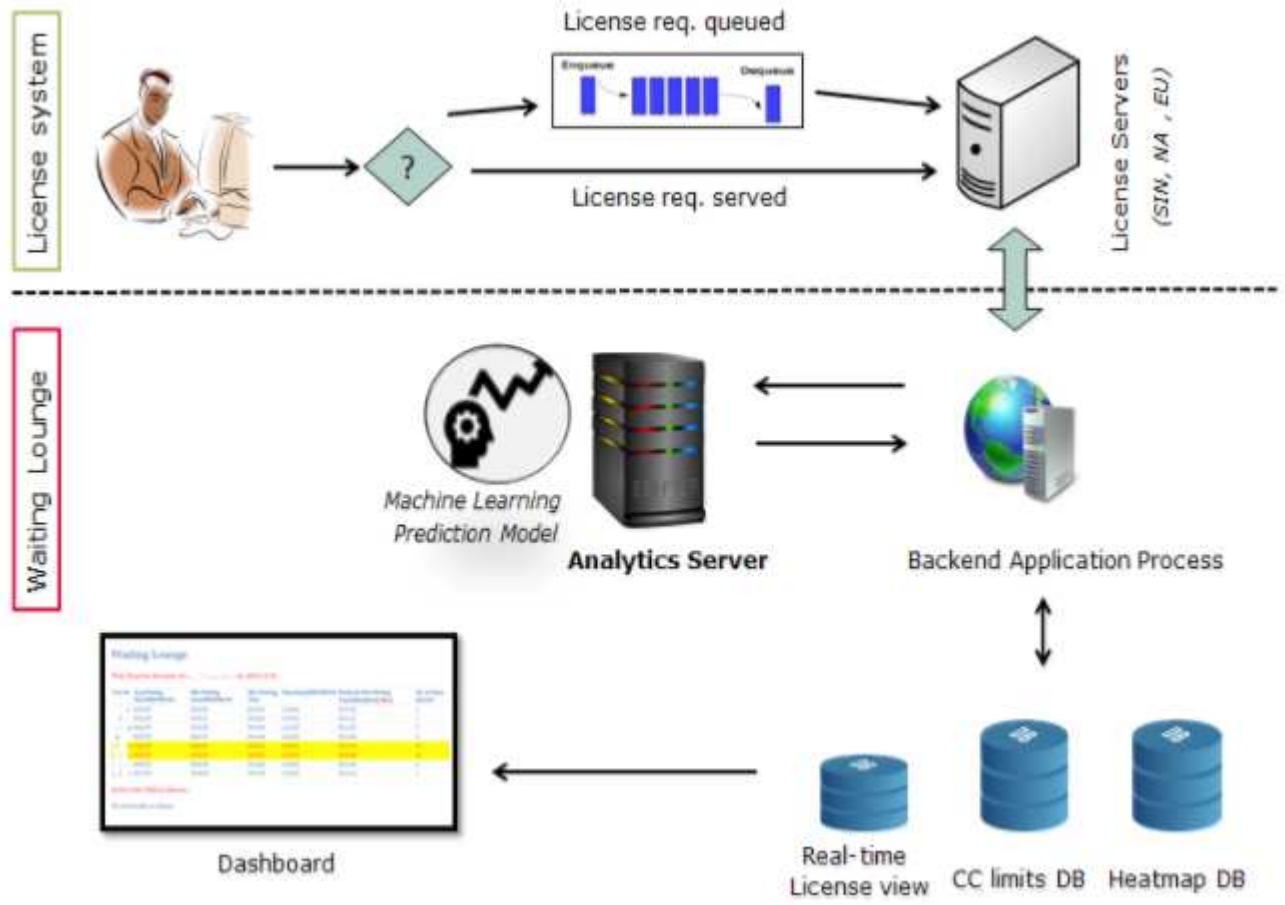

Figure 4.12: License Wait Time Prediction

As shown in figure. 4.12, implementation of Waiting Lounge application started with the data capturing part from license system. The user requests for the licenses and if the

licenses are not available, his requests are queued. All the license check-in, check-out and queued information is updated in the license statistics file called "lmstat".We capture the queued information on per 2 minute basis from lmstat, and store in the database. The data is then processed and given to the prediction model, for predicting the waiting time for the queued users. In this process, we are using R as the technology; the data is given to the XG-Boost model with a number of parameters. And the wait time is then predicted for the users.

4.10 XGBoost- eXtended Gradient Boosting Decision Tree Algorithm

This algorithm goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines.

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the AdaBoost algorithm that weights data points that are hard to predict.

Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

This approach supports both regression and classification predictive modeling problems.

Table 4.1: Parameters Given to XGBoost

Column Name	**Description**
UserId	User Ids
Feature	Tool name of a vendor
Host	Machine on which the process is running
JobId	Job Id given to the process
Version	License Version
Server	Server address (Link)
Port	Port no. for the server
ProcessId	Process Id
Event	License Used/License Queued
StartTime	Start time of the checkout
EndTime	End time of the checkout
LastUpdated	Time stamp of the last update in the database
LicenseCount	Count of checked out licenses
PredictedMod1	Model 1 Result
Vendor	Vendor Name

PredictedMod2	Model 2 Result
PredictedMod3	Model 3 Result
PredictedMod4	Model 4 Result
duration	Checkout Duration in seconds
hour	Hour of the checkout time
day	Day of the checkout time
COSTCENTER	Costcenter of the user
minLicAvail	Minimum License available for a particular tool
maxLicAvail	Maximum license available for a particular tool
startDate	Start Date of the checkout
p15StartTime	Start time of maximum checkout in last 15 mins
p30StartTime	Start time of maximum checkout in last 30 mins
countServiceDeadP15	Count of checkouts which are finished in last 15 mins
countServiceAliveP15	Count of checkouts which are running in last 15 mins
countServiceArrivalP15	Count of checkouts arrived in last 15 mins
meanServiceDurationDeadP15	Mean execution time of the finished checkouts of last 15 mins
countServiceDeadP30	Count of checkouts which are finished in last 30 mins
countServiceAliveP30	Count of checkouts which are running in last 30 mins
countServiceArrivalP30	Count of checkouts arrived in last 30 mins
meanServiceDurationDeadP30	Mean execution time of the finished checkouts of last 30 mins
countQueuedDeadP15	Count of queues which are finished (without getting license) in last 15 mins
countQueuedAliveP15	Count of queues in last 15 mins
countQueuedArrivalP15	Count of queues arrived in last 15 mins
meanQueuedDurationDeadP15	Mean waiting time of the queues finished in last 15 mins
countQueuedDeadP30	Count of queues which are finished (without getting license) in last 30 mins
countQueuedAliveP30	Count of queues in last 30 mins
countQueuedArrivalP30	Count of queues arrived in last 30 mins
meanQueuedDurationDeadP30	Mean waiting time of the queues finished in last 30 mins
meanServiceInterArrivalTimeP15	Mean duration between the jobs arrival in last 15 mins. Suppose the jobs are coming at 1st, 4th and 6th minute, then the value will be (3+2/3)
meanServiceInterArrivalTimeP30	Mean duration between the jobs arrival in last 30 mins.

meanQInterArrivalTimeP15	Mean duration between the queues arrival in last 15 mins.
meanQInterArrivalTimeP30	Mean duration between the queues arrival in last 30 mins.

Chapter 5

System Testing and Results

Testing is a methodology of executing a system to guarantee that characterized info will deliver real results that concur with obliged yields. In addition to a product venture, mistake can be launched at any stage amid the advancement. For every period of the product advancement cycle there are diverse systems for recognizing and disposal mistakes that begin in that stage. However a few mistakes will reflect in the code. Testing performs an extremely significant part for quality certification and for guaranteeing the reliabilities of the product. The nature of the framework relies on upon its outline, improvement, testing and execution. Shortcomings in any of these territories will genuinely influence the quality and accordingly estimation of the framework to its clients. When the code has been produced, trying of the modules starts usage closes with formal tests.

5.1 Test Environment

The system software was tested on the following platform.

Hardware

- Server on Unix machine and client on Windows machine as well as UNIX machine

Software

- Operating system Windows XP/UNIX
- Dedicated R-server
- Perl CGI
- HTML/CSS, MYSQL DB, Javascript
- Internet Explorer - 8, Google Chrome, Firefox

5.2 Automated Tools Testing

5.2.1 Unit Testing

Unit testing centres check on the littlest unit of programming outline, the product segment or module. Utilizing the segment level outline portrayal as an aide, critical control ways are tried to reveal slips inside the limit of the module. The unit testing is a white box

situated testing. Most importantly the module interface is tried to guarantee that the data legitimately streams into furthermore, out of the project until under test. At that point the neighbourhood information structure is tried to guarantee the information put away briefly keeps up its uprightness amid all ventures in an execution. Limit conditions are tried to guarantee that the module works appropriately at limits built to breaking point or limit handling. Every single autonomous way through the control structure are worked out to guarantee that all announcements in a module have been executed in any event once. Lastly, all blunders taking care of ways are tried. In this venture the testing is done by up approach. Beginning with littlest and least level modules and transforming each one in turn. For every module a driver and comparing stubs were likewise composed. On the off chance that any blunders discovered they were remedied promptly and the unit was tried once more.

5.2.2 Integration Testing

Integration testing is a coherent augmentation of unit testing. In its least difficult structure, two units that have as of now been tried are joined into a segment and the interface between them is tried. A segment, in this sense, alludes to a coordinated total of more than one unit. The thought is to test blends of pieces and in the long run grow the methodology to test your modules with those of different gatherings. In the long run all the modules making up a procedure are tried together. Any slips found when joining units are likely identified with the interface between units. This technique lessens the quantity of potential outcomes to a far more straightforward level of investigation. In this product, the base up coordination testing drew nearer has been utilized, beginning with the littlest and most minimal level modules and continuing each one in turn. For every module the tests were directed and the outcomes were noted down.

5.2.3 System Testing

System Testing of software or hardware is a testing conducted on a complete integrated system to evaluate the systems compliance with its specified requirements. System Testing falls under the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. Since the Equipment Manager is still under development to incorporate more services and features for completeness, system testing is not carried out.

Table 5.1: Unit Testing

Description	**Expected Result**
Test for application window properties	All the properties of the windows must be properly aligned and displayed

Test for mouse operations	All the mouse operations like click, drag, etc. must perform the necessary operations Without any exceptions.

Table 5.2: Test Cases

Test case ID	**Test case description**	**I/P Given**	**Expected O/P**	**Actual O/P**	**Test status P/F**
TC01	Testing the authentication process for the tool	Given Inputs are username and password	LDAP authentication should happen and user will be able to login to tool	User has accessed the tool with LDAP Authentication	Pass
TC02	Selecting the various drop down options	Selected option vendor name and features name and month	Tool should generate heatmap web page	Heatmap page has been generated	Pass
TC03	Testing if we do not select feature itself	Select only vendor and month	JavaScript pop up should open to say please select required feature	JavaScript POP up box generated	Pass
TC04	Testing the LOGFILE generation in a hierarchical structure	Perl script automatically captures the data	LOGFILE will be written by Perl script with valid information	LOGFILE generated as required	Pass
TC05	Testing the moving of the LOGFILE to the MySQL DB	Perl script to push data form LOGFILE to DB	MySQL DB should get updated with correct information	MySQL DB is getting updated with correct information	Pass

Table 5.3: Performance Testing

Description	Expected Result
This is required to assure that an application perforce adequately, having the capability to handle many peers, delivering its results in expected time and using an acceptable level of resource and it is an aspect of Operational management.	Should handle large input values, and produce accurate result in an Expected time.

Table 5.4: Security Testing

Description	Expected Result
Checking that the user identification is Authenticated.	In case failure it should have access to the tool features.
Check whether all data captured is correct or not?	Printed data is all correct and matching with log data

5.3 Automation Results

5.3.1 GUI for the System

Let us first look at the graphical user interface which has been developed for monitoring Real-time EDA License Usage.

i The official intranet link to GUI of the license monitoring system at Infineon Technologies is https://heatmap.vih.infineon.com/cgi-bin/heatmap.php. This is an internally hosted link in the organization. Whenever, we open the web link via any browser it pops up an authentication box which asks for user's identification and password for authorizing the user.

ii Once the user is authenticated, it will redirect to the Heatmap page which has multiple links in the navigation bar for other applications as Vendor Utilization, Quarterly Heatmap, Treemaps, Invoices, Usage Reports and other applications. User can easily navigate to other pages as well as per his role (Administrator or End-User).

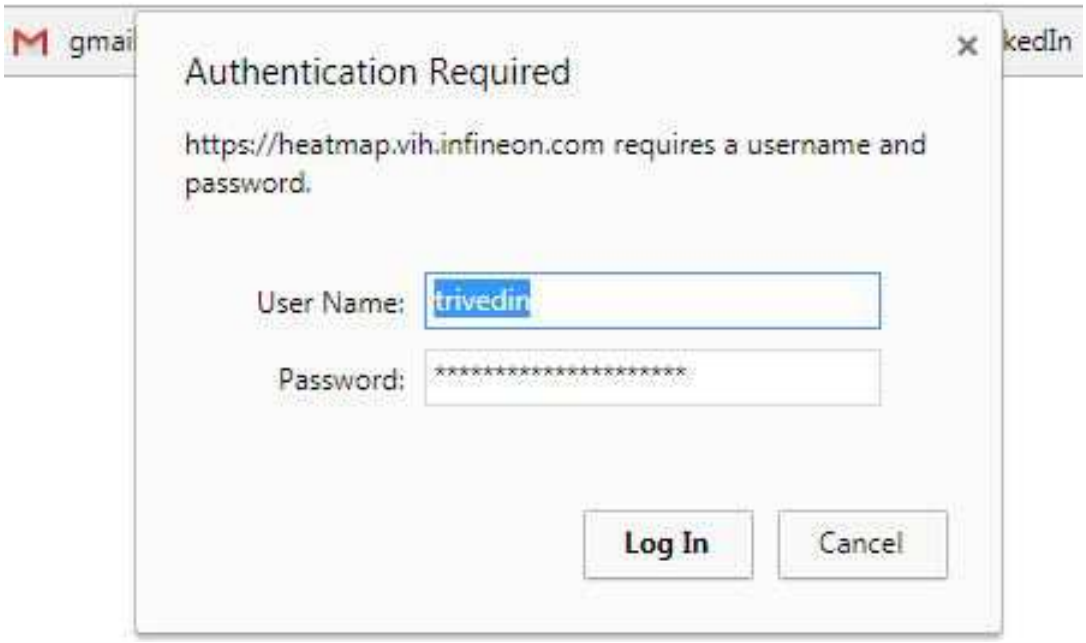

Figure 5.1: Login Authentication for the System

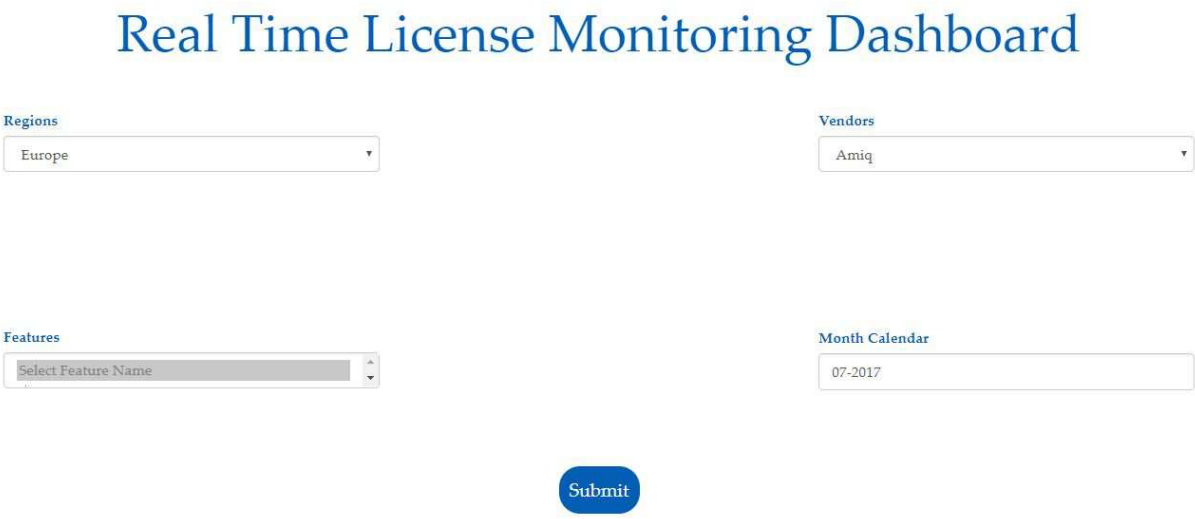

Figure 5.2: Heatmap GUI

iii The first view of the GUI is shown in figure 5.2, where user needs to select the Region first. Once the region is selected , according to the region, vendor list will be displayed and user selects the vendor from the displayed vendor list as in figure 5.3.

iv After selecting the vendor, respective features list of the selected vendor is generated and user will choose any one of the listed feature, as shown in figure 5.4.

v Hourly peak of license usage is represented in the form of a Heatmap. It is the view of day wise hourly based license usage for any given vendor/feature combination.

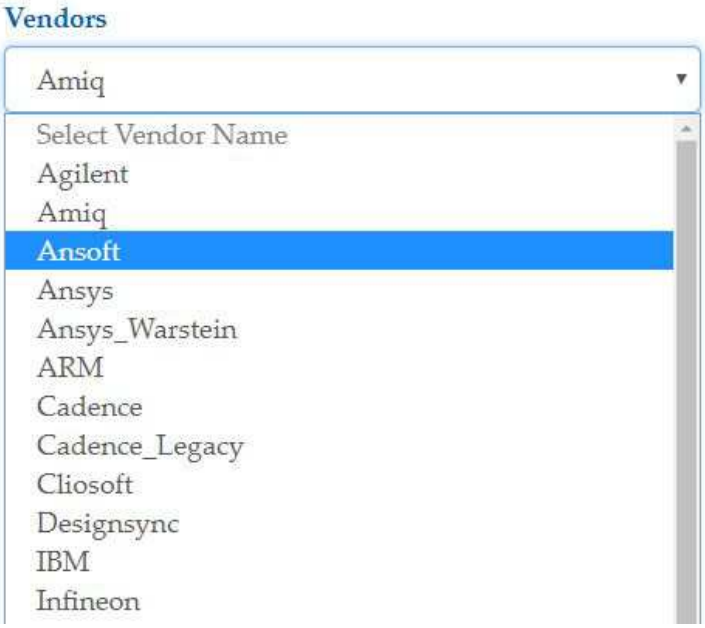

Figure 5.3: Selecting Options

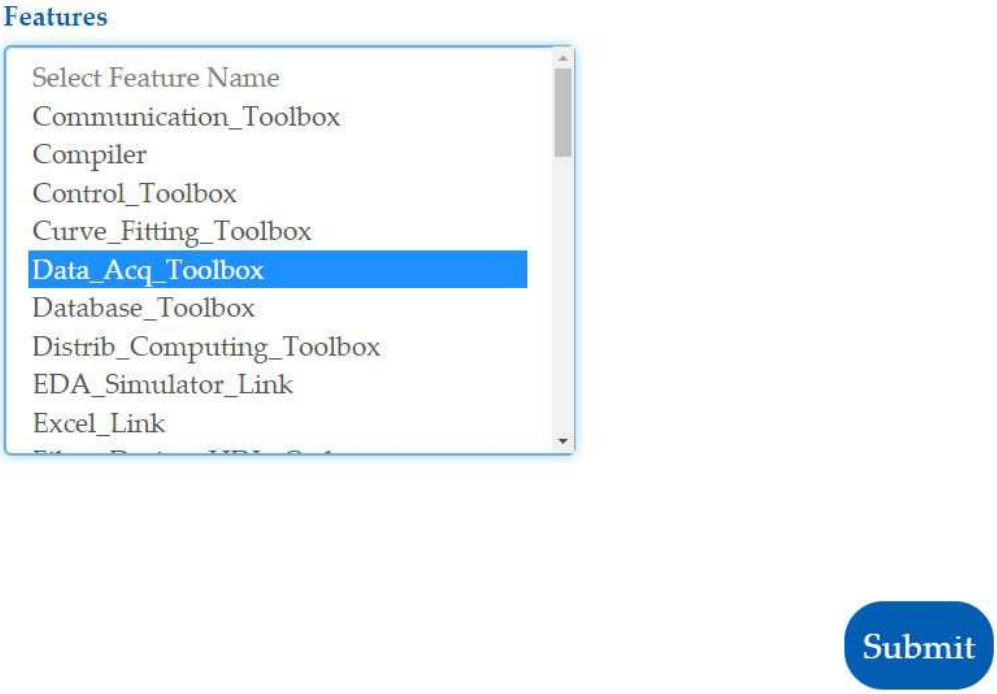

Figure 5.4: Selecting a feature

This is as shown in figure 5.5 .

vi In figure 5.6, you can see the License View. This application displays the license checkouts for an individual user for all the packages over different servers with all the necessary information like user's identity, server, execution host, start time, checkout time, etc. This helps the user to be informed of his own checkouts, and if

Time Zone: CET
Current Time: Wednesday 2017-07-26 13:20:10

Heatmap of [redacted]

legend 0 13 **Max Peak for the Month** **Change in Available number of Licenses**

Date/Time	Day	Installed/Available	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
01-07-2017	Saturday	98/13	10	10	9	6	7	7	7	8	9	8	9	8	8	8	6	6	4	7	7	6	8	9	6	5
02-07-2017	Sunday	98/13	7	8	5	4	3	5	6	6	6	6	6	6	6	4	4	2	4	3	4	4	5	3	4	4
03-07-2017	Monday	98/13	4	5	3	3	3	4	4	3	3	5	7	9	11	12	12	9	9	8	7	8	8	7	5	8
04-07-2017	Tuesday	98/13	7	9	10	9	8	7	8	6	8	9	10	11	11	10	13	13	12	13	12	11	11	13	12	10
05-07-2017	Wednesday	98/13	12	12	8	7	6	5	6	4	8	9	8	11	13	13	13	13	13	13	13	13	13	13	12	13
06-07-2017	Thursday	98/13	13	12	11	12	10	11	13	12	11	10	11	11	13	13	13	13	12	13	13	13	13	13	12	9
07-07-2017	Friday	98/13	9	9	10	9	8	9	10	12	11	11	11	11	13	13	7	7	7	7	9	9	9	9	9	9
08-07-2017	Saturday	98/13	3	3	3	4	3	3	3	2	7	4	5	6	6	7	6	6	5	5	7	7	8	8	8	8
09-07-2017	Sunday	98/13	7	8	9	9	8	8	8	9	9	8	7	6	4	6	7	6	6	6	6	5	6	6	6	4
10-07-2017	Monday	98/13	4	3	4	4	4	3	3	3	5	9	10	9	10	9	7	6	6	5	6	7	10	11	10	12
11-07-2017	Tuesday	98/13	12	11	9	10	7	7	8	13	12	13	12	7	10	12	10	7	7	7	8	9	12	11	10	12
12-07-2017	Wednesday	98/13	11	12	9	11	10	10	7	6	8	8	9	10	9	9	7	9	9	10	10	11	11	10	10	11
13-07-2017	Thursday	98/13	10	10	7	8	9	12	11	9	10	11	11	11	8	9	10	10	11	10	11	7	6	7	7	8
14-07-2017	Friday	98/13	6	4	5	6	7	7	6	6	5	7	7	8	10	8	10	13	12	10	8	8	9	8	7	7
15-07-2017	Saturday	98/13	7	8	8	7	7	6	6	6	6	8	7	7	7	6	6	6	8	7	7	8	8	8	7	7
16-07-2017	Sunday	98/13	6	7	6	6	7	6	6	5	6	6	5	5	5	5	5	5	5	6	5	4	5	5	5	5
17-07-2017	Monday	98/13	5	4	3	2	3	3	3	4	5	7	7	8	8	8	9	12	11	9	10	11	11	11	11	9

Figure 5.5: Output of Heatmap

he is not using any of his checked-out license, he can view that in this application and can release the zombie licenses. Also, this process will help the users to get some of the released licenses in requirement.

Licenses checked-out

License Server:		vihlal01.vih.infineon.com	Region:	Europe
User Id	Feature Name	Execution Host	Start Time	Checkout Duration
chjacobs	TLSTOK	[redacted]	Tue 7/25 12:33	0 days 00 hours 06 minutes 11 seconds
chjacobs	TLSTOK	[redacted]	Tue 7/25 12:13	0 days 00 hours 26 minutes 11 seconds
License Server:		**ulicserv2.vih.infineon.com**	**Region:**	**Europe**
User Id	**Feature Name**	**Execution Host**	**Start Time**	**Checkout Duration**
chjacobs	DC-Expert	[redacted]	Tue 7/25 12:15	0 days 00 hours 24 minutes 13 seconds
chjacobs	DC-Expert	[redacted]	Tue 7/25 12:29	0 days 00 hours 10 minutes 13 seconds
chjacobs	DC-Extension	[redacted]	Tue 7/25 12:16	0 days 00 hours 23 minutes 13 seconds

Figure 5.6: License View

vii Next utility is the Watchdog, this application helps the administrators in dealing with the long and the hung checkouts. The checkout time for a license is considered to be long, if it exceeds the set average checkout time for a particular tool. Daily, a mail is triggered to the GLM Team with user's identity, package name, server and region, checkout duration on comparing the current time duration of the checkout with the set package-wise average checkout time.

5.4 Analysis of Queues Formation

5.4.1 Experiment Results for Method I - Queuing Theory

License checkout data for March 2017 of site1 for Vendor1's feature1 from was extracted LiSA database. For March, the number of available licenses varied between 18 and 20. The average arrival time for license requests was 3.75 minutes and average service time was 59.06 minutes.

5.4.1.1 Analysis of March 2017 M/M/C results

Table 5.5: March 2017 M/M/C results

Number of licenses available	Arrival Rate	Service Rate	Average				
			Number of Requests		Waiting Time In		Utilization of licenses
			System	Waiting in Queue	System	Queue	
18	0.26667	0.01693	19.14	3.39151764	71.78	12.72	87.0%
19	0.26667	0.01693	17.39	1.63594654	65.19	6.13	83.0%
20	0.26667	0.01693	16.60	0.84896559	62.24	3.18	79.0%

With the help of the model we can calculate the waiting times, number of requests and utilization for different values of available licenses. In March, the number of licenses for a particular tool was increased from 18 to 20 during the course of the month. Each row shown in Table 5.5 shows the model results if the license number was set to a particular value for the entire month. This helps in critically analysing different scenarios for different conditions and designing the license configurations.

For example, though the license number was increased for short-term demands, utilization would not have reached its maximum (with a utilization of 87.0%) with the initial configuration of 18 licenses, which could have been maintained for the entire month. When the license number was increased to 20, the utilization goes down to 79.0%, leaving a lot of idle capacity.

Since we have a mathematical model, we can also calculate the probabilities of the events like waiting time in queue or total time in system (system time includes the time for servicing and waiting time in queue), this is helpful in getting better details, much more

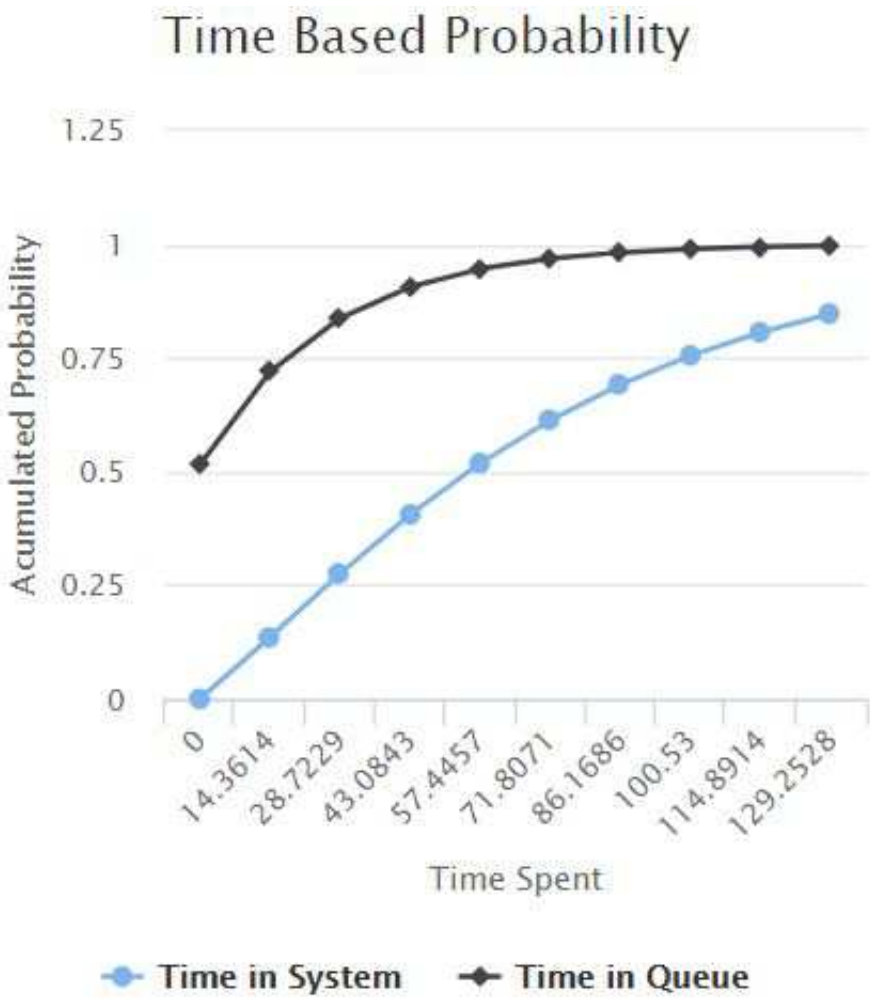

Figure 5.7: March 2017 Time based probability

than what the average numbers say. For example in March, with 20 licenses the probability of waiting time in queue being 24 minutes was 96.0%. A time based probability graph drawn for time spent in queue and system is shown in the figure below Figure 5.7.

5.4.1.2 Analysis March 14th M/M/C Results

Though the utilization of the licenses was not at its maximum at 87.0%, the licences were still increased twice because of short term demands on 14th and 15th March. The reason for the increase was that there were many queues in the license servers than normal. We tried out the queuing model for all the 24 hours individually for 14th and 15th March.

Queuing theory results for March 14th is shown in Table 5.6. For this period, average arrival time was 3.27 min and average service time was 60.73 min. With 18 licenses, the model fails for March 14th as the demand exceeded the capacity. This is a common behaviour as, on some days maximum EDA users work in parallel and submit multiple requests in parallel, causing sudden rise in arrival rate. Because the arrival rate is greater

than the servicing capacity of the server, the model fails. As the demand was great, it was essential to increase the license number to 19 to finish the jobs within the same day.

Table 5.6: March 14th, 2017 M/M/C results

Number of licenses Available	Arrival Rate	Service Rate	Average				
			Number of Requests		Waiting Time In		Utilization of licenses
			System	Waiting in Queue	System	Queue	
18	0.30057	0.01647	-	-	-	-	-
19	0.30057	0.01647	38.05	19.79	126.58	65.85	96.0%
20	0.30057	0.01647	24.51	6.26	81.56	20.83	91.0%

5.4.1.3 Analysis March 15th M/M/C Results

For the period of March 15th, the arrivals were observed to be more frequent with an average arrival time of 1.71 min, but at the same time service time reduced drastically to 30.11 minutes. This is the reason as shown in Table 5.7, with 18, 19 the model works, as the demand is within the capacity. Unlike March 14th, the actual increase in license could have been avoided on March 15th, by maintaining the license at 19.

Table 5.7: March 15th, 2017 M/M/C results

Number of licenses Available	Arrival Rate	Service Rate	Average				
			Number of Requests		Waiting Time In		Utilization of licenses
			System	Waiting in Queue	System	Queue	
18	0.5848	0.03321	57.80	40.19	98.84	68.73	98.0%
19	0.5848	0.03321	26.00	8.39	44.46	14.35	93.0%
20	0.5848	0.03321	21.15	3.54	36.17	6.06	88.0%

When the entire month of March was analysed as a whole, it did not give the true picture as to why the license numbers were increased during the month. But when March 14th and 15th were analysed individually, the real license demand was captured. So to capture further detail on hour wise demand, we analysed individual job for each of the 24 hours for these days. However as the time frame was small, the arrival rate and service rate, did not really fit in the required exponential distribution, calculations became complex and results were unreliable. This is primarily because of the non-representative sudden increase in the number of arrival rate. It is important to use representative arrival rates and service rates even when trying to predict numbers. So with the help of queuing theory we can get atleast a days level of detail for the waiting times and utilization, and it is recommended to use days data for analysing peak demand. However it is also a question of how much demand is allowed for a particular day. Since there are idle times in the system as found out through the months analysis, individual days license demand

though important one might choose to ignore if its an abnormal spike and believe that the demand would be dissolved the subsequent day.

5.4.2 Experiment Results for Method II- Average Based

We tried an alternate method for prediction of wait times for license availability at any given time using descriptive statistics and exploratory data analysis. The same feature1 usage dataset from LiSA for the month of March-2017 was used.

5.4.2.1 Understanding distribution of runtimes for month of March-2017

With the help of the available historical data of previous checkout durations at any given point in time we obtain a frequency distribution which explains the categories inside which majority of checkout durations were completed. Figure 5.8 represents the percentage of checkouts completed within different intervals of time during the month of March, 2017.

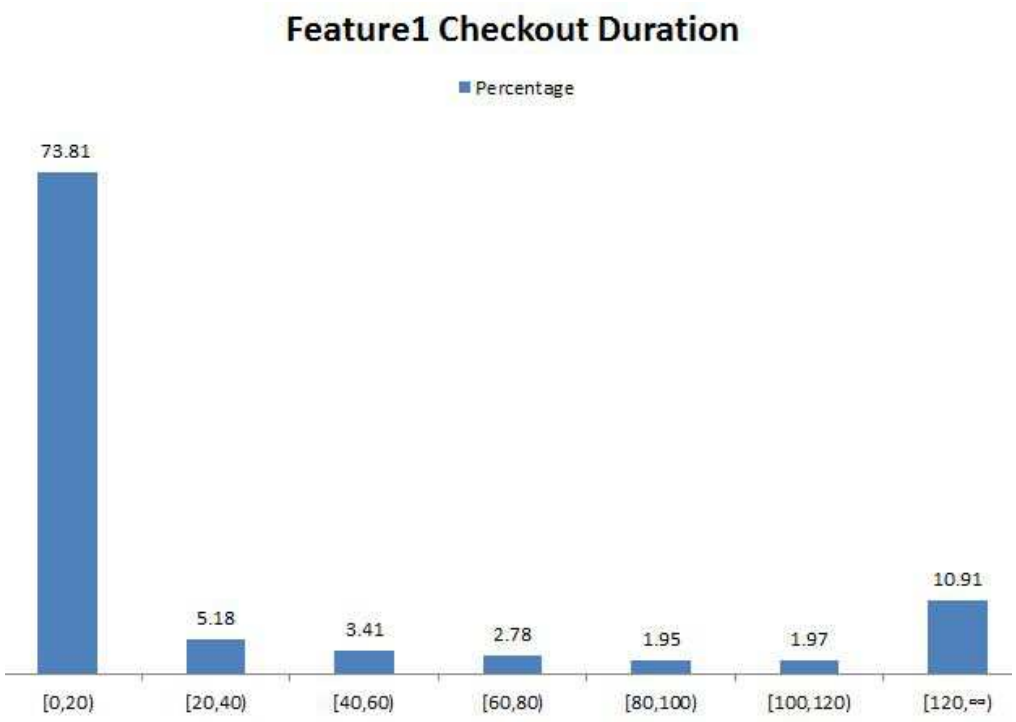

Figure 5.8: March 2017 Feature1 Checkouts

5.4.2.2 Fix initial conditions for month of March-2017

Wait time predictions are mostly useful in situations where the installed number of licenses does not meet the demand. In such situations, multiple queues are formed and user requests get delayed. In order to solve the problem of wait time prediction for such

crucial situations, of the entire March data we need to select data which would be representative of the peak/busy days for our analysis. We will now select a busy day of March which will best represent such situations. Looking at the heat map below figure 5.9 we selected March 15th, 2017, since there were continuous peak hours with maximum license usage.

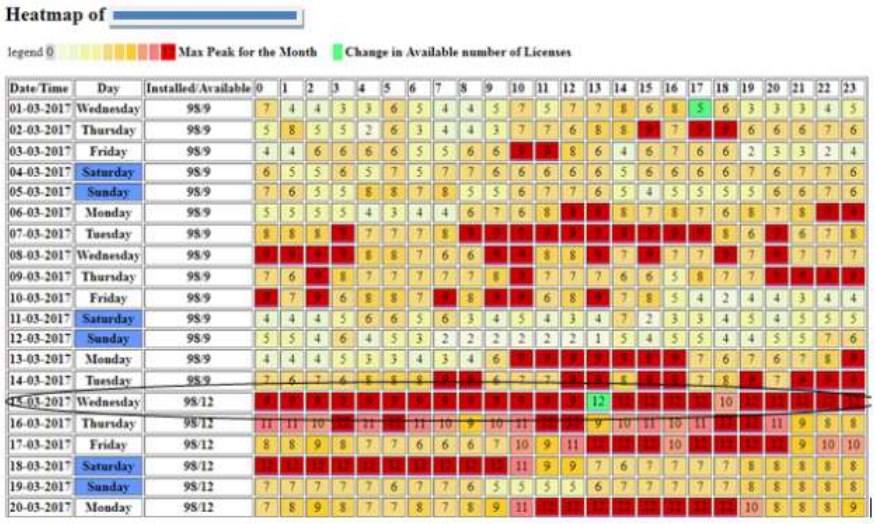

Figure 5.9: March 2017 Feature1 Heatmap

Based on our selection of the representative data (a busy day March 15th), we calculate the distribution of runtimes on that particular day as shown in Table 5.8.

Table 5.8: March 15th 2017 Runtime distribution

Interval	**Frequency**	**Percentage**
[0, 10)	343	72.3
[10, 20)	18	3.8
[20, 30)	5	1.1
[30, 40)	9	1.9
[40, 50)	18	3.8
[50, 60)	3	0.6
[60,∞)	78	16.5

From the above distribution for the run times on March 15th, 2017, it is evident that:

i 76.0% of the license requests were completed within 20 minutes.

ii 24.0% of the license requests were completed after 20 minutes.

It is important to note that 76.0% of the license requests are invoked by short runners (requests which use a license within 20 minutes). This is in line with the Pareto principle (also known as 80-20 rule) which states that for many events; roughly 80.0% of the effects come from 20.0% of the causes. Since, the top 20.0% of the license requests i.e., requests which use a license within 20 minutes have around 76.0% impact on the overall pattern of license requests, we concentrate more on these requests. We set these findings as a base to predict wait times.

Arrived base conditions

i The percentage split between short runners and long runners is 76, 24 at any point in time.

ii The standard duration for short runners is less than or equal to 20 minutes.

5.4.2.3 Validation on month March 2017

We test the above two base conditions if these can be generalized and applied to any situation by validating them against:

5.4.2.3.1 Validation using data from a complete month March 2017 Table 5.9 explains the percentage split between short and long user checkouts during the month of March, 2017, and Table 5.10 explains the wait time statistics.

Table 5.9: Runtime Statistics for complete month March 2017

Run time parameters	**Results**
Total no of checkouts	6034
Total no of checkouts less than 20 minutes	4277
Percentage of checkouts less than 20 minutes	71.0%
Split between short runners to long runners for the whole month	71, 29

Table 5.10: Wait time Statistics for complete month March 2017

Wait time parameters	**Results**
Total no of queued checkouts	125
No of wait times less than 20 minutes	123
Percentage of wait times less than 20 minutes	98.4%
Average of all the wait times for the complete month	3.47 minutes

5.4.2.3.2 Validation using data at a random point on 16th March, 2017 First random validation point: 16th March, 2017 at 8:45 A.M. Let us see if the base conditions are true when data from the start of the month till the random point 8:45 A.M on 16th March 2017 is taken. Table 5.11 for Runtime Statistics and Table 5.12 for Wait Time Statistics, explain the percentage split between short and long user checkouts before 8:45 A.M on 16th March, 2017.

Table 5.11: Random runtime Statistics on 16th March, 2017

Parameters(related to run times	**Results**
Percentage of checkouts less than 20 minutes	69.2%
Ratio between short runners to long runners for the whole month	69 : 31

Table 5.12: Random runtime Statistics on 16th March, 2017

Parameters(related to wait times	**Results**
Total no of queued requests before 8:45 A.M	4
No of queued requests below 20 minutes	4
Percentage of wait times less than 20 minutes	100.0%

Average of all the wait times for the complete day	2.40minutes
Note: For the whole day, only one request was above 20 minutes, and it had lasted for 20.98 minutes	

i The wait time on an average for the whole month of March has been 3.40 minutes, except for two instances which had wait times of 20.98 minutes 27.85 minutes.

ii The maximum predicted wait time would be 20 minutes, since both the base conditions were more or less met when we validated them against the whole month and at a random point. (i.e. the ratio between short runners which were completed within 20 minutes) to long runners for the whole month revolved around 70:30 and the average of wait times at both the validation points were less than 20 minutes.)

iii Hence, the wait time for any user request can be predicted at any point in a day, (based on the behaviour of checkouts before the considered point in the same day). Also, the ratio of short runners Vs long runners can be assumed to be 70:30. However, it could vary at different instances.

Hence, we could predict the wait time for any user request, if licenses are not free at any given point of time, based on the above exploratory and descriptive analysis of the past data.

5.5 Waiting Lounge and Prediction Accuracy

Waiting Lounge is for tracking queued license users of different tools was initially started with Vendor1 and Vendor2. It shows average, minimum and maximum waiting time for individual users of particular tool on feature basis along with the predicted wait times for the queued licenses. Sometimes when the license limit is reached, some licenses are queued. We are also capturing the queued license information for predicting the wait-times.

We have used 26 parameters, which are prepared based on the analysis described in section 5.4. These parameters are passed to the XGBoost and wait-times are predicted for the queued license requests. This forecasting helps the user to have an estimate of how long he has to wait for a particular license.

We have done the experiment using the data for Vendor1. We then experimented by increasing the dataset where the attributes of Vendor1 and Vendor2 were included. Finally, Vendor3 was also considered in the forecasting experiment.

For the first experiment, when Vendor1 was included, we achieved an accuracy of 80.0%. In second situation, when the data for Vendor1 and Vendor2 was included, the accuracy was nearly same as that was in the case of Vendor1.

However, when we considered Vendor3 in the scenario, the accuracy was dropped to 65.0%. This motivated us to consider the forecasting model in which the three vendors are considered separately. Also we have altered the database schema as per each vendor. Initially, there were only two tables (Main Table and Archive Table) in database for all the three vendors. With consideration of Vendor3, we have separate tables corresponding to each vendor for archiving data.

This increased the accuracy for the three situations to 92.0%. On analysing the accuracy dropdown after considering the dataset of Vendor3, it was concluded that the format of license statistics file of each vendor is different, which is resulting in irregular and incorrect data values in the tables, affecting the training of the model. Hence, we started archiving the data in different tables for each vendor. Presently, we are running 4 models with XGBoost method and giving different parameters to them for testing with different training-testing ratio. The best result is coming from 80:20 for training and testing. Figure 5.10 shows the accuracy variations, in percentage, over the time period for the model using classification method.

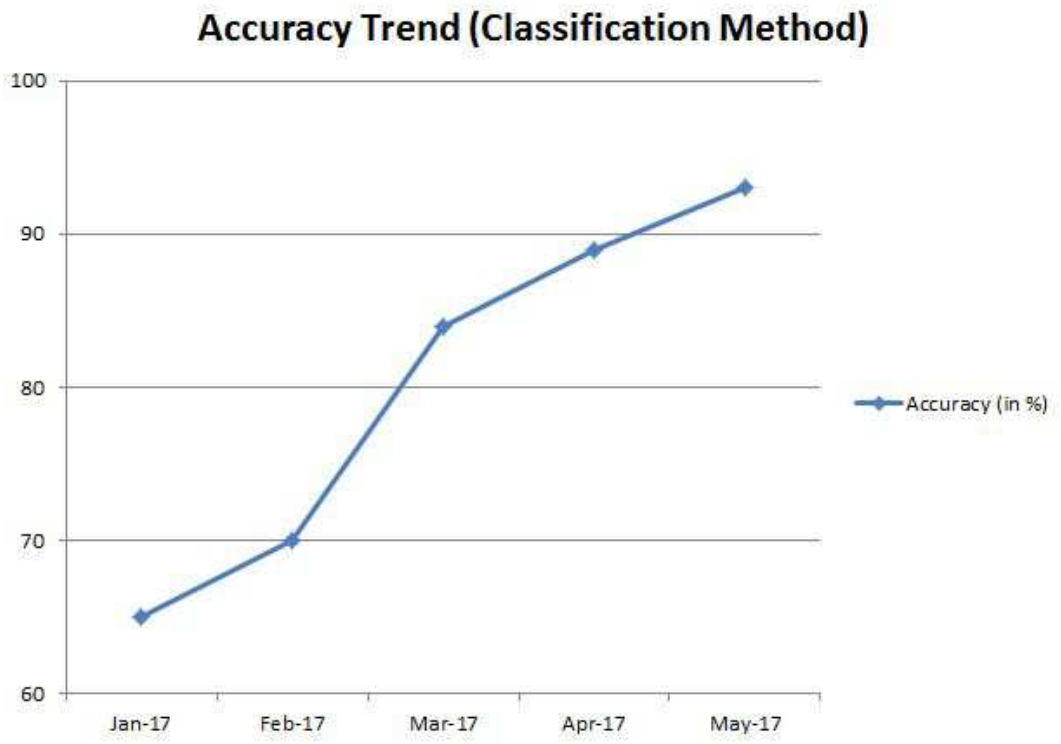

Figure 5.10: Accuracy Trend for Wait-Time Prediction (Classification)

For wait-times prediction, we have used Perl for data archiving, XGBoost for forecasting the wait-times and PHP is used for displaying information on the web-page. Figure 5.11 is a snapshot of a larger scenario for a tool of Vendor1, which shows the actual wait-time of a queued user, calculated with the values of start time of the queue and the end time of the queue. The column-Predicted Wait-Time(Beta) displayed in the same snapshot shows the end output of the prediction model, which gives a range of minutes to the user to have an estimate of the time he has to stay in the queue.

Min Waiting Time(HH:MM:SS)	Max Waiting Time	Timestamp(HH:MM:SS)	Predicted Wait-Time(Beta)
00:47:59	00:47:59	19:33:02	10-20 min
00:14:00	00:14:00	18:11:02	10-20 min
00:10:00	00:10:00	17:57:02	5-10 min
00:10:01	00:10:01	15:47:01	5-10 min
00:33:59	00:33:59	12:55:02	10-20 min
00:06:00	00:06:00	12:51:02	0-5 min
00:06:00	00:30:00	12:51:02	20-30 min
00:01:59	00:04:00	12:47:02	0-5 min
00:02:00	00:36:00	12:57:02	10-20 min
00:06:01	00:06:01	12:29:01	0-5 min
00:04:00	00:04:00	11:51:02	0-5 min
00:01:59	00:30:00	12:53:02	20-30 min
00:02:00	00:32:00	12:55:02	10-20 min
00:02:00	00:05:59	10:21:02	0-5 min
00:02:00	00:02:02	10:07:02	0-5 min
00:01:59	00:01:59	10:05:03	0-5 min

Figure 5.11: Predicted Wait-Times Snapshot (Classification)

We have started this forecasting experiment with the regression method. Recently, we considered the classification method also for predicting the wait-times thinking of giving a better and easier to understand output to the end-user. Figure 5.12 shows the accuracy variations, in percentage, over the time period for the model using regression method.

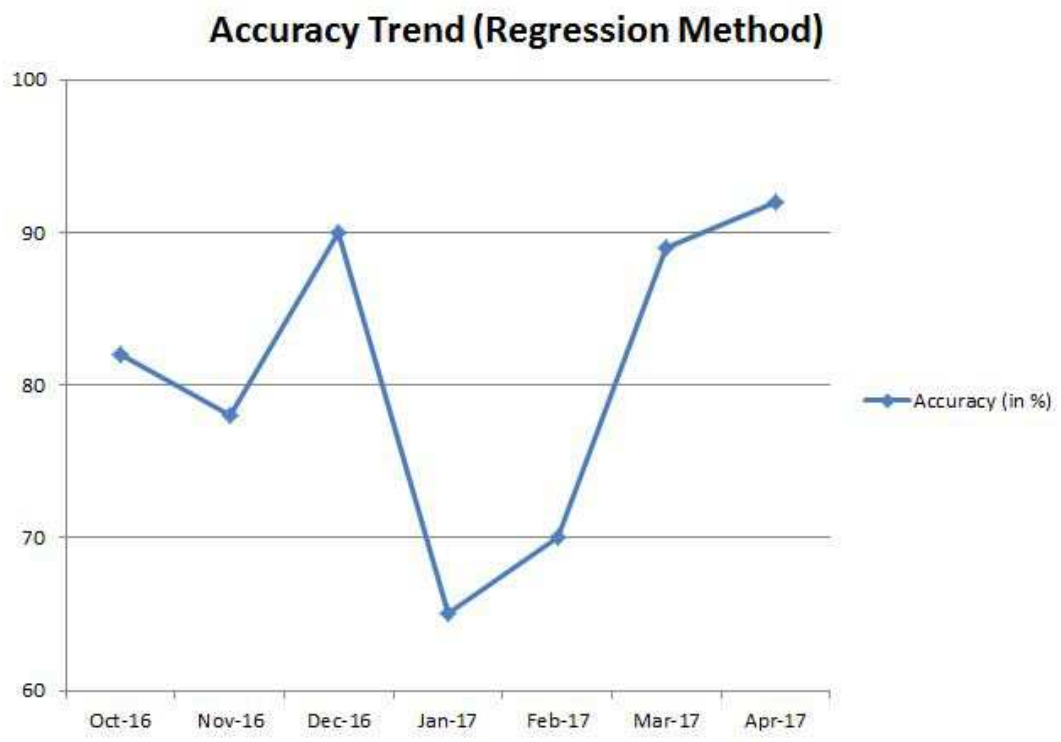

Figure 5.12: Accuracy Trend for Wait-Time Prediction (Regression)

Figure 5.13 shows the end output of the prediction model which is using regression method for the wait-times forecasting. The column-Predicted Wait-Time(Beta) displayed in the same snapshot depicts the predicted wait time for an individual user in a fixed value (in minutes), which is in queue for a tool license of Vendor1.

Min Waiting Time(HH:MM:SS)	Max Waiting Time	Timestamp(HH:MM:SS)	Predicted Wait-Time(Beta)
00:47:59	00:47:59	19:33:02	00:04:35
00:14:00	00:14:00	18:11:02	00:29:13
00:10:00	00:10:00	17:57:02	00:09:43
00:10:01	00:10:01	15:47:01	00:07:05
00:33:59	00:33:59	12:55:02	00:13:51
00:06:00	00:06:00	12:51:02	00:19:45
00:06:00	00:30:00	12:51:02	00:19:45
00:01:59	00:04:00	12:47:02	00:05:59
00:02:00	00:36:00	12:57:02	00:10:24
00:06:01	00:06:01	12:29:01	00:31:51
00:04:00	00:04:00	11:51:02	00:12:27
00:01:59	00:30:00	12:53:02	00:17:03
00:02:00	00:32:00	12:55:02	00:13:51
00:02:00	00:05:59	10:21:02	00:04:47
00:02:00	00:02:02	10:07:02	00:11:20
00:01:59	00:01:59	10:05:03	00:12:53

Figure 5.13: Predicted Wait-Times Snapshot (Regression)

Figure 5.14 shows the average waiting times for a month on vendor basis, i.e., considering the wait times of all the tools of a vendor.

Figure 5.14: Monthly Wait Time for Vendors

Figure 5.15 shows the average waiting time of selected features of a particular vendor for a month.

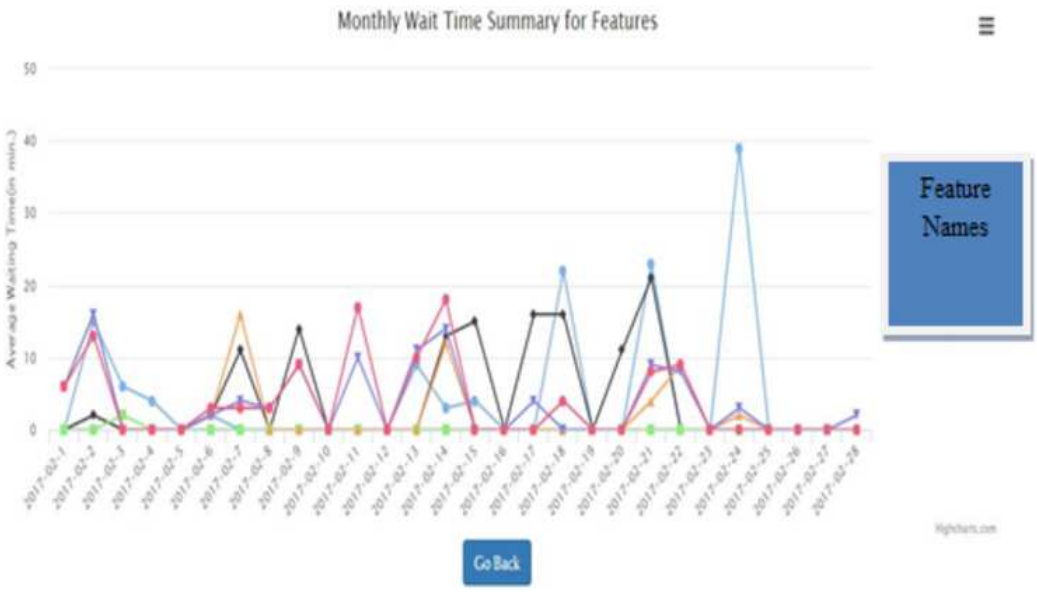

Figure 5.15: Monthly Wait Time for Features

Chapter 6

Conclusions and Future Works

This chapter is the concluding part of the thesis and also proposes some suggestions towards which the present work can be further extended. Section 6.1 brings out the overall conclusions of the research work carried out in this thesis and in section 6.3 suggestions regarding the future research directions and possible extensions of the work presented in the thesis are made.

6.1 Conclusion

These requirements target increased flexibility of the license management to cope with the flexibility and security of the license tokens and integrity of the execution environment. In the course of the project, these requirements will result in a number of developments in enhancing the dashboard to monitor license usage in real-time, which will be used in applying Data Analytics and Machine Learning, helping the Team in achieving the ultimate target of minimizing the EDA yearly budget.

6.2 Thesis Contribution

In this thesis, some development tools are mentioned like Heatmap for having real-time hourly license usage in world-wide areas. Also, there is an application Watchdog, which is used for handling zombies and hung licenses, helping in better management. Also, in Waiting Lounge application, Wait-times prediction for queued requests is introduced which can help the user in scheduling their job execution time, as well as the Global License Managers to manage the license availability in an effective manner, so that users do not have to wait for a longer period.

6.3 Scope for Future Work

Research is an iterative and continuous procedure. The work presented in the thesis focuses on minimizing the EDA cost without affecting the R&D effectiveness, using development as well as research. There are several directions in which this work could be expanded.

Some of the suggestions for future work in this direction are:

i Heatmap tool GUI can be further improved by addons like Trend page for expensive tools and project wise license usage tracking.

ii In Waiting Lounge, currently we have different tables handling the data for different vendors. This can be optimized to have only single dedicated DB Architecture, which can handle all the vendors added in the future.

iii Also, reading method of the license statistics files for different vendors can be optimized to a generic level, which can handle all the existing type of license statistics data present in the company.

References

[1] Jan Schmidt, Petr Fier, and Jirí Balcárek. On robustness of eda tools. In *Digital System Design (DSD), 2014 17th Euromicro Conference on*, pages 427–434. IEEE, 2014.

[2] Flexera. Flexera, 2017.

[3] RepriseSoftware. Software license management, 2017.

[4] Sergiu Costea and Bogdan Warinschi. Secure software licensing: Models, constructions, and proofs. In *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th*, pages 31–44. IEEE, 2016.

[5] D. V. Lindley. The theory of queues with a single server. *Mathematical Proceedings of the Cambridge Philosophical Society*, 48(2):277289, 1952.

[6] David G Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

[7] Dinesh R Bettadapur. Software licensing models in the eda industry. In *Design Automation Conference 1998. Proceedings of the ASP-DAC'98. Asia and South Pacific*, pages 235–239. IEEE, 1998.

[8] Herrick J Johnson, Margaret Olson, Stuart Jones, Stephanie Bodoff, Stephen C Bertrand, and Paul H Levine. Network license server, June 11 1991. US Patent 5,023,907.

[9] Bo Chen, Wei-wei Zhang, Ling Yu, and Ping Jiang. Cloud licensing model for. net software protection. In *Computer Science & Education (ICCSE), 2012 7th International Conference on*, pages 1069–1074. IEEE, 2012.

[10] Petr Fišer, Jan Schmidt, and Jiří Balcárek. Sources of bias in eda tools and its influence. In *Design and Diagnostics of Electronic Circuits & Systems, 17th International Symposium on*, pages 258–261. IEEE, 2014.

[11] Jethro B de Guzman, Remedios G Ado, et al. Mobile emergency response application using geolocation for command centers. *International Journal of Computer and Communication Engineering*, 3(4):235, 2014.

[12] James T O'Connor. Present value analysis applied to eda software alternatives. In *Economics of Design, Test, and Manufacturing, 1994. Proceedings., Third International Conference on the*, page 119. IEEE, 1994.

[13] Vineet Sharma, SAM Rizvi, and S Zeeshan Hussain. Distributed software and license key management an initiative to stop software piracy. In *UBICC, Ubiquitous Computing & Communication Journal*, volume 4, 2009.

[14] George Coulouris, Jean Dollimore, and Tim Kindberg. Election algorithm, bully algo & ring based algo. *Distributed Systems*, pages 445–448, 2006.

[15] Leili Noorian and Mark Perry. Autonomic software license management system: an implementation of licensing patterns. In *Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on*, pages 257–263. IEEE, 2009.

[16] Mikhail J Atallah and Jiangtao Li. Enhanced smart-card based license management. In *E-Commerce, 2003. CEC 2003. IEEE International Conference on*, pages 111–

119. IEEE, 2003.

[17] Yawei Zhang, Lei Jin, Xiaojun Ye, and Dongqing Chen. Software piracy prevention: Splitting on client. In *Security Technology, 2008. SECTECH'08. International Conference on*, pages 62–65. IEEE, 2008.

[18] Daniel Ferrante. Software licensing models: what's out there? *IT Professional*, 8(6), 2006.

[19] PYTHON UNIT IV and UNIT V Perl. Open source software unit i introduction 9.