ENERGY EFFICIENT IOT VIRTUALIZATION FRAMEWORK WITH PEER TO PEER NETWORKING AND PROCESSING

A PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, ANANTHAPURAMU

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

SK. Ziyauddin	**(H.T. No. 19JN1A0595)**
T. Lokesh	**(H.T. No. 19JN1A05B0)**
P. Nanda Kumar	**(H.T. No. 19JN1A0586)**
M. Sathya Prakash	**(H.T. No. 19JN1A0569)**

Under the Esteemed Guidance of

Mr. P. Mohan M.Tech., (Ph.D.),
Assistant Professor,

Department of Computer Science and Engineering
SREE VENKATESWARA COLLEGE OF ENGINEERING
NAAC 'A' Grade Accredited Institution, An ISO 9001::2015 Certified Institution
(Approved by AICTE, New Delhi and Affiliated to JNTUA, Ananthapuramu)
NORTH RAJUPALEM(VI), KODAVALURU(M), S.P.S.R NELLORE (DT) – 524 316

2019-2023

SREE VENKATESWARA COLLEGE OF ENGINEERING

NAAC 'A' Grade Accredited Institution, An ISO 9001:: 2015 Certified Institution
(Approved by AICTE, New Delhi and Affiliated to JNTUA, Ananthapuramu)
NORTH RAJUPALEM(VI), KODAVALURU(M), S.P.S.R NELLORE (DT) – 524 316

Department of
COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Project entitled as **"ENERGY EFFICIENT IOT VIRTUALIZATION FRAMEWORK WITH PEERTO PEER NETWORKING AND PROCESSING"** that is being submitted by **SK. Ziyauddin (18JN1A0595), T. L o k e s h (19JN1A05B0), P. N a n d a K u m a r (19JN1A0586), M. Sathya Prakash (19JN1A0569)** in partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** by **Jawaharlal Nehru** is **Technological University Anantapur, Ananthapuramu** during the academic year **2022-2023**. It certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Signature of the guide

Mr. P. Mohan

Signature of Head of the Dept.

Dr K. Venkata Nagendra

Principal

Dr. P. Kumar Babu

External Viva-Voce conducted on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The satisfaction and elation that accompany the successful completion of any task would be incomplete without the mention of the people who have made it a possibility. It is a great privilege to express our gratitude and respect to all those who have guided and inspired us during the course of the project work.

We are highly thankful to **Mr. P. MOHAN, M.Tech., (Ph.D.), Assistant Professor** in the **Department of Computer Science and Engineering** for his inspiring guidance and for providing the background knowledge to deal with the problem at every phase of our project in a systematic manner.

We have immense pleasure in expressing our sincere thanks and deep sense of gratitude to **Dr. K. VENKATA NAGENDRA, M.Tech., Ph.D., Professor & Head of the Department of Computer Science and Engineering** for extending necessary facilities for the completion of the project. With immense respect, we express our sincere gratitude to **Dr. P. BABU NAIDU, Chairman** and **Dr. P. KUMAR BABU, Principal** for permitting us to take up our project work and to complete the project successfully.

We would like to express our sincere thanks to all our faculty members of the Department for their continuous cooperation, which has given us the cogency to build-up adamant aspiration over the completion of our project.

SK. Ziyauddin	19JN1A0595
T. Lokesh	19JN1A05B0
P. Nanda Kumar	19JN1A0586
M. Sathya Prakash	19JN1A0569

Dept. of Computer Science and Engineering
Sree Venkateswara College of Engineering,
NH 16 Bypass Road, North Rajupalem, Nellore.

DECLARATION

We hereby declare that the project entitled **"Energy Efficient IOT Virtualization Framework with Peer to Peer Networking and Processing"** has been done by us under the guidance of **Mr. P. Mohan M.Tech., (Ph.D.), Assistant Professor, Department of Computer Science & Engineering.** This project work has been submitted to **SREE VENKATESWARA COLLEGE OF ENGINEERING** as a part of partial fulfillment of the requirements for the award of degree of **Bachelor of Technology.**

We also declare that this project report has not been submitted at any time to another Institute or University for the award of any degree.

Project Associates,

SK. Ziyauddin	19JN1A0595
T. Lokesh	19JN1A05B0
P. Nanda Kumar	19JN1A0586
M. Sathya Prakash	19JN1A0569

TABLE OF CONTENTS

ABSTRACT

In this project, an energy efficient IoT virtualization framework with peer-to-peer (P2P) networking and edge processing is proposed. In this network, the IoT task processing requests are served by peers. IoT objects and relays that host virtual machines (VMs) represents the peers in the proposed P2Pnetwork.We have considered three scenarios to investigate the saving in power consumption and the system capabilities in terms of task processing. The first scenario is a relay only scenario, where the task requests are processed using relays only. The second scenario is an object only scenario, where the task requests are processed using the IoT objects only. The last scenario is a hybrid scenario, where the task requests are processed using both IoT objects and VMs. We have developed a mixed integer linear programming (MILP) model to maximize the number of processing tasks served by the system, and minimize the total power consumed by the IoT network. Based on the MILP model principles, we developed an energy efficient virtualized IoT P2P networks heuristic (EEVIPN).

LIST OF FIGURES

List of Abbreviations

UML	:	Unified Modeling Language.
IOT	:	Internet of Things
WSN	:	Wireless Sensor Network
RFC	:	Request For Comments
EEBDN	:	Energy Efficient Big Data Networks
PN	:	Processing Nodes
CBDN	:	Classical Big Data Networks
MILP	:	Mixed Integer Linear Programming
VM	:	Virtual Machine
SDLC	:	Software Development Life Cycle
RTM	:	Requirements Traceability Matrix
SRS	:	Software Requirements Specification
JVM	:	Java Virtual Machine
AWT	:	Abstract Window Toolkit

1. INTRODUCTION

1.1 Introduction

The dramatic recent developments in IoT are mainly driven by the tremendous need and benefits that can be gained from connecting our physical world to the Internet. It is expected that there will be 50 billion (and by some estimates, more) IoT interconnected devices in the coming years. This growth in the number of connected devices opens the doors to new applications, for example in agriculture, transportation, manufacturing, smart homes, smart healthcare, and M2M communications. Many challenges such as energy efficiency, reliability, security, interoperability and scalability have to be overcome before the planned growth in the number and functionalities of IoT can be realized. Given the expected number of devices, one of the most important challenges is energy efficiency and hence greening the associated networks, which grabbed attention in both the academic and industrial domains. Cloud computing is investigated as one of the solutions to the energy efficiency challenge in networks and data centers. However, with the large data generated by the connected IoT objects (expected to generate 2.3 trillion gigabytes of data every day by year 2020), emerging cloud computing with IoT poses new challenges which have to be addressed. Among these challenges is the hunger for more processing capabilities, high communication bandwidth, security, and latency requirements. A number of solutions were suggested to address these issues. The work started with distributed content placement, thus bringing content closer to users, distributed data centers, thus bringing the processing capabilities closer to users and IoT devices and distributed processing of big data, where processing the huge data generated by IoT devices near the source can extract knowledge from the data and hence transmit the small volume `extracted knowledge' messages, thus saving network and processing resources and hence energy. However, a different and potentially more efficient solution, advocated here, is to process the IoT data by the IoT objects themselves or by the devices in the nearest layer to these objects. According to Allied Business Intelligence (ABI), it is expected that 90% of the data created by the endpoints will be processed and stored locally rather than being handled by the conventional clouds. Since some complicated data processing tasks cannot be done by most of the IoT devices and sensors because of their limited capabilities, edge computing is proposed to provide more resources to serve such tasks in efficient and fast ways. One of the suggested ways to do this is the dynamic

installation of virtual machines (VMs) in the edge cloud to process the raw data generated by the tasks requested by the IoT objects. The processed results are then sent back to the objects. In this we considered a single IoT network consisting of IoT network elements (relays, coordinators and gateways). In this data processing and traffic aggregation were done by VMs hosted in cloudlets, where these mini clouds are distributed over the IoT network elements. The work was extended in this where two separated IoT networks were considered with the deployment of a Passive Optical Access Network (PON). The main goal of our previous work is to investigate the potential energy efficiency gains that can be made if a use is made of distributed cloudlets at the edge of the network compared to centralized cloudlets at highest layer of the implemented model. There is a recent trend in research toward proposing IoT platforms based on local computing close to the objects such as fog and edge computing. Such platforms have many common characteristics with our proposed architecture. In this, a combination of fog computing and microgrid is proposed in order to reduce the energy consumed by IoT applications. A set of measurements and experiments were implemented considering different processing and traffic requirements. In this, dynamic decisions can be made by the proposed IoT gateway to minimize the consumed energy by choosing the most efficient location for processing a task in the fog or in the cloud. This decision is affected by the type of deployed IoT application, weather forecasting and the availability of renewable sources. An edge computation platform is presented in where the design of an IoT gateway virtualized environment for IoT applications is proposed using lightweight virtualization technologies. In this work, IoT data processing can be achieved by making use of containerbased virtualization technologies such as Docker containers. Although IoT and wireless sensor networks (WSN) share many features, IoT is a more encompassing concept and term compared to WSN. A WSN can be part of an IoT.in WSN sensors are assumed to communicate wirelessly, therefore a wired set of sensors is strictly not a WSN, but such wired set of sensors can be part of an IoT. In IoT, the devices can be contacted via Internet, which is not a prerequisite in WSN. WSNs focus more on optimizing the use of limited resources, whereas IoT focuses in many cases on quality of service, resilience and data processing as in the current paper. In addition, IoT devices can make use of P2P communication capabilities and architectures. A number of advantages could be realized by using P2P communication systems compared to conventional communication systems such as energy efficiency, traffic reduction and reliability. Based on the potential energy efficiency advantage, we introduce our energy

efficient IoT network considering a combination of P2P communication between the IoT objects and edge computing while installing VMs in the relays. Computing tasks and the communication between the peers in our network is achieved through two stages. In the rest stage, objects send the requests for tasks to be served by other peers (represented by IoT objects and relays hosting VMs) through the directly connected relays in the network. In the second stage the results of the processed tasks are received. We assume the traffic generated by task requests is reduced after processing by different percentages depending on the complexity of the requested tasks.

1.2 Objective of the Project

In this project, we have investigated the energy efficiency of an IoT virtualization framework with P2P network and edge computing. This investigation has been carried out by considering three different scenarios. A MILP was developed to maximize the number of processing tasks served by peers and minimize the total power consumption of the network.

2. LITERATURE SURVEY

``A practical evaluation of information processing and abstraction techniques for the Internet of Things,'

The term Internet of Things (IoT) refers to the interaction and communication between billions of devices that produce and exchange data related to real-world objects (i.e. things). Extracting higher level information from the raw sensory data captured by the devices and representing this data as machine-interpretable or human-understandable information has several interesting applications. Deriving raw data into higher level information representations demands mechanisms to find, extract, and characterize meaningful abstractions from the raw data. These meaningful abstractions then have to be presented in a human and/or machine-understandable representation. However, the heterogeneity of the data originated from different sensor devices and application scenarios such as e-health, environmental monitoring, and smart home applications, and the dynamic nature of sensor data make it difficult to apply only one particular information processing technique to the underlying data. A considerable amount of methods from machine-learning, the semantic web, as well as pattern and data mining have been used to abstract from sensor observations to information representations. This paper provides a survey of the requirements and solutions and describes challenges in the area of information abstraction and presents an efficient workflow to extract meaningful information from raw sensor data based on the current state-of-the-art in this area. This paper also identifies research directions at the edge of information abstraction for sensor data. To ease the understanding of the abstraction workflow process, we introduce a software toolkit that implements the introduced techniques and motivates to apply them on various data sets.

``Future edge cloud and edge computing for Internet of Things applications,''

The Internet is evolving rapidly toward the future Internet of Things (IoT) which will potentially connect billions or even trillions of edge devices which could generate huge amount of data at a very high speed and some of the applications may require very low latency. The traditional cloud infrastructure will run into a series of difficulties due to centralized computation, storage, and networking in a small number of datacenters, and due to the relative long distance between the edge devices and the remote datacenters. To tackle this challenge, edge cloud and edge computing seem to be a promising possibility

which provides resources closer to the resource-poor edge IoT devices and potentially can nurture a new IoT innovation ecosystem. Such prospect is enabled by a series of emerging technologies, including network function virtualization and software defined networking. In this survey paper, we investigate the key rationale, the state-of-the-art efforts, the key enabling technologies and research topics, and typical IoT applications benefiting from edge cloud. We aim to draw an overall picture of both ongoing research efforts and future possible research directions through comprehensive discussions.

``Internet of Things: A survey on enabling technologies, protocols, and applications,''

This project provides an overview of the Internet of Things (IoT) with emphasis on enabling technologies, protocols, and application issues. The IoT is enabled by the latest developments in RFID, smart sensors, communication technologies, and Internet protocols. The basic premise is to have smart sensors collaborate directly without human involvement to deliver a new class of applications. The current revolution in Internet, mobile, and machine-to-machine (M2M) technologies can be seen as the first phase of the IoT. In the coming years, the IoT is expected to bridge diverse technologies to enable new applications by connecting physical objects together in support of intelligent decision making. This paper starts by providing a horizontal overview of the IoT. Then, we give an overview of some technical details that pertain to the IoT enabling technologies, protocols, and applications. Compared to other survey papers in the field, our objective is to provide a more thorough summary of the most relevant protocols and application issues to enable researchers and application developers to get up to speed quickly on how the different protocols fit together to deliver desired functionalities without having to go through RFCs and the standards specifications. We also provide an overview of some of the key IoT challenges presented in the recent literature and provide a summary of related research work. Moreover, we explore the relation between the IoT and other emerging technologies including big data analytics and cloud and fog computing. We also present the need for better horizontal integration among IoT services. Finally, we present detailed service use-cases to illustrate how the different protocols presented in the paper fit together to deliver desired IoT services.

``A survey: Internet of Things (IOT) technologies, applications and challenges,''

The main aim of this project is to discuss the Internet of things in wider sense and prominence on protocols, technologies and application along related issues. The main factor IoT concept is the integration of different technologies. The IoT is empowered by the hottest developments in RFID, smart sensors, communication technologies, and Internet protocols. Primary hypothesis is to have smart sensor dealing directly to deliver a class of applications without any external or human participation. Recently development in Internet and smart phone and machine-to-machine M2M technologies can be consider first phase of the IoT. In the coming years IoT is expected to be one of the main hub between various technologies by connecting smart physical objects together and allow different applications in support of smart decision making. In this paper we discuss IoT architecture and technical aspect that relate to IoT. Then, give over view about IoT technologies, protocols and applications and related issues with comparison of other survey papers. Our main aim to provide a framework to researcher and application developer that how different protocols works, over view of some key issues of IoT and the relation between IoT and other embryonic technologies including big data analytics and cloud computing.

``Energy efficient big data networks: Impact of volume and variety,''

In this project, we study the impact of big data's volume and variety dimensions on energy efficient big data networks (EEBDN) by developing a mixed integer linear programming (MILP) model to encapsulate the distinctive features of these two dimensions. First, a progressive energy efficient edge, intermediate, and central processing technique is proposed to process big data's raw traffic by building processing nodes (PNs) in the network along the way from the sources to datacenters. Second, we validate the MILP operation by developing a heuristic that mimics, in real time, the behavior of the MILP for the volume dimension. Third, we test the energy efficiency limits of our green approach under several conditions where PNs are less energy efficient in terms of processing and communication compared to data centers. Fourth, we test the performance limits in our energy efficient approach by studying a “software matching” problem where different software packages are required to process big data. The results are then compared to the classical big data networks (CBDN) approach where big data is only processed inside centralized data centers. Our results revealed that up to 52% and

47% power saving can be achieved by the EEBDN approach compared to the CBDN approach, under the impact of volume and variety scenarios, respectively. Moreover, our results identify the limits of the progressive processing approach and in particular the conditions under which the CBDN centralized approach is more appropriate given certain PNs energy efficiency and software availability levels.

``Virtualization framework for energy efficient IoT networks,''

In this project, we introduce a Mixed Integer Linear Programming (MILP) model to design an energy efficient cloud computing platform for Internet of Things (IoT) networks. In our model, the IoT network consisted of four layers. The first (lowest) layer consisted of IoT devices, e.g. temperature sensors. The networking elements (relays, coordinators and gateways) are located within the upper three layers, respectively. These networking elements perform the tasks of data aggregation and processing of the traffic produced by IoT devices. The processing of IoT traffic is handled by Virtual Machines (VMs) hosted by distributed mini clouds and located within the IoT networking elements. We optimized the number of mini clouds, their location and the placement of VMs to reduce the total power consumption induced by traffic aggregation and processing. Our results showed that the optimal distribution of mini clouds in the IoT network could yield a total power savings of up to 36% compared to processing IoT data in a single mini cloud located at the gateway layer.

``Interconnecting Fog computing and microgrids for greening IoT,''

The Internet of things (IoT) is hailed as the next big phase in the evolution of the Internet. IoT devices are rapidly proliferating, owing to widespread adoption by industries in various sectors. Unlike security and privacy concerns, the energy consumption of the IoT and its applications has received little attention by the research community. This paper explores different options for deploying energy-efficient IoT applications. Specifically, we evaluate the use of a combination of Fog computing and microgrids for reducing the energy consumption of IoT applications. First, we study the energy consumption of different types of IoT applications (such as IoT applications with differing traffic generation or computation) running from both Fog and Cloud computing. Next, we consider the role of local renewable energy provided by microgrids, and local weather forecasting, along with Fog computing. To evaluate our proposal, energy consumption modeling, practical experiments and measurements were performed. The

results indicate that the type of IoT application, the availability of local renewable energy, and weather forecasting all influence how a system makes dynamic decisions in terms of saving energy

``Enabling data processing at the network edge through lightweight virtualization technologies,'

Cloud computing plays a crucial role in making Internet of Things (IoT) becoming a central part of future industries, society and people's life. However, the increase of the number of devices connected to the different networks, the huge amount of data produced by them, and the advanced requirements of many IoT applications, has resulted in new technical challenges. This has led to the introduction of an edge-computing approach, which is intended to make the management of such networks more efficient. The paradigm aims to move part of the data processing operations close to the data sources. Such operations are performed by means of network entities - like IoT gateways or local servers - near the IoT device. In this paper, we describe the design of a multifunctional IoT gateway that, making use of lightweight virtualization technologies such as Docker containers, allows managing different services, including data processing services, so as to enable the migration towards the edge-based approach. Furthermore, we show how the introduction of container-based technologies can bring several benefits without impacting the gateway performance.

3. ANALYSIS

Introduction

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process.

3.1 Existing System

In existing systems, one of the most challenges are energy efficiency and hence greening the associated networks, which grabbed attention in both the academic and industrial domains. Cloud computing is investigated as one of the solutions to the energy efficiency challenge in networks and data centers with the large data generated by the connected IoT objects, emerging cloud computing with IoT poses new challenges which have to be addressed. Among these challenges is the hunger for we need more processing capabilities, high communication bandwidth, security, and latency requirements.

Disadvantages of Existing System:

1. Security is less.
2. Communication failures are occurred.

3.2 Proposed System

In this project, an energy efficient IoT virtualization framework with peer-to-peer (P2P) networking and edge processing is proposed. In this network, the IoT task processing requests are served by peers. IoT objects and relays that host virtual machines (VMs) represents the peers in the proposed P2Pnetwork.We have considered three scenarios to investigate the saving in power consumption and the system capabilities in terms of task processing. The first scenario is a relay only scenario, where the task requests are processed using relays only. The second scenario is an object only scenario, where the task requests are processed using the IoT objects only. The last scenario is a hybrid scenario, where the task requests are processed using both IoT objects and VMs. We have developed a mixed integer linear programming (MILP) model to maximize the

number of processing tasks served by the system, and minimize the total power consumed by the IoT network. Based on the MILP model principles, we developed an energy efficient virtualized IoT P2P networks heuristic (EEVIPN).

Advantages of Proposed System:

1. Saving the power consumption.

3.3 PROCESS MODEL USED WITH JUSTIFICATION

SDLC (Umbrella Model):

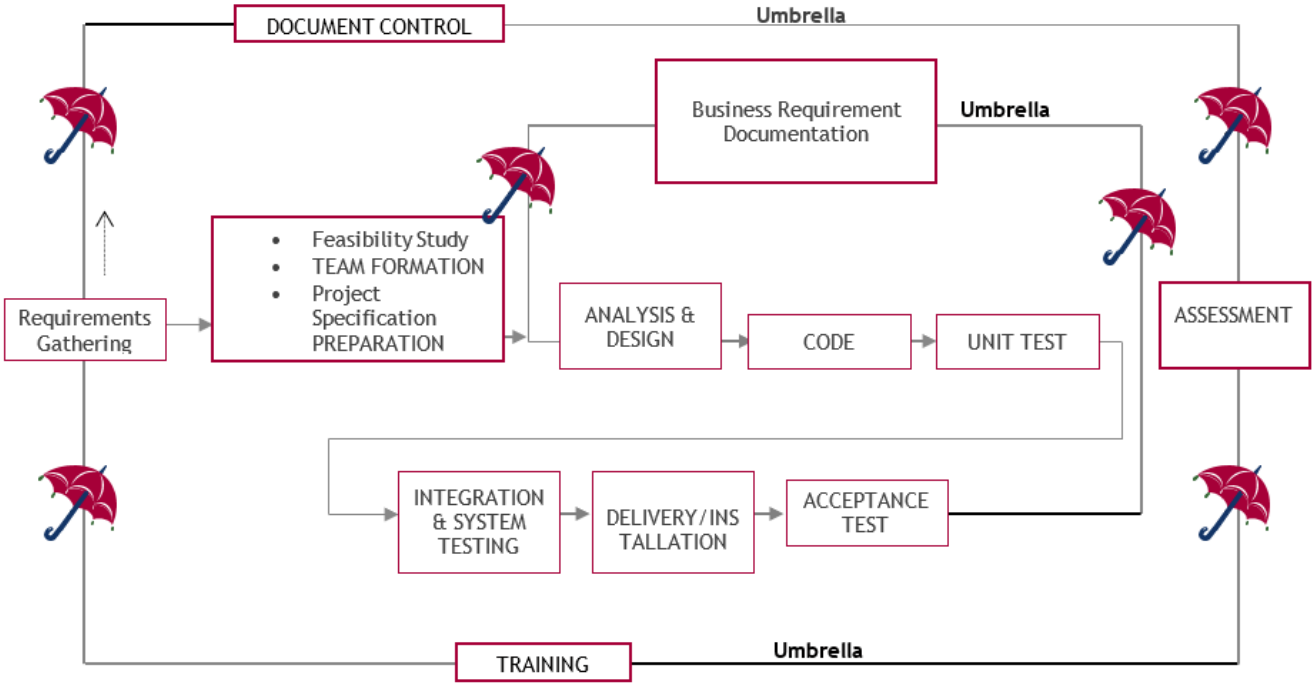

Fig 1: Umbrella Model

It is a standard which is used by software industry to develop good software.

3.3.1Stages in SDLC:

- Requirement Gathering
- Analysis
- Designing
- Coding
- Testing
- Maintenance

Requirements Gathering stage:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

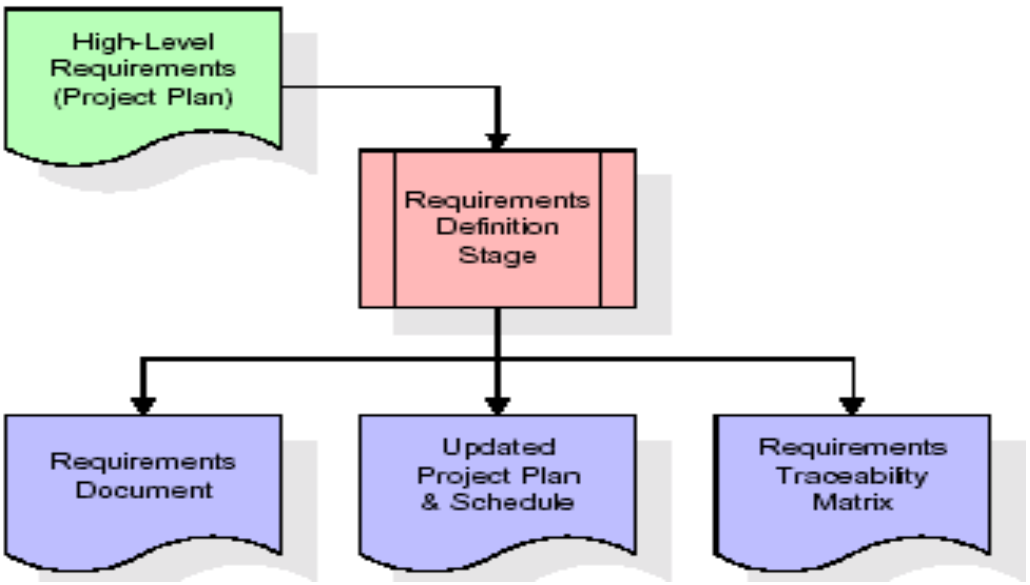

Fig 2: Requirement Gathering

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

- Feasibility study is all about identification of problems in a project.
- No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
- Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

Analysis Stage:

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.

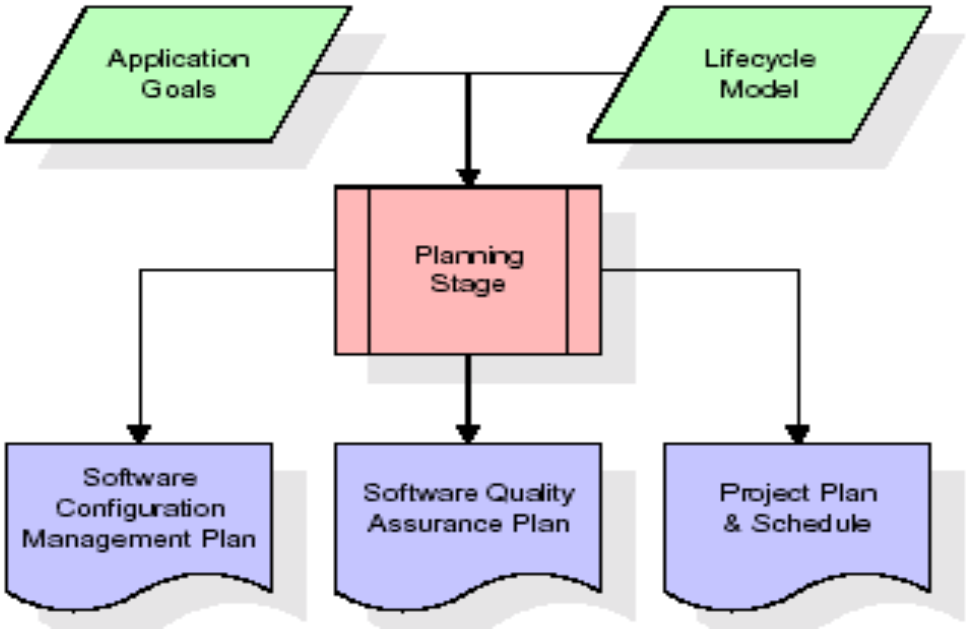

Fig 3: Analysis Stage

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

Designing Stage:

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

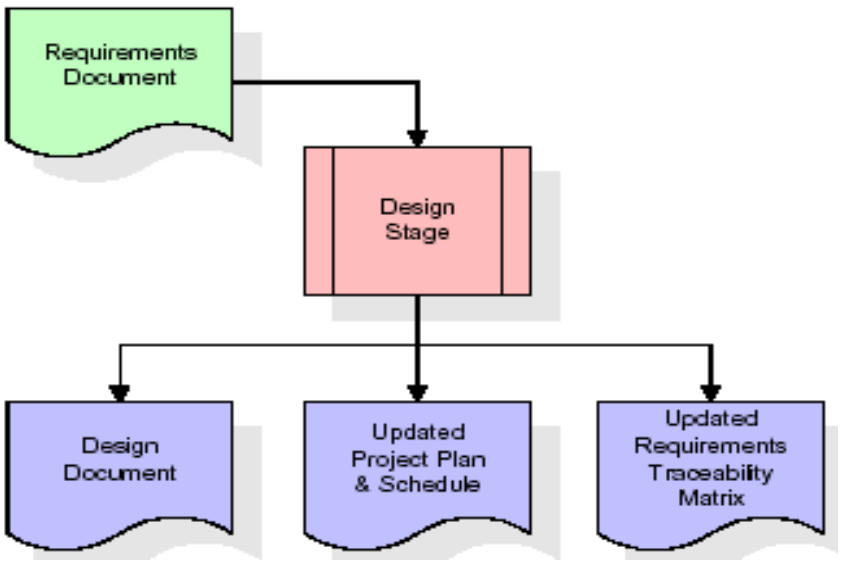

Fig 3: Designing Stage

When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

Development (Coding) Stage:

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

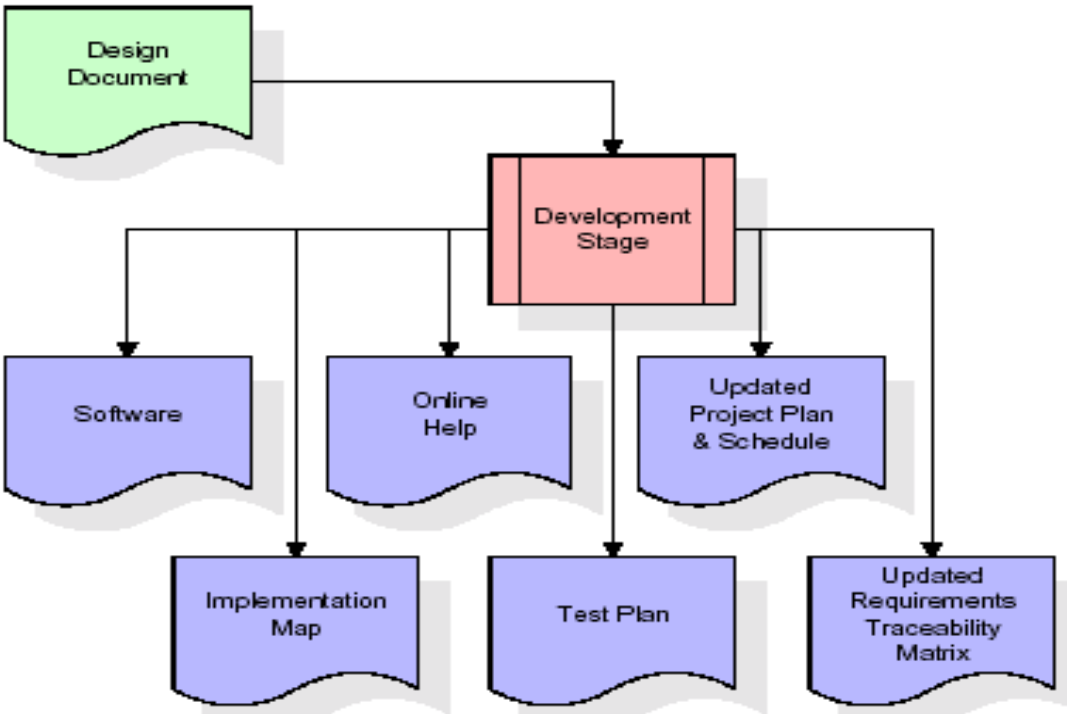

Fig 4: Development Stage

The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of

the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

Integration & Test Stage:

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

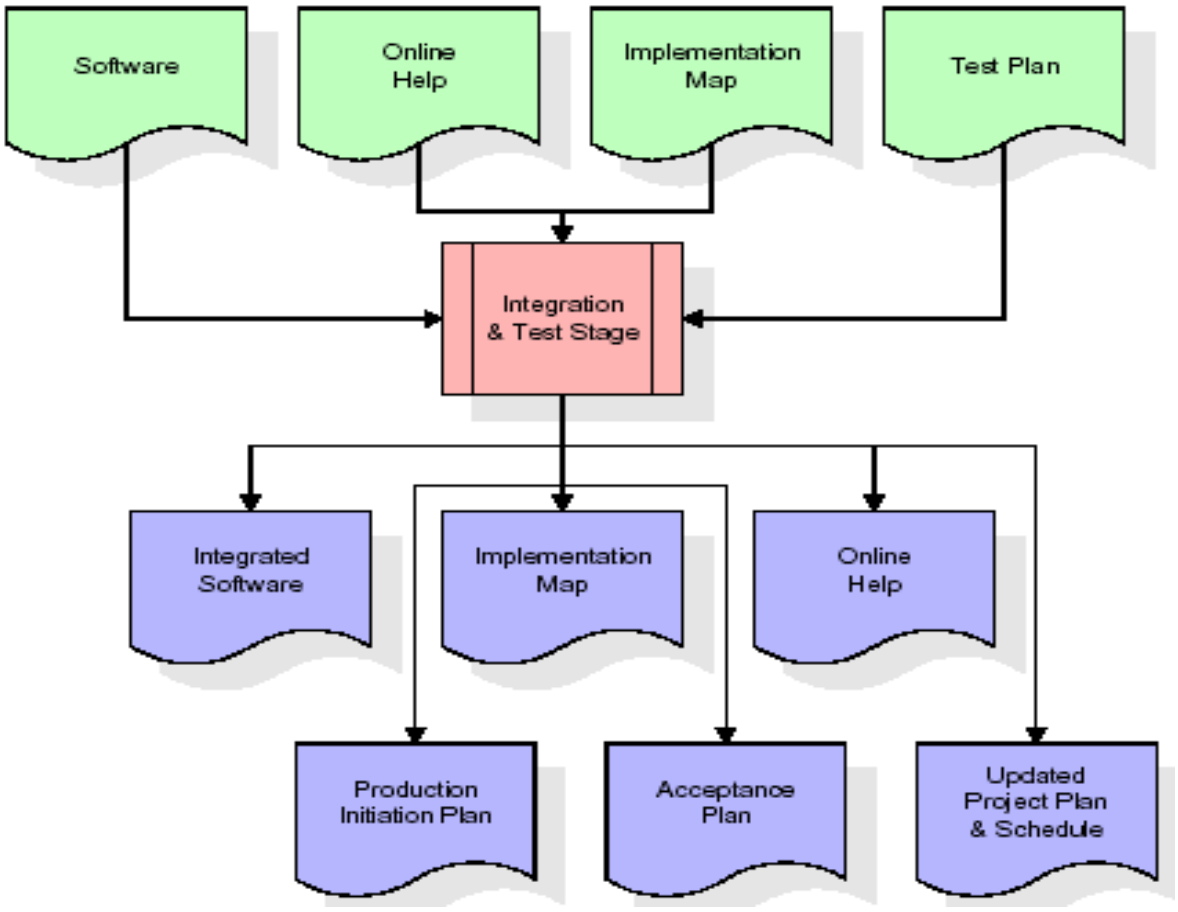

Fig 5: Integration & Test Stage

The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

Installation & Acceptance Test:

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.

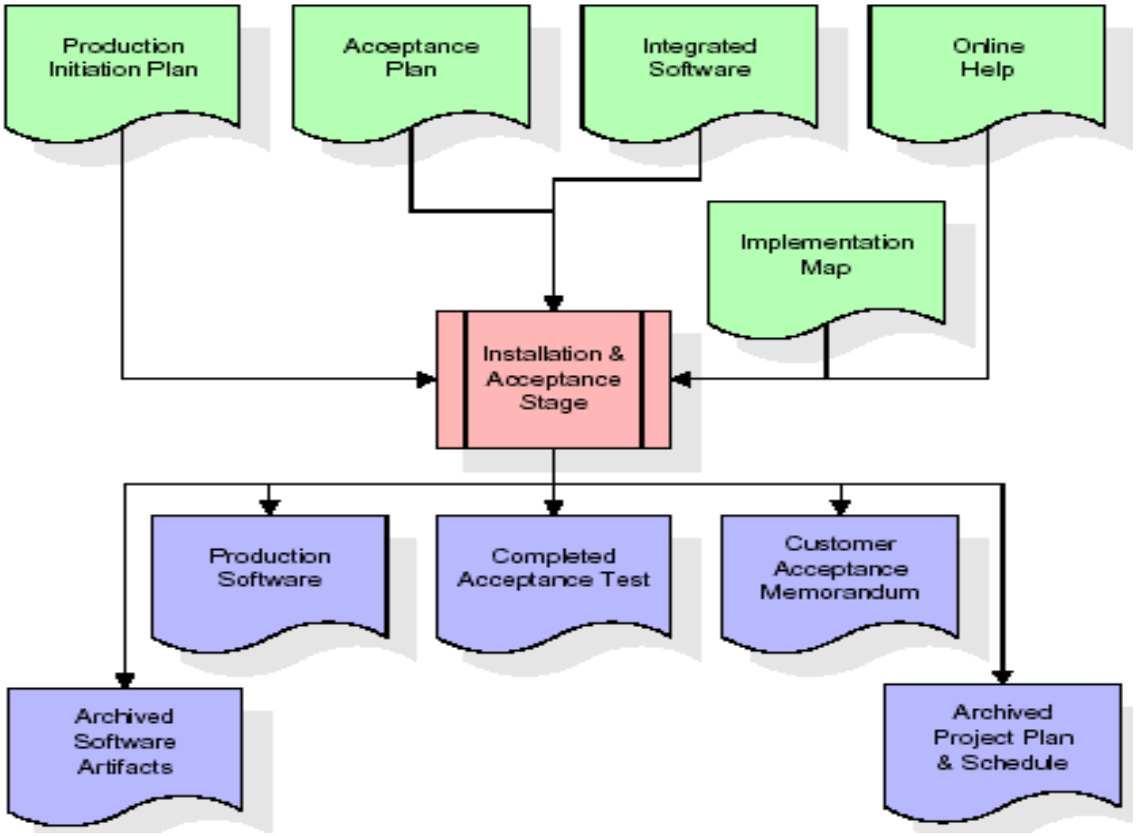

Fig 6: Installation & Acceptance Test

The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

Maintenance:

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

3.4 Software Requirement Specification

3.4.1. Overall Description

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Nonfunctional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- Business requirements describe in business terms *what* must be delivered or accomplished to provide value.

- Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)

- Process requirements describe activities performed by the developing organization. For instance, process requirements could specify. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economic feasibility for certain.

- OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration.

- TECHNICAL FEASIBILITY

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles.

3.4.2. External Interface Requirements

User Interface

The user interface of this system is a user-friendly Java Graphical User Interface.

Hardware Interfaces

The interaction between the user and the console is achieved through Java capabilities.

Software Interfaces

The required software is JAVA1.6.

Operating Environment

Windows XP, Linux.

HARDWARE REQUIREMENTS:

- **Processor - Intel i3**
- Speed - 1.30 Ghz
- RAM - 4 GB
- Hard Disk - 256 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

SOFTWARE REQUIREMENTS:

- Operating System : Windows XP
- Programming Language : Java

4. DESIGN

4.1UML diagrams

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

- **User Model View**

  i. This view represents the system from the user's perspective.

  ii. The analysis representation describes a usage scenario from the end-users perspective.

- **Structural Model view**

  i. In this model the data and functionality are arrived from inside the system.

  ii. This model view models the static structures.

- **Behavioral Model View**

  It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- **Implementation Model View**

  In this the structural and behavioral as parts of the system are represented as they are to be built.

- **Environmental Model View**

  In these the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

4.1 Class diagram: -

The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. A class with three sections, in the diagram, classes is represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

Class diagram for VM:

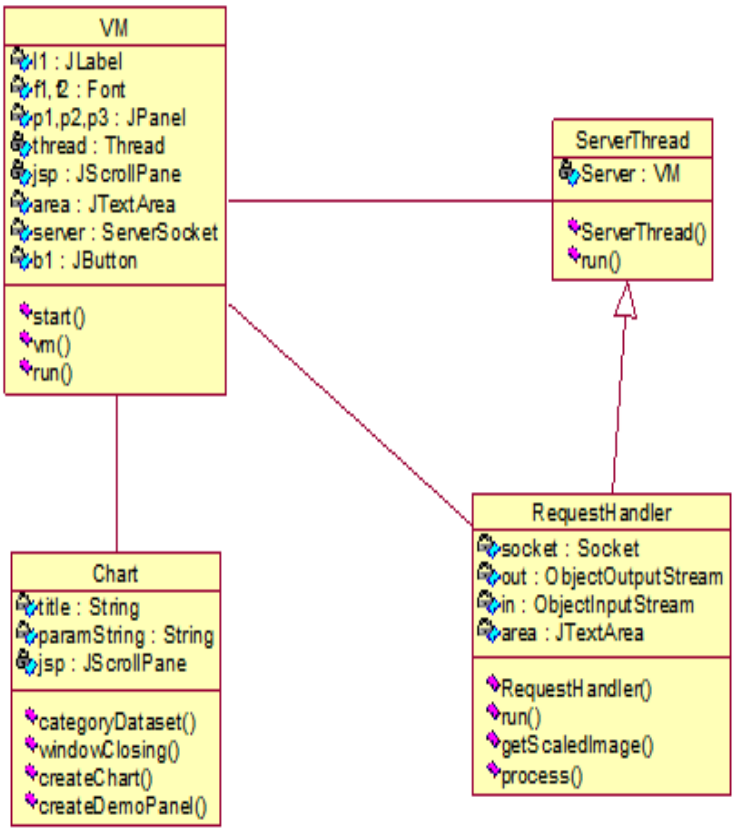

Fig 4.1.1: Class diagram for VM

Class Diagram for Simulation:

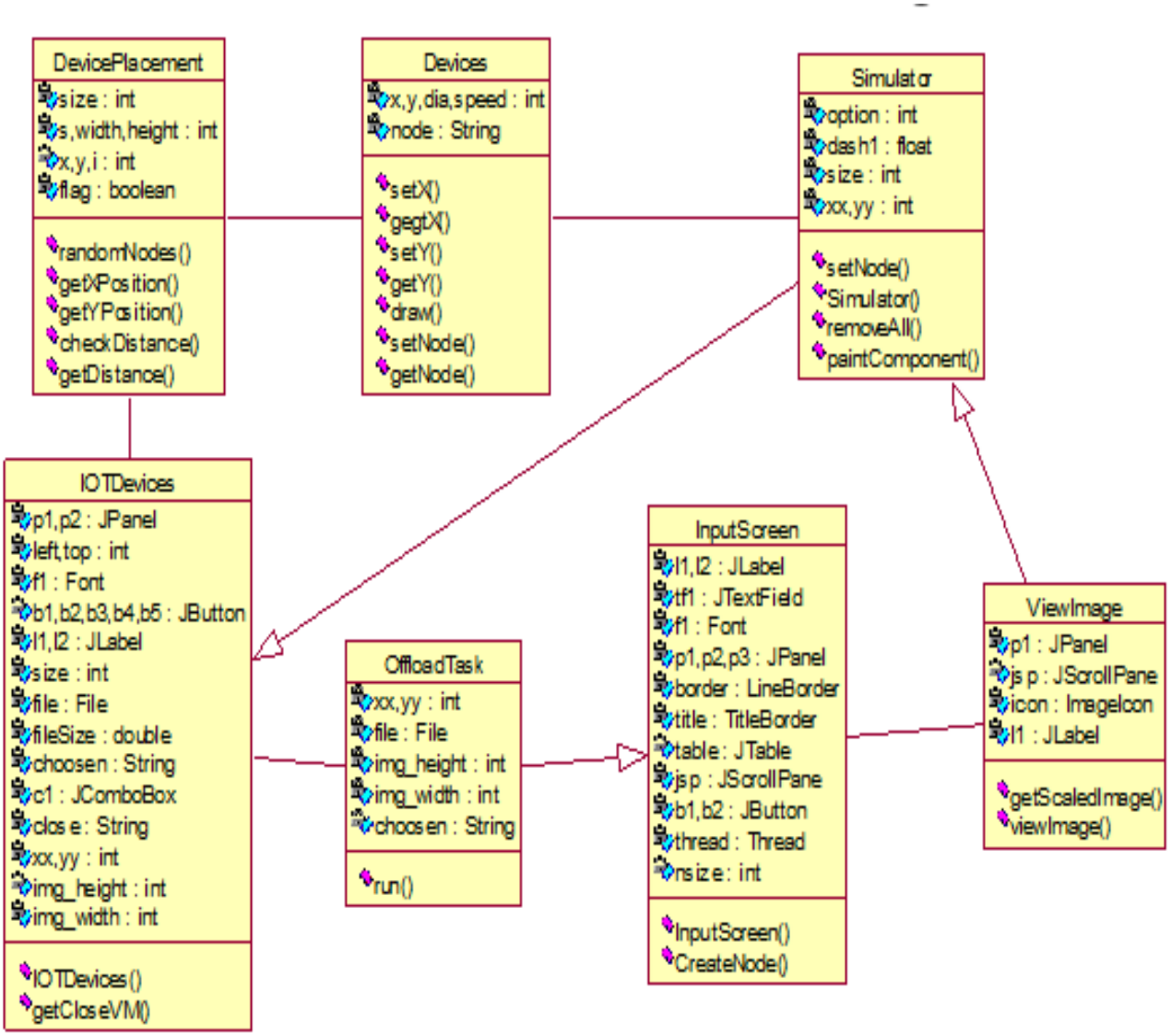

Fig4.1.2: Class Diagram for Simulation

4.2 Use Case Diagram: -

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

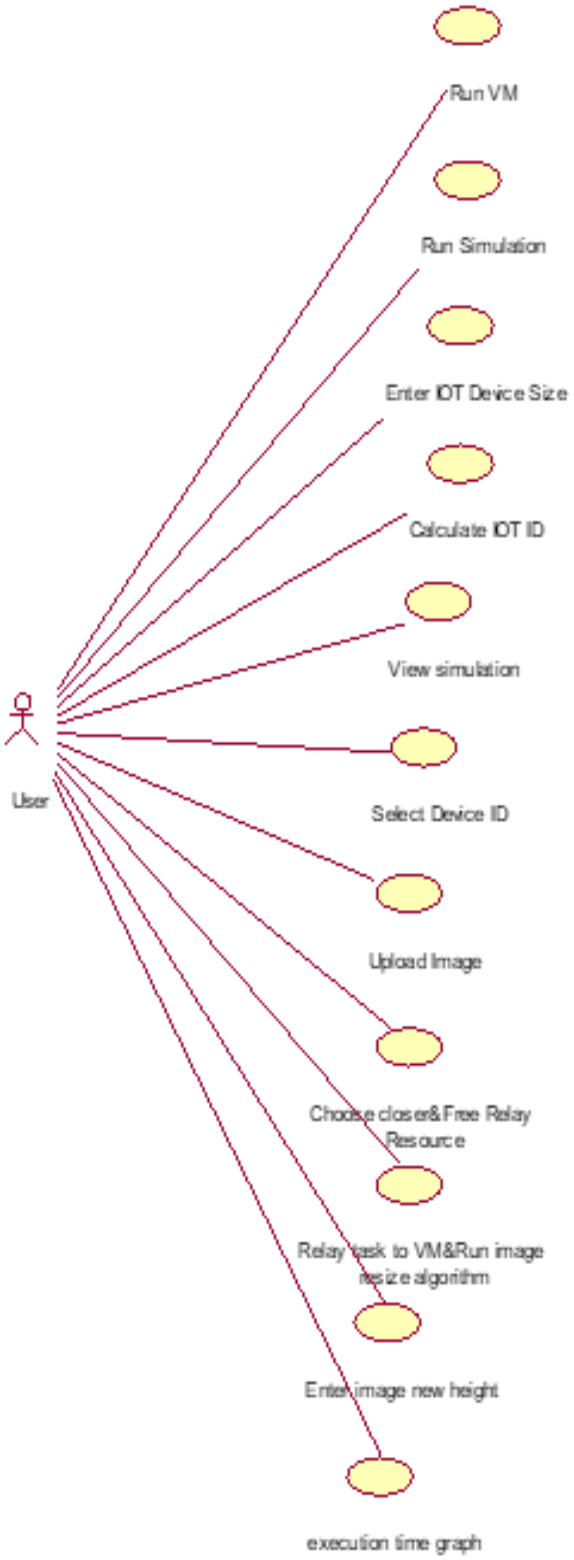

Fig 4.2.1: Use case diagram

4.3. Sequence Diagram:

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

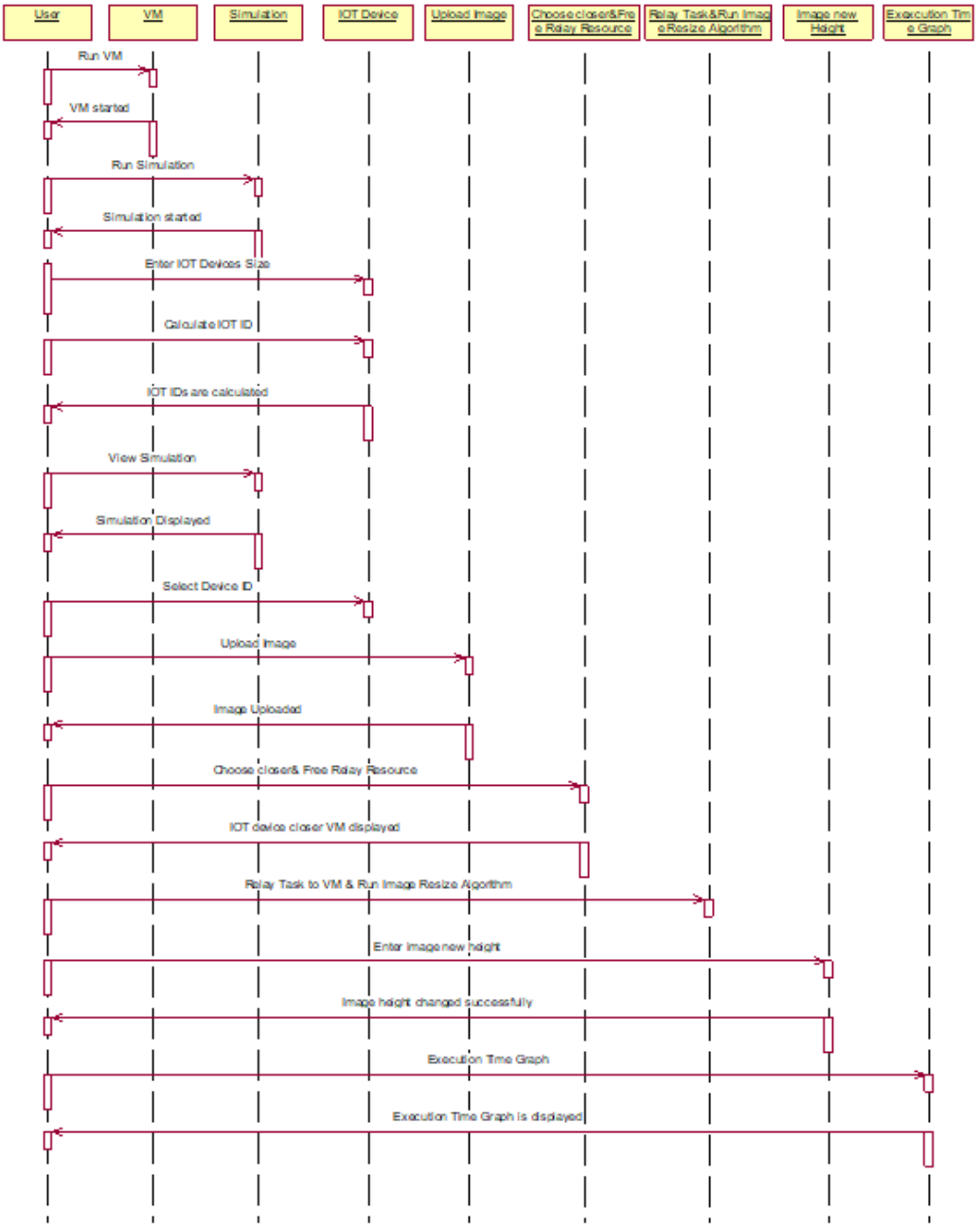

Fig 4.3.1: Sequence diagram

4.4 Collaboration diagram

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

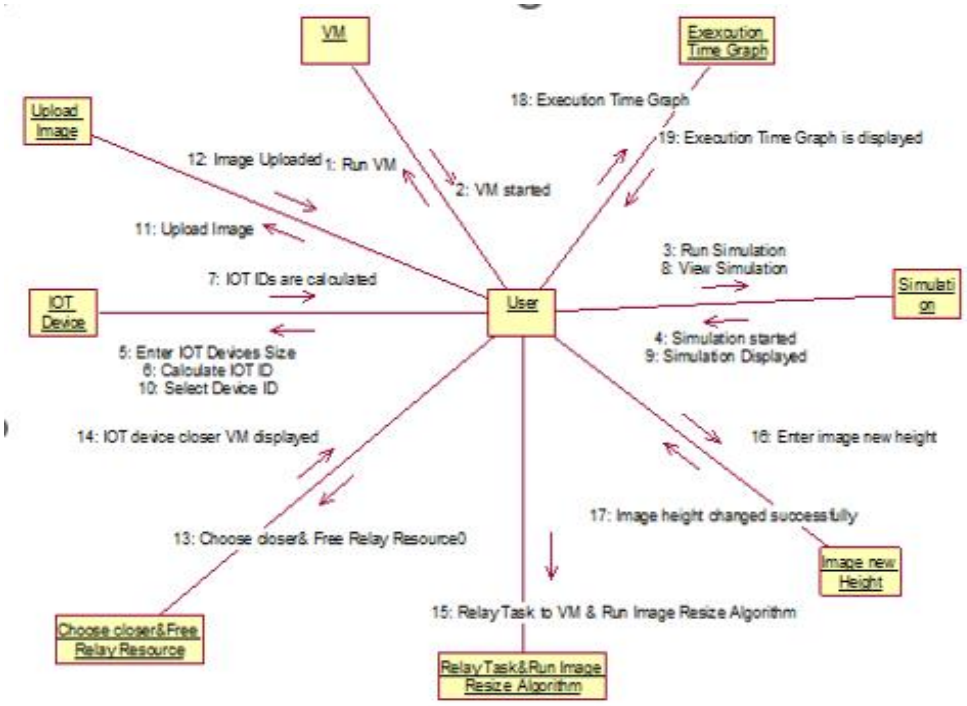

Fig 4.4.1: Collaboration diagram

4.5 Component Diagram

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

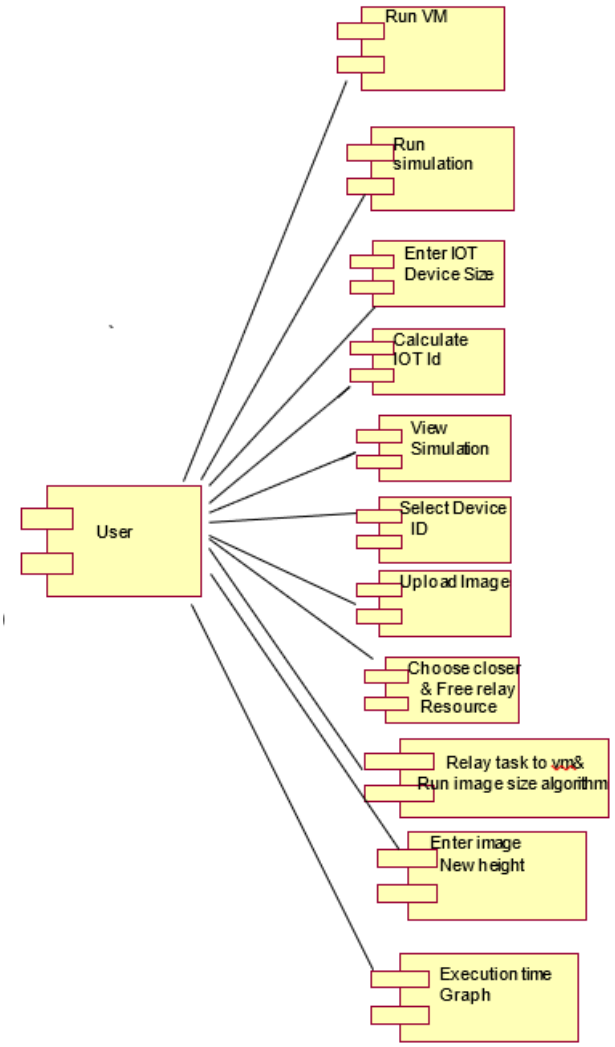

Fig 4.5.1: Component diagram

4.6 Deployment Diagram

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

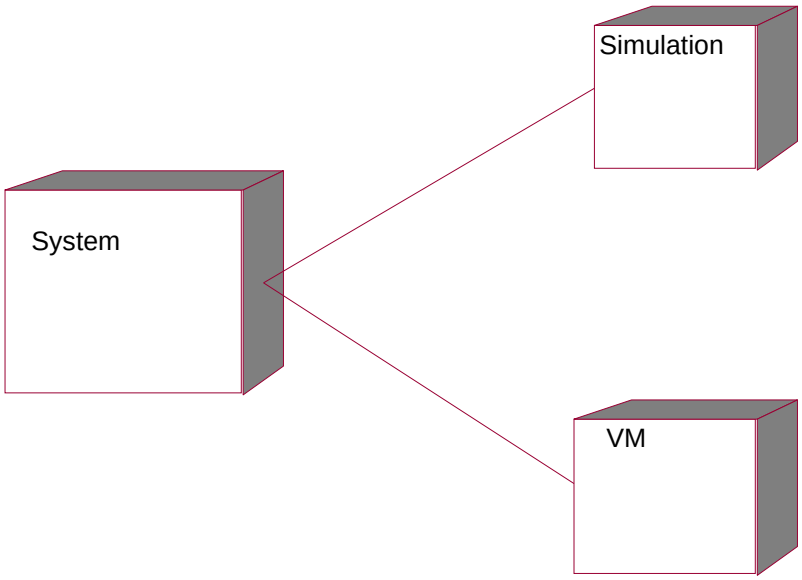

Fig 4.6.1: Deployment diagram

4.7 Activity diagram:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

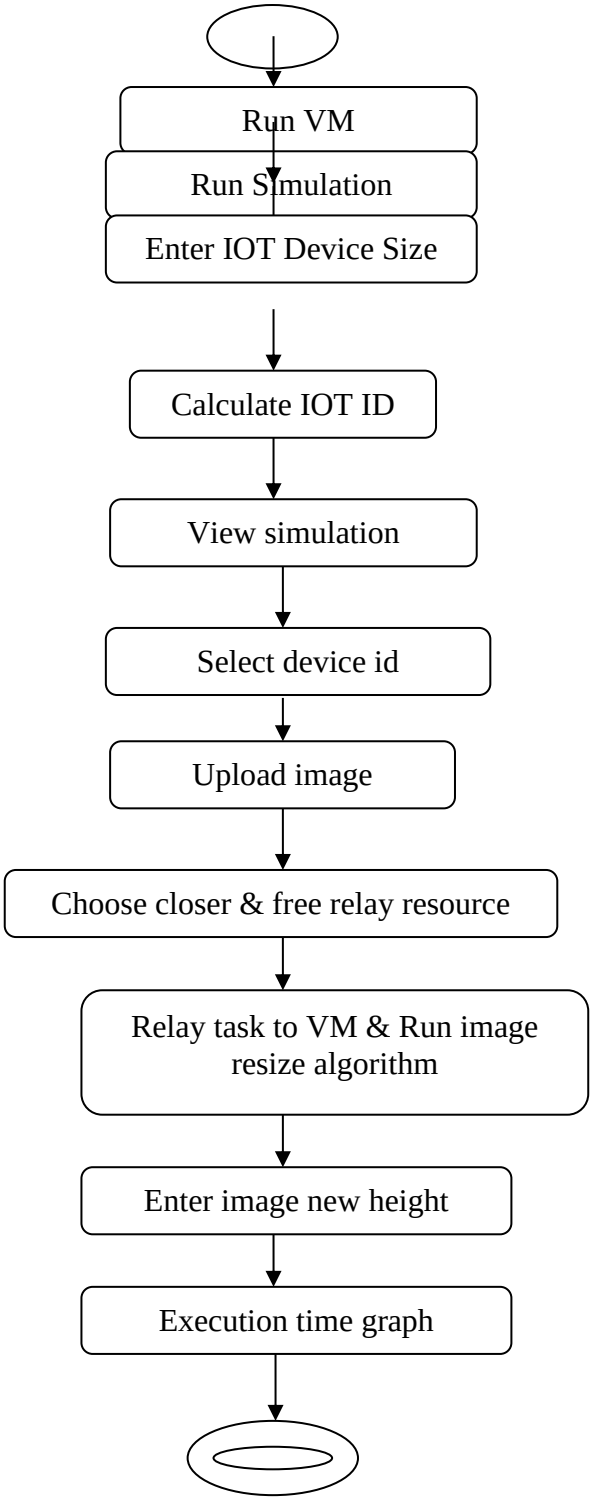

Fig 4.7.1: Activity diagram

4.8 Data Flow Diagram:

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

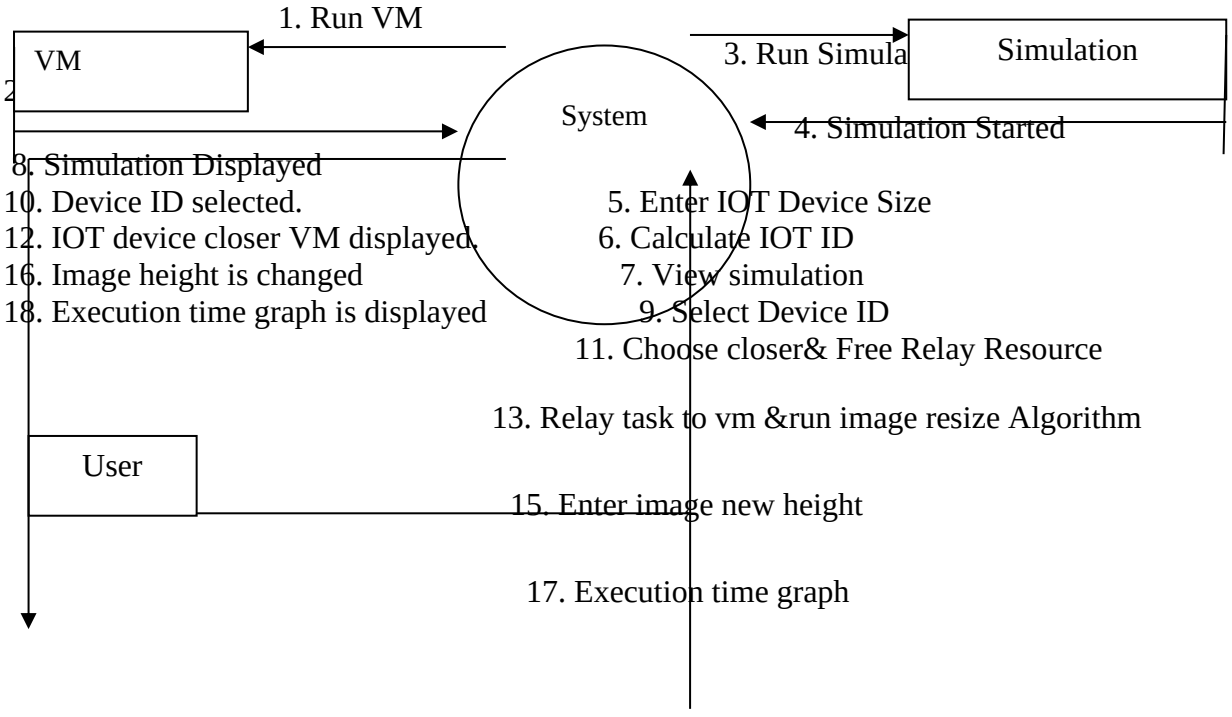

Fig 4.8.1: Data Flow Diagram

5. IMPLEMENTATION

5.1. Introduction of technologies used

About Java:

Initially the language was called as "oak" but it was renamed as "java" in 1995.The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer's language
- Java is cohesive and consistent
- Except for those constraint imposed by the Internet environment. Java gives the programmer, full control Finally, Java is to Internet Programming where c was to System Programming.

Importance of Java to the Internet

Java has had a profound effect on the Internet. This is because; java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs. in the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

Applications and applets

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++ .Java's ability to create Applets makes it important. An Applet I san application, designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet I actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can be react to the user input and dynamically change.

Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of code

When you compile the code, the Java compiler creates machine code (called byte code)for a hypothetical machine called Java Virtual Machine(JVM). The JVM is supposed t executed the byte code. The JVM is created for the overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines .This machine is called Java Virtual Machine.

Compiling and interpreting java source code.

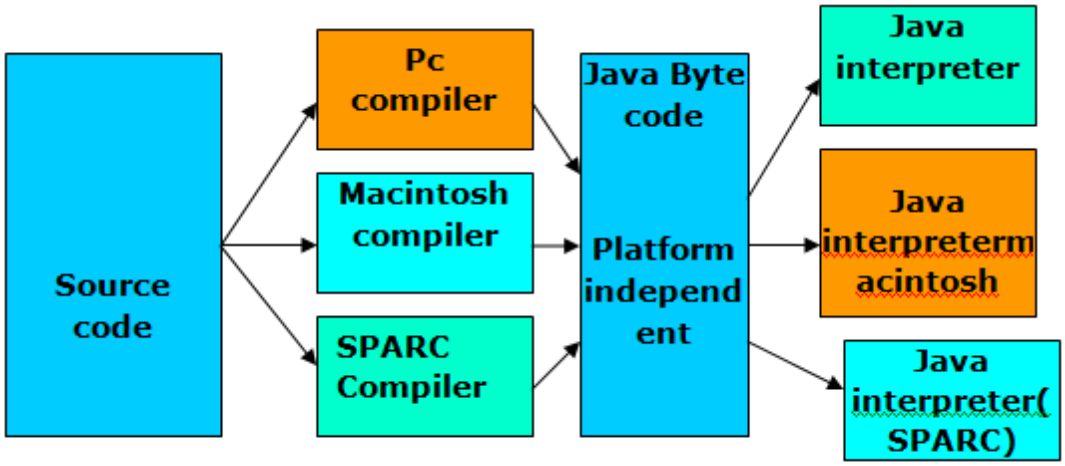

Fig 5.1.1: Java Architecture

During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be an Intel Pentium windows 95 or sun SPARCstation running Solaris or Apple Macintosh running system and all could receive code from any computer through internet and run the Applets.

Simple:

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ Programmer, learning Java will oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

Object oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank state. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

Robust

The multi-platform environment of the web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java? Java is strictly typed language; it checks your code at compile time and runtime.

Java virtually eliminates the problems of memory management and deal location, which is completely automatic. In a well-written Java program, all run-time errors can and should be managed by your program.

5.2 AWT and Swings:

AWT:

Graphical User Interface:

The user interface is that part of a program that interacts with the user of the program. GUI is a type of user interface that allows users to interact with electronic devices with images rather than text commands. A class library Java which is known as AWT for writing graphical programs. The AWT contains several graphical widgets which can be added and positioned to the display area with a layout manager.

As the Java programming language, the AWT is not platform-independent. AWT uses system peers object for constructing graphical widgets. A common set of tools is provided by the AWT for graphical user interface design. The implementation of the user interface elements provided by the AWT is done using every platform's native GUI toolkit. One of the AWT's significance is that the look and feel of each platform can be preserved.

Components:

A graphical user interface is built of graphical elements called components. A *component* is an object having a graphical representation that can be displayed on the screen and that can interact with the user. Components allow the user to interact with the program and provide the input to the program. In the AWT, all user interface components are instances of class Component or one of its subtypes. Typical components include such items as buttons, scrollbars, and text fields.

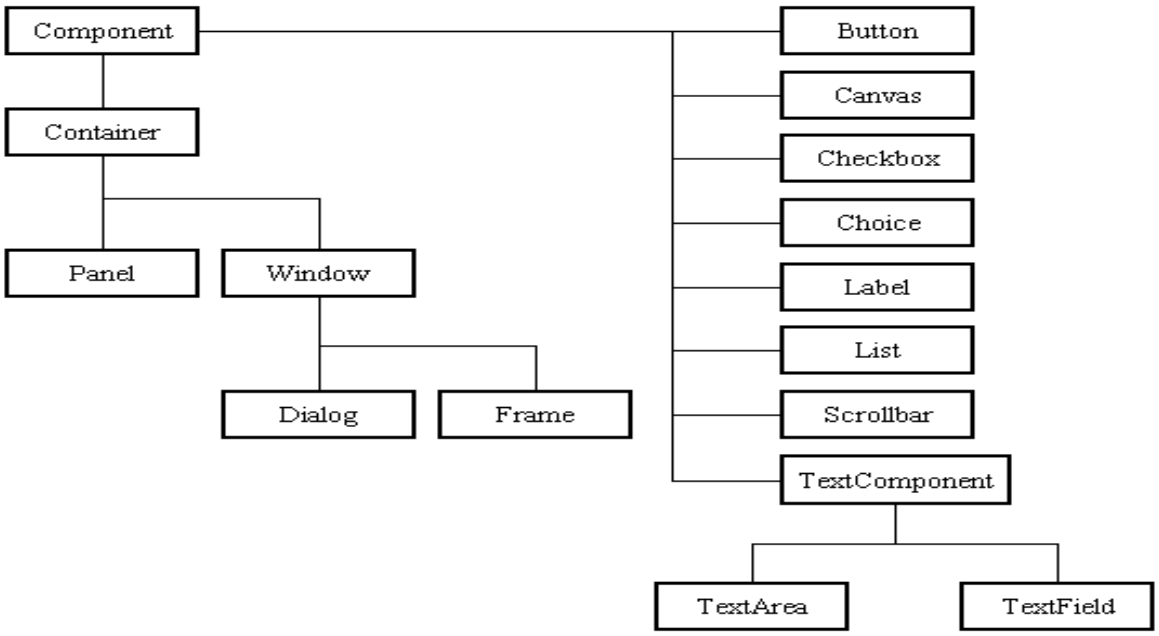

Fig 5.1.2: Types of Components

Before proceeding ahead, first we need to know what containers are. After learning containers, we learn all components in detail.

Containers:

Components do not stand alone, but rather are found within containers. In order to make components visible, we need to add all components to the container. Containers contain and control the layout of components. In the AWT, all containers are instances of class Container or one of its subtypes. Components must fit completely within the container that contains them. For adding components to the container we will use add() method.

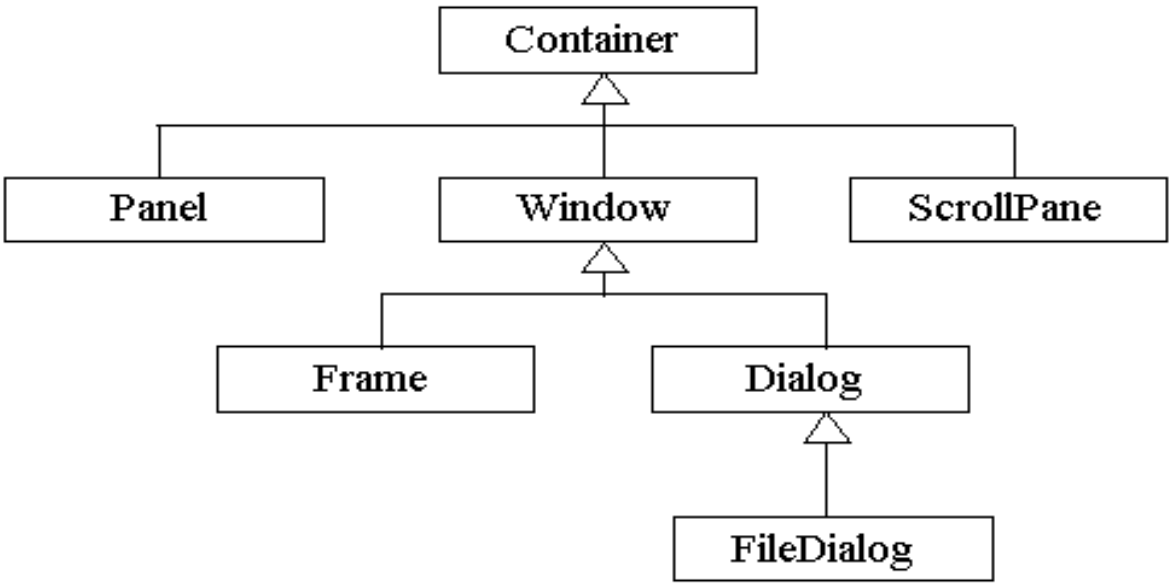

Fig 5.1.3: Types of containers

Basic GUI Logic:

The GUI application or applet is created in three steps. These are:

- Add components to Container objects to make your GUI.
- Then you need to setup event handlers for the user interaction with GUI.
- Explicitly display the GUI for application.

A new thread is started by the interpreter for user interaction when an AWT GUI is displayed. When any event is received by this new thread such as click of a mouse, pressing of key etc then one of the event handlers is called by the new thread set up for GUI. One important point to note here is that the event handler code is executed within the thread.

Creating a Frame:

Method1:

In the first method we will be creating frame by extending Frame class which is defined in java.awt package. Following program demonstrate the creation of a frame.

```
import java.awt.*;
public class FrameDemo1 extends Frame
{
  FrameDemo1()
  {
        setTitle("Label Frame");
        setVisible(true);
        setSize(500,500);
  }
  public static void main(String[] args)
  {
        new FrameDemo1 ();
  }
}
```

In the above program we are using three methods:

setTitle: For setting the title of the frame we will use this method. It takes String as an argument which will be the title name.

SetVisible: For making our frame visible we will use this method. This method takes Boolean value as an argument. If we are passing true then window will be visible otherwise window will not be visible.

SetSize: For setting the size of the window we will use this method. The first argument is width of the frame and second argument is height of the frame.

Method 2:

In this method we will be creating the Frame class instance for creating frame window. Following program demonstrate Method2.

```
import java.awt.*;

public class FrameDemo2

{

  public static void main(String[] args)

  {

        Frame f = new Frame();

        f.setTitle("My first frame");

        f.setVisible(true);

        f.setSize(500,500);

  }

}
```

Types of Components:

1) Labels:

This is the simplest component of Java Abstract Window Toolkit. This component is generally used to show the text or string in your application and label never perform any type of action.

```
        Label l1 = new Label("One");

        Label l2 = new Label("Two");

        Label l3 = new Label("Three",Label.CENTER);
```

In the above three lines we have created three labels with the name “one, two, three”. In the third label we are passing two arguments. Second argument is the justification of the label. Now after creating components, we will be adding it to the container.

```
add(l1);

add(l2);

add(l3);
```

We can set or change the text in a label by using the **setText( )** method. You can obtain the current label by calling **getText( )**. These methods are shown here:

void setText(String *str*)

String getText( )

1) Buttons:

This is the component of Java Abstract Window Toolkit and is used to trigger actions and other events required for your application. The syntax of defining the button is as follows :

```
Button l1 = new Button("One");

Button l2 = new Button("Two");

Button l3 = new Button("Three");
```

We can change the Button's label or get the label's text by using the Button.setLabel(String) and Button.getLabel() method.

2) CheckBox:

A check box is a control that is used to turn an option on or off. It consists of a small box that can either contain a check mark or not. There is a label associated with each check box that describes what option the box represents. You change the state of a check box by clicking on it. The syntax of the definition of Checkbox is as follows :

```
Checkbox Win98 = new Checkbox("Windows 98/XP", null, true);

Checkbox winNT = new Checkbox("Windows NT/2000");
```

Checkbox solaris = new Checkbox("Solaris");

Checkbox mac = new Checkbox("MacOS");

The first form creates a check box whose label is specified in first argument and whose group is specified in second argument. If this check box is not part of a group, then cbGroup must be **null**. (Check box groups are described in the next section.) The value true determines the initial state of the check box is checked. The second form creates a check box with only one parameter.

To retrieve the current state of a check box, call **getState( )**. To set its state, call **setState( )**. You can obtain the current label associated with a check box by calling **getLabel( )**. To set the label, call **setLabel( )**. These methods are as follows:

boolean getState( )

void setState(boolean *on*)

String getLabel( )

void setLabel(String *str*)

Here, if *on* is **true**, the box is checked. If it is **false**, the box is cleared. The string passed in *str* becomes the new label associated with the invoking check box.

3) Radio Button:

This is the special case of the Checkbox component of Java AWT package. This is used as a group of checkboxes which group name is same. Only one Checkbox from a Checkbox Group can be selected at a time. Syntax for creating radio buttons is as follows:

CheckboxGroup cbg = new CheckboxGroup();

Checkbox Win98 = new Checkbox("Windows 98/XP", cbg , true);

Checkbox winNT = new Checkbox("Windows NT/2000",cbg, false);

Checkbox solaris = new Checkbox("Solaris",cbg, false);

Checkbox mac = new Checkbox("MacOS",cbg, false);

For Radio Button we will be using CheckBox class. The only difference in Checkboxes and radio button is in Check boxes we will specify null for checkboxgroup but whereas in radio button we will be specifiying the checkboxgroup object in the second parameter.

4) Choice:

The Choice class is used to create a pop-up list of items from which the user may choose. Thus, a Choice control is a form of menu. Syntax for creating choice is as follows:

```
Choice os = new Choice();
/* adding items to choice */
        os.add("Windows 98/XP");
        os.add("Windows NT/2000");
        os.add("Solaris");
        os.add("MacOS");
```

We will be creating choice with the help of Choice class. Pop up list will be creating with the creation of object, but it will not have any items. For adding items we will be using add() method defined in Choice class.

To determine which item is currently selected, you may call either **getSelectedItem( )** or **getSelectedIndex( )**. These methods are shown here:

```
String getSelectedItem( )
int getSelectedIndex( )
```

The **getSelectedItem( )** method returns a string containing the name of the item. **getSelectedIndex( )** returns the index of the item. The first item is at index 0. By default, the first item added to the list is selected.

5) List:

List class is also same as choice but the only difference in list and choice is, in choice user can select only one item whereas in List user can select more than one item. Syntax for creating list is as follows:

List os = new List(4, true);

First argument in the List constructor specifies the number of items allowed in the list. Second argument specifies whether multiple selections are allowed or not.

```
/* Adding items to the list */

os.add("Windows 98/XP");

os.add("Windows NT/2000");

os.add("Solaris");

os.add("MacOS");
```

In list we can retrieve the items which are selected by the users. In multiple selection user will be selecting multiple values for retrieving all the values we have a method called getSelectedValues() whose return type is string array. For retrieving single value again we can use the method defined in Choice i.e. getSelectedItem().

6) TextField:

Text fields allow the user to enter strings and to edit the text using the arrow keys, cut and paste keys. TextField is a subclass of TextComponent. Syntax for creating list is as follows:

```
TextField tf1 = new TextField(25);

TextField  tf2 = new TextField();
```

In the first text field we are specifying the size of the text field and the second text field is created with the default value. **TextField** (and its superclass **TextComponent**) provides several methods that allow you to utilize a text field. To obtain the string currently contained in the text field, call **getText( )**. To set the text, call **setText( )**. These methods

are as follows:

String getText( )

void setText(String *str*)

We can control whether the contents of a text field may be modified by the user by calling **setEditable( )**. You can determine editability by calling **isEditable( )**. These methods are shown here:

boolean isEditable( )

void setEditable(boolean *canEdit*)

isEditable( ) returns **true** if the text may be changed and **false** if not. In **setEditable( )**, if *canEdit* is **true**, the text may be changed. If it is **false**, the text cannot be altered.

There may be times when we will want the user to enter text that is not displayed, such as a password. We can disable the echoing of the characters as they are typed by calling **setEchoChar( )**.

7) TextArea:

TextArea is a multiple line editor. Syntax for creating list is as follows:

TextArea area = new TextArea(20,30);

Above code will create one text area with 20 rows and 30 columns. **TextArea** is a subclass of **TextComponent**. Therefore, it supports the **getText( )**, **setText( )**, **getSelectedText( )**, **select( )**, **isEditable( )**, and **setEditable( )** methods described in the preceding section.

TextArea adds the following methods:

void append(String *str*)

void insert(String *str*, int *index*)

void replaceRange(String *str*, int *startIndex*, int *endIndex*)

The **append( )** method appends the string specified by *str* to the end of the current text. **insert( )** inserts the string passed in *str* at the specified index. To replace text, call

replaceRange( ). It replaces the characters from *startIndex* to *endIndex*–1, with the replacement text passed in *str*.

Layout Managers:

A layout manager automatically arranges controls within a window by using some type of algorithm. Each **Container** object has a layout manager associated with it. A layout manager is an instance of any class that implements the **LayoutManager** interface. The layout manager is set by the **setLayout( )** method. If no call to **setLayout( )** is made, then the default layout manager is used. Whenever a container is resized (or sized for the first time), the layout manager is used to position each of the components within it. The **setLayout( )** method has the following general form:

void setLayout(LayoutManager layoutObj)

Here, layoutObj is a reference to the desired layout manager. If you wish to disable the layout manager and position components manually, pass **null** for layoutObj. If we do this, you will need to determine the shape and position of each component manually, using the setBounds( ) method defined by Component.

Void setBounds(int x , int y , int width, int length)

In which first two arguments are the x and y axis. Third argument is width and fourth argument is height of the component.

Java has several predefined **LayoutManager** classes, several of which are described next. You can use the layout manager that best fits your application.

Flow Layout:

FlowLayout is the default layout manager. This is the layout manager that the preceding examples have used. **FlowLayout** implements a simple layout style, which is similar to how words flow in a text editor. Components are laid out from the upper-left corner, left to right and top to bottom. When no more components fit on a line, the next one appears on the next line. A small space is left between each component, above and below, as well as left and right. Here are the constructors for **FlowLayout**:

FlowLayout( )

FlowLayout(int *how*)

FlowLayout(int *how*, int *horz*, int *vert*)

The first form creates the default layout, which centers components and leaves five pixels of space between each component. The second form lets you specify how each line is

aligned. Valid values for *how* are as follows:

FlowLayout.LEFT

FlowLayout.CENTER

FlowLayout.RIGHT

These values specify left, center, and right alignment, respectively. The third form allows you to specify the horizontal and vertical space left between components in *horz* and *vert*, respectively.

BorderLayout:

The **BorderLayout** class implements a common layout style for top-level windows. It has four narrow, fixed-width components at the edges and one large area in the center. The four sides are referred to as north, south, east, and west. The middle area is called the center. Here are the constructors defined by **BorderLayout**:

BorderLayout( )

BorderLayout(int *horz*, int *vert*)

The first form creates a default border layout. The second allows you to specify the horizontal and vertical space left between components in *horz* and *vert*, respectively. **BorderLayout** defines the following constants that specify the regions:

BorderLayout.CENTER BorderLayout.SOUTH

BorderLayout.EAST BorderLayout.WEST

BorderLayout.NORTH

When adding components, you will use these constants with the following form of **add( )**, which is defined by **Container**:

void add(Component compObj, Object region);

Here, compObj is the component to be added, and region specifies where the component will be added.

GridLayout:

GridLayout lays out components in a two-dimensional grid. When you instantiate a **GridLayout**, you define the number of rows and columns. The constructors supported by **GridLayout** are shown here:

GridLayout( )

GridLayout(int numRows, int numColumns)

GridLayout(int numRows, int numColumns, int horz, int vert)

The first form creates a single-column grid layout. The second form creates a grid layout with the specified number of rows and columns. The third form allows you to specify the

horizontal and vertical space left between components in *horz* and *vert*, respectively. Either *numRows* or *numColumns* can be zero. Specifying *numRows* as zero allows for unlimited-length columns. Specifying *numColumns* as zero allows for unlimited-length rows.

Swings:

About Swings:

Swing is important to develop Java programs with a graphical user interface (GUI). There are many components which are used for the building of GUI in Swing. The Swing Toolkit consists of many components for the building of GUI. These components are also helpful in providing interactivity to Java applications. Following are components which are included in Swing toolkit:

- list controls
- buttons
- labels
- tree controls
- table controls

All AWT flexible components can be handled by the Java Swing. Swing toolkit contains far more components than the simple component toolkit. It is unique to any other toolkit in the way that it supports integrated internationalization, a highly customizable text package, rich undo support etc. Not only this you can also create your own look and feel using Swing other than the ones that are supported by it. The customized look and feel can be created using Synth which is specially designed. Not to forget that Swing also contains the basic user interface such as customizable painting, event handling, drag and drop etc.

The Java Foundation Classes (JFC) which supports many more features important to a GUI program comprises of Swing as well. The features which are supported by Java Foundation Classes (JFC) are the ability to create a program that can work in different languages, the ability to add rich graphics functionality etc.

There are several components contained in Swing toolkit such as check boxes, buttons, tables, text etc. Some very simple components also provide sophisticated functionality. For instance, text fields provide formatted text input or password field behavior. Furthermore, the file browsers and dialogs can be used to one's need and can even be customized.

Difference between Swings and AWT:

Swings	AWT
Swings are the light weight components.	AWTs are the heavy weight components.
Swings are developed by using pure java language.	AWTs are developed by using C and C++.
We can have different look and feel in swings.	This feature is not available in awt.
Swing has many advanced features like JTabel, JTabbedPane and JTree	This is not available in AWT.

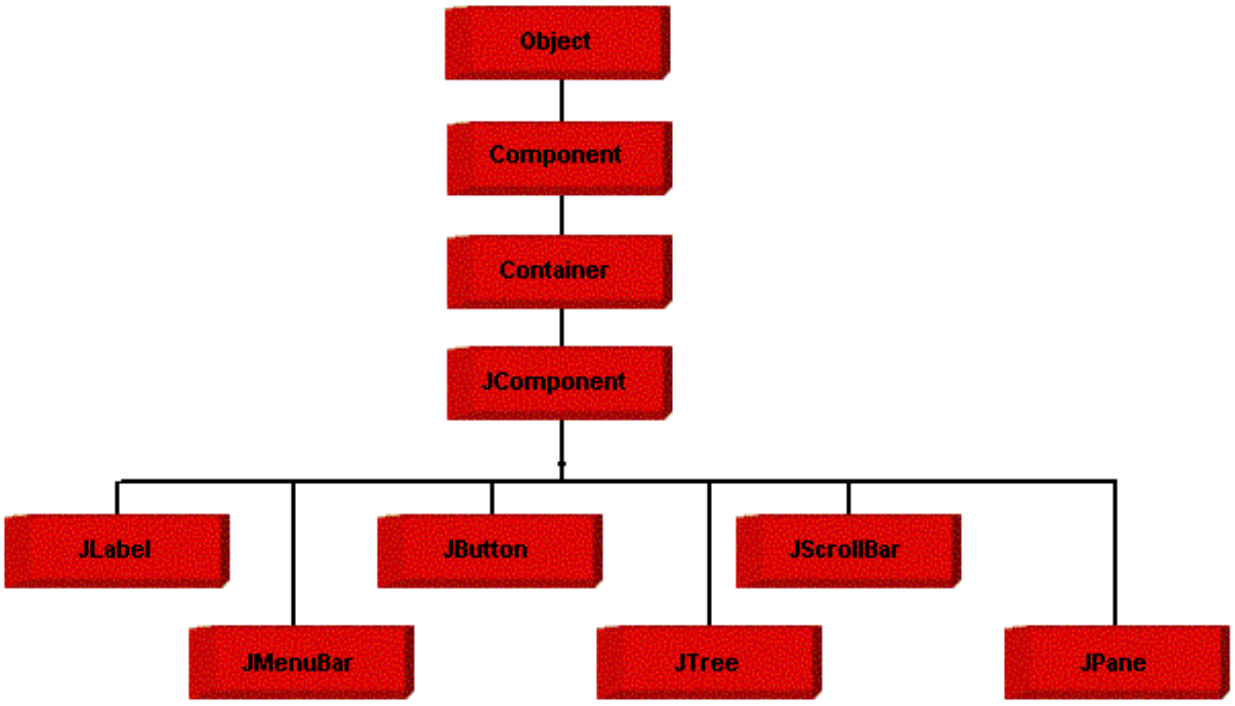

Fig 5.1.4: Java Swing Class Hierarchy

Swing Components:

All the components which are supported in AWT same components are also supported in Swings with a slight change in their class name.

AWT Components	Swing Components
Label	JLabel
TextField	JTextField
TextArea	JTextArea
Choice	JComboBox
Checkbox	JCheckBox
List	JList
Button	JButton
-	JRadioButton
-	JPasswordField
-	JTable
-	JTree
-	JTabbedPane
MenuBar	JMenuBar
Menu	JMenu
MenuItem	JMenuItem
-	JFileChooser
-	JOptionPane

We will discuss only those components which are not discussed in AWT chapter.

JTabbedPane class:

The JTabbedPane container allows many panels to occupy the same area of the interface, and the user may select which to show by clicking on a tab.

Constructor

```
JTabbedPane tp = new JTabbedPane();
```

Adding tabs to the JTabbedPane

Add tabs to a tabbed pane by calling `addTab` and passing it a String title and an instance of a class which should be called when we pressed a tab. That class should be a subclass of JPanel.

addTab(“String”,instance);

Example program:

```
import javax.swing.*;
import java.awt.*;
public class TabbedPaneDemo extends JFrame
{
    TabbedPaneDemo()
    {
        setLayout(new FlowLayout(FlowLayout.LEFT));
        setTitle("Tabbed Demo");
        setVisible(true);
        setSize(500,500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JTabbedPane pane = new JTabbedPane();
        pane.addTab("Countries",new Count());
        pane.addTab("Cities",new Cit());
        add(pane);
    }
    public static void main(String a[])
    {
        new TabbedPaneDemo();
    }
}
class Count extends JPanel
```

```
{
    Count()
    {
        JButton b1 = new JButton("India");
        JButton b2 = new JButton("SriLanka");
        JButton b3 = new JButton("Australia");
        add(b1);
        add(b2);
        add(b3);
    }
}
class Cit extends JPanel
{
    Cit()
    {
        JCheckBox cb1 = new JCheckBox("Hyderabad");
        JCheckBox cb2 = new JCheckBox("Banglore");
        JCheckBox cb3 = new JCheckBox("Pune");
        add(cb1);
        add(cb2);
        add(cb3);
    }
}
```

JMenuBar, JMenu, JMenuItem

A top-level window can have a menu bar associated with it. A menu bar displays a list of top-level menu choices. Each choice is associated with a drop-down menu. This concept is implemented in Java by the following classes: JMenuBar, JMenu, and JMenuItem. In general, a menu bar contains one or more JMenu objects. Each JMenu object contains a list of JMenuItem objects. Each JMenuItem object represents something that can be selected by the user. To create a menu bar, first create an instance of JMenuBar. This class only defines the default constructor. Next, create instances of JMenu that will define the selections displayed on the bar. Following are the constructors for Menu:

JMenu( )

JMenu(String optionName)

Here, optionName specifies the name of the menu selection. The first form creates an empty menu. Individual menu items are of type MenuItem. It defines these constructors:

JMenuItem( )

JMenuItem(String itemName)

Here, itemName is the name shown in the menu.

5.3 Sample Code

Simulator.java:

```
package com;

import java.awt.Dimension;

import javax.swing.JComponent;

import java.awt.geom.Rectangle2D;

import java.awt.BasicStroke;

import java.awt.GradientPaint;

import java.awt.Graphics2D;

import java.awt.Graphics;

import java.util.ArrayList;
```

```
import java.awt.Color;

import java.util.Random;

import java.net.Socket;

import java.io.ObjectOutputStream;

import java.io.ObjectInputStream;

import java.text.DecimalFormat;

public class Simulator extends JComponent{

    String col="empty";

    public int option=0;

    public ArrayList<Devices> devices = new ArrayList<Devices>();

    float dash1[] = {10.0f};

    BasicStroke dashed = new BasicStroke(1.0f,BasicStroke.CAP_BUTT,BasicStroke.JOIN_MITER,10.0f, dash1, 0.0f);

    BasicStroke rect=new BasicStroke(1f,BasicStroke.CAP_ROUND,BasicStroke.JOIN_ROUND,1f,new float[] {2f},0f);

    int size;


    Devices source = null;

    int xx,yy;

public void setNode(Devices source,int xx,int yy){

    this.source = source;

    this.xx = xx;
```

```
        this.yy = yy;
    }


    public Simulator(int size) {
        super.setBackground(new Color(81,123,138));
        this.size = size;
       this.setBackground(new Color(81,123,138));
    }


    public ArrayList<Devices> getList(){
        return devices;
    }
    public void removeAll(){
        option=0;
        devices.clear();
        col="empty";
        repaint();
    }


    public void paintComponent(Graphics g1){
        super.paintComponent(g1);
        GradientPaint gradient = new GradientPaint(0, 0, Color.blue, 175, 175,
```

```
Color.red,true);
        Graphics2D g = (Graphics2D)g1;
        g.setPaint(gradient);
        g.setStroke(rect);
        Rectangle2D rectangle = new Rectangle2D.Double(100,10,200,40);
        g.setStroke(rect);
        g.draw(rectangle);
        g.drawString("Relay/VM 1",180,40);
        rectangle = new Rectangle2D.Double(400,10,200,40);
        g.setStroke(rect);
        g.draw(rectangle);
        g.drawString("Relay/VM 2",480,40);
        rectangle = new Rectangle2D.Double(800,10,200,40);
        g.setStroke(rect);
        g.draw(rectangle);
        g.drawString("Relay/VM 3",880,40);


        if(option == 0){
                for(int i=0;i<devices.size();i++){
                        Devices d = devices.get(i);
                        if(d.getNode() != null){
                                d.draw(g,"fill");
```

```
                        g.drawString(d.getNode(),d.x+10,d.y+50);
                    }
                }
                g.setPaint(gradient);
            }
            if(option == 1){
                for(int i=0;i<devices.size();i++){
                    Devices d = devices.get(i);
                    if(d.getNode() != null){
                        d.draw(g,"fill");
                        g.drawString(d.getNode(),d.x+10,d.y+50);
                    }
                }
                g.drawLine(source.x+10,source.y+10,xx,yy+10);
                g.setPaint(gradient);
            }
        }
        }
```

DevicePlaement.java:

```
package com;
import java.util.Random;
import java.awt.Point;
```

```
import java.util.ArrayList;

public class DevicePlacement{

        static int size=40;

        static Simulator g;



public static void randomNodes(int s, int width, int height,Simulator nodes){

        g=nodes;

    randomNodes(s, width, height);

}

public static void randomNodes(int s, int width, int height){

        int x = getXPosition(100,900);

        int y = getYPosition(150,550);

        for(int i=1;i<=s;i++){

                boolean flag = checkDistance(x,y);

                if(!flag){

                        Devices d = new Devices(new Point(x, y), size);

                        d.setNode("IOT"+i);

                        g.devices.add(d);

                }else{

                        i = i - 1;

                }

                x = getXPosition(100,900);
```

```
                y = getYPosition(150,550);
        }
    }
    public static int getXPosition(int start,int end){
        Random rn = new Random();
        int range = end - start + 1;
        return rn.nextInt(range) + start;
    }
    public static int getYPosition(int start,int end){
        Random rn = new Random();
        int range = end - start + 1;
        return rn.nextInt(range) + start;
    }
    public static boolean checkDistance(int x,int y){
        boolean flag = false;
        for(int i=0;i<g.devices.size();i++){
                Devices d = g.devices.get(i);
                double d1 = getDistance(x,y,d.x,d.y);
                if(d1 < 50){
                        flag = true;
                        break;
                }
```

```
        }

        return flag;

    }

    public static double getDistance(int n1x,int n1y,int n2x,int n2y) {

        int dx = (n1x - n2x) * (n1x - n2x);

        int dy = (n1y - n2y) * (n1y - n2y);

        int total = dx + dy;

        return Math.sqrt(total);

    }

    }
```

RequestHandler.java:

```
package com;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.io.FileInputStream;

import java.io.File;

import javax.swing.JTextArea;

import java.io.FileOutputStream;

import java.awt.Graphics2D;

import java.awt.Image;

import java.awt.RenderingHints;
```

```
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
public class RequestHandler extends Thread{
   Socket socket;
   ObjectOutputStream out;
   ObjectInputStream in;
      JTextArea area;
public RequestHandler(Socket soc,JTextArea area){
   socket=soc;
      this.area=area;
      try{
       out = new ObjectOutputStream(socket.getOutputStream());
       in = new ObjectInputStream(socket.getInputStream());
       }catch(Exception e){
       e.printStackTrace();
   }
}
@Override
public void run(){
      try{
                process();
   }catch(Exception e){
```

```
            e.printStackTrace();
        }
    }
    public static BufferedImage getScaledImage(BufferedImage srcImg, int w, int h){
        BufferedImage resizedImg = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
        Graphics2D g2 = resizedImg.createGraphics();
        g2.setRenderingHint(RenderingHints.KEY_INTERPOLATION, RenderingHints.VALUE_INTERPOLATION_BILINEAR);
        g2.drawImage(srcImg, 0, 0, w, h, null);
        g2.dispose();
        return resizedImg;
    }
    public void process()throws Exception{
            Object input[]=(Object[])in.readObject();
            String type=(String)input[0];
            if(type.equals("image")){
                    long start = System.currentTimeMillis();
                    long start1 = start - 2000;
                    long start2 = start - 6000;
                    String file = (String)input[1];
                    byte img[] = (byte[])input[2];
                    int width = Integer.parseInt((String)input[3]);
```

```
int height = Integer.parseInt((String)input[4]);

String vm = (String)input[5];

String key = file+","+width+","+height;

FileOutputStream fout = new FileOutputStream(file);

fout.write(img,0,img.length);

fout.close();

BufferedImage bi = ImageIO.read(new File(file));

bi = getScaledImage(bi,width,height);

int index = file.lastIndexOf(".")+1;

String ext = file.substring(index,file.length());

ImageIO.write(bi,ext,new File(file));

FileInputStream fin = new FileInputStream(file);

byte b[] = new byte[fin.available()];

fin.read(b,0,b.length);

fin.close();

File temp = new File(file);

temp.delete();

Object res[] = {b};

area.append(vm+" received & processed image resize to "+width+" and "+height+"\n");

out.writeObject(res);

out.flush();
```

```
                long end = System.currentTimeMillis();

                VM.relay = end - start;

                VM.object = end - start2;

                VM.hybrid = end - start1;

        }

    }

    }
```

6. TESTING

Implementation and Testing:

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so

critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

Integration Testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

Acceptance Testing

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

Test Case Id	Test Case Name	Test Case Desc.	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Run VM and Simulator	Verify the VM and Simulator started or not	Without VM and Simulator	Users cannot do further operations	VM and Simulator are started	High	High
02	Enter IOT Device size	Verify IOT Device size is enter or not	Without entering the IOT Device size	It cannot display the IOT Device size	It can display the IOT Device size	High	High
03	View Simulation	Verify the simulation is displayed or not	Without assigning each id to device	The simulation cannot displayed	The simulation can displayed	High	High
04	Upload Image	Verify the image is uploaded or not	Without selecting any device id	we cannot upload image	selected image will send to vm	High	High
05	Choose closer &Free Relay Resource	Verify closer &Free Relay Resource are finded or not	without run this Choose closer &Free Relay Resource	we cannot find free and closer relay/vm	we cannot find free and closer relay/vm	High	High
06	Relay Task o vm & Run Image Resize Algorithm	Verify the node behavior chart is displayed or not	Without saving the abnormal weight of the sensors	The Node behavior Chart is not displayed	The Node behavior Chart is displayed successfully	High	High

07	Relay Task to vm & Run Image Resize Image Resize Algorithm	Verify Relay Task to vm & Run Image Resize Image Resize Algorithm run or not	Without finding closer relay/vm	The algorithm will not run	The algorithm will run and allow to enter image new height	High	High
08	Execution Time Graph	Verify the node behavior chart is displayed or not	Without saving the abnormal weight of the sensors	The Node behavior Chart is not displayed	The Node behavior Chart is displayed successfully	High	High

7. SCREEN SHOTS

First double click on 'run.bat' file from 'VM' folder to get below screen and let it run

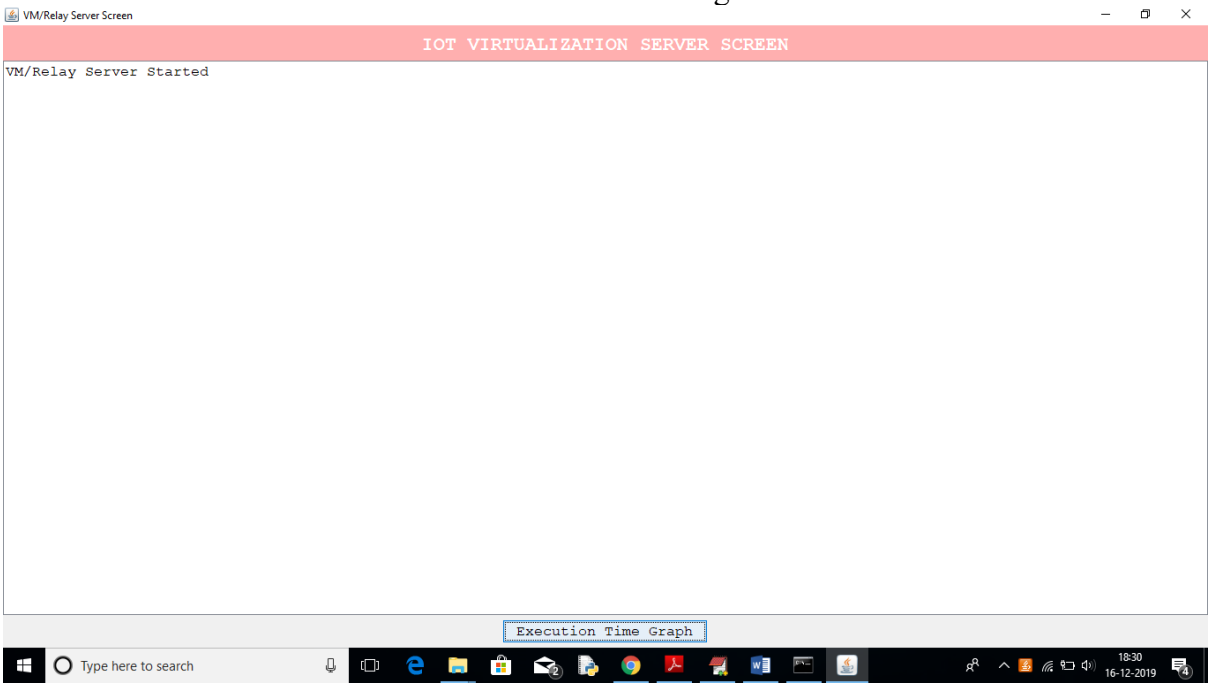

Fig 7.1: Run Virtual Machine

Now double click on 'run.bat' file from 'Simulation' folder to get below screen

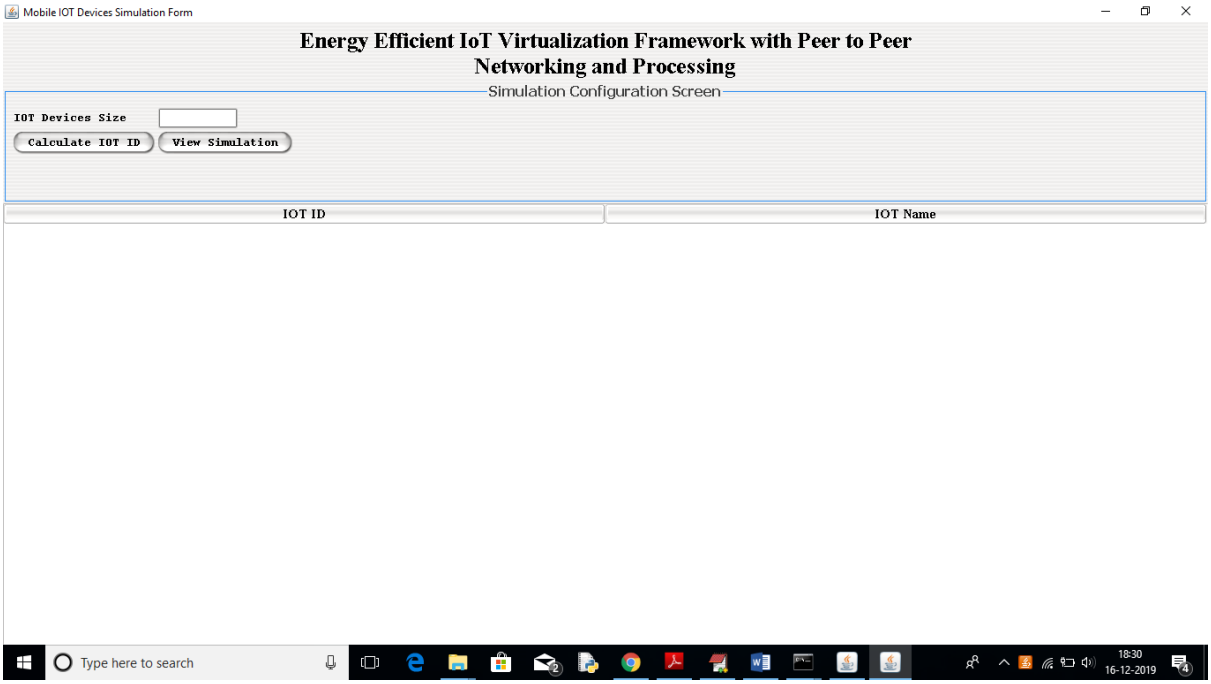

Fig 7.2 Simulation Configuration Screen

In above screen enter IOT Devices Size and then click on 'Calculate IOT ID' button to assign ID to each IOT device

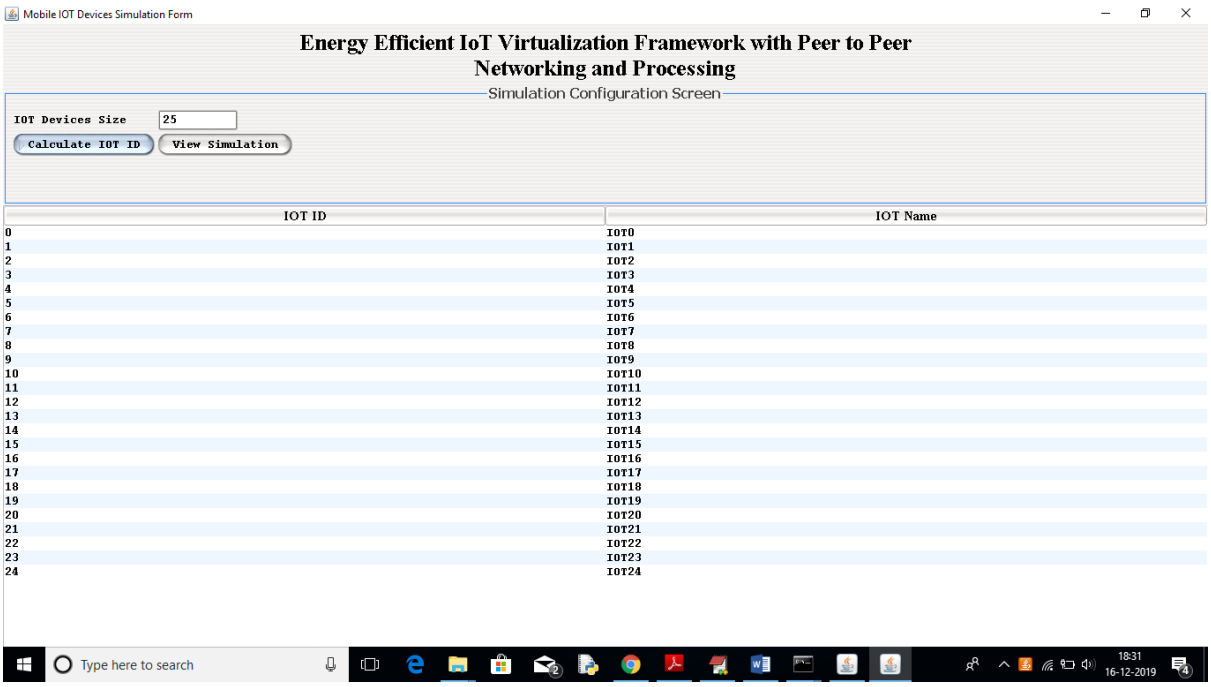

Fig 7.3: Generating IOT Devices

In above screen I entered IOT Device Size as '25' and then click on 'Calculate IOT ID' button to assign ID to each device. Now click on 'View Simulation' button to get below screen

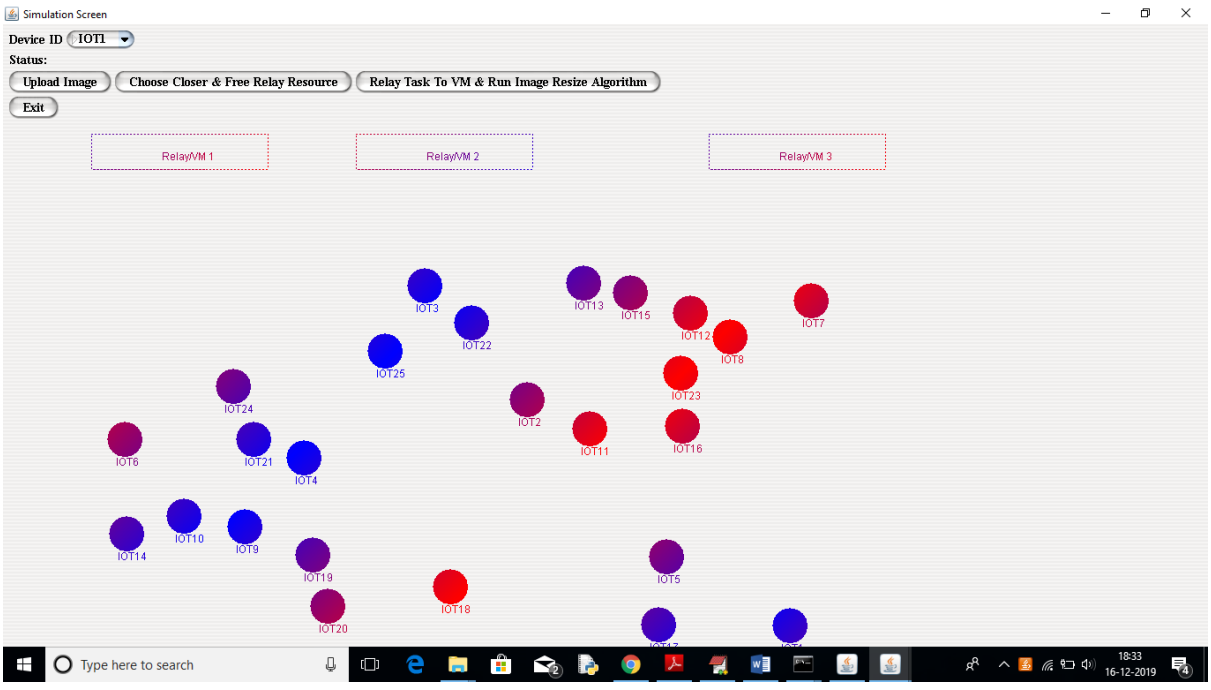

Fig7.4: Selecting IOT Device

In above screen each circle represents as one IoT device and we need to select any device ID from top drop down box and then click on "Upload Image" button to upload image and then this image will send to VM server to resize it

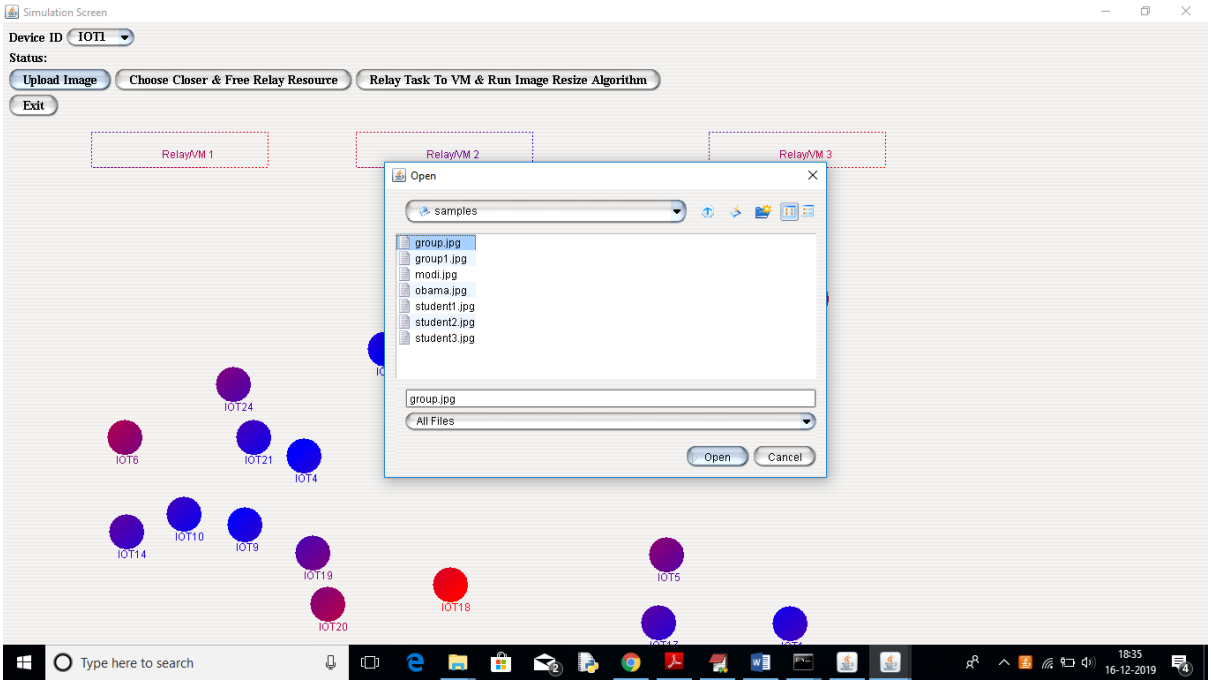

Fig 7.5: Uploading Image

In above screen I selected one image and then click on 'Open' button to load it. After loading image click on 'Choose Closer & Free Relay Resource' button to find free and closer Relay/VM.

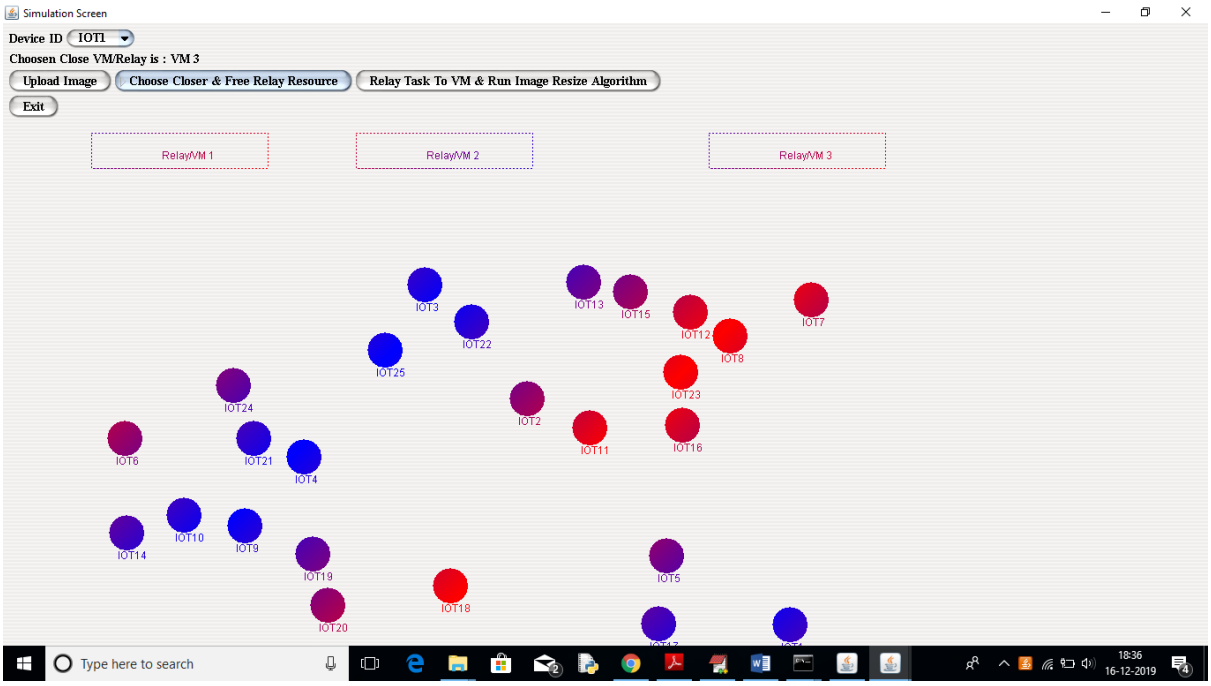

Fig 7.6: Selecting Closest VM

In above screen below Device ID we can see for selected IOT1 device closer VM is VM3. Now click on 'Relay Task to VM & Run Image Resize Algorithm' button to enter new image height and width and then send that task to VM

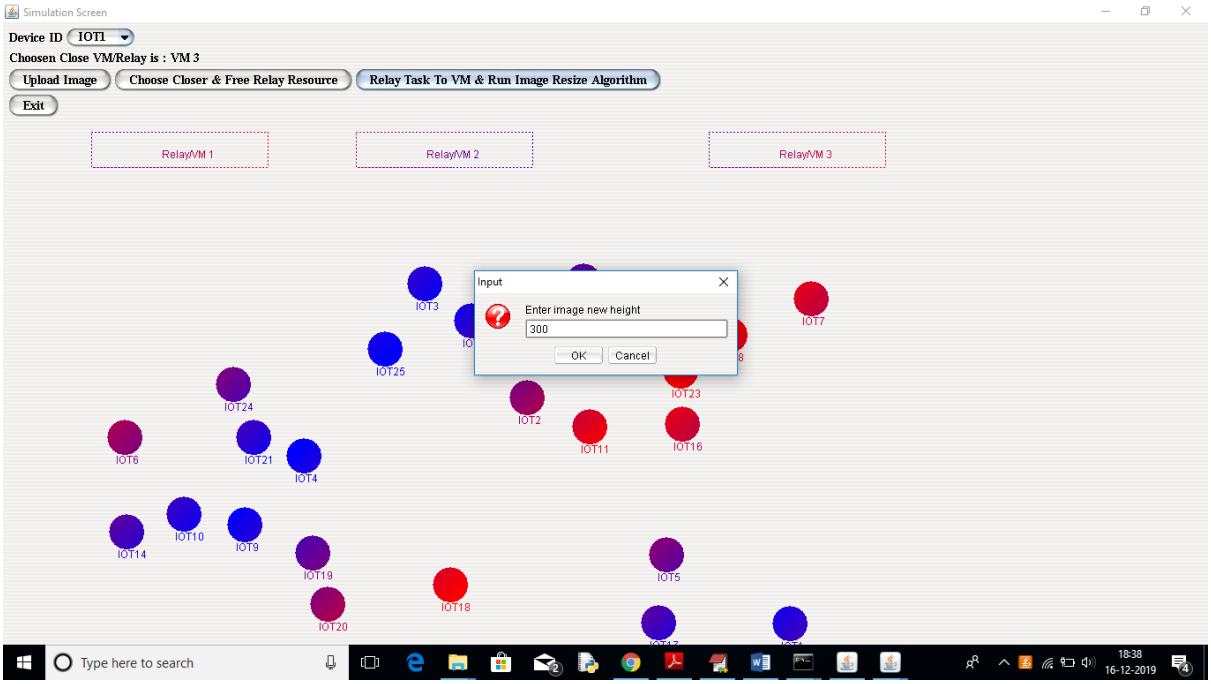

Fig 7.7: Image Configuration

In above screen entering image new height and width as 300 and 300. Now click 'OK' to offload task

Fig 7.8: Data Transferring

In above screen a line between IOT device and VM server indicates data is transferring between them and we get new resize image in below screen

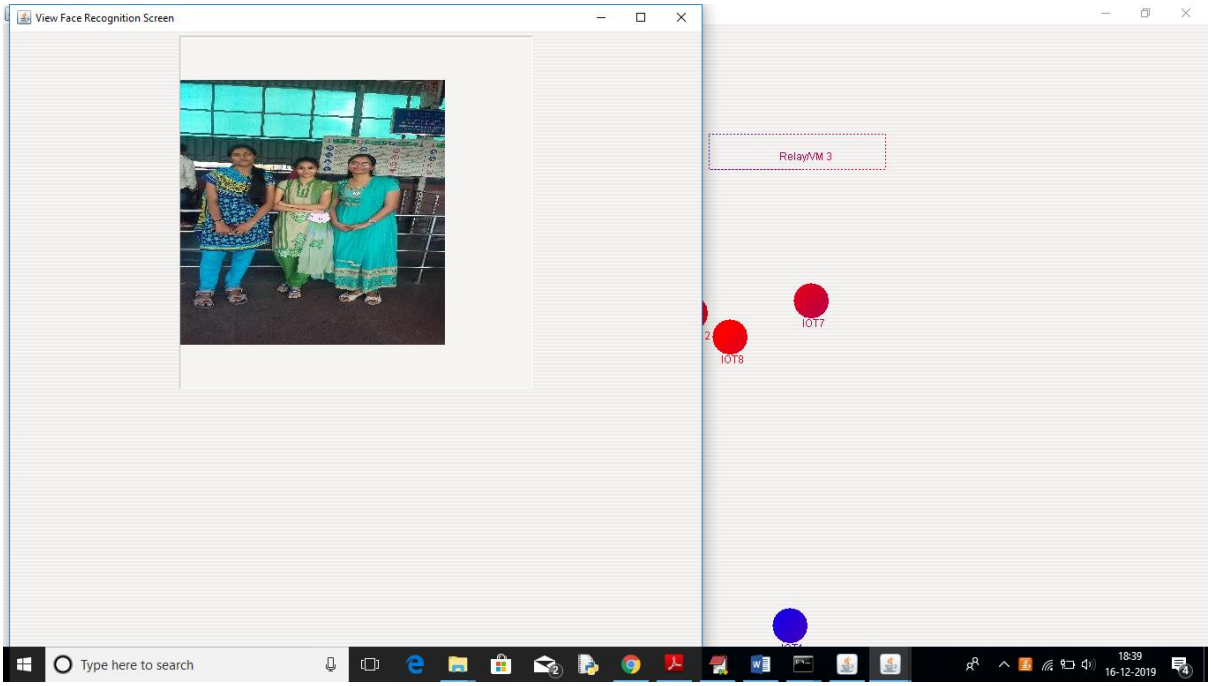

Fig 7.9: Output

Similarly, you can select any IOT device and then upload image and resize to any size and test. Now we can see execution time graph at 'VM' screen below

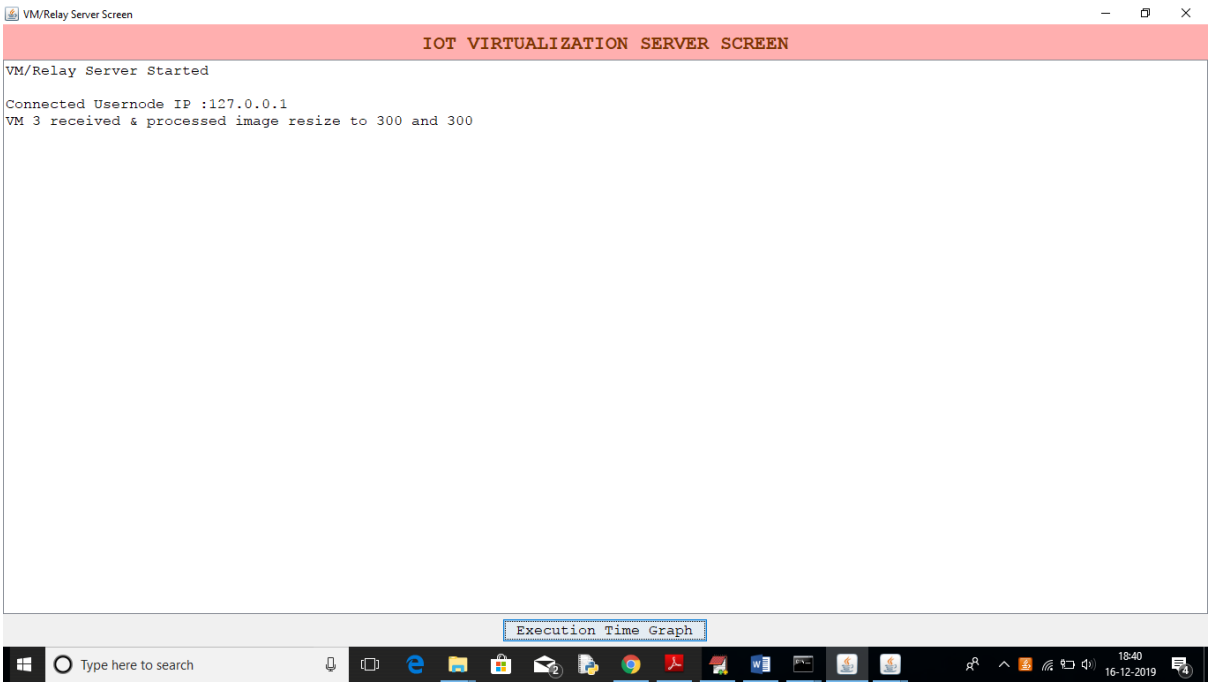

Fig 7.10: Server Requests

In above screen we can see message that VM 3 received request to resize image for 300 and 300. Now click on 'Execution Time Graph' button in above screen to get below graph

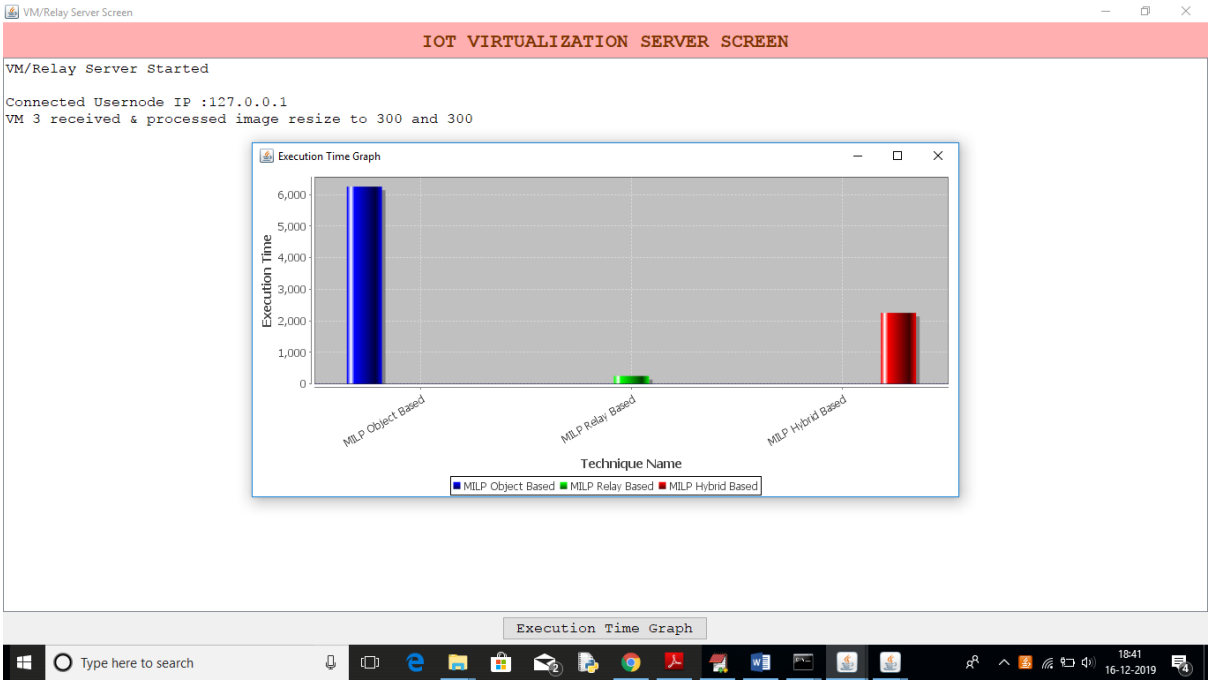

Fig 7.11: Execution Time graph

In above graph x-axis represents technique name and y-axis represents execution time and we can see MILP Relay technique took less execution time and its energy consumption will be less.

8. CONCLUSION

In this project, we have investigated the energy efficiency of an IoT virtualization framework with P2P network and edge computing. This investigation has been carried out by considering three different scenarios. A MILP was developed to maximize the number of processing tasks served by peers and minimize the total power consumption of the network. Our results show that the hybrid scenario serves up to 77% (57% on average) processed task requests, but with higher energy consumption compared with other scenarios. The relays only scenario can serve 74% (57% on average) of the processing task requests with 8% of power saving and 28% (22% on average) of task requests can be successfully handled by applying the objects only scenario with 62% power saving. The results also revealed the low percentage of addressed task requests in the objects only scenario resulting from the capacity limit of the IoT objects' processors. In addition, the small difference between the serving percentage of hybrid scenario and relays only scenario resulted from the allowed internal processing of objects in the hybrid scenario. For real time implementation, we have developed the EEVIPN heuristic based on the MILP model concepts. The heuristic achieved a comparable power efficiency and comparable number of executed tasks to the MILP model. The hybrid Scenario in the heuristic executes up to 74% of the total tasks (MILP 77%), up to 74% of tasks by the relays only scenario (MILP 74%) while the objects only scenario executes up to 21% of the tasks (MILP 28%). It should be noted that due to channel impairment and/or network congestion, link failures may occur, and hence retransmissions may become necessary. These retransmissions can have an impact on power consumption and therefore it is of interest to study the impact of resilience on energy consumption.

9. REFERENCES

[1] F. Ganz, D. Puschmann, P. Barnaghi, and F. Carrez, ``A practical evaluation of information processing and abstraction techniques for the Internet of Things," IEEE Internet Things J., vol. 2, no. 4, pp. 340354, Aug. 2015.

[2] J. Pan and J. McElhannon, ``Future edge cloud and edge computing for Internet of Things applications," IEEE Internet Things J., vol. 5, no. 1, pp. 439-449, Feb. 2018.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, ``Internet of Things: A survey on enabling technologies, protocols, and applications," IEEE Commun. Surveys Tuts., vol. 17, no. 4, pp. 2347-2376, 4th Quart., 2015.

[4] S. H. Shah and I. Yaqoob, ``A survey: Internet of Things (IOT) technologies, applications and challenges," in Proc. IEEE Smart Energy Grid Eng. (SEGE), Oshawa, ON, Canada, Aug. 2016, pp. 381-385.

[5] L. Nonde, T. E. H. El-Gorashi, and J. M. H. Elmirghani, ``Energy ef- cient virtual network embedding for cloud networks," J. Lightw. Technol., vol. 33, no. 9, pp. 1828-1849, May 1, 2015.

[6] A. Q. Lawey, T. E. H. El-Gorashi, and J. M. H. Elmirghani, ``BitTorrent content distribution in optical networks," J. Lightw. Technol., vol. 32, no. 21, pp. 3607-3623, Nov. 1, 2014.

[7] X. Dong, T. E. H. El-Gorashi, and J. M. H. Elmirghani, ``Green IP over WDM networks with data centers," J. Lightw. Technol., vol. 29, no. 12, pp. 1861-1880, Jun. 11, 2011.

[8] X. Dong, T. E. H. El-Gorashi, and J. M. H. Elmirghani, ``On the energy efciency of physical topology design for IP over WDM networks," J. Lightw. Technol., vol. 30, no. 12, pp. 1931-1942, Jun. 1, 2012.

[9] M. Chiang and T. Zhang, ``Fog and IoT: An overview of research opportunities," IEEE Internet Things J., vol. 3, no. 6, pp. 854-864, Dec. 2016.

[10] N. I. Osman, T. El-Gorashi, L. Krug, and J. M. H. Elmirghani, ``Energyef cient future high-denition TV," J. Lightw. Technol., vol. 32, no. 13, pp. 23642381, Jul. 1, 2014.

[11] A.Q. Lawey, T. E. H. El-Gorashi, and J. M. H. Elmirghani, ``Distributed energy efcient clouds over core networks," J. Lightw. Technol., vol. 32, no. 7, pp. 1261-1281, Apr. 1, 2014.

[12] A. M. Al-Salim, A. Q. Lawey, T. E. H. El-Gorashi, and J. M. H. Elmirghani, ``Energy efcient big data networks: Impact of volume and variety," IEEE Trans. Netw. Service Manage., vol. 15, no. 1, pp. 458-474, Mar. 2018.

[13] Z. T. Al-Azez, A. Q. Lawey, T. E. H. El-Gorashi, and J. M. H. Elmirghani, ``Virtualization framework for energy efcient IoT networks," in Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet), Niagara Falls, ON, Canada, Oct. 2015, pp. 7477.

[14] Z. T. Al-Azez, A. Q. Lawey, T. E. H. El-Gorashi, and J. M. H. Elmirghani, ``Energy efcient IoT virtualization framework with passive optical access networks," in Proc. 18th Int. Conf. Transparent Opt. Netw. (ICTON), Trento, Italy, Jul. 2016, pp. 1-4.

[15] F. Jalali, A. Vishwanath, J. de Hoog, and F. Suits, ``Interconnecting Fog computing and microgrids for greening IoT," in Proc. IEEE Innov. Smart Grid Technol.-Asia (ISGT-Asia), Melbourne, VI C, Australia, Nov. 2016, pp. 693-698.