A

Project Report On

TRUE DISTRIBUTED VIRTUAL COMPILER EDITOR

Submitted By

Deepak Mistry	**B80578551**
Deepak Pagar	**B80578557**
Tejas Jadhav	**B80578530**
Sumit Wakde	**B80578583**

Under the guidance of

Prof. S. G. Kamble

In partial fulfillment of

Bachelor of Engineering

[B. E. Information Technology]

[May 2013]

AT

Department of Information Technology

Sinhgad Institute of Technology Science

Narhe, Pune 411041

Affiliated to

University of Pune

Sinhgad Institute of Technology And Science

Department of Information Technology

Narhe, Pune 411041

CERTIFICATE

This is certify that the project entitled **"True Distributed Virtual Compiler Editor",** submitted by **Mr.Deepak Mistry (B80578551), Mr. Deepak Pagar (B80578557), Mr .Tejas Jadhav (B80578530), Mr.Sumit Wakde (B80578583)** are record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Information Technology) at Sinhgad Institute of Technology And Science, Pune under the University of Pune. This work is done during year 2012-2013, under our guidance.

_____

(Prof. S. G. Kamble)

Project Guide

_____ _____

Prof. T.D.Khadtare

HOD, IT Department

Dr. S.N.Mali

Principal SITS

Examination:

Examiner _____

Date:

Acknowledgements

I am profoundly grateful to **Prof. S. G. Kamble** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

I would like to express deepest appreciation towards **Dr. S. N. Mali**, Principal SITS, Pune, **Prof. T. D. Khadtare** HOD Information Technology Department whose invaluable guidance supported me in completing this project.

At last I must express my sincere heartfelt gratitude to all the staff members of Information Technology Department who helped me directly or indirectly during this course of work.

Deepak Mistry

Deepak Pagar

Tejas Jadhav

Sumit Wakde

CONTENTS

Abstract

Distributed computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort.

The aims to describe a compiler which helps to reduce the problem of probability and storage space by making use of the concept of distributed computing. The ability to use different compilers allows a programmer to pick up the fastest or more convenient tool to compile the code and remove the errors.

Moreover an,application can be used remotely throughout any network connection (wired/wireless and it is platform independent .The errors/outputs) of code are more convenient way.

Also,trouble of installing the compiler on each computer is avoided.Thus,these advantages make this application ideal for conducting examinations.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Distributed computing builds on decades of research in virtualization, utility computing, and more recently networking, web and software services.

It implies a service oriented architecture, reduced information technology overhead for the end-user, great flexibility, reduced total cost of ownership and on-demand services among other advantages. The National Institute of Standards and Technology (NIST) defines Distrib ted Computing as a model for enabling easy, on-demand network access to a shared pool of configurable computing resources(e.g.,network ,servers,storage,applications,and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. It does not require the end-user to know the physical location and configuration of the system that provides these services to the end user.

The main disadvantage of distributed computing is the loss of control over the infrastructure used by the users. However, this disadvantage is eclipsed by many an advantages that distributed computing offers. Some of them are lower costs, better computing, location independence, better security. There are five known ways of providing distributed computing currently viz. public, private, community, combined and hybrid computing.

A compiler, which is the heart of any computing system, transforms source code from a higher level language to a lower, machine level language. This is

mainly done in order to create executable files which can then be run in order to execute the program and its instructions. Every compiler primarily consists of three parts viz.

1. The Front end: This checks the semantics and syntax of higher level code (written by the user).Other functions like type checking and error reporting are also performed by the frontend.
2. The middle end: This performs the optimization though removal or useless code, relocation of computation depending on the correct.
3. The backend: This is the part where the translation of the language actually takes place.

1.2 Background

We are providing a centralized compiling scheme, where centralized storage for all the compilers are stored. There is no need to maintain separate compilers or SDKs at client side. User authentic tion and personalized task distribution. i.e. the administrator will be able to assign user-id, password personalized tasks to all the clients. Both the error stream and output stream of the compiler will be captured and the output will be sent to client.

A database of all the codes written by the clients will be maintained. A client may retrieve the stored-code at any later instanc .The administrator will have full authority to compile execute the codes stored by clients for evaluation. All the Statistical details of compilation time, execution time, etc. will be maintained at the server side. Direct comparison of outputs of all the clients can be done at the server side. A client may be assigned a different task every time he/she logs in depending on the will of the administrator.

It is efficient for conducting practical examinations, since every client will be assigned a different login id and password. The administrator may create, edit and delete client profiles anytime.

1.3 Motivation

In order to solve the problem at least partially in the area of programming a software package was developed that allows for interfacing of various compilers. Three compilers: GNU, Microsoft, and Borland (Inprise) are used in the Bradley University intranet. based front end allows to access them without any restrictions regarding the computer system requirements thus allowing for their use on different operating system platforms and also on older machines with lesser performance. Access to selected commercial software components is enabled based on the users computer name and IP address, and also limited by password system.

1.4 Need

The main advantage of distributed computing is of faster processing. Also, many processors can be used remotely, without the knowledge of the user(s), in order to expedite the processing. Thus, keeping this main advantage in mind, the main reason for creating the project is to provide a centralized compiling scheme for organizations or institutions. Also, it will act as a centralized repository for all the codes written. The other major advantage that this system will have over the others is that it will make the users system lightweight i.e. there will be no need to maintain separate compilers/SDKs at the client-side. Thus, for educational institutions this will prove to be highly efficient. Also, the process of maintenance and distribution of dynamic usernames and passwords will be greatly simplified .Also authentication and personalized task distribution will be possible.

Chapter 2

Literature Survey

2.1 Load Balancing Concept.

2.1.1 Description

Application-layer peer-to-peer (P2P) networks are considered to be the most important development for next-generation Internet infrastructure. For these systems to be effective, load balancing among the peers is critical. Most structured P2P systems rely on ID-space partitioning schemes to solve the load imbalance problem and have been known to result in an imbalance factor in the zone sizes. This paper makes two contributions. First, we propose addressing the virtual-server-based load balancing problem systematically using an optimization-based approach and derive an effective algorithm to rearrange loads among the peers.

We demonstrate the superior performance of our proposal in general and its advantages over previous strategies in particular. We also explore other important issues vital to the performance in the virtual server framework, such as the effect of the number of directories employed in the system and the performance ramification of user registration strategies. Second, and perhaps more significantly, we systematically characterize the effect of heterogeneity on load balancing algorithm performance and the conditions in which heterogeneity may be easy or

hard to deal with based on an extensive study of a wide spectrum of load and capacity scenarios.

2.1.2 Related work

The load balance problem for heterogeneous overlay net works has attracted much attention in the research community only recently. Load balance actions are executed by VS reassignment algorithms executed on directory nodes. The performance of M2M is shown to be superior to DM2M.Both algorithms are intuitively appealing, but there are several important issues with the general M2M approach.

2.2 J2SE SDK

2.2.1 Description

(a) Java Platform, Standard Edition or Java SE (formerly known up to version 5.0 as Java 2 Platform, Standard Edition or J2SE), is a collection of Java programming language APIs useful to many Java platform programs.

(b) The Java Platform, Enterprise Edition includes all of the classes in the Java SE, plus a number which are more useful to programs running on servers than on workstations

2.3 Tomcat Apache

(a) Apache Tomcat is a web container developed at the Apache Software Foundation.

(b) Java code to run in co-operation with a web server using Tomcat Apache.

(c) Tomcat includes its own HTTP server internally.

(d) Tomcat is a web server that supports servlets and JSPs.

(e) Tomcat comes with the compiler that compiles JSPs into servlets.

(f) Tomcat can also function as an independent web server.

(g) Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments.

2.4 Object Serialization

- Object Serialization is a concept of converting Java Classes or Java Class Objects into Byte Streams.
- A Byte Stream can be saved as files and can be loaded later.
- This is used to maintain the database of clients, etc.
- Hence we use object serialization to convert Objects into Byte Streams and vice versa.
- It follows a single-threaded programming model, and possesses the following traits:

(a) Platform independence

(b) Extensibility

(c) Component-Oriented:

(d) Customizable

(e) Configurable

(f) Lightweight UI

(g) Loosely-Coupled/MVC

2.5 Process Builder

- This class is used to create operating system processes.
- The start() method creates a new Process instance with those attributes.The start() method can be invoked repeatedly from the same instance to create new sub-processes with identical or related attributes.
- We use process builder throughout the project to call other compilers from our java program.

- A command-commands of O.S is identified by process builder. Execution of that command builder invokes related files and arguments of that command.
- An environment, which is a system-dependent mapping from variables to values. The initial value is a copy of the environment of the current process.
- Modifying a process builder's attributes will affect processes subsequently started by that object's start() method, but will never affect previously started processes or the Java process itself.

2.6 Advantages

(a) Distributed computing involves several computers on a network working together.

(b) Java is designed to make distributed computing easy with the networking capability that is inherently integrated into it.

(c) Writing network programs in Java is like sending and receiving data to and from a file.

2.7 Disadvantages

(a) Applications using the java tend to use much of the system resources .

(b) Also, loss of trade secrets and by passing of license is the major problem caused by reverse engineering.

(c) Regular garbage check and collection makes the application pause for sometime from execution.

Chapter 3

Proposed Work

3.1 Problem Statement

To develop a project that helps avoid having various types of compilers in a system. The system helps us compile programs coded in any languages irrespective of the compiler being available in our system.

3.2 Features

- To provide a centralized compiling scheme for an organization or institution.
- Centralized storage for all the codes written.
- No need to maintain separate compilers or SDKs at client side.
- User authentication and personalized task distribution. i.e. the administrator will be able to assign user-id, password personalized tasks to all the clients.
- The codes will be compiled centrally and the results will be displayed at client-side application.
- Both the error stream and output stream of the compiler will be captured and the output will be sent to client.
- Servlets will be used for client-server communication.
- A database of all the codes written by the clients will be maintained.

- A client may retrieve the stored-code at any later instance.
- The administrator will have full authority to compile execute the codes stored by clients for evaluation.
- Statistical details of compilation time, execution time, etc. will be maintained at the server side.
- Extremely efficient for educational institutions since maintenance of compilers needs to be done at only server side.
- Direct comparison of outputs of all the clients can be done at the server side.
- A client may be assigned a different task every time he/she logs in depending on the will of the administrator.
- Efficient for conducting practical examinations, since every client will be assigned a different log in id and password.
- The administrator may create, edit and delete client profiles anytime.

3.3 Scope

- **General:** Scope of the project is not just limited to client side compiling but rather centralized compiling at the server side along with processing-task transmission.
- **Extended :** Instead of using several high speed costly servers we can use one single ultra high speed server and simple low end cheaper clients.

3.4 Goals

- To improvise efficiency of the system by using optimized algorithms available.
- To have a system that consumes less time to process a complex task
- To Provide Server-side processing means users can quickly access single pages rather than downloading entire documents resulting in faster access times and reduced network traffic.

- Administrators no longer need to install or track users versions of software or update multiple applications on every client.

3.5 Objectives

- **Main Purpose:** The purpose of the project was to allow students to get familiar with different compilers and compiler optimization techniques rather than make another huge GUI application to wrap compilers.
- **Main Use :** The major use of the Distributed Compiler is in conducting university practical examinations through our project. A separate cloud would be constructed at the university location that is the server.

3.6 Constraints

- Every student would in advance have a user id and password to log in and start with the examination.
- There would be a URL provided to the students which would lead them directly to the login page of the application.

3.7 Applications

- **Conduct College Practical's:** Efficient for conducting practical examinations, since every client will be assigned a different log in id and password.Extremely efficient for educational institutions since maintenance of compilers needs to be done at only server side.
- **Centralized Code Database Servers:** Centralised storage for all the codes written.A database of all the codes written by the clients will be maintained.A client may retrieve the stored-code at any later instance.

Chapter 4

Project Design

4.1 Hardware and Software Requirements

4.1.1 Hardware Requirements

- Processor: Processor equal to 2.2 GHz.
- Memory: 256 GB RAM or more.
- Input:Keyboard,Mouse,PCs.,Lan wire,Switch(wired connection).

4.1.2 Software Requirements

- Java JDK (J2SE).
- Tomcat Apache Server.
- NetBeans IDE .

The architecture shown below,describes how the different components of the sytem interact and there working collaboratively to achieve the desired functionality of the system.

The system mainly consists of Client,Central Server,Compiling Server ,and the GUI which shows how all these components interect with each other.

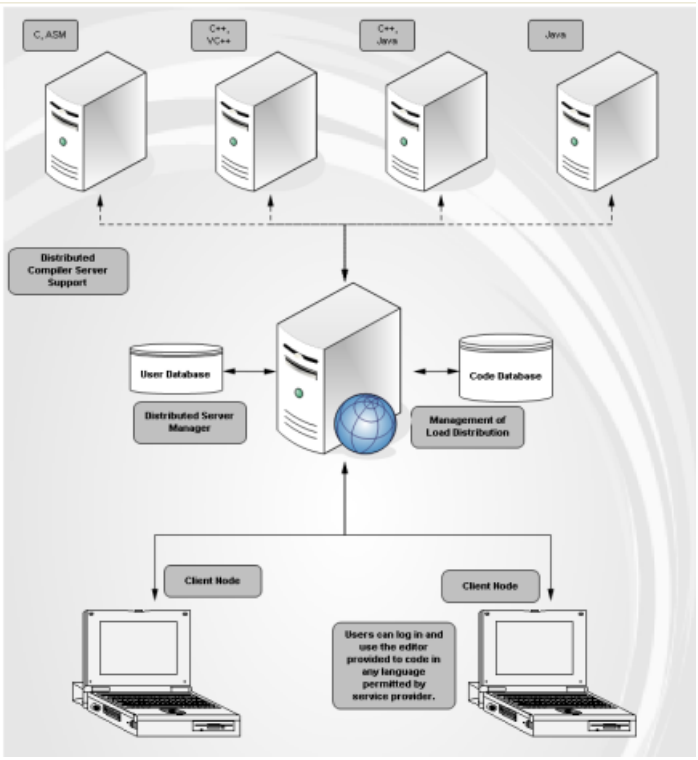

Figure 4.1: System Architecutre

Client side programs are applets which are transferred to the client machine entirely on the request of the user. They are executed only on the client machine and not on the server.This allows for sharing the computational cost between the server and client.This approach is used when program to be transferred to users is moderate in size, is cached on client machine or the data to be transferred between server and client in case the application is run on the server is very large in volume.

In case of platform independent solutions, such as Java,causing lesser computational performance may be prohibitive. With the distributed Compiler, much less information has to be passed on, to the server. The server executes instructions based on the given information and sends the results back to the local machine that made the initial request. This is used when the software package is large or is not to be released to user, or when amount of data to be transferred is small.

However, large number of clients that access the server simultaneously would make CGI- based approach undesired. These

software design problems were considered and solved in the Distributed Compiler. The user interface is programmed in java language using servlets. It was assumed that the user will use his or her favourite text editor to create and correct program files. This assumption allowed to create a very simple front-end that that loads quickly and is platform independent.

Although the front end is designed to be as simple as possible with only a few commonly used options, it is sufficiently functional and can be used quickly. The server side part of the application is implemented using java servlets that handles the communication between a user and compiler.The script does the file managing, runs compilers and processes the compilation results. The result is the source code listing or a list of errors sent back to the user.

4.2 Functional Analysis of Different Modules

- **Client Side GUI :** To be developed using SWING API. This module will provide user with forms to interact with main server.
- **Server Side GUI :** Again developed using SWING API, this module will allow administrator to login and interact with server database.
- **Servlets :** These java codes will be written for actual client-server communication. These are stored at server side and are accessed using server address URLs. Tomcat is needed in order to call these servlets from client slide modules. These will be written for following sub modules a. fetch code
  b. submit code
  c. compile code
  d. execute code
  e. login

- **Server Database :** The database at server side is created and maintained using Java Object Serialization. The data libraries will be defined for following sub-modules a. User ID's, tasks, password, permissions, etc. and
  b. Code, Filename, Class name, etc.
  c. Log, Time, User, Activity, etc

- **Client Side Sub Modules :**1. Editor: Implemented using Java I/O this module provides user with facilities to save/load/reload/edit source code.
  2. Server Calls: These modules are implemented to pass current parameters to server and receive response from server so that it can be presented to the user.
  3. Login: Initial login module to authenticate user.
  4. External Editor: This module is provided so that user can edit/mange code in an external compiler. This module is to be implemented using ProcessBuilder (java feature)

- **Server Side Sub Modules :**1. User ID Management: Module wherein administrator can create/edit/manage user id's / tasks / passwords etc.
  2. Code Review: Administrator can review all the code submitted by every user.
  3. Code Compiler: A module wherein administrator can compiler and view the compilation results of submitted code.
  4. Execution: Similar module like 3 in order to execute the code instead of compiling it.
  5. Log: A module for administrator to view all the log/activities of clients.

4.3 System Specifications

4.3.1 Use Case diagram

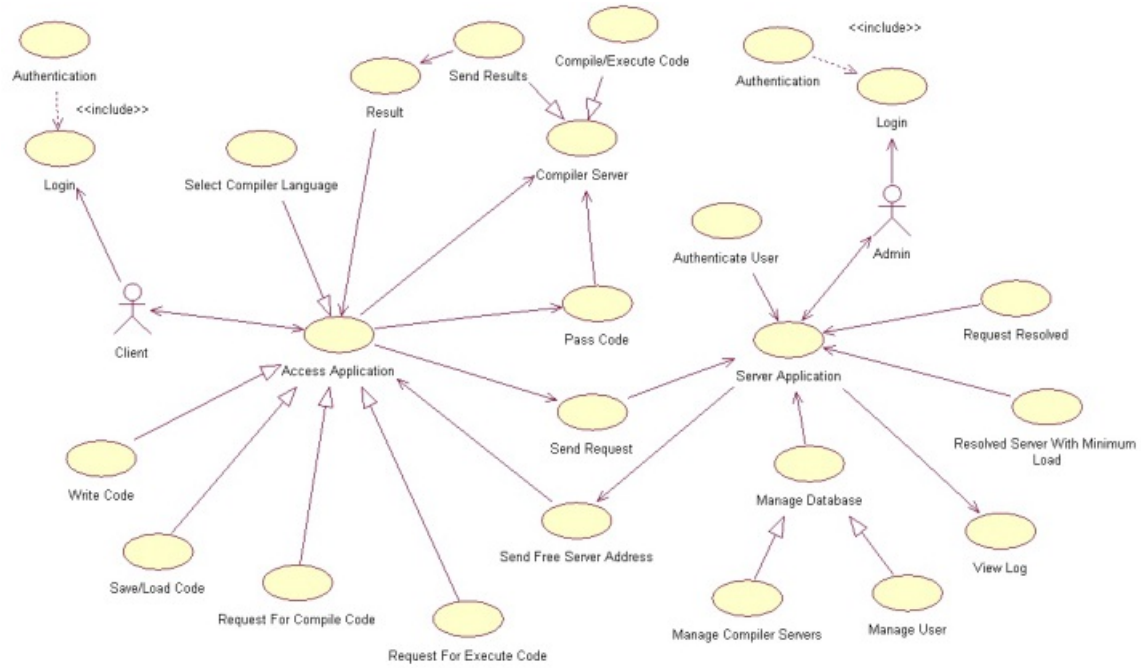

Figure 4.2: Use Case Diagram

Use Case diagram are set of use cases actors and their relationship. They represent the use case view of a system. In above case we have shown the different functionality that user can have with the system.

4.3.2 Class Diagram

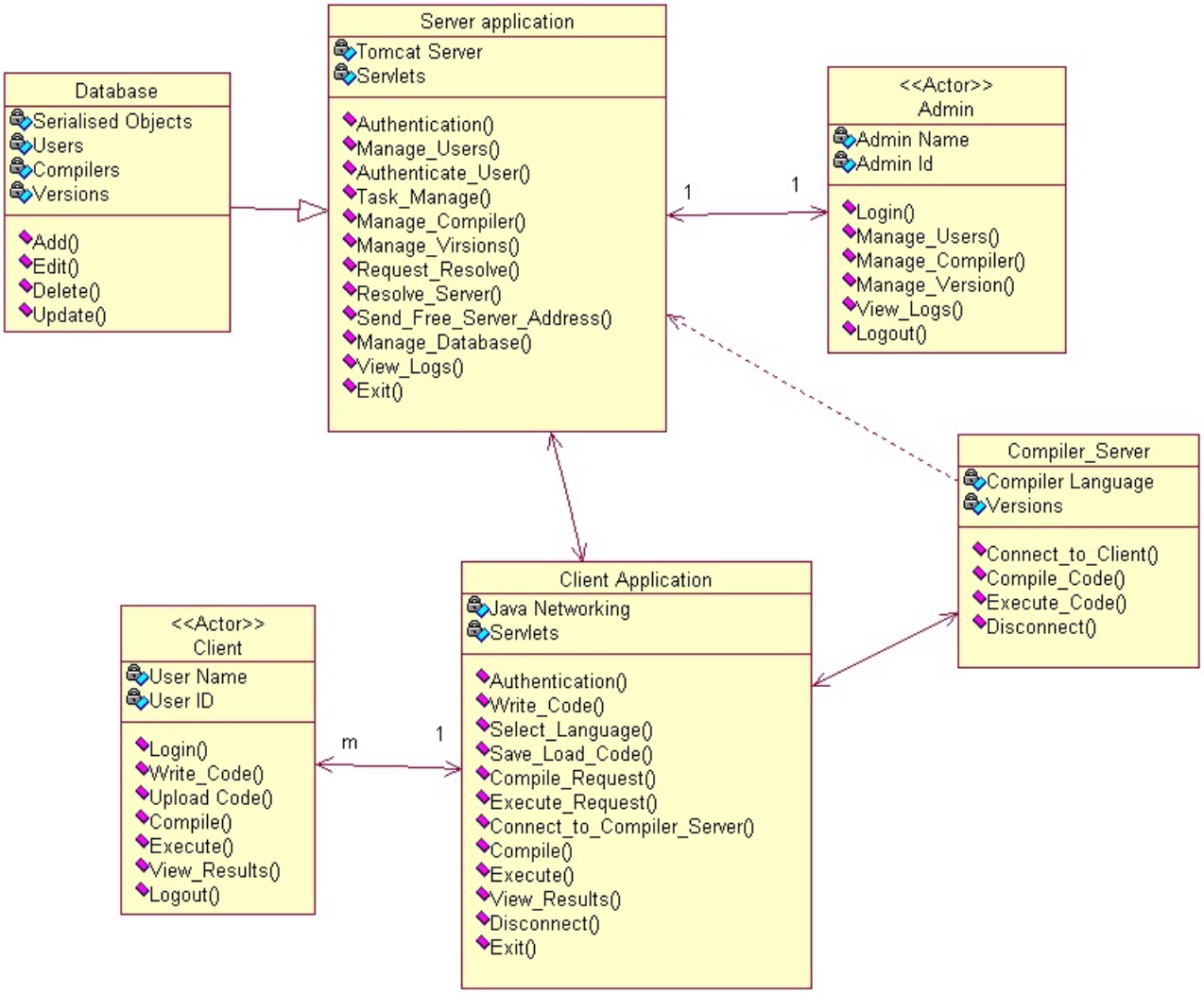

Figure 4.3: Class Diagram

Class diagram consist of classes, interfaces, association and collaboration. Class diagram basically represent the object oriented view of a system which is static in nature.

4.3.3 Activity Diagram

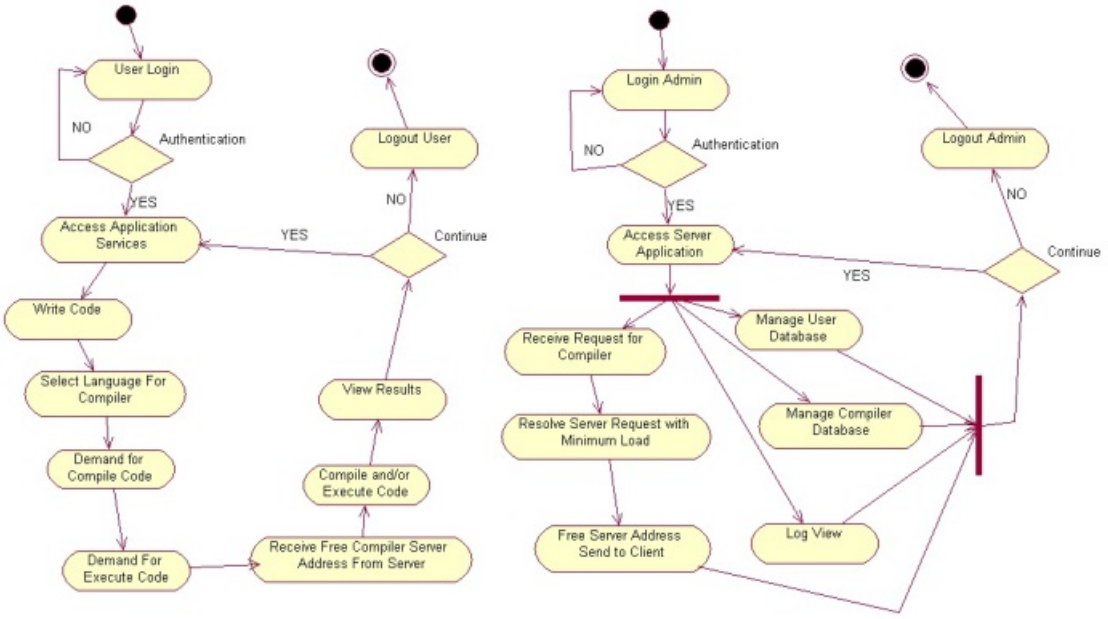

Figure 4.4: Activity Diagram

Activity diagram describes the flow of control in a system. So it consists of activities and links. The flow can be sequential, concurrent and branch. In this diagram we have shown how the activities are performed during execution.

4.3.4 Sequence Diagram

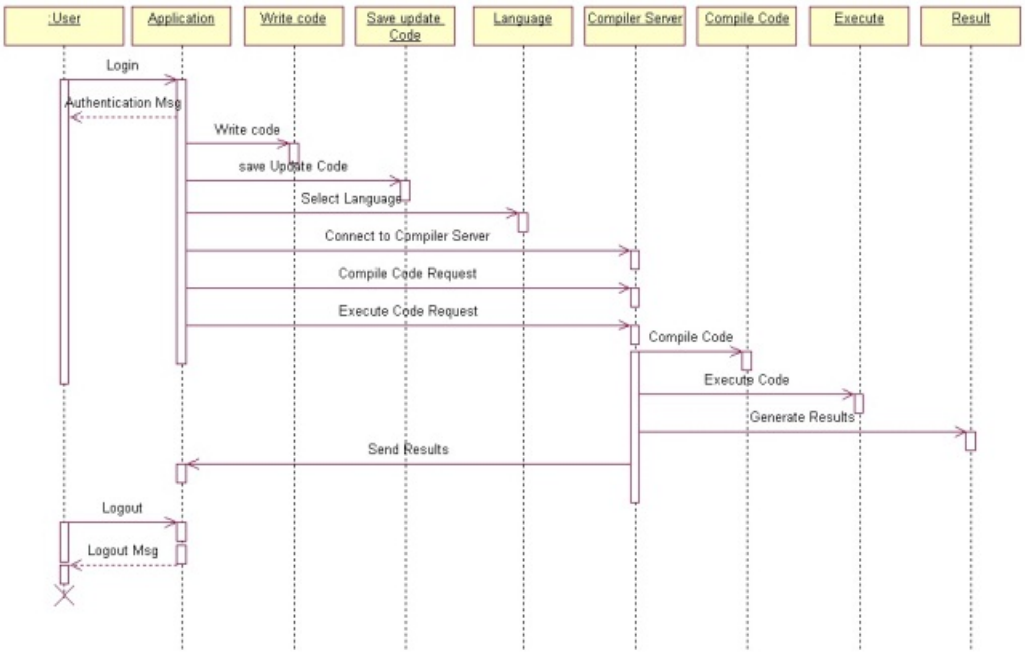

Figure 4.5: Sequence Diagram for Client Side

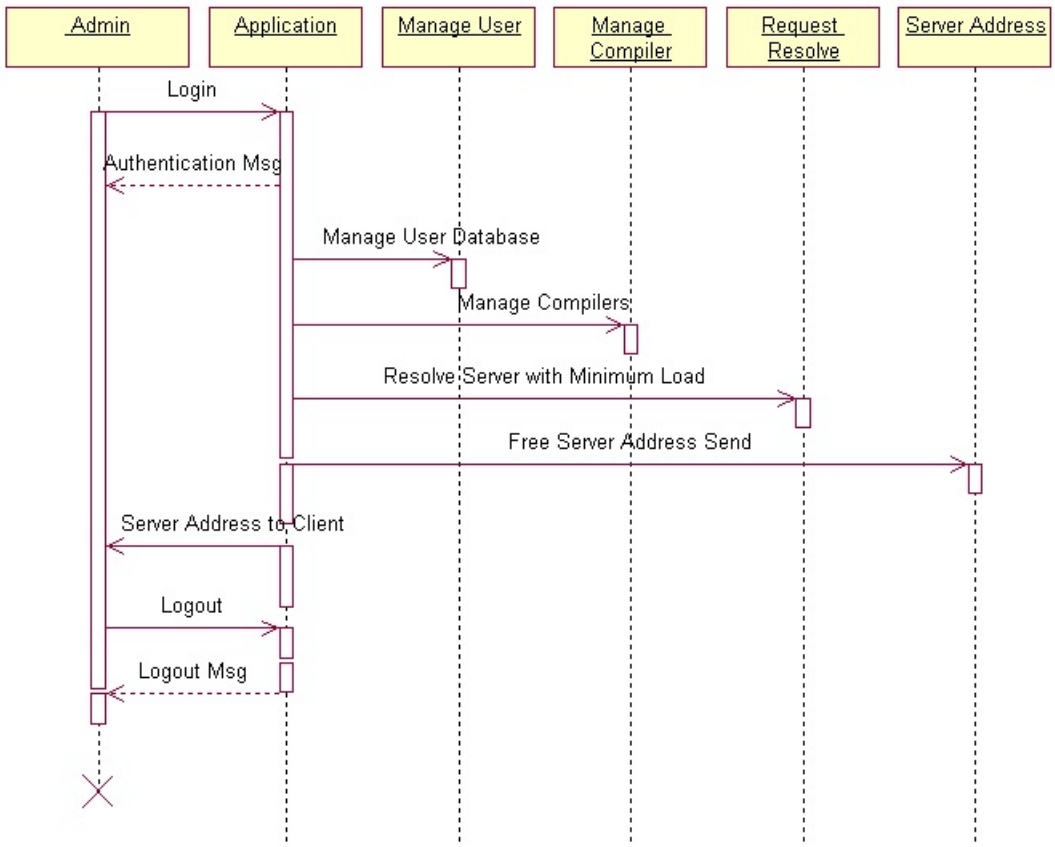

Figure 4.6: Sequence Diagram for Server Side

Sequence diagram is an interaction diagram in which we show some sequences, which are the sequence of message flowing from one object to another. Interaction among the components is very important from implementation and execution perspective.

So sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality.

4.3.5 Communication Diagram

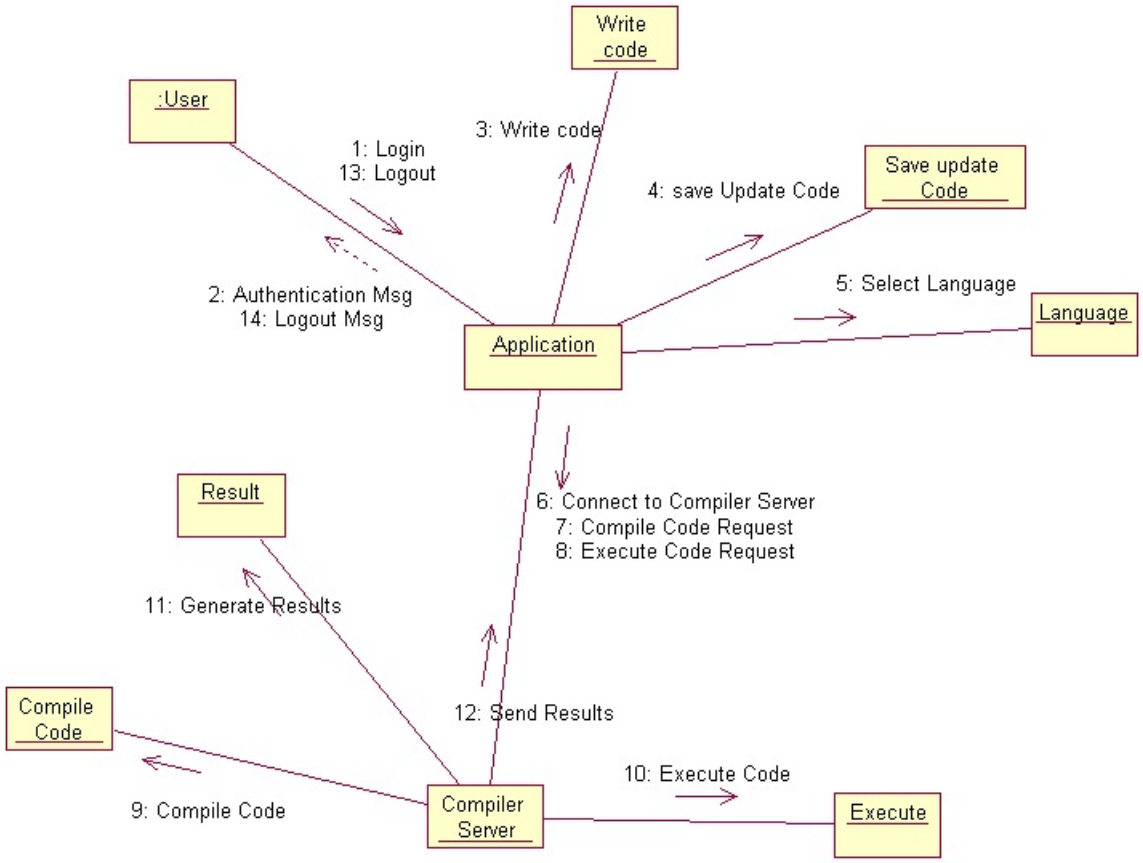

Figure 4.7: Communication Diagram for Client Side

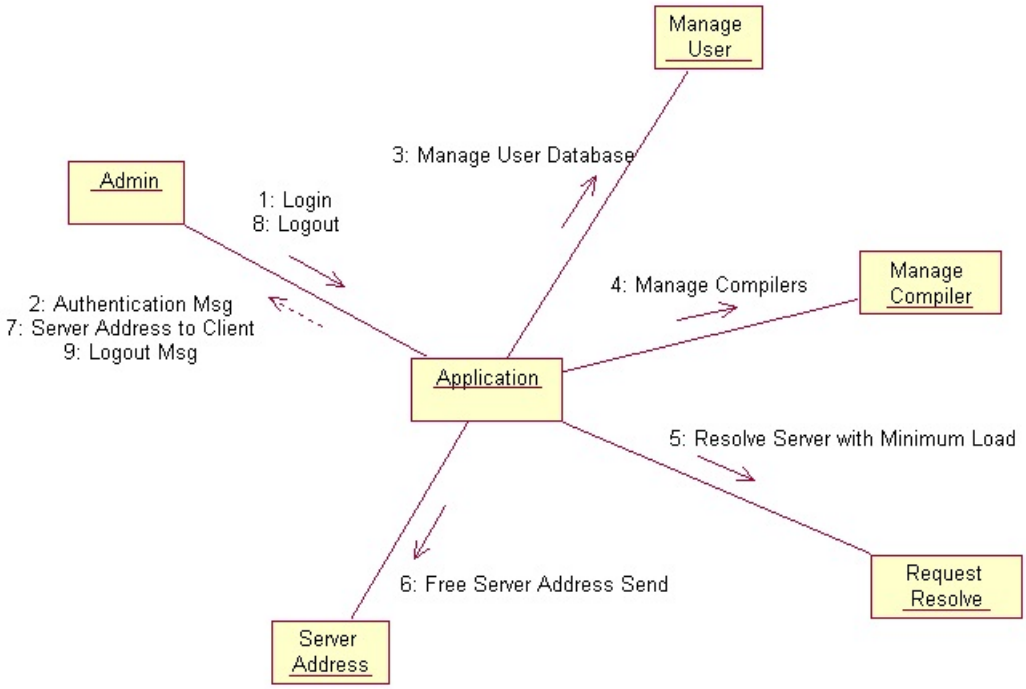

Figure 4.8: Communication Diagram for Server Side

Communication Diagram represents the structural organization of a system and show objects and how messages are send between the linked objects and thereby imply their relations.

Communication diagram add another perspective to an interaction by focusing on the links between the participants. They are specially good at showing which links are needed between participants to pass an interaction messages.In this diagram we have visualize the organization of objects and their interaction.

4.3.6 State Diagram

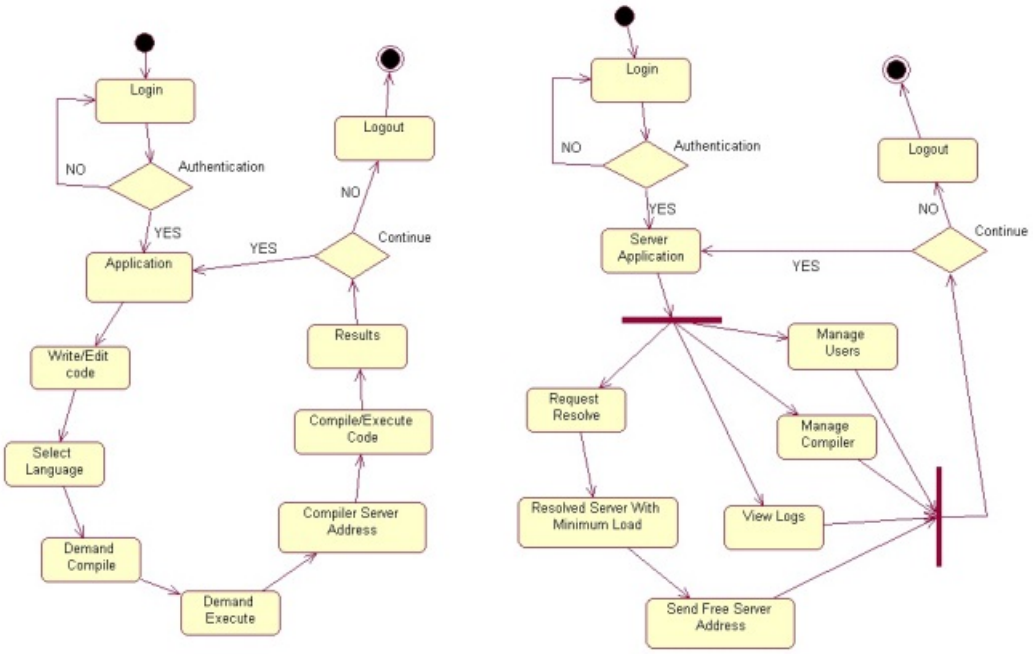

Figure 4.9: State Diagram

State Diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface, etc. It capture the behavior of the system. They are used in special position of software and hardware systems like, Real-time,mission critical systems,etc.

In this we have visualize the reaction of a system by internal/external factors.

4.3.7 Package Diagram

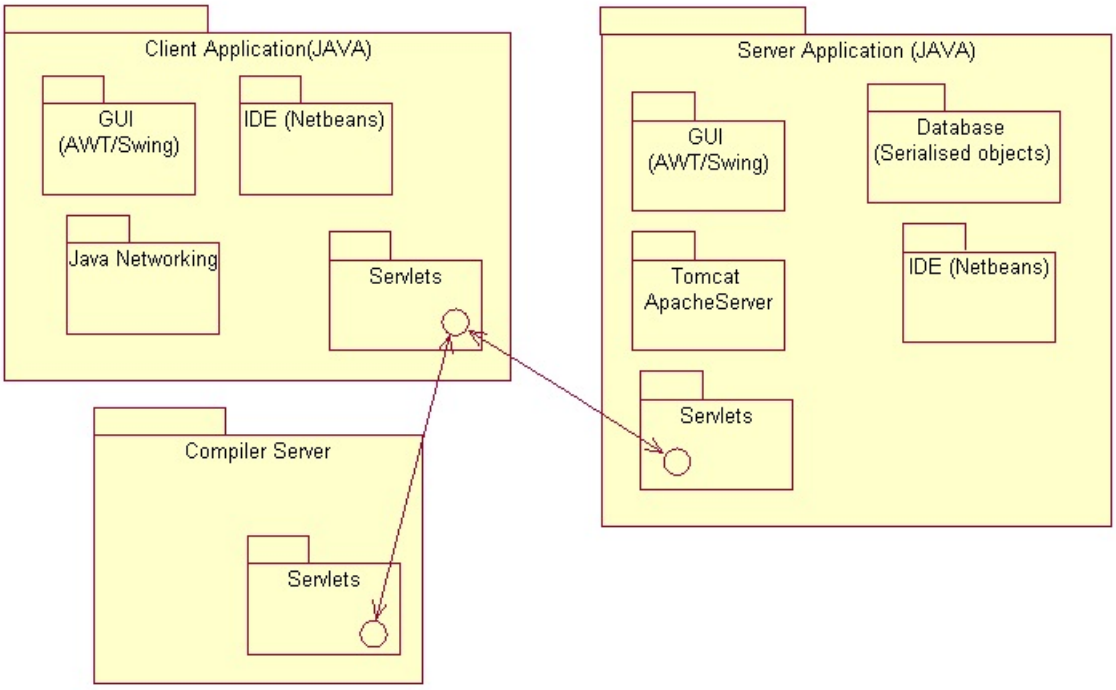

Figure 4.10: Package Diagram

A package diagram in the Unified Modeling Language depicts the dependencies between the packages that make up a model.

Package diagrams can use packages that represent the different layers of a software system to illustrate the layered architecture of a software system. The dependencies between these packages can be adorned with labels / stereotypes to indicate the communication mechanism between the layers.

4.3.8 Component Diagram

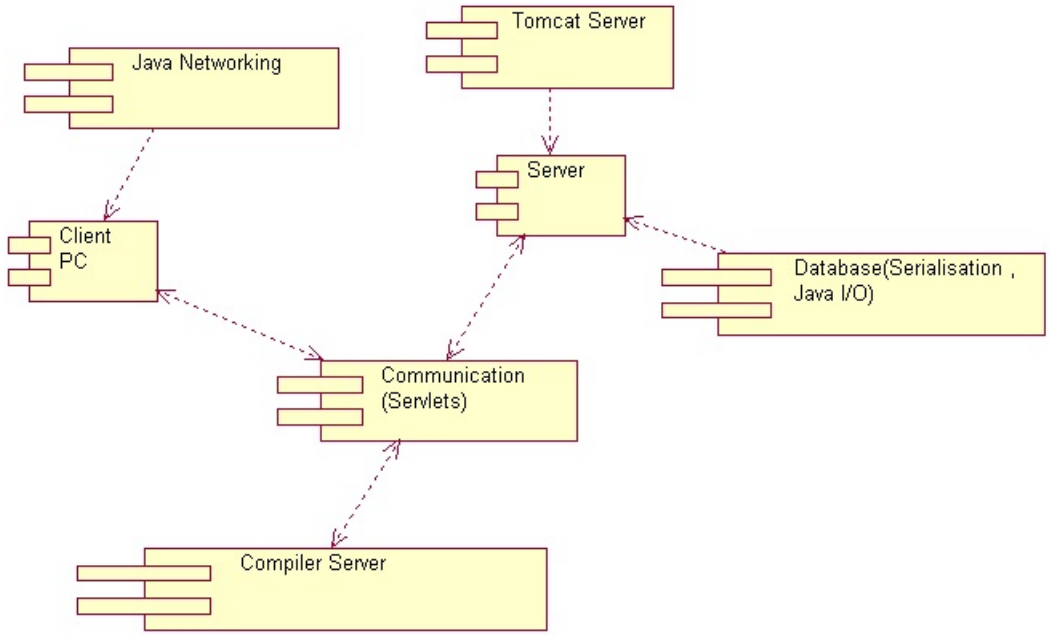

Figure 4.11: State Diagram

Component diagrams illustrate the pieces of software, embedded controllers, etc., that will make up a system. A component diagram has a higher level of abstraction than a Class Diagram - usually a component is implemented by one or more classes (or objects) at runtime.

The difference between package diagrams and component diagrams is that Component Diagrams offer a more semantically rich grouping mechanism. With component diagrams all of the model elements are private, whereas package diagrams only display public items.

4.3.9 Deployment Diagram

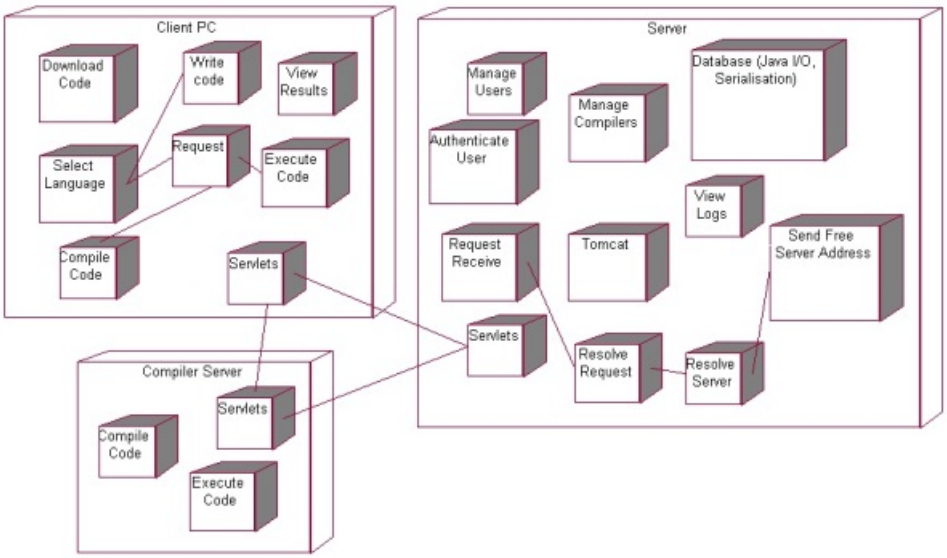

Figure 4.12: Deployment Diagram

Deployment diagram are set of nodes and their relationships. This nodes are physical entities where the components are deployed. In this we have visualize the deployment view of a system.

4.3.10 Data Flow Diagram (DFD)

Data Flow Diagram is a graphical representation that depicts information flow and the transformation that are applied as data flow from input to output.

DFD Level 0

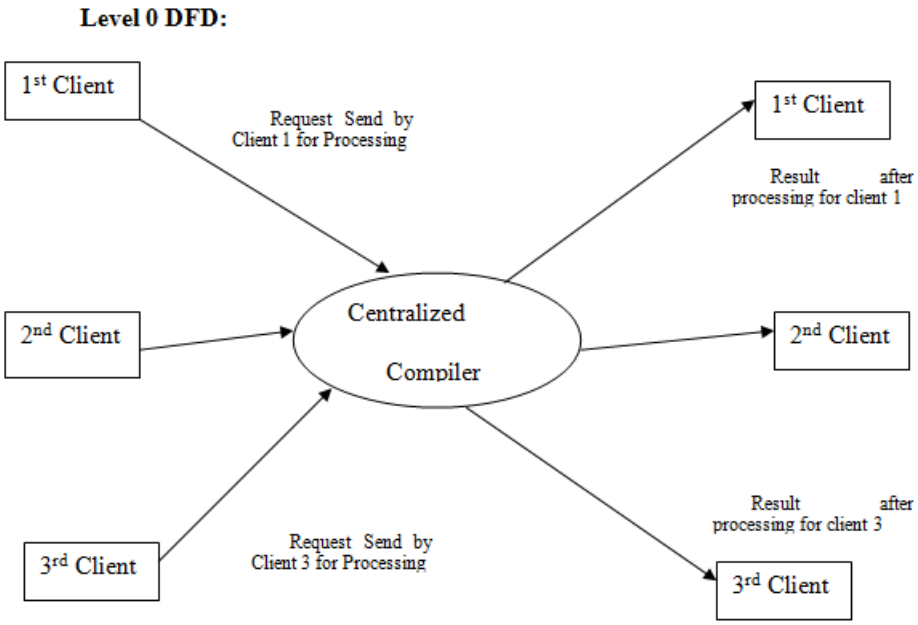

Figure 4.13: Data Flow Diagram (Level-0)

This is called as Fundamental/Context level DFD. It represents the entire software element as a single bubble with input and output data.

DFD Level 1

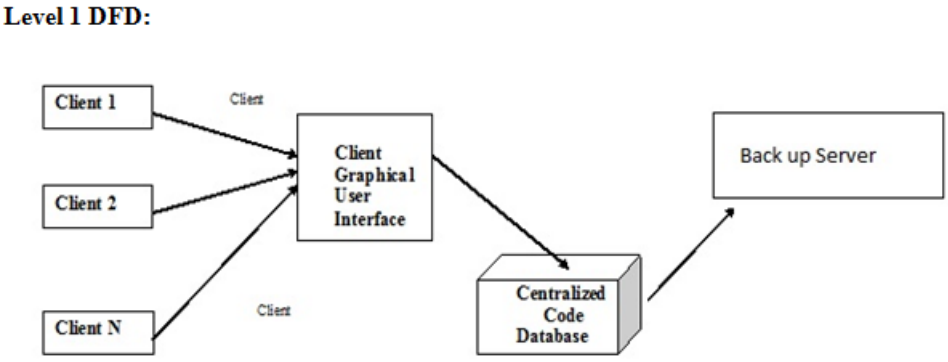

Figure 4.14: Data Flow Diagram (Level-1)

In this level there is a detail description of the software where the entire software is represented by 2/3 or more bubbles.

DFD Level 1

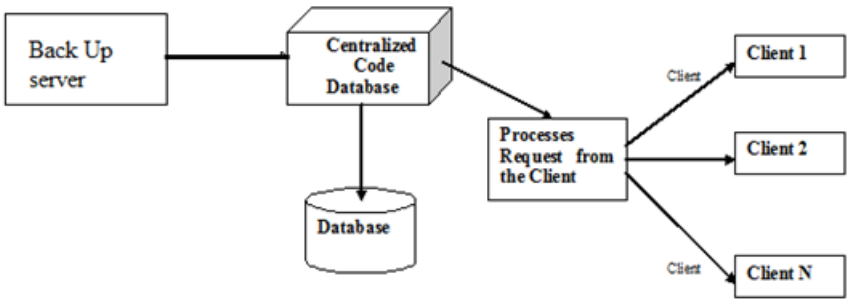

Figure 4.15: Data Flow Diagram (Level-1)

Chapter 5

Schedule of Work

5.1 Plan of Execution

- Identification : Searching for different project ideas. Identifying and finalizing one of them for further implementation.
- Conceptualization and design : The concepts required for building the project are studied in detail. Also the high level designing is done at this stage. Preliminary presentation is given for more clarification of project.
- Detailed design : At this stage, low-level designing is done. The overall architecture of the project was finalized and design decomposition was done.
- Coding : Actual implementation of the project starts at this stage. Coding for each of the modules will be done. Coding and testing will take approximately 10 to 12 weeks.
- Unit Testing : Initially the application will be tested over a number use cases. The individual modules were tested for the correct functionality.
- Integration Testing : All modules will be integrated and then testing of whole integrated component will be performed.
- System Testing : The product was tested in the context of the entire system. Different hardware platforms will be used for system testing and the performance will be monitored.

- Documentation : A detailed document about the project shall be prepared at this stage.

Requirement Gathering and Plan for the initial part of the project was as follows:

- Understanding the problem definition.
- Understanding the current scenario of the system.
- Gathering information about required Software Resources.
- Gathering information about required Hardware Resources.
- Preparing preliminary design of overall workflow of project.
- Deciding the modules required for overall execution

5.2 Software Risk Management

5.2.1 Introduction

Risk is a possibility of loss or injury. The definition of risk in the software engineering environment that we will use is exposure to harm or loss, as this includes not only the possibility of risks, but their impact as well. Using risk management techniques, we alleviate the harm or loss in a software project. All risk cannot be avoided, but by performing risk management, we can attempt to ensure that the right risks are taken at the right time. Risk taking is essential to progress and failure is often a key part of learning. Risk management involves Risk Identification, Risk Analysis and Risk Prioritization.

5.2.2 Risk Identification

Our development team identified some potential risks to the project. These risks were analyzed and were classified into various categories depending upon the threat they posed to the project. Some of these risks were generic risks while others were product specific risks. A

considerable amount of time was spent in analyzing the product specific risks.

Risks	Probability	Impact
Machine Failure	30%	Critical
Wrong estimate of size and effort	40%	Critical
Overly optimistic schedule	30%	Critical
Error prone modules require more testing and implementation work	30%	Critical
Integrating new code with existing code	35%	Critical
Requirements Change	45%	Critical
Staff size is small and inexperienced	10%	Marginal
Technology to be built is unexplored	20%	Marginal
Project scope is vast with limited time	20%	Marginal

Table 5.1: Risk Table

5.2.3 Risk Analysis

We analyzed all the risks individually and we came up with the classification of risks on the basis of their impact on project schedule. These risks were rated as follows:

- Catastrophic.
- Critical.
- Marginal.
- Negligible.

These were then reviewed and consensus on the impact was reached.

5.2.4 Risk Prioritization

After analyzing the risks, the risks were prioritized based on the review and consensus among the developers.

5.2.5 Overview of Risk Mitigation, Monitoring, Management

Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague

a software project. A risk is a potential problem it might happen, it might not. But regardless of outcome, its a really good idea to identify it, access its probability of occurrence and estimate its impact.

For any project being developed, it is empirical for the team to monitor and manage risks. Risks are the unexpected delays and hindrances that are faced by software team and need to be actively managed for rapid development.

The key functions of software risk management are to identify, address and eliminate sources of risk before they become threats to successful completion of a software project.
An effective strategy must address three issues:

- Risk avoidance.
- Risk monitoring.
- Risk management and contingency planning.

5.3 Project Schedule

5.3.1 Functional decomposition

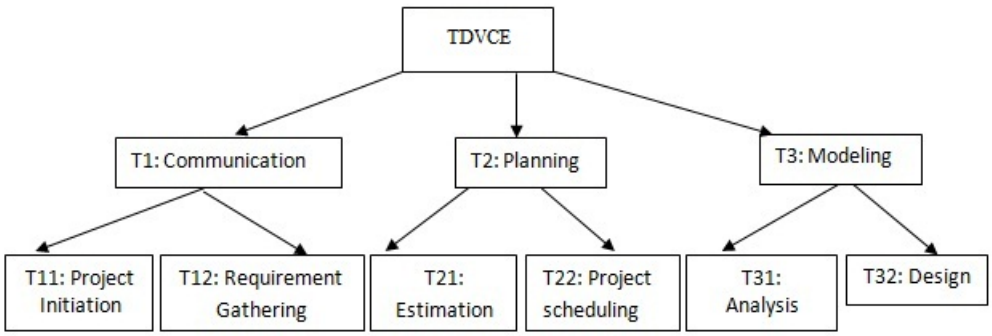

Figure 5.1: Functional Decompostion

The different task sets that are critical for the project are determined. There are six tasks in all leading towards the development of our project. The breakdown of the major functional task set is as follows:

- T1: Communication
  Software development process starts with the communication between customer and developer. According to need of project, we gathered the requirements related to project.

- T2: Planning
  In the planning phase, we have established the cost, schedule, list of deliverables and delivery Dates. Scope of Planning specifies the in-scope requirements for the project. Budget Planning specifies the budgeted cost to be incurred in the completion of the Project.

- T3: Modeling
  It includes detailed requirement analysis and project design.

- T11: Project Initiation
  The project initiation phase involves the construction of the server side application and the client side application also setting up of the protocols for the communication between the client and server.

- T12: Requirement Gathering
  In the requirement gathering phase, we generally noted down the basic requirements of the customers and what kind of services they generally require and understood all requirements by discussing them in detail.

- T21: Estimation
  In the estimation phase we have generally found the estimation of the projects total life cycle and scope of the project. The estimation phase is useful for the estimation of the total cost required for the

project.

- T22: Project Scheduling
  In this phase we consider the scheduling of the resources, an effort estimate for each task. The reason for this is that a schedule itself is an estimate: each date in the schedule is estimated, and if those dates do not have the buy-in of the people who are going to do the work, the schedule will be inaccurate.

- T31: Analysis
  In this phase we have studied the estimation of all the costs or problems of a problems before work on it is started.

- T32: Design
  In this phase we started working with the actual coding of the project and the project development. Also the project implementation and delivery and deployment of the project was done after this phase.

5.3.2 Project Schedule

The plan of execution will be as follows:

Sr.No.	Activity	Estimate completion Data
01.	Different ideas for project	August 2012
02.	Topic Finalisation	August 2012
03.	UML's and DFD diagrams	September 2012
04.	Study of various platforms	October 2012
05.	Platform finalisation	October 2012
06.	Platform learning	December 2012
07.	Development of GUI	January 2013
08.	Development of Modules 1-2 And their unit testing	Before February 2013
09.	Development of Modules 3-4 And their unit testing	Before March 2013
10.	System Testing	April 2013
11.	Modifications	April 2013
12.	Finalisation	May 2013
13.	Documentation	before 1st week of June 2013

Table 5.2: Project Schedule

5.4 Staff Organisation

5.4.1 Team Structure

The team structure for the project is identified. Roles are defined. We have decided to keep the team structure of 4 highly flexible throughout the project. Each individual shall contribute equally through all the phases of the project namely Problem Definition, Requirements Gathering and Analysis, Design, Coding, Testing and Documentation.

The overall design issues will be primarily tackled by all team members. For every stage in design each member shall be responsible for bringing relevant knowledge to the discussion table following which there will be an identification and allotment of roles to each individual

towards development of a module, all in consultation with our internal project guide.

The division of work shall change for every module of the software to be written as per the requirements imposed by that module. However, keeping in mind that one of the key goals of this project is trying to gain certain amount of expertise in the various areas of the project, it shall be ensured that each individual is exposed to different tasks with a view to enhance his knowledge.

In order to ensure that the project work goes in parallel, the team would be split in pairs and work distributed amongst these pairs. The pairing of individuals would also be dynamic so that the aforementioned goal of knowledge enhancement is satisfied. In spite of this work division cross team knowledge sharing would be encouraged so that each pair gets to know what the other pair does.

5.4.2 Management reporting and communication

Mechanisms for progress reporting and inter/intra team communication are identified. The teams shall be created dynamically as per requirements at different stages of the project. The teams shall communicate every 2 days by actual meeting or through remote means of communication like telephone, instant messaging or mail.

The project group will communicate with the internal guide every week or sooner as per project requirements with a view to collectively plan out further development and communicate the current status of the project.

Chapter 6

Implementation and Testing

6.1 Implementation Details

The Implementation process begins after management has accepted the system. Implementations consist of the installation of the new system software(computer programs, procedures and forms), and personnel. The implementation phase is often the most difcult phase of the system development.

During implementation of the system, problems that had not been anticipated during the analysis and design phase often appear solutions to these problems usually required modication to the original design.

Following are the activities which take place during the system implementation:
a. Program the new system.
b. Writing programs.
c. Debugging and documenting program.
d. Converting data from old to new system.

6.1.1 GUI Design of True Distributed Virtual Compiler Editor

Central Server Login

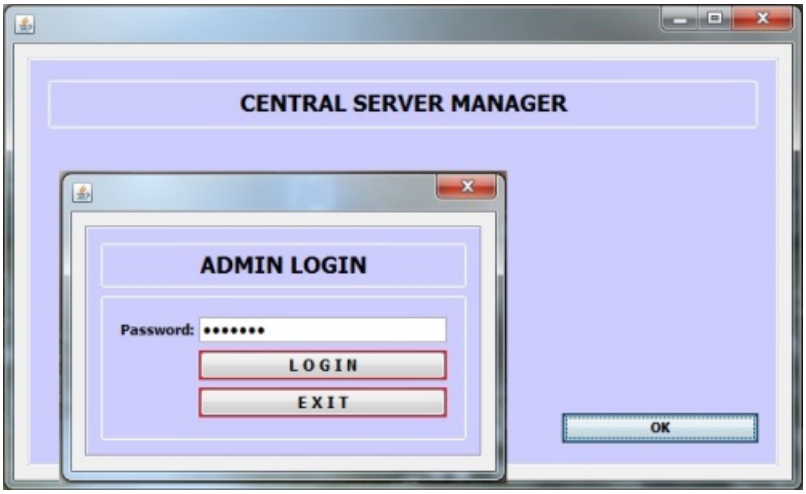

Figure 6.1: Central Server Login

Central Server Homepage

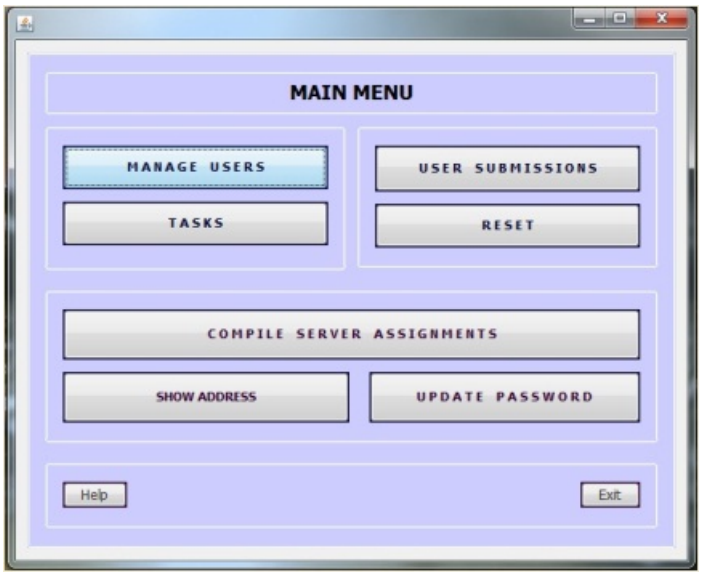

Figure 6.2: Central Server Homepage

Compile Server Login

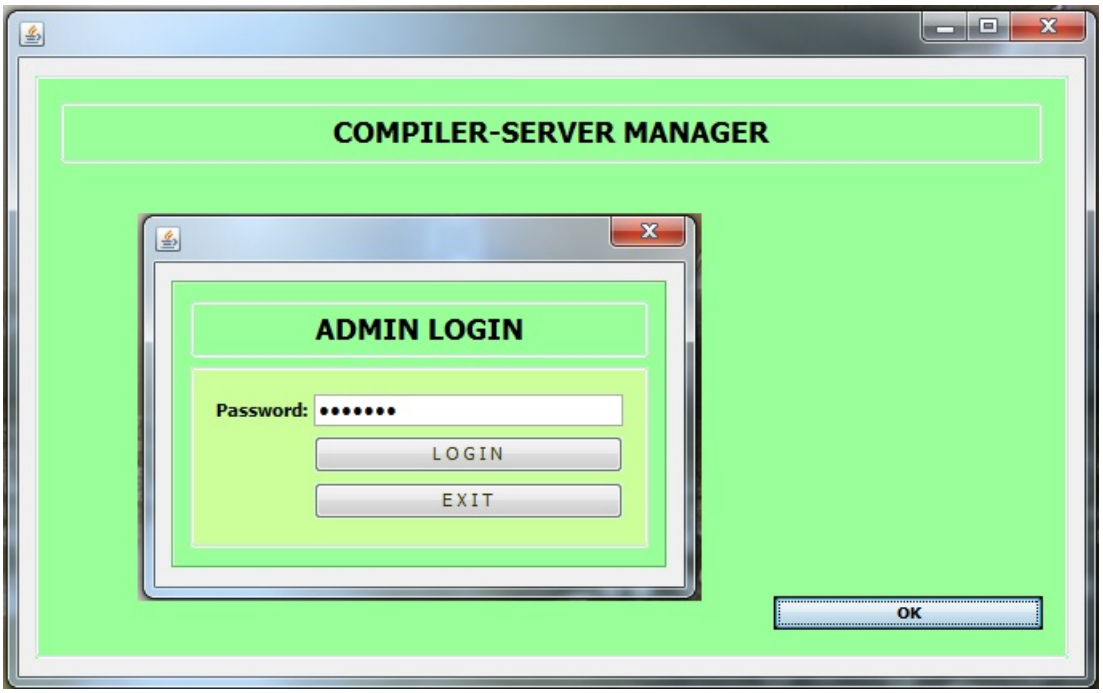

Figure 6.3: Compile Server Login

Compile Server Homepage

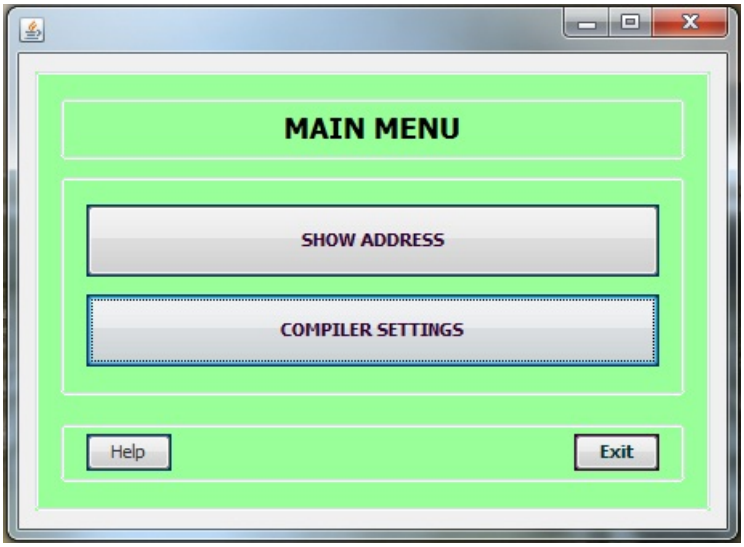

Figure 6.4: Compile Server Homepage

Distributed Client Login

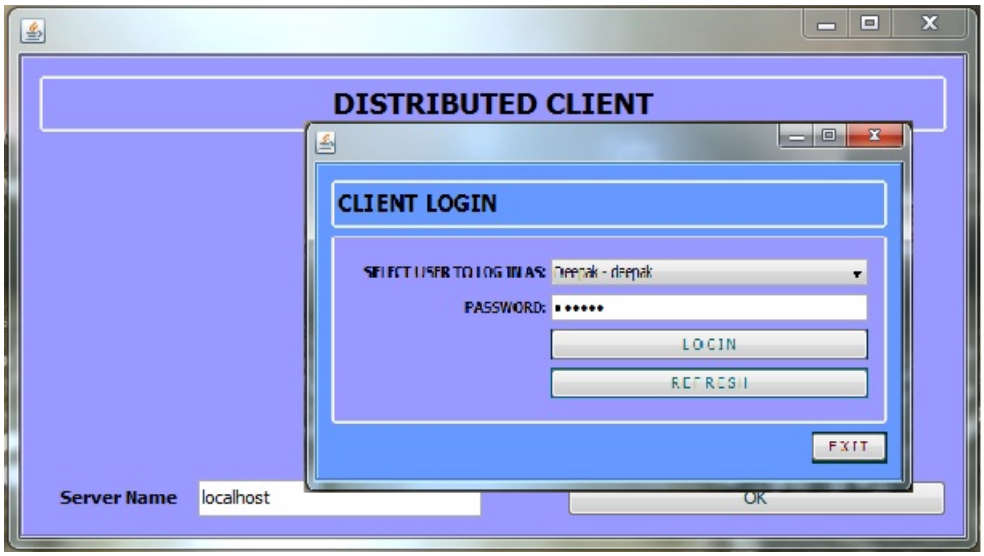

Figure 6.5: Distributed Client Login

Distributed Client Homepage

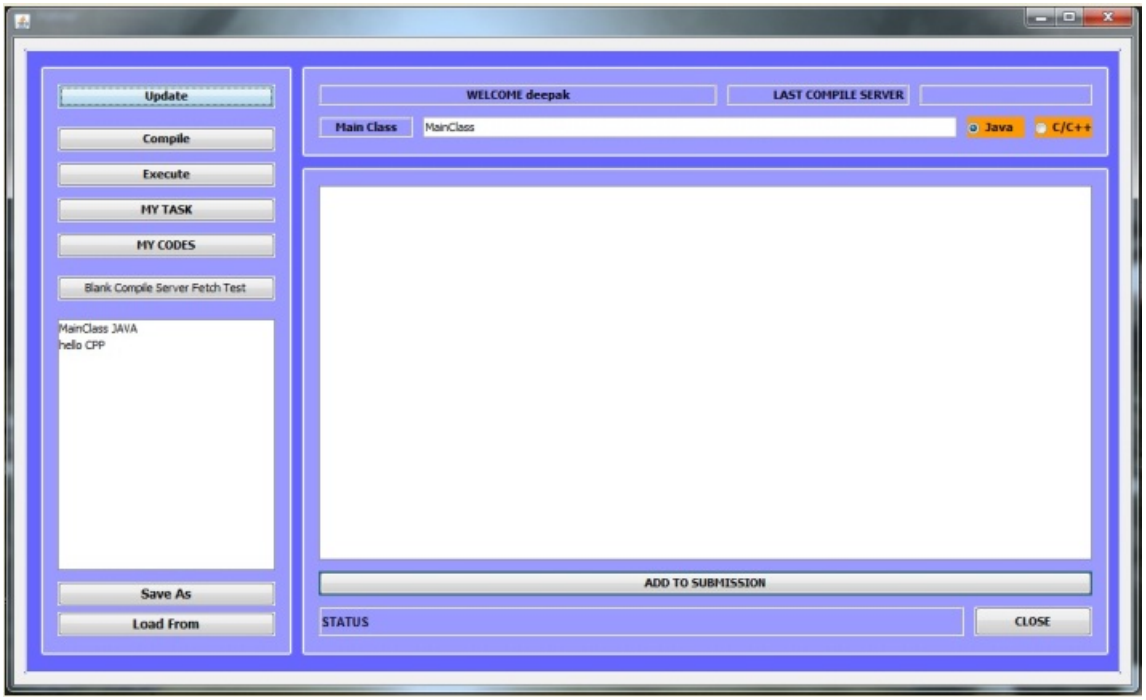

Figure 6.6: Distributed Client Login

6.1.2 Software Development Model

The software life cycle is a general model of the software development process,including all the activities and work products required to develop a software system. A software life cycle model is a particular abstraction representing a software life cycle. We have used Incremental Model as our software development approach.

Incremental Model

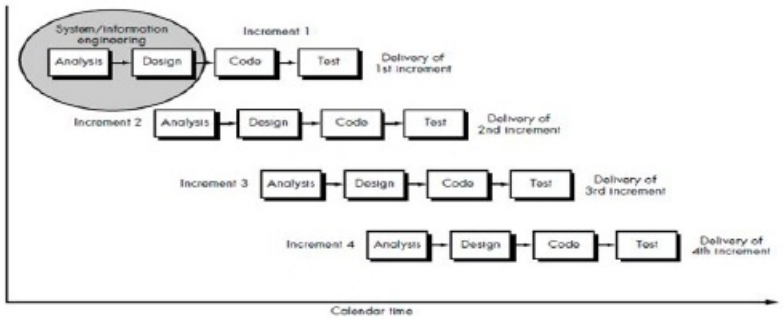

Figure 6.7: Incremental Model

The Incremental Model combines elements of the linear sequential model with the iterative philosophy . Each linear sequence produces deliverable ”increment of the software.”

Incremental Model is iterative in nature. In this model, the rst increment is often a core product. In core product, basic requirements are addressed. The core product is used for evaluation and as a result of evaluation plan is developed for next increment. This process is repeated following the delivery of each increament, until complete product is produced.

6.2 Testing

As the software is created and added to the developing system, testing is performed to ensure that it is working correctly and efciently. Testing

is generally focused on two areas:internal efciency and external effectiveness. The goal of external effectiveness testing is to verify that the software is working according to system design, and that is performing all necessary functions or sub-functions. The goal of internal testing is to make sure that the computer code is efcient, standardized, and well documented. Testing can be a laborintensive process, due to its iterative nature

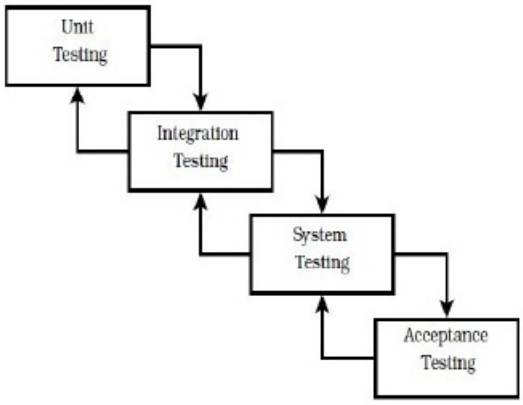

Figure 6.8: Tesing Process

6.2.1 Unit Testing

The primary goal of unit testing id to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you except. Each unit is tested separately before integrating them into modules to test the interfaces between modules. The most common approach to unit testing requires drivers and stubs to be written. The driver simulates a calling unit and the stub simulates a called unit.

6.2.2 Integrating Testing

Integration Testing starts at the module level when various modules are integrated with each other to form a sub-system or system. More stress is given on interfaces between the modules. Integration Testing focuses on the design and construction of the software architecture.

6.2.3 Regression Testing

Each time a new module is added as part of integration testing, the software changes. New data ow paths are established, new I/O may occur and new control logic is invoked. These changes may cause problems with functions that previously worked awlessly. In context of an integration test strategy, regression testing is the reexecution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.

6.2.4 Black Box Testing

It is a testing strategy which does not need any knowledge of internal design or code.This testing technique treats the system as black box or a closed box. The base of the Black Box Testing strategy lies in the selection of appropriate data as per functionality and testing it against the functional specications in order to check for normal and abnormal behavior of the system. In order to implement Black Box Testing strategy, the tester is needed to be thorough with the requirements specications of the system.

6.2.5 White Box Testing

It is the approach of testing that takes into account internal mechanism of a system or component, types include branch testing, path testing etc. Aspects missed out in the black box testing are tested in white box testing.

Test Case Id	Test Case Name	Description	Steps Carried out	Expected Results	Actual Result
TC1	Login form	Validation of username and password	1. Enter correct username and correct password & click submit 2. Enter correct username and wrong password & click submit. 3. Enter wrong username and correct password & click submit. 4. Enter wrong username and wrong password & click submit.	Well come page is displayed. Error Message = "Invalid username or password" is displayed. Error Message = "Invalid username or password" is displayed. Error Message = "Invalid username or password" is displayed.	Well come page is displayed. Error Message = "Invalid username or password" is displayed. Error Message = "Invalid username or password" is displayed. Error Message = "Invalid username or password" is displayed.

Figure 6.9: Test Case(1)

Test Case ID	Test Case Name	Description	Steps Carried out	Expected Results	Actual Result
TC3	Change setting form	Validation of Password	1. Enter Password And then again Re-Enter Password. 2. Enter corrected Password and Re-Enter Password is wrong	Password changed successfully Password not matched	Password changed successfully Password not matched

Figure 6.10: Test Case(2)

Chapter 7

Conclusion and Future Scope

In todays ever expanding and demanding world there is always a need as well as scope to enhance the capability of individual with the help of technology. In todays world where computers are speed enabled machine all the interactions are done using Internet technology.

In addition to expose to professional world this project has made us learn a lot of team spirit. It was not altogether a smooth going process indeed. There were many times, when we came across difficult situations. But we achieved experience in application of JAVA Methodology and tools. Understand the importance of Domain Knowledge and have achieved it after transfer from Classroom to an Organizational Environment.

We have provided a centralized compiling scheme, where centralized storage for all the compilers are stored. There is no need to maintain separate compilers or SDKs at client side. User authentication and personalized task distribution. i.e. the administrator will be able to assign user-id, password personalized tasks to all the clients. Both the error stream and output stream of the compiler will be captured and the output will be sent to client.

Future Enhancement:

As we know that todays world is completely dependent on the internet and online tools. We can enhance our idea and can make our tool as cloud compiling, so that anyone can use it at any time.

We can also complie our programs from remote locations with the help of concept cloud computing.

So like this possibilities are endless

References

[1] Java Server Programming 1.4 Edition-Black Book.

[2] The Complete Reference, Patrick Naughton,Herbert Schildt.

[3] Core Servlets and Java Server Pages , Marty Hall, Larry Brown

[4] weet, W. and Geppert, L., http:// It has changed everything, especially our engineering thinking, IEEE Spectrum, January 1997, pp. 23-37.

[5] amposano, R.; Deering, S.; DeMicheli, G.; Markov, L.; Mastellone, M.; Newton, A.R.; Rabaey, J.; Rowson, J.; Whats ahead for Design on the Web, IEEE Spectrum, September 1998, pp. 53-63.

[6] ank Shiffman, Making Sense of Java, www.disordered.org/Java-QA.html

[7] ank Shiffman, Boosting Java Performance: Native Code and JIT Compilers,www.disordered.org/Java-JIT.html

[8] undavaram, S.,. CGI Programming on the World Wide Web. OReilly Associates, Inc., 1996.

[9] all,L., Christiansen, T., Schwartz, R.L. Programming Perl, OReilly Associates, Inc., 1996