A TRUST-BASED AGENT LEARNING APPROACH FOR ENHANCING MOBILE CLOUD COMPUTING SERVICE

PROJECT REPORT
21AD1513- INNOVATION PRACTICES LAB

Submitted by

ENIYA S	**211422243070**
EVELIN BERCIYA C	**211422243071**
HARINI T	**211422243090**

in partial fulfillment of the requirements for the award of degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123

ANNA UNIVERSITY: CHENNAI-600 025

November, 2024

BONAFIDE CERTIFICATE

Certified that this mini project report **“A TRUST-BASED AGENT LEARNING APPROACH FOR ENHANCING MOBILE CLOUD COMPUTING SERVICE”** is the bonafide work of **“ENIYA S (211422243070) , EVELIN BERCIYA C (211422243071), HARINI T (211422243090)”** who carried out this project work under my supervision.

INTERNAL GUIDE	**HEAD OF THE DEPARTMENT**
Dr. .P.RAJESWARI,M.E., Ph.D.,	**Dr.S.MALATHI, M.E.,Ph.D.,**
Associate Professor	**Professor and Head,**
DEPARTMENT OF AI&DS,	**DEPARTMENT OF AI&DS,**
PANIMALAR ENGINEERING COLLEGE	**PANIMALAR ENGINEERING COLLEGE**
CHENNAI-600 123	**CHENNAI-600 123**

Certified that the above-mentioned students were examined in the End Semester mini project on Innovation Practices Laboratory (21AD1513) held on _____________

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ABSTRACT

Mobile cloud computing has the features of resource constraints, openness and uncertainty which leads to the high uncertainty on its Quality of Service (QOS) provision and serious security risks. Therefore, when faced with the complex service requirements, an efficient and reliable service composition approach is extremely important. In addition, preference learning is also a key factor to improve user experiences. In order to address them, this paper introduces a three-layered trust enabled service composition model for the mobile cloud computing systems. Based on fuzzy comprehensive evaluation method, we design a novel and integrated trust management model. Service brokers are equipped with a learning module enabling them to better analyze customers' service preferences especially in cases when the details of a service request are not totally disclosed. Because traditional methods cannot totally reflect the autonomous collaboration between the mobile cloud entities, a prototype system based on the multi-agent platform JADE is implemented to evaluate the efficiency of the proposed strategies. The experimental results show that our approach improves the transaction success rate and user satisfaction.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D.,** and **Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for providing us with necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr. K. MANI, M.E., Ph.D.,** who facilitated us in completing the project.

We thank the Head of the Artificial Intelligence and Data Science Department **Dr. S. MALATHI, M.E., Ph.D.,** for the support extended throughout the project. We would like to thank our supervisor **Dr. P. RAJESWARI, M.E., Ph.D.,** and all faculty members of the Department of AI&DS for their advice and encouragement for the successful completion of the project.

ENIYA S,

EVELIN BERCIYA C,

HARINI T

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
DFD	DATA FLOW DIAGRAM
AIC	AVAILABILITY INTERGRITY AND CONFIDENTIALITY
UML	UNIFIED MODELING LANGUAGE
GUI	GRAPHICAL INTERFACE
TCP	TRANSMISSION CONTROL PROTOCOL
UDP	USER DATAGRAM PROTOCOL
IP	INTERNET PROTOCOL
FCE	FUZZY COMPREHENSIVE EVALUATION

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPTION

Mobile cloud computing is the application of cloud computing in mobile Internet. It refers to the delivery and use mode of IT resources or information services to provide obtain infrastructure, platform, software (or applications) through mobile network in an on-demand and scalable manner. Mobile cloud computing builds up a hybrid application environment of cloud computing, Internet and mobile ends, improving the computational and storage capability of mobile terminals and providing users with a more rich and colorful functional experience. However, mobile cloud computing inherits both the advantages and disadvantages of cloud computing and mobile internet. The features of resource constraints, openness and uncertainty lead to the high uncertainty and unstable Quality of Service (QOS) provision and serious security risks. Especially in the face of complex service requirements, how to achieve efficient service composition, and how to ensure the credibility of combined services has become hot issue in the mobile cloud computing researches. Many valuable task scheduling and service composition strategies have been proposed for the traditional Internet environment. However, they cannot cope well with the active collaboration of participants in the mobile cloud computing markets. For this reason, agent-based cloud computing models are introduced. Mobile cloud systems based on multi-agent architecture are much easier to reflect the autonomy, intelligence and initiative of cloud entities, and to realize the independent evolution of the cloud service market, which is closer to the essence of a commercial market. To meet these requirements, we propose a Trust-based Agent Learning Model for Service Composition (TALMSC) in mobile cloud computing environments. We design a novel trust management model based on fuzzy comprehensive evaluation method and propose a trust-enabled service composition model. In order to obtain customers' service preferences and accelerate service classification, we equip service brokers with a learning module based on a two-stage improved Fuzzy C-Means (FCM) learning mechanism which can also improve the transaction success rate and user satisfaction. To make TALMSC more efficient, satisfactory and reliable, in the construction of our approach, the following key questions are addressed:

- What is the most suitable framework for the multi-agent based mobile cloud scheduling model? How can agents interact with each other?

● Since trust is fuzzy and context-aware, what is an efficient and integrated trust management model?

● What is the suitable learning algorithm to learn customers' service preferences? In contrast to the existing research works, this project mainly focuses on the impact of the external mechanisms on the service scheduling process.

1.2 OBJECTIVE

1) Data Secure Storage: In order to prevent data leakage and increase the difficulty of attack, this paper presents a method combining data distribution and data encryption to improve data storage security.

2) Hierarchical Key Management: To protect the key and prevent the attacker from using the key to recover the data, this paper introduces secret sharing and key hierarchy derivation algorithm in combination with user password to enhance key security.

3) Experimental Evaluation and Analysis: The security analysis and experimental results show that CSSM can effectively guarantee the security of data storage, and the increased performance cost is acceptable to users Remainder of the paper is organized as follows: A brief overview of CSSM mechanism is made in Section 2. Section 3 explains the proposed mechanism, and Section 4 introduces the implementation of CSSM. The experimental evaluations have been shown in Section 5. We discussed several variants and extensions of CSSM in Section 6. Finally, Section 7 concludes our work.

Cyber Security

Cyber Security is primarily about people, processes, and technologies working together to encompass the full range of threat reduction, vulnerability reduction, deterrence, international engagement, incident response, resiliency, and recovery policies and activities, including computer network operations, information assurance, law enforcement, etc.

Cyber Security is the protection of Internet-connected systems, including hardware, software, and data from Cyber Attacks. It is made up of two words one is cyber and other is security. Cyber is related to the technology which contains systems, network and programs or data. Whereas security related to the protection which includes systems security, network security and application and information security.

It is the body of technologies, processes, and practices designed to protect networks, devices, programs, and data from attack, theft, damage, modification or unauthorized access. It may also be referred to as information technology security**.**

We can also define cyber security as the set of principles and practices designed to protect our computing resources and online information against threats. Due to the heavy dependency on computers in a modern industry that store and transmit an abundance of confidential and essential information about the people, cyber security is a critical function and needed insurance of many businesses.

Cyber Security Goals

The objective of Cyber security is to protect information from being stolen, compromised or attacked. Cyber security can be measured by at least one of three goals

1. Protect the confidentiality of data.
2. Preserve the integrity of data.
3. Promote the availability of data for authorized users.

These goals form the confidentiality, integrity, availability (CIA) triad, the basis of all security programs. The CIA triad is a security model that is designed to guide policies for information security within the premises of an organization or company. This model is also referred to as the **AIC (Availability, Integrity, and Confidentiality)** triad to avoid the confusion with the Central Intelligence Agency. The elements of the triad are considered the three most crucial components of security.

The CIA criteria are one that most of the organizations and companies use when they have installed a new application, creates a database or when guaranteeing access to some data. For data to be completely secure, all of these security goals must come into effect. These are security policies that all work together, and therefore it can be wrong to overlook one policy.

CHAPTER 2

LITERATURE SURVEY

1. Mobile cloud computing: Challenges and future research directions Author: Talal H. Noor

Mobile cloud computing promises several benefits such as extra battery life and storage, scalability, and reliability. However, there are still challenges that must be addressed in order to enable the ubiquitous deployment and adoption of mobile cloud computing. Some of these challenges include security, privacy and trust, bandwidth and data transfer, data management and synchronization, energy efficiency, and heterogeneity. We present a thorough overview of mobile cloud computing and differentiate it from traditional cloud computing. Also presented here is a generic architecture that evaluates 30 recently proposed mobile cloud computing research architectures (i.e., published since 2010). This is achieved by utilizing a set of assessment criteria. Finally, we discuss future research challenges that require further attention.

2. TSLAM: A Trust-enabled Self-Learning Agent Model for Service Matching in the Cloud Market

Author: WENJUAN LI

With the rapid development of cloud computing, various types of cloud services are available in the marketplace. However, it remains a significant challenge for cloud users to find suitable services for two major reasons: (1) Providers are unable to offer services in complete accordance with their declared Service Level Agreements, and (2) it is difficult for customers to describe their requirements accurately. To help users select cloud services efficiently, this article presents a Trust enabled Self-Learning Agent Model for service Matching (TSLAM). TSLAM is a multi-agent-based three-layered cloud service market model, in which different categories of agents represent the corresponding cloud entities to perform market behaviors. The unique feature of brokers is that they are not only the service recommenders but also the participants of market competition. We equip brokers with a learning module enabling them to capture implicit service demands and find user preferences. Moreover, a distributed and lightweight trust model is designed to help cloud entities make service decisions. Extensive experiments prove that TSLAM is able to optimize the cloud service matching process and compared to the state-of-the-art studies, TSLAM improves user satisfaction and the transaction success rate by at least 10%.

3. Research on Trust Management Strategies in Cloud Computing Environment

Author: Wenjuan LI

In order to protect the security of cloud entities and better practice cloud's objectives of providing low cost and on-demand services, this paper proposes a novel cloud trust transaction framework and also a new trust fuzzy comprehensive evaluation based cloud service discovery algorithm. The new algorithm uses trust multi-dimensional vector to represent the credibility of providers when offer different types of services and applies fuzzy comprehensive evaluation method to classify services. The simulation experiments show that the proposed algorithm ensures relatively high trust accuracy. Besides it has certain advantages in rejecting malicious nodes and improving the transactions' success rate compared to other kindred solution.

4. Gen Trust: A genetic trust management model for peer-to-peer systems Author: Ugur Eray Tahta

In recent years, peer-to-peer systems have attracted significant interest by offering diverse and easily accessible sharing environments to users. However this flexibility of P2P systems introduces security vulnerabilities. Peers often interact with unknown or unfamiliar peers and become vulnerable to a wide variety of attacks. Therefore, having a robust trust management model is critical for such open environments in order to exclude unreliable peers from the system. In this study, a new trust model for peer-to-peer networks called Gen Trust is proposed. Gen Trust has evolved by using genetic programming. In this model, a peer calculates the trustworthiness of another peer based on the features extracted from past interactions and the recommendations. Since the proposed model does not rely on any central authority or global trust values, it suits the decentralized nature of P2P networks. Moreover, the experimental results show that the model is very effective against various attackers, namely individual, collaborative, and pseudo spoofing attackers. An analysis on features is also carried out in order to explore their effects on the results. This is the first study which investigates the use of genetic programming on trust management.

5. Location-Aware Service Recommendation With Enhanced Probabilistic Matrix Factorization

Author: YUYU YIN

Owing to the ever-growing popularity of mobile computing, a large number of services have been developed for a variety of users. Considering this, recommending useful services to users is an urgent problem that needs to be addressed. Collaborative filtering (CF) approaches have been successfully adopted for services recommendation. Nevertheless, the prediction accuracy of the existing CF approaches is likely to reduce due to many reasons, such as inability to use side information and high data spar city, which further lead to low quality of services recommendation. In order to solve these problems, some model based CF approaches have been proposed. In this paper, we propose a novel quality of service prediction approach based on probabilistic matrix factorization (PMF), which has the capability of incorporating network location (an important factor in mobile computing) and implicit associations among users and services. First, we propose a novel clustering method that is capable of utilizing network location to cluster users. Based on the clustering results, we further propose an enhanced PMF model. The proposed model also incorporates the implicit associations among users and services. In addition, our model incorporates the implicit relationships between the users and the services. We conducted experiments on one real-world data set, and the experimental results show that our model outperforms the compared methods.

CHAPTER 3

SYSTEM ANALYSIS

3.1 INTRODUCTION

System analysis refers to the study of existing system in terms of system goals. The system analysis of a project includes the basic analysis for the project development, the required data to develop the project, the cost factor considered for the project development and other related factors.

3.2 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

- To analyze whether the software will meet organizational requirements
- To determine whether the software can be implemented using the current technology and within the specified budget and schedule
- To determine whether the software can be integrated with other existing software

Types of Feasibility

Commonly considered feasibility includes:

3.2.1 Technical Feasibility
3.2.2 Social Feasibility
3.2.3 Economic Feasibility

3.2.1 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system. Technical feasibility also performs the following tasks:

Analyzes the technical skills and capabilities of the software development team members Determines whether the relevant technology is stable and established ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

3.2.2 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2.3 Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased. Software is said to be economically feasible if it focuses on the below listed issues:

- Cost incurred on software development to produce long-term gains for an organization.
- Cost required to conducts full software investigation such as, requirements elicitation and requirements analysis.
- Cost of hardware, software, development team, and training.

3.3 EXISTING SYSTEM

- In our existing, data owner uploads the files in cloud in encrypted format only, then data user and cloud just view the files added by data owner
- The data use, cannot download the data owner files without secret key.
- Data user send request to data owner, data owner send to authority, and authority will send secret key to user.
- Then data user, decrypt the files. The major drawback in cloud is whether the cloud is trusted cloud (or) not because user Upload the files contain sensitive Information's.

3.3.1 DISADVANTAGES OF EXISTING SYSTEM:

- To maximize the confidentiality of cloud data storage, the proposed mechanism should make full use of the advantages of the method and effectively control its disadvantages.
- Data owner don't know about that cloud is trusted cloud (or) not.
- May leak some sensitive information in untrusted cloud.

3.4 PROPOSED SYSTEM

In our proposed introduces a three-layered trust enabled service composition model for the mobile cloud computing systems.

Based on fuzzy comprehensive evaluation method, we design a novel and integrated trust management model. Service brokers are equipped with a learning module enabling them to better analyze customers' service preferences especially in cases when the details of a service request are not totally disclosed.

3.4.1 ADVANTAGES OF PROPOSED SYSTEM:

- High security and more effective.
- Data owner know about that cloud is trusted cloud (or) not through broker agent.
- Provider agent checks user agent file is safe (or) not.
- This system gives more accuracy compared to another system.

3.5 SYSTEM REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS

•	Processor	: I5
•	Speed	: 2.2 GHz
•	RAM	: 2 GB
•	Hard Disk	: 160 GB

3.5.2 SOFTWARE REQUIREMENTS

•	Platform	: Windows7
•	Front-end	: HTML,CSS,JS
•	Back-end	: SQL Server 2008 R2

3.6 LANGUAGE SPECIFICATION

3.6.1 FEATURES OF JAVA

Java Technology

- ➢ Java technology is both a programming language and a platform.

The Java Programming Language

- ➢ The Java programming language is a high-level language that can be characterized by all of the following buzzwords:
  - ▪ Simple
  - ▪ Architecture neutral
  - ▪ Object oriented
  - ▪ Portable
  - ▪ Distributed
  - ▪ High performance
  - ▪ Interpreted

- Multithreaded
- Robust
- Dynamic
- Secure

➢ With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

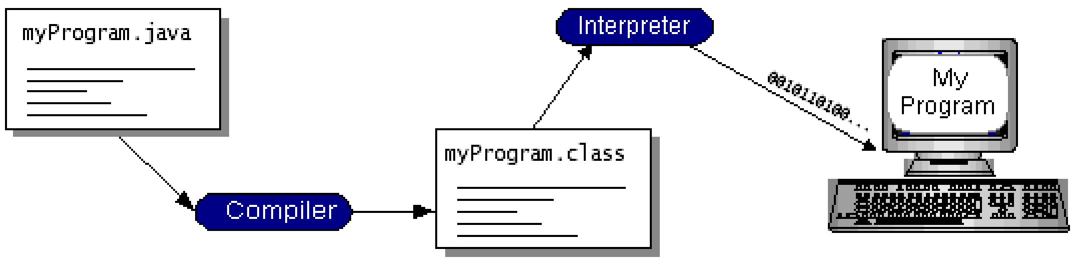
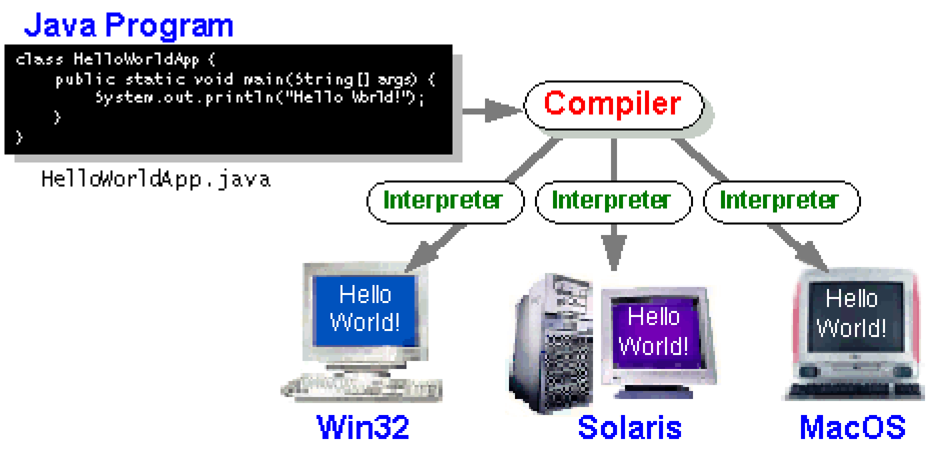


Figure No: 1 Java Virtual Machine Diagram

It can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Figure No: 2 Java Program Diagram

The Java Platform

A platform is the hardware or software environment in which program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MACOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The Java Virtual Machine (Java VM)
- The Java Application Programming Interface (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported on to various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, what Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

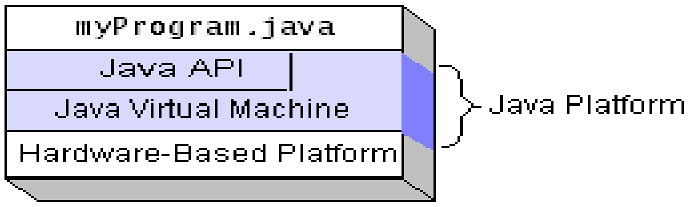


Figure No: 3 Java Platform Diagram

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you have surface the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server. How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

The essentials: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

Applets: The set of conventions used by applets.

Networking: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.

Internationalization: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.

Security: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.

Software components: Known as JavaBeans TM, can plug into existing component architectures.

Object serialization: Allows lightweight persistence and communication via Remote Method Invocation (RMI).

Java Database Connectivity (JDBCTM): Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

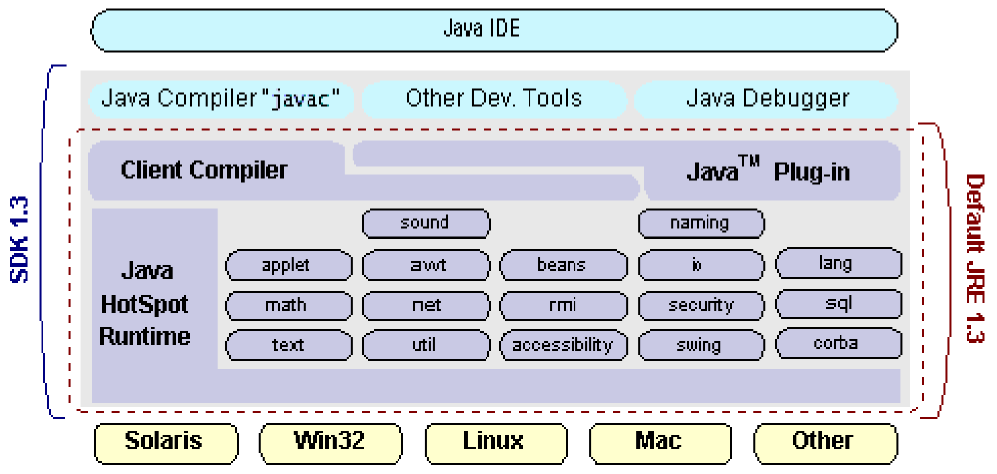


Figure No: 4 Java Database Connectivity (JDBC) Diagram

How Will Java Technology Change My Life?

It can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

Get started quickly: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.

Write less code: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.

Write better code: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.

Develop programs more quickly: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.

Avoid platform dependencies with 100% Pure Java: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java ^{TM} Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.

Write once, run anywhere: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.

Distribute software more easily: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to.

Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even

Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true.

The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. **SQL Level API**

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to "generate" JDBC code and to hide many of JDBC's complexities from the end user

2. **SQL Conformance**

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. **JDBC must be implemental on top of common database interface**

The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. **Provide a Java interface that is consistent with the rest of the Java system**

Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. **Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. **Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

7. **Keep the common cases simple**

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally, we decided to product the implementation using Java Networking.

And for dynamically updating the cache table we go for MS Access database.

Java ha two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

Simple	Architecture-neutral
Object-oriented	Portable
Distributed	High-performance
Interpreted	multithreaded
Robust	Dynamic
Secure	

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

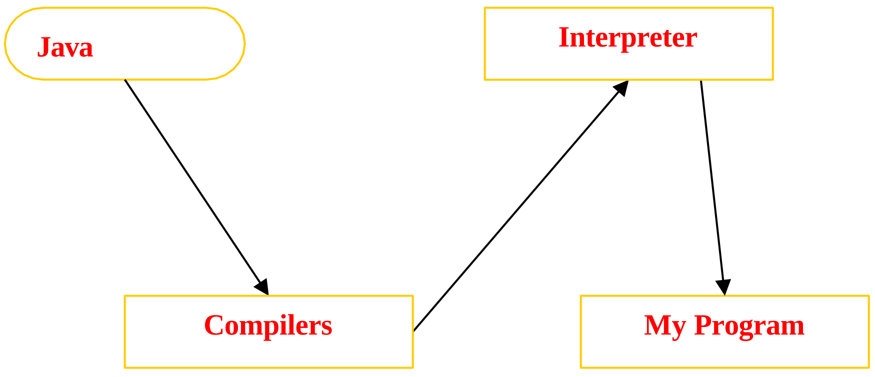


Figure No: 5 Java byte codes

It can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

Networking

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram’s

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address

The 32 bit address is usually written as 4 integers separated by dots.

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

```
#include   <sys/types.h>

#include <sys/socket.h>

int socket(int family, int type, int protocol);
```

Here "family" will be AF_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

J Free Chart

J Free Chart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. J Free Chart's extensive feature set includes:

- A consistent and well-documented API, supporting a wide range of chart types;
- A flexible design that is easy to extend, and targets both server-side and client-side applications;
- Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

- J Free Chart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licences (LGPL), which permits use in proprietary applications.

1. Map Visualizations

- Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

- Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

- Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XY Plot class in J Free Chart;

- Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to J Free Chart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of J Free Chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

The property editor mechanism in J Free Chart only handles a small subset of the properties that can be set for charts. Extends (or implements)this mechanism to provide greater end-user control over the appearance of the charts.

J2ME (Java 2 Micro edition)

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems."

Announced in June 1999 at the Java One Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

1. General J2ME Architecture

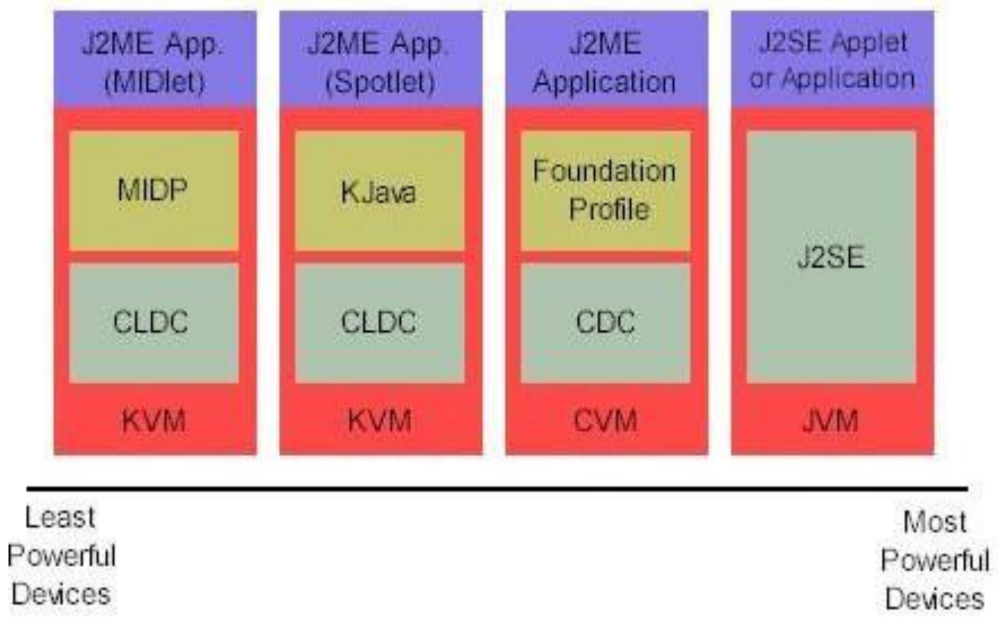


Figure No: 6 Java runtime Environment

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes.

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

2. Developing J2ME applications

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role pre verification plays in this process.

3. Design considerations for small devices

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

Keep it simple: Remove unnecessary features, possibly making those features a separate, secondary application.

Smaller is better: This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

Minimize run-time memory use: To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

4. Configurations overview

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

Connected Limited Device Configuration (CLDC) is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

Connected Device Configuration (CDC) is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

5. J2ME profiles

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: K Java and MIDP. Both K Java and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

Profile 1: K Java

K Java is Sun's proprietary profile and contains the K Java API. The K Java profile is built on top of the CLDC configuration. The K Java virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. K Java contains a Sun-specific API that runs on the Palm OS. The K Java API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. Learn more about the K Java API later in this tutorial when we develop some sample applications.

Profile 2: MIDP

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like K Java, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

* java.lang

* java.io

* java.util

* javax.microedition.io

* javax.microedition.lcdui

* javax.microedition.midlet

* javax.microedition.rms

CHAPTER 4

SYSTEM DESIGN

SYSTEM ARCHITECTURE

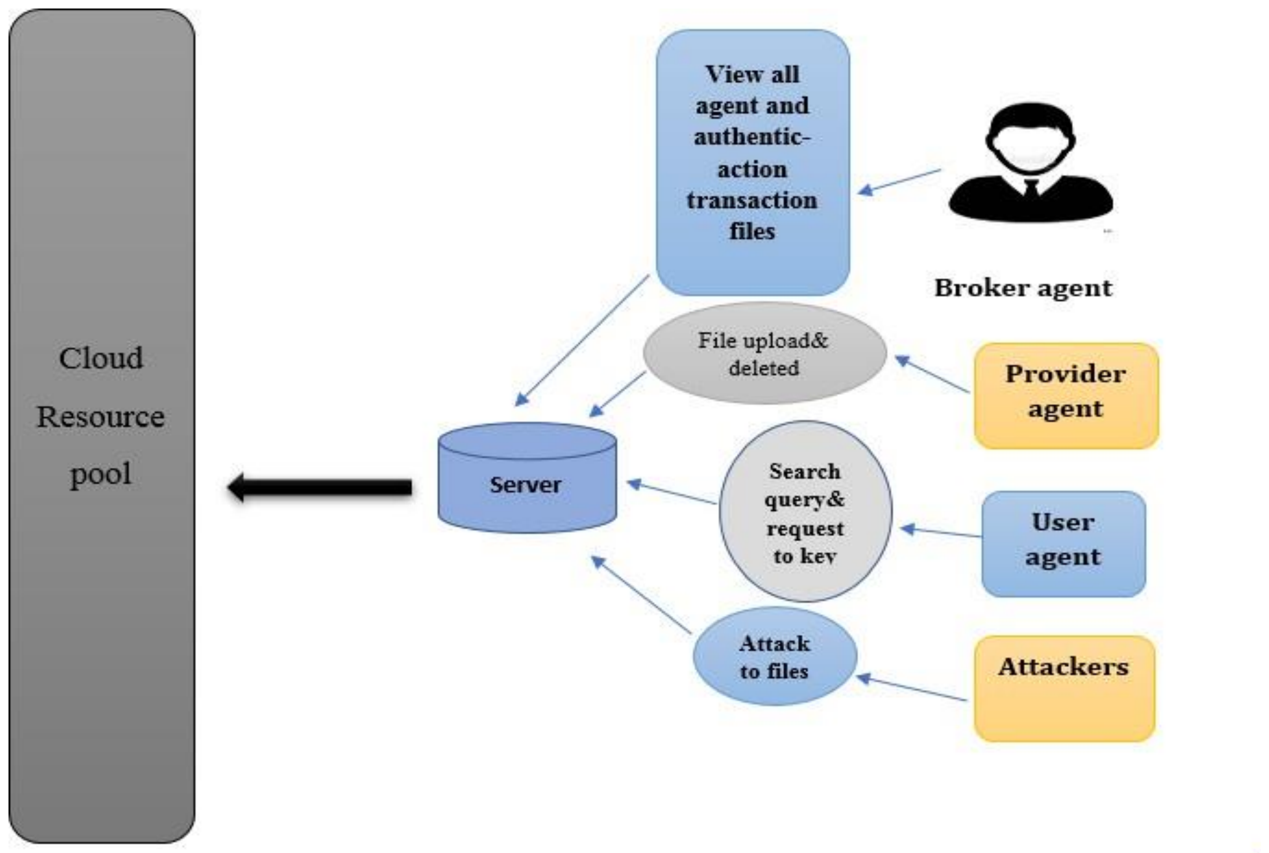


Figure no: 7 System Architecture for Mobile Cloud Computing

4.1 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of the processing.

Symbols used in a Data Flow Diagram

Symbol	Meaning
▭	The boxes represent external Entities.
⬭	A process or task that is performed by the system.
⊏	A data store reveals the storage of data.
↘	The data flow lines include arrows to show the direction of data element movement.

DATA FLOW DIAGRAM:

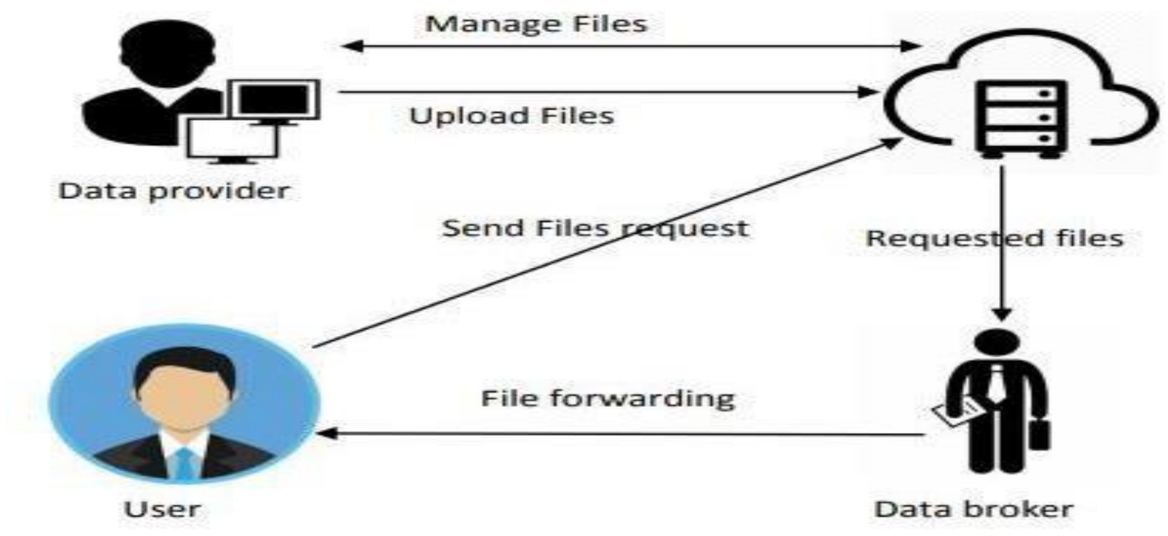


Figure No: 8 Data Flow Diagram for Mobile Cloud Computing

4.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general purpose modeling language in the field of object-oriented software engineering. The standard is managed and was created by, the Objected Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to or associated with UML.

The UML is a standard for specifying, Visualizing, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

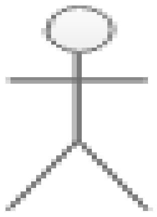
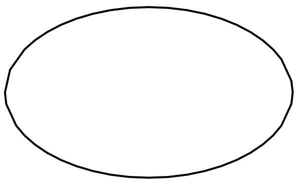
The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS OF UML

- Provides users ready-to-use, expressive visual modeling languages so that they can develop and exchange meaningful models.
- Provides extendibility and specification mechanisms to extend the core concepts.
- Be independent of particular programming language and development process

4.2.1 USECASE DIAGRAM

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components a user or another system that will interact with the system modeled. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

SYMBOL	Meaning
	An **actor**. It specifies a role played by a user or any other system that interacts with the subject.
	A **use case**. It is a list of steps, typically defining interactions between an actor and a system, to achieve a goal.

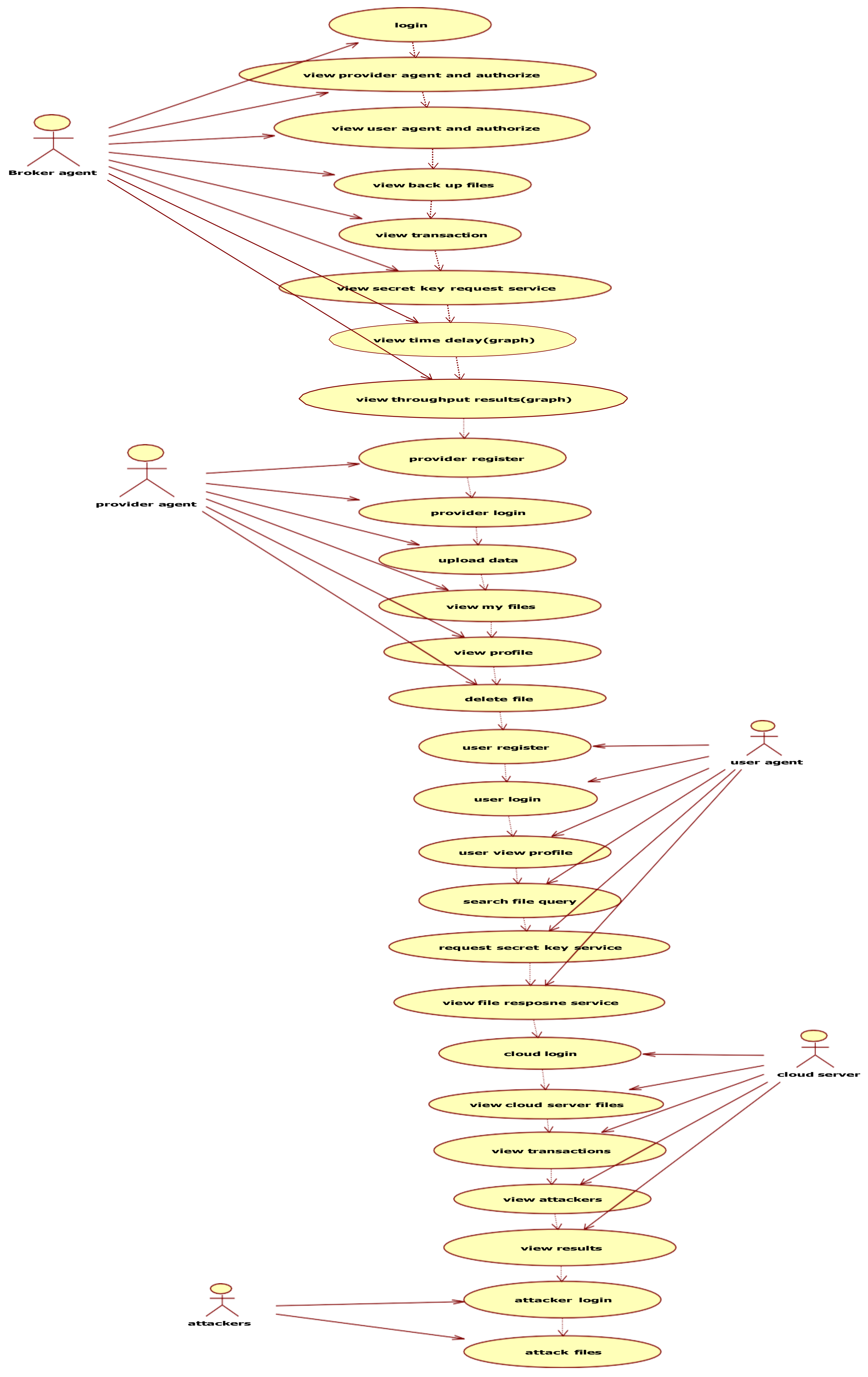
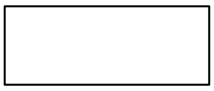
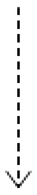
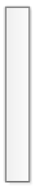
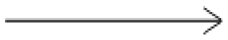


Figure No: 9 UML Diagram for Mobile Cloud Computing

4.2.2 SEQUENCE DIAGRAM

The sequence diagram shows how different objects interact with each and the order of those interactions occurs. They are ordered by time. They are useful if someone wants to review the flow of logic through a scenario.

Symbol	Meaning
	An **object** represents a class or object, in UML. They demonstrate how an object will behave in the context of the system.
	**Lifeline** represents the passage of time as it extends downward.
	**Activate** is used to denote participant activation. Once a participant is activated, its lifeline appears.
	A **Message** is an element that defines a specific kind of communication between instances in an interaction

The following symbols are used in a sequence diagram

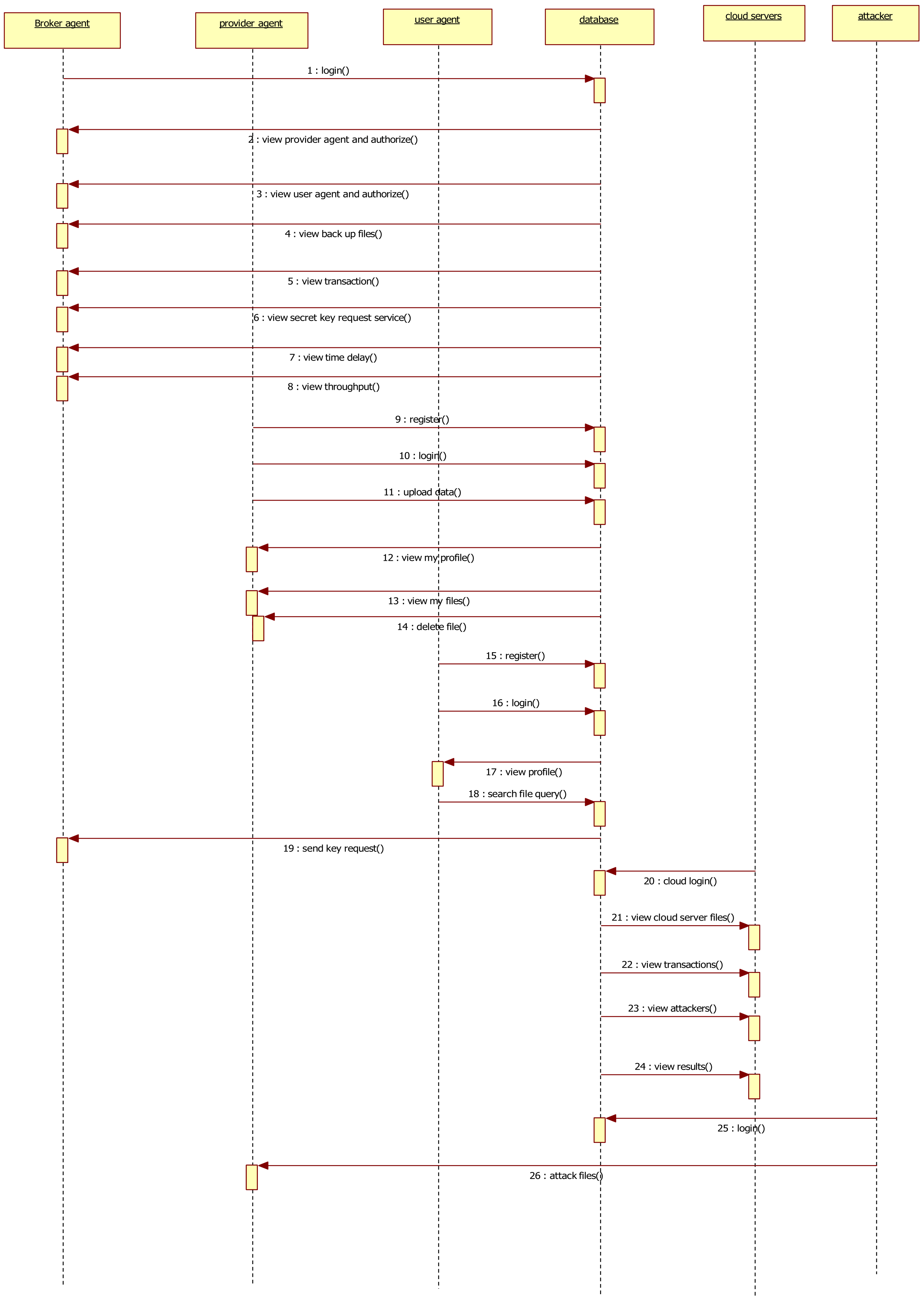


Figure No: 10 Sequence Diagram of Mobile Cloud Computing works

CHAPTER 5

SYSTEM DEVELOPMENT

5.1 MODULE DESCRIPTION

In this project, we have five modules

1. Broker agent

2. Provider agent

3. User agent

4. Cloud servers

5. Attackers

1. Broker agent

- Login
- View provider agent and authorize
- View user agent and authorize
- View back up files (added by provider)
- View transactions
- View secret key request service
- View time delay results(graph)
- View throughput results(graph)
- Logout

2. Provider agent

- Register
- Login
- Upload data
- View my files
- View my profile
- Verify
- Delete file
- Logout

3. User agent

- Register
- Login
- View my profile

- Search file query
- Request secret key service
- View file response service
- Logout

4. **Cloud servers**
    - Login
    - View cloud server files
    - View transactions
    - View attackers
    - View results(graph)
    - Logout
5. **Attackers**
    - Login
    - Attack files
    - Logout

MODULES DESCRIPTION:

1. Broker agent

The broker agent logs into their account and has the ability to view and authorize provider agents and user agents for their account access. They can also view backup files uploaded by providers, examine transaction details, and review secret key request service information. Additionally, the broker agent can view graphical representations of time delay results and throughput results. The broker agent can log out of their account when finished.

2. Provider agent

The provider agent registers their details and, once authorized by the broker agent, can log into their account. They can upload data, view and manage their files, check their profile, verify data accuracy, and delete files as needed. The provider agent can log out of their account when done.

3. User agent

The user agent registers their details and, upon authorization by the broker agent, can log into their account. They can view their profile, perform file searches, request secret key services, and review file responses. The user agent can log out of their account when finished.

4. Cloud servers

The cloud server logs in to access and view server files, examine transaction details, monitor potential attackers, and review results through graphical representations. The cloud server can log out of their account when done.

5. Attackers

The attackers log into their account to perform file attacks and then log out when finished.

CHAPTER 6

SYSTEM TESTING

6.1 INTRODUCTION

The purpose of testing is to discover errors. Testing is the process to discover every conceivable fault or error or the weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 OBECTIVES OF TESTING

Main objective of testing is to find errors. The success of testing in revealing errors in programs depends on the critical test cases. A good test case is one that has a high probability of finding an as-yet-undiscovered error or fault. A successful testing is one that reveals undiscovered errors to make the software more rugged and reliable.

Errors can be injected at any stage during development. In each phase different techniques for detecting and eliminating errors are incorporated. Testing should be planned based on the complexity of the activity or work. The testing for this project has been done on an incremental basis, in which components and subsystems of the system are tested separately before integrating them to form the system for system testing.

6.3 TYPES OF TESTING

- White Box Testing
- Black Box Testing

6.3.1 WHITE BOX TESTING

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

6.3.2 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box test, as most other kinds of test, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box and cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

The steps involved in black box test case design are

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison Testing

6.4 TESTING STRATEGIES

6.4.1 UNIT TESTING

It deals with the testing of the smallest units of the system design namely the module. This helps in identifying the syntax error/low-level logical errors at the lowest level possible. Unit testing positive result indicates that the module as such is error free and is limited to the boundary of the module itself. This means that the modules by it is error-free and may have errors when connected to other modules. This means that there may be massive logical errors, which can't be identified at the level of unit testing. The procedural design descriptions are helpful in testing the control paths within the boundary of the module. The unit testing methodology can be done to many modules of the same system simultaneously.

6.4.2 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check the components or software applications, e.g. components in a software system or –one step up- software applications at the company level-interact without error.

It involves testing of modules in series with regard to control flow system. This helps identifying the logical error that is the possible due to invalid number of arguments, invalid data type sent across the modules during the control flow. In other words, integration acts as a fine tune in finding out flaws in the system with more than one module in series. The approaches in integration testing namely, top-down approach give a better way to test the sub-system connected in series as in the control flow.

6.4.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
System/Procedures	: interfacing system or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current test is determined.

6.4.4 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

CHAPTER 7

SYSTEM IMPLEMENTATION

Implementation is the most crucial stage in achieving a successful system and giving the user's confidence that the new system is workable and effective, implementation of modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the program specification, the computer system and its environment is tested to the satisfaction of the user.

The system that has been developed is accepted, proved to be satisfactory for the user, and so the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions quickly.

Initially, as a first step the executable form of the application is to be created and loaded in the common server machine which is accessible to all users and the server is to be connected to a network. The final stage is to document the entire system which provides component and the operating procedure of the system. The implementation involves the following things:

- Proper planning
- Investigation of the system and constraints
- Design the method to achieve the changeover
- Training the methods to achieve the changeover
- Training of the staff in the change phase.

CHAPTER 8

SYSTEM MAINTENANCE

8.1 INTRODUCTION

In software maintenance, an enormous mass of potential problems and cost lies under the surface. Software maintenance is far more than fixing mistakes. Analyst and programmers append for more time in maintaining the program than they do writing them. Few tools and techniques are available for maintenance. The literature on maintenance contains very few entries when compared to development activities.

The software maintenance is classified into four tasks:

8.1.1 Corrective maintenance

8.1.2 Adaptive maintenance

8.1.3 Perfective maintenance

8.1.4 Preventive maintenance

8.1.1 CORRECTIVE MAINTENANCE

This type of maintenance implies removing errors in a program, which might have crept in the system due to faulty design or wrong assumptions. Thus, in corrective maintenance, processing or performance failures are required.

8.1.2 ADAPTIVE MAINTENANCE

In adaptive maintenance, program functions are changed to enable the information needs of the user. This type of maintenance may become necessary because of organizational changes which may include:

- Change in the organizational procedures,
- Change in organizational objectives, goals, policies, etc.
- Change in forms,
- Change in information needs of managers.
- Change in system controls and security needs, etc.

8.1.3 PERFECTIVE MAINTENANCE

Perfective maintenance means adding new programs or modifying the existing programs to enhance the performance of the information system. These types of maintenance are undertaken to respond to user’s additional needs which may be due to the changes within or outside of the organization. Outside changes, primarily environmental changes, which may be in the absence of system maintenance, render the information system in effective and inefficient. This environmental change includes:

a) Changes in governmental policies, laws, etc.
b) Economic and competitive conditions
c) New technology

The results obtained from the evaluation process help the organization to determine whether its information systems are effective and efficient or otherwise.

8.1.4 PREVENTIVE MAINTENANCE

Preventive maintenance is a modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults.

CHAPTER 9

CONCLUSION

This project proposed a novel trust-enabled service composition model for mobile cloud environments. TALMSC is a three-tier mobile cloud market model which includes the mobile cloud users (customers), the service providers, and the service inter mediator (broker). Brokers are the key entities who manage the providers and help the users to find the most suitable providers/resources. In order to improve the efficiency, reliability and satisfaction of service scheduling, trust and learning mechanisms are introduced in the service matching process. A novel integrated trust model based on FCE method is proposed.

CHAPTER 10

FUTURE ENHANCEMENT

The future scope is more the new trust mechanism is comprehensive, context-aware, and able to combine the direct trust with the recommendation trust. In addition, a two-stage improved FCM algorithm is designed to improve the learning ability of brokers.

APPENDIX 1

SAMPLE CODING

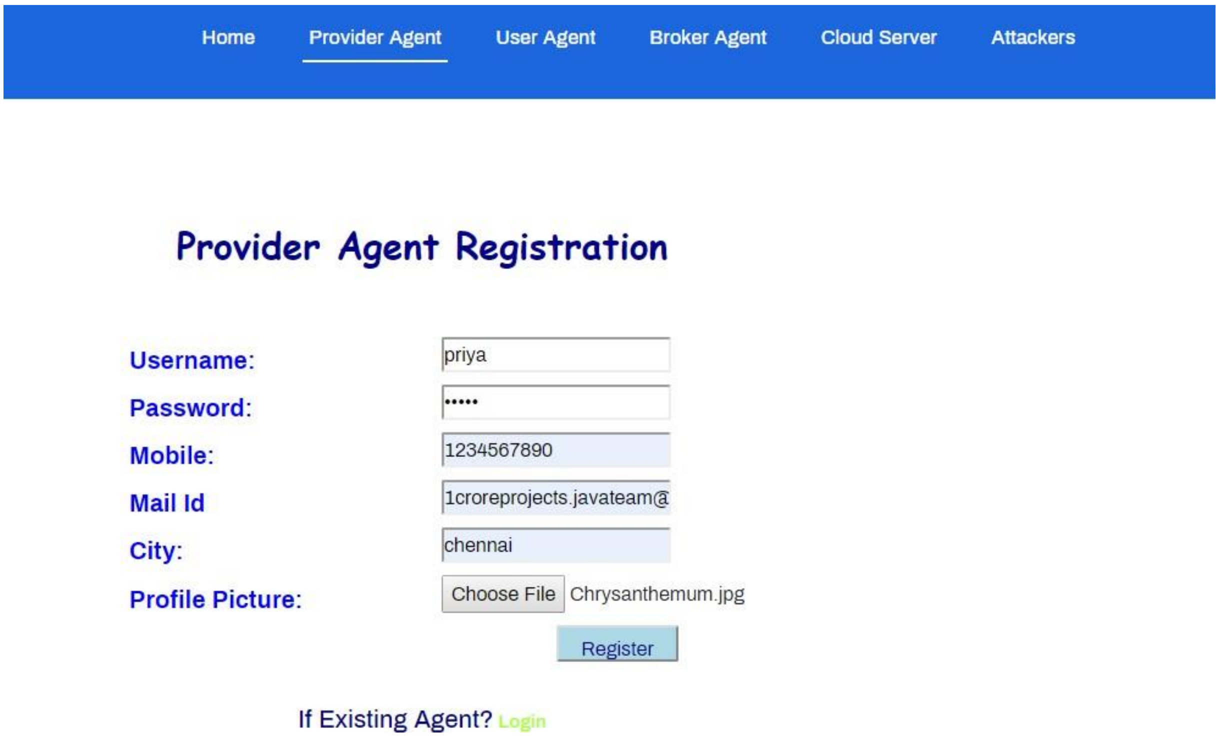
```
package dbServices;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DB {
private static final String URL "jdbc:mysql://localhost:3306/spam_filtering";
private static final String USER = "root";
private static final String PASSWORD = "root";
public Connection get Connection() throws SQL Exception {
return Driver Manager. get Connection(URL, USER, PASSWORD);
}
}
import java.io.IO Exception;
import java.io.Print Writer;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(urlPatterns = {"/Login_Action"})
public class Login_Action extends Http Servlet {
protected void process Request(Http Servlet Request request,
Http Servlet Response response)
   throws ServletException, IOException {
 response.setContentType("text/html;charset=UTF-8");
 try(Print Writer out = response.getWriter()) {
            String username = request.getParameter("username");
            String password = request.getParameter("password");
  if ("admin@gmail.com".equalsIgnoreCase(username) &&
"admin".equals(password)) {


                out.println("<script>alert('Welcome  Admin');</script>");
```

```
              Request Dispatcher rd = request.get Request Dispatcher("User_Home.jsp");
              rd.include(request, response);
  } else {
              out.println("<script>alert('Invalid  Login');</script>");
 RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
 rd.include(request, response);
            }
        } catch (Exception e) {
            e.printStackTrace();
             }
    }
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    public String getServletInfo() {
        return "Handles user login";
    }
}
import dbServices.DB;
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
public class movie extends HttpServlet {
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            ServletContext sc = request.getSession().getServletContext();
            List<String> m = new ArrayList<>();
            String finalImage = "";
            boolean isMultipart =
ServletFileUpload.isMultipartContent(request);
            if (!isMultipart) {
                out.println("<script>alert('File Not Uploaded');</script>");
                return;
            }
            FileItemFactory factory = new DiskFileItemFactory();
            ServletFileUpload upload = new ServletFileUpload(factory);
            List<FileItem> items;
            try {
                items = upload.parseRequest(request);
            } catch (FileUploadException e) {
                e.printStackTrace();
                out.println("<script>alert('File upload failed.');</script>");
                return;
            }
for (FileItem item : items) {
                if (item.isFormField()) {
                    String value = item.getString();
                    m.add(value);
                } else {
                    String itemName = item.getName();
                    finalImage = new File(itemName).getName();
                    File savedFile = new File(sc.getRealPath("video") +
File.separator + finalImage);
                    try {
                        item.write(savedFile);
                    } catch (Exception e) {
                        e.printStackTrace();
                        out.println("<script>alert('File save failed.');</script>");
                        return;
                    }
                }
            }
try (Connection con = new DB().getConnection();
                PreparedStatement ps = con.prepareStatement("INSERT
INTO movie(id, oname, file, skey, video, des) VALUES (?, ?, ?, ?, ?,
?)")) {
                ps.setString(1, m.get(0));
                ps.setString(2, m.get(1));
                ps.setString(3, m.get(2));
                ps.setString(4, m.get(3));
                ps.setString(5, finalImage);
                ps.setString(6, m.get(4));

                ps.executeUpdate();

                out.println("<script>alert('File uploaded
successfully');</script>");

            } catch (SQLException ex) {

                ex.printStackTrace();

                out.println("<script>alert('Database error
occurred.');</script>");

            }
```

```
RequestDispatcher rd =
request.getRequestDispatcher("User_Home.jsp");
            rd.include(request, response);
        }
    }
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    public String getServletInfo() {
        return "Handles movie file upload";
    }
}
```

APPENDIX 2

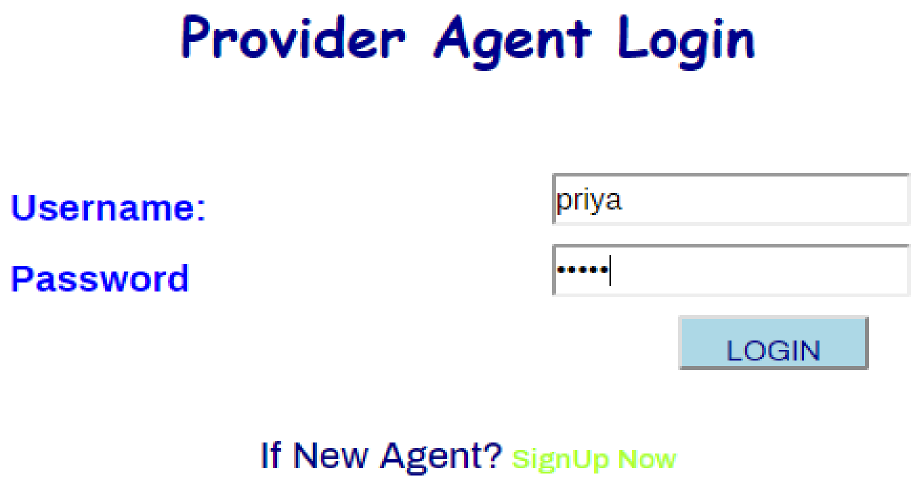
OUTPUT



Provider Agent Registration

Username:	priya
Password:	•••••
Mobile:	1234567890
Mail Id	1croreprojects.javateam@
City:	chennai
Profile Picture:	Choose File Chrysanthemum.jpg

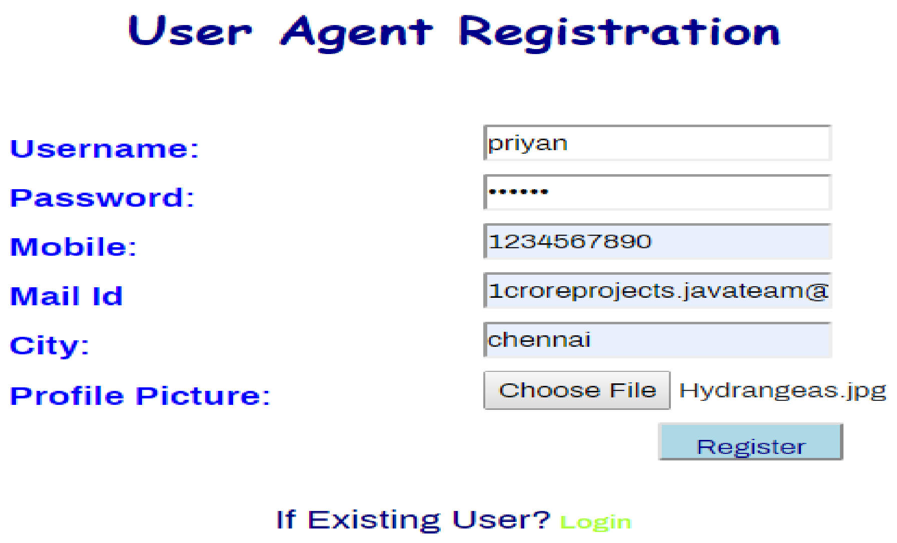
In the above image shows the provider Agent Registration with Register.



Provider Agent Login

Username:	priya
Password	•••••

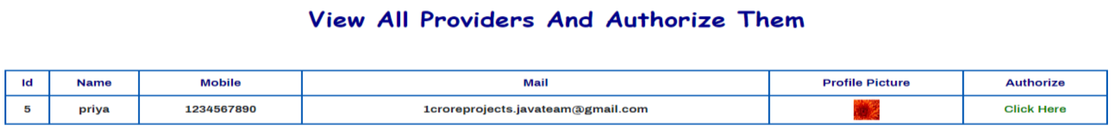
In the above image, shows the provider Agent can login the page with Username and Password.



User Agent Registration

Username:	priyan
Password:	••••••
Mobile:	1234567890
Mail Id	1croreprojects.javateam@
City:	chennai
Profile Picture:	Choose File Hydrangeas.jpg

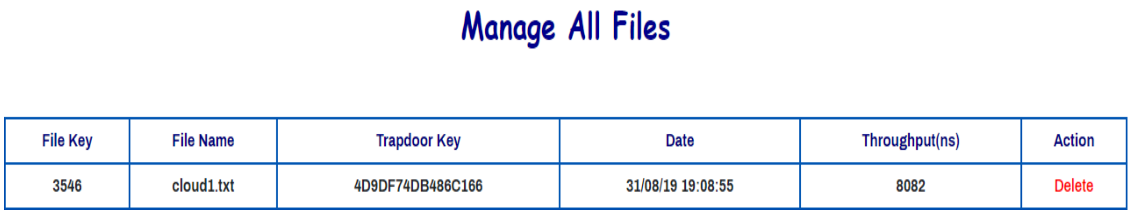
In the above page, User Agent can enter their Personal Details.



View All Providers And Authorize Them

Id	Name	Mobile	Mail	Profile Picture	Authorize
5	priya	1234567890	1croreprojects.javateam@gmail.com		Click Here

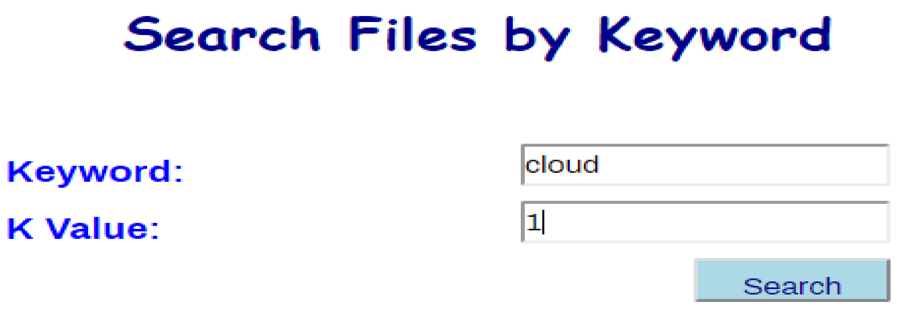
In the above page, View All Provider and Authorize Them can be added Successfully.



Manage All Files

File Key	File Name	Trapdoor Key	Date	Throughput(ns)	Action
3546	cloud1.txt	4D9DF74DB486C166	31/08/19 19:08:55	8082	Delete

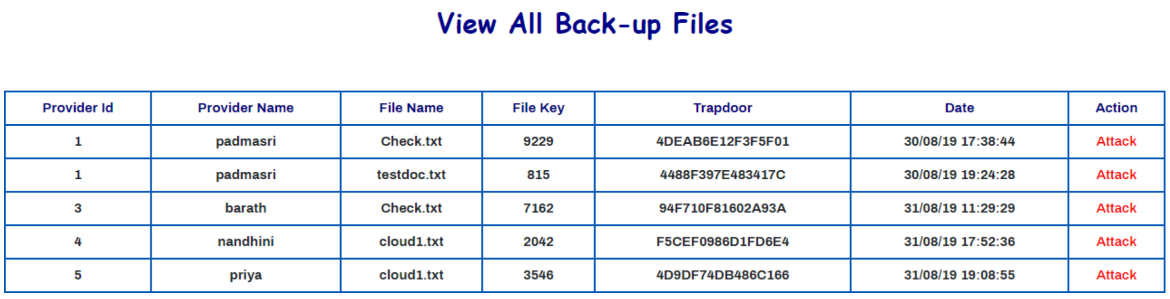
In the above image, Manage All Files can be Delete.



Search Files by Keyword

Keyword:	cloud
K Value:	1

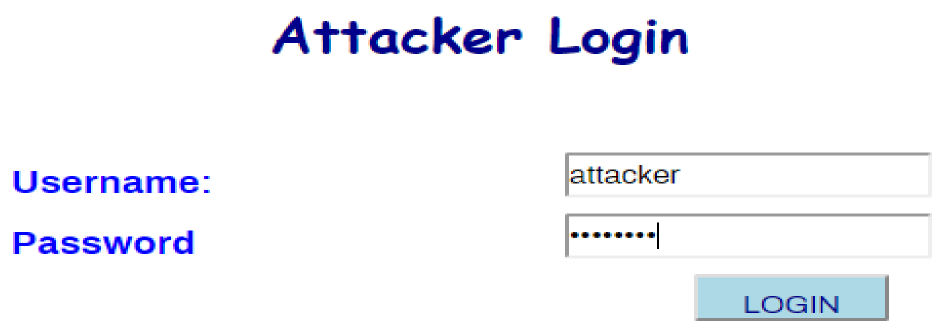
In the above image, shows the provider Agent Registration with Register.



View All Back-up Files

Provider Id	Provider Name	File Name	File Key	Trapdoor	Date	Action
1	padmasri	Check.txt	9229	4DEAB6E12F3F5F01	30/08/19 17:38:44	Attack
1	padmasri	testdoc.txt	815	4488F397E483417C	30/08/19 19:24:28	Attack
3	barath	Check.txt	7162	94F710F81602A93A	31/08/19 11:29:29	Attack
4	nandhini	cloud1.txt	2042	F5CEF0986D1FD6E4	31/08/19 17:52:36	Attack
5	priya	cloud1.txt	3546	4D9DF74DB486C166	31/08/19 19:08:55	Attack

In the above image, shows the View All Back-up Files with attack.



Attacker Login

Username:	attacker
Password	•••••••••

In the above image, Attacker Login with Username and Password with Login.

REFERENCES

1. Bhardwaj, F. Al-Turjman, M. Kumar, T. Stephan, and L. Mostarda, ‘‘Capturing-the-invisible (CTI): Behavior-based attacks recognition in IOT-oriented industrial control systems’’, Vol.8, pp. 104956-104966, 2020.
2. S. Srivastava, ‘‘Image forensics based on lighting estimation”, Vol.19, No. 03, 1950014 (2019).
3. N. Uddin, ‘‘Image forensic based on lighting estimation’’, Vol.19, No. 03, 1950014 (2019).
4. J. Li, Y. Zhang, X. Chen, and Y. Xiang, ‘‘Secure attribute-based data sharing for resource- limited users in cloud computing’’,
5. Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, and I. You, ‘‘Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing”.
6. The Open Stack Project. OSSA-2015-006: Unauthorized Delete of Versioned Swift Object.
7. The Open Stack Project. OSSA-2015-016: Information Leak Via Swift Tempurls.
8. T. Noora, S. Zeadallyb, A. Alfazic, and Q. Sheng, “Mobile cloud computing: Challenges and future research directions,” Journal of Network and Computer Applications, vol. 115, pp.70– 85, Aug. 2018.
9. C. euang, X. Mei, G. Zhao, and J. Wu et al, “Transaction modeling and execution analysis of uncertainty composition service in mobility computing environments,” Science China: Information Science, vol. 45, no. 1,pp.70-96, Jan. 2015.
10. S. Deng, L. Huang, H. Wu, W. Tan, J. Taheri, A. Zomaya, and Z. Wu. “Toward Mobile Service Computing: Opportunities and Challenges,” IEEE Cloud Computing, vol. 3, no.4, pp32-41, 2016.
11. K. Sim, “Agent-based Approaches for Intelligent Inter cloud Resource Allocation,” IEEE Trans. on Cloud Computing, 2018, early access, DOI 10.1109/TCC.2016.2628375.
12. K. Sim, “Agent-based cloud commerce”, in Proc. of the IEEE International Conf. on Industrial Engineering and Engineering Management, Hong Kong, China ,2006, pp. 717-721.