

INTEGRATION OF MULTI BANK MULTI USER IN SINGLE CARD WITH USER BEHAVIOR MONITORING USING HMM & FORMULA VERIFICATION

A PROJECT REPORT

Submitted by

A.N.JAYAKEERTHI(412513104044)

R.PAVITHRA(412513104081)

V.PRITHIKKA(412513104087)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



SRI SAIRAM ENGINEERING COLLEGE::CHENNAI 600 044

ANNA UNIVERSITY::CHENNAI 600 025

APRIL 2017

ANNA UNIVERSITY::CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“INTEGRATION OF MULTI BANK MULTI USER IN SINGLE CARD WITH USER BEHAVIOR MONITORING USING HMM & FORMULA VERIFICATION”** is the bonafide work of **“A.N.JAYAKEERTHI(412513104044), R.PAVITHRA(412513104081), V.PRITHIKKA(412513104087)”** who carried out the project work under my supervision.

SIGNATURE

Dr.B.Latha,M.E, Ph.D

HEAD OF DEPARTMENT

PROFESSOR

Department of Computer

Science and Engineering

Sri Sairam Engineering College,

Chennai-44.

SIGNATURE

Mrs.M.Kiruthiga Devi,M.E

SUPERVISOR

ASSISTANT PROFESSOR

Department of Computer

Science and engineering

Sri sairam engineering College,

Chennai-44.

Submitted for the Anna University Examination held on_____

(INTERNAL EXAMINER)

(EXTERNAL EXAMINER)

ACKNOWLEDGEMENT

We would like to express our profound gratitude to our beloved (Late) Chairman, **Thiru.MJF.Ln.LEO MUTHU** for having provided plenty of resources.

We express our gratitude to our **CEO Mr. J. SAI PRAKASH** for his constant encouragement in completing this project.

We extend solemn thanks to our principal **Dr.C.V.JAYAKUMAR** for having given us spontaneous and whole-hearted encouragement for completing this project.

We are indebted to our Head of the Department **Dr.B.LATHA**, for her support during the entire course of this project.

Our sincere thanks to our Project Coordinator **Mrs.R.SHARMIKHA SREE**, Associate Professor for her kind support in bringing out this project successfully.

Our sincere thanks to our Supervisor **Mrs.M.KIRUTHIGA DEVI**, Assistant Professor, for helping us in successful completion of this project.

Finally, we would like to thank all the staff members of the Department of Computer Science and Engineering, our parents, friends and all others who contributed directly and indirectly for the successful completion of our project.

ABSTRACT

Usage of ATM cards for transaction has become very common, hence managing of multiple accounts becomes tedious. A solution to this issue is to integrate multi bank multi user in a single card along with user behavior monitoring. In this system Big Data, Business analytical and RFID like technology are integrated to provide solution to this most challenge oriented activity. The implementation involves developing an application for a Banking sector particularly for a Debit / ATM card section. RFID smart card is used as ATM Card for transaction, users can create account and get the card from the bank. They can access their cards with unique identity numbers accordingly. Users can include their family members' accounts details in the same card and their behavior is monitored through HMM algorithm, verified using OTP and authenticated using formula based method. They can withdraw cash from their accounts after successful authentication of the corresponding PIN numbers and OTP along with formula based authentication.

TABLE OF CONTENTS

CHAP NO:	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1.	INTRODUCTION	
	1.1Project Introduction	1
	1.2Characteristics Of Big Data	2
	1.3Classification Of Big Data	3
	1.4Advantages of Big Data	4
	1.5Disadvantages of Big Data	4
2.	LITERATURE SURVEY	7
3.	SYSTEM ANALYSIS	
	3.1Objective	11
	3.2Existing System	11
	3.3 Proposed System	12
4.	SYSTEM SPECIFICATIONS	
	4.1Requirement Analysis	13
	4.2Hardware and software requirement	
	4.2.1 Hardware Requirement	13
	4.2.2 Software Requirement	13
	4.3 Technology used	14
	4.3.1 MYSQL	14
	4.3.2 N-Tier Architecture	19

4.3.3	JAVA	20
4.3.4	Apache Tomcat	25
5.	PROJECT DESCRIPTION	
5.1	Overview Of The Project	27
5.2	Modules	28
5.3	Modules Explanation	29
5.3.1	Creation Of Database For Storing User Details	29
5.3.2	Integration Of Multiple Bank and Multiple Use in RFID Card	30
5.3.3	Tracking Of Transaction Using HMM Model	30
5.3.4	Formula Based Authentication	30
6.	SYSTEM DESIGN	
6.1	Architecture Diagram	33
6.2	Data Flow Diagram	34
6.3	UML Diagrams	36
6.3.1	Use Case Diagram	37
6.3.2	Sequence Diagram	38
6.3.3	Collaboration Diagram	39
6.3.4	Activity Diagram	40
6.3.5	Class Diagram	41
7.	SYSTEM TESTING	
7.1	Coding Standard	43
7.2	Testing Strategies	44
8.	CONCLUSION AND FUTURE ENHANCEMENT	
8.1	Conclusion	48
8.2	Future Enhancement	48

APPENDEX-1: SOURCE CODING	49
APPENDEX-2: SCREEN SHOTS	66
REFERENCES	69

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
4.1	MYSQL Architecture	16
4.2	Internal Components of MYSQL	17
4.3	MYSQL Database	18
4.4	N-Tier Architecture	19
4.5	Software Development Process	22
4.6	API and JVM	23
6.1	System Architecture	33
6.2	User registration to bank server	34
6.3	Money withdrawal with pin verification	34
6.4	User Behavior monitoring using HMM	35
6.5	Use Case Diagram	37
6.6	Sequence Diagram	38
6.7	Collaboration Diagram	39
6.8	Activity Diagram	40
6.9	Class Diagram	41

LIST OF ABBREVIATIONS

- | | |
|----------|---|
| 1. RFID | - Radio Frequency Identification |
| 2. RDBMS | - Relational DataBase Management System |
| 3. DAS | - Direct Attached Storage |
| 4. NAS | - Network Attached Storage |
| 5. SAN | - Storage Area Network |
| 6. HDD | - Hard Disk Drive |
| 7. HDFS | -Hadoop Distributed File System |
| 8. SQL | -Structured Query Language |

CHAPTER 1

INTRODUCTION

1.1 PROJECT INTRODUCTION

Information technology (IT) not only introduces convenience, but creates many new improvement opportunities which were impossible in the past. For example, advances of business intelligence (BI) methods and data mining techniques have brought huge improvements to modern business operations. Nowadays, in the “big data era,” a massive amount of data is available for all kinds of industrial applications. For example, the cloud service can be considered as a data warehouse which provides a useful source of data.

Wireless sensor networks can be used to collect useful data ubiquitously. In other words, it is now easier to collect data than ever before. That being said, extracting and utilizing useful information from such huge and dynamic databases for “big data” is far from easy. Since these data are linked to real-time events, they can be employed, if properly (e.g., via BI schemes), for rescheduling or replanning activities in business applications which finally reduce the level of risk and improve profitability and efficiency of the operations. This undoubtedly can supplement traditional optimization techniques, which are *a priori* in nature. Such data migration problem is very important yet challenging as the volume of big data is growing quickly.

Developed a service optimization model for handling big data stored in cloud systems when privacy is a critical concern (e.g., the medical data). Service quality may be compromised if a cloud server refuses to provide the data due to the privacy issue. Such optimization model can maximize the service quality and is verified by a simulation study. Another application of big data is on smart grids. The demand of a cloud-based smart grid system and derived the optimal pricing

strategy, based on the big data on real-time consumption. The approach is possible due to the data mining algorithm the authors developed. The relationship between cloud systems and big data models will be further discussed in Section II. Owing to the importance of big data analytics for business applications, this paper is developed. With respect to the core topic on big data analytics for business operations and risk management.

The three big sections are, namely: 1) BI and data mining; 2) industrial systems reliability and security; and 3) business operational risk management (ORM). Each of these sections: 1) examines some carefully selected papers; 2) outlines the related research challenges; and 3) proposes the future research directions. To the best of our knowledge, this is the first paper in the literature which focuses on how big data analytics can be employed for reducing systems risk and enhancing efficiency in business operations.

1.2 CHARACTERISTICS OF BIG DATA

Big data is a term utilized to refer to the increase in the volume of data that are difficult to store, process, and analyze through traditional database technologies. The nature of big data is indistinct and involves considerable processes to identify and translate the data into new insights. The term “big data” is relatively new in IT and business. However, several researchers and practitioners have utilized the term in previous literature. For instance, referred to big data as a large volume of scientific data for visualization. Several definitions of big data currently exist.” Meanwhile and defined big data as characterized by three Vs: volume, variety, and velocity. The terms volume, variety, and velocity were originally introduced by Gartner to describe the elements of big data challenges. IDC also defined big data technologies as “a new generation of technologies and architectures, designed to economically extract value from very large volumes of a

wide variety of data, by enabling the high velocity capture, discovery, and/or analysis.” specified that big data is not only characterized by the three Vs mentioned above but may also extend to four Vs, namely, volume, variety, velocity, and value This 4V definition is widely recognized because it highlights the meaning and necessity of big data.

- **VOLUME**

Refers to the amount of all types of data generated from different sources and continue to expand. The benefit of gathering large amounts of data includes the creation of hidden information and patterns through data analysis Collecting longitudinal data requires considerable effort and underlying investments. Nevertheless, such mobile data challenge produced an interesting result similar to that in the examination of the predictability of human behavior patterns or means to share data based on human mobility and visualization techniques for complex data.

- **VARIETY**

The different types of data collected via sensors, smart phones, or social networks. Such data types include video, image, text, audio, and data logs, in either structured or unstructured format. Most of the data generated from mobile applications are in unstructured format. For example, text messages, online games, blogs, and social media generate different types of unstructured data through mobile devices and sensors. Internet users also generate an extremely diverse set of structured and unstructured data.

- **VELOCITY**

It refers to the speed of data transfer. The contents of data constantly change because of the absorption of complementary data collections, introduction of previously archived data or legacy collections, and streamed data arriving from multiple sources

- **VALUE**

An important aspect of big data; it refers to the process of discovering huge hidden values from large datasets with various types and rapid generation

1.3 DISADVANTAGES OF BIGDATA

Big data has various challenges. Some of the major challenges are

- **SCALABILITY**

Scalability is the ability of the storage to handle increasing amounts of data in an appropriate manner. Scalable distributed data storage systems have been a critical part of cloud computing infrastructures. The lack of cloud computing features to support RDBMSs associated with enterprise solutions has made RDBMSs less attractive for the deployment of large-scale applications in the cloud. This drawback has resulted in the popularity of NoSQL

A NoSQL database provides the mechanism to store and retrieve large volumes of distributed data. The features of NoSQL databases include schema-free, easy replication support, simple API, and consistent and

flexible modes. Different types of NoSQL databases, such as key-value column-oriented, and document-oriented, provide support for big data. Shows a comparison of various NoSQL database technologies that provide support for large datasets.

▪ **AVAILABILITY**

Availability refers to the resources of the system accessible on demand by an authorized individual. In a cloud environment, one of the main issues concerning cloud service providers is the availability of the data stored in the cloud. For example, one of the pressing demands on cloud service providers is to effectively serve the needs of the mobile user who requires single or multiple data within a short amount of time. Therefore, services must remain operational even in the case of a security breach.

In addition, with the increasing number of cloud users, cloud service providers must address the issue of making the requested data available to users to deliver high-quality services. Lee et al. introduced a multi-cloud model called “rain clouds” to support big data exploitation. “Rain clouds” involves cooperation among single clouds to provide accessible resources in an emergency. Schroeck et al. predicted that the demand for more real time access to data may continue to increase as business models evolve and organizations invest in technologies required for streaming data and smart phones.

▪ **DATA INTEGRITY**

A key aspect of big data security is integrity. Integrity means that data can be modified only by authorized parties or the data owner to prevent

misuse. The proliferation of cloud-based applications provides users the opportunity to store and manage their data in cloud data centers. Such applications must ensure data integrity. However, one of the main challenges that must be addressed is to ensure the correctness of user data in the cloud.

▪ **TRANSFORMATION**

Transforming data into a form suitable for analysis is an obstacle in the adoption of big data. Owing to the variety of data formats, big data can be transformed into an analysis workflow in two ways

CHAPTER 2

LITERATURE REVIEW

2.1 RECENT DEVELOPMENT IN BIG DATA ANALYTICS FOR BUSINESS OPERATIONS AND RISK MANAGEMENT

Clouds provide a powerful computing platform that enables individuals and organizations to perform variety levels of tasks such as: use of online storage space, adoption of business applications, development of customized computer software, and creation of a “realistic” network environment. The number of people using cloud services has dramatically increased and lots of data has been stored in cloud computing environments. In this paper, three cloud service models were compared; cloud security risks and threats were investigated based on the nature of the cloud service models. Real world cloud attacks were included to demonstrate the techniques that hackers used against cloud computing systems

2.2 SUPPLY CHAIN RISK ANALYSIS WITH MEAN VARIANCE MODELS A TECHNICAL REVIEW

One-manufacturer and -retailer supply chain facing uncertain demand. The manufacturer sells a perishable product to the retailer. Different from the traditional supply chain models based on risk neutrality, this article takes the viewpoint of the behavioural theory and assumes that the retailer is loss averse. The objective is to design the supply contract that provides a win–win coordination mechanism between the manufacturer and the retailer. Specifically, two types of contracts, buyback contract and markdown-price contract, are analysed. This article investigates the role of contracts mitigating the loss-aversion effect, which decreases the order quantity of the retailer and the total channel profit. As a comparison, these two types of contracts that ignore loss aversion are also

discussed. The analytical and numerical results shed light on how a manufacturer can design a contract to improve the channel performance. In particular, it is shown that these two types of contracts can coordinate the supply chain and arbitrarily allocate the expected channel profit between the manufacturer and the retailer.

2.3 NOISE TOLERANCE UNDER RISK MINIMIZATION

Noise tolerant learning of classifiers is explored. The actual training set given to the learning algorithm is obtained from this ideal data set by corrupting the class label of each example. The probability that the class label of an example corrupted is a function of the feature vector of the example. This would account for most kinds of noisy data one encounters in practice. We show that risk minimization under 0-1 loss function has impressive noise tolerance properties and that under squared error loss is tolerant only to uniform noise; risk minimization under other loss functions is not noise tolerant. We conclude the paper with some discussion on implications of these theoretical results.

2.4 ITEM-LEVEL RFID IN A RETAIL SUPPLY CHAIN WITH STOCKOUT-BASED SUBSTITUTION

Cloud computing and internet of things (IoT) have provided a promising opportunity to resolve the challenges caused by the increasing transportation issues. We present a novel multilayered vehicular data cloud platform by using cloud computing and IoT technologies. Two innovative vehicular data cloud services, an intelligent parking cloud service and a vehicular data mining cloud service, for vehicle warranty analysis in the IoT environment are also presented. Two modified data mining models for the vehicular data mining cloud service, a Naïve Bayes model and a Logistic Regression model, are presented in detail. Challenges and directions for future work are also provided.

2.5 OPTIMAL BI-OBJECTIVE REDUNDANCY ALLOCATION FOR SYSTEMS RELIABILITY AND RISK MANAGEMENT

The concept of connecting multiple objects together in an Internet-based architecture. Applications built around this concept are constantly growing in variety and quantity. The Internet of Things (IoT) has provided a promising opportunity to build powerful applications by leveraging the growing ubiquity of Radio Frequency Identification (RFID) and wireless sensors devices. In IoT enabled systems data from all connected devices can be generated quite rapidly the volume may be huge and types of data can be various. Processing & Analysis provides proper management of data. There are various email systems available worldwide. The idea IoT based email system is extension to existing email system. In such a system devices can send the notifications in email format, automatic actions are performed if necessary. For such a system, Application framework is necessary and which aims to provide a framework which would allow or facilitates the users to create their IoT applications. In this paper we explained how the IoT based email system is designed with suitable framework and how data can be processed for further analysis.

2.6 RISK MANAGEMENT MODELS FOR SUPPLY CHAIN: A SCENARIO ANALYSIS OF OUTSOURCING TO CHINA

Cloud computing introduces flexibility in the way an organization conducts its business. On the other hand, it is advisable for organizations to select cloud service partners based on how prepared they are owing to the uncertainties present in the cloud. This study is a conceptual research which investigates the impact of some of these uncertainties and flexibilities embellished in the cloud. First, we look at the assessment of security and how it can impact the supply chain operations using entropy as an assessment tool. Based on queuing theory, we look

at how scalability can moderate the relationship between cloud service and the purported benefits. We aim to show that cloud service can only prove beneficial to supply partners under a highly secured, highly scalable computing environment and hope to lend credence to the need for system thinking as well as strategic thinking when making cloud service adoption decisions.

CHAPTER 3

SYSTEM ANALYSIS

3.1 OBJECTIVE

The most fundamental challenge for Big Data applications is to explore the large volumes of data and extract useful information or knowledge for future actions. In this project, the multiple accounts of multiple banks in a single card. Another important objective is to overcome an emergency situation where the user don't have money in his/her account. It also enhance the security and also prevents unauthorized access formula based authentication and user behavior tracking with HMM model.

3.2 EXISTING SYSTEM

Big data is used in many applications in banking sector such as fraud detection and prevention Big data is also used to provide enhanced compliance to cater billions of customers' needs in a meaningful way. RFID cards are used in item level inventory tracking and attendee tracking.

DISADVANTAGES

- There is no RFID technology
- Security is less
- Every user having individual card in family

3.2 PROPOSED SYSTEM

- In this system multiple accounts are integrated in a single card. When the user creates a account a formula is to be attached to his account.(any variable (a-z), arithmetic sign(+,-)).
- As the user swipes the RFID card, various accounts linked to it are displayed where he/she needs to choose one.
- If the user chooses his/her own account the user behavior is tracked using HMM , if there is the drastic change in the behavior the panel with variables and their corresponding value is displayed .
- The user computes using his formula and withdraws the amount required.
- If the user chooses some other account, the user behavior is checked and the OTP is send to the corresponding account owner, and the formula is revealed to the user.
- He computes value after the panel is displayed, enters the OTP and withdraws the cash.
- The owner can change the formula n number of times.

3.2.1 ADVANTAGES

- This system reduces multiple numbers of ATM cards for every account.
- Shoulder surfing attack is prevented.
- User can withdraw cash using their single ATM card from their family members' account in case of emergency.
- Security is ensured by the implementation of formula based authentication.
- Big data helps to analyze huge volume of data.

CHAPTER 4

SYSTEM SPECIFICATIONS

4.1 REQUIREMENT ANALYSIS

Requirement analysis determines the requirements of a new system. This project analyses on product and resource requirement, which is required for this successful system. The product requirement includes input and output requirements it gives the wants in term of input to produce the required output. The resource requirements give in brief about the software and hardware that are needed to achieve the required functionality.

4.2 HARDWARE AND SOFTWARE SPECIFICATION

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. The software requirements are the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it.

4.2.1 HARDWARE REQUIREMENT

- Hard disk : 120 GB
- Monitor : 15' color with vgi card support
- Ram : Minimum 256 MB
- Processor : Pentium iv and above (or) equivalent
- Processor speed : Minimum 500 MHZ

4.2.2 SOFTWARE REQUIREMENT

- Operating system : Windows XP
- Languages : Java
- Data Base : Mysql
- IDE : Net Beans

4.3 TECHNOLOGIES USED

4.3.1 MYSQL

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by MySQL AB. MySQL AB is a commercial company, founded in 1995 by the MySQL developers. It is a second generation Open Source company that unites Open Source values and methodology with a successful business model.

The MySQL software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. MySQL is a registered trademark of MySQL AB.

▪ MySQL IS A DATABASE MANAGEMENT SYSTEM

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a

computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- **MySQL DATABASES ARE RELATIONAL**

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data. The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax. SQL is defined by the ANSI/ISO SQL Standard.

▪ MySQL ARCHITECTURE

My SQL architecture consists of three layers as shown in fig 4.1.

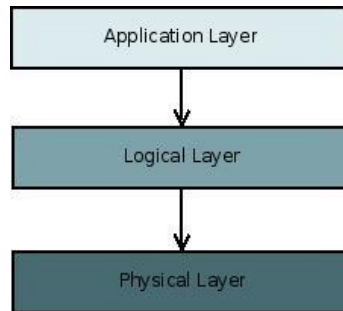


Fig 4.1 MySQL Architecture

- The application layer contains common network services for connection handling, authentication and security. This layer is where different clients interact with MySQL these clients can be written in different API's: .NET, Java, C, C++, PHP, Python, Ruby, Tcl, Eiffel, etc...
- The Logical Layer is where the MySQL intelligence resides; it includes functionality for query parsing, analysis, caching and all built-in functions (math, date...). This layer also provides functionality common across the storage engines.
- The Physical Layer is responsible for storing and retrieving all data stored in “MySQL”. Associated with this layer are the storage engines, which MySQL interacts with very basic standard API's.

▪ SOME INTERNAL COMPONENTS

The internal components of mysql are shown in fig 4.2.

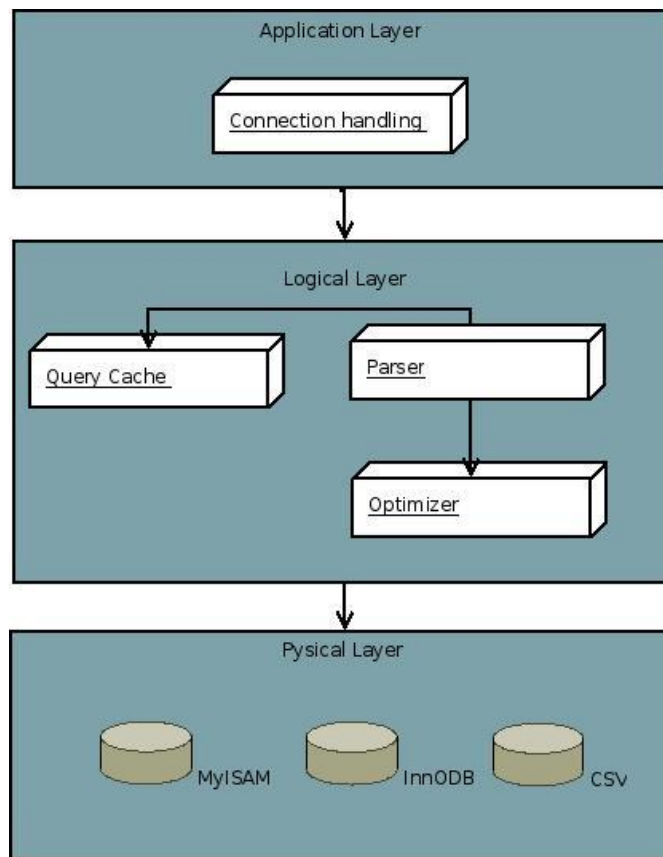


Fig 4.2 Internal components of MySQL

Each client connection gets its own thread within the server process. When clients (applications) connect to the MySQL server, the server needs to authenticate them. Before even parsing the query, the server consults the query cache, which only stores SELECT statements, along with their result sets. The storage engine does affect how the server optimizes query.

➤ STORAGE ENGINES

MySQL supports several storage engines that act as handlers for different table types. MySQL storage engines include both those that handle transaction-safe tables and those that handle non-transaction-safe tables as shown in fig 4.3.

Feature	MyISAM	Memory	InnoDB	Archive	NDB
Storage limits	256TB	RAM	64TB	None	384EB
Transactions	No	No	Yes	No	Yes
Locking granularity	Table	Table	Row	Row	Row

Fig 4.3 MySQL Database

➤ MYSQL FILES

In Linux the configuration file is typically located at `/etc/mysql/my.cnf`, but may vary for the different Linux flavours, this also applies to the database files themselves which are located in the `/var/lib/MySQL`.

Regardless of the storage engine, every MySQL table you create is represented, on disk, by an `.frm` file, which describes the table's format (i.e. the table definition).

`/var/lib/mysql/db.frm` #Table definition

`/var/lib/mysql/db.MYD` #MyISAM data file

`/var/lib/mysql/db.MYI` #MyISAM Index file

`/var/lib/mysql/ibdata1` #Innodb data file

➤ MYSQL CLUSTER

MySQL Cluster is a technology that enables clustering of in-memory databases in a shared-nothing system. The shared-nothing architecture enables the system to work with very inexpensive hardware, and with a minimum of specific requirements for hardware or software.

MySQL Cluster is designed not to have any single point of failure. In a shared-nothing system, each component is expected to have its own memory and disk, and the use of shared storage mechanisms such as network shares, network file systems, and SANs is not recommended or supported.

4.3.2 N-TIER ARCHITECTURE

The architecture of N-Tier is shown in fig 4.4. It describes about the process done by the content developers, carriers, end users and OEMs.



Fig 4.4 N-Tier Architecture

4.3.3 JAVA

The Java platform is the ideal platform for network computing. Running across all platforms from servers to cell phones to smart cards, Java technology unifies business infrastructure to create a seamless, secure, networked platform for your business. The Java platform benefits from a massive community of developers and supporters that actively work on delivering Java technology-based products and services as well as evolving the platform through an open, community-based, standards organization known as the Java Community Process program. Java technology can be found in cell phones, on laptop computers, on the Web, and even trackside at Formula One Grand Prix races.

- **BUSINESS BENEFITS**

- A richer user experience - Whether you're using a Java technology-enabled mobile phone to play a game or to access your company's network, the Java platform provides the foundation for true mobility. The unique blend of mobility and security in Java technology makes it the ideal development and deployment vehicle for mobile and wireless solutions.
- The ideal execution environment for Web services - The Java and XML languages are the two most extensible and widely accepted computing languages on the planet, providing maximum reach to everyone, everywhere, every time, to every device and platform.
- Enabling business from end to end - Java offers a single, unifying programming model that can connect all elements of a business infrastructure.

▪ **FEATURES OF JAVA**

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is considered by many as one of the most influential programming languages of the 20th century, and widely used from application software to web application.

▪ **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
- Portal
- Distributed
- High performance

In the Java programming language, all source code is first written in plain text files ending with the `.java` extension. Those source files are then compiled into `.class` files by the `javac` compiler. A `.class` file does not contain code that is native to your processor; it instead contains *byte codes* — the machine language of the Java Virtual Machine (Java VM). The `java` launcher tool then runs your application with an instance of the Java Virtual Machine as shown in fig 4.5.

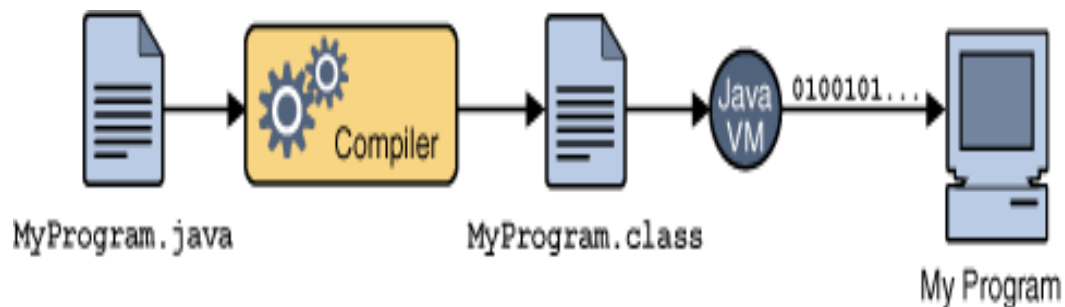


Fig 4.5 Software Development Process

Because the Java VM is available on many different operating systems, the same `.class` files are capable of running on Microsoft Windows, the Solaris Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the Java Hotspot virtual machine, perform additional steps at runtime to give your application a performance boost. This includes various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code.

▪ **JAVA PLATFORM**

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms

can be described as a combination of the operating system and underlying 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*
- The *Java Application Programming Interface (API)*

▪ **JAVA VIRTUAL MACHINE**

A Java virtual machine (JVM) is a virtual machine that can execute Java byte code. It is the code execution component of the Java software platform. A Java virtual machine is a program which executes certain other programs, namely those containing Java byte code instructions. JVMs are most often implemented to run on an existing operating system, but can also be implemented to run directly on hardware. A JVM provides an environment in which Java byte code can be executed, enabling such features as automated exception handling, which provides root-cause debugging information for every software error (exception), independent of the source code.

▪ **JAVA API**

The API is a large collection of ready-made software components as shown in fig 4.6 that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as *packages*.

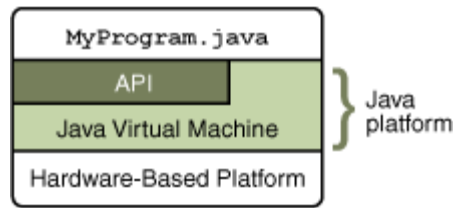


Fig 4.6 API and JVM

▪ **JAVA RUNTIME ENVIRONMENT**

The Java Runtime Environment, or JRE, is the software required to run any application deployed on the Java Platform. End-users commonly use a JRE in software packages and Web browser plug-in. Sun also distributes a superset of the JRE called the Java 2 SDK , which includes development tools such as the Java compiler, Javadoc, Jar and debugger.

▪ **JAVASERVER PAGE**

Java Server Pages (JSPs) are server-side Java EE components that generate responses, typically HTML pages, to HTTP requests from clients. JSPs embed Java code in an HTML page by using the special delimiters `<%` and `%>`. A JSP is compiled to a Java *servlet*, a Java application in its own right, the first time it is accessed. After that, the generated servlet creates the response.

▪ **SWING APPLICATION**

Swing is a graphical user interface library for the Java SE platform. It is possible to specify a different look and feel through the pluggable look and feel system of Swing. Clones of Windows, GTK+ and Motif are supplied by Sun. Apple also provides an Aqua look and feel for Mac OS X. Where prior implementations of these looks and feels may have been

considered lacking, Swing in Java SE 6 addresses this problem by using more native GUI widget drawing routines of the underlying platforms.

- **JAVA PLATFORM, STANDARD EDITION**

Java Platform, Standard Edition or Java SE is a widely used platform for programming in the Java language. It is the Java Platform used to deploy portable applications for general use. In practical terms, Java SE consists of a virtual machine, which must be used to run Java programs, together with a set of libraries (or "packages") needed to Allow the use of file systems, networks, graphical interfaces, and so on, from within those programs.

4.4.4 APACHE TOMCAT

Tomcat, developed by Apache is a standard reference implementation for Java servlets and JSP. It can be used standalone as a Web server or be plugged into a WebServer like Apache, Netscape Enterprise Server, or Microsoft Internet Information Server. There are many versions of Tomcat. This tutorial uses Tomcat 5.5.9 as an example. The tutorial should also apply to all later versions of Tomcat. To start Tomcat, you have to first set the JAVA_HOME environment variable to the JDK home directory using the following command. The JDK home directory is where your JDK is stored.

- **The Tomcat Manager Web Application**

The Tomcat manager web application is packaged with the tomcat server. It is installed in the context path of manager and provides the basic functionality to manage web applications running in the tomcat server from

any web browser. Some of the provided functionality includes the ability to install, start, stop, remove, and report on web applications.

- **The Server**

The first container element referenced in this snippet is the `<Server>` element. It represents the entire Catalina servlet engine and is used as a top-level element for a single Tomcat instance. The `<Server>` element may contain one or more `<Service>` containers.

- **The Service**

The next container element is the `<Service>` element, which holds a collection of one or more `<Connector>` elements that share a single `<Engine>` element. N-number of `<Service>` elements may be nested inside a single `<Server>` element.

- **The Connector**

The next type of element is the `<Connector>` element, which defines the class that does the actual Handling requests and responses to and from a calling client application.

- **The Engine**

The third container element is the `<Engine>` element. Each defined `<Service>` can have only one `<Engine>` element and this single `<Engine>` component handles all requests received by all of the defined `<Connector>` components defined by a parent service.

- **The Host**

The <Host> element defines the virtual hosts that are contained in each instance of a Catalina <Engine>. Each <Host> can be a parent to one or more web applications, with each being represented by a <Context> component.

- **The Context**

The <Context> element is the most commonly used container in a Tomcat instance. Each <Context> element represents an individual web application that is running within a defined <Host>. There is no limit to the number of contexts that can be defined within a <Host>.

CHAPTER 5

PROJECT DESCRIPTION

5.1 OVERVIEW OF THE PROJECT

Significant events occurring in the real world often trigger changes in the data that one can acquire from sources related to the event, and conversely, changes in time series data are often signs of important events happening at that time. Therefore, detecting changes in time-series data has long been a problem of great interest for researchers from various areas. This technique, which is often referred to as change-point detection, can be directly applied to various situations such as intrusion detection in computer networks , fault detection in machines, and fraud detection in credit card use.

It could also be employed to preprocess and segment time-series data. For time-series prediction, dramatic changes in the data would be detrimental to the performance of the prediction model, and therefore, the data should be segmented beforehand using a change-point detection technique. Segmenting time series data can also be used for signal processing.

One approach to the problem of change-point detection is to fit a stochastic model to the sequence of data and determine when the data deviates from the built model. For example, auto-regressive models are employed to construct a sequence of probability density functions which describe the underlying structure of the time-series data, and the deviation is evaluated by logarithmic loss. These methods rely on parametric models, and their applicability is frequently limited. In these methods, the common approach is to focus on two subsets of data that arrive in intervals before and after time t , which we call as the reference set and the test set respectively, and to evaluate the dissimilarity between the two sequences.

The effectiveness and the performance of change-point detection Techniques depend on how the two sequences are modeled, and how the two models are compared. For example, two one-class support vector machines are trained independently on the reference set and the test set, and the two resulting hyper planes are compared in the feature space to evaluate the dissimilarity between the two sets. There are also methods that focus on the subspace spanned by the trajectory matrix of the sequence or methods that focus on density ratio estimation. Many of these methods have demonstrated great performance in different settings. However, there is a downside that these methods have in common which is that all of these methods assume that there is only one vector associated with each time step.

5.2 MODULES

- A modular design reduces complexity, facilitates change (a critical aspect of software maintainability), and results in easier implementation by encouraging parallel development of different part of system. Software with effective modularity is easier to develop because function may be compartmentalized and interfaces are simplified. Software architecture embodies modularity that is software is divided into separately named and addressable components called modules that are integrated to satisfy problem requirements.
- Modularity is the single attribute of software that allows a program to be intellectually manageable. The five important criteria that enable us to evaluate a design method with respect to its ability to define an effective modular design are: Modular decomposability, Modular Comps ability, Modular Understandability, Modular continuity, Modular Protection.

- The following are the modules of the project, which is planned in aid to complete the project with respect to the proposed system, while overcoming existing system and also providing the support for the future enhancement.

- **LIST OF MODULES**

- ❖ *Creation of database for storing user details.*
- ❖ *Integration of multiple banks and multiple users in rfid card.*
- ❖ *Tracking of transaction using hmm model.*
- ❖ *Formula based authentication*

5.3 MODULES EXPLANATION

Modules are used to implement a project in a step by step manner and the possibility in the occurrence of risks is highly reduced when implemented in modules.

5.3.1 CREATION OF DATABASE FOR STORING USER DETAILS

In this module the user needs to create an account and then they are allowed to access the Network.

Once the user creates an account, they login into their account and request the job from the service provider. Based on the user's request, the service provider will process the user's request and respond to them. All the user details will be stored in the database of the Service Provider. The user Interface frame is designed to Communicate with the Server using programming language like java.

By sending the request to Server Provider, the User can access the requested data if they are authenticated by the Service Provider. Bank Service Provider contains information about the users in their Data Storage. Bank Service provider also maintains all the User information that are required for the authentication of the user at the time of logging into the account. The User information will be stored in

the Database of the Bank Service Provider. To communicate with the Client and the with the other modules of the Company server, the Bank Server will establish connection between them. For this Purpose User Interface Frame is created.

5.3.2 INTEGRATION OF MULTIPLE BANKS AND MULTIPLE USER IN RFID CARD

In this module, we can design and implementation of family member registration. Using single card like credit and debit for entire family members. But maintain unique PIN numbers for different banks. We will provide a button add “Family card” in our user card. Now user can add his family members bank atm details also along with pin number details. User can include like further bank account no, bank name, pin number same way for other family members also.

5.3.3 TRACKING OF TRANSACTION USING HMM MODEL

Hidden markov model used for user behaviour analysis of cash withdrawal. Hidden markov model is applied to understand users money withdrawal sequence in which first condition is total amount withdrawal in every month. Second one is Frequency of withdrawal of money using credit card. User can withdraw the cash as per money requirement and time frequency is also monitored & recorded. During registration of the card, user has to give a formula for secured authentication system. User can also add multiple bank accounts in single card.

5.3.4 FORMULA BASED AUTHENTICATION

In this module, security is provided by using formula like $(A+B-C)$ while registration. In this formula using alphabets and two operators like (+ and -). The formula is constant, but numbers will randomly change for every transaction. User is not required to provide the formula at any time, user is only required to submit the answer after substitution of the corresponding values in their formula. This formula based authentication is required only when user tries to withdraw money beyond the permitted 10% extra and increases the withdrawal frequency. Once user is registered by specifying his master bank account details & formula for authentication. Now user can add his family card details.

▪ 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.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- OPERATIONAL FEASIBILITY

▪ ECONOMICAL 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.

- **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 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.

- **OPERATIONAL 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.

CHAPTER 6

SYSTEM DESIGN

6.1 ARCHITECTURE DIAGRAM

Architecture diagram is a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation. It shows the relationships between different components of system.

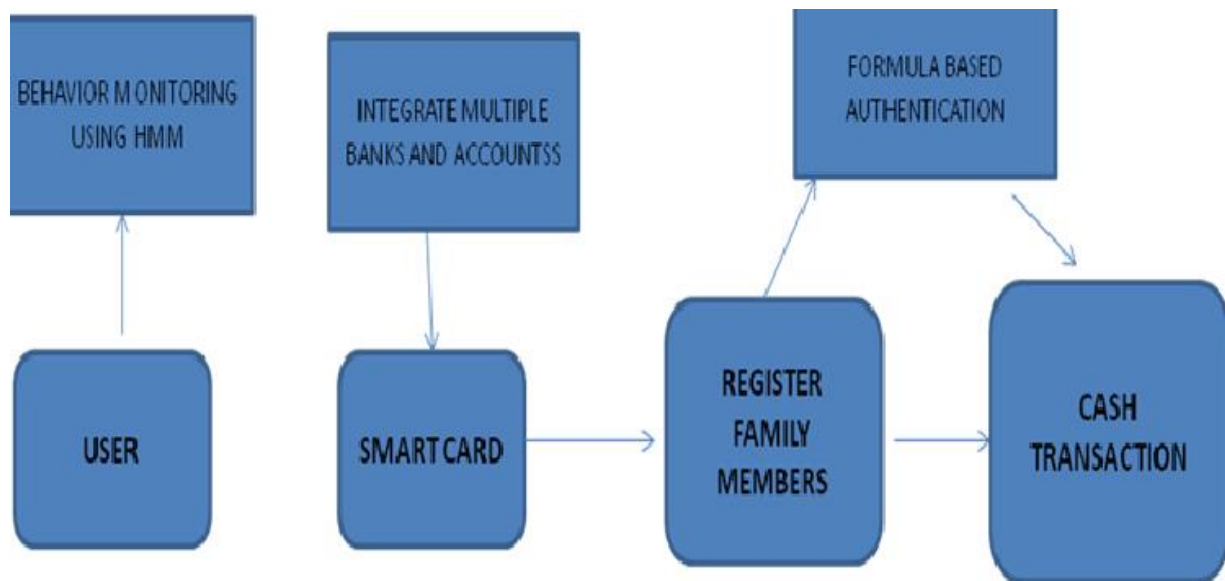


Fig 6.1 System Architecture

6.2 DATA FLOW DIAGRAM

LEVEL 0:

The users register their account details in the bank. The details are stored in the server which stores all the details in the bank. The user requests the data directly from the server. The family details are updates to the server and thus enable us to integrate various accounts in the single account.

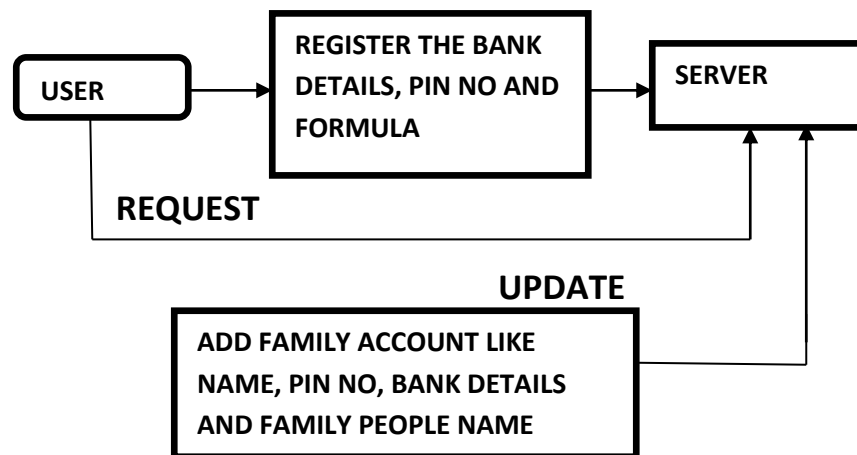


Fig 6.2 User Registration to the bank server

LEVEL 1

The user withdraws money from the server by requesting for the required amount and by entering the pin number. The pin number is verified for authentication and the requested money is provided to the requested user.

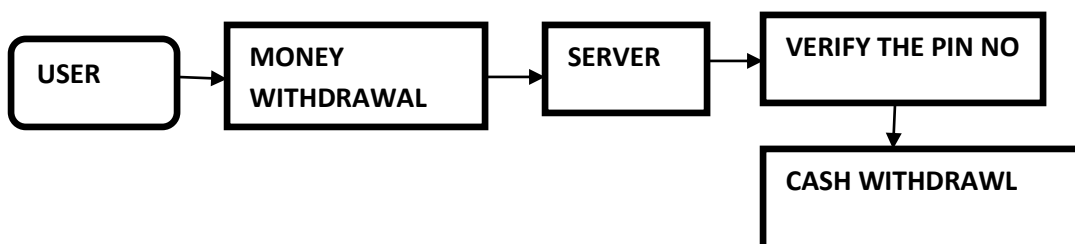


Fig 6.3 Money Withdrawal after pin number verification

LEVEL 2

In this level from the huge volume of bigdata the user behavior is monitored and various other criteria such as data analysis of excess money withdrawal is monitored and the unauthorized behavior is prevented from accessing the account

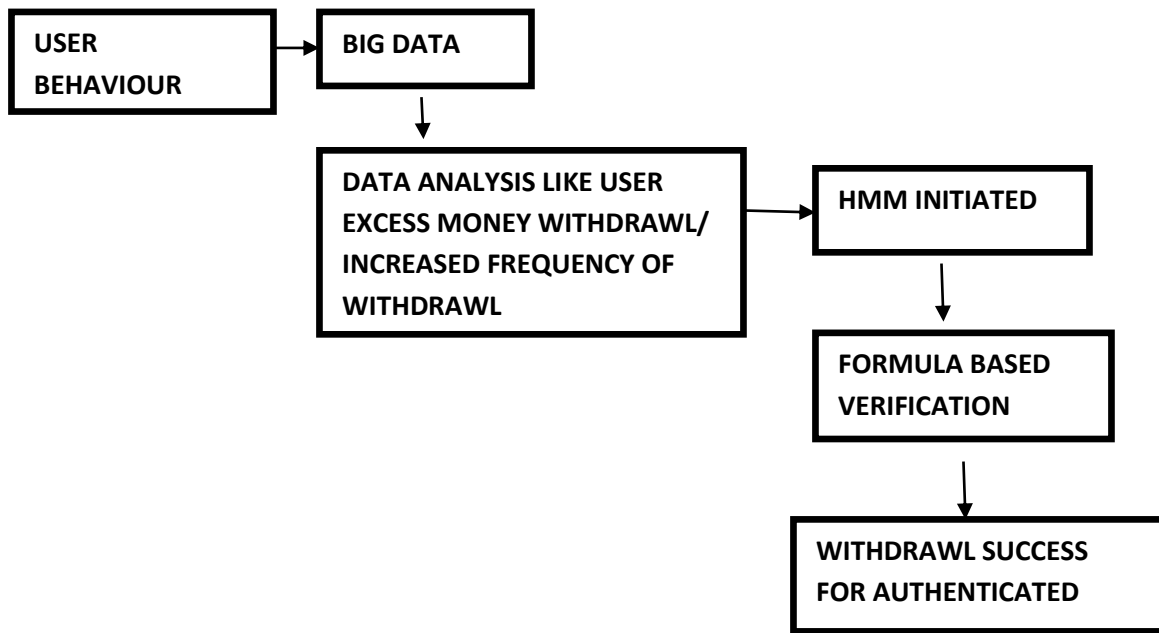


Fig 6.4 User Behavior monitoring using HMM and Formula based Authentication

6.3 UML DIAGRAMS

UML is simply another graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

ADVANTAGES

- To represent complete systems (instead of only the software portion) concepts
- To establish an explicit coupling between concepts and executable code
- To take into account the scaling factors that are inherent to complex and critical systems
- To creating a modeling language usable by both humans and machines

UML defines several models for representing systems

- The class model captures the static structure
- The state model expresses the dynamic behavior of objects
- The use case model describes the requirements of the user
- The interaction model represents the scenarios and messages flows
- The implementation model shows the work units
- Deployment model provides details pertain to process allocation

6.3.1 USE CASE DIAGRAM

Use case diagrams overview the usage requirement for system. they are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe “the meant” of the actual requirements. A use case describes a sequence of action that provide something of measurable value to an action and is drawn as a horizontal ellipse.

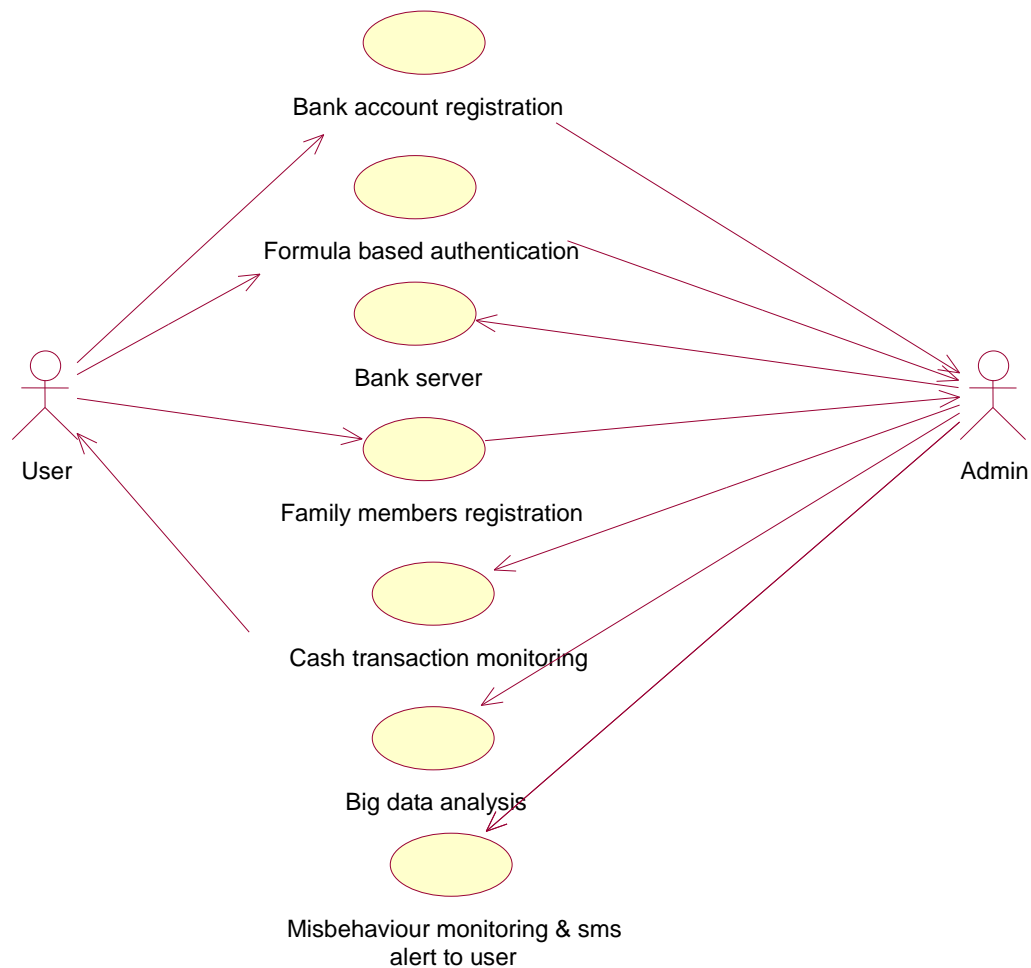


Fig 6.5 Use Case Diagram

6.3.2 SEQUENCE DIAGRAM

Sequence diagram model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and commonly used for both analysis and design purpose. Sequence diagram are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system.

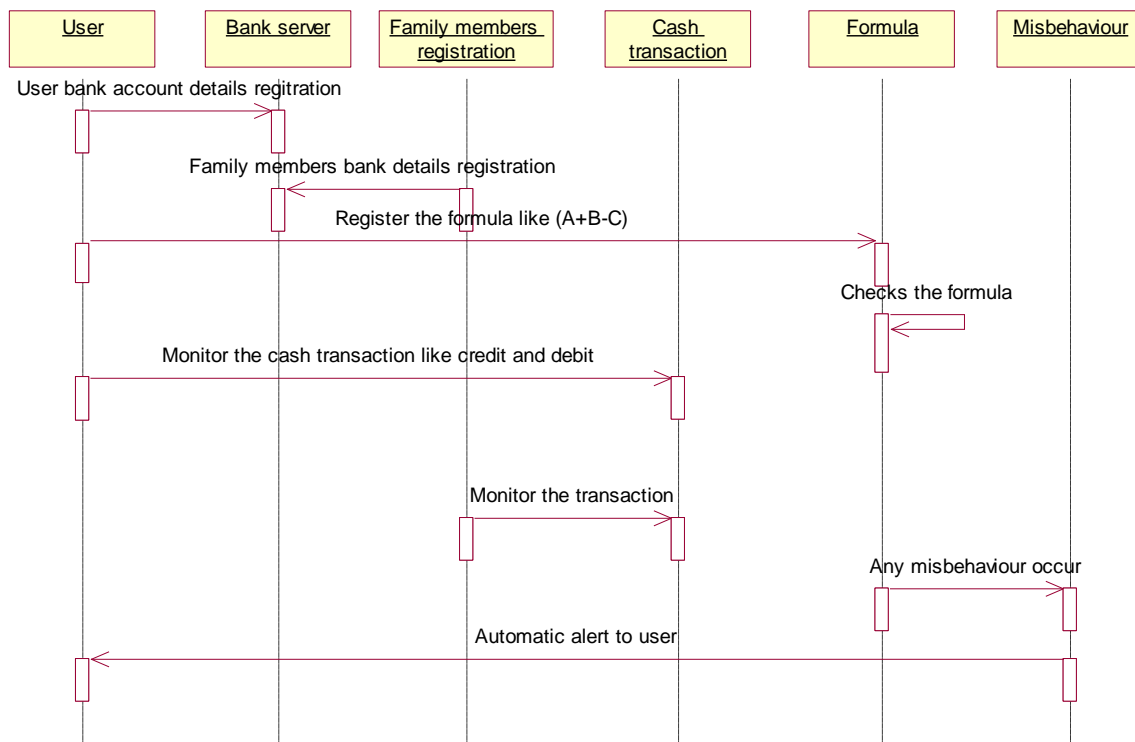


Fig 6.6 Sequence Diagram

6.3.3 COLLABORATION DIAGRAM

Another type of interaction diagram is the collaboration diagram. A collaboration diagram represents a collaboration, which is a set of objects related in a particular context, and interaction, which is a set of messages exchange among the objects within the collaboration to achieve a desired outcome.

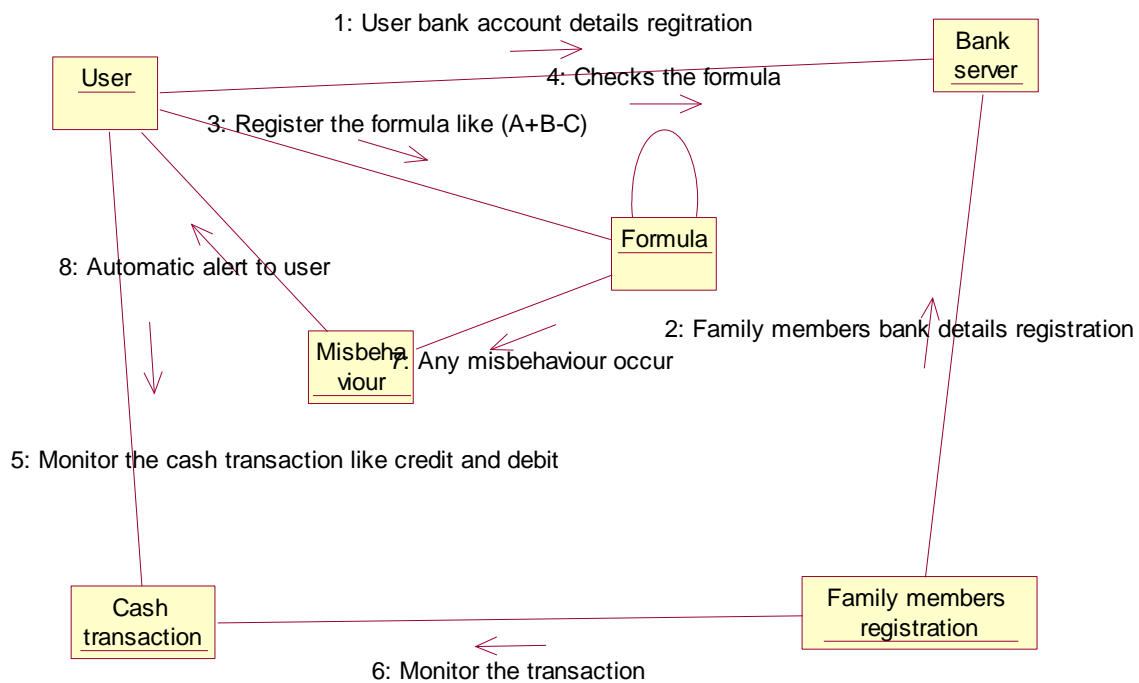


Fig 6.7 Collaboration Diagram

6.3.4 ACTIVITY DIAGRAM

Activity diagram are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. Activity diagram consist of Initial node, activity final node and activities in between.

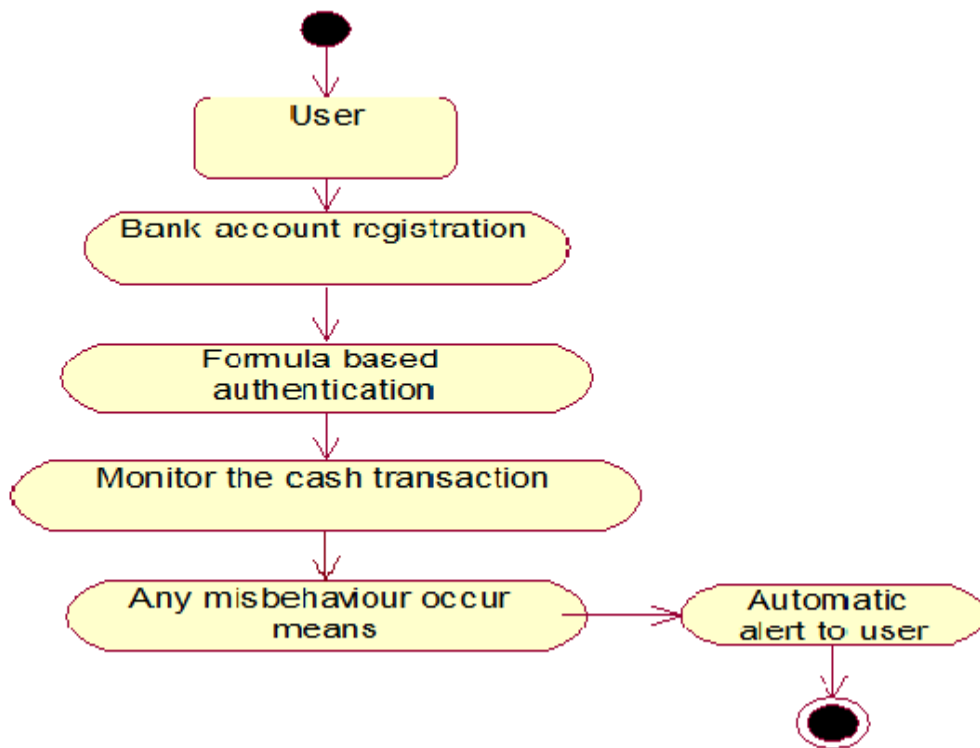


Fig 6.8 Activity Diagram

6.3.5 CLASS DIAGRAM

Class diagrams are used to caegorize various modues based on the sequence in which the project is executed

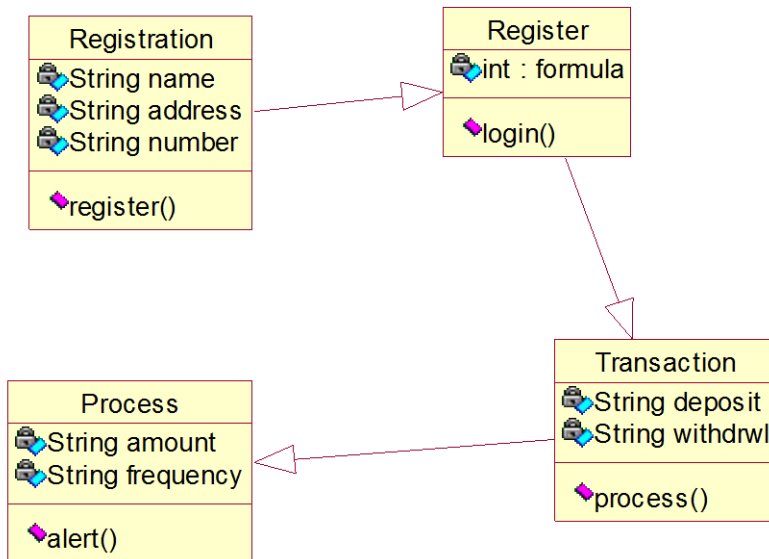


Fig 6.9 Class Diagram

CHAPTER 7

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or 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.

7.1 CODING STANDARD

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

- Program should be simple, clear and easy to understand.
- Naming conventions
- Value conventions
- Script and comment procedure
- Exception and error handling

7.2 TESTING STRATEGIES

▪ UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

▪ FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

▪ PERFORMANCE TEST

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

▪ **STRESS TEST**

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

▪ **STRUCTURED TEST**

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

▪ **INTEGRATION TESTING**

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take

untested modules and build a program structure tester should identify critical modules. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

▪ **WHITE BOX TESTING**

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases
- Graph matrices Control

▪ **BLACK BOX TESTING**

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been

adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

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

▪ **USER ACCEPTANCE TESTING**

The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing.

1. Input Screen design.
2. Output screen design.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

A project should have a proper conclusion in order to be proposed as a model and to be implemented in future. It can also be enhanced for future use and can be implemented in various projects.

8.1 CONCLUSIONS

There is sufficient supporting evidence to conclude that data-driven approaches would be a growing research methodology/ philosophy in business operations. Countless application domains can be influenced by this big data fad. BI systems are definitely on the list as such systems highly rely on the input data to generate valuable outputs. Hence it is not surprising that the research focal points have been scattered around different disciplines.. In this connection, emerging big-data-oriented research may need some adjustments. Synergizing multiple research methodologies could be one direction. Data mining is still the core engine of BI systems but previous data mining algorithms are very application-oriented. In addition, coupling with the big data era, it may be the right time to think about mining ontology's, rather than just algorithms.

8.2 FUTURE ENHANCEMENT

Another future work might be to consider a situation where the elements in each bag are correlated. In time series analysis, signals are often preprocessed by removing the predictable component. The resulting innovation time series is an i.i.d. sequence, and this is the assumption we have made in this paper. However, considering correlation in the data could be an interesting topic for additional research.

APPENDEX-1

SAMPLE CODING

```
package com.nura.ui;
import com.nura.dao.impl.CCDtlsDAOImpl;
import com.nura.dao.impl.UserDetailsDAOImpl;
import com.nura.entity.CCDtls;
import com.nura.entity.UserDetails;
import constants.Constants;
import static constants.Constants.FORMULA_CHAR;
import javax.swing.JOptionPane;
public class RelativeAC extends javax.swing.JFrame {
    private UserDetails _usrDtls;
    public RelativeAC(){

    }
    public RelativeAC(UserDetails _usrDtls) {
        this._usrDtls = _usrDtls;
        initComponents();
        tf_ccNumber.setEditable(false);
        tf_parentAC.setText(""+_usrDtls.getId());
        tf_parentAC.setEditable(false);
        tf_ccNumber.setText(_usrDtls.getCreditCardNo());
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-BEGIN: initComponents
```

```
private void initComponents() {  
    jPanel1 = new javax.swing.JPanel();  
    jLabel1 = new javax.swing.JLabel();  
    jLabel2 = new javax.swing.JLabel();  
    jLabel3 = new javax.swing.JLabel();  
    jLabel4 = new javax.swing.JLabel();  
    jLabel5 = new javax.swing.JLabel();  
    jLabel6 = new javax.swing.JLabel();  
    userName = new javax.swing.JTextField();  
    pwd = new javax.swing.JPasswordField();  
    rePwd = new javax.swing.JPasswordField();  
    mailId = new javax.swing.JTextField();  
    addr = new javax.swing.JTextField();  
    phno = new javax.swing.JTextField();  
    tf_ccNumber = new javax.swing.JTextField();  
    tf_pinNumber = new javax.swing.JTextField();  
    register = new javax.swing.JButton();  
    exit = new javax.swing.JButton();  
    jLabel7 = new javax.swing.JLabel();  
    jLabel8 = new javax.swing.JLabel();  
    jLabel9 = new javax.swing.JLabel();  
    jLabel10 = new javax.swing.JLabel();  
    cb_A1 = new javax.swing.JComboBox();  
    cb_Op = new javax.swing.JComboBox();  
    cb_A2 = new javax.swing.JComboBox();  
    cb_A3 = new javax.swing.JComboBox();  
    jLabel12 = new javax.swing.JLabel();  
}
```

```

        cb_BankName = new javax.swing.JComboBox();
        tf_BankACNo = new javax.swing.JTextField();
        jLabel13 = new javax.swing.JLabel();
        tf_parentAC = new javax.swing.JTextField();
        cb_SecOp = new javax.swing.JComboBox();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

setTitle("User Registration");
setResizable(false);
jPanel1.setForeground(new java.awt.Color(0, 51, 51));
jLabel1.setFont(new java.awt.Font("Times New Roman", 1, 18)); //
NOI18N
jLabel1.setForeground(new java.awt.Color(0, 51, 51));
jLabel1.setText("User Name");
jLabel2.setBackground(new java.awt.Color(51, 51, 51));
jLabel2.setFont(new java.awt.Font("Times New Roman", 1, 18)); //
NOI18N
jLabel2.setForeground(new java.awt.Color(0, 51, 51));
jLabel2.setText("Password");
jLabel3.setBackground(new java.awt.Color(51, 51, 51));
jLabel3.setFont(new java.awt.Font("Times New Roman", 1, 18)); //
NOI18N
jLabel3.setForeground(new java.awt.Color(0, 51, 51));
jLabel3.setText("Re-Type Password");
jLabel4.setBackground(new java.awt.Color(51, 51, 51));

```

```

        jLabel4.setFont(new java.awt.Font("Times New Roman", 1, 18)); //
NOI18N
        jLabel4.setForeground(new java.awt.Color(0, 51, 51));
        jLabel4.setText("Mail Id");
        jLabel5.setBackground(new java.awt.Color(51, 51, 51));
        jLabel5.setFont(new java.awt.Font("Times New Roman", 1, 18)); //
NOI18N
        jLabel5.setForeground(new java.awt.Color(0, 51, 51));
        jLabel5.setText("Address");
        jLabel6.setBackground(new java.awt.Color(51, 51, 51));
        jLabel6.setFont(new java.awt.Font("Times New Roman", 1, 18)); //
NOI18N
        jLabel6.setForeground(new java.awt.Color(0, 51, 51));
        jLabel6.setText("Phone Number");
        userName.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
        pwd.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
        rePwd.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
        mailId.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
        addr.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
        phno.setFont(new java.awt.Font("Times New Roman", 1, 14)); //
NOI18N

```

```

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(register,
javax.swing.GroupLayout.PREFERRED_SIZE, 103,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
        .addComponent(exit, javax.swing.GroupLayout.PREFERRED_SIZE,
104, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(115, 115, 115))
        .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(74, 74, 74)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING, false)
        .addComponent(jLabel12,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel10,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel9,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

```

```

        .addComponent(jLabel8,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel7,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel5,
javax.swing.GroupLayout.DEFAULT_SIZE, 123, Short.MAX_VALUE)
        .addComponent(jLabel4,
javax.swing.GroupLayout.DEFAULT_SIZE, 123, Short.MAX_VALUE)
        .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE, 123, Short.MAX_VALUE)
        .addComponent(jLabel2,
javax.swing.GroupLayout.DEFAULT_SIZE, 123, Short.MAX_VALUE)
        .addComponent(jLabel6,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel3,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel13,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING, false)

```

```

        .addGroup(jPanel1Layout.createSequentialGroup()
            .addComponent(cb_A1,
                javax.swing.GroupLayout.PREFERRED_SIZE, 45,
                javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELA
                TED)

            .addComponent(cb_Op,
                javax.swing.GroupLayout.PREFERRED_SIZE, 40,
                javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
                D)

            .addComponent(cb_A2, 0,
                javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addComponent(addr, javax.swing.GroupLayout.DEFAULT_SIZE,
            145, Short.MAX_VALUE)
        .addComponent(mailId)
        .addComponent(rePwd)
        .addComponent(pwd)
        .addComponent(userName)
        .addComponent(phno,
            javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(tf_ccNumber)
        .addComponent(tf_pinNumber)
        .addComponent(tf_BankACNo)

```



```

        .addComponent(cb_BankName, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(tf_parentAC))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)

.addComponent(cb_SecOp,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(cb_A3,
javax.swing.GroupLayout.PREFERRED_SIZE, 56,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap())
);

jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL,
new java.awt.Component[] {jLabel1, jLabel2, jLabel4, jLabel5});

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)

.addGroup(jPanel1Layout.createSequentialGroup())

```

```
.addGap(48, 48, 48)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
.addComponent(jLabel13,  
javax.swing.GroupLayout.DEFAULT_SIZE, 34, Short.MAX_VALUE)
```

```
.addComponent(tf_parentAC))
```

```
.addGap(18, 18, 18)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
.addComponent(userName,  
javax.swing.GroupLayout.PREFERRED_SIZE, 34,  
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addComponent(jLabel1,  
javax.swing.GroupLayout.DEFAULT_SIZE, 34, Short.MAX_VALUE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
.addComponent(pwd,  
javax.swing.GroupLayout.PREFERRED_SIZE, 34,  
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addComponent(jLabel2,  
javax.swing.GroupLayout.DEFAULT_SIZE, 34, Short.MAX_VALUE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
    .addComponent(rePwd,  
        javax.swing.GroupLayout.PREFERRED_SIZE, 34,  
        javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
    .addComponent(jLabel3,  
        javax.swing.GroupLayout.DEFAULT_SIZE, 30, Short.MAX_VALUE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
    .addComponent(mailId,  
        javax.swing.GroupLayout.PREFERRED_SIZE, 34,  
        javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
    .addComponent(jLabel4,  
        javax.swing.GroupLayout.DEFAULT_SIZE, 33, Short.MAX_VALUE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
    .addComponent(cb_BankName))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
```

```
    .addComponent(jLabel12,  
    javax.swing.GroupLayout.DEFAULT_SIZE, 33, Short.MAX_VALUE)  
    .addComponent(tf_BankACNo))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
```

```
    .addComponent(tf_ccNumber)  
    .addComponent(jLabel8,  
    javax.swing.GroupLayout.DEFAULT_SIZE, 31, Short.MAX_VALUE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
```

```
    .addComponent(tf_pinNumber)  
    .addComponent(jLabel9,  
    javax.swing.GroupLayout.DEFAULT_SIZE, 32, Short.MAX_VALUE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
```

```
    .addComponent(cb_SecOp,  
        javax.swing.GroupLayout.DEFAULT_SIZE, 34, Short.MAX_VALUE)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
```

```
    .addComponent(cb_A3)  
    .addComponent(jLabel10,  
        javax.swing.GroupLayout.DEFAULT_SIZE, 34, Short.MAX_VALUE)  
    .addComponent(cb_A1)  
    .addComponent(cb_Op)  
    .addComponent(cb_A2)))  
.addGap(30, 30, 30)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```
    .addComponent(register,  
        javax.swing.GroupLayout.PREFERRED_SIZE, 38,  
        javax.swing.GroupLayout.PREFERRED_SIZE)  
    .addComponent(exit,  
        javax.swing.GroupLayout.PREFERRED_SIZE, 38,  
        javax.swing.GroupLayout.PREFERRED_SIZE))
```

```

        .addGap(34, 34, 34))    );

    jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL, new
java.awt.Component[] {jLabel1, jLabel2, jLabel3, jLabel4, jLabel5,
jLabel6});

    jPanel1Layout.linkSize(javax.swing.SwingConstants.VERTICAL, new
java.awt.Component[] { addr, mailId, phno, pwd, rePwd, userName });

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(67, Short.MAX_VALUE)))
        .addGroup(layout.createSequentialGroup()
            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(0, 0, Short.MAX_VALUE))
    );

```

```

pack();

} // </editor-fold> // GEN-END: initComponents

private void registerActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_registerActionPerformed
    // TODO add your handling code here:
    if (validateFields()) {
        UserDetails _usrDtls = new UserDetails();
        _usrDtls.setMobileNumber(phno.getText());
        _usrDtls.setUserName(userName.getText().trim());
        _usrDtls.setPassword(pwd.getText());
        _usrDtls.setEmailId(mailId.getText());
        _usrDtls.setAddress(addr.getText());
        _usrDtls.setBankAcNo(tf_BankACNo.getText());
        _usrDtls.setBankName(cb_BankName.getSelectedItem().toString());
        _usrDtls.setCcPwd(tf_pinNumber.getText());
        _usrDtls.setFormula(cb_A1.getSelectedItem().toString() + "#" +
cb_Op.getSelectedItem().toString()
        + "#" + cb_A2.getSelectedItem().toString() + "#" +
cb_SecOp.getSelectedItem().toString() + "#" +
cb_A3.getSelectedItem().toString());
        _usrDtls.setCreditCardNo(tf_ccNumber.getText());
        _usrDtls.setccLimit(constants.Constants.CC_LIMIT);
        _usrDtls.setParentAc(Long.parseLong(tf_parentAC.getText()));
        UserDetailsDAOImpl _userImpl = new UserDetailsDAOImpl();
        _userImpl.saveUsrDtls(_usrDtls);
        UserDetails _usrDtlsRec =
        _userImpl.validateUsrDtls(userName.getText().trim(), pwd.getText());
    }
}

```

```

        CCDtls _ccDtls = new CCDtls();
        _ccDtls.setCcLimit(constants.Constants.CC_LIMIT);
        _ccDtls.setUserId(_usrDtlsRec.getId());
        _ccDtls.setCcNo(_usrDtlsRec.getCreditCardNo());
        CCDtlsDAOImpl _ccImpl = new CCDtlsDAOImpl();
        _ccImpl.saveCCDtls(_ccDtls)

        JOptionPane.showMessageDialog(this, "Details saved
successfully");}

} //GEN-LAST:event_registerActionPerformed

public boolean validateFields() {
    boolean valid = false;
    if (userName.getText().isEmpty() || pwd.getText().isEmpty()
        || rePwd.getText().isEmpty() || mailId.getText().isEmpty()
        || addr.getText().isEmpty() || phno.getText().isEmpty()) {
        valid = false;
    } else {
        valid = true;
    }
    return valid;
}

public void clearFields() {
    userName.setText("");
    addr.setText("");
    pwd.setText("");
    rePwd.setText("");
    mailId.setText("");
    phno.setText("");
}

```



```

java.util.logging.Logger.getLogger(RelativeAC.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(RelativeAC.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex)
java.util.logging.Logger.getLogger(RelativeAC.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    }
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new RelativeAC(_usrDtls).setVisible(true);
        }
    });
// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JTextField addr;
private javax.swing.JComboBox cb_A1;
private javax.swing.JComboBox cb_A2;
private javax.swing.JComboBox cb_A3;
private javax.swing.JComboBox cb_BankName;
private javax.swing.JComboBox cb_Op;
private javax.swing.JComboBox cb_SecOp;
private javax.swing.JButton exit;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel12;

```

```
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel1;
private javax.swing.JTextField mailId;
private javax.swing.JTextField phno;
private javax.swing.JPasswordField pwd;
private javax.swing.JPasswordField rePwd;
private javax.swing.JButton register;
private javax.swing.JTextField tf_BankACNo;
private javax.swing.JTextField tf_ccNumber;
private javax.swing.JTextField tf_parentAC;
private javax.swing.JTextField tf_pinNumber;
private javax.swing.JTextField userName;
// End of variables declaration//GEN-END:variables
}
```

APPENDIX-2

SNAP SHOTS

Running command prompt with hadoop commands to access the transaction application using the credit card.

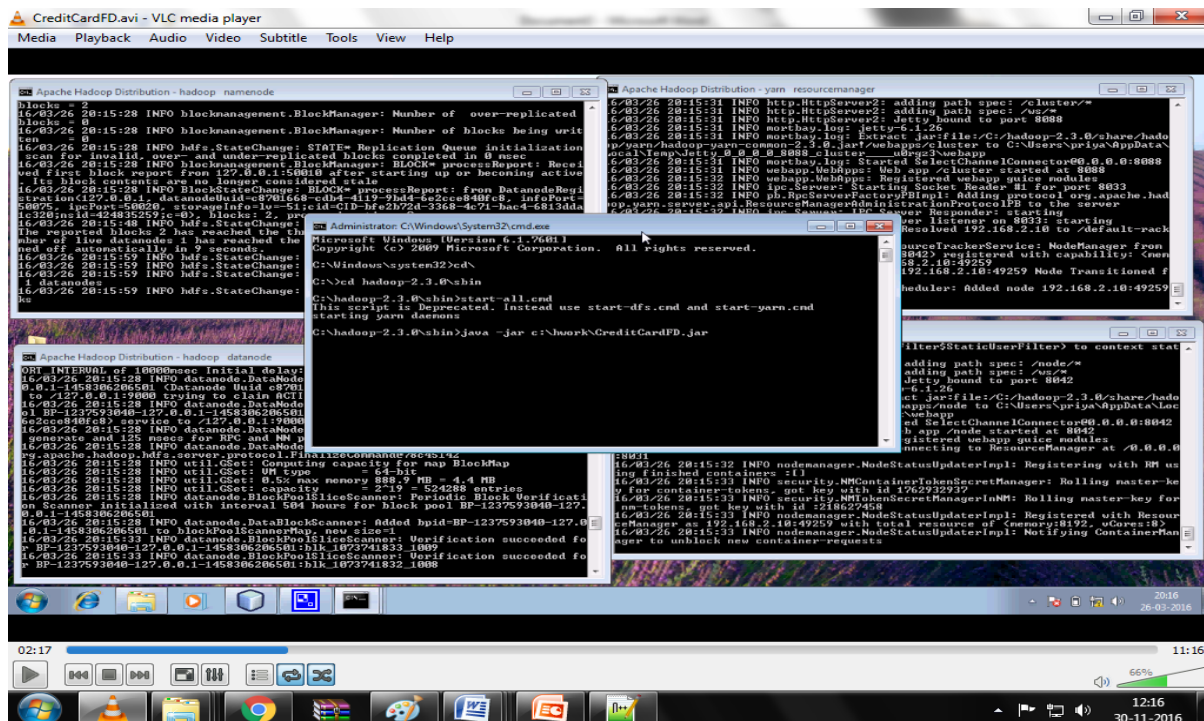


Fig A.1 Opening the application using hadoop commands

Front end of the application appears and provides access to the transaction. The username and password is entered if the user is already registered and the required amount is withdrawn.

The image shows a 'User Login' window with a light gray background. It contains two input fields: 'User Name' and 'Password'. Below these fields are four buttons: 'Sign-In', 'Sign-Up', 'Relative A/C', and 'Exit'. The window has a standard title bar with a minimize, maximize, and close button.

Fig A-2 Frontend User registration Form

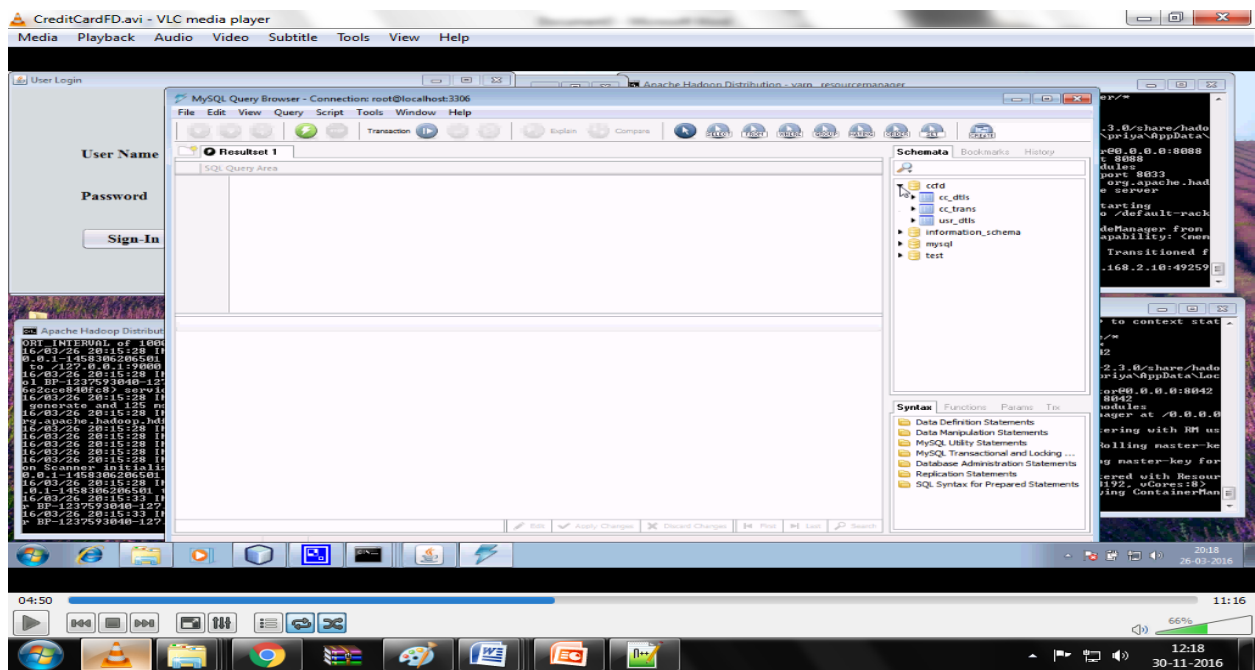
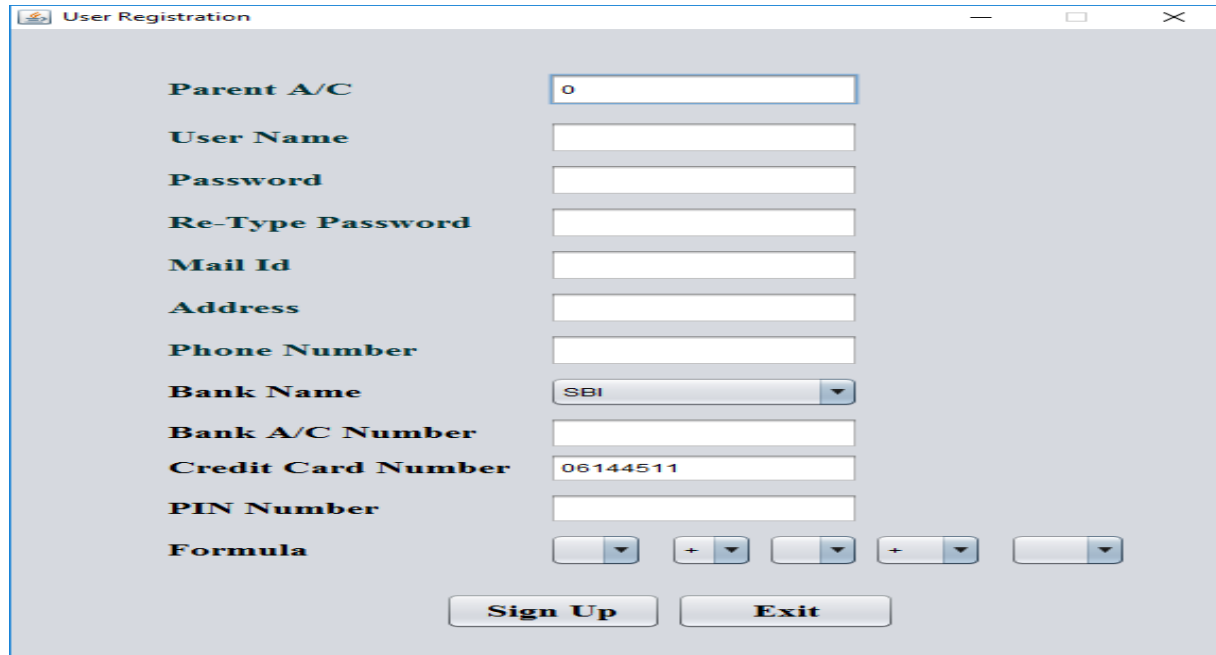


Fig A-3 SQL dataBase to save transaction Details

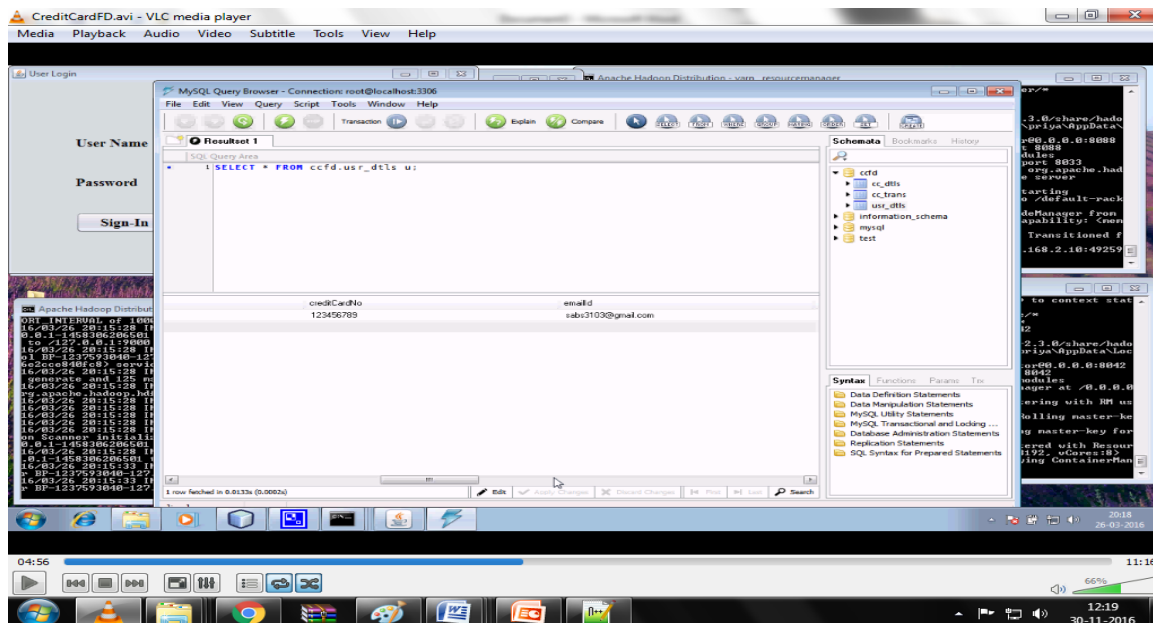
The relative account is registered to the corresponding account or the card holder. The formula for the particular relative account is setup and the sign up is selected in order to register to the account.



A screenshot of a 'User Registration' window. It contains several input fields for user details: Parent A/C (with value 0), User Name, Password, Re-Type Password, Mail Id, Address, Phone Number, Bank Name (dropdown menu showing SBI), Bank A/C Number, Credit Card Number (with value 06144511), PIN Number, and Formula (with a dropdown menu). At the bottom, there are 'Sign Up' and 'Exit' buttons.

Fig A-4 User Detail Registration to the card

The transaction details stored in the MYSQL and can have access to all the transactions that are made to the various accounts attached to the card



REFERENCES

- [1]Tsan-Ming Choi, Member, IEEE, Hing Kai Chan, Senior Member, IEEE, and Xiaohang Yue” Recent Development in Big Data Analytics for Business Operations and Risk Management” May2016.
- [2]C.-H. Chiu and T.-M. Choi, “Supply chain risk analysis with mean variance models: A technical review,” *June 2016*
- [3] Z. Hong, C. K. M. Lee, and X. Nie, “Proactive and reactive purchasing planning under dependent demand, price, and yield risks,” *OR Spectr.*, vol. 36, no. 4, pp. 1055–1076, 2014.
- [4] N. Manwani and P. S. Sastry, “Noise tolerance under risk minimization,”*IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 1146–1151, Jun. 2013.
- [5] H. K. Chan and F. T. S. Chan, “Early order completion contract approach to minimize the impact of demand uncertainty on supply chains,” *IEEE Trans. Ind. Informat.*, vol. 2, no. 1, pp. 48–58, Feb. 2013.
- [6] G. M. Gaukler, “Item-level RFID in a retail supply chain with stockout-based substitution,” *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 362–370, May 2011.

- [7] K. Govindan, A. Jafarian, M. E. Azbari, and T.-M. Choi, “Optimal bi-objective redundancy allocation for systems reliability and risk management,” *IEEE Trans. Cybern.*, June 2011.
- [8] B. Shen, T.-M. Choi, Y. Wang, and C. K. Y. Lo, “The coordination of fashion supply chains with a risk-averse supplier under the markdown money policy,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 2, pp. 266–276, Mar. 2011.
- [9] H. M. Markowitz, *Portfolio Selection: Efficient Diversification of Investment*. New York, NY, USA: Wiley, 1959.
- [10] D. D. Wu and D. Olson, “Enterprise risk management: A DEA VaR approach in vendor selection,” *Int. J. Prod. Res.*, vol. 48, no. 16, pp. 4919–4932, 2010.
- [11] D. D. Wu and D. Olson, “Enterprise risk management: Coping with model risk in a large bank,” *J. Oper. Res. Soc.*, vol. 61, no. 2, pp. 179–190, 2010.
- [12] D. L. Olson and D. D. Wu, “Risk management models for supply chain: A scenario analysis of outsourcing to China,” *Supply Chain Manag. Int. J.*, vol. 16, no. 6, pp. 401–408, 2011.