`

CREDIT CARD FRAUD DETECTION USING ADABOOST AND MAJORITY VOTING

A MAJOR PROJECT REPORT

Submitted by

THIRUPATHI SAI SHASHANK GOUD

(19841A05F9)

KOTHAPALLI KOUSHIK PAVAN

(19841A05H3)

PENDLIMADGULA VISHNU VARDHAN GOUD

(19841A05E7)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

AURORA'S TECHNOLOGICAL AND RESEARCH INSTITUTE

Approved by AICTE and Affiliated to JNTUH (Accredited By NAAC with 'A' Grade)

Parvathapur, Uppal, Medipally(M), Medchal(D), Hyderabad – 500098.

JUNE 2023

`

AURORA'S TECHNOLOGICAL AND RESEARCH INSTITUTE

Approved by AICTE and Affiliated to JNTUH (Accredited By NAAC with 'A' Grade)

Parvathapur, Uppal, Medipally(M), Medchal(D), Hyderabad – 500098.

.

DECLARATION

We hereby declare that the work described in this project, entitled '**CREDIT CARD FRAUD DETECTION USING ADABOOST AND MAJORITY VOTING'** which is being submitted by us in partial fulfilment for the award of Bachelor of Technology in Computer Science and Engineering to **AURORA'S TECHNOLOGICAL AND RESEARCH INSTITUTE** is the result of investigation carried by us under the guidance of **Mr. Rohit Kumar, Assistant Professor, CSE.**

The work is original and has not been submitted for any degree of this or any other university.

Place: Hyderabad

Date:

THIRUPATHI SAI SHASHANK (19841A05F9)

KOTHAPALLI KOUSHIK PAVAN (19841A05H3)

PENDLIMADGULA VISHNU VARDHAN (19841A05E7)

`

CERTIFICATE

Certified that this project report **"CREDIT CARD FRAUD DETECTION USING ADABOOST AND MAJORITY VOTING"** is the bonafide work of "**T. SAI SHASHANK (19841A05F9), K. KOUSHIK PAVAN (19841A05H3), P. VISHNU VARDHNA (19841A05E7)"** who carried out the project work under my supervision.

GUIDE

Mr. Rohit Kumar

Assistant Professor

Dept. of CSE

PROJECT COORDINATOR

Dr. M. Saravanan

Associate Professor

Dept. of CSE

HEAD OF THE DEPARTMENT

Ms. A. Durga Pavani

Dept. of CSE

DIRECTOR

Mr. Srikanth Jatla

EXTERNAL EXAMINER

`

ACKNOWLEDGMENT

This work has been done during the project period and it was a very good opportunity to put theoretical knowledge into planned exercise with an aim to solve a real time problem and also to develop confidence to face various practical situations.

We would also like to express our gratitude to **Mr. Srikanth Jatla, Director**, Aurora's Technological and Research Institute for providing us with a congenial atmosphere and encouragement.

We express our sincere thanks to Head of the Department **Ms. A. Durga Pavani,** for giving us the support and her kind attention and valuable guidance to us throughout this course.

We express our sincere thanks to Project Coordinator **Dr. M. Saravanan,** for helping us to complete our project work by giving valuable suggestions.

We convey thanks to our project guide **Mr. Rohit Kumar**, Department of Computer Science and Engineering, for providing encouragement, constant support and guidance which was of great help to complete this project successfully

`

ABSTRACT

Credit card fraud is a serious problem in financial services. Billions of dollars are lost due to credit card fraud every year. There is a lack of research studies on analysing real-world credit card data owing to confidentiality issues. In this paper, machine learning algorithms are used to detect credit card fraud. Standard models are firstly used. Then, hybrid methods which use AdaBoost and majority voting methods are applied. To evaluate the model efficacy, a publicly available credit card data set is used. Then, a real-world credit card data set from a financial institution is analysed. In addition, noise is added to the data samples to further assess the robustness of the algorithms. The experimental results positively indicate that the majority voting method achieves good accuracy rates in detecting fraud cases in credit cards.

`

TABLE OF CONETNTS

`

LIST OF FIGURES

`

`

LIST OF ABBREVATIONS

API	-	Application Program Interface
HTML	-	Hyper Text Markup Language
IDE	-	Integrated Development Environment
JDBC	-	Java Database Connectivity
JIT	-	Just In Time
JSP	-	Java Server Pages
JVM	-	Java Virtual Machine
LAN	-	Local Area Network
ODBC	-	Open Database Connectivity
OSN	-	Online Server Name
SGML	-	Standard Generalised Markup Language
SM	-	Social Media
SQL	-	Structured Query Language
UML	-	Unified Modelling Language

`

CHAPTER 1

INTRODUCTION

Fraud is a wrongful or criminal deception aimed to bring financial or personal gain. In avoiding loss from fraud, two mechanisms can be used: fraud prevention and fraud detection. Fraud prevention is a proactive method, where it stops fraud from happening in the first place. On the other hand, fraud detection is needed when a fraudulent transaction is attempted by a fraudster. Credit card fraud is concerned with the illegal use of credit card information for purchases. Credit card transactions can be accomplished either physically or digitally. In physical transactions, the credit card is involved during the transactions. In digital transactions, this can happen over the telephone or the internet. Cardholders typically provide the card number, expiry date, and card verification number through telephone or website. With the rise of e-commerce in the past decade, the use of credit cards has increased dramatically. The number of credit card transactions in 2011 in Malaysia were at about 320 million, and increased in 2015 to about 360 million. Along with the rise of credit card usage, the number of fraud cases have been constantly increased. While numerous authorization techniques have been in place, credit card fraud cases have not hindered effectively. Fraudsters favour the internet as their identity and location are hidden. The rise in credit card fraud has a big impact on the financial industry.

The global credit card fraud in 2015 reached to a staggering USD $21.84 billion. Loss from credit card fraud affects the merchants, where they bear all costs, including card issuer fees, charges, and administrative charges. Since the merchants need to bear the loss, some goods are priced higher, or discounts and incentives are reduced. Therefore, it is imperative to reduce the loss, and an effective fraud detection system to reduce or eliminate fraud cases is important. There have been various studies on credit card fraud detection. Machine learning and related methods are most commonly used, which include artificial neural networks, rule-induction techniques, decision trees, logistic regression, and support vector machines. These methods are used either standalone or by combining several methods together to form hybrid models. IEEE In this paper, a total of twelve machine learning algorithms are used for detecting credit card fraud. The algorithms range from standard neural networks to deep learning

`

models. They are evaluated using both benchmark and real-world credit card data sets. In addition, the AdaBoost and majority voting methods are applied for forming hybrid models. To further evaluate the robustness and reliability of the models, noise is added to the real-world data set. The key contribution of this paper is the evaluation of a variety of machine learning models with a real-world credit card data set for fraud detection. While other researchers have used various methods on publicly available data sets, the data set used in this paper are extracted from actual credit card transaction information over three months.

1.1 What Is Credit Card Fraud Detection?

Credit card fraud detection is the process of identifying purchase attempts that are fraudulent and rejecting them rather than processing the order. There are a variety of tools and techniques available for detecting fraud, with most merchants employing a combination of several of them.

Payment cards are easy to use because you only need to transmit a few simple numbers to the bank in order to identify your account and authorize the transaction. This simplicity makes them vulnerable as well. It’s very hard to practice rigorous data security on a few simple numbers that must be shared with the parties you’re transacting with.

In the card-present space, EMV chips have gone a long way towards reducing in-person fraud. These chips tokenize account numbers, making them less vulnerable to data theft, and cannot be counterfeited as easily as magnetic stripe cards.

In the card-not-present transaction environment of e-commerce, there is no technological solution equivalent to the EMV chip in terms of widespread use and effectiveness. There are some possible solutions on the horizon, most notably Click to Pay, but unlike EMV chips, these require action on the customer's part to sign up for the service. That means there's a significant barrier to adoption that can't be overcome by some equivalent of the EMV liability shift that ensured widespread acceptance of EMV chips.

.

`

Credit card fraud costs the global economy more than $24 billion per year, and the numbers keep going up. Smaller merchants suffer the most from the impact of fraud, and that's why it's so important to have tools and practices in place to detect fraud in its early stages.

1.2 AdaBoost Algorithm in Machine Learning

Adaptive Boosting, or most commonly known AdaBoost, is a Boosting algorithm. This algorithm uses the method to correct its predecessor. It pays more attention to under fitted training instances by the previous model. Thus, at every new predictor the focus is more on the complicated cases more than the others.

It fits a sequence of weak learners on different weighted training data. It starts by predicting the original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy.

Mostly, AdaBoost uses decision stamps. But we can use any machine learning algorithm as base learner if it accepts weight on training data set. We can use AdaBoost algorithm in machine learning for both classification and regression problems.

Let us consider the example of the image mentioned above. In order to build an AdaBoost classifier, consider that as a first base classifier a Decision Tree algorithm is trained to make predictions on our training data. Applying the following methodology of AdaBoost, the weight of the misclassified training instances is increased. Then the second classifier is trained and the updated weights are acknowledged. It repeats the procedure over and over again.

At the end of every model prediction, we end up boosting the weights of the misclassified instances so that the next model does a better job on them, and so on.

`

This sequential learning technique might sound similar to Gradient Descent, except that instead of tweaking a single predictor's parameter to minimize the cost function, AdaBoost adds predictors to the ensemble, gradually making it better.

One disadvantage of this algorithm is that the model cannot be parallelized since each predictor can only be trained after the previous one has been trained and evaluated.

Below are the steps for performing the AdaBoost algorithm:

1. Initially, all observations are given equal weights.
2. A model is built on a subset of data.
3. Using this model, predictions are made on the whole dataset.
4. Errors are calculated by comparing the predictions and actual values.
5. While creating the next model, higher weights are given to the data points which were predicted incorrectly.
6. Weights can be determined using the error value. For instance,the higher the error the more is the weight assigned to the observation.
7. This process is repeated until the error function does not change, or the maximum limit of the number of estimators is reached.

1.3 Majority Voting

Voting methods are based on a democratic (weighted) process that combines the predictions provided by the classification models independently calibrated on a number of available analytical sources. The most simple and intuitive approach is based on majority voting rule, which assigns a sample on the basis of the most frequent class assignment ("loose" method), whereas the sample is not classified in case of ties.

`

Protective approaches are derived by applying specific limits, which means that a sample is assigned only if the frequency of assignments to a class is higher than a selected threshold. For example, “intermediate” and “strict” majority voting criteria classify a sample if the agreement of predictions is higher than or equal to 75% and 100%, respectively. The strict majority voting represents the case of full prediction agreement of all the considered models.

Imagine to have five analytical data blocks, each one associated with a specific (and independent) classification model, which can assign samples to class *A* or class *B*. Note that samples can also be nonassigned (na); for example, the analytical measurements associated with the model *P*5 were not available for the sample *S*4 or the sample *S*4 was an outlier in the analytical chemical space of *P*5 and thus no prediction was provided. The agreement of predictions is evaluated by the class frequency, that is, the ratio of the number of predictions in a specific class over the number of available predictions. Finally, majority voting rules assign a sample to the class associated with the highest frequency of predictions, unless the specific restrictions on the percentage of class agreement are not fulfilled.

`

CHAPTER 2

LITERATURE SURVEY.

2.1 Title: Machine Learning For Credit Card Fraud Detection System

Authors: Lakshmi S V S S1, Selvani Deepthi Kavila2

The rapid growth in E-Commerce industry has lead to an exponential increase in the use of credit cards for online purchases and consequently they has been surge in the fraud related to it. In recent years, for banks has become very difficult for detecting the fraud in credit card system. Machine learning plays a vital role for detecting the credit card fraud in the transactions. For predicting these transactions banks make use of various machine learning methodologies, past data has been collected and new features are been used for enhancing the predictive power. The performance of fraud detecting in credit card transactions is greatly affected by the sampling approach on data-set, selection of variables and detection techniques used.

This paper investigates the performance of logistic regression, decision tree and random forest for credit card fraud detection. Dataset of credit card transactions is collected from Kaggle and it contains a total of 2,84,808 credit card transactions of a European bank data set. It considers fraud transactions as the “positive class” and genuine ones as the “negative class”. The data set is highly imbalanced, it has about 0.172% of fraud transactions and the rest are genuine transactions. The author has been done oversampling to balance the data set, which resulted in 60% of fraud transactions and 40% genuine ones. The three techniques are applied for the dataset and work is implemented in R language. The performance of the techniques is evaluated for different variables based on sensitivity, specificity, accuracy and error rate. The result shows of accuracy for logistic regression, Decision tree and random forest classifier are 90.0, 94.3, 95.5 respectively. The comparative results show that the Random Forest performs better than the logistic regression and decision tree techniques.

`

2.2 Title: A Cost-Sensitive Decision Tree Approach For Fraud Detection

Authors: Yusuf Sahin a, Serol Bulkan b, Ekrem Duman c

With the developments in the information technology, fraud is spreading all over the world, resulting in huge financial losses. Though fraud prevention mechanisms such as CHIP&PIN are developed for credit card systems, these mechanisms do not prevent the most common fraud types such as fraudulent credit card usages over virtual POS (Point Of Sale) terminals or mail orders so called online credit card fraud. As a result, fraud detection becomes the essential tool and probably the best way to stop such fraud types. In this study, a new cost-sensitive decision tree approach which minimizes the sum of misclassification costs while selecting the splitting attribute at each non-terminal node is developed and the performance of this approach is compared with the well-known traditional classification models on a real-world credit card data set. In this approach, misclassification costs are taken as varying. The results show that this cost-sensitive decision tree algorithm outperforms the existing well-known methods on the given problem set with respect to the well-known performance metrics such as accuracy and true positive rate, but also a newly defined cost-sensitive metric specific to credit card fraud detection domain. Accordingly, financial losses due to fraudulent transactions can be decreased more by the implementation of this approach in fraud detection systems

2.3 Title: Credit Card Fraud Detection Using Hidden Markov Model

Authors: Abhinav Srivastava; Amlan Kundu; Shamik Sural; Arun Majumdar

The Internet has taken its place beside the telephone and the television as an important part of people's lives. Consumers rely on the Internet to shop, bank and invest online. Most online shoppers use credit cards to pay for their purchases. As credit card becomes the most popular mode of payment, cases of fraud associated with it are also increasing. In this paper, we model the sequence of operations in credit card transaction processing using a Hidden Markov Model (HMM) and show how it can be used for the detection of frauds. An HMM is trained with normal behaviour of cardholder. If an incoming credit card transaction is not accepted by the HMM with sufficiently high probability, it is considered to be fraudulent. We present detailed experimental results to show the effectiveness of our approach.

`

2.4 Title: Real-Time Credit Card Fraud Detection Using Computational Intelligence

Authors: Jon T.S. Quah, M. Sri Ganesh

Online banking and e-commerce have been experiencing rapid growth over the past few years and show tremendous promise of growth even in the future. This has made it easier for fraudsters to indulge in new and abstruse ways of committing credit card fraud over the Internet. This paper focuses on real-time fraud detection and presents a new and innovative approach in understanding spending patterns to decipher potential fraud cases. It makes use of self-organization map to decipher, filter and analyse customer behaviour for detection of fraud.

2.5 Title: Credit Card Fraud Detection Using Data Mining Techniques

Authors: Lawrence Borah, Saleena B, Prakash B

Credit card fraud has become the most significantly popular issues in the credit card industry. The principle thought process is to distinguish the various kinds of credit card misrepresentation and to audit elective procedures that have been utilized in fraud detection. Depending upon various types of credit card frauds faced by financial institutions like banks, or credit card industries, many different measures can be implemented to reduce the frauds. The objective of the usage of these strategies and techniques is to minimize credit card fraud. There are certain open problems with the existing techniques that result in some genuine credit card clients as misclassified fraudulent. This paper focuses on incorporating the best classification algorithm from a set of four different algorithms, which is likely to suggest the degree of fraudulent activities in the field of financial domain.

`

2.6 Title: A Novel Model For Credit Card Fraud Detection Using Artificial Immune Systems

Authors: Neda Soltani Halvaiee, Mohammad Kazem Akbari

The amount of online transactions is growing these days to a large number. A big portion of these transactions contains credit card transactions. The growth of online fraud, on the other hand, is notable, which is generally a result of ease of access to edge technology for everyone. There has been research done on many models and methods for credit card fraud prevention and detection. Artificial Immune Systems is one of them. However, organizations need accuracy along with speed in the fraud detection systems, which is not completely gained yet. In this paper we address credit card fraud detection using Artificial Immune Systems (AIS), and introduce a new model called AIS-based Fraud Detection Model (AFDM). We will use an immune system inspired algorithm (AIRS) and improve it for fraud detection. We increase the accuracy up to 25%, reduce the cost up to 85%, and decrease system response time up to 40% compared to the base algorithm.

2.7 Title: Credit Card Fraud Detection: A Fusion Approach Using Dempster–Shafer Theory and Bayesian Learning

Authors: Suvasini Panigrahi, Amlan Kundu, Shamik Sural

We propose a novel approach for credit card fraud detection, which combines evidences from current as well as past behaviour. The fraud detection system (FDS) consists of four components, namely, rule-based filter, Dempster–Shafer adder, transaction history database and Bayesian learner. In the rule-based component, we determine the suspicion level of each incoming transaction based on the extent of its deviation from good pattern. Dempster–Shafer's theory is used to combine multiple such evidences and an initial belief is computed. The transaction is classified as normal, abnormal or suspicious depending on this initial belief. Once a transaction is found to be suspicious, belief is further strengthened or weakened according to its similarity with fraudulent or genuine transaction history using Bayesian learning. Extensive simulation with stochastic models shows that fusion of different evidences has a very high positive impact on the performance of a credit card fraud detection system as compared to other methods.

`

2.8 Title: Detecting credit card fraud by Modified Fisher Discriminant Analysis

Authors: Nader Mahmoodi, Ekrem Duman

In parallel to the increase in the number of credit card transactions, the financial losses due to fraud have also increased. Thus, the popularity of credit card fraud detection has been increased both for academicians and banks. Many supervised learning methods were introduced in credit card fraud literature some of which bears quite complex algorithms. As compared to complex algorithms which somehow over-fit the dataset they are built on, one can expect simpler algorithms may show a more robust performance on a range of datasets. Although, linear discriminant functions are less complex classifiers and can work on high-dimensional problems like credit card fraud detection, they did not receive considerable attention so far.

This study investigates a linear discriminant, called Fisher Discriminant Function for the first time in credit card fraud detection problem. On the other hand, in this and some other domains, cost of false negatives is very higher than false positives and is different for each transaction. Thus, it is necessary to develop classification methods which are biased toward the most important instances. To cope for this, a Modified Fisher Discriminant Function is proposed in this study which makes the traditional function more sensitive to the important instances. This way, the profit that can be obtained from a fraud/legitimate classifier is maximized. Experimental results confirm that Modified Fisher Discriminant could eventuate more profit.

`

2.9 Title: Detecting credit card fraud by genetic algorithm and scatter search

Authors: Ekrem Duman, M. Hamdi Oz Celik

In this study we develop a method which improves a credit card fraud detection solution currently being used in a bank. With this solution each transaction is scored and based on these scores the transactions are classified as fraudulent or legitimate. In fraud detection solutions the typical objective is to minimize the wrongly classified number of transactions. However, in reality, wrong classification of each transaction do not have the same effect in that if a card is in the hand of fraudsters its whole available limit is used up. Thus, the misclassification cost should be taken as the available limit of the card. This is what we aim at minimizing in this study. As for the solution method, we suggest a novel combination of the two well known meta-heuristic approaches, namely the genetic algorithms and the scatter search. The method is applied to real data and very successful results are obtained compared to current practice.

2.10 Existing System

Three methods to detect fraud are presented. Firstly, clustering model is used to classify the legal and fraudulent transaction using data clusterization of regions of parameter value. Secondly, Gaussian mixture model is used to model the probability density of credit card user's past behaviour so that the probability of current behaviour can be calculated to detect any abnormalities from the past behaviour. Lastly, Bayesian networks are used to describe the statistics of a particular user and the statistics of different fraud scenarios. The main task is to explore different views of the same problem and see what can be learned from the application of each different technique.

2.10.1 Disadvantages

- There is no Majority Voting technique for credit card fraud detection.
- There is no Machine Learning Techniques in the existing system.

`

2.11 Proposed System

machine learning algorithms are used for detecting credit card fraud. The algorithms range from standard neural networks to deep learning models. They are evaluated using both benchmark and real-world credit card data sets. In addition, the AdaBoost and majority voting methods are applied for forming hybrid models. To further evaluate the robustness and reliability of the models, noise is added to the real-world data set. The key contribution of this paper is the evaluation of a variety of machine learning models with a real-world credit card data set for fraud detection.

2.11.1 Advantages

- The system is very fast due to AdaBoost Technique.
- Effective Majority Voting techniques

`

CHAPTER 3

SOFTWARE REQUIREMENT ANALYSIS

3.1 Technologies Used

Front end or User Interface Design

The entire user interface is planned to be developed in browser specific environment with a touch of Intranet-Based Architecture for achieving the Distributed Concept.

The browser specific components are designed by using the HTML standards, and the dynamism of the designed by concentrating on the constructs of the Java Server Pages.

Communication or Database Connectivity Tier5

The Communication architecture is designed by concentrating on the Standards of Servlets and Enterprise Java Beans. The database connectivity is established by using the Java Data Base Connectivity.

The standards of three-tire architecture are given major concentration to keep the standards of higher cohesion and limited coupling for effectiveness of the operations.

3.2 About Java

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

`

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.

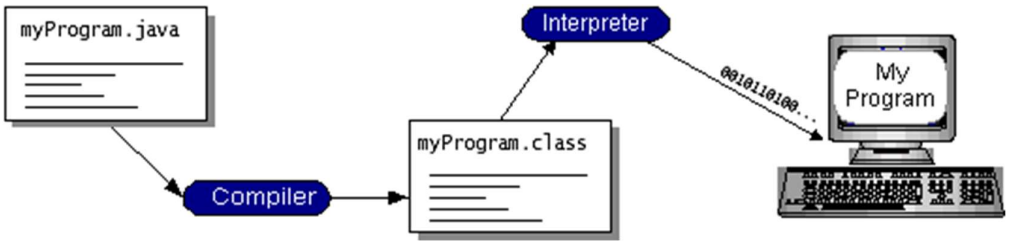


Fig No. 3.1 Process of java program

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.

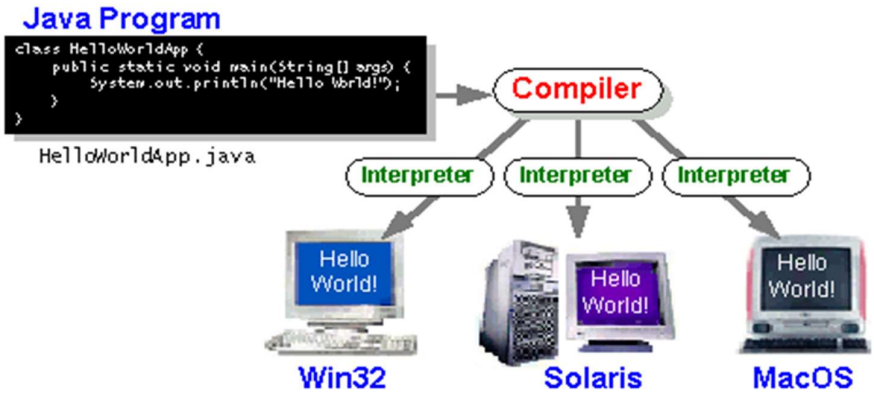


Fig No. 3.2 Process of Java Virtual Machine

`

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

- Java is a programmer’s language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

Finally, Java is to Internet programming where C was to system programming.

Importance of Java to the Internet

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

Java can be used to create two types of programs:

Applications

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java’s ability to create Applets makes it important.

Applets

An Applet is an application designed to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

`

3.3 Features of Java

Security

Every time you that you download a “normal” program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scanned them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a “firewall” between a network application and your computer.

When we use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

Portability

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed. As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java’s solution to these two problems is both elegant and efficient.

The Byte code

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

`

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

Java Virtual Machine (JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

Overall Description

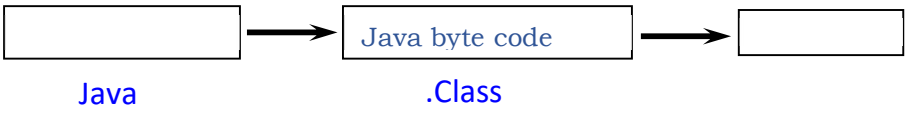


Fig No. 3.3 Java Virtual Machine

Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a. class file, which contains the byte code. The. Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

`

Java Architecture

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

Compilation of code

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

Compiling and interpreting Java Source Code

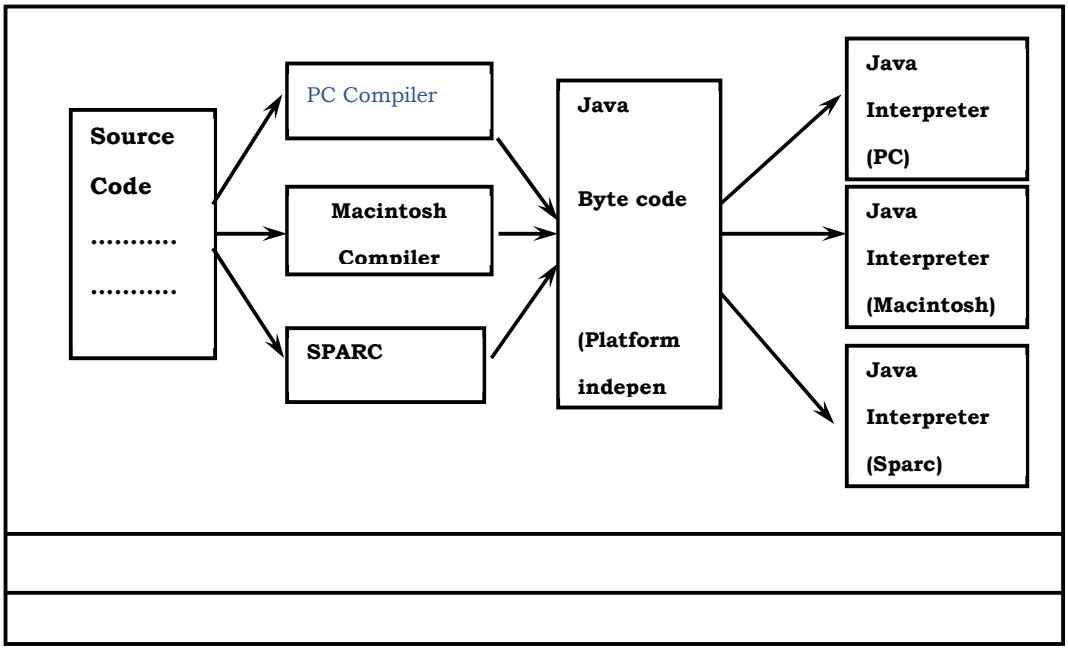


Fig No. 3.4 Compiling And Interpreting Java Source Code

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

`

Simple

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

Object-Oriented

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

Robust

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

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

3.4 Hyper Text Markup Language

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can

`

navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

Basic HTML Tags:

<!> Specifies comments

<A> </A> Creates hypertext links

<B> </B> Formats text as bold

<BIG> </BIG> Formats text in large font.

<BODY> </BODY> Contains all tags and text in the HTML document

<CENTER> </CENTER> Creates text

<DD> </DD> Definition of a term

<DL> </DL> Creates definition list

<FONT> </FONT> Formats text with a particular font

<FORM> </FORM> Encloses a fill-out form

<FRAME> </FRAME> Defines a particular frame in a set of frames

<H#> </H#> Creates headings of different levels

<HEAD> </HEAD> Contains tags that specify information about a document

<HR> </HR> Creates a horizontal rule

<HTML> </HTML> Contains all other HTML tags

<META> </META> Provides meta-information about a document

<SCRIPT> </SCRIPT> Contains client-side or server-side script

<TABLE> </TABLE> Creates a table

`

<TD> </TD> Indicates table data in a table

<TR> </TR> Designates a table row

<TH> </TH> Creates a heading in a table

Advantages

- A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
- HTML is platform independent.
- HTML tags are not case-sensitive.

3.5 Java Database Connectivity

What Is JDBC? JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually any relational database. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of Java and JDBC lets a programmer write it once and run it anywhere.

What Does JDBC Do? Simply put, JDBC makes it possible to do three things:

- Establish a connection with a database
- Send SQL statements
- Process the results.

JDBC versus ODBC and other APIs

At this point, Microsoft's ODBC (Open Database Connectivity) API is that probably the most widely used programming interface for accessing relational databases. It offers the ability to connect to almost all databases on almost all platforms. So why not just use ODBC from Java? The answer is that you can use ODBC from Java, but this is best done with the help of JDBC in the form of the JDBC-ODBC

`

Bridge, which we will cover shortly. The question now becomes "Why do you need JDBC?" There are several answers to this question:

1. ODBC is not appropriate for direct use from Java because it uses a C interface. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications.
2. A literal translation of the ODBC C API into a Java API would not be desirable. For example, Java has no pointers, and ODBC makes copious use of them, including the notoriously error-prone generic pointer "void *". You can think of JDBC as ODBC translated into an object-oriented interface that is natural for Java programmers.
3. ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the other hand, was designed to keep simple things simple while allowing more advanced capabilities where required.
4. A Java API like JDBC is needed in order to enable a "pure Java" solution. When ODBC is used, the ODBC driver manager and drivers must be manually installed on every client machine. When the JDBC driver is written completely in Java, however, JDBC
5. code is automatically installable, portable, and secure on all Java platforms from network computers to mainframes.

Two-tier and Three-tier Models

In the two-tier model, a Java applet or application talks directly to the database. This requires a JDBC driver that can communicate with the particular database management system being accessed. A user's SQL statements are delivered to the database, and the results of those statements are sent back to the user. The database may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the database as the server. The network can be an Intranet, which, for example, connects employees within a corporation, or it can be the Internet.

`

In the three-tier model, commands are sent to a "middle tier" of services, which then send SQL statements to the database. The database processes the SQL statements and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it.

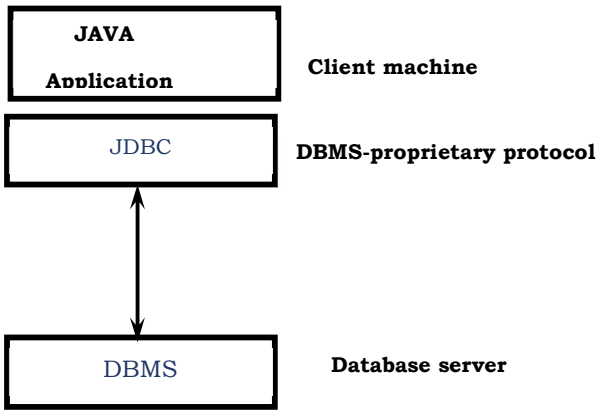


Fig No. 3.5 Two Tier and Three Tier Models

Possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that when there is a middle tier, the user can employ an easy-to-use higher-level API which is translated by the middle tier into the appropriate low-level calls. Finally, in many cases the three-tier architecture can provide performance advantages.

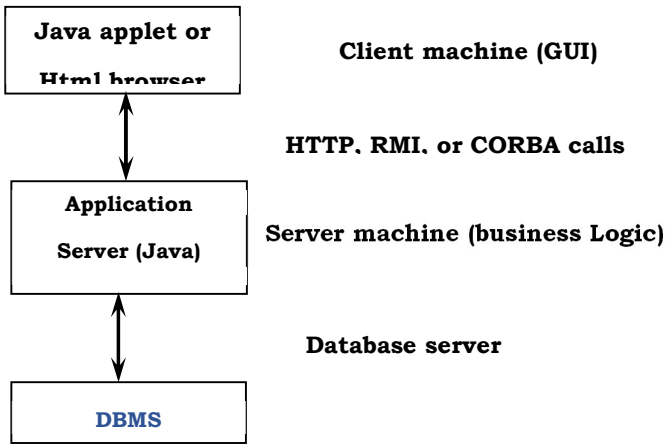


Fig No. 3.6 Process of two tier models

`

Until now the middle tier has typically been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java byte code into efficient machine-specific code, it is becoming practical to implement the middle tier in Java. This is a big plus, making it possible to take advantage of Java's robustness, multithreading, and security features. JDBC is important to allow database access from a Java middle tier.

3.6 JDBC Driver Types

The JDBC drivers that we are aware of at this time fit into one of four categories:

- JDBC-ODBC bridge plus ODBC driver
- Native-API partly-Java driver
- JDBC-Net pure Java driver
- Native-protocol pure Java driver

JDBC-ODBC Bridge

If possible, use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC. It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge (that is, the Bridge native library, the ODBC driver manager library, the ODBC driver library, and the database client library).

What Is the JDBC- ODBC Bridge?

The JDBC-ODBC Bridge is a JDBC driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge implements JDBC for any database for which an ODBC driver is available. The Bridge is implemented as the sun.jdbc.odbc Java package and contains a native library used to access ODBC. The Bridge is a joint development of Intersolv and JavaSoft.

3.7 Java Server Pages (JSP)

Java server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards, and a mature re-usable

`

component model .The Java Server Pages architecture enables the separation of content generation from content presentation. This separation not eases maintenance headaches, it also allows web team members to focus on their areas of expertise. Now, web page designer can concentrate on layout, and web application designers on programming, with minimal concern about impacting each other's work.

Features of JSP

Portability

Java Server Pages files can be run on any web server or web-enabled application server that provides support for them. Dubbed the JSP engine, this support involves recognition, translation, and management of the Java Server Page lifecycle and its interaction components.

Components

It was mentioned earlier that the Java Server Pages architecture can include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components current supported include Java Beans, and Servlets.

Processing

A Java Server Pages file is essentially an HTML document with JSP scripting or tags. The Java Server Pages file has a JSP extension to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a Servlet on the server side. The Servlet that is generated outputs real content in straight HTML for responding to the client.

Access Models

A Java Server Pages file may be accessed in at least two different ways. A client's request comes directly into a Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform particular well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean as properties. The page uses such Beans to generate dynamic content and present it back to the client.

`

In both of the above cases, the page could also contain any valid Java code. Java Server Pages architecture encourages separation of content from presentation.

Steps in the execution of a JSP Application:

1. The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page.
2. This request is transferred to the JavaWebServer. At the server side JavaWebServer receives the request and if it is a request for a jsp file server gives this request to the JSP engine.
3. JSP engine is program which can understands the tags of the jsp and then it converts those tags into a Servlet program and it is stored at the server side. This Servlet is loaded in the memory and then it is executed and the result is given back to the JavaWebServer and then it is transferred back to the result is given back to the JavaWebServer and then it is transferred back to the client.

JDBC connectivity

The JDBC provides database-independent connectivity between the J2EE platform and a wide range of tabular data sources. JDBC technology allows an Application Component Provider to:

- Perform connection and authentication to a database server
- Manager transactions
- Move SQL statements to a database engine for preprocessing and execution
- Execute stored procedures
- Inspect and modify the results from Select statements.

3.8 Apache Tomcat 6.0 Web Server

Apache Tomcat is an open source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process. Web Servers like Apache Tomcat support only web components while an application server supports web components as well as business components (BEAs Weblogic, is one of the popular application server).

`

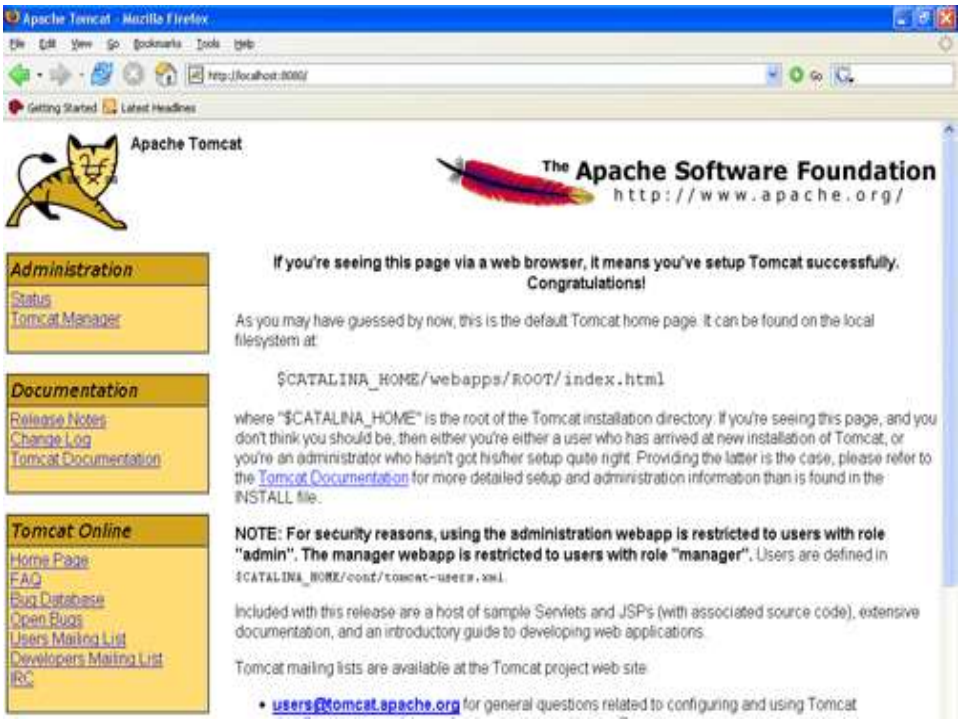


Fig No. 3.7 Apache Tomcat Server

3.9 Client Server

Over view:

With the varied topic in existence in the fields of computers, Client Server is one, which has generated more heat than light, and also more hype than reality. This technology has acquired a certain critical mass attention with its dedication conferences and magazines. Major computer vendors such as IBM and DEC, have declared that Client Servers is their main future market. A survey of DBMS magazine reveled that 76% of its readers were actively looking at the client server solution. The growth in the client server development tools from $200 million in 1992 to more than $1.2 billion in 1996.

Client server implementations are complex but the underlying concept is simple and powerful. A client is an application running with local resources but able to request the database and relate the services from separate remote server. The software mediating this client server interaction is often referred to as MIDDLEWARE.

`

The typical client either a PC or a Work Station connected through a network to a more powerful PC, Workstation, Midrange or Main Frames server usually capable of handling request from more than one client. However, with some configuration server may also act as client. A server may need to access other server in order to process the original client request.

The key client server idea is that client as user is essentially insulated from the physical location and formats of the data needs for their application. With the proper middleware, a client input from or report can transparently access and manipulate both local database on the client machine and remote databases on one or more servers. An added bonus is the client server opens the door to multi-vendor database access indulging heterogeneous table joins.

What is a Client Server

Two prominent systems in existence are client server and file server systems. It is essential to distinguish between client servers and file server systems. Both provide shared network access to data but the comparison dens there! The file server simply provides a remote disk drive that can be accessed by LAN applications on a file by file basis. The client server offers full relational database services such as SQL-Access, Record modifying, Insert, Delete with full relational integrity backup/ restore performance for high volume of transactions, etc. the client server middleware provides a flexible interface between client and server, who does what, when and to whom.

Why Client Server

Client server has evolved to solve a problem that has been around since the earliest days of computing: how best to distribute your computing, data generation and data storage resources in order to obtain efficient, cost effective departmental an enterprise-wide data processing. During mainframe era choices were quite limited. A central machine housed both the CPU and DATA (cards, tapes, drums and later disks).

`

Access to these resources was initially confined to batched runs that produced departmental reports at the appropriate intervals. A strong central information service department ruled the corporation. The role of the rest of the corporation limited to requesting new or more frequent reports and to provide hand written forms from which the central data banks were created and updated. The earliest client server solutions therefore could best be characterized as “SLAVE-MASTER”.

Time-sharing changed the picture. Remote terminal could view and even change the central data, subject to access permissions. And, as the central data banks evolved in to sophisticated relational database with non-programmer query languages, online users could formulate adhoc queries and produce local reports with out adding to the MIS applications software backlog. However remote access was through dumb terminals, and the client server remained subordinate to the Slave\Master

`

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1 Software Requirements

Operating System	:	Windows family
Technology	:	Java, JSP, HTML
IDE	:	NetBeans

4.2 Hardware Requirements

Processer	:	I3 or I5 or above
Ram	:	Min 4 GB
Hard Disk	:	Min 100 G

`

CHAPTER 5

SOFTWARE DESIGN

5.1 Architectural Design

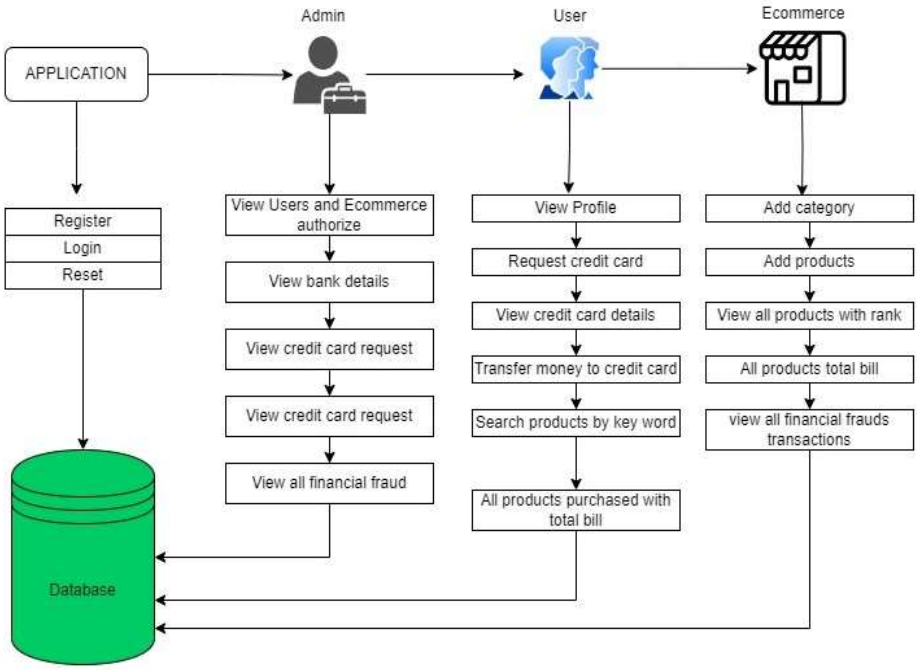


Fig No. 5.1 System Architecture

5.2 Modules

Bank Admin

In this module, the admin has to login by using valid user name and password. After login successful he can do some operations such as Bank Admin's Profile ,View Users and Authorize ,View Ecommerce Website Users and Authorize, Add Bank ,View Bank Details ,View Credit Card Requests, View all Products with rank ,View all Financial Frauds ,View all Financial Frauds with Random Forest Tree With wrong CVV ,View all Financial Frauds with Random Forest Tree with Expired Date Usage ,List Of all Users with Majority of Financial Fraud ,Show Product Rank In Chart ,Show Majority Voting With Wrong CVV Fraud in chart ,Show Majority Voting with Expiry date Usage in chart.

`

Viewing and Authorizing Users

In this module, the admin can view the list of users who all registered. In this, the admin can view the user's details such as, user name, email, address and admin authorize the users.

View Chart Results

Show Product Rank In Chart, Show Majority Voting With Wrong CVV Fraud in chart, Show Majority Voting with Expiry date Usage in chart.

Ecommerce Users

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like, Add Category, Add Products, view all Products with rank, and View all Purchased Products with total bill, View All Financial Frauds.

End Users

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like, View My Profile, Manage Bank Account, Request Credit Card, View Credit Card Details, Transfer Money to Your Credit Card Account, Search for Products by Keyword, View all Purchased Products with Total Bill.

`

5.3 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 Object Management Group.

The goal is for UML to become a common language for creating models of objec 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 Unified Modeling Language is a standard language for specifying, Visualization, 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 object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Goals:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.

`

5.3.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

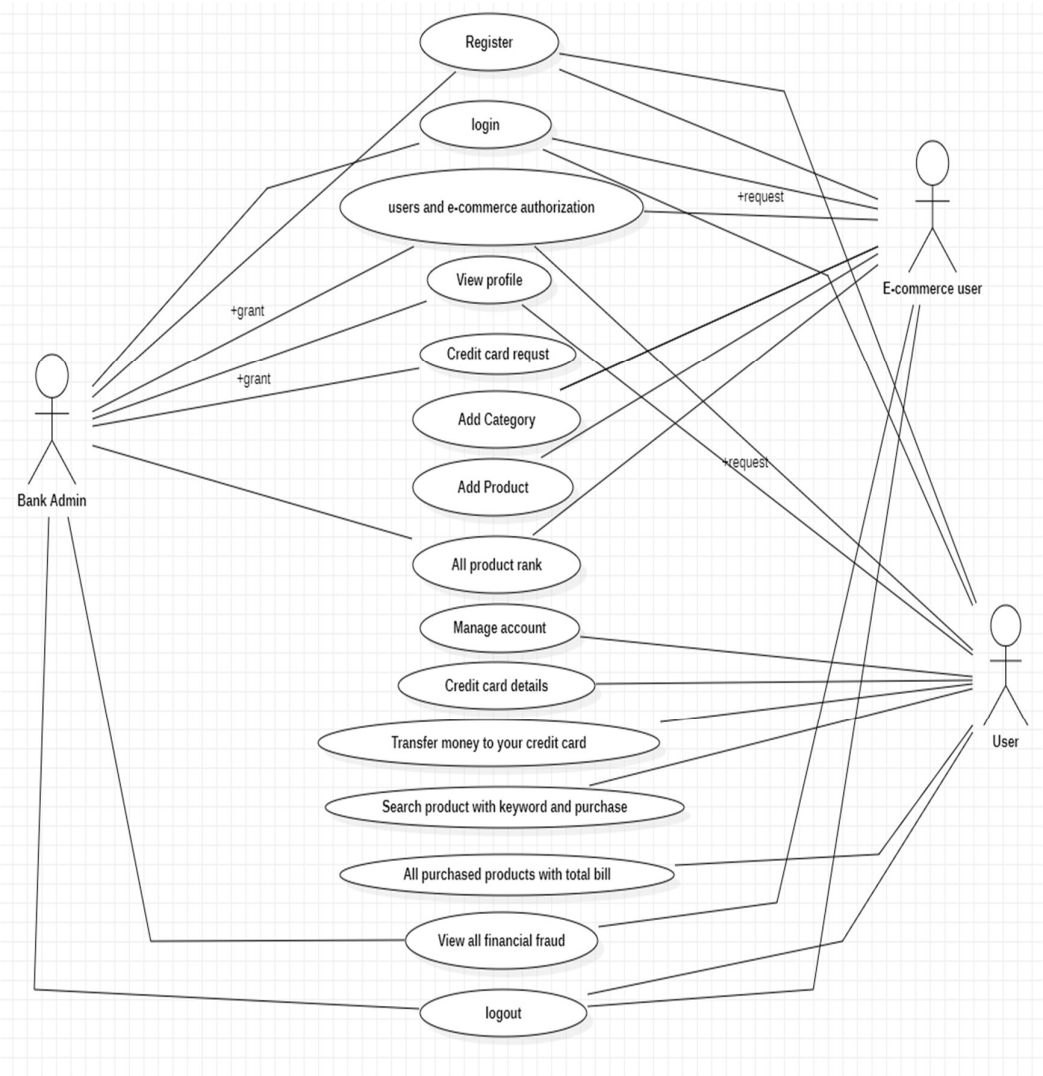


Fig No. 5.2 Use case Diagram

`

5.3.2 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

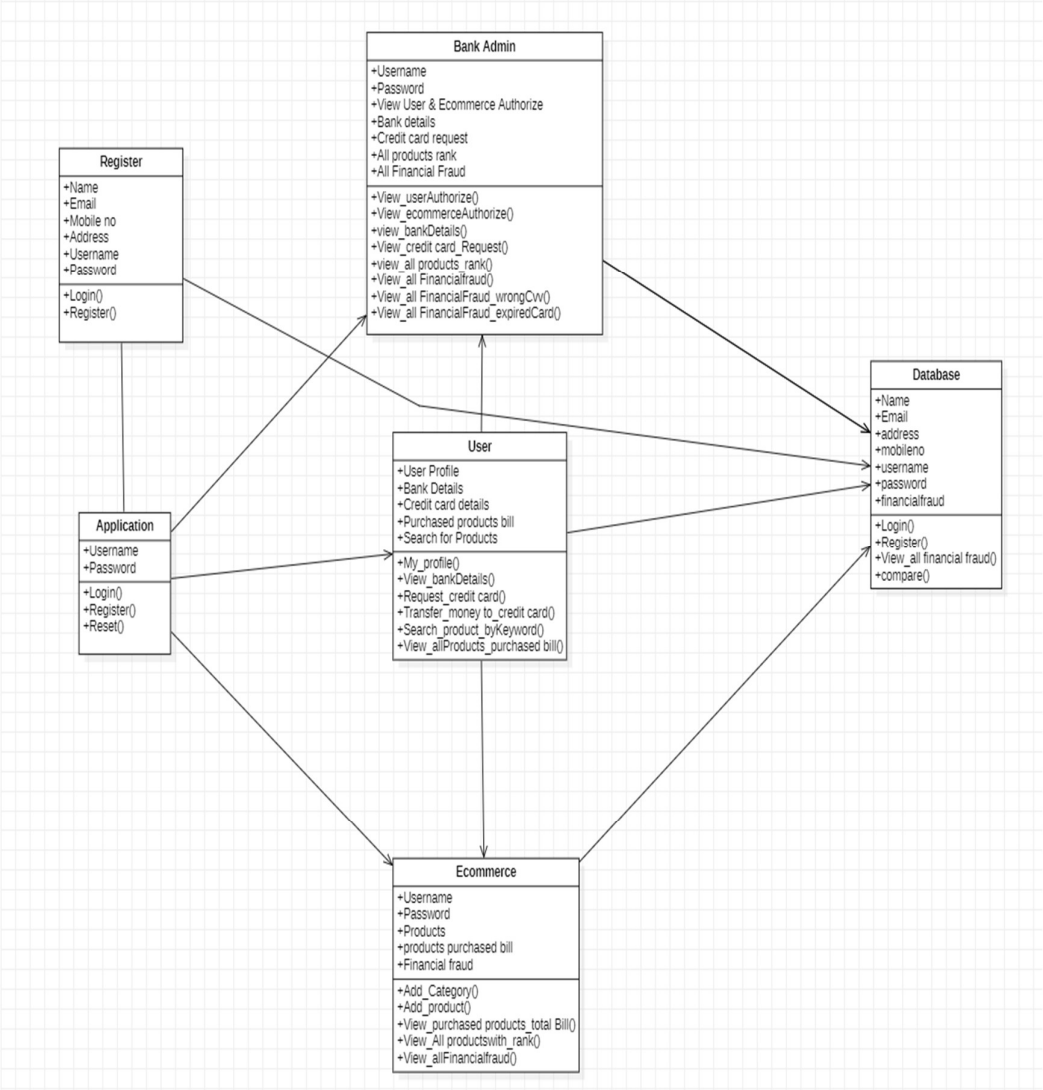


Fig No. 5.3 Class Diagram

`

5.3.4 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

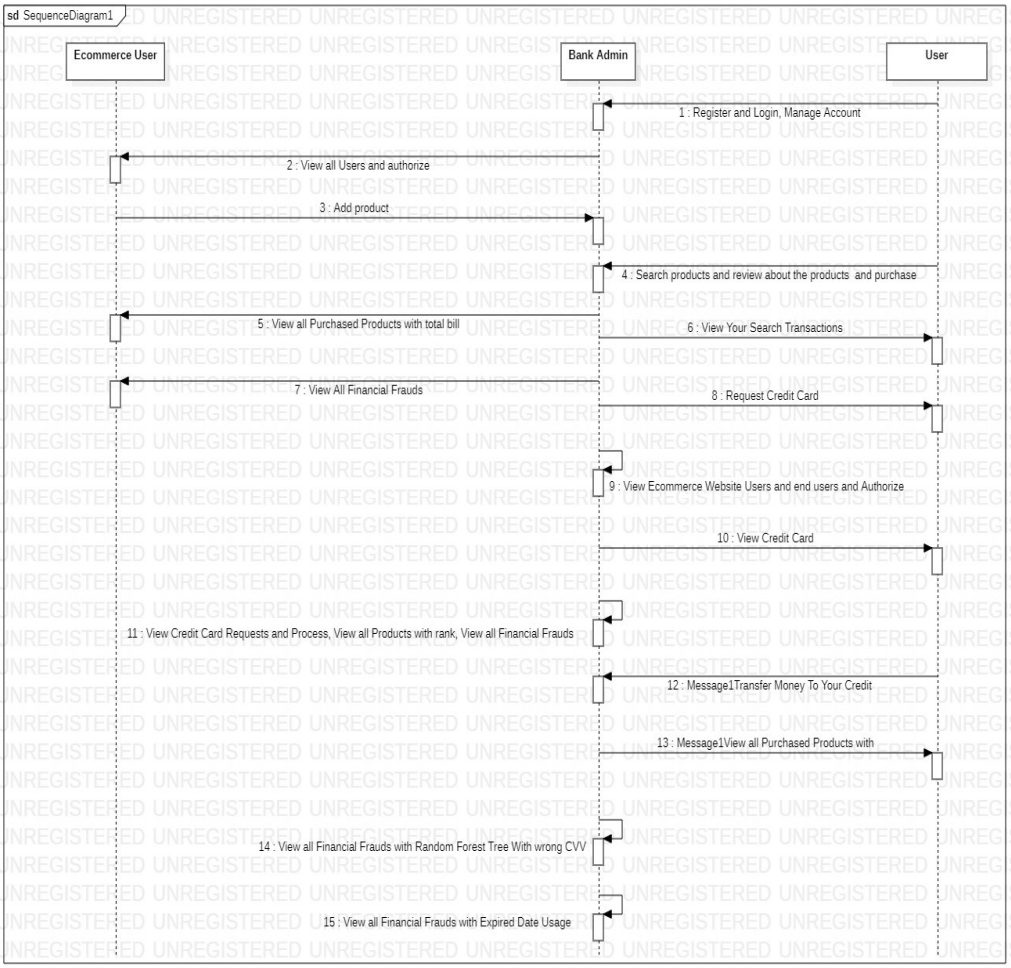


Fig No. 5.4 Sequence Diagram

Sequence Diagrams captures:

- The interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
- High-level interactions between user of the system and the system, between the system and other systems, or between subsystems

`

CHAPTER 6

SOURCE CODE

Get_pic.jsp

```
<%@ include file="connect.jsp" %>
<%@ page import="java.sql.*,java.io.*,java.util.*" %>

<%
        try{
            String pic=request.getParameter("picture");
                String p_Name=request.getParameter("p_Name");
                int id = Integer.parseInt(request.getParameter("id"));
            byte[] imageData=null;
            Blob b=null;
                if(pic.equalsIgnoreCase("bankadmin"))
                {
                                Statement st=connection.createStatement();
                                String strQuery = "select image from bankadmin where
                                                                id="+id+"" ;
                                ResultSet rs = st.executeQuery(strQuery);
                                rs = st.executeQuery(strQuery);
                                if(rs.next())
                                {
                                b = rs.getBlob(1);
                             imageData=b.getBytes(1, (int)b.length());
                            }
                            response.setContentType("image/gif");
                            OutputStream o=response.getOutputStream();
                            o.write(imageData);
                            o.flush();
                            o.close();
                        }
                        else if(pic.equals("userimage"))
```

`

```
{
                Statement st=connection.createStatement();
                String strQuery = "select image from user where
                                                id="+id+"" ;
                ResultSet rs = st.executeQuery(strQuery);
                String imgLen="";
                if(rs.next())
                {
                b = rs.getBlob(1);
        imageData=b.getBytes(1, (int)b.length());

        }
        response.setContentType("image/gif");
        OutputStream o=response.getOutputStream();
        o.write(imageData);
        o.flush();
        o.close();

}
else if(pic.equals("productimage"))
 {
                Statement st=connection.createStatement();
                String strQuery = "select p_image from products where
                                                id="+id+"" ;
                ResultSet rs = st.executeQuery(strQuery);
                String imgLen="";
                if(rs.next())
                {
                b = rs.getBlob(1);
        imageData=b.getBytes(1, (int)b.length());

        }
        response.setContentType("image/gif");
        OutputStream o=response.getOutputStream();
```

```
`
                    o.write(imageData);
                    o.flush();
                    o.close();


                }
                else if(pic.equals("bankimage"))
                 {
                            Statement st=connection.createStatement();
                            String strQuery = "select b_image from bank where
                                                    id="+id+"“;
                            ResultSet rs = st.executeQuery(strQuery);
                            String imgLen="";
                            if(rs.next())
                            {
                            b = rs.getBlob(1);
                     imageData=b.getBytes(1, (int)b.length());
                    }
                    response.setContentType("image/gif");
                    OutputStream o=response.getOutputStream();
                    o.write(imageData);
                    o.flush();
                    o.close();
                }
                else{}
        }
        catch (Exception e){
        e.printStackTrace();
        }
%>
</body>
</html>
```

Connect.jsp

`

```
<title></title>
<%@ page import="java.sql.*"%>
<%@ page import="java.util.*" %>
<%
	Connection connection = null;
	try {
		Class.forName("com.mysql.jdbc.Driver");
		connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/cc_fraud","root","root");
	String sql="";
	}
	catch(Exception e)
	{
		System.out.println(e);
	}
%>
```

View all Financial fraud.jsp

```
<%@ include file="connect.jsp" %>
<title>All Financial Frauds</title>
<%

String s1,s2,s3,s4,s5,s6,s7;
int i=0;
try
{
	String query="select * from financial_fraud";
	Statement st=connection.createStatement();
	ResultSet rs=st.executeQuery(query);
	while ( rs.next() )
	{
		i=rs.getInt(1);
		s1=rs.getString(2);
```

`

```
                    s2=rs.getString(4);
                    s3=rs.getString(5);
                    s4=rs.getString(6);
                    s5=rs.getString(8);
                    s6=rs.getString(9);
                    s7=rs.getString(10);
                    if(s7.equalsIgnoreCase("Wrong CVV"))
                    else{
 <td width="17" height="0" align="center"  valign="middle"><div align="center"
class="style12">
                    <b><%out.println(s7);%></b>
                  </div></td>
               }%>
</tr>
  }
connection.close();
catch(Exception e)
{
out.println(e.getMessage());
}
   protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
          throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");

      try{
         OutputStream out=response.getOutputStream();
    DefaultCategoryDataset barchart=new DefaultCategoryDataset();


          String query="select title from cybercomment ";
          ResultSetReturnImpl r=new ResultSetReturnImpl();
         ResultSet rs=r.getExecuteQuery(query);
         while(rs.next()){
```

```
`
            String title=rs.getString("title");


          String quer1="select count(title),title from cybercomment where
title='"+title+"'";
          ResultSetReturnImpl r1=new ResultSetReturnImpl();
         ResultSet rs1=r1.getExecuteQuery(quer1);


          while(rs1.next()){
              String status=rs1.getString("title");
              int ex=rs1.getInt(1);
               barchart.addValue(ex,"",status);
          }
         }
          JFreeChart BarChartObject=ChartFactory.createBarChart("", "", "count",
barchart, PlotOrientation.VERTICAL, true, true, false);
     response.setContentType("image/png");
     ChartUtilities.writeChartAsPNG(out,BarChartObject,580,400);
       }catch(Exception e){
        System.out.println(e);
       }
   }

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
      processRequest(request, response);
   }

   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
      processRequest(request, response);
```

```
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

`

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. There are various types of test. Each test type addresses a specific testing requirement.

7.1 Types of Tests

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

`

Functional test

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.
Systems/Procedures	: interfacing systems 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 tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

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

`

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 tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

`

CHAPTER 8

OUTPUT SCREENS

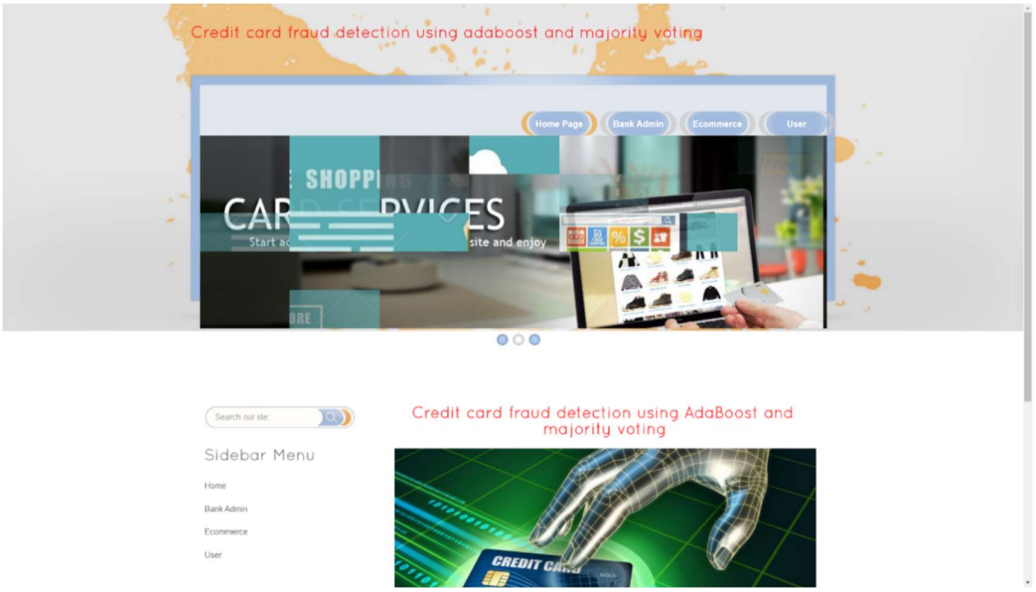


Fig No. 8.1 Home screen

The Above Screen is about the Home Screen of the project.

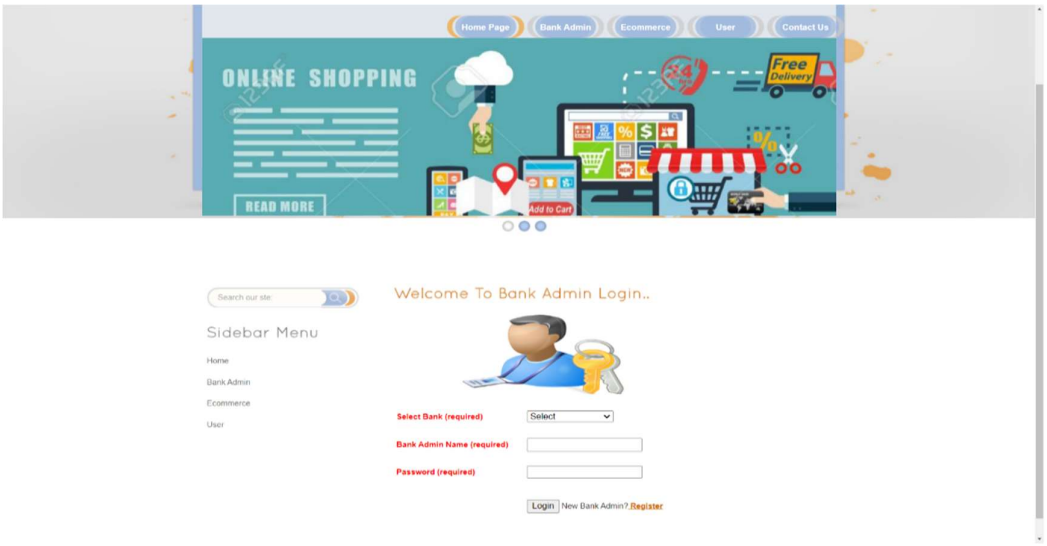


Fig No. 8.2 Bank Admin Login Screen

The above screen is about the Admin Login Page.

`

Fig No. 8.3 Bank Admin Home screen

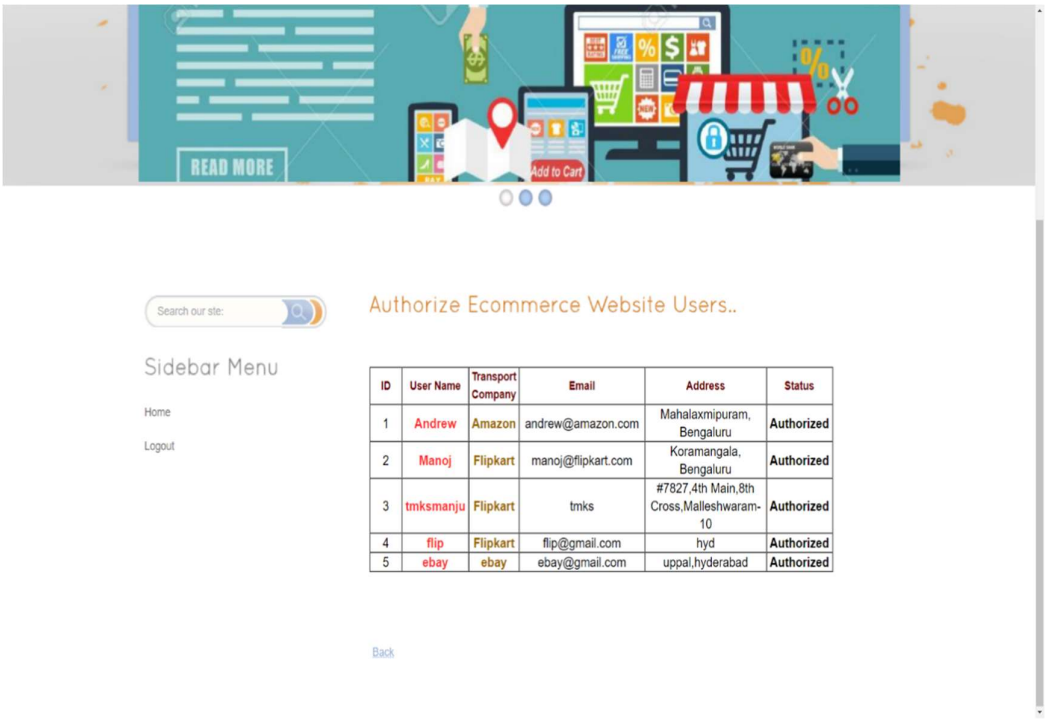
The Above screen is about Bank admin home screen.

ID	User Image	User Name	Mobile	Email	Address	Status
1		Sujan	9535866270	sujan.naik7@yahoo.com	BG Road, Bengaluru	Authorized
9		Ashwin	9535866270	ashwinmustari6@gmail.com	Kengeri, Bengaluru	Authorized
10		Sharan	9535866270	Sharan@gmail.com	Vijay Nagar, Bengaluru	Authorized
11		Shivaji	9535866270	shivaji@gmail.com	Hesaraghatta, Bengaluru	Authorized
12		Manjuanth	9535866270	tmksmanju13@gmail.com	#7827,11th Cross,Malleshwaram,Bangalore-10	Authorized

Fig No. 8.4 View users and authorize

The above screen is about the all users authorize by the bank admin

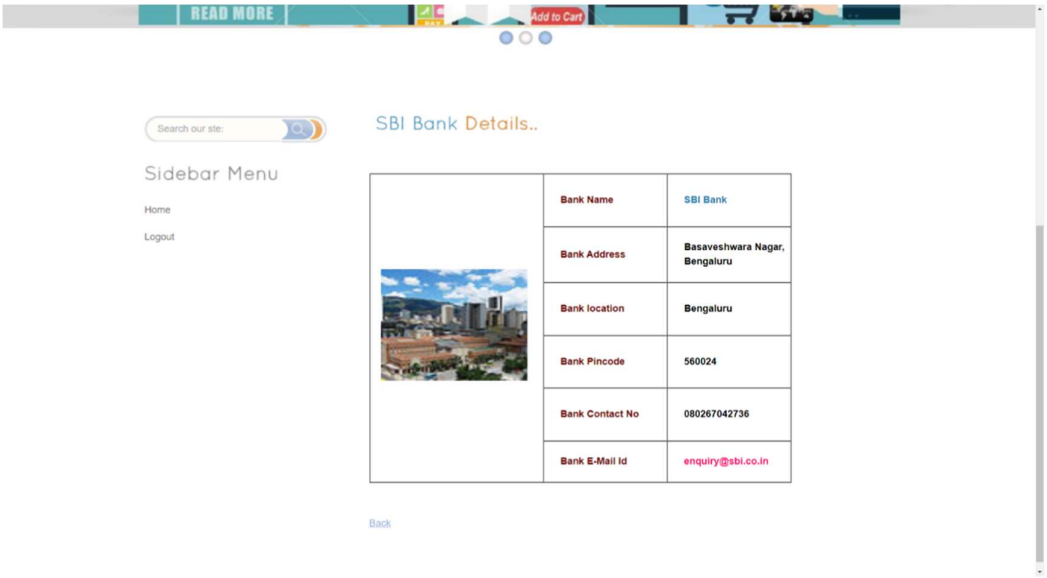
`



ID	User Name	Transport Company	Email	Address	Status
1	Andrew	Amazon	andrew@amazon.com	Mahalaxmipuram, Bengaluru	Authorized
2	Manoj	Flipkart	manoj@flipkart.com	Koramangala, Bengaluru	Authorized
3	tmksmanju	Flipkart	tmks	#7827,4th Main,8th Cross,Malleshwaram-10	Authorized
4	flip	Flipkart	flip@gmail.com	hyd	Authorized
5	ebay	ebay	ebay@gmail.com	uppal,hyderabad	Authorized

Fig No. 8.5 View all Ecommerce Authorize

The above screen is about the all ecommerce authorize by the bank admin.



Bank Name	SBI Bank
Bank Address	Basaveshwara Nagar, Bengaluru
Bank location	Bengaluru
Bank Pincode	560024
Bank Contact No	080267042736
Bank E-Mail Id	enquiry@sbi.co.in

Fig No. 8.6 View Bank Details

The above screen is about the admin bank details.

`

Fig No. 8.7 Ecommerce login page

The above screen is about the Ecommerce login page

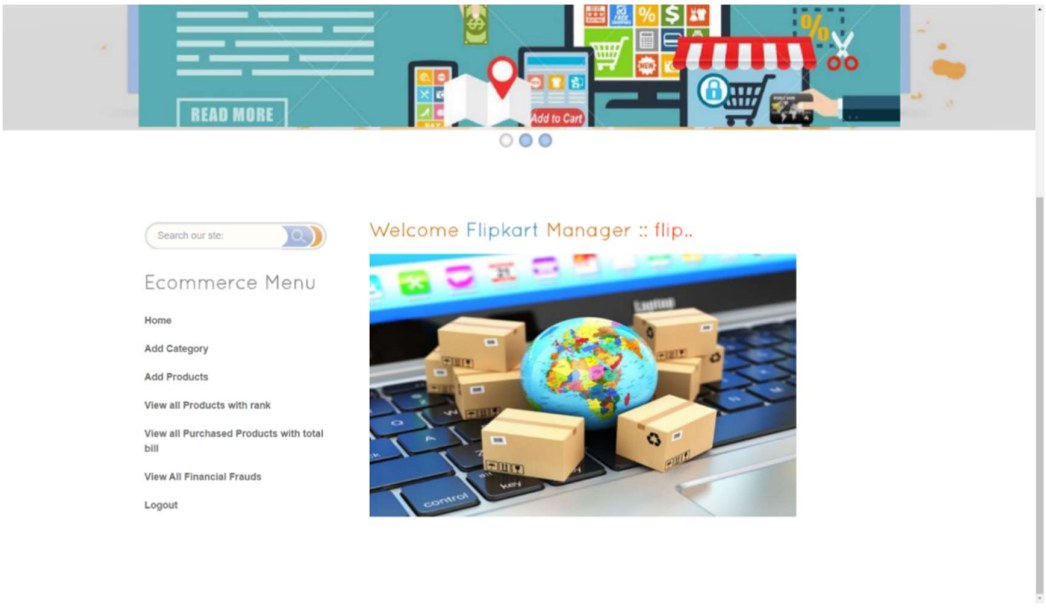


Fig No. 8.8 Ecommerce Home

The above screen is about the Ecommerce home page.

`

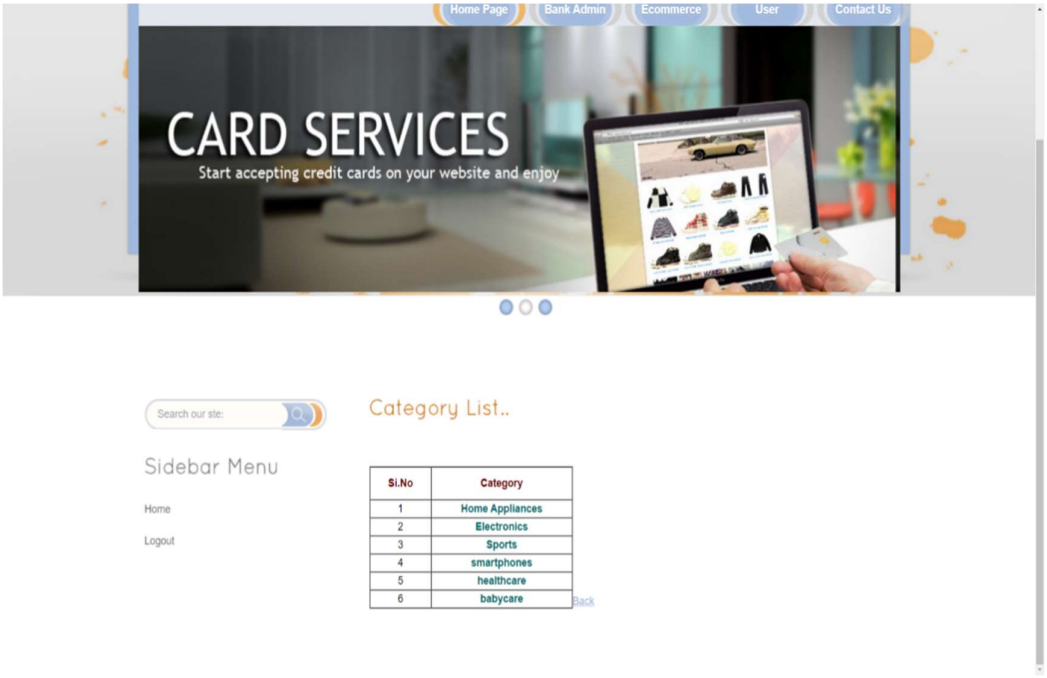


Fig No. 8.9 Products Category list

The above screen is about the products category list in the ecommerce site.

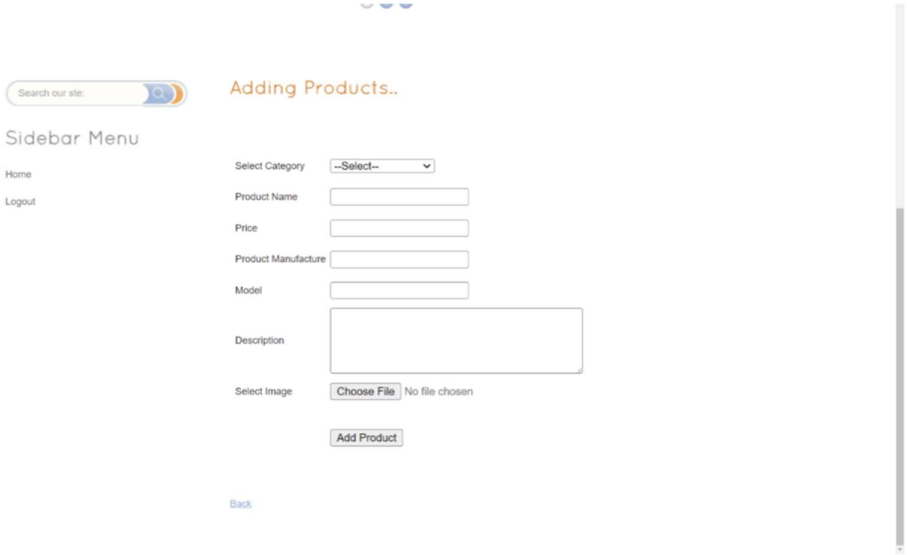


Fig No. 8.10 Add Products

The above screen is about the Adding products to the ecommerce site.

`

Fig No. 8.11 All products with rank

The above screen is about viewing of all product with the ranks in the ecommerce site.

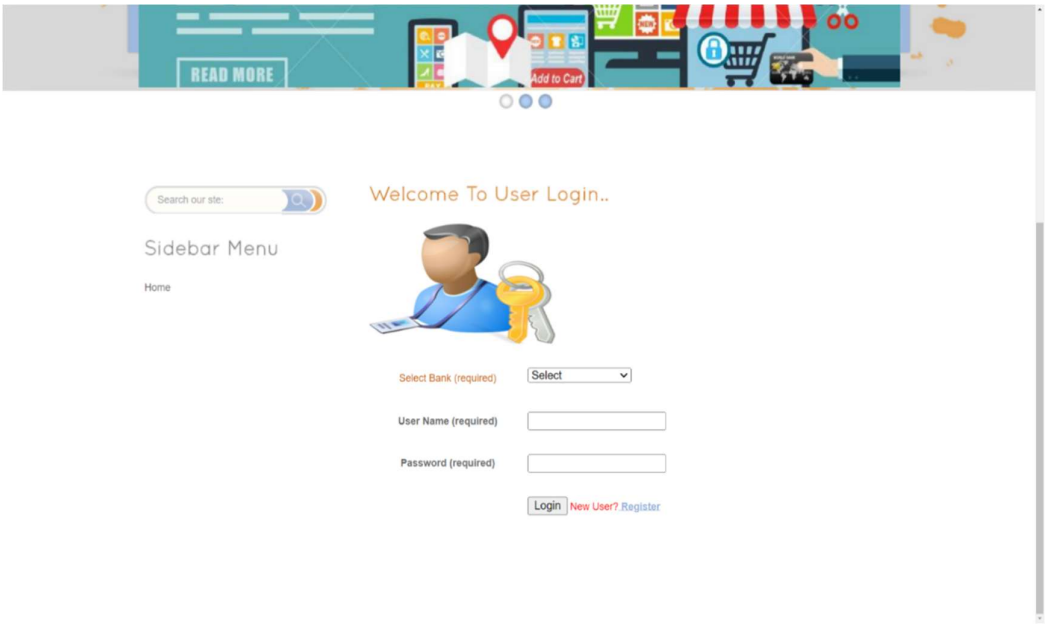


Fig No. 8.12 User Login page

The above screen is about the user login page.

`

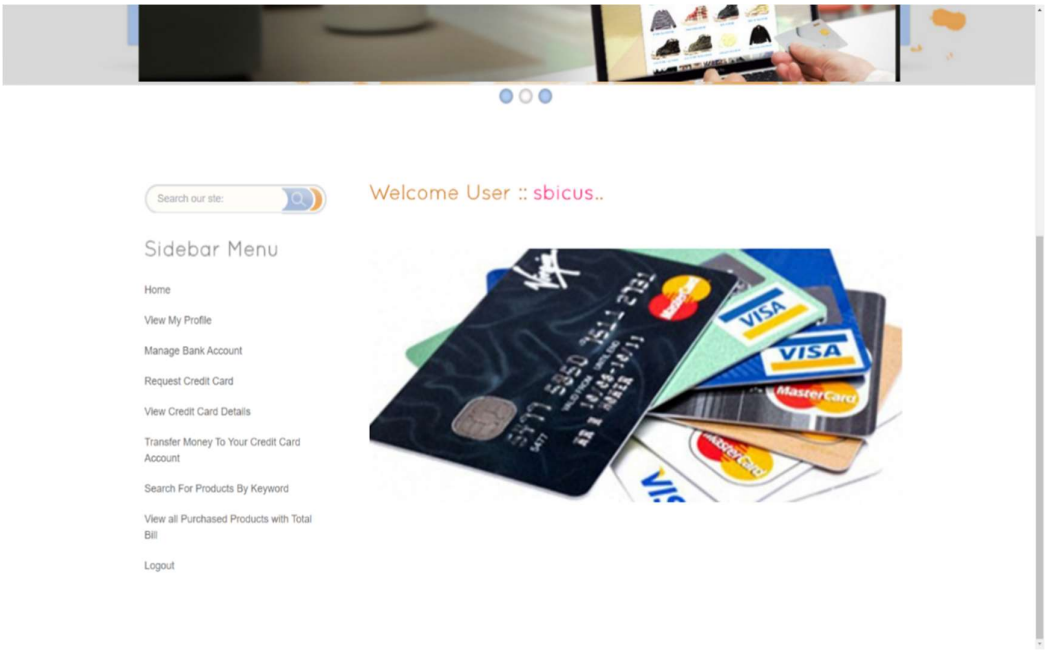


Fig No. 8.13 User Home page

The above screen is about the user home page and operations.

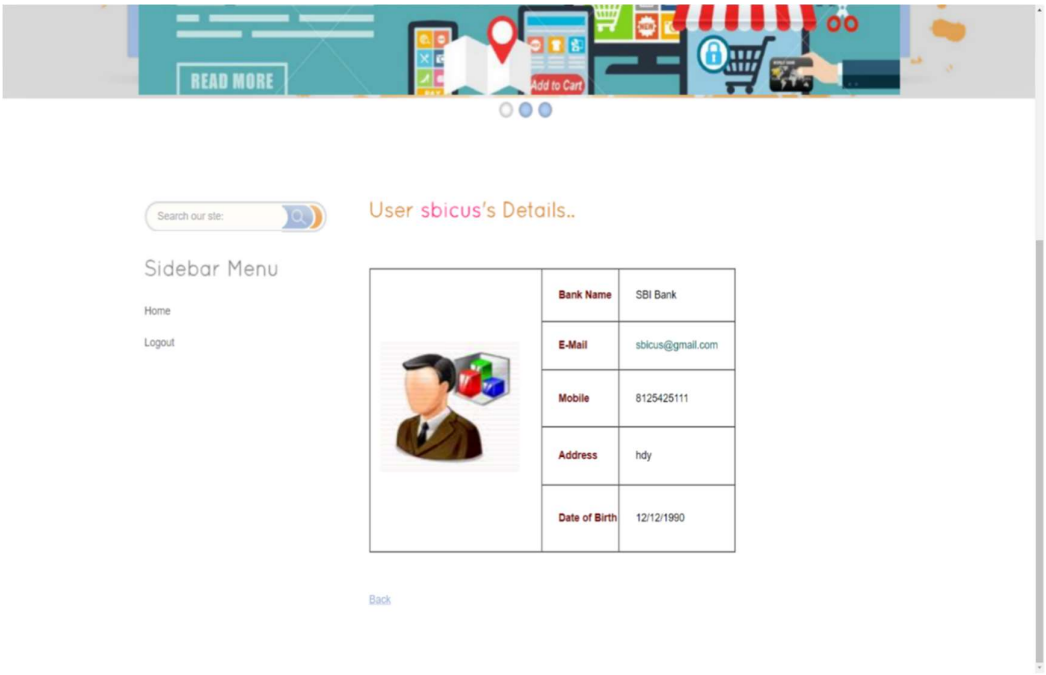


Fig No. 8.14 User Profile

The above screen is about the user profile.

`

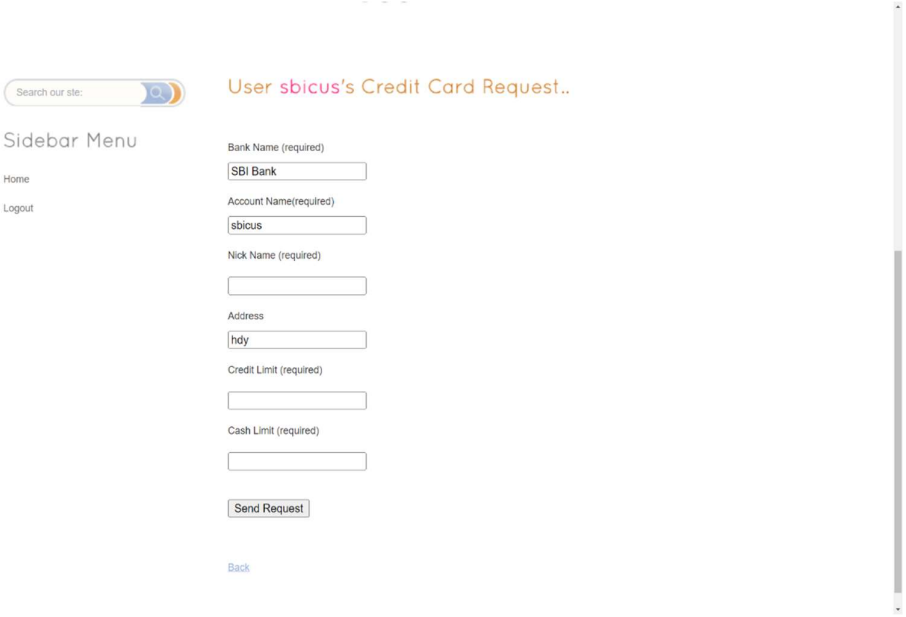


Fig No. 8.15 Credit card Request

The above screen is about user request for the credit card send to bank admin.

Fig No. 8.16 View credit card details

The above screen about details of the credit card that is generated by bank admin.

`

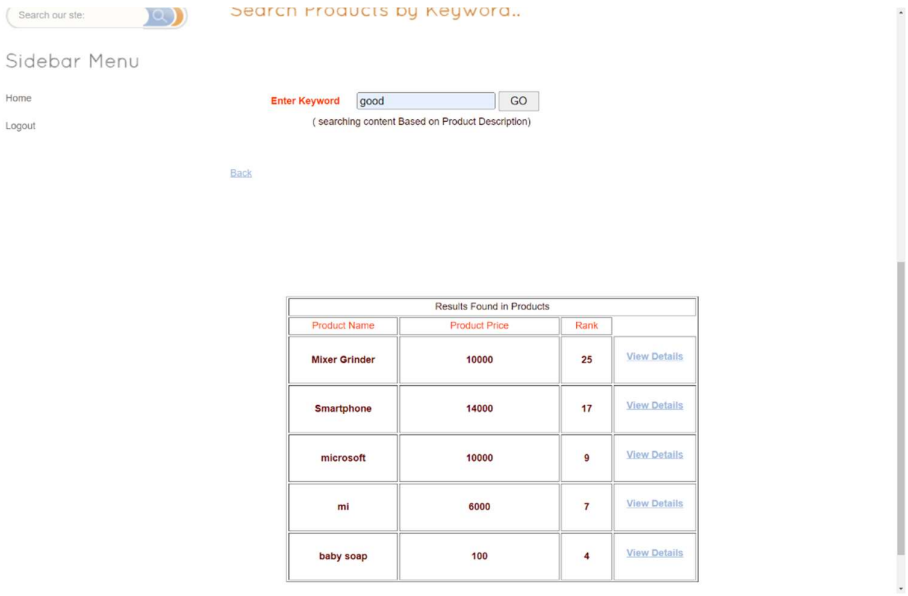


Fig No. 8.17 Search product by key word.

The above screen is about the user searching for the product by using the keyword.

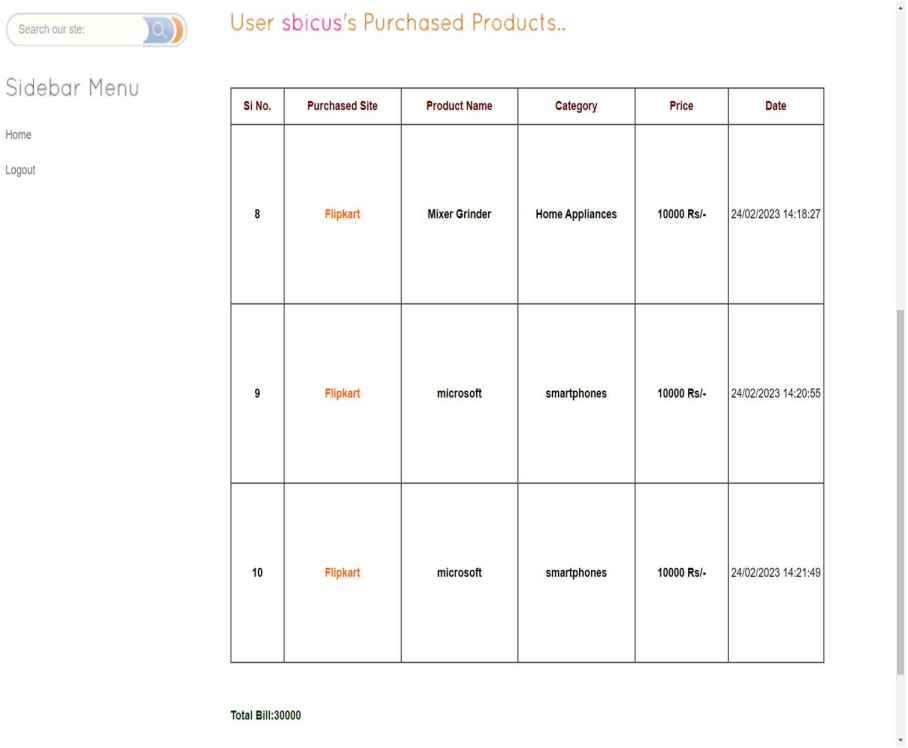
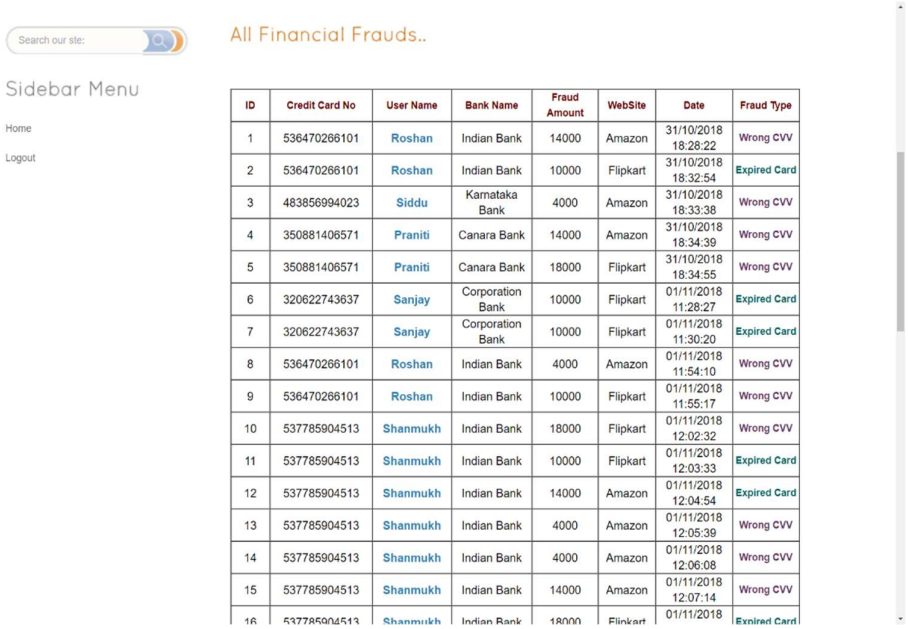


Fig No. 8.18 Purchased products bill

The above screen about the purchases made by the user and their total bill amount.

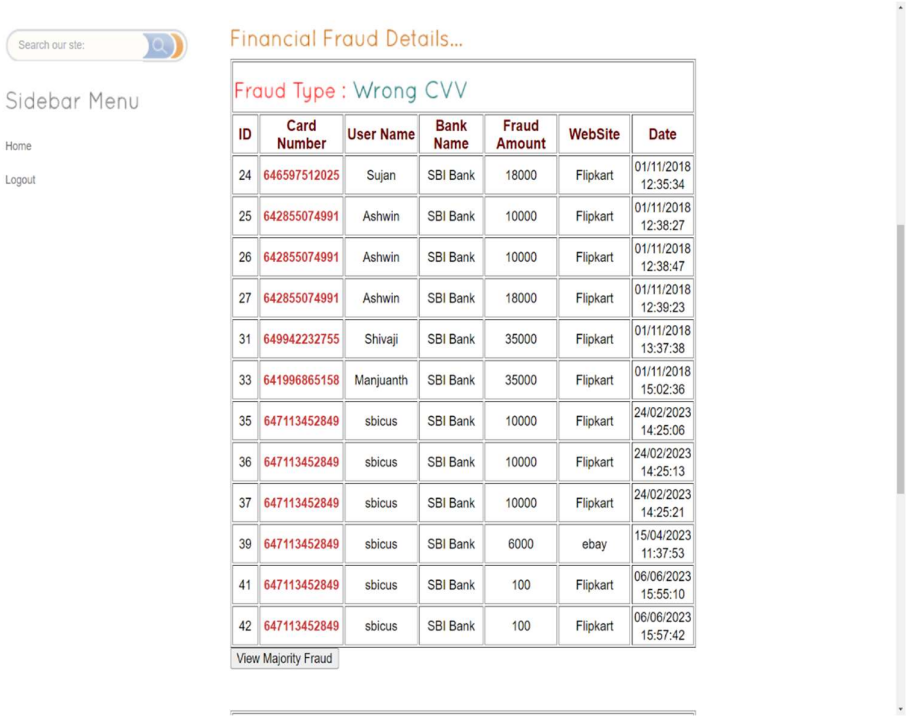
`



ID	Credit Card No	User Name	Bank Name	Fraud Amount	WebSite	Date	Fraud Type
1	536470266101	Roshan	Indian Bank	14000	Amazon	31/10/2018 18:28:22	Wrong CVV
2	536470266101	Roshan	Indian Bank	10000	Flipkart	31/10/2018 18:32:54	Expired Card
3	483856994023	Siddu	Karnataka Bank	4000	Amazon	31/10/2018 18:33:38	Wrong CVV
4	350881406571	Praniti	Canara Bank	14000	Amazon	31/10/2018 18:34:39	Wrong CVV
5	350881406571	Praniti	Canara Bank	18000	Flipkart	31/10/2018 18:34:55	Wrong CVV
6	320622743637	Sanjay	Corporation Bank	10000	Flipkart	01/11/2018 11:28:27	Expired Card
7	320622743637	Sanjay	Corporation Bank	10000	Flipkart	01/11/2018 11:30:20	Expired Card
8	536470266101	Roshan	Indian Bank	4000	Amazon	01/11/2018 11:54:10	Wrong CVV
9	536470266101	Roshan	Indian Bank	10000	Flipkart	01/11/2018 11:55:17	Wrong CVV
10	537785904513	Shanmukh	Indian Bank	18000	Flipkart	01/11/2018 12:02:32	Wrong CVV
11	537785904513	Shanmukh	Indian Bank	10000	Flipkart	01/11/2018 12:03:33	Expired Card
12	537785904513	Shanmukh	Indian Bank	14000	Amazon	01/11/2018 12:04:54	Expired Card
13	537785904513	Shanmukh	Indian Bank	4000	Amazon	01/11/2018 12:05:39	Wrong CVV
14	537785904513	Shanmukh	Indian Bank	4000	Amazon	01/11/2018 12:06:08	Wrong CVV
15	537785904513	Shanmukh	Indian Bank	14000	Amazon	01/11/2018 12:07:14	Wrong CVV
16	537785904513	Shanmukh	Indian Bank	18000	Flipkart	01/11/2018	Expired Card

Fig No. 8.19 All Financial Fraud.

The above screen is about the all financial fraud made by the user using credit card.



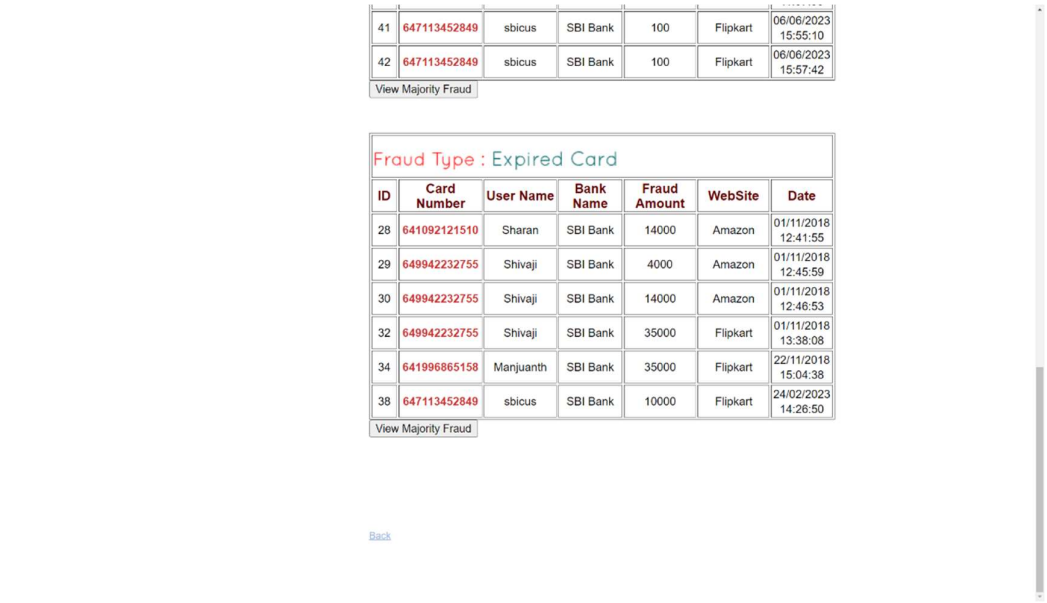
ID	Card Number	User Name	Bank Name	Fraud Amount	WebSite	Date
24	646597512025	Sujan	SBI Bank	18000	Flipkart	01/11/2018 12:35:34
25	642855074991	Ashwin	SBI Bank	10000	Flipkart	01/11/2018 12:38:27
26	642855074991	Ashwin	SBI Bank	10000	Flipkart	01/11/2018 12:38:47
27	642855074991	Ashwin	SBI Bank	18000	Flipkart	01/11/2018 12:39:23
31	649942232755	Shivaji	SBI Bank	35000	Flipkart	01/11/2018 13:37:38
33	641996865158	Manjuanth	SBI Bank	35000	Flipkart	01/11/2018 15:02:36
35	647113452849	sbicus	SBI Bank	10000	Flipkart	24/02/2023 14:25:06
36	647113452849	sbicus	SBI Bank	10000	Flipkart	24/02/2023 14:25:13
37	647113452849	sbicus	SBI Bank	10000	Flipkart	24/02/2023 14:25:21
39	647113452849	sbicus	SBI Bank	6000	ebay	15/04/2023 11:37:53
41	647113452849	sbicus	SBI Bank	100	Flipkart	06/06/2023 15:55:10
42	647113452849	sbicus	SBI Bank	100	Flipkart	06/06/2023 15:57:42

View Majority Fraud

Fig No. 8.20 All Financial Fraud with wrong CVV

The above screen is about the all-financial fraud with wrong CVV made by user.

`



41	647113452849	sbicus	SBI Bank	100	Flipkart	06/06/2023 15:55:10
42	647113452849	sbicus	SBI Bank	100	Flipkart	06/06/2023 15:57:42

View Majority Fraud

Fraud Type : Expired Card

ID	Card Number	User Name	Bank Name	Fraud Amount	WebSite	Date
28	641092121510	Sharan	SBI Bank	14000	Amazon	01/11/2018 12:41:55
29	649942232755	Shivaji	SBI Bank	4000	Amazon	01/11/2018 12:45:59
30	649942232755	Shivaji	SBI Bank	14000	Amazon	01/11/2018 12:46:53
32	649942232755	Shivaji	SBI Bank	35000	Flipkart	01/11/2018 13:38:08
34	641996865158	Manjuanth	SBI Bank	35000	Flipkart	22/11/2018 15:04:38
38	647113452849	sbicus	SBI Bank	10000	Flipkart	24/02/2023 14:26:50

View Majority Fraud

Back

Fig No. 8.21 All Financial Fraud with Expired card

The above screen is about the all financial fraud with expired card made by user.

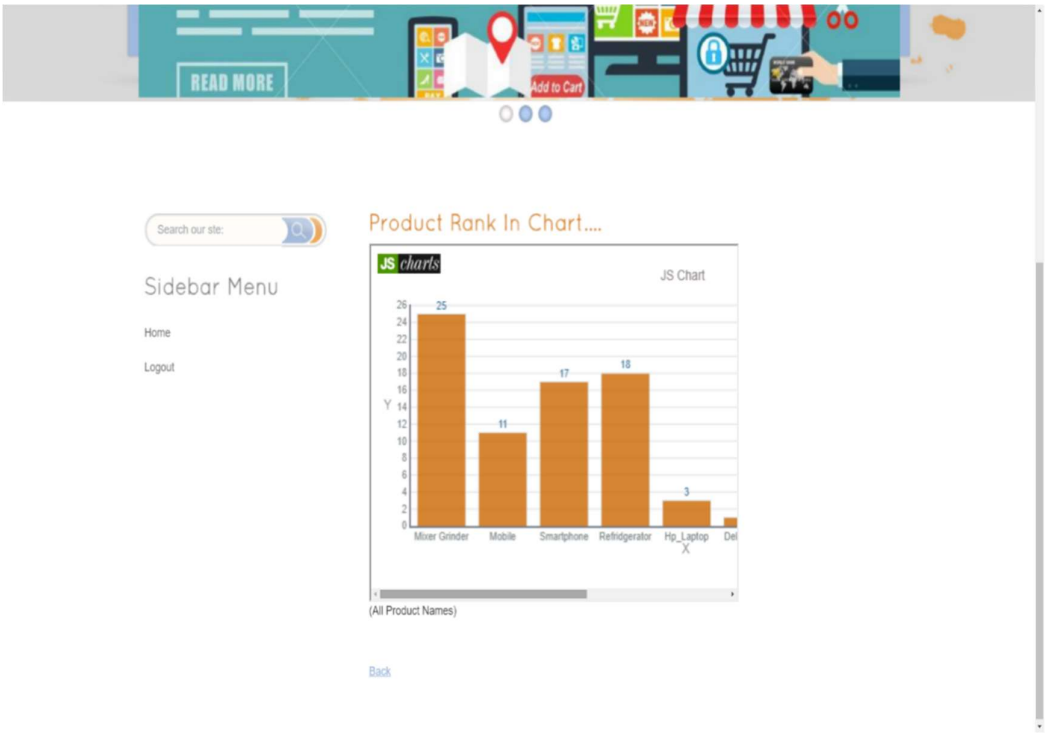


Fig No. 8.22 All Rank Chart

The above screen is about the all products and their ranks.

`

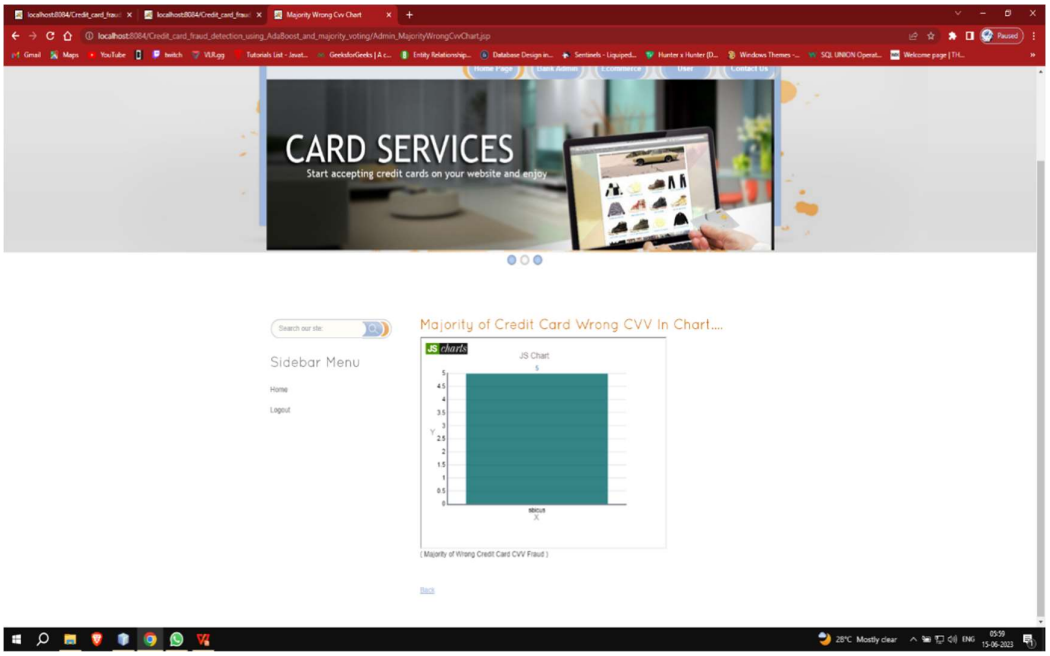


Fig No. 8.23 Majority Voting for wrong CVV

The above screen is about majority number of frauds made by using wrong CVV.

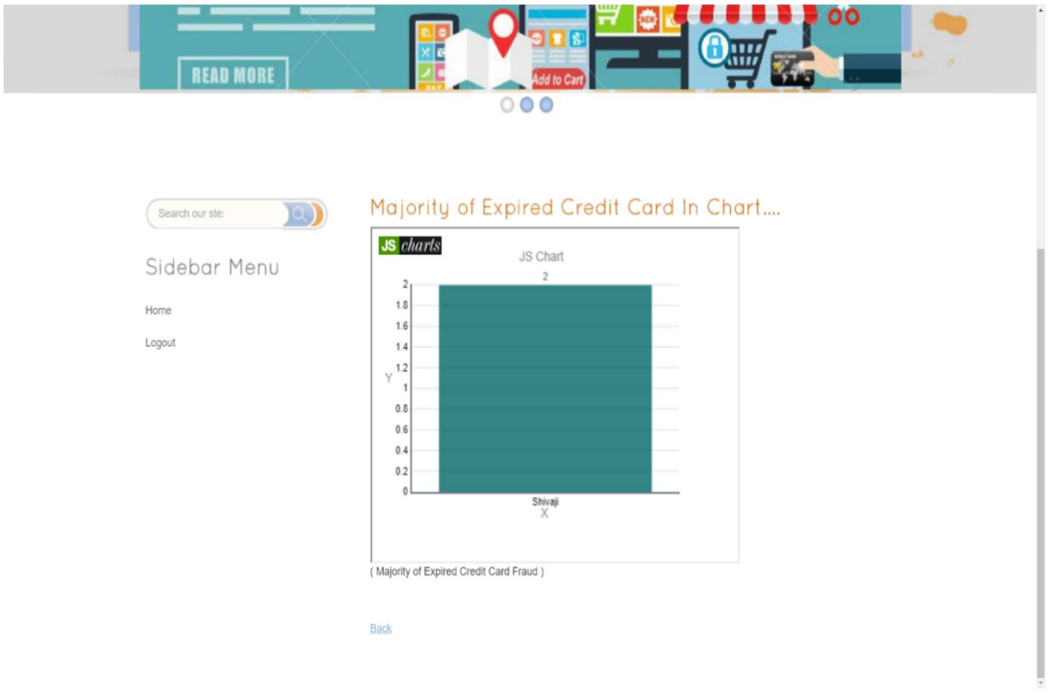


Fig No. 8.24 Majority Voting with expired card.

The above screen is about the majority number of frauds made by expired card.

`

CHAPTER 10

CONCLUSION

A study on credit card fraud detection using machine learning algorithms has been presented in this project. A number of standard models which include NB, SVM, and DL have been used in the empirical evaluation. A publicly available credit card data set has been used for evaluation using individual (standard) models and hybrid models using AdaBoost and majority voting combination methods. The MCC metric has been adopted as a performance measure, as it takes into account the true and false positive and negative predicted outcomes. The best MCC score is 0.823, achieved using majority voting. A real credit card data set from a financial institution has also been used for evaluation. The same individual and hybrid models have been employed. A perfect MCC score of 1 has been achieved using AdaBoost and majority voting methods. To further evaluate the hybrid models, noise from 10% to 30% has been added into the data samples. The majority voting method has yielded the best MCC score of 0.942 for 30% noise added to the data set. This shows that the majority voting method is stable in performance in the presence of noise. For future work, the methods studied in this paper will be extended to online learning models. In addition, other online learning models will be investigated. The use of online learning will enable rapid detection of fraud cases, potentially in real-time. This in turn will help detect and prevent fraudulent transactions before they take place, which will reduce the number of losses incurred every day in the financial sector.

`

CHAPTER 11

REFERENCES

1. Y. Sahin, S. Bulkan, and E. Duman, “A cost-sensitive decision tree approach for fraud detection,” Expert Systems with Applications, vol. 40, no. 15, pp. 5916–5923, 2013.
2. O. Adewumi and A. A. Akinyelu, “A survey of machine-learning and nature-inspired based credit card fraud detection techniques,” International Journal of System Assurance Engineering and Management, vol. 8, pp. 937–953, 2017.
3. A. Srivastava, A. Kundu, S. Sural, A. Majumdar, “Credit card fraud detection using hidden Markov model,” IEEE Transactions on Dependable and Secure Computing, vol. 5, no. 1, pp. 37–48, 2008.
4. The Nilson Report (October 2016) [Online]. Available: https://www.nilsonreport.com/upload/content_promo/The_Nilson_R eport_10-17-2016.pdf
5. J. T. Quah, and M. Sriganesh, “Real-time credit card fraud detection using computational intelligence,” Expert Systems with Applications, vol. 35, no. 4, pp. 1721–1732, 2008.
6. S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C., “Data mining for credit card fraud: A comparative study,” Decision Support Systems, vol. 50, no. 3, pp. 602–613, 2011.
7. N. S. Halvaiee and M. K. Akbari, “A novel model for credit card fraud detection using Artificial Immune Systems,” Applied Soft Computing, vol. 24, pp. 40–49, 2014.
8. S. Panigrahi, A. Kundu, S. Sural, and A. K. Majumdar, “Credit card fraud detection: A fusion approach using Dempster–Shafer theory and Bayesian learning,” Information Fusion, vol. 10, no. 4, pp. 354–363, 2009.
9. N. Mahmoudi and E. Duman, “Detecting credit card fraud by modified Fisher discriminant analysis,” Expert Systems with Applications, vol. 42, no. 5, pp. 2510–2516, 2015.
10. D. Sánchez, M. A. Vila, L. Cerda, and J. M. Serrano, “Association rules applied to credit card fraud detection,” Expert Systems with Applications, vol. 36, no. 2, pp. 3630–3640, 2009.

`

11. E. Duman and M. H. Ozcelik, “Detecting credit card fraud by genetic algorithm and scatter search,” Expert Systems with Applications, vol. 38, no. 10, pp. 13057–13063, 2011.
12. P. Ravisankar, V. Ravi, G. R. Rao, and I. Bose, “Detection of financial statement fraud and feature selection using data mining techniques,” Decision Support Systems, vol. 50, no. 2, pp. 491–500, 2011.
13. E. Kirkos, C. Spathis, and Y. Manolopoulos, “Data mining techniques for the detection of fraudulent financial statements,” Expert Systems with Applications, vol. 32, no. 4, pp. 995–1003, 2007.
14. F. H. Glancy and S. B. Yadav, “A computational model for financial reporting fraud detection,” Decision Support Systems, vol. 50, no. 3, pp. 595–601, 2011.
15. D. Olszewski, “Fraud detection using self-organizing map visualizing the user profiles,” Knowledge-Based Systems, vol. 70, pp. 324–334, 2014.