CREDIT FRAUD DETECTION USING MACHINE LEARNING

PROJECT REPORT

Submitted by

CHELLAPA HERLAN PIO R

Register No.: 20UBCS004

Under the guidance of

Dr.N.MOGANARANGAN,Ph.D

Professor & Head, Department of Computational Studies

in partial fulfillment of the requirements for the degree of

BACHELOR OF COMPUTER SCIENCE

DEPARTMENT OF COMPUTATIONAL STUDIES

SRI MANAKULA VINAYAGAR ENGINEERING COLLEGE

(An Autonomous Institution)

SCHOOL OF ARTS AND SCIENCE

MADAGADIPET, PUDUCHERRY - 605107

JULY 2023

SRI MANAKULA VINAYAGAR ENGINEERING COLLEGE
(An Autonomous Institution)
(Approved by AICTE, New Delhi & Affiliated to Pondicherry University)
(Accredited by NBA-AICTE, New Delhi, ISO 9001:2000 Certified Institution &
Accredited by NAAC with "A" Grade)
Madagadipet, Puducherry - 605 107

SCHOOL OF ARTS AND SCIENCE

DEPARTMENT OF COMPUTATIONAL STUDIES

BONAFIDE CERTIFICATE

This is to certify that the project work entitled "**CREDIT FRAUD DETECTION USING MACHINE LEARNING** " is a bonafide work done by **CHELLAPA HERLAN PIO R [REGISTER NO.: 20UBCS004]** in partial fulfillment of the requirement, for the award of B.Sc. Degree in Computer Science by Pondicherry University during the academic year 2022 - 2023.

PROJECT GUIDE **HEAD OF THE DEPARTMENT**

Submitted for the End Semester Practical Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I am very thankful and grateful to our beloved guide, **Dr.N.Moganarangan,Ph.D,** whose great support in valuable advices, suggestions and tremendous help enabled us in completing our project. He / She has been a great source of inspiration to us.

I also sincerely thank our Head of the Department, **Dr. N. MOGANARANGAN** whose continuous encouragement and sufficient comments enabled us to complete our project report.

My sincere thanks to **Dr. S. MUTHULAKSHMI**, Dean, School of Arts and Science, SMVEC, for her effort and valuable guidance on my project.

I thank all our **Staff members** who have been by our side always and helped us with our project. We also sincerely thank all the lab technicians for their help as in the course of our project development.

I would also like to extend our sincere gratitude and grateful thanks to our Director cum Principal **Dr. V. S. K. VENKATACHALAPATHY** for having extended the Research and Development facilities of the department.

I am grateful to our Founder Chairman **Shri. N. KESAVAN**. He has been a constant source of inspiration right from the beginning.

I would like to express our faithful and grateful thanks to our Chairman and Managing Director **Shri. M. DHANASEKARAN** for his support.

I would also like to thank our Vice Chairman **Shri. S. V. SUGUMARAN**, for providing us with pleasant learning environment.

I would like to thank our Secretary **Dr. K. NARAYANASAMY** for his support.

I wish to thank our **family members and friends** for their constant encouragement, constructive criticisms and suggestions that has helped us in timely completion of this project.

Last but not the least; we would like to thank the **ALMIGHTY** for His grace and blessings over us throughout the project.

ABSTRACT

Credit card fraud is a significant concern for financial institutions and their customers, with billions of dollars lost each year. Traditional fraud detection methods have proven to be insufficient in preventing fraudulent transactions. Therefore, machine learning techniques have been employed to develop more accurate and efficient fraud detection systems. This paper presents an abstract for credit card fraud detection using machine learning, which outlines the key challenges and the potential benefits of employing machine learning techniques in fraud detection.

The proposed system employs supervised learning algorithms such as logistic regression, decision trees, and random forests to identify fraudulent transactions. The system also utilizes unsupervised learning techniques like clustering to detect new patterns of fraud. The results of the study demonstrate that machine learning techniques can effectively identify fraudulent transactions with high accuracy, thereby helping to minimize financial losses due to fraud. Credit card fraud detection is an important task for financial institutions to protect their customers from fraudulent activities. Machine learning techniques have been widely adopted in the finance industry to detect fraudulent transactions in real-time.

This paper presents an overview of credit card fraud detection using machine learning techniques. It discusses various machine learning algorithms such as logistic regression, decision tree, random forest, and neural networks, and their application in credit card fraud detection. The paper also discusses the challenges and limitations of machine learning in fraud detection, including imbalanced data, overfitting, and interpretability. Finally, the paper concludes with future directions for credit card fraud detection using machine learning, such as the use of deep learning and reinforcement learning techniques.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The popularity of online shopping is growing day by day. According to an ACNielsen study conducted in 2005, one-tenth of the world's population is shopping online. Germany and Great Britain have the largest number of online shoppers, and credit card is the most popular mode of payment (59 percent). About 350 million transactions per year were reportedly carried out by Barclaycard, the largest credit card company in the United Kingdom, toward the end of the last century.

Retailers like Wal-Mart typically handle much larger number of credit card transactions including online and regular purchases. As the number of credit card users rises world-wide, the opportunities for attackers to steal credit card details and, subsequently, commit fraud are also increasing. The total credit card fraud in the United States itself is reported to be $2.7 billion in 2005 and estimated to be $3.0 billion in 2006, out of which $1.6 billion and $1.7 billion, respectively, are the estimates of online fraud.

In a physical-card-based purchase, the cardholder presents his card physically to a merchant for making a payment. To carry out fraudulent transactions in this kind of purchase, an attacker has to steal the credit card. If the cardholder does not realize the loss of card, it can lead to a substantial financial loss to the credit card company. In these kind of purchase, only some important information about a card (card number, expiration date, secure code) is required to make the payment.

Such purchases are normally done on the Internet or over the telephone. To commit fraud in these types of purchases, a fraudster simply needs to know the card details. Most of the time, the genuine cardholder is not aware that someone else has seen or stolen his card information. The only way to detect this kind of fraud is to analyze the spending patterns on every card and to figure out any inconsistency with respect to the "usual" spending patterns.

Fraud detection based on the analysis of existing purchase data of cardholder is a promising way to reduce the rate of successful credit card frauds. Since humans tend to exhibit specific behaviouristic profiles, every cardholder can be represented by a set of patterns containing information about the typical purchase category, the time since the last purchase, the amount

of money spent, etc. Deviation from such patterns is a potential threat to the system. Several techniques for the detection of credit card fraud have been proposed in the last few years.

Credit card fraud detection has drawn a lot of research interest and a number of techniques, with special emphasis on data mining and neural networks, have been suggested. Ghosh and Reilly have proposed credit card fraud detection with a neural network. They have built a detection system, which is trained on a large sample of labelled credit card account transactions.

These transactions contain exam-ple fraud cases due to lost cards, stolen cards, application fraud, counterfeit fraud, mail-order fraud, and nonreceived issue (NRI) fraud. Recently, Syeda et al. have used parallel granular neural networks (PGNNs) for improving the speed of data mining and knowledge discovery process in credit card fraud detection. A complete system has been implemented for this purpose.

Stolfo et al. suggest a credit card fraud detection system (FDS) using Metalearning techniques to learn models of fraudulent credit card transactions. Metalearning is a general strategy that provides a means for combining and integrating a number of separately built classifiers or models.

A metaclassifier is thus trained on the correlation of the predictions of the base classifiers. The same group has also worked on a cost-based model for fraud and intrusion detection. They use Python agents for Metalearning (JAM), which is a distributed data mining system for credit card fraud detection. A number of important performance metrics like True Positive—False Positive (TP-FP) spread and accuracy have been defined by them.

Aleskerov et al. present CARDWATCH, a database mining system used for credit card fraud detection.

The system, based on a neural learning module, provides an interface to a variety of commercial databases. Kim and Kim have identified skewed distribution of data and mix of legitimate and fraudulent transactions as the two main reasons for the complexity of credit card fraud detection.

Based on this observation, they use fraud density of real transaction data as a confidence value and generate the weighted fraud score to reduce the number of misdetections. Fan et al. suggest the application of distributed data mining in credit card fraud detection. Brause et al. have developed an approach that involves advanced data mining techniques and neural network algorithms to obtain high fraud coverage. Chiu and Tsai have proposed Web services and data mining techniques to establish a collaborative scheme for fraud detection in the banking

industry. With this scheme, participating banks share knowledge about the fraud patterns in a heterogeneous and distributed environment. To establish a smooth channel of data exchange, Web services techniques such as XML, SOAP, and WSDL are used. Phua et al. have done an extensive survey of existing data-mining-based FDSs and published a comprehensive report. Prodromidis and Stolfo use an agent-based approach with distributed learning for detect-ing frauds in credit card transactions.

It is based on artificial intelligence and combines inductive learning algorithms and Metalearning methods for achieving higher accuracy. Phua et al. suggest the use of metaclassifier similar to in fraud detection problems. They consider naive Bayesian C4.5, and Back Propagation neural networks as the base classifiers. A metaclassifier is used to determine which classifier should be considered based on skewness of data. Although they do not directly use credit card fraud detection as the target application, their approach is quite generic.

Vatsa et al. have recently proposed a game-theoretic approach to credit card fraud detection. They model the interaction between an attacker and an FDS as a multistage game between two players, each trying to maximize his payoff. The problem with most of the abovementioned approaches is that they require labelled data for both genuine, as well as fraudulent transactions, to train the classifiers. Getting real-world fraud data is one of the biggest problems associated with credit card fraud detection.

Also, these approaches cannot detect new kinds of frauds for which labelled data is not available. In contrast, we present a Hidden Markov Model (AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING)-based credit card FDS, which does not require fraud signatures and yet is able to detect frauds by considering a cardholder's spending habit. We model a credit card transaction processing sequence by the stochastic process of an AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING. The details of items purchased in individual transactions are usually not known to an FDS running at the bank that issues credit cards to the cardholders.

This can be represented as the underlying finite Markov chain, which is not observable. The transactions can only be observed through the other stochastic process that produces the sequence of the amount of money spent in each transaction. Hence, we feel that AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING is an ideal choice for addressing this problem.

CHAPTER 2

LITERATURE SURVEY

Rapid application development is a software development methodology that involves methods like iterative development and software prototyping. According to Whitten (2004), it is a merger of various structured techniques, especially data-driven Information Engineering, with prototyping techniques to accelerate software systems development.

In rapid application development, structured techniques and prototyping are especially used to define users' requirements and to design the final system. The development process starts with the development of preliminary data models and business process models using structured techniques. In the next stage, requirements are verified using prototyping, eventually to refine the data and process models. These stages are repeated iteratively; further development results in "a combined business requirements and technical design statement to be used for constructing new systems".

RAD approaches may entail compromises in functionality and performance in exchange for enabling faster development and facilitating application maintenance. Software project management is an activity of organising, planning and scheduling the software projects. The goal of software project management is to deliver the software product in given time and within the budget. It is also necessary that the software project should project should be developed in accordance with the requirements of the organisation. The software project managers are responsible for planning and scheduling the software projects. Managing software projects is difficult because of Following reasons:-

1. Other engineering products are visible. But the software is intangible. It is not a physical entity that can be touched. Hence software project managers can see the progress of software project with the help of documents only.
2. The large projects are different in nature. Hence even the managers who have adequate experiences of previous projects may find difficulties in the current project.
3. There are no standard software process. It changes from one organisation to another.

CHAPTER 3

SYSTEM STUDY

In this phase a detailed appraisal of the existing system is explained. This appraisal includes how the system works and what it does. It also includes finding out in more detail- what are the problems with the system and what user requires from the new system or any new change in system. The output of this phase results in the detail model of the system. The model describes the system functions and data and system information flow. The phase also contains the detail set of user requirements and these requirements are used to set objectives for the new system.

3.1 EXISTING SYSTEM:

In case of the existing system the fraud is detected after the fraud is done that is, the fraud is detected after the complaint of the holder. And so the card holder faced a lot of trouble before the investigation finish. And also as all the transaction is maintained in a log, we need to maintain a huge data, and also now a day's lot of online purchase are made so we don't know the person how is using the card online, we just

capture the ip address for verification purpose. So there need a help from the cybercrime to investigate the fraud. To avoid the entire above disadvantage we propose the system to detect the fraud in a best easy way.

The method and apparatus for pre-authorizing transactions includes providing a communications device to a vendor and a credit card owner. The credit card owner initiates a credit card transaction by communicating to a credit card number, and storing therein, a distinguishing piece of information that characterizes a specific transaction to be made by an authorized user of the credit card at a later time.

The information is accepted as "network data" in the data base only if a correct personal identification code (PIC) is used with the communication. The "network data" will serve to later authorize that specific transaction. The credit card owner or other authorized user can then only make that specific transaction with the credit card. Because the transaction is pre-authorized, the vendor does not need to see or transmit a PIC.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM:

These transactions contain exam-ple fraud cases due to lost cards, stolen cards, application fraud, counterfeit fraud, mail-order fraud, and nonreceived issue (NRI) fraud. Recently, Syeda et al. have used parallel granular neural networks (PGNNs) for improving the speed of data mining and knowledge discovery process in credit card fraud detection. A complete system has been implemented for this purpose.

Stolfo et al. suggest a credit card fraud detection system (FDS) using Metalearning techniques to learn models of fraudulent credit card transactions. Metalearning is a general strategy that provides a means for combining and integrating a number of separately built classifiers or models.

A metaclassifier is thus trained on the correlation of the predictions of the base classifiers. The same group has also worked on a cost-based model for fraud and intrusion detection. They use Python agents for Metalearning (JAM), which is a distributed data mining system for credit card fraud detection. A number of important performance metrics like True Positive—False Positive (TP-FP) spread and accuracy have been defined by them.

Aleskerov et al. present CARDWATCH, a database mining system used for credit card fraud detection.

The system, based on a neural learning module, provides an interface to a variety of commercial databases. Kim and Kim have identified skewed distribution of data and mix of legitimate and fraudulent transactions as the two main reasons for the complexity of credit card fraud detection.

Based on this observation, they use fraud density of real transaction data as a confidence value and generate the weighted fraud score to reduce the number of misdetections. Fan et al. suggest the application of distributed data mining in credit card fraud detection. Brause et al. have developed an approach that involves advanced data mining techniques and neural network algorithms to obtain high fraud coverage. Chiu and Tsai have proposed Web services and data mining techniques to establish a collaborative scheme for fraud detection in the banking industry. With this scheme, participating banks share knowledge about the fraud patterns in a heterogeneous and

distributed environment. To establish a smooth channel of data exchange, Web services techniques such as XML, SOAP, and WSDL are used. Phua et al. have done an extensive survey of existing data-mining-based FDSs and published a comprehensive report. Prodromidis and Stolfo use an agent-based approach with distributed learning for detecting frauds in credit card

3.2PROPOSED SYSTEM:

In this system ,we present a hidden morkov model(AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING) Which does not required fraud signatures and yet is able to detect frauds by considering a cardholder's spending habit. Card transaction processing sequence by the stochastic process of an AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING. The details of items purchased in individual transactions are usually not known to an FDS running at the bank that issues credit cards to the cardholder.

Hence, we feel that AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING is an ideal choice for addressing this problem. Another important advantage of the AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING based approach is a drastic reduction is the number reduction in the number of false positives transactions identified a malicious by an FDS although they are actually genuine.

An FDS runs at a credit card issuing bank. Each incoming transaction is submitted to the FDS for verification. FDS receives the card details and the values if purchases to verify, whether the transaction is genuine or not. The types of goods that are bought in that transaction are not known to the FDS. It tries to find nay anomaly in the transaction based on the spending profile of the cardholder, shipping address, shipping address, and billing.

Credit card fraud detection has drawn a lot of research interest and a number of techniques, with special emphasis on data mining and neural networks, have been suggested. Ghosh and Reilly have proposed credit card fraud detection with a neural network. They have built a detection system, which is trained on a large sample of labelled credit card account transactions.

Based on this observation, they use fraud density of real transaction data as a confidence value and generate the weighted fraud score to reduce the number of misdetections. Fan et al. suggest

the application of distributed data mining in credit card fraud detection. Brause et al. have developed an approach that involves advanced data mining techniques and neural network algorithms to obtain high fraud coverage. Chiu and Tsai have proposed Web services and data mining techniques to establish a collaborative scheme for fraud detection in the banking industry. With this scheme, participating banks share knowledge about the fraud patterns in a heterogeneous and distributed environment. To establish a smooth channel of data exchange,.

Credit card fraud detection has drawn a lot of research interest and a number of techniques, with special emphasis on data mining and neural networks, have been suggested. Ghosh and Reilly have proposed credit card fraud detection with a neural network. They have built a detection system, which is trained on a large sample of labelled credit card account transactions.
Based on this observation, they use fraud density of real transaction data as a confidence value and generate the weighted fraud score to reduce the number of misdetections. Fan et al. suggest the application of distributed data mining in credit card fraud detection.

Brause et al. have developed an approach that involves advanced data mining techniques and neural network algorithms to obtain high fraud coverage. Chiu and Tsai have proposed Web services and data mining techniques to establish a collaborative scheme for fraud detection in the banking industry. With this scheme, participating banks share knowledge about the fraud patterns in a heterogeneous and distributed environment. To establish a smooth channel of data exchange,.

An FDS runs at a credit card issuing bank. Each incoming transaction is submitted to the FDS for verification. FDS receives the card details and the values if purchases to verify, whether the transaction is genuine or not. The types of goods that are bought in that transaction are not known to the FDS. It tries to find nay anomaly in the transaction based on the spending profile of the cardholder, shipping address, shipping address, and billing.

Credit card fraud detection has drawn a lot of research interest and a number of techniques, with special emphasis on data mining and neural networks, have been suggested. Ghosh and Reilly have proposed credit card fraud detection with a neural network. They have built a detection system, which is trained on a large sample of labelled credit card account transactions.

3.2.1ADVANTAGES OF PROPOSED SYSTEM:

frauds in credit card transactions.

It is based on artificial intelligence and combines inductive learning algorithms and Metalearning methods for achieving higher accuracy. Phua et al. suggest the use of metaclassifier similar to in fraud detection problems. They consider naive Bayesian C4.5, and Back Propagation neural networks as the base classifiers. A metaclassifier is used to determine which classifier should be considered based on skewness of data. Although they do not directly use credit card fraud detection as the target application, their approach is quite generic.

Vatsa et al. have recently proposed a game-theoretic approach to credit card fraud detection. They model the interaction between an attacker and an FDS as a multistage game between two players, each trying to maximize his payoff. The problem with most of the abovementioned approaches is that they require labelled data for both genuine, as well as fraudulent transactions, to train the classifiers. Getting real-world fraud data is one of the biggest problems associated with credit card fraud detection.

Also, these approaches cannot detect new kinds of frauds for which labelled data is not available. In contrast, we present a Hidden Markov Model (AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING)-based credit card FDS, which does not require fraud signatures and yet is able to detect frauds by considering a cardholder's spending habit. We model a credit card transaction processing sequence by the stochastic process of an AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING. The details of items purchased in individual transactions are usually not known to an FDS running at the bank that issues credit cards to the cardholders.

This can be represented as the underlying finite Markov chain, which is not observable. The transactions can only be observed through the other stochastic process that produces the sequence of the amount of money spent in each transaction. Hence, we feel that AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING is an ideal choice for addressing this problem. Another important advantage of the AUTO ENCODER, LOCAL

CHAPTER 4
SYSTEM ANALYSIS

The purpose of system requirement specification is to produce the specification analysis of the task and also to establish complete information about the requirement, behavior and other constraints such as functional performance and so on. The goal of system requirement specification is to completely specify the technical requirements for the product in a concise and unambiguous manner.

4.1 Hardware Requirements:

Processor : Pentium 4

RAM : 4GB or more

Hard disk : 16 GB or more

4.2 Software Requirements:

The JDK is a development environment for building application, applets, and components using the Python programming language. The JDK include tools useful for developing and testing programs written in the Python programming language and running on the Python platform.

Contents of JDK :-

- Development Tools
- Runtime Environment
- Additional Libraries

4.2 CASE STUDY AND DATA

Objective :-

Software project management is an activity of organising, planning and scheduling the software projects. The goal of software project management is to deliver the software product in given time and within the budget. It is also necessary that the software project should project should be developed in accordance with the requirements of the organisation. The software project managers are responsible for planning and scheduling the software projects. Managing software projects is difficult because of Following reasons:-

4. Other engineering products are visible. But the software is intangible. It is not a physical entity that can be touched. Hence software project managers can see the progress of software project with the help of documents only.
5. The large projects are different in nature. Hence even the managers who have adequate experiences of previous projects may find difficulties in the current project.
6. There are no standard software process. It changes from one organisation to another.

Because of these difficulties the most of the software projects could not meet the budgets or fail to deliver on time.

Scope :-

- The detection of the fraud use of the card is found much faster than existing system.
- No need check the original user as we maintain a log .
- The log which is maintained will also be a proof for the bank for the transaction made.
- We can find the most accurate detection using this technique.
- This reduce the tedious work of an employee in bank.\

3.3 Cost Estimation :-

Software cost estimation is an important factor in software project. Large cost estimation errors can affect profit and cost of software project. If the project cost exceeds then it will seriously affect the project. But software cost and effort estimation will never be exact. The cost and effort estimation can be done in following

ways :-

- Delay the estimation as far as possible. This is not a feasible way of estimation. Because then we have to wait for long period to get project estimation.
- Use the estimates of earlier similar projects. If the current project is similar to earlier project then past efforts business conditions and deadlines are equivalent. But sometimes past efforts are not good indicators.
- Some decomposition techniques can be used to generate project cost and effort estimations.

The decomposition techniques are nothing but the “divide and conquer” techniques used for project estimation. That means decompose the project into major functions. Perform cost and effort estimations for these functionalities and related software activities.Thus for each major function, software cost and effort estimation can be obtained in step by step manner.

- Use of some empirical model for project cost eatimation.

The empirical cost estimation model is used for determining the cost of the project based on observations and experiences. The estimation can be made in the form of

$$E = f(Vi)$$

Where E is effort or cost or duration and f(V i) is the function for V i project estimation in terms of LOC or function point(fp).

There is another way of estimating a cost and that is use of automated estimation tools.

There are some systems available in which automated tools along with Graphical User Interface(GUI) are used to determine the cost of the project.

COCOMO Model:-

COCOMO is one of the most widely used software estimation models in the world. This model is developed in 1981 by Barry Boehm to give an estimate of the number of man-months it will take to develop a software product. COCOMO predicts the efforts and schedule of a software product based on size of the software. COCOMO stands for "Constructive Cost Model".

COCOMO has three different models that reflect the complexity

- Basic model
- Intermediate model
- Detailed model

Similarly there are three classes of software projects.

1) **Organic mode :-** In this mode, relatively small, simple software projects with a small team are handled. Such a team should have good application experiences to less rigid requirements.

2) **Semi-detached Projects :-** In this class an intermidiate projects in which teams with mixed experience level are handled. Such projects may have mix of rigid and less than rigid requirements.

3) **Embedded Projects :-** In this class, projects with tight hardware, software and operational constraints are handled.

Let us understand each model in detail.

1) Basic Model :- The basic COCOMO model estimates the software development effort using only Lines Of Code(LOC). Various equations in this model are –

$$E = ab\ (KLOC)^{bb}$$

$$D = C_b(E)d_b$$

$$P = E/D$$

Where E is the effort applied in person- months.

D is the development time in chronological months.

KLOC means kilo line of code for the project.

P is the total number of persons required to accomplish the project.

The coefficient a_b, b_b, c_b, d_b for three modes are as given below.

Software Projects	a_b	b_b	C_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi- detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table :- The coefficient for three modes

Merits of Basic COCOMO Model :-

1. Basic COCOMO model is good for quick, early, rough order of magnitude estimate of software project.

Limitations of Basic Model :-

The accuracy of this model is limited because it does not consider certain factors for cost estimation of software. These factors are hardware constraints, personal quality, and experience, modern techniques and tools.

CHAPTER 5
RESULT ANALYSIS

Process models are proposed in order to adopt the systematic approach in the software development. These models define the distinct set of activities, tasks and work products that are required to create high quality software. These process models are also called as prescriptive process model, generic software process models or software paradigms.

The process model can be defined as an abstract representation of process. The process model is chosen based on nature of software project.

5.1 Waterfall Model :-

The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance.

The waterfall development model originates in the manufacturing and construction industries: highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.

The first known presentation describing use of similar phases in software engineering was held by Herbert D. Benington at Symposium on advanced programming methods for digital computers on 29 June 1956. This presentation was about the development of software for SAGE. In 1983 the paper was republished with a foreword by Benington pointing out that the process was not in fact performed in strict top-down, but depended on a prototype.

The first formal description of the waterfall model is often cited as a 1970 article by Winston W. Royce, although Royce did not use the term "waterfall" in this article. Royce presented this model as an example of a flawed, non-working model. This, in fact, is how the

term is generally used in writing about software development—to describe a critical view of a commonly used software development practice.

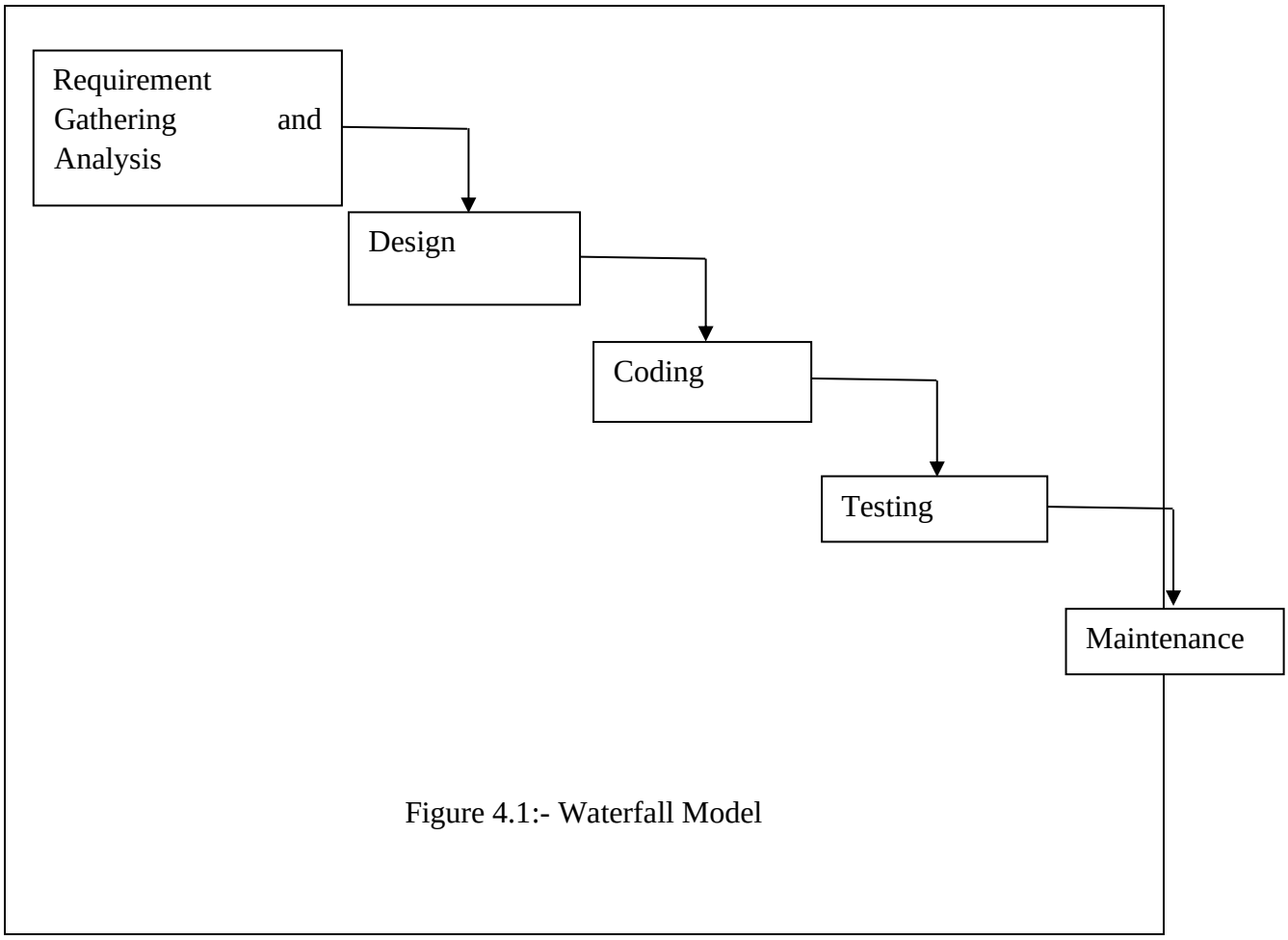


Figure 4.1:- Waterfall Model

5.2 RAD Model :-

Rapid application development (RAD) is a software development methodology that uses minimal planning in favour of rapid prototyping. The "planning" of software developed using RAD is interleaved with writing the software itself. The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements.

Rapid application development is a software development methodology that involves methods like iterative development and software prototyping. According to Whitten (2004), it is a merger of various structured techniques, especially data-driven Information Engineering, with prototyping techniques to accelerate software systems development.

In rapid application development, structured techniques and prototyping are especially used to define users' requirements and to design the final system. The development process starts with the development of preliminary data models and business process models using structured techniques. In the next stage, requirements are verified using prototyping, eventually to refine the data and process models. These stages are repeated iteratively; further development results in "a combined business requirements and technical design statement to be used for constructing new systems".

RAD approaches may entail compromises in functionality and performance in exchange for enabling faster development and facilitating application maintenance.

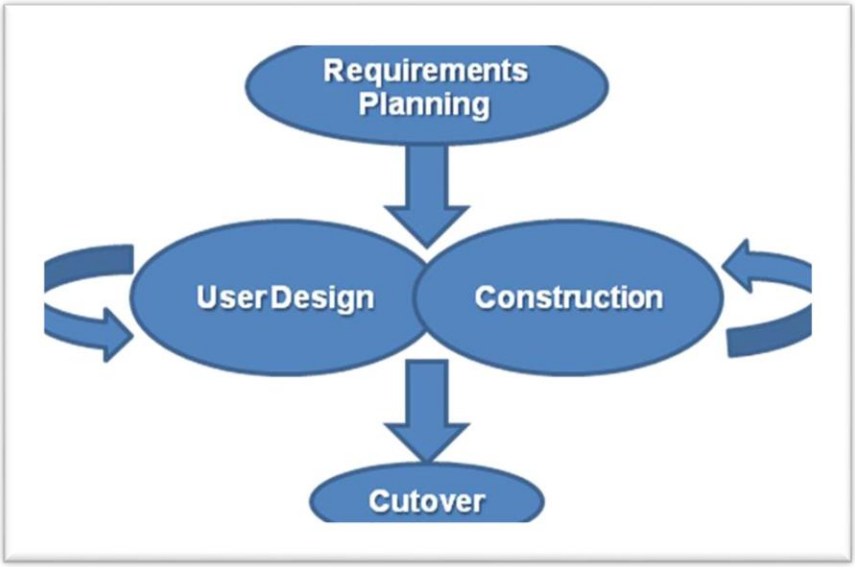


Figure 4.2 :- RAD Model In software Engineering

5.3 Spiral Model :-

The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. Also known as the spiral lifecycle model (or spiral development), it is a systems development method (SDM) used in information technology (IT). This model of

development combines the features of the prototyping and the waterfall model. The spiral model is intended for large, expensive and complicated projects.

The spiral model is divided into a number of framework activities. These framework activities are denoted by task regions.

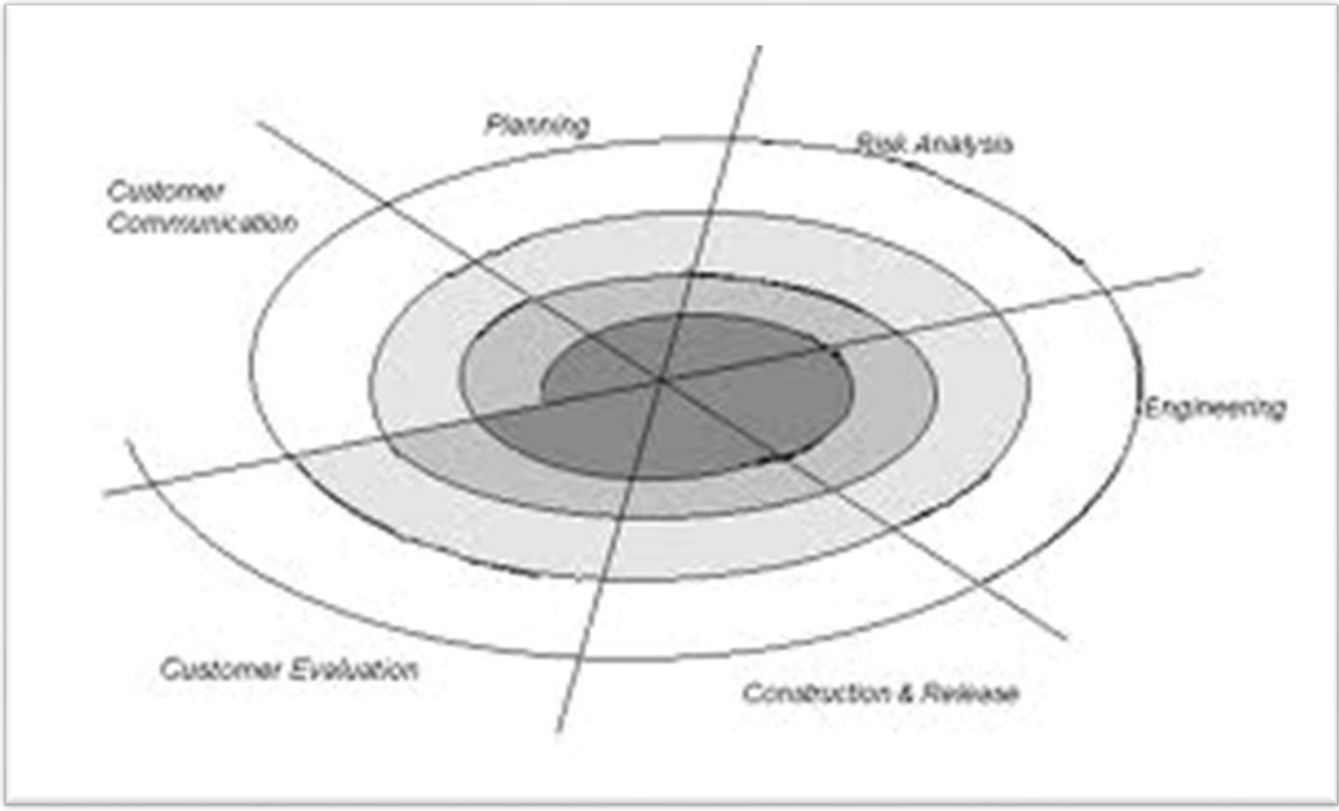


Figure 4.3:- Spiral Model in software Engineering

5.5 Verification and Validation Model :-

The V-model represents a software development process (also applicable to hardware development) which may be considered an extension of the waterfall model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes

represents time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.

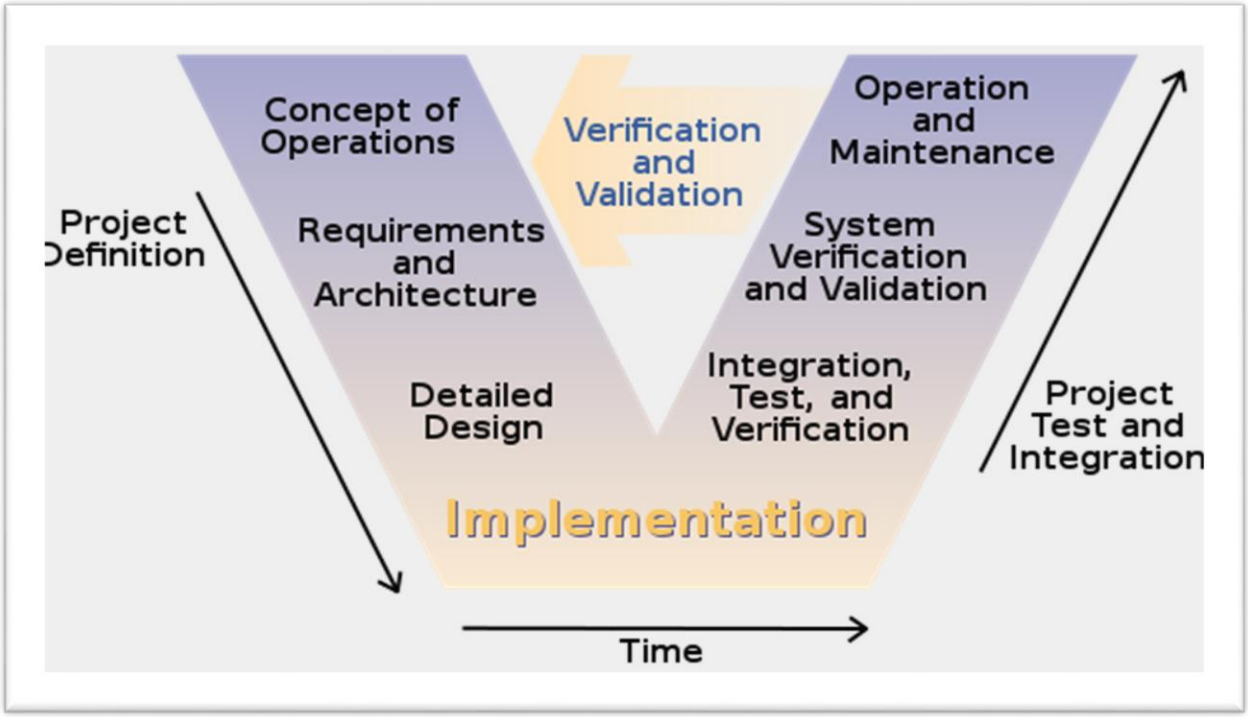


Figure 4.5 :- Verification and Validation Model in software Engineering

5.6 Prototyping Model :-

In prototyping model initially the requirement gathering is done. Developer and customer define overall objectives; identify areas needing more requirement gathering. Then a quick design is prepared. This design represents what will be visible to user-in input and output format. From the quick design a prototype is prepared. Customer or user evaluates the prototype in order to refine the requirements. Iteratively prototype is tuned for satisfying customer requirements. Thus prototype is important to identify the software requirements. When working prototype is built, developer use existing program fragments or program generates to throw away the prototype and rebuilt the system to high quality.

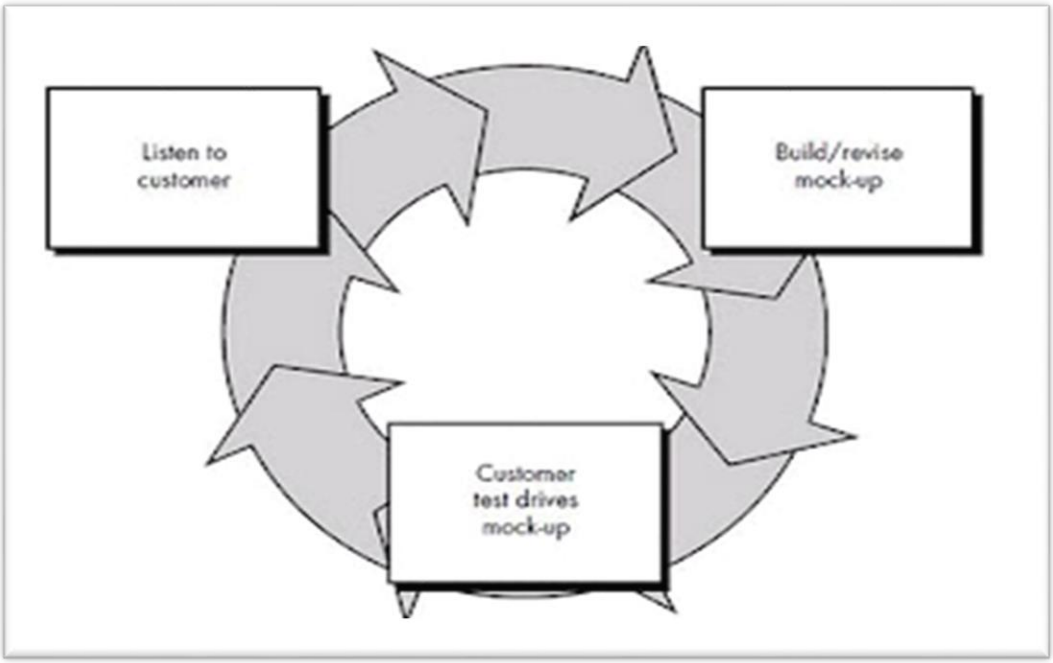


Figure 4.6 :- Incremental model in software engineering

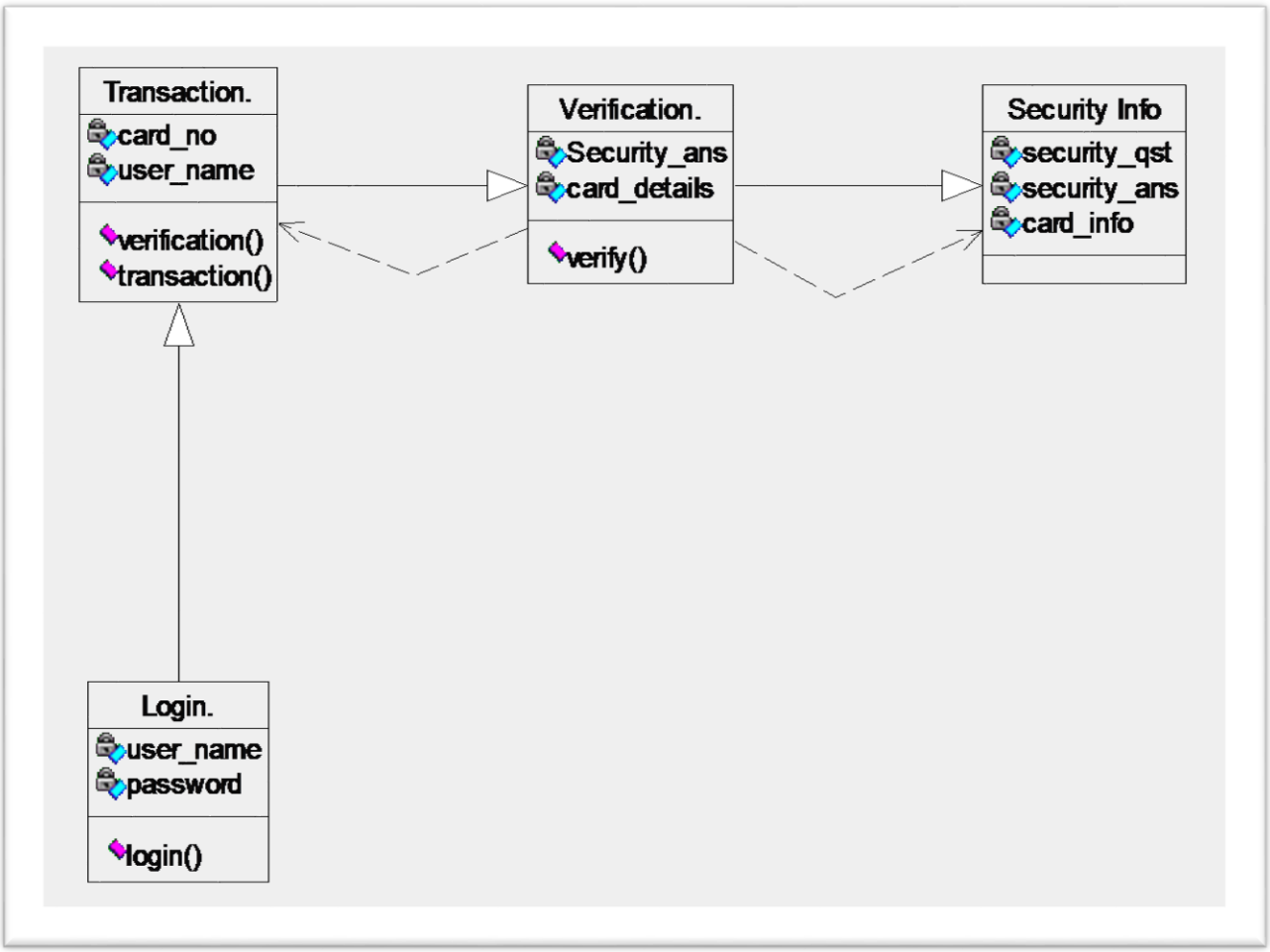


Figure 8.4 :- Class Diagram

CHAPTER 6
SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE:

System architecture is a conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

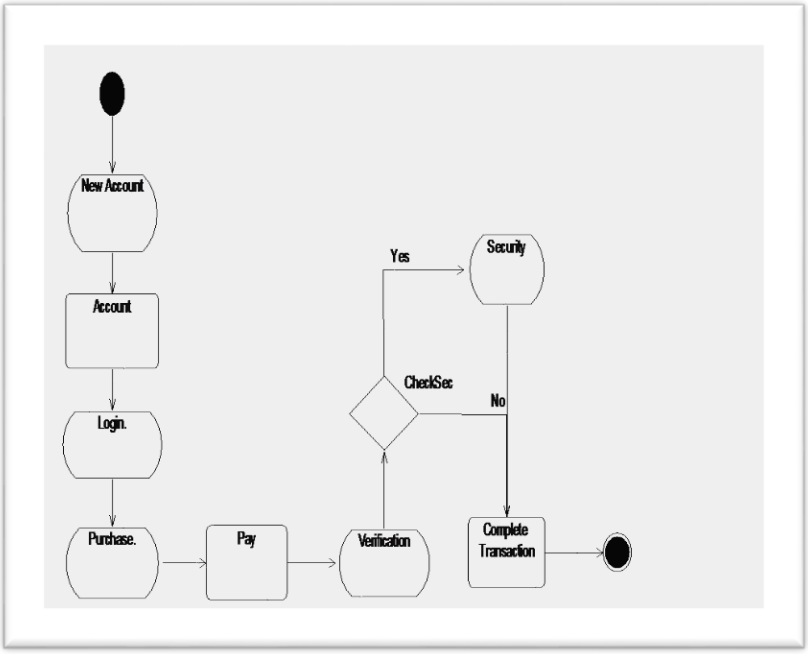
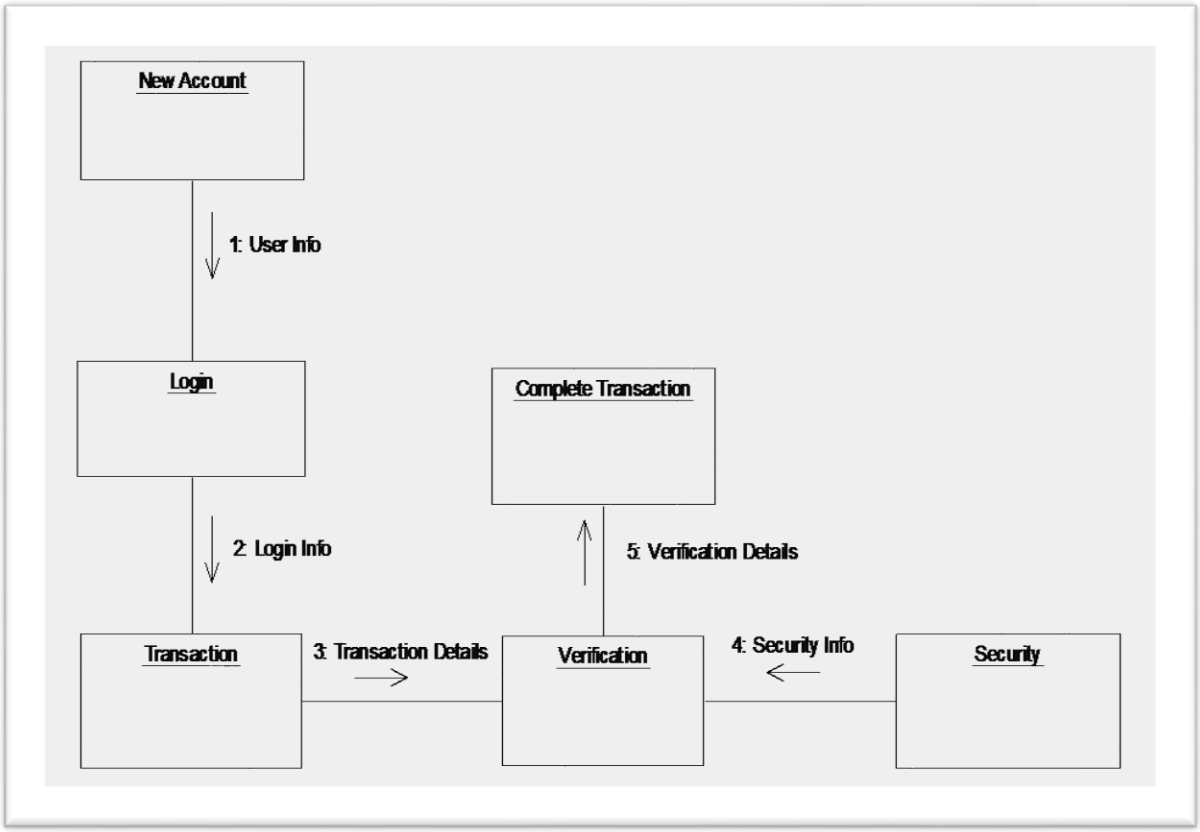


Fig:–8.7 Collaboration Diagram



CHAPTER 7

CODING

7.1 :-CODING FOR CUSTOMER REGISTRATION MODULE

```
import Python.sql.Connection;

import Python.sql.DriverManager;

import Python.sql.Statement;

import Python.util.Random;

import Pythonx.swing.JOptionPane;

public class Customer_Details extends Pythonx.swing.JFrame

{

   public Customer_Details()

{

      initComponents();

   }

   GEN-BEGIN:initComponents

   private void initComponents() {

         bfr=new StringBuffer();

      jPanel1 = new Pythonx.swing.JPanel();

      jPanel2 = new Pythonx.swing.JPanel();

      jLabel2 = new Pythonx.swing.JLabel();
```

```
jLabel15 = new Pythonx.swing.JLabel();

First = new Pythonx.swing.JTextField();

jLabel4 = new Pythonx.swing.JLabel();

Bank = new Pythonx.swing.JComboBox();

jLabel5 = new Pythonx.swing.JLabel();

Card = new Pythonx.swing.JComboBox();

jLabel7 = new Pythonx.swing.JLabel();

Account = new Pythonx.swing.JTextField();

jLabel9 = new Pythonx.swing.JLabel();

Dob = new Pythonx.swing.JTextField();

jLabel11 = new Pythonx.swing.JLabel();

Questions = new Pythonx.swing.JComboBox();

Last = new Pythonx.swing.JTextField();

jLabel3 = new Pythonx.swing.JLabel();

jLabel6 = new Pythonx.swing.JLabel();

Salary = new Pythonx.swing.JTextField();

Range = new Pythonx.swing.JLabel();

jLabel8 = new Pythonx.swing.JLabel();

jLabel10 = new Pythonx.swing.JLabel();

Email = new Pythonx.swing.JTextField();
```

```
        jLabel12 = new Pythonx.swing.JLabel();

        Answer = new Pythonx.swing.JTextField();

        Submit = new Pythonx.swing.JButton();

        Next = new Pythonx.swing.JButton();

        jLabel13 = new Pythonx.swing.JLabel();

        jLabel14 = new Pythonx.swing.JLabel();

        Number = new Pythonx.swing.JLabel();

        Password = new Pythonx.swing.JLabel();

        jLabel1 = new Pythonx.swing.JLabel();

 setDefaultCloseOperation(Pythonx.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setBackground(new Python.awt.Color(255, 102, 102));

        jPanel1.setLayout(null);

        jPanel2.setBackground(new Python.awt.Color(255, 153, 153));

        jLabel2.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));

        jLabel2.setText("First Name:");

        jLabel4.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));

        jLabel4.setText("Bank Name:");

        Bank.setFont(new Python.awt.Font("Palatino Linotype", 1, 14));

        Bank.setModel(new Pythonx.swing.DefaultComboBoxModel(new String[] {"SELECT",
"SBI", "BOM", "INDIAN", "AXIS" }));

        Bank.addActionListener(new Python.awt.event.ActionListener() {
```

```
        public void actionPerformed(Python.awt.event.ActionEvent evt) {
            CardActionPerformed(evt);
        }
    });
    jLabel5.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    jLabel5.setText("Type of cards:");
    Card.setFont(new Python.awt.Font("Palatino Linotype", 1, 14));
    Card.setModel(new Pythonx.swing.DefaultComboBoxModel(new String[] {"SELECT",
"PLATINUM", "MASTER CARD", "VISA", "MAESTRO" }));
    Card.addActionListener(new Python.awt.event.ActionListener() {
        public void actionPerformed(Python.awt.event.ActionEvent evt) {
            CardActionPerformed(evt);
        }
    });
    jLabel7.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    jLabel7.setText("A/C No:");
    jLabel9.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    jLabel9.setText("Dob:(d/m/yy)");
    jLabel11.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    jLabel11.setText("Questions:");
    Questions.setFont(new Python.awt.Font("Palatino Linotype", 1, 14));
```

```
    Questions.setModel(new Pythonx.swing.DefaultComboBoxModel(new String[] { "Favourite Film?", "Academic Concern?", "Favourite City?" }));
    Questions.addActionListener(new Python.awt.event.ActionListener()
    {
       public void actionPerformed(Python.awt.event.ActionEvent evt)
       {
          QuestionsActionPerformed(evt);
       }
    });
    jLabel3.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    jLabel3.setText("Last Name:");
    jLabel6.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    jLabel6.setText("Salary:");
    jLabel8.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    jLabel8.setText("Your Range:");
    jLabel10.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    jLabel10.setText("E_mail:");
    jLabel12.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    jLabel12.setText("Answers:");
    Submit.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
    Submit.setText("Submit");
```

```
Submit.addActionListener(new Python.awt.event.ActionListener()
{
   public void actionPerformed(Python.awt.event.ActionEvent evt)
   {
      SubmitActionPerformed(evt);
   }
});
Next.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));
Next.setText("Clear");
Next.addActionListener(new Python.awt.event.ActionListener()
{
   public void actionPerformed(Python.awt.event.ActionEvent evt)
   {
      NextActionPerformed(evt);
   }
});
First.addKeyListener(new Python.awt.event.KeyListener()
{
            public void keyReleased(Python.awt.event.KeyEvent ke)
            {
```

```
        }
        public void keyPressed(Python.awt.event.KeyEvent ke)
        {
                char ch=ke.getKeyChar();
                if(Character.isDigit(ch))
                {
                        JOptionPane.showMessageDialog(null,"Digit not allowed");
                        First.setText("");
                }
        }
        public void keyTyped(Python.awt.event.KeyEvent ke)
        {
               char ch=ke.getKeyChar();
                if(Character.isDigit(ch))
                {
                        JOptionPane.showMessageDialog(null,"Digit not allowed");
                        First.setText("");
                }
        }
});
```

```
   Last.addKeyListener(new Python.awt.event.KeyListener()
{
        public void keyReleased(Python.awt.event.KeyEvent ke)
        {
        }
        public void keyPressed(Python.awt.event.KeyEvent ke)
        {
                char ch=ke.getKeyChar();
                if(Character.isDigit(ch))
                {
                        JOptionPane.showMessageDialog(null,"Digit not allowed");
                        Last.setText("");
                }
        }
        public void keyTyped(Python.awt.event.KeyEvent ke)
        {
                char ch=ke.getKeyChar();
                if(Character.isDigit(ch))
                {
        JOptionPane.showMessageDialog(null,"Digit not allowed");
```

```
                    First.setText("");

                }

            }

        });

    jLabel13.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));

    jLabel13.setText("Card Number");

    jLabel14.setFont(new Python.awt.Font("Palatino Linotype", 1, 18));

    jLabel14.setText("Password");

    Pythonx.swing.GroupLayout jPanel2Layout = new Pythonx.swing.GroupLayout(jPanel2);

    jPanel2.setLayout(jPanel2Layout);

    jPanel2Layout.setHorizontalGroup(
jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)

        .addGroup(jPanel2Layout.createSequentialGroup()

            .addGap(185, 185, 185)

.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)

                .addGroup(jPanel2Layout.createSequentialGroup()

                    .addComponent(jLabel5, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 129, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

  .addPreferredGap(Pythonx.swing.LayoutStyle.ComponentPlacement.RELATED)

                    .addComponent(Card, 0, 174, Short.MAX_VALUE))
```

.addGroup(jPanel2Layout.createSequentialGroup()
.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel2, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 129, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jLabel4, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 129, Pythonx.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(Pythonx.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.TRAILING)

.addComponent(Bank, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 139, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(First, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 139, Pythonx.swing.GroupLayout.PREFERRED_SIZE)))

.addGroup(jPanel2Layout.createSequentialGroup()
.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jLabel9, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 129, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jLabel11, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 129, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jLabel7, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 129, Pythonx.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(Pythonx.swing.LayoutStyle.ComponentPlacement.RELATED)

```
.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.TRA
ILING)

                .addComponent(Dob,    Pythonx.swing.GroupLayout.Alignment.LEADING,
Pythonx.swing.GroupLayout.DEFAULT_SIZE, 174, Short.MAX_VALUE)

                .addComponent(Questions,
Pythonx.swing.GroupLayout.Alignment.LEADING, 0, 174, Short.MAX_VALUE)

                .addComponent(Account,    Pythonx.swing.GroupLayout.DEFAULT_SIZE,
174, Short.MAX_VALUE)))

            .addGroup(jPanel2Layout.createSequentialGroup()


.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEA
DING)

                .addComponent(jLabel14,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,                    140,
Pythonx.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jLabel13))
.addPreferredGap(Pythonx.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEA
DING, false)

                .addComponent(Number)

                .addComponent(Password,   Pythonx.swing.GroupLayout.DEFAULT_SIZE,
163, Short.MAX_VALUE))


.addPreferredGap(Pythonx.swing.LayoutStyle.ComponentPlacement.RELATED,
Pythonx.swing.GroupLayout.DEFAULT_SIZE,                Short.MAX_VALUE)))
```

.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(77, 77, 77)
.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(133, 133, 133)

.addComponent(Email, Pythonx.swing.GroupLayout.DEFAULT_SIZE, 143, Short.MAX_VALUE))

.addComponent(jLabel10, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 129, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jLabel6, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 129, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(Pythonx.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(Salary, Pythonx.swing.GroupLayout.DEFAULT_SIZE, 143, Short.MAX_VALUE))

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jLabel3, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 129, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

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

                .addComponent(Last,    Pythonx.swing.GroupLayout.DEFAULT_SIZE,
143, Short.MAX_VALUE))

            .addGroup(jPanel2Layout.createSequentialGroup()

                .addComponent(jLabel8,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,                    129,
Pythonx.swing.GroupLayout.PREFERRED_SIZE)


.addPreferredGap(Pythonx.swing.LayoutStyle.ComponentPlacement.RELATED)

                .addComponent(Range,   Pythonx.swing.GroupLayout.DEFAULT_SIZE,
143, Short.MAX_VALUE))

            .addGroup(Pythonx.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()

                .addGap(5, 5, 5)

                .addComponent(jLabel12,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,                    129,
Pythonx.swing.GroupLayout.PREFERRED_SIZE)


.addPreferredGap(Pythonx.swing.LayoutStyle.ComponentPlacement.RELATED)

                .addComponent(Answer,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,                    138,
Pythonx.swing.GroupLayout.PREFERRED_SIZE)))

            .addGap(122, 122, 122))

        .addGroup(jPanel2Layout.createSequentialGroup()
```

```
                .addGap(56, 56, 56)
                .addComponent(Submit, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 134, Pythonx.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(55, 55, 55)
                .addComponent(Next, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 89, Pythonx.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())))
        );
        jPanel2Layout.setVerticalGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(39, 39, 39)
                .addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(First, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(Last, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE))
```

.addGap(26, 26, 26)

.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel4, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(Bank, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(Salary, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jLabel6, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE))

.addGap(35, 35, 35)

.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel5, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(Card, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jLabel8, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(Range, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(32, 32, 32)

.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel10, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jLabel7, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(Account, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE)))

.addComponent(Email, Pythonx.swing.GroupLayout.Alignment.TRAILING, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE))

.addGap(30, 30, 30)

```
.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.TRAILING)
                .addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(Answer, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(Dob, Pythonx.swing.GroupLayout.PREFERRED_SIZE, Pythonx.swing.GroupLayout.DEFAULT_SIZE, Pythonx.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel9, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(31, 31, 31))
                .addGroup(jPanel2Layout.createSequentialGroup()
                    .addComponent(jLabel12, Pythonx.swing.GroupLayout.PREFERRED_SIZE, 28, Pythonx.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(18, 18, 18)))

.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel2Layout.createSequentialGroup()
```

```
.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.BAS
ELINE)

                .addComponent(Questions,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,
Pythonx.swing.GroupLayout.DEFAULT_SIZE,
Pythonx.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jLabel11,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,                                    28,
Pythonx.swing.GroupLayout.PREFERRED_SIZE))

            .addGap(45, 45, 45)


.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.BAS
ELINE)

                .addComponent(Number,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,                                    30,
Pythonx.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jLabel13,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,                                    28,
Pythonx.swing.GroupLayout.PREFERRED_SIZE))

            .addGap(30, 30, 30)


.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.BAS
ELINE)

                .addComponent(jLabel14,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,                                    27,
Pythonx.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(Password,
Pythonx.swing.GroupLayout.PREFERRED_SIZE,
Pythonx.swing.GroupLayout.DEFAULT_SIZE,
Pythonx.swing.GroupLayout.PREFERRED_SIZE))

                .addContainerGap(27, Short.MAX_VALUE))

            .addGroup(Pythonx.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()


.addGroup(jPanel2Layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.BAS
ELINE)

                    .addComponent(Submit,  Pythonx.swing.GroupLayout.PREFERRED_SIZE,
33, Pythonx.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(Next,     Pythonx.swing.GroupLayout.PREFERRED_SIZE,
33, Pythonx.swing.GroupLayout.PREFERRED_SIZE))

                .addGap(47, 47, 47))))

      );

     jPanel1.add(jPanel2);

      jPanel2.setBounds(-60, 40, 960, 520);

      jLabel1.setFont(new Python.awt.Font("Arial Black", 1, 24)); // NOI18N

      jLabel1.setForeground(new Python.awt.Color(51, 0,255));

      jLabel1.setText("User_Details");

      jPanel1.add(jLabel1);

      jLabel1.setBounds(310, 0, 260, 35);
```

```
Pythonx.swing.GroupLayout layout = new Pythonx.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, Pythonx.swing.GroupLayout.DEFAULT_SIZE, 897, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(Pythonx.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, Pythonx.swing.GroupLayout.DEFAULT_SIZE, 614, Short.MAX_VALUE)
        );
           pack();
    }// </editor-fold>//GEN-END:initComponents


    private void CardActionPerformed(Python.awt.event.ActionEvent evt)
     {//GEN-FIRST:event_CardActionPerformed
        String d=(String) Card.getSelectedItem();
        if(d.equals("PLATINUM"))
        {
         Range.setText("20000");
```

```
}
else if(d.equals("MASTER CARD"))
{
   Range.setText("30000");
} else if(d.equals("VISA"))
{
   Range.setText("40000");
} else
if(d.equals("MAESTRO"))
{
   Range.setText("50000");
}
 Random r=new Random();
  int num=r.nextInt(1000000);
  System.out.println(num);
 String s = new Integer(num).toString();
Number.setText(s);
final int PASSWORD_LENGTH = 8;
StringBuffer sb = new StringBuffer();
for (int x = 0; x < PASSWORD_LENGTH; x++)
```

```
        {
          sb.append((char)((int)(Math.random()*26)+97));
        }
        System.out.println(sb.toString());
        Password.setText(sb.toString());
    }//GEN-LAST:event_CardActionPerformed

        private void SubmitActionPerformed(Python.awt.event.ActionEvent evt) {//GEN-FIRST:event_SubmitActionPerformed
    int b=Bank.getSelectedIndex();
    int ct=Card.getSelectedIndex();
    if((b!=0) && (ct!=0) && (!First.getText().equals("")) && (!Last.getText().equals("")) && (!Account.getText().equals("")) && (!Salary.getText().equals("")) && (!Range.getText().equals("")) && (!Dob.getText().equals("")) && (!Email.getText().equals("")) && (!Answer.getText().equals("")))
     {
        Connection conn=null;
            Statement s=null;
            int limit=Integer.parseInt(Range.getText());
            System.out.println("\n\nLimit is := "+limit);
            int lower=limit*20/100;
            int middle=limit*50/100;
            int upper=limit*100/100;
```

```
String ss="INSERT INTO Customer_Details("+"First_Name,"+"Last_Name,"+"Bank_Name,"+"Salary,"+"Card_Type,"
+"Range,"+"AC_No,"+"DOB,"+"E_MAIL,"+"QUESTIONS,"+"ANSWERS,"+"CARD_NU
MBER,"+"PASSWORD,"+"lower_range,"+"middle_range,"+"upper_range)"

        +"VALUES('"+First.getText()+"','"+Last.getText()+"','"+
Bank.getSelectedItem()+"','"+Salary.getText()+"','"+Card.getSelectedItem()+"',"+Integer.pars
eInt(Range.getText())+",'"+Account.getText()+"','"+Dob.getText()+"','"+Email.getText()+"','
"+Questions.getSelectedItem()+"','"+Answer.getText()+


','"+Number.getText()+"','"+Password.getText()+"',"+lower+","+middle+","+upper+")";

    try {

            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

            conn=DriverManager.getConnection("jdbc:odbc:Credit");

            // JOptionPane.showMessageDialog(null,"Connection Established");

       s=conn.createStatement();

          s.executeUpdate(ss);

            JOptionPane.showMessageDialog(null, "Data Successfully  Inserted");

            conn.close();

            this.dispose();

    }

    catch (Exception e)

    {

       System.out.println(e);
```

```
    JOptionPane.showMessageDialog(null,e);
}
     First.setText("");
Last.setText("");
Bank.setSelectedIndex(0);
Card.setSelectedIndex(0);
Dob.setText("");
Email.setText("");
Questions.setSelectedIndex(0);
Salary.setText("");
Password.setText("");
Range.setText("");
Number.setText("");
Answer.setText("");
Account.setText("");
}
else
{
                JOptionPane.showMessageDialog(null,"Fill complete Details ");
}
```

```
    }

    private void NextActionPerformed(Python.awt.event.ActionEvent evt)

{

        First.setText("");

        Last.setText("");

        Bank.setSelectedIndex(0);

        Card.setSelectedIndex(0);

        Dob.setText("");

        Email.setText("");

        Questions.setSelectedIndex(0);

        Salary.setText("");

        Password.setText("");

        Range.setText("");

        Number.setText("");

        Answer.setText("");

        Account.setText("");

    }

    private void QuestionsActionPerformed(Python.awt.event.ActionEvent evt) {//GEN-FIRST:event_QuestionsActionPerformed

        // TODO add your handling code here:

    }//GEN-LAST:event_QuestionsActionPerformed
```

```
    public static void main(String args[])
    {
        new Customer_Details().setVisible(true);
}
    GEN-BEGIN:variables
    private Pythonx.swing.JTextField Account;
    private Pythonx.swing.JTextField Answer;
    private Pythonx.swing.JComboBox Bank;
    private Pythonx.swing.JComboBox Card;
    private Pythonx.swing.JTextField Dob;
    private Pythonx.swing.JTextField Email;
    private Pythonx.swing.JTextField First;
    private Pythonx.swing.JTextField Last;
    private Pythonx.swing.JButton Next;
    private Pythonx.swing.JLabel Number;
    private Pythonx.swing.JLabel Password;
    private Pythonx.swing.JComboBox Questions;
    private Pythonx.swing.JLabel Range;
    private Pythonx.swing.JTextField Salary;
    private Pythonx.swing.JButton Submit;
```

```
private Pythonx.swing.JLabel jLabel10;
private Pythonx.swing.JLabel jLabel11;
private Pythonx.swing.JLabel jLabel12;
private Pythonx.swing.JLabel jLabel13;
private Pythonx.swing.JLabel jLabel14;
private Pythonx.swing.JLabel jLabel15;
private Pythonx.swing.JLabel jLabel2;
private Pythonx.swing.JLabel jLabel3;
private Pythonx.swing.JLabel jLabel4;
private Pythonx.swing.JLabel jLabel5;
private Pythonx.swing.JLabel jLabel6;
private Pythonx.swing.JLabel jLabel7;
private Pythonx.swing.JLabel jLabel8;
private Pythonx.swing.JLabel jLabel9;
private Pythonx.swing.JPanel jPanel1;
private Pythonx.swing.JPanel jPanel2;
 }
```

MAINTENANCE

Software maintenance in software engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes.

A common perception of maintenance is that it merely involves fixing defects. However, one study indicated that the majority, over 80%, of the maintenance effort is used for non-corrective actions (Pigosky 1997). This perception is perpetuated by users submitting problem reports that in reality are functionality enhancements to the system.

Software maintenance and evolution of systems was first addressed by Meir M. Lehman in 1969. Over a period of twenty years, his research led to the formulation of Lehman's Laws (Lehman 1997). Key findings of his research include that maintenance is really evolutionary development and that maintenance decisions are aided by understanding what happens to systems (and software) over time. Lehman demonstrated that systems continue to evolve over time. As they evolve, they grow more complex unless some action such as code refactoring is taken to reduce the complexity.

The key software maintenance issues are both managerial and technical. Key management issues are: alignment with customer priorities, staffing, which organization does maintenance, estimating costs. Key technical issues are: limited understanding, impact analysis, testing, and maintainability measurement. Four types are encountered during the support phase

Correction:

Even with the best quality assurance activities, it is likely that the customer will uncover defect in the software. Corrective maintenance changes the software to correct defects.

Adaptation:

Over time, the original environment (e.g. CPU operating system, business rules, external product characteristics) for which the software was developed is likely to change. Adaptive maintenance result in modification to the software to accommodate change to its external environment.

Enhancement:

As software is used, the customer/user will recognise additional function that will provide benefit. The perfective maintenance the extend the software beyond its original functional requirement.

Prevention:

Computer software deteriorates due to change, and because of this, preventive maintenance, often called software reengineering must be conducted to enable the software to serve the need of its end user. In essence, preventive maintenance makes changes to computer program so that they can be more easily corrected, adapted, and enhanced.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

In this project, we have proposed an application of AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING in credit card fraud detection. The different steps in credit card transaction processing are represented as the underlying stochastic process of an AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING. We have used the ranges of transaction amount as the observation symbols, whereas the types of item have been considered to be states of the AUTO ENCODER, LOCAL OUTLIER FACTOR, KMEANS CLUSTERING. We have suggested a method for finding the spending profile of cardholders, as well as application of this knowledge in deciding the value of observation.

CHAPTER 9
FUTURE ENHANCEMENT

FUTURE SCOPE :-

- Detailed reporting on suspicious activity
- Option to hold questionable transactions for review
- Ability to block transactions from historically fraudulent IPs
- Robust searching feature allows you to quickly locate transactions
- Choose from pre-drafted customer responses, or customize your own
- Receive instant notification of questionable activity directly to your inbox
- Customizable filters for easy identification of fraudulent charges, including but not limited to:
    - Set minimum and maximum amounts
    - Monitor amount of charges applied to a card per hour
    - Scan for address discrepancies
    - Locate abnormal activity from a single IP address, and block IPs known to be fraudulent

BENEFITS:-

- Decreased fees and other costs associated with credit card fraud
- Increased confidence in legitimacy of transactions
- Settings tailored to your business necessities
- Advanced technology to take your credit card security to the next level
- We based intuitive design is quick to learn and easy to use
- Seamlessly integrates with other E-Complish solutions using the online payments system.

CHAPTER 10

REFERENCES

[1] “Global Consumer Attitude Towards On-Line Shopping,”

http://www2.acnielsen.com/reports/documents/2005_cc_online shopping.pdf, Mar. 2007.

[2] D.J. Hand, G. Blunt, M.G. Kelly, and N.M. Adams, “Data Mining for Fun and Profit,” Statistical Science, vol. 15, no. 2, pp. 111-131, 2000.

[3] “Statistics for General and On-Line Card Fraud,” http://www. epaynews.com/statistics/fraud.html, Mar. 2007.

[4] S. Ghosh and D.L. Reilly, “Credit Card Fraud Detection with a Neural-Network,” Proc. 27th Hawaii Int’l Conf. System Sciences: Information Systems: Decision Support and Knowledge-Based Systems, vol. 3, pp. 621-630, 1994.

[5] M. Syeda, Y.Q. Zhang, and Y. Pan, “Parallel Granular Networks for Fast Credit Card Fraud Detection,” Proc. IEEE Int’l Conf. Fuzzy Systems, pp. 572-577, 2002.

[6] S.J. Stolfo, D.W. Fan, W. Lee, A.L. Prodromidis, and P.K. Chan, “Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results,” Proc. AAAI Workshop AI Methods in Fraud and Risk Management, pp. 83-90, 1997.

[7] S.J. Stolfo, D.W. Fan, W. Lee, A. Prodromidis, and P.K. Chan, “Cost-Based Modeling for Fraud and Intrusion Detection: Results from the JAM Project,” Proc. DARPA Information Survivability Conf. and Exposition, vol. 2, pp. 130-144, 2000.

[8] E. Aleskerov, B. Freisleben, and B. Rao, “CARDWATCH: A Neural Network Based Database Mining System for Credit Card Fraud Detection,” Proc. IEEE/IAFE: Computational Intelligence for Financial Eng., pp. 220-226, 1997.

[9] M.J. Kim and T.S. Kim, “A Neural Classifier with Fraud Density Map for Effective Credit Card Fraud Detection,” Proc. Int’l Conf. Intelligent Data Eng. and Automated Learning, pp. 378-383, 2002.

[10] C. Chiu and C. Tsai, “A Web Services-Based Collaborative Scheme for Credit Card Fraud Detection,” Proc. IEEE Int’l Conf. e-Technology, e-Commerce and e-Service, pp. 177-181, 2004.

[11] V. Vatsa, S. Sural, and A.K. Majumdar, “A Game-theoretic Approach to Credit Card Fraud Detection,” Proc. First Int’l Conf. Information Systems Security, pp. 263-276, 2005.

[12] S.S. Joshi and V.V. Phoha, “Investigating Hidden Markov Models Capabilities in Anomaly Detection,” Proc. 43rd ACM Ann. Southeast Regional Conf., vol. 1, pp. 98-103, 2005.

[13] S.B. Cho and H.J. Park, “Efficient Anomaly Detection by Modeling Privilege Flows Using Hidden Markov Model,” Computer and Security, vol. 22, no. 1, pp. 45-55, 2003.

[14] D. Ourston, S. Matzner, W. Stump, and B. Hopkins, “Applications of Hidden Markov Models to Detecting Multi-Stage Network Attacks,” Proc. 36th Ann. Hawaii Int’l Conf. System Sciences, vol. 9, pp. 334-344, 2003.

[15] T. Lane, “Hidden Markov Models for Human/Computer Interface Modeling,” Proc. Int’l Joint Conf. Aicial Intelligence, Workshop Learning about Users, pp. 35-44, 1999.