

1. INTRODUCTION

1.1 INTRODUCTION

The Bank Management System project is a software application designed to streamline the operations of a bank. The system provides an efficient platform for managing various banking activities such as account opening, customer management, transaction processing, loan management, and financial reporting. The system is designed to be user-friendly, secure, and scalable, and it can be customized to meet the specific needs of the bank.

The system provides an easy-to-use interface for customers to access their accounts, perform transactions, and view their transaction history. It also provides a comprehensive dashboard for bank administrators, clerk and managers to monitor the system's performance and track key performance indicators.

This system also provide a feature for a bank to predict whether their customers leaves their bank or not by using classification alogorithms.

Overall, the Bank Management System project is a valuable tool for banks looking to stay competitive in today's rapidly changing financial landscape

1.2 PROBLEM STATEMENT

The problem statement for the project is to address the challenge of customer churn in the banking industry. Customer churn refers to the phenomenon of customers leaving a bank and switching to competitors or completely discontinuing their banking services. This is a significant issue for banks as it results in lost revenue, decreased market share, and increased customer acquisition costs. The project aims to develop a predictive model that can accurately identify customers who are at risk of churning, enabling banks to take proactive measures to retain these customers. By understanding the factors that contribute to churn and implementing targeted retention strategies, banks can reduce customer attrition, improve customer satisfaction, and drive long-term profitability.

1.3 SCOPE AND RELEVANCE OF THE PROJECT

The system aims to address the challenge of customer churn in the banking industry. By leveraging data analysis techniques and utilizing the bank's customer database, the system can accurately predict customer churn and provide actionable insights for proactive retention strategies. This helps banks optimize their business outcomes by reducing customer attrition and improving customer satisfaction. By implementing targeted retention measures based on churn predictions, banks can strengthen customer relationships, enhance profitability, and remain competitive in the dynamic banking landscape. The system's practical application and cost-effectiveness make it a valuable tool for banks seeking sustainable growth and success in the industry.

1.4 OBJECTIVES

The objectives of customer churn prediction in a bank management system are:

- **Building a Predictive Customer Churn Model:** By using classification algorithms, the system aims to develop a model that can accurately predict whether a customer will leave the bank or not. This predictive capability allows the bank to proactively identify customers at risk of churn and take appropriate retention measures.
- **Incorporating Essential Bank Functionalities:** The bank management system is designed to encompass all essential bank-related tasks and functionalities. It provides customers with an easy-to-use interface to access their accounts, perform transactions, and view their transaction history. This ensures a seamless and convenient banking experience for customers.

2. SYSTEM ANALYSIS

2.1 INTRODUCTION

System analysis is the process of gathering and interpreting data and facts diagnosing problems to the system. In the development of software, structure analysis is required. During the analysis, the information is collected in the form of answers to the question for collecting information from existing documents. The system study and analysis, deals with study of current system. This is the most critical process of information development. It can be defined as a problem solving technique which consists of four phases that can be successfully completed by the applying appropriate skill and carefully addressing each dimension of the information system. After the feasibility analysis the next step is the study of the current system. The purposes of this phase are to learn how the current system operates. The analyst identifies the problems, limitations and the constraints and forms preliminary solutions finally. The analyst updates the feasibility estimates and presents the findings as a problem statement for formal study of phase reports.

2.2 EXISTING SYSTEM

Infosys Finacle is a widely used banking solution that provides comprehensive banking functionality, including core banking, payments, treasury, lending, and wealth management. The existing system based on Infosys Finacle focuses on managing various banking operations, customer accounts, transactions, and financial processes. It offers features such as account opening, fund transfers, balance inquiries, loan management, and reporting. However, it may not have an integrated customer churn prediction feature.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

1. **Lack of Predictive Analytics:** The existing system may not have built-in predictive analytics capabilities to analyze customer data and identify patterns indicative of potential churn.
2. **Limited Data Integration:** The existing system may have constraints in integrating and consolidating data from multiple sources, such as transaction history, customer demographics, and interaction logs.
3. **Manual Analysis and Reporting:** Without automated tools for churn analysis and reporting, bank personnel may rely on manual processes to identify potential churn indicators. This approach is time-consuming and prone to human error, leading to delayed response times and inaccurate predictions.
4. **Lack of Real-time Insights:** The existing system may provide limited real-time insights into customer behavior and churn likelihood. Without timely access to updated customer data and predictive analytics, the bank may miss critical opportunities to intervene and retain at-risk customers.

5. **Training and Expertise Requirements:** Implementing a customer churn prediction system requires specialized knowledge in data analytics and machine learning. The existing system may not offer sufficient training resources or require additional expertise, posing challenges for banks to leverage advanced churn prediction techniques effectively.

2.3 PROPOSED SYSTEM

In the proposed system, we can enhance the existing Infosys Finacle system by integrating a customer churn prediction feature. This addition would allow the bank to predict and identify customers who are likely to churn or leave the bank in the near future.

This feature involves integrating additional data sources such as customer data into the system. Using machine learning algorithms, a churn prediction model is developed and integrated into Finacle. This model continuously analyzes customer data to identify those at high risk of churn in real-time. To retain these customers, personalized offers, improved service, and targeted marketing campaigns can be implemented. The system's performance is regularly monitored to assess the effectiveness of the churn prediction model and the retention strategies.

2.3.1 ADVANTAGES OF PROPOSED SYSTEM

The proposed system for customer churn prediction in the bank management system offers several advantages:

1. **Improved Customer Retention:** By accurately identifying customers at risk of churning, the system enables proactive retention efforts. The bank can take targeted actions, such as personalized offers and enhanced customer service, to retain valuable customers and minimize attrition.
2. **Enhanced Customer Insights:** The system utilizes advanced analytics and machine learning algorithms to analyze customer data comprehensively. This provides deeper insights into customer behavior, preferences, and patterns, helping the bank understand customer needs and tailor services accordingly.
3. **Real-time Churn Monitoring:** The system continuously monitors customer data in real-time, allowing for timely detection of churn indicators and immediate intervention. This proactive approach enables the bank to address customer concerns promptly and take preventive measures to minimize churn.
4. **Automated Prediction:** The system automates the process of churn prediction by utilizing machine learning algorithms. This eliminates the need for manual analysis and prediction, saving significant

time and effort for bank personnel.

2.4 FEASIBILITY STUDY

All projects are feasible when given unlimited resources and infinite time. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study will decide if the proposed system will be cost effective from the business point of view and if it can be developed in the given existing budgetary constraints. The feasibility study should be relatively cheap and quick. The result should inform the decision of whether to go ahead with a more detailed analysis. The feasibility study is done in these phases.

- Operational feasibility
- Technical feasibility
- Economic feasibility.

2.4.1 TECHNICAL FEASIBILITY

The technical feasibility study evaluates both hardware and software requirements for the project. It assesses whether the project can be executed using the current equipment and software technology available.

Furthermore, our system has specific software requirements, including Python, HTML, MySQL, and a minimum operating system of Windows 7 or higher. As these software components are readily available, the system meets the technical feasibility criteria. By ensuring compatibility with existing hardware and software resources and leveraging widely used frameworks and datasets, the proposed system can be developed and deployed effectively.

2.4.2 OPERATIONAL FEASIBILITY

Operational feasibility refers to the practicality and usability of the proposed system within the operational environment. In the case of your system, it is stated that the system will be operational and operative once it is developed and installed. Additionally, it is mentioned that the system can be easily operated by the users.

Based on this information, it can be inferred that the proposed system demonstrates operational feasibility. It suggests that the system is expected to be used effectively by the intended users and is compatible with the operational processes of the bank. The ease of operation further enhances the feasibility of the system,

indicating that users will be able to navigate and utilize the system efficiently.

It is important to note that operational feasibility should also consider factors such as user acceptance, training requirements, and the overall impact on operational efficiency. However, based on the provided information, the operational feasibility of the system appears to be favorable.

2.4.3 ECONOMIC FEASIBILITY

The economic feasibility assesses the financial viability and cost-effectiveness of implementing the proposed system. It involves analyzing the costs associated with developing, implementing, and maintaining the system, as well as the potential benefits and return on investment (ROI) that the system can deliver. Considerations include the costs of acquiring necessary hardware and software, training personnel, and the expected financial impact of reducing customer churn and improving customer retention.

2.5 SOFTWARE ENGINEERING PARADIGM APPLIED

Software engineering paradigm refers to a systematic approach or methodology used in software development projects. It provides a framework for managing the software development process, organizing tasks, and ensuring the successful delivery of a high-quality software product. There are various software engineering paradigms, such as Waterfall, Agile, Scrum, and Spiral, each with its own characteristics and benefits.

In the context of the "Customer Churn Prediction in Bank" project, the chosen software engineering paradigm is Agile. Agile is a widely used approach in software development known for its iterative and collaborative nature. It emphasizes customer involvement, adaptability, and continuous improvement.

By adopting Agile as the software engineering paradigm, the project aims to foster a responsive and collaborative development process. It enables the project team to deliver functional segments of the customer churn prediction in bank system at regular intervals, gather feedback, and incorporate improvements based on the changing requirements. This iterative and customer-focused approach ensures the successful implementation of the proposed system.

3. SYSTEM DESIGN

3.1 INTRODUCTION

System design is an interactive process through which requirement are transmitted to a “blue print” for constructing the software initial. the blue print depicts a holistic view of software that is design is represented at a high-level abstraction a level that can be directly traced to specific data, functional and behavioural requirement. As design interaction occur subsequent refinement leads to design representation at much lower level of abstraction. System design is a creative art of inventing and developing input, data bases, off line files, method and procedures, for processing data to get meaning full output that satisfy the organization objectives. Through the design phase consideration to the human factor, that is inputs to the users will have on the system. Some of the main factors that have to be noted using the design of the system are:

- **Practicability:** System must be capable of being operated over a long period of time and must have ease of use
- **Efficiency:** Make better use of available resources. Efficiency involves accuracy, timeliness and comprehensiveness of system output.
- **Cost:** Aim of minimum cost and better results.
- **Security:** Ensure physical security of data.

3.2 DATABASE DESIGN

Database design is one of the most important parts of the system design phase. In a database environment, data available are used by several users instead of each program managing its own data, authorized users share data across application with the database software managing the data as an entity. Primary key is one of the candidate key that are chosen to be the identifying key for the entire table. Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly deletion anomaly. The database design of this this system is in second normal form and every non key attribute is functionally depends only on the primary key. Here are the most commonly used normal forms:

- First normal form (1NF)
- Second normal form(2NF)
- Third normal form (3NF)
- Boyce Codd normal form (BCNF)

First Normal Form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

To make the values atomic the name of the user was further divided into first name and last name

Second Normal Form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table. An attribute that is not part of any candidate key is known as non-prime attribute.

Third Normal Form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.
- An attribute that is not part of any candidate key is known as non-prime attribute. In other words, 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency $X \rightarrow Y$ at least one of the following conditions hold:
 - X is a super-key of table
 - Y is a prime attribute of table
 - An attribute that is a part of one of the candidate keys is known as prime attribute

Boyce Codd Normal Form (BCNF)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency $X \rightarrow Y$, X should be the super key of the table.

3.2.1 ENTITY RELATIONSHIP MODEL

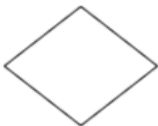
An entity relationship model (ER-model) is an abstract way to describe a database. It usually starts with a relational database, which stores data in tables. Some of the data in this table point to data in other tables for instance, your entry in the database could point to several entries for each of the phone number that is yours. The ER-model would say that you're an entity, and relationship between you and the phone numbers is 'has a phone number '. Diagrams created to design these entities and relationships are called entity relationship diagrams or ER diagrams. An entity maybe defined as a thing which is recognized as being

capable of being independent existence and which can be uniquely identified. An entity is an abstraction from complexities of a domain. When we speak of an entity, we normally speak of some aspects of the real world which can be distinguished from other aspects of the real world. An entity may be physical object such as a house or a car, an event such as a house sale or a car service, or a concepts such as customer transaction order.

Entity



Relationship



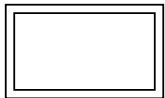
Attribute



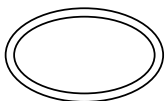
Line



Weak Entity



Multivalued Attribute



ER-DIAGRAM

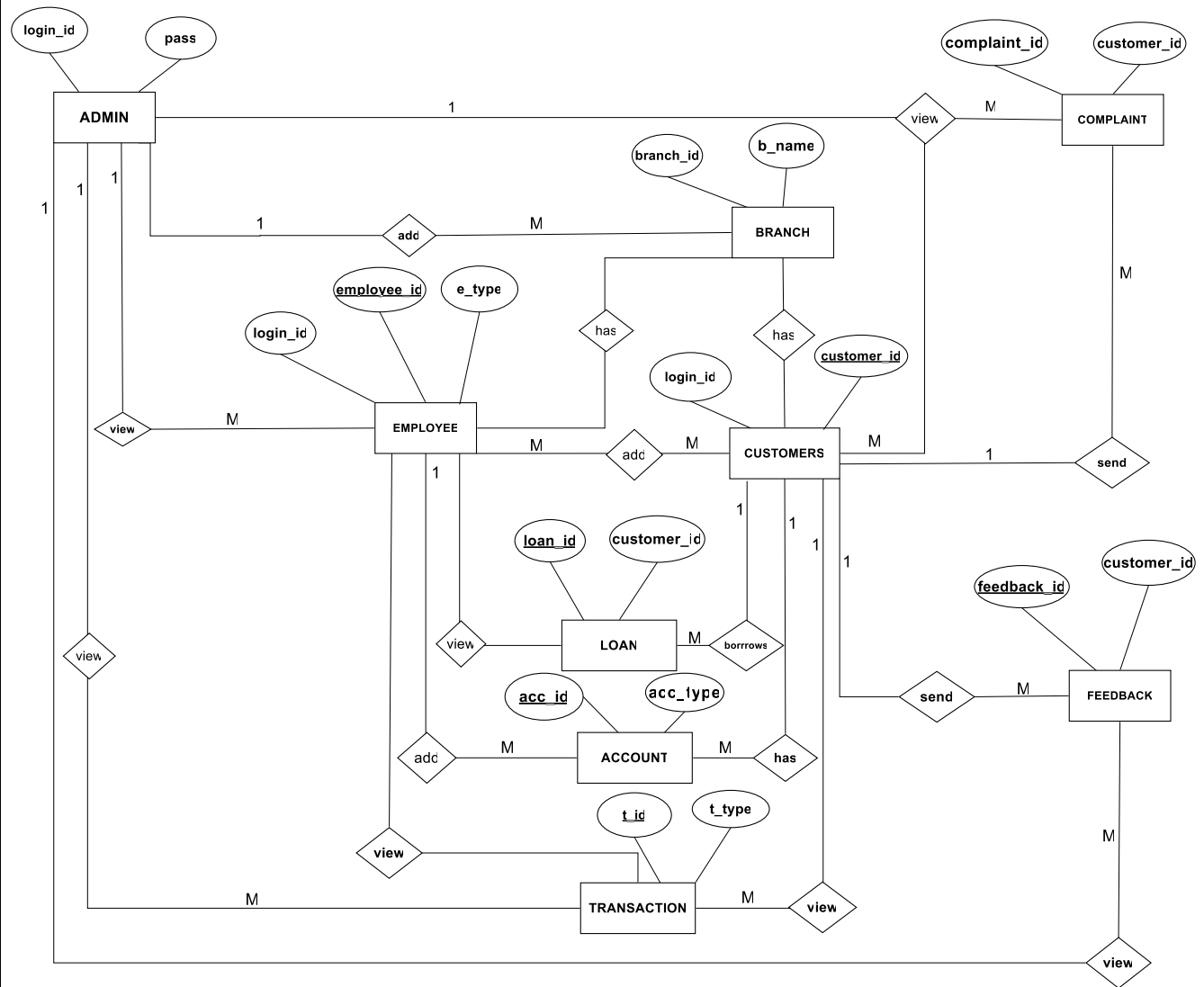


FIG 1: ER DIAGRAM

3.2.2 TABLE DESIGN

1. TABLE NAME: Login

DESCRIPTION: Login Details

PRIMARY KEY: login_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Login id	Integer (10)	Login id	Primary key
Username	Varchar (25)	Login username	Not null
Password	Varchar (25)	Login password	Not null
User type	Varchar (25)	Type of user	Not null

Table 1: Login

2. TABLE NAME: employee

DESCRIPTION: employee Details

PRIMARY KEY: employee_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
employee id	Integer (10)	Id of packager	Primary key
Login_id	Integer (10)	Login id	Foreign key
Branch_id	Integer (10)	Branch id	Foreign key
fname	Varchar (25)	First name	Not null
lname	Varchar (25)	Last name	Not null
dob	date	Date of birth	Not null
image	Varchar (25)	Image	Not null
m_status	Varchar (25)	Martial status	Not null
gender	Varchar (25)	gender	Not null
Employee_type	Varchar (25)	Employee type	Not null

address	Varchar (25)	Address of employee	Not null
zipcode	Varchar (25)	Zip code	Not null
Place	Varchar (25)	Place of employee	Not null
District	Varchar (25)	District of employee	Not null
Phone	Integer (10)	Phone number of employee	Not null
Email	Varchar (25)	Email id of employee	Not null
Dateofjoin	date	Employee join date	Not null
status	Varchar (25)	Status	Not null

Table 2: employee

3. TABLE NAME: Customer

DESCRIPTION: Customer Details

PRIMARY KEY: customer_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Customer id	Integer (10)	Id of customer	Primary key
Login_id	Integer (10)	Login id	Foreign key
Branch_id	Integer (10)	Branch id	Foreign key
First name	Varchar (25)	First name of customer	Not null
Last name	Varchar (25)	Last name of customer	Not null
dob	Date	Date of join	Not null
photo	Varchar (25)	Photo of customer	Not null
M_status	Varchar (25)	Martial status of customer	Not null
gender	Varchar (25)	gender	Not null
phone	Varchar (25)	Phone number	Not null
email	Varchar (25)	email	Not null
address	Varchar (25)	Address	Not null
city	Varchar (25)	city	Not null

state	Varchar (25)	State	Not null
ziocode	Varchar (25)	Zip code	Not null
Country	Varchar (25)	Country	Not null
Education	Varchar (25)	Education of customer	Not null
Monthly_salary	Varchar (25)	Monthly salary of customer	Not null
idproof	Varchar (25)	Id proof type	Not null
idno	Integer(11)	Id proof number	Not null
Proof_files	Varchar (25)	House name of customer	Not null
dateofjoin	date	Date of customer join in bank	Not null
status	Varchar (25)	status	Not null

Table 3: customer

4. TABLE NAME: branch

DESCRIPTION: branch Details

PRIMARY KEY: branch d

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Branch_id	Integer (10)	Id of category	Primary key
name	Varchar (25)	Name of branch	Not null
Branch_location	Varchar (25)	Location of branch	Not null
Ifsc_code	Varchar (25)	Ifsc code	Not null
phone	Integer(11)	Phone number	Not null

Table 4: branch

5. TABLE NAME: savingsacc

DESCRIPTION: savings account details

PRIMARY KEY: savings_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Savings_id	Integer (10)	Savings id	Primary key
Customer_id	Integer (10)	Customer id	Foreign key
Branch_id	Integer (10)	Branch id	Foreign key
Acc_no	Varchar (25)	Account number	Not null
ifsccode	Varchar (25)	Image of place	Not null
Balance	Integer (20)	Balance of customer	Not null
Acc_started_date	date	Account started date	Not null
status	Varchar(20)	Status of account	Not null

Table 5: savingsacc

6. TABLE NAME: depositacc

DESCRIPTION: fixed deposit account Details

PRIMARY KEY: deposit_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Deposit_id	Integer (10)	Deposit id	Primary key
Customer_id	Integer (10)	Customer id	Foreign key
Branch_id	Integer (10)	Branch id	Foreign key
Acc_no	Integer (10)	Account number	Foreign key
Ifsc_code	Integer (10)	Ifsc code	Not null
Deposit_amt	Integer (20)	Deposit amount	Not null
tenure	Integer (20)	tenure	Not null
Maturity_date	date	Maturity date	Not null

Deposit_date	date	date of deposit	Not null
Depsoit_type	Varchar(25)	Type of deposit	Not null
Interest_rate	Int(11)	Interest rate	Not null
Interest_amt	Int(11)	Interest amount	Not null
Interest_earn	Int(11)	Interest earn	Not null
Acc_to	Int(11)	Interest amount send to	Not null
Last_transaction_date	date	Last interest update date	Not null
Acc_status	Varchar(20)	status	Not null

Table 6: depositacc

7. TABLE NAME: loanacc

DESCRIPTION: loan account Details

PRIMARY KEY: loan_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Loan_id	Integer (10)	Loan id	Primary key
Customer_id	Integer (10)	Customer id	Foreign key
Branch_id	Integer(10)	Branch id	Foreign key
Acc_no	Integer(11)	Account number	Not null
Ifsc_code	Integer (11)	Ifsc code	Not null
Loan_type	Varchar (25)	Type of loan	Not null
Maturity_date	date	Maturity date	Not null
tenure	Integer(11)	tenure	Not null
Interest_date	date	Interest date	Not null
Interest_amt	Integer(11)	Interest amount	Not null
Issued_amount	Integer(11)	Issued loan amount	Not null
Remaining_amount	Integer(11)	Remaining loan amount	Not null
Date_issued	date	Date issued	Not null

Emi_payment_date	date	Emi payment date	Not null
Acc_status	Varchar(25)	status	Not null

Table 7: loanacc

8. TABLE NAME: loanpayment

DESCRIPTION: loan payment Details

PRIMARY KEY: payment_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Payment_id	Integer (10)	Payment id	Primary key
Customer_id	Integer(10)	Customer id	Foreign key
Loan_id	Integer(10)	Loan id	Foreign key
Branch_id	Integer (10)	Branch id	Foreign key
amount	Integer (10)	Amount paid	Not null
date	date	date	Not null

Table 8: loanpayment

9. TABLE NAME: loanborrower

DESCRIPTION: loan borrower details

PRIMARY KEY: borrower_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Payment_id	Integer (10)	Payment id	Primary key
Customer_id	Integer(10)	Customer id	Foreign key
Branch_id	Integer (10)	Branch id	Foreign key
Loan_id	Integer (10)	Loan id	Foreign key

Table 9: loanborrower

10. . TABLE NAME: cheque

DESCRIPTION: loan account details

PRIMARY KEY: loan_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Cheque_id	Integer (10)	Cheque id	Primary key
Customer_id	Integer (10)	Customer id	Foreign key
Account_id	Integer(10)	Account id	Foreign key
Branch_id	Integer (10)	Branch id	Foreign key
Cheque_no	Varchar (25)	Cheque no	Not null
Dateofissue	Date	Date of issue	Not null
status	Varchar(25)	status	Not null

Table 10: cheque

11. . TABLE NAME: Complaint

DESCRIPTION: Complaint Details

PRIMARY KEY: complaint_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Complaint id	Integer (10)	Id of complaint	Primary key
Customer id	Integer (10)	Id of customer	Foreign key
Complaint	Varchar (25)	Complaint	Not null
Replay	Varchar (25)	Replay	Not null
Date	Date	Date of complaint	Not null

Table 11: complaint

12. . TABLE NAME: debitcard

DESCRIPTION: debit card Details

PRIMARY KEY: debit_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Debit_id	Integer (10)	Debit id	Primary key
Customer_id	Integer (10)	Customer id	Foreign key
Account_id	Integer(10)	Account id	Foreign key
Branch_id	Varchar (25)	Branch id	Foreign key
Debit_no	Integer(11)	Debit card number	Not null
cv	Integer(11)	Cv number	Not null
Expiry_date	date	Expiry date of debit card	Not null
status	Varchar(20)	Status of debit card	Not null

Table 12: debitcard

13. TABLE NAME: creditcard

DESCRIPTION: credit card Details

PRIMARY KEY: credit_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Credit_card_id	Integer (10)	Debit id	Primary key
Customer_id	Integer (10)	Customer id	Foreign key
Branch_id	Integer(10)	Account id	Foreign key
Card_no	Integer(10)	Card number	Foreign key
Card_limit	Varchar (25)	Card limit	Foreign key
cv	Integer(11)	Cv number	Not null
Expiry_date	date	Expiry date of debit card	Not null
status	Varchar(20)	Status of debit card	Not null

Table 13: creditcard

14. . TABLE NAME: transaction

DESCRIPTION: transaction details

PRIMARY KEY: transacion_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Transaction_id	Integer (10)	Id of enquiry	Primary key
Customer_id	Integer (10)	Id of customer	Foreign key
Branch_id	Varchar (25)	Id of packager	Foreign key
Account_id	Varchar (25)	Details of enquiry	Foreign key
Trans_no	Integer(11)	Transaction number	Not null
Trans_type	Varchar(20)	Transaction type	Not null
amount	Integer(11)	Amount	Not null
Date	Date	Date of enquiry	Not null

Table 14: transaction

15. TABLE NAME: bankproduct

DESCRIPTION: babk product Details

PRIMARY KEY: product_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Product_id	Integer (10)	Id of rate	Primary key
Customer_id	Integer (10)	Rating details	Not null
count	Varchar (25)	Review details	Not null

Table 15: bankproducts

16. TABLE NAME: notescount

DESCRIPTION: bank note count details

PRIMARY KEY: count_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Count_id	Integer (10)	Count id	Primary key
Branch_id	Integer (10)	Branch id	Foreign key
Note_type	Integer (10)	Type of notes	Not null
count	Varchar (25)	Count of notes	Not null

Table 16: notescount

17. TABLE NAME: feedback

DESCRIPTION: Feedback Details

PRIMARY KEY: feedback_id

FIELD NAME	TYPE	DESCRIPTION	CONSTRAINTS
Feedback_id	Integer (10)	Feedback id	Primary key
Customer_id	Integer (10)	Customer id	Foreign Key
Branch_id	Integer (10)	Branch id	Foreign Key
message	Varchar (25)	Date of chat	Not null
reply	Varchar(25)	Reply message	Not null
date	date	date	Not null

Table 17: feedback

3.3 OBJECT ORIENTED DESIGN – UML DIAGRAMS

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system. There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after (as part of documentation).

The two broadest categories that encompass all other types are Behavioral UML diagram and Structural UML diagram. As the name suggests, some UML diagrams try to analyze and depict the structure of a system or process, whereas other describe the behavior of the system, its actors, and its building components. The different types are broken down as follows:

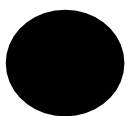
- Activity Diagram
- Class Diagram
- Component Diagram
- Object Diagram
- Package Diagram
- Sequence Diagram

3.3.1 ACTIVITY DIAGRAM

Activity diagrams are probably the most important UML diagrams for doing business process modeling. In software development, it is generally used to describe the flow of different activities and actions. These can be both sequential and in parallel. They describe the objects used, consumed or produced by an activity and the relationship between the different activities. All the above are essential in business process modeling.

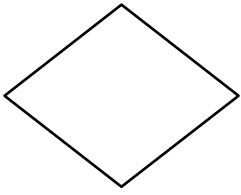
BASIC ACTIVITY SYMBOLS

Initial State



It depicts the initial stage or beginning of the set of actions.

Decision Box



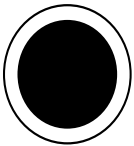
It makes sure that the control flow or object flow will follow only one path.

Action Box



It is the stage where all the control flows and object flows end.

Final State



It is the stage where all the control flows and object flows end.

ACTIVITY DIAGRAM

ADMIN

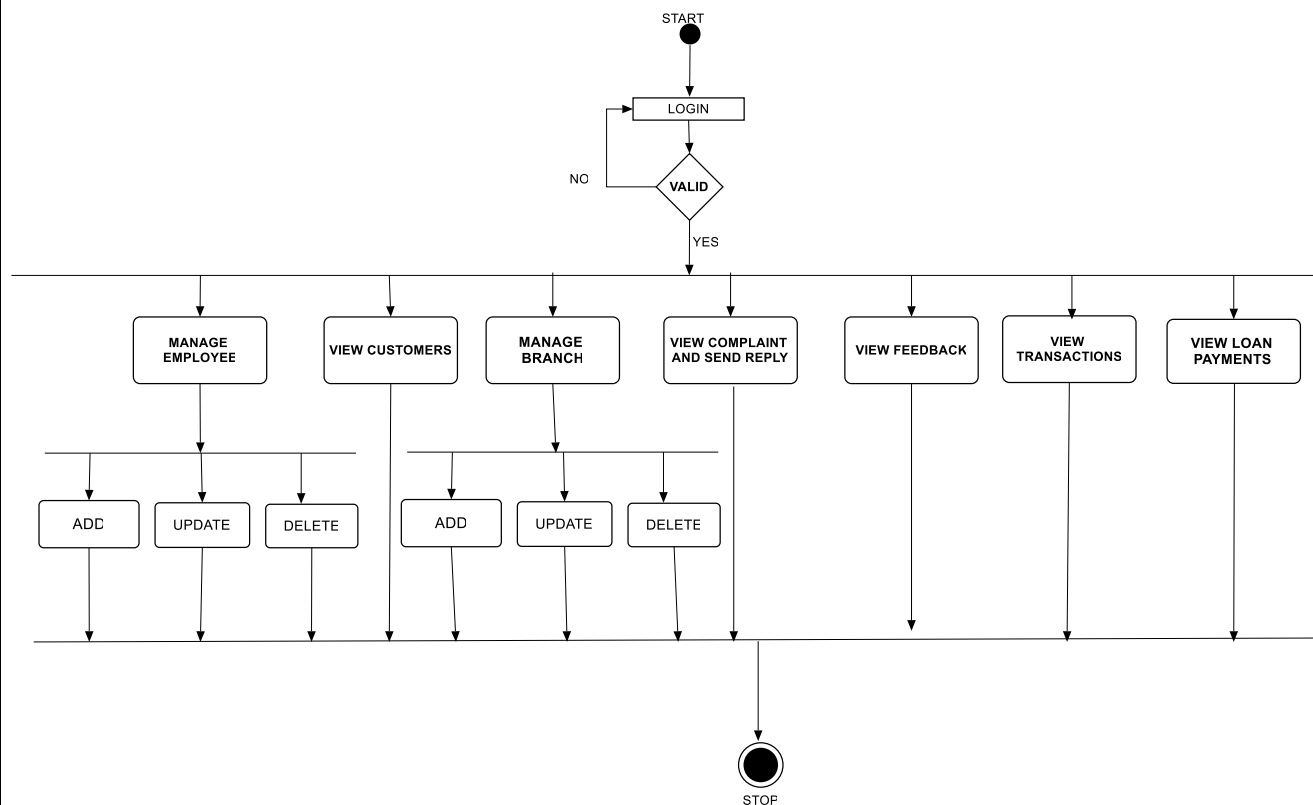


FIG 2: ACTIVITY DIAGRAM FOR ADMIN

CUSTOMER

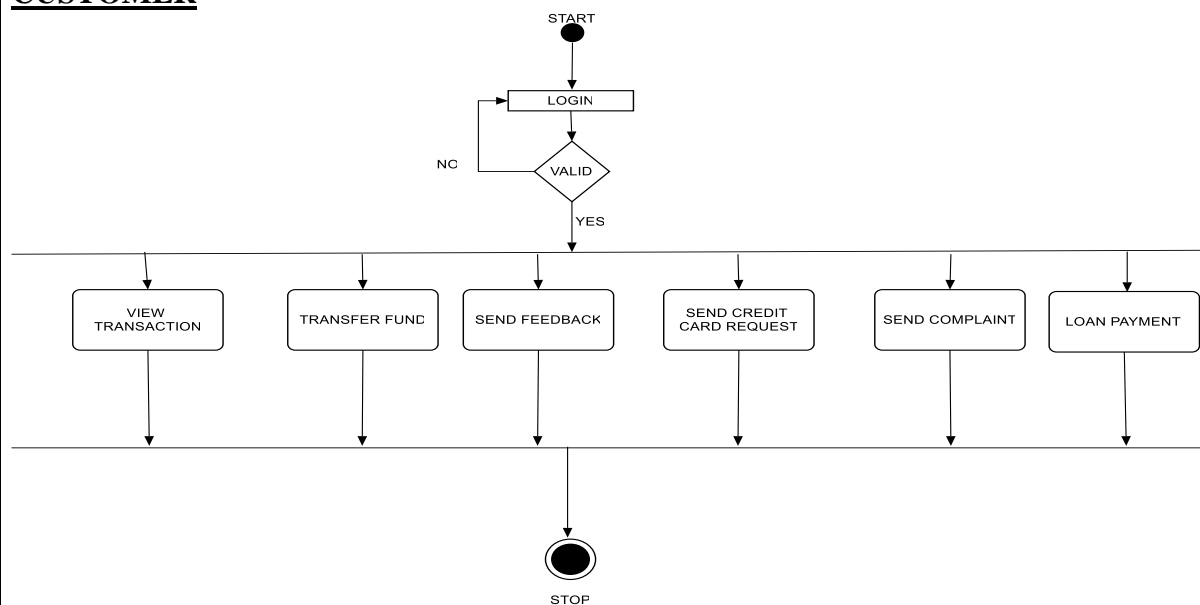


FIG 3:ACTIVITY DIAGRAM FOR CUSTOMER

CLERK

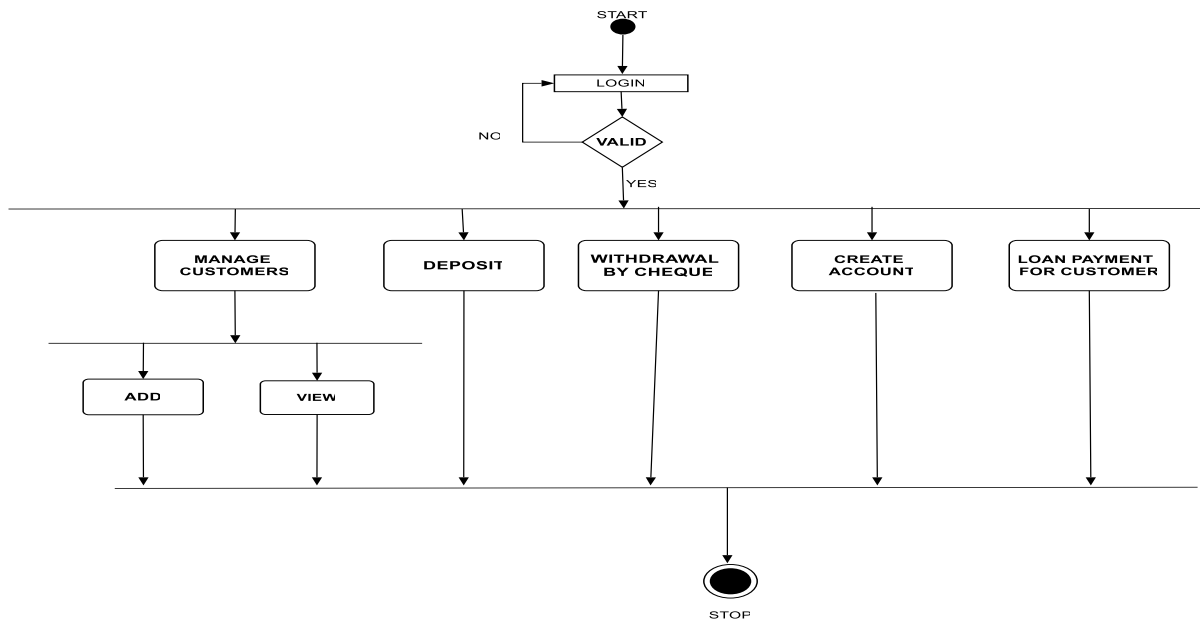


FIG 4:ACTIVITY DIAGRAM FOR CLERK

MANAGER

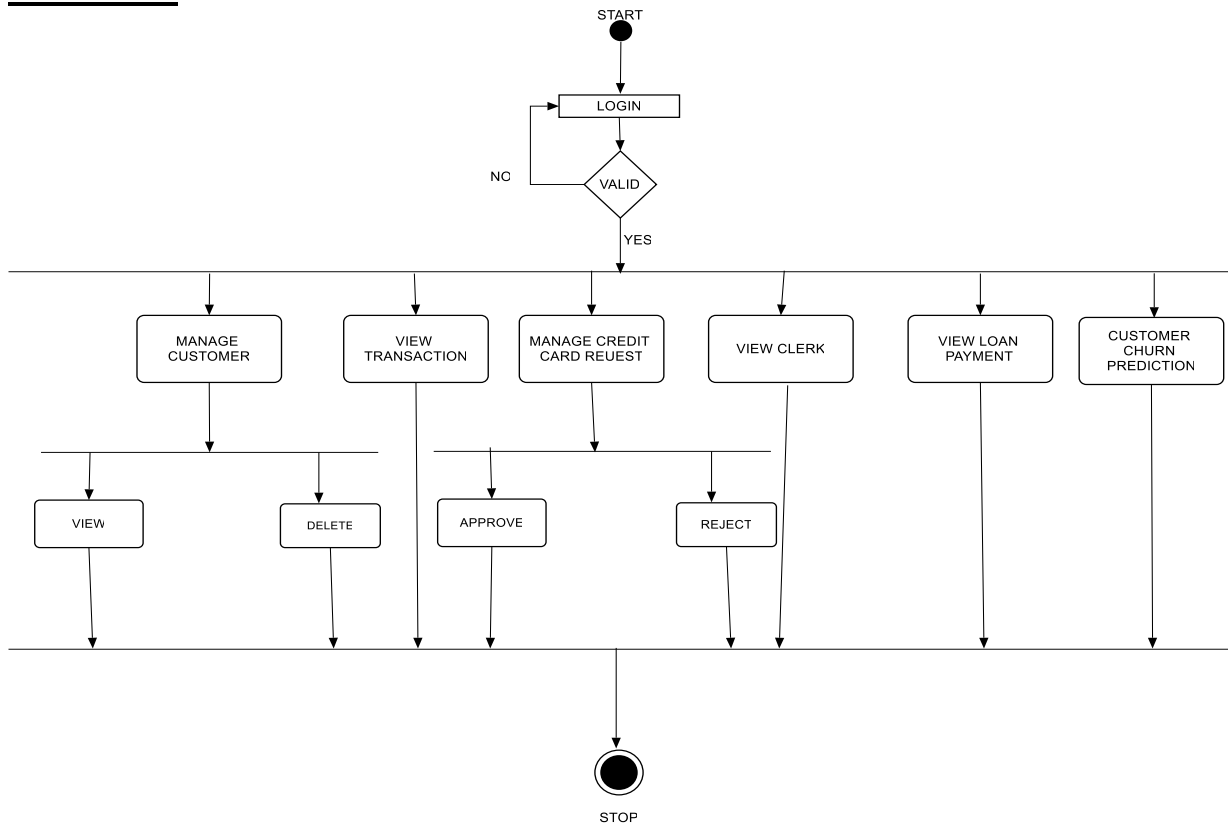
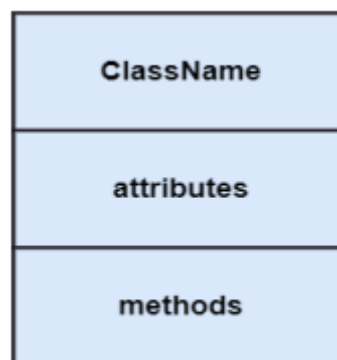


FIG 5:ACTIVITY DIAGRAM FOR MANAGER

3.3.2 CLASS DIAGRAM

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code. It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. The class diagram is made up of three sections:

- Upper Section: The upper section encompasses the name of the class. A class is a representation of similar objects that shares the same relationships, attributes, operations, and semantics.
- Middle Section: The middle section constitutes the attributes, which describe the quality of the class.
- Lower Section: The lower section contain methods or operations. The methods are represented in the form of a list, where each method is written in a single line. It demonstrates how a class interacts with data.



CLASS DIAGRAM

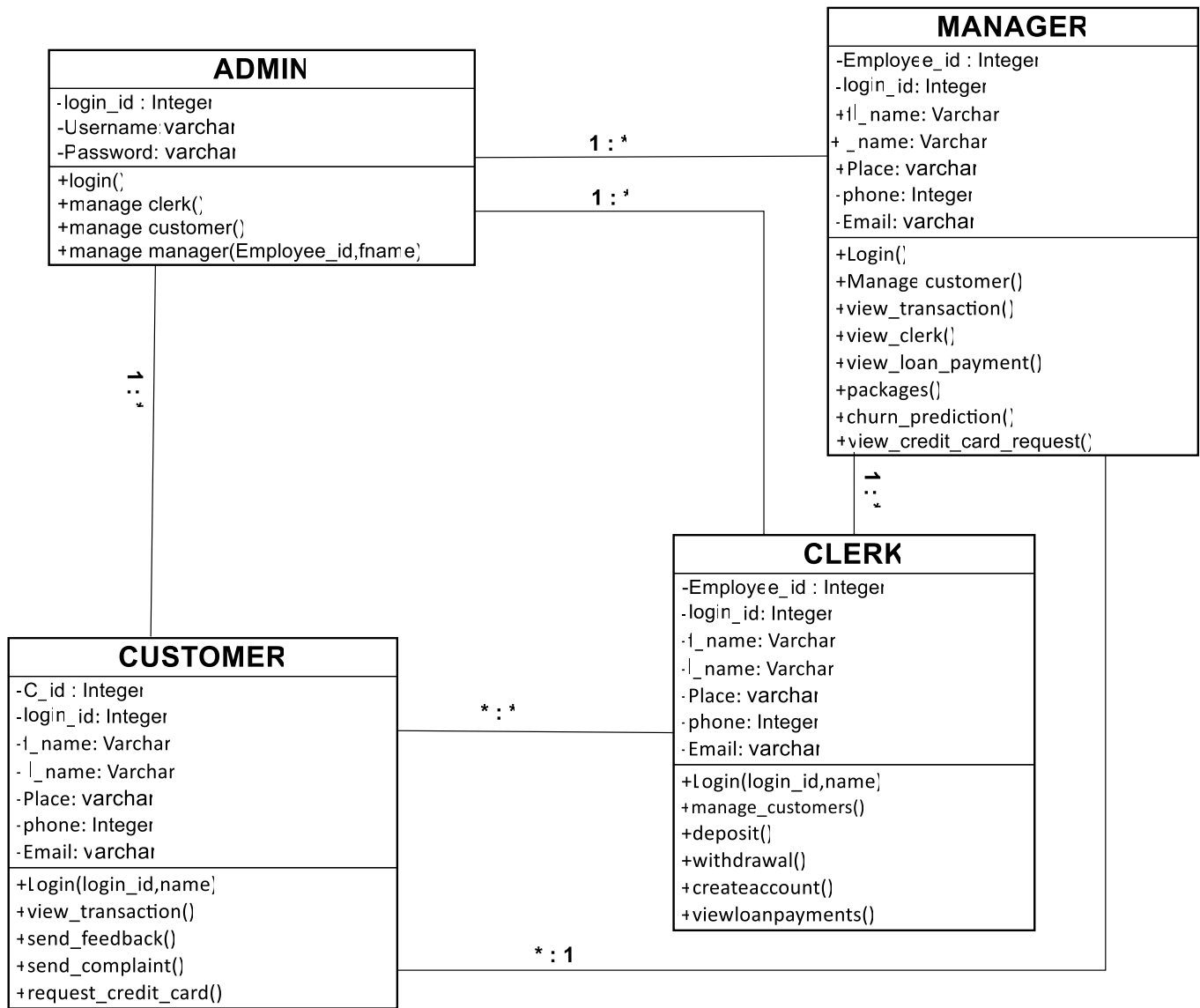


FIG 6: CLASS DIAGRAM

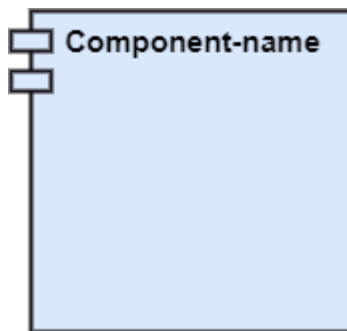
3.3.3 COMPONENT DIAGRAM

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

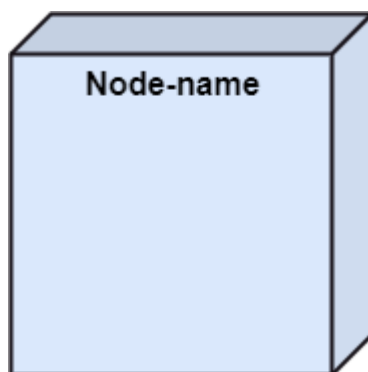
It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

BASIC COMPONENT SYMBOL

a) A component



b) A node



COMPONENT DIAGRAM

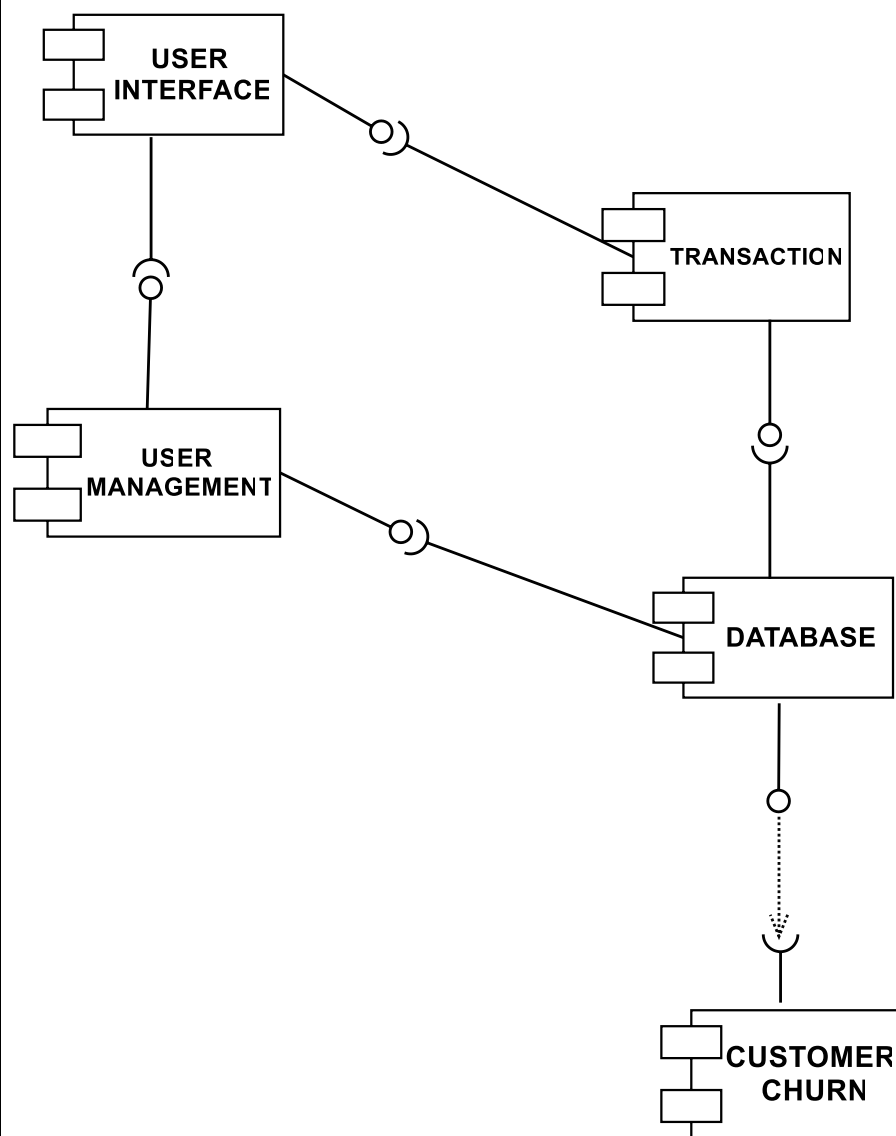


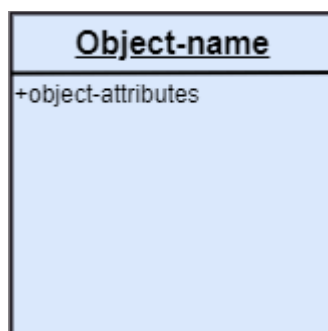
FIG 7: COMPONENT DIAGRAM

3.3.4 OBJECT DIAGRAM

Object diagrams are dependent on the class diagram as they are derived from the class diagram. It represents an instance of a class diagram. The objects help in portraying a static view of an object-oriented system at a specific instant.

Both the object and class diagram are similar to some extent; the only difference is that the class diagram provides an abstract view of a system. It helps in visualizing a particular functionality of a system.

BASIC OBJECT SYMBOL



OBJECT DIAGRAM

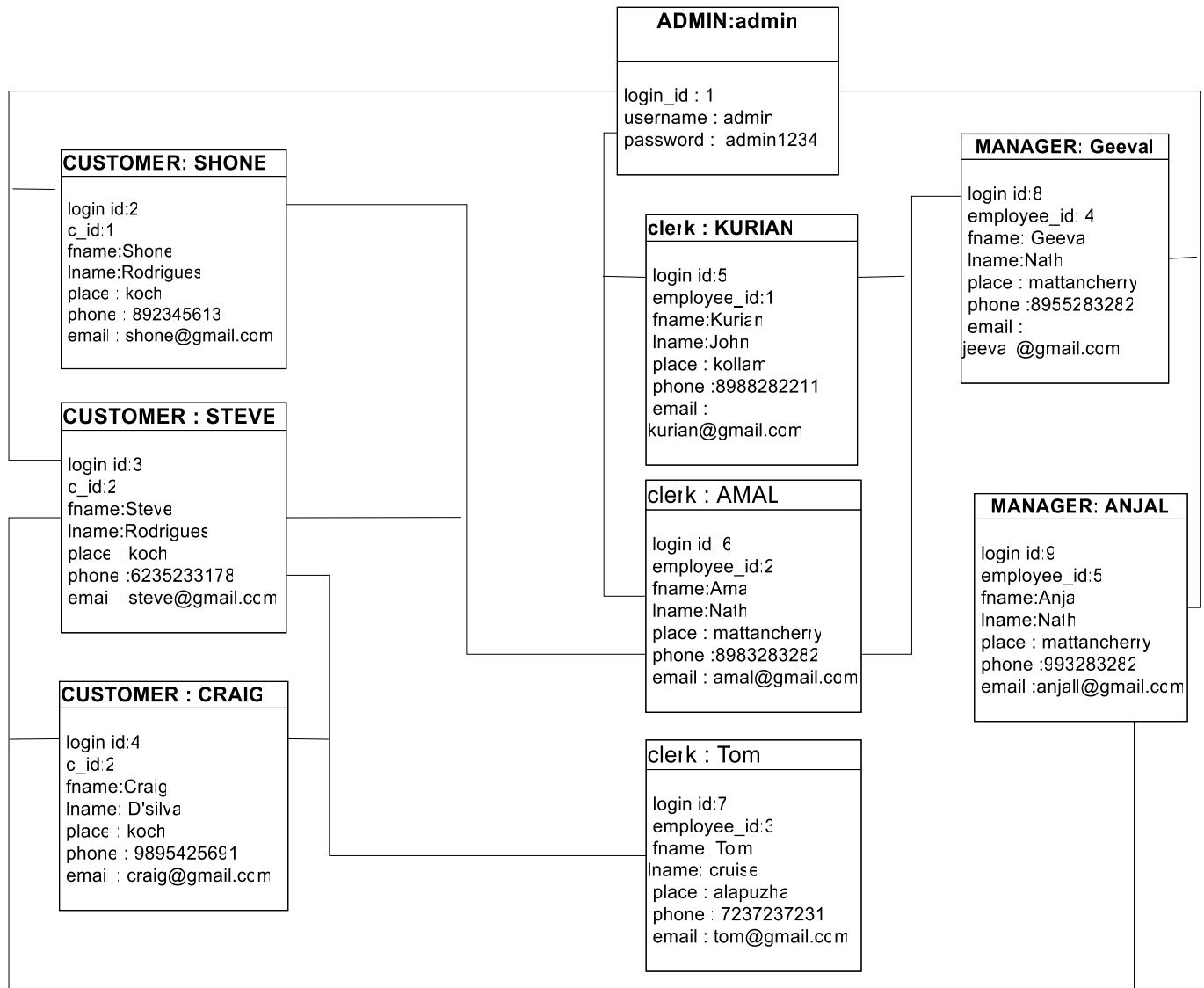


FIG 8: OBJECT DIAGRAM

3.3.5 PACKAGE DIAGRAM

Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages. A package is a grouping of related UML elements, such as diagrams, documents, classes, or even other packages. Each element is nested within the package, which is depicted as a file folder within the diagram, then arranged hierarchically within the diagram. Package diagrams are most commonly used to provide a visual organization of the layered architecture within any UML classifier, such as a software system.

BASIC PACKAGE SYMBOLS

a) Package



Groups common elements based on data, behavior, or user interaction

b) Dependency



Depicts the relationship between one element (package, named element, etc) and another

PACKAGE DIAGRAM

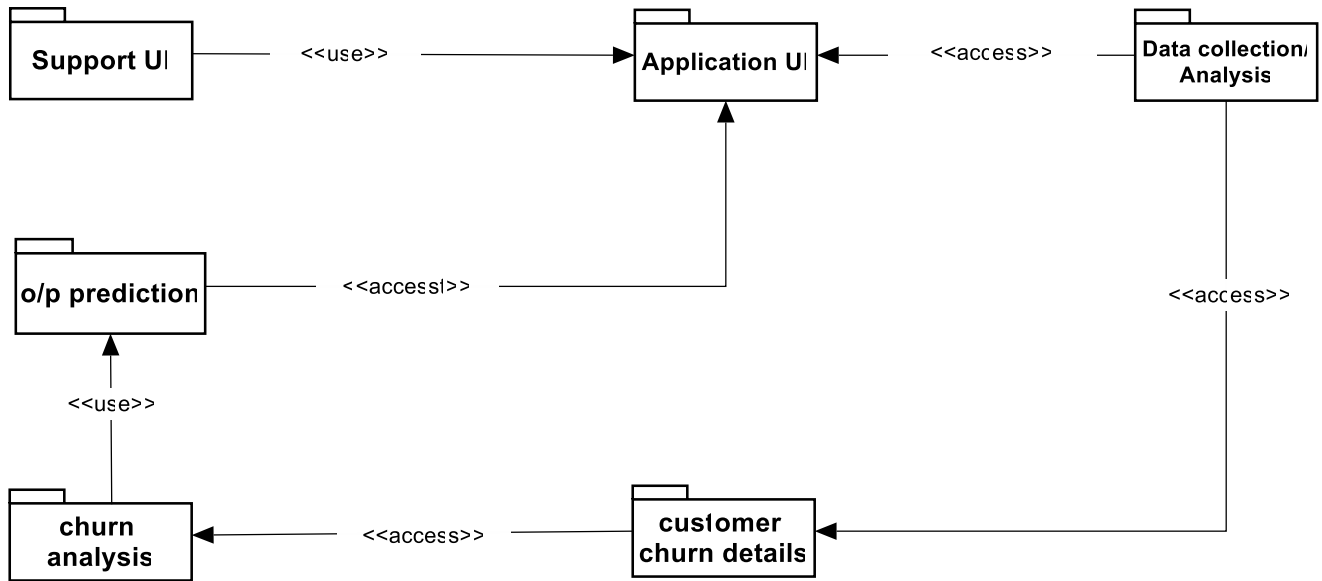


FIG 9 : PACKAGE DIAGRAM

3.3.6 SEQUENCE DIAGRAM

Sequence diagrams are probably the most important UML diagrams among not only the computer science community but also as design-level models for business application development. Lately, they have become popular in depicting business processes, because of their visually self-explanatory nature. As the name suggests, sequence diagrams describe the sequence of messages and interactions that happen between actors and objects. Actors or objects can be active only when needed or when another object wants to communicate with them. All communication is represented in a chronological manner.

Benefits of sequence diagrams

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

SEQUENCE DIAGRAM

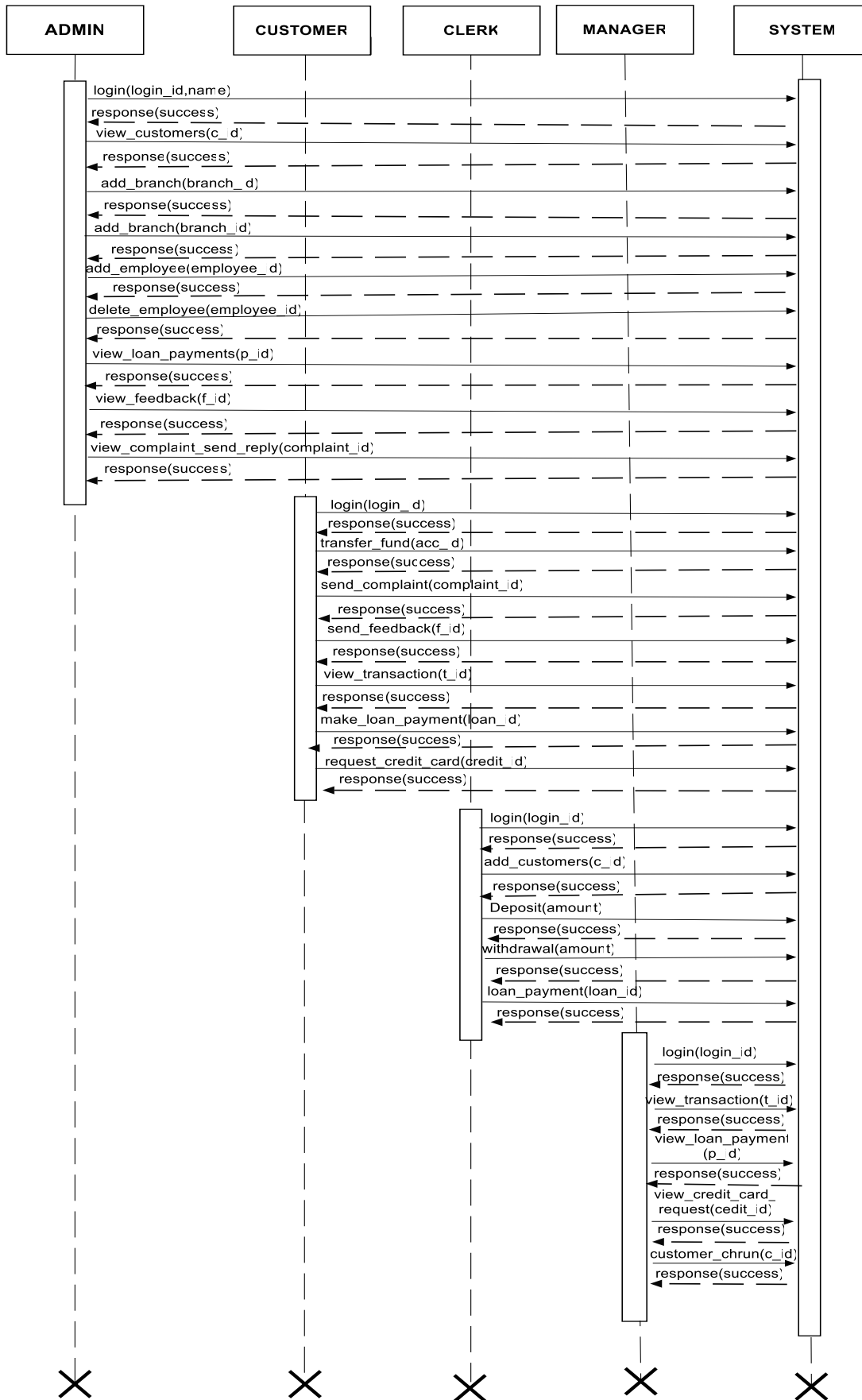


FIG 10: SEQUENCE DIAGRAM

3.4 MODULAR DESIGN

Modular design, or modularity in design, is a design principle that subdivides a system into smaller parts called modules, which can be independently created, modified, replaced, or exchanged with other modules or between different systems.

A modular design is an approach for product designing which is used to produce a complete product by integrating or combining smaller parts that are independent of each other. With the modular design approach, a complex product can be broken down or divided into smaller and simpler components that are independently designed and manufactured. Each of these individual components is then integrated (or assembled) together to form the final product.

3.4.1 STRUCTURE CHART

A structure chart (SC) in software engineering and organizational theory is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's name. The tree structure visualizes the relationships between modules.

Structure Chart represent hierarchical structure of modules. It breaks down the entire system into lowest functional modules, describe functions and sub-functions of each module of a system to a greater detail. Modules at top level called modules at low level. Components are read from top to bottom and left to right. When a module calls another, it views the called module as black box, passing required parameters and receiving results.

In the project “Customer churn at bank”, the entire system is broken down into 4 user modules. With the help of the structure chart, the functions and sub-functions of each module is represented.

STRUCTURE CHART

ADMIN

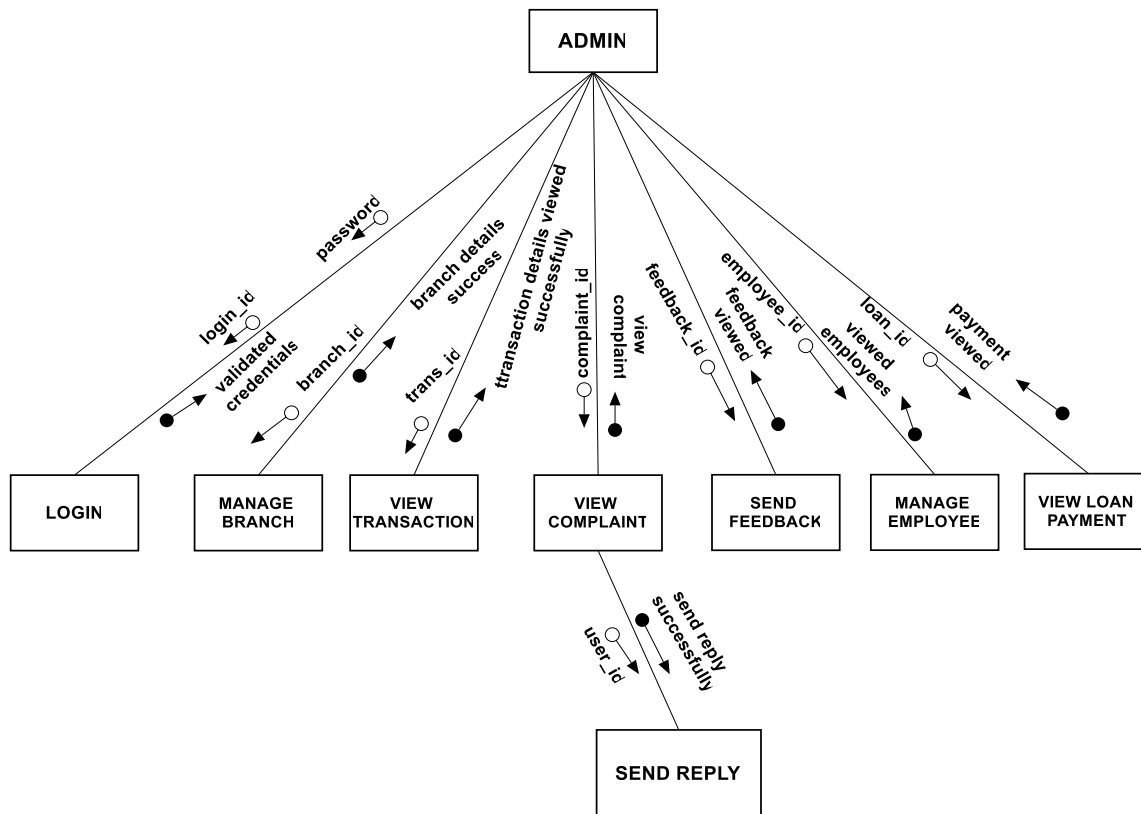


FIG 11: STRUCTURE CHART FOR ADMIN

CUSTOMER

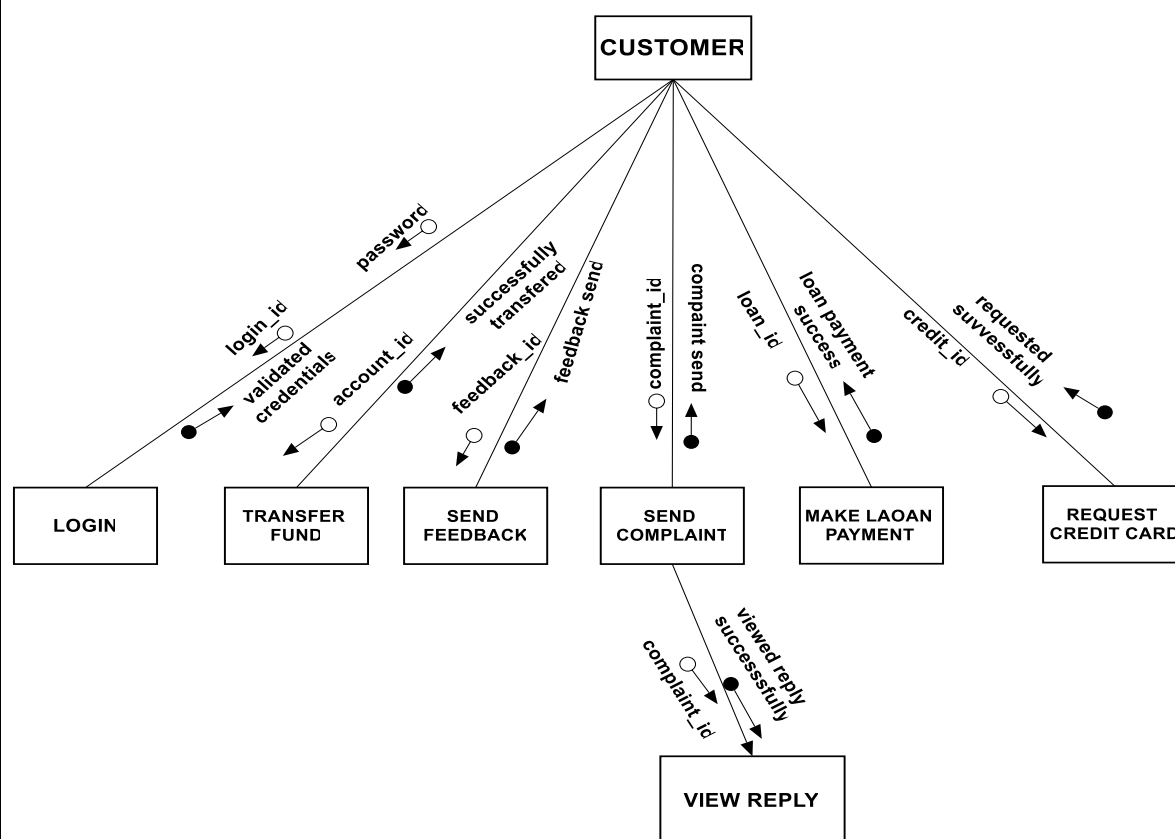


FIG 12: STRUCTURE CHART FOR STAFF

MANAGER

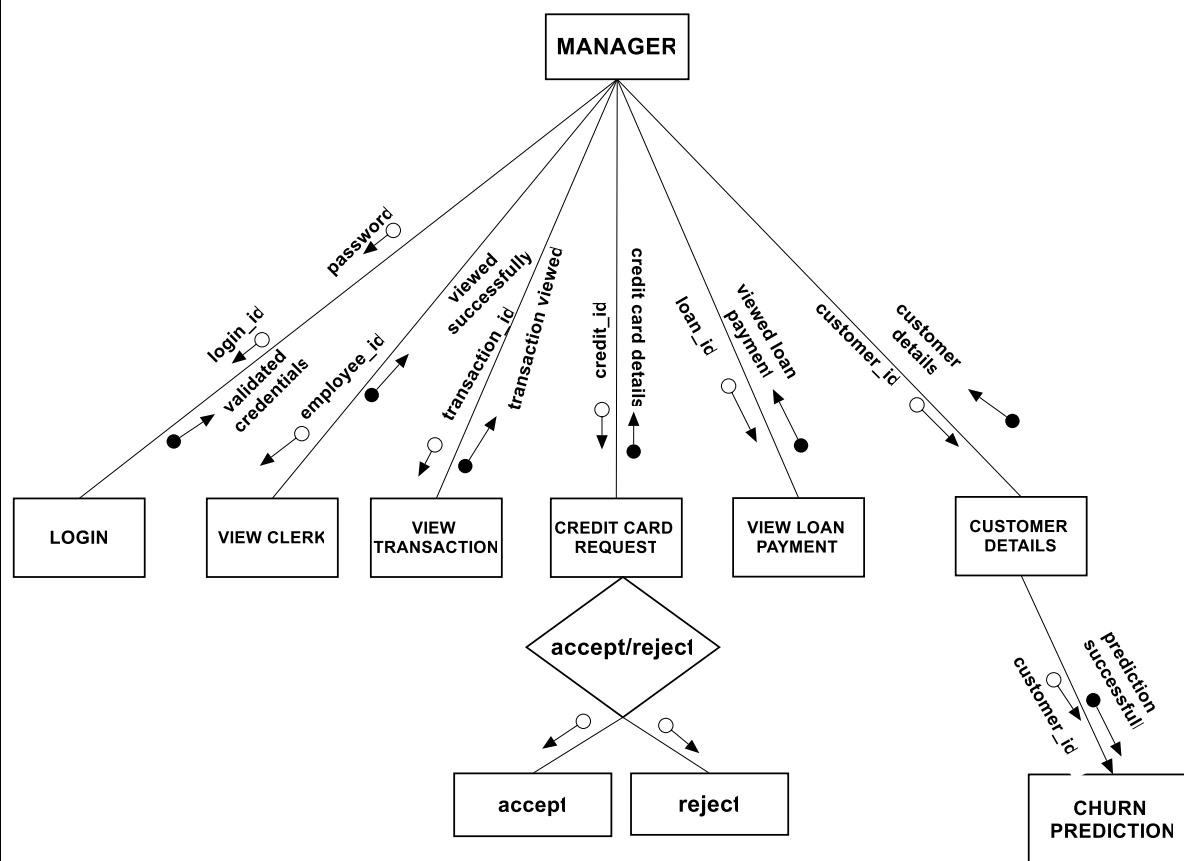


FIG 13: STRUCTURE CHART FOR MANAGER

CLERK

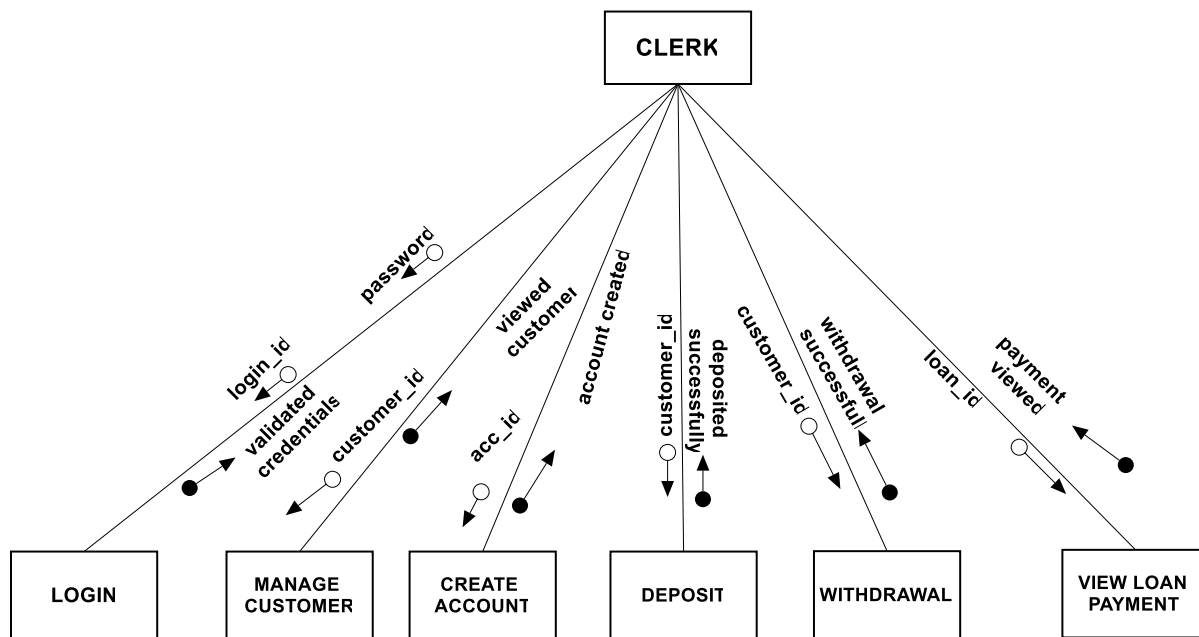


FIG 14: STRUCTURE CHART FOR USER

3.4.2 MODULE DESCRIPTION

A software system is always divided into several sub systems that makes it easier for the development. A software system that is structured into several subsystems makes it easy for the development and testing. The different subsystems are known as the modules and the process of dividing an entire system into subsystems is known as modularization or decomposition. The system under consideration has been divided into several modules taking in consideration the above-mentioned criteria.

1. ADMIN

- **Login :** Admin can login to this application
- **View customers :** Admin can view customers
- **Add managers :** admin can managers.
- **Add clerk :** Admin can add clerk
- **View complaint :** Admin can view complaints from the user and reply.
- **View Feedback :** Admin can view feedback from the user
- **add branch :** Admin can add branch
- **Send bank notification:** send notification to user

2. MANAGER

- **Login** : Manager can login to this system
- **Manage customers**: Can view customers
- **View transaction** : can view customer transaction
- **View loan payment** : can approve/reject/view loan request
- **view clerk** : manager can view clerk
- **credit card request** – manager can approve or request credit card
- **view loan payment** – manager can view loan payments of user
- **send message** – send any message to customers
- **customer churn prediction** – predict whether the customer will leave the bank or not

3. CLERK

- **Login** : login to this system
- **Add customers** : add customer to the bank and view them and create bank account for them .
- **Create account** :
 1. Saving account
 2. Loan account
 3. Fixed deposit account
- **Deposit/withdrawal** : deposit/withdrawal of customer
- **Loan payment for customer** – loan payment of customer

4. CUSTOMER

- **Login** : login to the system
- **View balance** : can view his balance-
- **View his profile** : customer can view his profile.
- **View transaction** : customer can view his previous transaction
- **Request credit card** – can request loan available.
- **Transfer fund** : can transfer fund from one acc to another.
- **Pay loan payments online** – customers can pay loan payments online
- **Send feedback** : sent feedback to admin
- **Send complaint**: can sent complaint to admin
- **View bank notification** : can view notification from bank

5. CHURN PREDICTION

- Predict whether a customer leaves the bank or not by using Classification algorithms.

Functional modules are:

1. USER MANAGEMENT MODULE

- **User Registration/login:** The User Management Module allows new users (customers) to register themselves by providing their personal information, such as name, contact details, and identification documents. Users can log in to the system using their unique credentials, such as username and password, to access their account and perform banking operations
- **User Profiles:** Each user has a profile associated with their account, which contains their personal information, contact details, and account settings. Users can update their profile information as needed.
- **Role-Based Access:** The User Management Module assigns different roles to users, such as admin, manager, clerk, or customer, based on their responsibilities and permissions within the system. This ensures appropriate access control and security.
- **User Dashboard:** Customers can access a personalized dashboard that provides an overview of their accounts, transaction history, balances, and other relevant information.
- **User Management:** Admin users can manage user accounts, including creating new accounts, modifying user details, assigning roles, and deactivating or deleting user accounts as necessary.

2. ACCOUNT MODULE

- **Account Creation:** Bank clerks can create new customer accounts by capturing relevant information such as name, contact details, identification documents, and account type (e.g., savings, fixed deposit).
- **Account Management:** Bank clerks and administrators can manage customer accounts, update customer information, and modify account settings as needed.
- **Account Balance:** Customers can view their account balance in real-time, allowing them to monitor their funds and track their financial transactions.
- **Account Statements:** Customers can generate account statements that provide a comprehensive overview of their transaction history, including details of deposits, withdrawals
- **Account Notifications:** Customers can receive account-related notifications, such as balance updates, transaction alerts, or account activity notifications, to stay informed

- about their financial activities.

3. TRANSACTION MODULE

- **Transfer Funds:** Allows customers to transfer money between their accounts or to other accounts within the bank.
- **Deposit/Withdrawal/check:** Enables customers to deposit money into their accounts or make withdrawals.
- **Transaction History:** Allows customers to view their transaction history, including details of deposits, withdrawals, and transfers

4. LOAN MANAGEMENT MODULE

- **Loan Disbursement:** Once a loan request is approved, this module facilitates the disbursement of funds to the customer's account.
- **Loan Repayment Management:** The Loan Module includes features for managing loan repayments, including setting up repayment schedules, calculating installment amounts, interest, and tracking payment history.

5. CUSTOMER CHURN PREDICTION MODULE

- Predict whether a customer leaves the bank or not by using Classification algorithms.

3.5 INPUT DESIGN

Input design is the process of converting user-oriented input to a based format. Inaccurate input data are the most common cause of errors in data processing. Errors entered by data entry operators can be controlled by input design. The goal of designing input data is to make data entry as easy, logical and free from errors. When we approach input data design; we design the data source documents that capture the data and then select the media used to enter them into computer. User-friendly screen format can reduce the burden on end users, who are not highly proficient in computers.

An important step in input design stage is a design of source document. Source document is the form in which the data can initially capture. The next step is the design of the document layout. In the layout organizes the document by placing information, where it will be noticed and establishes the appropriate sequence of items. In our system, almost all inputs are being taken from the databases. To avoid adequate inputs we have to select necessary values from the databases and arrange it to the appropriate controls.

Form 1 – Login page

Sign in

f

G+

in

or use your account

username

Password

Forgot your password?

LOGIN

Welcome Back!

To keep connected with us please login with your personal info

READ MORE

ABOUT US

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed tristique bibendum neque vel viverra. Nam malesuada nisl at turpis ultrices, ut suscipit sapien hendrerit.

LINKS

Home

About Us

Contact Us

CONTACT US

123 Main St

City, State 12345

Phone: 555-555-5555

Email: info@example.com

f

tw

ig

© 2023 MyCompany. All Rights Reserved.

Form 2 – Add employee

INVEST

search

Steve Rodriguez

Dashboard

dd-mm-yyyy

Dashboard

employee

Branch

customers

General Information

First name

Last name

dd-mm-yyyy

dob

Choose File

No file chosen

Choose photo

Uploaded Image will shown below

Select marital status

Status

Select gender

Gender

Select Branch

Branch

Select employee

Employee

Contact Details

Address

Zip Code

Place

Select a district

District

91

Code +

Phone Number

Your Email

Register

Go back to home page

Go Back

Form 3 – Add customers

The screenshot displays a web application interface for adding a new customer. At the top, there is a header with the 'INVEST' logo, a search bar, and a user profile for 'craig d silva'. The main content area is divided into two columns. The left column, titled 'General Information', contains fields for 'First name', 'Last name', 'Date of Birth' (with a date picker), a 'Choose File' button for a photo, a 'Select marital status' dropdown, a 'Select gender' dropdown, a 'Code +' field, and an 'Email' field. The right column, titled 'Details', contains fields for 'Address', 'city', 'state', 'zipcode', 'country', 'Select your education' (dropdown), 'monthly salary', 'Select your ID' (dropdown), and 'ID NUMBER'. A sidebar on the left shows navigation links for 'Dashboard', 'customers', 'Open new Account', 'transactions', and 'loan'. A 'Go Back' button is located at the bottom left of the sidebar.

3.6 OUTPUT DESIGN

Computer output is the most important and direct source of information to the user. Efficient and intelligent output design improves the system's relationship and helps user decision-making. In the output design it is determine how the implementation is to be played for immediate need and also the hardcopy output. A major form of input is a hardcopy from the printer. Printouts should be designed around the output requirement of the user. Printers, CRT screen display are the examples for providing computer based output.

The output design associated with the system includes the various reports of the table generations and query executions. Output design is one of the, most important features of the information system. The logical design of an information system is analogous to an engineering blue print of an automobile. It shows the major features and how they are related to one another. The outputs, inputs and databases are designed are in this phase.

Form 1 – view employees

Dashboard
 employee
 Branch
 customers

Name	Age	Marital Status	Gender	Branch	Employee Type	Address	Place	District	Phone	Email	Manage
sas fdsdfs	2023-07-08	married	male	tip	manager	Henry Hernandez Notting Estate 123 Notting Lane Nottingham NG1 1AJ England	kochi	Alappuzha	6235233153	steverodrigues369@gmail.com	View Details
reena rodrigues	45	married	female	tip	clerk	Henry Hernandez Notting Estate 123 Notting Lane Nottingham NG1 1AJ England	kochi	Kottayam	6235233150	steverodrigues976@gmail.com	View Details
shone newniz	45	single	male	tip	clerk	Henry Hernandez Notting Estate 123 Notting Lane Nottingham NG1 1AJ England	kochi	Kasaragod	6235233153	steverodrigues976@gmail.com	View Details

Go back to home page

[Go Back](#)

Form 2- view employees

Dashboard
 employee
 Branch
 customers

Name	dob	Gender	Branch	Address	State	Phone	Email	Manage
steve SSS	2023-07-07	male	tip	dfdsfsdf	kerala	815693886158	krishna@gmail.com	View Details
shone james	1995-05-15	male	tip	rggrgrgrdgrdgrgfgfgfd	kerala	623523315	robingrace491@gmail.com	View Details
shone rodrigues	1990-03-15	male	tip	kurshu parambu road kuroshu parambu house nazareth	karnadaka	623523315	tobeymagu16@gmail.com	View Details
abiram SSS	1975-05-19	male	tip	yhtyftytyt	kerala	623523316	steverodrigues369@gmail.com	View Details

[Previous](#)
[Next](#)

Go back to home page

[Go Back](#)

Form 3 – churn prediction details

INVEST

search

🌙

👤

steve
rodrigues

Dashboard

customers

Churn Customers

credit card

dd-mm-yyyy

Customer Churn

Customer ID: 2

CreditScore: 595

Age: 48

Tenure: 5

Balance: 27414

NumOfProducts: 5

HasCrCard: 0

IsActiveMember: (1: active, 0: not active): 0

EstimatedSalary: 25000

The Customer will leave the bank

←

Go back
to home page

Go Back

Possible reasons to Churn

Reason for leaving: Inactive Customer

Reason for leaving: No Credit Card

Suggestions for the bank:

Inactive Customer

1. Send personalized and targeted re-engagement campaigns


2. Offer exclusive promotions or discounts for inactive customers

3. Provide tailored product recommendations based on past interactions

Probability of leaving: 95.32763361930847%

Probability of Leaving

Probability of Staying



4.SYSTEM ENVIRONMENT

4.1 INTRODUCTION

System analysis or study is an important phase of any system development process. The system is studied to the minutest details and analyzed. The system analyst plays the role of an interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the inputs to the system are identified.

The outputs from the organization are traced to through the various processing that the input phases through in the organization. A detailed study of this process must be made by various techniques like interviews; questionnaires etc. the data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of the system functions. This system is called existing system. Now the existing system is close study and problem solver tries to sort out difficulties that the enterprise faces.

4.2 SOFTWARE REQUIREMENTS SPECIFICATION

A software requirement specification (SRS), a requirements specification for a software system, is a complete description of the behaviour of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints) the software requirements specification document enlists all necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with the project team and customer.

Operating System: WINDOWS 8 or above for better performance

Front end: HTML , CSS

Back end: PYTHON, MYSQL, Android studio

Software: Sub Lime Text, WAMP

Web Browser: Internet Explorer/Google Chrome/Firefox

Web Server: Apache

4.3 HARDWARE REQUIREMENTS SPECIFICATION

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application.

Processor: Intel Pentium or above.

Hard Disc: 320GB.

Display Type: PC Display.

Keyboard: PC/AT Enhanced PS/2Keyboard (110/10Key).

Mouse: First/Pilot Mouse Serial (c48).

Input Device: Mouse, keyboard

Output Device: Monitor, Mobile Display

4.4 TOOLS, PLATFORMS

4.4.1 FRONT END TOOL

HTML

The Hyper Text Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web receive HTML documents from a web server or from local storage and render the documents into multimedia web pages.

HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Browsers do not display the HTML tags but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages.

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.

SUBLIME TEXT EDIT 3:

Sublime Text is a shareware cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses.

The following is a list of features of Sublime Text

- Goto Anything, quick navigation to files, symbols, or lines
- Command palette
- uses adaptive matching for quick keyboard invocation of arbitrary commands
- Simultaneous editing: simultaneously make the same interactive changes to multiple selected areas
- Python-based plugin API
- Project-specific preferences
- Extensive customizability via JSON settings files, including project-specific and platform-specific settings
- Cross-platform (Windows, macOS, and Linux)
- Supportive Plugins for cross-platform
- Compatible with many language grammars from TextMate.

4.4.2 BACK END TOOL

PYTHON

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020. Python

consistently ranks as one of the most popular programming languages.

MySQL

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds. Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems. Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

4.4.3 OPERATING SYSTEM

WINDOWS 10

The latest version of the Microsoft Windows operating system is Windows 10. While previous versions of Windows were primarily meant to work on desktop and laptop computers, Windows 10 is also meant to function on tablets. Windows 10 Operating System was created by Microsoft to provide a more personal computing experience across a variety of devices. This is an experience that is optimized for each device type while remaining familiar to all users. Windows 10 will run on a wide range of devices, including PCs, tablets, phones, Xbox One, Microsoft HoloLens, and the Surface Hub.

Windows 10 is a collection of Microsoft operating systems that are part of the Windows NT operating system family. It is the successor of Windows 8.1, which was unveiled approximately two years ago on July 15, 2015, and released to the general public on July 29, 2015.

As a follow-up to Windows 8, Microsoft released Windows 10 in July 2015. Instead of releasing a new, full-fledged operating system as a successor, Microsoft has stated that it will continue to update Windows 10.

Anyone who installs Windows on a legacy PC can update it immediately from Windows 7 or Windows 8 to Windows 10 without having to re-imaging or go through time-consuming system wipes and upgrades.

5. SYSTEM PLANNING AND SCHEDULING

5.1 INTRODUCTION

A Software Project is the complete methodology of programming advancement from requirement gathering to testing and support, completed by the execution procedures, in a specified period to achieve intended software product. Project schedule simply means a mechanism that is used to communicate and know about that tasks are needed and has to be done or performed and which organizational resources will be given or allocated to these tasks and in what time duration or time frame work is needed to be performed. The key to a successful project is planning. When undertaking any kind of project, the first thing should do creating a project plan. Often project planning is ignored in favor of getting on with the work. However, many people fail to realize the value of a project plan in saving time, money and many problems. A project is successful when the needs of the stakeholders have been met. A stakeholder is anybody directly or indirectly impacted by the project.

5.2 PLANNING A SOFTWARE PROJECT

Project planning is at the heart of the project life cycle, and tells everyone involved where you're going and how you're going to get there. The planning phase is when the project plans are documented, the project deliverables and requirements are defined, and the project schedule is created. It involves creating a set of plans to help guide your team through the implementation and closure phases of the project. The plans created during this phase will help you manage time, cost, quality, changes, risk, and related issues. They will also help you control staff and external suppliers to ensure that you deliver the project on time, within budget, and within schedule.

The purpose of the project planning phase is to:

- Establish business requirements.
- Establish cost, schedule, list of deliverables, and delivery dates.
- Establish resources plans.
- Obtain management approval and proceed to the next phase.
- The basic processes of project planning are:
 - Scope planning – specifying the in-scope requirements for the project to facilitate creating the work breakdown structure.
 - Preparation of the work breakdown structure – spelling out the breakdown of the project into tasks and sub-tasks.
 - Project schedule development – listing the entire schedule of the activities and

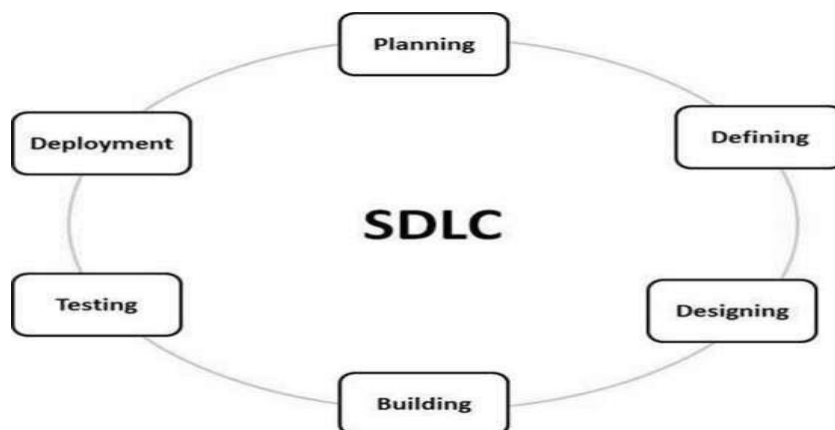
detailing their sequence of implementation.

- Resource planning – indicating who will do what work, at which time, and if any special skills are needed to accomplish the project tasks.
- Budget planning – specifying the budgeted cost to be incurred at the completion of the project.
- Procurement planning – focusing on vendors outside your company and subcontracting.
- Risk management – planning for possible risks and considering optional contingency plans and mitigation strategies.
- Quality planning – assessing quality criteria to be used for the project.

5.2.1 STEPS INVOLVED IN PLANNING A SYSTEM

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



A typical Software Development Life Cycle consists of the following stages –

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification. This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

5.3 GANTT CHART

A gantt chart is a horizontal bar chart used in project management to visually represent a project plan over time. Gantt charts typically show you the timeline and status—as well as who's responsible—for each task in the project.

- Here's a quick look at the details a gantt chart enables you to capture at a glance:
 - How a project breaks down into tasks
 - When each task will begin and end
 - How long each task will take
 - Who's assigned to each task
 - How tasks relate to and depend on each other
 - When important meetings, approvals, or deadlines need to happen
 - How work is progressing in a project

The full project schedule from start to finish.

In other words, a gantt chart is a super-simple way to communicate what it will take to deliver a project on time and budget. That means it's a whole lot easier to keep your project team and stakeholders on the same page from the get-go.

GANNT CHART

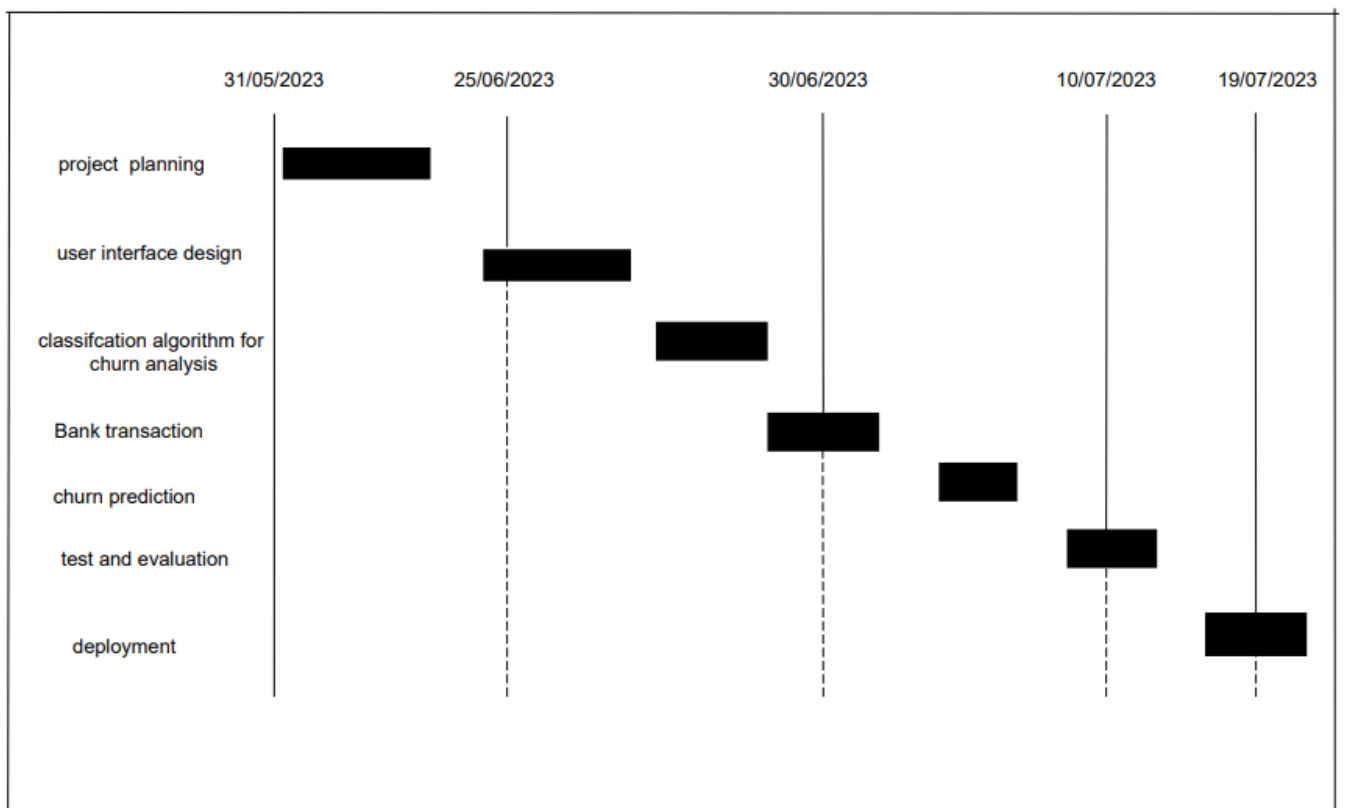


Fig 15 : Gannt Chart

5.4 PERT CHART

A PERT chart is a network diagram utilized in the Program Evaluation Review Technique (PERT) to address a project's course of events. It permits project managers to gauge the span of projects dependent on the analysis of task arrangements. A PERT chart network diagram incorporates numbered hubs, directional bolts, and different enhanced paradigm enhanced the PERT chart and lengthwise, while hubs are achievements.

The PERT chart model will permit you to see these components. The data created by a PERT chart can be brought into project booking programming. With programming, you can utilize a Gantt chart to make a valuable plan for getting work done, deal with your assets and follow through on schedule and under a financial plan.

Utilizing a PERT chart can:

- Explain time imperatives for your group.
- Offer a definite perspective on the grouping in which tasks ought to be performed.
- Assist you with dealing with your time and assets across your group all the more viably.
- Diminish waste and expenses as you complete your project.

NOTATIONS FOR PERT CHART

- **Task:** A task should include name of the task, task ID, task duration, starting date and the ending date of the task.
- **Dependency:** The time dependency relation will link and indicate the time frame of the compilation of the proceeding task on hand and the start of the next task of the project tasks.
- **Lanes:** In PERT Chart, lanes help you arrange project base time for your chart by allowing you to separate the chart with lines and into horizontal zones. The horizontal zones represent the work and responsibilities of an assigned task, work, or planned concept for the group of functions.

PERT CHART

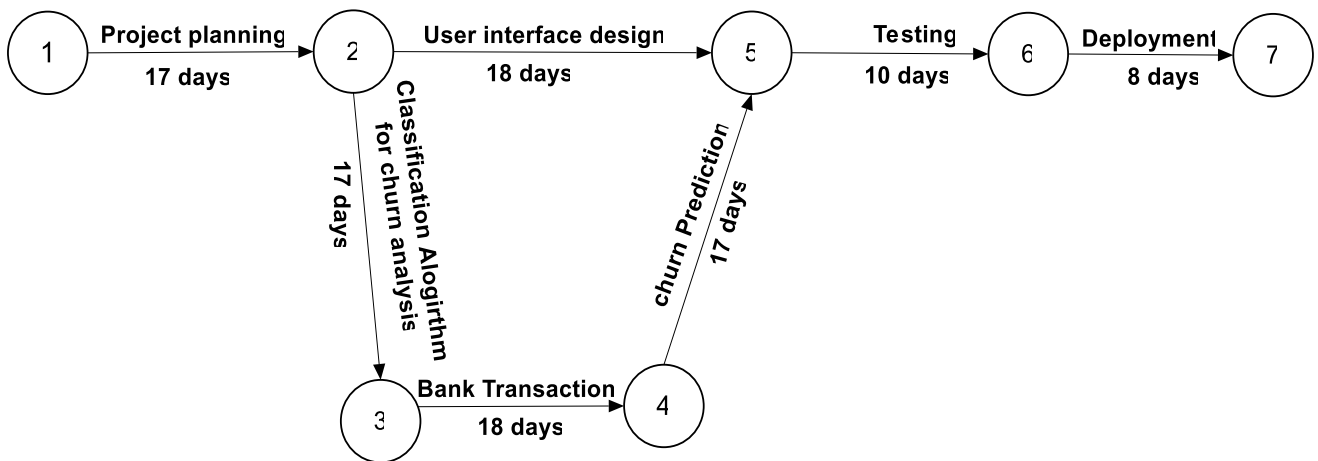


Fig 16 : Pert Chart

6. SYSTEM COST ESTIMATION

6.1 INTRODUCTION

A project can only come together with all the necessary materials and labour, and those materials and labours cost money. Putting together a budget that keeps costs to a minimum, while maximizing the project's quality and scope can be challenging. This is why proper cost estimation is important.

Cost estimation in project management is the process of forecasting the financial and other resources needed to complete a project within a defined scope. Cost estimation accounts for each element required for the project—from materials to labour—and calculates a total amount that determines a project's budget. An initial cost estimate can determine whether an organization greenlights a project, and if the project moves forward, the estimate can be a factor in defining the project's scope. If the cost estimation comes in too high, an organization may decide to pare down the project to fit what they can afford (it is also required to begin securing funding for the project). Once the project is in motion, the cost estimate is used to manage all of its affiliated costs in order to keep the project on budget.

There are two key types of costs addressed by the cost estimation process:

1. Direct costs: Costs associated with a single area, such as a department or the project itself. Examples of direct costs include fixed labour, materials, and equipment.
2. Indirect costs: Costs incurred by the organization at large, such as utilities and quality control

6.2 FUNCTIONAL POINT BASED ESTIMATION

The basic and primary purpose of the functional point analysis is to measure and provide the software application functional size to the client, customer, and the stakeholder on their request. Further, it is used to measure the software project development along with its maintenance, consistently throughout the project irrespective of the tools and the technologies.

Allan J. Albrecht initially developed function Point Analysis in 1979 at IBM and it has been further modified by the International Function Point Users Group (IFPUG). FPA is used to make estimate of the software project, including its testing in terms of functionality or function size of the software product. However, functional point analysis may be used for the test estimation of the product. The functional size of the product is measured in terms of the function point, which is a standard of measurement to measure the software application. FPs of an application is found out by counting the number and types of functions used in the applications.

Function point analysis is a method used to estimate the size and complexity of software development projects based on the functionality provided by the system. It quantifies the functionality in terms of function points, which are derived from the different features and functionalities of the software system. While function point estimation is relevant for software development projects, it is not directly applicable to customer churn prediction.

In this project, the estimation involves predicting whether the customer leaves the bank or not based on factors like customer age, transaction details active or not and bank products,. This is a classification problem, where you aim to develop a model that can accurately predict whether the customer leaves the bank or not based on its attributes. The estimation is based on historical data and the relationships between the input features and the corresponding customer details

7. SYSTEM IMPLEMENTATION AND TESTING

7.1 INTRODUCTION

The implementation is the final state and it is an important phase. It involves the individual programming; system testing, user training and the operational running of developed proposed system that constitutes the application subsystems. A major task of preparing for implementation is education of users, which should really have been taken place much earlier in the project when they were being involved in the investigation and design work. During the implementation phase system actually takes physical shape. In order to develop a system implemented planning is very essential.

The implementation phase of the software development is concerned with translating design specification into source code. The user tests the developed system and changes are made according to their needs. Our system has been successfully implemented. Before implementation several tests have been conducted to ensure that no errors are encountered during the operation. The implementation phase ends with an evaluation of the system after placing into the operation for a period of time. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from old system to new system. The system can be implemented only after testing is done and is found to be working to specifications. The implementation stage is a systems project in its own right. The implementation stage involves following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of method to achieve change over.
- Evaluation of the changeover method.

7.2 CODING

7.2.1 CODING STANDARDS

During the development of the application I follow standard python coding. Python uses indentation to give you a visual indication of the structure of our code. Additionally we have an interactive interpreter which provides a default representation for many objects and data structures. Consistency of style, and using the interpreters representation for objects, will make it easier for you to parse and absorb the meaning of the code. The Python programming language has evolved over the past year as one of the most favorite programming languages. This language is relatively easy to learn than most of the programming languages. It is a multi-paradigm, it has lots of open source modules that add up the utility of the language and it is gaining popularity in data science and web development community. However, you can use the benefits of Python only when you know how to express better with your code.

The following are the some python coding standards used in this project. Indentation:

When programming in Python, indentation is something that you will definitely use. However, you should be careful with it, as it can lead to syntax errors. The recommendation is therefore to use 4 spaces for indentation. And also this for loop with print statement is indented with 4 spaces. Imports: Importing libraries and/or modules is something that you'll often do when you're working with Python for data science. As you might already know, you should always import libraries at the start of your script. Comments: Comments are used for in code documentation in Python. They add to the understanding of the code. There are lots of tools that you can use to generate documentation, such as comments and doc-strings, for your own module. Comments should be more verbose so that when someone reads the code, the person would get the proper understanding of the code and how it is being used with other pieces of the code. Comments start with the symbol. Anything written after the hashtag does not get executed by the interpreter. You write documentation strings or doc-strings at the start of public modules, files, classes and methods. These type of comments start with `'''` and end with `'''`. Maximum Line Length: Generally, it's good to aim for a line length of 79 characters in your Python code. It is possible to open files side by side to compare. You can view the whole expression without scrolling horizontally which adds to better readability and understanding of the code.

7.2.2 SAMPLE CODES

Admin.py

```
from flask import *
from database import *
import uuid
import os
from datetime import datetime
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

app.secret_key = 'your_secret_key'
admin=Blueprint('admin',__name__)
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="bank"
)

@admin.route('/adminhome')
def adminhome():
    months = [
        'January', 'February', 'March', 'April',
        'May', 'June', 'July', 'August', 'September',
        'October', 'November', 'December'
    ]
    cursor = db.cursor()
    cursor.execute("SELECT MONTH(date_time), SUM(amount) FROM transaction GROUP BY MONTH(date_time)")
    data = cursor.fetchall()
    labels = []
    values = []
    for month in months:
        month_num = datetime.strptime(month, '%B').month
        found = False
        for row in data:
            if row[0] == month_num:
                labels.append(month)
                values.append(row[1])
                found = True
                break
        if not found:
            labels.append(month)
            values.append(0)
    cursor.execute("select sum(transaction_amount) from transactions")
    total_amount = cursor.fetchone()[0]
    cursor.execute("select uname from login where login_type='admin';")
    name = cursor.fetchone()[0]
    cursor.execute("""
        SELECT note_type, SUM(count) AS total_count
        FROM notescount
        GROUP BY note_type
        ORDER BY note_type DESC;
```

```

""")

# Fetch all rows from the result
notes = cursor.fetchall()

# =====feedbacks
cursor.execute("SELECT f.messages, f.date_time, c.photo,c.fname,c.lname FROM feedbacks f INNER JOIN
customers c ON f.customer_id = c.cid ORDER BY f.date_time LIMIT 3")

feedback_messages = cursor.fetchall()

cursor = db.cursor()
cursor.execute("SELECT c.fname,c.lname,c.photo,tt.amount,tt.date_time,tt.t_type FROM customers
c,o_transaction t,transaction tt where c.cid=tt.customer_id LIMIT 4")
online_transaction=cursor.fetchall()
query = "SELECT COUNT(*) FROM complaints where reply='0'"

cursor.execute(query)
count = cursor.fetchone()[0]
print("max date=",query)
if 'count_removed' in session:
    session.pop('count_removed') # Remove the 'count_removed' flag from session

return render_template('adminhome.html', labels=labels,
values=values,total_amount=total_amount,name=name,feedback_messages=feedback_messages,count=count,onlin
e_transaction=online_transaction,notes=notes)

@admin.route('/adminviewcomplaints', methods=['POST', 'GET'])
def adminviewcomplaints():

    cursor = db.cursor()
    query = "SELECT c.messages, c.date_time, cu.fname, cu.lname, cu.photo, c.reply, c.complaint_id FROM
complaints c INNER JOIN customers cu ON c.customer_id = cu.cid WHERE c.reply = '0'"
    cursor.execute(query)
    messages = cursor.fetchall()

    if "add" in request.form:
        id=request.form['complaint_id']
        reply = request.form['reply']
        print("reply===",reply)
        cursor.execute("UPDATE complaints SET reply = %s WHERE complaint_id = %s", (reply, id))
        flash("Send reply successfully")
        return redirect(url_for('admin.adminviewcomplaints'))

    return render_template('adminviewcomplaints.html', messages=messages)

@admin.route('/adminaddemployee', methods=['post', 'get'])
def adminaddemployee():
    cursor = db.cursor()
    cursor.execute("SELECT branch_name FROM branch")
    branch_names = [row[0] for row in cursor.fetchall()]
    if 'add' in request.form:
        fname = request.form['fname']

```



```

'%s', '%s', '%s', '%s', 'active')" % (id, branch_id, fname, lname,
age, img, status, gender, branch, employee_type, address, zipcode, place, district, phone, email)
    insert(q)
    # Execute the employee query here
    flash("Registration successful...")
    return redirect(url_for('admin.adminaddemployee'))

return render_template('adminaddemployee.html', branch_names=branch_names)

@admin.route('/adminaddbranch', methods=['post', 'get'])
def adminaddbranch():
    if 'add' in request.form:
        bname = request.form['bname']
        location = request.form['location']
        phoneno = request.form['phone']
        ifsccode = request.form['ifsccode']

        q = "INSERT INTO branch VALUES(null, '%s', '%s', '%s', '%s')" % (bname, location, phoneno, ifsccode)
        insert(q)
        # Execute the employee query here
        flash("Registration successful...")
        return redirect(url_for('admin.adminaddbranch'))
    return render_template('adminaddbranch.html')

@admin.route('/adminmanageemployee', methods=['post', 'get'])
def adminmanageemployee():

    cursor = db.cursor()
    cursor.execute("SELECT * FROM employee order by employee_id desc LIMIT 10")
    employees = cursor.fetchall()

    cursor.execute("select uname from login where login_type='admin'")
    name = cursor.fetchone()[0]
    cursor = db.cursor()
    cursor.execute("SELECT branch_name FROM branch")
    branch_names = [row[0] for row in cursor.fetchall()]
    return
    render_template('adminmanageemployee.html', employees=employees, name=name, branch_names=branch_names)

@admin.route('/viewdetails', methods=['post', 'get'])
def viewdetails():
    name1 = request.args.get('name1')
    name2 = request.args.get('name2')
    age = request.args.get('age')
    image = request.args.get('image')
    maritalStatus = request.args.get('maritalStatus')
    gender = request.args.get('gender')
    branch = request.args.get('branch')
    employeeType = request.args.get('employeeType')
    address = request.args.get('address')
    place = request.args.get('place')
    district = request.args.get('district')
    phone = request.args.get('phone')
    email = request.args.get('email')

```



```

cursor = db.cursor()

cursor.execute("select uname from login where login_type='admin';")
name3 = cursor.fetchone()[0]
return render_template('viewdetails.html', name1=name1, name2=name2,name3=name3,age=age, image=image,
maritalStatus=maritalStatus, gender=gender, branch=branch, employeeType=employeeType, address=address,
place=place, district=district, phone=phone, email=email)

```

manager.py

```

from flask import *
from database import *
from datetime import datetime,timedelta
import datetime
import joblib
import pandas as pd
import random
import os
# model_path = os.path.join('C:', 'mca project 2023', 'BankManagement', 'bank', 'churn_predict_model')
model = joblib.load('C:/mca project 2023/BankManagement/bank/churn_predict_model')
# # model = joblib.load('churn_predict_model')
# C:\mca project 2023\BankManagement\bank\churn_predict_model
app.secret_key = 'your_secret_key'
admin=Blueprint('admin',__name__)
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="bank"
)

manager=Blueprint('manager',__name__)

@manager.route('/managerhome')
def managerhome():
    months = [

```

```

    'January', 'February', 'March', 'April',
    'May', 'June', 'July', 'August', 'September',
    'October', 'November', 'December'
]
cursor = db.cursor()
cursor.execute("SELECT MONTH(transaction_date), SUM(transaction_amount) FROM transactions
GROUP BY MONTH(transaction_date)")
data = cursor.fetchall()
labels = []
values = []
for month in months:
    month_num = datetime.datetime.strptime(month, '%B').month
    found = False
    for row in data:
        if row[0] == month_num:
            labels.append(month)
            values.append(row[1])
            found = True
            break
    if not found:
        labels.append(month)
        values.append(0)
cursor.execute("select sum(transaction_amount) from transactions")
total_amount = cursor.fetchone()[0]
cursor.execute("select employee_fname,employee_lname,email from employee where
loginid='%s'"%(session['logid']))
name = cursor.fetchall()
# =====feedbacks
cursor.execute("select message,date from feedback order by date LIMIT 4")
feedback_messages = cursor.fetchall()
cursor = db.cursor()
return render_template('managerhome.html', labels=labels,
values=values,total_amount=total_amount,name=name,feedback_messages=feedback_messages)

@manager.route('/managermanagecustomers',methods=['post','get'])
def managermanagecustomers():

    cursor = db.cursor()
    cursor.execute("select employee_fname,employee_lname,email from employee where
loginid='%s'"%(session['logid']))
    name = cursor.fetchall()
    cursor.execute("SELECT * FROM customers where branch_id=(select branch_id from employee
where employee_id='%s')"%(session['mid']))
    employees = cursor.fetchall()
    date=datetime.datetime.now()
    if 'add' in request.form:

        messages=request.form['messages']
        customer_id=request.form['customer_id']

```

```

min_allowed_time = date - timedelta(hours=1)
cursor.execute("SELECT MAX(date) FROM bank_messages WHERE customer_id = %s",
(customer_id,))
last_submission_time = cursor.fetchone()[0]
cursor.execute("select branch_id from customers where cid='%s'" % customer_id)
details=cursor.fetchone()
print(details)
branch_id=details[0]

if last_submission_time is None or last_submission_time < min_allowed_time:
    bank_messages="INSERT INTO bank_messages(customer_id,branch_id,messages,date)
VALUES ( %s, %s, %s,%s)"
    bank_messages_values = (customer_id, branch_id,messages,date)
    cursor.execute(bank_messages, bank_messages_values)
    # cursor.execute("SELECT * FROM customers where branch_id=(select branch_id from
employee where employe_id='%s')"%(session['mid']))
    # employees = cursor.fetchall()
    flash("successfully send notification")
else:
    print('send after 1 hoursuccess')

return render_template('managermanagecustomers.html',employees=employees,name=name)

```

```

@manager.route('/customerchurnprediction', methods=['POST'])
def customerchurnprediction():
    if 'add' in request.form:
        creditscore=request.form['credit_score']
        age = request.form['age']
        tenure = request.form['tenure']
        balance = request.form['balance']
        num_of_products = request.form['num_of_products']
        has_cr_card = request.form['has_cr_card']
        is_active_member = request.form['is_active_member']
        estimated_salary = request.form['estimated_salary']
        geography_germany = request.form['geography_germany']
        geography_spain = request.form['geography_spain']
        gender = request.form['gender']
        prediction_data = pd.DataFrame({
            'CreditScore': [creditscore],
            'Age': [age],
            'Tenure': [tenure],
            'Balance': [balance],
            'NumOfProducts': [num_of_products],
            'HasCrCard': [has_cr_card],
            'IsActiveMember': [is_active_member],
            'EstimatedSalary': [estimated_salary],
            'Geography_Germany': [geography_germany],

```

```

        'Geography_Spain': [geography_spain],
        'Gender_Male': [gender]
    })

    prediction = model.predict(prediction_data)
    probability = model.predict_proba(prediction_data)[0][1] * 100
    stay_probability = 100 - probability
    if prediction == 1:
        return render_template('predictionresult.html', prediction_text="The Customer will leave the
bank",
                               probability=probability, stay_probability=stay_probability)
    else:
        return render_template('predictionresult.html', prediction_text="The Customer will not leave the
bank",
                               probability=stay_probability, stay_probability=probability)
    return render_template('customerchurnprediction.html')

```

clerk.py

```

from flask import *
from database import *
import uuid
import demjson
# import datetime
from datetime import datetime, timedelta
import smtplib
# import schedule
# import time
from dateutil.relativedelta import relativedelta
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import random
app.secret_key = 'your_secret_key'
admin=Blueprint('admin',__name__)
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="bank"
)
clerk=Blueprint('clerk',__name__)

@clerk.route('/clerkhome')

```

```

def clerkhome():
    cursor = db.cursor()
    cursor.execute("select employee_fname,employee_lname,email from employee where
loginid='%s'"%(session['loginid']))
    name = cursor.fetchall()
    cursor.execute("SELECT branch_name FROM branch")
    branch_names = [row[0] for row in cursor.fetchall()]
    print(branch_names)
    return render_template('clerkhome.html',name=name)

@clerk.route('/clerkmanagehome')
def clerkmanagehome():
    return render_template('clerkmanagehome.html')

@clerk.route('/clerkadduser', methods=['post', 'get'])
def clerkadduser():
    cursor = db.cursor()
    # cursor.execute("SELECT e.branch,e.employee_fname from employee e,branch b where
e.branch_id=b.branch_id and e.branch_id=(select branch_id from employee where
employee_id='%s'"%(session['loginid'])) and employee='clerk'")
    cursor.execute("SELECT branch from employee where employee_id='%s'" % session['clid'])
    # cursor.execute("SELECT e.branch, e.employee_fname FROM employee e, branch b WHERE
e.branch_id = b.branch_id AND e.branch_id = (SELECT branch_id FROM employee WHERE
employee_id = '{ }') AND employee_id='{ }'".format(session['clid'], session['loginid']))

    branch_names = [row[0] for row in cursor.fetchall()]
    print(branch_names)
    if 'add' in request.form:
        fname = request.form['fname']
        lname = request.form['lname']
        dob = request.form['dob']
        i=request.files['image']
        img="uploads/"+str(uuid.uuid4())+i.filename
        i.save('bank/static/'+img)
        status = request.form['status']
        gender = request.form['gender']
        phone = request.form['phone']
        email = request.form['email']
        branch = request.form['branch']
        address = request.form['address']
        city = request.form['city']
        state = request.form['state']
        zipcode = request.form['zipcode']
        country = request.form['country']
        education = request.form['education']
        msalary = request.form['msalary']

        idproof = request.form['idproof']
        idnumber = request.form['idnumber']
        idupload = request.files['image1']

```



```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://kit.fontawesome.com/c7705f70af.js" crossorigin="anonymous"></script>
  <link rel="stylesheet" type="text/css" href="/static/css/login.css">
  <!-- <link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.5.2/css/bootstrap.min.css' /> -->
  <title>Login form</title>
</head>
{% with messages = get_flashed_messages() %}
{% if messages %}
  <ul class=flashes>
    {% for message in messages %}
      <script type="text/javascript">alert("{{ message }}")</script>
    {% endfor %}
  </ul>
{% endif %}
{% endwith %}
<div class="container" id="container">

  <div class="form-container sign-in-container">
    <form method="post">
      <h1>Sign in</h1>
      <div class="social-container">
        <a href="#" class="social"><i class="fab fa-facebook-f"></i></a>
        <a href="#" class="social"><i class="fab fa-google-plus-g"></i></a>
        <a href="#" class="social"><i class="fab fa-linkedin-in"></i></a>
      </div>
      <span>or use your account</span>
      <input type="username" name="username" placeholder="username" />
      <input type="password" name="password" placeholder="Password"
id="password"/>
      <a href="#">Forgot your password?</a>
      <input type="submit" name="submit" value="Login" class="button" />
    </form>
  </div>
  <div class="overlay-container">
    <div class="overlay">
      <div class="overlay-panel overlay-left">
        <h1>Welcome Back!</h1>
        <p>To keep connected with us please login with your personal info</p>
        <button class="ghost" id="signIn">Sign In</button>
      </div>
      <div class="overlay-panel overlay-right">
        <h1>Welcome Back!</h1>
        <p>To keep connected with us please login with your personal info</p>
        <button class="ghost" id="signUp">Read more</button>
      </div>
    </div>
  </div>

```

```

    </div>
</div>

<!-- <footer>
    <p>
        Created with <i class="fa fa-heart"></i> by
        <a target="_blank" href="https://florin-pop.com">Florin Pop</a>
        - Read how I created this and how you can join the challenge
        <a target="_blank" href="https://www.florin-pop.com/blog/2019/03/double-slider-sign-in-
up-form/">here</a>.
    </p>
</footer> -->

<br><br>
<footer>
    <div class="footer-container">
        <div class="footer-section">
            <h3>About Us</h3>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed tristique bibendum neque
vel viverra. Nam malesuada nisl at turpis ultrices, ut suscipit sapien hendrerit.</p>
        </div>
        <div class="footer-section">
            <h3>Links</h3>
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">About Us</a></li>
                <li><a href="#">Contact Us</a></li>
            </ul>
        </div>
        <div class="footer-section">
            <h3>Contact Us</h3>
            <p>123 Main St<br>City, State 12345<br>Phone: 555-555-5555<br>Email:
info@example.com</p>
            <div class="social-media-icons">
                <a href="#"><i class="fab fa-facebook-f"></i></a>
                <a href="#"><i class="fab fa-twitter"></i></a>
                <a href="#"><i class="fab fa-instagram"></i></a>
            </div>
        </div>
    </div>
    <div class="footer-bottom">
        <p>&copy; 2023 MyCompany. All Rights Reserved.</p>
    </div>
</footer>
<!-- <script src="/static/js/login.js"></script> -->
</body>
</html>

```

clerkaddcustomer.html

```
{% include 'clerksidebar.html' %}
```



```

<section class="middle">
<div class="header">
<h1>Dashboard</h1>
<input type="date">
</div>
<form method="post" enctype="multipart/form-data">
<section class="h-100 h-custom gradient-custom-2" style="width: 125%;">
<div class="container py-5 h-100">
<div class="row d-flex justify-content-center align-items-center h-100">
<div class="col-12">
<div class="card card-registration card-registration-2" style="border-radius: 15px;">
<div class="card-body p-0">
<div class="row g-0">
<div class="col-lg-6">
<div class="p-5">
<h3 class="fw-normal mb-5" style="color: #4835d4;">General Infomation</h3>

<div class="row">
<div class="col-md-6 mb-4 pb-2">

<div class="form-outline">
<input type="text" id="form3Examplev2" name="fname" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplev2">First name</label>
</div>

</div>
<div class="col-md-6 mb-4 pb-2">

<div class="form-outline">
<input type="text" id="form3Examplev3" name="lname" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplev3">Last name</label>
</div></div>
</div>

<div class="mb-4 pb-2">
<div class="form-outline">
<input type="date" id="form3Examplev4" name="dob" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplev4">Date of Birth</label>
</div>
</div>

<div class="mb-4 pb-2">

<div class="form-outline">
<input type="file" name="image" class="form-control form-control-lg" id="inputGroupFile01"
onchange="previewImage(event)">
<label for="inputGroupFile01" class="form-label">Choose photo</label>

</div>
</div>

```

```

<div class="mb-4">
<img id="image-preview" class="img-fluid" alt="Uploaded Image will shown below" style="width:
70%;">
</div>

<div class="mb-4 pb-3">
<select name="status" class="form-control form-control-lg p-2">
<option disabled selected value>Select martial status</option>
<option value="single">single</option>
<option value="married">married</option>
</select>
<label class="form-label" for="form3Examplev4">Status</label>
</div>

<div class="mb-4 pb-3">
<select name="gender" class="form-control form-control-lg p-2">
<option disabled selected value>Select gender</option>
<option value="male">male</option>
<option value="female">female</option>
</select>
<label class="form-label" for="form3Examplev4">Gender</label>
</div>
<div class="row">
<div class="col-md-5 mb-4 pb-2">

<div class="form-outline form-white">
<input type="text" id="form3Examplea7" class="form-control form-control-lg" value="91" readonly/>
<label class="form-label" for="form3Examplea7">Code +</label>
</div>

</div>

<div class="col-md-7 mb-4 pb-2">

<div class="form-outline form-white">
<input type="text" id="form3Examplea8" name="phone" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplea8">Phone Number</label>
</div>

</div>
</div>

<div class="mb-4 pb-2">
<div class="form-outline">
<input type="email" id="form3Examplev4" name="email" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplev4">Email</label>
</div>
</div>
<div class="mb-4 pb-2">
<div class="form-outline">
{% for branch in branch_names %}
<input type="text" id="form3Examplev4" name="branch" value="{{ branch }}" class="form-control

```

```

form-control-lg" readonly/>
<label class="form-label" for="form3Examplev4">branch</label>
{% endfor %}
</div>
</div>

</div>
</div>
<div class="col-lg-6 bg-indigo text-white">
<div class="p-5">
<h3 class="fw-normal mb-5">Details</h3>

<div class="mb-4 pb-2">
<div class="form-outline form-white">
<textarea name="address" id="form3Examplea2" class="form-control form-control-lg"></textarea>
<label class="form-label" for="form3Examplea2">Address</label>
</div>
</div>

<div class="row">
<div class="col-md-5 mb-4 pb-2">

<div class="form-outline form-white">
<input type="text" id="form3Examplea4" name="city" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplea4">city</label>
</div>

</div>
<div class="col-md-7 mb-4 pb-2">

<div class="form-outline form-white">
<input type="text" id="form3Examplea5" name="state" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplea5">state</label>
</div>

</div>
</div>

<div class="row">
<div class="col-md-5 mb-4 pb-2">

<div class="form-outline form-white">
<input type="text" id="form3Examplea4" name="zipcode" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplea4">zipcode</label>
</div>

</div>
<div class="col-md-7 mb-4 pb-2">

<div class="form-outline form-white">
<input type="text" id="form3Examplea5" name="country" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplea5">country</label>

```

```

</div>

</div>
</div>

<div class="mb-4 pb-3">
<select name="education" class="form-control form-control-lg p-2">
<option disabled selected value>Select your education</option>
<option value="high school">high school</option>
<option value="under graduate">under graduate</option>
<option value="Master degree">Master degree</option>
<option value="phd">phd</option>

</select>
<label class="form-label" for="form3Examplev4">education</label>
</div>

<div class="mb-4 pb-2">
<div class="form-outline">
<input type="number" id="form3Examplev4" name="msalary" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplev4">monthly salary</label>
</div>
</div>

<div class="mb-4 pb-3">
<select name="idproof" class="form-control form-control-lg p-2">
<option disabled selected value>Select your id</option>
<option value="adhar card">adhar card</option>
<option value="voter id">voter id</option>

<option value="driving lisence">driving lisence</option>
<option value="passport">passport</option>
<option value="student id">student id</option>

</select>
<label class="form-label" for="form3Examplev4">form of identification</label>
</div>

<div class="mb-4 pb-2">
<div class="form-outline">
<input type="number" id="form3Examplev4" name="idnumber" class="form-control form-control-lg" />
<label class="form-label" for="form3Examplev4">ID NUMBER</label>
</div>
</div>

<div class="mb-1 pb-2">

<label for="images" class="drop-container">
<i class="fas fa-cloud-upload-alt"></i>
<span class="drop-title">Drop files here</span>
or

```

```
<input type="file" name="image1" id="images" class="file1" required>

</label>
<label class="form-label" for="form3Examplev4">CHOOSE ID CARD TO UPLOAD</label>
</div>

<div class="form-outline">
<input type="file" name="image1" class="form-control form-control-lg" id="inputGroupFile01"
onchange="previewImage(event)">
<label for="inputGroupFile01" class="form-label">Choose photo</label>

</div>
</div>

<input type="submit" name="add" value="Add customer" class="btn btn-primary btn-lg">

</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</section>
</form>
</section>
</main>
</body>
</html>
```

7.2.3 CODE VALIDATION AND OPTIMIZATION

Validation is the process, whether we are building the right product i.e., to validate the product which we have developed is right or not. Activities involved in this is Testing the software application In simple words, Validation is to validate the actual and expected output of the software The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements It's a High-Level Activity. Validation is a dynamic process of testing the real product.

Validation is intended to ensure a product, service, or system (or portion thereof, or set thereof) results in a product, service, or system (or portion thereof, or set thereof) that meets the operational needs of the user. For a new development flow or verification flow, validation procedures may involve modeling either flow and using simulations to predict faults or gaps that might lead to invalid or incomplete verification or development of a product, service, or system (or portion thereof, or set

thereof).

A set of validation requirements (as defined by the user), specifications, and regulations may then be used as a basis for qualifying a development flow or verification flow for a product, service, or system (or portion thereof, or set thereof). Additional validation procedures also include those that are designed specifically to ensure that modifications made to an existing qualified development flow or verification flow will have the effect of producing a product, service, or system (or portion thereof, or set thereof) that meets the initial design requirements, specifications, and regulations; these validations help to keep the flow qualified. It is a process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance of fitness for purpose with end users and other product stakeholders. This is often an external process.

7.3DEBUGGING

Debugging, or tracking down the source of errors and erroneous result, is an important task that all developers need to perform before they allow end-user to use their applications.

The only effective way to eliminate logic error is to test every path of your application with every possible data values that a user could enter. This is difficult to manage without effective planning. We should create our test plan at the same time we are designing the application, and we should update these plans as you modify the application design.

7.4UNIT TESTING

Unit testing in the Bank Management System project is performed to ensure the accuracy and reliability of individual modules that constitute the system. The primary objective is to build a comprehensive bank management system with various functionalities, such as account opening, customer management, transaction processing, loan management, and financial reporting.

During unit testing, each module is examined independently to identify any errors and to validate the correctness of the applied logic and coding. The focus is on ensuring that each module functions correctly and meets the specific requirements outlined in the project objectives.

In addition to functionality, unit testing also verifies the adherence to standardized practices and design principles. This includes testing the proper tab order of fields, differentiating between capital and small letters, attaching scroll bars when necessary, displaying informative messages at the appropriate locations (such as the bottom of the screen), and maintaining consistent width and height of each field.

The unit testing process also considers the seamless integration of multiple screens. For example, when querying data from one block to populate another block, the unit tests ensure that the process is executed accurately, preserving data integrity and providing a smooth user experience.

Control blocks are thoroughly examined during unit testing to ensure their authenticity and compliance with established standards. This involves manual cross-checking to verify the correctness of the control block implementation.

Furthermore, unit testing pays attention to maintaining a consistent look and feel across the entire system. This involves standardizing aspects such as the height of text fields, the color scheme of forms, and the font usage to create a cohesive user interface.

By conducting comprehensive unit testing, the Bank Management System project aims to deliver a robust and reliable software application that meets the specific objectives of streamlining bank operations, enhancing customer experience, and enabling efficient management of various banking activities.

7.5 INTEGRATION TESTING

Integration testing plays a crucial role in the project of customer churn prediction at a bank. It is a phase in software testing where individual software modules are combined and tested as a group. Integration testing occurs after unit testing and before validation testing. The purpose of integration testing is to verify the functional, performance, and reliability requirements placed on major design items.

In the context of customer churn prediction, integration testing ensures that the different modules and functionalities of the system work together harmoniously when integrated. This includes modules such as data preprocessing, feature engineering, machine learning models, customer segmentation, and churn prediction algorithms. Integration testing validates that data flows correctly between these modules and that the integration points, data transfer, and communication between different modules are functioning as expected.

By conducting integration testing, potential issues and inconsistencies between the various components can be identified and resolved early on in the development process. It helps ensure the smooth operation and functionality of the entire churn prediction system, enhancing its overall reliability and performance. Integration testing also helps validate that the system meets the objectives of accurately predicting customer churn and providing actionable insights to the bank for customer retention strategies.

Overall, integration testing is an essential step in the customer churn prediction project, ensuring the seamless integration and proper functioning of the different components to achieve accurate and reliable churn predictions for the bank.

7.6 SYSTEM TESTING

System testing is a critical phase in the project of customer churn prediction at a bank. It is a type of software testing that evaluates the compliance of the complete integrated system with the corresponding requirements. In system testing, the integrated modules that have passed integration testing are taken as input. In the context of customer churn prediction, system testing ensures that the entire system, including modules such as data preprocessing, feature engineering, machine learning models, customer segmentation, and churn prediction algorithms, functions seamlessly and meets the desired requirements. The goal of system testing is to detect any defects or irregularities within the integrated units and the system as a whole. System testing is carried out based on the system requirement specifications and functional requirement specifications, or both. It tests the design, behavior, and customer expectations of the system. By performing system testing, potential issues such as incorrect data flow, functionality gaps, or compatibility problems can be identified and resolved before the system is deployed.

Additionally, system testing evaluates the performance, security, and usability of the customer churn prediction system. It ensures that the system delivers accurate churn predictions, effectively identifies customers at risk of churn, provides a user-friendly interface for data analysis and reporting, and facilitates smooth customer retention strategies.

Overall, system testing plays a vital role in guaranteeing the reliability, functionality, and quality of the customer churn prediction system at the bank, helping to achieve accurate predictions, optimize customer retention efforts, and enhance the overall customer experience.

7.6.1 TEST PLAN AND TEST CASES

A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specification and other requirements. A test plan is usually prepared by or with significant input from Test Engineers. A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can help find problems in the requirements or design of an application.

PROJECT TITLE: CUSTOMER CHURN PREDICTION AT BANK**Module: Admin**

Unit/sub module	Input	Expected output	Output obtained	Status	Reason	Remedy
Login	incorrect username or password	Login not allowed	Login successfully	failure	Validation not done correctly	Update the code
Add branch	Branch_id,branch_name	Added successfully	Added successfully	Success	Details entered correctly	Nil
Add clerk	employee_id, name	Added successfully	Added successfully	Success	Details entered correctly	Nil
Add Manager	employee_id, name	Added successfully	Added successfully	Success	Details entered correctly	Nil
View clerk	employee_id	clerk can be displayed	clerk can be displayed	Success	Selected correct button	Nil
View manager	employee_id	manager can be displayed	Manager can be displayed	Success	Selected correct button	Nil
View customer	Customer_id	customer cannot be displayed	Displayed the vehicle details	failure	Validation not done correctly	Update the code
View transaction	Transaction_id	Transaction details can be viewed	Transaction details can be viewed	Success	Selected correct button	Nil

View feedbacks	Feedback_id	Feedback viewed successfully	Feedback viewed successfully	Success	Selected correct button	Nil
View complaint & send reply	Compliant_id	Complaint viewed successfully	complaint viewed successfully	Success	Selected correct button	Nil

Module : manager

Unit/sub module	Input	Expected output	Output obtained	Status	Reason	Remedy
Login	Incorrect username or password	Login not allowed	Login successfully	Failure	Validation not done correctly	Update the code
View clerk	Employee_id	clerk can be displayed	clerk can be displayed	Success	Select correct button	Nil
View customers	Customer_id	customer can be displayed	customer can be displayed	Success	Select correct button	Nil
View transaction	Transaction_id, customer_id	Transaction details can be viewed	Transaction details cannot be viewed	failure	Validation not done correctly	Update the code
View loan Payments	Payment_id	Loan payment can be viewed	Loan payment can be viewed	Success	Select correct button	Nil
View credit card request	Credit_card_id	Request can be viewed	Request can be viewed	success	Select correct button	Nil
Customer churn prediction	Customer_id	Show prediction result	Show prediction result	success	Select correct button	Nil

Module : clerk

Unit/sub module	Input	Expected output	Output obtained	Status	Reason	Remedy
Login	incorrect username or password	Login not allowed	Login successfully	failure	Validation not done correctly	Update the code
View customer	Customer_id	customer can be displayed	customer can be displayed	Success	Select correct button	Nil
Add customer	Customer_id	Added successfully	Not added successfully	failure	Validation not done correctly	Update the code
Create account	Customer_id, account_id	Added successfully	added successfully	success	Details entered correctly	Nil
Make transaction	Customer_id, amount	Transaction successfull	Transaction not successfull	failure	Validation not done correctly	Update the code

Module : customers

Unit/sub module	Input	Expected output	Output obtained	Status	Reason	Remedy
Login	incorrect username or password	Login not allowed	Login successfully	failure	Validation not done correctly	Update the code
Transfer cash	Transaction_id, amount	Amount send successfully	Amount send successfully	Success	Details entered correctly	Nil

Request credit card	Credit_request_id	Requested successfully	Requested successfully	Success	Selected correct button	Nil
Make loan payment	Payment_id, amount	Payment success	Payment success	Success	Selected correct button	Nil
View transaction	Transaction_id	Transaction details can be viewed	Transaction details cannot be viewed	failure	Validation not done correctly	Update the code
Send feedbacks	Feedback_id	Feedback send successfully	Feedback send successfully	Success	Selected correct button	Nil
Send complaint & view reply	complaint_id	Complaint send successfully	Complaint not send successfully	failure	Validation not done correctly	Update the code

8. SYTEM MAINTENANCE

8.1 INTRODUCTION

The maintenance is an important phase of any system. Maintenance of the System should be done accurately and with specific care for proper running system. Maintenance involves the software industry captive, typing up the system resources. It means restoring something to its original condition. Maintenance involves a wide range of activities including correcting, coding and design errors, updating documentation and test data and upgrading user support. Maintenance is continued till the product is re-engineered or deployed to another platform. Maintenance is also done based on fixing the problems reported, changing the interface with other software or hardware enhancing the software.

8.2 MAINTENANCE

Software maintenance is the modification of a software product after delivery to correct faults, to improve performance or other attributes. Maintenance is the ease with which a program can be corrected if any error is encountered, adapted if its environment changes or enhanced if the customer desires a change in requirement. Maintenance follows conversation to extend that changes are necessary to maintain satisfactory operations relative to changes in the user's environment. Maintenance often includes minor enhancements or corrections to problems that surface in the system's operation. Maintenance is also done based on fixing the problems reported, changing the interface with other software or hardware enhancing the software.

Any system developed should be secured and protected against possible hazards. The system should be maintained and upgraded according to technological advancements. It ensures data integrity, data control and security. The system must be protected from fire and other natural calamities. The backup copies of the data must be maintained daily so that to prevent the loss of data due to various reasons. Security measures provided to prevent unauthorized access of the database at various levels.

Software maintenance is divided into the following three categories:-

- corrective maintenance
- Adaptive maintenance
- perfect maintenance

Corrective maintenance has to do with the removal of residual errors present in the product when it is delivered as well as errors introduced into the software during its maintenance. Adaptive maintenance modifies the software to keep it up to date with its operative environment. It may be needed because of changes in the user requirements, changes in target platforms, or changes in external interfaces. Perfective maintenance involves changing the software to improve some of its qualities. The request to perfective maintenance may come directly from the software engineer, in order to improve the status of the product on the market, or they may come.

9. SYSTEM SECURITY MEASSURES

9.1 INTRODUCTION

The security of a computer system is a crucial task. It is a process of ensuring the confidentiality and integrity of the OS. Security is one of most important as well as the major task in order to keep all the threats or other malicious tasks or attacks or program away from the computer's software system. A system is said to be secure if its resources are used and accessed as intended under all the circumstances, but no system can guarantee absolute security from several of various malicious threats and unauthorized access.

The security of a system can be threatened via two violations:

Threat: A program that has the potential to cause serious damage to the system.

Attack: An attempt to break security and make unauthorized use of an asset.

Security violations affecting the system can be categorized as malicious and accidental threats. Malicious threats, as the name suggests are a kind of harmful computer code or web script designed to create system vulnerabilities leading to back doors and security breaches. Accidental Threats, on the other hand, are comparatively easier to be protected against. Example: Denial of Service DDoS attack.

Security can be compromised via any of the breaches mentioned:

Breach of confidentiality: This type of violation involves the unauthorized reading of data.

Breach of integrity: This violation involves unauthorized modification of data.

Breach of availability: It involves unauthorized destruction of data.

Theft of service: It involves the unauthorized use of resources.

Denial of service: It involves preventing legitimate use of the system. As mentioned before, such attacks can be accidental in nature.

Henceforth, based on the above breaches, the following security goals are aimed:

Integrity- The objects in the system mustn't be accessed by any unauthorized user & any user not having sufficient rights should not be allowed to modify the important system files and resources.

Secrecy- The objects of the system must be accessible only to a limited number of authorized users. Not everyone should be able to view the system files.

Availability - All the resources of the system must be accessible to all the authorized users

9.2. OPERATING SYSTEM LEVEL SECURITY

Operating system security (OS security) is the process of ensuring OS integrity, confidentiality and availability. OS security refers to specified steps or measures used to protect the OS from threats, viruses, worms, malware or remote hacker intrusions. OS security encompasses all preventive-control techniques, which safeguard any computer assets capable of being stolen, edited or deleted if OS security is compromised. OS security encompasses many different techniques and methods which ensure safety from threats and attacks. OS security allows different applications and programs to perform required tasks and stop unauthorized interference. OS security may be approached in many ways, including adherence to the following:

- Performing regular OS patch updates
- Installing updated antivirus engines and software
- Scrutinizing all incoming and outgoing network traffic through a firewall
- Creating secure accounts with required privileges only (i.e., user management)

9.3. DATABASE LEVEL SECURITY

Database security refers to the various measures organizations take to ensure their databases are protected from internal and external threats. Database security includes protecting the database itself, the data it contains, its database management system, and the various applications that access it. Organizations must secure databases from deliberate attacks such as cyber security threats, as well as the misuse of data and databases from those who can access them.

In the last several years, the number of data breaches has risen considerably. In addition to the considerable damage these threats pose to a company's reputation and customer base, there are an increasing number of regulations and penalties for data breaches that organizations must deal with, such as those in the General Data Protection Regulation (GDPR)—some of which are extremely costly. Effective database security is key for remaining compliant, protecting organizations' reputations, and keeping their customers.

Security concerns for internet-based attacks are some of the most persistent challenges to database security. Hackers devise new ways to infiltrate databases and steal data almost daily. Organizations must ensure their database security measures are strong enough to withstand these attacks.

Some of these cyber security threats can be difficult to detect, like phishing scams in which user credentials are compromised and used without permission. Malware and ransomware are also common cyber security threats.

Another critical challenge for database security is making sure employees, partners, and contractors with database access don't abuse their credentials. These exfiltration vulnerabilities are difficult to guard against because users with legitimate access can take data for their own purposes. Edward Snowden's compromise of the NSA is the best example of this challenge. Organizations must also make sure users with legitimate access to database systems and applications are only privy to the information they need for work. Otherwise, there's greater potential for them to compromise database security.

There are three layers of database security: the database level, the access level, and the perimeter level. Security at the database level occurs within the database itself, where the data live. Access layer security focuses on controlling who is allowed to access certain data or systems containing it. Database security at the perimeter level determines who can and cannot get into databases. Each level requires unique security solutions.

9.4 SYSTEM-LEVEL SECURITY

System-level security refers to the architecture, policy and processes that ensure data and system security on individual computer systems. It facilitates the security of standalone and/or network computer systems/servers from events and processes that can exploit or violate its security or stature. System-level security is part of a multi-layered security approach in which information security (IS) is implemented on an IT infrastructure's different components, layers or levels. System-level security is typically implemented on end-user computer and server nodes.

It ensures that system access is granted only to legitimate and trusted individuals and applications. The key objective behind system-level security is to keep system secure, regardless of security policies and processes at other levels. If other layers or levels are breached, the system must have the ability to protect itself. Methods used to implement system-level security are user/ID login credentials, antivirus and system-level firewall applications.

10. FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT

10.1 INTRODUCTION

Many applications (and wider database analytics systems) will go through what the industry likes to call the software application development lifecycle (SDLC) when they are first built. The SDLC describes the process of planning, analysis, design, build, test and implementation. But the lifecycle has further steps into maintenance, enhancement, and progression. Vendors are now appearing that are specifically describing themselves as application enhancement and maintenance specialists. User demands today change so rapidly, it makes sense for organizations to look for ways to extend functionality rather than go through rip-and-replace cycles, which are, ultimately, always costly.

The future enhancement and scope for further development of the customer churn prediction system at the bank hold immense potential for improving its effectiveness and delivering additional value. Some key areas for enhancement include incorporating enhanced feature engineering techniques to capture more comprehensive customer insights, exploring advanced machine learning algorithms for improved accuracy, developing real-time churn monitoring capabilities, implementing automated retention strategies, refining customer segmentation and targeting, integrating with CRM systems, establishing robust performance monitoring mechanisms, expanding to new channels and platforms, integrating with feedback and complaint systems, and fostering collaboration between data scientists and business units. These enhancements will enable the bank to proactively identify churn risks, personalize retention strategies, optimize customer engagement, and ultimately improve customer satisfaction, retention rates, and profitability. Regular updates and adaptations to changing customer behaviors and market dynamics are essential for sustained success.

10.2 MERITS OF THE SYSTEM

The system for customer churn prediction offers several merits that can benefit users and stakeholders:

1. **Improved Customer Retention:** The customer churn prediction system enables the bank to identify customers who are likely to churn in advance. By implementing targeted retention strategies, such as personalized offers, proactive customer service, and tailored engagement campaigns, the bank can effectively reduce customer attrition and improve overall customer retention rates.
2. **Cost Savings:** By focusing on retaining existing customers, the bank can minimize the costs associated with acquiring new customers. Customer acquisition can be an expensive process, involving marketing campaigns, lead generation, and onboarding efforts. The churn prediction

system helps allocate resources more efficiently by prioritizing retention efforts and reducing the need for costly acquisition activities.

3. **Enhanced Customer Experience:** The system enhances the overall customer experience by enabling the bank to understand customer preferences, needs, and behaviors. With this information, the bank can personalize interactions, offer relevant products and services, and provide proactive support, ultimately leading to higher customer satisfaction and loyalty.
4. **Time and Cost Savings:** The customer churn prediction system provides significant time and cost savings for the bank. By accurately identifying customers at risk of churn, the bank can focus its resources on targeted retention efforts, rather than investing time and money in acquiring new customers. This leads to more efficient resource allocation and cost-effective customer management.
5. **Improved Decision-Making:** The system enhances the bank's decision-making capabilities by providing valuable insights into customer behavior and churn patterns. With access to accurate churn predictions and related customer data, the bank can make informed decisions on retention strategies, product offerings, and customer engagement initiatives. This leads to more effective decision-making and better outcomes for the bank.
6. **User-Friendly Interface:** The customer churn prediction system features a user-friendly interface that is easy to navigate and understand. It allows bank personnel to access and analyze churn-related information effortlessly. This intuitive interface enables quick identification of at-risk customers and facilitates the implementation of targeted retention measures. The user-friendly nature of the system enhances user satisfaction and promotes efficient utilization of the system.

10.3 LIMITATIONS OF THE SYSTEM

1. **Data Availability and Quality:** The accuracy and effectiveness of the churn prediction system heavily rely on the availability and quality of data. Limited or inconsistent data, incomplete customer profiles, and data privacy constraints can pose challenges in accurately predicting churn and implementing effective retention strategies. Ensuring data completeness, accuracy, and privacy compliance is essential for the system's success.
2. **Model Accuracy and Adaptability:** While the churn prediction model may demonstrate a high level of accuracy in predicting churn for existing customers, it may face challenges when applied to new or evolving customer segments. Changes in customer behavior, market dynamics, or external factors may require continuous monitoring and adjustment of the model to maintain its effectiveness.
3. **Retention Strategy Implementation:** The system's effectiveness also depends on the bank's ability to effectively implement retention strategies based on the churn predictions. Challenges may arise

in coordinating and executing personalized retention efforts, ensuring timely and relevant interventions, and measuring the impact of these strategies on customer retention.

10.4 FUTURE ENHANCEMENT OF THE SYSTEM

potential future enhancement for the system is to implement a feedback loop that continuously collects user feedback and transaction data. By leveraging this feedback and data, the system can refine its predictive algorithms, improve accuracy over time, and provide personalized recommendations based on user preferences and historical patterns. This enhancement would enable the system to adapt and evolve with changing market dynamics and user needs, ultimately enhancing the user experience and the effectiveness of the churn prediction capabilities.

11.ANNEXURE

11.1 ORGANIZATION PROFILE

KMM college of Arts and Science is a part of Jai Bharath Educational Foundation established in 2002.

KMM college is an institute of higher education located in Thrikkakkara, Kochi in Ernakulam district of Kerala. The college is affiliated to Mahatma Gandhi University. KMM College of Arts & Science offers Under Graduate Programmes in B.Com. (Computer Application/Taxation) | BBA | BCA | BSc.

(Computer Science/Mathematics) | BSW | B.A Communicative English & Post Graduate Programmes in M.Com. (Finance/ E- Commerce & Banking), BSc Apparel and Fashion Design, MA English & M.Sc. Mathematics, MCA, MBA.

KMM College of Arts & Science strives to create a future society where ignorance, inequality, ill-health, illiteracy, poverty, and powerlessness can be eradicated and where equitable development can take place providing equal access to education to all members of the society. The institution believes in the strong linkage of training, education, research, and action for sustainable development. Education is the key to enhance knowledge that creates both opportunities as well as ability to make right choices. KMM is an institution committed to excellence and our objective is to produce students with the ability to think critically, rationally, and communicate effectively. We strive to create a stimulating learning environment for a truly academic community of students, teachers, and professionals. The college provides exciting facilities and opportunities to enable the youngsters to work and live in a variety of settings within and outside the country.

11.2 DOCUMENT GLOSSARY

- SDS :System Design Specifications
- DB :Database
- UML :Unified Modelling Language
- SQL :Structured Query Language
- ER - Entity Relationship Diagram
- OS - Operating System
- UML - Unified Modeling Language
- HTML - Hypertext Markup Language

11.2 FIGURES , TABLES

LIST OF FIGURES

Figure	Title	Page No
1	ER Diagram	13
2	Activity Diagram of admin	26
3	Activity Diagram of customer	27
4	Activity Diagram of clerk	27
5	Activity Diagram of manager	29
6	Class Diagram	29
7	Component Diagram	31
8	Object Diagram	33
9	Package Diagram	35
10	Sequence Diagram	37
11	Structure chart of Admin	39
12	Structure chart of customer	40
13	Structure chart of manager	41
14	Structure chart of clerk	42
15	GAANT Chart	60
16	Pert	62

LIST OF TABLES

Table No	Title	Page No
1	login	14
2	employee	14
3	customer	15
4	branch	16
5	savingsacc	17
6	depositacc	17
7	Loanacc	18
8	loanpayment	19
9	loanborrower	19
10	cheque	20
11	complaint	20
12	debitcard	21
13	creditcard	21
14	transaction	22
15	bankproduct	22
16	notescount	22
17	feedback	23

11.4 REFERENCES

WEBSITE

- <http://www.tutorialspoint.com>
- <http://www.codeacademy.com>
- <https://en.wikipedia.org/wiki/Feasibilitystudy>
- <https://en.wikipedia.org/wiki/Systemtesting>
- <https://www.mysql.com>