**A**

***PROJECT REPORT***

*on*

**Bank Management System**

*Submitted in partial fulfilment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

****

Session: - Jan-June 2024

Submitted by

Akshat Audichya (20ETCCS005)

Kanishk Asawara (20ETCCS059)

8th Semester CSE

Under Guidance of

Aaditya Maheshwari

Assistant Professor

Computer science

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001**

**MAY - 2024**

**CONTENTS**

**A**

***PROJECT REPORT***

*on*

**Bank Management System**

*Submitted in partial fulfilment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**



**Session: - Jan-June 2024**

Submitted by

Akshat Audichya (20ETCCS005)

Kansihk Asawara (20ETCCS059)

8th Semester

Under Guidance of

Aaditya Maheshwari

Assistant Professor

Computer Science

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001**

**MAY - 2024**



Department of Computer Science and Engineering

Techno India NJR Institute of Technology, Udaipur-313001

**Certificate**

This is to certify that project work titled Bank Management System by **Akshat Audichya** as successfully carried out in the Department of Computer Science and Engineering, TINJRIT and the report is approved for submission in the partial fulfillment of the requirements for award of degree of Bachelor of Technology in Computer Science and Engineering.

Akhilesh Arya Aaditya Maheshwari

Assistant Professor Assistant Professor

Dept. of CSE TINJRIT, Udaipur Dept. of CSE TINJRIT, Udaipur

Date...................... Date......................



Department of Computer Science and Engineering

Techno India NJR Institute of Technology, Udaipur-313001

**Certificate**

This is to certify that project work titled Bank Management System by **Kanishk Asawara** as successfully carried out in the Department of Computer Science and Engineering, TINJRIT and the report is approved for submission in the partial fulfillment of the requirements for award of degree of Bachelor of Technology in Computer Science and Engineering.

Akhilesh Arya Aaditya Maheshwari

Assistant Professor Assistant Professor

Dept. of CSE TINJRIT, Udaipur Dept. of CSE TINJRIT, Udaipur

Date...................... Date......................



Department of Computer Science and Engineering

Techno India NJR Institute of Technology, Udaipur-313001

**Examiner Certificate**

This is to certify that the following students **Akshat Audichya, Kanishk Asawara** of final year B.Tech. (Computer Science and Engineering), was examined for the project work titled ***Bank Management System*** during the academic year 2023 – 2024 at Techno India NJR Institute of Technology, Udaipur

**Remarks:**

**Date:**

Signature Signature

(**Internal Examiner**) (**External Examiner**)

Name :- ……………………… Name :- ………………………

Designation:- ……………….. Designation:- ………………..

Department: - ………………. Department: - ……………….

Organization:- ……………… Organization:- ………………

**Preface**

The financial sector thrives on efficiency and security. In today's digital landscape, banks require robust systems to manage complex financial transactions and vast amounts of customer data. This project explores the development of a comprehensive bank system using SAP's ABAP (Advanced Business Application Programming) platform.

This document details the design, development, and functionalities of the created bank system. It aims to demonstrate the power and flexibility of ABAP in building a secure and scalable banking solution.

This preface is followed by sections that delve into:

* **Project Scope:** A clear outline of the functionalities and features incorporated into the bank system.
* **System Design:** A detailed explanation of the system architecture, data structures, and chosen ABAP development tools.
* **Development Process:** A step-by-step walkthrough of the development process, including coding standards and testing procedures.
* **Functionalities:** A comprehensive overview of the core functionalities offered by the bank system, such as account management, transaction processing, and reporting.
* **Security Considerations:** A discussion on the security measures implemented within the system to protect sensitive financial data.
* **Conclusion:** A summary of the project's achievements, potential limitations, and future enhancements.

This project serves as a valuable resource for those interested in leveraging SAP ABAP for developing banking solutions. It showcases the platform's capabilities and paves the way for further exploration and customization to cater to specific banking needs.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001

**ACKNOWLEDGMENT**

We take this opportunity to record our sincere thanks to all who helped us to successfully complete this work. Firstly, We are grateful to our supervisor **Kutubuddin Bohra** for his invaluable guidance and constant encouragement, support and most importantly for giving us the opportunity to carry out this work.

We would like to express our deepest sense of gratitude and humble regards to our

**Head of Department Dr. Rimpy Bishnoi** for giving invariable encouragement in our endeavors and providing necessary facility for the same. Also a sincere thanks to all faculty members of CSE, TINJRIT for their help in the project directly or indirectly.

Finally, We would like to thank my friends for their support and discussions that have proved very valuable for us. We are indebted to our parents for providing constant support, love and encouragement. We thank them for the sacrifices they made so that we could grow up in a learning environment. They have always stood by us in everything we have done, providing constant support, encouragement and love

**Akshat Audichya (20ETCCS005)**

**Kanishk Asawara (20ETCCS059)**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Abstract** | ………………………………………………………………. | i |
| **Contents** | ………………………………………………………………. | ii – iii |
| **List of Figures** ……………………………………………………............ | | iv |
| **Chapter 1** | INTRODUCTION…………………………………………... | 1-6 |
|  | * 1. Existing system …..……….……………………………. | 2 |
|  | 1.2 Propose System…………………………………………. | 3 |
|  | 1.3 Requirements…………………………………………… | 4 |
| **Chapter 2** | WORKING…………………………………………………. | 7-12 |
|  | * 1. Core Functionalities…………..………………………... | 7 |
|  | * 1. Processes……………………...………………………… | 7 |
|  | * 1. Code Overview………….……………………………… | 8 |
|  | 2.3.1 Top block.………………………………………… | 8 |
|  | 2.3.2 Event Block…….....……………………………… | 9 |
|  | 2.3.3 Subroutine Block....………….…………………… | 11 |
| **Chapter 3** | PROJECT OVERVIEW..………………………………………. | 13-18 |
|  | * 1. Insight of the Project………….……………………………. | 13 |
|  | * 1. Existing system …..……….………………………………. | 15 |
|  | * 1. Limitations of System……………………………………… | 15 |
|  | * 1. Future Scope……………………………………………….. | 16 |
|  | * 1. Data Dictionary……………………………………………. | 17 |
| **Chapter 4** | PROGRAM…………………………………………………. | 19-24 |
|  | 4.1 Top Block of Program…………..………………………... | 19 |
|  | 4.2 Event Block of Program………..………………………... | 20 |
|  | 4.3 Subroutine Block of Program.…..………………………... | 23 |
| **Chapter 5** | CONCLUSION..………………………………………………. | 25 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Fig 2.1 | Event Block Working….…………………………………....... | 10 |
| Fig 2.2 | Subroutine Block Working….……………………………....... | 12 |
| Fig 3.1 | Creating Account…………………………………………….. | 13 |
| Fig 3.2 | Delete Transaction……………………………………………. | 13 |
| Fig 3.3 | Alter Transaction…………………………………………….. | 14 |
| Fig 3.4 | Display Transaction………………………………………….. | 14 |
| Fig 3.5 | Data Dictionary Diagram…………………………………….. | 18 |

**Chapter 1: INTRODUCTION**

The "Bank Management System" is a software application designed to facilitate the management of accounts within a bank. Its primary purpose is to address the financial needs of customers in a banking environment, offering various features to perform banking tasks efficiently. Moreover, it aims to enhance the user experience by providing functionalities beyond those typically found in conventional banking software.

Here's a detailed breakdown of the key aspects:

**1. Addressing Financial Needs**: The software is developed to cater to the financial requirements of customers in a banking environment. This includes tasks such as account management, depositing money, withdrawing funds, and viewing account reports. By providing these services, the system aims to streamline banking operations and enhance customer satisfaction.

**2. Software Development Objective:** The primary objective of developing this software is to create a solution for managing financial applications within a banking context. It seeks to meet the diverse needs of end-users by offering a range of functionalities tailored to banking tasks.

**3. Enhancing User Workspace:** The software intends to improve the user workspace by introducing additional functionalities not typically available in traditional banking software. This could involve features such as advanced account management options, personalized settings, and user-friendly interfaces. By enhancing the workspace, the system aims to provide a more intuitive and efficient banking experience for customers.

**4. Project Scope**: The project encompasses various components, including account opening, depositing money, and withdrawing funds. These functionalities are essential for managing accounts within a bank and are therefore included in the system's scope. Additionally, the system allows users to view reports of all accounts present, enabling them to monitor their financial activities effectively.

**5. System Specification:** The presentation outlines the specifications for the system, detailing its features and functionalities. This includes information on account creation processes, deposit and withdrawal procedures, and report generation capabilities. By providing a comprehensive overview of the system's specifications, stakeholders can gain a clear understanding of its capabilities and functionalities.

In summary, the "Bank Management System" is a software application designed to address the financial needs of customers in a banking environment. It offers a range of functionalities for managing accounts, depositing and withdrawing funds, and generating account reports. Moreover, it aims to enhance the user experience by providing additional features not typically found in traditional banking software.

* 1. **EXISTING SYSTEM:**

The existing system work manually. The existing system has got lot of intricacies within itself and need lot of human effort and paper works. All above the data need to be maintained on ledgers and maintaining this is a tedious and risky process. As the transactions increases, so the data too. So the task of maintaining them increases exponentially. To view a data may need lot of paper to be searched.

**Some of the negative aspects of the existing system are as follows:**

1. **Time Consuming:**

There is a lot of time consumes in the bank, whenever we open an account, deposit, withdraw or pass a loan than because of many customers with his/ her different purpose, than we wait for our turn sometimes 2 to 3 hours.

1. **Reliability:**

This banking system is not fully reliable whenever the computer system is create a problem and not work properly than sometime our data is damaged or lost.

1. **Man Power:**

In this project man power is fully used. A number of employee need to manage the banking system.

1. **Less Accurate:** This system is not fully accurate, because sometime computer system create a problem in working, than the computer system also give us wrong results.

**To overcome these, the proposed system has been suggested.**

**1.2 PROPOSED SYSTEM:**

In this project we are going to explain about **Banking Management System.** This project have facility to opening account, depositing and withdrawing money. The proposed system is a computerized one. This has greater accuracy and efficiency. This takes only limited time for calculation.

The proposed system can be used to maintain efficiently the BANKING schedule. In larger organizations employees are large. At that time also the proposed system is useful and helpful. The system includes users Administrator(HR) level.

In this project we have a admin login feature, we want to fill the admin and password then we enter to home page, The home page facility show all the include features about this banking project. We have a number of options like open account, delete account, modify account, add employee record, delete employee record, modify employee record, deposit money, we can easily choose any option according to our own requirement. We have also feature of validation.

We also take view of all the account list information according to date. We have also information record about this bank and its director.

**1.3 REQUIREMENTS:**

* Functional Requirements
* Non- Functional Requirements
* System Requirements

Functional requirements:

* **SAP Client Installation:** The SAP GUI (Graphical User Interface) client software needs to be installed on your computer.
* **Network Connectivity:** A stable network connection is required to connect to the SAP server where the application resides.
* **Valid User ID and Password:** You'll need a unique ID and password assigned by the system administrator to access SAP.
* **Authorization:** Your user ID will have specific permissions for tasks and data access within the system.
* **Understanding of Relevant SAP Modules:** SAP offers a wide range of modules like Finance (FICO), Human Resources (HCM), etc. You'll need basic knowledge of the specific module(s) you'll be using.
* **Knowledge of Business Processes:** Familiarity with the business processes supported by the SAP modules you'll be working with is helpful.

Non-Functional requirements:

* **Security:** This is paramount. SAP systems handle sensitive business data, so robust security measures are crucial. This includes user access controls, data encryption, and regular security audits.
* **Performance:** SAP systems should be responsive and handle large data volumes efficiently. Response times for transactions and reports should meet user expectations.
* **Availability:** Minimizing downtime is essential. SAP systems should be highly available to ensure business continuity. This involves disaster recovery plans and system redundancy.
* **Scalability:** The system should adapt to changing needs. As a business grows, the SAP system should be able to handle increased data volumes and users.
* **Maintainability:** The system should be easy to maintain and update. This includes clear code documentation and user-friendly tools for system administrators.
* **Usability:** The user interface should be intuitive and easy to learn for personnel with varying levels of SAP expertise.
* **Compliance:** The system should comply with relevant industry regulations and data privacy laws.

System Requirements:

* **CPU:** A modern processor with multiple cores is recommended for efficient handling of complex tasks.
* **RAM:** The amount of RAM needed depends on the specific SAP modules used and the number of users, but generally 4GB is a minimum and more RAM improves performance.
* **Storage:** Enough hard disk space is needed to store the SAP software, user data, and temporary files. The exact amount depends on the specific modules used.
* **Network:** A reliable network connection is essential for communication between users and the SAP system.
* **Operating System:** The specific operating system supported will depend on the SAP product and version. SAP generally supports major operating systems like Windows and Linux.
* **Database:** SAP systems rely on a database to store data. SAP offers its own database (SAP HANA) but other databases like Oracle are also supported.
* **SAP Software:** This includes the specific SAP modules your

organization requires, and the SAP GUI (graphical user interface) for users to interact with the system.

* **SAP Business One:** This version of SAP is designed for small and medium businesses and has lower system requirements than the larger SAP ERP systems.
* **Virtualization:** SAP systems can be deployed on virtual machines, which can help to optimize resource utilization.
* **Cloud deployment:** SAP offers cloud-based deployments of its software, which can alleviate some of the hardware and software management burden for organizations.

**Chapter 2: WORKING**

This section outlines the core functionalities and processes within the developed bank system built using SAP ABAP.

### **2.1 CORE FUNCTIONALITIES**

The bank system caters to various banking operations, including:

* **Account Management:**
  + Create new accounts (savings, current, etc.)
  + Manage customer details and account information
  + Close existing accounts
* **Transaction Processing:**
  + Deposit and withdrawal functionalities
  + Fund transfers between accounts (internal and potentially external)
  + Bill payments and loan repayments
  + Integration with external payment gateways (if applicable)
* **Customer Management:**
  + Maintain customer profiles and contact information
  + Manage user access and security permissions
* **Reporting:**
  + Generate account statements and transaction history
  + Create reports on overall bank activity (balances, deposits, withdrawals, etc.)
  + Allow for customized reports based on user needs
* **Security:**
  + User authentication and authorization controls
  + Data encryption for sensitive information
  + Audit trails to track user activity

### **2.2 PROCESSES**

The functionalities mentioned above are achieved through a series of integrated processes within the ABAP system:

1. **User Interaction:** Users (customers and bank staff) interact with the system through a user-friendly interface (SAP GUI or web interface, depending on the implementation).
2. **Data Capture and Validation:** User input for transactions, account modifications, etc., is captured and validated for accuracy and completeness.
3. **ABAP Processing:** ABAP programs handle the core functionalities. They interact with the database to perform actions like creating accounts, processing transactions, generating reports, etc.
4. **Database Interaction:** The system utilizes a database (SAP HANA or a compatible database) to store all account information, transaction history, and other relevant data.
5. **Security Checks:** Throughout the process, security measures like user authentication, authorization checks, and data encryption safeguard sensitive information.
6. **Output and Reporting:** The system generates appropriate outputs based on user actions. This can include confirmations for successful transactions, account statements, or comprehensive reports.

**2.3 CODE OVERVIEW**

The overall programming in building this project is done in module pool where the code is divided into three sections

1. **TOP BLOCK:** The top section of a Module Pool program in SAP ABAP primarily consists of two key elements:
   1. **Screen Painter Definition:** This defines the visual layout of the screen using a graphical interface. You won't directly write code in this section, but rather use tools to position UI elements like:
      1. **Input Fields:** Where users enter data (text, numbers, dates, etc.).
      2. **Output Fields:** Display pre-defined or calculated information.
      3. **Pushbuttons:** Trigger specific actions within the program.
      4. **Checkboxes and Radio Buttons:** Allow users to select options.
      5. **Text Boxes:** Display static information or instructions.
      6. **Tables:** Display rows and columns of data.
   2. **Screen Attributes:** This section defines overall properties of the screen using ABAP code within the Module Pool program. These attributes typically appear at the beginning of the program and control the screen's behavior. Here are some important attributes:
      1. **Screen Number:** A unique identifier for the screen (e.g., 100).
      2. **Screen Title:** The title displayed at the top of the screen.
      3. **Screen Type:** Defines the screen behavior (e.g., dialog box or modal window).
      4. **Next Screen:** Specifies the screen to navigate to after completing this screen (optional).
      5. **Program Name:** Links the screen to the ABAP program that handles its logic.
2. **EVENT BLOCK:** It is the crucial section of code that control the program's behavior in response to user interaction with the screen elements. These blocks are placed strategically within the program to execute at specific points during the screen's lifecycle.

There are two main event blocks defined within a Module Pool:

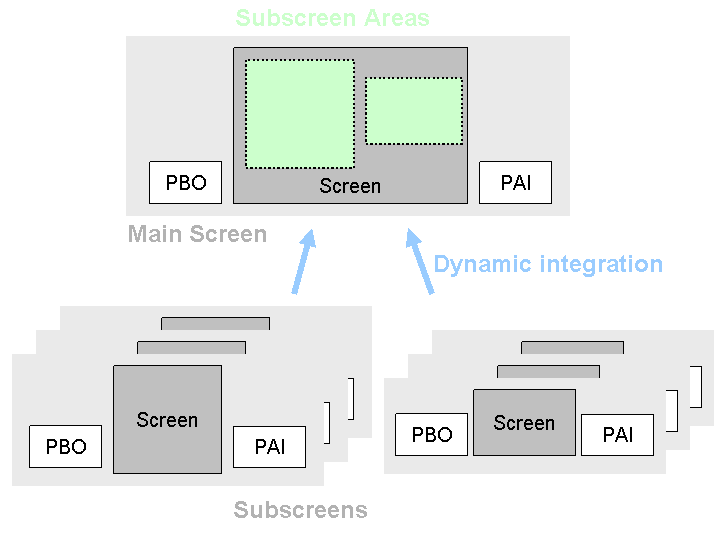
* 1. **PROCESS BEFORE OUTPUT (PBO):**

**2.1.1 Execution:** This block executes **before** the screen is displayed to the user.

* + 1. **Purpose:** This is typically used for:
    - **Initializing screen elements:** Setting default values for input fields, hiding or showing specific elements based on conditions, etc.
    - **Fetching data:** Retrieving data from the database or other sources to populate the screen before display (e.g., displaying account balance information based on a customer ID).
  1. **PROCESS AFTER INPUT (PAI):**

**2.2.1 Execution:** This is the most critical event block. It executes **after** the user interacts with the screen elements (e.g., pressing a button, entering data, selecting an option).

* + 1. **Purpose:** This is where the core logic of the program is implemented based on user input. Here you can:
    2. **Validate user input:** Ensure data entered by the user meets expected criteria (e.g., checking if entered amount is a valid number and within acceptable limits).
    3. **Perform calculations:** Based on user input, perform calculations (e.g., calculating interest on a loan).
    4. **Update database records:** Update relevant database tables based on user actions (e.g., recording a new account transaction).
    5. **Navigate to other screens:** Control the flow of the program and potentially navigate to different screens within the module pool.



**Fig: 2.1**

1. **SUBROUTINE BLOCK:** In SAP ABAP, a subroutine block within a module pool is a structured programming construct used to organize and encapsulate reusable code within a larger program. Here's a detailed explanation:

**3.1. Purpose of Subroutine Blocks:** Subroutine blocks are used to group related functionality together within a module pool program. This promotes modular programming, making the code easier to understand, maintain, and debug. By encapsulating specific tasks or processes within subroutines, developers can achieve better code organization and reusability. The `subroutine\_name` is the name of the subroutine block, and `[OUTPUT]` is an optional keyword used to specify that the subroutine can return values.

**3.2. Parameters:** Subroutines can accept parameters just like ABAP functions or methods. Parameters are declared within the `FORM` statement and can be used to pass data into the subroutine or receive data from it.

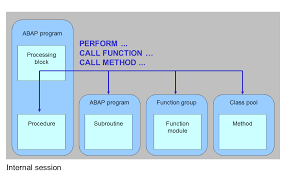
**3.3. Calling Subroutines:** Subroutines are called using the `PERFORM` statement followed by the name of the subroutine block. If parameters are defined, they are passed within parentheses.

**3.4. Scoping:** Subroutines have their own local data scope, meaning variables declared within a subroutine are local to that subroutine and cannot be accessed from outside. This helps in maintaining data integrity and avoids conflicts with variables in other parts of the program.

**3.5. Reuse and Maintainability:** By encapsulating specific logic within subroutines, developers can easily reuse the same logic across multiple parts of the program. This promotes code reusability and reduces redundancy. Additionally, if changes are required to the logic within a subroutine, they only need to be made in one place, improving maintainability.

**3.6. Control Flow:** Subroutines can be called from anywhere within the module pool program, allowing for flexible control flow. They can be called conditionally, iteratively, or sequentially based on the requirements of the program.

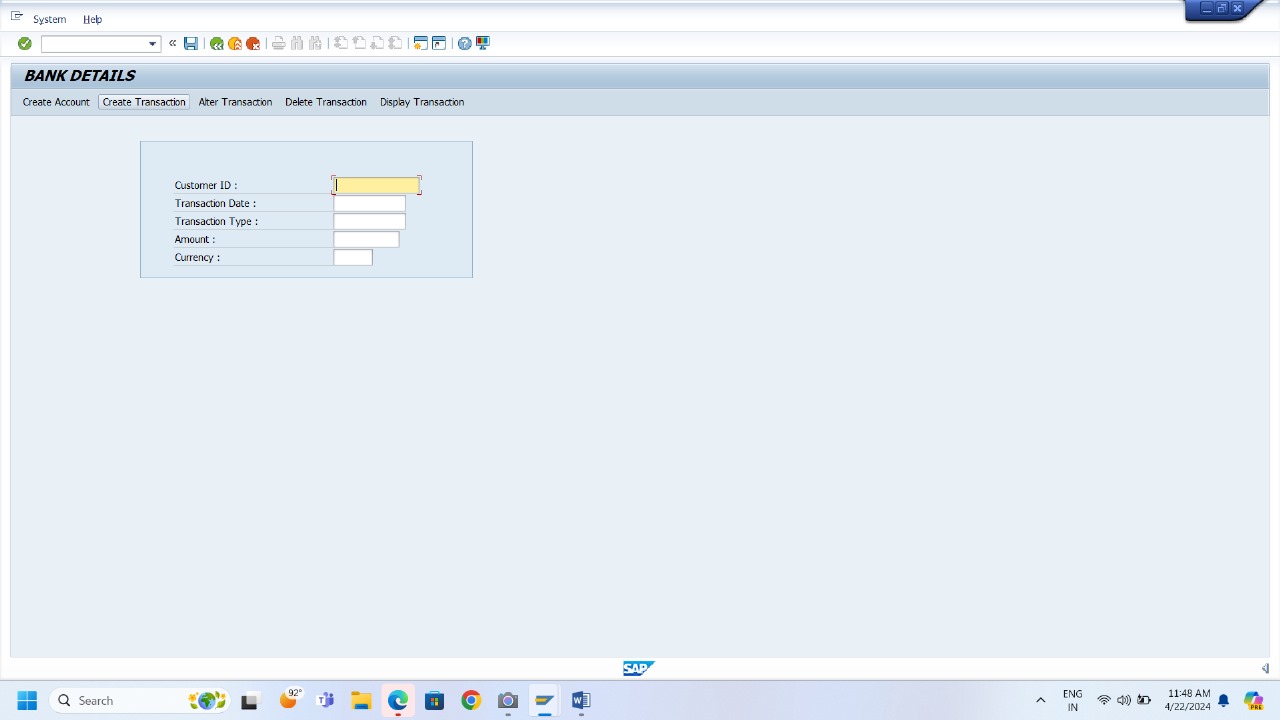
In summary, subroutine blocks in SAP ABAP module pool programs provide a structured way to organize and encapsulate reusable code. They promote modularity, code reusability, and maintainability, making it easier to develop and maintain complex SAP applications.



**FIG: 2.2**

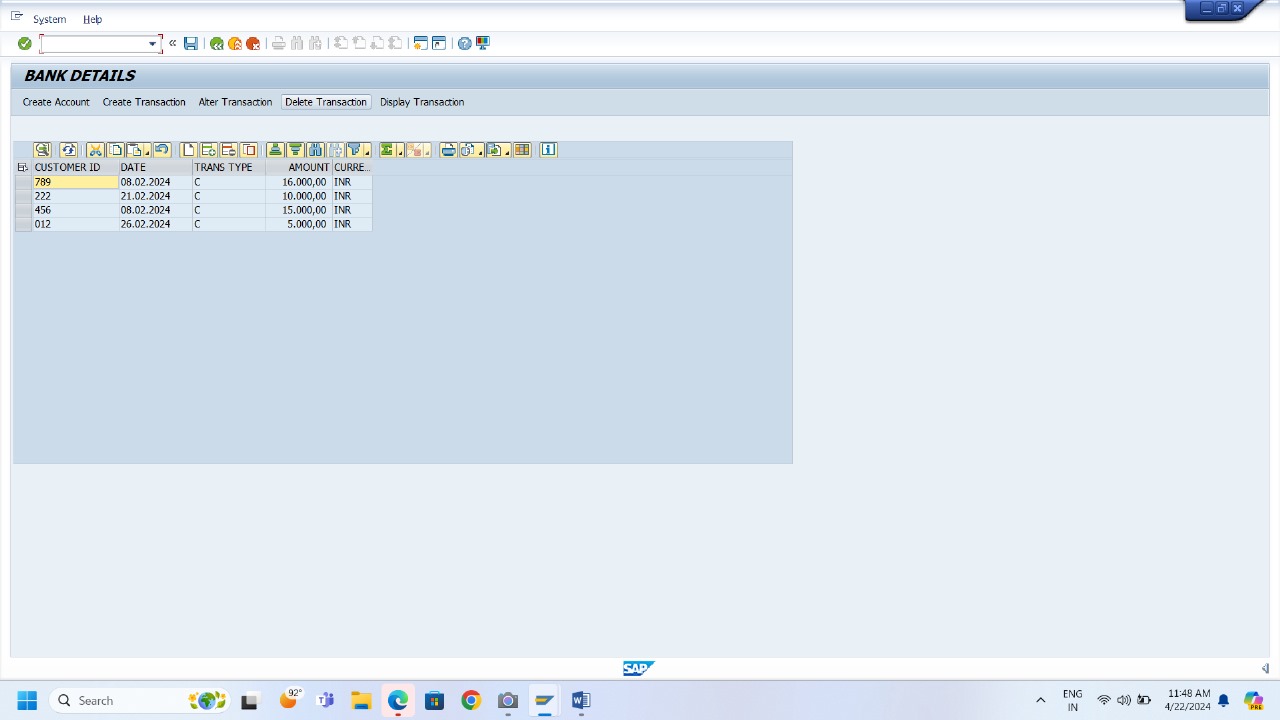
**Chapter 3: PROJECT OVERVIEW**

**3.1 INSIGHTS OF PROJECT:**

****

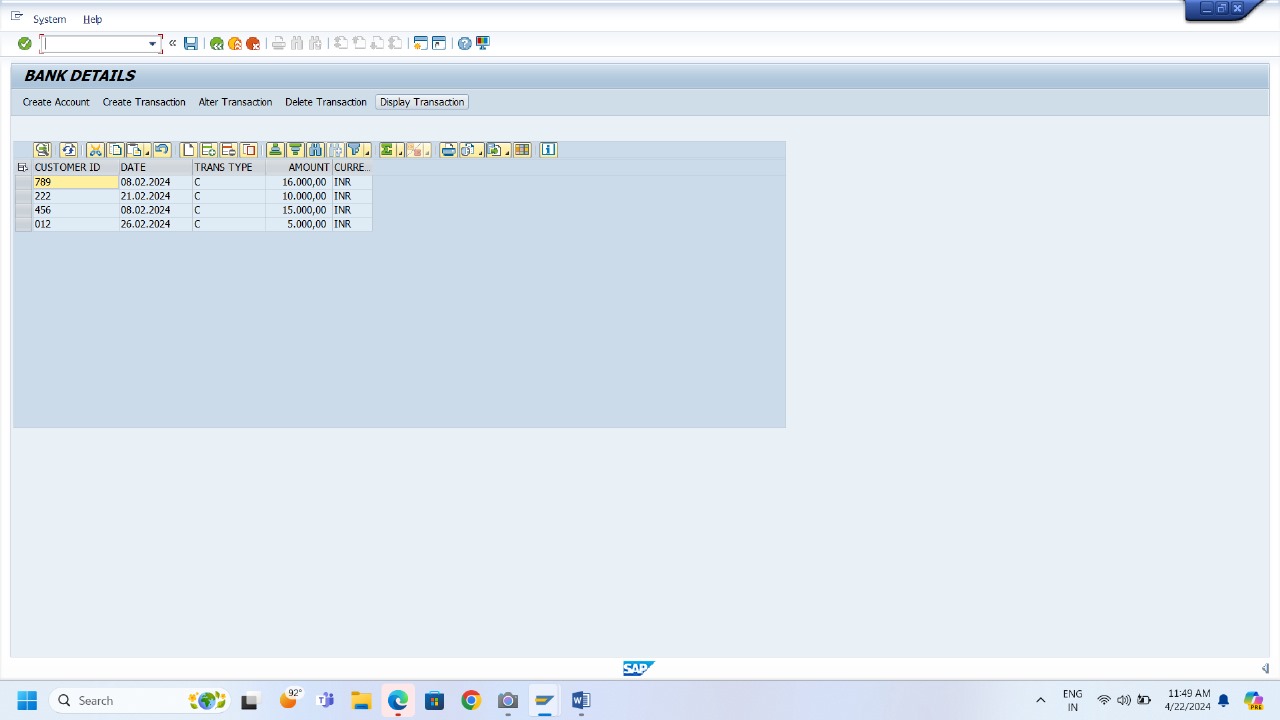
**FIG: 3.1**

The fig 3.1 shows the user experience of creating and account in the bank system. The user will enter their name, ID, transaction type i.e either credit or debit and the amount of money.

****

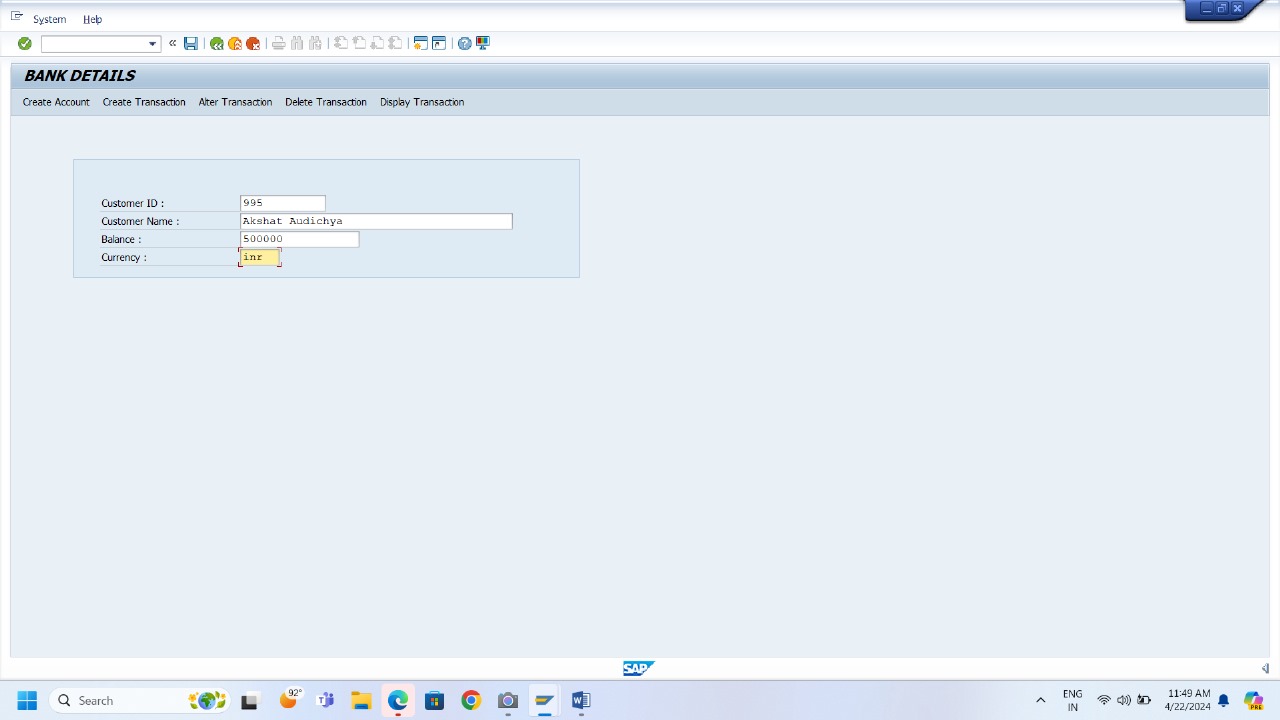
**FIG: 3.2**

The fig 3.2 represents the deletion of transaction which means the user can delete their transaction and only their transaction. When a transaction is deleted the amount mentioned is added or removed, depending on the type of transaction so as to make it back to its original balance.

****

**FIG: 3.3**

The fig 3.3 depicts the screen of alteration which gives the user to alter the transaction in case if the user has entered the wrong amount or the transaction type. The changes will be reflected in the balance of the user and if there is insufficient amount for the alteration, then the user will be notified for the respective case.

****

**FIG: 3.4**

The fig 3.4 gives the brief of how the user is created when a new entry is created. First field is the Customer ID which is autogenerated by the system using the function module. The user needs to enter the username and the amount of initial balance to be deposited with the respective currency.

**3.2 ADVANTAGES OF SYSTEM:**

* Proposed system enables Bank to find out the particular Account.
* This system also helps to know about the deposit and withdraw amount for particular person.
* This system have feature to search customer record by two option one is acc\_number and second is by customer name and also feature to update, delete account.
* In this system we also store employee record and search, modify, delete operations also occurs.
* Reduction of errors and viruses due to non requirement of the internet.
* This system have feature to see the account detail according to date.
* Updating of data is easy in computerized system.

**3.3 LIMITATIONS OF THE PROJECT:**

**Functional Limitations:**

* **Limited Transaction Types:** The system might only handle basic transactions like deposits, withdrawals, and potentially transfers between accounts within the same bank. It might not support more complex transactions like bill payments, loan applications, or investment management.
* **Limited Account Types:** The system might only cater to a limited number of account types (e.g., savings accounts, checking accounts). It might not handle more complex account structures like joint accounts, investment accounts, or loan accounts.
* **Limited Reporting:** The system might offer basic reporting functionalities like account statements or transaction history. It might lack in-depth analytics or reporting tools for comprehensive analysis of customer behavior or banking trends.
* **Scalability:** The system might not be scalable to handle a large number of users or a high volume of transactions. This could lead to performance issues as the user base grows.
* **Security Considerations:** While security is crucial, the description doesn't mention specific security features implemented. There might be limitations in user authentication, data encryption, or audit trails for comprehensive transaction tracking.

**Technical Limitations:**

* **Development Platform:** The description doesn't specify the development platform or programming language used. Depending on the chosen platform, there might be limitations in terms of functionality, performance, or integration capabilities.
* **Database Design:** A simple database design might struggle to handle complex data structures or queries needed for more advanced features in the future.
* **User Interface:** The focus might be on core functionalities, potentially neglecting a user-friendly and intuitive interface. This could lead to a less user-friendly experience for customers.

**Project Scope Limitations:**

* **Project Timeframe and Budget:** The project might have been limited by time or budget constraints, restricting the scope of functionalities that could be implemented.
* **Development Team Expertise:** The development team's skills and experience might have influenced the complexity of features that could be implemented within the project.

**3.4 FUTURE SCOPE:**

In the future more software companies will hire this software program because now a days the need for the speed in the day-to-day life has become essential. As competition increases, companies by considering old version, they develop more efficient versions for individual success.

In future we can make a link of this project with networking. We can also convert this project in Hibernate. Hibernate is a framework. It storage large amount of database.

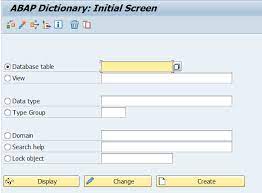
In future we can also add the facility to change the password and recovery the forgotten password..

**3.5 DATA DICTIONARY:**

Data dictionary is the collection of complete data is used in some process. It can also to be called the whole databases that are used in the project. Data is stored in different data bases. Database is a collection of different table and tables further are collection of records in which each record is made up of primary unit called fields’ data fields are the entity where the information can be stored and accessed as and when required.

The components of Data Dictionatry are:

* **Data Objects:** These are the fundamental elements that store information within a database. They can include:
* **Tables:** Structured collections of data rows and columns that represent entities or concepts.
* **Views:** Virtual tables based on underlying tables, providing specific subsets or transformations of data.
* **Domains:** Define the valid values and data types (e.g., text, number, date) that an attribute can hold.
* **Fields/Columns:** Individual attributes within a table that hold specific data points.
* **Metadata:** This is "data about data." It describes the characteristics and properties of data objects, providing crucial information for understanding and managing the database. Here are some key metadata elements:
* **Data Name:** A unique identifier for each data object (e.g., table name, column name).
* **Data Type:** Specifies the type of data an attribute can hold (e.g., text, number, date).
* **Size/Length:** Defines the amount of storage space allocated for each data element.
* **Description:** Provides a textual explanation of the data object's purpose and meaning.
* **Constraints:** Defines rules that restrict the data values allowed within an attribute (e.g., mandatory fields, primary key constraints).
* **Relationships:** Defines how data objects are related to each other (e.g., foreign key relationships between tables).

****

**FIG: 3.5**

**Chapter 4: PROGRAM**

**4.1 TOP BLOCK OF PROGRAM:**

\*&---------------------------------------------------------------------\*  
\*& Include ZT039\_BANK\_TOP                           - Module Pool      ZT039\_BANK\_TASK  
\*&---------------------------------------------------------------------\*  
PROGRAM zt039\_bank\_task.  
  
DATA: gt\_bank       TYPE TABLE OF zt039\_bank,  
      gs\_bank       TYPE zt039\_bank,  
      gs\_bank1      TYPE zt039\_bank,  
      gt\_bank\_info  TYPE TABLE OF zt039\_bank\_info,  
      gs\_bank\_info  TYPE zt039\_bank\_info,  
      gs\_bank1\_info TYPE zt039\_bank\_info,  
      lt\_fieldcat   TYPE lvc\_t\_fcat,  
      wa\_fieldcat   TYPE lvc\_s\_fcat,  
      lo\_obj        TYPE REF TO cl\_gui\_custom\_container,  
      lo\_grid       TYPE REF TO cl\_gui\_alv\_grid,  
      gs\_layout     TYPE lvc\_s\_layo,  
      flag          TYPE c,  
      test          TYPE c,  
      delete        TYPE c,  
      create\_t      TYPE c,  
      create\_a      TYPE c,  
      e\_row         TYPE i,  
      e\_value       TYPE c,  
      e\_col         TYPE i,  
      es\_col\_id     TYPE lvc\_s\_col,  
      row           TYPE  lvc\_t\_row,  
      rows          TYPE lvc\_s\_row,  
      roid          TYPE  lvc\_t\_roid,  
      bank\_class    TYPE REF TO zt039\_bank\_class,  
      lv\_var        TYPE sy-dynnr.

* 1. **EVENT BLOCK OF PROGRAM:**

\*----------------------------------------------------------------------\*  
\*\*\*INCLUDE ZT039\_BANK\_TASK\_STATUSO01.  
\*----------------------------------------------------------------------\*  
\*&---------------------------------------------------------------------\*  
\*& Module STATUS OUTPUT  
\*&---------------------------------------------------------------------\*  
\*&  
\*&---------------------------------------------------------------------\*  
MODULE status OUTPUT.  
  SET PF-STATUS 'STATUS'.  
  SET TITLEBAR 'TITLE'.  
CONSTANTS: cntl\_true TYPE i VALUE 1,  
  
cntl\_false TYPE i VALUE 0.  
  
DATA:  
  
h\_picture TYPE REF TO cl\_gui\_picture,  
  
h\_pic\_container TYPE REF TO cl\_gui\_custom\_container.  
  
DATA: graphic\_url(255),  
  
graphic\_refresh(1),  
  
g\_result LIKE cntl\_true.  
  
DATA: BEGIN OF graphic\_table OCCURS 0,  
  
line(255) TYPE x,  
  
END OF graphic\_table.  
  
DATA: graphic\_size TYPE i.  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
  
DATA: l\_graphic\_xstr TYPE xstring,  
  
l\_graphic\_conv TYPE i,  
  
l\_graphic\_offs TYPE i.  
  
CALL METHOD cl\_ssf\_xsf\_utilities=>get\_bds\_graphic\_as\_bmp  
  
EXPORTING  
  
p\_object = 'GRAPHICS'  
  
p\_name = 'DILIGENT' "IMAGE NAME - Image name from SE78  
  
p\_id = 'BMAP'  
  
p\_btype = 'BCOL' "(BMON = black&white, BCOL = colour)  
  
RECEIVING  
  
p\_bmp = l\_graphic\_xstr  
  
EXCEPTIONS  
  
not\_found = 1  
  
OTHERS = 2.  
  
graphic\_size = XSTRLEN( l\_graphic\_xstr ).  
  
CHECK graphic\_size > 0.  
  
l\_graphic\_conv = graphic\_size.  
  
l\_graphic\_offs = 0.  
  
WHILE l\_graphic\_conv > 255.  
  
graphic\_table-line = l\_graphic\_xstr+l\_graphic\_offs(255).  
  
APPEND graphic\_table.  
  
l\_graphic\_offs = l\_graphic\_offs + 255.  
  
l\_graphic\_conv = l\_graphic\_conv - 255.  
  
ENDWHILE.  
  
graphic\_table-line = l\_graphic\_xstr+l\_graphic\_offs(l\_graphic\_conv).  
  
APPEND graphic\_table.  
  
CALL FUNCTION 'DP\_CREATE\_URL'  
  
EXPORTING  
  
type = 'image' "#EC NOTEXT  
  
subtype = cndp\_sap\_tab\_unknown " 'X-UNKNOWN'  
  
size = graphic\_size  
  
lifetime = cndp\_lifetime\_transaction "'T'  
  
TABLES  
  
data = graphic\_table  
  
CHANGING  
  
url = graphic\_url  
  
EXCEPTIONS  
  
dp\_invalid\_parameter = 1  
  
dp\_error\_put\_table = 2  
  
dp\_error\_general = 3  
  
OTHERS = 4 .  
  
IF sy-subrc <> 0.  
  
MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno  
  
WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.  
  
EXIT.  
  
ENDIF.  
  
CREATE OBJECT h\_pic\_container  
  
EXPORTING container\_name = 'ALV\_AREA1'.  
  
CREATE OBJECT h\_picture EXPORTING parent = h\_pic\_container.  
  
CALL METHOD h\_picture->load\_picture\_from\_url  
  
EXPORTING  
  
url = graphic\_url  
  
IMPORTING  
  
RESULT = g\_result.  
ENDMODULE.

**4.3 SUBROUTINE BLOCK OF PROGRAM:**

\*&---------------------------------------------------------------------\*  
\*& Include          ZT039\_BANK\_F01  
\*&---------------------------------------------------------------------\*  
  
\*&---------------------------------------------------------------------\*  
\*& Form delete  
\*&---------------------------------------------------------------------\*  
\*& text  
\*&---------------------------------------------------------------------\*  
\*& -->  p1        text  
\*& <--  p2        text  
\*&---------------------------------------------------------------------\*  
FORM delete .  
  SELECT \* FROM zt039\_bank INTO TABLE gt\_bank.  
  CALL METHOD lo\_grid->get\_selected\_rows  
    IMPORTING  
      et\_index\_rows = row.          " Indexes of Selected Rows  
  
  LOOP AT row INTO rows.  
    READ TABLE gt\_bank\_info INTO gs\_bank\_info INDEX rows-index.  
    READ TABLE gt\_bank INTO gs\_bank WITH KEY cust\_id = gs\_bank\_info-cust\_id.  
    IF gs\_bank\_info-trans\_type = 'C'.  
      gs\_bank-cust\_balance = gs\_bank-cust\_balance - gs\_bank\_info-amount.  
    ENDIF.  
    IF gs\_bank\_info-trans\_type = 'D'.  
      gs\_bank-cust\_balance = gs\_bank-cust\_balance + gs\_bank\_info-amount.  
    ENDIF.  
    MODIFY gt\_bank FROM gs\_bank TRANSPORTING cust\_balance WHERE cust\_id = gs\_bank-cust\_id.  
    UPDATE zt039\_bank FROM TABLE gt\_bank.  
    DELETE FROM zt039\_bank\_info WHERE cust\_id = gs\_bank\_info-cust\_id AND t\_date = gs\_bank\_info-t\_date.  
    MESSAGE 'deleted successfully' TYPE 'S'.  
  ENDLOOP.  
  
  
ENDFORM.

**Chapter 5: CONCLUSION**

This project successfully developed a core functionality for your bank system using SAP ABAP and Module Pools. The bank management system provides a foundation for further development and integration with other banking functionalities.

**Key Achievements:**

* Developed a user-friendly interface using a Module Pool, allowing user to create account, make transactions, display transactions, alter and delete the transaction in real time.
* Implemented core logic within ABAP code to handle complex calculations performed while making changes in the balance and currency conversions
* Utilized module pool, transaction codes, forms, subroutines, data dictionary, etc. for efficient and maintainable code.

Overall, this project demonstrates the successful application of SAP ABAP for building core functionalities within your bank system. By following best practices and leveraging the potential of Module Pools and ABAP programming, you can continue to develop and enhance your banking system to meet the evolving needs of your customers.

**Bibliography:**

<https://www.sapabap.com/>

<https://www.sap.com/india/industries/banking.html>

<https://www.geeksforgeeks.org/what-is-sap-abap-data-dictionary-ddic/>

<https://www.tutorialspoint.com/sap_abap/index.htm>