

DATA STRUCTURE IN C

PROJECT REPORT

SUBMITTED BY: SUBMITTED TO:

NAME : GITANSHU SONKAR NAME : MR PARWAN SINGH

UID: 24BCA10179 Designation: Assistant

SUBJECT CODE: 24CAT-151

COURSE: BCA Professor

Acknowledgement

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I would like to express my sincere gratitude to all those who supported and guided me throughout the development of this project, titled **“Bank Customer Management System.”**

First and foremost, I extend my heartfelt thanks to my project guide, [**MR PARWAN SINGH**], for their continuous encouragement, valuable insights, and expert supervision during every phase of this project. Their guidance has played a crucial role in shaping the direction and quality of this work.

I am also thankful to my institution and all faculty members of the [**UNIVERSITY INSTITUTE OF COMPUTING**] department for providing the resources and environment necessary for completing this project. Their support and feedback helped me enhance my technical knowledge and practical skills.

A special thanks to my friends and classmates who provided moral support, shared their thoughts, and offered constructive feedback whenever I faced challenges during development.

Finally, I would like to express my deepest appreciation to my family for their unwavering support and motivation throughout this journey. This project has been a great learning experience, and I am truly grateful to everyone who contributed to its successful complete.

Abstract

In the modern financial landscape, the need for efficient and secure data management systems is crucial for any banking institution. This project aims to develop a **Bank Customer Management System**, an application designed to store, retrieve, update, and delete customer information in a structured and reliable manner. The system allows bank staff to manage customer data seamlessly, providing functionalities such as adding new customers, viewing existing records, updating customer details, and removing outdated or incorrect information.

The application leverages a database-driven architecture to ensure data consistency and quick access to records. Through an intuitive interface, authorized personnel can interact with the system efficiently without needing extensive technical knowledge. The system is built with scalability and security in mind, supporting essential operations while preparing the foundation for future enhancements like role-based access control, audit trails, and online customer access.

By automating customer data handling, the system reduces manual errors, improves response time, and ensures the integrity and confidentiality of sensitive information. This project not only addresses a critical operational need in banking but also demonstrates the application of software engineering principles in building real-world data management solutions

Introduction

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In today’s digital age, banks are increasingly adopting technology to streamline their operations and improve customer service. One of the key areas where technology plays a vital role is in managing customer information. A bank needs to maintain accurate records of its customers, which include their personal details, account types, balances, and other essential data. This project focuses on developing a Bank Customer Management System — an application designed to store, retrieve, update, and delete customer information efficiently and securely.

The system provides bank staff with easy access to customer data through a user-friendly interface and ensures that operations like updating customer details or deleting records can be done seamlessly. By digitizing customer data management, the application eliminates the need for manual record-keeping, reduces the risk of data loss, and enhances overall operational efficiency.

**Problem Definition**

Traditional methods of managing customer information using paper files or static databases are inefficient, time-consuming, and prone to errors. Banks need a reliable and secure way to handle large volumes of customer data. Manual processes can lead to data duplication, loss of information, and delays in accessing or modifying records.

Moreover, in today’s fast-paced environment, customers expect quick responses from banks regarding their account information. The lack of a centralized, dynamic system makes it difficult for banks to serve their customers efficiently. There is a pressing need for an application that allows authorized staff to access, update, or delete customer records in real-time while maintaining data integrity and security.

**Objectives**

**The main objective of this project is to develop an application that enables a bank to manage its customer data effectively. Key goals include:**

* **Creating a secure and user-friendly interface for bank staff.**
* **Allowing staff to add new customer records with comprehensive details.**
* **Enabling quick search and retrieval of existing customer information.**
* **Providing functionality to update customer data when needed.**
* **Allowing deletion of customer records if required.**
* **Maintaining data integrity, confidentiality, and security.**
* **Ensuring fast performance and reliable storage using a structured database.**

**This project aims to improve operational efficiency and reduce errors, while also preparing the system for potential future enhancements like audit logging or integration with online banking platforms**

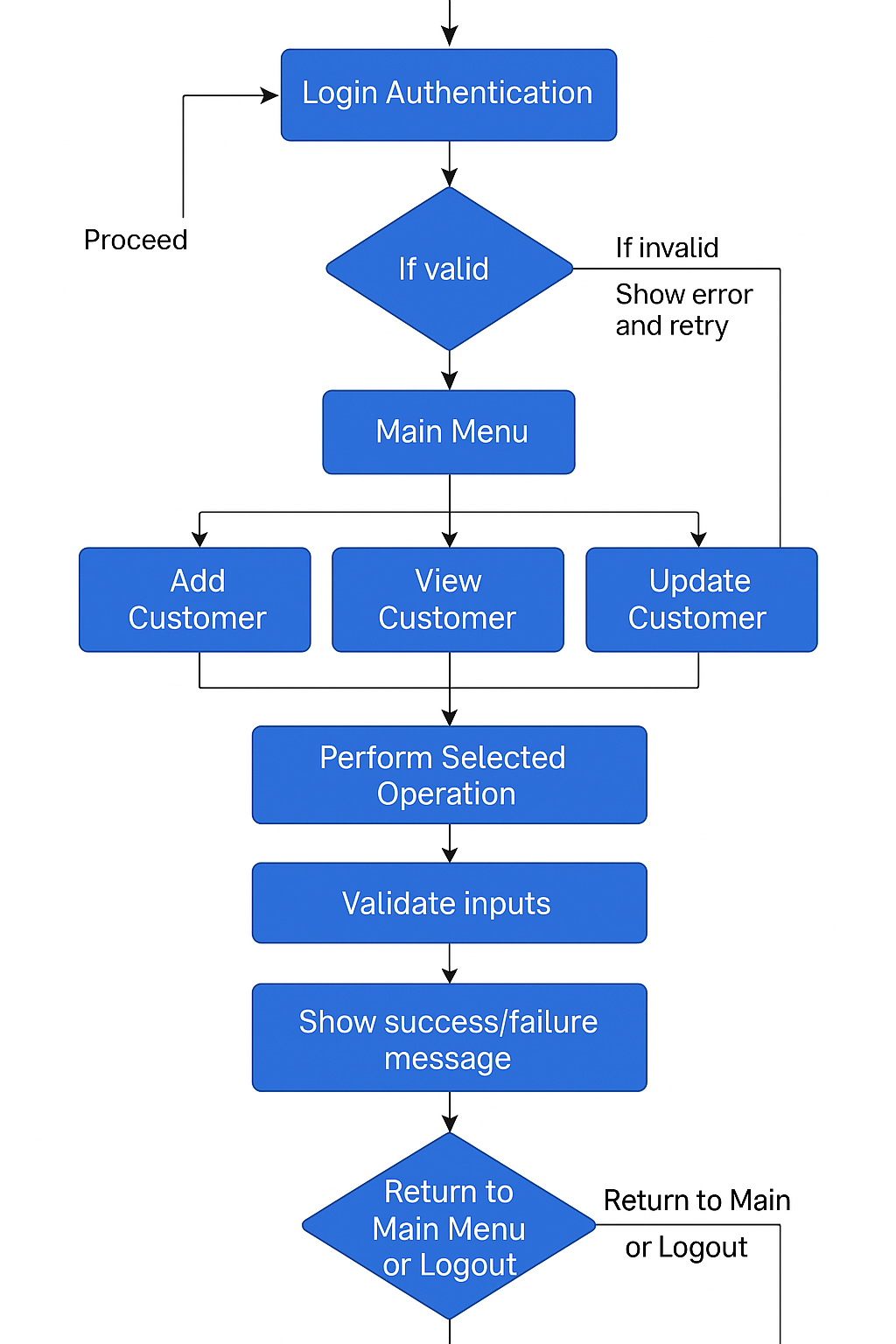
**System Design**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The system is designed using a modular architecture to promote scalability and ease of maintenance. It includes three main layers: the user interface, the application logic, and the database layer.

* **User Interface (UI):** A clean and intuitive interface allows staff to enter, search, update, and delete customer records.
* **Application Layer:** This is the core of the system, responsible for validating inputs, managing user sessions, and handling operations like data updates and deletions.
* **Database Layer:** Customer data is stored in a structured relational database, such as MySQL or SQLite, ensuring data consistency and easy querying.

The application uses a CRUD (Create, Read, Update, Delete) structure, with each function implemented as a module. This separation of concerns makes the system flexible and easy to expand with future features such as user roles, transaction history, or automated alerts.



**Source Code**

#include <iostream>

#include <string>

using namespace std;

const int MAX\_CUSTOMERS = 100;

struct Customer {

int id;

string name;

string address;

string phone;

string email;

string accountType;

double balance;

};

Customer customers[MAX\_CUSTOMERS];

int customerCount = 0;

// Function to add a new customer

void addCustomer() {

if (customerCount >= MAX\_CUSTOMERS) {

cout << "Database full. Cannot add more customers." << endl;

return;

}

Customer c;

cout << "Enter Customer ID: ";

cin >> c.id;

cin.ignore();

cout << "Enter Name: ";

getline(cin, c.name);

cout << "Enter Address: ";

getline(cin, c.address);

cout << "Enter Phone: ";

getline(cin, c.phone);

cout << "Enter Email: ";

getline(cin, c.email);

cout << "Enter Account Type: ";

getline(cin, c.accountType);

cout << "Enter Balance: ";

cin >> c.balance;

customers[customerCount++] = c;

cout << "Customer added successfully!" << endl;

}

// Function to display all customers

void viewCustomers() {

for (int i = 0; i < customerCount; i++) {

cout << "\nCustomer ID: " << customers[i].id

<< "\nName: " << customers[i].name

<< "\nAddress: " << customers[i].address

<< "\nPhone: " << customers[i].phone

<< "\nEmail: " << customers[i].email

<< "\nAccount Type: " << customers[i].accountType

<< "\nBalance: " << customers[i].balance << "\n";

}

}

// Function to update customer by ID

void updateCustomer() {

int id;

cout << "Enter Customer ID to update: ";

cin >> id;

for (int i = 0; i < customerCount; i++) {

if (customers[i].id == id) {

cout << "Enter new Name: ";

cin.ignore();

getline(cin, customers[i].name);

cout << "Customer updated successfully!" << endl;

return;

}

}

cout << "Customer not found." << endl;

}

// Function to delete a customer by ID

void deleteCustomer() {

int id;

cout << "Enter Customer ID to delete: ";

cin >> id;

for (int i = 0; i < customerCount; i++) {

if (customers[i].id == id) {

for (int j = i; j < customerCount - 1; j++) {

customers[j] = customers[j + 1];

}

customerCount--;

cout << "Customer deleted successfully!" << endl;

return;

}

}

cout << "Customer not found." << endl;

}

int main() {

int choice;

do {

cout << "\n--- Bank Customer Management System ---\n";

cout << "1. Add Customer\n2. View Customers\n3. Update Customer\n4. Delete Customer\n5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: addCustomer(); break;

case 2: viewCustomers(); break;

case 3: updateCustomer(); break;

case 4: deleteCustomer(); break;

case 5: cout << "Exiting system.\n"; break;

default: cout << "Invalid choice. Try again.\n";

}

} while (choice != 5);

return 0;

}

**Challenges Faced**

**During the development of the system, several challenges were encountered. One major issue was ensuring data integrity and preventing accidental deletion or corruption of records. Implementing robust input validation was necessary to avoid invalid data entry, especially for fields like phone numbers and email addresses.**

**Another challenge was designing a user interface that was both functional and easy to use, especially for bank staff who may not be very tech-savvy. Balancing functionality with simplicity required careful planning and iteration.**

**Database connection and transaction handling were also tricky, especially when multiple updates happened simultaneously. Ensuring the application remained responsive and didn't crash due to unexpected input or operations took significant debugging and testing.**

**Future Enhancements**

**To make the system more powerful and efficient, several enhancements can be considered:**

* **Role-Based Access Control: Different permissions for admins and regular staff.**
* **Audit Logs: Record all changes made to customer data for security and compliance.**
* **Online Integration: Allow customers to view or update their data through a secure web portal.**
* **Backup & Recovery: Automated data backups to prevent loss in case of system failure.**
* **Advanced Search: Add filters and keyword-based search for large databases.**
* **Notification System: Send emails or SMS alerts to customers for updates or reminders.**

**These enhancements will not only improve the usability and security of the system but also prepare it for real-world deployment in a banking environment.**

**Conclusion**

**This project successfully demonstrates how a bank can use a simple yet powerful application to manage customer data efficiently. The system ensures accurate record-keeping and provides all essential functionalities like adding, viewing, updating, and deleting customer information. With a structured database and user-friendly interface, the system addresses key operational challenges faced by banks in handling customer data manually.**

**While the current version of the system handles basic functionalities, it lays the foundation for more advanced features such as audit tracking, online access, and role-based controls. The modular design ensures the application can evolve with the bank's needs over time, making it a practical and scalable solution for modern banking.**

**References**

**** [**Python SQLite Documentation**](https://docs.python.org/3/library/sqlite3.html)

** W3Schools SQL Tutorial**

** Real Python – Working with Databases**

**** [**Stack Overflow**](https://stackoverflow.com/) **– For community help and code fixes**

** Code examples and UI inspiration from various GitHub repositories**