**OOP Project Second Semester…**

**THE BANKING SYSTEM !!**

**Group Members:**

Shivam 24K-0957

Jeevan 24K-0821

Yogita 24K-0929

**1. Introduction**

* **Background**:  
  This project focuses on building a secure and dynamic Bank Management System using Object-Oriented Programming (OOP) in C++. It simulates real-world banking tasks with secure data handling, modular design, and an interactive graphical interface using the **Raylib library**.
* **Problem Statement**:  
  Existing systems often fail to adapt services based on customer engagement or account history. This project solves that by integrating a personalized service layer based on **account age**, using OOP and GUI principles.
* **Objectives**:
  + Implement a fully OOP-driven Bank Management System
  + Use C++ to model users, accounts, transactions, and system logic
  + Apply OOP principles: **abstraction, encapsulation, inheritance, polymorphism**, and **static variables**
  + Store and manage all data securely with file handling
  + Encrypt passwords with a secret key
  + Build a GUI using **Raylib**
  + Introduce **account age-based facilities** (calculated using the current date and account creation date)

**2. Scope of the Project**

* **Inclusions**:
  + User roles: **Account Holder** and **Admin**
  + Account types: **Saving** and **Current**
  + Card types: **CreditCard** and **DebitCard**
  + A Transaction class to log all transfers and withdrawals
  + Password encryption using a secret key
  + GUI features (login, transfer, admin panel, etc.) using Raylib
  + Full OOP integration:
    - **Encapsulation** of private data
    - **Inheritance** for Account and Card hierarchies
    - **Abstraction** to expose simplified banking interfaces
    - **Polymorphism** for overrideable operations (e.g., applyCharges(), calculateInterest())
  + **Account Age Calculation**:
    - Capture the **account creation date**
    - Use system date to compute **account age**
    - Based on account age, provide facilities like:
      * Bonus interest for accounts older than 2 years
      * Loan eligibility after 1 year
      * Reduced transaction fees for long-standing users

**3. Project Description**

* **Overview**:  
  The project mimics real banking operations through a structured OOP approach in C++. It includes all essential banking features and enhances realism with account aging, transaction logs, and secure password handling. A GUI developed with **Raylib** will provide an intuitive interface for users and admins.
* **Technical Requirements**:
  + C++ compiler (e.g., GCC, MSVC)
  + Raylib library for GUI
  + Visual Studio, Code::Blocks, or VS Code
  + Date library (e.g., ctime) for date/time operations
  + MS Office for documentation
* **Project Phases**:
  + Planning and class diagram design
  + Base implementation of classes and core features
  + File handling and encryption integration
  + Account age features and conditional logic
  + GUI development and feature integration
  + Testing and final report

**4. Methodology**

* **Approach**:  
  Agile-inspired, incremental development. CLI-based prototype is built first, followed by Raylib GUI integration. New features like account age logic are tested independently before integration.
* **Team Responsibilities**:

**Shivam**: classes implementation and relations

**Yogita**: file handling

**Jeevan**: GUI Implementaion

**5. Expected Outcomes**

* **Deliverables**:
  + A C++-based GUI banking application
  + Modular, well-documented source code
  + Secure file-based data storage
  + Realistic features including transaction logging, encryption, and age-based services
  + Visual interface built with Raylib for login, deposit, withdrawal, and admin control
* **Relevance**:  
  This project is highly relevant to ICT and OOP coursework. It demonstrates application of class hierarchies, real-time data handling, and user interaction in a modern GUI environment, with added depth via account age–driven decision logic.

**6. Resources Needed**

* **Software**:
  + C++ IDE (Visual Studio, Code::Blocks)
  + Raylib library
  + MS Office (report and presentation)
  + Optional: libraries for date/time operations
* **Other Resources**:
  + Online guides on file I/O and Raylib GUI handling
  + Tutorials on date/time calculations in C++
  + Instructor feedback for logic design and feature validation

**Additionally:**

I will try to make another class of atm simulation which directly withdraw or transfer amount from atm without log in to account. And I also have other functionalites in my mind so if I got time will try to implement.