University of Tunis
Higher Institute of Digital Engineering

DATA STRUCTURES PROJECT
BANKING MANAGEMENT SYSTEM

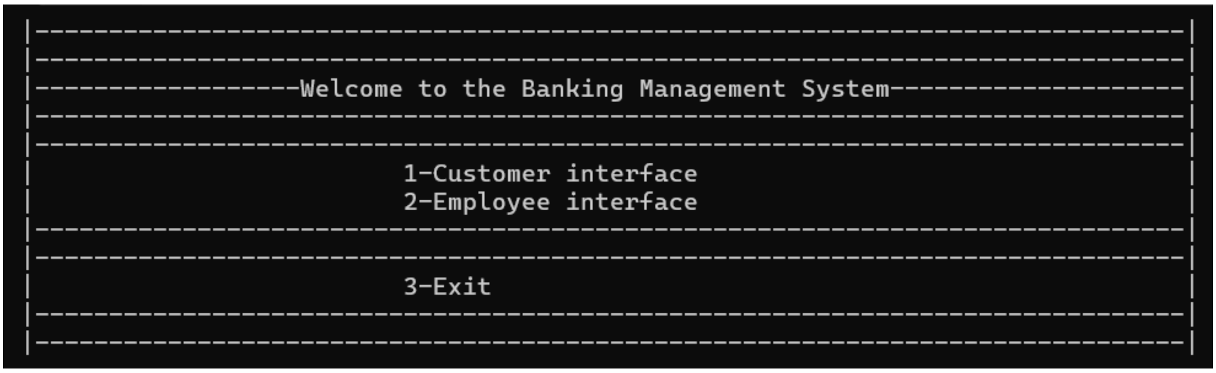
Professor:	Rihab BOUSLAMA
Level:	PI 1
Academic Year:	2025/2026

ENVIRONMENT:
The project should be implemented in C++ language

Project Submission Guidelines
Content to submit: A Google Drive link to the project, containing all source files and a detailed, clear report. The link should be sent via email.
The presentation of the project will be taken into account.
The project must be completed by groups of 4 students from the same class.
The project evaluation will be conducted in two stages:
First stage: Based on the present specifications, to be submitted by **07/12/2025**, with an initial in-person presentation on **10/12/2025**.
Second stage: A second in-person evaluation on **15/12/2025**.

The project aims to develop a banking management system capable of managing employee records, customer accounts, financial transactions, bank branches, and loan operations.

The application should include two separate interfaces: one for customers and another for employees.



```
|------------------------------------------------------------------------------|
|------------------------------------------------------------------------------|
|------------------Welcome to the Banking Management System--------------------|
|------------------------------------------------------------------------------|
|------------------------------------------------------------------------------|
|                         1-Customer interface                                 |
|                         2-Employee interface                                 |
|------------------------------------------------------------------------------|
|------------------------------------------------------------------------------|
|                         3-Exit                                               |
|------------------------------------------------------------------------------|
|------------------------------------------------------------------------------|
```

Part 1: Customer interface:

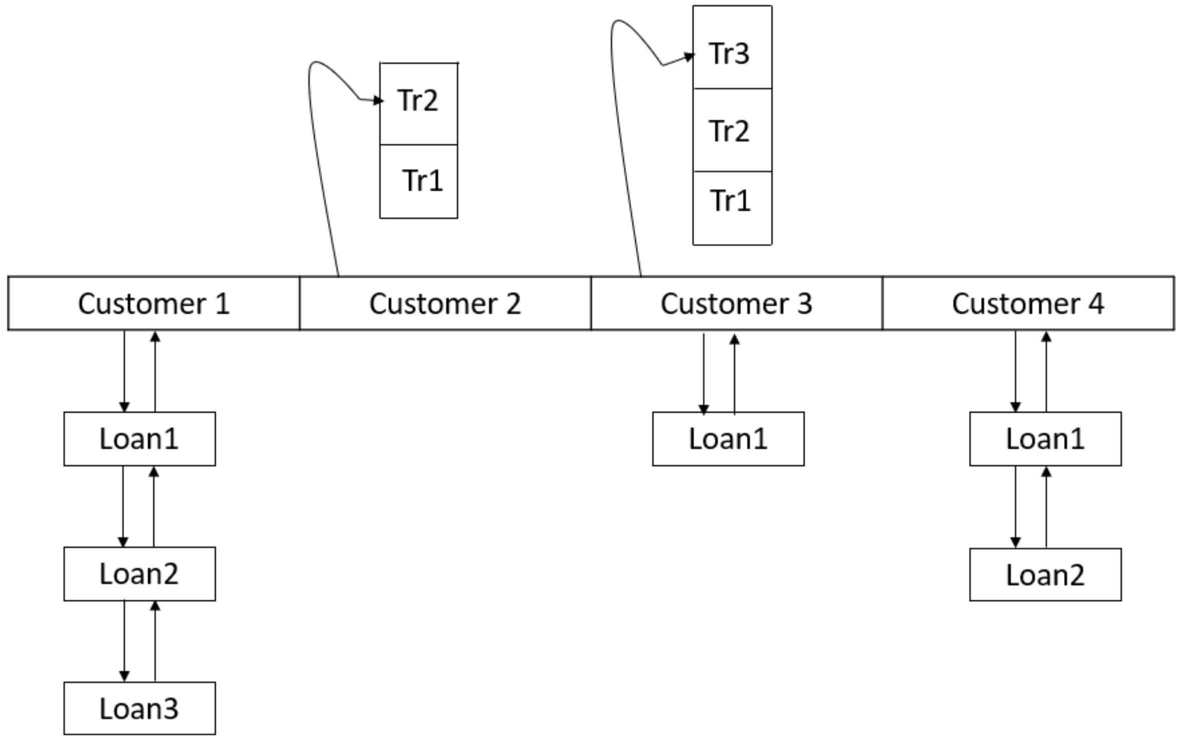
Each customer is associated with an account, which is characterized by the following attributes:

- Account number (should be unique)
- Account type
- IBAN
- Branch code

- Account holder name
- Opening date
- Status (active, inactive, closed)
- Balance (in Tunisian dinars)

For each account, we should also store the list of loans associated with that account (using a doubly linked list) and the daily transactions performed on that account (using a stack).

In the end, we will obtain a structure like this:



For each loan we want to store the following attributes:

- Loan ID
- Loan type (car, home, student, business)
- Principal amount
- Interest rate
- Amount paid
- Remaining balance
- Start date
- End date
- Loan status (active, completed, overdue)

Each transaction is characterized by:

- Transaction ID
- Account number
- Type (deposit, withdrawal)
- Amount
- Date

Tasks:

1) Each customer must log in before accessing the system.
2) Each customer can view the list of their loans along with the corresponding details.

3) Each customer should be able to submit a new loan request (more details are provided in Part 2).
4) Each customer can perform one of the following transactions:
    a. Withdraw money (available denominations: 10, 20, 50 TND)
    b. Deposit money into their account (the amount must be specified)
5) Each customer can view the list of transactions made during the day.
6) Each customer can undo only the last transaction.

Part 2: Employee Space:

Each employee is characterized by the following attributes:

- Id (should be unique)
- Name
- Last name
- Address
- Salary
- Hire date
- Bank branch (store the code of the branch or 1 for head office)

Employees should be stored in an array.

Tasks:

1) Add, delete, or modify an employee.
2) Display the list of employees in alphabetical order by last name.
3) Display the list of employees grouped by bank branch.
4) Display the earliest recruited and most recently recruited employees.
5) Add a customer account.
6) Display the list of accounts.
7) Change the status of an account.
8) Delete all accounts whose status is closed and store them in an array of archived accounts.
9) Display the list of loans for a specific customer.
10) Change the status of a loan.
11) Delete loans whose status is completed and store them in a singly linked list named “completed_loans”.
12) Manage all loan requests based on the request date using a first-in, first-out (FIFO) mechanism. Each loan should be either accepted or declined:
    - If declined, it should be permanently deleted.
    - If accepted, it should be added to the customer’s list of loans and removed from the list of loan requests.
13) Manage a daily transaction log stack for each account to support undo, rollback, or crash recovery. At the end of the day, once an employee finalizes the day, all transactions should be copied into a singly linked list for tracking purposes, after which customers can no longer cancel those transactions.

Part 3: statistics:

The system should provide the following information:

1. Total number of loans.
2. Number of loans by type.
3. Number of loans by status.

4. Active loans within a specified date range (the user should provide the start and end dates).
5. Customer with the highest number of loans.
6. Customer with the highest account balance.
7. Customer with the lowest account balance.
8. Total number of employees.
9. Number of employees per bank branch.

Note:

1) All arrays and lists should be initialized with default values. When the application is launched, these data structures should be populated from files containing all the necessary data.
2) Ensure that all changes to accounts, loans, and transactions remain consistent and fully functional during the session, allowing the user to navigate the application seamlessly.
3) A user interface is not mandatory, but implementing one can be considered a bonus.
4) Using GitHub is not required, but doing so can also be considered a bonus.