Task 1:

**Planning Discussion:**

**Possible User Stories-Create an account:**

**Customer:**

* + As a new customer, I want to be able to create a new account online so that I can access banking services without having to visit a branch.
  + As a customer, I want to be able to provide my personal information securely online so that my account can be created quickly and easily.
  + As a customer, I want to be able to choose the type of account I want to open, such as a checking or savings account, so that I can select the account that meets my needs.
  + As a customer, I want to be able to review the terms and conditions of the account, including fees and interest rates, so that I can make an informed decision.
  + As a customer, I want to be able to fund my account by linking it to an existing bank account or by making a deposit, so that I can start using my account right away.
  + As a customer, I want to be able to set up online banking services, including online statements, bill pay, and mobile banking, so that I can manage my account from anywhere.
  + As a customer, I want to be able to receive my account information and login details securely so that I can access my account online.
  + As a customer, I want to be able to contact customer support if I have any questions or issues with my account so that I can receive help when I need it.

**Programmer:**

* As a programmer, I want to create a secure and reliable system for opening new bank accounts, so that customer data is protected and the system is dependable.
* As a programmer, I want to ensure that the system adheres to relevant regulations and industry standards, such as Know Your Customer (KYC) and Anti-Money Laundering (AML) regulations, so that the bank is compliant with legal requirements.
* As a programmer, I want to design the system in a modular way, so that different components can be easily tested and updated without affecting other parts of the system.
* As a programmer, I want to use best practices for database design and management, so that customer data is stored securely and can be easily accessed by the system.
* As a programmer, I want to use test-driven development to ensure that the system functions as intended and that any bugs or issues are detected early in the development process.
* As a programmer, I want to incorporate automated testing and continuous integration into the development process, so that the system can be tested quickly and frequently.
* As a programmer, I want to use a modern technology stack and programming language, so that the system is scalable and easy to maintain.
* As a programmer, I want to create a user-friendly interface for customers to open new accounts, so that the process is easy to understand and navigate.

**Tester:**

* As a tester, I want to ensure that the system for opening new bank accounts is easy to use and navigate, so that customers can complete the process without confusion or frustration.
* As a tester, I want to verify that the system accurately captures all required customer data, such as name, address, and identification information, so that the bank has complete and accurate records.
* As a tester, I want to test the system's validation rules, so that customers cannot submit incomplete or invalid information.
* As a tester, I want to verify that the system securely stores customer data and that it is protected from unauthorized access.
* As a tester, I want to ensure that the system is accessible and compatible with different devices and browsers, so that customers can access the system from anywhere.
* As a tester, I want to verify that the system accurately generates account numbers and other identifying information, so that customers can easily identify their accounts.
* As a tester, I want to ensure that error messages are clear and helpful, so that customers can understand what went wrong and how to correct the problem.
* As a tester, I want to test the system's integration with other bank systems, such as credit checks and identity verification services, so that the bank can quickly and accurately process new account requests.

**Possible Iterations-Create an account:**

Iteration 1: Minimum Viable Product (MVP)

* Create a simple user interface for entering basic customer information (name, address, phone number)
* Add basic validation rules to ensure the data entered is complete and accurate
* Store customer information in a database
* Generate a unique account number for each new account
* Verify that the system securely stores customer data

Iteration 2: Additional Information and Verification

* Add fields for additional customer information, such as social security number and date of birth
* Add integration with external services to verify customer identity and perform credit checks
* Improve the validation rules to prevent invalid or fraudulent submissions
* Implement error handling and notifications for incomplete or incorrect submissions

Iteration 3: Enhancements and Improvements

* Add support for multiple account types (savings, checking, etc.)
* Improve the user interface to make it more user-friendly and intuitive
* Optimize the database design for performance and scalability
* Add support for international customers and currencies
* Conduct user testing to identify any areas for improvement or issues with the system

Iteration 4: Security and Compliance

* Implement additional security measures to protect customer data from unauthorized access
* Ensure compliance with industry regulations and standards, such as Know Your Customer (KYC) and Anti-Money Laundering (AML) regulations
* Conduct security and penetration testing to identify any vulnerabilities or weaknesses in the system

Iteration 5: Advanced Features

* Add support for online account opening, with electronic signature and document upload capabilities
* Implement a referral program to incentivize existing customers to refer new customers
* Add integration with the bank's mobile app for a seamless and integrated experience
* Implement a dashboard for bank administrators to monitor new account activity and identify potential issues or fraud.

**Possible User Stories-deposit cash:**

**Customer:**

* As a customer, I want to be able to deposit cash into my account through an ATM, so that I don't have to go to a bank branch during business hours.
* As a customer, I want to be able to deposit cash into my account through a mobile app, so that I can do it from the comfort of my own home.
* As a customer, I want to receive a confirmation email or message after depositing cash into my account, so that I have a record of the transaction.
* As a customer, I want to be able to deposit cash into someone else's account, so that I can easily transfer money to friends or family.
* As a bank teller, I want to be able to accept cash deposits from customers in person, so that I can ensure the accuracy of the transaction and help customers with any questions they may have.

**Programmer:**

* As a programmer, I need to implement a secure and reliable API for depositing cash, so that external systems can integrate with the bank's deposit system.
* As a programmer, I need to ensure that the deposit process is optimized for performance and can handle a large volume of transactions, so that the system can scale effectively.
* As a programmer, I need to implement logic for handling errors and exceptions during the deposit process, so that users are informed of any issues and the system remains stable.
* As a programmer, I need to integrate with third-party services for verification and fraud detection, so that the bank can ensure the security of deposits and prevent fraudulent activity.
* As a programmer, I need to ensure that the system meets all regulatory and compliance requirements related to cash deposits, so that the bank avoids legal and financial penalties.

**Tester:**

* As a tester, I want to ensure that the ATM accurately counts and accepts cash deposits in various denominations and conditions, so that customers can trust the system and avoid errors or malfunctions.
* As a tester, I want to verify that the mobile app allows users to securely and easily deposit cash into their accounts, using clear and intuitive instructions and error messages, so that they can complete the transaction without confusion or frustration.
* As a tester, I want to confirm that the confirmation email or message sent to customers after a cash deposit contains all the relevant information (e.g., date, time, amount, account number), so that they can track their transactions and reconcile their records.
* As a tester, I want to test the feature of depositing cash into someone else's account, making sure that the user is prompted to enter the correct account details and that the recipient is notified of the deposit, to avoid any potential fraud or mistakes.
* As a tester, I want to ensure that the bank tellers can handle cash deposits efficiently and securely, using proper procedures and equipment, and that they can resolve any issues or discrepancies with the customers in a timely and professional manner, to maintain the bank's reputation and customer satisfaction.

**Possible Iterations-deposit cash:**

Iteration 1: ATM Deposit

* Implement ATM deposit feature to allow customers to deposit cash into their accounts without having to visit a bank branch.
* Integrate with existing account system to ensure the deposit is credited to the correct account.
* Implement confirmation message to be displayed on the ATM screen after a successful deposit.

Iteration 2: Mobile App Deposit

* Implement mobile app deposit feature to allow customers to deposit cash from their mobile devices.
* Implement security measures to ensure the safety of customer information during the transaction.
* Allow customers to specify which account the deposit should be credited to.

Iteration 3: Third-Party Deposit

* Implement third-party deposit feature to allow customers to deposit cash into other people's accounts.
* Implement authentication measures to ensure that only authorized individuals can make a deposit.
* Implement confirmation message to be sent to both the depositor and recipient after a successful transaction.

Iteration 4: Recurring Deposit

* Implement recurring deposit feature to allow customers to schedule deposits to be made at regular intervals.
* Allow customers to specify the frequency and amount of the deposits.
* Implement confirmation message to be sent after each successful deposit.

Iteration 5: Bank Teller Deposit

* Implement bank teller deposit feature to allow customers to deposit cash in person at a bank branch.
* Integrate with existing account system to ensure the deposit is credited to the correct account.
* Implement customer service measures to assist customers with any questions or issues during the transaction.

**Possible User stories-withdraw cash:**

* As a customer, I want to be able to withdraw cash from an ATM, so that I can access my funds anytime, anywhere.
* As a customer, I want to be able to withdraw cash from a bank branch, so that I can receive assistance from a teller if needed.
* As a customer, I want to be able to set a daily withdrawal limit, so that I can control my spending and prevent fraud.
* As a programmer, I want to ensure that the withdrawal process is secure and encrypted, so that customer information is protected.
* As a tester, I want to verify that the correct amount of cash is dispensed during a withdrawal, and that the customer's account is debited accordingly.

**Possible Iterations-withdraw cash:**

Iteration 1:

* As a customer, I want to be able to withdraw cash from an ATM, so that I don't have to go to a bank branch during business hours.

Iteration 2:

* As a customer, I want to be able to withdraw cash from an ATM using my debit card, so that I can access my funds easily and securely.

Iteration 3:

* As a customer, I want to be able to select the amount of cash I want to withdraw from an ATM, so that I can get the exact amount of money I need.

Iteration 4:

* As a customer, I want to receive a receipt after withdrawing cash from an ATM, so that I have a record of the transaction.

Iteration 5:

* As a customer, I want to be able to withdraw cash from a bank teller, so that I can access my funds in person if needed.

Iteration 6:

* As a customer, I want to be able to withdraw cash from a bank teller using my debit card or by providing identification and account information.

Iteration 7:

* As a customer, I want to receive a confirmation from the bank teller after withdrawing cash, so that I have a record of the transaction.

Iteration 8:

* As a customer, I want to be able to set a daily limit on the amount of cash I can withdraw, so that I can control my spending and protect my account.

Iteration 9:

* As a customer, I want to be able to temporarily suspend my ability to withdraw cash, in case I lose my debit card or suspect fraudulent activity.

Iteration 10:

* As a programmer, I want to ensure that the withdraw cash feature is secure, reliable, and efficient, so that customers can access their funds easily and securely.

**Possible User stories-transfer from different accounts:**

**Customer:**

* As a customer, I want to be able to transfer funds between my own accounts (e.g. checking and savings), so that I can manage my finances more efficiently.
* As a customer, I want to be able to transfer funds to other bank accounts, both within and outside of the bank, so that I can pay bills or send money to family and friends.
* As a customer, I want to be able to set up recurring transfers, so that I can automatically move money between my accounts on a regular basis.

**Programmer:**

* As a programmer, I need to ensure that the transfer feature is secure, so that customers' sensitive financial information is protected.
* As a programmer, I need to ensure that the transfer feature is user-friendly and intuitive, so that customers can easily complete their transactions.
* As a programmer, I need to ensure that the transfer feature is reliable and efficient, so that customers' transactions are processed quickly and accurately.

**Tester:**

* As a tester, I need to ensure that the transfer feature works as expected for different types of accounts (e.g. checking, savings, credit card), so that customers can transfer funds between any accounts they need to.
* As a tester, I need to ensure that the transfer feature works seamlessly with other features of the bank system (e.g. account balances, transaction history), so that customers have a consistent experience across all aspects of the system.
* As a tester, I need to ensure that the transfer feature works correctly with different currencies and exchange rates, so that customers can transfer money internationally if needed.

**Possible Iterations-transfer from different accounts:**

Iteration 1:

* Customers can transfer funds between their own accounts only
* Transfer feature is available only on the bank's website
* Basic security measures, such as password authentication, are implemented
* Basic error handling, such as insufficient funds or incorrect account numbers, are implemented
* Transfer feature is tested for functionality and basic security

Iteration 2:

* Customers can transfer funds to other accounts within the same bank
* Transfer feature is available on the bank's website and mobile app
* Enhanced security measures, such as two-factor authentication, are implemented
* More advanced error handling, such as account limits and fraud detection, are implemented
* Transfer feature is tested for usability, security, and error handling

Iteration 3:

* Customers can transfer funds to accounts outside of the bank
* International transfers are supported, with accurate exchange rates and currency conversions
* Transfer feature is available on third-party platforms, such as PayPal or Venmo
* Additional security measures, such as biometric authentication or transaction monitoring, are implemented
* Transfer feature is tested for international transfers, third-party integrations, and enhanced security

**Possible User stories-view account details:**

**Customer:**

* As a customer, I want to be able to view my account balance, so that I can check my financial status.
* As a customer, I want to be able to view my transaction history, so that I can monitor my spending and ensure that there are no unauthorized transactions.
* As a customer, I want to be able to view my account details, such as account number and type, so that I have all the necessary information about my accounts.

**Programmer:**

* As a programmer, I need to ensure that customers' account details are secure and protected, so that only authorized individuals can access the information.
* As a programmer, I need to ensure that account details are displayed accurately and clearly, so that customers can easily understand their financial status.
* As a programmer, I need to ensure that the account details feature is easy to navigate, so that customers can quickly find the information they need.

**Tester:**

* As a tester, I need to ensure that account details are displayed accurately and in real time, so that customers can see their most up-to-date financial information.
* As a tester, I need to ensure that the account details feature is accessible on multiple devices, such as mobile phones and tablets, so that customers can view their accounts from anywhere.
* As a tester, I need to ensure that the account details feature is reliable and performs well, even during high traffic times, so that customers can access their information without interruption.

**Possible Iterations-view account details:**

Iteration 1:

* Customers can view their account balance and transaction history on the bank's website only
* Basic security measures, such as password authentication, are implemented
* Account details are displayed in a simple, easy-to-read format
* Account details feature is tested for basic functionality and security

Iteration 2:

* Customers can view their account details on the bank's mobile app as well
* Additional account details, such as interest rates and account type, are displayed
* Account details are displayed in a more visual and interactive format
* Enhanced security measures, such as two-factor authentication, are implemented
* Account details feature is tested for mobile compatibility, enhanced security, and usability

Iteration 3:

* Customers can view their account details on third-party platforms, such as personal finance management apps or investment platforms
* Advanced account details, such as investment portfolios or credit score, are displayed
* Customizable account details settings, such as notifications or alerts, are available
* Additional security measures, such as fraud detection or account freezing, are implemented
* Account details feature is tested for third-party integrations, advanced account details, and enhanced security.

**Implementable tasks:**

**User story1-create an account:**

* Create a registration form that collects necessary information such as name, contact details, and identification documents.
* Implement a validation mechanism to ensure that all required fields are filled and that the provided information is accurate and valid.
* Integrate the registration form with a secure database to store new account details and customer information.
* Implement a verification process for new accounts, such as sending a confirmation email or text message to the customer.
* Develop a user interface for customers to log in to their newly created accounts and access banking services.
* Implement security measures such as password requirements, two-factor authentication, and encryption to protect customer information and prevent fraud.
* Develop a system for customers to link their new account to their existing bank accounts, if applicable.
* Implement error handling mechanisms for common registration issues, such as duplicate account numbers or invalid identification documents.
* Provide clear and detailed instructions for customers to follow during the registration process, including any required documentation.
* Test the registration process thoroughly for functionality, security, and usability, and make necessary improvements based on feedback.

**Test-cases:**

* Verify that the user can access the account creation page from the bank's website.
* Verify that the account creation page is user-friendly and easy to understand.
* Verify that the user can select the type of account they want to create.
* Verify that the user can provide all the necessary personal information, such as name, address, and contact details.
* Verify that the user can set up login credentials, such as a username and password.
* Verify that the user can provide funding details, such as a credit card or bank account number, to set up the new account.
* Verify that the user can review their account information and make any necessary changes before submitting the account application.
* Verify that the user receives a confirmation email or message after submitting the application.
* Verify that the user can log in to their new account after it has been approved.
* Verify that the user can access basic account features, such as checking their balance or making a deposit, after logging in.

**User story2-deposit cash:**

* Develop a secure online portal for third-party deposits, where users can enter their deposit information and payment details.
* Implement a validation mechanism to ensure that all required fields are filled and that the provided information is accurate and valid.
* Integrate the deposit portal with a secure database to store third-party deposit information and payment details.
* Implement a verification process for third-party deposits, such as sending a confirmation email or text message to the depositor.
* Develop a user interface for bank employees to review and approve third-party deposits, and to ensure that the deposits are valid and legitimate.
* Implement security measures such as encryption and two-factor authentication to protect sensitive deposit information and prevent fraud.
* Develop a system for customers to track the status of their third-party deposits and receive updates on the approval process.
* Implement error handling mechanisms for common deposit issues, such as insufficient funds or invalid payment details.
* Provide clear and detailed instructions for third-party depositors to follow during the deposit process, including any required documentation.
* Test the deposit process thoroughly for functionality, security, and usability, and make necessary improvements based on feedback.

**Test Cases:**

1. Positive test case: Valid deposit amount

* Input: Deposit amount of $500
* Expected Output: Deposit is successful, and the account balance is increased by $500.

1. Positive test case: Multiple deposits in one transaction

* Input: Two deposit amounts of $200 and $300
* Expected Output: Both deposits are successful, and the account balance is increased by $500.

1. Negative test case: Invalid deposit amount

* Input: Deposit amount of $0
* Expected Output: Deposit fails, and an error message is displayed stating that the deposit amount must be greater than 0.

1. Negative test case: Invalid third-party account number

* Input: Invalid account number
* Expected Output: Deposit fails, and an error message is displayed stating that the account number is invalid or does not exist.

1. Negative test case: Insufficient funds

* Input: Deposit amount greater than the account balance
* Expected Output: Deposit fails, and an error message is displayed stating that there are insufficient funds for the deposit.

1. Positive test case: International deposit

* Input: Deposit amount of 100 euros
* Expected Output: Deposit is successful, and the account balance is increased by the equivalent amount in US dollars.

1. Negative test case: Suspicious deposit activity

* Input: Large deposit amount from an unknown third party
* Expected Output: Deposit is flagged for review, and the bank's security team is notified to investigate the transaction.

1. Positive test case: Automated deposit confirmation

* Input: Deposit confirmation email or text message
* Expected Output: Customer receives automated confirmation of the deposit and the new account balance.

1. Negative test case: Duplicate deposit

* Input: Two identical deposit transactions in a short time period
* Expected Output: The first deposit is successful, and the second deposit fails with an error message stating that a duplicate transaction has been detected.

1. Positive test case: Successful deposit with third-party service integration

* Input: Deposit made through a third-party payment service such as PayPal or Venmo
* Expected Output: Deposit is successful, and the account balance is increased by the deposited amount.

Task 2:

**Project planning: using Gantt Chart**

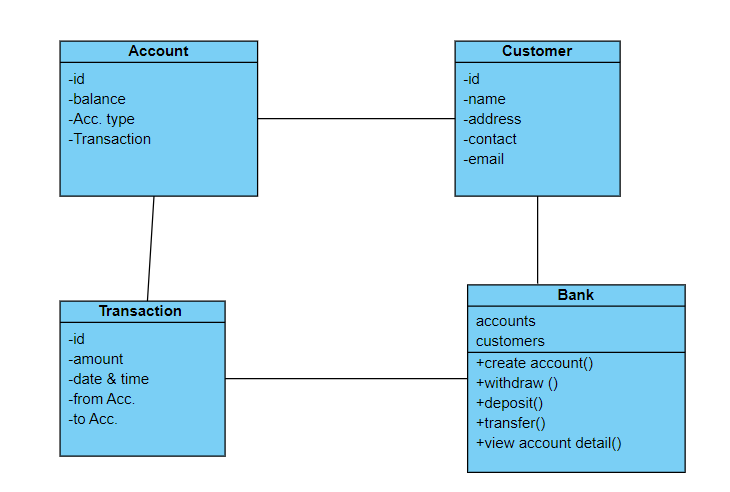
**Gantt Chart:**

Task 3:

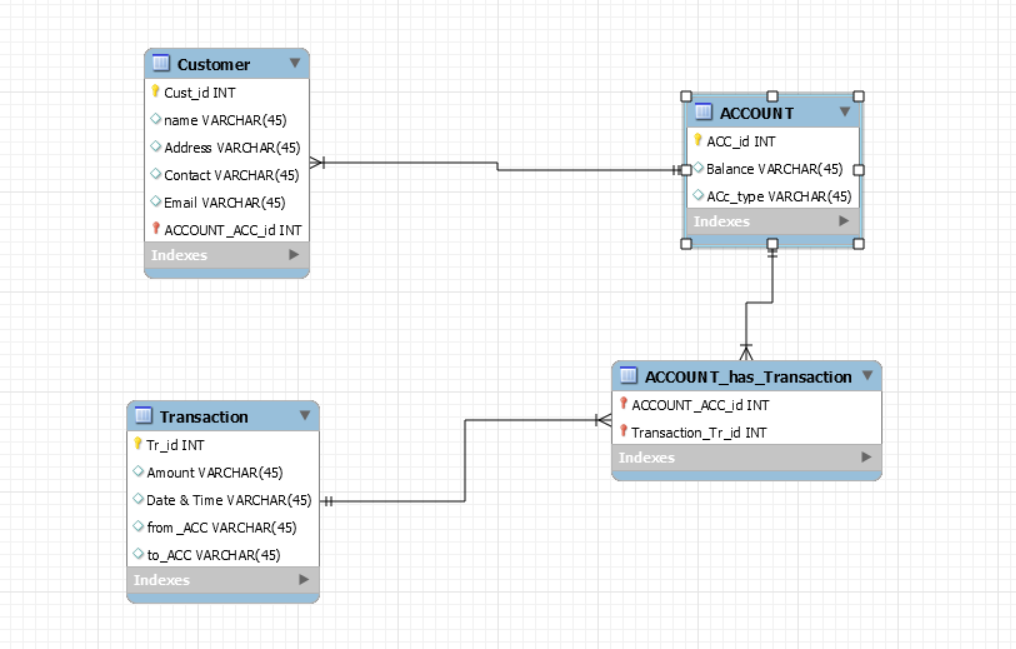
**Prototype design:**

**System design:** structural diagrams

**Class diagram:**

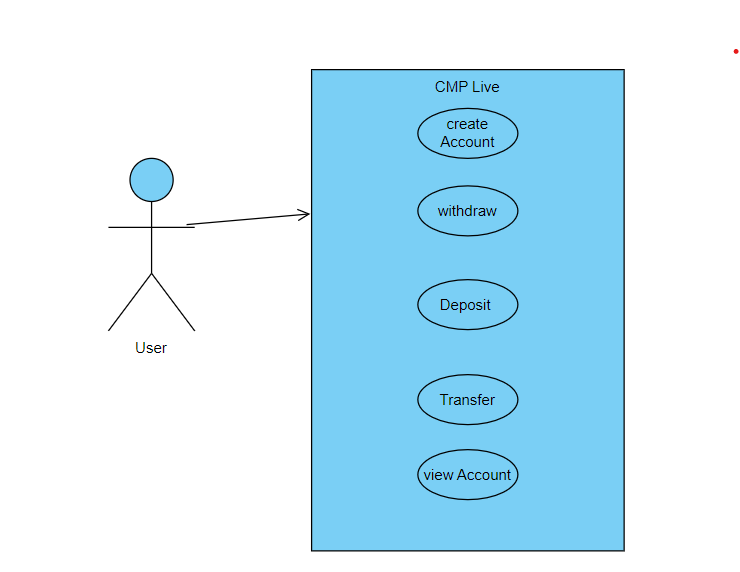
****

**Entity Relationship Diagram:**

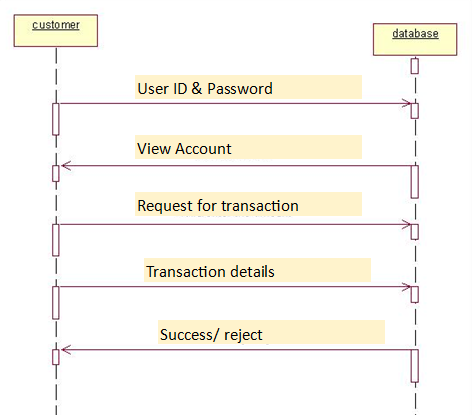
****

**behavioral diagrams:**

**Use-case Diagram:**



**Sequence Diagram:**



Task 4:

To maintain versions of all source files of the banking software, we can use a code control system such as Git. Here's an example of how to use Git to manage the versions of our banking software:

* Create a Git repository: Navigate to the directory where your source files are stored and create a new Git repository by running the command git init. This will create a hidden .git directory which stores all the version control data.
* Add files to the repository: Use the command git add <filename> to add all the source files to the repository. You can use git add . to add all files in the current directory.
* Commit changes: Use the command git commit -m "Initial commit" to create the first commit in the repository. This saves a snapshot of the current state of the files in the repository.
* Make changes to the code: Edit the source files as needed, adding new features or fixing bugs.
* Stage changes: Use the command git add <filename> to stage the changes to the file for the next commit.
* Commit changes: Use the command git commit -m "Description of changes" to commit the changes to the repository.
* View commit history: Use the command git log to view a history of all the commits made to the repository, including the commit message, date, and author.
* Revert changes: If you need to undo changes made in a commit, use the command git revert <commit-hash> to create a new commit that reverses the changes made in the specified commit.
* Create branches: Use the command git branch <branch-name> to create a new branch in the repository. This allows you to work on new features or fixes without affecting the main codebase.
* Merge branches: Once a feature or fix is complete, use the command git merge <branch-name> to merge the changes from the branch into the main codebase.

**Task 5:**

#include <iostream>

#include <string>

using namespace std;

class Account {

public:

Account() {} // Default constructor

Account(int id, string name, float balance) { // Constructor with parameters

acc\_id = id;

acc\_name = name;

acc\_balance = balance;

}

void deposit(float amount) {

acc\_balance += amount;

cout << "Amount deposited: " << amount << endl;

cout << "New balance: " << acc\_balance << endl;

}

void withdraw(float amount) {

if (acc\_balance < amount) {

cout << "Insufficient balance" << endl;

}

else {

acc\_balance -= amount;

cout << "Amount withdrawn: " << amount << endl;

cout << "New balance: " << acc\_balance << endl;

}

}

void transfer(Account& account, float amount) {

if (acc\_balance < amount) {

cout << "Insufficient balance" << endl;

}

else {

acc\_balance -= amount;

account.acc\_balance += amount;

cout << "Amount transferred: " << amount << endl;

cout << "New balance for account " << acc\_id << ": " << acc\_balance << endl;

cout << "New balance for account " << account.acc\_id << ": " << account.acc\_balance << endl;

}

}

void display() {

cout << "Account ID: " << acc\_id << endl;

cout << "Account name: " << acc\_name << endl;

cout << "Account balance: " << acc\_balance << endl;

}

private:

int acc\_id;

string acc\_name;

float acc\_balance;

};

int main() {

Account account1(1001, "John Doe", 5000.00);

Account account2(1002, "Jane Doe", 8000.00);

int choice;

float amount;

do {

cout << endl;

cout << "\*\*\*\*\* Bank System \*\*\*\*\*" << endl;

cout << "1. Create an account" << endl;

cout << "2. Withdraw cash" << endl;

cout << "3. Deposit cash" << endl;

cout << "4. Transfer cash from one account to another" << endl;

cout << "5. View account details" << endl;

cout << "6. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: {

int id;

string name;

float balance;

cout << "Enter account ID: ";

cin >> id;

cout << "Enter account name: ";

cin >> name;

cout << "Enter account balance: ";

cin >> balance;

Account new\_account(id, name, balance);

cout << "Account created successfully" << endl;

new\_account.display();

break;

}

case 2: {

cout << "Enter amount to withdraw: ";

cin >> amount;

account1.withdraw(amount);

break;

}

case 3: {

cout << "Enter amount to deposit: ";

cin >> amount;

account1.deposit(amount);

break;

}

case 4: {

cout << "Enter amount to transfer: ";

cin >> amount;

account1.transfer(account2, amount);

break;

}

case 5: {

account1.display();

break;

}

Default:

Cout<<”invalid command’<<endl;

}

}

While(choice<=6);

Getch();

}

**Task 6:**

Run this code in CPP compiler