$ BANK MANAGEMENT SYSTEM $

UML REPORT

Academic Year 2022-23 ODD SEMESTER

Department with Specialization: Computer Science and Engineering with Specialisation in Artificial Intelligence & Machine Learning

Semester : III
Course Code : 18CSC202J
Course Title : OBJECT ORIENTED DESIGN AND PROGRAMMING

Submitted By

Kavya Reddy Vutukuri (Reg No: RA2111026010261)
Manoj Sai Abhithesh (Reg No: RA2111026010229)

Under the Guidance of
Dr. M. Uma
Associate Professor

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203
NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY Project Report** titled **"BANK MANAGEMENT SYSTEM"** is the Bonafide work of

Kavya Reddy Vutukuri (RA2111026010261)
Manoj Sai Abhithesh (RA2111026010229)

who undertook the task of completing the project within the allotted time.

Signature of the Guide
Dr. M.Uma
Associate Professor
Department of CINTEL,
SRM Institute of Science and Technology

Signature of the II Year Academic Advisor

Professor and Head
Department of CINTEL
SRM Institute of Science and Technology

About the course:-

18CSC202J/ 18AIC203J - Object Oriented Design and Programming are 4 credit courses with **L T P C as 3-0-2-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

Objectives:

The student should be made to:

- Learn the basics of OOP concepts in C++
- Learn the basics of OOP analysis and design skills.
- Be exposed to the UML design diagrams.
- Be familiar with the various testing techniques

Course Learning Rationale (CLR): The purpose of learning this course is to:

1. Utilize class and build domain model for real-time programs
2. Utilize method overloading and operator overloading for real-time application development programs
3. Utilize inline, friend and virtual functions and create application development programs
4. Utilize exceptional handling and collections for real-time object-oriented programming applications
5. Construct UML component diagram and deployment diagram for design of applications
6. Create programs using object-oriented approach and design methodologies for real-time application development

Course Learning Outcomes (CLO): At the end of this course, learners will be able to:

1. Identify the class and build domain model
2. Construct programs using method overloading and operator overloading
3. Create programs using inline, friend and virtual functions, construct programs using
   standard templates
4. Construct programs using exceptional handling and collections
5. Create UML component diagram and deployment diagram
6. Create programs using object oriented approach and design methodologies

Table 1: Rubrics for Laboratory Exercises
(Internal Mark Splitup:- As per Curriculum)

CLAP-1	5=(2(E-lab Completion) + 2(Simple Exercises)(from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-2	7.5=(2.0(E-lab Completion)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-3	7.5=(2.0(E-lab Completion(80 Pgms)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	**2 Mark -** E-lab Completion **80 Program** Completion from 10 Session (Each session min 8 program) **2 Mark -** Code to UML conversion GCR Exercises **3.5 Mark - Hacker Rank** Coding challenge completion
CLAP-4	5= 3 (Model Practical) + 2(Oral Viva)	• **3 Mark** – Model Test • **2 Mark** – Oral Viva
Total	25	

COURSE ASSESSMENT PLAN FOR OODP LAB

S.No	List of Experiments	Course Learning Outcomes (CLO)	Blooms Level	PI	No of Programs in each session
1.	Implementation of I/O Operations in C++	CLO-1	Understand	2.8.1	10
2.	Implementation of Classes and Objects in C++	CLO-1	Apply	2.6.1	10
3,	To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram.	CLO-1	Analysis	4.6.1	Mini Project Given
4.	Implementation of Constructor Overloading and Method Overloading in C++	CLO-2	Apply	2.6.1	10
5.	Implementation of Operator Overloading in C++	CLO-2	Apply	2.6.1	10
6.	Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams	CLO-2	Analysis	4.6.1	Mini Project Given
7.	Implementation of Inheritance concepts in C++	CLO-3	Apply	2.6.1	10
8.	Implementation of Virtual function & interface concepts in C++	CLO-3	Apply	2.6.1	10
9.	Using the identified scenarios in your project, draw relevant state charts and activity diagrams.	CLO-3	Analysis	4.6.1	Mini Project Given
10.	Implementation of Templates in C++	CLO-3	Apply	2.6.1	10
11.	Implementation of Exception of Handling in C++	CLO-4	Apply	2.6.1	10
12.	Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram.	CLO-5	Analysis	4.6.1	Mini Project Given
13.	Implementation of STL Containers in C++	CLO-6	Apply	2.6.1	10
14.	Implementation of STL associate containers and algorithms in C++	CLO-6	Apply	2.6.1	10
15.	Implementation of Streams and File Handling in C++	CLO-6	Apply	2.6.1	10

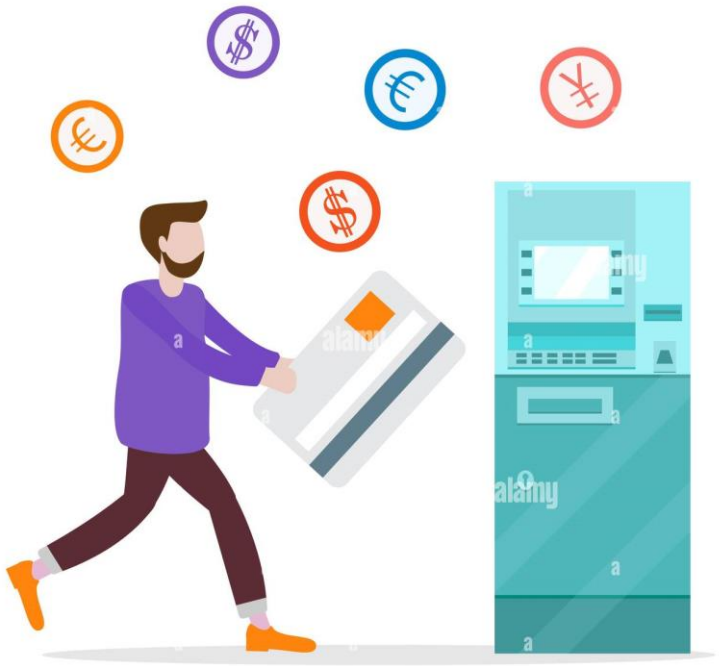
LIST OF EXPERIMNENTS FOR UML DESIGN AND MODELLING:

To develop a mini project by following the exercises listed below.

1. To develop a problem statement.
2. Identify Use Cases and develop the Use Case model.
3. Identify the conceptual classes and develop a domain model with UML Class diagram.
4. Using the identified scenarios, find the interaction between objects and represent them
   using UML Sequence diagrams.
5. Draw relevant state charts and activity diagrams.
6. Identify the User Interface, Domain objects, and technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

Suggested Software Tools for UML:
StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit

Table of Contents

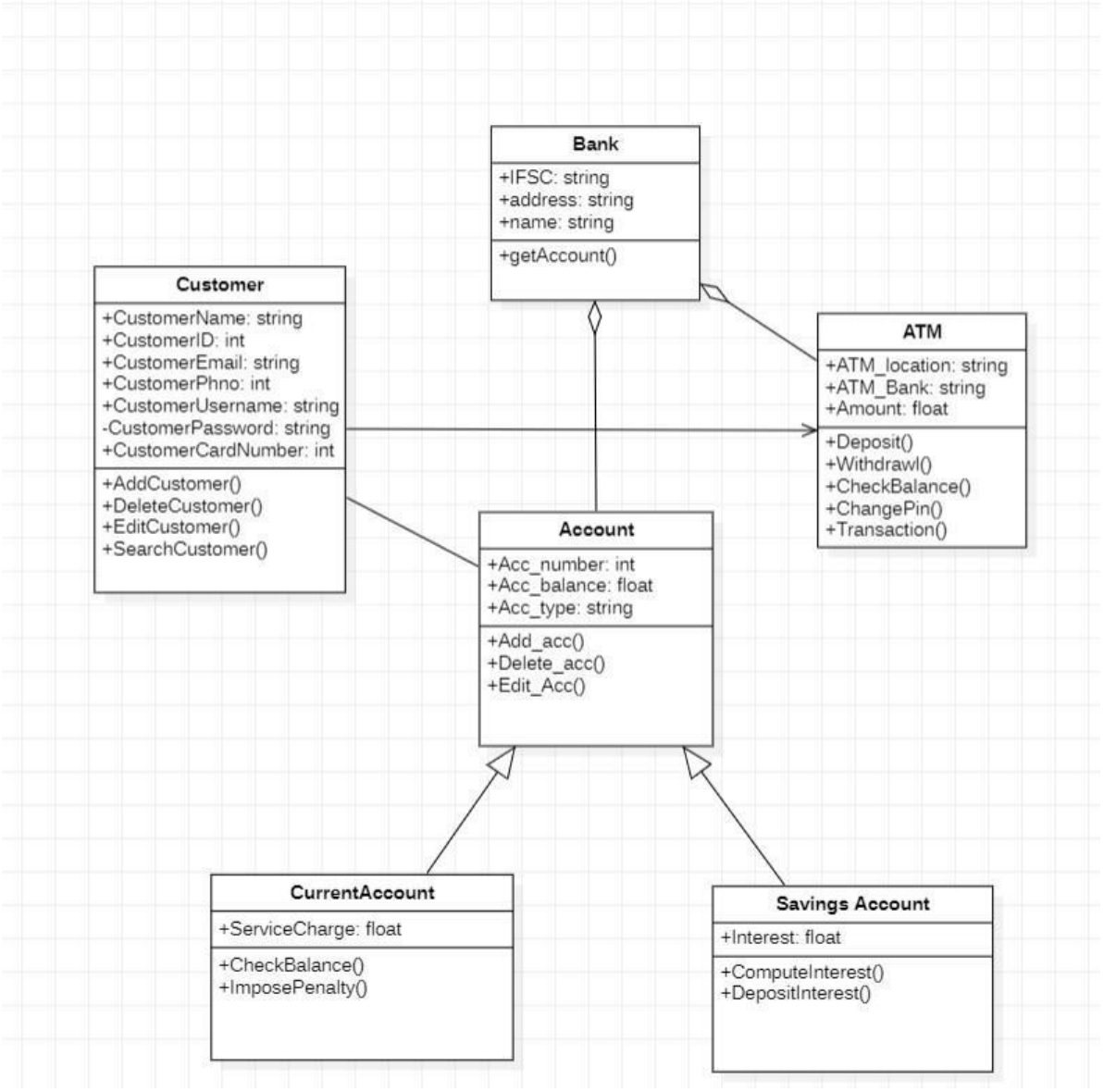


Abstract:

The Bank Management System is an application for maintaining a person's account in a bank. In this project I tried to show the working of a banking account system and cover the basic functionality of a Bank Management System. To develop a project for solving financial applications of a customer in banking environment to nurture the needs of an end banking user by providing various ways to perform banking tasks. Also, to enable the user's workspace to have additional functionalities which are not provided under a conventional banking project. The Bank Account Management System undertaken as a project is based on relevant technologies. The main aim of this project is to develop software for Bank Account Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software. The project analyses the system requirements and then come up with he requirements specifications. It studies other related systems and then come up with system specifications.

The system is then designed in accordance with specifications to satisfy the requirements. The system is designed as an interactive and content management system. The content management system deals with data entry, validation confirm and updating whiles the interactive system deals with system interaction with the administration and users. Thus, above features of this project will save transaction time and therefore increase the efficiency of the system.

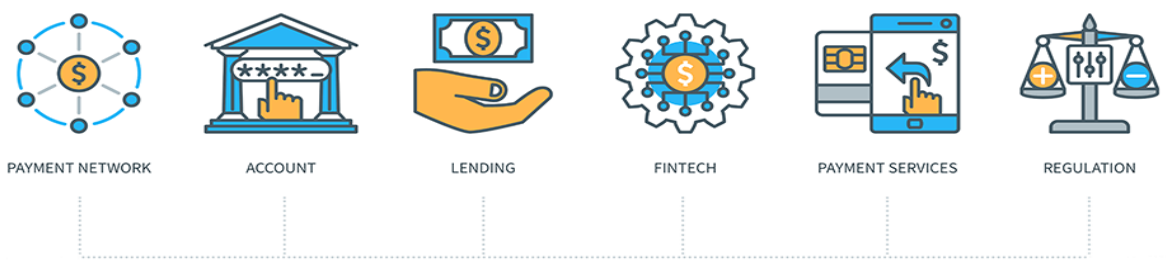
Class Diagram:



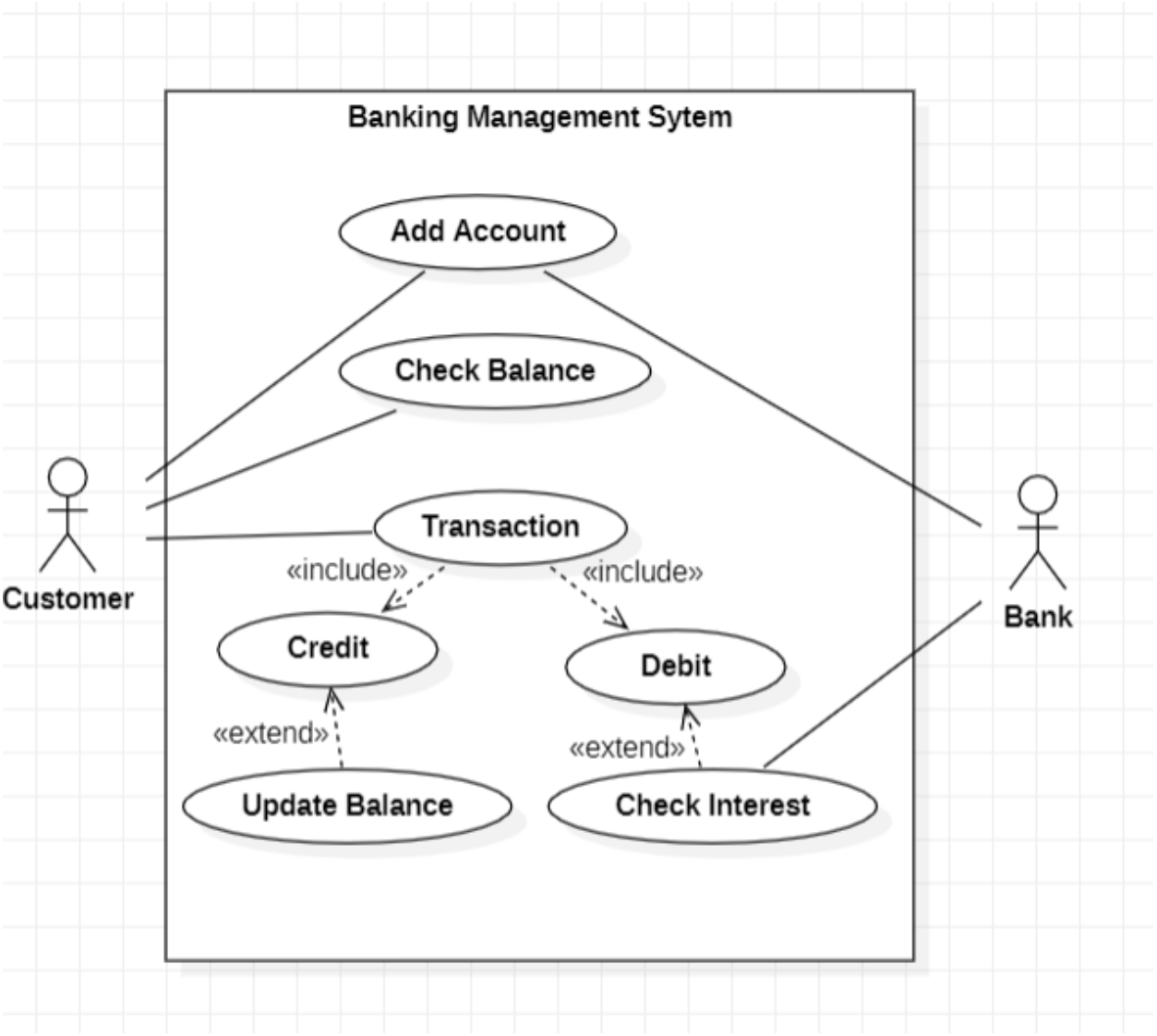
Relationship Among Classes:

Class Bank is base class that holds the IFSC code, address of the ban and name as its attributes and get Account as behavior. Class ATM and Class Account are derived classes of Bank linked to it by a relationship called Aggregation. An aggregation specifies a whole-part relationship between an aggregate (a whole) and a constituent part, where the part can exist independently from the aggregate. Aggregations are denoted by a hollow-diamond adornment on the association.

Class Account consists of account number, balance and account type as attributes and AddAccount, Delete Account and Edit Account as its behavior. Furthermore, Current Account and Savings Account are linked to Account through generalization or inheritance. Class ATM and Customer are linked by directed association and Class Customer is linked to Account by association.



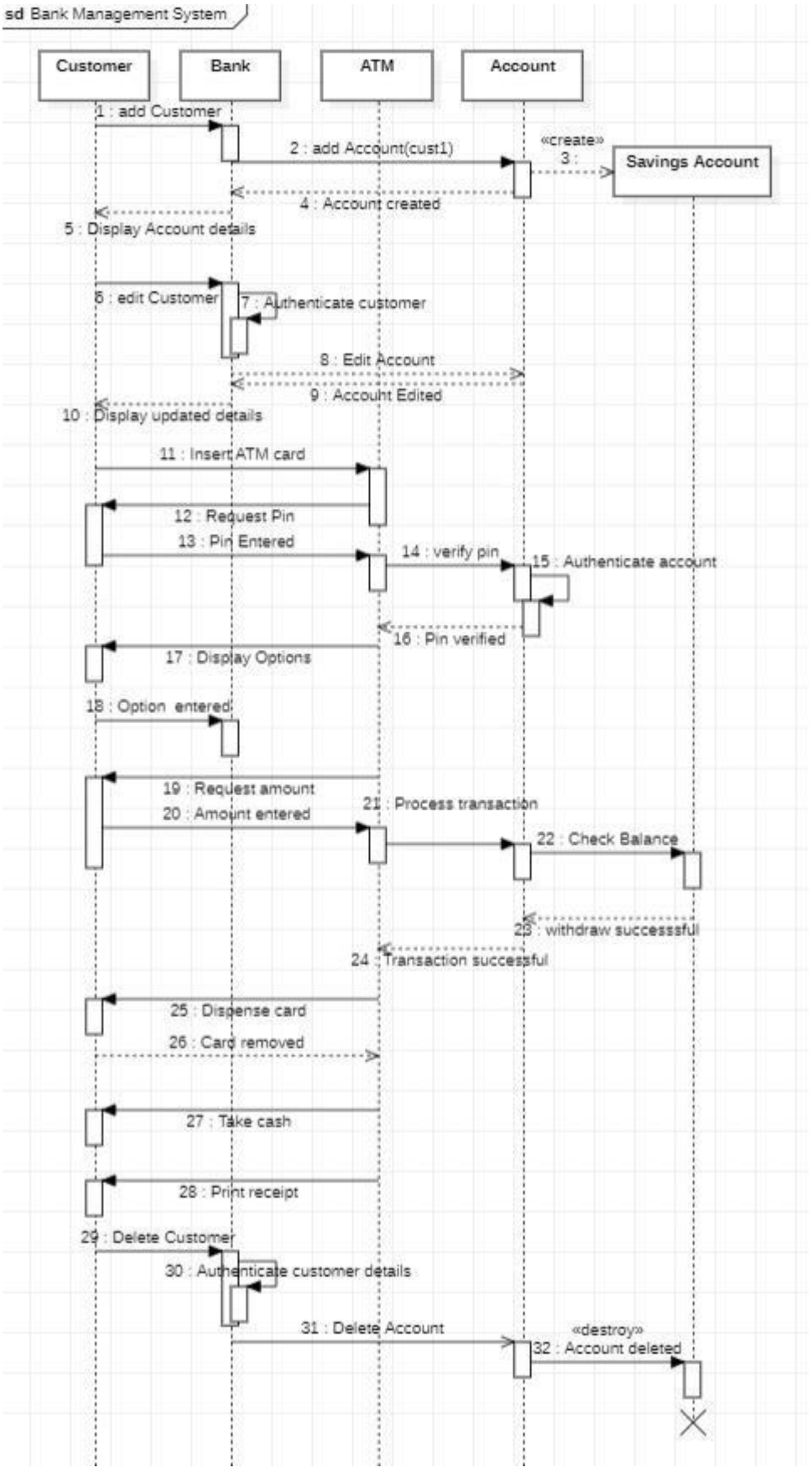
Use Case Diagram



This use case diagram consists of two actors, Customer and Bank (System). The customer requests to add account. The banking system creates a new account for the customer. The customer checks account balance. The customer initiates transaction. The transaction includes credit and debit. Credit transaction will extend to updating the account balance. On the other hand, the debit transaction would extend to checking the interest and debit the amount. The banking system is in association with check interest and add account. Whereas the customer has an association relationship with add account, check balance and transaction processes.

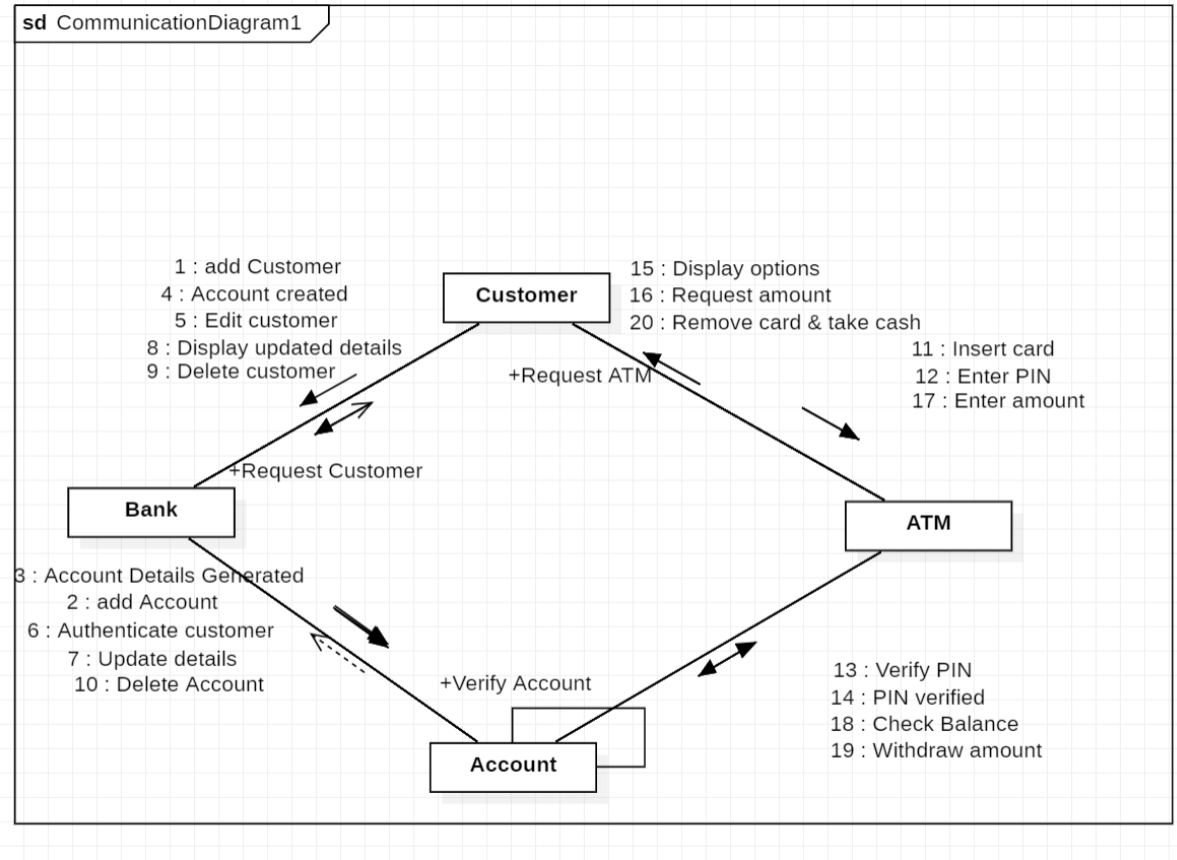
Interaction Diagram :

Sequence Diagram :



In this sequence diagram 5 lifelines are created: Customer, Bank, ATM, Account and Savings Account. Following operations take place in the sequence diagram. Firstly, Customer requests to add customer to Bank. Bank requests to create account. Account Created. Account details displayed to customer. Then customer requests to edit account. Bank authenticates customer. Bank edits Account details. Account details edited and updated details are displayed to customer. Customer inserts ATM card, ATM requests PIN from the customer, Customer enters PIN, PIN verified by the Account, Account authenticated. PIN verified message sent to ATM. ATM displays options to Customer. Customer enters option. ATM requests amount. Amount entered by Customer. Transaction is then processed by the Account. Account requests to check balance from savings account. Withdraw successful message sent to Account. Account sends Transaction Successful message to ATM. ATM asks customer to dispense card. Card removed by customer, then ATM asks customer to take cash. Receipt printed by ATM to customer. Customer requests Bank to delete customer, Bank authenticates customer details and deletes account. Lifeline Savings Account destroyed.

Collaboration Diagram



This collaboration diagram focuses on the relationship of objects- how they associate and connect through messages in a sequence rather than interactions. In this collaboration diagram, the interaction between various lifelines is shown such as Bank, Customer, ATM and Account.

The following operations are shown in this collaboration diagram.

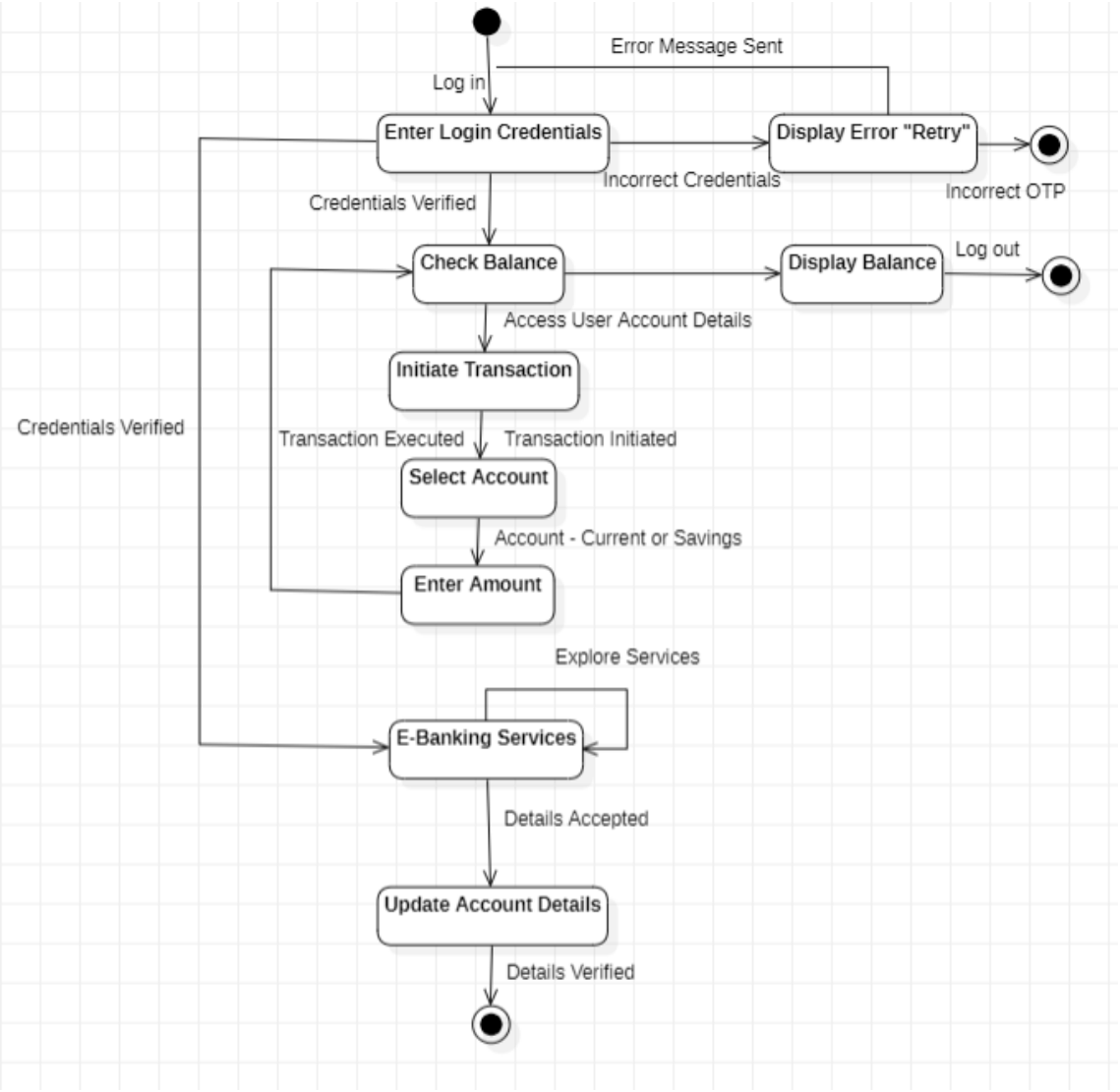
1. addCustomer- Customer sends message to Bank to add new customer account.
2. addAccount- Bank sends message to Account to create account. Account replies that account details are generated to Bank. Bank sends a reply to customer that account is created.
3. EditCustomer: Customer sends a message to Bank to edit customer details. Bank sends a message to Account to Authenticate customer.

4. Account verifies customer credentials. Bank sends a message to customer to update details. Bank replies to customer by displaying updated details.

5. DeleteCustomer: Customer sends a request to Bank delete customer account. Bank sends a message to Account to authenticate customer, Account deletes customer account.
6. Customer inserts card into the ATM, ATM sends a reply to enter PIN. PIN entered.
7. ATM sends a message to Account to verify PIN, PIN verified message sent to ATM.
8. ATM displays options to Customer.
9. Customer enters amount to ATM.
10. ATM sends a message to Account to check account balance.
11. Account withdraws amount and sends message to ATM.
12. ATM sends message to Customer to remove card and take cash.

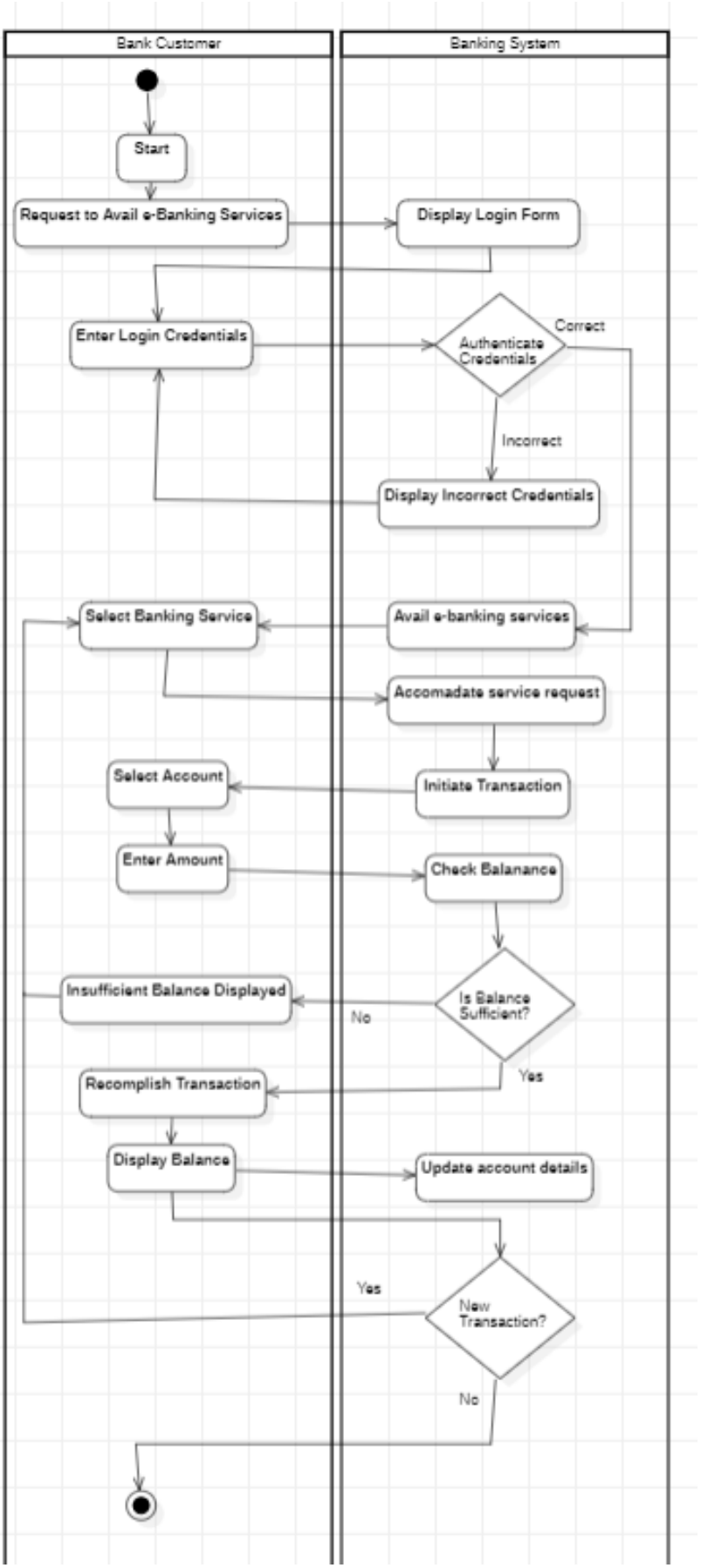
Behavioral Diagram:

State Chart Diagram



In the state chart diagram, the user logs into the e-Banking service. The login credentials are entered. If incorrect login credentials are entered error message displayed and it reaches final state. Otherwise, if credentials entered correctly, balance is checked. User opts to check balance display balance and log out. If user opts to access account details, then initiate transaction. Select type of account, whether current or savings. Then, enter amount. Transaction is then executed, and process returns to the balance stage. Once, the login credentials are verified the user accesses the e-banking services offered. The user explores the services. Details are accepted and account details are updated of the user. Then the updated details are verified, and process reaches to final state.

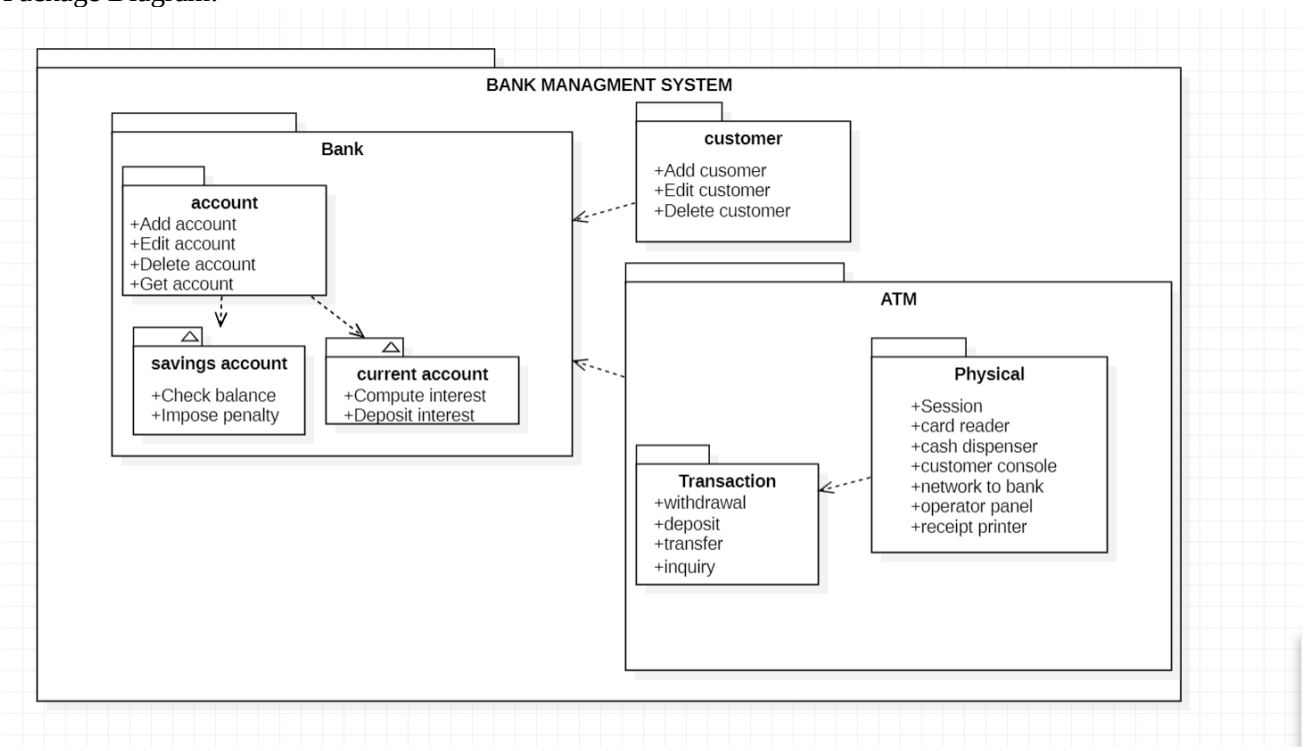
Activity Diagram



Activity Diagram

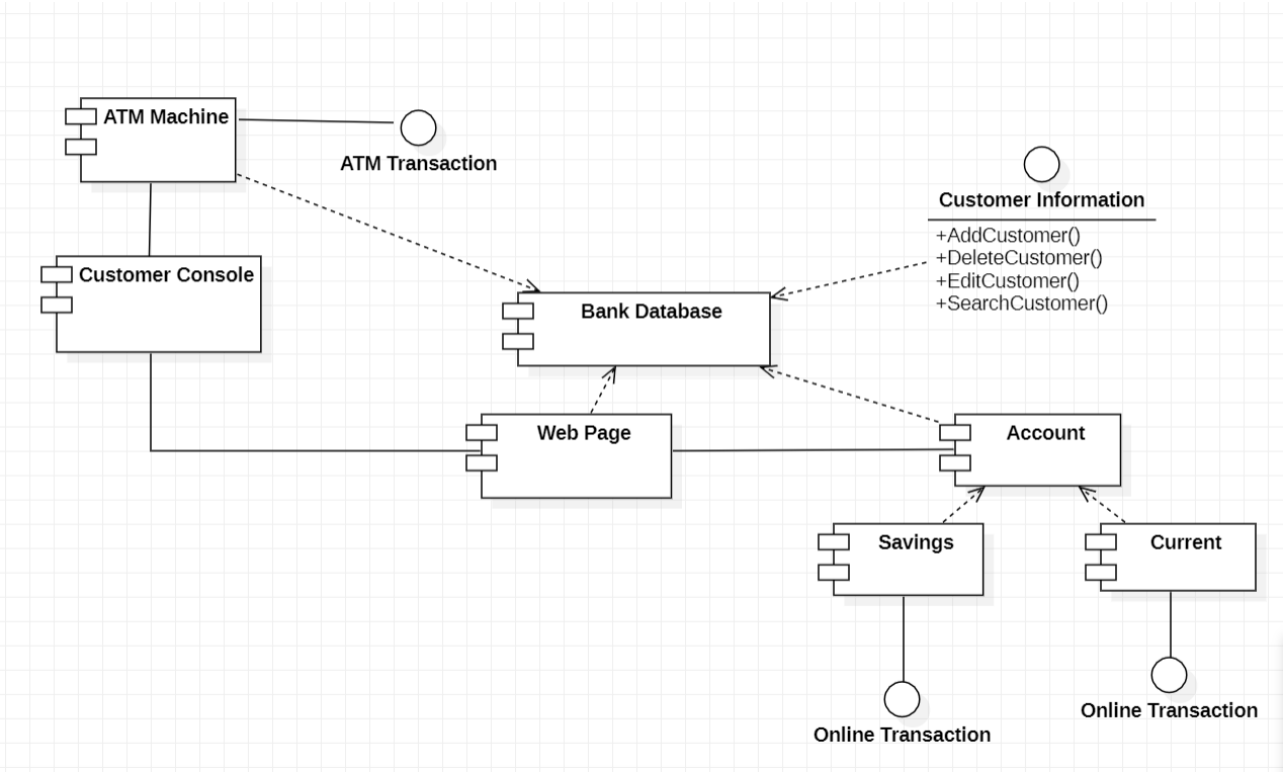
In this Activity Diagram, initially the user requests to avail e-banking services. The banking system displays the login page. The banking customer enters login credentials. The login credentials are verified. If the login credentials are correct, then e-banking services are availed. If wrong credentials are entered, error message displayed and returns to login page. If login credentials correct, e-services are availed. The customer selects the banking services. The requested service is accommodated by the banking system. The banking system then initiates the transaction. The user selects the account type (Current or Savings). Then customer enters the amount. Then, the balance is checked. If the balance is sufficient then process the transaction and display updated balance. If balance is insufficient, then return to the home page to select another banking service. Then, the user account details are updated. If the customer avails a new transaction, then the process is repeated otherwise the process ends and action reaches the final state.

Package Diagram:



The bank management system is a package that consists of sub packages Bank, Customer and ATM. The Bank package consists of account information. The Account package consists of member functions to add, edit, delete and access account. The Account package is related to two sub packages Savings Account and Current Account. Savings Account consists of checking balance, impose penalty functionalities. In addition, current account the member functions include compute interest, deposit interest. Customer package is linked to Bank Package. ATM package consists of two sub packages Transaction and Physical. Transaction package consists of functions such as withdrawal, deposit, transfer and inquiry. Whereas the Physical package consists of ATM session, card reader, cash dispenser, customer console, network to bank, operator panel and receipt printer. ATM package is linked to Bank Package through dependency relationship.

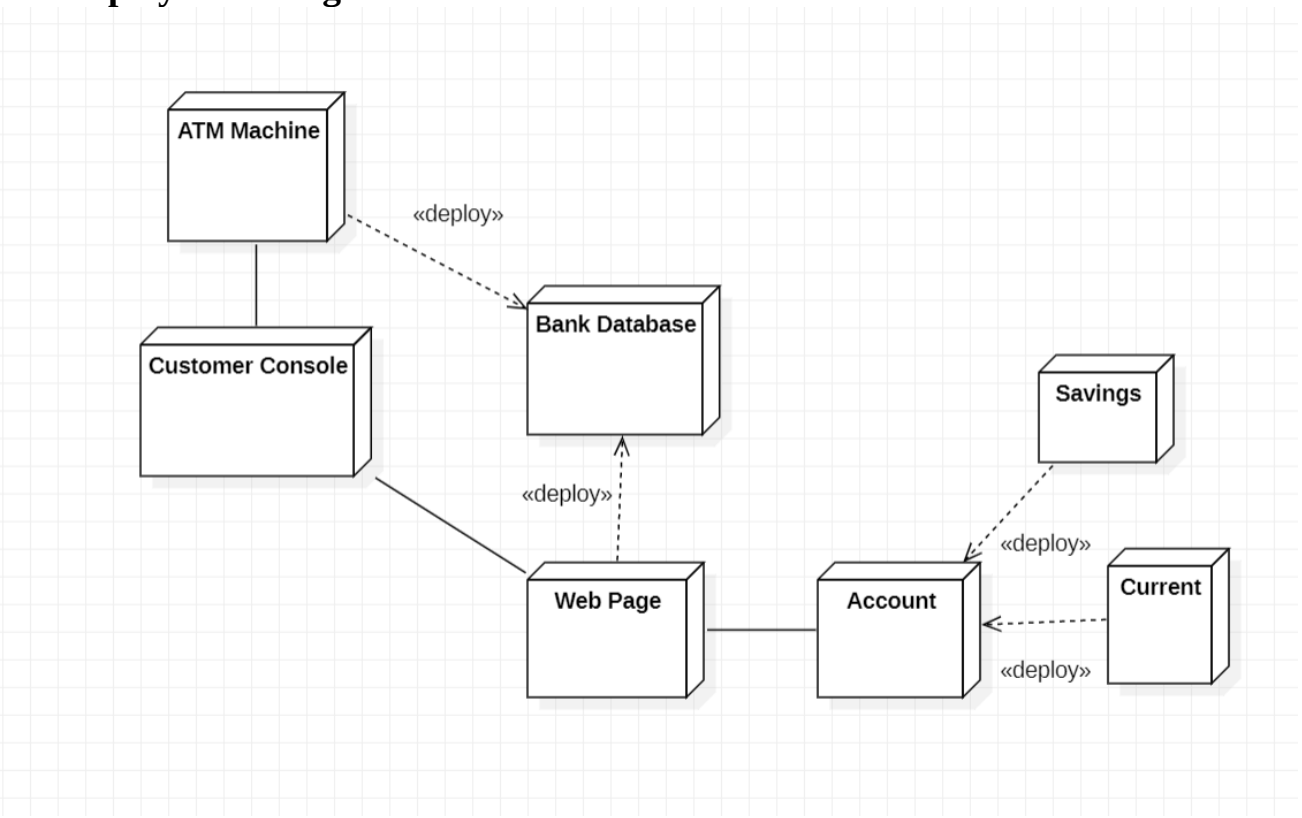
Component Diagram:



The customer data flows into the Bank database. The customer information is a required interface to store customer details. ATM transaction interface links with the ATM Machine.

Data passes from ATM transaction to ATM Machine and then to Customer Console. From Customer console the data is transferred to Web page of the banking site which access the bank database. From web page the data flows in the account component. Account components consist of two subcomponents savings and transaction. Savings and Current component transfers data to the online transactions.

Deployment Diagram:



The deployment diagram consists of Bank Database node. ATM machine node, web page is deployed to bank database. ATM Machine is linked to Customer Console. Customer console transfers data to the web page node. The web page node transfers data to Account node. Two nodes' Savings and Current are deployed to Account node. The ATM machine and customer console are the hardware components whereas the web page, bank database and account are the software components. Customer interacts with the ATM machine and web page. The ATM machine accesses the bank database to enable transaction physically the web page deploys to bank database to enable online transaction by transfer of data from account.

Conclusion:

Here are the UML Diagrams that composes a Bank Management System. Each of the UML Diagrams has a major role in achieving a well-developed and functional Bank Management System.

The UML Diagrams works together to achieve the most desired functions of the Bank Management System Project. All of these were designed to guide programmers and beginners about the behavior and structure of Bank Management System.

By completing all the given Diagrams, the Bank Management Project System development would be much easier and attainable. So those UML diagrams were given to teach you and guide you through your project development journey. You can use all the given UML diagrams as your reference or have them for your project development. The ideas presented in UML Diagrams were all based on Bank Management System requirements.

References

https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm

https://www.javatpoint.com/uml-diagrams

https://docs.staruml.io/working-with-uml-diagrams/use-case-diagram