

# **BANK MANAGEMENT SYSTEM**

---

## **PROJECT REPORT**

**18CSC202J/ 18AIC203J - OBJECT ORIENTED DESIGN AND  
PROGRAMMING LABORATORY**

**(2018 Regulation)**

**II Year/ III Semester**

**Academic Year: 2022 -2023**

**By**

**MALE CHARANJEETH KUMAR(RA2111026010082)**

**THANGALA NITHIN KUMAR REDDY(RA2111026010117)**

**Under the guidance of**

**Dr. S. AMUDHA**

**Assistant Professor**

**Department of Computational Intelligence**



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Kattankulathur, Kancheepuram**

**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 **MALE CHARANJEETH KUMAR(RA2111026010082), THANGALA NITHIN KUMAR REDDY (RA2111026010117)**

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

### **Signature of the Guide**

Dr. S. Amudha

### **Assistant Professor**

Department of CINTEL,

SRM Institute of Science and Technology

### **Signature of the II Year Academic Advisor**

Dr.N.Arivazhagan

### **Professor and Head**

Department of CINTEL

SRM Institute of Science and Technology

**About the course:-**

18CSC202J/ 8AIC203J - 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)

<b>CLAP-1</b>	5=(2(E-lab Completion) + 2(Simple Exercises)( from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
<b>CLAP-2</b>	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
<b>CLAP-3</b>	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)	<b>2 Mark - E-lab Completion 80 Program</b> Completion from 10 Session (Each session min 8 program) <b>2 Mark - Code to UML</b> conversion GCR Exercises <b>3.5 Mark - Hacker Rank</b> Coding challenge completion
<b>CLAP-4</b>	5= 3 ( Model Practical) + 2( Oral Viva)	<ul style="list-style-type: none"> <li>• <b>3 Mark</b> – Model Test</li> <li>• <b>2 Mark</b> – Oral Viva</li> </ul>
<b>Total</b>	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 EXPERIMENTS 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

## **ABSTRACT**

The Bank Account 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 Account Management System. To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks. Also to enable the user's work space 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. This project is developed using c++ use for database connection. Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer, while proving compliance and staying on the schedule and within budget. The impact of a poorly expressed requirement can bring a business out of compliance or even cause injury or death. Requirements definition and management is an activity that can deliver a high, fast return on investment. The project analyzes the system requirements and then comes up with the 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 design is then implemented with c++. 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.

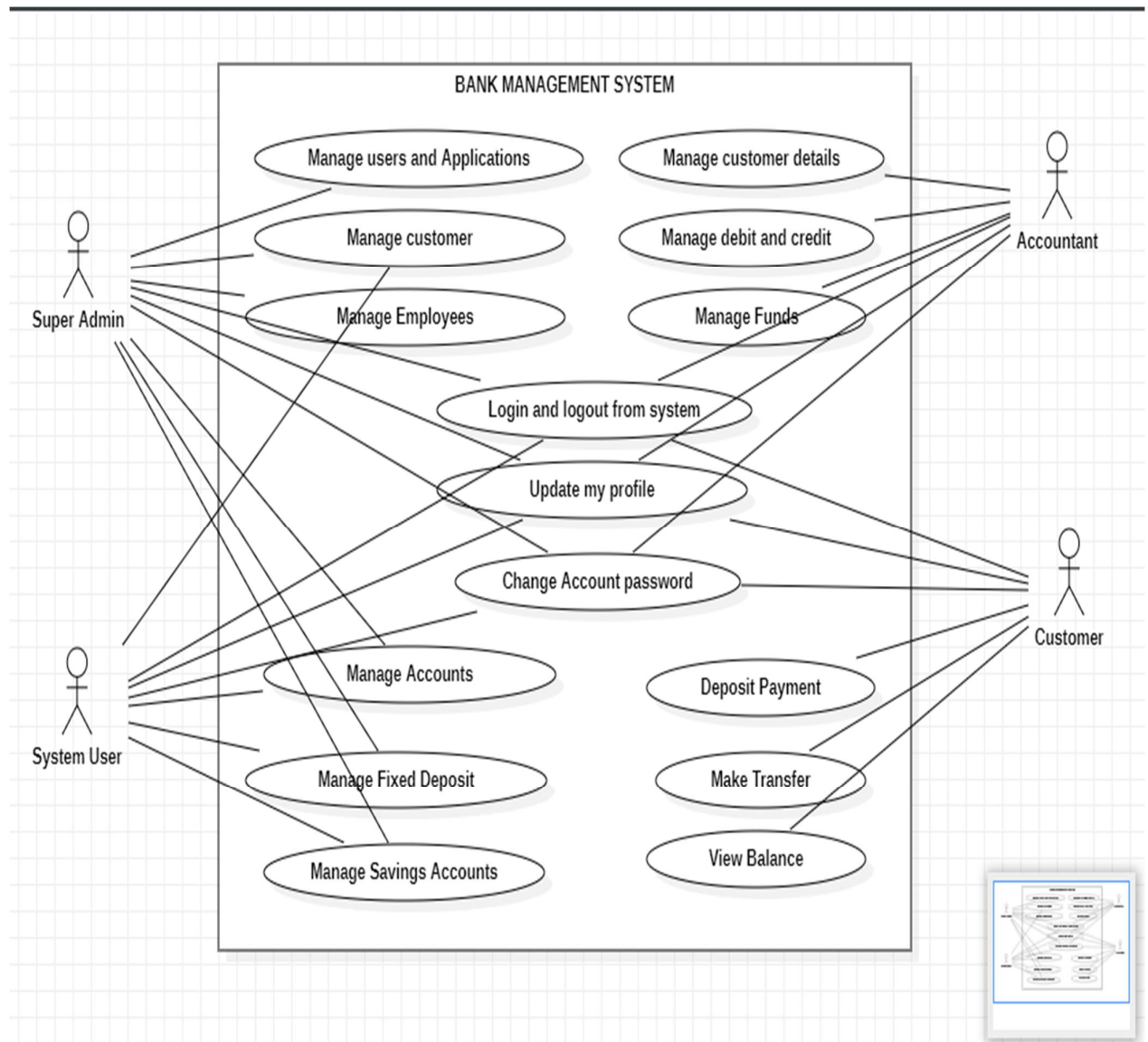


## MODULE DESCRIPTION

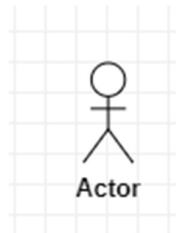
These are the main modules of the project:

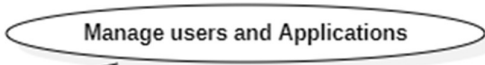
- **Balance Module:** We can create, read, update and delete balance from the module
- **Accounts Module:** All the operations related to Accounts, is managed by this module
- **Savings Account Module:** Saving Account Module is used to manage the Savings Account
- **Current Account Module:** It has been developed for managing the current Account
- **Customer Module:** It manages the customer
- **Employees Module:** Employees operations will be managed by Employees module

# USE CASE DIAGRAM



The above use case diagram is a graphic depiction of the interaction among the elements of the Bank Management System. It represents the methodology used in the system analysis to identify, clarify and organize system requirements of the Bank Management System. The components used in the use case diagram are-

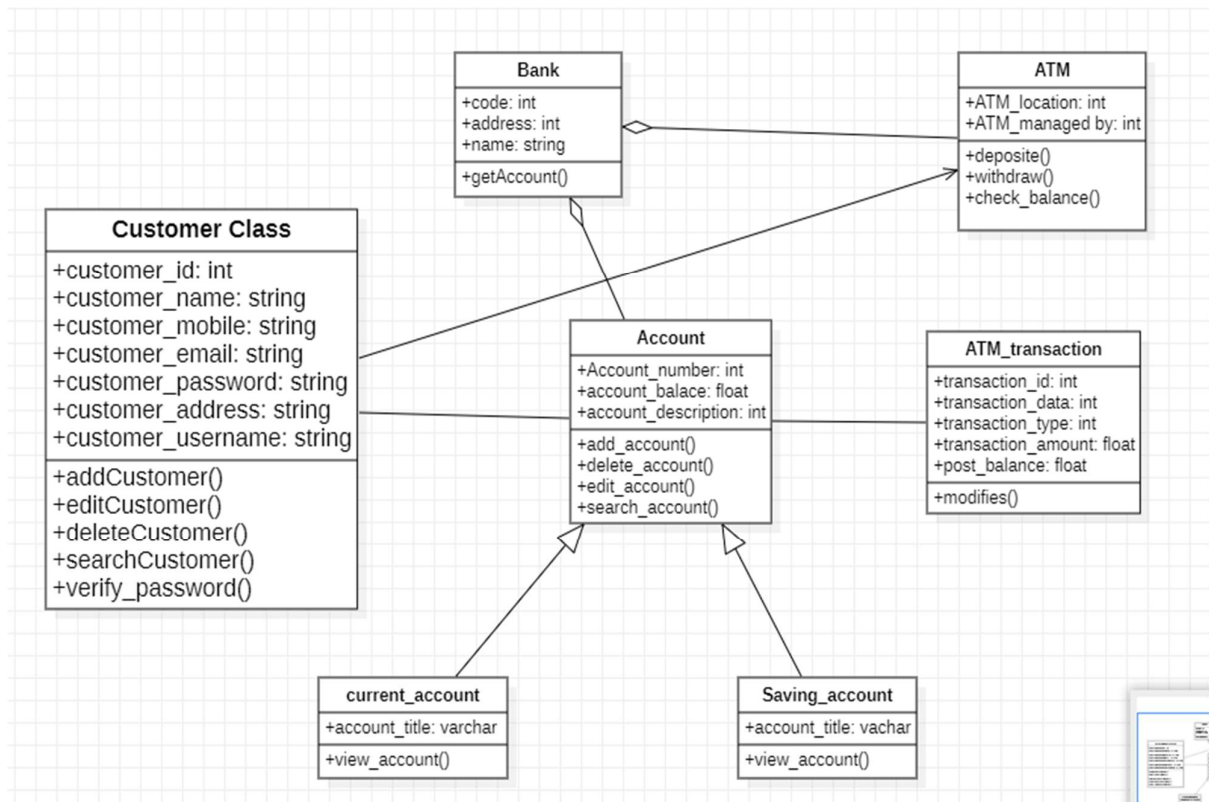


- Actors: - are the entities that interface with the system, the actors of these project was super admin, system user, accountant, Customer
- The rectangular box is known as use case subject where use case is drawn inside of it.
- Use Case:  - are represented in the form oval shape it is used to represent what are the applications can perform by any actor.
- Solid arrow representation is know as association and it is known as association between the actor and the use case.

The relationship among actors and use case:

- ✓ Super admin can perform all the tasks related to managing users and full applications, Manage customers, Manage employees, Login and logout from system, update profiles, change account password, manage accounts, manage fixed deposit, manage saving Accounts
- ✓ System user performs all the tasks similar to super admin except manage employees and manage users.
- ✓ Accountant can perform Manage customer deposits, manage debit and credit, manage funds
- ✓ Customers can perform deposit payment, make transfer, view balance.

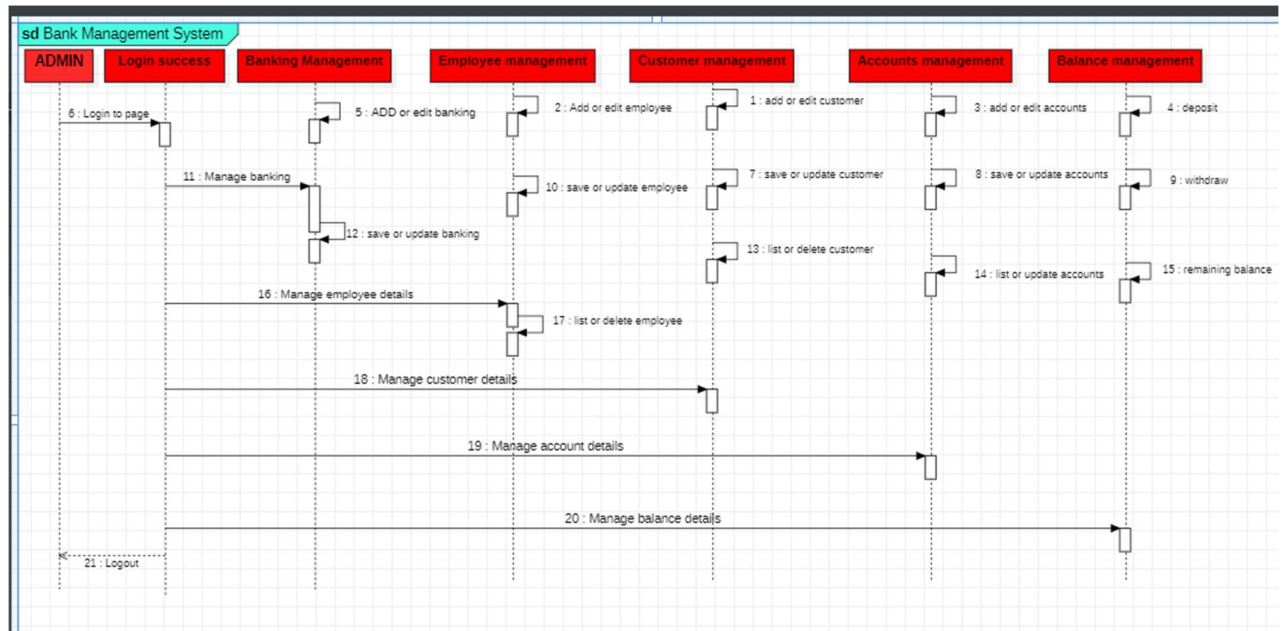
# CLASS DIAGRAM



Bank Management system class diagram describes the structure of a Bank Management system Class, their attributes, operations (or methods) and the relationship among objects.

- The main classes of the bank management system are Customer\_class, current\_account, savings\_account, Account, ATM\_transaction, ATM, Bank.
- ✓ Each rectangular box is used to represent the single class and it is divided into 3 portions-1<sup>st</sup> portion is used to write the class name, 2<sup>nd</sup> portion is used for writing attributes of the respective class and 3<sup>rd</sup> portion for representing operations of the respective classes.
- ✓ One class is interconnected with another class with different types of arrows, some of such arrows which are used aggregation, Generalization, Association and direct Association.

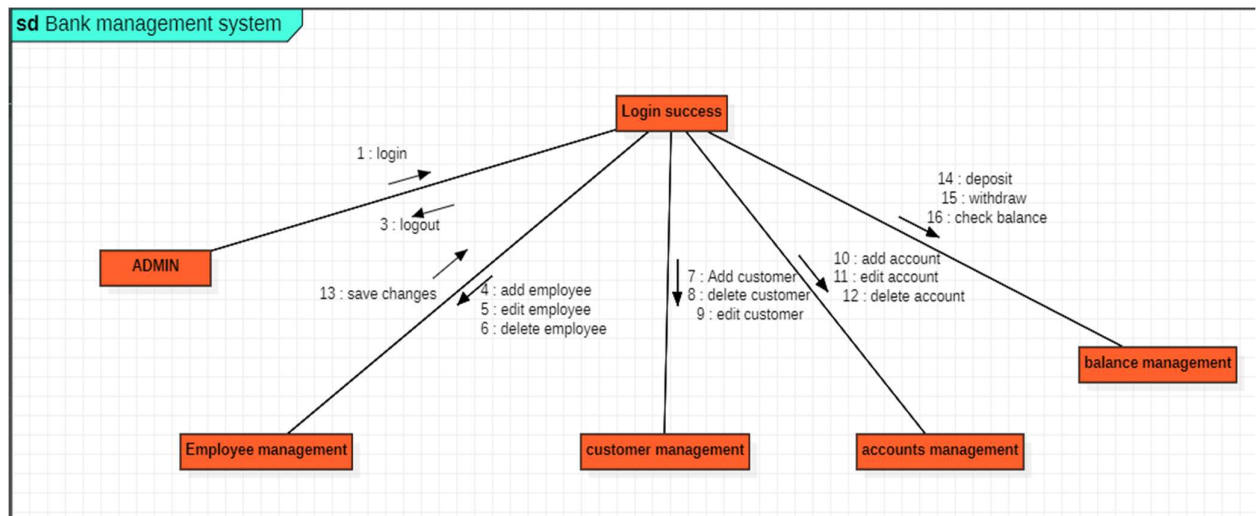
# SEQUENCE DIAGRAM



Sequence diagram is emphasizes on time sequence of message from one object to another object.

- A “lifeline” is a named elements which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. It is represented in the form of dotted lines in vertical direction.
- The objects of our project are Admin, login success, Banking management, Employees management, customer management, Accounts management, Balance Management. Objects are represented in a rectangular box.
- In between objects messages are used, communication between objects are depicted using messages. The messages appear in a sequential order on the life time. We represent messages using arrows
- Different types of messages used in the above sequence diagram are Synchronous messages, Asynchronous messages, self messages, Reply messages.

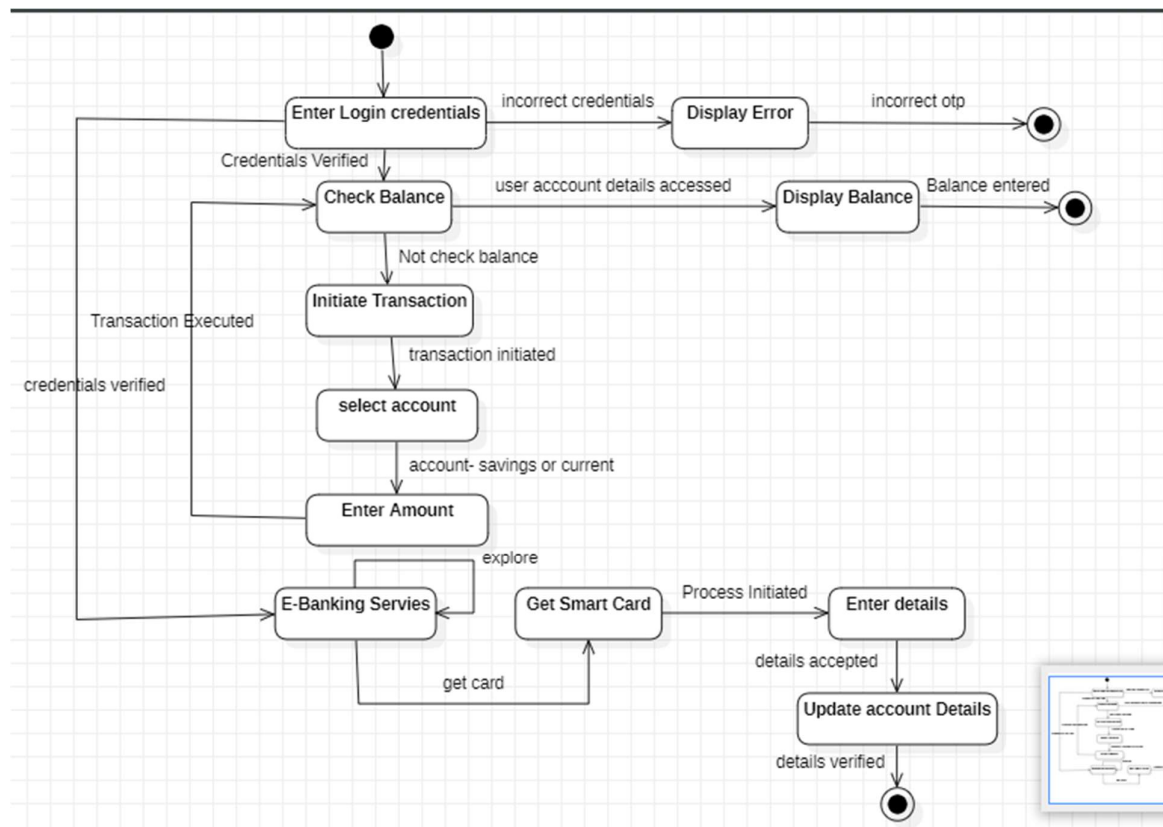
# Collaboration diagram




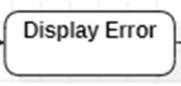

**Collaboration diagram** depicts the relationships and interactions among software objects. They are used to interact among the software objects. They are used to understand the object architecture within a system rather than the flow of a message as in a sequence diagram. They are also known as communication diagrams.

- Its syntax is similar to that of sequence diagram except that life time don't have tails.
- Messages passed over sequencing is indicated by numbering each message hierarchically.
- The objects are login success, admin, employee management, customer management, accounts management, balance management
- First admin will enter password if login is success it will enter into the site other wise it pop up login is failed.
- Now admin can perform all the tasks which ever is need for example adding a customer or removing the user depositing the money
- After performing all the tasks admin will logout from the banking system.

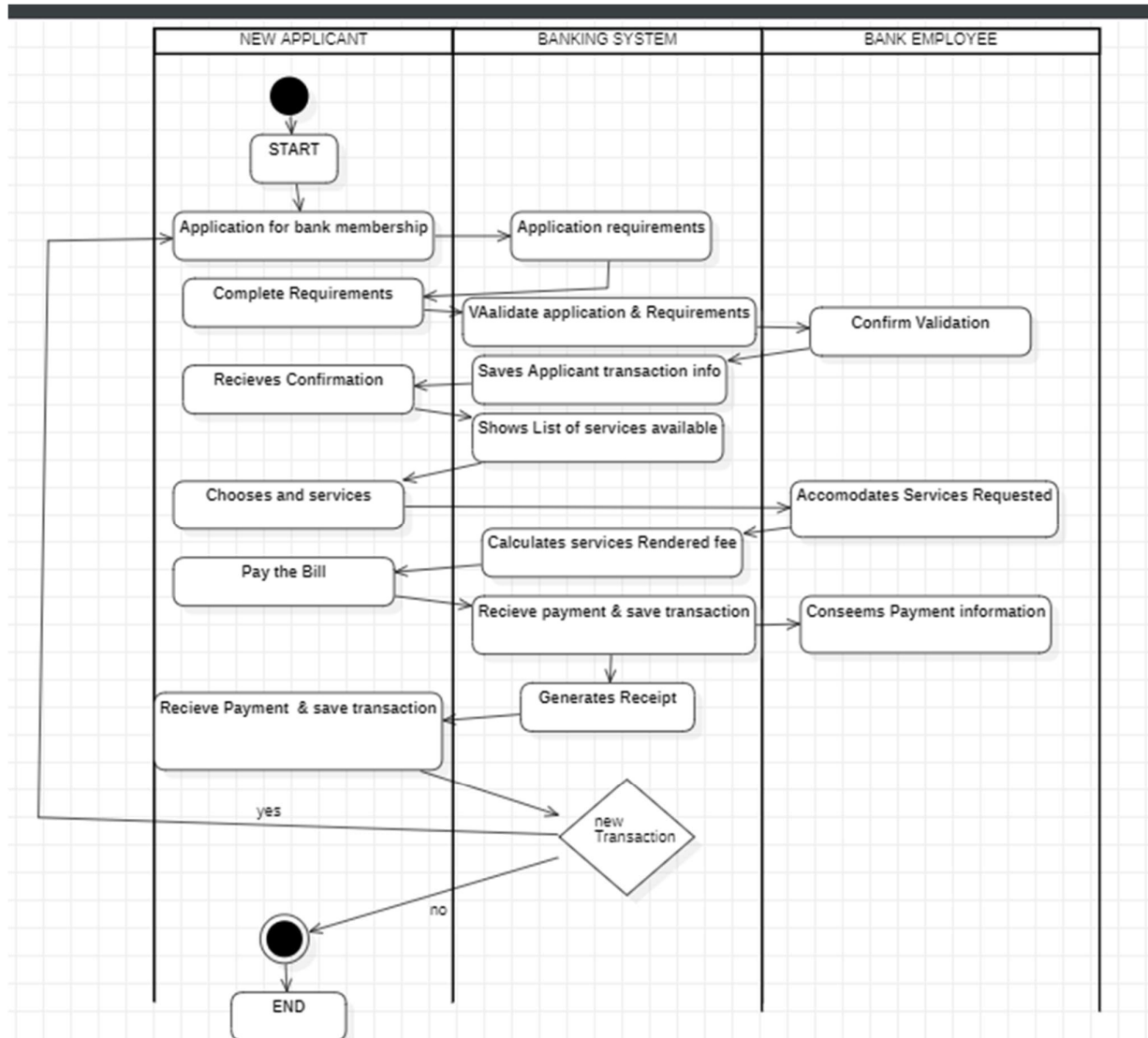
# State Chart Diagram



State chart diagram are also called state machine diagrams. It is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represent the behavior using finite state transitions. It describes the flow of control from one state to another state

- First state is “initial state”  it is the starting point of the state chart diagram that it is where activity starts
- State:  represents a condition of a modelled entity for which some action is performed. The state is indicated by using a rectangle with rounded corners and contains compartments.
- We use solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.
- Final state: the end of the state chart diagram is represented by a solid circle surrounded with circle 

# Activity Diagram





Activity diagram is UML behavior diagram which emphasis on the sequence and condition of the flow. It shows a sequence of actions or flow of control in a system. It is like to a flowchart or a flow diagram. It is frequently Acrused in business process modelling. They can also describe the steps in a use case diagram. The model activities are either sequential or concurrent.

- Activity: it is used to illustrate a set of actions. It shows the non-interruptible actions of object



- Action flow: it is also called edges and paths. It is shows switching from one action state to another. It represented as an arrowed line.
- Decisions and branching: A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, adda diamond between the two activities.



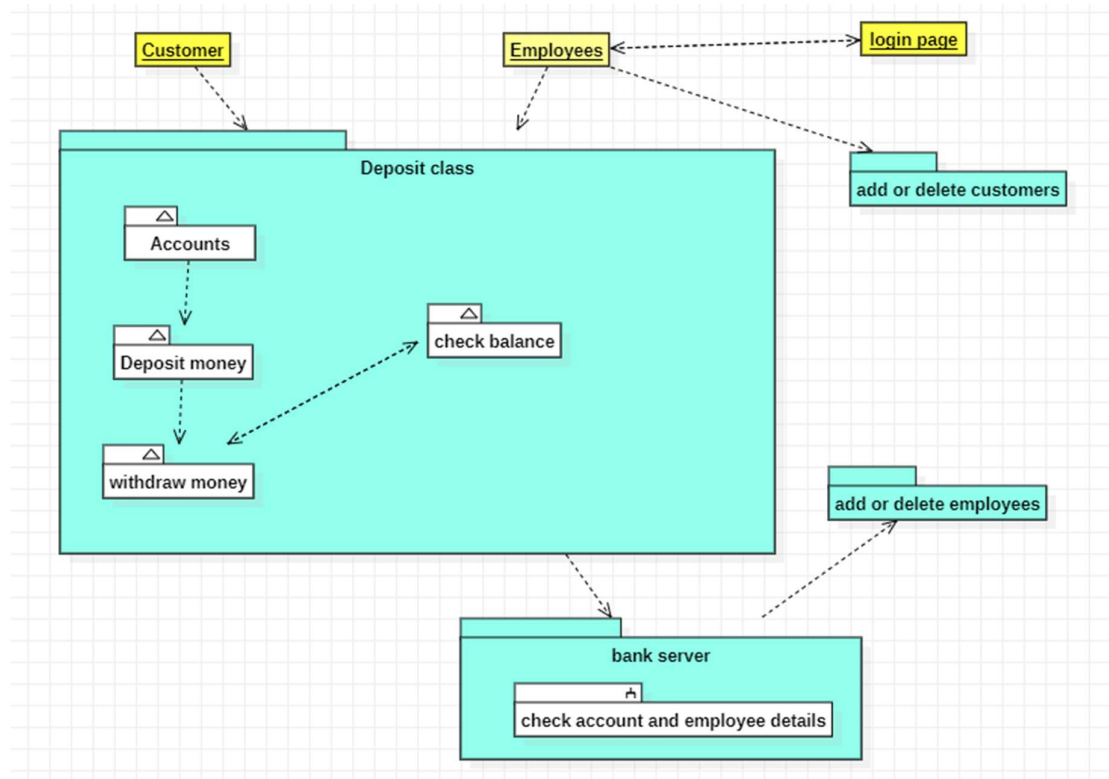
- Swimlane and partition: a way to group activities performed by the same actor on an activity diagram to group activities in a single thread.
- Starting of the activity with a dot



- Ending of the activity with a dot and circle surrounded around it.



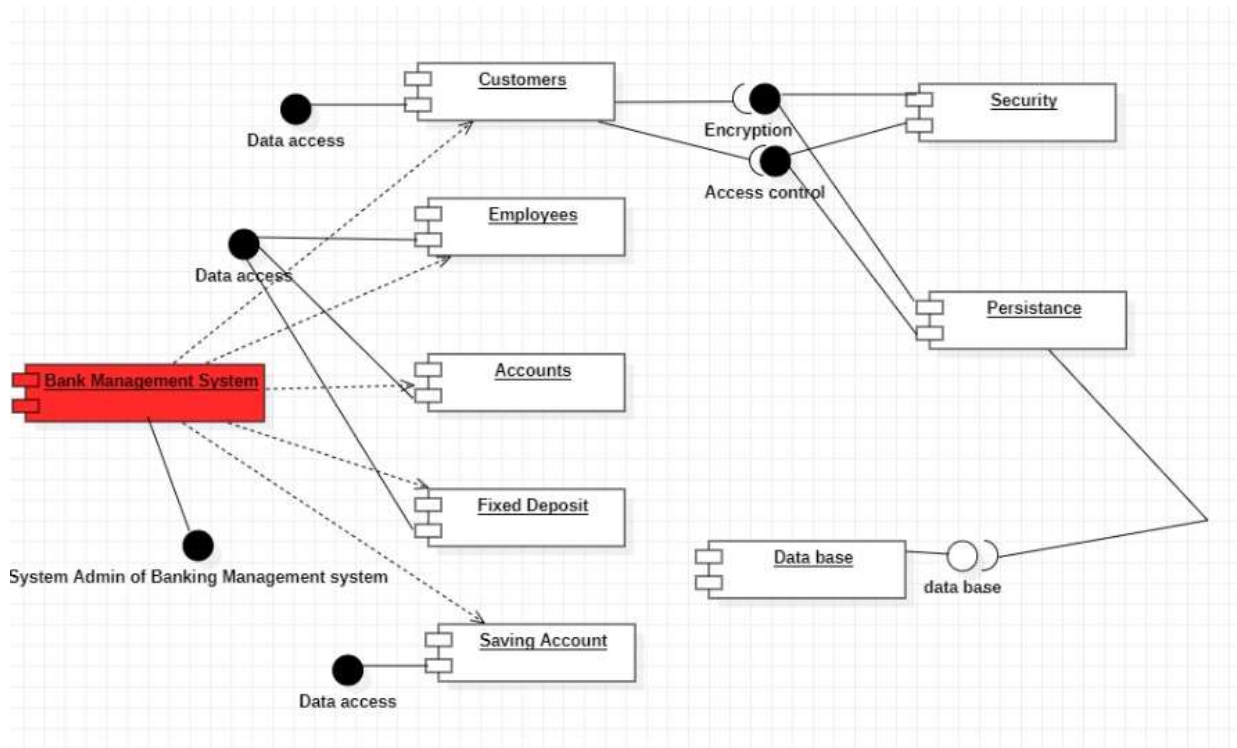
# Package Diagram



All the interrelated classes and interfaces of the system when grouped together form a package. To represent all these interrelated classes and interface UML provides package diagram. **Package diagram** helps in representing the various packages of a software system and dependencies between them. It is used to illustrate the layered architecture of a software system.

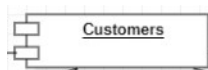
- ✓ Package appear as rectangle with small tabs at the top.
- ✓ The package name is on the tab or inside the rectangle.
- ✓ The dotted arrows are dependencies.
- ✓ One package depends on another if changes in the other could possibly forces changes in the first.
- ✓ Model show only a subset of the contained elements according to some criterion.

# Component Diagram



A component diagram shows the physical view of the system. We combine packages or individual entities to form components. We can depict various components and their dependencies using a component diagram.

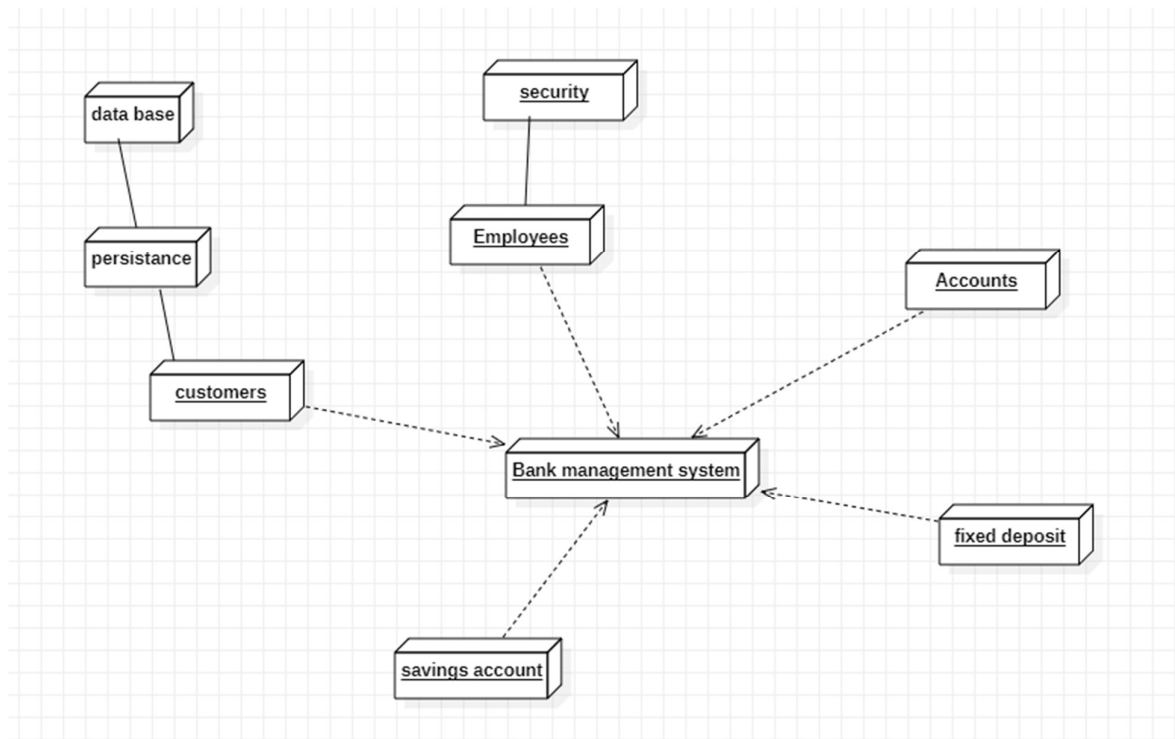
- Component: component is used to represent any part of a system for UML diagrams are made.



Various components of our project are Bank management system, customers, Employees, Accounts, Fixed deposit, Savings account, Data base, security, persistence.

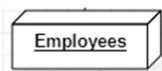
- Association: a structural relationship describing a set of links connected between objects. it is represented with dotted arrow.

# Deployment Diagram



A deployment diagram shows the physical placement of components in nodes over a network. A deployment can be drawn by identifying nodes and components. A deployment diagram usually describes the resources required for processing and the installation of software components in those resources.

- Node: a node represents a physical component of the system. Node is used to represent physical part of a system like server, network etc..



Various nodes of our project are Bank management system, customers, Employees, Accounts, Fixed deposit, Savings account, Data base, security, persistence.

- Association: A structural relationship describing a set of links connected between objects, it is represented with dotted arrow.

## Conclusion

Hence The Above are the representation of the Bank Management system using UML Diagrams. This gives a clear idea about the working of the Bank management system in all the processes and the things that can be done by the system. And the above there is the sample code that runs the bus reservation system which is represented by the UML diagrams.

## Reference

- [T4tutorials.com](http://T4tutorials.com)
- [Geeksforgeeks.org](http://Geeksforgeeks.org)
- Budd, T.(1997b), an introduction in object oriented programming, 2<sup>nd</sup> edn, Addison-wesely.