G.TEJAPRATHAPVARMA CH.SC.U4CSE24117

OBJECT ORIENTED PROGRAMMING (23CSE111)

LAB RECORD

AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF COMPUTING, CHENNAI

BONAFIDE CERTIFICATE

This is to certify that the Lab Record work for 23CSE111- Object Oriented Programming Subject submitted by ***CH.SC.U4CSE24117 – G .TEJAPRATHAPVARMA*** in **"ComputerScience and Engineering"** is a Bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Chennai.

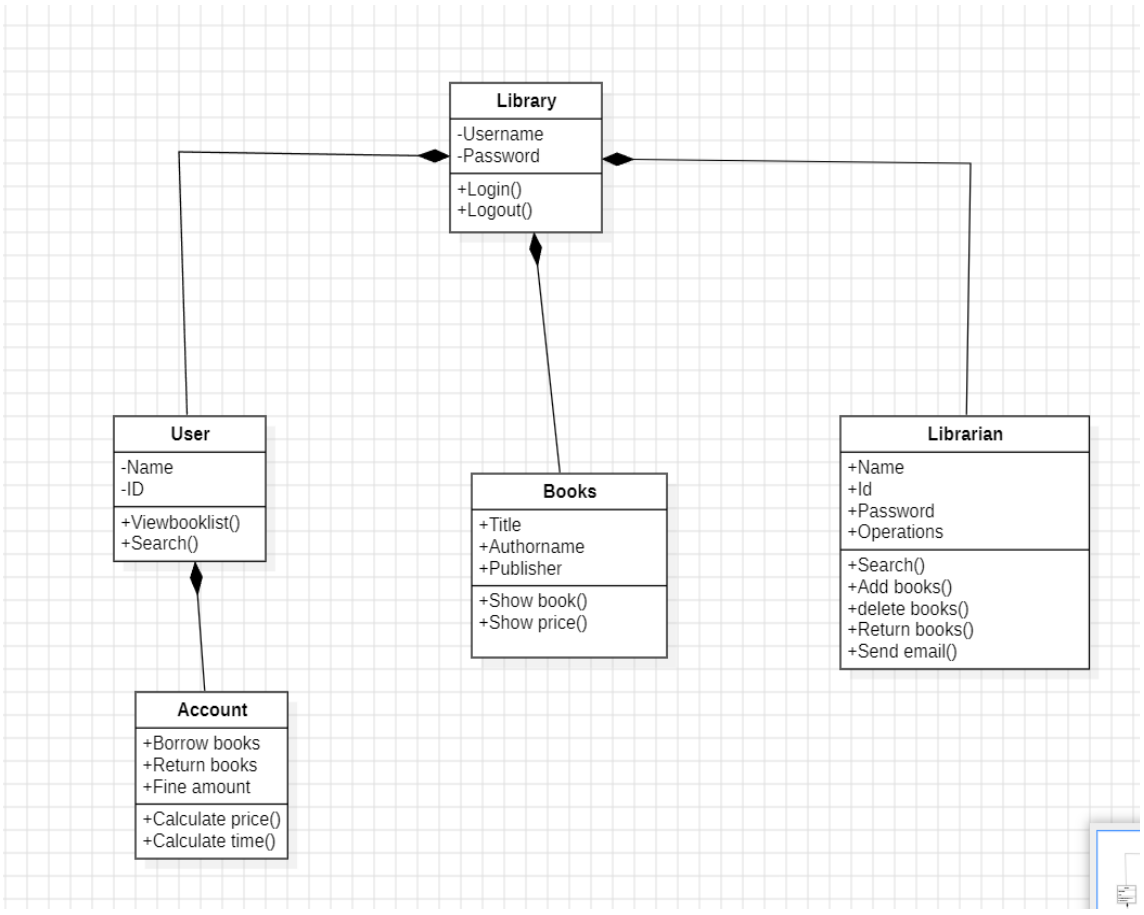
This Lab examination held on

Internal Examiner 1 Internal Examiner 2

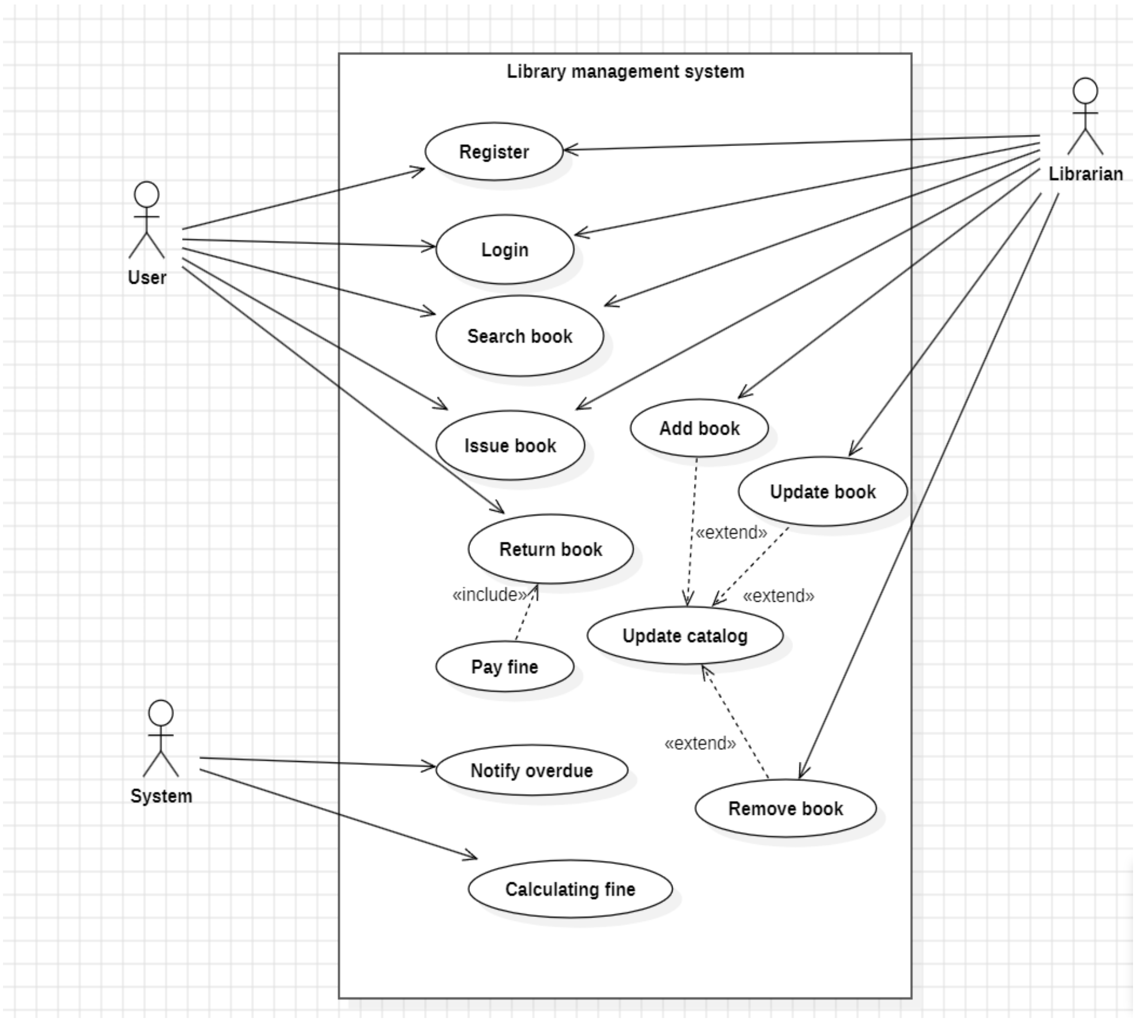
INDEX

1.LIBRARY MANAGEMENT SYSTEM

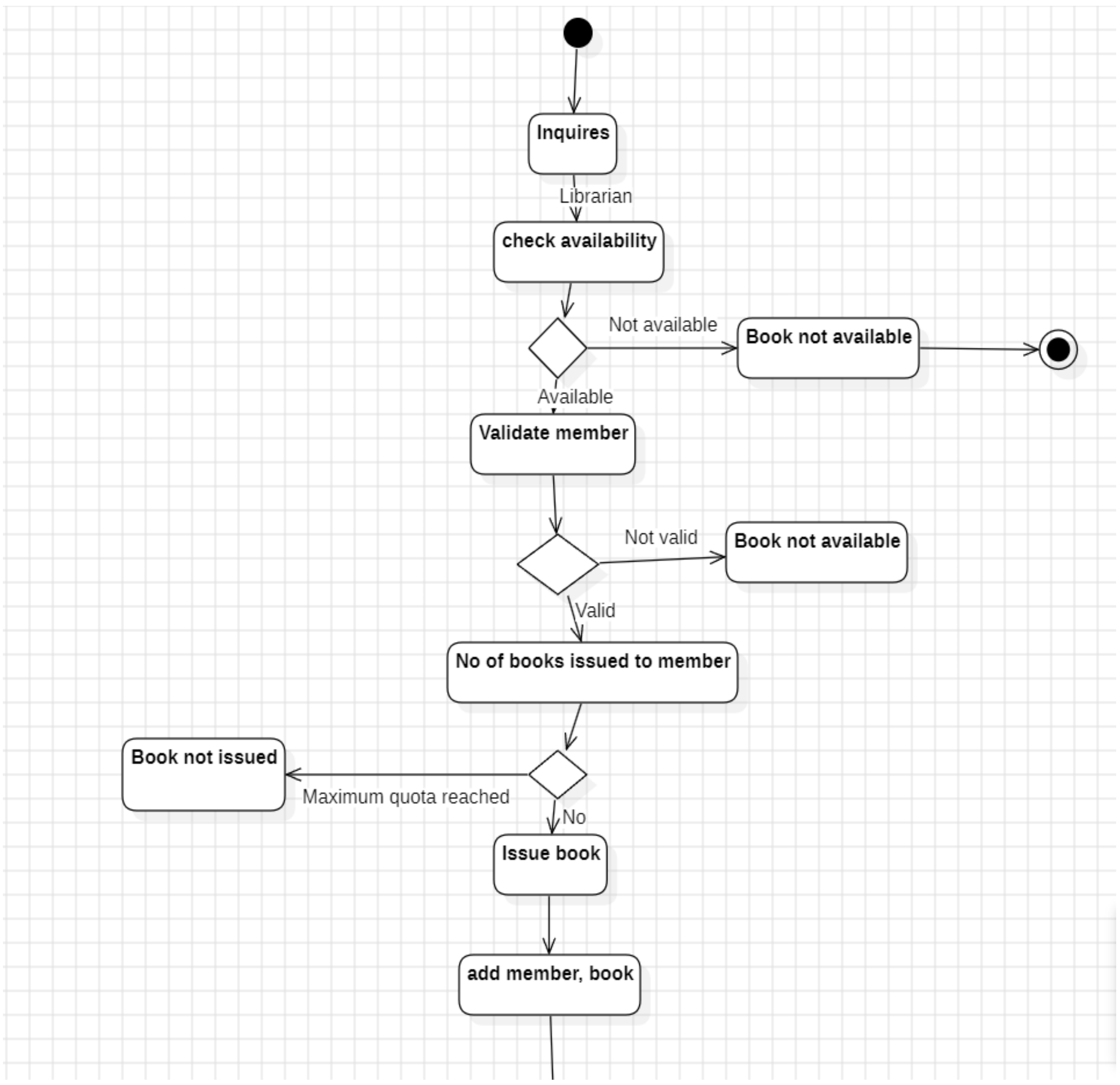
1.a) CLASS Diagram:



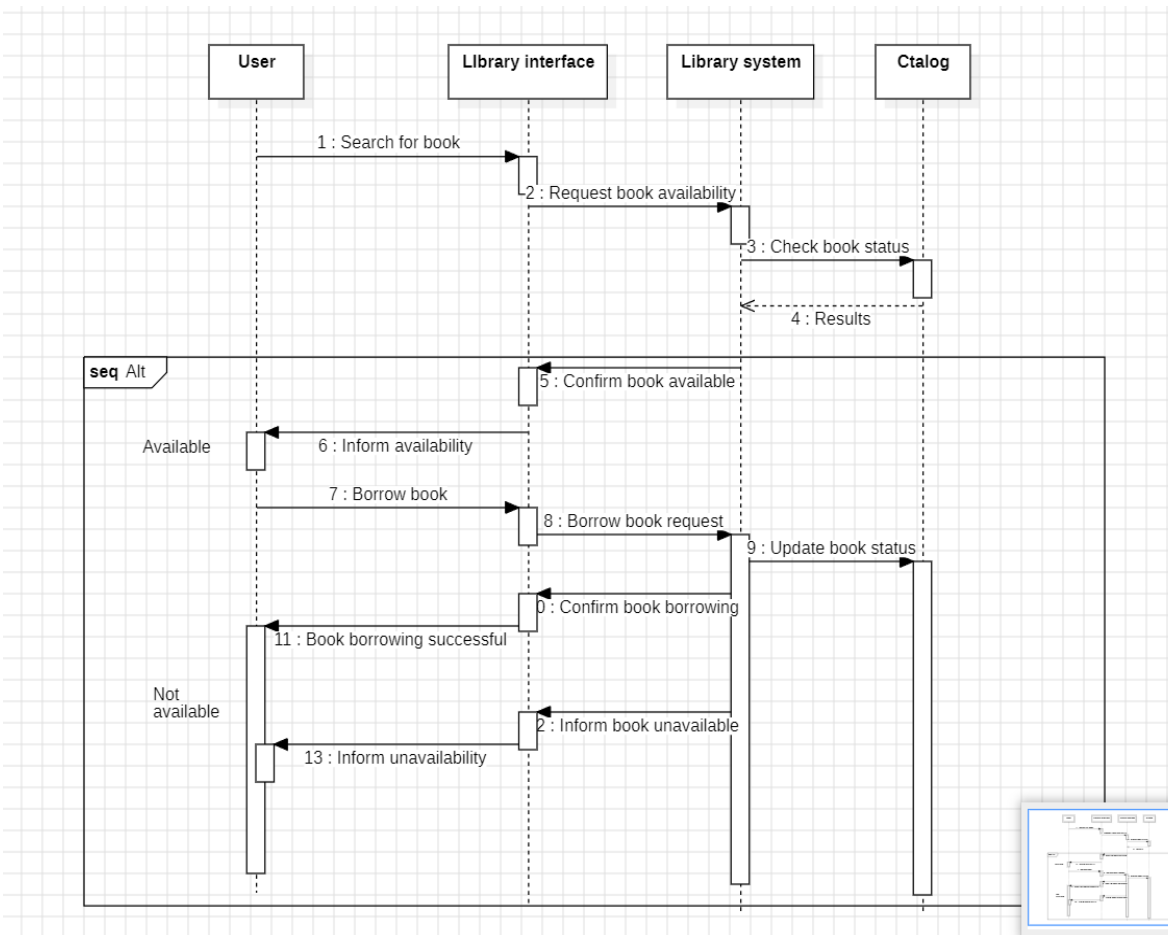
1b) USE CASE DIAGRAM



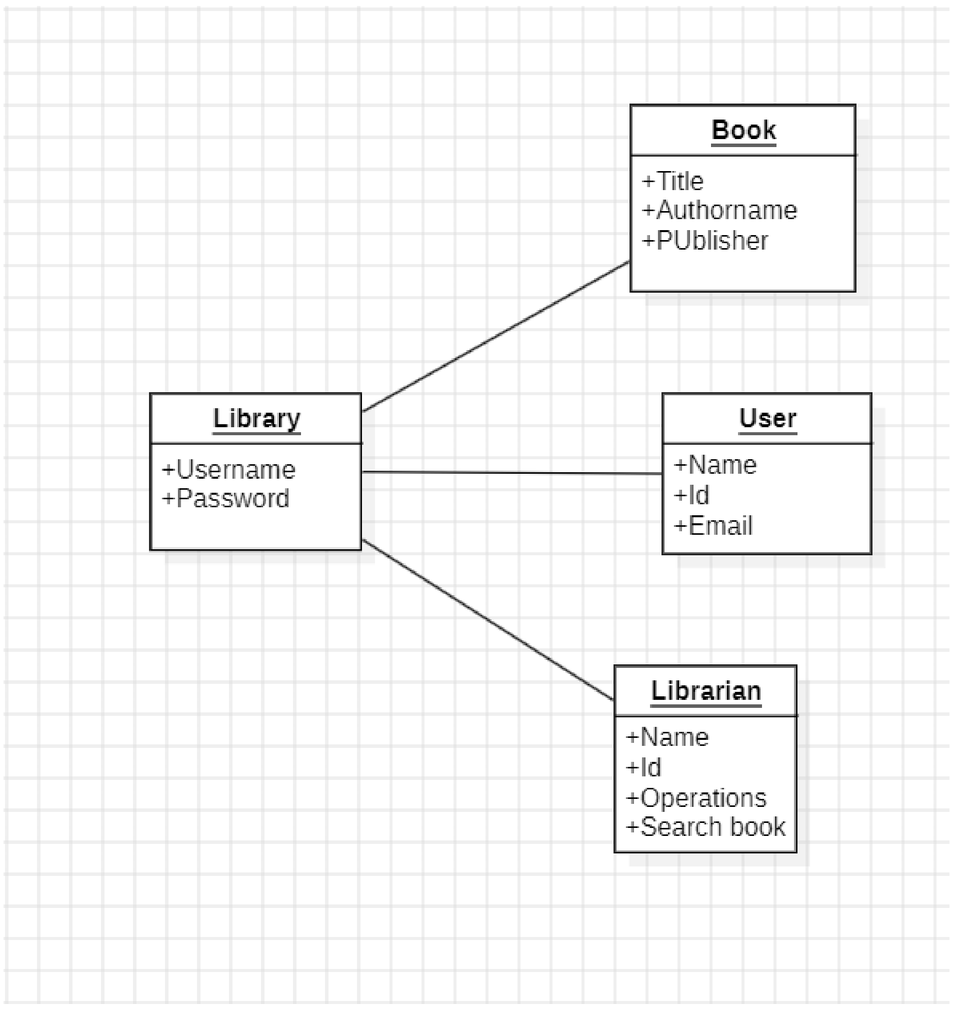
1c) state diagram



1d) sequence diagram

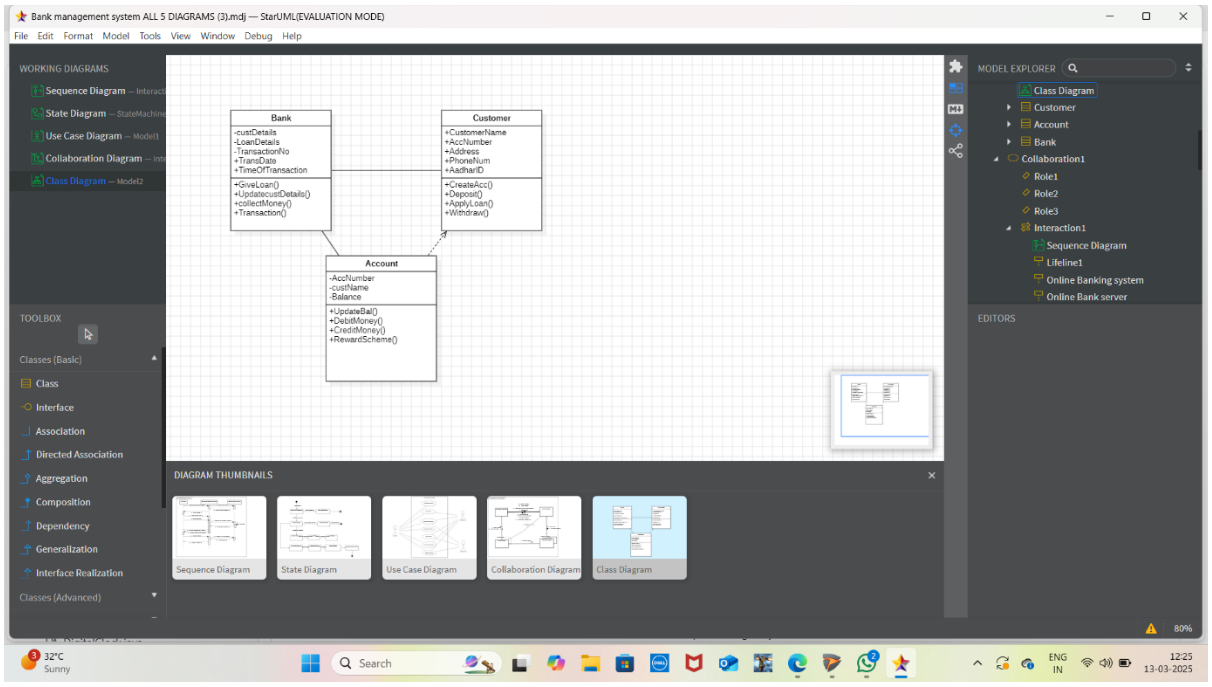


1e) object diagram

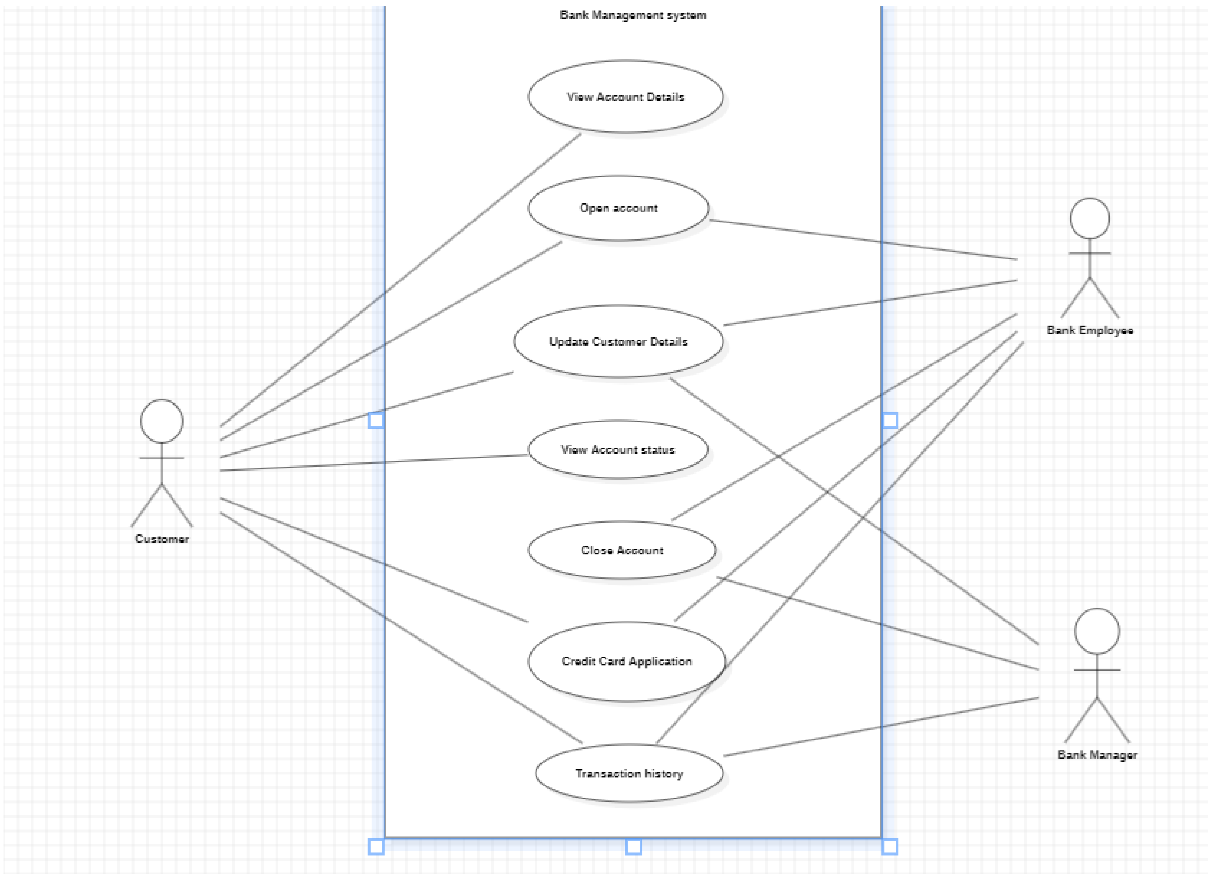

Book
+Title
+Authorname
+PUblisher
Library
+Username
+Password
User
+Name
+Id
+Email
Librarian
+Name
+Id
+Operations
+Search book

Bank management system

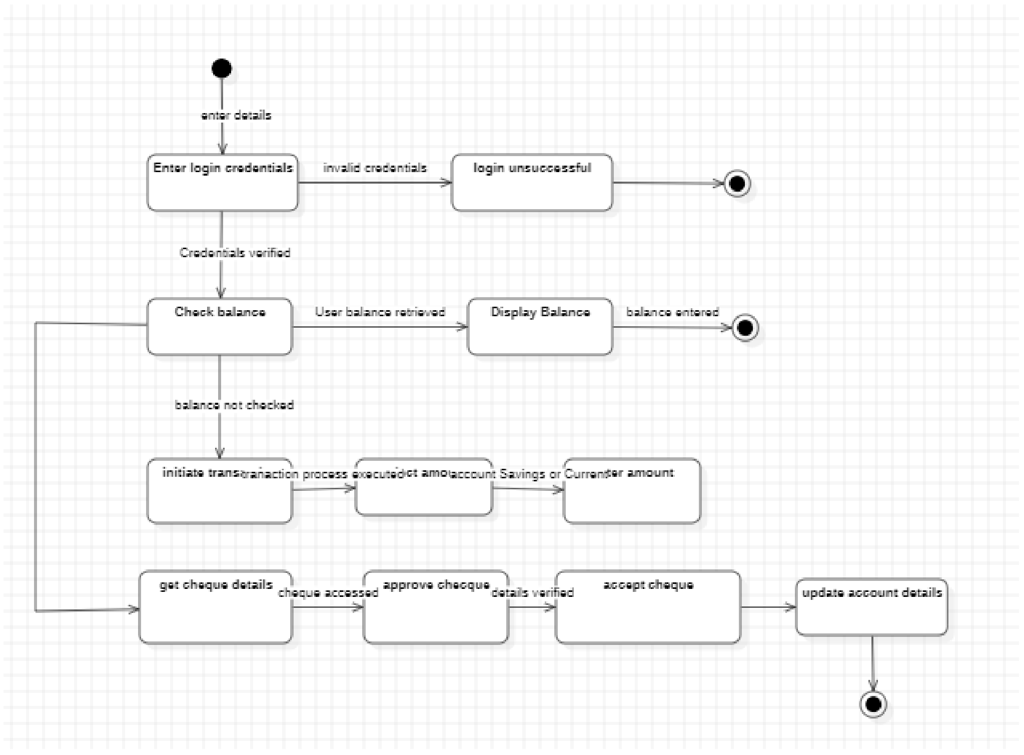
2a) class diagram



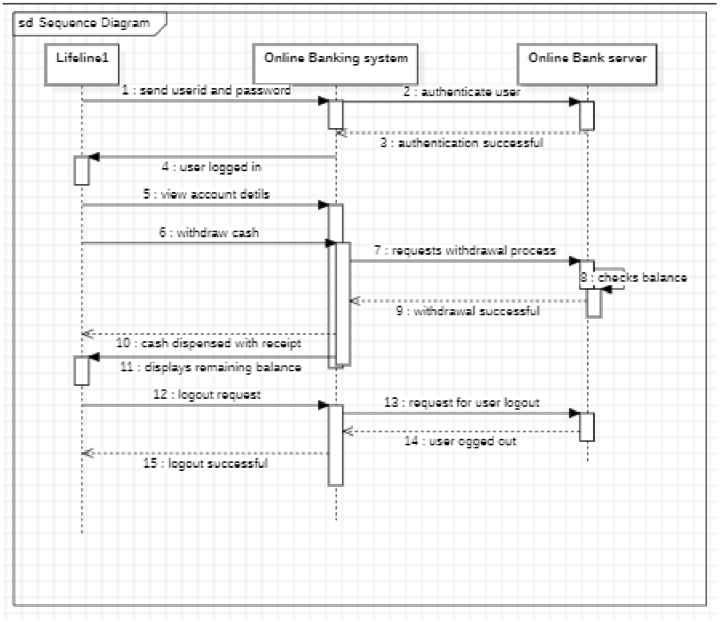
2b) use case diagram



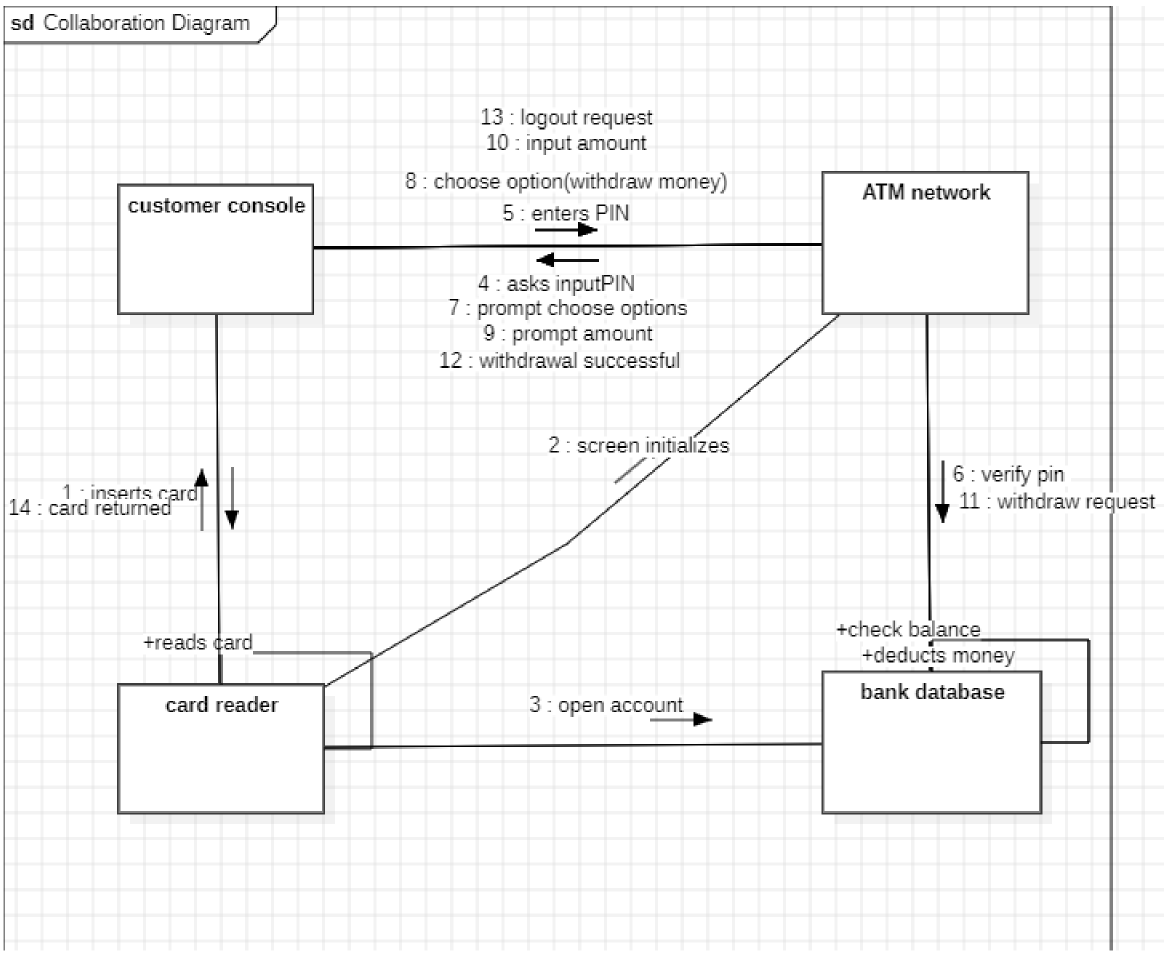
2c) state diagram



2d) SEQUENCE DIAGRAM



2e) COLLABRATION DIAGRAM


sd Collaboration Diagram
customer console
ATM network
card reader
bank database
13 : logout request
10 : input amount
8 : choose option(withdraw money)
5 : enters PIN
4 : asks inputPIN
7 : prompt choose options
9 : prompt amount
12 : withdrawal successful
2 : screen initializes
1 : inserts card
14 : card returned
6 : verify pin
11 : withdraw request
+reads card
+check balance
+deducts money
3 : open account

BASIC JAVA

3A) chat bot

```
import java.util.Scanner;

public class Chatbot {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        while (true) { // Outer loop for conversation
            System.out.print("You: ");
            String input = sc.nextLine();
            if (input.equalsIgnoreCase("exit")) {
                System.out.println("Chatbot: Bye! Have a great day!");
                break;
            }

            for (int i = 1; i <= 1; i++) { // Inner loop for extra responses
                System.out.println("Chatbot: Hmm, interesting! Tell me more.");
            }
        }
    }
}
```

OUTPUT:

```
You: Hello
Chatbot: Hmm, interesting! Tell me more.
You: How are you?
Chatbot: Hmm, interesting! Tell me more.
You: exit
Chatbot: Bye! Have a great day!
```

3B) ContactManager

```
import java.util.HashMap;
import java.util.Scanner;

public class ContactManager {
    public static void main(String[] args) {
        HashMap<String, String> contacts = new HashMap<>();
        Scanner sc = new Scanner(System.in);

        while (true) { // Outer loop: Handles the menu and user choices
            System.out.println("1. Add Contact  2. View Contacts  3. Exit");
            System.out.print("Choose an option: ");
            int choice = sc.nextInt();
            sc.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    while (true) { // Inner loop: Allows multiple entries at once
                        System.out.print("Enter name: ");
                        String name = sc.nextLine();
                        System.out.print("Enter phone number: ");
                        String phone = sc.nextLine();
                        contacts.put(name, phone);

                        System.out.print("Do you want to add another contact? (yes/no): ");
                        String response = sc.nextLine();
                        if (!response.equalsIgnoreCase("yes")) break; // Exit inner loop
                    }
                    break;
                case 2:
                    System.out.println("Contacts:");
                    if (contacts.isEmpty()) {
                        System.out.println("No contacts available.");
                    } else {
                        for (String key : contacts.keySet()) { // Iterating through contacts using
a loop
                            System.out.println(key + ": " + contacts.get(key));
                        }
                    }
                    break;
                case 3:
                    System.out.println("Exiting...");
                    return; // Exit the program
                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        }
    }
}
```

OUTPUT:

```
1. Add Contact  2. View Contacts  3. Exit
Choose an option: 1
Enter name: Alice
Enter phone number: 1234567890
Do you want to add another contact? (yes/no): yes
Enter name: Bob
Enter phone number: 9876543210
Do you want to add another contact? (yes/no): no
```

```
1. Add Contact  2. View Contacts  3. Exit
Choose an option: 2
Contacts:
Alice: 1234567890
Bob: 9876543210

1. Add Contact  2. View Contacts  3. Exit
Choose an option: 3
Exiting...
```

3C) DigitalClock

```java
import java.text.SimpleDateFormat;
import java.util.Date;

public class DigitalClock {
    public static void main(String[] args) {
        int hoursToDisplay = 1; // Display clock for 1 hour
        for (int i = 0; i < hoursToDisplay * 3600; i++) { // Outer loop for 1 hour
            Date date = new Date();
            SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");
            System.out.print("\r" + formatter.format(date));

            // Inner loop for 1-second intervals
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
}
```

OUTPUT:

```
14:30:01
14:30:02
14:30:03
...
```

3D) ExpenseTracker

```
import java.util.Random;

public class MazeGenerator {
    public static void main(String[] args) {
        int rows = 10, cols = 10;
        char[][] maze = new char[rows][cols];
        Random random = new Random();

        for (int i = 0; i < rows; i++) { // Outer loop for rows
            for (int j = 0; j < cols; j++) { // Inner loop for columns
                maze[i][j] = random.nextBoolean() ? '#' : ' ';
            }
        }

        maze[0][0] = 'S'; // Start
        maze[rows - 1][cols - 1] = 'E'; // End

        for (char[] row : maze) { // Loop to print the maze
            for (char cell : row) {
                System.out.print(cell);
            }
            System.out.println();
        }
    }
}
```

OUTPUT:

```
S  # #   #
# #  ## #
 #  #  # #
###  ##  #
 #   ##  #
  ##  ##
#   #  #
##  ##   #
 #   ###
  #  #  E
```

3E) MazeGenerator

```
import java.util.Random;

public class MazeGenerator {
    public static void main(String[] args) {
        int rows = 10, cols = 10;
        char[][] maze = new char[rows][cols];
        Random random = new Random();

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                maze[i][j] = random.nextBoolean() ? '#' : ' ';
            }
        }

        maze[0][0] = 'S'; // Start
        maze[rows - 1][cols - 1] = 'E'; // End

        for (char[] row : maze) {
            for (char cell : row) {
                System.out.print(cell);
            }
            System.out.println();
        }
    }
```

OUTPUT:

```
S  # #   #
# #  ## #
 #  #  # #
###  ##  #
 #   ##  #
  ##  ##
#   #  #
##  ##   #
 #   ###
  #  #  E
```

3F) PasswordGenerator

```
import java.util.Random;
import java.util.Scanner;

public class PasswordGenerator {
    public static void main(String[] args) {
        String chars =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@
#$%^&*()";
        Scanner sc = new Scanner(System.in);

        System.out.print("How many passwords do you want to generate? ");
        int numPasswords = sc.nextInt();

        for (int p = 0; p < numPasswords; p++) { // Outer loop for multiple passwords
            StringBuilder password = new StringBuilder();
            Random random = new Random();
            int length = 12; // Desired password length

            for (int i = 0; i < length; i++) { // Inner loop for characters in each password
                int index = random.nextInt(chars.length());
                password.append(chars.charAt(index));
            }
            System.out.println("Generated Password " + (p + 1) + ": " + password);
        }
    }
}
```

OUTPUT:

```
How many passwords do you want to generate? 3
Generated Password 1: G7#kd8@Pq9!X
Generated Password 2: Xy&8Lp$23AoM
Generated Password 3: B!m3^KqL7@9Z
```

3G) SimpleEncryption

```
import java.util.Scanner;

public class SimpleEncryption {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter text to encrypt: ");
        String text = sc.nextLine();
        System.out.print("Enter shift value: ");
        int shift = sc.nextInt();
        StringBuilder encrypted = new StringBuilder();

        for (char c : text.toCharArray()) {
            if (Character.isLetter(c)) {
                char base = Character.isLowerCase(c) ? 'a' : 'A';
                c = (char) ((c - base + shift) % 26 + base);
            }
            encrypted.append(c);
        }

        System.out.println("Encrypted Text: " + encrypted);
    }
}
```

OUTPUT:

```
Enter text to encrypt: Hello World
Enter shift value: 3
Encrypted Text: Khoor Zruog
```

3H) Stopwatch

```
import java.util.Scanner;

public class Stopwatch {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Press Enter to start the stopwatch.");
        sc.nextLine();
        long startTime = System.currentTimeMillis();

        System.out.println("Press Enter to stop the stopwatch.");
        sc.nextLine();
        long endTime = System.currentTimeMillis();

        System.out.println("Elapsed Time: " + (endTime - startTime) / 1000.0 + "
seconds.");
    }
}
```

OUTPUT:

```
Press Enter to start the stopwatch.
(You press Enter)
Press Enter to stop the stopwatch.
(You press Enter after some time)
Elapsed Time: 5.32 seconds.
```

3l) TicTacToe

```
import java.util.Scanner;

public class TicTacToe {
    static char[][] board = { {'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'} };
    static char currentPlayer = 'X';

    public static void main(String[] args) {
        playGame();
    }
```

```
static void playGame() {
    while (true) {
        printBoard();
        playerMove();
        if (checkWin()) {
            printBoard();
            System.out.println("Player " + currentPlayer + " wins!");
            break;
        }
        if (isBoardFull()) {
            printBoard();
            System.out.println("The game is a tie!");
            break;
        }
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
    }
}

static void printBoard() {
    for (char[] row : board) {
        for (char cell : row) {
            System.out.print(cell + " ");
        }
        System.out.println();
    }
}

static void playerMove() {
    Scanner sc = new Scanner(System.in);
```

```
        System.out.print("Player " + currentPlayer + ", enter your move (1-9): ");
        int move = sc.nextInt();
        int row = (move - 1) / 3;
        int col = (move - 1) % 3;
        if (board[row][col] != 'X' && board[row][col] != 'O') {
            board[row][col] = currentPlayer;
        } else {
            System.out.println("Invalid move! Try again.");
            playerMove();
        }
    }

    static boolean checkWin() {
        for (int i = 0; i < 3; i++) {
            if ((board[i][0] == currentPlayer && board[i][1] == currentPlayer && board[i][2]
== currentPlayer) ||
                (board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i]
== currentPlayer)) {
                return true;
            }
        }
        return (board[0][0] == currentPlayer && board[1][1] == currentPlayer &&
board[2][2] == currentPlayer) ||
               (board[0][2] == currentPlayer && board[1][1] == currentPlayer &&
board[2][0] == currentPlayer);
    }

    static boolean isBoardFull() {
        for (char[] row : board) {
            for (char cell : row) {
                if (cell != 'X' && cell != 'O') {
                    return false;
                }
            }
        }
        return true;
    }
}
```

OUTPUT:

```
1 2 3
4 5 6
7 8 9

Player X, enter your move (1-9): 5
1 2 3
4 X 6
7 8 9

Player O, enter your move (1-9): 1
O 2 3
4 X 6
7 8 9

... (game continues)

Player X wins
```

3J) UnitConverter

```
import java.util.Scanner;
```

```
public class UnitConverter {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        while (true) { // Outer loop for multiple conversions
            System.out.print("Enter distance in kilometers (or type '-1' to exit): ");
            double kilometers = sc.nextDouble();
            if (kilometers == -1) break;

            for (int i = 1; i <= 1; i++) { // Inner loop for repeated confirmation
                double miles = kilometers * 0.621371;
                System.out.println(kilometers + " kilometers is equal to " + miles + "
miles.");
            }
        }
    }
}
```

```
OUTPUT: Enter distance in kilometers (or type '-1' to exit): 5
5.0 kilometers is equal to 3.106855 miles.

Enter distance in kilometers (or type '-1' to exit): 10
10.0 kilometers is equal to 6.21371 miles.

Enter distance in kilometers (or type '-1' to exit): -1
```