Task 1

I. I have selected Queue since the ticketing system is based First-In First-Out principle. A limited number of ticket buyers must wait in the queue until their time comes. and tickets will be issued in the First-In-First-Out strategy.

II. Queue using array

```
public class Queue
{
    private int front;
    private int rear;
    private int noOfItems;
    private int maxSize;
    private int[] queueArray;

    public Queue(int size)
    {
        maxSize = size;
        front = 0;
        rear = -1;
        noOfItems = 0;
        queueArray = new int[maxSize];
    }

    public boolean isFull()
    {
        return (rear == maxSize - 1);
    }

    public boolean isEmpty()
    {
        return (noOfItems == 0);
    }

    public void enqueue(int item)
    {
        if (isFull())
        {
            System.out.println("The queue is full");
        }
        else
        {
            queueArray[++rear] = item;
            noOfItems++;
        }
    }

    public int deque()
    {
        if (isEmpty())
        {
            System.out.println("The Queue is Empty");
            return 0;
        }
        else
        {
            noOfItems--;
            return queueArray[front++];
        }
```

```
    }

}
```

III. In my ticketing system I have created four java classes,

1. TicketingSystem – I have implemented the data structure here using an array
2. Main – The execution is done by using this class
3. Admin – the deque method is controlled by the admin
4. Buyer – buyer needs to add his/her details to purchase a ticket

- TicketingSystem

```
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellType;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Scanner;

public class TicketingSystem
{
    private int front;
    private int rear;
    private int noOfBuyers;
    private int maxNoOfTickets;
    private String[] buyerArray;
    private Scanner sc = new Scanner(System.in);

    public TicketingSystem(int size, int front, int rear, int
noOfBuyers,String[] buyerArray )
    {
        maxNoOfTickets = size;
        this.front = front;
        this.rear = rear;
        this.noOfBuyers = noOfBuyers;
        this.buyerArray = buyerArray;
    }

    public boolean isFull()
    {
        return (rear == maxNoOfTickets - 1);
    }

    public boolean isEmpty()
    {
        return (noOfBuyers == 0);
    }

    public void enqueue(XSSFSheet sheetOne, String name, XSSFWorkbook
workbook, String filePath) throws IOException {
        buyerArray[++rear] = name;
        noOfBuyers++;
```

```
        XSSFRow row = sheetOne.createRow(rear);
        XSSFCell cell = row.createCell(0);
        cell.setCellValue(name);

        FileOutputStream outStream = new FileOutputStream(filePath);
        workbook.write(outStream);
        outStream.close();

        System.out.println("Your name has added to the ticket queue, your
ticket will be issued with in 24 hours");

    }
    public String dequeue(XSSFSheet sheetOne,XSSFWorkbook workbook, String
filePath) throws IOException {
        noOfBuyers--;
        XSSFRow row = sheetOne.createRow(front);
        XSSFCell cell = row.createCell(0);
        cell.setCellValue("__");

        FileOutputStream outStream = new FileOutputStream(filePath);
        workbook.write(outStream);
        outStream.close();

        return buyerArray[front++];

    }
}
```

- Main

```
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.InputMismatchException;
import java.util.Objects;
import java.util.Scanner;

public class Main
{
    static int day = 0;
    static int option = 0;
    static String excelFilePathOne = ".\\dataFiles\\TicketingSystem.xlsx";
    static Scanner sc = new Scanner(System.in);
    static String[] buyerArray = new String[1000];
    static int front = 0;
    static int rear = -1;
    static int size = 1000;

    static String excelFilePathTwo = ".\\dataFiles\\PurchaseDetails.xlsx";
    static FileInputStream inputStreamOne;
    static int adminOption = 2;
```

```
    static int buyerOption = 2;

    static {
        try {
            inputStreamOne = new FileInputStream(excelFilePathOne);
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    static XSSFWorkbook workbookOne;

    static {
        try {
            workbookOne = new XSSFWorkbook(inputStreamOne);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    static XSSFSheet sheetOne = null;

    static FileInputStream inputStreamTwo;

    static {
        try {
            inputStreamTwo = new FileInputStream(excelFilePathTwo);
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    static XSSFWorkbook workbookTwo;

    static {
        try {
            workbookTwo = new XSSFWorkbook(inputStreamTwo);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    static XSSFSheet sheetTwo = null;
    public static void main(String[] args) throws IOException {

        System.out.println("If you are the admin: press 1\nIf you are a
ticket buyer: press 2");
        while (option!=1 && option!=2) {
            System.out.println("Enter your option: ");
            try {
                option = sc.nextInt();
                if (option != 1 && option != 2) {
                    throw new Exception("Enter an integer according to the
option");
                }
            } catch (InputMismatchException e) {
                System.out.println("Enter an integer according to the
option");
            } catch (Exception e) {
                System.out.println(e);
            } finally {
```

```
                sc.nextLine();
            }
        }



        if (option == 1)
        {
            Admin admin = new Admin();
            admin.enterPassword();

            while (adminOption == 2)
            {
                enterDay();
                referSheet();
                buyerArray = new String[1000];
                int noOfBuyers = loadBuyerNames();
                TicketingSystem tc = new
TicketingSystem(size,front,rear,noOfBuyers,buyerArray);
                performAdminTask(tc);
                adminOption = admin.enterOption();
                while (adminOption == 1)
                {
                    performAdminTask(tc);
                    adminOption = admin.enterOption();
                }
            }

        }
        else
        {
            Buyer buyer = new Buyer();
            System.out.print("Ticket price is Rs: 1500\nIf you want to buy:
press 1\nIf not: press 2\n");
            int buyTicketOption = buyer.selectBuyTicketOption();
            if (buyTicketOption == 1)
            {
                while (buyerOption == 2)
                {
                    enterDay();
                    referSheet();
                    buyerArray = new String[1000];
                    int noOfBuyers = loadBuyerNames();
                    TicketingSystem tc = new
TicketingSystem(size,front,rear,noOfBuyers,buyerArray);
                    performBuyerTask(buyer,tc);
                    buyerOption = buyer.buyerOption();
                    while (buyerOption == 1)
                    {
                        performBuyerTask(buyer,tc);
                        buyerOption = buyer.buyerOption();
                    }
                }

            }
        }
    }
    public static void enterDay()
    {
        day = 0;
```

```
        while (day!=1 && day!=2 && day!=3)
        {
            System.out.println("Enter the day using numbers: ");
            try
            {
                day = sc.nextInt();
                if (day!=1 && day!=2 && day!=3)
                {
                    throw new Exception("Enter an integer according to the
day");
                }
            }
            catch (InputMismatchException e)
            {
                System.out.println("Enter an integer according to the
day");
            }
            catch (Exception e)
            {
                System.out.println(e);
            }
            finally
            {
                sc.nextLine();
            }
        }
    }

    public static void referSheet()
    {
        if (day == 1)
        {
            sheetOne = workbookOne.getSheet("dayOne");
        }
        if (day == 2)
        {
            sheetOne = workbookOne.getSheet("dayTwo");
        }
        if (day == 3)
        {
            sheetOne = workbookOne.getSheet("dayThree");
        }
        sheetTwo = workbookTwo.getSheet("sheetOne");
    }

    public static int loadBuyerNames()
    {
        int rows = size;
        int noOfBuyers = 0;

        for (int r = 0; r<rows; r++)
        {
            XSSFRow row = sheetOne.getRow(r);

            if (row == null)
            {
                buyerArray[r] = null;
            }
            else
            {
                XSSFCell cell = row.getCell(0);
```

```
                if (Objects.equals(cell.getStringCellValue(), "__"))
                {
                    buyerArray[r] = "__";
                }
                else
                {
                    buyerArray[r] = cell.getStringCellValue();
                }
            }

        }

//         for (int i = 0; i<buyerArray.length; i++)
//         {
//             System.out.println(buyerArray[i]);
//         }


        for (int i = 0; i<buyerArray.length; i++)
        {
            if (Objects.equals(buyerArray[i], "__"))
            {
                for (int j = i; j<buyerArray.length; j++)
                {
                    if(buyerArray[j] == null)
                    {
                        front = j-1;
                        rear = j-1;
                        noOfBuyers = 0;
                        break;
                    }
                    else if (buyerArray[j] != null &&
!Objects.equals(buyerArray[j], "__"))
                    {
                        front = j;
                        rear = j;
                        for (int k = rear; k<buyerArray.length; k++)
                        {
                            int temp = k-1;
                            rear = temp;
                            noOfBuyers = (rear - front)+1;
                            if (buyerArray[k] == null)
                                break;
                        }
                        break;
                    }
                }
                break;
            }
            else if (buyerArray[i] != null &&
!Objects.equals(buyerArray[i], "__"))
            {
                front = 0;
                rear = 0;
                for (int l = rear; l<buyerArray.length; l++)
                {
                    int temp = l-1;
                    rear = temp;
                    noOfBuyers = (rear - front)+1;
                    if (buyerArray[l] == null)
                        break;
```

```
                }
                break;
            }
            else if(buyerArray[i] == null)
            {
                front = 0;
                rear = -1;
                noOfBuyers = 0;
                break;
            }
        }

        return noOfBuyers;
    }

    public static int checkCardNumber(String[] cardNumberArray, String
cardNumber)
    {
        for (int i = 0; i<cardNumberArray.length; i++)
        {
            if (Objects.equals(cardNumberArray[i], cardNumber))
            {
                return i;
            }
        }
        return -1;
    }

    public static void enterPurchaseDetails(String buyerName, String
cardNumber, XSSFSheet sheetTwo, XSSFWorkbook workbook, String filePath)
throws IOException {
        int maxRow = sheetTwo.getLastRowNum();
        XSSFRow row = sheetTwo.createRow(maxRow+1);

        Cell cell1 = row.createCell(0);
        cell1.setCellValue(buyerName);

        Cell cell2 = row.createCell(1);
        cell2.setCellValue(cardNumber);

        FileOutputStream outStream = new FileOutputStream(filePath);
        workbook.write(outStream);
        outStream.close();
    }

    public static void performBuyerTask(Buyer buyer,TicketingSystem tc)
throws IOException {

        boolean queueStatus = tc.isFull();
        if(queueStatus)
        {
            if (day==3)
            {
                System.out.println("Sorry, All the tickets were issued");
                return;
            }
            else
            {
                System.out.println("Sorry , All the tickets for the day
were issued, try for another day");
            }
```

```
        }
        else
        {
            String[] accountNumberArray =
buyer.loadAccountNumbers(sheetTwo);
            String buyerName = buyer.enterBuyerName();
            String accountNumber = buyer.enterCardNumber();

            int status = checkCardNumber(accountNumberArray,accountNumber);
            if (status != -1)
            {
                XSSFRow row = sheetTwo.getRow(status);
                XSSFCell cell = row.getCell(0);
                if (!Objects.equals(cell.getStringCellValue(), buyerName))
                {
                    System.out.println("Sorry, You cannot transfer your
ticket for another person");
                }
                else
                {
                    buyer.enterCardExpiryDate();
                    buyer.enterCVCNumber();

enterPurchaseDetails(buyerName,accountNumber,sheetTwo,workbookTwo,excelFile
PathTwo);

tc.enqueue(sheetOne,buyerName,workbookOne,excelFilePathOne);
                }
            }
            else
            {
                buyer.enterCardExpiryDate();
                buyer.enterCVCNumber();

enterPurchaseDetails(buyerName,accountNumber,sheetTwo,workbookTwo,excelFile
PathTwo);

tc.enqueue(sheetOne,buyerName,workbookOne,excelFilePathOne);
            }
        }
    }

    public static void performAdminTask(TicketingSystem tc ) throws
IOException {
        boolean queueStatus = tc.isEmpty();
        if (queueStatus)
        {
            System.out.println("The customer queue is empty");
        }
        else {
            System.out.println("You issued a ticket for " +
tc.dequeue(sheetOne,workbookOne,excelFilePathOne));
        }
    }

}
```

- Admin

```
import org.apache.poi.xssf.usermodel.XSSFSheet;

import java.util.InputMismatchException;
import java.util.Objects;
import java.util.Scanner;

public class Admin
{
    private int adminOption;
    final private String adminPassword = "adminPassword";
    private Scanner sc = new Scanner(System.in);

    Admin()
    {
        adminOption = 0;
    }

    public void enterPassword()
    {
        String password = "";
        while (!Objects.equals(password, adminPassword))
        {
            System.out.print("Enter your password: ");
            try
            {
                password = sc.next();
                if (!Objects.equals(password, adminPassword))
                {
                    throw new Exception("Your password is incorrect, Enter
again");
                }
            }
            catch (Exception e)
            {
                System.out.println(e);
            }
        }
    }

    public int enterOption()
    {
                adminOption = 0;
                System.out.println("If you want to issue a ticket for
another buyer with in same day: press 1\nIf you want to issue a ticket for
another buyer for another day: press2\nIf not: press 3");
                while (adminOption!=1 && adminOption!=2 && adminOption!=3)
                {
                    System.out.print("Enter your option: ");
                    try
                    {
                        adminOption = sc.nextInt();
                        if (adminOption != 1 && adminOption != 2 &&
adminOption!=3) {
                            throw new Exception("Enter an integer according
to the option");
                        }
                    }
                    catch (InputMismatchException e) {
                        System.out.println("Enter an integer according to
the option");
                    } catch (Exception e) {
```

```
                        System.out.println(e);
                    } finally {
                        sc.nextLine();
                    }
                }
                return adminOption;
    }


}
```

- Buyer

```
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Buyer
{
    int buyTicketOption;
    final Scanner sc = new Scanner(System.in);
    int buyerOption;
    String[] accountNumberArray = new String[3000] ;
    public int selectBuyTicketOption()
    {
        while (buyTicketOption!=1 && buyTicketOption!=2)
        {
            System.out.print("Enter your option: ");
            try {
                buyTicketOption = sc.nextInt();
                if (buyTicketOption != 1 && buyTicketOption != 2) {
                    throw new Exception("Enter an integer according to the
option");
                }
            } catch (InputMismatchException e) {
                System.out.println("Enter an integer according to the
option");
            } catch (Exception e) {
                System.out.println(e);
            } finally {
                sc.nextLine();
            }
        }
        return buyTicketOption;
    }

    public int buyerOption()
    {
        buyerOption = 0;
        System.out.println("If you want to buy a ticket again with in same
day: press 1\nIf you want to buy a ticket for another day: press2\nIf not:
press 3");
        while (buyerOption!=1 && buyerOption!=2 && buyerOption!=3)
        {
            System.out.print("Enter your option: ");
```

```
            try
            {
                buyerOption = sc.nextInt();
                if (buyerOption != 1 && buyerOption != 2 && buyerOption!=3)
{
                    throw new Exception("Enter an integer according to the
option");
                }
            }
            catch (InputMismatchException e) {
                System.out.println("Enter an integer according to the
option");
            } catch (Exception e) {
                System.out.println(e);
            } finally {
                sc.nextLine();
            }
        }
        return buyerOption;
    }


    public String[] loadAccountNumbers(XSSFSheet sheetTwo)
    {
        int rows = sheetTwo.getLastRowNum();
        for (int r = 0; r<=rows; r++)
        {
            XSSFRow row = sheetTwo.getRow(r);
            XSSFCell cell = row.getCell(1);
            accountNumberArray[r] = cell.getStringCellValue();
        }
        return accountNumberArray;
    }
    public String enterCardNumber()
    {
        String cardNumber = "0";
        System.out.print("Card Number: ");
        while (cardNumber.length() != 16)
        {
            try
            {
                cardNumber = sc.next();
                if (cardNumber.length() != 16)
                    throw new Exception("Your card number must be contained
16 numbers");
            }
            catch (Exception e)
            {
                System.out.println(e);
            }
        }
        return cardNumber;
    }

    public String enterBuyerName()
    {
        System.out.print("Name: ");
        String name = sc.next();
        return name;
    }
```

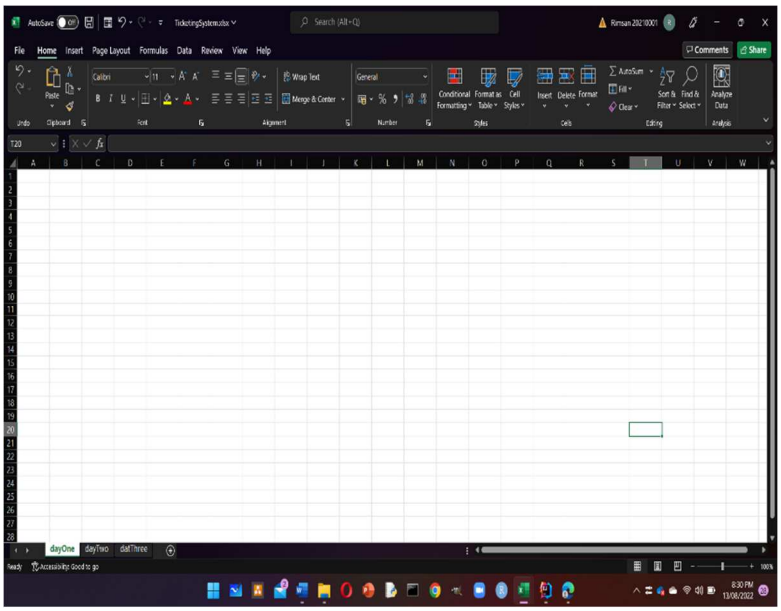
```
    public void enterCVCNumber()
    {
        String CVCNumber = "0";
        System.out.print("Enter CVC number: ");
        while (CVCNumber.length() != 3)
        {
            try
            {
                CVCNumber = sc.next();
                if (CVCNumber.length() != 3)
                    throw new Exception("Your CVC number must be contained
3 numbers");
                try
                {
                    Integer.parseInt(CVCNumber);
                }
                catch (Exception e)
                {
                    System.out.println("Your CVC number must be contained
only numbers");
                }
            }
            catch (Exception e)
            {
                System.out.println(e);
            }
        }
    }

    public void enterCardExpiryDate()
    {
        System.out.print("Enter the expiry date of your card: ");
        String expiryDate = sc.next();
    }
}
```

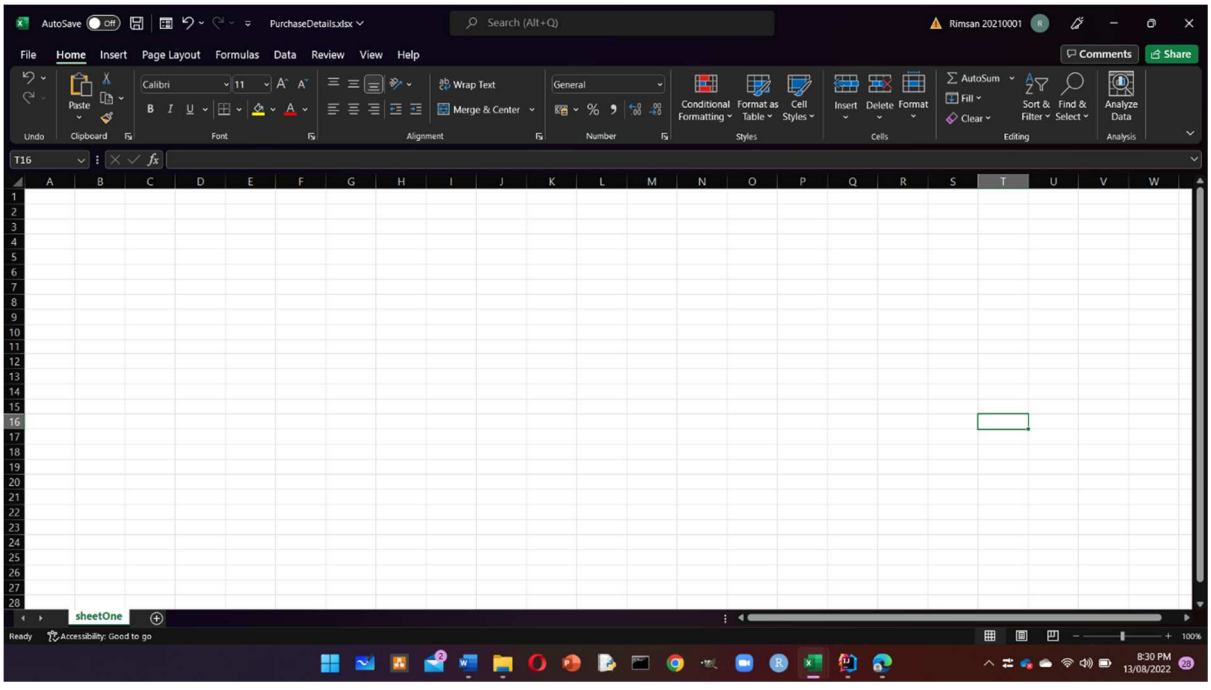
Outputs:

- In here I have used two excel workbooks to maintain a buyer queue every three days, and purchase details of buyers.

1. TicketingSystem.xlsx
   sheet – dayOne



PurchseDetails.xlsx

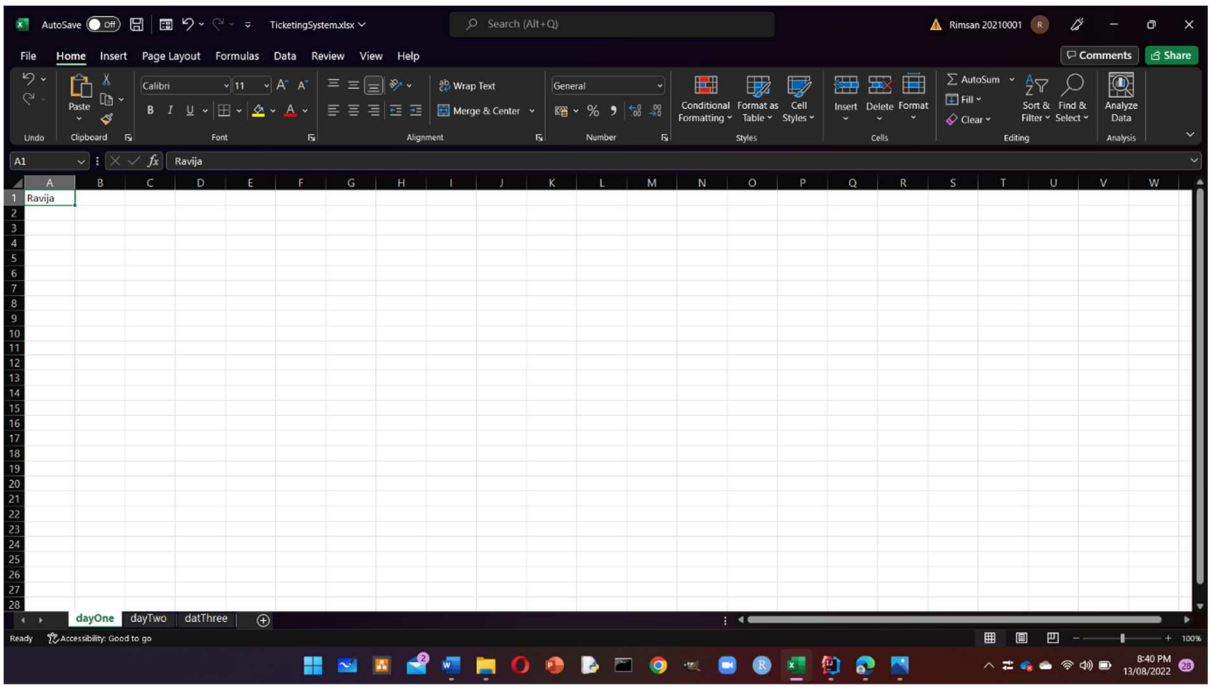


If you are the admin: press 1
If you are a ticket buyer: press 2
Enter your option:
2
Ticket price is Rs: 1500
If you want to buy: press 1
If not: press 2

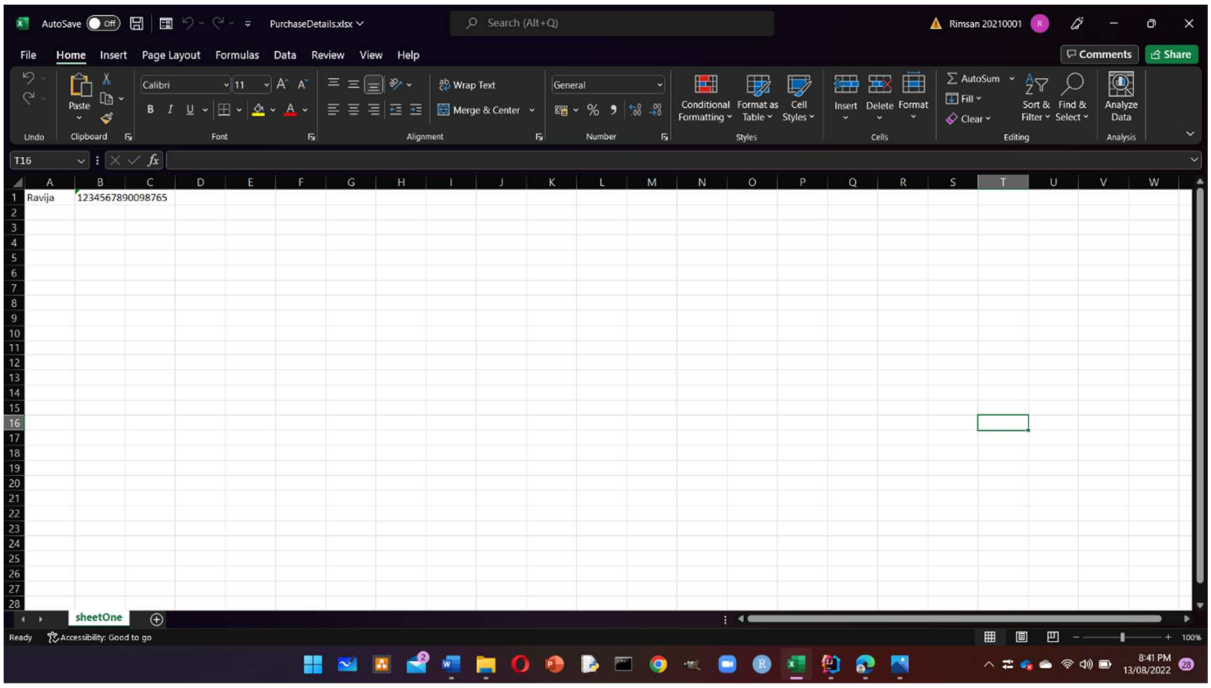
Enter your option: 1
Enter the day using numbers:
1
Name: Ravija
Card Number: 1234567890098765
Enter the expiry date of your card: 2/25
Enter CVC number: 456
Your name has added to the ticket queue, your ticket will be issued with in 24 hours
If you want to buy a ticket again with in same day: press 1
If you want to buy a ticket for another day: press2
If not: press 3
Enter your option: 1
Name: Rashadha
Card Number: 1234567890098765
Sorry, You cannot transfer your ticket for another person
If you want to buy a ticket again with in same day: press 1
If you want to buy a ticket for another day: press2
If not: press 3
Enter your option: 3

Process finished with exit code 0

TicketingSystem.xlsx
sheet – dayOne



PurchseDetails.xlsx



2.

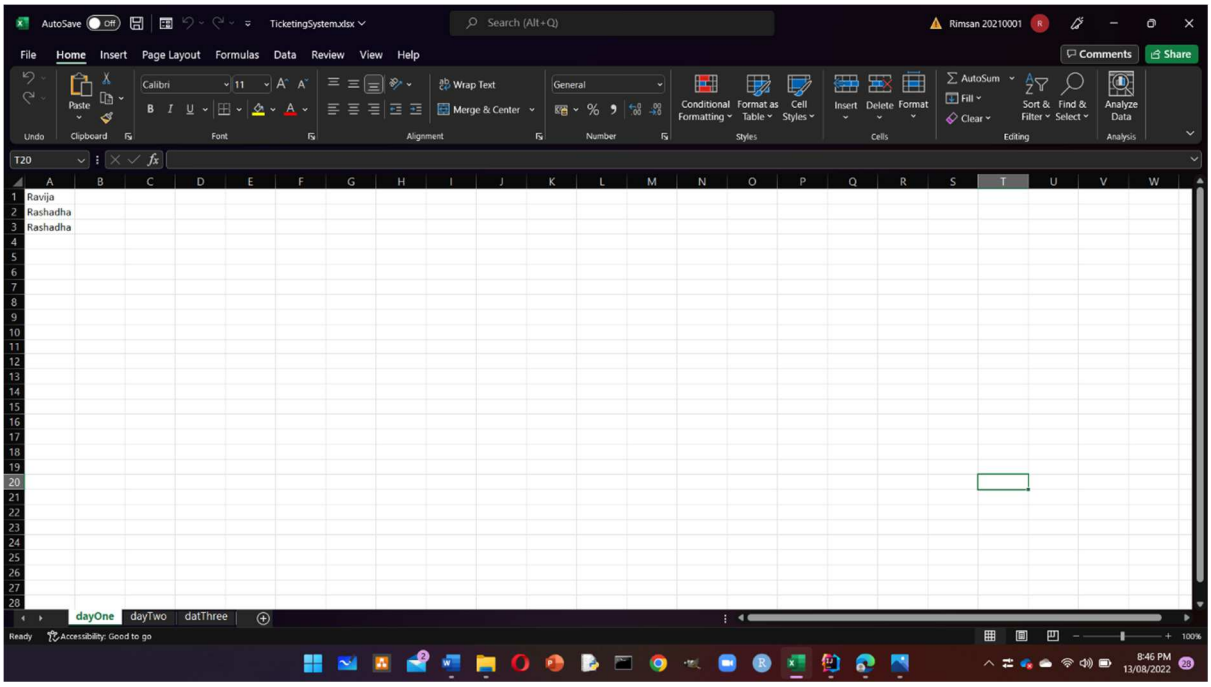
If you are the admin: press 1
If you are a ticket buyer: press 2
Enter your option:
2
Ticket price is Rs: 1500
If you want to buy: press 1
If not: press 2

Enter your option: 1
Enter the day using numbers:
1
Name: Rashadha
Card Number: 2345543223455432
Enter the expiry date of your card: 2/25
Enter CVC number: 012
Your name has added to the ticket queue, your ticket will be issued with in 24 hours
If you want to buy a ticket again with in same day: press 1
If you want to buy a ticket for another day: press2
If not: press 3
Enter your option: 1
Name: Rashadha
Card Number: 2345543223455432
Enter the expiry date of your card: 2/25
Enter CVC number: 012
Your name has added to the ticket queue, your ticket will be issued with in 24 hours
If you want to buy a ticket again with in same day: press 1
If you want to buy a ticket for another day: press2
If not: press 3
Enter your option: 2
Enter the day using numbers:
2
Name: Rashadha
Card Number: 2345543223455432
Enter the expiry date of your card: 2/25
Enter CVC number: 012
Your name has added to the ticket queue, your ticket will be issued with in 24 hours
If you want to buy a ticket again with in same day: press 1
If you want to buy a ticket for another day: press2
If not: press 3
Enter your option: 3

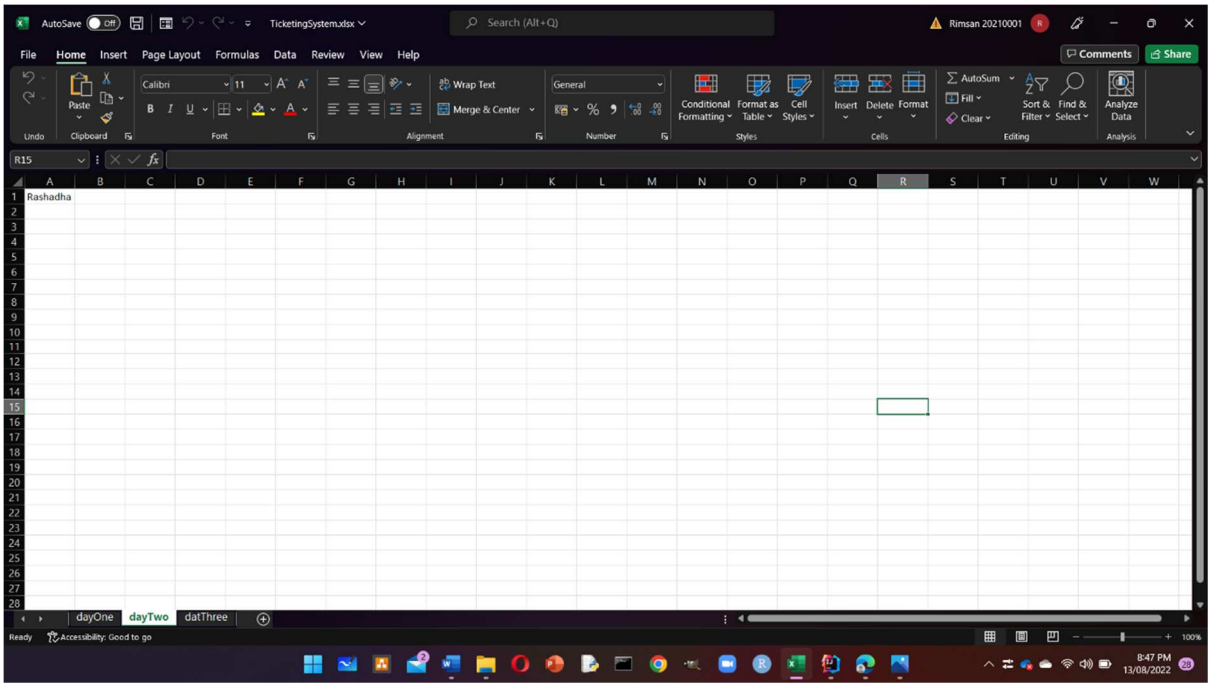
Process finished with exit code 0

TicketingSystem.xlsx

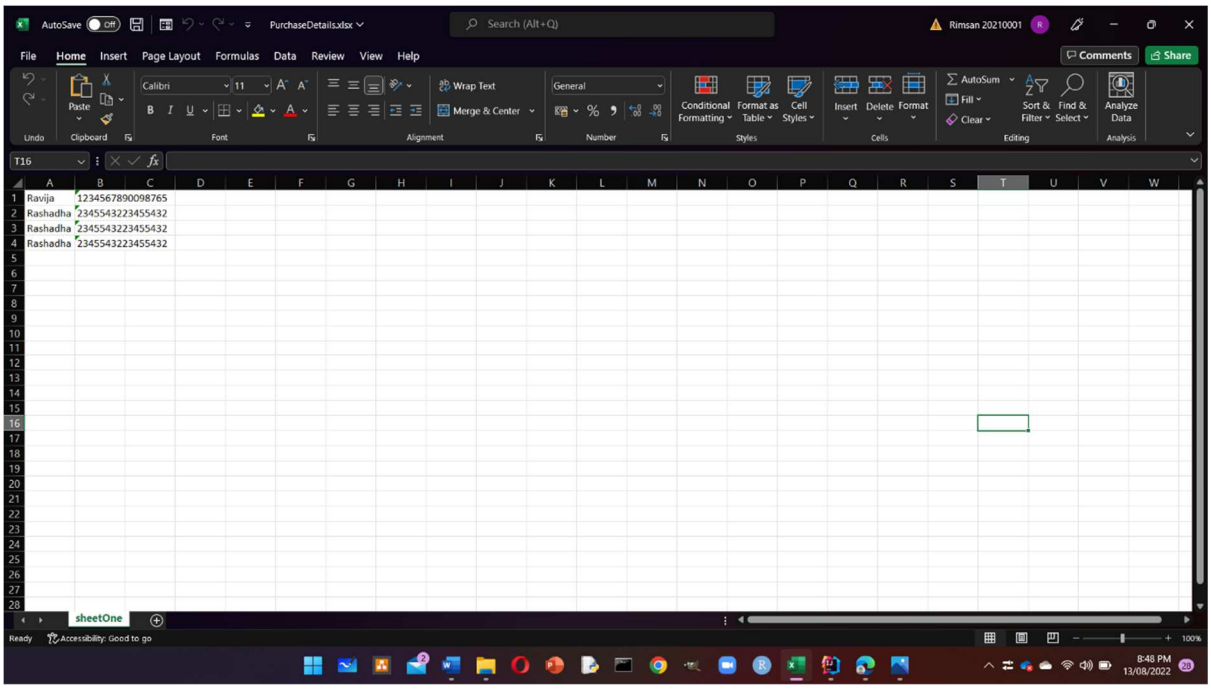
sheet – dayOne



sheet – dayTwo

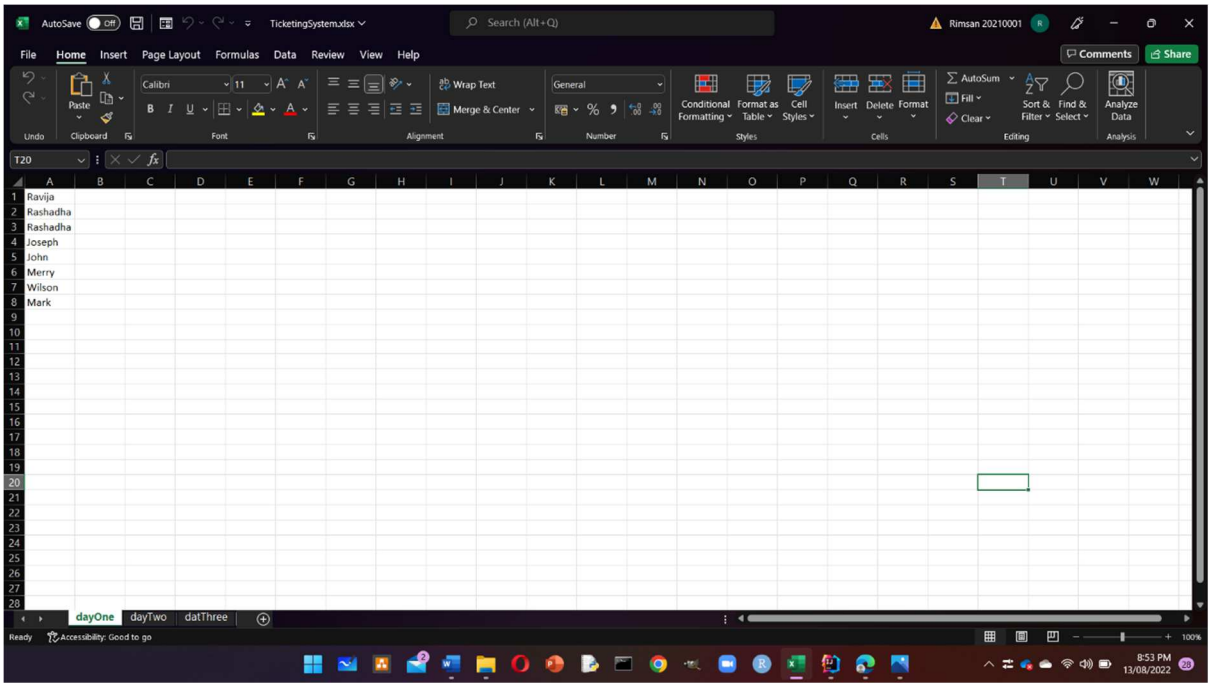


purchaseDetails.xlsx



3.

TicketingSystem.xlsx
sheet – dayOne

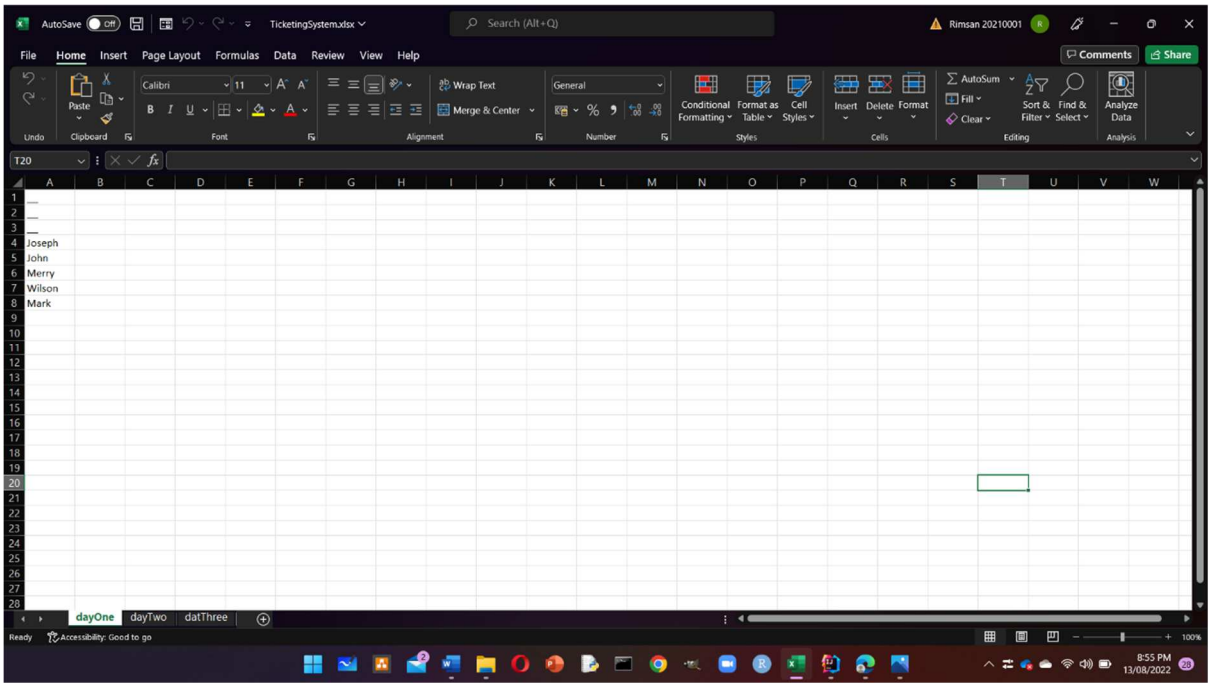


If you are the admin: press 1
If you are a ticket buyer: press 2
Enter your option:
1
Enter your password: adminPassword

Enter the day using numbers:
1
You issued a ticket for Ravija
If you want to issue a ticket for another buyer with in same day: press 1
If you want to issue a ticket for another buyer for another day: press2
If not: press 3
Enter your option: 1
You issued a ticket for Rashadha
If you want to issue a ticket for another buyer with in same day: press 1
If you want to issue a ticket for another buyer for another day: press2
If not: press 3
Enter your option: 1
You issued a ticket for Rashadha
If you want to issue a ticket for another buyer with in same day: press 1
If you want to issue a ticket for another buyer for another day: press2
If not: press 3
Enter your option: 3

Process finished with exit code 0

TicketingSystem.xlsx
sheet – dayOne

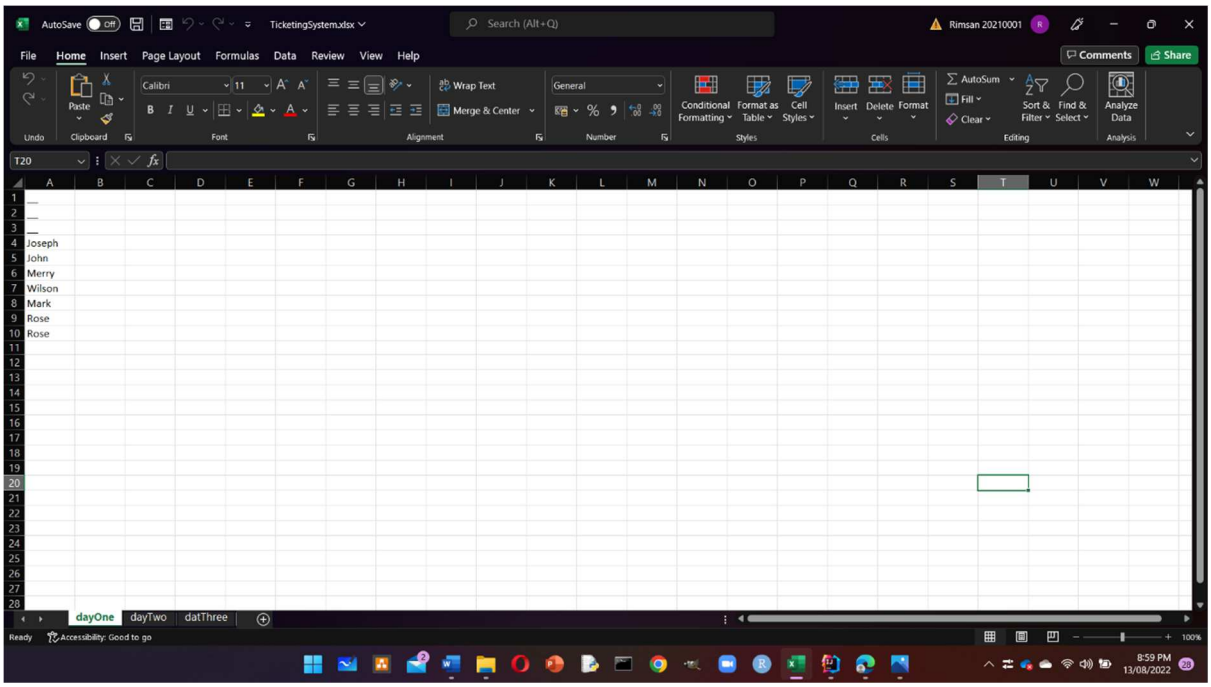


4.

If you are the admin: press 1
If you are a ticket buyer: press 2
Enter your option:
2
Ticket price is Rs: 1500
If you want to buy: press 1
If not: press 2
Enter your option: 1
Enter the day using numbers:
1
Name: Rose
Card Number: 1234098765566565
Enter the expiry date of your card: 2/24
Enter CVC number: 147
Your name has added to the ticket queue, your ticket will be issued with in 24 hours
If you want to buy a ticket again with in same day: press 1
If you want to buy a ticket for another day: press2
If not: press 3
Enter your option: 1
Name: Rose
Card Number: 1234098765566565
Enter the expiry date of your card: 2/24
Enter CVC number: 147
Your name has added to the ticket queue, your ticket will be issued with in 24 hours
If you want to buy a ticket again with in same day: press 1
If you want to buy a ticket for another day: press2
If not: press 3
Enter your option: 3

Process finished with exit code 0

TicketingSystem.xlsx
sheet – dayOne



Task 2

a.

Linear Search Algorithm

```
public class LinearSearch
{
    public static void main(String[] args)
    {
        int[] array = {1, 2, 3, 5, 7, 8, 9, 10, 4, 6};
        int value = 9;
        int index = search(array,value);
        if (index == -1)
        {
            System.out.println(value+" is not found in the array");
        }
        else
        {
            System.out.println(value+" is found at array index "+index);
        }
    }
    public static int search(int[] array, int x)
    {
        int n = array.length;
        for (int i = 0; i<n; i++)
        {
            if (array[i] == x)
            {
                return i;
            }
```

```
        }
        return -1;
    }
}
```

Binary Search Algorithm

```
public class BinarySearch
{
    public static void main(String[] args)
    {
        int[] array = {2,5,8,12,16,23,38,56,72,91};
        System.out.println(search(array,0,array.length-1,72));

    }
    public static int search(int[] array, int start, int end, int key)
    {
        if (end >= start)
        {
            int mid =(start + end)/2;

            if (array[mid] == key)
            {
                return mid;
            }
            if (array[mid] > key)
            {
                return search(array, start, mid - 1, key);
            }
            return search(array, mid+1,end,key);
        }
        return -1;
    }
}
```

b.

Linear Search	Binary Search
• It's a sequential search algorithm. • The key that you are searching for in the given array is searching in sequential order. • If the key is found, it returns the index number of that key, if it's not found, it returns -1. • Complexity – O(n)	• If we are using binary search in an array, we must ensure that the array is sorted. • It follows the divide and conquers approach. • The array is divided into two halves and the key is compared with the middle most element of the array,

	✓ If a match occurs, then the index of the element is returned. ✓ if the middle element is greater than the key, then the key is searched in the left sub-array of the middle element. ✓ Otherwise the key is searched in the right sub-array of the middle element. ✓ The process is continues until the size of subarray reduces to zero. Complexity – O(log n)

c. Insertion sort

```
public class InsertionSort
{
    public static void main(String[] args)
    {
```

```
        int[] array = {4,3,2,10,12,1,5,6};
        sort(array);
        for (int data : array)
        {
            System.out.println(data);
        }

    }
    public static void sort(int[] array)
    {
        int n = array.length;

        for (int i = 1; i<n; i++)
        {
            int key = array[i];
            int j = i -1;

            while (j>=0 && array[j]>key)
            {
                array[j+1] = array[j];
                j=j-1;
            }
            array[j+1] = key;
        }
    }
}
```

Bubble sort

```
public class BubbleSort
{
    public static void main(String[] args)
    {
        int[] arr = {77,42,35,12,101,5};
        sort(arr);
        for (int data: arr)
        {
            System.out.println(data);
        }
    }
    public static void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i<n; i++)
        {
            for (int j = 0; j < n-i-1; j++)
            {
                if (arr[j] > arr[j+1])
                {
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
        }
    }
}
```

d.

Insertion Sort	Bubble Sort
• In insertion sort, the array is virtually split into sorted and unsorted parts. • Values from the unsorted part are picked and placed at the correct position in the sorted part. • Finally, the array will be sorted in ascending order. • Complexity – o(n^2)	• In bubble sort we will the largest value be moved to the end of the array using pairwise comparisons and swapping. • We have to repeat the same process for all the elements. • Complexity – O(n^2)