

# Acknowledgement

I would like to express our special thanks of gratitude to our OOPS teacher “Mr Debendra Muduli” who gave us the golden opportunity to do the casestudy.

It helped us in doing research and came to know about so many new

things, so we are very thankful to you because under your guidance and

support we were able to complete this case study report on time.

# DECLARATION

I hereby declare that report entitled "Ary Management System " given in OOPS Case Study is submitted by me and has been carried out by intense efforts and facts at observation and some research under the guidance and motivation of subject teacher "Mr Debendra Muduli".

# Context

<b>SL No.</b>	<b>Outlines</b>	<b>Page NO.</b>
<b>2</b>	<b>Abstract</b>	<b>5</b>
<b>3</b>	<b>Introduction</b>	<b>6</b>
<b>4</b>	<b>Feasibility Study</b>	<b>9</b>
<b>5</b>	<b>Source Code</b>	<b>10</b>
<b>6</b>	<b>Output</b>	<b>17</b>
<b>7</b>	<b>Conclusion</b>	<b>24</b>

# Student Details

<b>Ahmad Ashraf Zargar</b>	<b>Roll No CSE20080</b>	<b>Regd No 20010073</b>
--------------------------------	-----------------------------	-----------------------------

# **Abstract**

The Army Management System project is a program that keeps track of an Army's allocated supplies, soldier allocation, resource allocation and other important information required for a Management System. This project demonstrates the operation of an Army Management System and covers the essential functions of management software with respect to users. It develops a project for resolving basic requirements in a management environment to meet the needs of an end-user by providing multiple ways to complete management chores. Additionally, this project is to provide additional features to the user's workspace that are not available in a traditional management project or file-based project. This project's primary goal is to create an Army management system program. This project was designed to make it simple and quick to complete previously impossible processes with manual systems which are now possible with different types of software.

# Introduction

This program is used for the creation of a management system which will be used by the army for managing their necessary requirements. This project will be programmed using JAVA (OOPS), which provides us with the features of OOPS and JAVA.

Features of Java include:

- Platform independent
- Simple
- Secure
- Robust and many more

This case study's end goal is to create a management system which can be used in place of the old file-based system, that used to exist with a new programming based system. This will provide security one of the most important part of management for the army, easy in modifying, and updating the information.

The Army Management System is made based on Army requirements to handle and store the data fed by the users of the system.

This project intends to introduce more user-friendliness in various activities such as record updating, maintenance, and searching. The search of record has been made quite simple as all the details of the soldier can be obtained by simply keying in the identification of that army regiment. Similarly, record maintenance and updation can also be accomplished by using the attributes with all the details being automatically generated.

Features:

- - This project allows the users to insert, update, deletion data with greater efficiency.

- - The project also reduces the redundancy which can occur in the database/ file-based system.
- - The records can be secured for unauthorized access.

## **Problem Statement**

An Army has a commanding officer, which is in control of all the regiments, their soldier allocation, resource allocation and mission allocation.

The commanding officer is identified by a code, which needs to be entered to access the program.

The program can be used to create regiments, display regiments, and inventory details of regiments.

Each regiment has a captain who is in charge of the regiment, and each regiment has a base camp, and a mission allocated to them.

The regiment also contains information about the soldiers present in it with their name, height and BMI details.

Inventory deals with the purpose of allocation of resources whether it be guns, food or health care provider.

After that, we have to display the information of a regiment and the inventory details of that regiment.

The details of the regiment and inventory are also stored in the file-based “.txt” format for future uses.



# FEASIBILITY STUDY

Feasibility is the process of defining exactly what is and what strategic issue needs to be considered to assess its feasibility, or likelihood of succeeding.

A feasibility study tries to determine whether a given solution would work out or not to solve the problem, but to acquire its scope. It focuses on the following:

1. Meet user requirements.
2. Best utilization of available resources
3. Develop a cost-efficient system
4. Develop a technically feasible system

The question which is answered by the feasibility study for this case study

1. What resources are available for the given candidate system? Is the problem worth solving?
2. What is the likely impact of the candidate system on the organization?
- 3.. How well does it fit within the organization's master plan?

Feasibility study For Army Management System:

Yes, it is technically feasible to design such a system. There are technology and software available which can be used to easily design Army Management Systems. As modern devices are available with storage space which is in Gigabytes and TeraBytes, our system can have storage for a long time before needing to upgrade. The data required will be instantly delivered without any delay. It is a very reliable system and a very safe system where data security is of the most importance. Our system is also economically feasible. It can be installed easily but the initial cost of implementation can be higher in comparison to the file-based system but the benefit of once implementing this system is immense and will be beneficial in the coming number of years.

# Source Code

## Main Class:

```
package ArmyManagementSystem;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.*;
import java.io.File;

public class Main extends Thread{
    static final String commName = "Ahmad Ashraf";
    final int commCode = 4321;

    public static void main(String[] args){
        Main commander = new Main();
        System.out.print("Enter Identification Code: ");
        Scanner scan = new Scanner(System.in);
        int IdCode = scan.nextInt();
        if (IdCode == commander.commCode){
            System.out.println("WELCOME COMMANDER: "+commName+"\n\n");
            boolean continueMenu = true;
            while(continueMenu) {
                System.out.println("*****\n");
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                System.out.println("1. Create Regiment");
                System.out.println("2. Display a Regiment");
                System.out.println("3. Inventory Detail Of Regiment");
                System.out.println("4. Exit");

                int choice = scan.nextInt();
                switch (choice) {
                    case 1: {
                        try {
                            Thread.sleep(1000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                        System.out.print("Enter Regiment Name: ");
                        String RegimentName = scan.next();
                        System.out.print("\nEnter Regiment Base Camp: ");
                        String RegimentBase = scan.next();
                        System.out.println();
                        Regiment regiment1 = new Regiment(RegimentName, RegimentBase);

                        break;
                    }
                    case 2: {
                        try {
                            Thread.sleep(1000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            }
        }
    }
}
```

```

    }
    System.out.print("Enter Name of Regiment: ");
    String RegimentName = scan.next();
    DisplayRegiment displayRegiment = new DisplayRegiment(RegimentName);
    try {
        displayRegiment.detailRegiment();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    break;
}
case 3: {
    try {
        Thread.sleep(2000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    Inventory inventory = new Inventory();
    System.out.print("Enter Regiment Name: ");
    String name = scan.next();
    inventory.main(name);
}
case 4: {
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    continueMenu = false;
    break;
}

default:
    break;
}
}
}
}
}
}

```

## Regiment Class:

```

package ArmyManagementSystem;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;
import java.io.File;

```

```

public class Regiment {
    String RegimentName;

```

```

String RegimentCaptain;
String Mission;
String BaseCamp;
static Scanner scan = new Scanner(System.in);
FileWriter solFile;

Regiment (String RegimentName, String BaseCamp){
    this.RegimentName = RegimentName;
    this.BaseCamp = BaseCamp;
    this.RegimentCaptain = allocateCaptain();
    this.Mission = allocateMission();
    {
        try {
            solFile = new FileWriter(RegimentName+"SoldierDetail.txt");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    try {
        addRegimentDetail();
        addSoldier();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

void addRegimentDetail(){
    try {
        solFile.write("REGIMENT NAME: " + RegimentName + "\n\n");
        solFile.write("REGIMENT BASE: " + BaseCamp + "\n\n");
        solFile.write("REGIMENT CAPTAIN: " + RegimentCaptain + "\n\n");
        solFile.write("MISSION: " + Mission + "\n\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

void addSoldier() throws IOException {
    System.out.print("Enter the no of Soldiers: ");
    int noSol = scan.nextInt();
    ArrayList<String> soldier= new ArrayList<>();
    solFile.write("\nSOLDIER\t NAMES\t HEIGHT\t BMI\n");
    for (int i=1; i<=noSol; i++){
        System.out.print("Enter Soldier Name: ");
        String name = scan.next();
        System.out.print("\nEnter Soldier Height: ");
        double height = scan.nextDouble();
        System.out.print("\nEnter Solider BMI: ");
        double bmi = scan.nextDouble();
        soldier.add(name);
        solFile.write("SoldierID " +i+ " : " +name+"\t"+height+"\t"+bmi+"\n");
    }
    solFile.close();
}

String allocateMission(){
    System.out.print("Enter Name of Mission: ");
    String Mission = scan.next();
    return Mission;
}

String allocateCaptain(){

```

```

        System.out.print("Enter Captain Name: ");
        String RegimentCaptain = scan.next();
        return RegimentCaptain;
    }
}

```

## Inventory Class:

```

package ArmyManagementSystem;
import java.io.*;
import java.util.*;

public class Inventory extends Thread{
    final int totalGuns = 20;
    static int availableGuns=20;
    HashMap <String, String> inventory = new HashMap<>();
    Scanner scan = new Scanner(System.in);
    void main(String Name){
        boolean continueMenu = true;
        while (continueMenu){
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println("INVENTORY OF "+Name+" REGIMENT\n");
            System.out.println("*****");
            System.out.println("1. Allocate Guns");
            System.out.println("2. Allocate Health Facilities");
            System.out.println("3. Allocate Food");
            System.out.println("4. Change Health Facilities");
            System.out.println("5. Allocate more Food");
            System.out.println("6. Allocate more Guns");
            System.out.println("7. Display Inventory");
            System.out.println("8. Update File");
            System.out.println("9. EXIT");

            int userChoice = scan.nextInt();
            switch (userChoice){
                case 1: {
                    System.out.print("Enter Number Of Guns: ");
                    int Guns = scan.nextInt();
                    allocateGuns(Guns);
                    break;
                }
                case 2: {
                    System.out.print("Enter Name Of Health Provider: ");
                    String healthName = scan.next();
                    allocateHeath(healthName);
                    break;
                }
                case 3: {
                    System.out.println("Enter Food Allocate(Days): ");
                    String foodDays = scan.next();
                    allocateFood(foodDays);
                }
            }
        }
    }
}

```

```

        break;
    }
    case 4: {
        System.out.print("Enter Name Of New Health Provider: ");
        String healthName = scan.next();
        ChangeHeath(healthName);
        break;
    }
    case 5: {
        System.out.print("Allocate More Food: ");
        String moreFood = scan.next();
        allocateMoreFood(moreFood);
        break;
    }
    case 6: {
        System.out.print("Allocate More Guns: ");
        int Guns = scan.nextInt();
        allocateMoreGuns(Guns);
        break;
    }
    case 7: {
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.print("Enter Name of Regiment: ");
        String RegimentName = scan.next();
        DisplayInventory displayInventory = new DisplayInventory(RegimentName);
        try {
            displayInventory.detailInventory();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        break;
    }
    case 8: {
        System.out.print("Enter File Name: ");
        String FileName = scan.next();
        updateFile(FileName);
        break;
    }
    case 9: {
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        continueMenu = false;
        break;
    }
}
}
}
}
void allocateGuns(int guns){
    if (availableGuns>guns){
        availableGuns-=guns;
        String Guns=String.valueOf(guns);
        this.inventory.put("Guns", Guns);
    }
}

```

```

        else{
            System.out.println("Guns Resource Insufficient");
        }
    }
    void allocateMoreGuns(int guns){
        if (availableGuns>guns){
            availableGuns-=guns;
            String Guns=String.valueOf(guns);
            this.inventory.replace("Guns", Guns);
        }
        else{
            System.out.println("Guns Resource Insufficient");
        }
    }
    void allocateHeath(String healthName){
        this.inventory.put("HealthProvider", healthName);
    }
    void ChangeHeath(String healthName){
        this.inventory.replace("HealthProvider", healthName);
    }
    void allocateFood(String foodDays){
        this.inventory.put("Food", foodDays);
    }
    void allocateMoreFood(String foodDays){
        this.inventory.replace("Food", foodDays);
    }
    void updateFile(String Name){
        try {
            BufferedWriter bf = new BufferedWriter(new FileWriter("/Users/ahmadashraf/Desktop/java/CaseStudy/"+Name+"Inventory.txt"));
            for (Map.Entry<String, String> entry : inventory.entrySet()) {

                bf.write(entry.getKey() + ":" + entry.getValue());

                bf.newLine();
            }
            bf.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

## Display Regiment:

```

package ArmyManagementSystem;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class DisplayRegiment {
    String RegimentName;
    DisplayRegiment(String RegimentName){

```

```

        this.RegimentName="/Users/ahmadashraf/Desktop/java/
CaseStudy/"+RegimentName+"SoldierDetail.txt";
    }
    void detailRegiment() throws FileNotFoundException {
        File detail = new File(RegimentName);
        Scanner scan = new Scanner(detail);
        while(scan.hasNextLine()){
            String line = scan.nextLine();
            System.out.println(line);
        }
        scan.close();
    }
}

```

## Display Inventory:

```

package ArmyManagementSystem;

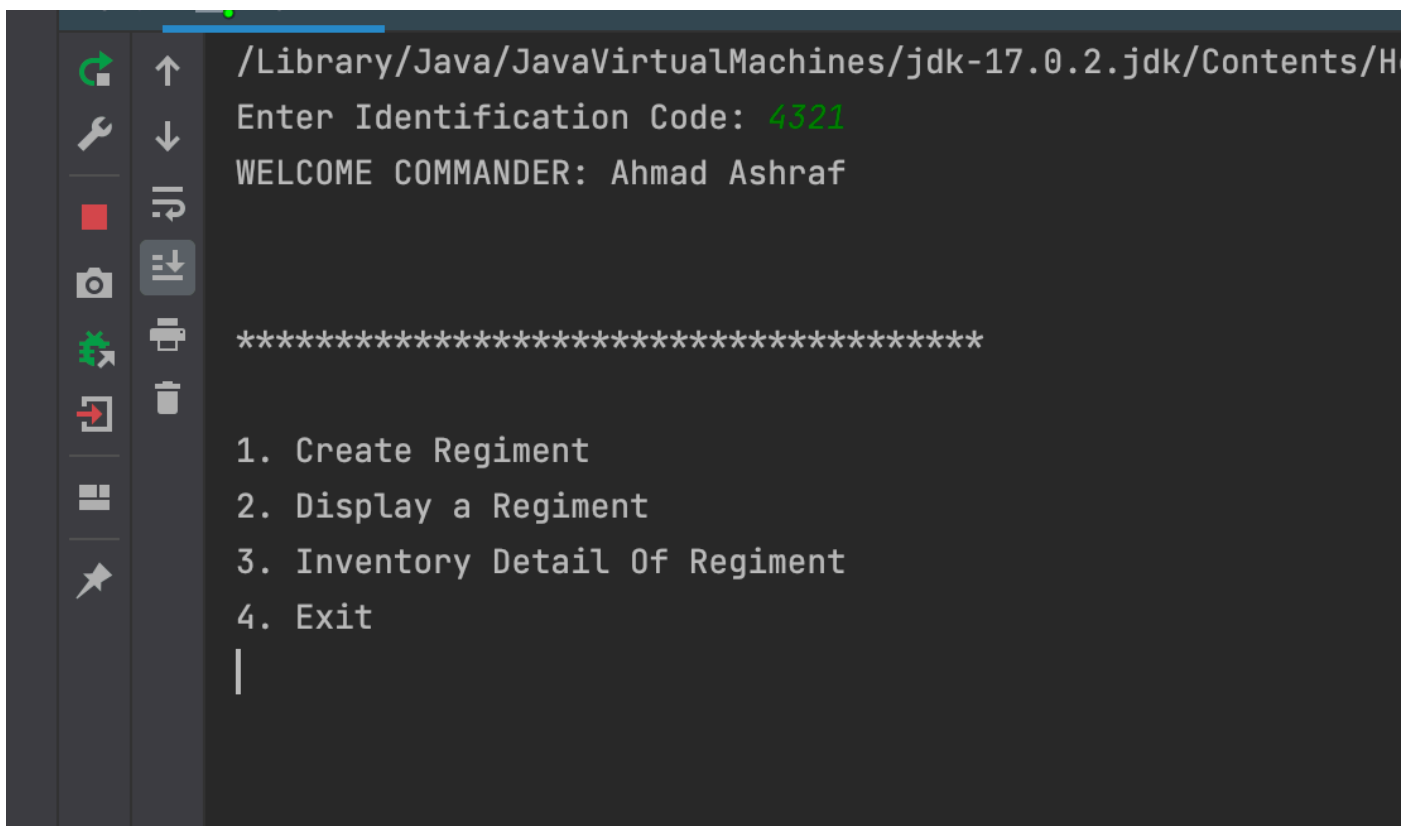
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class DisplayInventory {
    String RegimentName;
    DisplayInventory(String RegimentName){
        this.RegimentName="/Users/ahmadashraf/Desktop/java/CaseStudy/"+RegimentName+"Inventory.txt";
    }
    void detailInventory() throws FileNotFoundException {
        File detail = new File(RegimentName);
        Scanner scan = new Scanner(detail);
        while(scan.hasNextLine()){
            String line = scan.nextLine();
            System.out.println(line);
        }
        scan.close();
    }
}

```



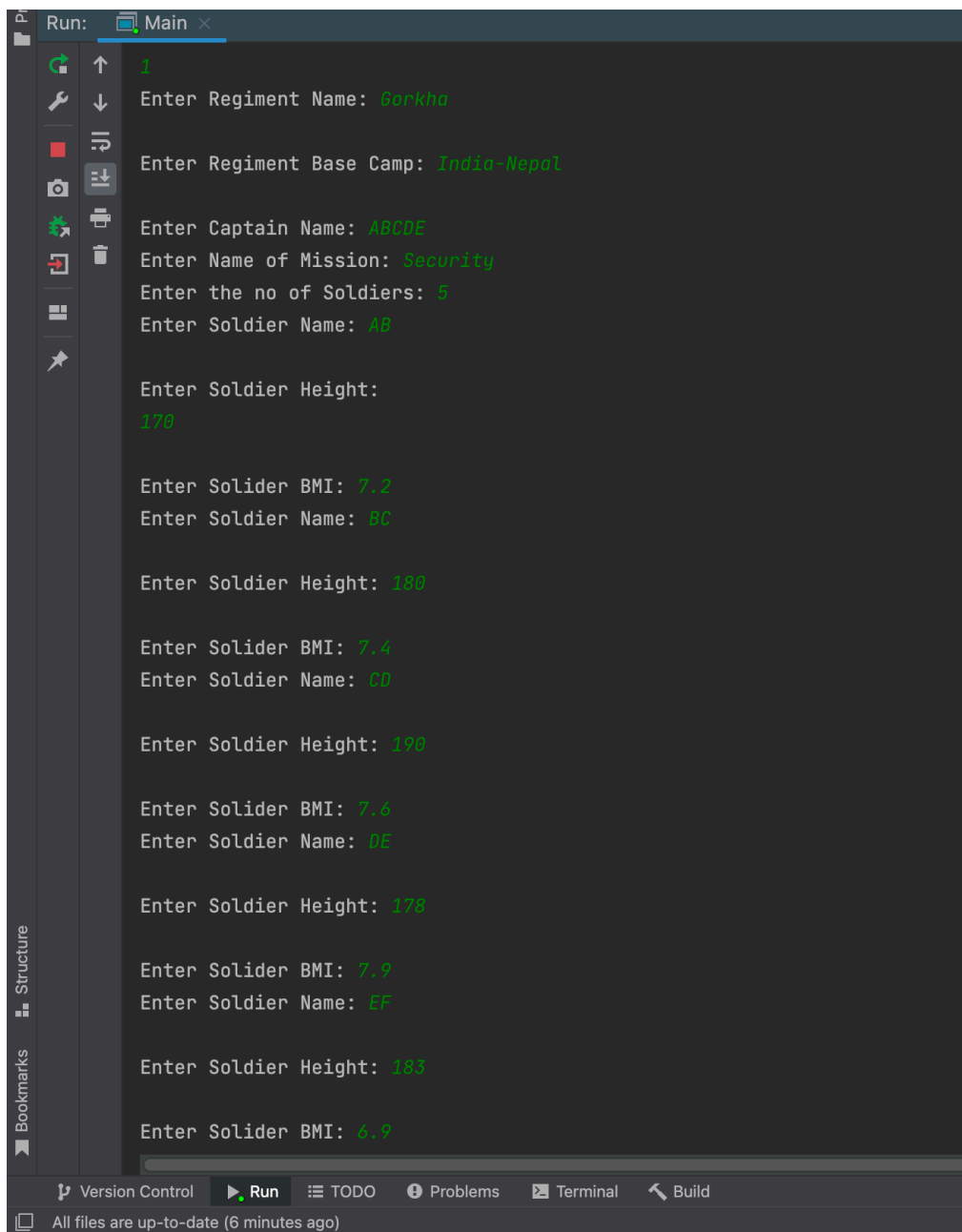
# Output

A screenshot of a Java IDE terminal window. The terminal has a dark background with a vertical toolbar on the left containing icons for undo, redo, run, debug, and other IDE functions. The output text is as follows:

```
/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/H
Enter Identification Code: 4321
WELCOME COMMANDER: Ahmad Ashraf

*****

1. Create Regiment
2. Display a Regiment
3. Inventory Detail Of Regiment
4. Exit
|
```



The image shows a screenshot of a Visual Studio Code (VS Code) terminal window. The terminal is titled "Run: Main x" and displays the output of a program. The program prompts the user to enter various details, and the user's inputs are shown in green text. The inputs are: Regiment Name: Gorkha, Regiment Base Camp: India-Nepal, Captain Name: ABCDE, Mission Name: Security, Number of Soldiers: 5, Soldier Name: AB, Soldier Height: 170, Soldier BMI: 7.2, Soldier Name: BC, Soldier Height: 180, Soldier BMI: 7.4, Soldier Name: CD, Soldier Height: 190, Soldier BMI: 7.6, Soldier Name: DE, Soldier Height: 178, Soldier BMI: 7.9, Soldier Name: EF, Soldier Height: 183, and Soldier BMI: 6.9. The terminal interface includes a left sidebar with icons for Explorer, Search, Run and Debug, and Source Control. The bottom status bar shows "All files are up-to-date (6 minutes ago)".

```
Run: Main x
1
Enter Regiment Name: Gorkha
Enter Regiment Base Camp: India-Nepal
Enter Captain Name: ABCDE
Enter Name of Mission: Security
Enter the no of Soldiers: 5
Enter Soldier Name: AB

Enter Soldier Height:
170

Enter Solider BMI: 7.2
Enter Soldier Name: BC

Enter Soldier Height: 180

Enter Solider BMI: 7.4
Enter Soldier Name: CD

Enter Soldier Height: 190

Enter Solider BMI: 7.6
Enter Soldier Name: DE

Enter Soldier Height: 178

Enter Solider BMI: 7.9
Enter Soldier Name: EF

Enter Soldier Height: 183

Enter Solider BMI: 6.9
```

Version Control Run TODO Problems Terminal Build

All files are up-to-date (6 minutes ago)

```
Enter Soldier Height: 170
Enter Solider BMI: 7.9
Enter Soldier Name: EF
Enter Soldier Height: 183
Enter Solider BMI: 6.9
*****
1. Create Regiment
2. Display a Regiment
3. Inventory Detail Of Regiment
4. Exit
2
Enter Name of Regiment: ChinarCorps
REGIMENT NAME: ChinarCorps

REGIMENT BASE: Kashmir

REGIMENT CAPTAIN: Ashraf

MISSION: Security

SOLDIER      NAMES      HEIGHT  BMI
SoldierID 1 :Ahmad  172.0   7.1
SoldierID 2 :Zargar 182.0   7.6
```

```
2. Display a Regiment
3. Inventory Detail Of Regiment
4. Exit
2
Enter Name of Regiment: Gorkha
REGIMENT NAME: Gorkha

REGIMENT BASE: India-Nepal

REGIMENT CAPTAIN: ABCDE

MISSION: Security

SOLDIER      NAMES      HEIGHT  BMI
SoldierID 1 :AB 170.0   7.2
SoldierID 2 :BC 180.0   7.4
SoldierID 3 :CD 190.0   7.6
SoldierID 4 :DE 178.0   7.9
SoldierID 5 :EF 183.0   6.9
*****

1. Create Regiment
2. Display a Regiment
3. Inventory Detail Of Regiment
4. Exit
```

```
*****

1. Create Regiment
2. Display a Regiment
3. Inventory Detail Of Regiment
4. Exit
3
Enter Regiment Name: ChinarCorps
INVENTORY OF ChinarCorps REGIMENT

*****

1. Allocate Guns
2. Allocate Health Facilities
3. Allocate Food
4. Change Health Facilities
5. Allocate more Food
6. Allocate more Guns
7. Display Inventory
8. Update File
9. EXIT
```

```
7/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/
Enter Identification Code: 4321
WELCOME COMMANDER: Ahmad Ashraf

*****

1. Create Regiment
2. Display a Regiment
3. Inventory Detail Of Regiment
4. Exit
3
Enter Regiment Name: ChinarCorps
INVENTORY OF ChinarCorps REGIMENT

*****

1. Allocate Guns
2. Allocate Health Facilities
3. Allocate Food
4. Change Health Facilities
5. Allocate more Food
6. Allocate more Guns
7. Display Inventory
8. Update File
9. EXIT
7
Enter Name of Regiment: ChinarCorps
Guns:10
HealthProvider:TATA
Food:25Days
|
```

```
INVENTORY OF GORKHA REGIMENT
*****
1. Allocate Guns
2. Allocate Health Facilities
3. Allocate Food
4. Change Health Facilities
5. Allocate more Food
6. Allocate more Guns
7. Display Inventory
8. Update File
9. EXIT
2
Enter Name Of Health Provider: TATA
INVENTORY OF Gorkha REGIMENT

*****
1. Allocate Guns
2. Allocate Health Facilities
3. Allocate Food
4. Change Health Facilities
5. Allocate more Food
6. Allocate more Guns
7. Display Inventory
8. Update File
9. EXIT
3
Enter Food Allocate(Days):
450days
INVENTORY OF Gorkha REGIMENT

*****
1. Allocate Guns
2. Allocate Health Facilities
3. Allocate Food
```

```
Run: Main x
5. Allocate more Food
6. Allocate more Guns
7. Display Inventory
8. Update File
9. EXIT
3
Enter Food Allocate(Days):
450days
INVENTORY OF Gorkha REGIMENT

*****
1. Allocate Guns
2. Allocate Health Facilities
3. Allocate Food
4. Change Health Facilities
5. Allocate more Food
6. Allocate more Guns
7. Display Inventory
8. Update File
9. EXIT
8
Enter File Name: Gorkha
INVENTORY OF Gorkha REGIMENT

*****
1. Allocate Guns
2. Allocate Health Facilities
3. Allocate Food
4. Change Health Facilities
5. Allocate more Food
6. Allocate more Guns
7. Display Inventory
8. Update File
9. EXIT
|
```

```
Project
Main.java × Regiment.java × DisplayInventory.java × In
1 REGIMENT NAME: ChinarCorps
2
3 REGIMENT BASE: Kashmir
4
5 REGIMENT CAPTAIN: Ashraf
6
7 MISSION: Security
8
9
10 SOLDIER NAMES HEIGHT BMI
11 SoldierID 1 :Ahmad 172.0 7.1
12 SoldierID 2 :Zargar 182.0 7.6
13
```

```
CaseStudy > ChinarCorpsInventory.txt
Project
Main.java × Regiment.java × DisplayInventory.java × In
1 Guns:10
2 HealthProvider:TATA
3 Food:25Days
4
```

```
Project
Main.java × Regiment.java × DisplayInventory.java × Inven
1 REGIMENT NAME: Gorkha
2
3 REGIMENT BASE: India-Nepal
4
5 REGIMENT CAPTAIN: ABCDE
6
7 MISSION: Security
8
9
10 SOLDIER NAMES HEIGHT BMI
11 SoldierID 1 :AB 170.0 7.2
12 SoldierID 2 :BC 180.0 7.4
13 SoldierID 3 :CD 190.0 7.6
14 SoldierID 4 :DE 178.0 7.9
15 SoldierID 5 :EF 183.0 6.9
16
```

# Conclusion

In this Case Study, we have created a management system for Army. Which has all the latest features. This project is to provide additional features to the user's workspace that are not available in a traditional management project or file-based project. This project's primary goal is to create an Army management system program. This project was designed to make it simple and quick to complete previously impossible processes with manual systems which are now possible with different types of software.

In future, we would like to include a backend using SQL or PHP giving it more powerful features. In addition, we would also like to include a frontend like a webpage which would make it more user friendly. These are the goals for future development of this project.