UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SECJ2154

OBJECT ORIENTED PROGRAMMING

Systematic Squad

PROJECT

Bluestone Food Support

LECTURER	SECTION
Dr. Nies Hui Wen	10

Prepared by:

NAME	MATRIC
Toya Lazmin Khan	A20EC0284
Adnan Shafi	A20EC0255
Fayruz Tahzeeb Rahman	A20EC4019
GM Shaheen Shah Shimon	A20EC0266

Contents

1.0 Introduction

Near the end of 2019, the world faced a huge outbreak of a respiratory disease named COVID-19. It was later declared a pandemic and is still an ongoing global pandemic. COVID-19 took the lives of many loved ones and urged the world to go into a strict lockdown. This resulted in a huge loss to the economy and many small businesses as well as well-established businesses ended up shutting down temporarily or completely. Not only that, every place involving social gatherings was shut down until further notice. This made a huge chunk of the population lose their jobs and medical bills stripped money due to covid-19. People having a standard lifestyle ended up living in poverty.

On the first of April 2022, Malaysia declared COVID-19 an endemic disease and took steps to redo life the way it was. Unfortunately, even though a normal lifestyle commenced, there were many people who did not get their jobs back and are still struggling to make both ends meet.

The system we intend on making is targeted toward those people who are going through a hard time due to food insufficiency. Food is a prime need in a person's life and without it, people cannot survive. Even though we cannot help the needy from other aspects, we are willing to take the initiative to reduce a person's hunger through our system. Our system is also focused on the people who are willing to help people through the extra money they have. Moreover, people who have bought food in abundance and cannot finish it in time have the opportunity to donate it instead of letting it go to waste. The last feature of the system is to be able to volunteer in our noble act, for example: helping to deliver parcels from the donor to the seeker. This system is oriented towards the kind of process where a donor donates and the people in need receive their donation. Initially, our system will be based in Malaysia and we will be working with other organizations as well. The system is primarily implemented through a mobile application where all the aforementioned features will be available. Our system will be connected to multiple food supply chains and catering services for the continuous supply of food and to keep in stock. Our application will contain various features, the most prominent ones of which are 'Order food', where people or organizations will be able to order food and either pick it up from locations or get them delivered and 'Donate', where people or restaurants can either donate money or leftover food items and 'Volunteering', where people can apply to be volunteers.

Even though the system will be based on the covid-19 affected people, it is free for any person in need to be able to seek help from the system.

2.0 Project Planning

2.1 Relation between classes

The volunteer, donor, and receiver are associated with the organizer by aggregation. The object of the volunteer, donor, and receiver are declared in the organizer class. The class people and restaurants are the subclasses of the class donor. The donor is the parent class, and the people and restaurant are the child classes. The relationship of donors with people and restaurants is called inheritance.

2.1.1 Association

Donor class is associated with the organizer class

```
restaurant rest = new restaurant(); // restaurant class object

Vector<Receiver> rec = new Vector<Receiver>(); // creating a vector to store the receiver details

Vector<Receiver> acc = new Vector<Receiver>(); // creating vector of accepted receiver

Vector<volunteer> vol = new Vector<volunteer>();// creating a vector to store the volunteer details

Vector<donor> don = new Vector<donor>();// creating a vector to store the donor details

Vector<people> pp = new Vector<people>();// creating a vector to store the people donor

Vector<restaurant> rr = new Vector<restaurant>();// creating a vector to store the restaurant donor

ArrayList<String> donorNameArray = new ArrayList<String>();
ArrayList nameArray = new ArrayList();

Scanner inp = new Scanner(System.in);
```

2.1.2 Aggregation

Aggregation relationship is shown between the organizer and the receiver class. It is a HAS-A relationship.

```
public class organizer {
    private volunteer Volunteer;  //COMPOSITION
    protected Receiver receiver;  //AGGREGATION

    public organizer(volunteer Volunteer, Receiver receiver) {
        this.Volunteer = Volunteer;
        this.receiver = receiver;
    }

    // method Volunteer class to take input of volunteer
    public void takeInput(Vector<volunteer> vol, ArrayList nameArray) {
        Volunteer.takeInput(vol, nameArray);
    }

    // method to call displayVolunteer of Volunteer class
    public void displayVolunteerMethod(Vector<volunteer> vol) {
        Volunteer.displayVolunteer(vol);
    }

    // method to call deliverItem of Volunteer class
    public void deliverItemMethod(Vector<volunteer> vol, Vector<Receiver> acc, people people, restaurant rest,
            String type,
            int value) {
        Volunteer.deliverItem(vol, acc, people, rest, type, value);
    }

    // method to call input of receiver class
```

Fig: Aggregation

2.1.3 Composition

Composition relationship is shown between the organizer and the volunteer class.

```
public class organizer {
    private volunteer Volunteer;  //COMPOSITION
    protected Receiver receiver;  //AGGREGATION

    public organizer(volunteer Volunteer, Receiver receiver) {
        this.Volunteer = Volunteer;
        this.receiver = receiver;
    }

    // method Volunteer class to take input of volunteer
    public void takeInput(Vector<volunteer> vol, ArrayList nameArray) {
        Volunteer.takeInput(vol, nameArray);
    }

    // method to call displayVolunteer of Volunteer class
    public void displayVolunteerMethod(Vector<volunteer> vol) {
        Volunteer.displayVolunteer(vol);
    }

    // method to call deliverItem of Volunteer class
    public void deliverItemMethod(Vector<volunteer> vol, Vector<Receiver> acc, people people, restaurant rest,
            String type,
            int value) {
        Volunteer.deliverItem(vol, acc, people, rest, type, value);
    }

    // method to call input of receiver class
```

2.1.4 Polymorphism

The toString() is implemented in the organizer class and is a polymorphism

```
    case 2: {
        if (don.size() == 0) {
            System.out.println(x: "NO Donor found\n");
        } else {
            System.out.println(x: "All Donor Information is as follows:");

            // printing the information of donor type==people
            for (int i = 0; i < pp.size(); i++) {
                System.out.println(pp.get(i).toString());          //POLYMORPHISM
            }
            // printing the information of donor type==restaurant
            for (int f = 0; f < rr.size(); f++) {
                System.out.println(rr.get(f).toString());          //POLYMORPHISM
            }

        }
        break;
    }

    case 3: {
        Organizer.takeInput(vol, nameArray);
        break;
    }
    case 4: {
```

2.1.5 Inheritance

People class extends donor , people is a subclass of the superclass donor .

```java
// people class extends donor class
class people extends donor {      //INHERITANCE
    private double amount;

    people() {
        amount = 0.0;
    }

    public people(String name, String address, String type, double amount) {
        super(name, address, type);
        this.amount = amount;
    }

    public people(ArrayList<String> donorNameArray, Vector<donor> don, Vector<people> pp) {
        Scanner inp = new Scanner(System.in);
        amount = 0.0;
        type = "People";// inherited from donor class

        // System.out.println("people");
        boolean isMatch = donorInput(donorNameArray); // function of superclass donor;

        if (!isMatch) {
            System.out.print(s: "Enter Amount of money: ");
            amount = inp.nextDouble();

            // storing the information of people donor in a vector type donor
```

Restaurant class extends donor , restaurant is a subclass of the superclass donor .

```java
// restaurant class extend donor
class restaurant extends donor {  //INHERITANCE
    private String foodType;

    restaurant() {
        foodType = "";
    }

    public restaurant(String name, String address, String type, String foodType) {
        super(name, address, type);
        this.foodType = foodType;
    }

    public restaurant(ArrayList donorNameArray, Vector<donor> don, Vector<restaurant> rr) {
        Scanner inp = new Scanner(System.in);
        foodType = "";

        System.out.println(x: "Restaurant");

        type = "Restaurant"; // inherited from donor class
```

2.2 UML Diagram

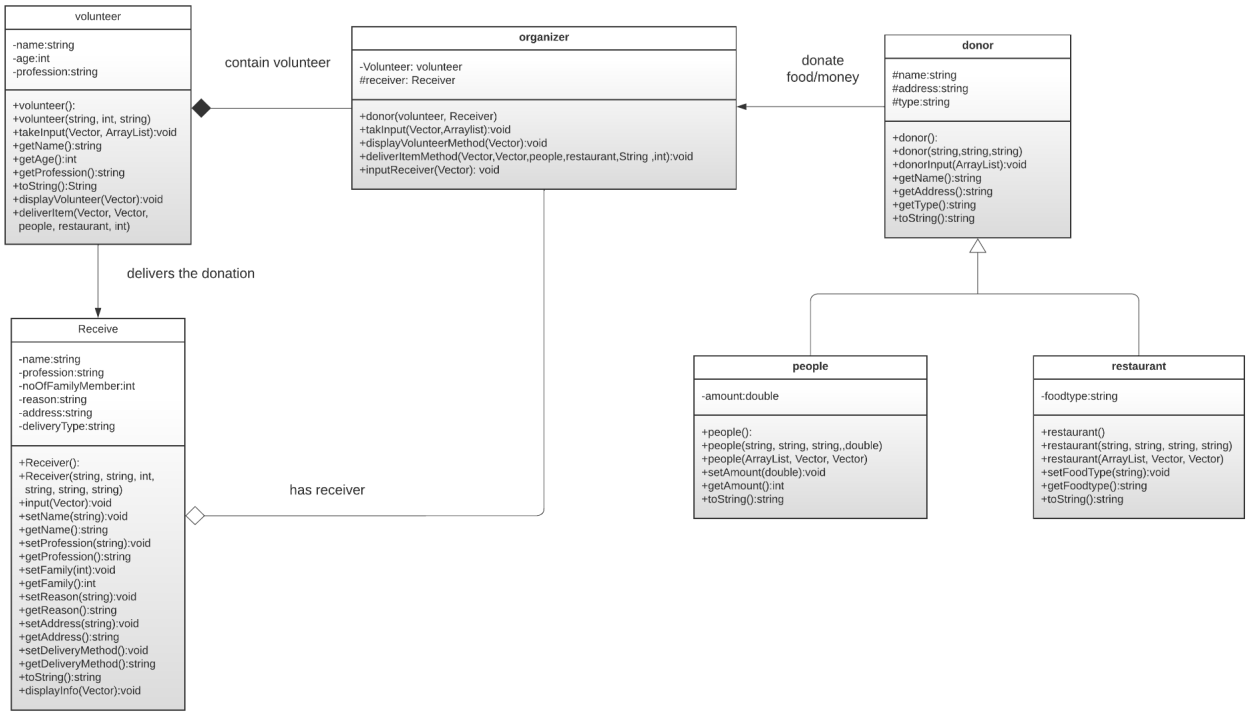
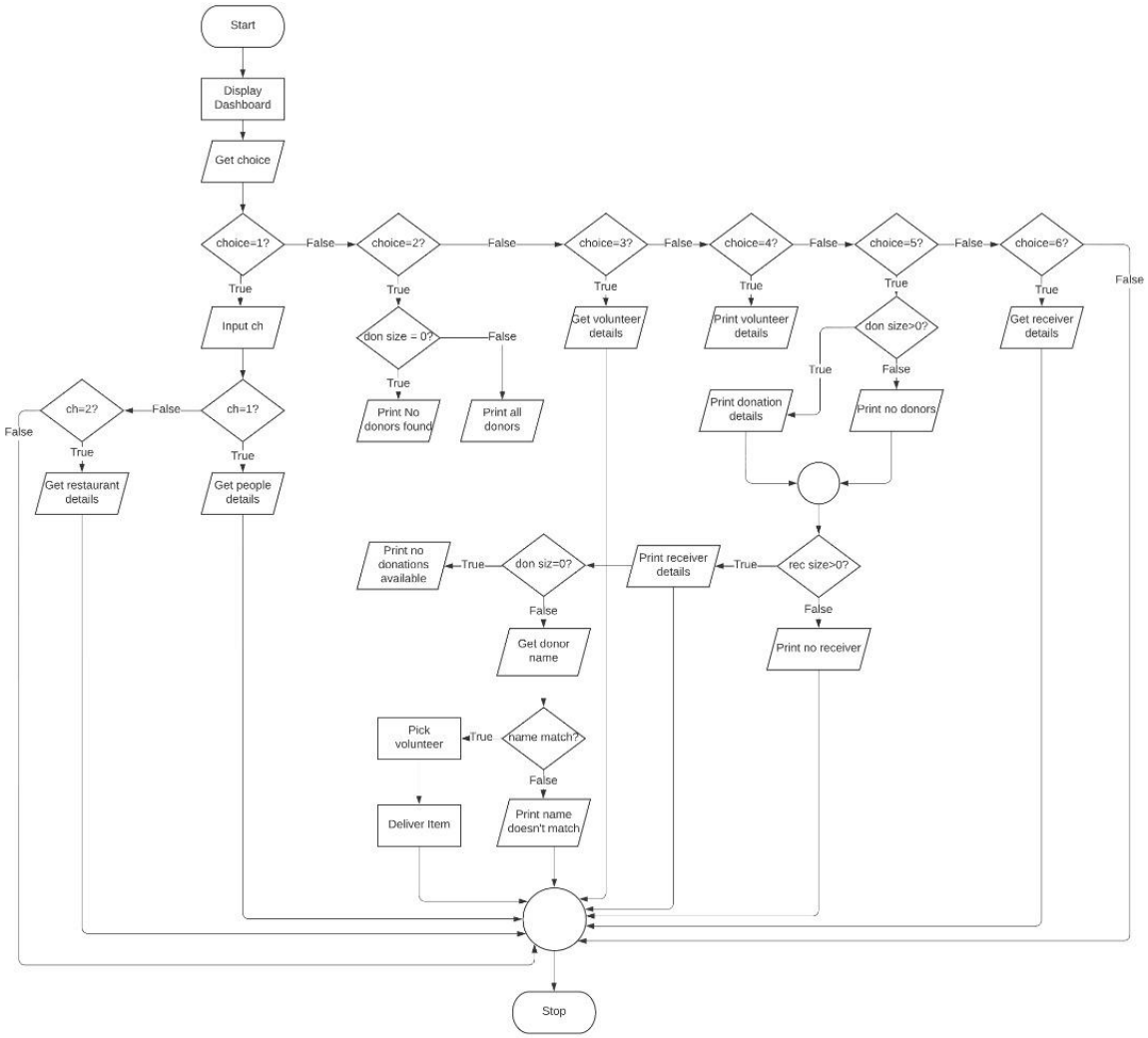


Fig: UML Class Diagram

2.3 Flowchart



2.4 Classes

1.organizer

Attributes: Volunteer:volunteer, Receiver: receiver

Methods: donor(volunteer, Receiver), takeInput(vector,ArrayList), displayVolunteerMethod(vector),deliverItemMethod(vector,vector,people,restaurant,string,int), inputReceiver(vector)

2.volunteer

Attributes: name, age, profession

Methods: volunteer(), volunteer(string,string,string), takeInput(vector,ArrayList), getName(), getAge(), getProfession(), toString(), displayVolunteer(vector), deliverItem(vector,vector, people, restaurant, int)

3.receiver

Attributes: name, profession, noOfFamilyMember, reason,address , deliveryType

Methods: Receiver(), Receiver(string,string,int,string,string,string), input(vector), getName(), getProfession(),getFamily(), getReason(), getAddress(), getDeliveryMethod(),setName(string),setProfession(string),setFamily(int),setReason(string), setAddress(string),set DeliveeryType(),toString(),displayInfo(vector)

4.donor
Attributes: name, address, deliveryType, type
Methods: donor(), donor(string,string,string), donarInput(ArrayList), getName(), getType(), getAddress(), getDeliveryType(), toString()

5.people
Attributes*:*amount
Methods:people(), people(string,string,string,double),people(ArrayList,Vector, Vector), setAmount(double),getAmount(),toString()

6.restaurant
Attributes:foodType
Methods: restaurant(), restaurant(string,string,string,string), restaurant(ArrayList,vector,vector), setFoodType(), getFoodType(), toString()

3.0 Project Design

For the project design, we used many java concepts like array, arrayList, vector , scanner etc.

Our project's main class, the organizer, contains the public main method. This class will be used to execute our code. In order to conduct all of the functions, the objects of the volunteer, receiver, and donor classes are declared inside the organizer class.

For the class volunteer, we have included a method previously mentioned in the UML diagram that takes input( ), here we use the Scanner class to get input from the user. Through the scanner object, the user inputs the volunteer's name, age, and profession. We use vectors to store the inputs taken from the user. The attributes of the volunteer are first stored in one vector and then an ArrayList is created to store the vectors in it. Other than that, another ArrayList is formed where only the volunteer's name is stored. When the menu is displayed and the volunteer information is asked, the user can type in the information. When all the typed volunteer information is asked to print, the system prints the typed information. For the donor class, we use the method donorInput() to take input from the user. The same concept is used in the donor class as well. Inputs are stored in the scanner object and then the donor information is saved in a vector and then stored as a group of vector objects in ArrayList.The donor name is separately stored in an ArrayList. When the menu is displayed and the donor information is asked, the user can type in the information. When all the typed donor information is asked to print, the system prints the typed information. The donorInput()method is called in 2 of donors inherited class People and Restaurant.

For class people and restaurants, inheritance is used to form a relationship between the donor

class and these 2 classes. Where the donor is the parent class and people and restaurants are the children's classes. People and restaurants specify which group of donors are they i.e they are either people or food given from restaurants.

To take input from the user, we use the Scanner class inside the input( ) method for the receiver class. It will also assign the input to the class variable. Furthermore, the getter method is used to return the value allocated to the instance variable. Because only one user can apply for a donation at a time, we didn't use an ArrayList or vector. We then use the display( ) method to show the user's information as well as the delivery type

4.0 Project Outcome

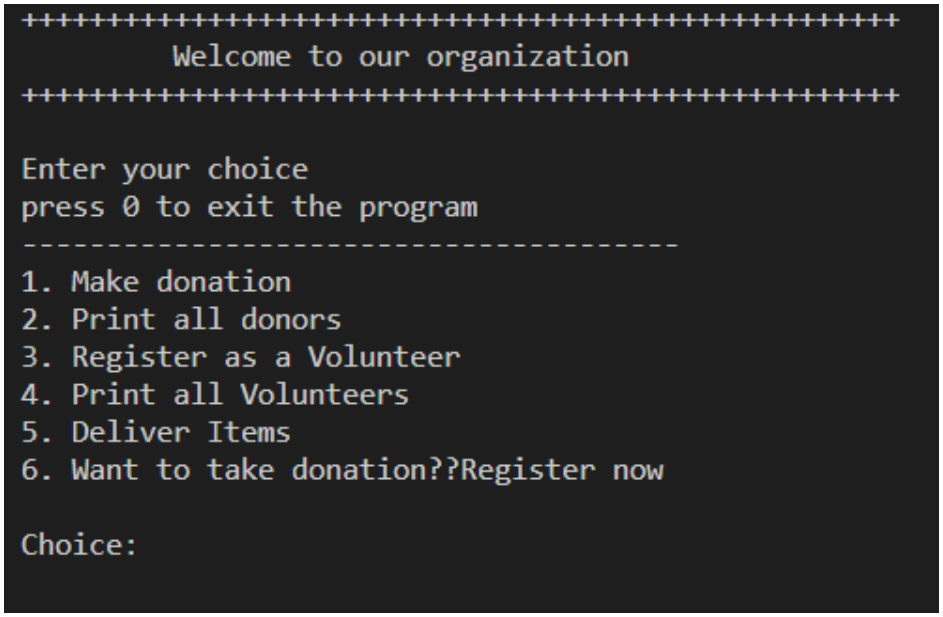


Fig: Menu of the code

At first, we will run the organizer class, then a menu will be displayed as above figure. When the option to make a donation is selected by typing 1 as the user's choice, the interface shows if the donation has to be made by people or from a restaurant. If people are selected, the interface will ask the people for their name, address, and amount of money which will then be saved in the system. If the option restaurant is rather selected, the same information is asked from them.

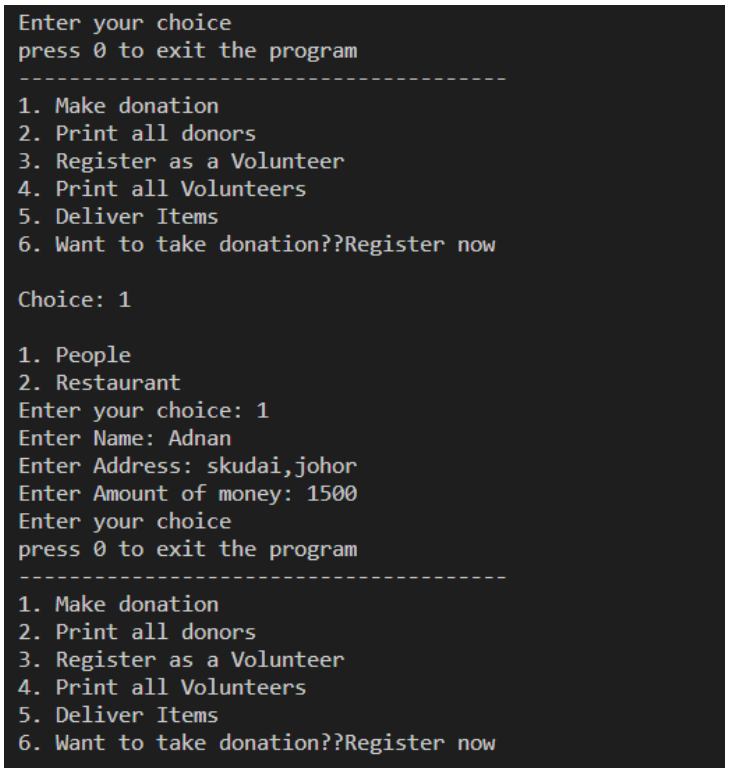


Fig: Choice 1- Make Donation - Choice 1 -people selected

When choice 2 from the main menu is selected all the donor information is shown in the display

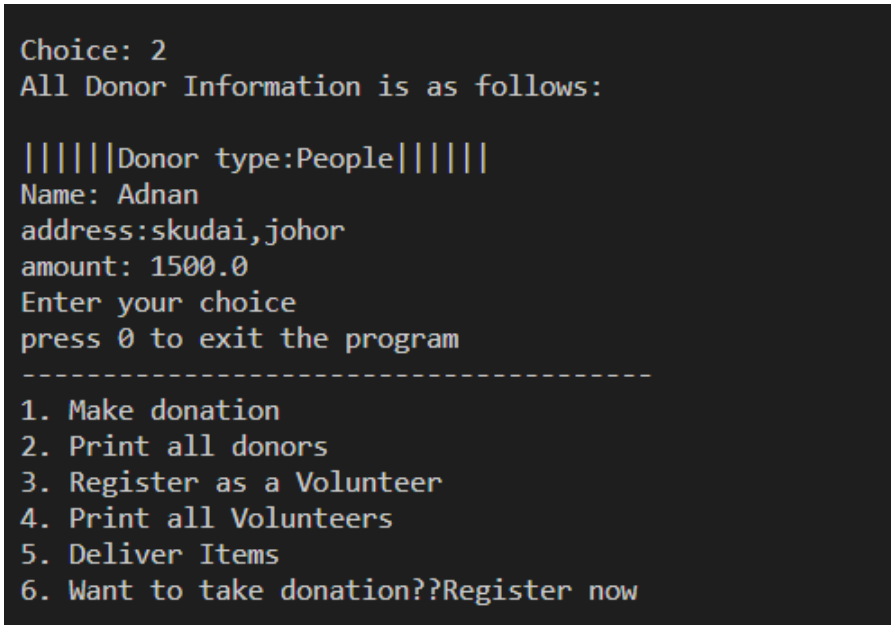


Fig: Choice 2: Print all donors

When choice 3 is selected, the system lets the Volunteer be registered as a member of the organization.

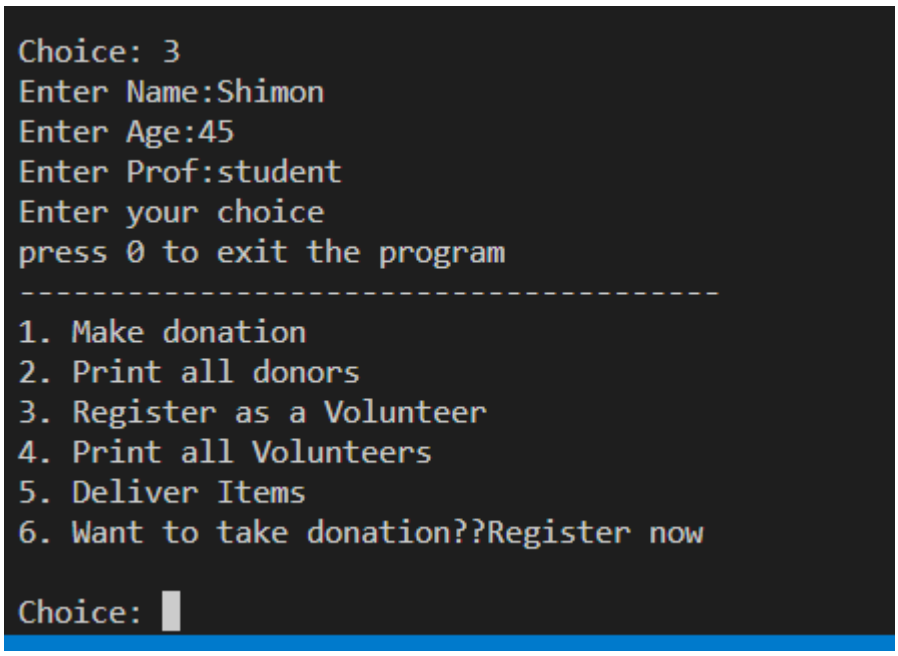


Fig: Choice 3: Register as Volunteer

When choice 4 is selected all the volunteer information is shown in the display

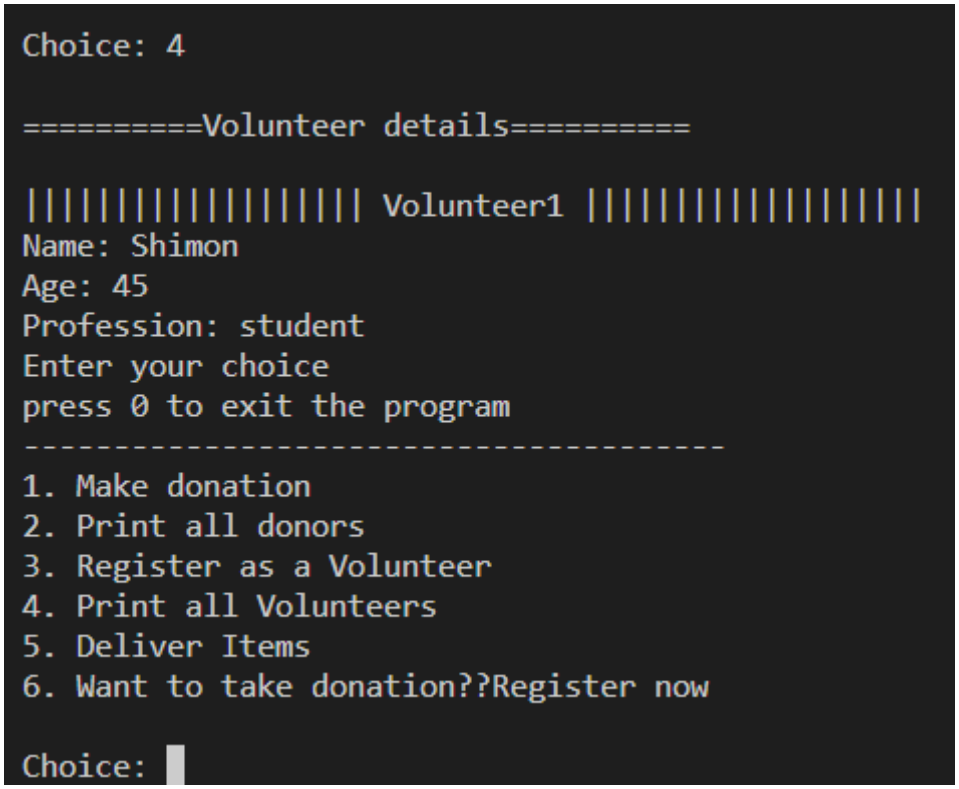


Fig: Choice 4: Print all volunteers

Option 5 will be activated once all the field has a person to make the delivery possible. Option 1 is needed for having donated goods. When using select option 4 a volunteer will be assigned to deliver the donation to the receiver. Option 6 is for receivers, those who want to take donations from the organization. Once it is selected, the user will be asked to provide the name, profession, number of families, the reason why the user is requesting for donation, and finally the address of the user. Afterward, the system will display the delivery option, according to user choice the delivery option will be assigned.

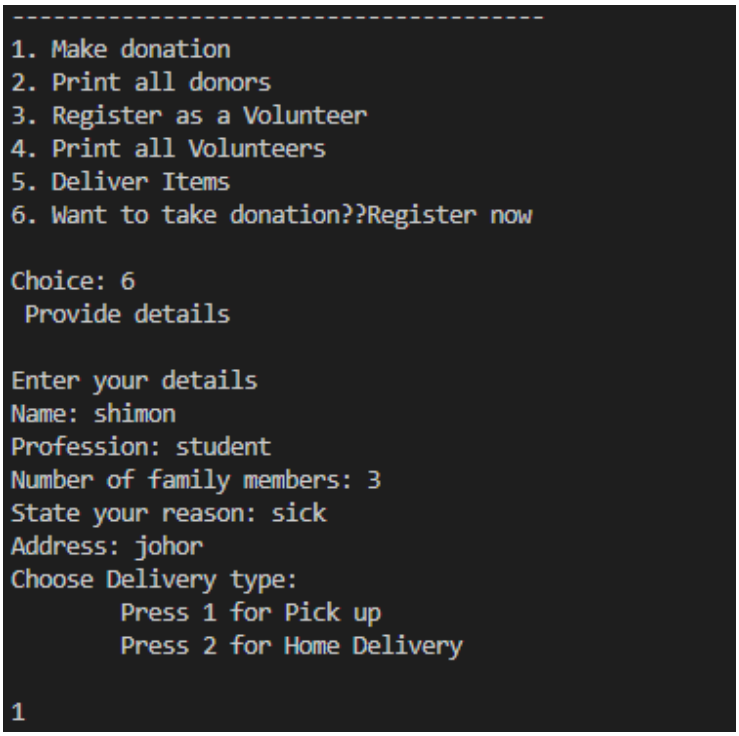


Fig: Choice 6: Receiver asks for donation

When option 5 will be selected from the main menu. Two more options will be displayed, “review receiver” and “donate”. Admin can review the receiver and accept or reject their application for the donation. Based on the admin choice the receiver will be added to a new vector called acc type Receiver or the receiver will be deleted. When the “donate option” is selected, the system will check the donor and accepted receiver list. If both are the empty system will display a print message. If there are donors and receivers, the system will ask to write the name of the donor whose donation the admin wants to donate. The system will now check the delivery method a receiver wants. If it is the “Home delivery” method, a volunteer will be assigned to deliver the donation to the receiver. If the input name matches the donor name in the list, the donation will be performed otherwise it will print “Donor name does not match”.

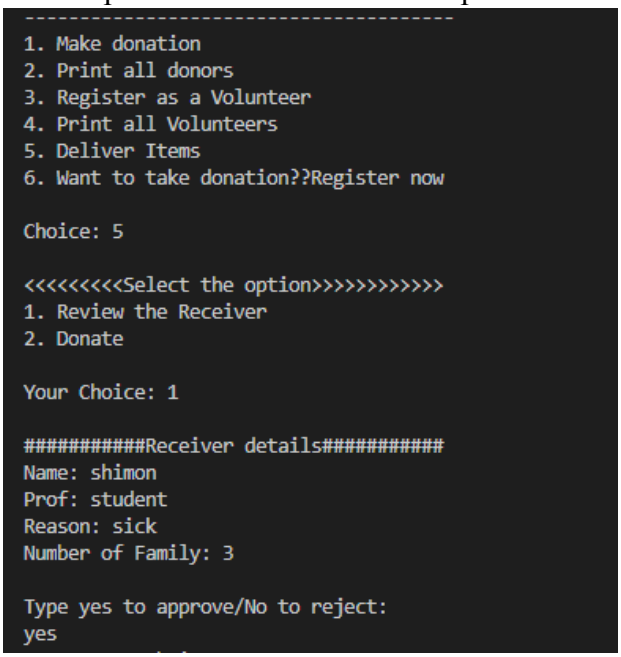


Fig: Choice 5:Admin approves the receiver

```
-------------------------------------
1. Make donation
2. Print all donors
3. Register as a Volunteer
4. Print all Volunteers
5. Deliver Items
6. Want to take donation??Register now

Choice: 5

<<<<<<<<<Select the option>>>>>>>>>>>>
1. Review the Receiver
2. Donate

Your Choice: 1

###########Receiver details###########
Name: shawon
Prof: student
Reason: sick
Number of Family: 3

Type yes to approve/No to reject:
No
The receiver has been rejected
```

Fig: Choice 5:Admin rejects the receiver

When the "Donate" option is selected, the system will display the donor list and accepted receiver.

```
-------------------------------------
1. Make donation
2. Print all donors
3. Register as a Volunteer
4. Print all Volunteers
5. Deliver Items
6. Want to take donation??Register now

Choice: 5

<<<<<<<<<Select the option>>>>>>>>>>>>
1. Review the Receiver
2. Donate

Your Choice: 2
++++++++++++++++++++List of Donation++++++++++++++++++++
NO.  Name              Address        Type             Donation
1    toya              johor          People           400.0(money)
2    hehe              jani na        People           500.0(money)
++++++++++++++++++++List of Accepted Receiver++++++++++++++++++++
NO.  Name              Address        No. of Family    Reason        Delivery Method
1    shimon            johor               3           sick            Pick up
```

Fig: Choice 5: Option Donate is selected

```
Type donors name to donate the receiver: hehe
shimon has got donation from hehe. Amount of money 500.0
```

Fig: Choice 5: Donor name matches, donation performed

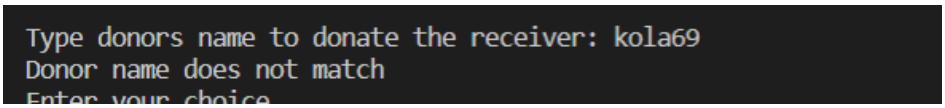


Fig: Choice 5: Donor name does not match

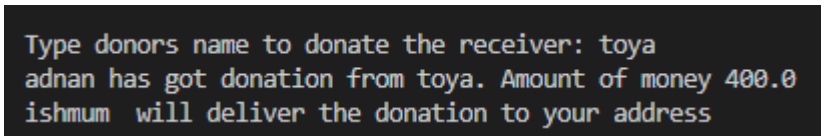


Fig: Choice 5: volunteer assigned

5.0 Conclusion

This system provides the main features that a food bank charitable organization needs. This is a computerized system that is utilized by the employees, donors, and potential volunteers to operate the organization. In this system there are various tasks such as making donations, printing all donors, registering as a volunteer, printing all-volunteer, delivering items, and receiving donations. The main inputs of the system are the information of the donors, receivers, and volunteers for their registration for their respective tasks in the system. The outputs are basically the donors giving a donation to the receiver and volunteers eventually making a delivery to the receiver carrying the donation. All these functions will help the food bank work in a more efficient way. This will benefit both the donor and the staff who manages the organization.

There are many aspects of the donation system that can be improved by our system. Firstly, the system is fully computerized so it saves a lot of time taking notes on the list of donors and receivers. This system saves the organization from errors that are made due to having a manual system, for example losing files that have donors and receivers listed. Moreover, in this system, the donor after donating and money, and the receiver after receiving the money are then removed from the queue so it prevents the same receiver from receiving the donation twice as the manual system might forget to cross out the receiver who already received the donation.

6.0 Task Distribution

Category	Task and their contributer
Coding	Class Donor -Toya Class Receiver-Tahzeeb Class People- Adnan Class Restaurant - Gm shaheen Class Volunteer - Toya Class Organizer - Gm Shaheen
Report	Introduction -Toya UML Class Diagram -Tahzeeb Project Planning - All Project Outcome - Tahzeeb & Gm Shaheen Project Design - Adnan Conclusion- Adnan & Toya
Presentation	Slides - Adnan Flowchart- Tahzeeb Explanation - All

7.0 Group Presentation Youtube Video Link:

Link: https://youtu.be/bXO4RI1lRtk

.