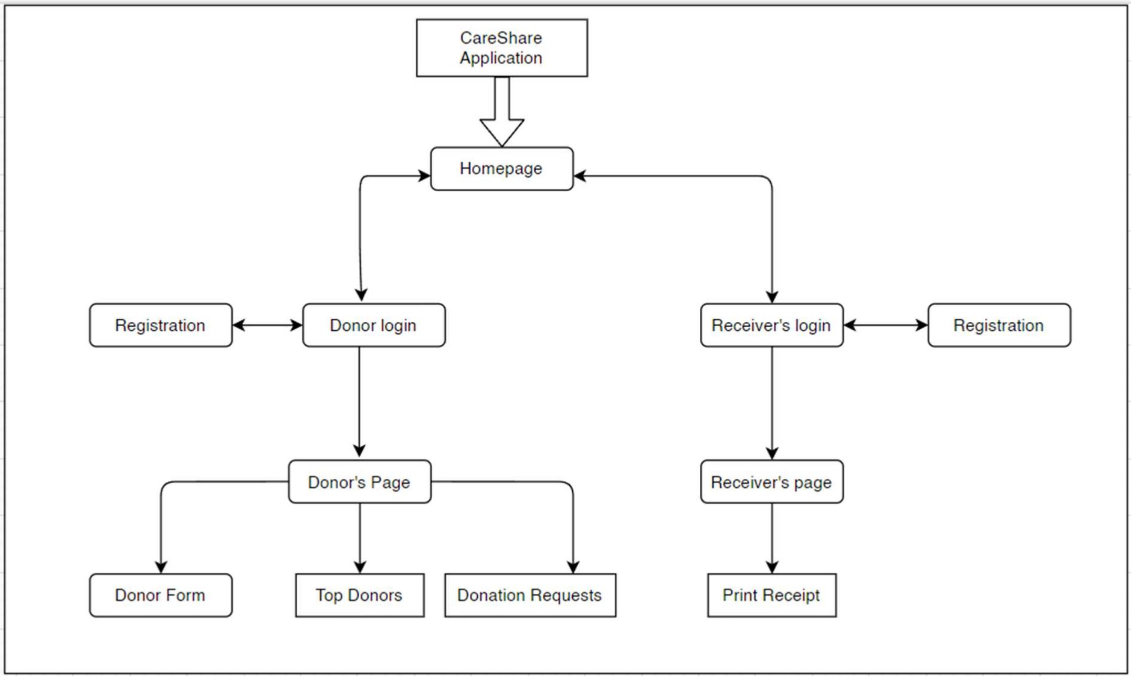
Criterion B: Design and Implementation

Basic Hierarchical Structure/Overview of GUI



List of Classes

The following briefly outlines function of the classes in this application:

HomePage: allows the end-user to navigate to either the donor or the receiver’s page**.**

LoginPage: allows donors to login to the system with their account. Also allows the user to navigate to registration page.

Registration: allows donors to register new accounts

DonorsPage: allows the user to navigate to the donation form, view the donation requests, and the top donor.

DonationForm: This is the donation form page. It allows the user to donate.

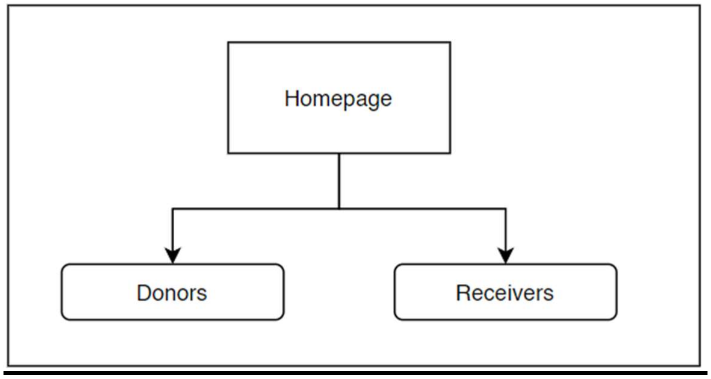
My_CNX: This class establishes the connection to the database.

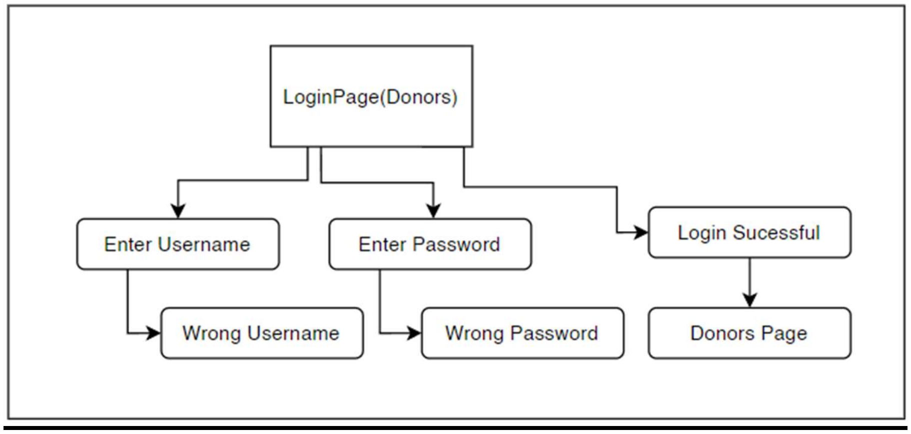
ReceiverHomepage: Allows receivers to register an account

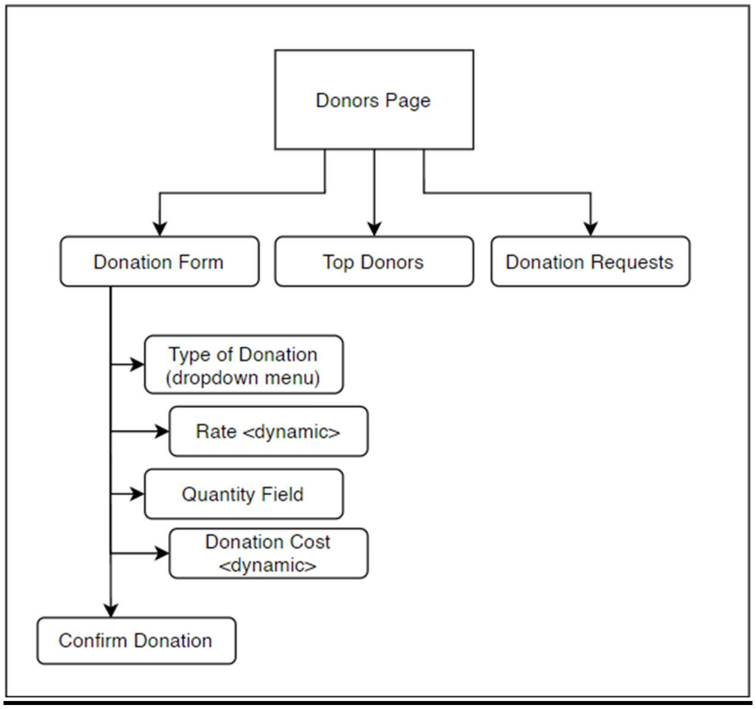
ReceiverLogin: This allows the user to login to their account.

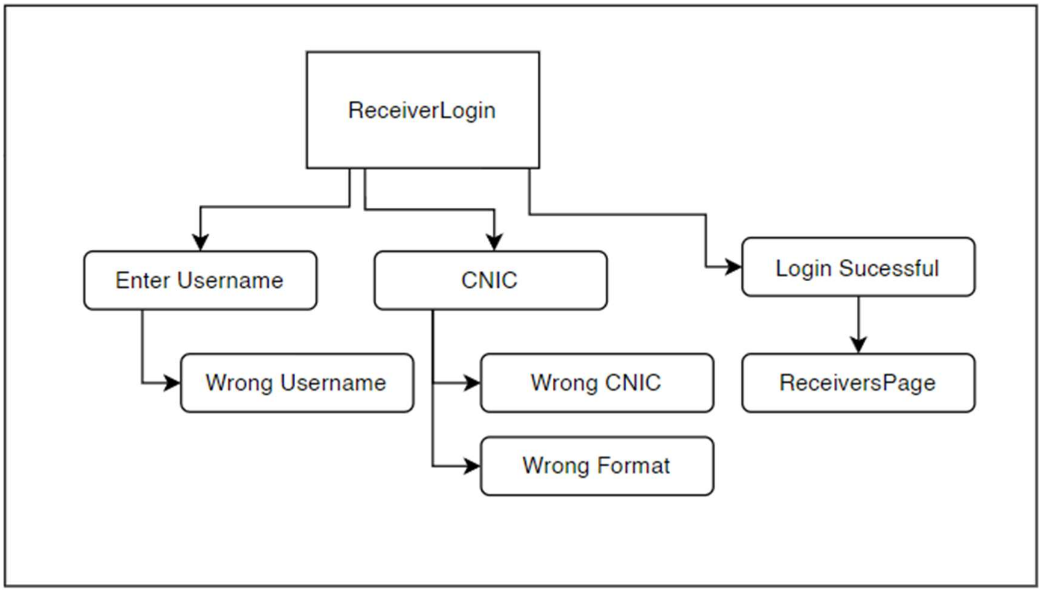
ReceiverPage: The page where receivers can add donations to their cart and confirm these donations as well as print a receipt of the donations that they receive.

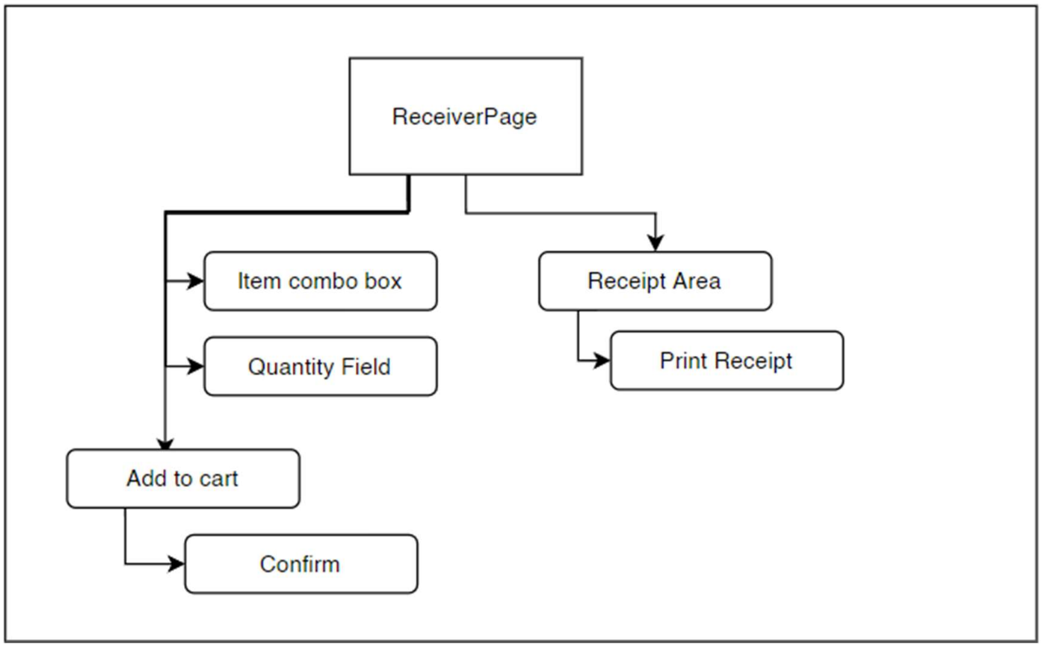
Hierarchical Input Process Output models



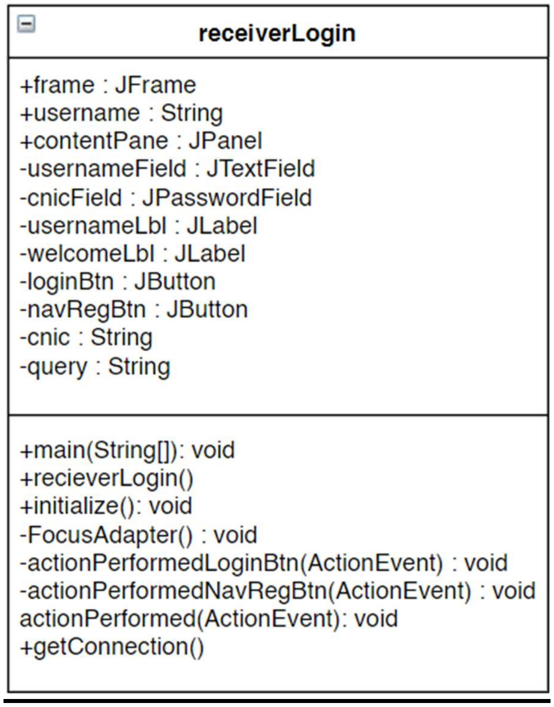





ReceiverLogin
Enter Username
CNIC
Login Sucessful
Wrong Username
Wrong CNIC
ReceiversPage
Wrong Format


ReceiverPage
Item combo box
Receipt Area
Quantity Field
Print Receipt
Add to cart
Confirm

Class (Unified Model Language) Diagrams



receiverLogin

+frame : JFrame

+username : String

+contentPane : JPanel

-usernameField : JTextField

-cnicField : JPasswordField

-usernameLbl : JLabel

-welcomeLbl : JLabel

-loginBtn : JButton

-navRegBtn : JButton

-cnic : String

-query : String

+main(String[]): void

+recieverLogin()

+initialize(): void

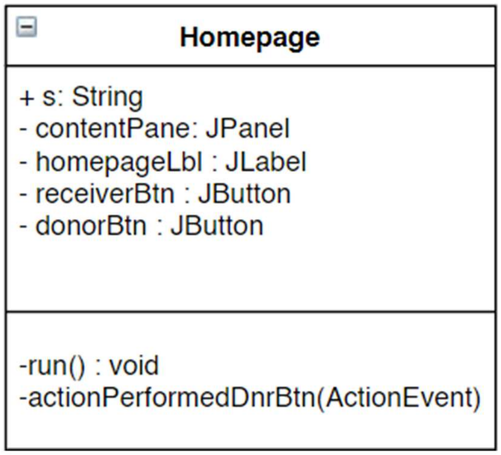
-FocusAdapter() : void

-actionPerformedLoginBtn(ActionEvent) : void

-actionPerformedNavRegBtn(ActionEvent) : void

actionPerformed(ActionEvent): void

+getConnection()



Homepage

+ s: String

- contentPane: JPanel

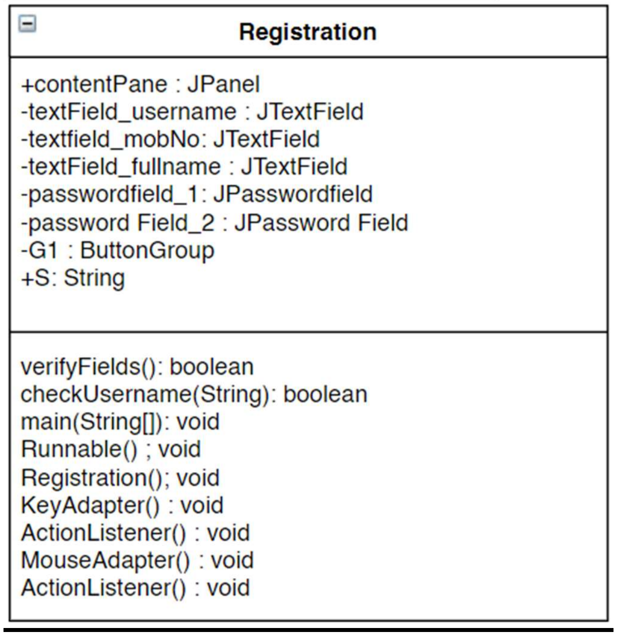
- homepageLbl : JLabel

- receiverBtn : JButton

- donorBtn : JButton

-run() : void

-actionPerformedDnrBtn(ActionEvent)



Registration

+contentPane : JPanel

-textField_username : JTextField

-textfield_mobNo: JTextField

-textField_fullname : JTextField

-passwordfield_1: JPasswordfield

-password Field_2 : JPassword Field

-G1 : ButtonGroup

+S: String

verifyFields(): boolean

checkUsername(String): boolean

main(String[]): void

Runnable() ; void

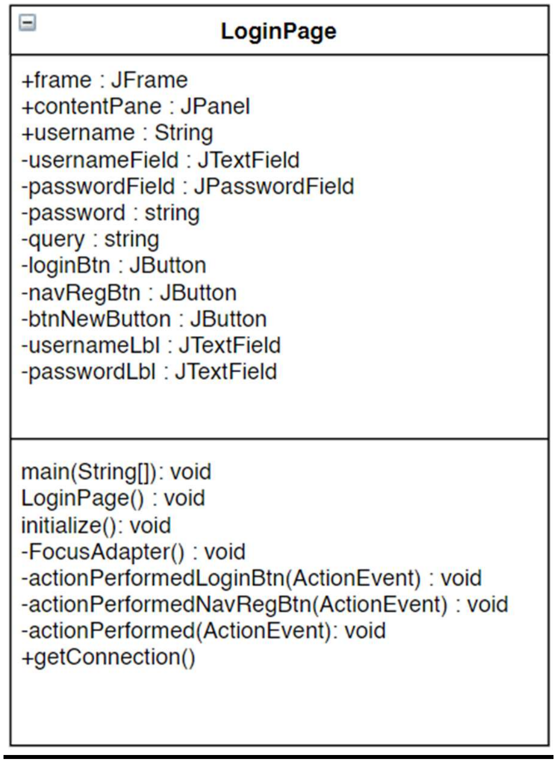
Registration(); void

KeyAdapter() : void

ActionListener() : void

MouseAdapter() : void

ActionListener() : void



LoginPage

+frame : JFrame

+contentPane : JPanel

+username : String

-usernameField : JTextField

-passwordField : JPasswordField

-password : string

-query : string

-loginBtn : JButton

-navRegBtn : JButton

-btnNewButton : JButton

-usernameLbl : JTextField

-passwordLbl : JTextField

main(String[]): void

LoginPage() : void

initialize(): void

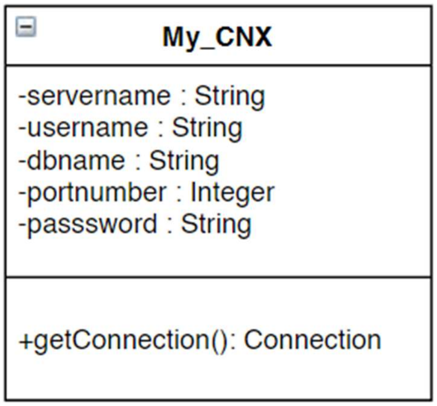
-FocusAdapter() : void

-actionPerformedLoginBtn(ActionEvent) : void

-actionPerformedNavRegBtn(ActionEvent) : void

-actionPerformed(ActionEvent): void

+getConnection()



My_CNX

-servername : String

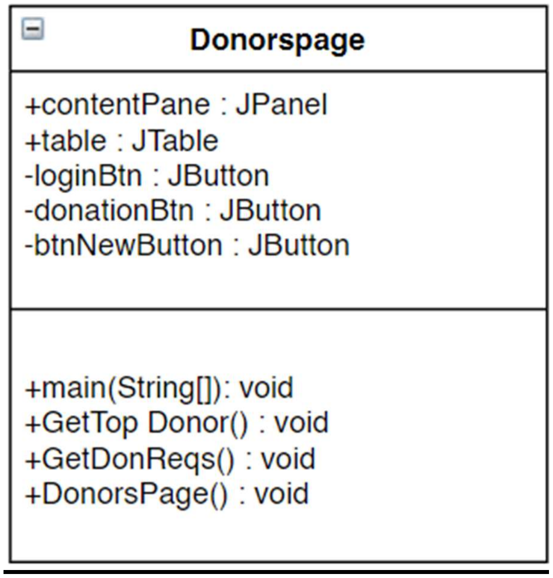
-username : String

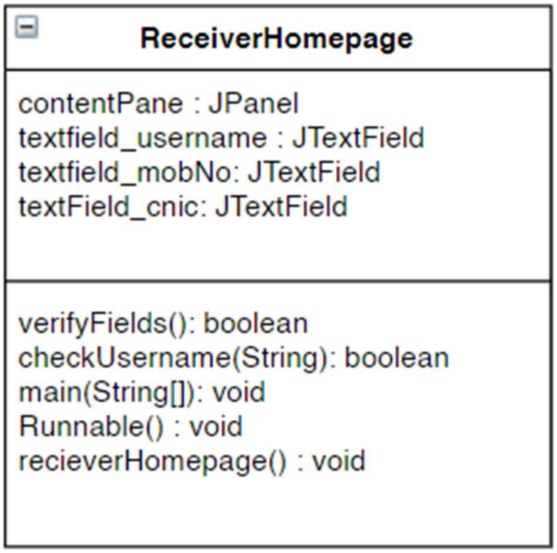
-dbname : String

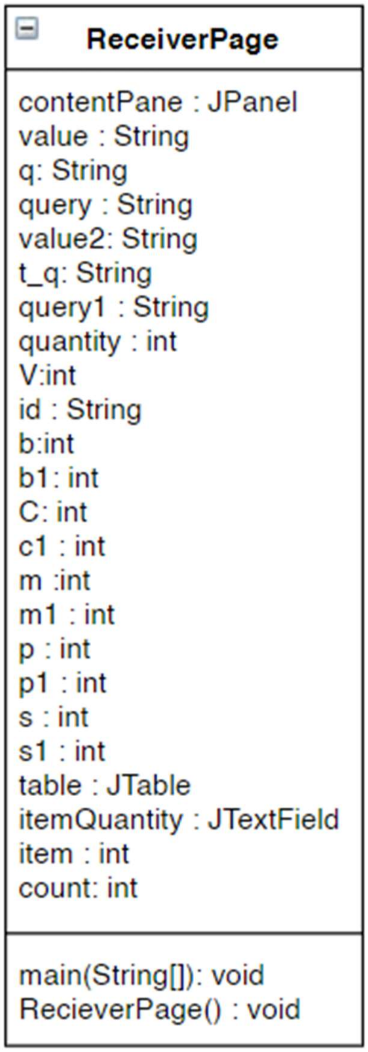
-portnumber : Integer

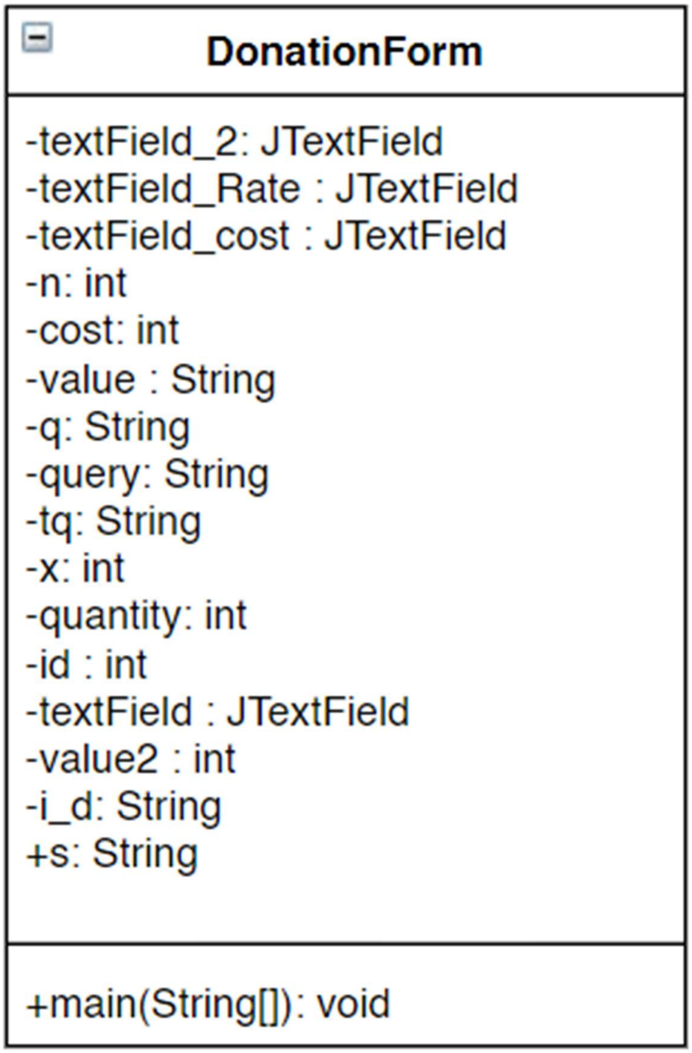
-passsword : String

+getConnection(): Connection


Donorspage
+contentPane : JPanel
+table : JTable
-loginBtn : JButton
-donationBtn : JButton
-btnNewButton : JButton
+main(String[]): void
+GetTop Donor() : void
+GetDonReqs() : void
+DonorsPage() : void

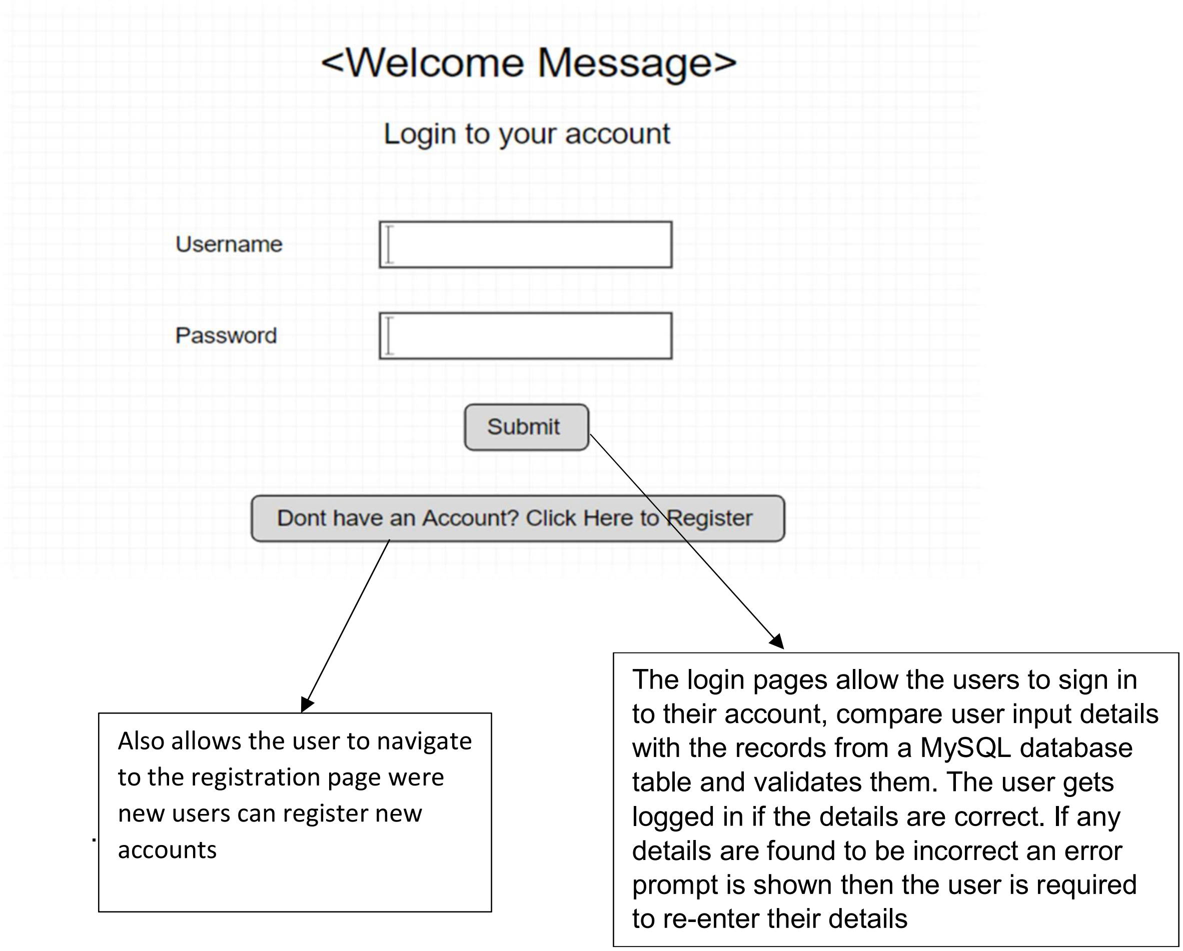

ReceiverHomepage
contentPane : JPanel
textfield_username : JTextField
textfield_mobNo: JTextField
textField_cnic: JTextField
verifyFields(): boolean
checkUsername(String): boolean
main(String[]): void
Runnable() : void
recieverHomepage() : void


ReceiverPage
contentPane : JPanel
value : String
q: String
query : String
value2: String
t_q: String
query1 : String
quantity : int
V:int
id : String
b:int
b1: int
C: int
c1 : int
m :int
m1 : int
p : int
p1 : int
s : int
s1 : int
table : JTable
itemQuantity : JTextField
item : int
count: int
main(String[]): void
RecieverPage() : void


DonationForm
-textField_2: JTextField
-textField_Rate : JTextField
-textField_cost : JTextField
-n: int
-cost: int
-value : String
-q: String
-query: String
-tq: String
-x: int
-quantity: int
-id : int
-textField : JTextField
-value2 : int
-i_d: String
+s: String
+main(String[]): void

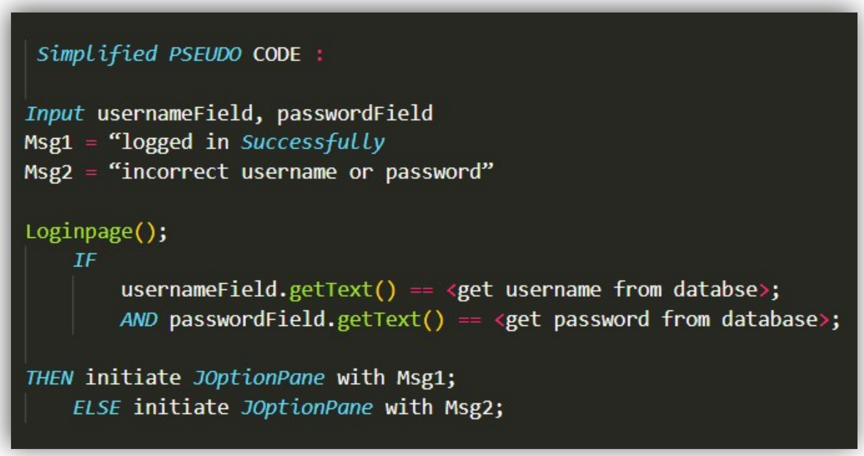
Login Page(s)

GUI design

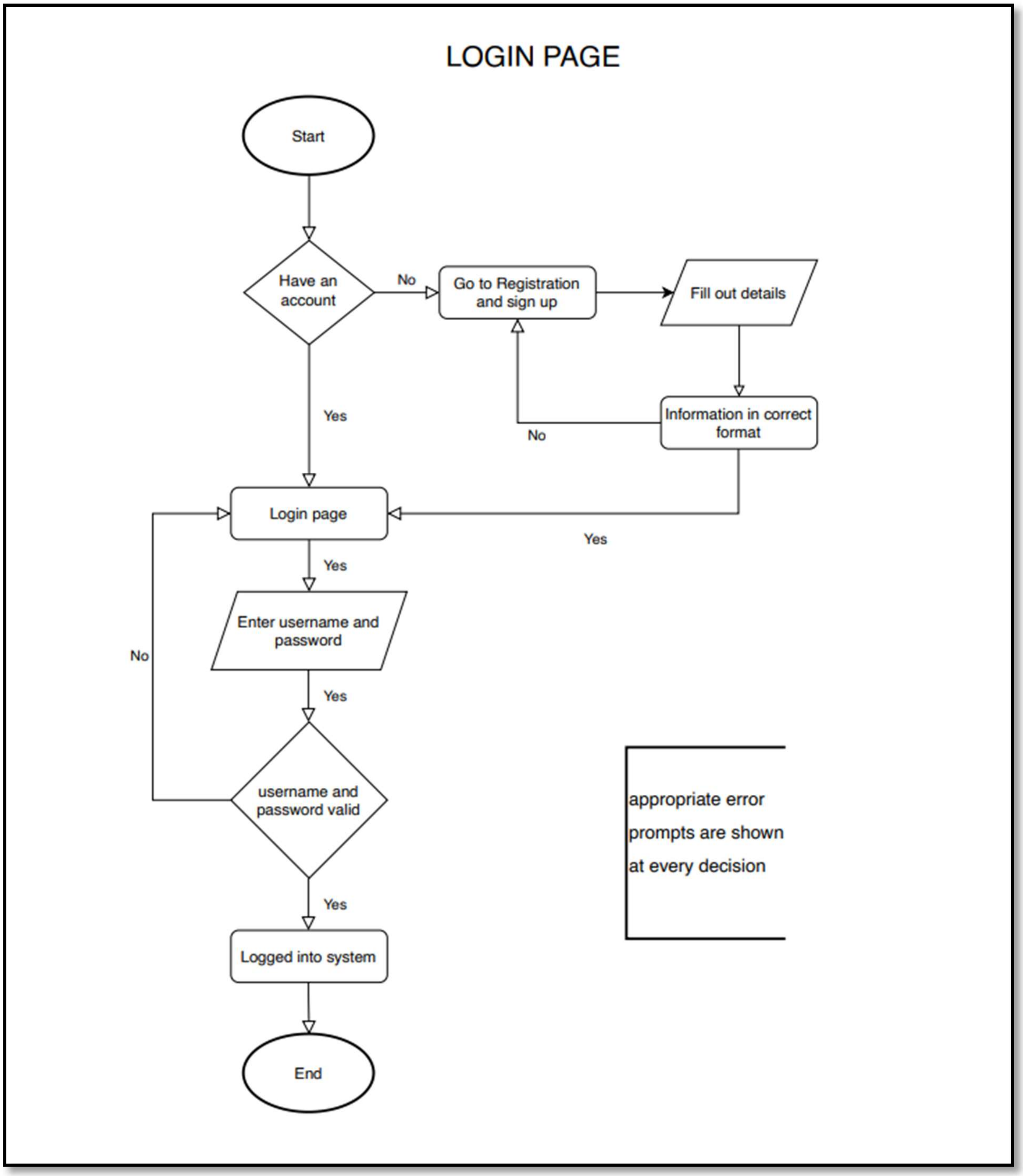


Also allows the user to navigate to the registration page were new users can register new accounts

The login pages allow the users to sign in to their account, compare user input details with the records from a MySQL database table and validates them. The user gets logged in if the details are correct. If any details are found to be incorrect an error prompt is shown then the user is required to re-enter their details



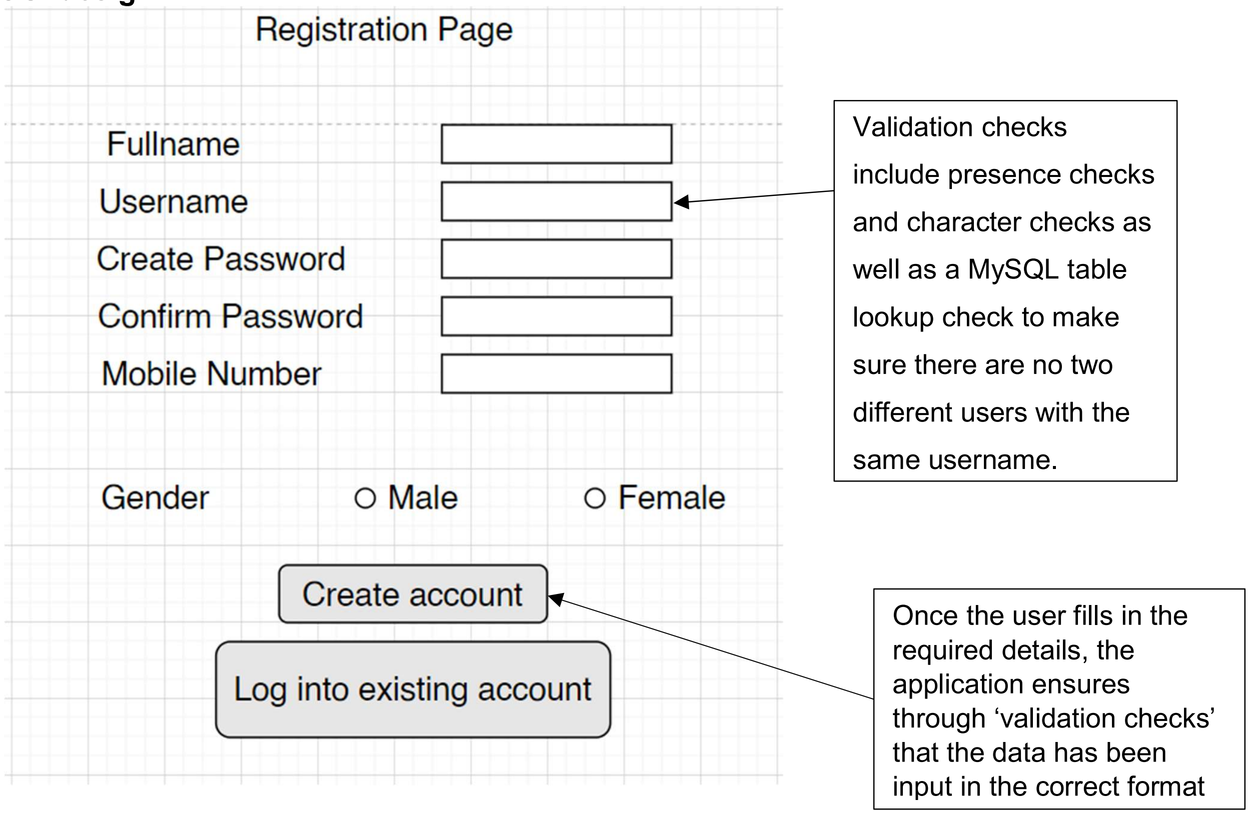
```
 Simplified PSEUDO CODE :

Input usernameField, passwordField
Msg1 = “logged in Successfully
Msg2 = “incorrect username or password”

Loginpage();
    IF
        usernameField.getText() == <get username from databse>;
        AND passwordField.getText() == <get password from database>;

THEN initiate JOptionPane with Msg1;
    ELSE initiate JOptionPane with Msg2;
```


LOGIN PAGE
Start
Have an account
No
Go to Registration and sign up
Fill out details
Information in correct format
No
Yes
Login page
Yes
Yes
Enter username and password
Yes
No
username and password valid
Yes
Logged into system
End
appropriate error prompts are shown at every decision

Flowchart showing the login process

Registration Page

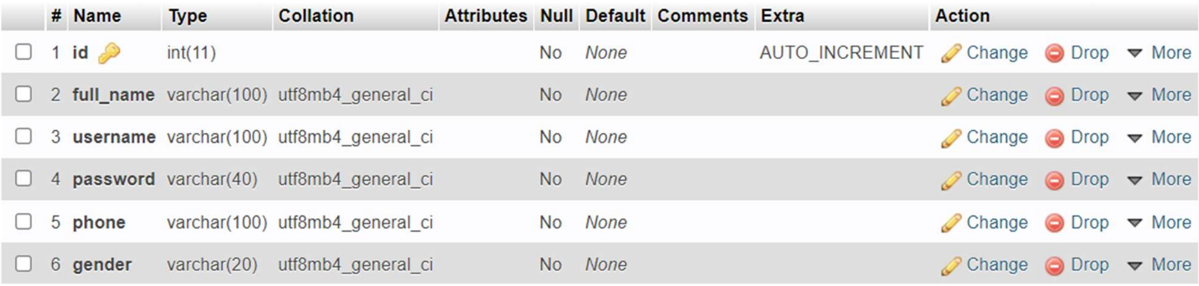
GUI design



Validation checks include presence checks and character checks as well as a MySQL table lookup check to make sure there are no two different users with the same username.

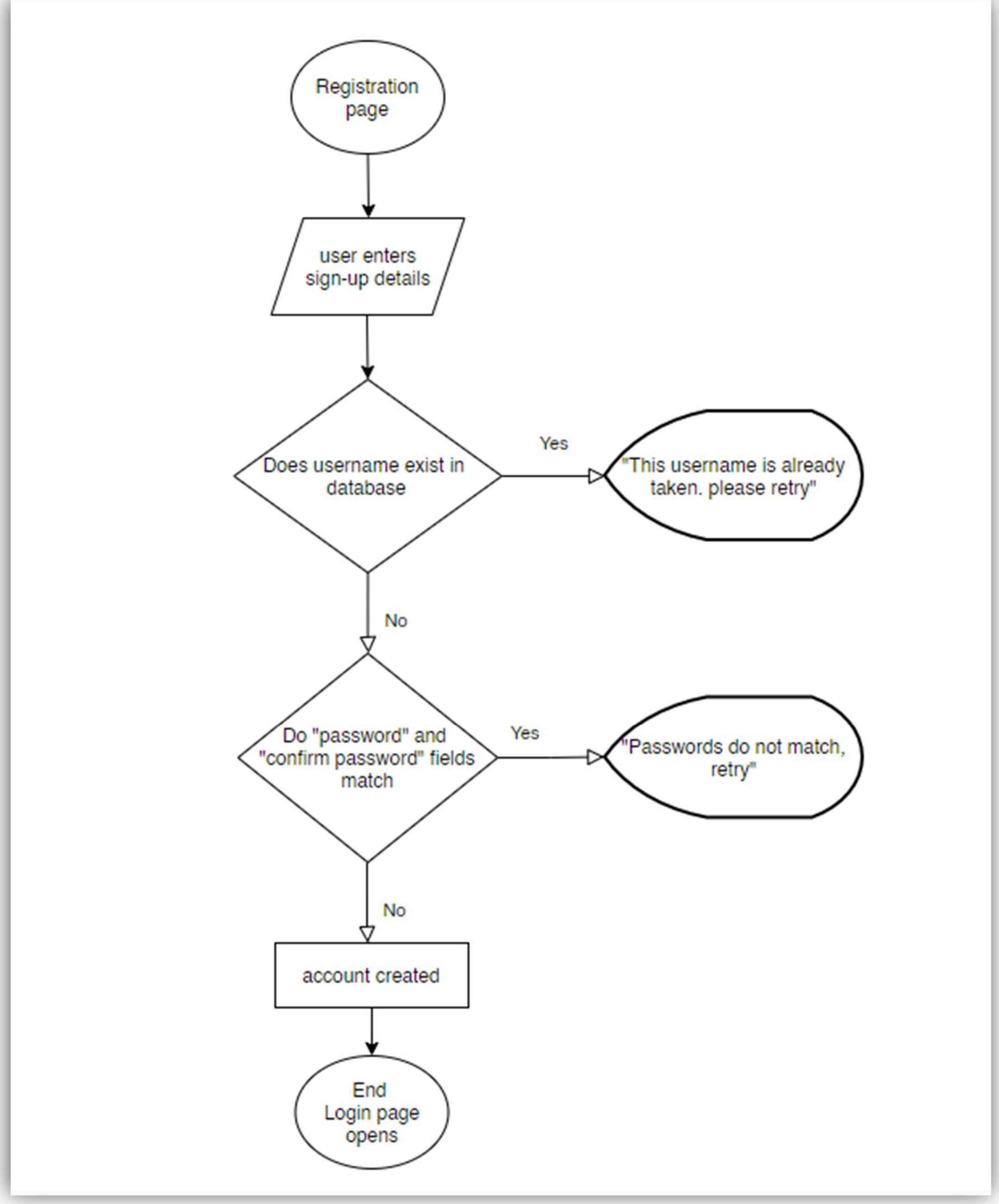
Once the user fills in the required details, the application ensures through 'validation checks' that the data has been input in the correct format

Structure of 'users' table (connected to registration and login pages)



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**id**	int(11)			No	*None*		AUTO_INCREMENT	Change	Drop	More
☐	2	**full_name**	varchar(100)	utf8mb4_general_ci		No	*None*			Change	Drop	More
☐	3	**username**	varchar(100)	utf8mb4_general_ci		No	*None*			Change	Drop	More
☐	4	**password**	varchar(40)	utf8mb4_general_ci		No	*None*			Change	Drop	More
☐	5	**phone**	varchar(100)	utf8mb4_general_ci		No	*None*			Change	Drop	More
☐	6	**gender**	varchar(20)	utf8mb4_general_ci		No	*None*			Change	Drop	More

Validation Method Flowchart

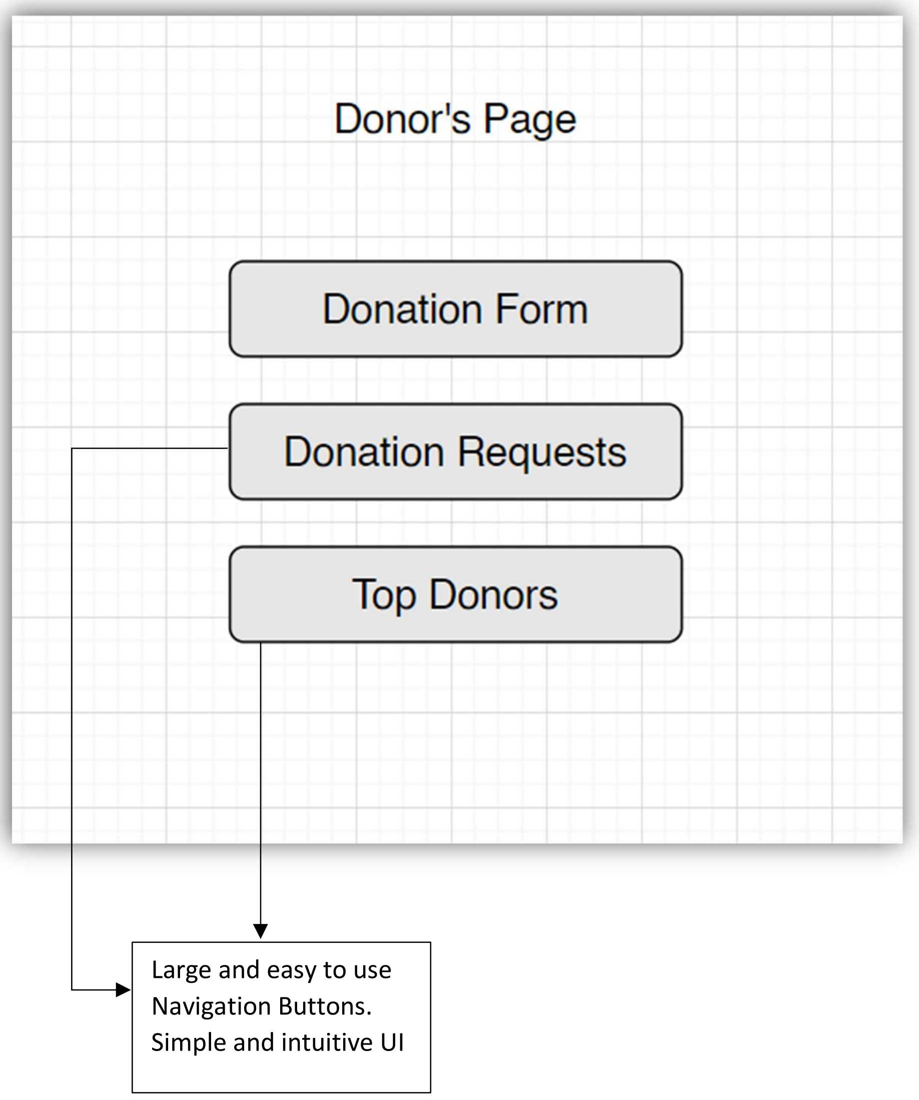

Registration page
user enters sign-up details
Does username exist in database
Yes
"This username is already taken. please retry"
No
Do "password" and "confirm password" fields match
Yes
"Passwords do not match, retry"
No
account created
End
Login page opens

Donor’s Page:

The donor’s page has three parts: -

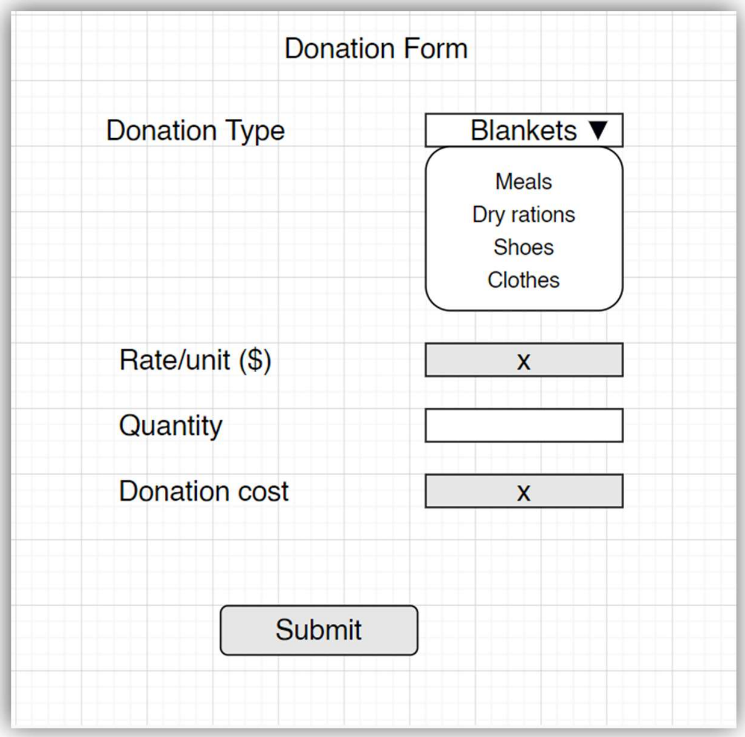
1. Donation Form
2. Donation Request
3. Top Donors

GUI Design



Donation Form

GUI Design



Drop Down Menu to select an option.

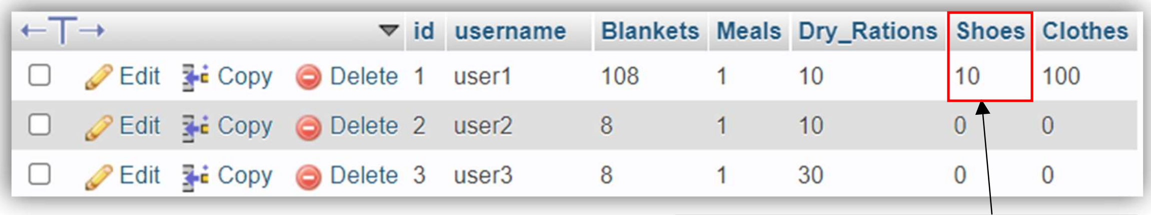
Based on the selection, a rate per unit will be shown. Different for every selection

A user than enters the quantity that they want to donate

Donation Cost is calculated by multiplying the quantity and the rate/unit fields.

Note: The donation is added into the account of the user that makes the donation. This allows the administrator to keep track of total number of donations made by a particular user.

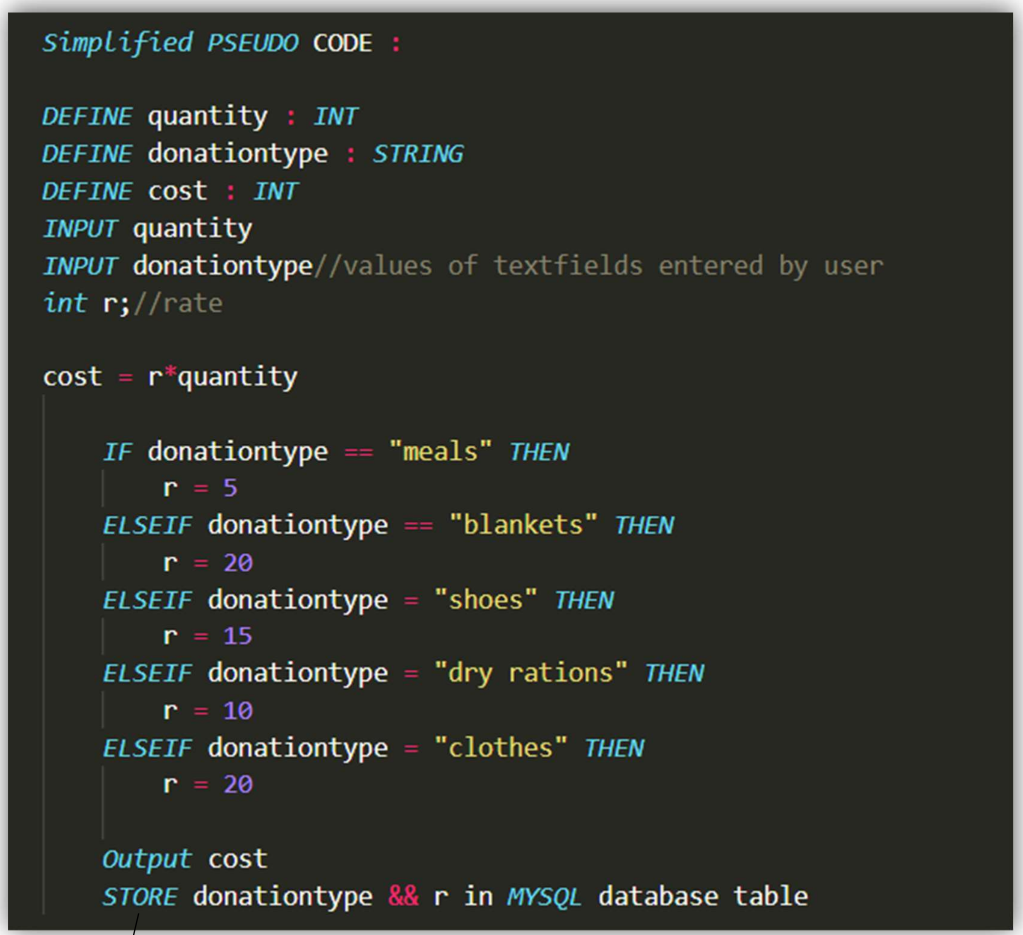
Structure of the 'donations' table:



←T→			▼	id	username	Blankets	Meals	Dry_Rations	Shoes	Clothes
□	Edit	Copy	Delete	1	user1	108	1	10	10	100
□	Edit	Copy	Delete	2	user2	8	1	10	0	0
□	Edit	Copy	Delete	3	user3	8	1	30	0	0

For example; If user 1 decides that he wants to donate 5 shoes, he will select "shoes" from the drop down enter "5" into the quantity field. When they press 'submit', the value of the cell corresponding to their name will be updated to reflect this change.

Note: The working of the accounts system will be explained in detail in Criterion C.

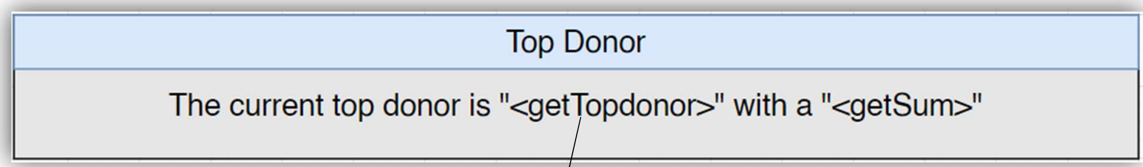


```
Simplified PSEUDO CODE :

DEFINE quantity : INT
DEFINE donationtype : STRING
DEFINE cost : INT
INPUT quantity
INPUT donationtype//values of textfields entered by user
int r;//rate

cost = r*quantity

    IF donationtype == "meals" THEN
        r = 5
    ELSEIF donationtype == "blankets" THEN
        r = 20
    ELSEIF donationtype = "shoes" THEN
        r = 15
    ELSEIF donationtype = "dry rations" THEN
        r = 10
    ELSEIF donationtype = "clothes" THEN
        r = 20

    Output cost
    STORE donationtype && r in MYSQL database table
```

The STORE function shows the insertion of donation details into the database table.

Top Donors Button

GUI Design

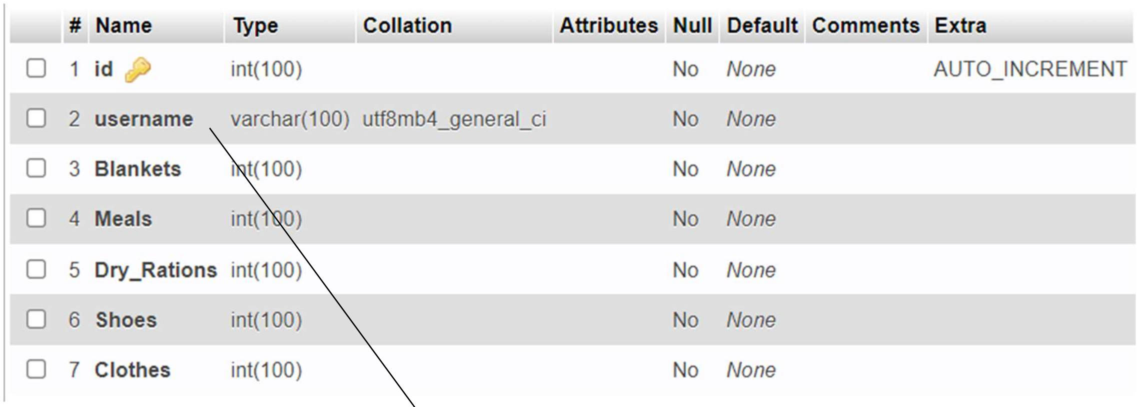


Top Donor

The current top donor is "<getTopdonor>" with a "<getSum>"

The “top donors” is a button, which upon pressing, pops up a dialog box that will tell the user the name of the donor who has made the largest amount of donations as well as the amount of donations made.

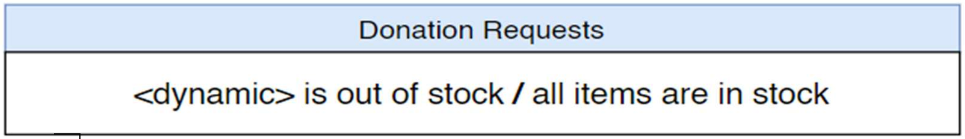
Structure of “donations” table



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
□	1	**id**	int(100)			No	*None*		AUTO_INCREMENT
□	2	**username**	varchar(100)	utf8mb4_general_ci		No	*None*		
□	3	**Blankets**	int(100)			No	*None*		
□	4	**Meals**	int(100)			No	*None*		
□	5	**Dry_Rations**	int(100)			No	*None*		
□	6	**Shoes**	int(100)			No	*None*		
□	7	**Clothes**	int(100)			No	*None*		

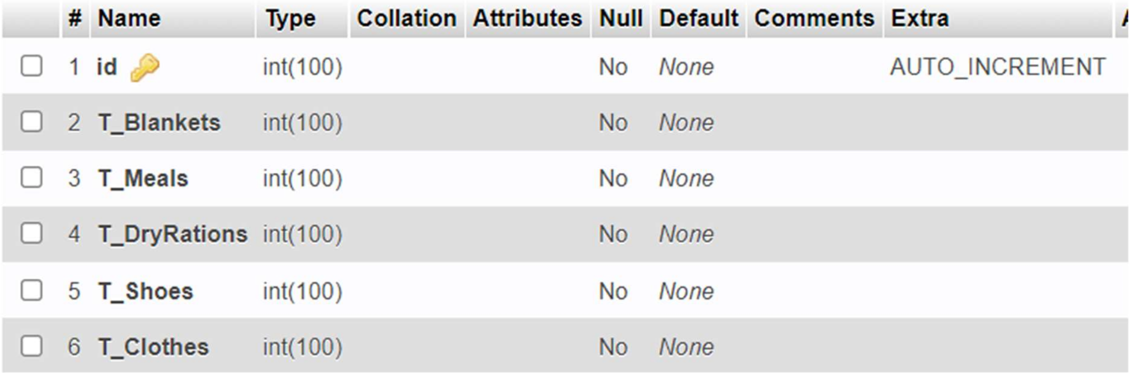
This is done by using two MySQL queries which sum the donations of each users and then compare them to find the donor with the largest sum of donations.

Donation Requests GUI

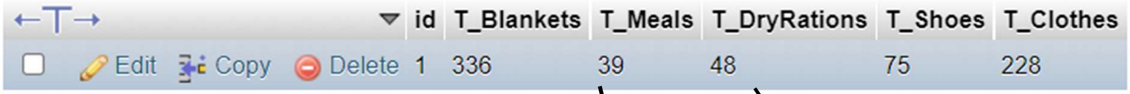


The “donation requests” is a button, which This will allow users to know what items are out of stock and therefore suggesting them to donate those items.

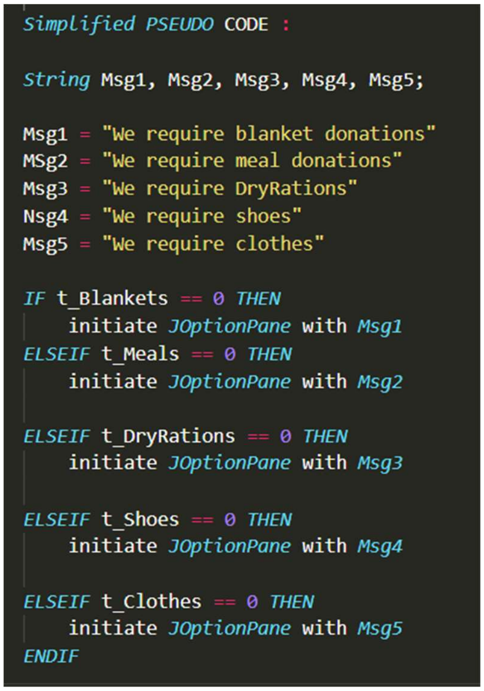
Structure of “totals” table



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
□	1	id 🔑	int(100)			No	*None*		AUTO_INCREMENT
□	2	T_Blankets	int(100)			No	*None*		
□	3	T_Meals	int(100)			No	*None*		
□	4	T_DryRations	int(100)			No	*None*		
□	5	T_Shoes	int(100)			No	*None*		
□	6	T_Clothes	int(100)			No	*None*		



←T→	▼	id	T_Blankets	T_Meals	T_DryRations	T_Shoes	T_Clothes
□	Edit Copy Delete	1	336	39	48	75	228



```
Simplified PSEUDO CODE :

String Msg1, Msg2, Msg3, Msg4, Msg5;

Msg1 = "We require blanket donations"
MSg2 = "We require meal donations"
Msg3 = "We require DryRations"
Nsg4 = "We require shoes"
Msg5 = "We require clothes"

IF t_Blankets == 0 THEN
    initiate JOptionPane with Msg1
ELSEIF t_Meals == 0 THEN
    initiate JOptionPane with Msg2

ELSEIF t_DryRations == 0 THEN
    initiate JOptionPane with Msg3

ELSEIF t_Shoes == 0 THEN
    initiate JOptionPane with Msg4

ELSEIF t_Clothes == 0 THEN
    initiate JOptionPane with Msg5
ENDIF
```

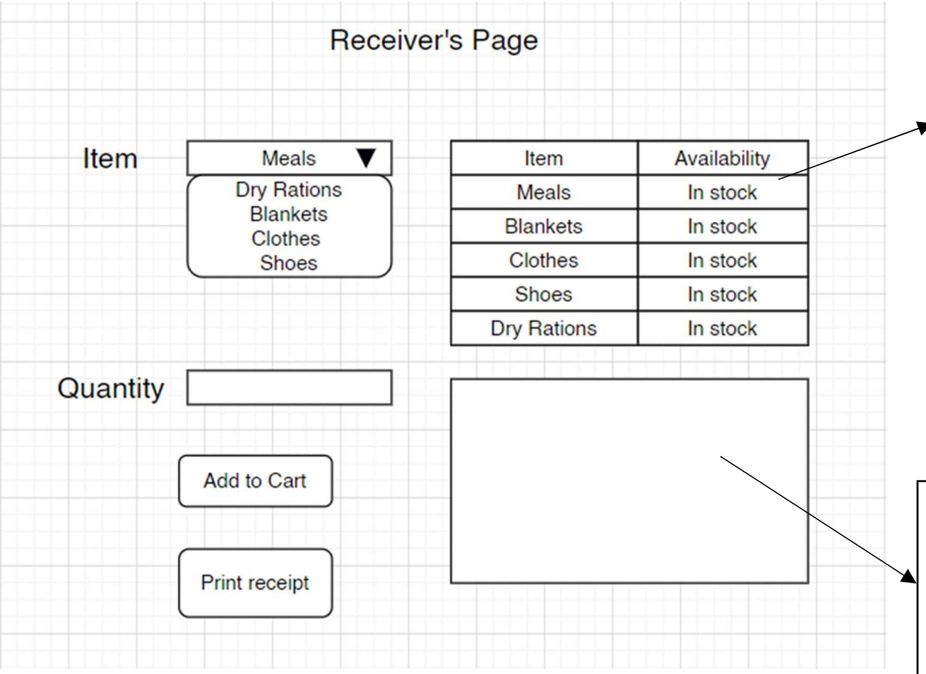
When the ‘donation requests’ button is pressed, it performs a lookup to see whether any of the items in the ‘totals’ table are out of stock i.e. value = 0

If an item is found to be out of stock, a dialogue box pops up informing this to the user

Totals calculated at run time and updated when change is made to “donations” table.

Receivers Page

GUI Design



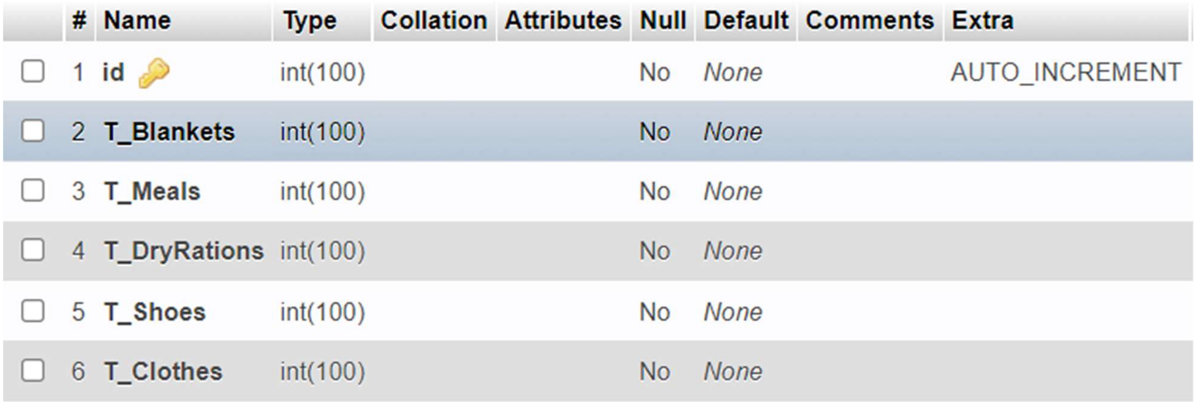
Item	Availability
Meals	In stock
Blankets	In stock
Clothes	In stock
Shoes	In stock
Dry Rations	In stock

Table performs a lookup using a MySQL query on the 'totals' table to find out whether anything is out of stock i.e. value = 0

If value for any of the items is found to be 0 then instead of in-stock the table will show "out of stock"

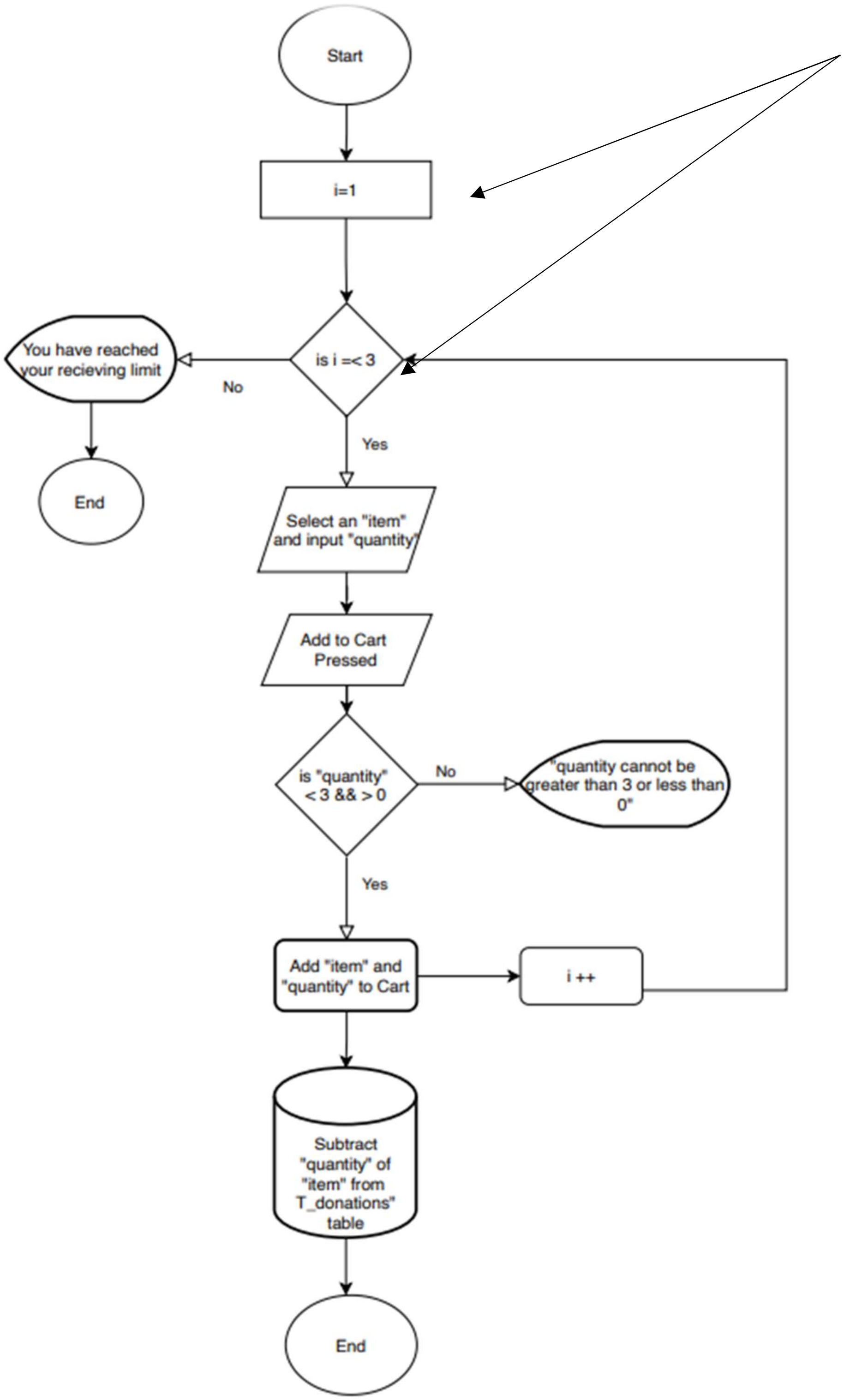
This is the user's cart. When the user adds an item to the cart, the cart gets updated to show the item and the quantity selected. A user can select a total of 3 items. i.e. 2 meals and a blanket

Linked "T_donations" Table Structure



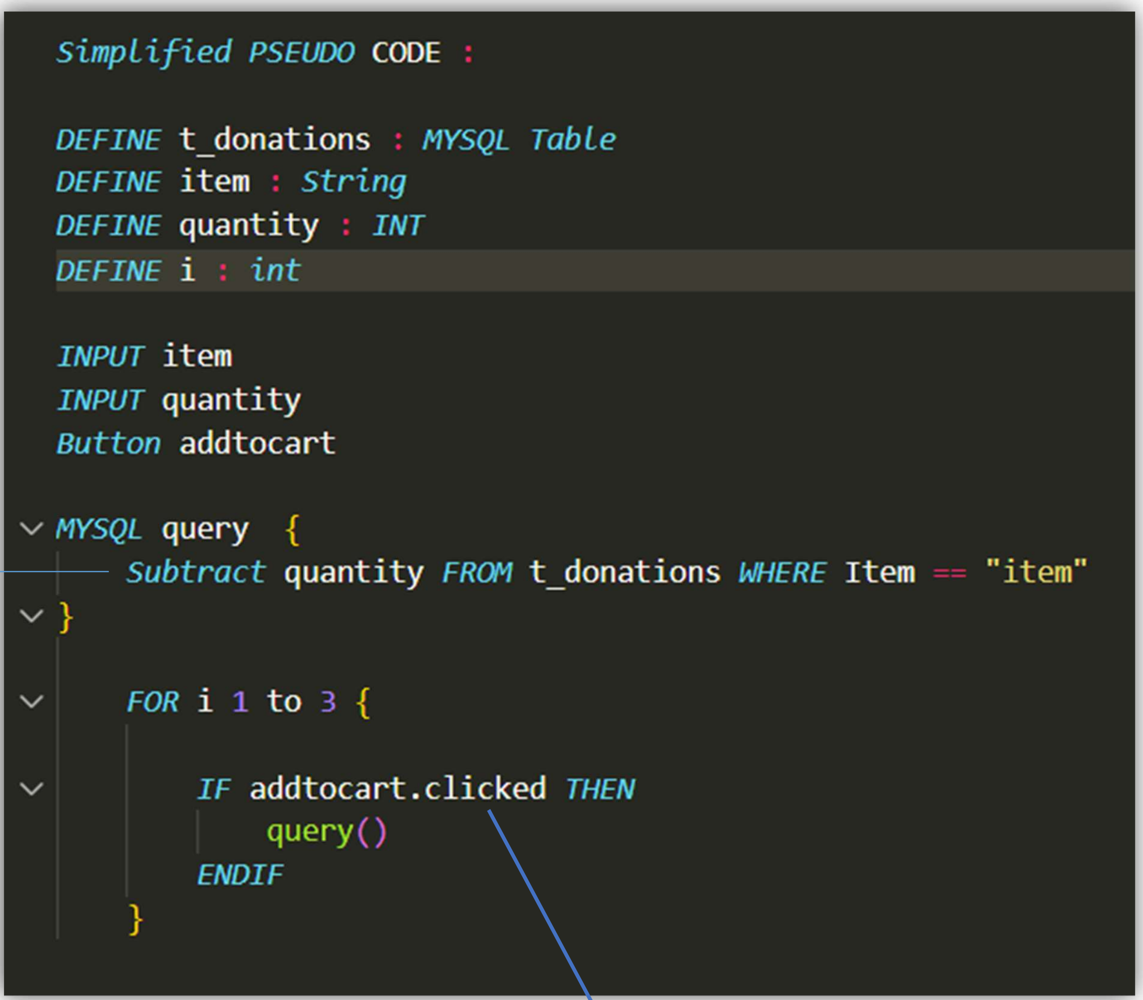
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**id**	int(100)			No	*None*		AUTO_INCREMENT
☐	2	**T_Blankets**	int(100)			No	*None*		
☐	3	**T_Meals**	int(100)			No	*None*		
☐	4	**T_DryRations**	int(100)			No	*None*		
☐	5	**T_Shoes**	int(100)			No	*None*		
☐	6	**T_Clothes**	int(100)			No	*None*		

Simplified Flowchart


Start
i=1
is i =< 3
No
You have reached your recieving limit
End
Yes
Select an "item" and input "quantity"
Add to Cart Pressed
is "quantity" < 3 && > 0
No
"quantity cannot be greater than 3 or less than 0"
Yes
Add "item" and "quantity" to Cart
i ++
Subtract "quantity" of "item" from T_donations" table
End

A loop is initialized here. This loop makes sure that users cannot add more than 3 items to their cart.

Throws an error prompt and disables input when a user tries to add more than 3 items



```
Simplified PSEUDO CODE :

DEFINE t_donations : MYSQL Table
DEFINE item : String
DEFINE quantity : INT
DEFINE i : int

INPUT item
INPUT quantity
Button addtocart

MYSQL query  {
    Subtract quantity FROM t_donations WHERE Item == "item"
}

    FOR i 1 to 3 {

        IF addtocart.clicked THEN
            query()
        ENDIF
    }
```

This query subtracts the 'item' that has been specified by the user, by the amount input "quantity", from the MySQL table.

addtocart.clicked means:

"when the add to cart button is clicked"

Test Plan

Success Criterion Tested	Input/Method	Expected Result
Visually Appealing *Criteria 1b*	'Client and users' comments on the general aesthetic of the application.	Elegant consistent themes and palette across the user interface.
User Friendliness *Criteria 1a*	Users were allowed to use the software and were then asked to fill out a questionnaire regarding the ease of use of the software.	Software is user friendly and most people who use the software don't encounter a problem while using it.
Accounts Log-in and registration with appropriate validation checks and error prompts *Criteria 2a*	Abnormal data checks to verify validation checks. Normal data checks to ensure working.	If data is not in correct format, appropriate error messages pop up. Users can sign into and out of their accounts.
Register new accounts both donors and receivers *Criteria 2a, 3a*	User inputs details. Abnormal data and normal data to test.	User can register a new account and details are saved in the users table.
Application should inform donors about the availability of items *Criteria 2d*	Upon pressing a specific button, a message showing he items currently out of stock pops up.	Items that are currently out of stock are shown in a pop-up panel.
Application should show the user who is currently the top donor (registered user with most donations) *Criteria 2c*	A button on the page should pop up a message.	A panel tells the reader which donor has the most donations and the number of donations the top donor has.
Each selected item on the donor's form has a different rate *Criteria 2b*	Each item is selected.	When an item is selected, the rate field is updated to show the rate of the item that has been selected.
A cost price is calculated by multiplying rate of selected item with the quantity input by the user *Criteria 2B*	The user inputs a quantity which is multiplied by the rate of the selected item.	Cost price is displayed correctly in the Cost price field.
Cost Price Field and Quantity field are un-editable *Criteria 2B*	The user attempts to edit the value of both fields.	The application does not allow users to change these fields as they are fixed by the administrator.

Receivers can only register if they have a unique CNIC (National Identification Card) in valid format (13 digits with dashes after 5th and 12th word) *Criteria 3a*	Duplicate data is entered. Abnormal data (wrong format) is entered.	Users can only login and register using a unique CNIC which is input in the correct format.
The application should show the receivers whether the items are in stock *Criteria 3a*	The users should be able to see which items are in stock. Users try to add items that are out of stock into their cart.	A table that shows availability of each type of item displays the availability of each item correctly. Users are unable to add items that are out of stock into their carts.
All donation inflow and outflow are logged correctly both at donors end and on receiver's end *Criteria 2e, 3d*	Receivers take items. Donors donate items.	Database tables update at runtime, reflecting changes to donation totals.
Receivers are able to print a receipt of their cart	Receivers attempt to print.	A pdf file is created showing the contents of the cart.