



# **Share Cloths**

## Comp204 Term Project

### Final Report

#### **Instructor**

Ahmet Soran  
Samet Tonyalı

#### **Group Members**

Tacettin Batuhan Bostancı  
Ayşe Şeyda Çalışkan  
Mustafa Demiröz  
Mehmet Anıl İrfanoğlu  
Dhiya Ulhaq

# TABLE OF CONTENTS

1) LIST OF FIGURES.....	3
2) ABSTRACT.....	4
3) REQUIREMENT ANALYSIS.....	4
4) SPECIFICATIONS.....	4
5) GENERAL DESCRIPTION.....	5
6) TOOLS/IDE.....	5
7) HIGH LEVEL DIAGRAM. ....	6
8) UML USE CASE DIAGRAM. ....	6
9) E-R DIAGRAM.....	7
10) DESIGN PHILOSOPHY.....	8
11) NORMALIZATION.....	10
12) ER TO RELATIONAL MAPPING.....	11
13) FUNCTIONAL DEPENDENCIES.....	11
14) DATABASE SCHEMA.....	23
15) VIEW.....	23
16) DATABASE APPLICATION.....	24
17) INTERFACE OF WEBSITE.....	25
18) SUMMARY.....	36

## LIST OF FIGURES

1) <i>Figure1.: Tools</i> .....	2
2) <i>Figure2.: UML Diagram</i> .....	2
3) <i>Figure2.: E-R Diagram</i> .....	3
4) <i>Figure2.: EER Diagram</i> .....	20
5) <i>Figure4: Login page</i> .....	26
6) <i>Figure5: Register page</i> .....	27
7) <i>Figure6: Information page</i> .....	28
8) <i>Figure7: Cloth donation page</i> .....	29
9) <i>Figure8: Cloth Information page</i> .....	29
10) <i>Figure9: Cloth entrance page</i> .....	30
11) <i>Figure10: Hangar page</i> .....	31
12) <i>Figure11: Delete Transporter page</i> .....	32
13) <i>Figure12: Create new transportation page</i> .....	33
14) <i>Figure13: Update Transportation page</i> .....	34
15) <i>Figure14: Create Transporter page</i> .....	35
16) <i>Figure15: Create Person In Need page</i> .....	36

## **Abstract**

This article contains means of implementation of a charity database system. Charities have significant obligation of credibility. The aim of this project is to make certain of the credibility of the help foundation by acquiring trust from the donators. The application offers for the donators to see status of their donation and to the charities to get the reputation of being trustworthy. The general mechanism of the project is outlined in the ER diagram, database design, system requirements, tools used, analysis and specifications in detail. Lastly, some screenshots from the application are given in the paper for better understanding of the project.

## **REQUIREMENT ANALYSIS**

The Share Clothes application have charity page that user will enter the needed information about the clothes that they gave.

The clothes will have different attributes such as cloth id, type, size, user id and hangar id. Till the cloth reaches the hangar, hangar id will be null.

Once user fill out the charity page program will create a transportation record and send a delivery personal to user's can bring those cloths to hangar by itself in that case there transporter id in the transportation record will be null.

The user is a person and in addition to user id, username and password it has person's attributes which are name, surname, sex and address id.

The user will be able to track down the status of the cloth in cloth status page.

People who are in need will have person id, upper size, lower size as body size and the number of clothes that he/she got aided.

The Gathering hangars that these cloths will be kept have three attributes hangar id, hangar name and address id.

In create transportation page cloths, transporter and people in need will be chosen and then transportation will be carried out. In first transportations are created as they have null arrival date, once the transportation completed arrival date is given to the transportation.

There are two pages for transportation updates one is when cloth arrived to hangar and the other is when cloth is arrived to people in need. In these pages there are active transportations which are not done yet and once you click to rows in table and update with arrival date then transportation is become completed.

## **SPECIFICATION**

As we know, the need for clothing is an important part of every person's life. Many organizations provide clothing aid to poor regions where this need cannot be met. After long observations, although we saw that these charities did not keep records to keep the donation confidential, we came to the conclusion that this actually caused many problems. The reason for this is that the clothes given go to other places rather than the actual destination. Therefore, we decided to set up a

database. Thanks to this system, users will open an account and donate clothes and give details about the clothes they give. The charity employees collecting these clothes will make a classification process based on the characteristics of the dress givers entered into the system. Finally, people who need clothes will register to the system and give some information. To give an example of this information, these are some elements that are important in giving clothes such as height, weight, body size. When all these things are implemented, a very effective clothe sharing system emerges. and thanks to these records, curious people will be able to confirm that the dresses are reaching the people who really need it.

## **GENERAL DESCRIPTION**

The cloth aid system our purpose is to make sufficient and organizable application with the MySQL database management system. In the system, there are 2 significant data one of them is people who need clothes and the other one is people who sent his/him clothes. In the cloth aid system, we will keep the recipient, donor information. Determining the size of the clothes that will go to help beforehand, understanding who needs which clothes by the system, we will be late for the wrong clothes to go. At the same time, we aim to make the aid campaign more contractual and efficient by keeping the information of the people who help and the people helped, the information of the transfer center, and the information of the people who will help. In this way, by having a general background by the charity organization, aid campaign information can be easily sent to the same people in future aid campaigns.

## **TOOLS/IDES**

### **MySQL Workbench:**

It is the most important program in order to complete Project. It will be used to create and manage database system of the Project.

### **NetBeans:**

This application is used to create desktop application. Using NetBeans is much more easier than using eclipse because there is no help while creating user interface in eclipse but in NetBeans, Java SWING has lots of features that can be used in projects.

### **Java:**

Java is used in NetBeans in order to create projects which works with MySQL. As we know Java is very detailed language and it offers lots of important features which are necessary to create very good project.

### **GitHub:**

It will be used to share the versions of mobile application.

## HIGH LEVEL DIAGRAM

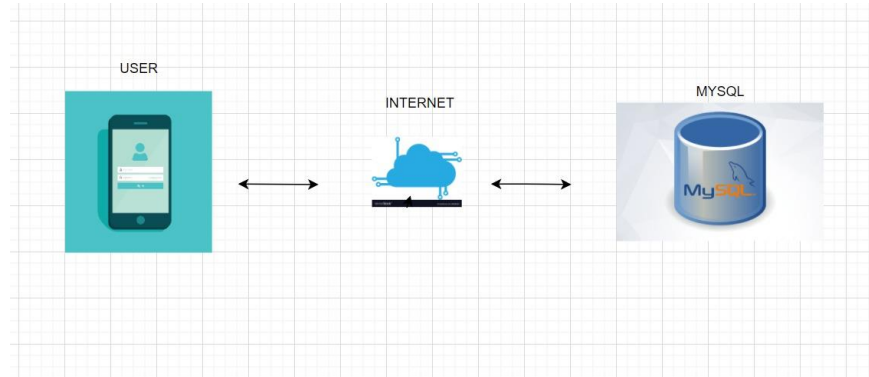


Figure1.: Tools

## UML USE CASE DIAGRAM

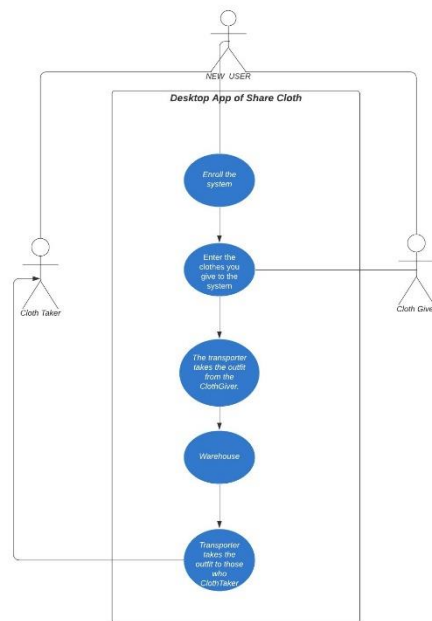


Figure2.: UML Diagram

## E-R DIAGRAM

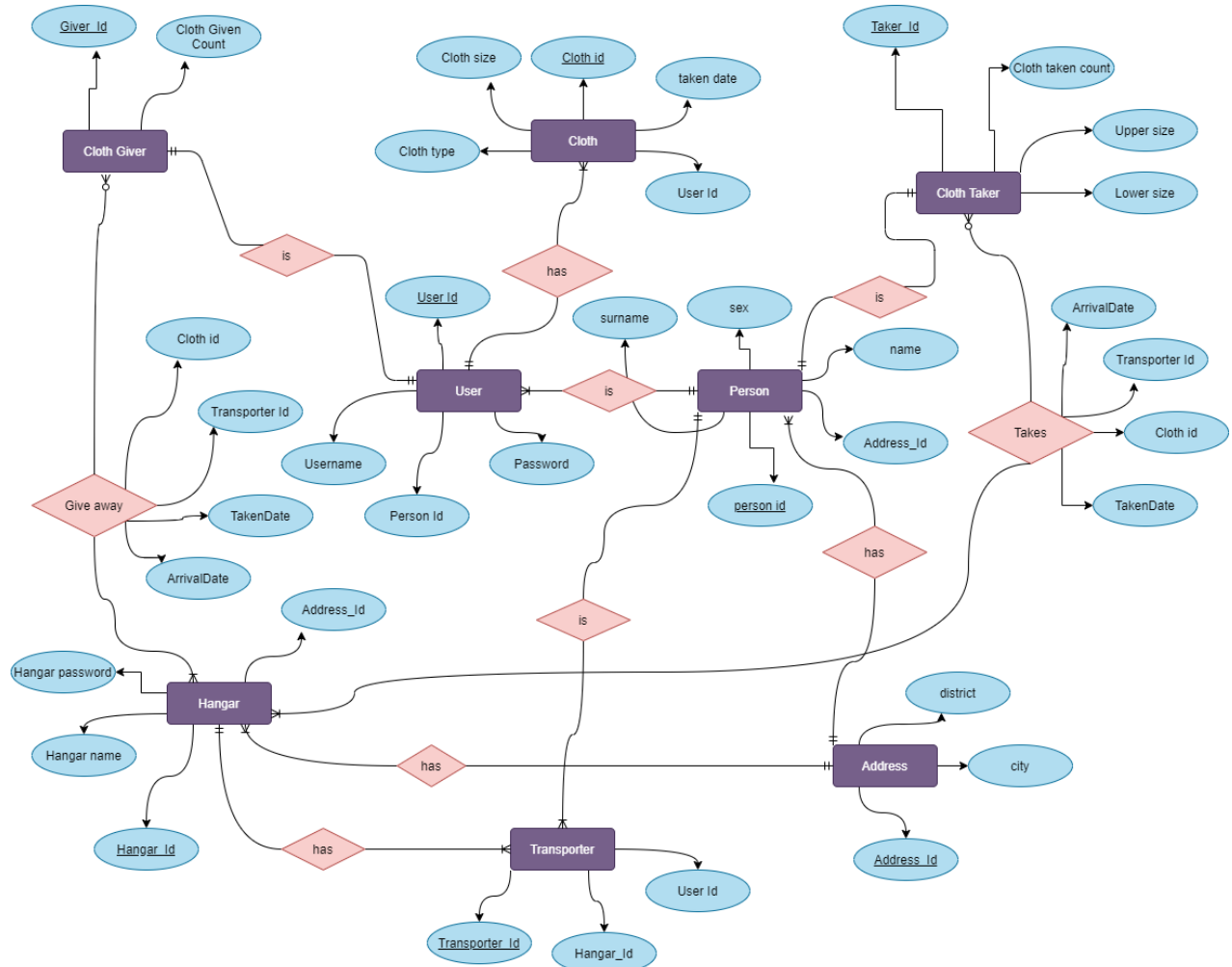


Figure2.: E-R Diagram

## DESIGN PHILOSOPHY

### ER Diagram:

- User
- Person
- Clothe\_giver
- Clothe\_taker
- Cloth
- Hangar
- Transporter
- Adress

In this part we will show the ER diagram to explain the whole system. By showing the ER diagram, sections that are not understood in the ER diagram will be understood.

**User:** gives its information to the **Clothe\_giver**, **Person**, **Cloth** and **Transporter**. Basically it keeps the general information of the user like: **username** and **password**. **User\_id** will be generated automatically and with this way other entities takes user entity information's.

**Person:** Person class **name**, **surname** **sex** and **person\_id(generated automatically)**. Also there are another two subclass which are **Clothe\_taker**, **Clothe\_giver** sub classes, those are connected to the **User** and **Person** entities.

**Clothe\_giver:** Clothe\_giver is a sub-class of **User** entity. **Clothe\_giver** keeps **user\_id**, **cloth\_given\_count**. Finally it has a foreign key comes from User Entity.

**Adress:** Adress has connection between person and it keeps, **city** name and **district**.

**Transporter:** have connections between takes, hangar, user and giveaway. It keeps the **user\_id** and **hangar\_id** and **transporter\_id**.

**Cloth:** is subclass of user. It keeps **cloth\_id (generated automatically)**, **user\_id** **cloth\_size**, **cloth\_type**, **hangar\_id**.

**Hangar:** keeps the information like **hangar\_id(generated automatically)**, **address\_id** **hangar\_name**.

**Clothe\_Taker:** Clothe\_taker is a sub-class of **Person** entity. **Clothe\_taker** keeps **person\_id**, **lowerSize**, **upperSize** and **cloth\_taken\_count**. Finally it has a foreign key comes from Perosn Entity.



## **CARDINALITIES**

### **MANY TO OPTIONAL MANY**

Warehouse, Cloth Giver

Warehouse, Cloth Taker

### **ONE TO MANY**

Person, User

User, Transporter

Warehouse, Transporter

Person, ClothTaker

User, Cloth Giver

Person, Cloth

Address, Person

Address, Hangar

## **USER PERMISSIONS**

### **CLOTH GIVER**

Add: User, Clothe type

Delete: User, Clothe type,

Update: User, Clothe type,

View: User, Clothe type, Warehouse

### **CLOTH TAKER**

Add: User, Clothe type

Delete: User, Clothe type,

Update: User, Clothe type,

View: User, Clothe type, Warehouse

### **WAREHOUSE**

Add: Hangar name

Delete: Hangar name

Update: Hangar name

View: Hangar name, Cloth giver, Cloth Taker

## PHASE 2

### NORMALIZATION

I will talk about the normalization process that we have done for a more consistent and stable operation of the database. In addition to stability, we envisioned preventing the parts that occupy space in the system due to duplicate data by developing a better design.

I will try to explain in detail what we are doing to achieve 1NF, the first step in normalization. As it is known in 1NF, one information should be stored in a cell in the database, if more than one data is stored, this situation does not comply with 1NF. We also had this problem at the design stage. Donations made and donations received by individuals can be considered as an example to reach 1NF. because a person may have made more than one donation or a person may own more than one garment at the same time, and these are data that cannot be kept in a single cell. Therefore, 1NF status has been tried to be provided by using takes and giveAway relations.

I will talk about another step, 2NF, in this section. As seen in our system, 1 person has more than one feature, but they are not kept in only 1 table. To give an example, the person in the system can also be a clothTaker person. Besides, this person can also be a user. and this user could be clothGiver or Transporter. In order to provide all these, we were able to talk about the same person with the help of a foreign key. We defined the primary key of one table as the foreign key for the primary key of the other table. To give an example, clothTaker is a person and this person can use his person information using his TakerId because the primary key that defines the ClothTaker actually has the same value as the primary key that has its person properties. The same relationship as seen in this system exists between ClothGiver and user. Transporter is also connected to the user with the same logic and the user is connected to the person with the key it contains. Thanks to the features I have explained here, the desired conditions are also met in 2NF.

Finally, I want to talk about what we did for the 3NF part. province example is related to address. As we know, city and district properties belong to the addressId, also known as the postal code. and this city and district information depends on the address id. As mentioned in the definition of 3NF, if 1 or more columns are connected to a column other than the primary key, a separate table can be opened for these columns and this column can be used as a primary key to define them. As a result, city and district data are linked to address id and address id is not primary key, so if we leave a single address id in person and store the values related to this id in another table, we achieve this goal. The second example is related to cloth. As we know, the data of a cloth does not change according to the owner of that cloth, it is completely dependent on the cloth. In this case, if this cloth information depends on the clothId rather than the personId, it would not be logical to keep this cloth information in the person, so we transferred the cloth information to another table and these data are stored here depending on the clothId.

Thanks to these steps, the system runs very stable, and whenever an update, delete, insert or update command is given, the system continues to work stably. At the same time, different information about a person is kept in different tables in the divided tables, and while this is done, the same data is not stored repeatedly in order not to fill the memory unnecessarily. Another thing is that we store all personal information in separate tables rather than in a single table, so if a person is not a transporter, we do not keep information about that he is not a transporter in vain. If we were to keep all the information on a table, whether a person is a transporter, whoever is a dresser or whoever is, obviously, no distinction could be made, so this kind of unnecessary information would be kept and memory would have been filled in vain. In summary, a quality database was created thanks to the normalization steps implemented.

### **E-R to Relational Mapping**

- **Normal Entities**

Address(Address\_id, District, City)

Person(Person\_id, Name, Surname, Sex, Address\_id)

User(User\_id, Person\_id, username, password)

Hangar(Hangar\_id, HangarName, Address\_id)

Transporter(Transporter\_id, User\_id, Hangar\_id)

Cloth(Cloth\_id, User\_id, ClothSize, ClothType, Hangar\_id)

Clothgiver(ClothGiver\_id, ClothGiven\_id)

Clohtaker(ClothTaker\_id, LowerSize, UpperSize, ClothTakenCount)

- **Relationships**

Takes(Takes\_id, ArrivalDate, Transporter\_id, Cloth\_id, TakenDate, ClothTaker\_id, Hangar\_id)

Giveaway(Giveaway\_id, Cloth\_id, Transporter\_id, TakenDate, ArrivalDate, ClothGiver\_id, Hangar\_id)

### **Functional Dependencies**

**Address(Address\_id, District, City)**

Address\_id → District

Address\_id → City

**Identification key:** Address\_id

**Person(Person\_id, Name, Surname, Sex, Address\_id)**

Person\_id → Name

Person\_id → Surname

Person\_id → Sex

Person\_id → Address\_id

Person\_id → District

Person\_id → City

**Identification key:** Person\_id

**User(User\_id,Person\_id,username,password)**

User\_id → Person\_id

User\_id → username

User\_id → password

User\_id → Name

User\_id → Surname

User\_id → Sex

User\_id → District

User\_id → City

**Identification key:** User\_id

**Hangar(Hangar\_id, HangarName,Adress\_id)**

Hangar\_id → HangarName

Hangar\_id → address\_id

Hangar\_id → District

Hangar\_id → City

**Identification key:** Hangar\_id

**Transporter(Transporter\_id, User\_id, Hangar\_id)**

Transporter\_id → User\_id

Transporter\_id → Hangar\_id

Transporter\_id → Person\_id

Transporter\_id → username

Transporter\_id → password

Transporter\_id → Name

Transporter\_id → Surname

Transporter\_id → Sex

Transporter\_id → Address\_id

Transporter\_id → District

Transporter\_id → City

**Identification key:** Transporter\_id

**Cloth(Cloth\_id, User\_id, ClothSize, ClothType, Hangar\_id)**

Cloth\_id → User\_id

Cloth\_id → ClothSize

Cloth\_id → ClothType  
Cloth\_id → Hangar\_id  
Cloth\_id → Person\_id  
Cloth\_id → Name  
Cloth\_id → Surname  
Cloth\_id → Sex

Cloth\_id → Address\_id  
Cloth\_id → District  
Cloth\_id → City  
Cloth\_id → HangarName  
**Identification key:** Cloth\_id

**Clothgiver(ClothGiver\_id, ClothGiven\_id)**

ClothGiver\_id → ClothGiven\_id  
ClothGiven\_id → ClothGiver\_id  
**Identification key:** ClothGiven\_id

**Clohtaker(ClothTaker\_id, LowerSize, UpperSize, ClothTakenCount)**

ClothTaker\_id → LowerSize  
ClothTaker\_id → UpperSize  
ClothTaker\_id → ClothTakenCount  
**Identification key:** ClothTaker\_id

**Takes(Takes\_id, ArrivalDate, Transporter\_id, Cloth\_id, TakenDate, ClothTaker\_id, Hangar\_id)**

Takes\_id → ArrivalDate  
Takes\_id → Transporter\_id  
Takes\_id → Cloth\_id  
Takes\_id → TakenDate  
Takes\_id → ClothTaker\_id  
Takes\_id → Hangar\_id  
Takes\_id → User\_id  
Takes\_id → Person\_id  
Takes\_id → username  
Takes\_id → password  
Takes\_id → Name  
Takes\_id → Surname  
Takes\_id → Sex

Takes\_id → Address\_id  
Takes\_id → District  
Takes\_id → City  
Takes\_id → LowerSize  
Takes\_id → UpperSize  
Takes\_id → ClothTakenCount  
Takes\_id → HangarName  
**Identification key:** Takes\_id

**Giveaway(Giveaway\_id, Cloth\_id, Transporter\_id, TakenDate, ArrivalDate, ClothGiver\_id, Hangar\_id)**

Giveaway\_id → Cloth\_id  
Giveaway\_id → Transporter\_id  
Giveaway\_id → TakenDate  
Giveaway\_id → ArrivalDate  
Giveaway\_id → ClothGiver\_id  
Giveaway\_id → Hangar\_id  
Giveaway\_id → User\_id  
Giveaway\_id → Person\_id  
Giveaway\_id → username  
Giveaway\_id → password  
Giveaway\_id → Name  
Giveaway\_id → Surname  
Giveaway\_id → Sex  
Giveaway\_id → Address\_id  
Giveaway\_id → District  
Giveaway\_id → City  
Giveaway\_id → ClothGiven\_id  
Giveaway\_id → HangarName  
**Identification key:** Giveaway\_id

## DATABASE SCHEMA

Address			
	Address_id	City	District
Type	INT	VARCHAR(30)	VARCHAR(30)
Key	PKey	Key	Key
Example	110210198	Malatya	Battalgazi

Person					
	Person_id	Name	Surname	Sex	Address_id
Type	INT	VARCHAR(50)	VARCHAR(50)	VARCHAR(10)	INT
Key	Pkey	Key	Key	Key	FKey
Example	5511652	Leyla	Yilmaz	Female	110210198

User				
	User_id	Person_id	username	password
Type	INT	INT	VARCHAR(50)	VARCHAR(50)
Key	PKey	FKey	Key	Key
Example	10	5511652	User52	1146fdfdf

Hangar			
	Hangar_id	Hangarname	Address_id
Type	INT	VARCHAR(20)	INT
Key	Pkey	Key	FKey
Exampe	156	Kanal	110210198



Transporter			
	Transporter_id	User_id	Hangar_id
Type	INT	INT	INT
Key	PKey	FKey	FKey
Example	2563	10	156

Cloth					
	Cloth_id	User_id	ClothSize	ClothType	Hangar_id
Type	INT	INT	INT	VARCHAR(50)	INT
Key	Pkey	FKey	Key	Key	FKey
Example	85201	10	38	skirt	156

ClothTaker				
	ClothTaker_id	LowerSize	UpperSize	ClothTaken Count
Type	INT	INT	INT	INT
Key	PKey	Key	Key	Key
Example	1235	36	40	15

ClothGiver		
	ClothGiver_id	ClothTaker_id
Type	INT	INT
Key	PKey	Fkey
Example	5698	3256

Takes							
	Takes_id	ArrivalDate	Transporter_id	Cloth_id	TakenDate	ClothTaker_id	Hangar_id
Type	INT	VARCHAR(20)	INT	INT	VARCHAR(20)	INT	INT
Key	PKey	Key	FKey	FKey	Key	FKey	FKey
Example	15445	12/05/2021	2563	85201	15/05/2021	3256	156

GiveAway							
	Giveaway_id	ArrivalDate	Transporter_id	Cloth_id	TakenDate	ClothGiver_id	Hangar_id
Type	INT	VARCHAR(20)	INT	INT	VARCHAR(20)	INT	INT
Key	PKey	Key	FKey	FKey	FKey	FKey	FKey
Example	56565	12/05/2021	2563	85201	15/05/2021	5698	156

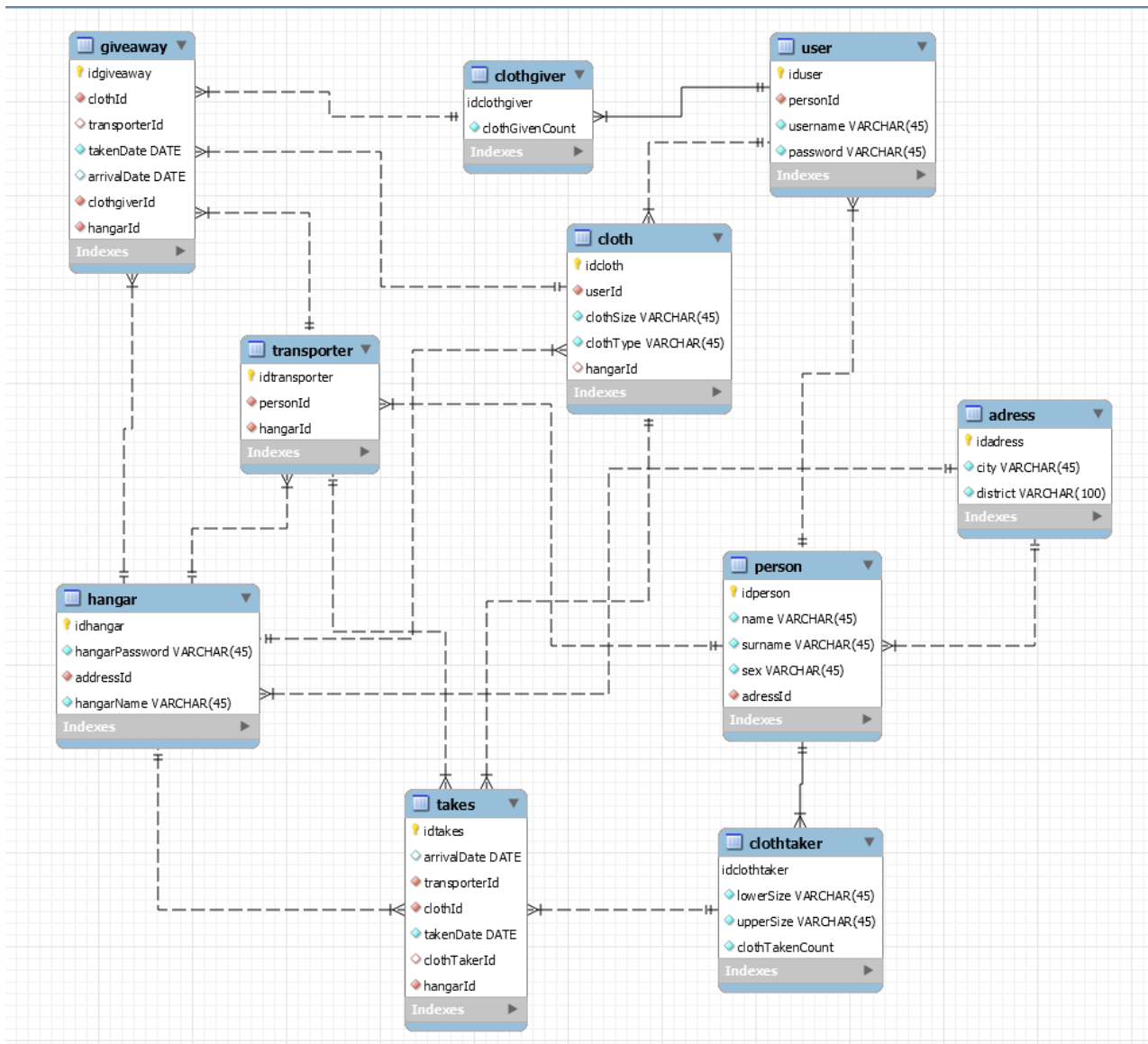


Figure3.: EER Database Schema

## SCRIPTS

### DDL CODES:

```
CREATE TABLE `adress` (
  `idadress` int NOT NULL AUTO_INCREMENT,
  `city` varchar(45) NOT NULL,
  `district` varchar(100) NOT NULL,
  PRIMARY KEY (`idadress`)
);
```

```
CREATE TABLE `person` (
```

```

`idperson` int NOT NULL AUTO_INCREMENT,
`name` varchar(45) NOT NULL,
`surname` varchar(45) NOT NULL,
`sex` varchar(45) NOT NULL,
`adressId` int NOT NULL,
PRIMARY KEY (`idperson`),
KEY `person_FK1_idx` (`adressId`),
CONSTRAINT `person_FK1` FOREIGN KEY (`adressId`) REFERENCES `adress`
(`idadress`)
);

```

```

CREATE TABLE `hangar` (
`idhangar` int NOT NULL AUTO_INCREMENT,
`hangarPassword` varchar(45) NOT NULL,
`addressId` int NOT NULL,
`hangarName` varchar(45) NOT NULL,
PRIMARY KEY (`idhangar`),
KEY `hangar_FK1_idx` (`addressId`),
CONSTRAINT `hangar_FK1` FOREIGN KEY (`addressId`) REFERENCES `adress`
(`idadress`)
);

```

```

CREATE TABLE `user` (
`iduser` int NOT NULL AUTO_INCREMENT,
`personId` int NOT NULL,
`username` varchar(45) NOT NULL,
`password` varchar(45) NOT NULL,
PRIMARY KEY (`iduser`),
KEY `user_FK1_idx` (`personId`),
CONSTRAINT `user_FK1` FOREIGN KEY (`personId`) REFERENCES `person`
(`idperson`)
);

```

```

CREATE TABLE `cloth` (
`idcloth` int NOT NULL AUTO_INCREMENT,
`userId` int NOT NULL,
`clothSize` varchar(45) NOT NULL,
`clothType` varchar(45) NOT NULL,
`hangarId` int DEFAULT NULL,
PRIMARY KEY (`idcloth`),
KEY `cloth_FK2_idx` (`hangarId`),
KEY `cloth_FK1_idx` (`userId`),
CONSTRAINT `cloth_FK1` FOREIGN KEY (`userId`) REFERENCES `user` (`iduser`),
CONSTRAINT `cloth_FK2` FOREIGN KEY (`hangarId`) REFERENCES `hangar`
(`idhangar`)
);

```

```

CREATE TABLE `transporter` (
  `idtransporter` int NOT NULL AUTO_INCREMENT,
  `personId` int NOT NULL,
  `hangarId` int NOT NULL,
  PRIMARY KEY (`idtransporter`),
  KEY `transporter_FK2_idx` (`hangarId`),
  KEY `transporter_FK1_idx` (`personId`),
  CONSTRAINT `transporter_FK1` FOREIGN KEY (`personId`) REFERENCES `person`
(`idperson`),
  CONSTRAINT `transporter_FK2` FOREIGN KEY (`hangarId`) REFERENCES `hangar`
(`idhangar`)
);

```

```

CREATE TABLE `clothgiver` (
  `idclothgiver` int NOT NULL,
  `clothGivenCount` int NOT NULL,
  PRIMARY KEY (`idclothgiver`),
  KEY `clothgiver_FK1_idx` (`idclothgiver`),
  CONSTRAINT `clothgiver_FK1` FOREIGN KEY (`idclothgiver`) REFERENCES `user`
(`iduser`)
);

```

```

CREATE TABLE `clothtaker` (
  `idclothtaker` int NOT NULL,
  `lowerSize` varchar(45) NOT NULL,
  `upperSize` varchar(45) NOT NULL,
  `clothTakenCount` int NOT NULL,
  PRIMARY KEY (`idclothtaker`),
  CONSTRAINT `clothtaker_FK1` FOREIGN KEY (`idclothtaker`) REFERENCES `person`
(`idperson`)
) ;

```

```

CREATE TABLE `giveaway` (
  `idgiveaway` int NOT NULL AUTO_INCREMENT,
  `clothId` int NOT NULL,
  `transporterId` int DEFAULT NULL,
  `takenDate` date NOT NULL,
  `arrivalDate` date DEFAULT NULL,
  `clothgiverId` int NOT NULL,
  `hangarId` int NOT NULL,
  PRIMARY KEY (`idgiveaway`),
  KEY `giveaway_FK1_idx` (`clothId`),
  KEY `giveaway_FK3_idx` (`clothgiverId`),
  KEY `giveaway_FK4_idx` (`hangarId`),
  KEY `giveaway_FK2_idx` (`transporterId`),
  CONSTRAINT `giveaway_FK1` FOREIGN KEY (`clothId`) REFERENCES `cloth`
(`idcloth`),

```

```

    CONSTRAINT `giveaway_FK2` FOREIGN KEY (`transporterId`) REFERENCES
`transporter` (`idtransporter`),
    CONSTRAINT `giveaway_FK3` FOREIGN KEY (`clothgiverId`) REFERENCES
`clothgiver` (`idclothgiver`),
    CONSTRAINT `giveaway_FK4` FOREIGN KEY (`hangarId`) REFERENCES `hangar`
(`idhangar`)
);

CREATE TABLE `takes` (
    `idtakes` int NOT NULL AUTO_INCREMENT,
    `arrivalDate` date DEFAULT NULL,
    `transporterId` int NOT NULL,
    `clothId` int NOT NULL,
    `takenDate` date NOT NULL,
    `clothTakerId` int DEFAULT NULL,
    `hangarId` int NOT NULL,
    PRIMARY KEY (`idtakes`),
    KEY `takes_FK4_idx` (`clothTakerId`),
    KEY `takes_FK1_idx` (`clothId`),
    KEY `takes_FK3_idx` (`hangarId`),
    KEY `takes_FK2_idx` (`transporterId`),
    CONSTRAINT `takes_FK1` FOREIGN KEY (`clothId`) REFERENCES `cloth` (`idcloth`),
    CONSTRAINT `takes_FK2` FOREIGN KEY (`transporterId`) REFERENCES `transporter`
(`idtransporter`),
    CONSTRAINT `takes_FK3` FOREIGN KEY (`hangarId`) REFERENCES `hangar`
(`idhangar`),
    CONSTRAINT `takes_FK4` FOREIGN KEY (`clothTakerId`) REFERENCES `clothtaker`
(`idclothtaker`)
);

```

## DML:

```

INSERT INTO adress (city,district) VALUES ("Ankara","Meydan");
INSERT INTO adress (city,district) VALUES ("İstanbul","Meydan");
INSERT INTO adress (city,district) VALUES ("Trabzon","Meydan");
INSERT INTO adress (city,district) VALUES ("İzmir","Meydan");
INSERT INTO adress (city,district) VALUES ("Bursa","Meydan");
INSERT INTO adress (city,district) VALUES ("Antalya","Meydan");
INSERT INTO adress (city,district) VALUES ("Kayseri","Meydan");
INSERT INTO adress (city,district) VALUES ("Konya","Meydan");
INSERT INTO adress (city,district) VALUES ("Samsun","Meydan");
INSERT INTO adress (city,district) VALUES ("Muğla","Meydan");

```

```

Insert INTO person(name,surname,sex,adressId) VALUES ("Mustafa","Demiröz","male",3);
Insert INTO person(name,surname,sex,adressId) VALUES ("Sukufe","Arsoy","female",2);
Insert INTO person(name,surname,sex,adressId) VALUES ("Nur","Güçlü","female",1);
Insert INTO person(name,surname,sex,adressId) VALUES ("Yazganalp","Sakarya","male",1);

```

```

Insert INTO person(name,surname,sex,adressId) VALUES ("Tarık","Güçlü","male",4);
Insert INTO person(name,surname,sex,adressId) VALUES ("Yücelen","Mansız","male",2);
Insert INTO person(name,surname,sex,adressId) VALUES ("Emine","Safak","female",2);
Insert INTO person(name,surname,sex,adressId) VALUES ("Deviner","Bilge","male",9);
Insert INTO person(name,surname,sex,adressId) VALUES ("Elif","Mansız","female",8);
Insert INTO person(name,surname,sex,adressId) VALUES ("Sanur","Yüksel","female",10);
Insert INTO person(name,surname,sex,adressId) VALUES ("Dilder","Karadeniz","female",5);
Insert INTO person(name,surname,sex,adressId) VALUES ("Alaaddin","Korutürk","male",2);
Insert INTO person(name,surname,sex,adressId) VALUES ("Duruk","Erdoğan","male",4);
Insert INTO person(name,surname,sex,adressId) VALUES ("Nursan","Yıldırım","female",8);
Insert INTO person(name,surname,sex,adressId) VALUES ("Aydinç","Demir","male",1);
Insert INTO person(name,surname,sex,adressId) VALUES
("Abdurrahman","Sener","male",2);
Insert INTO person (name,surname,sex,adressId) VALUES ("Bilge","Eraslan","female",1);
Insert INTO person (name,surname,sex,adressId) VALUES ("Ali","Zengin","male",2);
Insert INTO person (name,surname,sex,adressId) VALUES ("Ahmet","Erdoğan","male",8);
Insert INTO person (name,surname,sex,adressId) VALUES ("Mehmet","Durdu","male",9);
Insert INTO person (name,surname,sex,adressId) VALUES ("Ayse","Akgündüz","female",7);
Insert INTO person (name,surname,sex,adressId) VALUES ("Nazım","Bilir","male",6);
Insert INTO person (name,surname,sex,adressId) VALUES ("Müğber","Sezer","male",1);
Insert INTO person (name,surname,sex,adressId) VALUES ("Rıza","Akça","male",2);
Insert INTO person (name,surname,sex,adressId) VALUES ("Nadir","Türk","male",10);
Insert INTO person (name,surname,sex,adressId) VALUES ("Yaren","Şensoy","female",1);
Insert INTO person (name,surname,sex,adressId) VALUES ("Yudum","Aslan","female",4);
Insert INTO person (name,surname,sex,adressId) VALUES ("Beyza","Güçlü","female",1);
Insert INTO person (name,surname,sex,adressId) VALUES ("Akif","Ergül","male",9);
Insert INTO person (name,surname,sex,adressId) VALUES ("Melik","Soylu","male",1);
Insert INTO person (name,surname,sex,adressId) VALUES ("Berat","Aylak","male",2);
Insert INTO person (name,surname,sex,adressId) VALUES ("Ahmet","Aydın","male",1);
Insert INTO person (name,surname,sex,adressId) VALUES ("Melek","Erdal","female",2);
Insert INTO person (name,surname,sex,adressId) VALUES ("Nursena","Tütüncü","female",2);
Insert INTO person (name,surname,sex,adressId) VALUES ("Ali","Engin","male",2);
Insert INTO person (name,surname,sex,adressId) VALUES ("Elif","Tasdemir","female",2);
Insert INTO person (name,surname,sex,adressId) VALUES ("Öktem","Güvenç","male",1);
Insert INTO person (name,surname,sex,adressId) VALUES ("Nisa","Ateş","female",1);
Insert INTO person (name,surname,sex,adressId) VALUES ("Kadir","Yesilyurt","male",1);
Insert INTO person (name,surname,sex,adressId) VALUES ("Ugur","Gökdemir","male",2);
Insert INTO person (name,surname,sex,adressId) VALUES ("Mustafa","Öztaskın","male",2);
Insert INTO person(name,surname,sex,adressId) VALUES ("İsmail","Sulak","male",3);
Insert INTO person(name,surname,sex,adressId) VALUES ("Berat","Demircan","male",8);
Insert INTO person(name,surname,sex,adressId) VALUES ("Anıl","Gürdemir","male",7);
Insert INTO person (name,surname,sex,adressId) VALUES ("Batuhan","Demirtürk","male",6);
Insert INTO person (name,surname,sex,adressId) VALUES ("Emine","Akcan","female",4);
Insert INTO person (name,surname,sex,adressId) VALUES ("Merve","Yaldız","female",3);
Insert INTO person (name,surname,sex,adressId) VALUES ("Hüma","Bayazıt","female",7);
Insert INTO person (name,surname,sex,adressId) VALUES ("Zeynep","Altunsoy","female",6);
Insert INTO person (name,surname,sex,adressId) VALUES ("Eren","Ercan","male",5);

```



```

INSERT INTO user (personId,username,password) VALUES (1,"sandarsa","1234");
INSERT INTO user (personId,username,password) VALUES (2,"yolgezer","12345");
INSERT INTO user (personId,username,password) VALUES (3,"apollo","1234");
INSERT INTO user (personId,username,password) VALUES (4,"america","1234567");
INSERT INTO user (personId,username,password) VALUES (5,"nobody","pass");

```

```

INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (23487,1,"Ankara
Meydan Hangar");
INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (23485,2,"Istanbul
Meydan Hangar");
INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (34523,3,"Trabzon
Meydan Hangar");
INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (78643,4,"Izmir
Meydan Hangar");
INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (34532,5,"Bursa
Meydan Hangar");
INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (23487,6,"Antalya
Meydan Hangar");
INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (87483,7,"Kayseri
Meydan Hangar");
INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (53435,8,"Konya
Meydan Hangar");
INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (98564,9,"Samsun
Meydan Hangar");
INSERT INTO hangar(hangarPassword,addressId,hangarName) VALUES (98746,10,"Mugla
Meydan Hangar");

```

```

INSERT INTO cloth(userId,clothSize,clothType,hangarId) VALUES (1,"M","T-Shirt",1);
INSERT INTO cloth(userId,clothSize,clothType,hangarId) VALUES (2,"XXL","T-Shirt",2);
INSERT INTO cloth(userId,clothSize,clothType,hangarId) VALUES (3,"L","Pants",1);
INSERT INTO cloth(userId,clothSize,clothType,hangarId) VALUES (3,"S","Skirt",1);
INSERT INTO cloth(userId,clothSize,clothType,hangarId) VALUES (4,"M","T-Shirt",1);
INSERT INTO cloth(userId,clothSize,clothType,hangarId) VALUES (4,"L","Skirt",2);
INSERT INTO cloth(userId,clothSize,clothType,hangarId) VALUES (1,"XL","T-Shirt",1);
INSERT INTO cloth(userId,clothSize,clothType,hangarId) VALUES (2,"M","Pants",3);

```

```

INSERT INTO transporter(personId,hangarId) VALUES (15,1);
INSERT INTO transporter(personId,hangarId) VALUES (16,2);
INSERT INTO transporter(personId,hangarId) VALUES (17,3);
INSERT INTO transporter(personId,hangarId) VALUES (18,4);
INSERT INTO transporter(personId,hangarId) VALUES (19,5);

```

```
INSERT INTO transporter(personId,hangarId) VALUES (20,6);
```

```
INSERT INTO clothgiver(idClothGiver,clothGivenCount) VALUES (1,0);  
INSERT INTO clothgiver(idClothGiver,clothGivenCount) VALUES (2,0);  
INSERT INTO clothgiver(idClothGiver,clothGivenCount) VALUES (3,0);  
INSERT INTO clothgiver(idClothGiver,clothGivenCount) VALUES (4,0);  
INSERT INTO clothgiver(idClothGiver,clothGivenCount) VALUES (5,0);
```

```
INSERT INTO clohtaker(idClothTaker,lowerSize,upperSize,clothTakenCount) VALUES  
(25,"S","M",0);  
INSERT INTO clohtaker(idClothTaker,lowerSize,upperSize,clothTakenCount) VALUES  
(26,"M","M",0);  
INSERT INTO clohtaker(idClothTaker,lowerSize,upperSize,clothTakenCount) VALUES  
(27,"L","XL",0);  
INSERT INTO clohtaker(idClothTaker,lowerSize,upperSize,clothTakenCount) VALUES  
(28,"XXL","XL",0);  
INSERT INTO clohtaker(idClothTaker,lowerSize,upperSize,clothTakenCount) VALUES  
(29,"S","M",0);  
INSERT INTO clohtaker(idClothTaker,lowerSize,upperSize,clothTakenCount) VALUES  
(30,"M","M",0);
```

```
INSERT INTO giveaway(clothId,transporterId,takenDate,arrivalDate,clothGiverId,hangarId)  
VALUES (1,1,"2021-05-25","2020-06-01",3,1);  
INSERT INTO giveaway(clothId,transporterId,takenDate,clothGiverId,hangarId) VALUES  
(2,1,"2021-05-25",3,1);  
INSERT INTO giveaway(clothId,takenDate,clothGiverId,hangarId) VALUES (3,"2021-05-  
25",1,1);  
INSERT INTO giveaway(clothId,transporterId,takenDate,arrivalDate,clothGiverId,hangarId)  
VALUES (4,1,"2021-05-25","2020-06-01",3,1);
```

```
INSERT INTO takes(arrivalDate,transporterId,clothId,takenDate,clothTakerId,hangarId)  
VALUES ("2021-06-15",1,1,"2021-06-14",30,1);  
INSERT INTO takes(transporterId,clothId,takenDate,clothTakerId,hangarId) VALUES  
(1,2,"2021-06-14",25,1);
```

### **APPLICATION DELETE OPERATION:**

With the link provided here you can check delete operation on database with our application. As you wanted, we recorded a video which shows delete operation in our application.

**Video Link:** <https://www.youtube.com/watch?v=j6TRX4hVP1w>

## VIEWS

As you can see below, we added some crucial views in order to gather data from our database. These views will be used very actively in our projects interface. In this section, some database tables will be combined as views and will be used to process in the background more easily.

**1. Clothes in the hangar:** This gives the size, id of the available clothes in which hangar they are located.

```
Create view available_cloths_in_given_hangar as select idcloth, clothType,
clothSize, hangarName
from giveaway inner join cloth inner join hangar on giveaway.clothId=cloth.idcloth and
hangar.idhangar = cloth.hangarId where arrivalDate is not null and (clothId) not in
(select clothId from takes);
;
```

**2. Clothes that have left the hangar and have not reached the person:** This indicates that the donated clothes have left the hangar but have not yet reached the person in need.

```
Create view on_the_way_cloths as select
username, clothId, clothType, clothSize, hangarName, name, surname, transporterId from person
inner join takes inner join cloth inner join user inner join hangar on
hangar.idhangar=takes.hangarId and cloth.userId = user.iduser and takes.clothTakerId =
person.idperson and cloth.idcloth = takes.clothId where arrivalDate is null;
```

**3. Arrived clothes:** It displays information such as from which hangar the donated clothes came, the size, the date they reached the person.

```
Create view arrived_cloths as select username, clothId, clothType,
clothSize, hangarName, name, surname from person inner join takes inner join cloth inner join
user inner join hangar on hangar.idhangar=takes.hangarId and cloth.userId = user.iduser and
takes.clothTakerId = person.idperson and cloth.idcloth = takes.clothId where arrivalDate is not
null;
```

**4. Clothing that has or has not reached the hangar and not transferred to person in need:**  
It indicates clothing that has been donated but has not transferred to person in need.

```
Create view cloths_waiting_for_transfer as select
idcloth, clothType, clothSize, hangarName, username
from cloth inner join hangar inner join user on hangar.idhangar = cloth.hangarId and
cloth.userId = user.iduser where (idcloth) not in (select clothId from takes);
;
```

5.

**The clothes that the person gave for help but haven't arrive to hangar:** After the user gives a dress to the system, it checks whether the given dress has reached the hangar.

**Create view cloth\_didnt\_arrive\_hangar as select** idcloth, clothType, clothSize, hangarName **from** giveaway inner join cloth inner join hangar on giveaway.clothId=cloth.idcloth and hangar.idhangar = cloth.hangarId **where** arrivalDate is null and (clothId) not in (select clothId **from** takes);  
;

6.

**In this section, we use the lower body size and the upper body size, which are the body measurements of the person in need:** we use the users' names, surnames, lower bodies and upper bodies to give information about these users

**Create view clothTaking\_person as select** idperson, name, surname, lowerSize, upperSize, city, district **from** person inner join clohtaker inner join adress on person.idperson=clohtaker.idclohtaker and person.adressId=adress.idadress;

7.

**In this section we get all these transporters for creating new delivery:**

By using this view, we call a query which returns transporters whose hangar is given.

**Create view transporter\_information as select** idtransporter, name, surname, hangarName, hangarId, personId **from** transporter inner join hangar inner join person on hangar.idhangar = transporter.hangarId and person.idperson=transporter.personId;

## **TRIGGERS:**

1.

In this trigger, the aim is to ensure that when the user registers the system his/her lastname is going to save MySQL database management system as Upper Casse Letters.

**CREATE TRIGGER** trigger\_gender  
**BEFORE INSERT** on person  
for each row  
**SET** new.surname = UPPER(new.surname);

2.

In this trigger when the user donate his/her clothe(s) to the system. The count of the user should be updated. So, with this trigger we update the user givenCountNumber with this update we can use this number easily in other views.

**CREATE TRIGGER** update\_cloth\_count

**BEFORE UPDATE** on clothgiver  
for each row  
**SET** new.clothGivenCount = new.clothGivenCount;

### 3.

When we get the size of clothes from user. Sometimes, they can insert lowercases or uppercases. When we use triggers, we can fix these problem and when we show in the application all the sizes will be shown as upper cases. So, there would be no misunderstanding in the application.

**CREATE TRIGGER** upper\_cloth\_size  
**BEFORE INSERT** on cloth  
for each row  
**SET** new.clothSize = UPPER(new.clothSize);

## DATABASE APPLICATION

Some software applications were used to complete this project and develop its interface. Used from Netbeans, MySql to run and build this whole process.

When the main page of the program is entered, it is first displayed on the user page. There are two different options here as User and Admin. From this page, you can log in to the system by entering your information. If you have not registered before, you can go to the registration page and fill in the necessary information to register.

After registration or entrance, you will be directed to the 'information page', where personal information such as your name, surname, gender is included.

If you want to donate clothes after logging into the system as a user, you should click the 'cloth relieve' button. This will take you to the 'Donate your cloth' page where you can enter the features and other information of the outfit you wish to donate.

If you log in to the system as an admin instead of a user, you will be able to manage your work on several different pages.

For example, by coming to the 'Hangar' page, you can have a large part of the system. By clicking the 'add person in need' button on this page, you will be directed to that page and you

can add the personal information and body information of the people in need.

You can switch to this page by clicking on the 'update route' button, which is another button on the hangar page, and you can update the donated clothes reaching the hangar here.

You can check the location and delivery of the clothes on the 'Cloth information' page.

You will also be directed to 'cloth entrance' page by clicking the 'cloth entrance' button on the Hangar page. Thus, you can register the clothes coming to the hangar for donation on this page.

By coming to the 'Create e new transportation' page, you can match the clothes with the needy and create a new aid. You can also view the person who will carry this outfit here.

You can enter the information of the people who deliver the clothes by clicking the 'add transporter' button on the hangar page.

## **INTERFACE OF APPLICATION**

—□×

**WELCOME**

USER INFORMATION

USERNAME:


PASSWORD:

☒ User

☐ Admin

Login

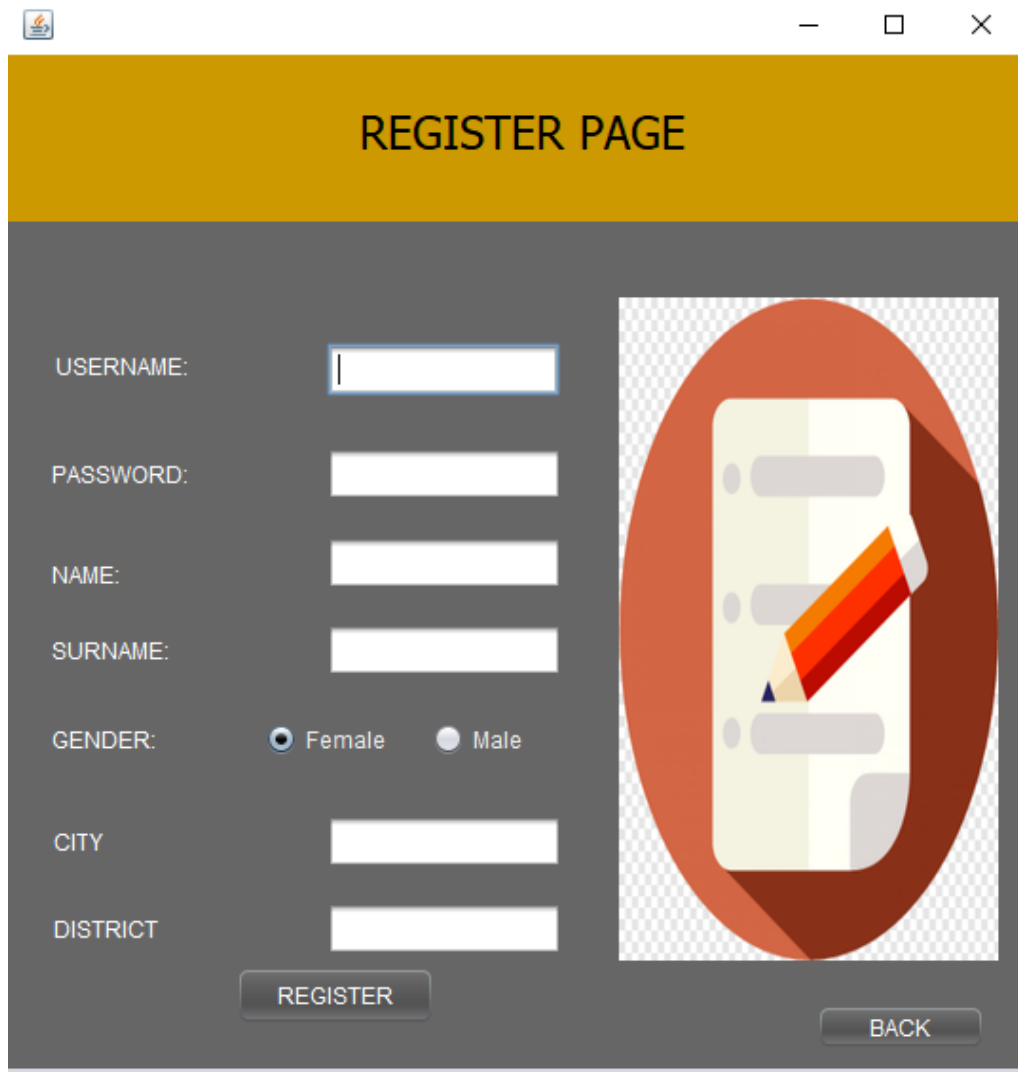
Register



**We happy to see you :)**

Figure4: Login page

31



REGISTER PAGE

USERNAME:

PASSWORD:

NAME:

SURNAME:

GENDER: ☒ Female ☐ Male

CITY

DISTRICT

*Figure5: Register page*



—

□

×

## Information Page

NAME	SURNAME	GENDER
Sukufe	Arsoy	female

Cloth Relieve

Cloth Status

Total cloth count 2

LOG OUT




Figure6: Information page

—

□

×

## Donate Your Cloth


Cloth Size:

Cloth Type:

Hangar:

Ankara Meydan Hangar

Donate



Back

Figure7: Cloth donation page

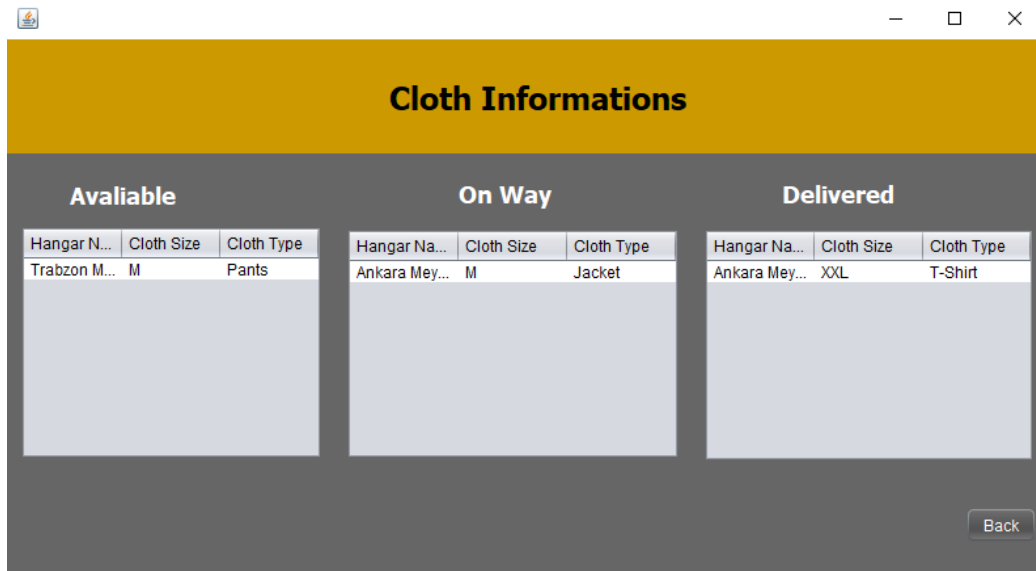


Figure8: Cloth Information page

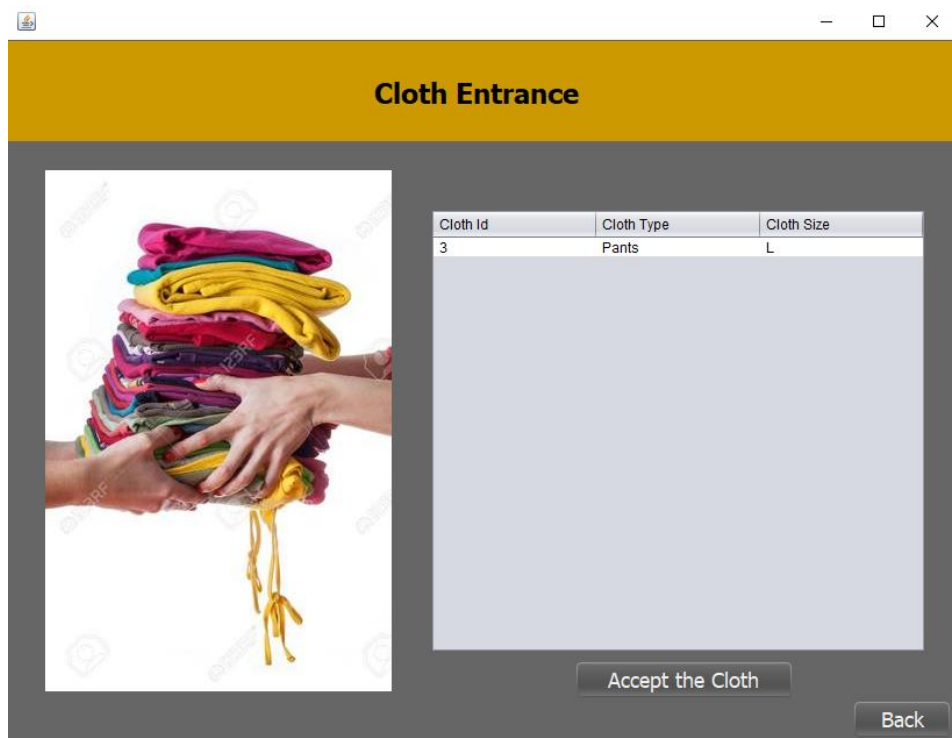
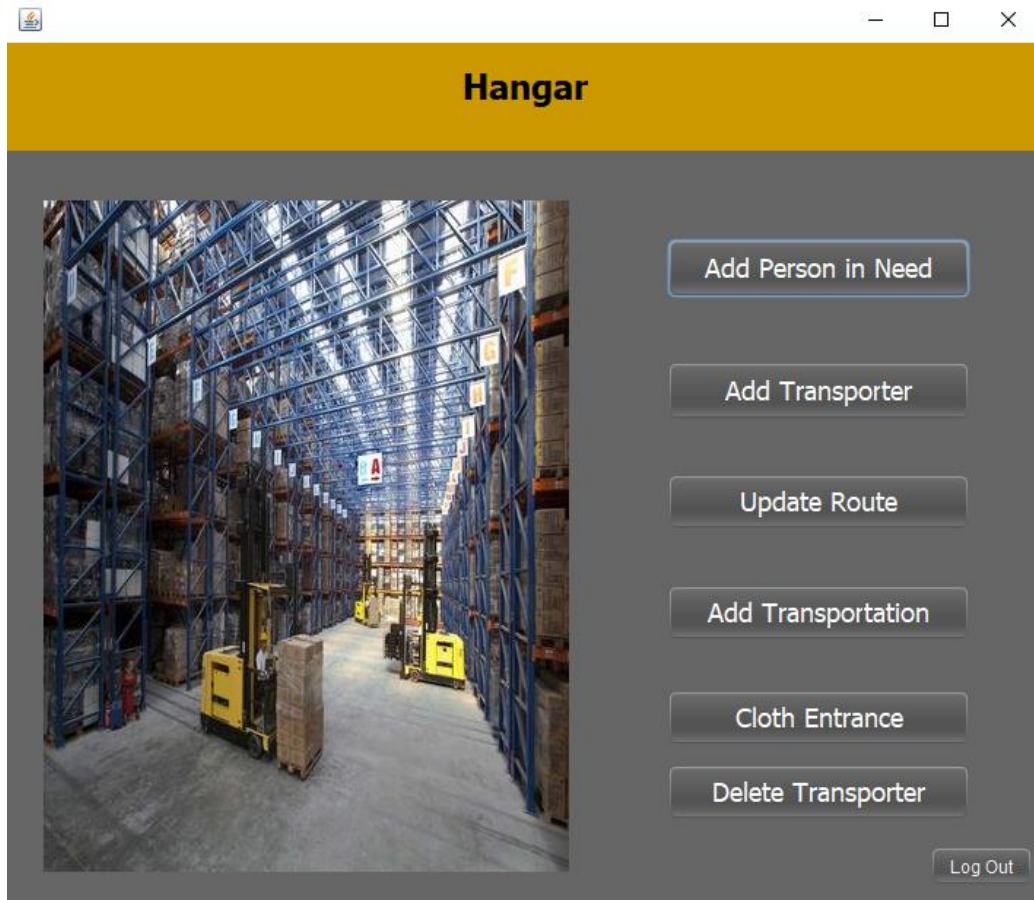
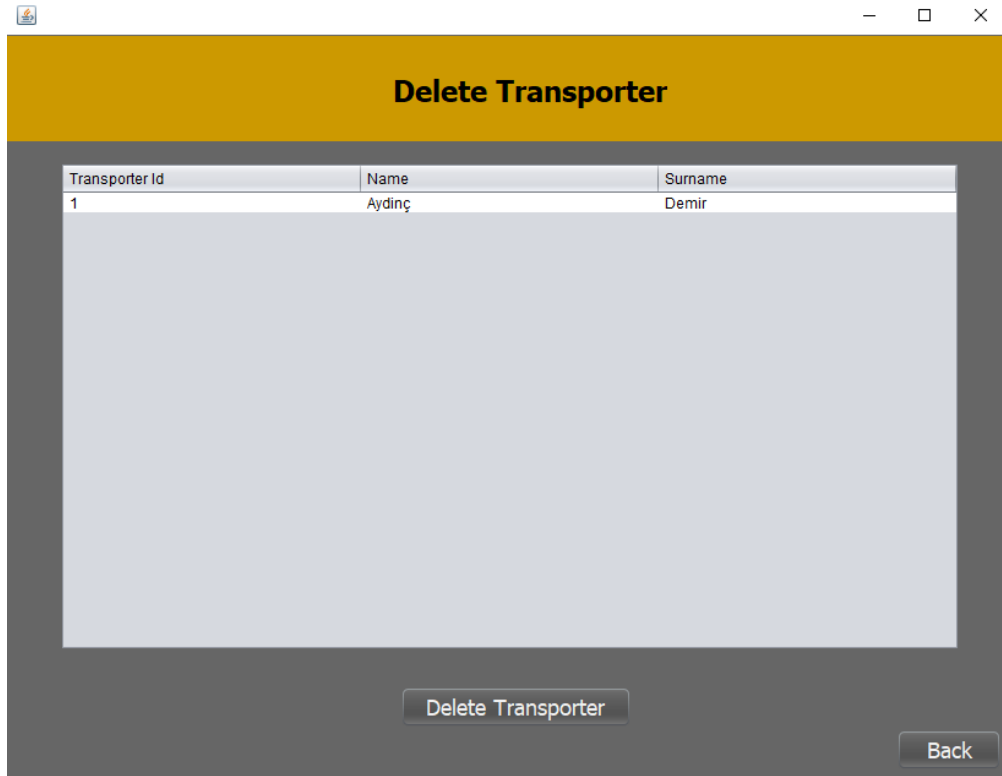


Figure9: Cloth entrance page



*Figure10: Hangar page*

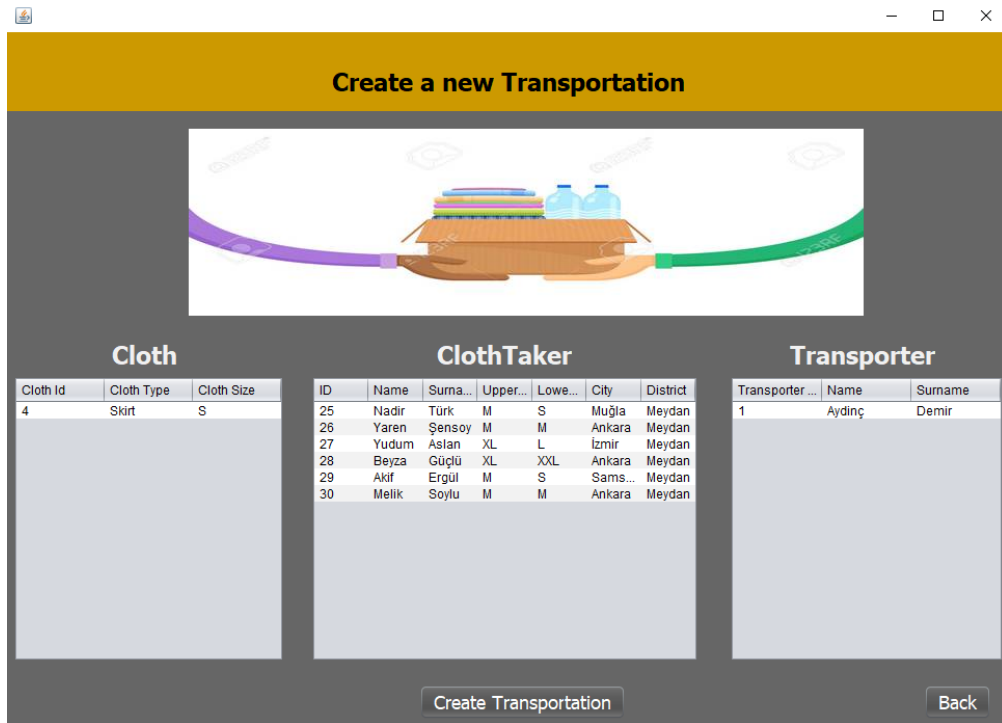


The screenshot shows a web application window titled "Delete Transporter". It features a table with three columns: "Transporter Id", "Name", and "Surname". The first row contains the values "1", "Aydiñ", and "Demir". Below the table is a large, empty rectangular area. At the bottom of the window, there are two buttons: "Delete Transporter" and "Back".

Transporter Id	Name	Surname
1	Aydiñ	Demir

Buttons: Delete Transporter, Back

Figure11: Delete Transporter page



The screenshot shows a web application window titled "Create a new Transportation". It features a central illustration of a hand holding a stack of clothes and two water bottles. Below the illustration are three tables: "Cloth", "ClothTaker", and "Transporter". At the bottom of the window, there are two buttons: "Create Transportation" and "Back".

Cloth Id	Cloth Type	Cloth Size
4	Skirt	S

ID	Name	Surna...	Upper...	Lowe...	City	District
25	Nadir	Türk	M	S	Muğla	Meydan
26	Yaren	Şensoy	M	M	Ankara	Meydan
27	Yudum	Astan	XL	L	Izmir	Meydan
28	Beyza	Güçlü	XL	XXL	Ankara	Meydan
29	Akıf	Ergül	M	S	Sams...	Meydan
30	Melik	Soylu	M	M	Ankara	Meydan

Transporter ...	Name	Surname
1	Aydiñ	Demir

Buttons: Create Transportation, Back

Figure12: Create new transportation page

Update Transportation

Arrival Date: 2021-06-06

Cloth Id	Cloth Type	Cloth Size	TransporterId
3	Shirt	M	1
8	Pants	L	1

Update Route

Back

Figure13: Update Transportation page

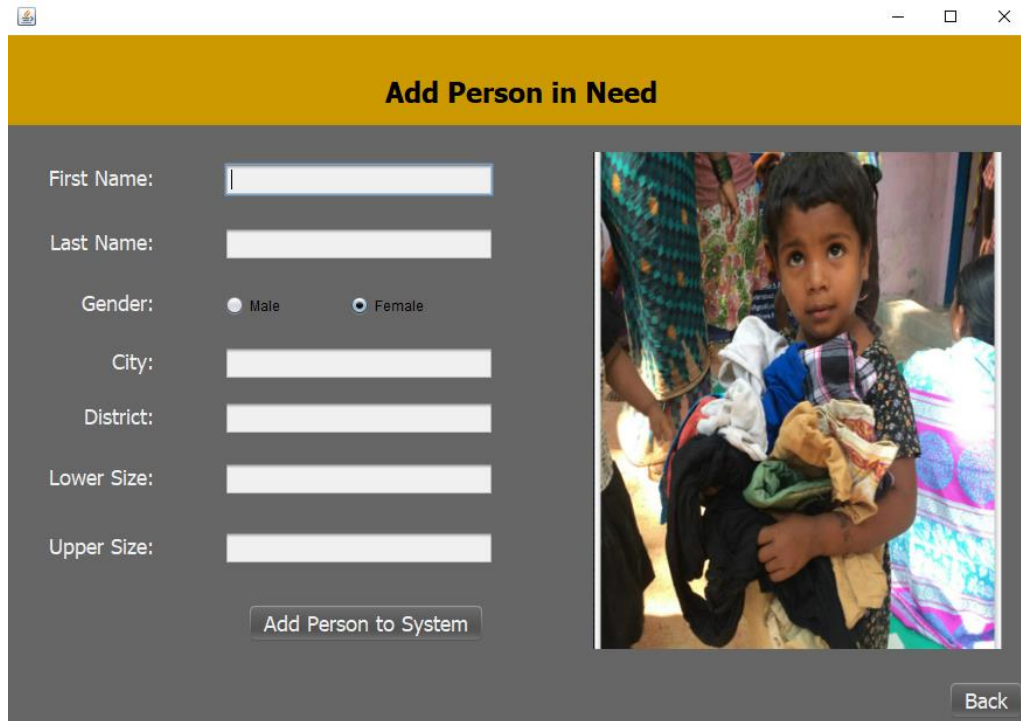
Add Transporter

First Name:
Last Name:
Gender:
☐ Male
☒ Female
City:
District:

Add Transporter

Back

Figure14: Create Transporter page



The screenshot shows a web application window titled "Add Person in Need". The form is set against a dark grey background with a yellow header bar. On the left, there are input fields for "First Name:", "Last Name:", "City:", "District:", "Lower Size:", and "Upper Size:". The "Gender:" field has two radio buttons, "Male" and "Female", with "Female" selected. Below these fields is a button labeled "Add Person to System". On the right side of the form is a photograph of a young child holding a bundle of clothes. At the bottom right of the form is a "Back" button.

*Figure15: Create Person In Need page*

## SUMMARY

This report is based on the project that creates a database that has been established to ensure that the connection between people who want to give their clothes to help and those who need them is faster and more effective. In the realization of this project, SQL to create the connection between people and systems, Flutter for the creation of the website, and programs such as Lucid App and Draw.io as additional resources were utilized. In the following parts of the report, general description, requirement analysis, specifications, IDEs, UML, High-level diagram, E-R diagram, design philosophy, cardinalities, and user permission of the project are mentioned in detail.