BLG 317E

SECOND HAND SHOPPING WEB SITE

Name	Ramazan Yetişmiş
Student No.	150190708
Department	Computer Engineering
Date	February 7, 2021

Contents

1. Motivation and Requirements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

1.1 Motivation

Nowadays we all are handling our daily-life via the internet of things. In addition, ecommerce has been improving with the internet. So, everything becomes a part of ecommerce. As a member of ITU, we all have some items that we do not need anymore and we want to get rid of them easily. Of course, we can go and throw them away, instead of that we may sell them to the people who are in need. In this way, we can gain money and space, also the buyer of the item will be profit by meeting his/her needs. My first goal is to create a website that allows the ITU students to sell or buy second-hand stuff. On the website, the students can make announcements about their stuff and easily place them, and they can see the other announcements on their profiles. In this way, we can reach the stuff we have been searching for. And these website users are all ITU members so this helps us to see the secondhand stuff before buying it. My other goal is security, as I mentioned above the participant of the website, are restricted as ITU members, so this makes users more careful because users' identities will not be encapsulated. So, they have to sell real stuff or they have to pay for what they buy. My third goal is easy access to the stuff, currently, the members are handling this type of operations via small communication groups, so we have to find one of the small groups and get contact with the person. This is not a very easy process, but as I mentioned with a user account on this website, we have easy access to every buyer and seller. My last goal is that users can comment and rate the other users according to their reliability, punctuation of the delivery, etc.

1.2 Requirements

- The tool helps users to create an account.
- The tool helps users to add the item that will be sold.
- The tool helps users to create/remove an announcement about the item.
- Users can list and manipulate his/her announcements.
- Users can see other announcements and list them according to their type.
- Users can comment and rate other users according to their experiences.
- All users can see his/her previous, expired or sold announcements.
- There will be no selling through the tool, users will communicate with each other individually via the user information part.
- On the main page, the items that are not sold and not the current users have, will be seen.
- The users can remove the announcement from the tool when the selling process occurs.

2. Conceptual Database Design

2.1 Data requirements

User Table

- Each user can have zero or more products [Product id foreign Key]
- Each user has user id [primary key] (INT)
- Each user has a name (VARCHAR)
- Each user has a Surname (VARCHAR)
- Each user has an email address (VARCHAR)
- Each user has a password (VARCHAR)
- Each user has a phone number (VARCHAR)
- One user can send zero or many purchase requests for different products (I mean a user can not send 2 or more purchased request for the same product)
- One user can make zero or many comments for different products.

Product Table

- Each product has a product id [primary key] (INT)
- A product can have one and only one user. User id [Foreign key]
- Each product has a name (VARCHAR)
- Each product has a price (FLOAT)
- Each product has a description (VARCHAR)
- Each product has a is sold variable (TINYINT)
- There can be one and only one type for each product. Type id [foreign key]

Type Table

- Each type has type an id [primary key] (INT)
- Each type has a category (VARCHAR)

Purchased Request Table

- Each purchased request has a sender id [foreign key] (INT)
- Each purchased request has a product id [foreign key] (INT)
- Each purchased request has status id (BOOL)
- For one product there can be send 0 or many requests.

Photo Table

- Each Photo has a photo id [primary key] (INT)
- Each Photo has photo path (VARCHAR)
- Each Photo has photo order (TINYINT)
- Each photo has a product id [Foreign key]
- A product can have zero or many photos.

Comment Table

- Each comment has a user id [foreign key] (INT)
- Each comment has a product id [foreign key] (INT)
- The above two foreign id is a composite key for this table.
- A product can have zero or many comments
- Each comment table has a comment that can be null (VARCHAR)
- Each comment table has a rating that can be null (TINYINT)

2.2 E/R Design

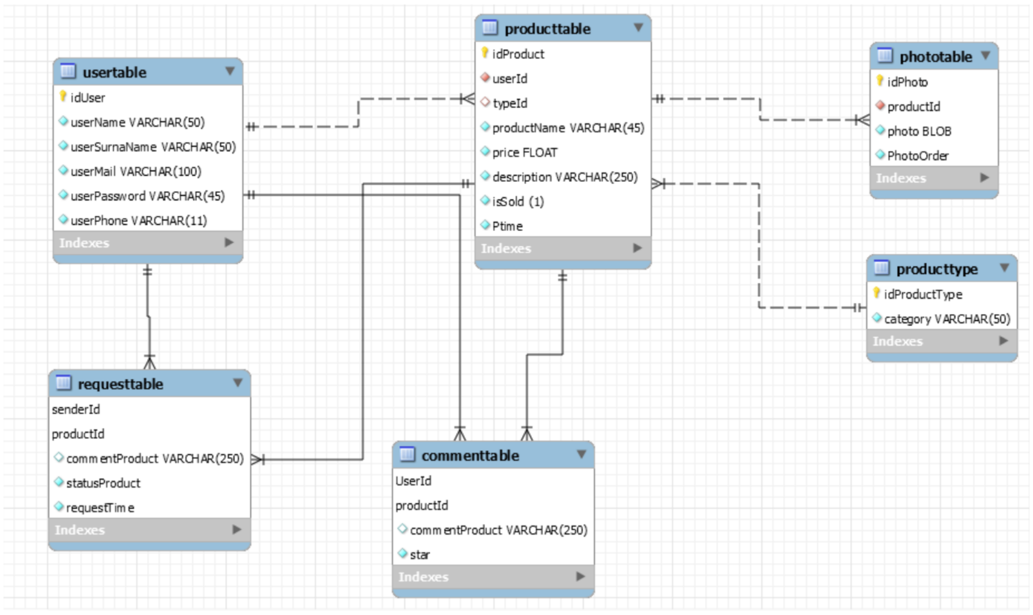


Figure 2.1: Er design

3. Logical Database Design

3.1 DDL Statements

```
CREATE TABLE ‘usertable‘ (
      ‘idUser‘ int NOT NULL AUTO_INCREMENT,
      ‘userName‘ varchar(50) NOT NULL,
      ‘userSurnaName‘ varchar(50) NOT NULL,
      ‘userMail‘ varchar(100) NOT NULL,
      ‘userPassword‘ varchar(200) NOT NULL,
      ‘userPhone‘ varchar(11) DEFAULT NULL,
      PRIMARY KEY (‘idUser‘),
      UNIQUE KEY ‘userMail_UNIQUE‘ (‘userMail‘),
      UNIQUE KEY ‘idUser_UNIQUE‘ (‘idUser‘)
    );

    CREATE TABLE ‘producttype‘ (
      ‘idProductType‘ int NOT NULL AUTO_INCREMENT,
      ‘category‘ varchar(50) NOT NULL,
      PRIMARY KEY (‘idProductType‘)
    );

    CREATE TABLE ‘producttable‘ (
      ‘idProduct‘ int NOT NULL AUTO_INCREMENT,
      ‘userId‘ int NOT NULL,
      ‘typeId‘ int DEFAULT NULL,
      ‘productName‘ varchar(45) NOT NULL,
      ‘price‘ float NOT NULL,
      ‘description‘ varchar(250) NOT NULL,
      ‘isSold‘ tinyint(1) NOT NULL,
      ‘Ptime‘ datetime NOT NULL,
      PRIMARY KEY (‘idProduct‘),
      UNIQUE KEY ‘idProduct_UNIQUE‘ (‘idProduct‘),
      KEY ‘userIdFk‘ (‘userId‘),
      KEY ‘typeIdFk‘ (‘typeId‘),
      CONSTRAINT ‘userIdFk‘ FOREIGN KEY (‘userId‘) REFERENCES ‘usertable‘
      (‘idUser‘) ON DELETE CASCADE ON UPDATE CASCADE
    );

    CREATE TABLE ‘phototable‘ (
      ‘idPhoto‘ int NOT NULL AUTO_INCREMENT,
      ‘productId‘ int NOT NULL,
      ‘photo‘ longblob NOT NULL,
      ‘PhotoOrder‘ int NOT NULL,
      PRIMARY KEY (‘idPhoto‘),
      KEY ‘productId‘ (‘productId‘),
      CONSTRAINT ‘phototable_ibfk_1‘ FOREIGN KEY (‘productId‘) REFERENCES
      ‘producttable‘ (‘idProduct‘) ON DELETE CASCADE ON UPDATE CASCADE
    );
    CREATE TABLE ‘requesttable‘ (
      ‘senderId‘ int NOT NULL,
      ‘productId‘ int NOT NULL,
      ‘statusProduct‘ int NOT NULL,
      ‘requestTime‘ datetime NOT NULL,
      PRIMARY KEY (‘senderId‘,‘productId‘),
      KEY ‘prIdFk‘ (‘productId‘),
      CONSTRAINT ‘prIdFk‘ FOREIGN KEY (‘productId‘) REFERENCES ‘producttable‘
      (‘idProduct‘) ON DELETE CASCADE ON UPDATE CASCADE,
      CONSTRAINT ‘usIdFk‘ FOREIGN KEY (‘senderId‘) REFERENCES ‘usertable‘
      (‘idUser‘) ON DELETE CASCADE ON UPDATE CASCADE
    );

CREATE TABLE ‘commenttable‘ (
    ‘UserId‘ int NOT NULL,
    ‘productId‘ int NOT NULL,
    ‘commentProduct‘ varchar(250) DEFAULT NULL,
    ‘star‘ double NOT NULL,
    ‘Ptime‘ datetime NOT NULL,
    PRIMARY KEY (‘UserId‘,‘productId‘),
    KEY ‘pFk‘ (‘productId‘),
    CONSTRAINT ‘pFk‘ FOREIGN KEY (‘productId‘) REFERENCES ‘producttable‘
    (‘idProduct‘) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT ‘uFk‘ FOREIGN KEY (‘UserId‘) REFERENCES‘usertable‘ (‘idUser‘)
    ON DELETE CASCADE ON UPDATE CASCADE
    ) ;
```

3.2 Tables

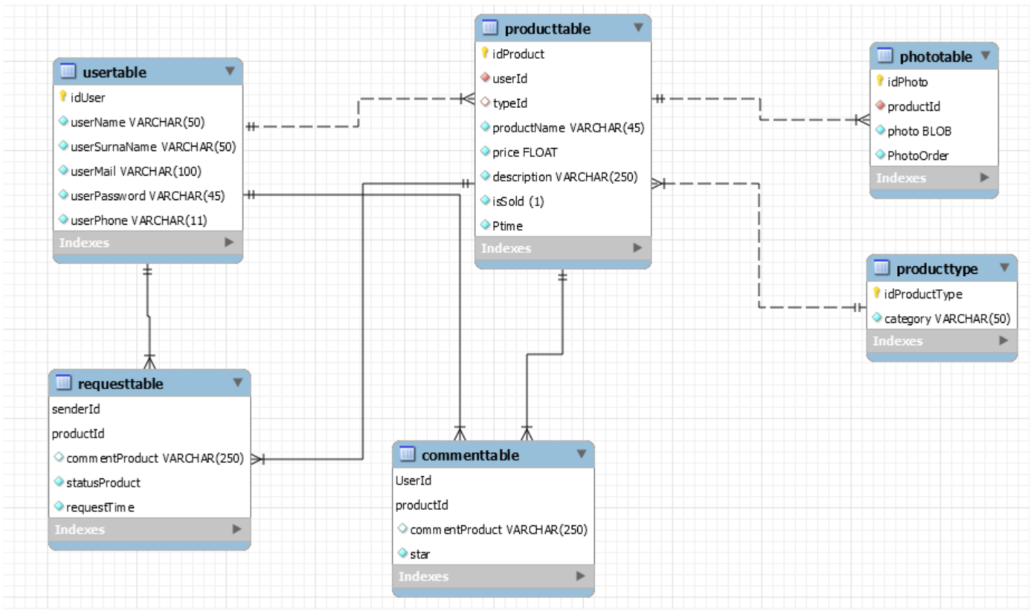


Figure 3.1: Tables

4. Database Normalization

4.1 Functional Dependencies

userTable

```
Primary Key: idUser
{idUser}-> {userName}
{idUser}-> {userSurname}
{idUser}-> {userMail}
{idUser}-> {userPassword}
{idUser}-> {userPhone}
{userMail}-> {userId}
```

Requesttable

```
Primary Key: {senderId,productId}
{senderId,productId} -> {statusProduct}
{senderId,productId} -> {requestTime}
```

commentTables

```
Primary Key: {senderId,productId}
{senderId,productId} -> {commentProduct }
{senderId,productId} -> {star}
```

productTable

```
Primary key:idProduct
Foreign Key: {userId, typeId}
{idProduct}->{userId}
{idProduct}->{typeId}
{idProduct}->{productName}
{idProduct}->{price}
{idProduct}->{description}
{idProduct}->{isSold}
{idProduct}->{Ptime}
```

photoTable

```
Primary Key:idPhoto
Foreign Key: productid
{idPhoto}-> {productId}
{idPhoto}->{photo}
{idPhoto}->{PhotoOrder}
```

productTypeTable

```
Primary Key:idProductType
{idProductType }->{category }
{category}->{idProductType }
```

4.2 Normalization

1NF: attribute values have to be atomic (no Multi valued attribute)

2NF: every non-key attribute depends on the primary key (no partial dependency)

3NF: non-key attributes do not depend on any attributes other than the primary key (no Transitive dependency)

BCNF: all functional dependencies must be on candidate keys (strict form of 3NF)

UserTable :

This table is appropriate for 1NF, because there are no attributes that contains more than 1 attributes.
$\{$idUser$\} \rightarrow \{$userName$\}$
$\{$idUser$\} \rightarrow \{$userSurname$\}$
every attribute has single value. So, it is atomic. This table also not violates 2NF because every non-key attribute depends on primary key, I mean no subset of the key can determine non-key attributes. For this table I have only one key so subset of the key is itself. The one and only one key determines the attributes. So, for the 3NF we have to look transitive dependency, this table does not contain transitive dependency. I mean non-key attributes do not depend on any attributes other than the primary key. For BCNF all left part of the dependency is key attributes so this Table is in BCNF form. (A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey)

Requesttable:

This table is appropriate for 1NF, because there are no attributes that contains more than 1 attributes.
$\{$senderId,productId$\} \rightarrow \{$statusProduct$\}$
$\{$senderId,productId$\} \rightarrow \{$requestTime$\}$ every attribute has single value. So, it is atomic. This table also not violates 2NF because every non-key attribute depends on primary key, I mean no subset of the key can determine non-key attributes.
For this table I have two candidate keys so. But with subset of the key, none of the non-key

attribute cannot be determined.For example
{senderId}→X {statusProduct}
{senderId} →X { requestTime }
So, for the 3NF we have to look transitive dependency, this table does not contain transitive dependency. I mean non-key attributes do not depend on any attributes other than the primary key.
For BCNF all left part of the dependency is key attributes so this Table is in BCNF form. (A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey)

commentTable:

This table is appropriate for 1NF, because there are no attributes that contains more than 1 attributes.
{senderId,productId}→{commentProduct}
{senderId,productId} → {star} every attribute has single value. So, it is atomic. This table also not violates 2NF because every non-key attribute depends on primary key, I mean no subset of the key can determine non-key attributes.
For this table I have two candidate keys so. But with subset of the key, none of the non-key attribute cannot be determined.For example
{senderId}→X {commentProduct }
{senderId}→X {star} So, for the 3NF we have to look transitive dependency, this table does not contain transitive dependency. I mean non-key attributes do not depend on any attributes other than the primary key.
For BCNF all left part of the dependency is key attributes so this Table is in BCNF form.(A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey)

productTable:

This table is appropriate for 1NF, because there are no attributes that contains more than 1 attributes.
{idProduct}→{userId}
{idProduct}→{typeId} every attribute has single value. So, it is atomic. This table also not violates 2NF because every non-key attribute depends on primary key, I mean no subset of the key can determine non-key attributes.
For this table I have only one key so subset of the key is itself. The one and only one key determines the attributes.
So, for the 3NF we have to look transitive dependency, this table does not contain transitive dependency. I mean non-key attributes do not depend on any attributes other than the primary key.
For BCNF all left part of the dependency is key attributes so this Table is in BCNF form.(A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey)

photoTable:

This table is appropriate for 1NF, because there are no attributes that contains more than 1 attributes.
{idPhoto}→ {productId}
{idPhoto}→{photo} every attribute has single value. So, it is atomic. This table also not violates 2NF because every non-key attribute depends on primary key, I mean no subset of the key can determine non-key attributes.

For this table I have only one key so subset of the key is itself. The one and only one key determines the attributes.
So, for the 3NF we have to look transitive dependency, this table does not contain transitive dependency. I mean non-key attributes do not depend on any attributes other than the primary key.
For BCNF all left part of the dependency is key attributes so this Table is in BCNF form. (A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey)

productTypeTable

This table is appropriate for 1NF, because there are no attributes that contains more than 1 attributes.
{idProductType}→ {category} every attribute has single value. So, it is atomic.
This table also not violates 2NF because every non-key attribute depends on primary key, I mean no subset of the key can determine non-key attributes.
For this table I have only one key so subset of the key is itself. The one and only one key determines the attributes.
So, for the 3NF we have to look transitive dependency, this table does not contain transitive dependency. I mean non-key attributes do not depend on any attributes other than the primary key.
For BCNF all left part of the dependency is key attributes so this Table is in BCNF form.(A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey)

5. Application Design and Implementation

5.1 Technical Manual

5.1.1 Architecture of your application

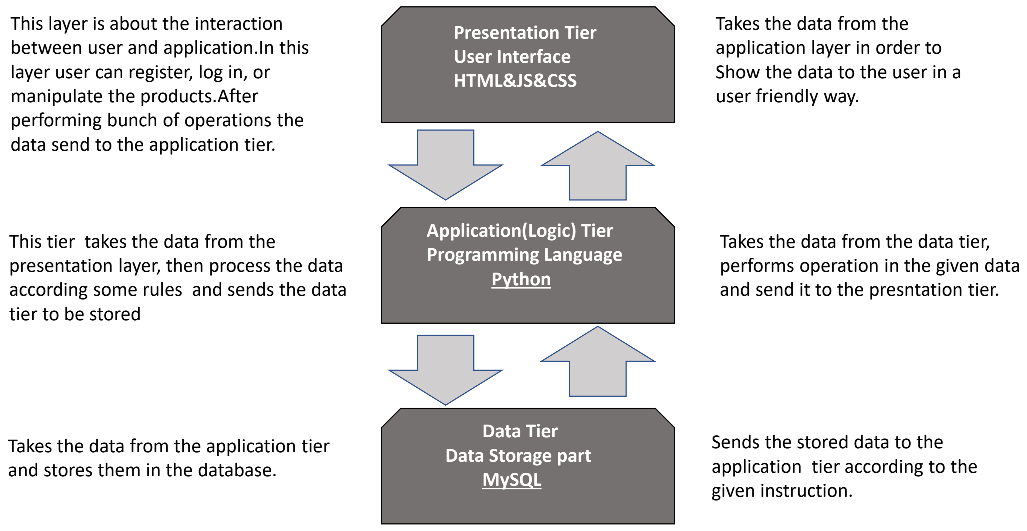

This layer is about the interaction between user and application.In this layer user can register, log in, or manipulate the products.After performing bunch of operations the data send to the application tier.
Presentation Tier
User Interface
HTML&JS&CSS
Takes the data from the application layer in order to Show the data to the user in a user friendly way.
This tier takes the data from the presentation layer, then process the data according some rules and sends the data tier to be stored
Application(Logic) Tier
Programming Language
Python
Takes the data from the data tier, performs operation in the given data and send it to the presntation tier.
Takes the data from the application tier and stores them in the database.
Data Tier
Data Storage part
MySQL
Sends the stored data to the application tier according to the given instruction.

Figure 5.1: Tiers of the application

5.1.2 List of SQL queries

```
Query:
  'SELECT * FROM usertable WHERE userMail = % s AND userPassword = % s'

Semantics:
    Gets the user in formations according to given
    mail and password

Query:
    'SELECT * FROM usertable  WHERE userMail = % s'
Semantics:
    lists the user information according to given mail.

Query:
    'SELECT * FROM usertable  WHERE idUser = % s'
```

```
Semantics:
    lists the user information according to given id.

Query:
    'INSERT INTO usertable VALUES (NULL,% s,%s, % s, % s,%s)'
Semantics:
    creates new user according to given parameters.

Query:
    'SELECT idProduct, productName, price FROM producttable
     WHERE userId = % s'
Semantics:
    List the given user's products info (name and id).
Query:
    '''SELECT userId,productName, price,description,isSold,category,
    idProductType FROM producttable
    INNER JOIN producttype ON typeId=idProductType
    WHERE idProduct=% s'''

Semantics:
   List the particular given product information and type of the product

Query:
   '''SELECT * FROM producttable
    Inner join phototable ON
    producttable.idProduct =phototable.productId
    where idProduct = % s'''

Semantics:
   Get the particular product & all its photos.

Query:
   '''SELECT * FROM producttable
    INNER JOIN usertable ON producttable.userId= usertable.idUser
    Inner join phototable ON producttable.idProduct =phototable.productId
    where idProduct = % s'''

Semantics:
   List the  the user&photos of the given product.

Query:
   '''SELECT   userId,idProduct,description, productName, price,photo
   FROM producttable
   inner join phototable ON idProduct=phototable.productId
   where PhotoOrder =1 and userId <> %s and issold=0
   group by userId,idProduct  '''

Semantics:
   List the  products&photo with these conditions:
   1)Not current users product
   2)Not sold
   and the first photo of the product.

Query:
   ''' SELECT   userId,idProduct,description, productName, price,photo
   FROM producttable
   inner join phototable ON idProduct=phototable.productId
   where PhotoOrder =1 and userId <> %s and issold=0  and typeId =%s
   group by userId, idProduct '''

Semantics:
   Filters the product according to given product type(Search bar)

Query:
   'DELETE FROM producttable WHERE idProduct = % s'

Semantics:
   Removes the current user's given product from

Query:
   'INSERT INTO producttable VALUES (NULL,% s,%s, % s, % s,% s, %s,%s)'

Semantics:
   Add new product to the current user

Query:
   'INSERT INTO requesttable VALUES (%s,% s,%s, % s)'

Semantics:
   Send new request to the product

Query:
   'SELECT * FROM requesttable INNER JOIN producttable ON
   productId= producttable.idProduct where senderId =%s'

Semantics:
    List the items that the current user sent request .

Query:
   '''SELECT senderId,idProduct,productName,price,description,userName,
   idUser,userSurnaName,isSold,userMail,userPhone FROM requesttable
   INNER JOIN producttable ON productId= producttable.idProduct
   INNER JOIN usertable ON senderId=idUser
   where productId in (select idProduct from producttable
   where userId=%s )'''
```

```
Semantics:
    List the items and buyers that request sent to the current
    user’s products


Query:
   ’UPDATE requesttable SET statusProduct = %s WHERE senderId =%s
   and productId= %s’

Semantics:
    Help user to accept or reject the given request.


Query:
   "SELECT * FROM producttype"

Semantics:
    List all types in the website


Query:
   ’UPDATE  producttable SET isSold = %s  where idProduct= %s’

Semantics:
    Change the status of the product (sold or available)

Query:
   ’UPDATE requesttable SET statusProduct = %s WHERE senderId = %s
   and productId = %s’
Query:
   ’UPDATE requesttable SET statusProduct = %s WHERE senderId <> %s
   and productId = %s’

Semantics:
    With the combination of these 2 queries we can sell the product
    to the given user.

Query:
   ’INSERT INTO commenttable VALUES (%s,% s,%s, % s ,%s)’

Semantics:
    Make comment to the product

Query:
   ’’’SELECT producttable.productName,producttable.UserId,productId
   ,commentProduct,star,commenttable.Ptime,userName,userSurnaName
   FROM commenttable
   INNER JOIN usertable ON commenttable.UserId= usertable.idUser
   INNER JOIN producttable ON productId=producttable.idProduct
   where productId in (select idProduct from producttable
   where producttable.userId = %s)'''

Semantics:
    List all the comments that is made to the current user's
    products

Query:
   "SELECT commentProduct,star,Ptime FROM commenttable where UserId =%s
   and productId=%s"

Semantics:
    List my comment that is made to the current product

Query:
   'SELECT count(photo) FROM phototable where productId=%s'

Semantics:
    Number of photos the given product has.

Query:
   'INSERT INTO phototable VALUES (NULL,%s,%s, %s)'

Semantics:
    Add new photo to the product.

Query:
   'SELECT idPhoto,photo FROM phototable where productId= %s'

Semantics:
    List all the photos of the product

Query:
   'SELECT statusProduct,count(*) as Nstat  FROM requesttable
   where senderId=%s group by statusProduct'

Semantics:
    Groups the status and list the number of status in each
    group accordingly.

Query:
   "UPDATE  producttable set productName=%s , price=%s ,
   description=%s ,typeId=%s where idProduct =%s"

Semantics:
    Change the current product information.
```

```
Query:
   "UPDATE usertable SET userName =%s , userSurnaName=%s ,
   userMail=%s ,userPhone=%s where idUser= %s"

Semantics:
    Change the current user information.

Query:
   "DELETE FROM usertable where idUser =%s"

Semantics:
    Remove current user’s account.

Query:
   "DELETE FROM requesttable where senderId =%s and productId=%s;"

Semantics:
    Remove current user’s request for the chosen produt.

Query:
   ’’’SELECT * FROM producttable
    INNER JOIN usertable ON producttable.userId= usertable.idUser
    Inner Join producttype ON producttype.idProductType=producttable.typeId
    where idProduct = % s’’’

Semantics:
   List one product with its user.

Query:
   ’’’UPDATE commenttable SET commentProduct= %s , star=%s , Ptime=%s
   where UserId=%s and productId=%s’’’

Semantics:
   Change the comment of the product.

Query:
   "DELETE FROM commenttable where UserId =%s and productId=%s;"

Semantics:
   Remove the comment from the product.


Query:
   "UPDATE usertable SET userName =%s , userSurnaName=%s ,
   userMail=%s ,userPhone=%s,userPassword= %s where idUser= %s"

Semantics:
   Change the user information.
```

5.1.3 Programming languages and DBMS

Programming languages that are used for this project are Python and Java Script, and used DBM is MySQL.

5.1.4 Data size

I did not outsourced any data for this application.I manually recorded the data and it is adequate for the demonstration purposes.

5.2 User Manual

In this part I will talk about the properties of the web-page.

Register

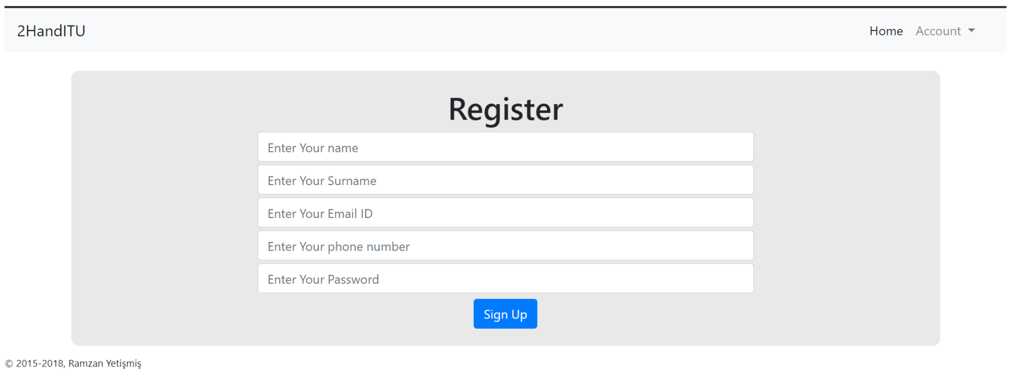
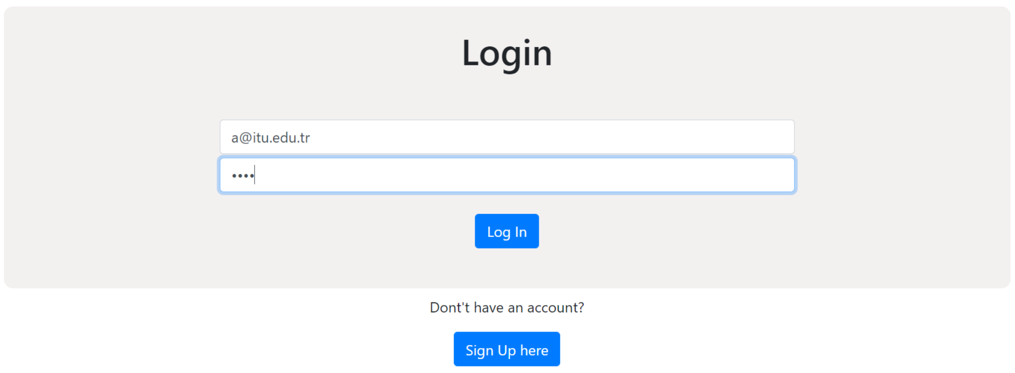


Figure 5.2: Register page

In order to register to the site you have to give your ITU mail and all the information are necessary.

Login


Login

a@itu.edu.tr

••••

Log In

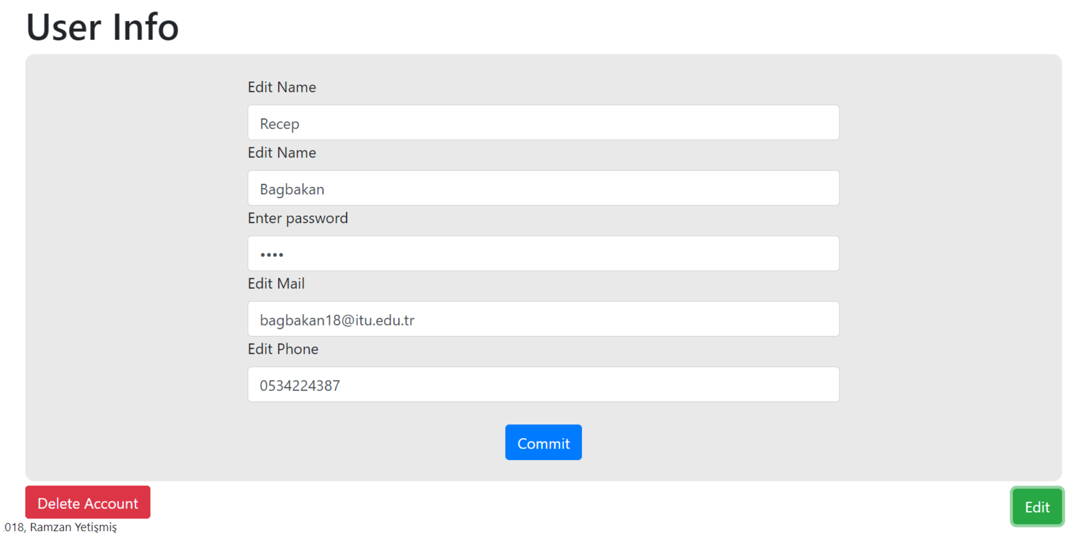
Don't have an account?

Sign Up here

Figure 5.3: Login page

In this page you can only log in with an existing user information if it is not exist it will return warning message.

User Page


User Info

Edit Name

Recep

Edit Name

Bagbakan

Enter password

••••

Edit Mail

bagbakan18@itu.edu.tr

Edit Phone

0534224387

Commit

Delete Account

Edit

018, Ramzan Yetişmiş

Figure 5.4: This age shows user information

Here you can change user settings and if you want you can delete your account.

Home page

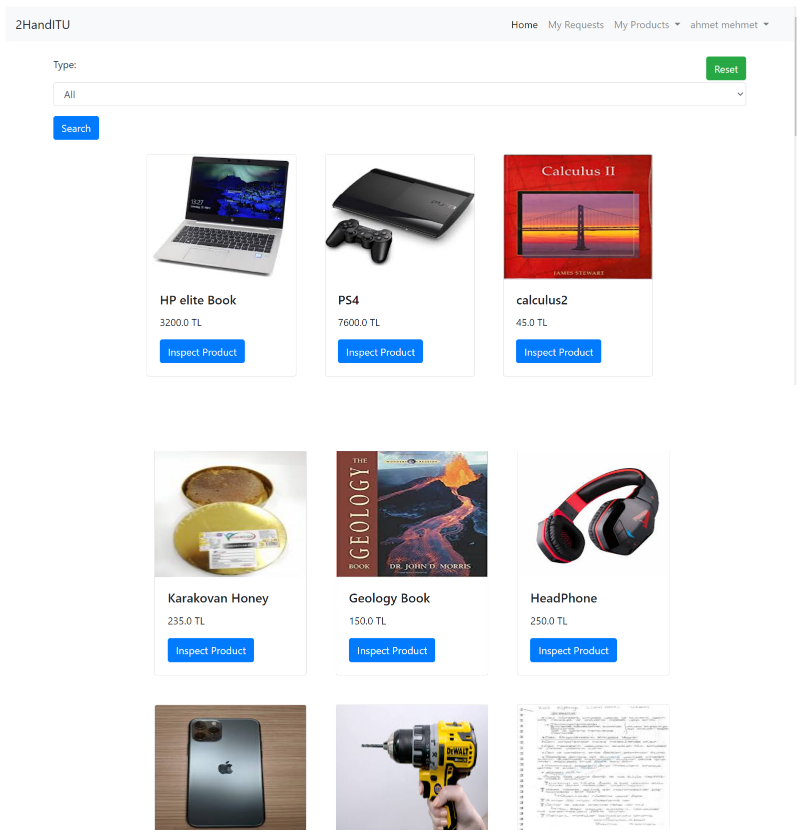


Figure 5.5: Home page

In the main page the products that are available are listed.You can request the product if you want.

Search Bar

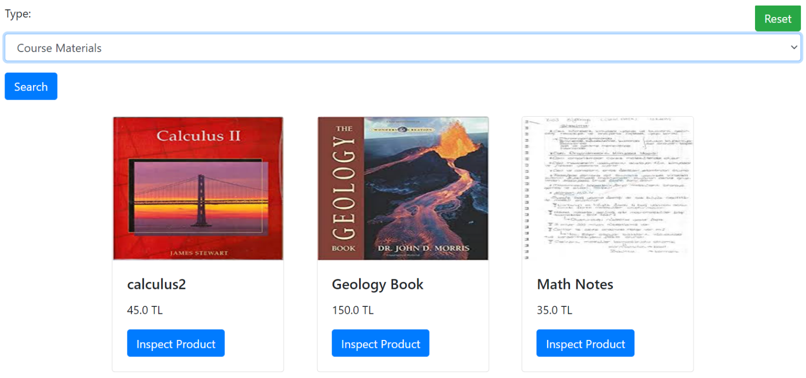
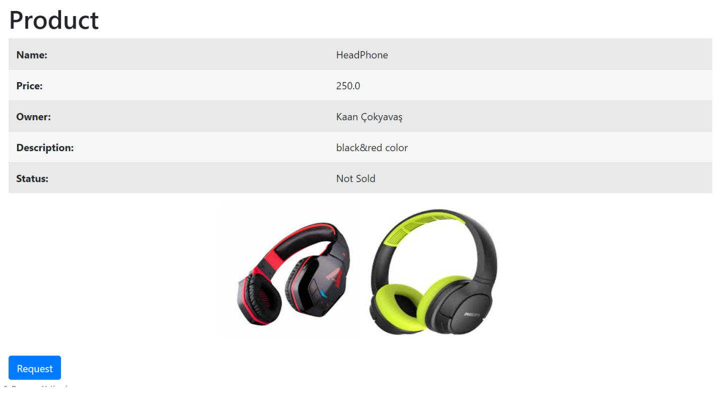


Figure 5.6: Home page

Here I selected the **"Course Materials"** and all products that are related are listed in the Home page.Next I will click **"inspect product"**.

Send Request



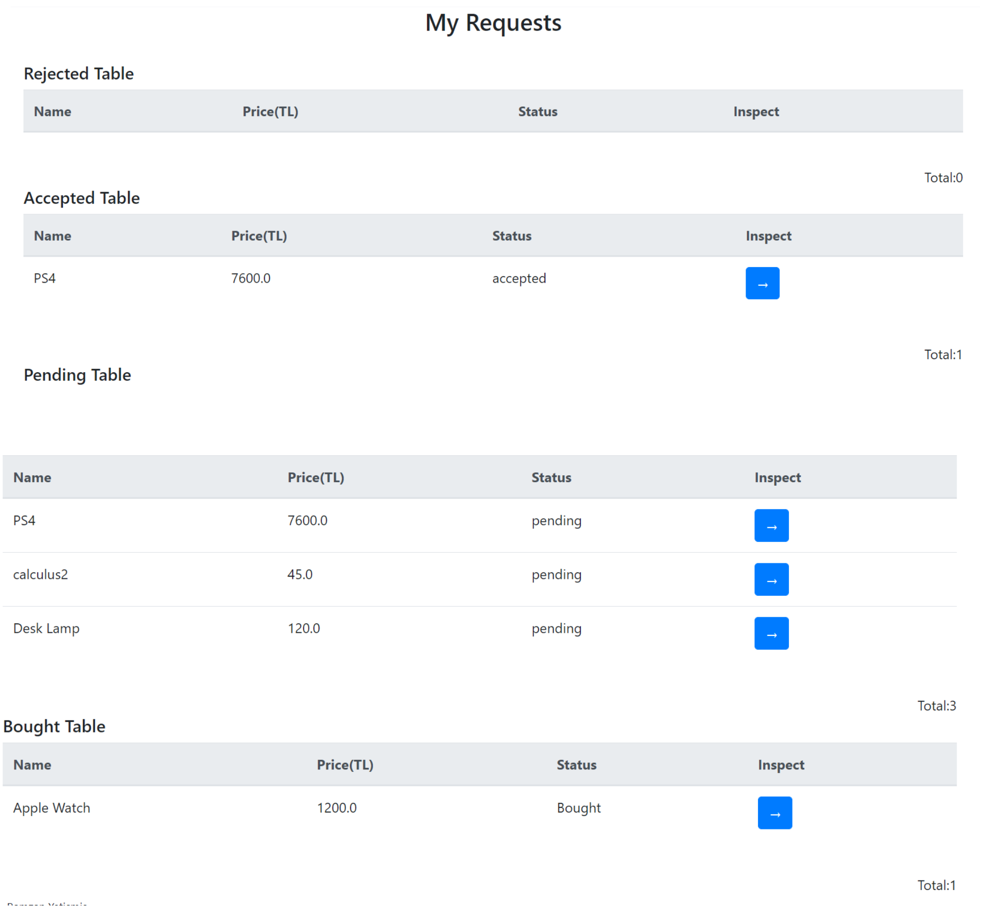
Product

Name:	HeadPhone
Price:	250.0
Owner:	Kaan Çokyavaş
Description:	black&red color
Status:	Not Sold

Figure 5.7: Request Product Page

Here you can see the owner name,description and whole photos of the product.But the user communication information hidden due to security issues.Then you can send request if you like the product.

MY REQUESTS



My Requests

Rejected Table

Name	Price(TL)	Status	Inspect

Total:0

Accepted Table

Name	Price(TL)	Status	Inspect
PS4	7600.0	accepted	→

Total:1

Pending Table

Name	Price(TL)	Status	Inspect
PS4	7600.0	pending	→
calculus2	45.0	pending	→
Desk Lamp	120.0	pending	→

Total:3

Bought Table

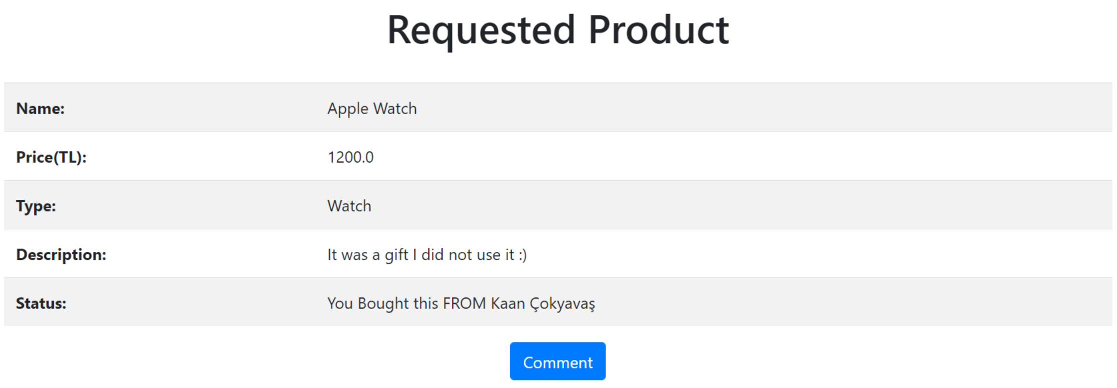
Name	Price(TL)	Status	Inspect
Apple Watch	1200.0	Bought	→

Total:1

Figure 5.8: My requests page

In this page you can see the last situation of your requests,such as if it rejected it will be in the rejected table etc.Then by clicking the arrow button you can go to **that product**.

Requested Product

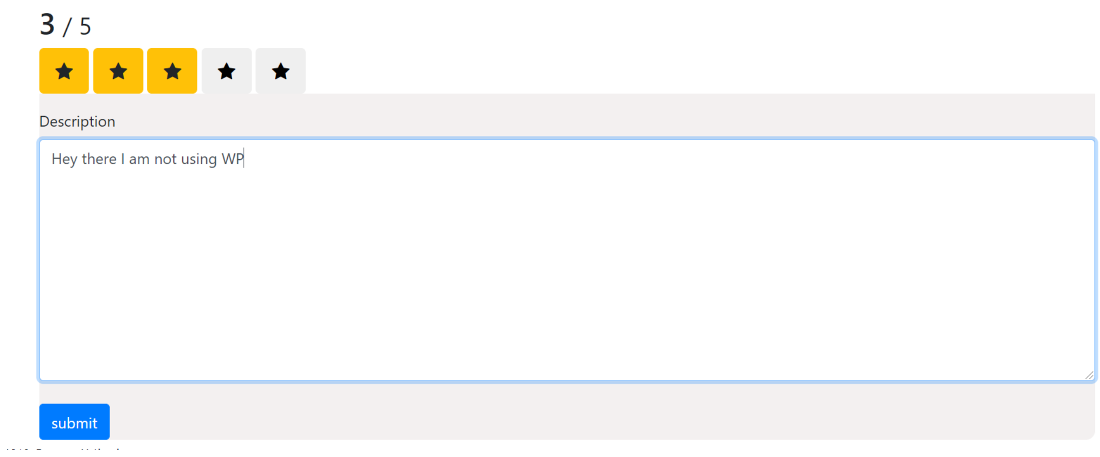

Requested Product

Name:	Apple Watch
Price(TL):	1200.0
Type:	Watch
Description:	It was a gift I did not use it :)
Status:	You Bought this FROM Kaan Çokyavaş

Figure 5.9: Follow the status of the product

In this page if your request is accepted you can see the owner's private information.If not it is blocked.And if you buy the product then you can comment it.

Comment Page



Hey there I am not using WP

Figure 5.10: Comment the product

You can comment the product and then change it later.

List My Products

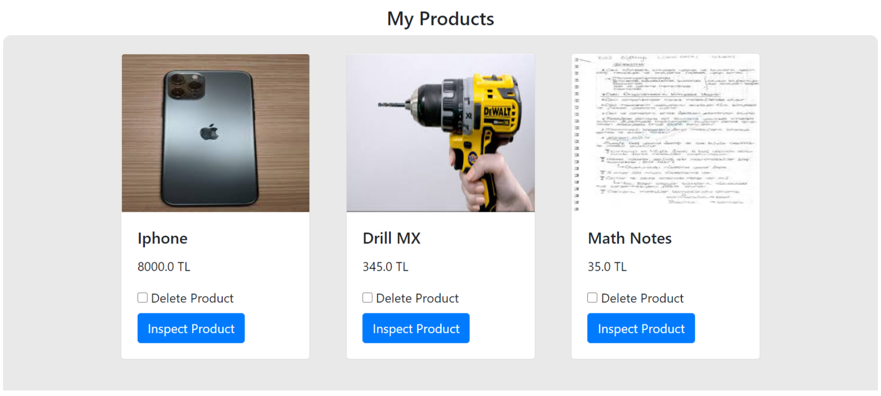
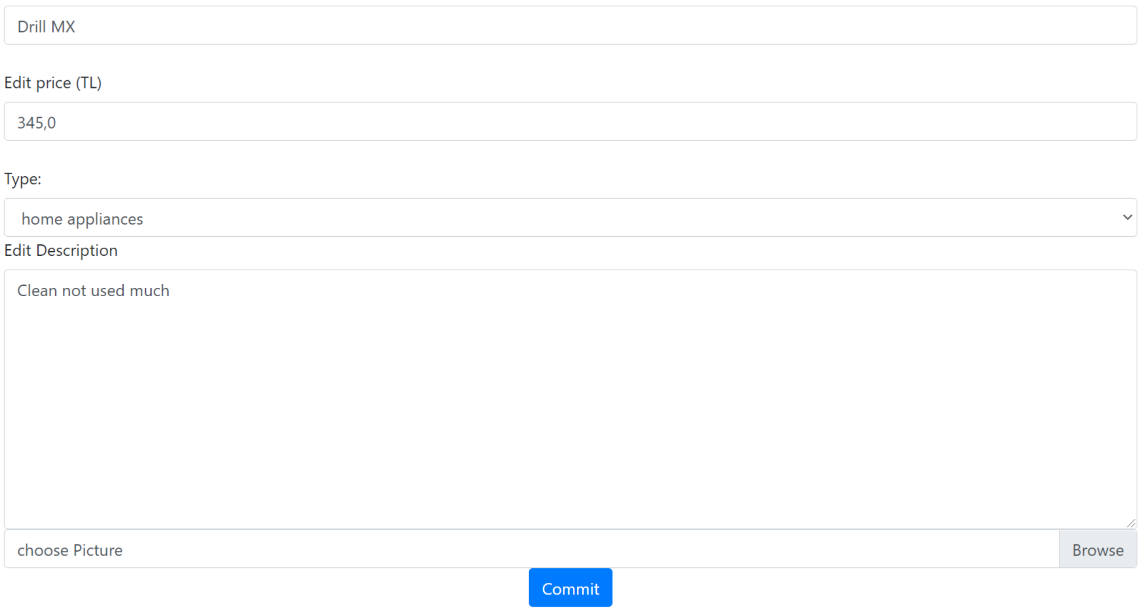


Figure 5.11: list my product

You can see the products and by clicking **delete** you can delete the item. And if you click **inspect** you can update the product.

Edit Product



Drill MX

Edit price (TL)

345,0

Type:

home appliances

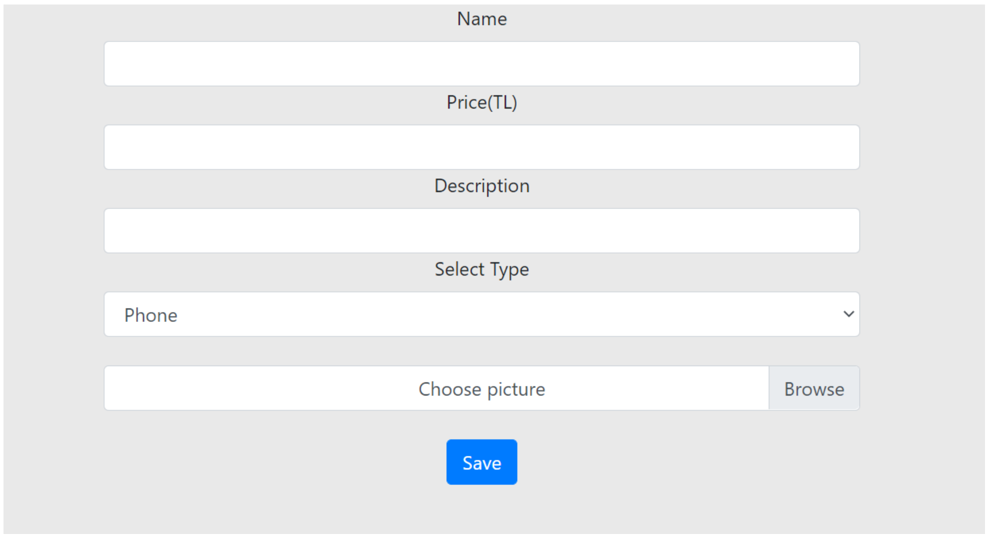
Edit Description

Clean not used much

Figure 5.12: Edit the product

You can change the information and add new photos to your product.

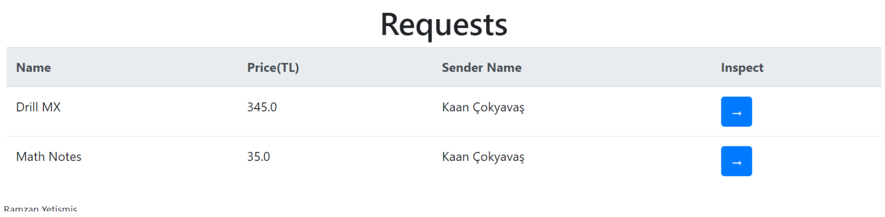
Add Product



Select Type: Phone

Figure 5.13: Add a product

List Requests



Requests

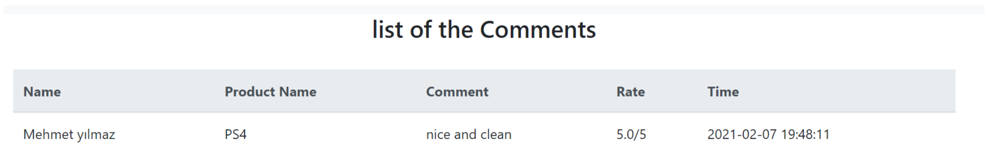
Name	Price(TL)	Sender Name	Inspect
Drill MX	345.0	Kaan Çokyavaş	→
Math Notes	35.0	Kaan Çokyavaş	→

Ramzan Yetismis

Figure 5.14: List requests

You see your products that are requested by another users.

List Comments



list of the Comments

Name	Product Name	Comment	Rate	Time
Mehmet yılmaz	PS4	nice and clean	5.0/5	2021-02-07 19:48:11

Figure 5.15: List comments to my products

Here there is only one comment but if you have multiple comments then you can see them too.

5.3 Installation Manual

1. Create a project application in heroku.
2. Create a file called **Procfile** and inside it write **"web: gunicorn "app:create_app()"**
3. Activate virtual environment
4. Log in heroku from terminal
5. Run the comment **"pip3 freeze > requirements.txt**
6. Run the comment **"git init"**
7. Run the comment **"heroku git: remote -a [nameOf the project]"**
8. Run the comment **"git add ."**
9. Run the comment **"git commit -am "your commit"**
10. Run the comment **"git push heroku master"**
11. Then go to the link given by heroku and your site is online.

References

https://www.w3schools.com
https://getbootstrap.com/docs/5.0/getting-started/introduction
https://flask.palletsprojects.com/en/1.1.x
https://jinja.palletsprojects.com/en/2.11.x
https://www.tutorialspoint.com/mysql
https://stackoverflow.com