CSE 222 PROJECT

GTU E-COMMERCE

TABLE OF CONTENTS

- Problem definition and explanation.
- Which data structures were used and why?
- Recursive solution methods and used sorting algorithms.
- Uml Diagrams and how to use object oriented programming principles?
- User scenarios.

1. PROBLEM DEFİNİTİON AND EXPLANATİON

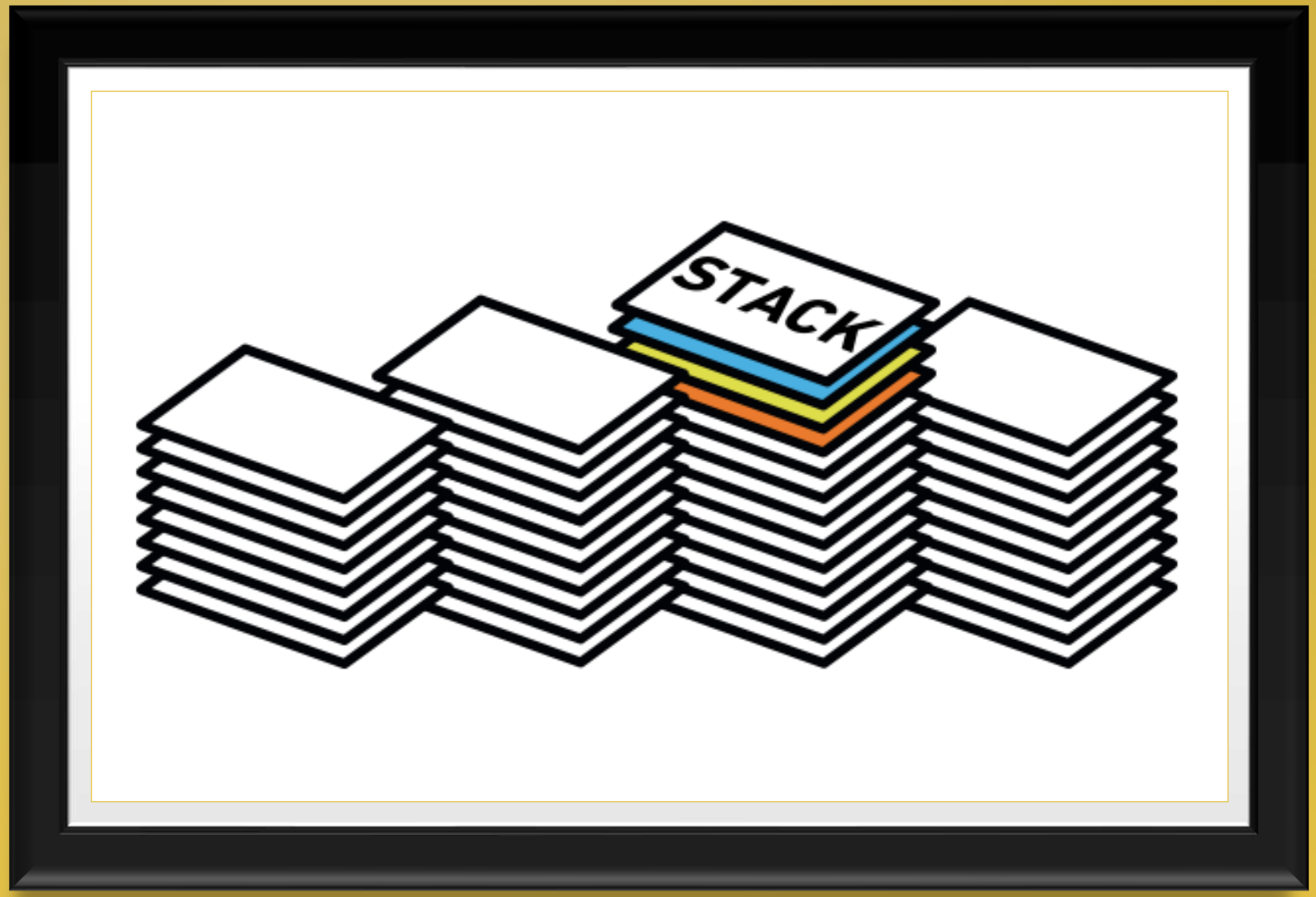
- An in-university e-commerce application. Why?
  - Electronic shopping systems are applications for all people and many examples have been applied. Gtu e-commerce is not for all people, but for a specific community of people.
  - A specific application for a more specific user group.

1. PROBLEM DEFİNİTİON AND EXPLANATİON

- Very convenient to use all data structures.
- When people shop at shopping sites, they may have a trust problem with the person or company that supplies the product they buy. As a solution to this problem, we think that intra-university trade in our project is more reliable for students and staff.
- Short delivery time.
- Because of the project addresses a specific group, and because this group has certain common characteristics, the product profile required by users will be the same. In this way a product will find buyers sooner.

2. WHICH DATA STRUCTURES WERE USED AND WHY?

- Stack data structure was used for user comments For the last comment to be at the top.

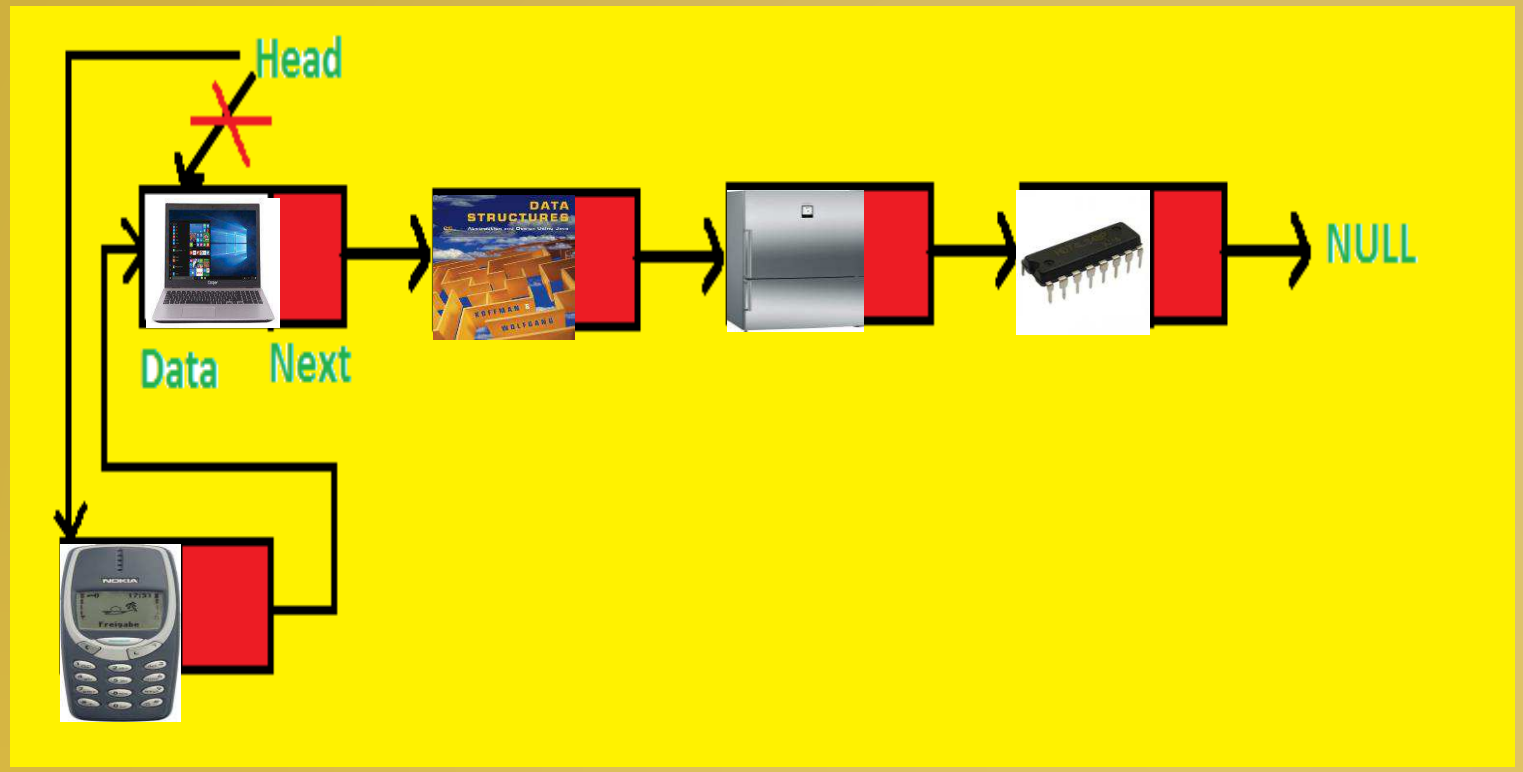


2. WHİCH DATA STRUCTURES WERE USED AND WHY?

- The ArrayList data structure is used to store information for products of the same type:

  →Dynamic structure.

  → Random access to elements using their index O(1) time



2. WHİCH DATA STRUCTURES WERE USED AND WHY?

- The linked list data structure was used to keep information of the products in the basket.

→Dynamic structure.

→add product to basket or remove product from basket begin / end constant O(1) time

Dont need shift all elements when removed product from interior basket

2. WHİCH DATA STRUCTURES WERE USED AND WHY?

- Map structure is used for keeping user names and passwords.
- Each user has a unique user name.
- In the map structure, key is unique e-mail and value is password <e-mail, password> and the user's password will be easily accessed and logged in.

2. WHİCH DATA STRUCTURES WERE USED AND WHY?

- To make product recommendations for related categories according to the products the user is looking at.

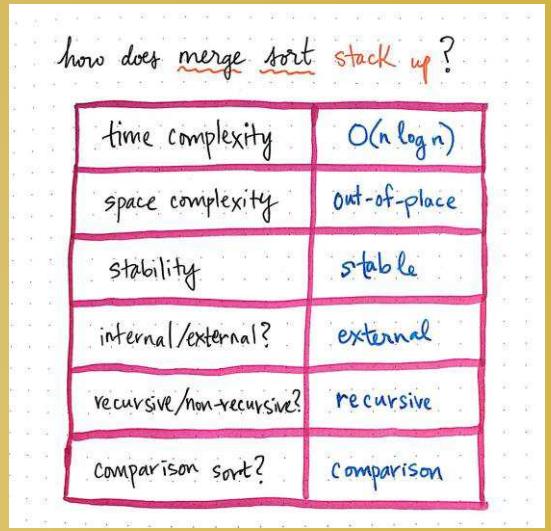
→According to the products in the category that the user is browsing, user category edges are created. These edges are initially given a maximum weight and as you travel in that category of products, the weight of that category is reduced. Shortest path is found and 3 random products are offered in that category.

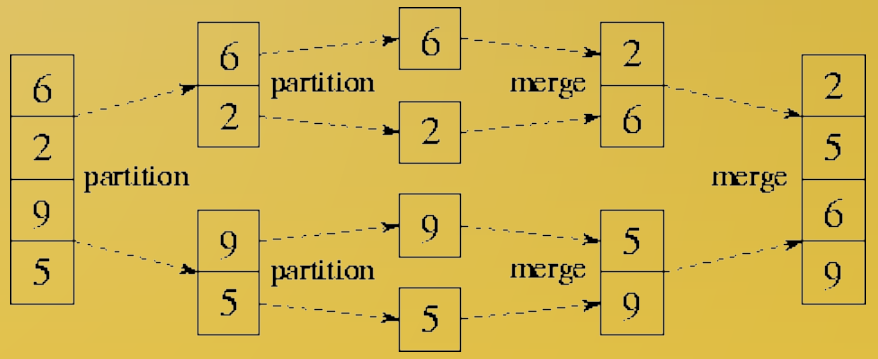
2. WHİCH DATA STRUCTURES WERE USED AND WHY?

- Avl tree is used for searching the products in the category.(Balanced) insead of Binary search tree because time comlexity always is O(logn).
- Since they're node based, rather than based on a contiguous chunk of memory, the cost of moving an element either into the collection or out of the collection is fast while still maintaining the sorted nature of the collection.

2. WHİCH DATA STRUCTURES WERE USED AND WHY?

- Sorting according to the prices of the products will use the merge sorting algorithm which is also a recursive sorting algorithm. Because the runtime is best - average and worse case nlogn. It is an efficient sorting algorithm.

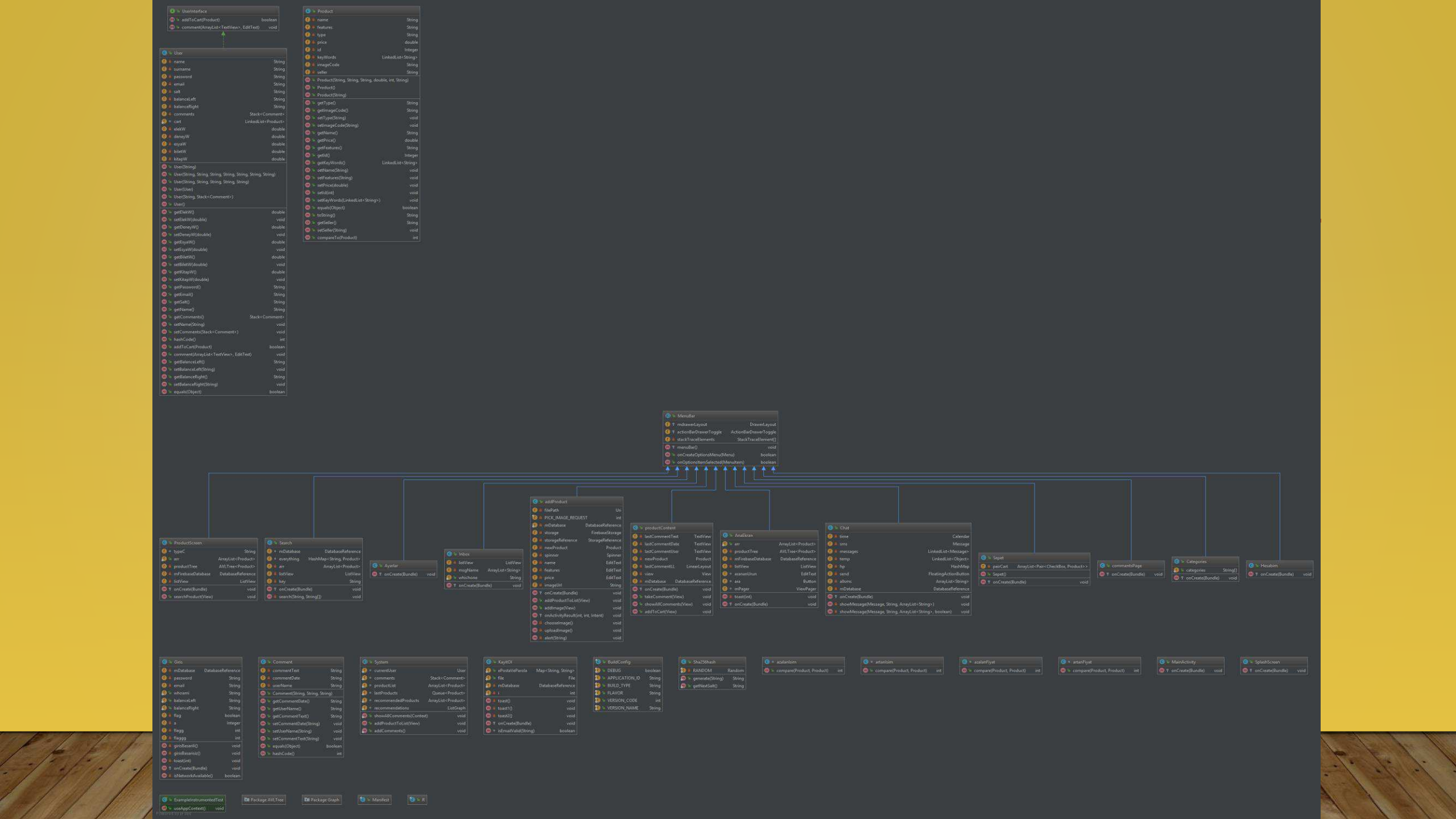




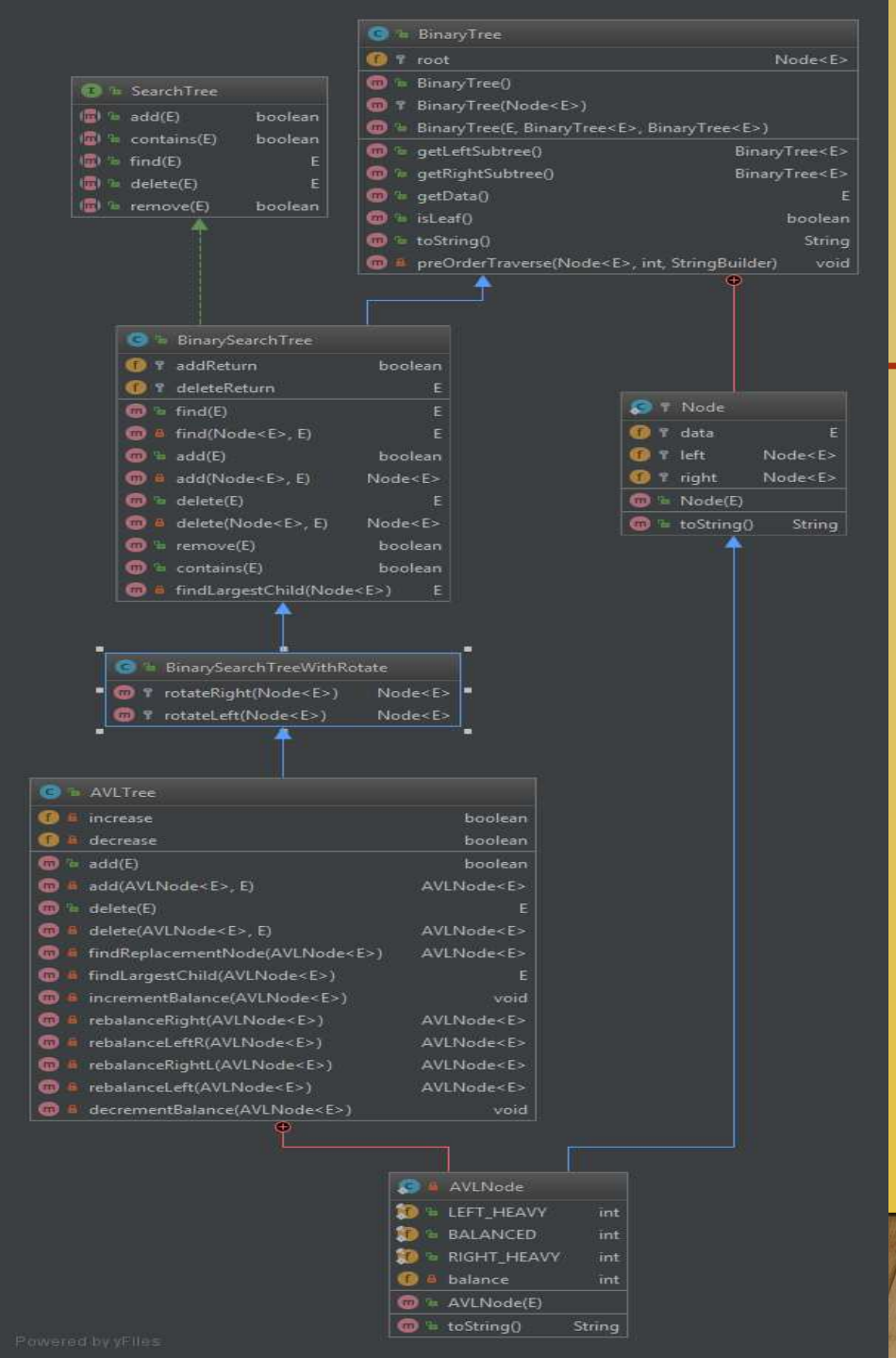
3. RECURSIVE SOLUTION METHODS AND USED SORTING ALGORITHMS.

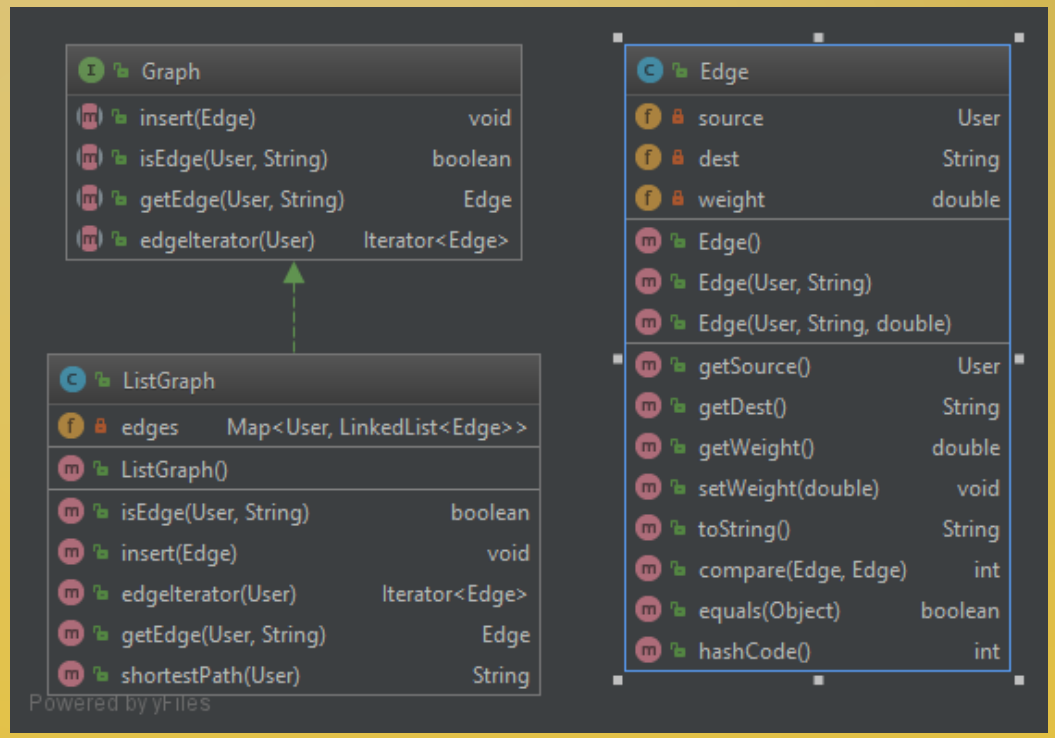
The recursive solution method was used while looking for the products in the categories. It was used to shorten the working time and to do more in less time.

The price order of the products was used the merge sort algorithm for the alphabetical sorting. Products used search tree for search


UserInterface
addToCart(Product) boolean
comment(ArrayList<TextView>, EditText) void
User
name String
surname String
password String
email String
salt String
balanceLeft String
balanceRight String
comments Stack<Comment>
cart LinkedList<Product>
elektW double
deneyW double
esyaW double
biletW double
kitapW double
User(String)
User(String, String, String, String, String, String, String)
User(String, String, String, String, String)
User(User)
User(String, Stack<Comment>)
User()
getElektW() double
setElektW(double) void
getDeneyW() double
setDeneyW(double) void
getEsyaW() double
setEsyaW(double) void
getBiletW() double
setBiletW(double) void
getKitapW() double
setKitapW(double) void
getPassword() String
getEmail() String
getSalt() String
getName() String
getComments() Stack<Comment>
setName(String) void
setComments(Stack<Comment>) void
hashCode() int
addToCart(Product) boolean
comment(ArrayList<TextView>, EditText) void
getBalanceLeft() String
setBalanceLeft(String) void
getBalanceRight() String
setBalanceRight(String) void
equals(Object) boolean
Product
name String
features String
type String
price double
id Integer
keyWords LinkedList<String>
imageCode String
seller String
Product(String, String, String, double, int, String)
Product()
Product(String)
getType() String
getImageCode() String
setType(String) void
setImageCode(String) void
getName() String
getPrice() double
getFeatures() String
getId() Integer
getKeyWords() LinkedList<String>
setName(String) void
setFeatures(String) void
setPrice(double) void
setId(int) void
setKeyWords(LinkedList<String>) void
equals(Object) boolean
toString() String
getSeller() String
setSeller(String) void
compareTo(Product) int
MenuBar
mdrawerLayout DrawerLayout
actionBarDrawerToggle ActionBarDrawerToggle
stackTraceElements StackTraceElement[]
menuBar() void
onCreateOptionsMenu(Menu) boolean
onOptionsItemSelected(MenuItem) boolean
addProduct
FilePath Uri
PICK_IMAGE_REQUEST int
mDatabase DatabaseReference
storage FirebaseStorage
storageReference StorageReference
newProduct Product
spinner Spinner
name EditText
features EditText
price EditText
imageUrl String
onCreate(Bundle) void
addProductToList(View) void
addImage(View) void
onActivityResult(int, int, Intent) void
chooseImage() void
uploadImage() void
alert(String) void
productContent
lastCommentText TextView
lastCommentDate TextView
lastCommentUser TextView
eesvProduct Product
lastCommentLL LinearLayout
view View
mDatabase DatabaseReference
onCreate(Bundle) void
takeComment(View) void
showAllComments(View) void
addToCart(View) void
AnaEkran
arr ArrayList<Product>
productTree AVLTree<Product>
mFirebaseDatabase DatabaseReference
listView ListView
aranamUrun EditText
ara Button
mPager ViewPager
toast(int) void
onCreate(Bundle) void
Chat
time Calendar
sms Message
messages LinkedList<Message>
temp LinkedList<Object>
hp HashMap
send FloatingActionButton
alisms ArrayList<String>
mDatabase DatabaseReference
onCreate(Bundle) void
showMessage(Message, String, ArrayList<String>) void
showMessage(Message, String, ArrayList<String>, boolean) void
Sepet
pairCart ArrayList<Pair<CheckBox, Product>>
Sepet()
onCreate(Bundle) void
commentsPage
onCreate(Bundle) void
Categories
categories String[]
onCreate(Bundle) void
Hesabim
onCreate(Bundle) void
ProductScreen
typeC String
arr ArrayList<Product>
productTree AVLTree<Product>
mFirebaseDatabase DatabaseReference
listView ListView
onCreate(Bundle) void
searchProduct(View) void
Search
mDatabase DatabaseReference
everything HashMap<String, Product>
arr ArrayList<Product>
listView ListView
key String
onCreate(Bundle) void
search(String, String[]) void
Ayarlar
onCreate(Bundle) void
Inbox
listView ListView
msgName ArrayList<String>
whichone String
onCreate(Bundle) void
Giris
mDatabase DatabaseReference
password String
email String
whoami String
balanceLeft String
balanceRight String
flag boolean
a integer
flagg int
flaggg int
girisBasarili() void
girisBasarisiz() void
toast(int) void
onCreate(Bundle) void
isNetworkAvailable() boolean
Comment
commentText String
commentDate String
userName String
Comment(String, String, String)
getCommentDate() String
getUserName() String
getCommentText() String
setCommentDate(String) void
setUserName(String) void
setCommentText(String) void
equals(Object) boolean
hashCode() int
System
currentUser User
comments Stack<Comment>
productList ArrayList<Product>
lastProducts Queue<Product>
recommendedProducts ArrayList<Product>
recommendations ListGraph
showAllComments(Context) void
addProductToList(View) void
addComments() void
KayitOl
ePostaVeParola Map<String, String>
file File
mDatabase DatabaseReference
i int
toast() void
toast1() void
toast2() void
onCreate(Bundle) void
isEmailValid(String) boolean
BuildConfig
DEBUG boolean
APPLICATION_ID String
BUILD_TYPE String
FLAVOR String
VERSION_CODE int
VERSION_NAME String
Sha256hash
RANDOM Random
generate(String) String
getNextSalt() String
azalanIsim
compare(Product, Product) int
artanIsim
compare(Product, Product) int
azalanFiyat
compare(Product, Product) int
artanFiyat
compare(Product, Product) int
MainActivity
onCreate(Bundle) void
SplashScreen
onCreate(Bundle) void
ExampleInstrumentedTest
useAppContext() void
Package AVLTree
Package Graph
Manifest
R

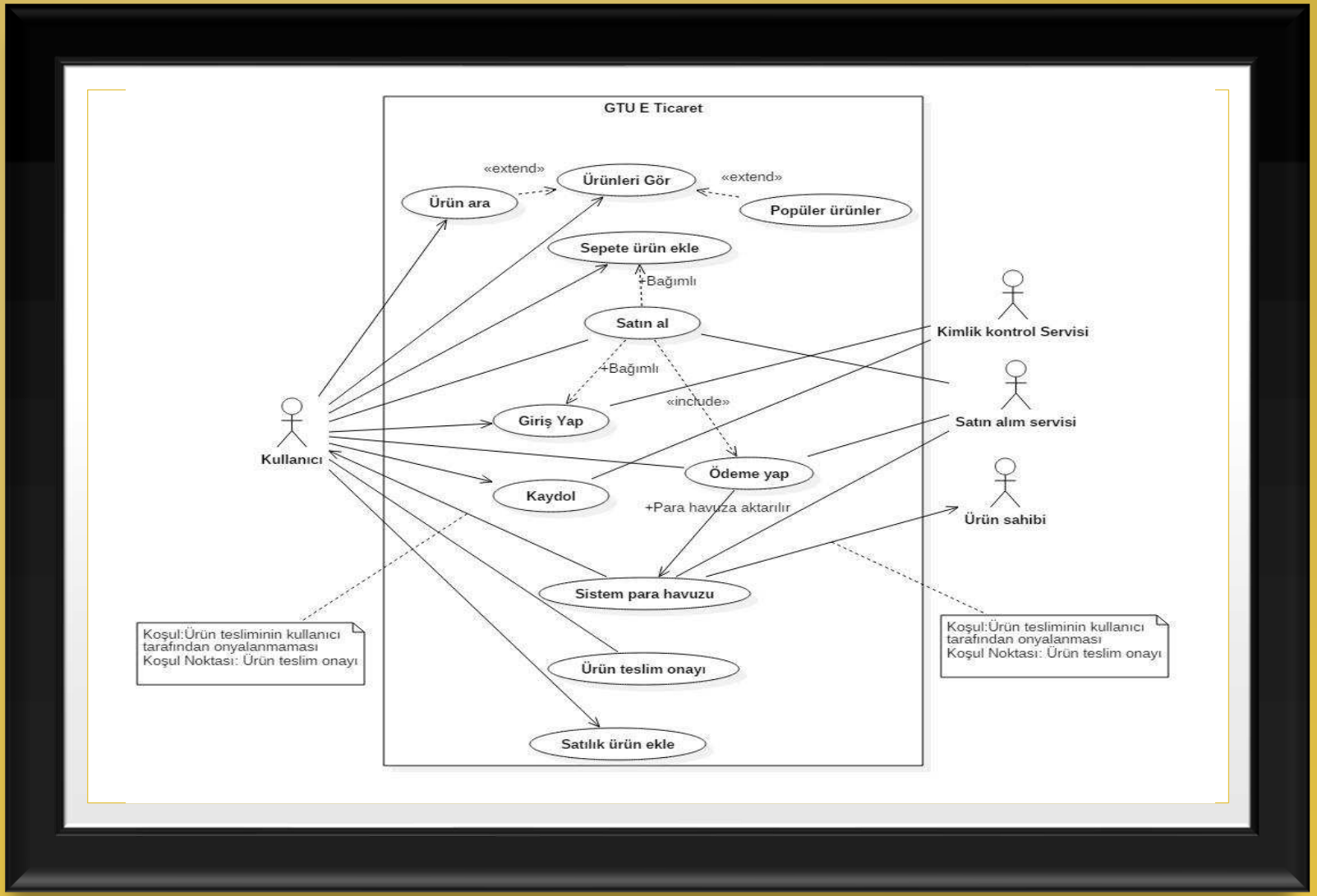
4.UML DIAGRAMS AND HOW TO USE OBJECT ORIENTED PROGRAMMING PRINCIPLES?





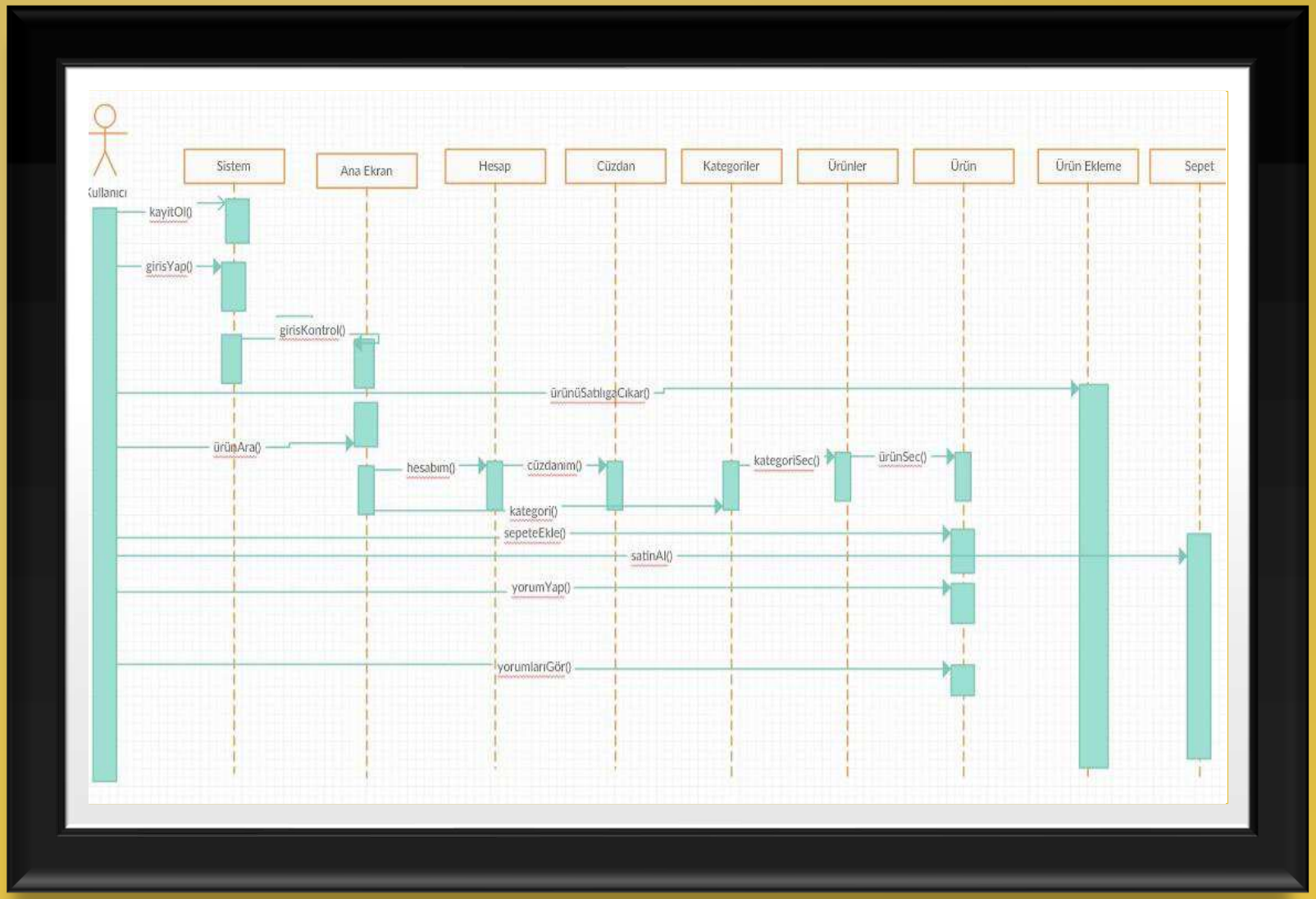
4.UML DIAGRAMS AND HOW TO USE OBJECT ORIENTED PROGRAMMING PRINCIPLES?

UseCase Diagram



4.UML DIAGRAMS AND HOW TO USE OBJECT ORIENTED PROGRAMMING PRINCIPLES?

Sequential Diagram



5. USER SCENARİOS.

- The user opens the application and registers in the system with e-mail with @ gtu.edu.tr name, first name, surname and the password specified.

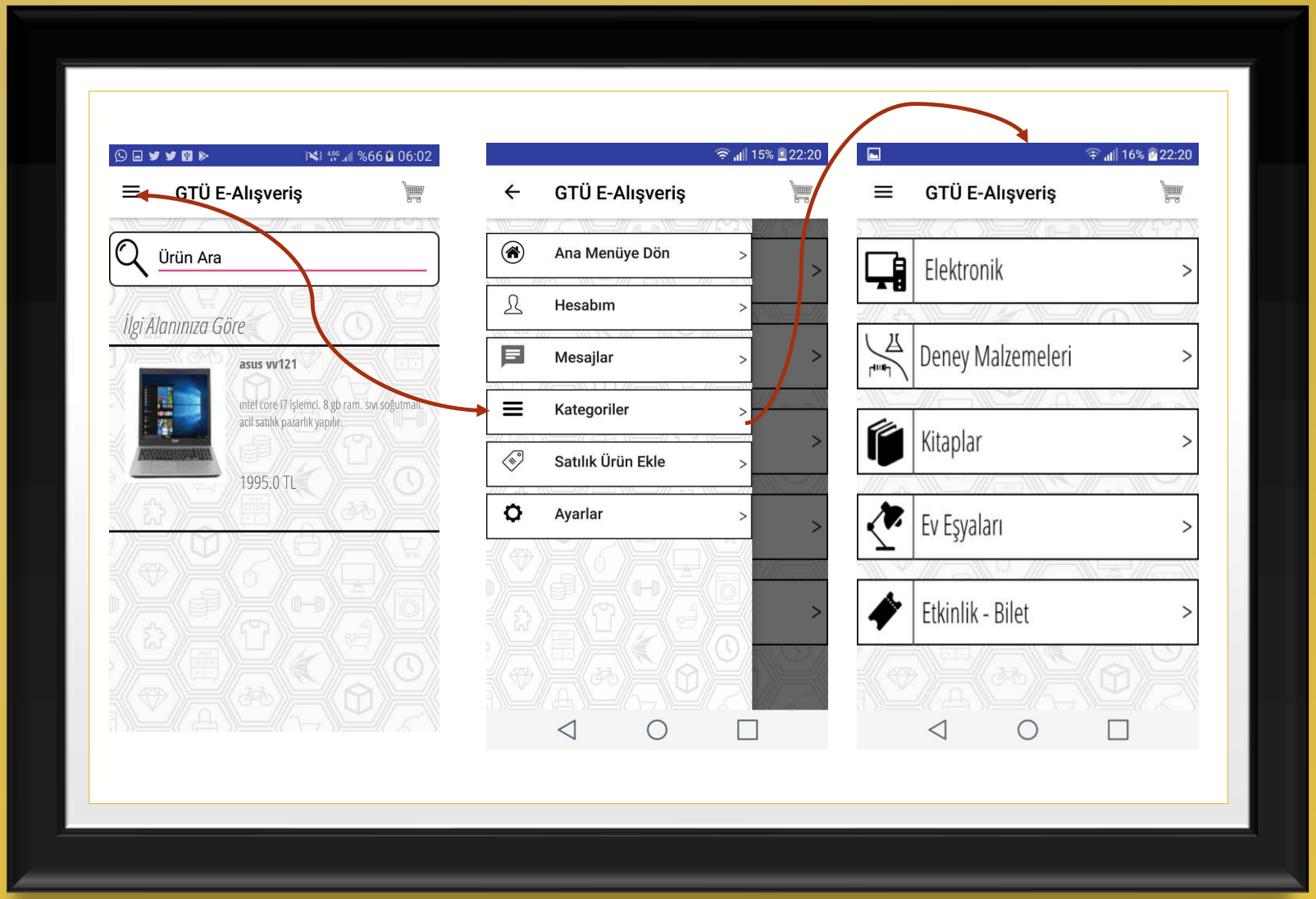
5. USER SCENARIOS.

THE USER OPENS THE APPLICATION. IT LOGS IN WITH USERNAME AND PASSWORD WITH E-MAIL EXTENSION.

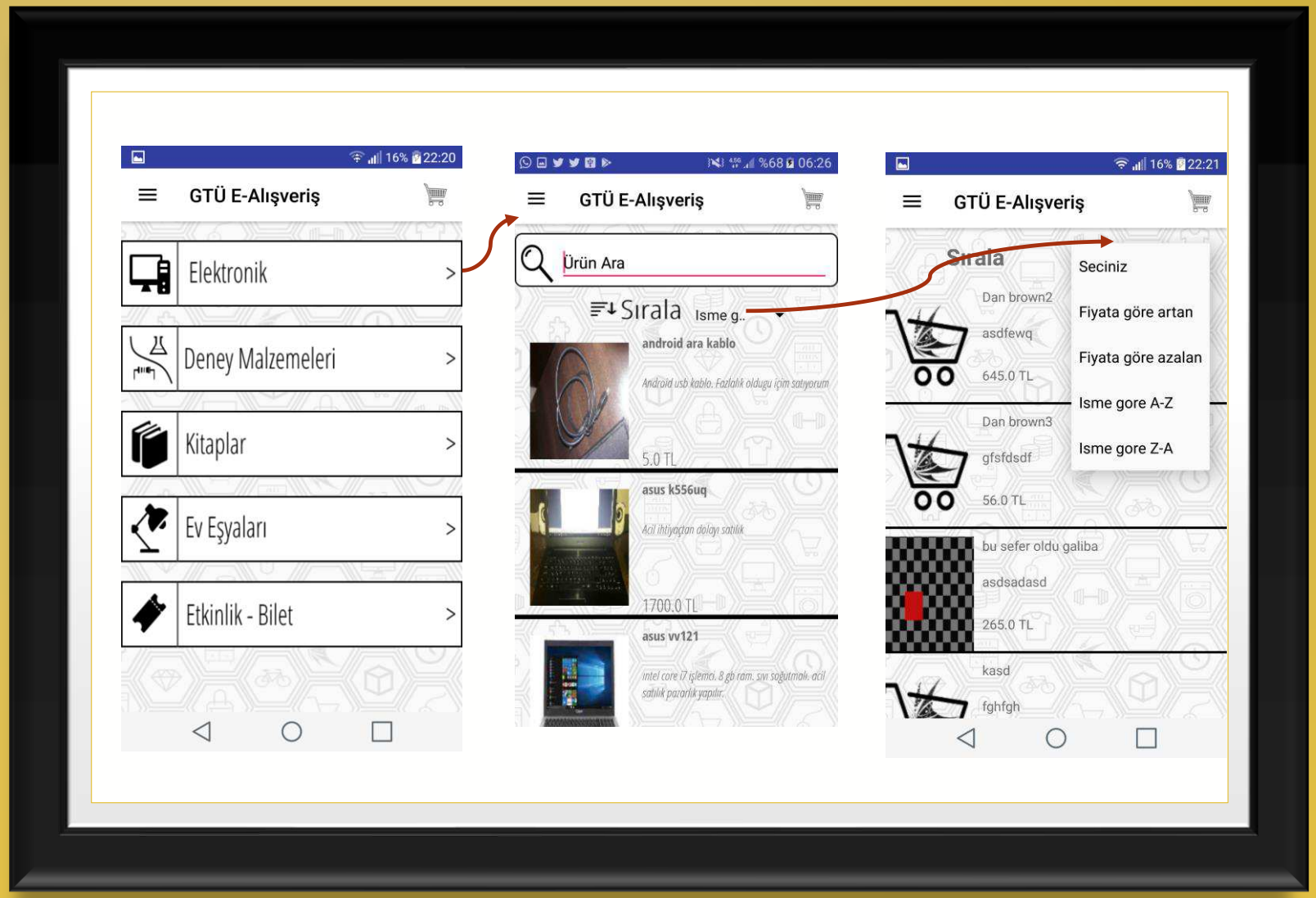
5. USER SCENARIOS.

IF E-MAIL IS NOT REGISTERED , USER IS WARNED.

5. USER SCENARİOS.

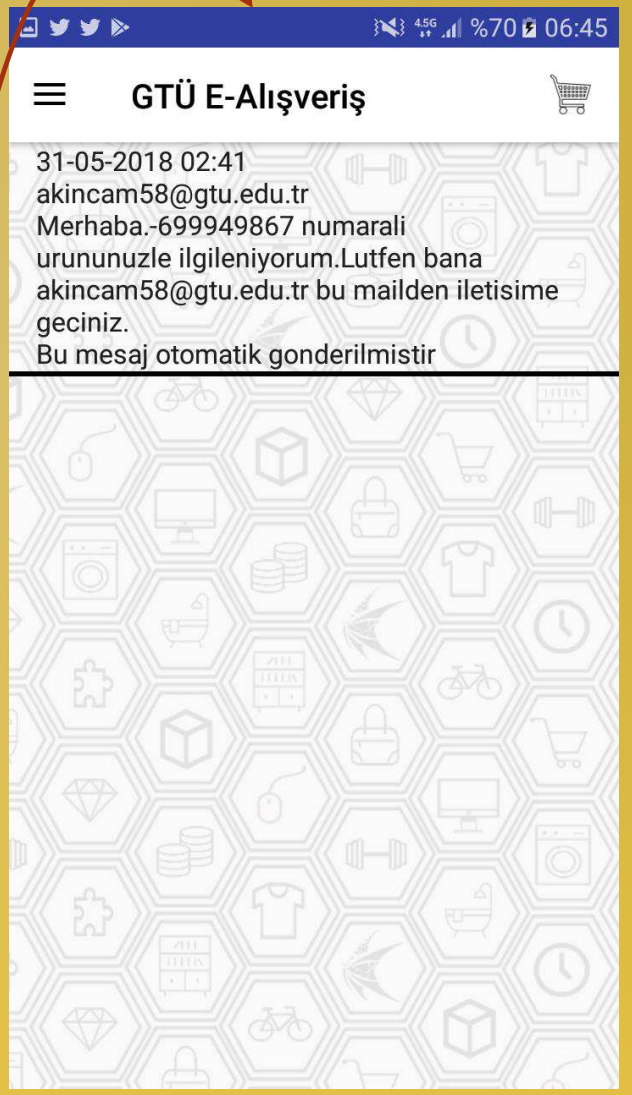


5. USER SCENARİOS.

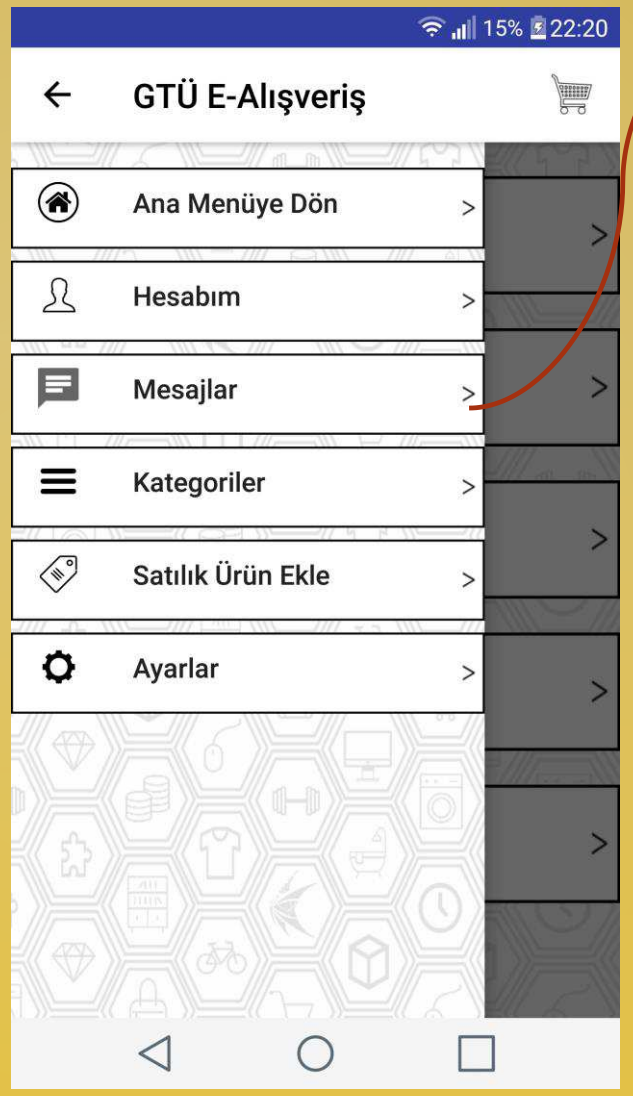


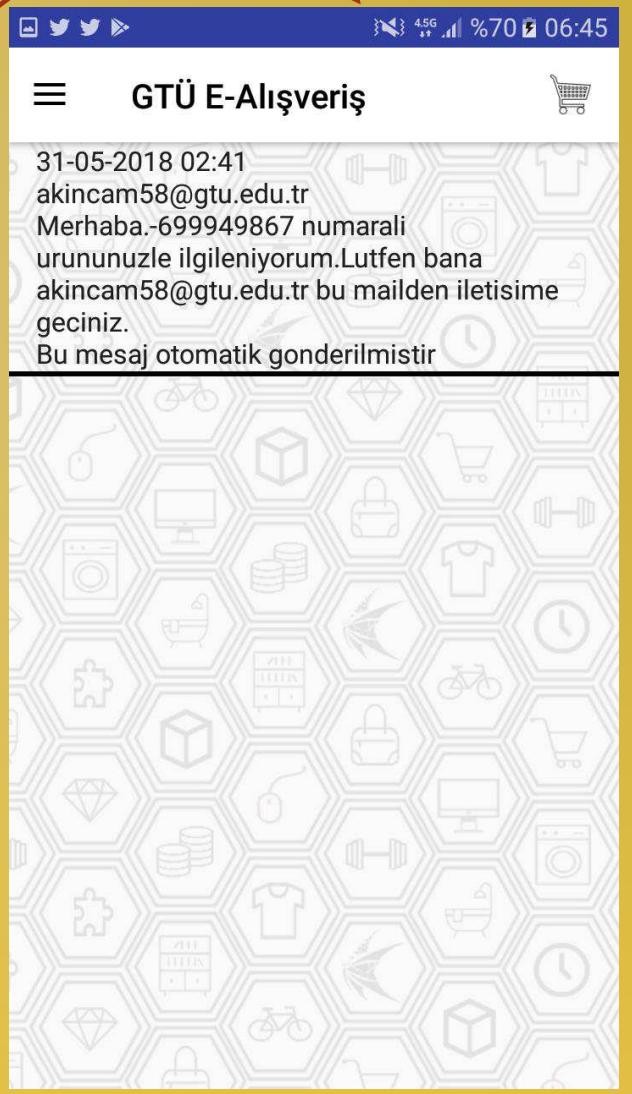
5. USER SCENARİOS.

5. USER SCENARİOS.

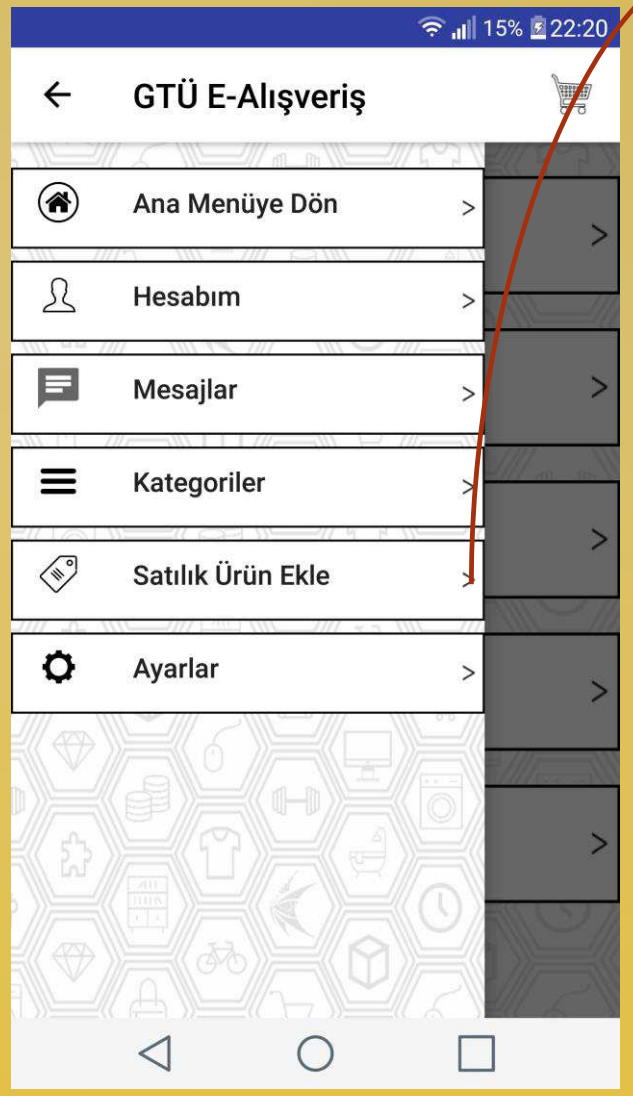


5. USER SCENARİOS.





5. USER SCENARİOS.



5. USER SCENARİOS.