Online Book Store Application

Group Members:

Abdul Hannan AYUBI	152120181097
Eddy Ngabo SHAMI	152120181093
Furkan TAŞKIN	152120181029
Osman ÇAĞLAR	152120181033
Serdar DEMİRTAŞ	152120181011

The Lecturer:

Asst. Prof. Uğur GÜREL

02/06/2021

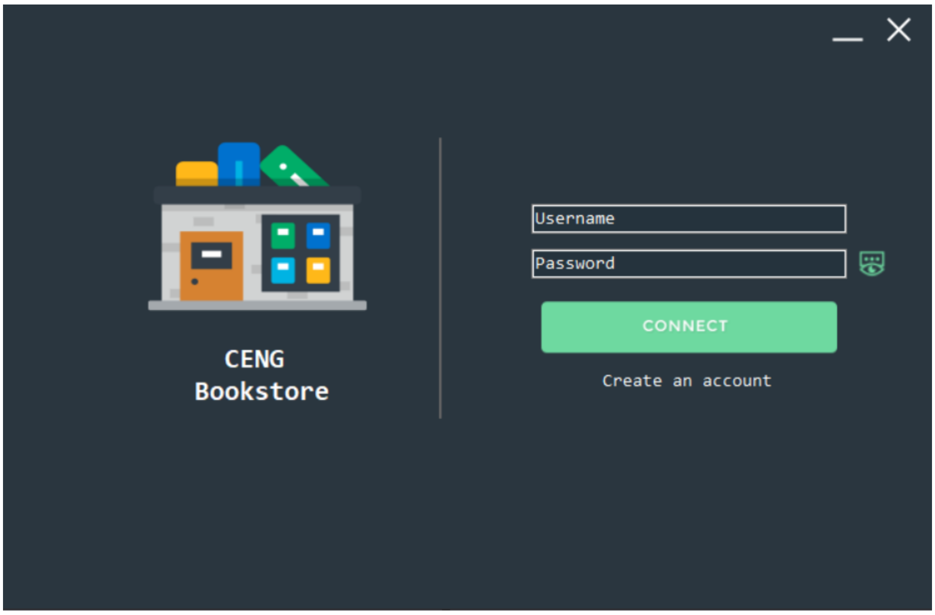
CONTENTS

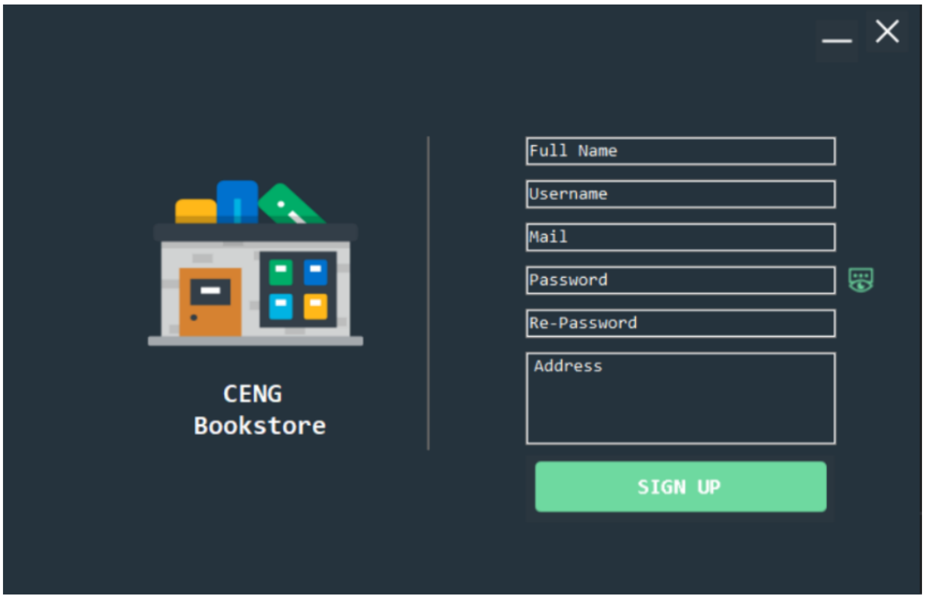
1. INTRODUCTION

In this project, we made a desktop application where users can sign up and buy products in different categories. Two types of users can enter the application, the first is the customer login and the other is the admin login. If the customer has a registered account, they can log in to the application, if they do not have an account, they can create a new account by clicking the "create an account" text. When logging in to the application, you can see the username and name at the top left, access the day date and time information at the bottom left, and log out with the "logout" button if desired. At the same time, the customer can switch between 3 different categories (books, magazines, music/cd) and click on the image of the product they want, access the detailed information about the product on the pop-up screen, and add as many products as they want to the shopping cart. If he/she cannot find the desired product in any category, he/she can search for the product name and access that product using the "Search Button". It can be displayed in the shopping cart with the "My Cart" button on the left panel. He/she can see the products he added to his/her cart in a list box and can remove the product from his/her cart by double-clicking on the product he/she wants to remove from his/her cart. He/she can see the total amount of his/her basket at the bottom right, and he/she can delete all the products in his/her basket with the "delete all products" button, and his/her can buy the products in his/her basket with the "buy" button. The invoice containing the purchased products is sent to the user via e-mail. If an admin log in, in addition to everything the customer can do, they can add and remove products to any category they want through the "admin panel", update the product they want, and with the "clear" button, they can delete the text boxes containing the information about the product they have created.

2. DESIGN

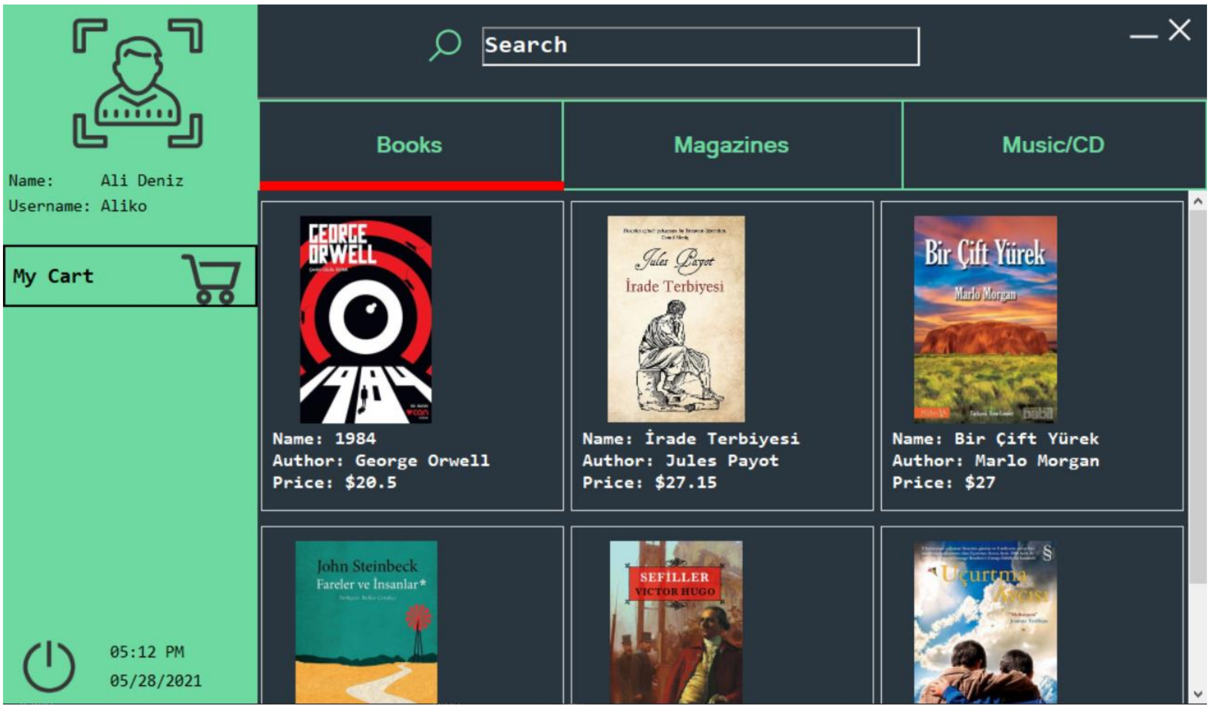
a. Login & Sign-Up Screens





The application starts with login screen. If the person has a user account, he/she can access the application via this screen. If the person is an admin, he/she also can access the application via their admin username and password. If the person is not sure about the password what he/she wrote, by clicking the “show password” image the person can see his/her password. In sign-up screen the person can sign up with his/her information, except admin. A user can only be given admin authority over the database.

Main Menu



Name: Ali Deniz
Username: Aliko

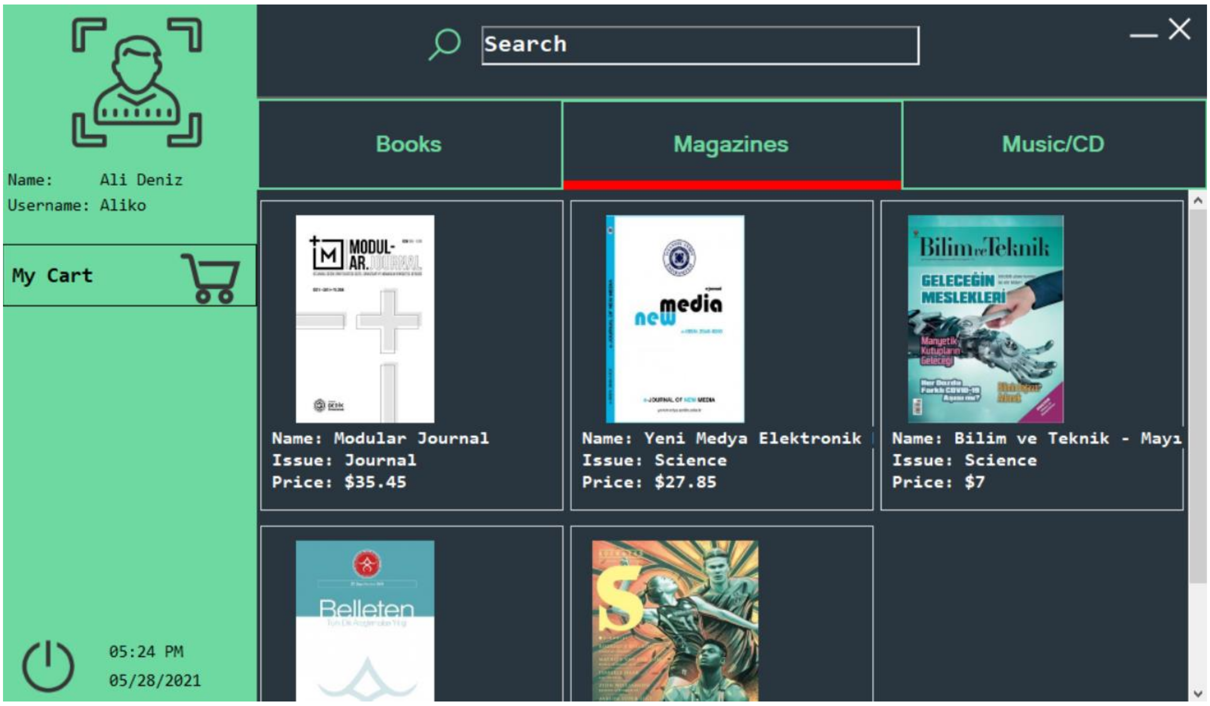
Name: 1984
Author: George Orwell
Price: $20.5

Name: İrade Terbiyesi
Author: Jules Payot
Price: $27.15

Name: Bir Çift Yürek
Author: Marlo Morgan
Price: $27

05:12 PM
05/28/2021

Books



Name: Ali Deniz
Username: Aliko

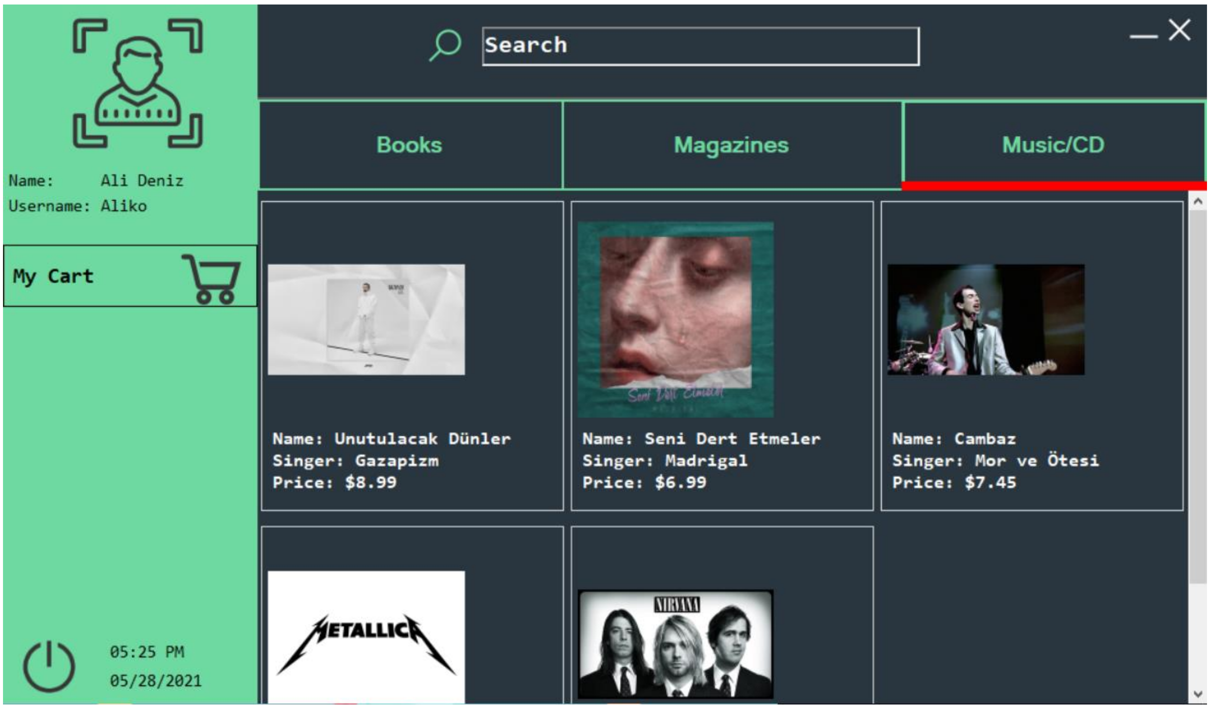
Name: Modular Journal
Issue: Journal
Price: $35.45

Name: Yeni Medya Elektronik
Issue: Science
Price: $27.85

Name: Bilim ve Teknik - Mayı
Issue: Science
Price: $7

05:24 PM
05/28/2021

Magazines



Name: Ali Deniz
Username: Aliko

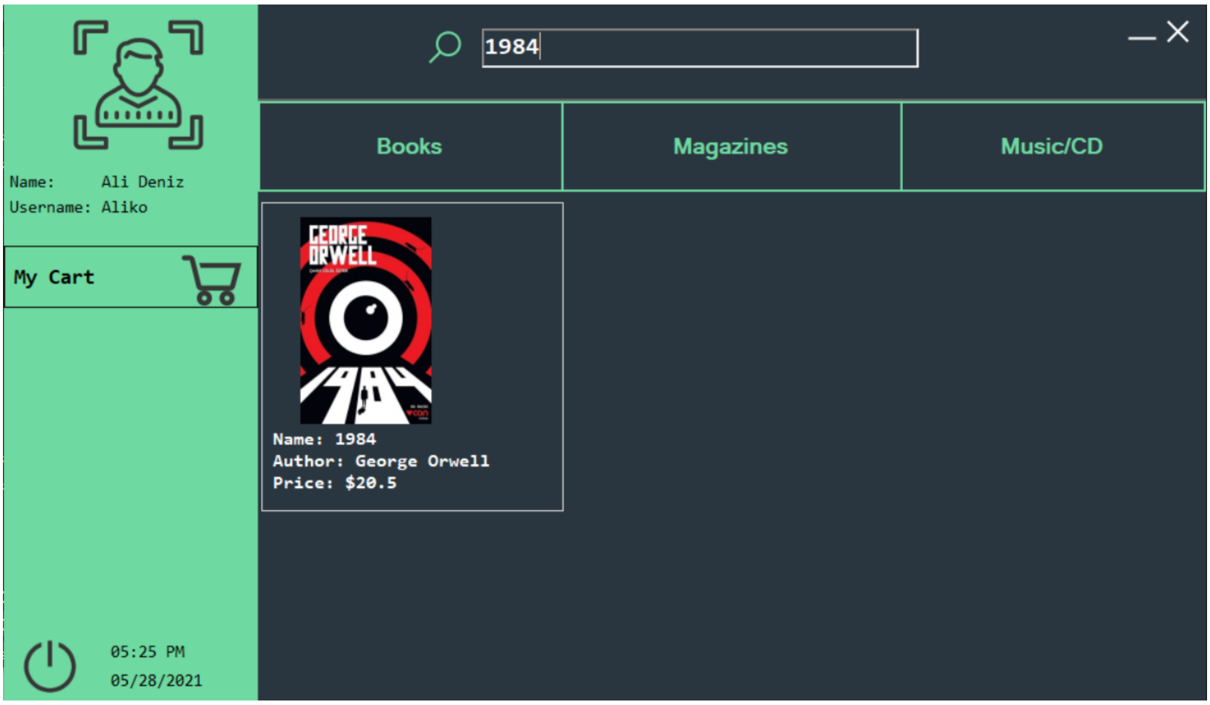
Name: Unutulacak Dünler
Singer: Gazapizm
Price: $8.99

Name: Seni Dert Etmeler
Singer: Madrigal
Price: $6.99

Name: Cambaz
Singer: Mor ve Ötesi
Price: $7.45

05:25 PM
05/28/2021

Music/CD



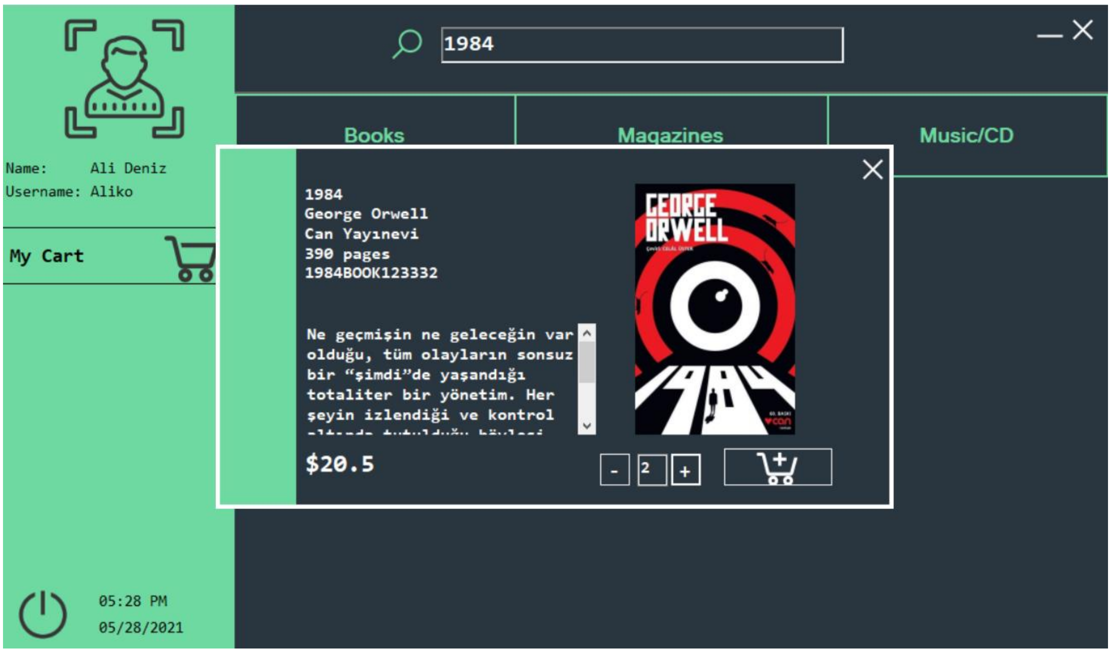
1984

Name: Ali Deniz
Username: Aliko

Name: 1984
Author: George Orwell
Price: $20.5

05:25 PM
05/28/2021

When using the search button



1984
George Orwell
Can Yayınevi
390 pages
1984BOOK123332

Ne geçmişin ne geleceğin var olduğu, tüm olayların sonsuz bir "şimdi"de yaşandığı totaliter bir yönetim. Her şeyin izlendiği ve kontrol

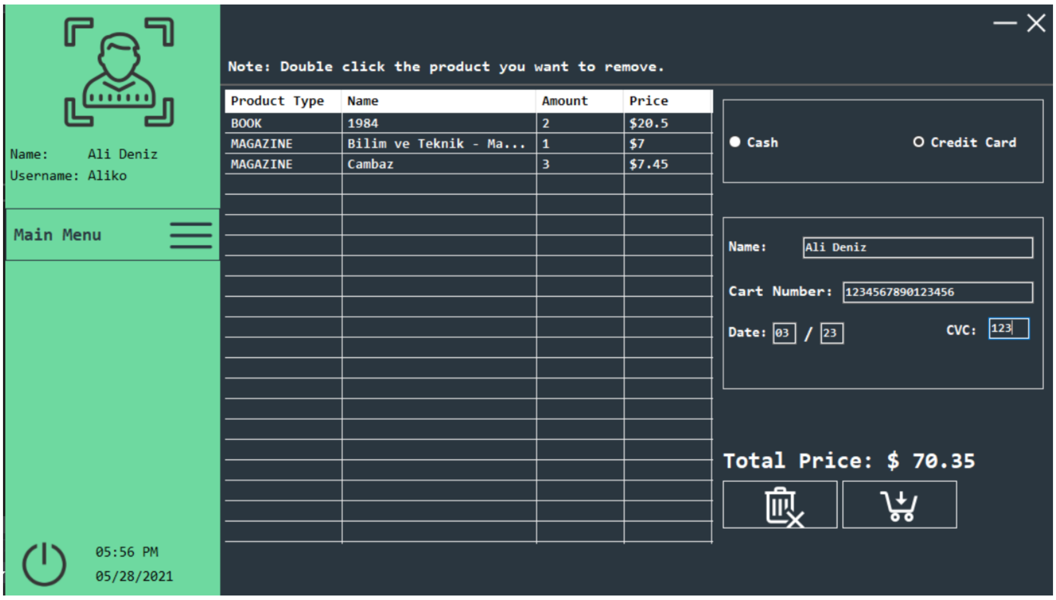
$20.5

- 2 +

The Pop-up Screen

When the user entered the application, he/she sees main menu. Main menu starts with books, he/she can change categories by clicking on their names and click on the image of the product they want, access the detailed information about the product on the pop-up screen, and add as many products as they want to the shopping cart. If they did not find the product they want, they can search the name of the product via the "Search Button". By using "My Cart" button, they can see their shopping cart.

b. My Cart



Note: Double click the product you want to remove.

Product Type	Name	Amount	Price
BOOK	1984	2	$20.5
MAGAZINE	Bilim ve Teknik - Ma...	1	$7
MAGAZINE	Cambaz	3	$7.45

● Cash ○ Credit Card

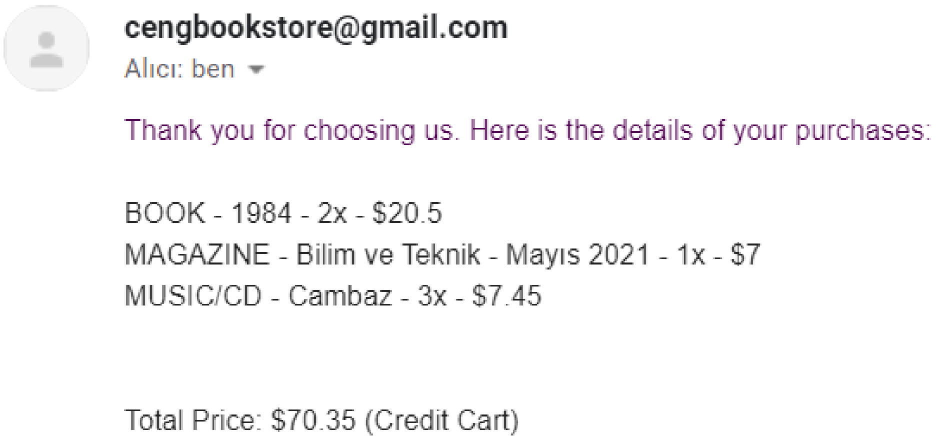
Name: Ali Deniz

Cart Number: 1234567890123456

Date: 03 / 23 CVC: 123

Total Price: $ 70.35

The user can see products which he/she added to his/her shopping cart in this page. If he/she wants to remove a product, he/she can double click the product he/she wants to remove or to remove all product he/she can use “Remove All” button. Our bookstore has just 2 payment type; cash and credit card. If the user chooses credit cart as payment type, he/she must input his/her credit card information. After purchasing the products in the shopping cart, the invoice goes to the mail.



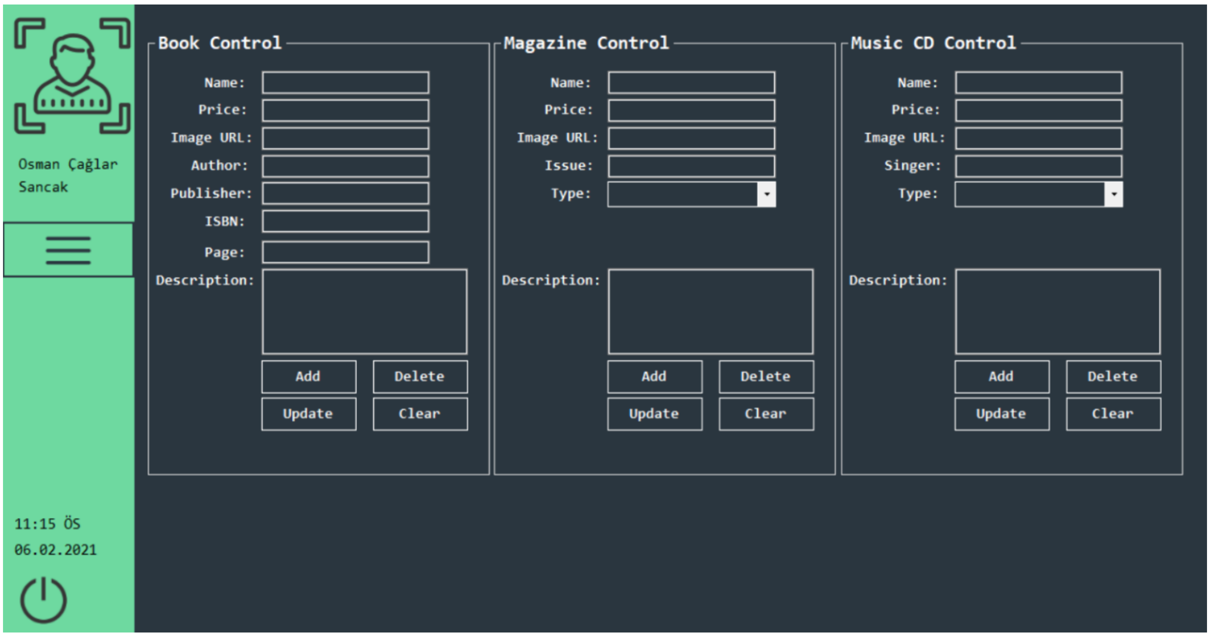
cengbookstore@gmail.com
Alıcı: ben

Thank you for choosing us. Here is the details of your purchases:

BOOK - 1984 - 2x - $20.5
MAGAZINE - Bilim ve Teknik - Mayıs 2021 - 1x - $7
MUSIC/CD - Cambaz - 3x - $7.45

Total Price: $70.35 (Credit Cart)

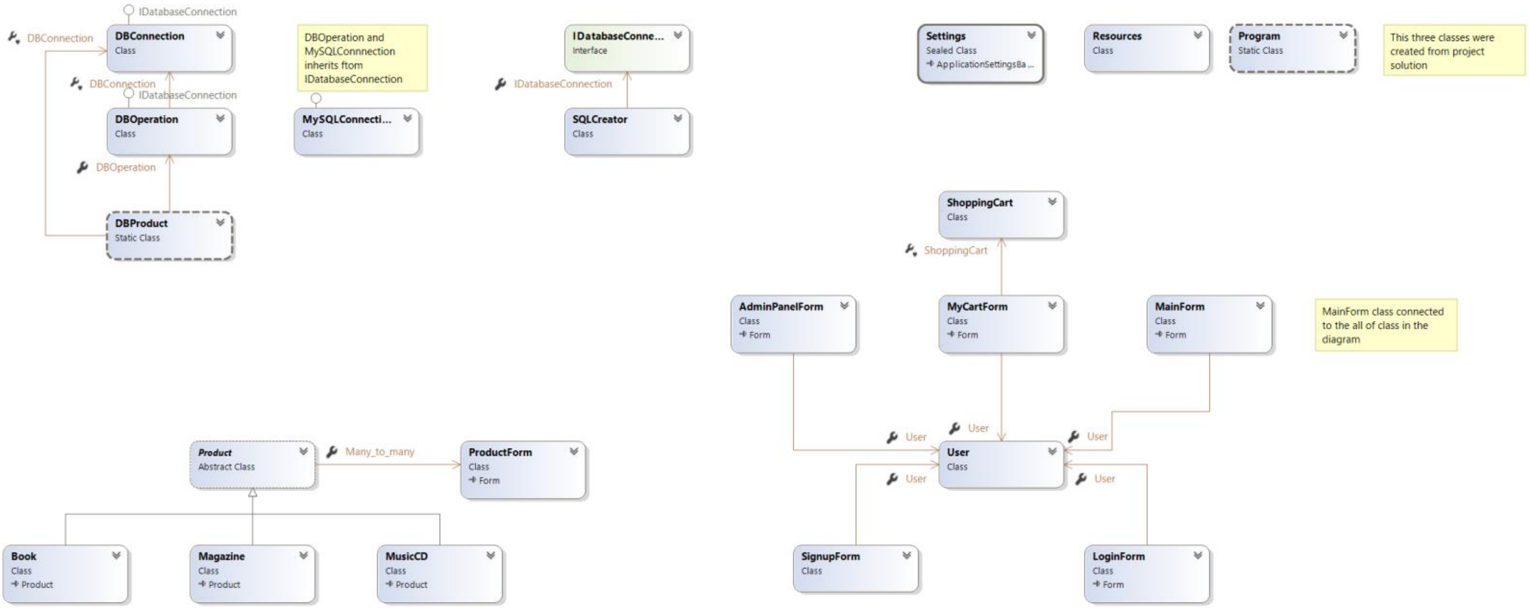
Invoice Mail



Admin Panel

This page is the Admin Panel, if you an admin, you can access to this page and you can control products from database with this panel. To update a product, you must enter the name of the product right, because the application access to product with the name and you can update all the information except the product name. If you want to delete the product you can do it just with the product name.

c. UML Diagram



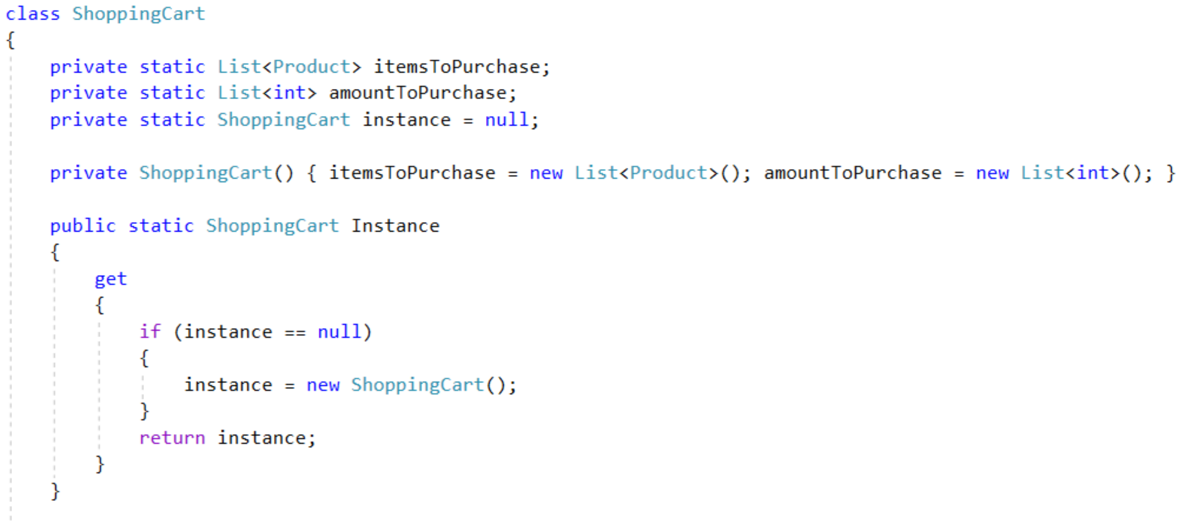
DBOperation and MySQLConnnection inherits ftom IDatabaseConnection

This three classes were created from project solution

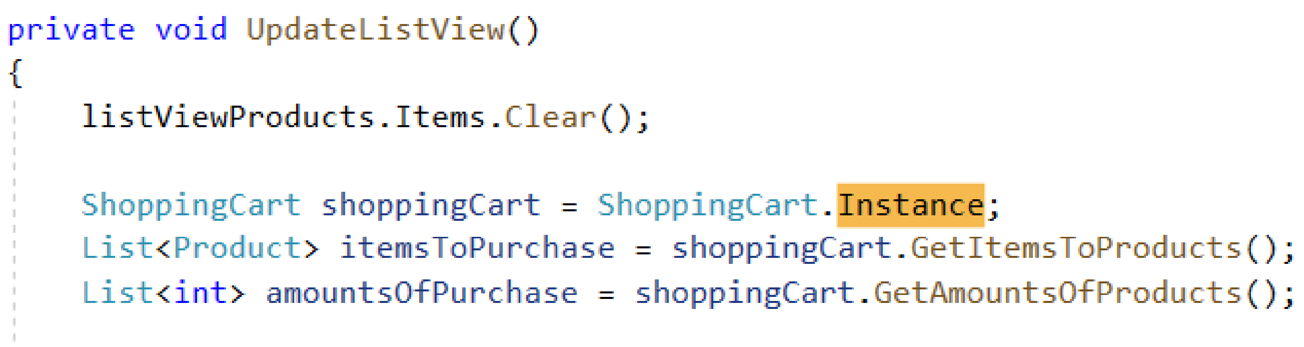
MainForm class connected to the all of class in the diagram

d. Design Patterns That We Used

Singleton Design Pattern



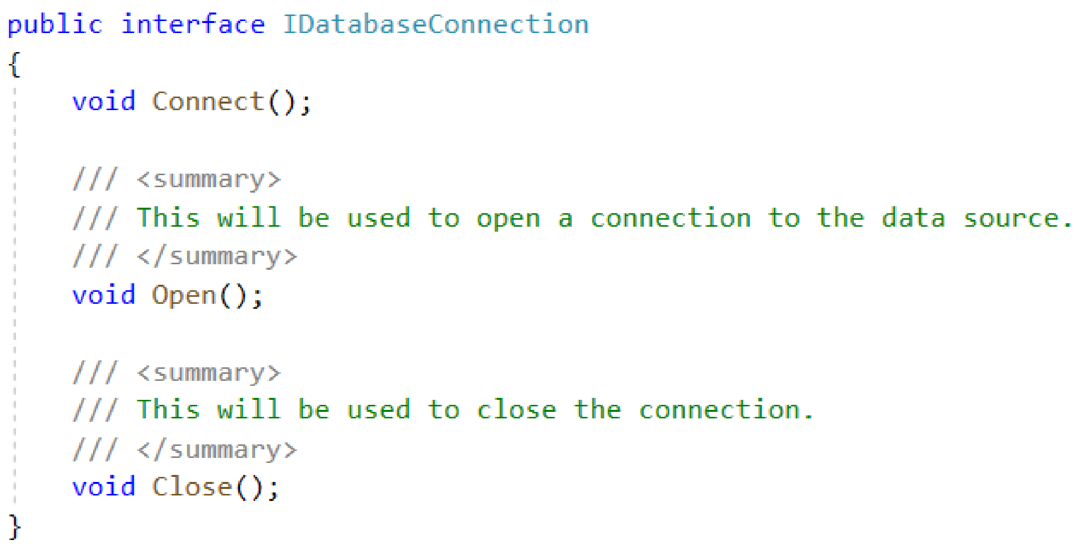
```
class ShoppingCart
{
    private static List<Product> itemsToPurchase;
    private static List<int> amountToPurchase;
    private static ShoppingCart instance = null;

    private ShoppingCart() { itemsToPurchase = new List<Product>(); amountToPurchase = new List<int>(); }

    public static ShoppingCart Instance
    {
        get
        {
            if (instance == null)
            {
                instance = new ShoppingCart();
            }
            return instance;
        }
    }
```



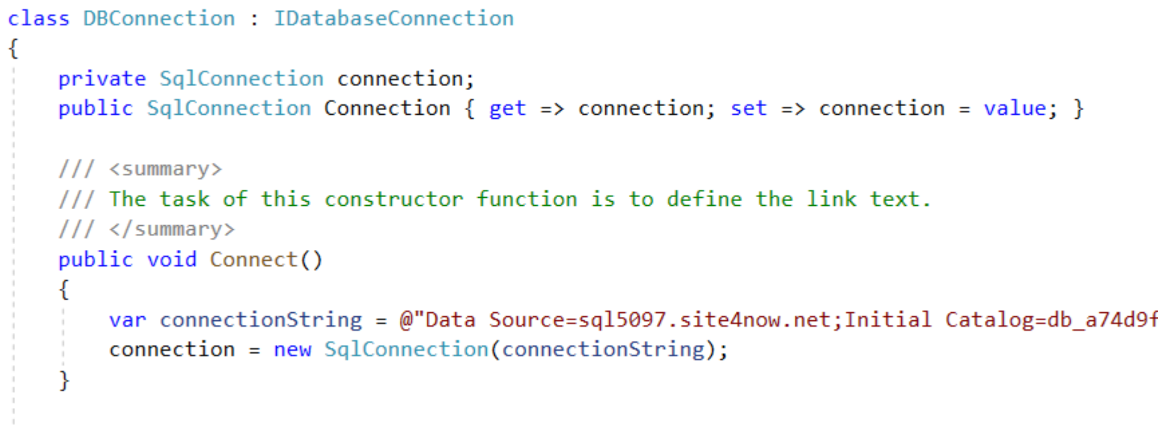
```
private void UpdateListView()
{
    listViewProducts.Items.Clear();

    ShoppingCart shoppingCart = ShoppingCart.Instance;
    List<Product> itemsToPurchase = shoppingCart.GetItemsToProducts();
    List<int> amountsOfPurchase = shoppingCart.GetAmountsOfProducts();
```

We used 2 design patterns in our project. First, Singleton Design Pattern, and the second one is Strategy Design Pattern. To make our code more readable, firstly we used Singleton Design Pattern for “ShoppingCart” class. As a result, each user can only have one shopping cart. We decided to use this design pattern in the “ShoppingCart” class in order not to overload the RAM.

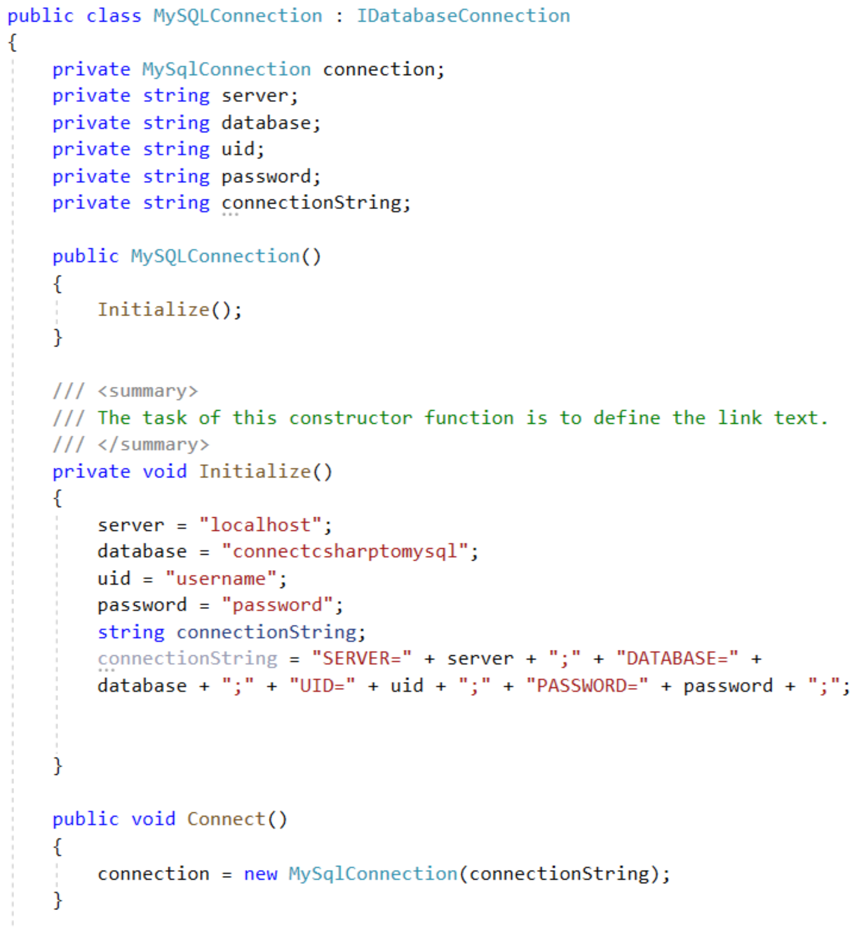
Strategy Design Pattern



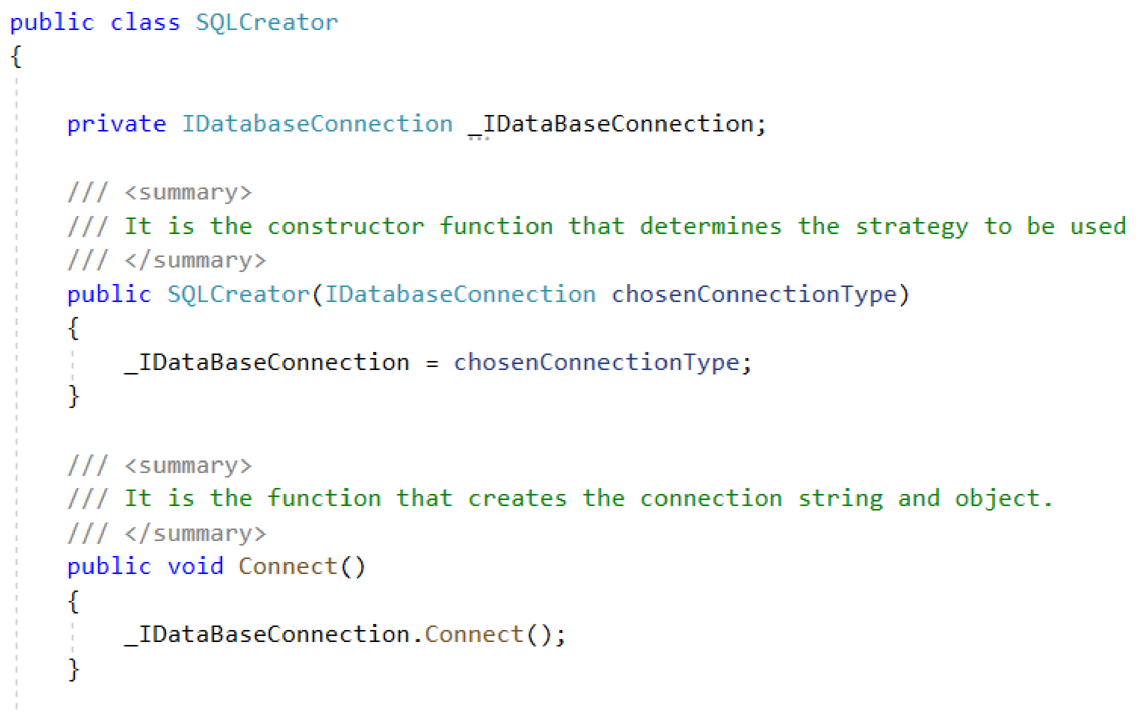
```
public interface IDatabaseConnection
{
    void Connect();

    /// <summary>
    /// This will be used to open a connection to the data source.
    /// </summary>
    void Open();

    /// <summary>
    /// This will be used to close the connection.
    /// </summary>
    void Close();
}
```



```
class DBConnection : IDatabaseConnection
{
    private SqlConnection connection;
    public SqlConnection Connection { get => connection; set => connection = value; }

    /// <summary>
    /// The task of this constructor function is to define the link text.
    /// </summary>
    public void Connect()
    {
        var connectionString = @"Data Source=sql5097.site4now.net;Initial Catalog=db_a74d9f
        connection = new SqlConnection(connectionString);
    }
```



```
public class MySQLConnection : IDatabaseConnection
{
    private MySqlConnection connection;
    private string server;
    private string database;
    private string uid;
    private string password;
    private string connectionString;

    public MySQLConnection()
    {
        Initialize();
    }

    /// <summary>
    /// The task of this constructor function is to define the link text.
    /// </summary>
    private void Initialize()
    {
        server = "localhost";
        database = "connectcsharptomysql";
        uid = "username";
        password = "password";
        string connectionString;
        connectionString = "SERVER=" + server + ";" + "DATABASE=" +
        database + ";" + "UID=" + uid + ";" + "PASSWORD=" + password + ";";


    }

    public void Connect()
    {
        connection = new MySqlConnection(connectionString);
    }
```



```
public class SQLCreator
{

    private IDatabaseConnection _IDataBaseConnection;

    /// <summary>
    /// It is the constructor function that determines the strategy to be used
    /// </summary>
    public SQLCreator(IDatabaseConnection chosenConnectionType)
    {
        _IDataBaseConnection = chosenConnectionType;
    }

    /// <summary>
    /// It is the function that creates the connection string and object.
    /// </summary>
    public void Connect()
    {
        _IDataBaseConnection.Connect();
    }
```

The purpose of using this pattern is to make database connections in different SQL types (MySQL, MSSQL). We created a class called “IDataBaseConnection”, this interface class has 3 function which called “Connect, Open and Close”. Classes that inherit this interface populate these functions according to their connection types. We also have “SQLCreator” class, and the purpose of this class ensures the continuity of the program by taking the MySQL or MSSQL type object we created into the parameter of the interface type in the constructor function it contains, finding its own specific SQL type with the help of polymorphism and running its functions.

3. DUTY OF TEAM MEMBERS

Abdul Hannan Ayubi	Eddy Ngabo Shami	Furkan Taşkın	Osman Çağlar	Serdar Demirtaş
Group Coordinator, Product Form, AdminPanel Form, Program Class	Product Class, Book Class, Magazine Class, MusicCD Class	Design, Login Form, SignUp Form, User Class, Report,	DBConnection Class, IDataBaseConnection Class, MySQLConnection Class, SQLCreator Class, MainMenu Form, DBProducts Class, DBOperations Class	Design, MyCart Form, ShoppingCart Class, UML Diagram

Note: Although each class seems to belong to a single group member above, we worked on most classes as a group. So, we can say that each group member works simultaneously.

Note-2: https://bitbucket.org/ABDULHANNANAYUBI/oopfinalproject/src/develop/