

Pharm App

Application prepared by:

Chiriac Dan-Constantin

Ratoi Liviu

Ursachi Gabriela

Cuptor Iuliana Stefania

Faculty of Automatic Control and Computer Engineering

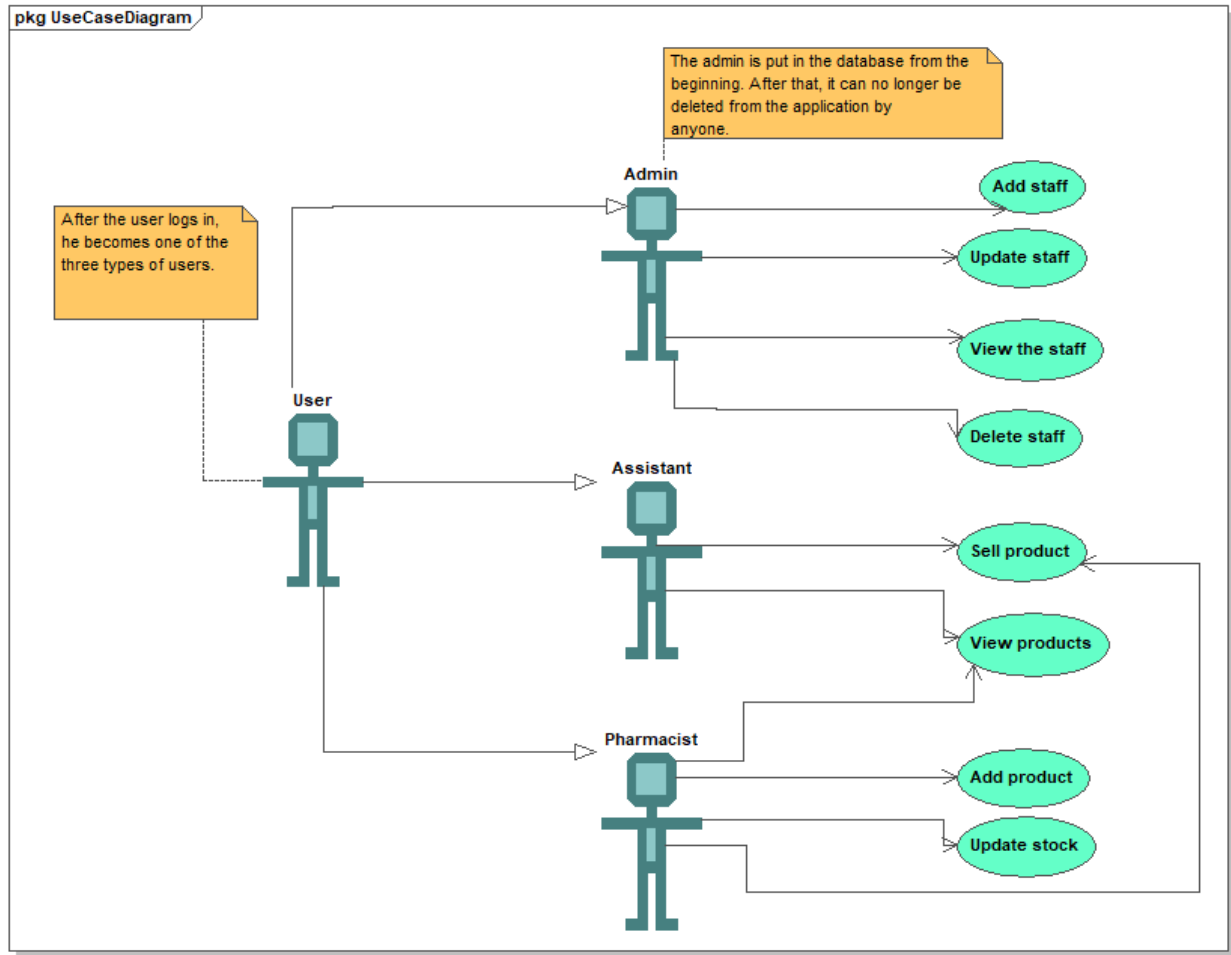
PharmApp is a desktop application for managing the staff of a pharmacy, as well as the administration of marketed products. On the one hand, it deals with the sale and supply of medicines, as well as checking stocks. Also, it considers the organization of the pharmacy staff and the allocation of rights according to the actions that each type of employee can exercise. It does not require a network connection, it uses local data storage to retain the necessary information.

Diagrams

```

classDiagram
    class ChangeConnection {
        +ChangeConnection()
        +CreateTables()
        +Insert()
        +SelectAllUsers()
        +SelectAllProducts()
        +SelectUser()
        +SelectProduct()
        +SelectUser()
        +SelectProduct()
        +DeleteUser()
        +GetInstances()
        +DB()
    }
    class PharmacyManagementDLL_Real_ActionManager {
        +RealActionManager()
        +AddNewProduct()
        +SellProduct()
        +AddToStock()
        +GetProducts()
        +AddUser()
        +UpdateUserPassword()
        +DeleteUser()
        +GetUsers()
    }
    class PharmacyManagementDLL_Permissions {
        -_permissionsList
        +Permissions()
        +RightsList()
    }
    class Items_DatabaseObjects_User {
        +id
        +Username
        +Password
        +Rights
        -jd
        -username
        -password
        -rights
        +User()
        +ToString()
    }
    class PharmacyManagementDLL_Proxy_ActionManager {
        +CurrentUser
        +Login()
        +AddNewProduct()
        +AddToStock()
        +GetProducts()
        +SellProduct()
        +AddUser()
        +UpdateUserPassword()
        +DeleteUser()
        +GetUser()
        +GetUsers()
        +RunAccess()
        +GetInstance()
        +ProxyActionManager()
    }
    ChangeConnection --> PharmacyManagementDLL_Real_ActionManager : _dbInstance
    ChangeConnection --> PharmacyManagementDLL_Permissions : _instance
    PharmacyManagementDLL_Real_ActionManager --> PharmacyManagementDLL_Permissions : _realActionManager
    PharmacyManagementDLL_Permissions --> PharmacyManagementDLL_Proxy_ActionManager : _permissions
    PharmacyManagementDLL_Proxy_ActionManager --> Items_DatabaseObjects_User : _currentUser
    PharmacyManagementDLL_Proxy_ActionManager --> PharmacyManagementDLL_Permissions : _permissions
    PharmacyManagementDLL_Proxy_ActionManager --> PharmacyManagementDLL_Real_ActionManager : _realActionManager
    PharmacyManagementDLL_Proxy_ActionManager --> ChangeConnection : _instance
    
```

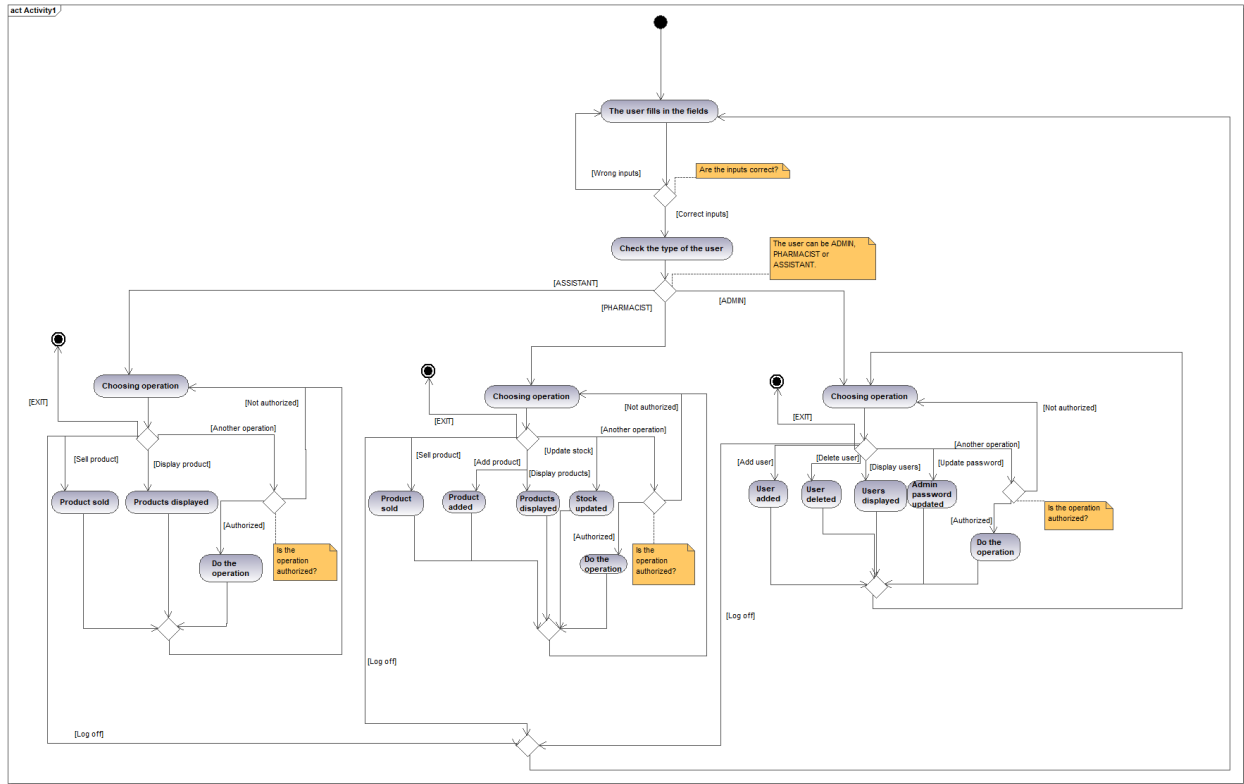
The use case diagram is shown below:



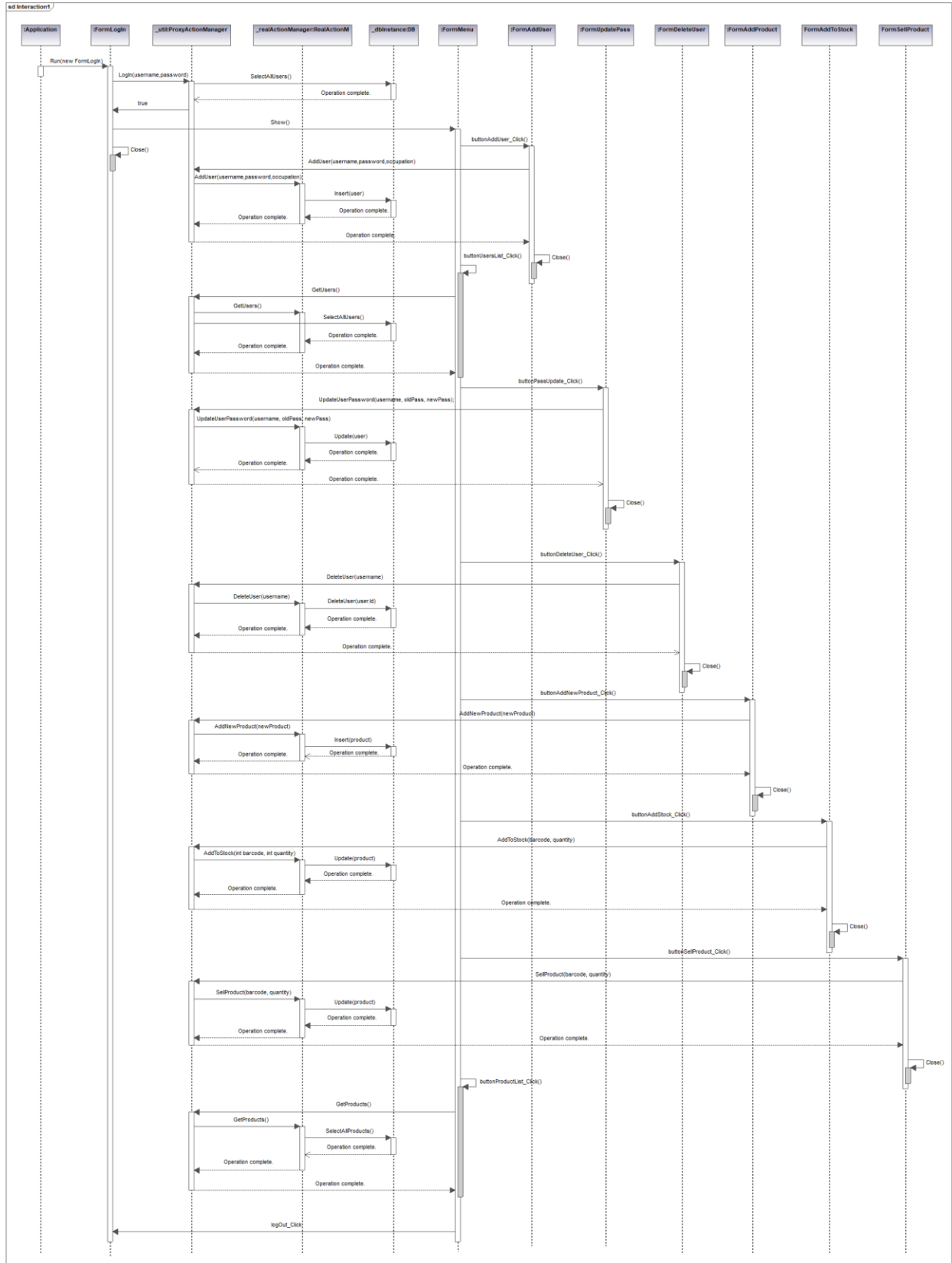
Generated by UModel

www.altova.com

The activity diagram is shown below:



The sequence diagram is shown below:

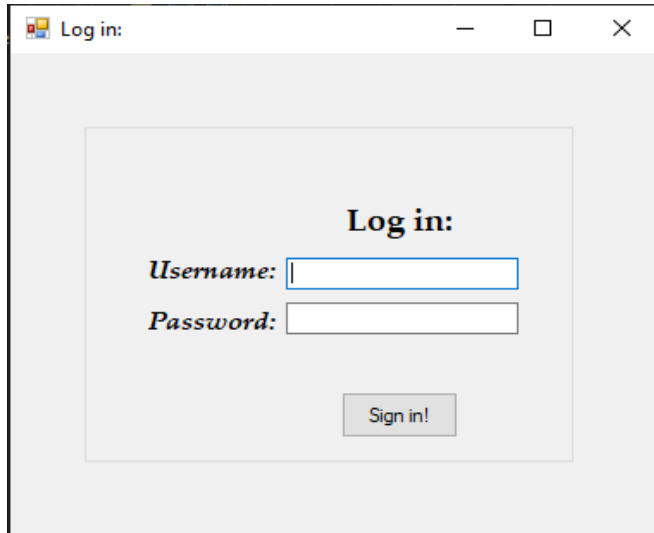


How the application works

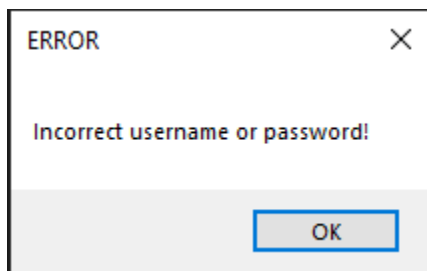
An user can be one of the following staff members: admin, pharmacist or assistant. If he is registered in the database, he can log in and do certain activities according to his rights. The application is divided into several interfaces as follows:

Log In interface

When the user starts the application, the Log In interface will open:

A screenshot of a Windows-style window titled "Log in:". Inside the window, there is a central gray box containing the text "Log in:" in bold. Below this, there are two input fields: "Username:" followed by a text box, and "Password:" followed by a text box. At the bottom of the gray box, there is a button labeled "Sign in!".

The user will have to enter the username and the password. Once the "Sign in!" button is pressed, a query will be made in the database and the user's existence will be verified. If the database does not contain the user, an error will be displayed on the screen:

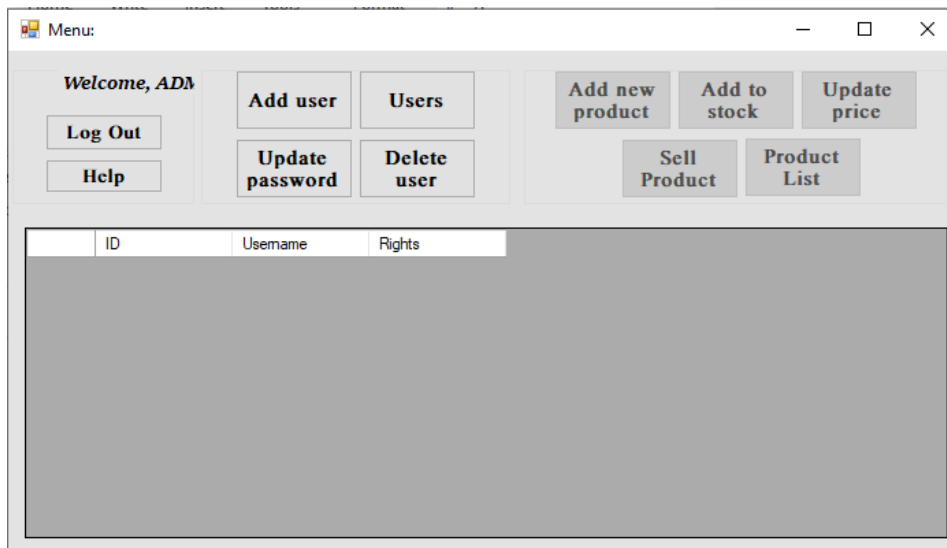
A screenshot of a small error dialog box. The title bar says "ERROR" with a close button (X). The main text inside the box reads "Incorrect username or password!". At the bottom right, there is an "OK" button.

If the database contains the user, he has successfully logged in and the interface will change to the menu interface.

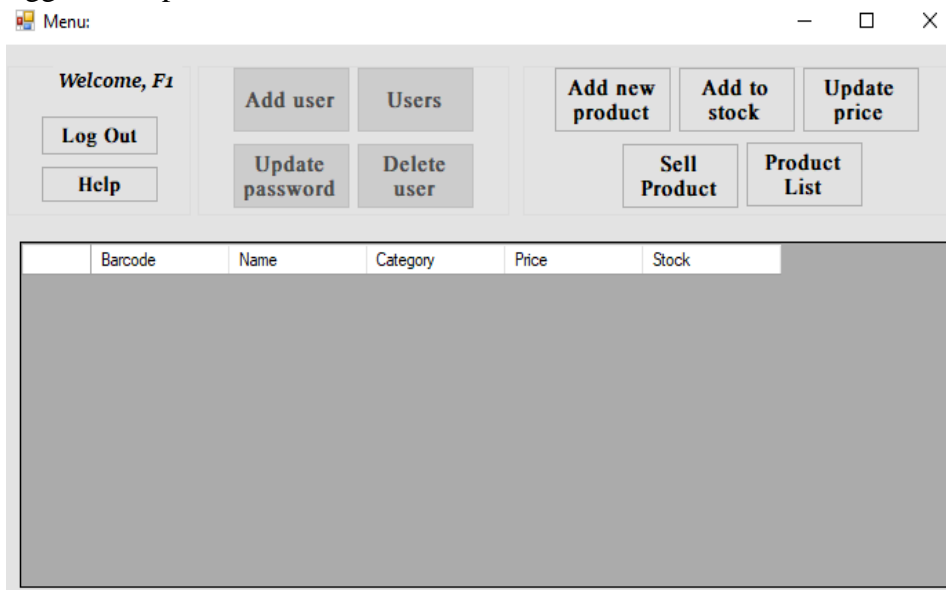
Menu interface

Once the user has successfully logged in, its rights are verified and depending on them, the Menu interface will be displayed with certain buttons disabled. (this depends on his rights)

User logged in as admin:



User logged in as pharmacist/assistant:



Add new product interface

When the employee presses the "Add new product" button from the Menu interface, this interface will be displayed:

The screenshot shows a window titled "Add new product:". Inside, there's a section labeled "NEW PRODUCT" containing five input fields: "Barcode :" (a text box with a cursor), "Category :" (a dropdown menu), "Product :" (a dropdown menu), "Price :" (a text box with "0.00" and up/down arrows), and "Initial stock :" (a text box with "0" and up/down arrows). An "Add!" button is located at the bottom right of the window.

After the employee fills in the fields and press the "Add!" button, a new product is inserted in the database. He must pay attention to some aspects because some errors may appear:

- The barcode needs to be unique in the database. If the product with the entered barcode exists, this error will be displayed:



The product with given barcode or name already exists!

OK

- Also, the barcode can't be a string. If it is not a number, this error will be displayed:



Please insert valid number for barcode.

OK

- If the "Category" field is completed on the keyboard and is not selected from the dropdown menu, the employee must ensure that the entered category is valid. Otherwise, this error is displayed:



Please select a valid product category.

OK

This is the list with the valid categories, that appears in the dropdown menu:

Analgesics
Anti-viral
Anti-fungal
Antibiotics
Antitussives
Antispasmodic
Beta blockers
Corticosteroid
Diuretics
Immunosuppressant's
Sedatives
Vitamin supplements

- If the "Product" field is completed on the keyboard and is not selected from the dropdown menu, the employee must ensure that the entered product is valid. Otherwise, this error is displayed:



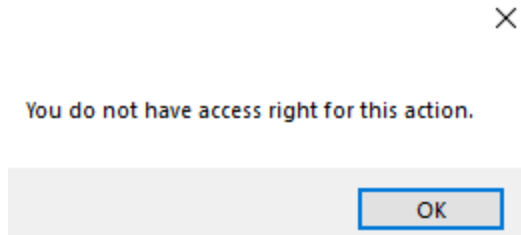
Please select a valid product name.

OK

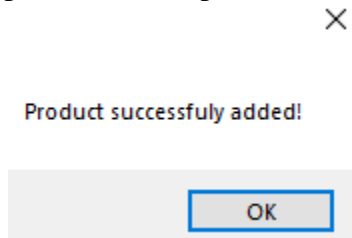
This is the list with the valid products, by each category:

Analgesics : Acetylsalicylic acid, Paracetamol, Ibuprofen, Ketoprofen, Naproxen
Anti-viral: Acyclovir, Tamiflu
Anti-fungal: Econazole, Fluconazole, Miconazole, Ketoconazole
Antibiotics: Azithromycin, Amoxicillin, Doxycycline, Cephalexin, Clindamycin
Antitussives: Robitussin, Codeine
Antispasmodic: Bentyl, Librax, Dicycloverine, Hyoscine, Atropine
Beta blockers: Metoprolol, Acebutolol, Bisoprolol, Propranolol
Corticosteroid: Dexamethasone , Cotelone, Medrol, Prednicot
Diuretics: Bumex , Spironolactone, Furosemid, Zaroxolyn
Sedatives: Xanax , Librium, Diazepam
Vitamin supplements: Vitamin D3, Biotin, Niaspan, Vitamin C

- On this interface, the user can't introduce negative price and stock, so an error will not be thrown at this attempt.
- If the user is logged in as assistant and is trying to introduce a new product, this error is displayed:



If the operation is completed successfully, this message will be displayed:



Add to stock interface

When the employee presses the "Add to stock" button from the Menu interface, this interface will be displayed:

A screenshot of a Windows application window titled "Add to stock:". The window has standard minimize, maximize, and close buttons in the top right corner. The main content area is titled "ADD TO PRODUCT STOCK" and contains two input fields: "Barcode:" followed by a text box, and "Quantity:" followed by a spinner box showing the value "0". At the bottom right of the window, there is an "Add" button with a blue border.

After the employee fills in the fields and presses the "Add" button, the stock field from the product that corresponds to the barcode is updated. He must pay attention to some aspects because some errors may appear:

- If the product is not found in the database, this error is displayed:



This product doesn' exist.Try to add a new product of this type.

OK

- On this interface, the user can't introduce negative quantity, so an error will not be thrown at this attempt.
- If the user is logged in as assistant and is trying to update the stock for a product, this error is displayed:



You do not have access right for this action.

OK

If the operation is completed successfully, this message will be displayed:

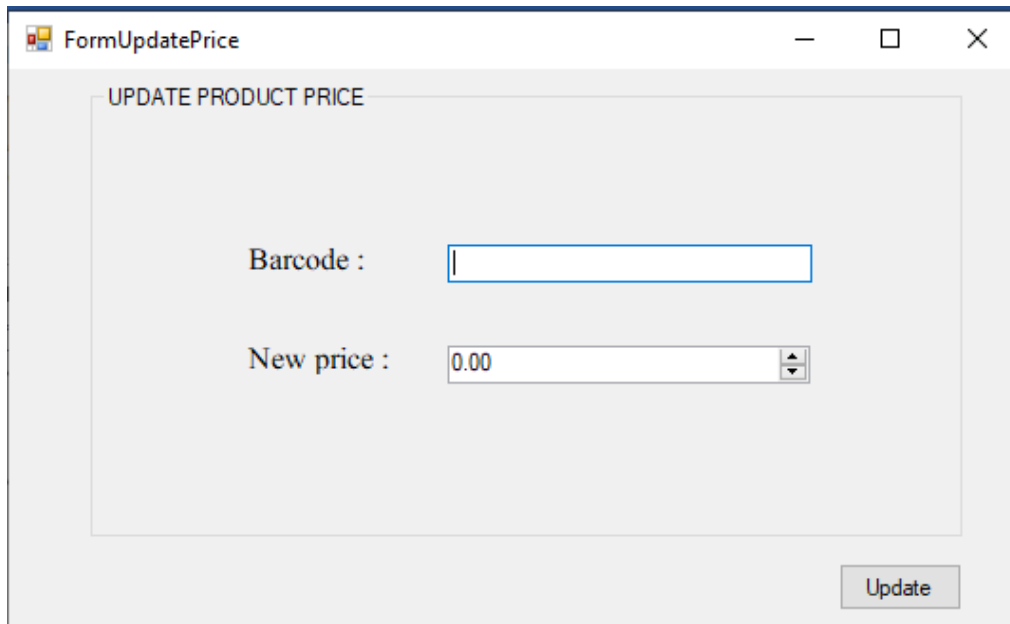


Stock was successfully updated

OK

Update price interface

When the employee presses the "Update price" button from the menu Interface, this interface will be displayed:

A screenshot of a Windows application window titled "FormUpdatePrice". The window contains a form titled "UPDATE PRODUCT PRICE". The form has two input fields: "Barcode :" with a text box and "New price :" with a numeric spinner box showing "0.00". At the bottom right of the form is an "Update" button.

After the employee fills in the fields and presses the "Update" button, the price field from the product that corresponds to the barcode is updated. He must pay attention to some aspects because some errors may appear:

- If the product is not found in the database, this error is displayed:

×

This product doesn' exist.Try to add a new product of this type.

OK

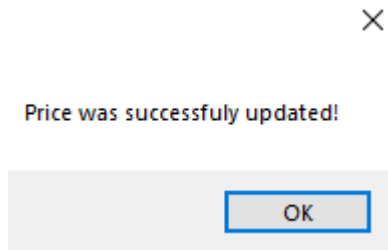
- On this interface, the user can't introduce negative price, so an error will not be thrown at this attempt.
- If the user is logged in as assistant and is trying to update the price for a product, this error is displayed:

×

You do not have access right for this action.

OK

If the operation is completed successfully, this message will be displayed:



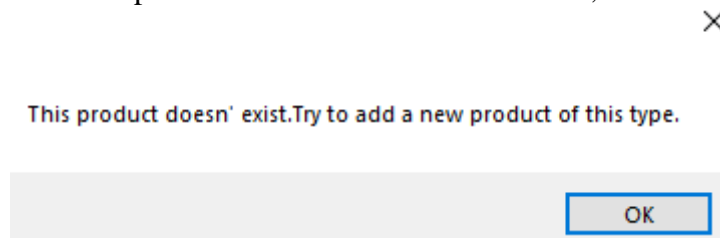
Sell product interface

When the employee presses the "Sell product" button from the Menu interface, this interface will be displayed:

A screenshot of the "Sell product" interface. It has a title bar "Sell product:" with standard window controls. The main area is titled "SELL PRODUCT". It contains two input fields: "Barcode:" with a text box, and "Quantity:" with a spinner box showing "0". At the bottom right is a "Sell" button.

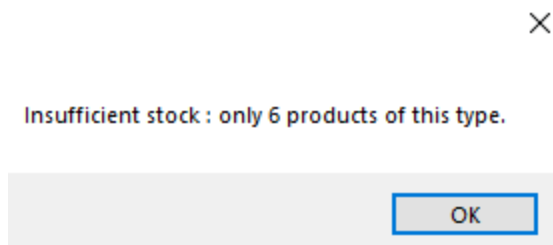
After the employee fills in the fields and presses the "Sell" button, the stock field from the product that corresponds to the barcode is updated. He must pay attention to some aspects because some errors may appear:

- If the product is not found in the database, this error is displayed:

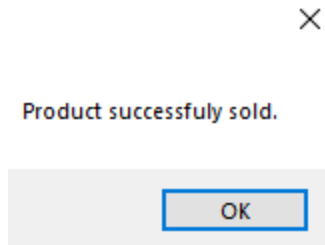


- On this interface, the user can't introduce negative quantity, so an error will not be thrown at this attempt.

- If the employee tries to sell more products than they are in stock, this error is displayed:



If the operation is completed successfully, this message will be displayed:



Selecting users from the Menu interface

When the employee presses the "Product List" button from the Menu interface, the interface will be displayed with all products from the database:

Menu: — □ ×

Welcome, F1

Log Out

Help

Add user

Users

Update password

Delete user

Add new product

Add to stock

Update price

Sell Product

Product List

	Barcode	Name	Category	Price	Stock
▶	12	Paracetamol	Analgesics	100	3
	13	Ketoconazole	Anti-fungal	21	45
	333	Acyclovir	Anti-viral	15.6	259
	3223	Miconazole	Anti-fungal	34	12
	4234	Doxycycline	Antibiotics	15	30
	12312	Codeine	Antitussives	12	34
	12321	Cephalexin	Antibiotics	0	0
	421421	Bentyl	Antispasmodic	0	0
	321321	Clindamycin	Antibiotics	0	0
	43243	Bisoprolol	Beta blockers	0	0

Add user interface

When the admin presses the "Add user" button from the Menu interface, this interface will be displayed:

Add new user: — □ ×

ADD NEW USER

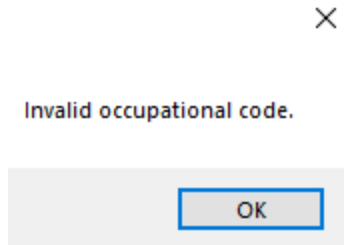
Username:

Password:

Occupation code:

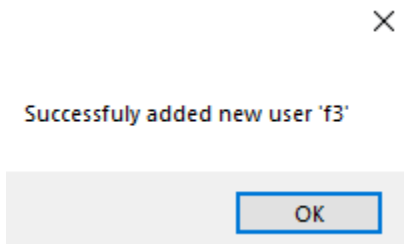
Add

After the admin fills in the fields and presses the "Add" button, a new user is inserted in the database. He must pay attention to the "occupation code" field because, if he enters an invalid code, this error will be displayed:



The valid occupation codes are: -1, 0, 1.

If the operation is completed successfully, this message will be displayed:



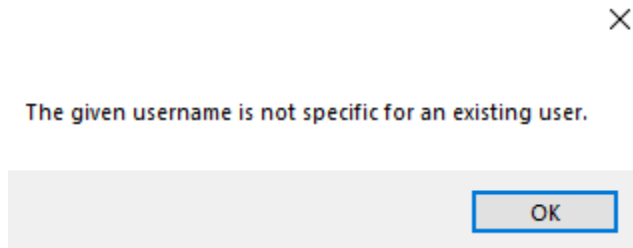
Update user interface

When the admin presses the "Update password" button from the Menu interface, this interface will be displayed:

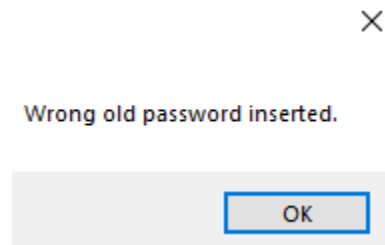
A window titled "Update Password:" with standard window controls (minimize, maximize, close). Inside the window, there is a section titled "UPDATE PASSWORD" containing three input fields: "User:", "Old password:", and "New password:". At the bottom right of the window is an "Update!" button.

After the admin fills in the fields and presses the "Update!" button, the selected user from the database is updated with the new password. He must pay attention to some aspects because some errors may appear:

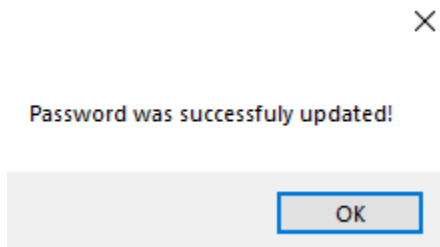
- The user must be in the database. If it does not exist, this error will be displayed:



- The old password must be correct. Otherwise, this error will be displayed:

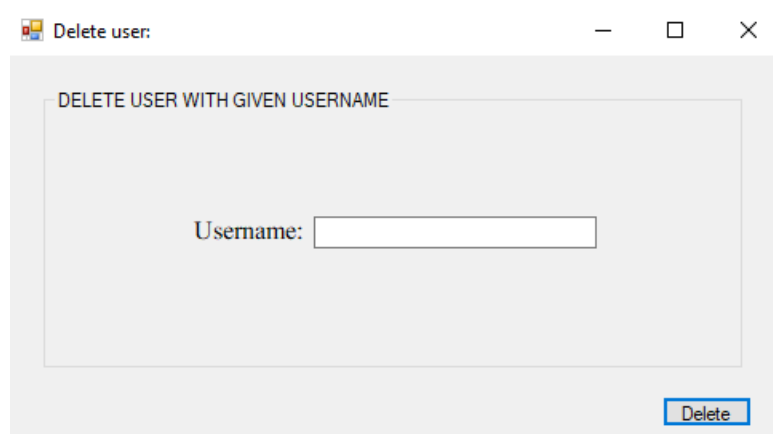


If the operation is completed successfully, this message will be displayed:

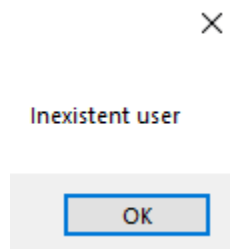


Delete user interface

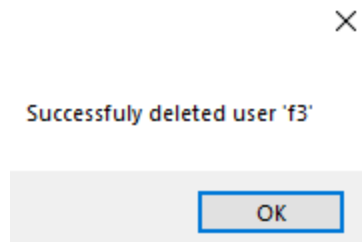
When the admin presses the "Delete user" button from the Menu interface, this interface will be displayed:



After the admin fills in the "Username" field and clicks the "Delete" button, the user with that username will be deleted from the database. The admin must ensure that the user exists in the database. Otherwise, this error will be displayed:

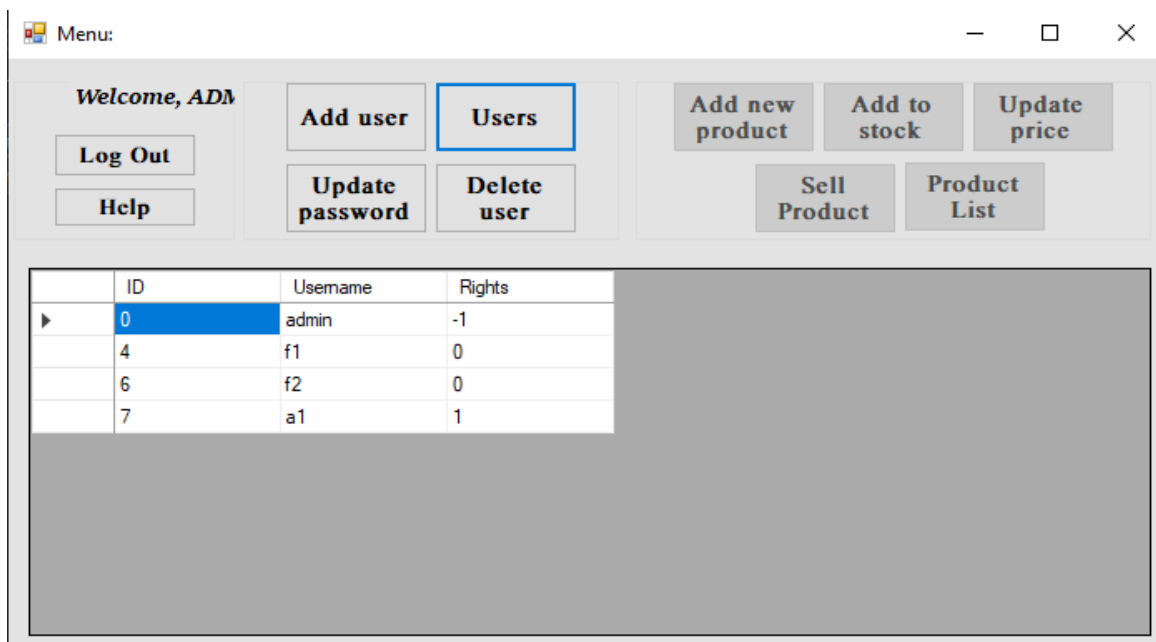


If the operation is completed successfully, this message will be displayed:



Selecting all users from the Menu interface

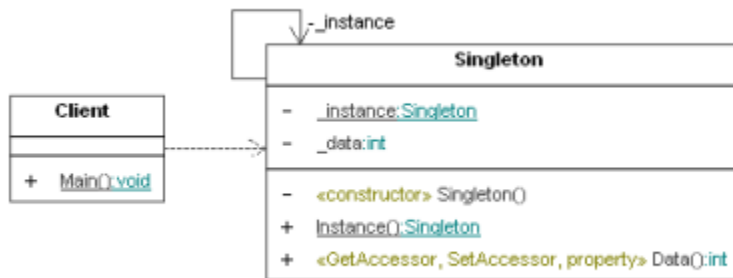
When the admin presses the "Users" button from the Menu interface, the interface will be displayed with all users from the database:



How the database works

For the application, SQLite database was used. The System.Data.SQLite package was installed to create a database connection. All CRUD operations were implemented in DB class from DataBaseManager dll.

Because we don't need more than one instance of the class, the singleton design pattern was used.



```

private DB(string dbName)
{
    if (Path.GetExtension(dbName) != ".db")
    {
        throw new InvalidExtensionException();
    }
    _connection = new SQLiteConnection(String.Format("Data Source={0}; Version = 3; New = True; Compress = True; ", dbName));
    try
    {
        _connection.Open();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
  
```

```

public static DB GetInstance(string dbName)
{
    if (instance == null)
    {
        instance = new DB(dbName);
    }
    //If the dbName is different from the path of the current connection, change it.
    if (dbName.Substring(0, dbName.Length - 3) != instance.Connection.DataSource)
    {
        instance.ChangeConnection(dbName);
    }
    return instance;
}
  
```

An example of an implemented CRUD operation is shown below:

```

public List<User> SelectAllUsers()
  
```

```

{
    List<User> users = new List<User>();
    SQLiteCommand command;
    command = _connection.CreateCommand();
    command.CommandText = "SELECT * FROM users";
    SQLiteDataReader sqlite_datareader = command.ExecuteReader();
    while (sqlite_datareader.Read())
    {
        int id = sqlite_datareader.GetInt32(0);
        string username = sqlite_datareader.GetString(1);
        string password = sqlite_datareader.GetString(2);
        int rights = sqlite_datareader.GetInt32(3);
        User usr = new User(id, username, password, rights);
        users.Add(usr);
    }
    return users;
}

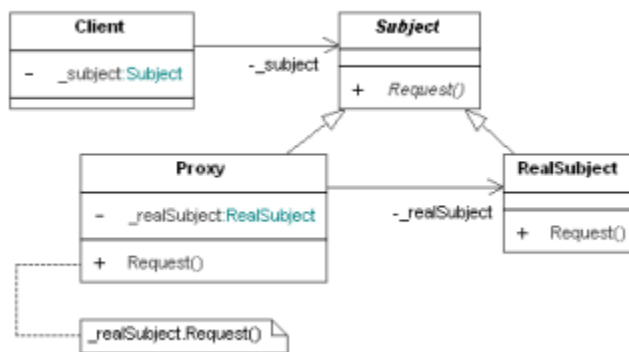
```

In the PharmacyManagementDLL, methods from this class are called like this:

```
Product product = _dbInstance.SelectProduct(barcode);
```

How users' rights have been limited

To limit and differentiate the user rights, the proxy design pattern was used.



In the PharmacyManagementDLL, this pattern was implemented with this classes:

- ProxyActionManager as Proxy;
- RealActionManager as RealSubject;
- IActionManager as Subject;

Depending on the user's rights, ProxyActionManager delegates the operation to the RealActionManager. The RealActionManager class accesses the database via the DB class from the DataBaseManager.dll . ProxyActionManager is also a singleton.

This is an example how the ProxyActionManager delegates the operation to the RealActionManager:

```
public void AddNewProduct(Product newProduct)
{
    try
    {
        //gives access only to those who are Pharmacists
        if (_permissions.RightsList(_currentUser.Rights).Contains(Constants.ModifyProductsDBRight))
        {
            _realActionManager.AddNewProduct(newProduct);
        }
        else
        {
            throw new PermissionDeniedException();
        }
    }
    //forwards exceptions for display a suggestive message on GUI
    catch (ConstraintViolatedException exc)
    {
        throw exc;
    }
    catch (InvalidStockException exc)
    {
        throw exc;
    }
}
```

The RealActionManager will access the database like this:

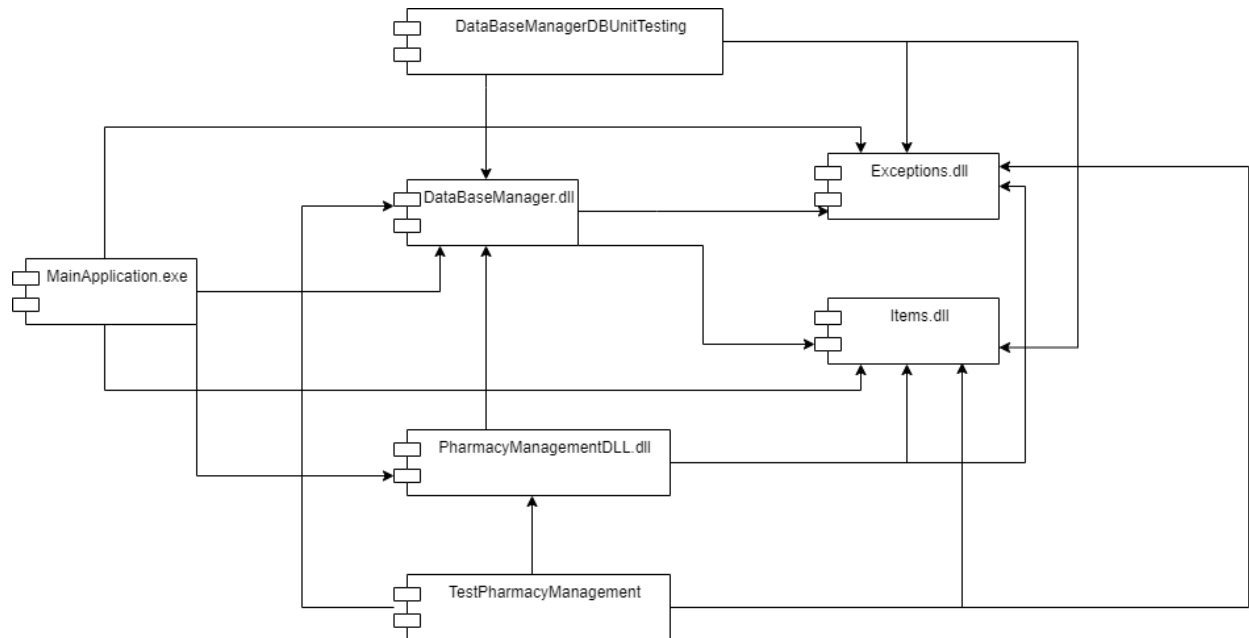
```
public void AddNewProduct(Product newProduct)
{
    if (newProduct.Stock < 0)
    {
        throw new InvalidDataException("Stock cannot be less than zero");
    }
    if (newProduct.Price < 0)
    {
        throw new InvalidDataException("Price cannot be less than zero");
    }
    try
    {
        _dbInstance.Insert(newProduct);
    }
    catch (ConstraintViolatedException exc)
    {
        throw new ConstraintViolatedException("The product with given barcode or name already exists!");
    }
    catch (InvalidStockException exc)
    {
        throw new InvalidStockException();
    }
}
```

DLL's

This application use this dll's:

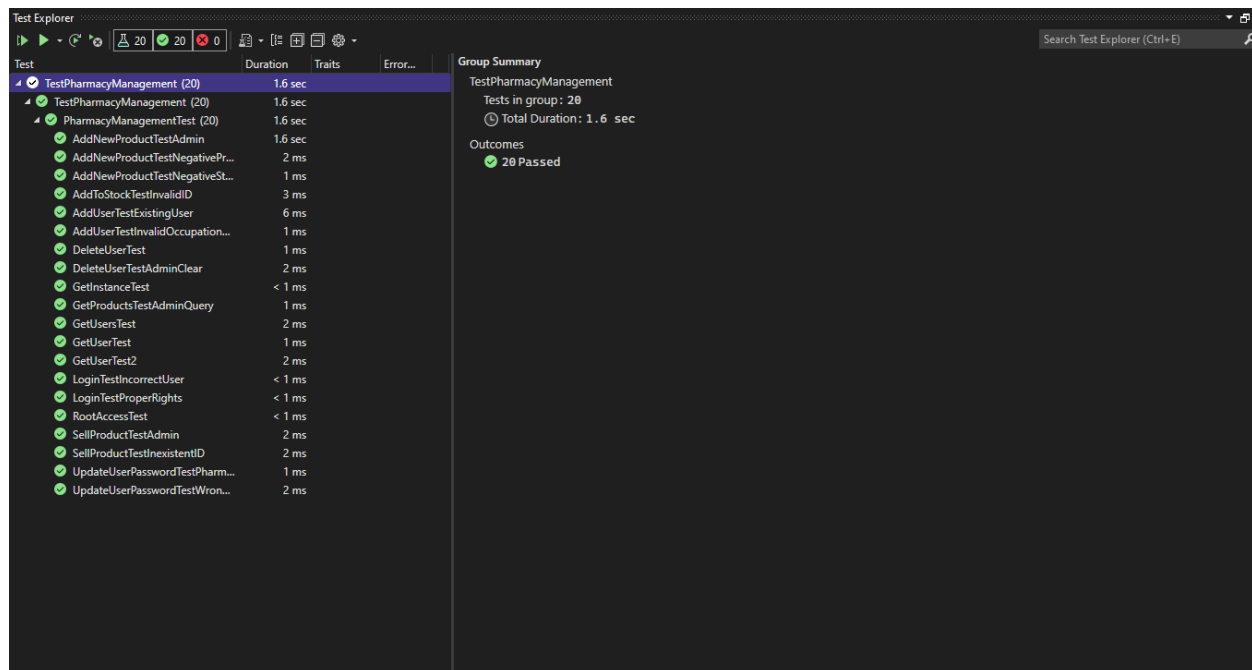
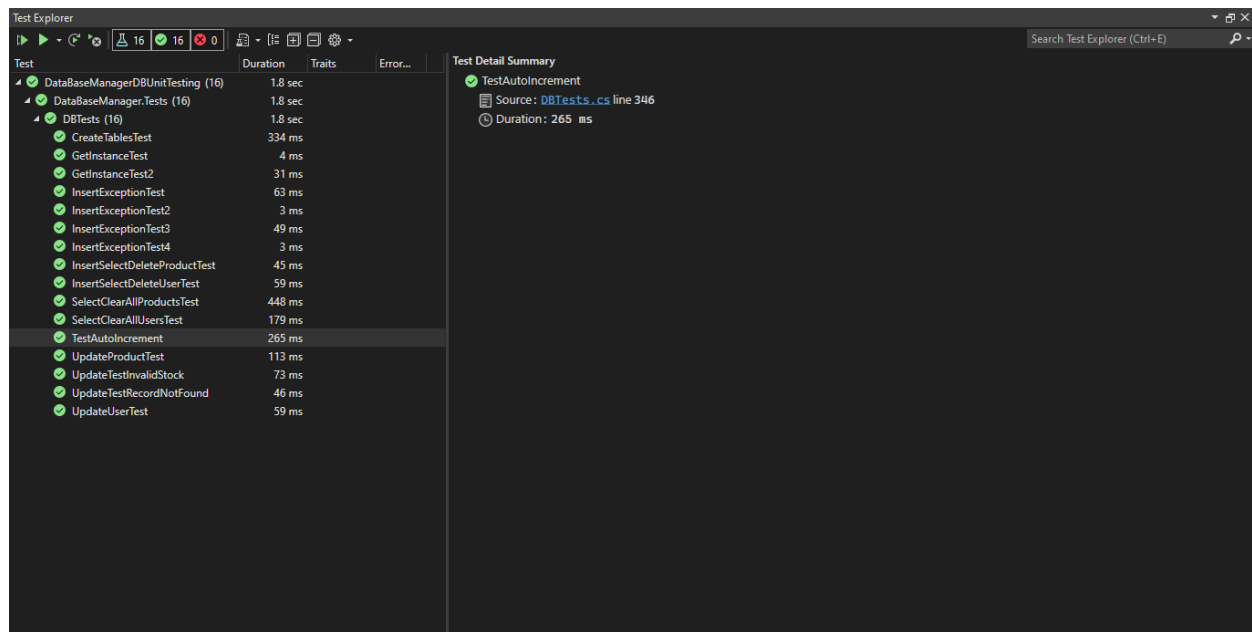
- Items.dll
- PharmacyManagementDLL.dll
- Exceptions.dll
- DataBaseManager.dll

Communication between modules are shown in this image:



Unit testing

Both DataBaseManager and PharmacyManagementDLL have the unit tests made, the results being in the next images:



As you can see in the pictures, all the tests are passed.

SRS DOCUMENTATION:

Software Requirements Specification

for

PharmApp

Version 1.0 approved

**Prepared by Chiriac Dan - Constantin
Cuptor Iuliana – Stefania
Ratoi Liviu
Ursachi Gabriela**

Faculty of Automatic Control and Computer Engineering

22 May 2022

Table of Contents

Table of Contents	XXV
Revision History	XXV
1. Introduction	26
1.1 Purpose.....	26
1.2 Document Conventions.....	26
1.3 Intended Audience and Reading Suggestions	26
1.4 Product Scope.....	26
1.5 References	26
2. Overall Description	27
2.1 Product Perspective	27
2.2 Product Functions	27
2.3 User Classes and Characteristics	28
2.4 Operating Environment.....	28
2.5 Design and Implementation Constraints	28
2.6 User Documentation	28
2.7 Assumptions and Dependencies	28
3. External Interface Requirements	28
3.1 User Interfaces.....	28
3.2 Hardware Interfaces	29
3.3 Software Interfaces	29
3.4 Communications Interfaces	29
4. System Features.....	29
5. Other Nonfunctional Requirements	29
5.1 Performance Requirements	29
5.2 Safety Requirements	29
5.3 Security Requirements	30
5.4 Software Quality Attributes	30
5.5 Business Rules.....	30
6. Other Requirements	30
Appendix A: Glossary	30
Appendix B: Analysis Models.....	30

Revision History

Name	Date	Reason For Changes	Version
PharmApp	22.05.2022	Release	1.0

1. Introduction

1.1 Purpose

PharmApp is a desktop application for managing the staff of a pharmacy, as well as the administration of marketed products. On the one hand, it deals with the sale and supply of medicines, as well as checking stocks. Also, it considers the organization of the pharmacy staff and the allocation of rights according to the actions that each type of employee can exercise.

It does not require a network connection, it uses local data storage to retain the necessary information. Details and specifications of the functionality of this application are defined in sections 3 and 4. An overview of the application is given in section 2, and a list of requirements is given in section 5. This documentation refers to the release version on this application.

1.2 Document Conventions

This document follows the IEEE standard formatting for software development. The standard defines a regular formatting this document follows including writing to be done in third-person, passive voice as well as readable and grammatically correct text.

1.3 Intended Audience and Reading Suggestions

This document is intended for both users and developers. Since a user needs information on how to use this application, he should continue to read section 3, 4 and 5. In the case of developers, because they need a detailed and depth understanding about the application, it is recommended to read whole document, with increased attention on section 2.

The document starts off with an overview of the functions and specifications for this app in section 2, then moves on to describe the requirements for interfacing with external hardware and software in section 3. Section 4 describes the application functions in great detail and section 5 lists various requirements the application must respects after completion.

1.4 Product Scope

PharmApp is a simple , easy-to-use application developed on .NET platform. It is being intended for the pharmaceutical field and it provides a solution for maintaining and retrieving data related to a pharmacy management in a secure context. For this purpose, the application has an authentication mechanism and it is intended for use only by users who already have an account created or by an administrator who has an account set up at the time of application release.

It does not require users to have advanced computer knowledge in order to use the application, because it offers a simple graphical interface that contains graphical controls suggestive of different types of actions.

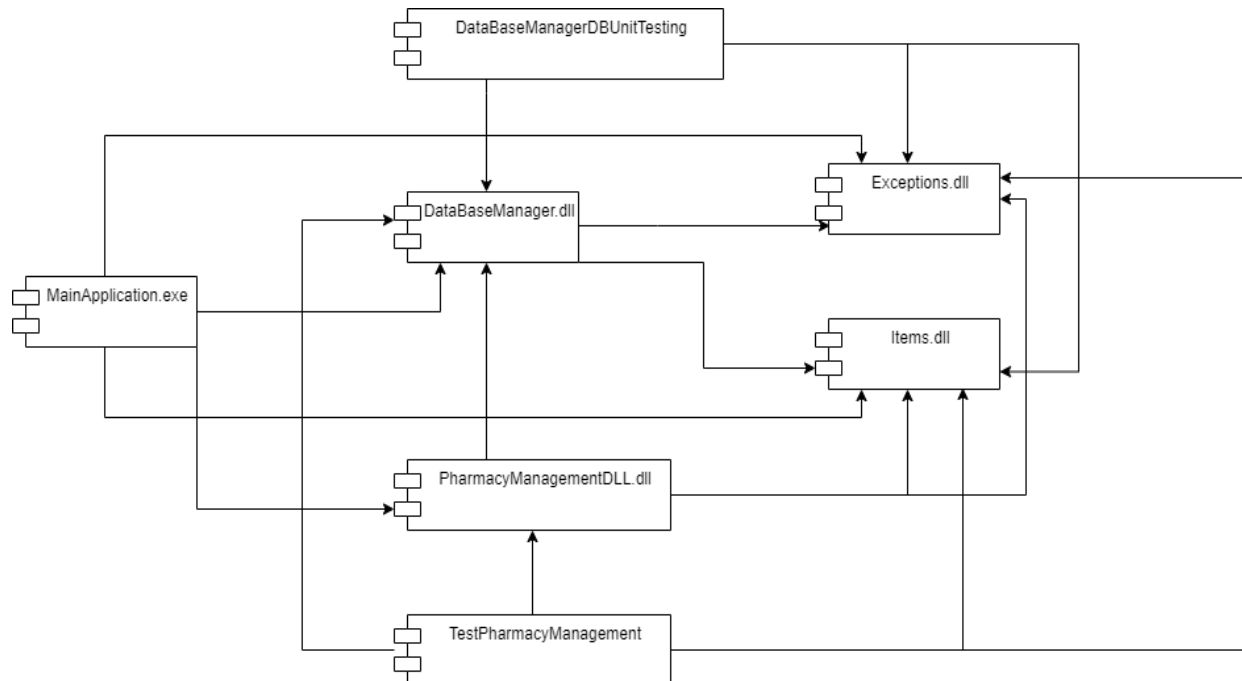
1.5 References

N/A

2. Overall Description

2.1 Product Perspective

PharmApp is an academic project, realized to practice with programming engineering concepts. It is a self-contained project, implemented as a Windows Forms Application using C# programming language and .NET framework. The data required for this application is stored within two tables from a SQLite database.



In the figure above are presented the application components and their dependency relations. Each module has been implemented as a DLL (Dynamic Link Library) file.

ExceptionsDll module provide custom exceptions definitions;

ItemsDll module is used for providing structures and basic object used in context of application;

DatabaseDll is a module that handles the interaction between application and SQLite database;

PharmacyManagementDll module implements the protection proxy mechanism in accordance with established acces right rules.

For each functional module is developed an unit test module.

Application entry point is represented by MainApplication.exe.

2.2 Product Functions

The major functions the PharmApp must perform for the end user are the following:

- ✓ Authentication mechanism - entering username and password account;
- ✓ Possibility to manage application users – this function is specific only for admin instance;
- ✓ Possibility to handle with pharmacy products, in terms of sale, stock update , or just information queries – this functionality is open to all pharmacy employees except the admin.

2.3 User Classes and Characteristics

This application is designed to be easy to use. It intends to provide the necessary functionalities for three categories of users:

The first class of users is represented by **pharmacy administrator** : it is required to be a person with good organizational skills because it's in his responsibility to manage database access.

Another class is represented by **pharmacy assistant** – this category must have knowledge of the types of drugs and therapeutic indications .

The third class of user is the **pharmacist** - an user with knowledge similar to assistants, but some new characteristics : attention to detail and a few years of experience in pharmaceutical field.

2.4 Operating Environment

Since PharmApp is developed using .NET Framework, it is compatible with only all version of Windows operating system starting with Windows XP Service Pack 3. Also, .NET Framework 4.7.2 or a newer version is required to be installed on the system.

2.5 Design and Implementation Constraints

Since PharmApp is being designed and implemented in a single semester as a project for Programming Engineering subject, it is possible that time is the most limiting factor in this development cycle. Another constraint is that application is supposed to have an authentication mechanism, to prevent unauthorized access, so we need a deep understanding of protection proxy pattern and databases knowledge, but also hash mechanism knowledge for securely storing sensitive data.

2.6 User Documentation

PharmApp contains a clear and explicit in-app help, accessible on Help Button Click. It provides the user guidance regarding the use of the application and the interpretation of the information meaning.

2.7 Assumptions and Dependencies

We are assuming that it will be a preconfigured admin account when launching the application. Otherwise, the application would be impossible to use .

3. External Interface Requirements

3.1 User Interfaces

The user interface will be separated into two main components : one window for authentication mechanism and another one for displaying buttons for each action that can be done in the context of managing a pharmacy.

When starting the application, a window with two text boxes will be displayed to enter account credentials : username and password. If the data entered does not correspond to an existing user,

an error message will be displayed, and use of the application will be impossible until valid data is entered.

If we assume that valid data has been entered, then the second window will be displayed. This one contains the username of authenticated user in the upper left corner and a 'logout' button in the upper right corner. It also contains one button for each possible action. If there are some disabled buttons, this means that the logged user hasn't access right for those actions. If the user access a button on whose action is included in his access rights, then a new smaller window will be displayed to allow them to insert the necessary data, in order to insert, update, delete or just query the information from database. If user tries to insert invalid data, they will be alerted by an error message displayed.

There is an explicit indicator inside user interface to open in-app help. It is displayed as '?' button.

3.2 Hardware Interfaces

There is not much heavy hardware needed to run this application. Hardware interfaces include a display monitor, a mouse and a keyboard. The mouse is used to interact through left click to allow the user to interact with certain objects displayed on the monitor. The keyboard is used to introduce necessary data in specific text or number entry spaces.

3.3 Software Interfaces

To run this application the user's system must runs any Windows operating system, starting with Windows XP Service Pack 3 and to have installed .NET Framework 4.7.2 or a newer version.

3.4 Communications Interfaces

No communication standard is required for this application. It is a standalone application, without dependencies on remote resources. It is also an offline application, so it's not necessary a network connection.

4. System Features

This application features are covered in-depth in the user guideline.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

This application is able to run on all version of Windows, starting with Windows XP Service Pack 3. It is recommended to have at least 512 MB RAM and at least 1 GHz CPU. No dedicated graphics card is needed.

5.2 Safety Requirements

Using this application does not present any safety risks.

5.3 Security Requirements

Since the authentication data is stored in a database, only the administrator will have access to this information. If an unauthorized person manages to access information about users' accounts, this could jeopardize the entire activity of the pharmacy. So it is recommended to update user's password constantly and to set strong password each time .

5.4 Software Quality Attributes

PharmApp is thought to be a simple, fast, easy to use and useful tool that offers the possibility of a correct management of databases. This software must be robust and as bug-free as possible to ensure a good management of a pharmacy activity. This application is designed to be used for as long as possible. There are been chosen maintainability and reliability over portability.

5.5 Business Rules

It is the policy of the development team to follow all codes of conduct established by the University.

6. Other Requirements

Appendix A: Glossary

N/A

Appendix B: Analysis Models

All diagrams are available in the electronic documentation.