

Library Project

Documentation

1. **Meet the team Durian:** ("{0} – {1} {2}", TAid, FirstNameBg, SecondNameBg):

- 1) *IlianaB* - Илиана Бобева
- 2) *pzubev* - Петър Зубев
- 3) *dani.ivanov.96* - Дани Иванов
- 4) *Andro0* - Андрей Киров
- 5) *Ioderunner* - Никола Богомиров
- 6) *nikistefanov* - Николай Стефанов
- 7) *messenger* - Михаил Петров
- 8) *D.Yanis* - Димитър Янис

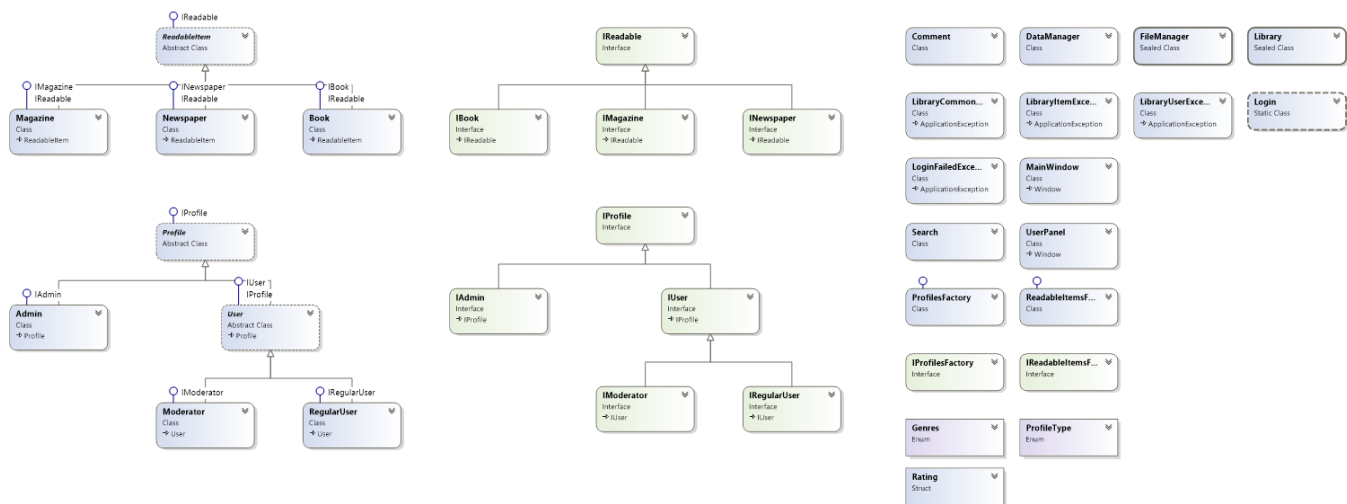
Program description: Product deliverable is an implemented logic and structure of an online Library API, allowing users to read and comment on books, newspapers and magazines that are present in the Library. The delivered product implements a clean UI via WPF/XAML.

2. **Repository Location:** <https://github.com/IlianaB/TA-TeamWork-CSharpOOP>

All code under MIT license.

3. **Description** of underlying logic and implementation: We have used the Simple Factory design pattern as a creational pattern of all objects in the Library. This design pattern separates the logic of creating objects from the business logic of the App. The two factories - [ProfilesFactory](#) and [ReadableItemsFactory](#), work with interfaces as return types for maximal level of abstraction. An additional design patten is Singleton (which is the Library itself).

The **Library Diagram:**



The abstract Profile class implements the **IProfile** interface. This class holds the common properties Name, Password and ProfileType for all profiles in the App. The **Profile** class is being inherited by the abstract **User** class and **Admin** class, which can be instantiated. Administrator have only managing authorities - to add and remove readable items on/from the Library. All the other users can keep statistics on their activities in the Library by adding or removing items in their custom lists Wish-to-read-list, currently-reading and read items. **Moderator** and **Regular user** are treated as subclasses of the abstract User class. The difference between them is that the Moderator can delete Comments, and Regular user can only report Comments for deletion. The readable items in our virtual Library are of 3 types: **Books**, **Magazines** and **Newspapers**, which inherit the abstract class ReadableItem. Those items a different set of properties but share a common genre from a predefined set, listed in the Genres enumeration, as well as some other identical properties, which are implemented in the base class. The **DataManager** class takes care of reading and writing information about profiles and readable items from text files (our data base). All the information about user and readable items is stored in the DB folder in our project structure.

The App provides searching in its own DB, through the Search class implementation. All readable items have a **Rating** structure. There are four Exception classes which throw user friendly messages.

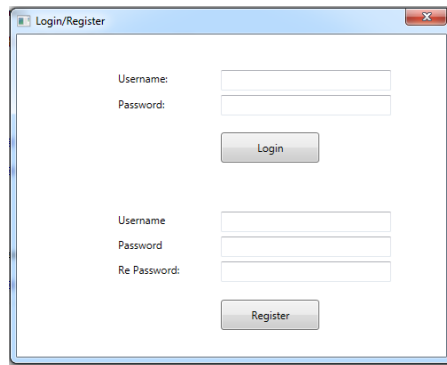
4. Requirements for the project vs. project implementation:

All requirements are being met incl. those for classes, interfaces enumerations, OOD, structures, polymorphism, inheritance, abstraction, exception handling.

The program implementation uses highest level of abstraction and polymorphism.

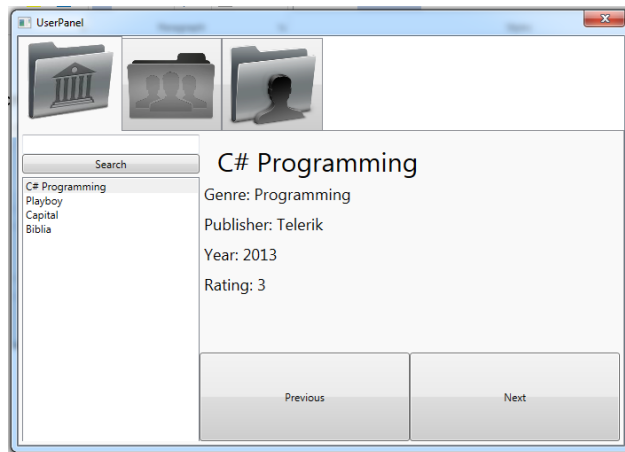
5. UI

Start screen:

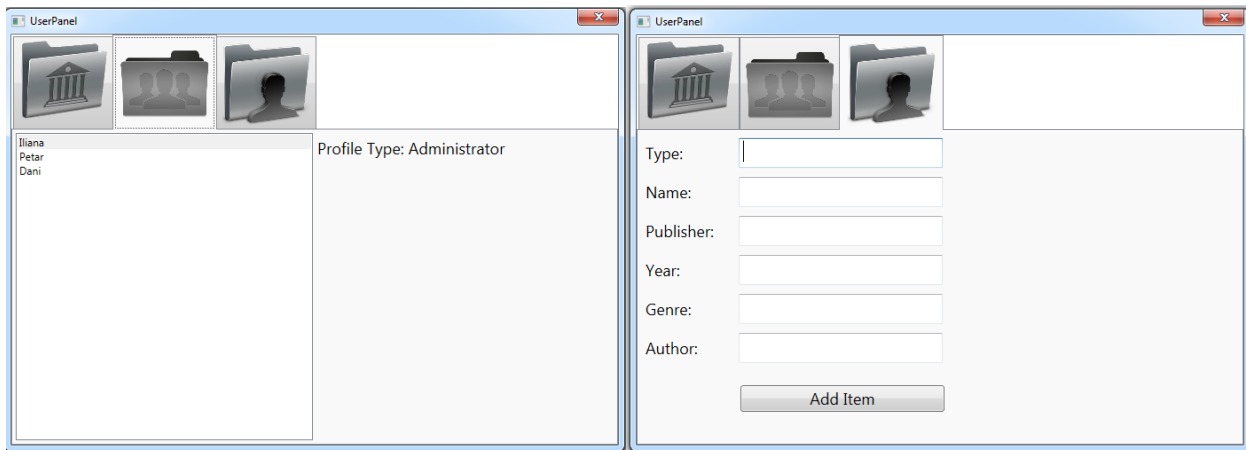


A window titled "Login/Register" with two sections. The top section is for login, with fields for "Username:" and "Password:" followed by a "Login" button. The bottom section is for registration, with fields for "Username", "Password", and "Re Password:" followed by a "Register" button.

Follow on screens:



A window titled "UserPanel" displaying a search results page. On the left is a search bar with a "Search" button and a list of results: "C# Programming", "Playboy", "Capital", and "Biblia". The main area shows details for "C# Programming": "Genre: Programming", "Publisher: Telerik", "Year: 2013", and "Rating: 3". At the bottom are "Previous" and "Next" buttons.



Two side-by-side "UserPanel" windows. The left window shows a list of users: "Iliana", "Petar", and "Dani", with "Profile Type: Administrator" displayed. The right window shows a form for adding a new item, with fields for "Type:", "Name:", "Publisher:", "Year:", "Genre:", and "Author:", followed by an "Add Item" button.

All entered information is being validated and pop up messages appear in case of validations being not passed through. All fields and menus are functioning as per the logic of the program.