|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/6wx3_nO0-B-v9q4dJPudU9qpXcrb_fl9t6qKqsYKGOJXrrY3CiXEA1RwZNubVZTYzdfQBhqGOLJL_8ayTPeFB_J-RArQ8siunxp6fANDVMl5RswOI_r5sEXmpchzT6b8a47avuMP | **DOKUZ EYLÜL UNIVERSITY**  **ENGINEERING FACULTY**  **DEPT. OF COMPUTER ENGINEERING** |

**LIBRARY FOR EVERYONE**

**CME 4414 Advances in Web Technologies**

**Term Project Report**

**Final Phase**

**2019-2020 SPRING**

**2013510074 MEHMET OĞUZHAN URHAN**

**2015510035 BERKE KADAM**

1. **System Requirements**
   1. **Introduction**

This will be a web application project that aims to deliver books to people without any fees. The books will be exchanged between the customers but instead of 1-on-1 exchange system all the books will be stored in a pool so that everyone can get them without any limitations or trying to find someone to exchange. You can think of it like an online book club.

There will be a list that every member needs to fill that shows the books they can lent to somebody else. Every book will be stored in the database and will be filtered in order to help customers get what they want. The project depends on its customers to be efficient so a powerful startup will be necessary. We are going to try getting a deal with a cargo company in order to make all the book deliveries free in return of advertisements displayed on the website. We are expecting to attract many people's attention with this service being completely free and delivering the books you want to your doorstep.

* 1. **Business Rules & Requirements**

The purpose of our project is to create a system that helps those who want to read books but cannot allocate financial resources for it. We can see it as a kind of social responsibility project. For this purpose, it is wide enough to appeal to everybody in the country because every user can contribute to the library by donating books. We want to automate and digitize this situation, because in the digitalized world, people have a hard time doing things physically. We digitalize systems such as book donations and public libraries and make them accessible to everyone and even take books to their houses with cargo support.

The system is controlled by the "Library for Everyone" company. What we need in the first place will be a lot of users, because this is how the system is maintained. So, we can say that it is a very user based web project. If we come to the technical details of the project and what the users can do and restrict on the site, firstly registration to the site will be required. Those who register will have the right to save their books on the system. One person has the right to donate as many books as he wants so that the pool will expand and we will not have any problems in stock. However, there are limits on borrowing books; because we want every user to benefit equally. First of all, no user can request two from the same book. A user can request a maximum of 1 books at a time and limitless in total.

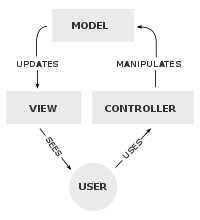
The most important of the system's needs is delivery, but this part concerns logistics officials, not software engineers. When we analyze the needs for the software, we need a strong and stable database, as we mentioned in the last paragraph. When we handle three important functions such as loading books, inquiring and requesting, we have a smooth software.

Project's success is highly dependant to the database. It has to sustain all the search operations and store the details of both users' and books'. Therefore, entities and the relationships have to be perfect. Users' information (name, address, previously taken books, given books and identity number for uniqueness) and books' information (name, author, publisher, ISBN, previous takers) have to be stored and all those information should not affect the performance of the system. Users is going to give all the information including the books that they can give to the community. Users have the authorization to edit and update their personal information. However, only admins have the authorization of deleting the books from the system (excluding the special cases like users wanting to take back the book they donated).

* 1. **Software Architecture**

**2.1 Model – View – Controller Architecture**

The Model-View-Controller, shortly MVC, is an architectural pattern that seperates the web application into three main components; the model, the view and the controller. Main reason for us to use this architectural pattern is with the help of our past experiences on Multitier or often called as n-tier architecture we can learn and use this model relatively easily.



Just like an n-tier architecture divides the workload onto layers like Business, Presentation etc. MVC pattern divides the load to three main components:

View; this component is the one that has all the UI parts of the application. In this component we can use HTML5, CSS in order to make it useful for the user.

Model; this component is the place where all the data-related operations are done. This component is significant because it has all the data and secures it from other components, providing security.

Controller; acts as a bridge between the View and Model components and manipulates data with the Model and afterwards reflects it to the View component. [1,2]

**2.1.1 Main reasons of choosing the MVC Architecture over others:**

Past experience in the ASP.NET and n-tier architecture, making the MVC for the first candidate to learn.

With MVC, we can produce reusable codes and so investing in the future projects.

MVC lets us see the every operation without the confusion, we are in control of everything and can make it work just like we wanted it to work.

* 1. **.NET Framework & ASP.NET**

..NET Framework (pronounced as "dot net") is a platform made up of tools, programming languages and libraries for building many different types of applications.

ASP.NET extends the .NET platform with tools and libraries that are used specifically for building web applications. [3]

**2.2.1 Main reasons of choosing the ASP.NET over others:**

It is open source and free which is a huge benefit for us. Also with this comes the vast source of information and support from other users.

.NET Framework supports most of the languages so we can write the web application in the language that we are familiar with, in this case C#.

ASP.NET is a lot faster than most of the other popular web frameworks like Node.js.

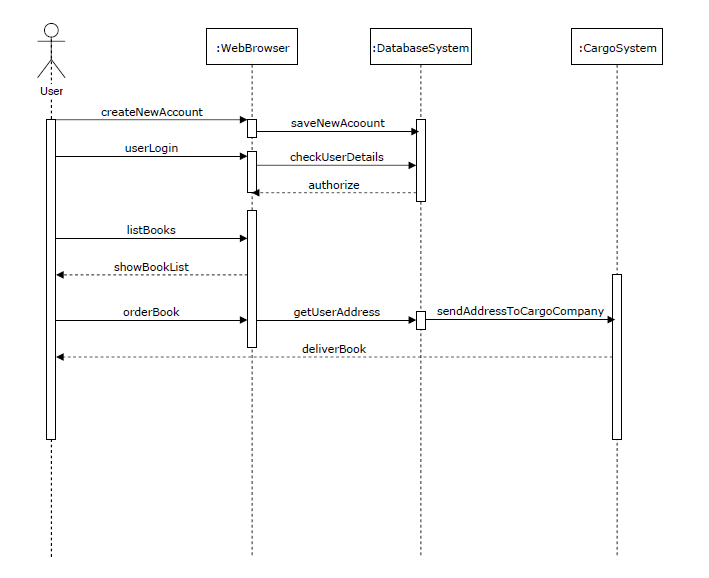
ASP.NET evaluates the complicated code on the server and sents the result HTML to the user, with this we can develop more advanced and secure web applications.

* 1. **Microsoft SQL Server**

MSSQL (Microsoft SQL Server) is a database system when it stores data used in any website or software. For example, articles, comments, user information and many more data in a blog can be stored with the help of MSSQL. MSSQL is the most used database type in Windows based servers and programming languages. We decided to use MSSQL in the database part because we will use Microsoft based .NET Framework and also our projects in DBMS course were developed with MSSQL so we’re familiar with it.

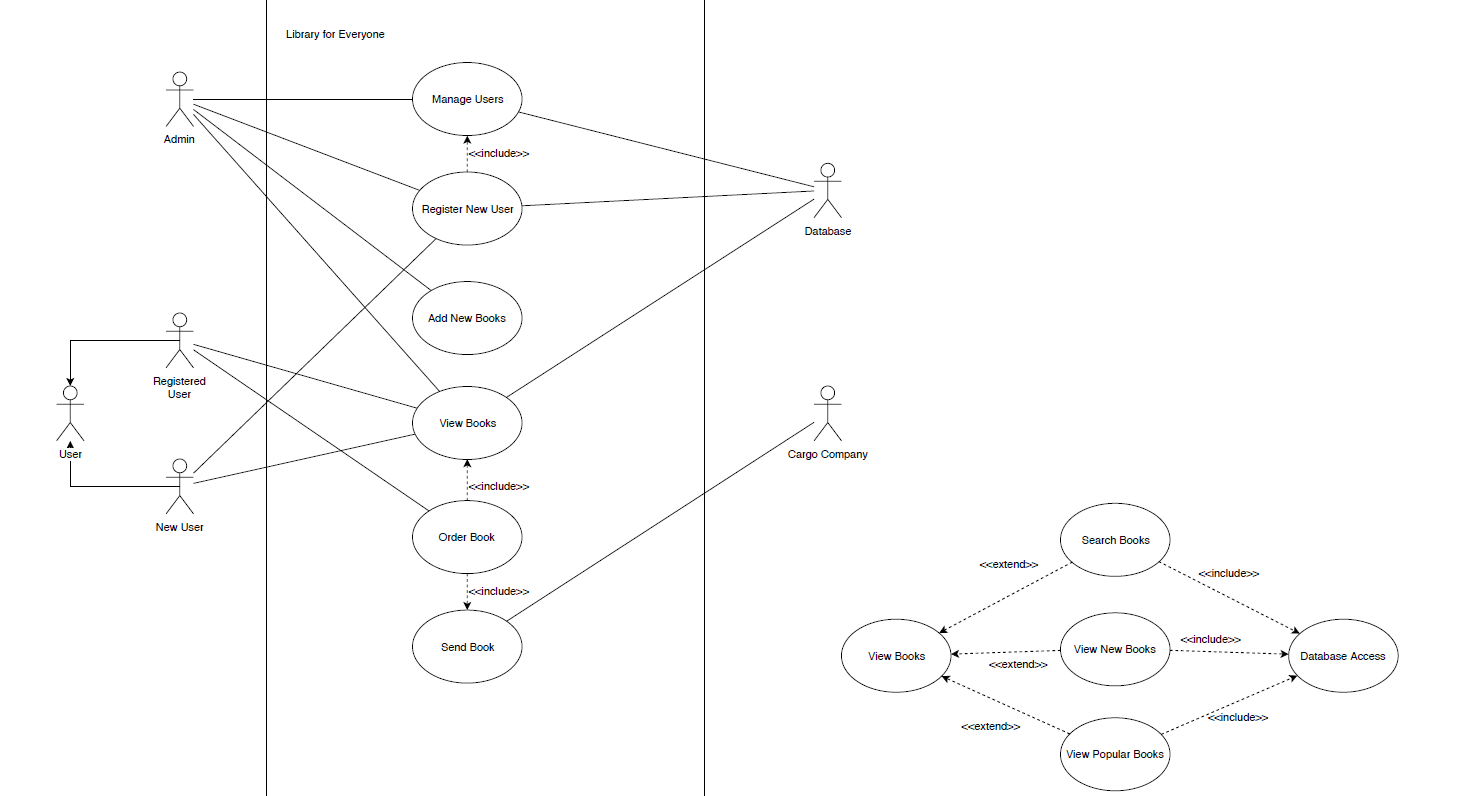
MSSQL is managed by its own editor, Server Management Studio. Operations such as creating the database, designing, entering restrictions can be done here. In addition, MSSQL uses T-SQL (Transact-SQL), which is slightly different from normal SQL syntax. It is not a programming language by itself, but we can say that it is the differentiated version of SQL. Developed by Microsoft and Sybase to interact with logical databases.

1. **Sequence Diagram**



Sequence diagram is a diagram that shows the interaction of objects in the system with each other and in what order. [1] In our project, the most interaction takes place between the user and the web browser. The person entering the site can perform registration, login, book listing and ordering through the site interface. All these processes we talk about interact with the system database in the background.

When a new user is registered, the user information entered on the site is recorded in the database. While logging in, the user is queried in our database and returned to the browser. When asked to list the books, a list is brought from the database and displayed to the user. In the book order section, it is necessary to interact with the cargo system. The user's address and the books requested are transferred to the cargo from the database and the delivery process is carried out by the cargo system.

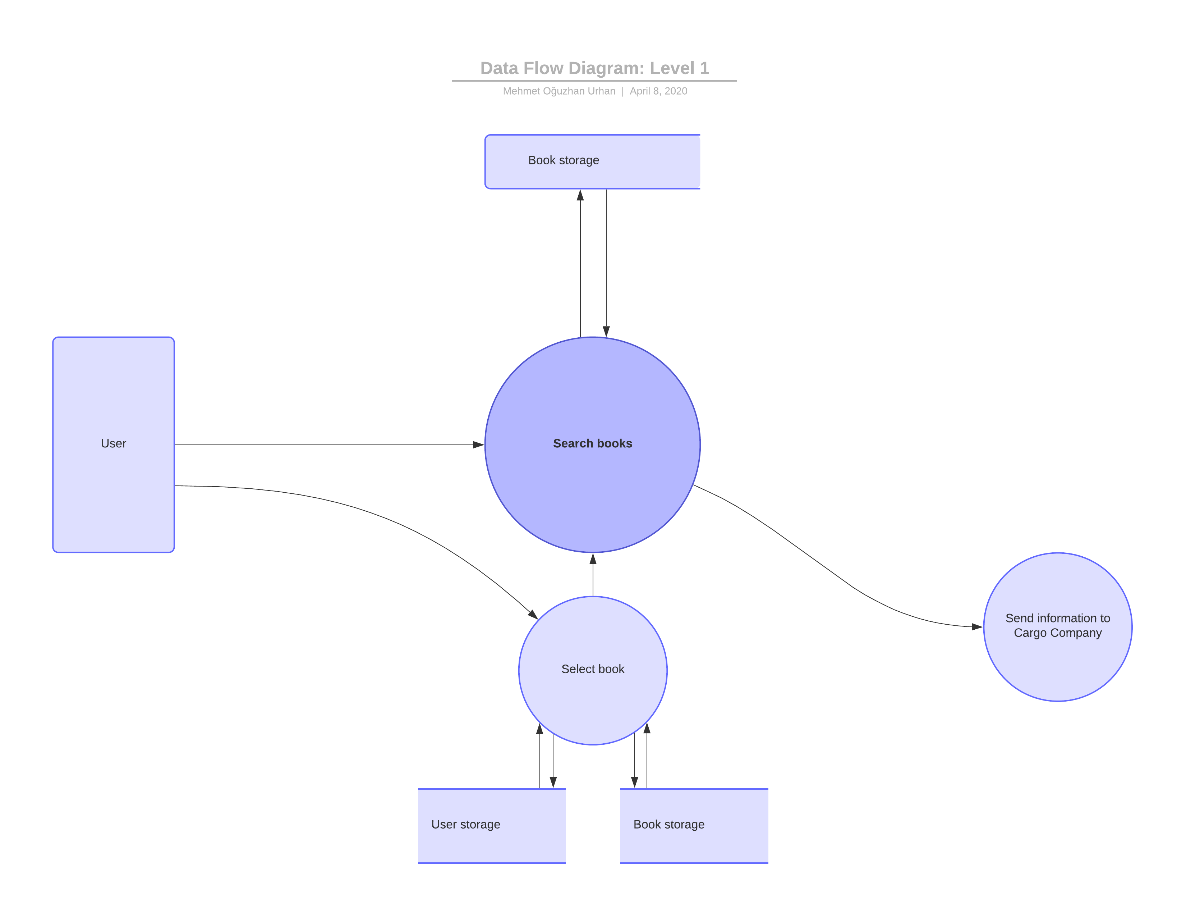
* 1. **Use Case Diagram**

Use case shows interactions of roles with the system, to accomplish specific tasks. It’s similar with sequence diagram, but we can see extra details and can see the different user types with use case. There is an admin role for the system, it can control users, add new books and register new users. All these operations are related to database.

There are two types of users, new and registered. Both can view the book list. Book viewing can be extended as we see at the right hand side of the diagram. Books can be search spesificly and also new or popular books can be seen as a seperate list. Registered user can also order books. Book delivery is made by cargo company.

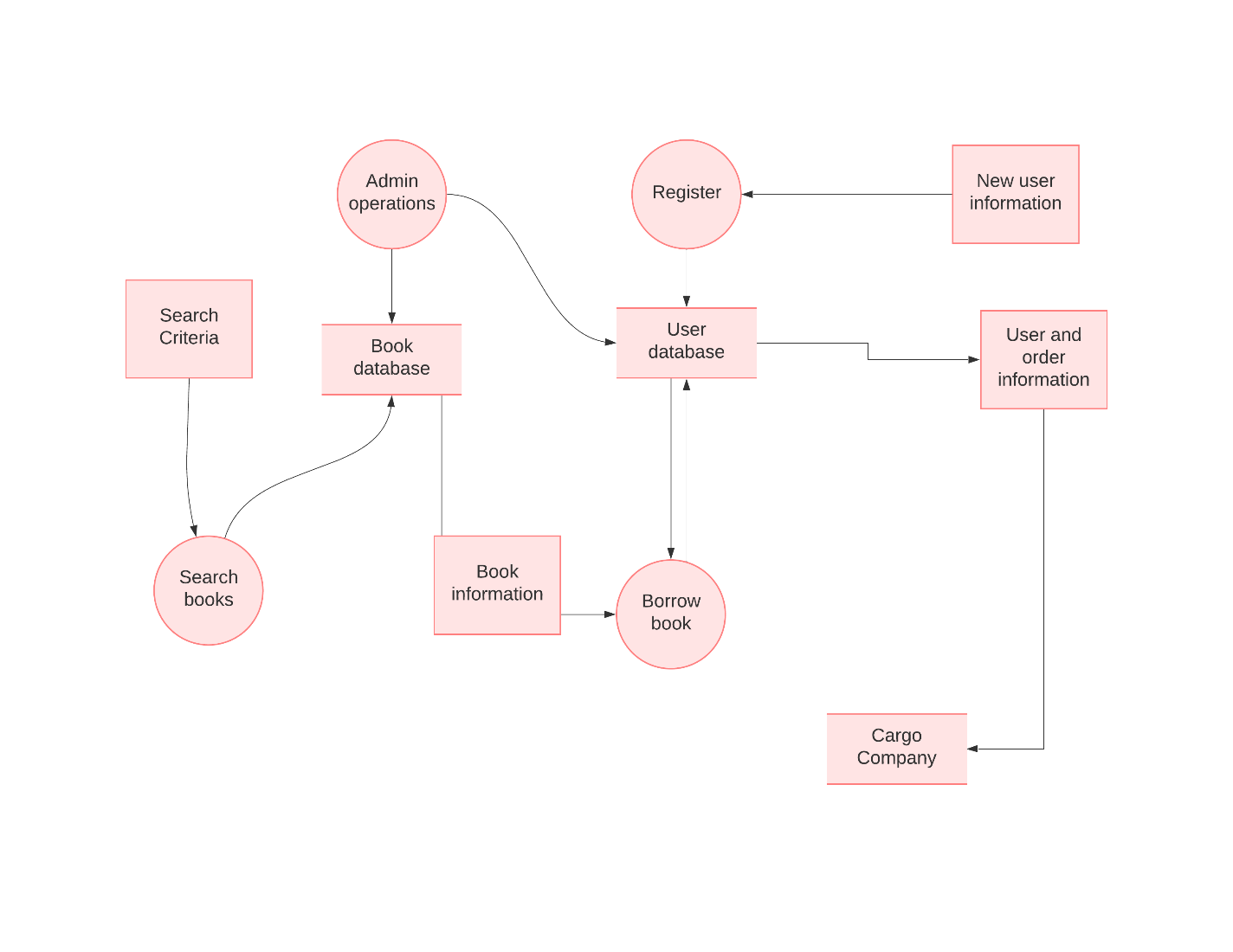
* 1. **Data Flow Diagram**

# 



DFD Level 0

DFD Level 1



DFD Level 2

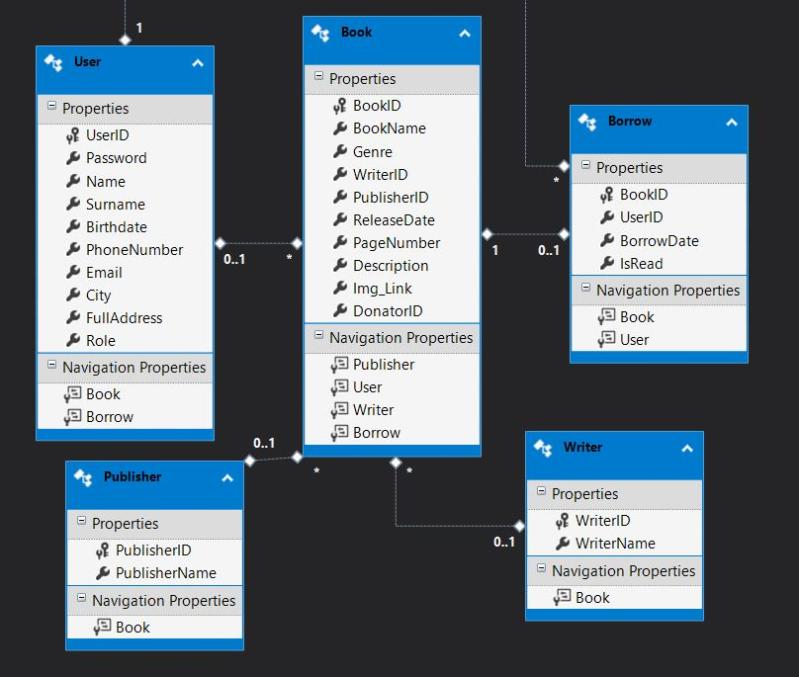
Data flow diagram shows how the information flows in between processes of the system. It uses leveled architecture which helps to show a general look at the system itself at its first level, more detailed look for the search operation and the flow of information between data storage units and lastly, we can see the operations in pseudocode.[3]

We can think level 0 as the "big picture". It's a book borrowing app that interacts with users and cargo company and overseed by admin. As proceeded to level 1, databases can be seen. Searcing takes place in level 1 because it's our project's most crucial operation. User can select books by searcing and so he/she interacts with Book Database at the background.

Level 2 is the most detailed data flow level. We have a complete view of how data flows in our system. User's search criteria input goes to search function that is working by Book Database. We get the information of the book that will be borrowed. It'll be added to user's database. So user's address and book goes to cargo company as an input.(order information) When cargo company delivers the book, our cycle completes.

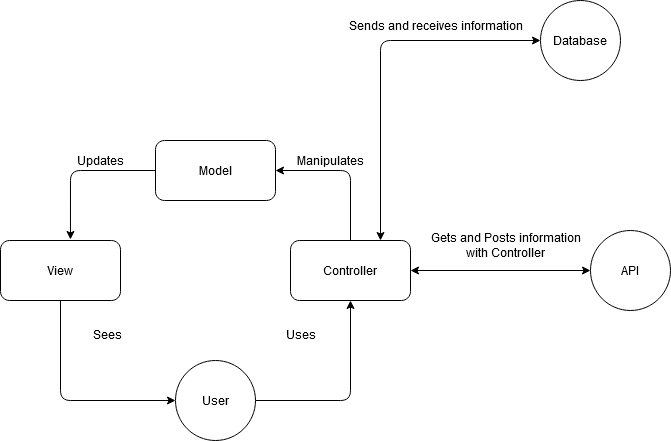
* 1. **Development Process**
  2. **First changes**

After the design phase of the project, we started implementing according to our planning. At first we started with the MSSQL Database, which ended with the updated and improved new database model in order to satisfy the api and user interface needs of the project.



Updated Database Schema

Updated database now have new tables for operations like borrowing a new book and with support for publishers and writers. With the implementation of roles to users we aim to have different web experiences for different roles like an admin to edit the website nearly fully.



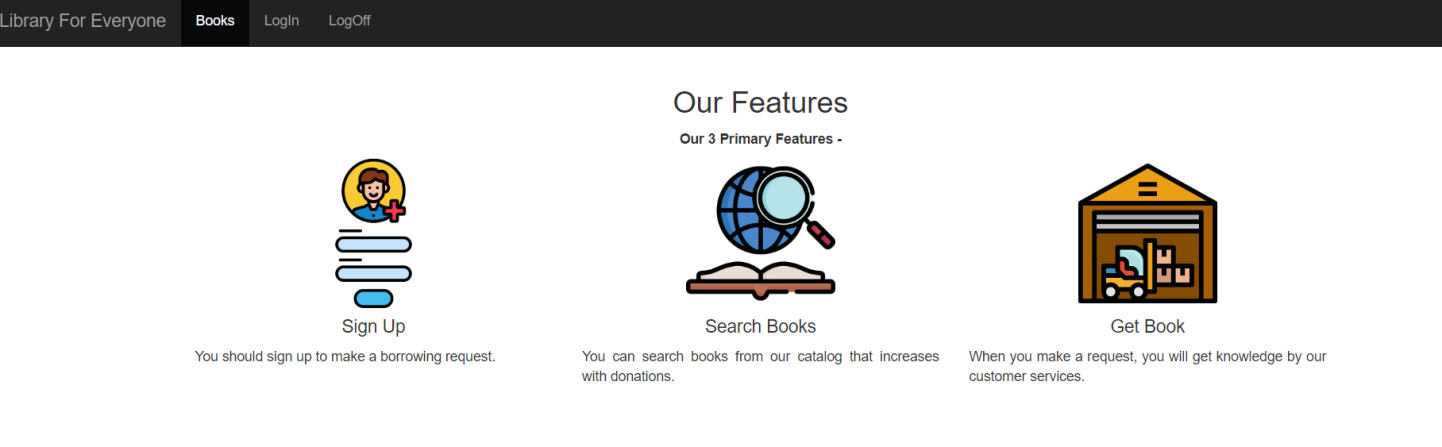
Main interaction diagram of the project

* 1. **Improvements and encountered problems**

Initially we started project according to MVC architecture and from there we implemented our database with stored procedures for database operations for adding and deleting items like users and books. From there we tried to bind the user experience according to their individual roles. Alas, we failed. Therefore we decided to start over with better implementation of database and their entities.

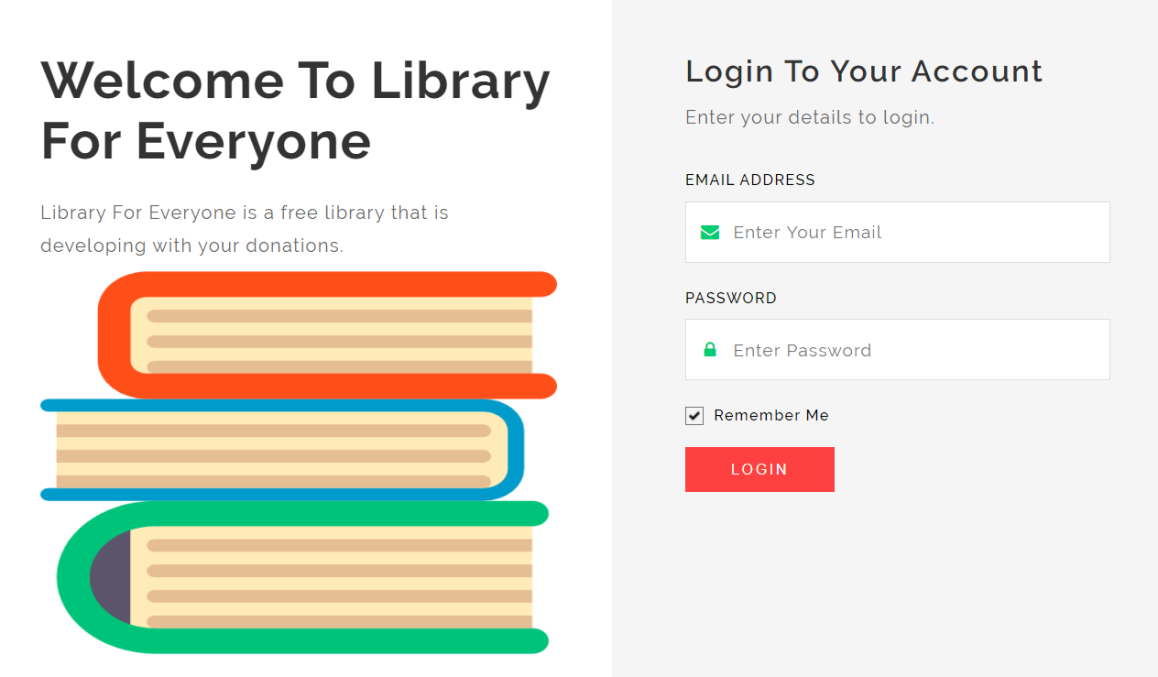
Afterwards, we implemented the roles and their authorities, taking into account the security concerns with this. For a user to borrow a book, a request is sent to the admins, from there if it gets approved, the book goes to borrowing system and makes its way to the user.

Following these improvements, we smoothed the user interface and general interactions with the application.

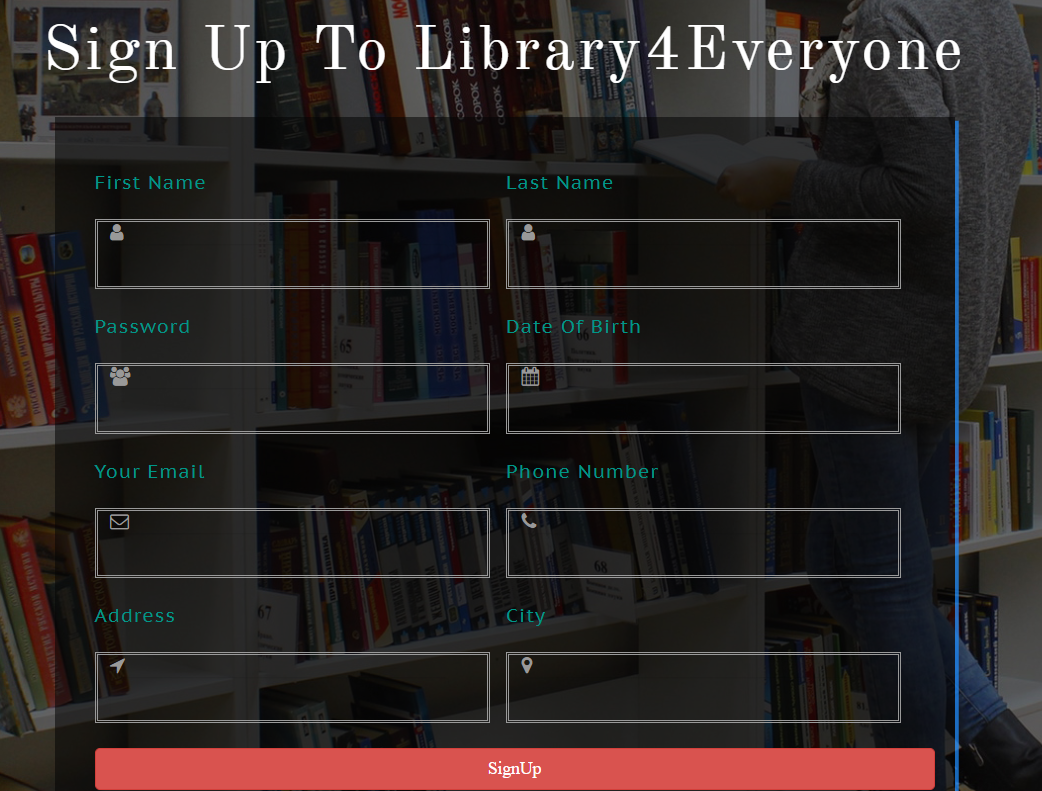


Home page of the Application

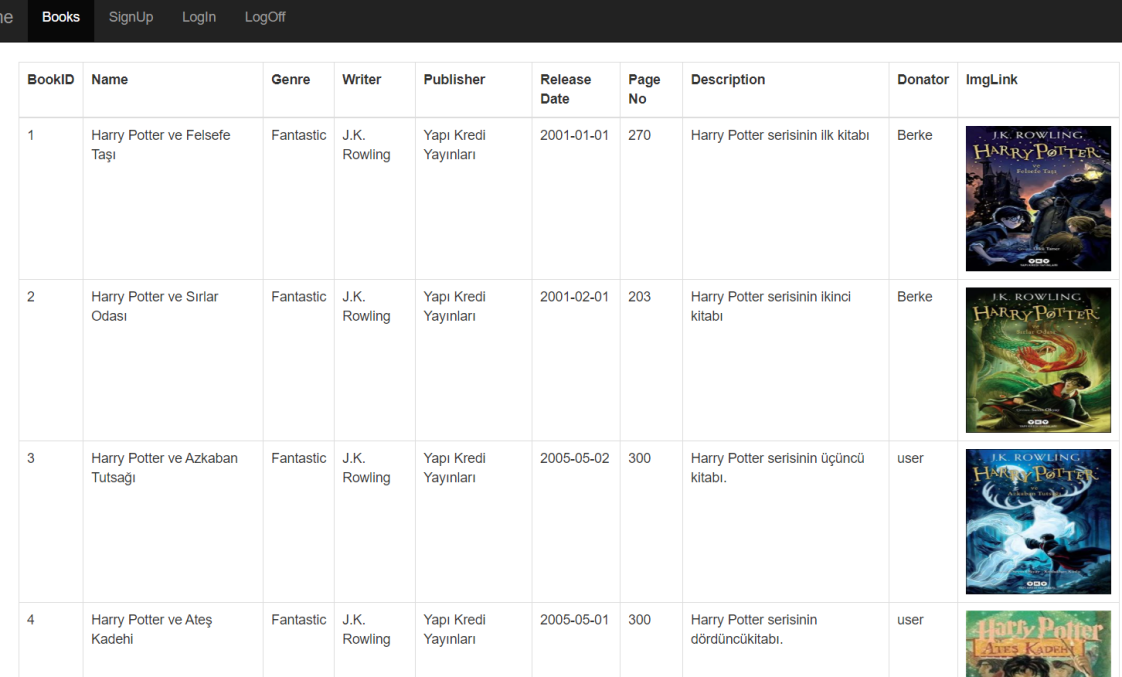
In the home page; users are welcomed by minimalistic design. Users can login or sign up to the application or they can browse the books of the library. After logging in, they can make a request for borrowing.



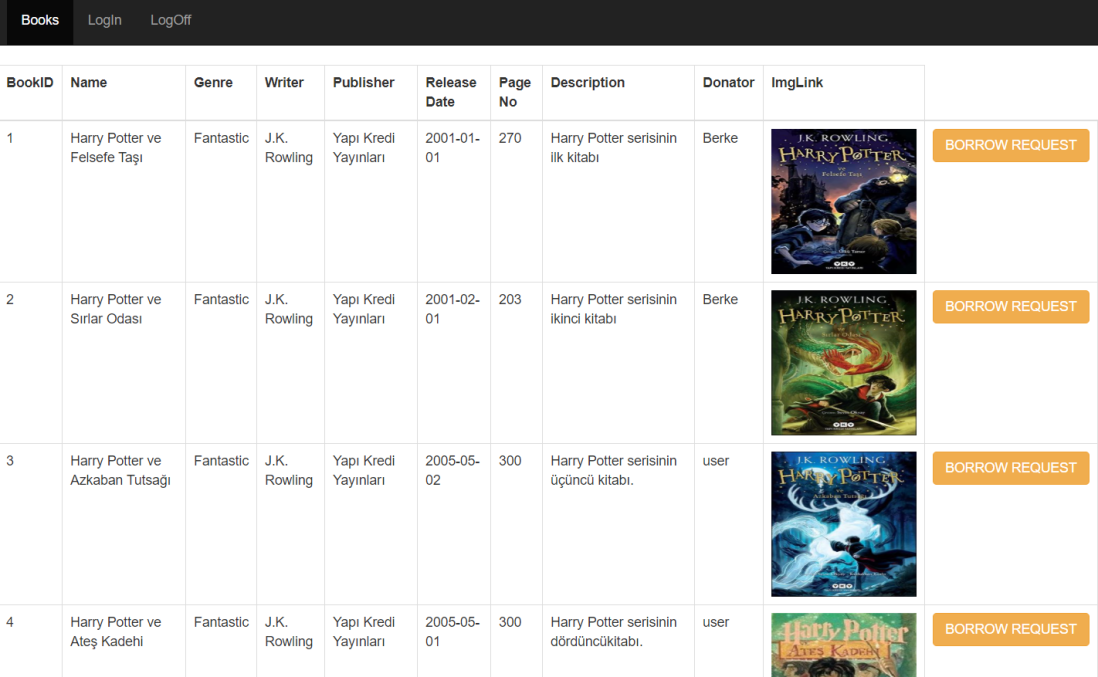
Login page of the application



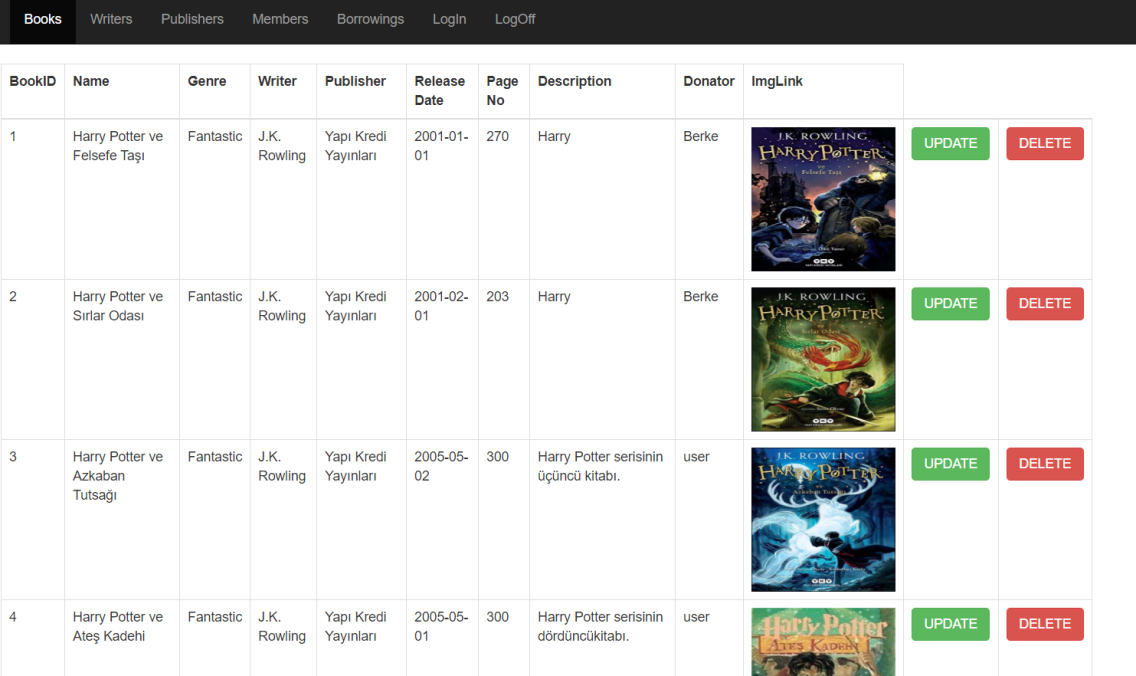
SignUp Page of Application



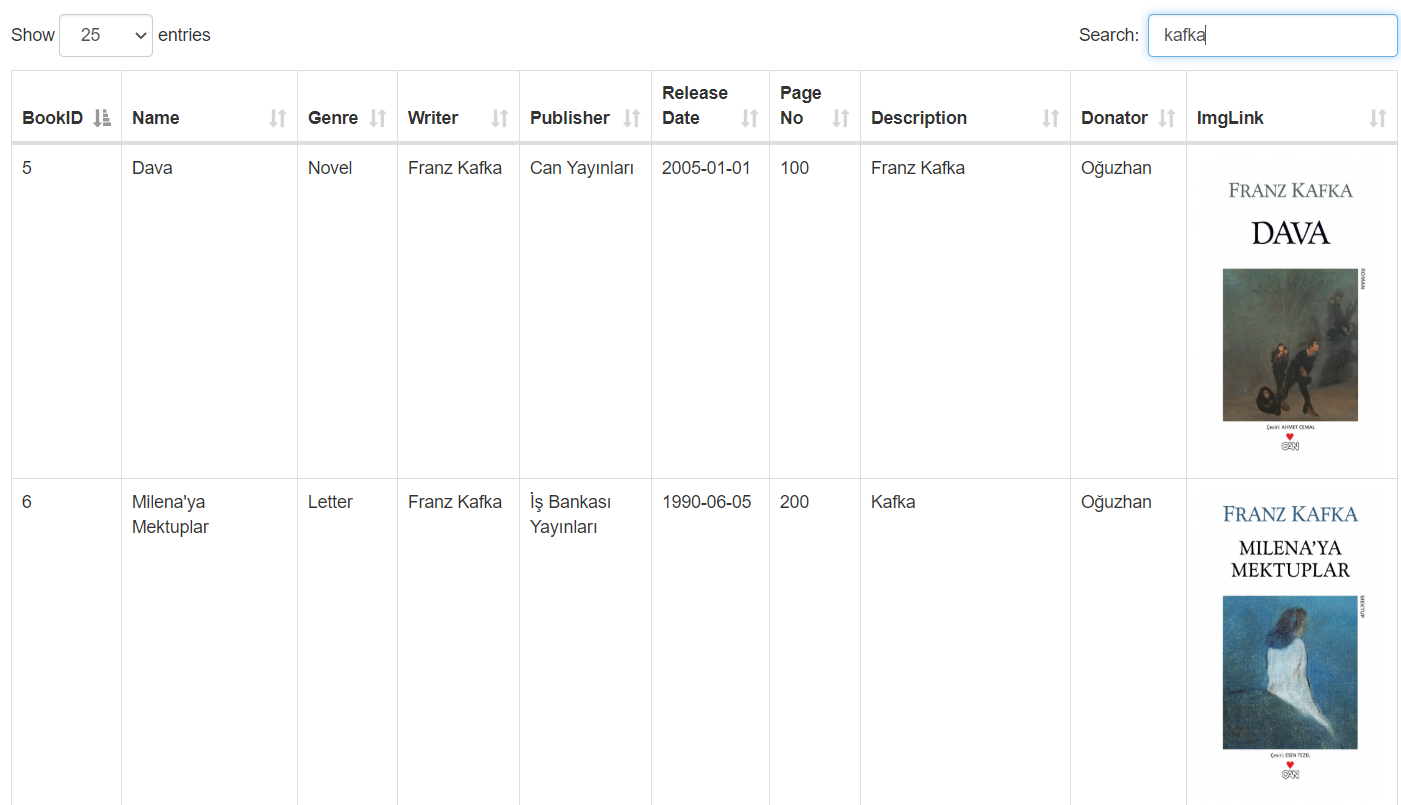
Book list page for guest user



Book list page for a registered user



Book list page for an admin



Search and paging of book list page

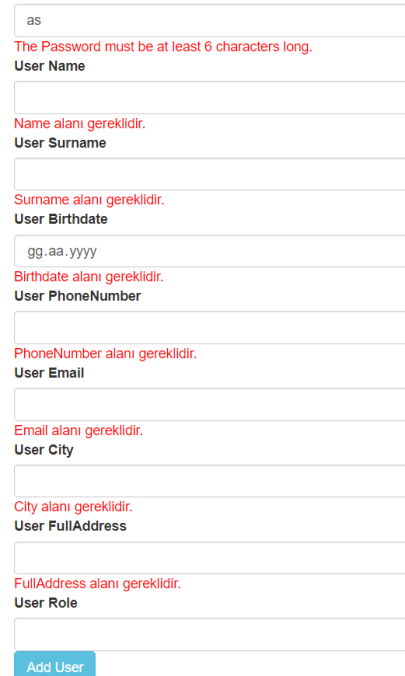
As can be seen in these screenshots, we aim to have a simple yet efficient user interface. Even without signing in, guest users can see the books that are in the library in order to obtain new members. In the book list page, if a user has admin authorization they can add, update or delete books. If it does not has that it can only make a request for adding or borrowing book.

* 1. **Security and privacy**

Concerning security and privacy, we decided to go with the implementation of our own server with our computers because of both financial and privacy concerns that came with KVKK in Turkey which blocks the unauthorized information storage in another countries.

When registering to the application, users are expected to have at least 6 character password, which can be improved more with the addition of upper and lowercase letters with support for special characters but we did not implemented this for now in order to not lose new users at register phase with that much work.

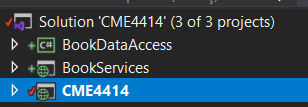
In the application, users are divided by their roles by using Web Security libraries and RoleProvider classes came with it.



Register phase obligations

* 1. **API**

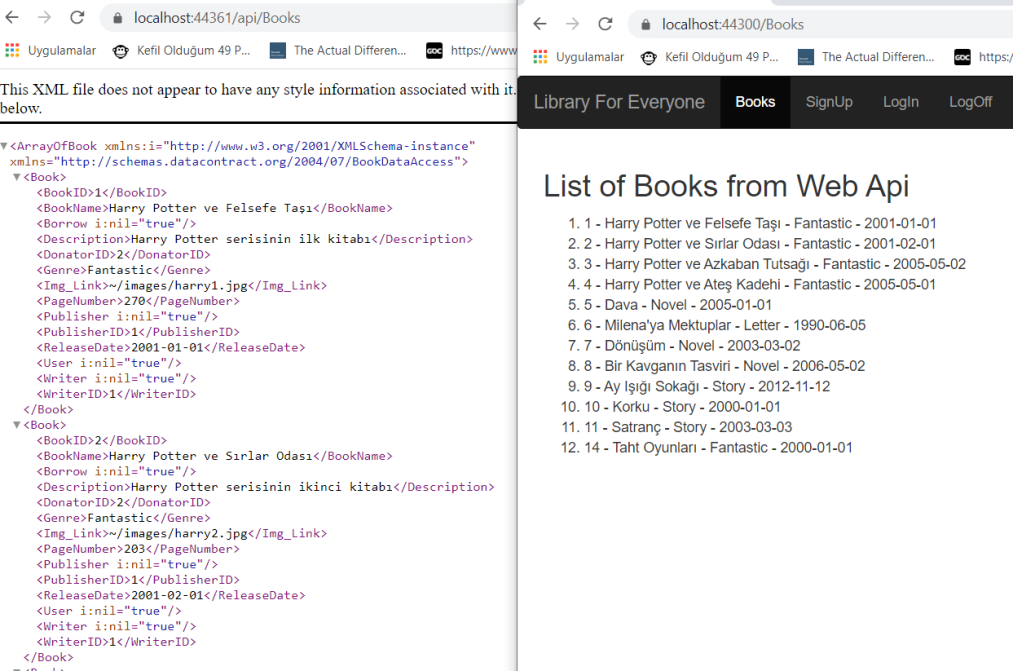
We implemented the use of web service as a tool to get and post the books from our database. Web services directly created from our database. It is controlled by the Web Api MVC project named BookServices.



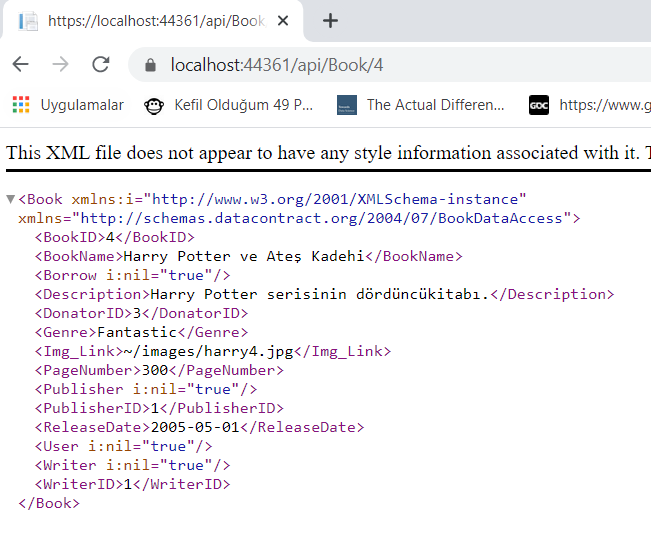
Our solution’s content

BookDataAccess is a .NET Framework Class Library that contacts with our MSSQL database via ADO.NET Data Access. We did it to make a fresh start and trying a new approach and we kept it because it was successful. Now for the API, data is coming from BookDataAccess; Get methods are in BookServices projects that communicates with CME4414. CME4414 uses ApiMessages and BookDataAccess to get BookEntity. Then http connections made in BooksController. (not to be confused with BookController – that controlls CRUD operations for Book table.)

BooksController gets data from <https://localhost:44361/api/Books> and send it to its View as a List. In this View (uses ApiMessages.BookEntity as model) we can select which data to show from XML.



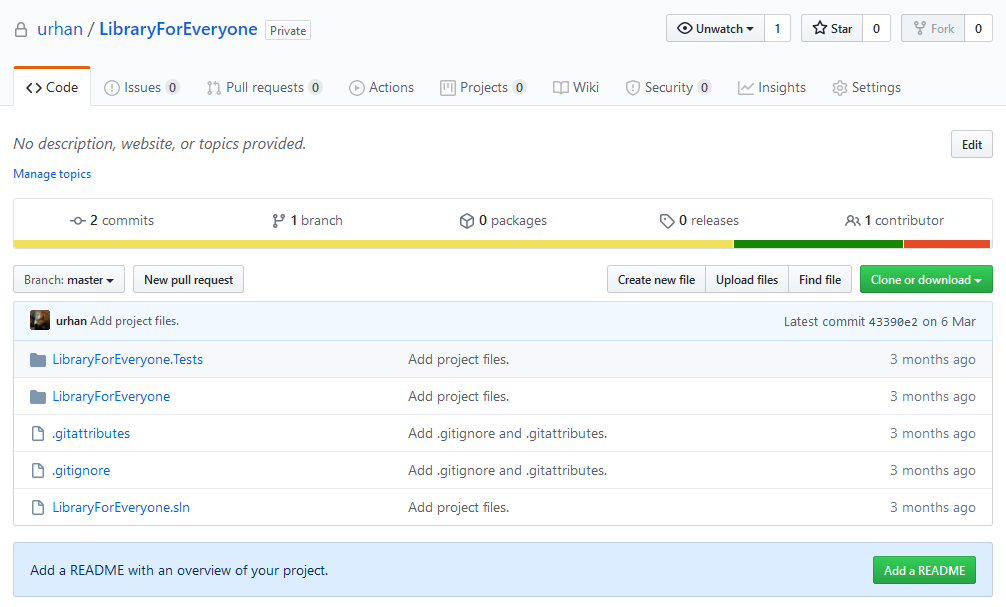
API Output to the web application



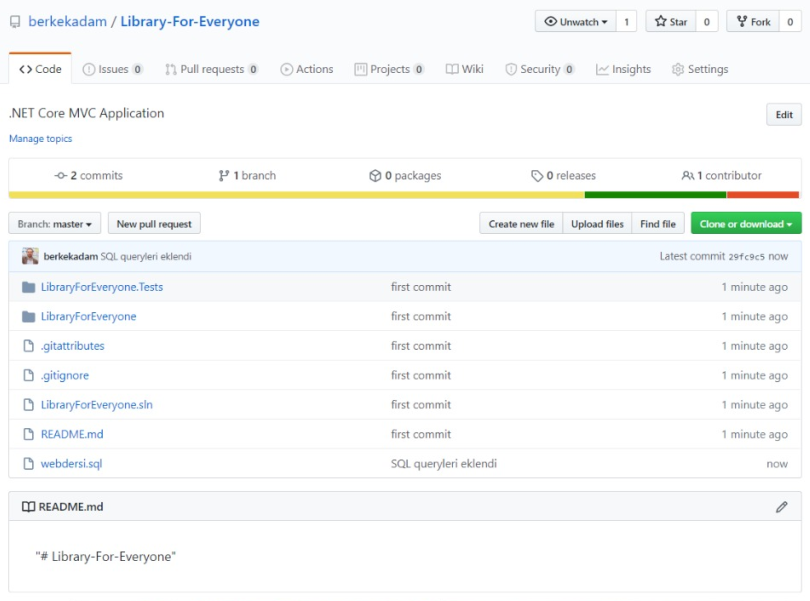
API Data output in XML format (for specific ID)

* 1. **Github**

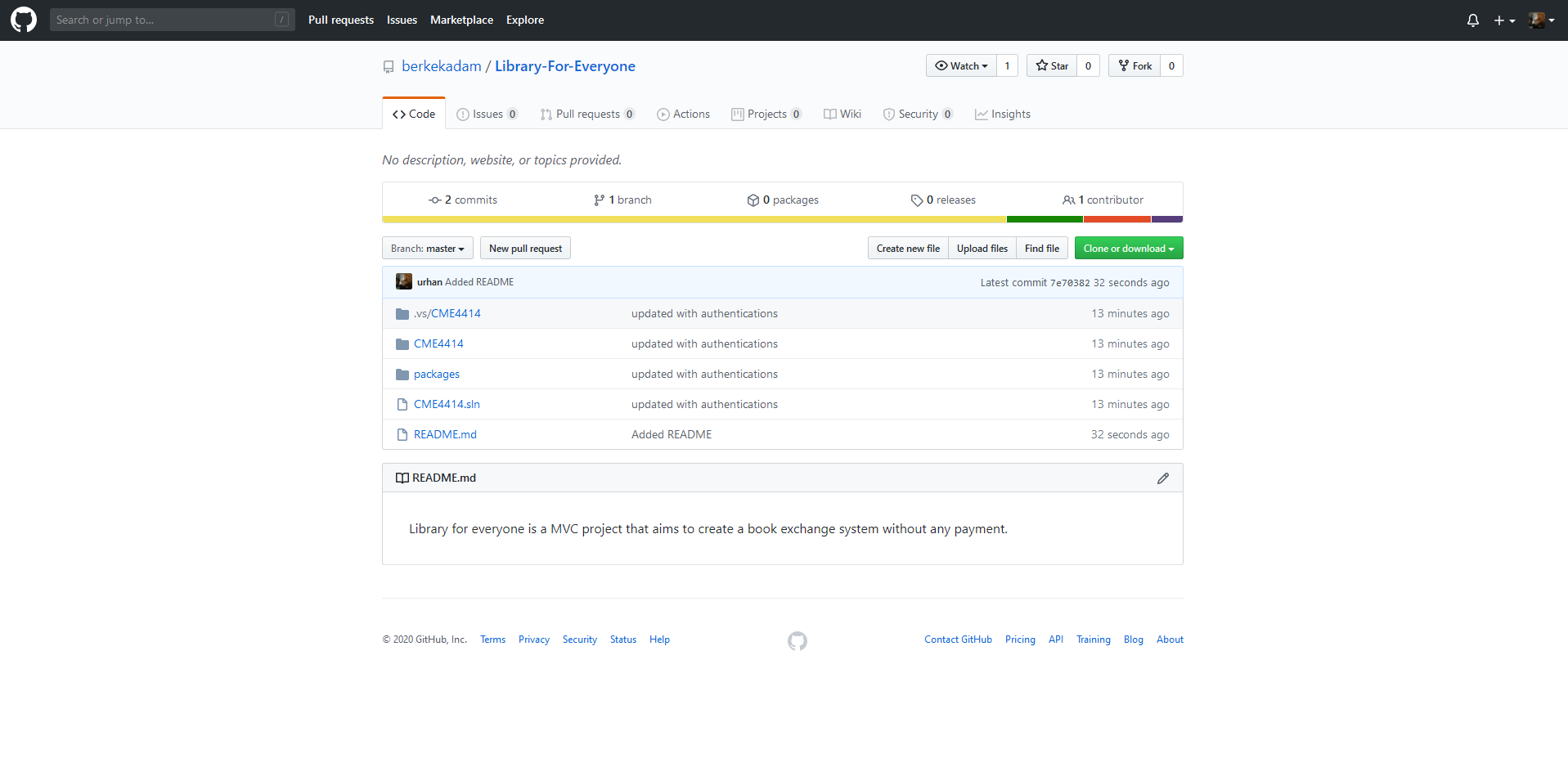
Development of this project is carried out with the use of version control tool Git on Github. However with the project restarted over many times in order to understand and implement the true and efficient form of the architecture with support for our own database and entities, we did not finished the project on the same repository.



Initial commit of the project on Github



Later version of the project, using queries



One of the last and cleaned commit of the project