Your Books Everywhere!

Analysis and Design Document

Student:Lazăr Denisa Ștefania

**Group:30238**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

It is commonly known that reading has a positive impact on one’s intellectual

development, by enriching both knowledge in certain domains and vocabulary.

The task for this assignment is to build a book management service. It will serve as remote library, so the user can borrow/buy different books from the comfort of his own home, without needing to go to a library that maybe is far apart.

# Functional Requirements

In this section, the functional requirements of the application will be discussed. They are as follows:

* **Account creation 🡪** The app must offer a user the possibility to create an account by completing a form with his/her personal details.
* **Payment plan choosing 🡪** A user can choose between the following payment plans: *weekly*(he/she can pay at the beginning of every week for being able to search and eventually borrow/buy different books), *monthly* or *yearly*.
* **Staff management 🡪** The library will be managed by staff, whose main tasks are to validate payments and returns made by the library’s users.
* **Library filtering 🡪** The books belonging to the library can be filtered by release date, author, title and genre.
* **Borrow service 🡪** If a certain book is available, the user can borrow it. Otherwise he/she will have to wait. It is important to mention that for each book there will be a waiting queue. So, if the desired book is not available the user will join the waiting queue.
* **Return service 🡪** Once a user has finished reading the book, he/she will return it. Following this, the book will be assigned to the next user in the waiting queue.
* **Recommendation service 🡪** The app will recommend different books to a user based on previous borrowed books or on specified genre and topic.

# Non-functional Requirements

This library management app has to be:

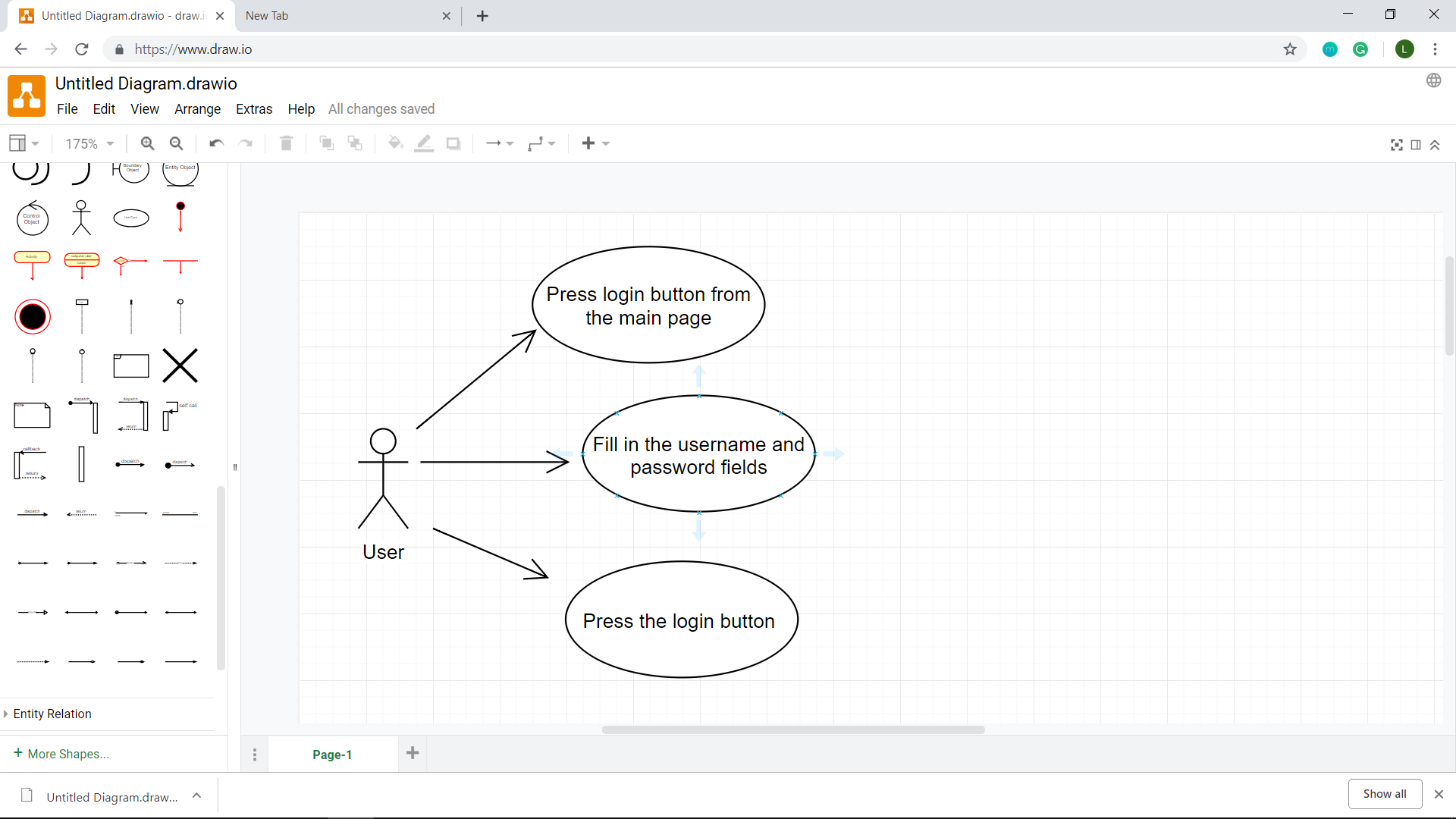
* **secure** (a potential attacker must not have access to the database or other sensitive information such as other users’ passwords)
* **portable** (it should be available to be used on different types of devices)
* **easily maintainable** (the effort that will be made for different upgrades/bug fixes must remain at a reasonable low level)

2. Use-Case Model

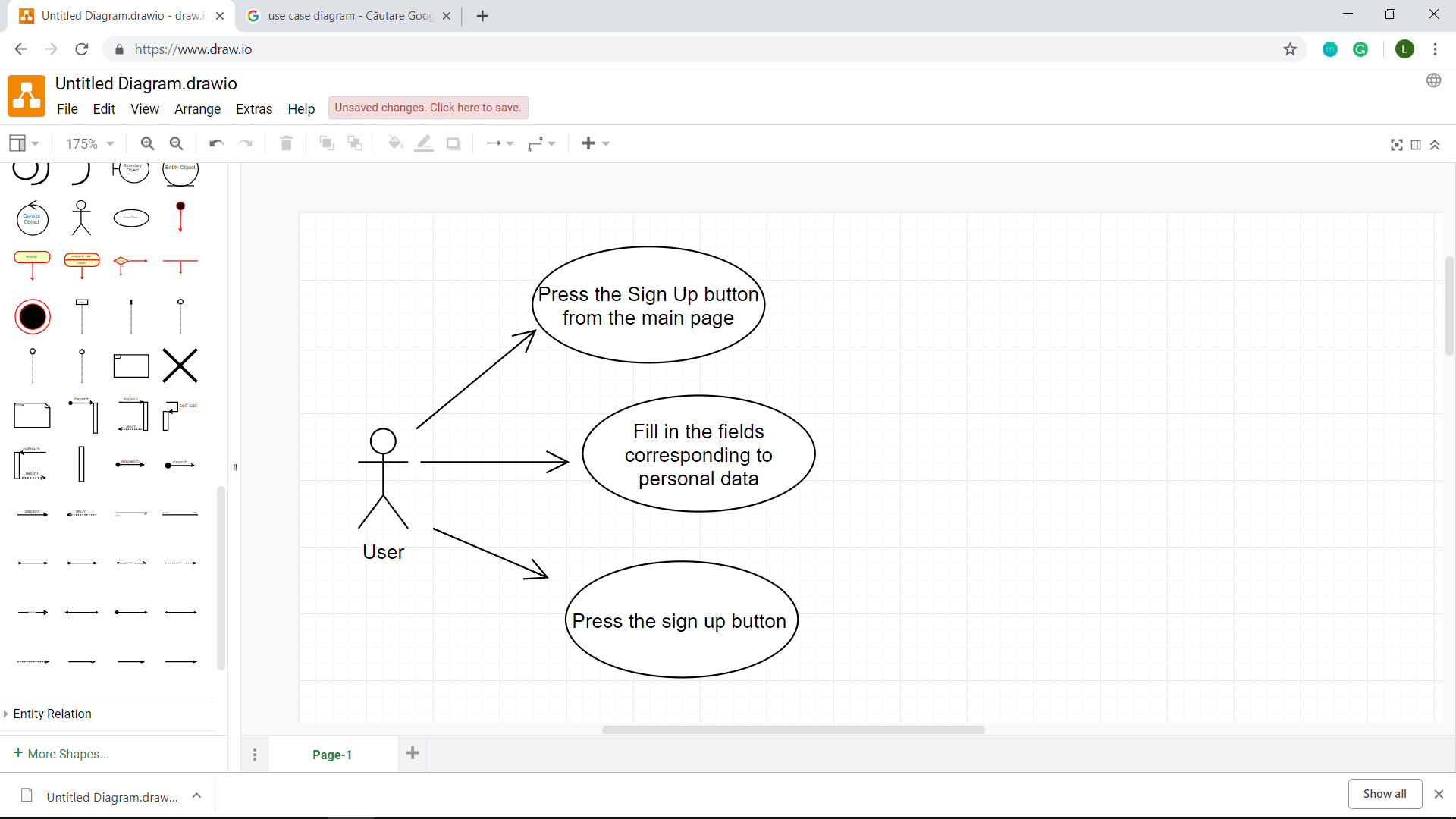
This section will show the use case diagrams corresponding to the application and describe one use case in a more detailed manner.

The use case diagrams are as follows:

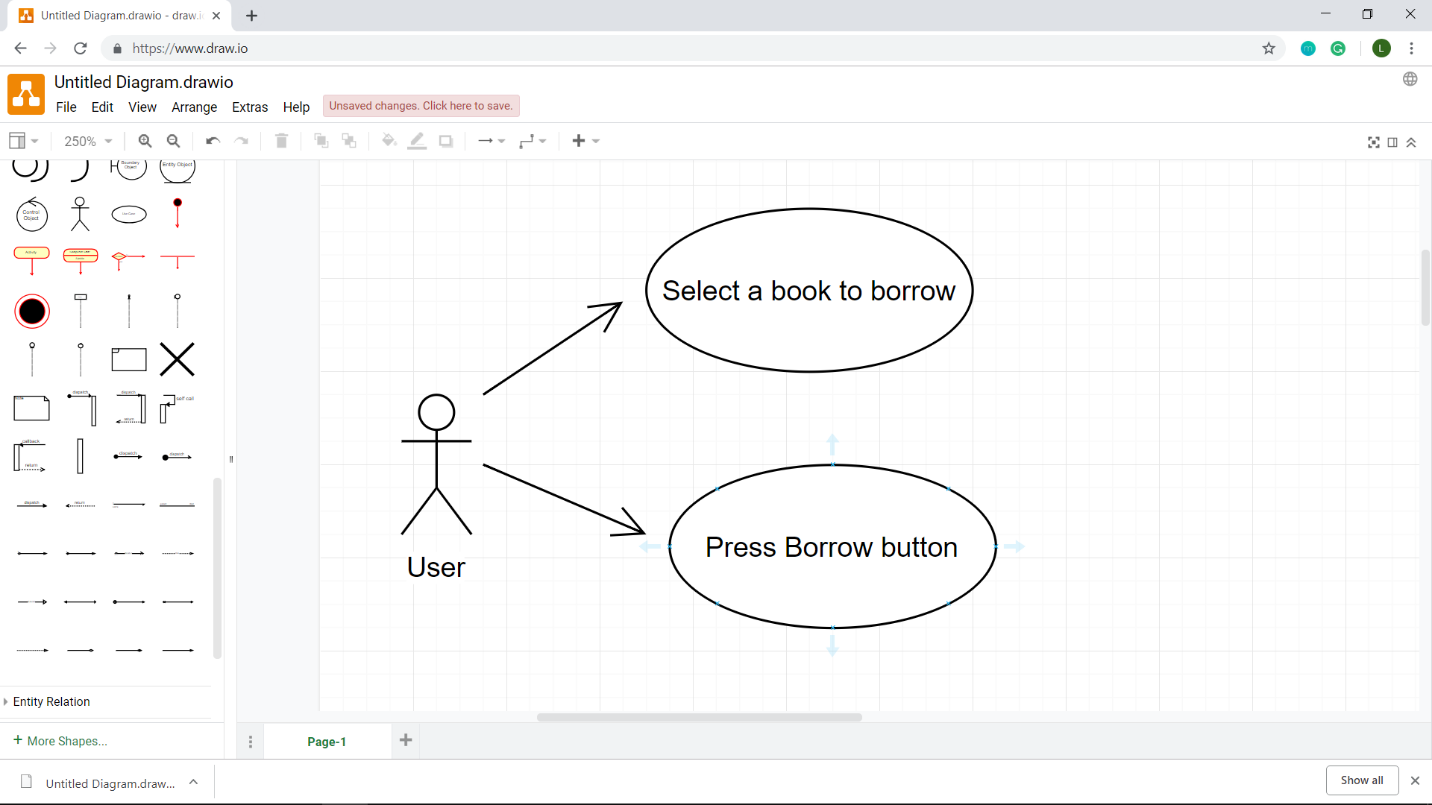
* User/Staff Login use case ()



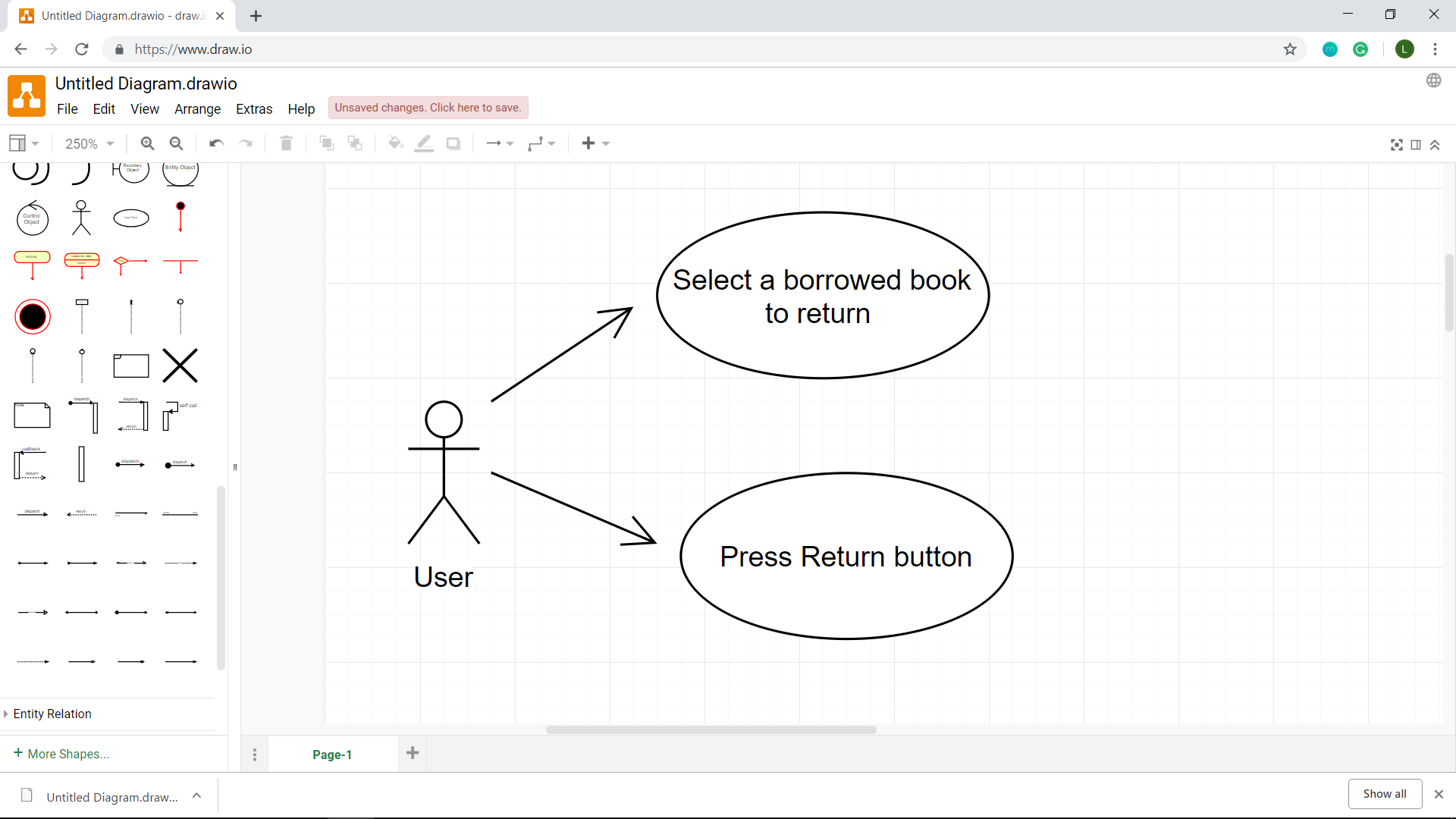
* User Sign Up use case



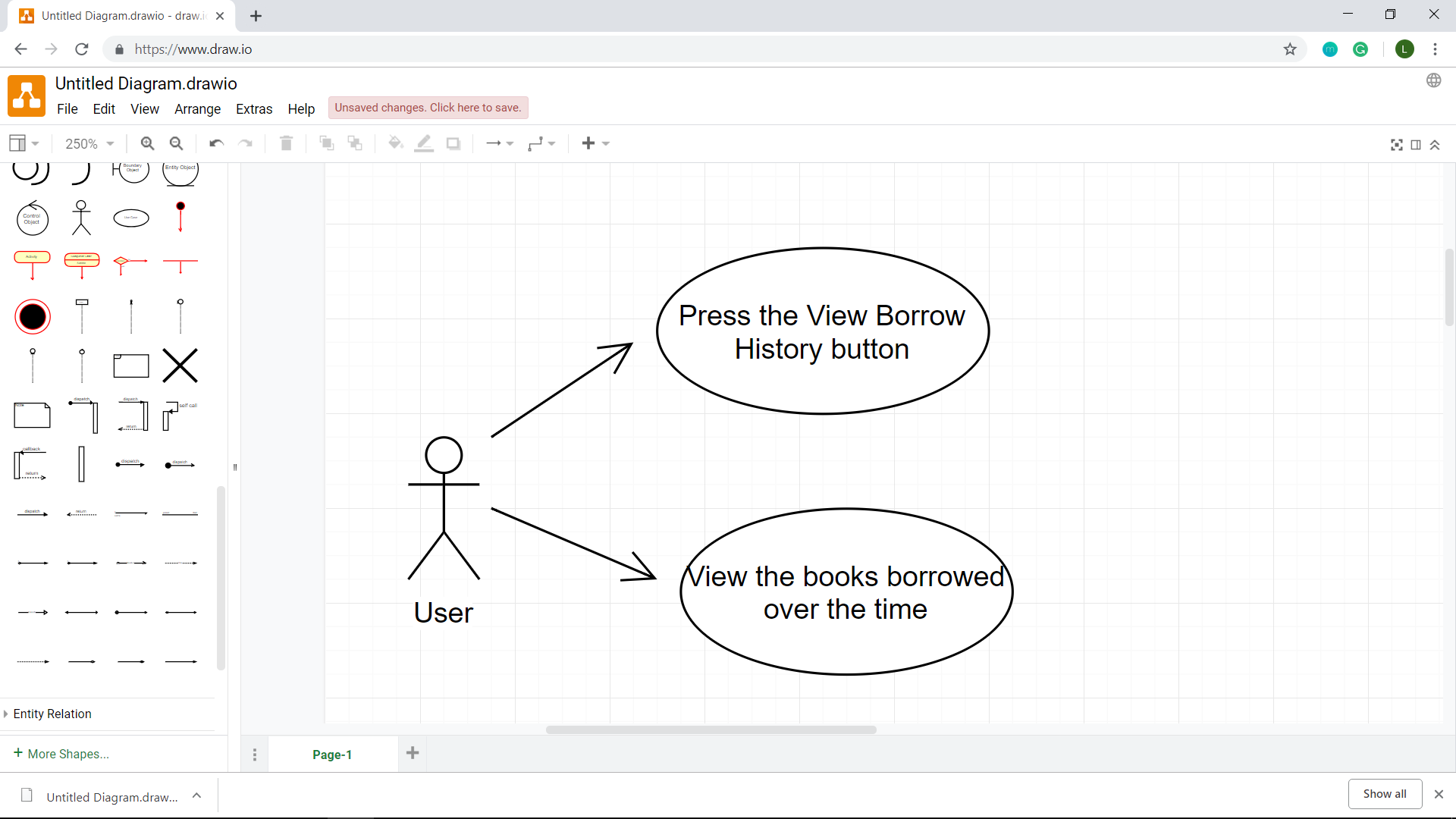
* Borrow a book use case



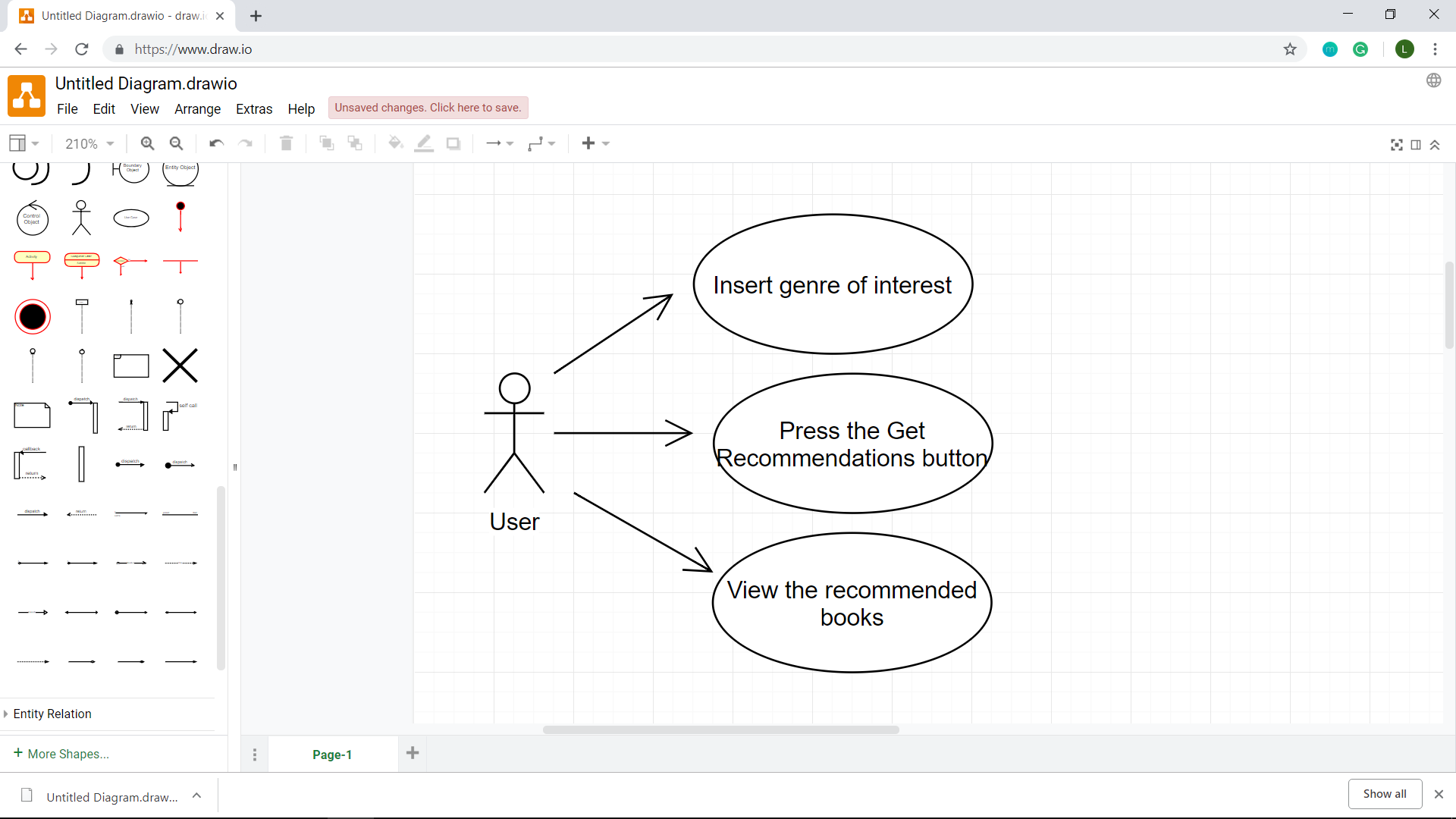
* Return a book use case



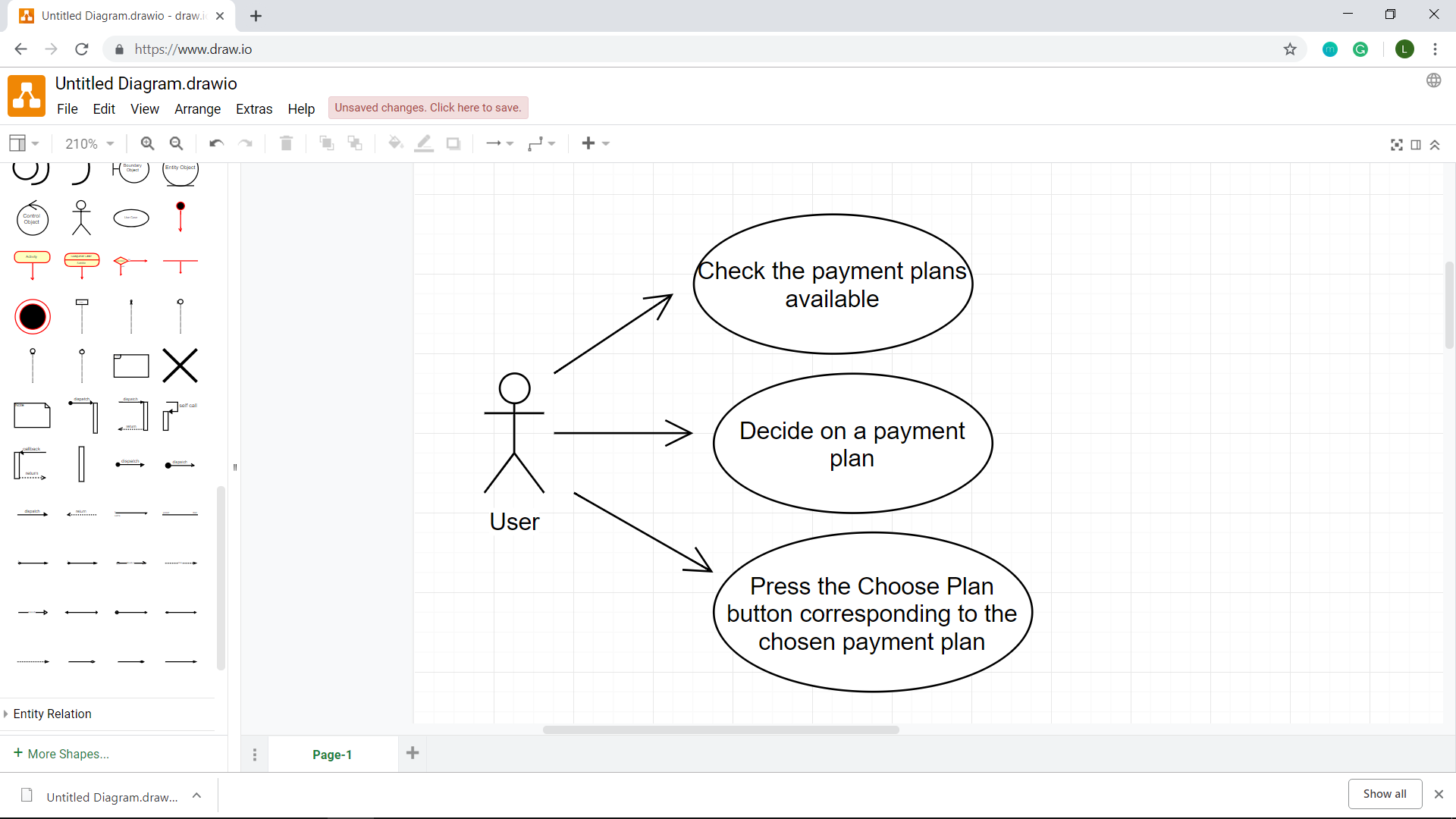
* View borrow history use case



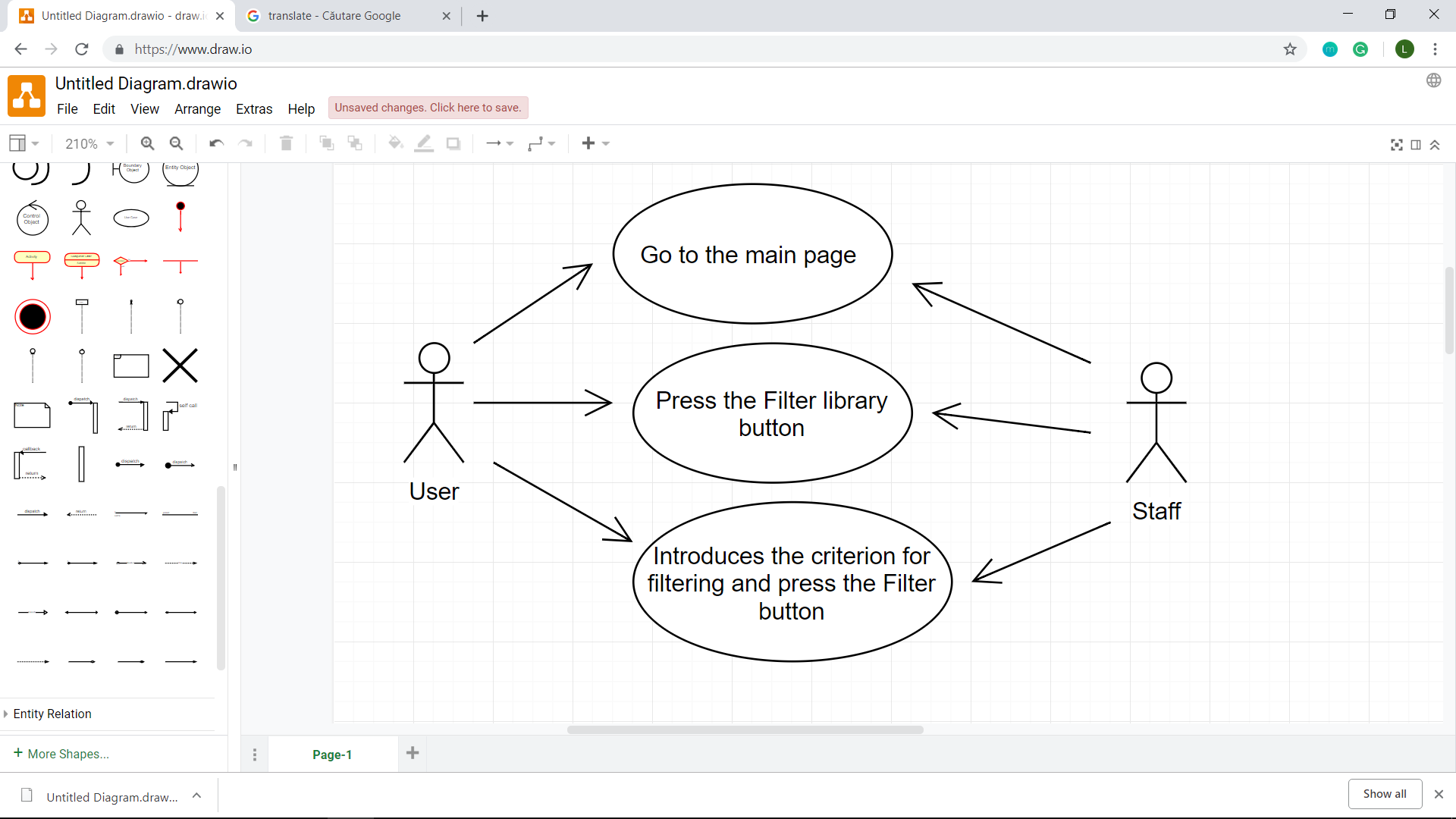
* Get book recommendations based on a choosen genre use case



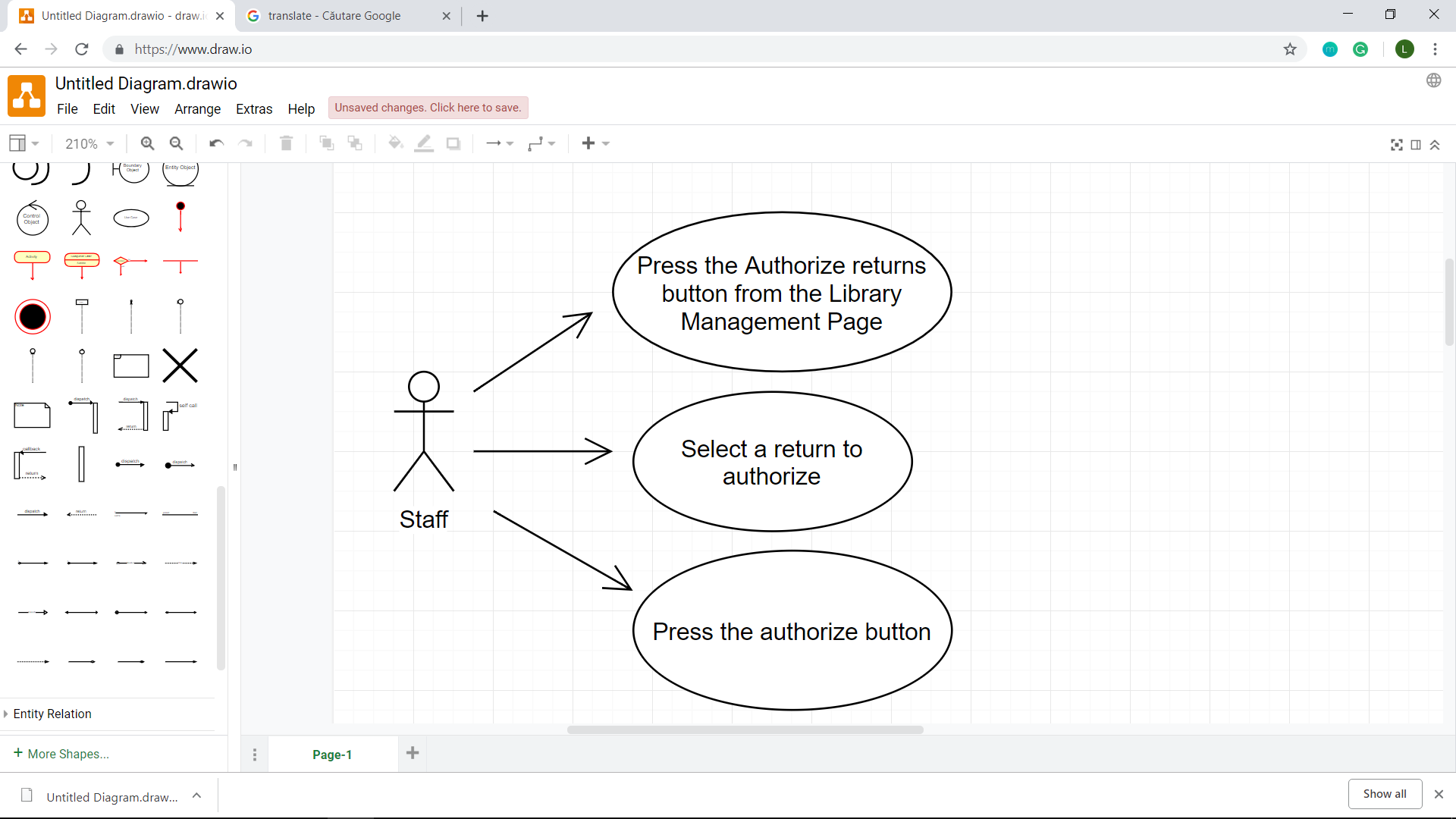
* Choose payment plan use case



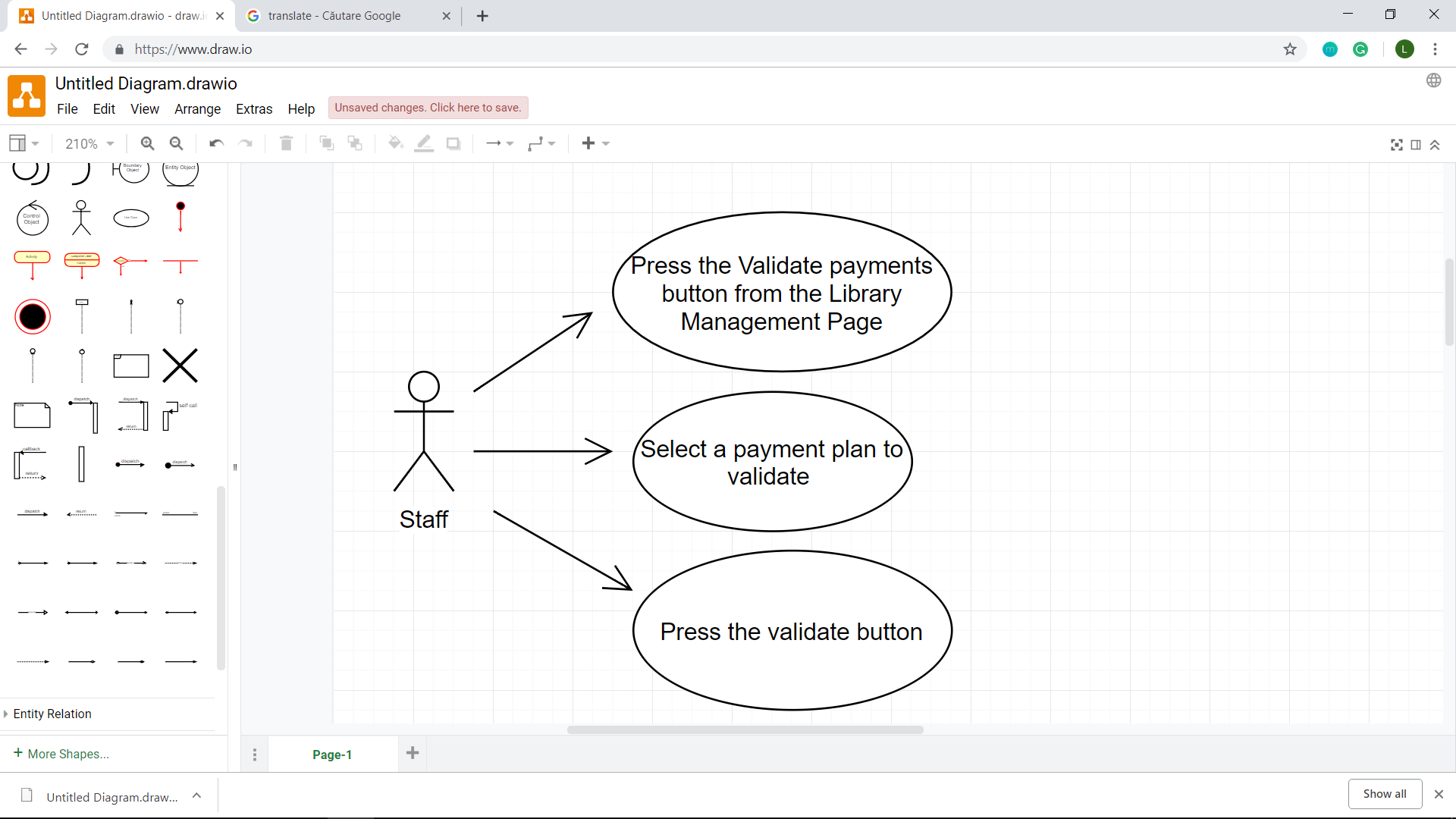
* Filter library use case



* Authorize returns use case



* Validate payments use case



A more detailed description of a use case, according to the required format is provided below:

**Use case**: User borrows a book

**Level**: user-goal level

**Primary actor**: a library client

**Main success scenario**: The library client first has to log in into his/her personal account. Then he/she has to go to the “Borrow” section. After that, the library client has to choose the book he/she is interested in and borrow it by pressing the “BORROW BOOK” button.

**Extensions**: for this use case, the success scenario is when the book is available; in this case, the book will be added to the list of borrowed books corresponding to that user. The failure scenario is when the book required by the user for borrowing is not available; in this case, he/she will join a waiting list and will be assigned the book when it will become available again.

3. System Architectural Design

**3.1 Architectural Pattern Description**

The required architecture for this assignment was a **layered architecture**. Thus, the app has three layers which will be briefly described next:

* **presentation layer** 🡪 this layer is responsible for graphical user interfaces and displaying data under a specific format; also, it sends data to the business logic layer for further processing and manipulation
* **business logic layer** 🡪 this layer is responsible for manipulating interaction between different types of objects corresponding to the application; another important role this layer has is to coordinate data flow between the presentation layer and the data access layer
* **persistence (data access) layer** 🡪 this layer separates the database engine used from the business logic, providing a few advantages such as: encapsulation of database logic in a single layer and easier migration to other storage engines
* **database layer** 🡪 provides data storage and isolates data from the other layers

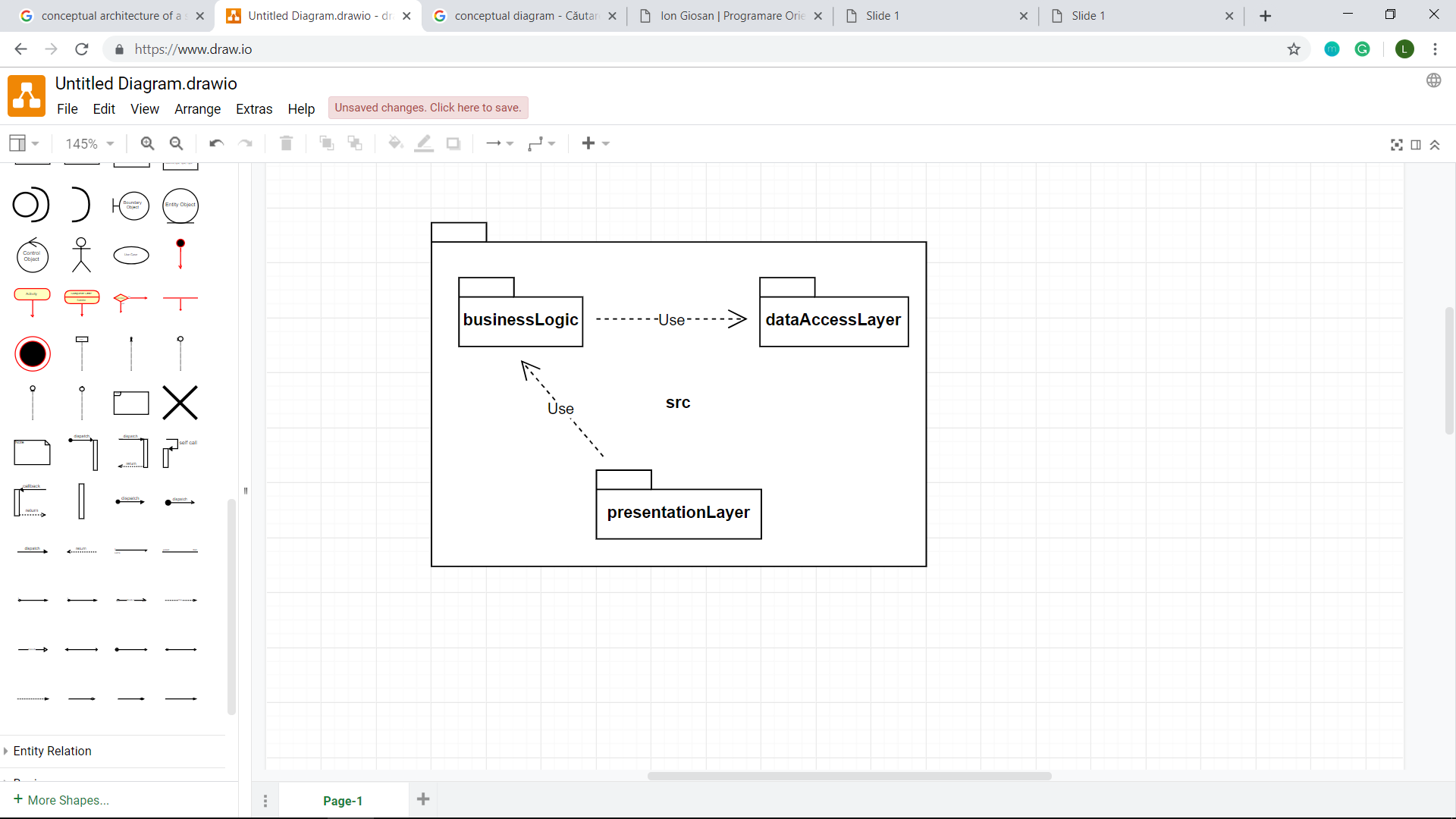
Following, there is presented a diagram which illustrates the layered architecture pattern:



Each layer is closed, meaning that data flows through all the layers, without skipping any of them. Also a layer should only know about the layer underneath it, without being aware of higher layers.

**3.2 Diagrams**

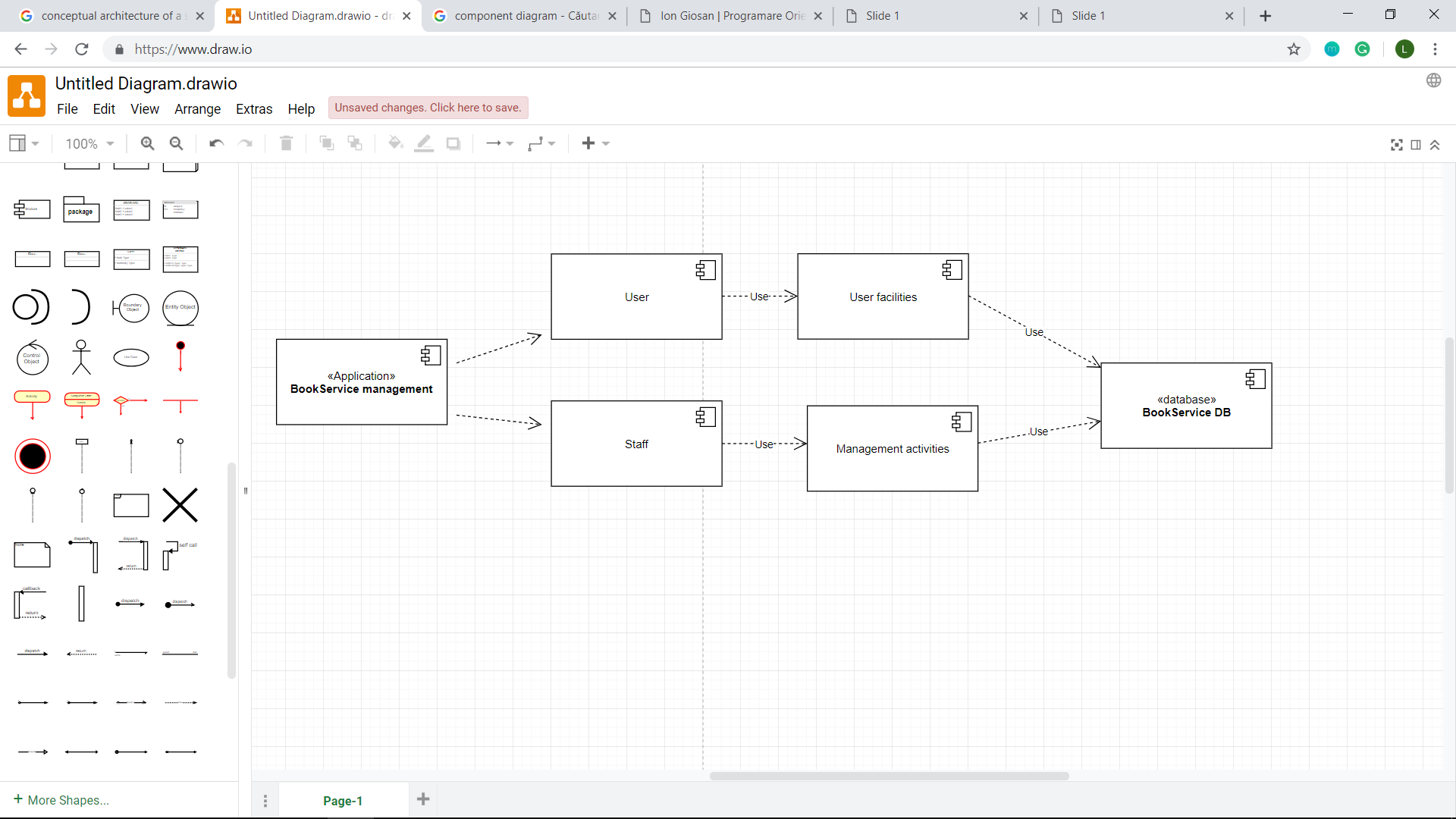
* Package diagram



* The diagram above represents the structure of the application from the packages point of view. The application was build using the Layers architectural pattern. A lower order level does not know about the layers above it and should use functions only from the layer directly under it. The order of levels (from highest to lowest order one) is :

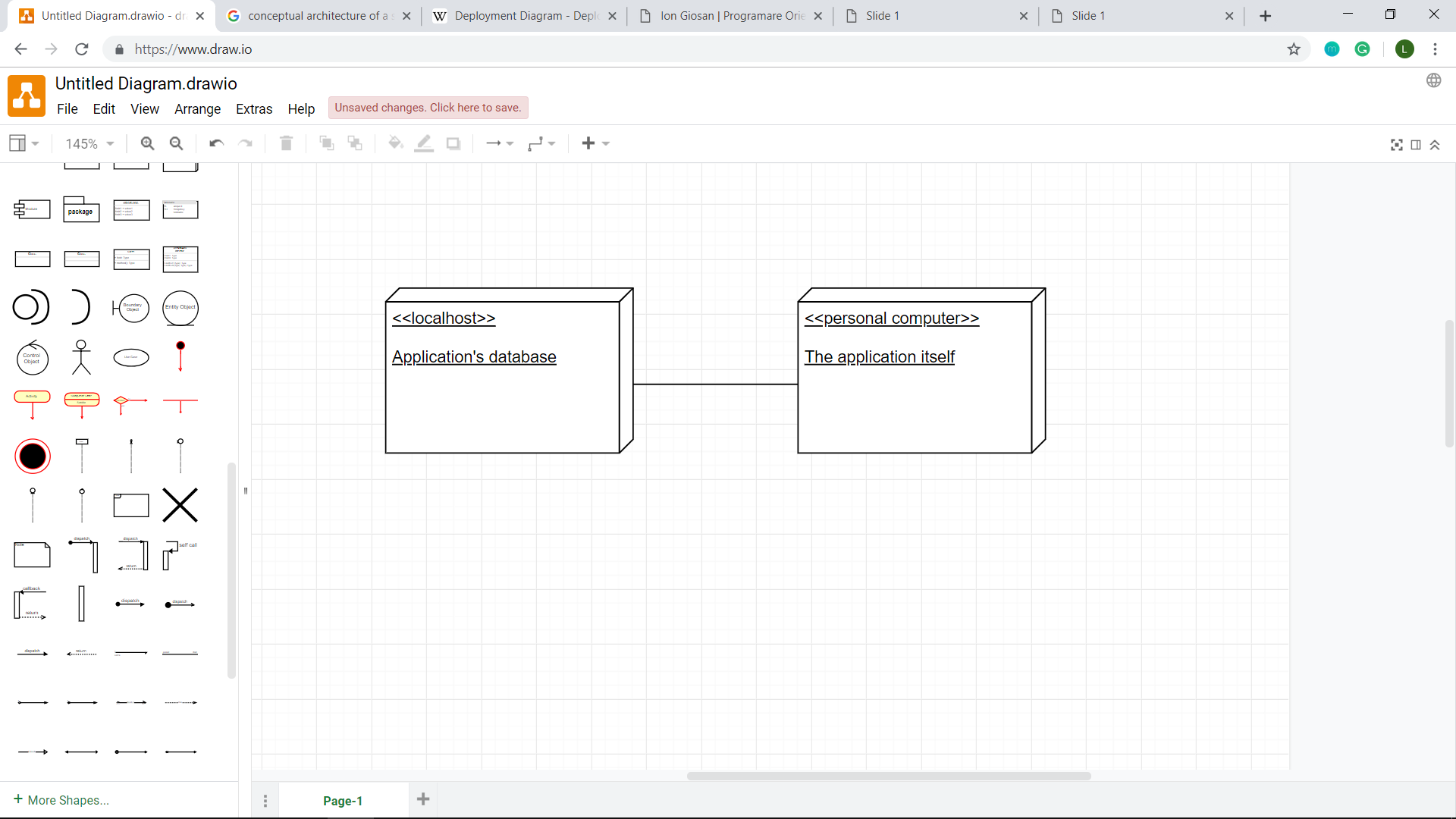
1. presentation layer (deals with user interface and with the way that data is presented to the user)
2. business logic layer (implements the functionalities that the application has)
3. data access layer (layer for accessing the data stored in the application’s database)

* Component diagram



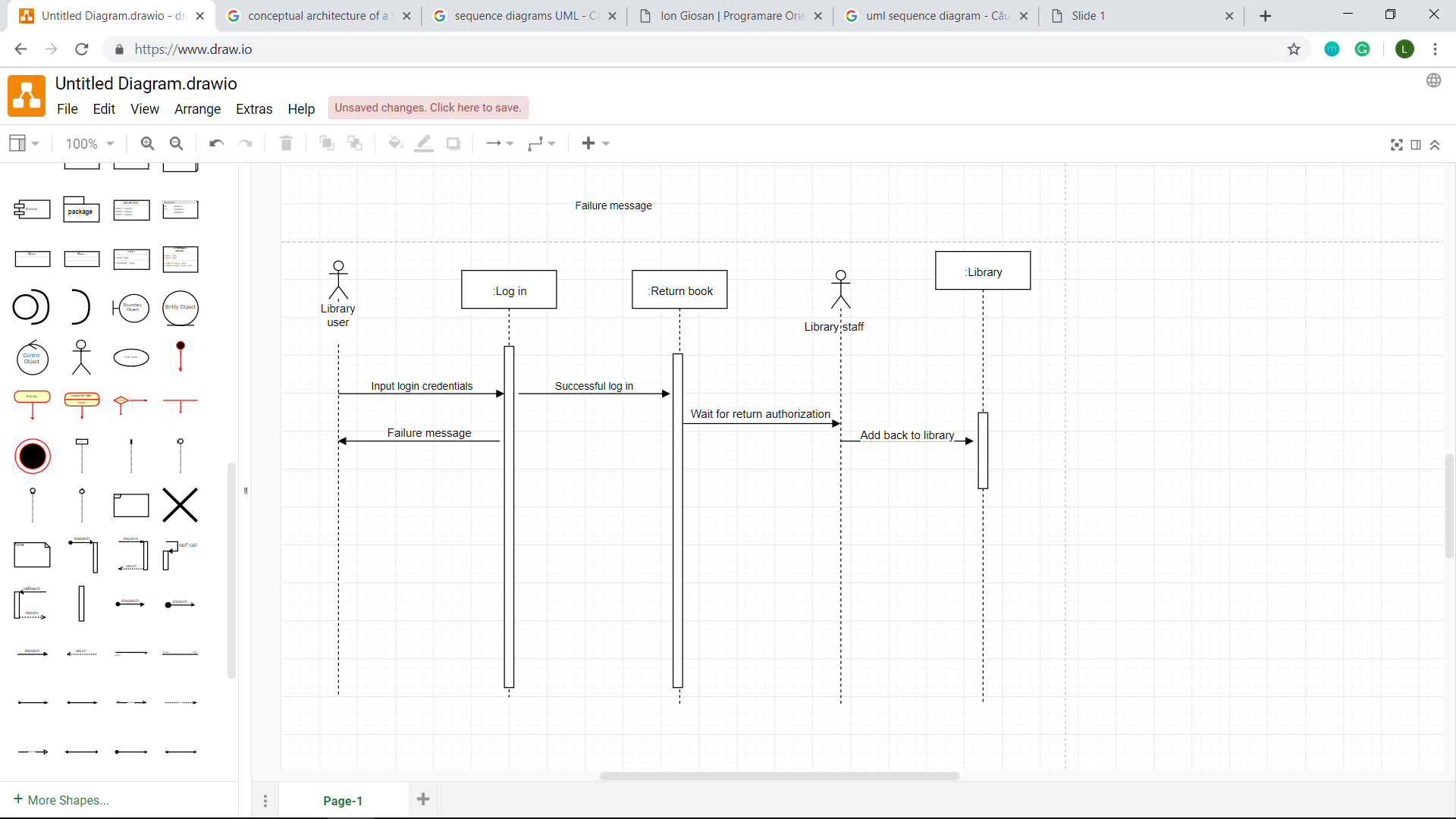
The main components of the application can be observed in the diagram above. The “user” and “staff components” model who can use the application and their characteristics. The “user facilities” component represents the actions that a user can do (log in, sign up, choose payment plan, borrow/return a book, view borrow history etc.). The “management activities” component represents the action that a staff member can do (validate payment plans and authorize returns).

Deployement diagram



For this stage of the application the database will be stored on localhost (I used PhpMyAdmin and XAMPP).

4. UML Sequence Diagrams

 The sequence diagram for the scenario consisting in a user returning a book is illustrated below:

5. Class Design

**5.1 Design Patterns Description**

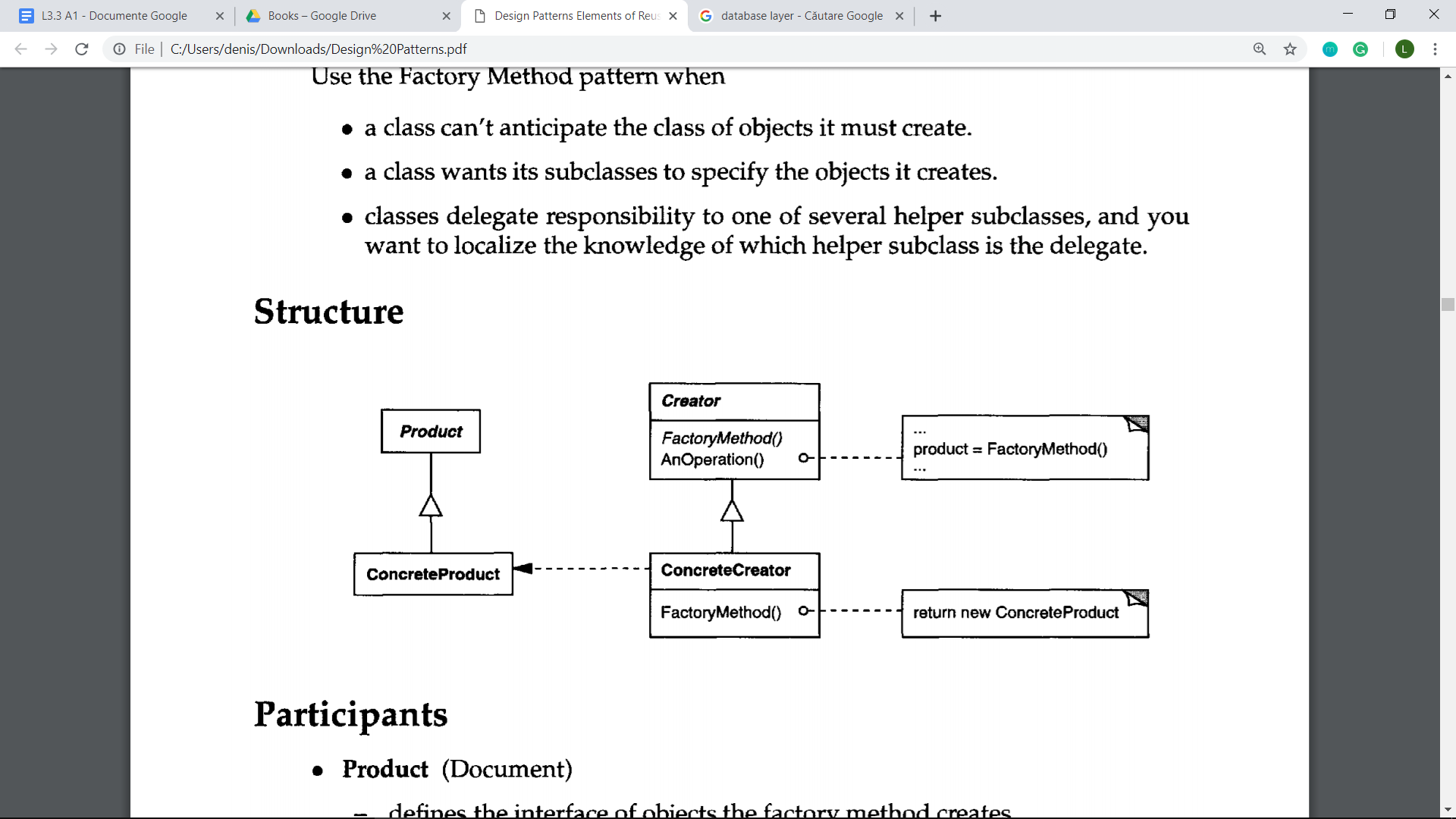
The design pattern used in the context of this assignment is the **Factory Method** creational pattern. The intent of this specific pattern is to define an interface for object creation, but by letting subclasses decide which class to instantiate. Another known name for this pattern is Virtual Constructor.

Applying this pattern is suitable in the following situations: when a class does not know the type of objects it needs to instantiate, when a class wants its subclasses to specify objects’ type or when a class delegates responsibility to one or more helper subclasses.

The major participants occurring when using this design pattern are:

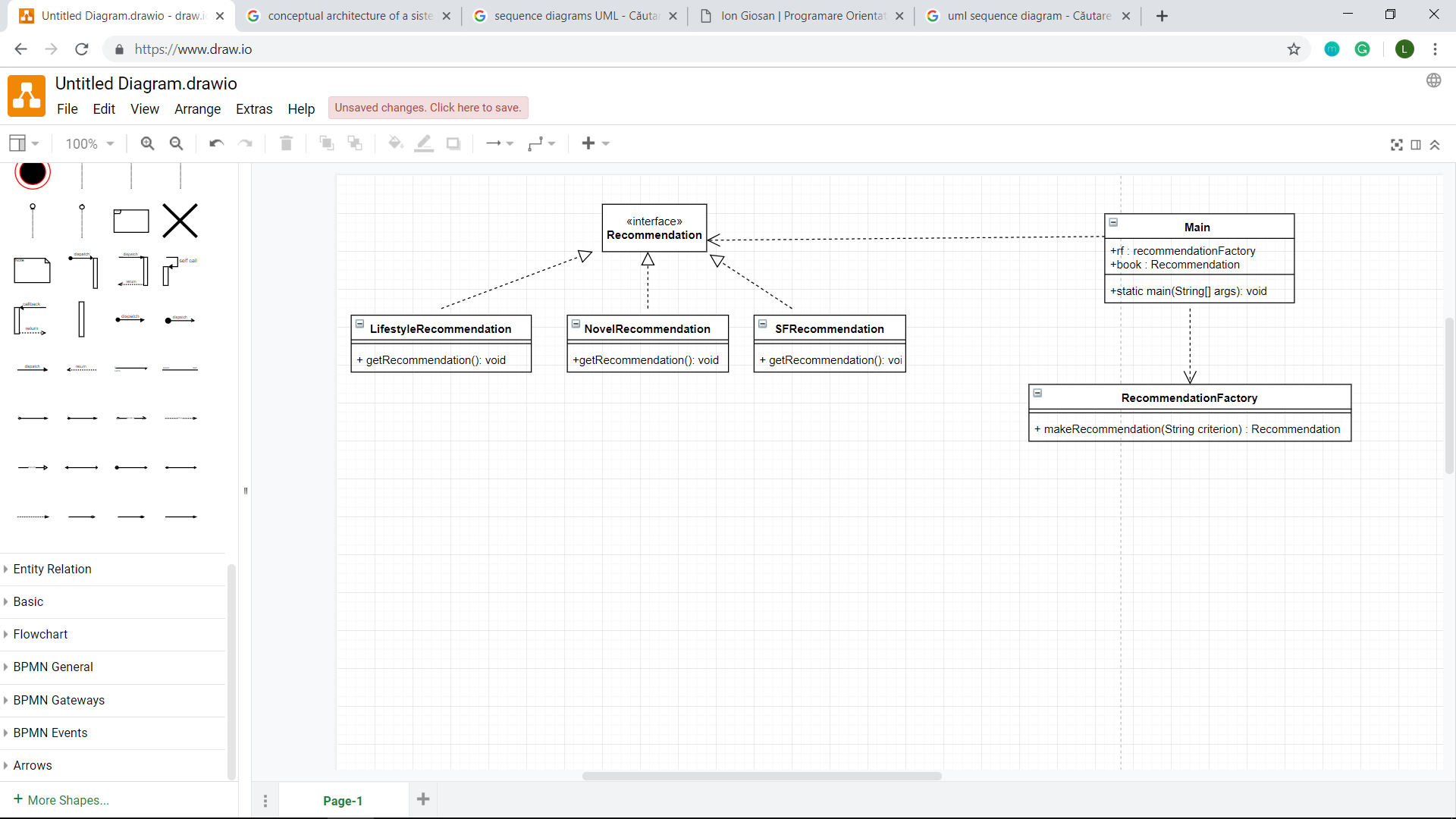
* **product** 🡪 it defines the interface for the objects that will be created by the factory method
* **concrete\_product** 🡪 this is the implementation of the product interface
* **creator** 🡪 declares the factory method which returns an instance of product
* **concrete\_creator** 🡪 overrides the factory method, so it returns an instance of a concrete product

The general structure of this pattern is illustrated in the following diagram:



**5.2 UML Class Diagram**

For making a recommendation based on a criterion introduced by the user, the factory method pattern is used . The class diagram corresponding to the implementation of this facility belonging to a user is showed next:



6. Data Model

A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to properties of the real world entities. Any data model can be viewed from three different perspectives: conceptual data model, logical data model and physical data model.

The data associated to the Book Management Service will be stored in a MySQL relational database.

7. System Testing

The system will be tested using the JUnit framework. Each unit (functionality/service) will be tested individually. A few testing results will be illustrated below.

After testing the individual units of the application, to assure that all the components inside the application work properly in relation to one another, I will use the application passing through all the main services it offers (user account creation, logging in, signing up, borrowing a book etc.)

8. Bibliography

**Layered Architecture** : <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>

**Factory method design pattern** : E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns. Elements of Reusable Object-Oriented Software,* Addison-Wesley Professional Computing Series, 1994

**Data model** : <https://en.wikipedia.org/wiki/Data_model>