<Book Store>

Student: Sechel Raluca-Rodica

**Group: 30233**

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

This is an application for the employees of a book store. The application has two types of user: a basic user which a regular user represented by the book store employee and an administrator user. All kinds of users need to provide a username and a password in order to use the application. The basic user can perform several operations such as: search books by genre/title/author and sell books. The administrator user can create/read/update/delete book’s information, create/read/update/delete employee’s information, generate PDF/CSV report on books out of stock. The information about books consists in genre, title, author, price and quantity, and the information about employee is represented by name, age and date of employment.

# Functional Requirements

In order to use the application, first you have to authenticate. You can choose one of the two possible users: basic user or administrator user. For the authentication you need a username and a password. After entering the username and password the application will validate the compatibility between them. If they match, you can perform the provided operations, if not, the application will show an error message: „Username and password don’t match!”. For username and password enter Strings.

If you want to authenticate as basic user with the right username and password, you can perform some book operation. So, here you can search for books using the title, the author or the book genre. If there is a book having the entered data, the book will be shown in the text area. If there are more books, you will see a list of books that meet the given condition. If you want to sell a book, the aplication will search for that book. If it does’t exist, in the text area you will see a message „The book doesn’t exist!”. On the other hand, if the book exists, but the quantity is 0, you will also get a message.

If you want to authenticate as an administrator with the right username and password, you can create/read/update/delete books and employees. The data will be shown in the text area if you choose to view the books or employees. On the other hand, an admin can generate reports on books out of stock. You can choose to generate PDF or CSV reports which will contain the books that are out of stock.

# Non-functional Requirements

* Availability

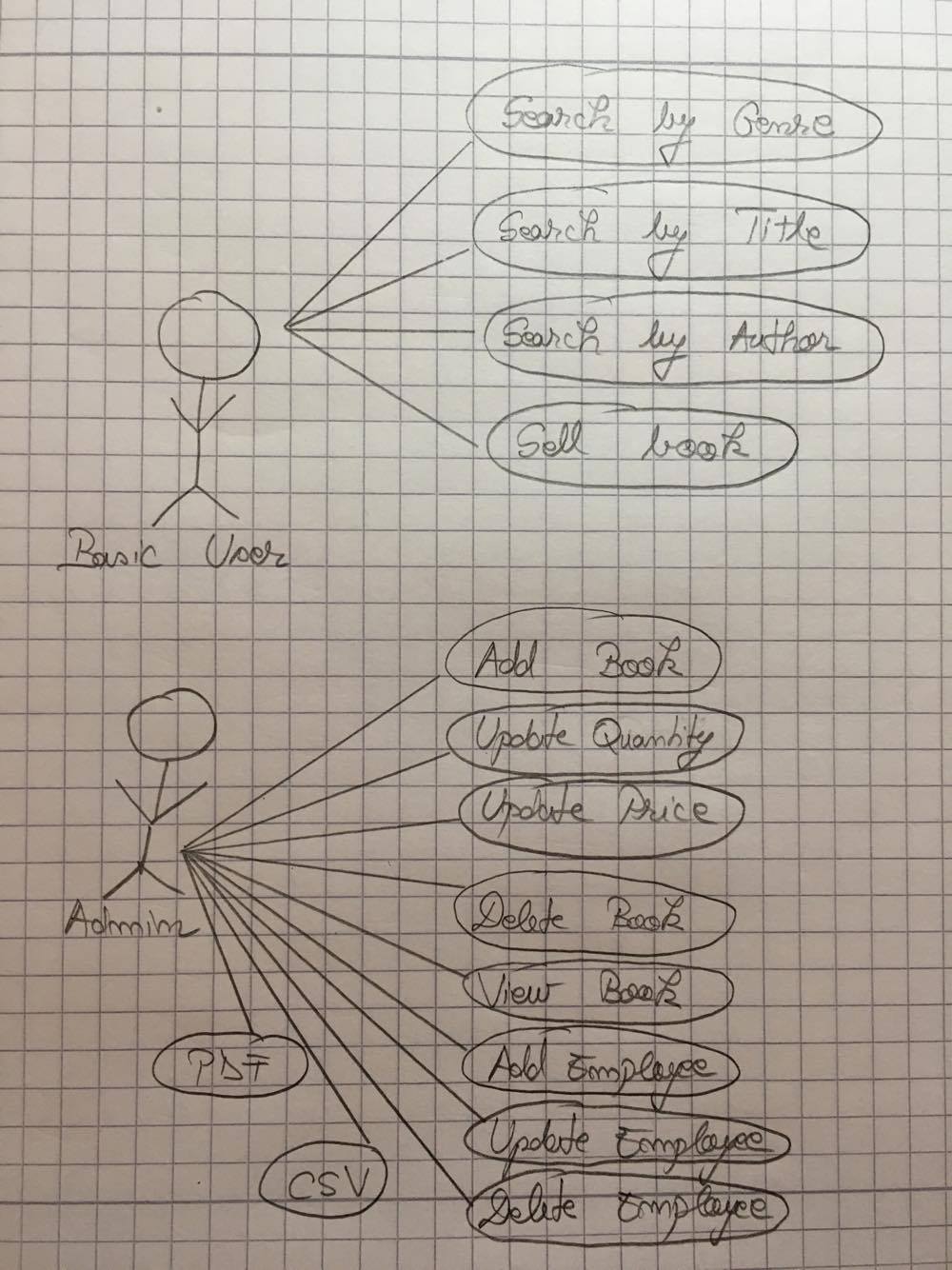
My system is available because it is capable to provide it’s intended services.

* Reusability

The application has some assets that can be used in some form within the software

product development process. For example we can use classes creating books and employees. Also, the class that models the xml files can be reused.

2. Use-Case Model



*Use case:* Sell book

*Level:* Sub-function

*Primary actor:* Employee

*Main success scenario:* The requested book is in stock and the customer has the amount of money to buy it.

*Extensions:* It is possible that the requested book is not available.

3. System Architectural Design

**3.1 Architectural Pattern Description**

MVC is a software architecture - the structure of the system - that separates domain/application/business (whatever you prefer) logic from the rest of the user interface. It does this by separating the application into three parts: the model, the view, and the controller.

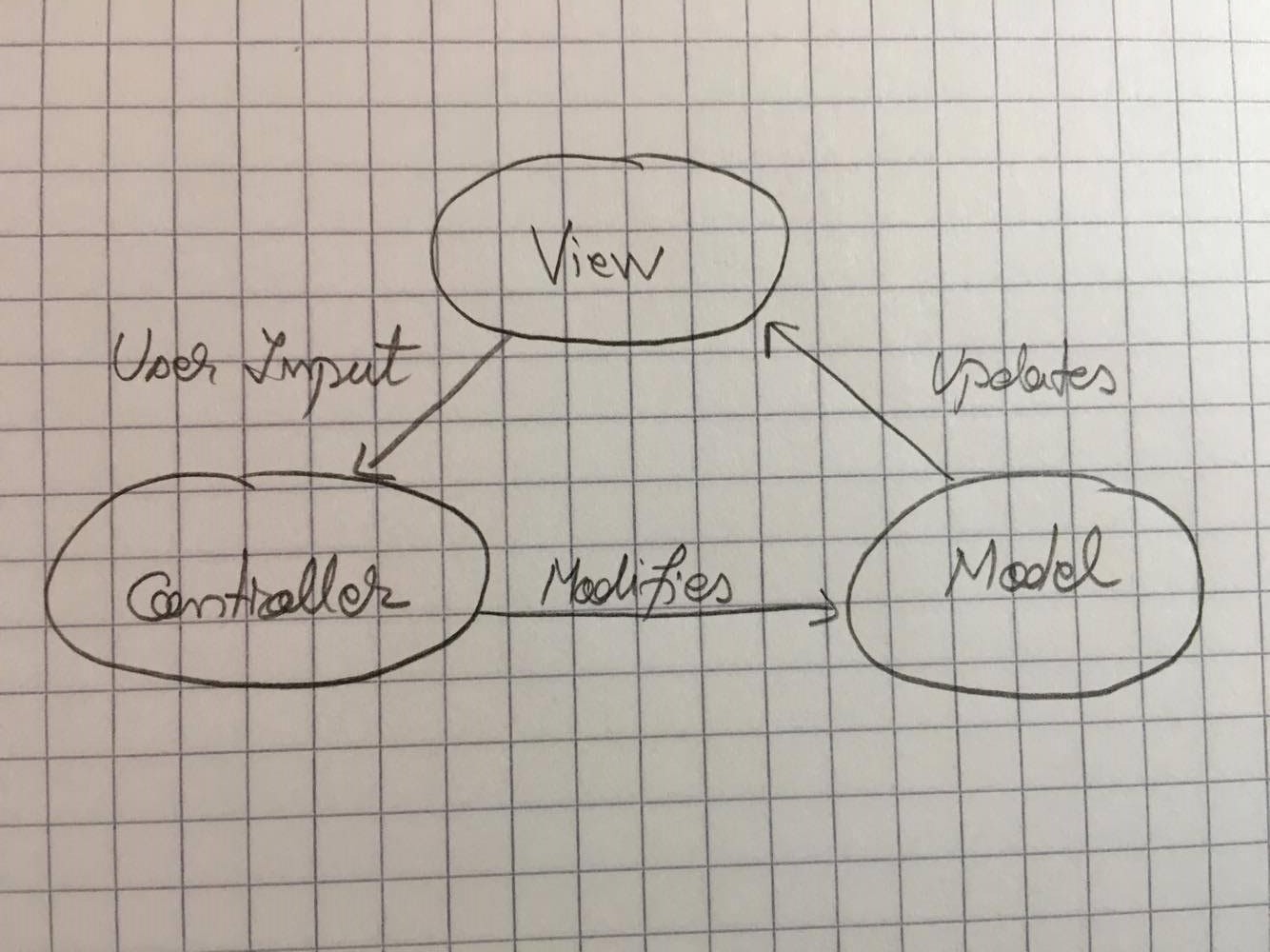
The model manages fundamental behaviors and data of the application. It can respond to requests for information, respond to instructions to change the state of its information, and even to notify observers in event-driven systems when information changes. This could be a database, or any number of data structures or storage systems. In short, it is the data and data-management of the application.

The view effectively provides the user interface element of the application. It'll render data from the model into a form that is suitable for the user interface.

The controller receives user input and makes calls to model objects and the view to perform appropriate actions.

All in all, these three components work together to create the three basic components of MVC

**3.2 Diagrams**

**

 The *model* is the central component of the pattern. It expresses the application's behavior in terms of the problem domain, independent of the user interface. It directly manages the data, logic and rules of the application.

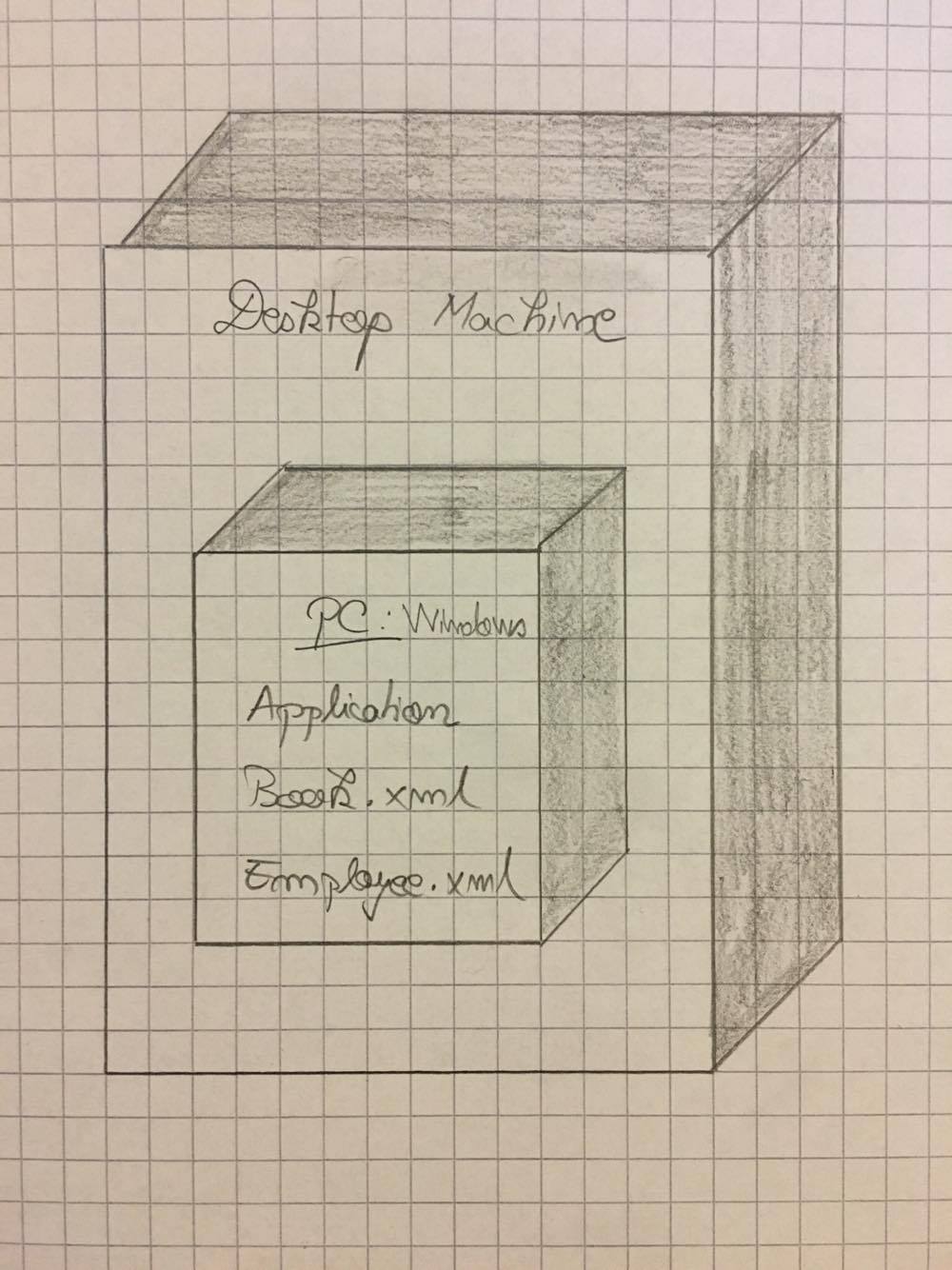
 A *view* can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

 The third part, the *controller*, accepts input and converts it to commands for the model or view.

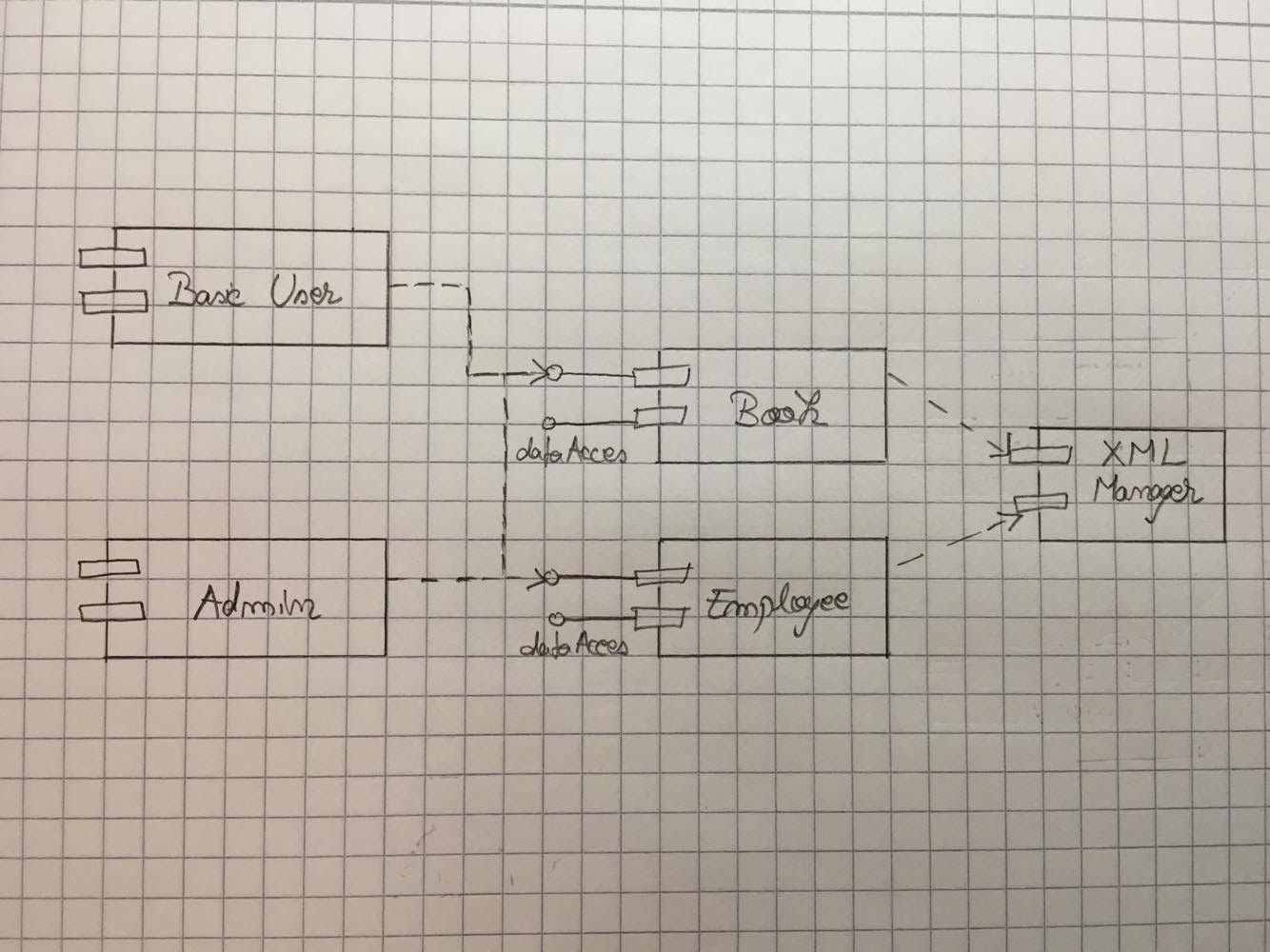
In addition to dividing the application into three kinds of components, the model–view–controller design defines the interactions between them.

* A *model* stores data that is retrieved according to commands from the controller and displayed in the view.
* A *view* generates new output to the user based on changes in the model.
* A *controller* can send commands to the model to update the model's stat. It can also send commands to its associated view to change the view's presentation of the model.

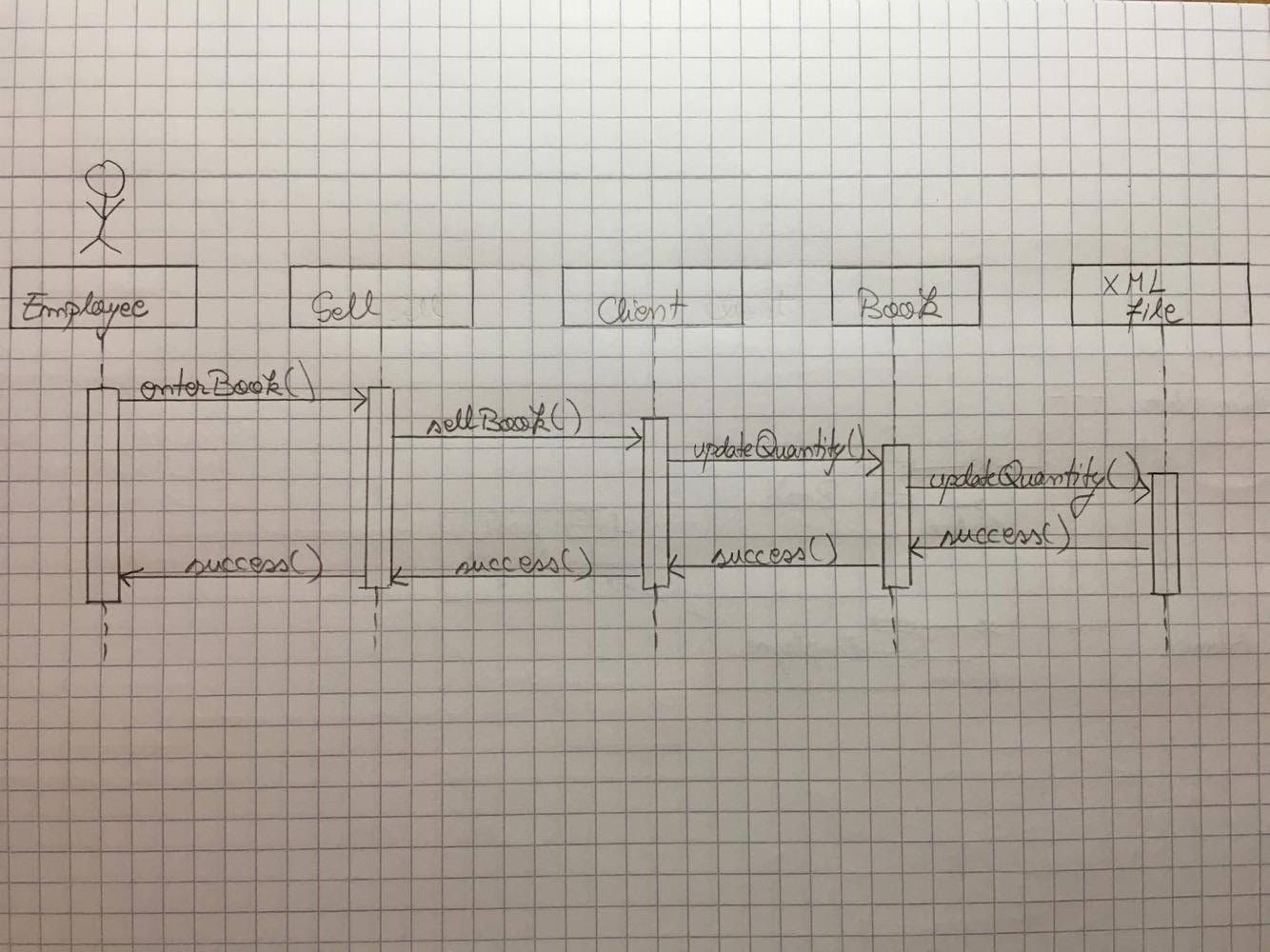
Here is the deployment diagram for my application:



Here is the component diagram for my application:



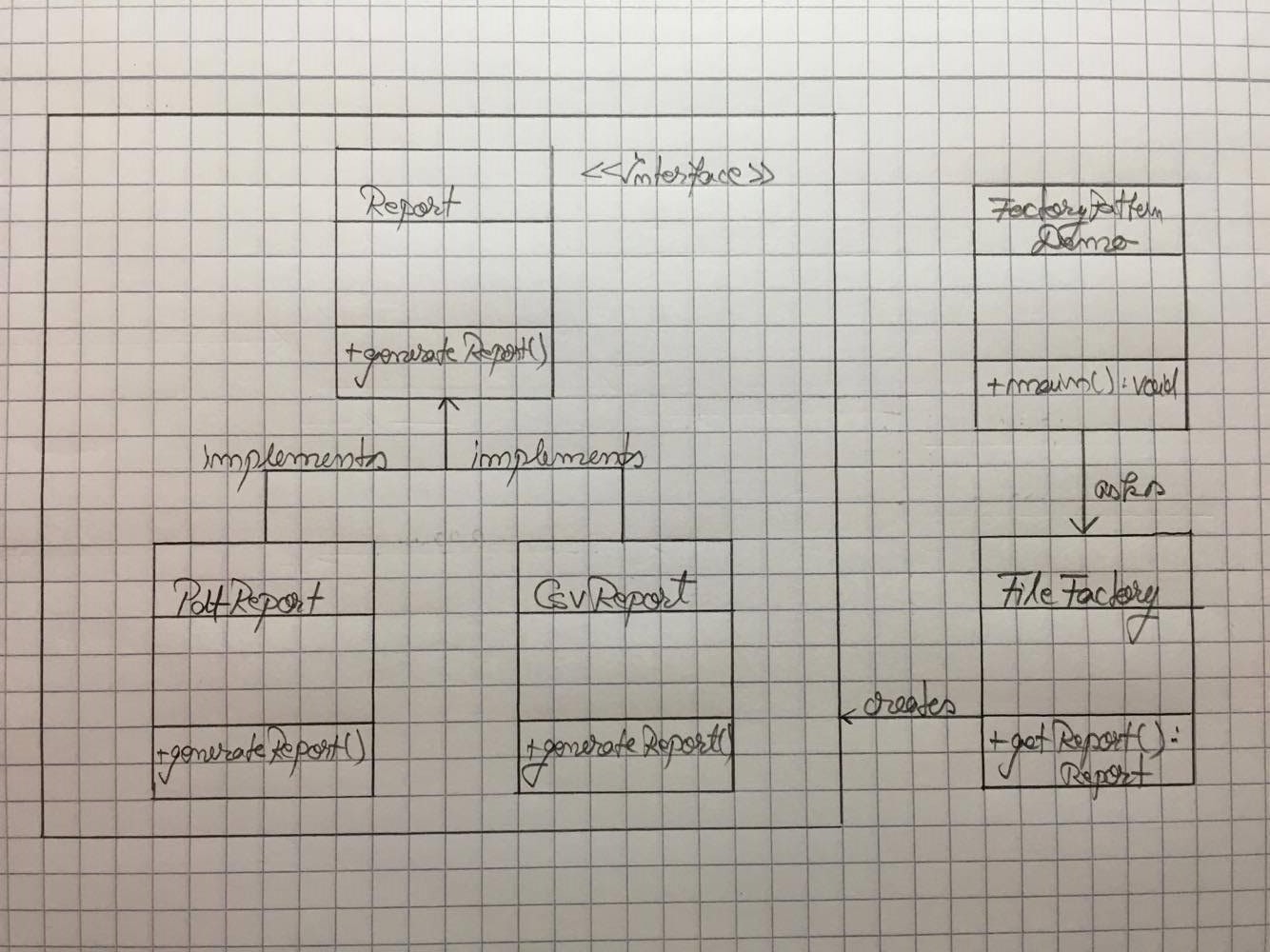
4. UML Sequence Diagrams

**

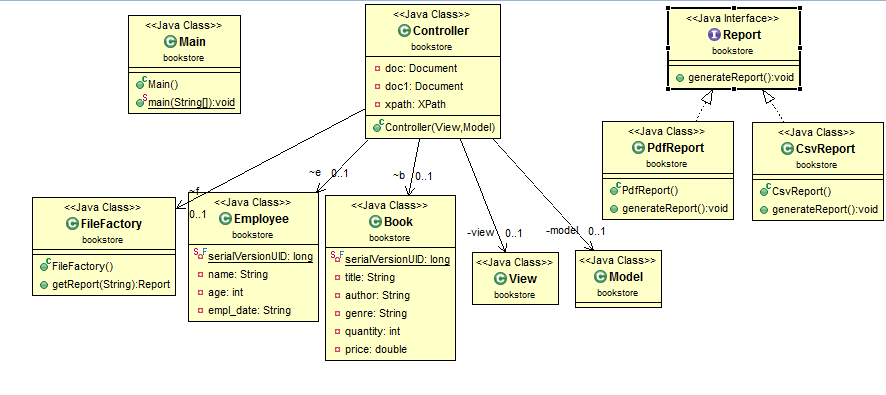
5. Class Design

**5.1 Design Patterns Description**

For generating a report I used Factory pattern. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface. I’ve created Report interface and concrete classes implementing the Report interface. After that, I’ve created a factory class FileFactory. The Controller will use *FileFactory* to get a *File* object. It will pass information (*PDF / CSV)*  to *FileFactory* to get the type of object it needs.



**5.2 UML Class Diagram**



6. Data Model

The XML data model follows the XPath 2.0 and the XQuery 1.0 data model. This data model provides an abstract representation of one or more XML documents or fragments.

The purpose of the data model is to define all permissible values of expressions in XPath, including values that are used during intermediate calculations. Every XPath expression takes as its input an instance of the data model and returns an instance of the data model. The XML data model is described in terms of sequences and items, atomic values, and nodes.

The XPath data model is based on the notion of a sequence. The value of an XPath expression is always a sequence. A sequence is an ordered collection of zero or more items. An item is either an atomic value or a node.

An atomic value is an instance of one of the built-in atomic data types that are defined by XML Schema.

A node conforms to one of the types of nodes that are defined for XPath. These node types include: document, element, attribute, text, processing instruction, comment, and namespace nodes.

Before an XPath expression can be processed, the input documents must be represented in the XML data model.

7. System Testing

For each CRUD operations I checked if the data exists in XML files. For example, if you want to sell a book, the application verifies if the book exists in stock and then verifies if the quantity is greater than 0. If those conditions aren’t satisfied, the applications shows a message like “The book doesn’t exist!”. If you want to search for a book, the application shows the titles if the data exists in XML files, and if it doesn’t exist it returns an empty list. On the other hand, if you want to access something that doesn’t exist in the XML files, you get an error message.

8. Bibliography

<https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_71/rzasp/rzasp_datamodel.htm>

<https://www.tutorialspoint.com/design_pattern/factory_pattern.htm>

<https://www.w3schools.com/xml/xml_xpath.asp>