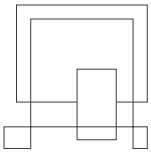


Read with Panda

Michał Karol Pompa 266494, Michaela Golhova 266099

Michal Ciebien 266908, Matej Michalek 266827

Daniela Koch 266502

Supervisor:

Jan Otto Munch Pedersen,

Jakob Knop Rasmussen

VIA University College

56435 characters

ICT Engineering

3rd semester

December 2018

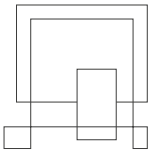
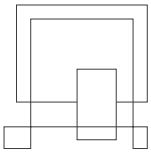
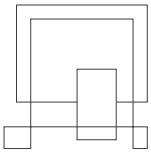


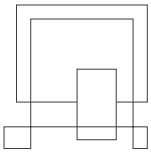
Table of content





Abstract

The following document presents the development of a system with the purpose of improving the accessibility of books. It is divided into nine subparts. The first one takes under consideration why the accessibility of books is a crucial factor in today's world. It is followed by the analysis of the system's needs, including what actors should it contain and what should their actions be, illustrated also on diagrams. Next is the design of the system, specifying among others the choice of the 3-tier architecture design pattern, the chosen technologies, e.x. the choice of a website for UI, the needed security and the solution for it and the use of sockets in communication with the database. Moreover, it presents the blueprint of the system. What is next, is the implementation part. In this chapter, one can see how the blueprint was converted into code. Some of the most important parts are the authentication of users with the use of session keys and UUIDs, sending requests is Java using the Spring MVC Framework and in C# using the REST web services, the communication with the database provided by the Hibernate ORM framework, creating the UI using React and the communication with the database server accomplished with TCP sockets. Afterwards is a section about tests, describing how they were conducted with Unit tests and with Requirement Test Descriptions. It is followed by a section about the results and the future of the project.

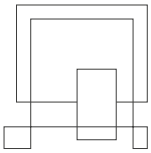


1 Introduction

Ernest Hemingway once said: “There is no friend as loyal as a book”. That sentence can be read on a few levels. One aspect of it is being shaped by a book. On that point an association between reading and the level of empathy can be distinguished, namely the less adults read, the smaller they define their empathy to be (Jamil Zaki, 2011). That leads to the statement, that reading boosts one’s quality of empathy, as well as the ability to see the world from other people's perspective (Christopher Ingraham, 2016). What is more, “a book is a device to ignite the imagination”. As Alan Bennett stated, reading develops imagination and creativity (Dennis J. Sumara, 2002). But this is not everything a book has to offer! Among the vast range of its advantages reading has been proven to reduce the stress level (Putai Jin, 1992) and improve one’s communication skills by extending one’s vocabulary (Gery Deer, 2016).

Taking under consideration all the enumerated benefits one can gain from books, reading should be a common habit and should be growing into importance and popularity as time passes. Unfortunately the reality is different. In 1982 the percentage of Americans reading books for pleasure was on the rate of almost 57%. That may seem as a small number, but the perspective changes while comparing with the percentages from 2015, which is only 43% of the American population. More than every second American did not read any work of literature during the year 2015. And that does not only count to the proverbial man in the street, but even the educated citizens, that are supposed to be the intelligence in the country, who have put enough effort in gaining knowledge, to achieve a graduate degree, are on the level of only 68% of “intelligence” who have read any work of literature for pleasure (Christopher Ingraham, 2016).

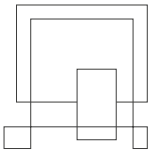
There are multiple reasons to the issue of the decreasing popularity of books. One of them is being surrounded by various types of distraction. As technology progresses, so does the number of ways of spending free time. Nowadays the act of reading a book has to compete for people’s attention with the internet, movies, or computer games. Unfortunately for the books, people tend to choose the latter ones (Christopher Ingraham, 2016; Michael Kozlowski, 2018). Additionally, is the matter of availability of



books. As most of the competitors of books can easily be found on the internet, the case is often looking differently for books. Even though a book can be found in an e-version, many people declare to prefer the classic printed versions. Moreover, paper based books increase one's reading comprehension more than e-books (Hanho Jeong, 2012). That leads to the issue of availability. While to find a book one has to access multiple sides and search for it, or even (God forbid!) go out and look in a library or a store, still not having the certainty of finding it, it is just simpler to use the time on the internet.

Bearing everything that has been mentioned in mind, it can clearly be seen that the humanity is in a need of a way to increase the popularity of books again. One approach to this issue could be combining the traditional books with technology. As it has been stated, people fancy technology. That is why combining books with technology could attract them, if done in an appealing manner. Another approach could be simplifying the process of looking for a book. Namely, connecting libraries with bookstores. In that case, readers would have a bigger range of artworks to choose from and would save time they would normally have spent on looking for books.

More details can be found in the project description appended in Appendix A.

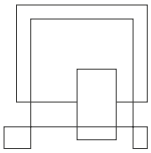


2 Requirements

Considering the description of the needed functionalities of the system, a list of requirements is stated below. The requirements are divided into functional and non-functional.

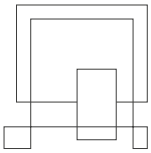
2.1 Functional Requirements

1. The guest should be able to create an account with a name, surname, address, e-mail address and phone number
2. The user should be able to log in to the system
3. The user should be able to log out from the system
4. The customer should be able to see his/her personal information
5. The customer should be able to borrow books from the library
6. The customer should be able to buy books from the bookstore
7. The customer should be able to see books he/she has borrowed and has not given back in his/her account
8. The user should be able to see details of books
9. The customer and guest should be able to search for books (by inputting the name, author, isbn, year or category, filtered by (specific) bookstore/library)
10. The administrator of an institution should be able to search for books in his/her institution
11. The administrator of an institution should be able to manage books in the institution (add, delete)
12. The administrator of an institution should be able to see the list of borrowed books and mark chosen ones as returned
13. The administrator of an institution should be able to confirm book orders
14. The administrator of the system should be able to manage administrators of the bookstores and libraries
15. The system should send reminder emails 3 days before the return date
16. The system should be sending confirmation emails to the users after the administrator of the bookstore has confirmed the users order



2.2 Non-Functional Requirements

1. The system must be written in Java and C#
2. The system must be using a database for persistence
3. The system must be based on a 3-tier architecture
4. Communication in the system must be done with the web service and sockets (with a communication protocol)
5. The system must include a GUI dedicated for the clients(administrator of the system, administrator of a library, administrator of a bookstore, customer)
6. The system must store passwords in a secure way



3 Analysis

The first stage of creating a project is analysis. This is where the needs of the customer are analyzed and defined. It is the most important part of the system development, as it creates the base of it. Therefore, next stages as Design, Implementation and Test were derived from deep analysis of this stage.

As the purpose was to make a system, that would improve the accessibility of books by combining functionalities of bookstores and libraries into one system, the main questions considering the problem are as follows:

- What functionalities does a bookstore provide?
- What functionalities does a library provide?
- What should the user be able to do in the system?
- What would make the user interface attractive?
- What information should be stored about users?
- Is membership needed?
- What work methodology should be used while developing the project?

Extended analysis takes into consideration every issue, but sometimes there are some difficulties that are not possible to overcome and that is why creating a list of delimitations is needed. Due to specific requirements and the lack of time the delimitations are summarized as follows:

- The system will connect only one bookstore with one library. However, the architecture and design should support connecting more institutions.
- The membership in the Library won't be taken into consideration
- The system will not fulfill the requirements with minor importance and one with normal, which are:
    - The administrator of the system should be able to manage administrators of the bookstores and libraries
    - The customer should be able to see his/her personal information
    - The customer should be able to see books he/she has borrowed and has not given back in his/her account
    - The system should send reminder emails 3 days before the return date

3.1 System analysis

Basing on requirements, one can proceed with analyzing the main parts of the system's needs. The first thing to start with is analyzing the domain. The outcome can be seen in the domain model diagram, Figure 1. Moreover, a Data Glossary, explaining the used words through the development of the system can be found in Appendix B.

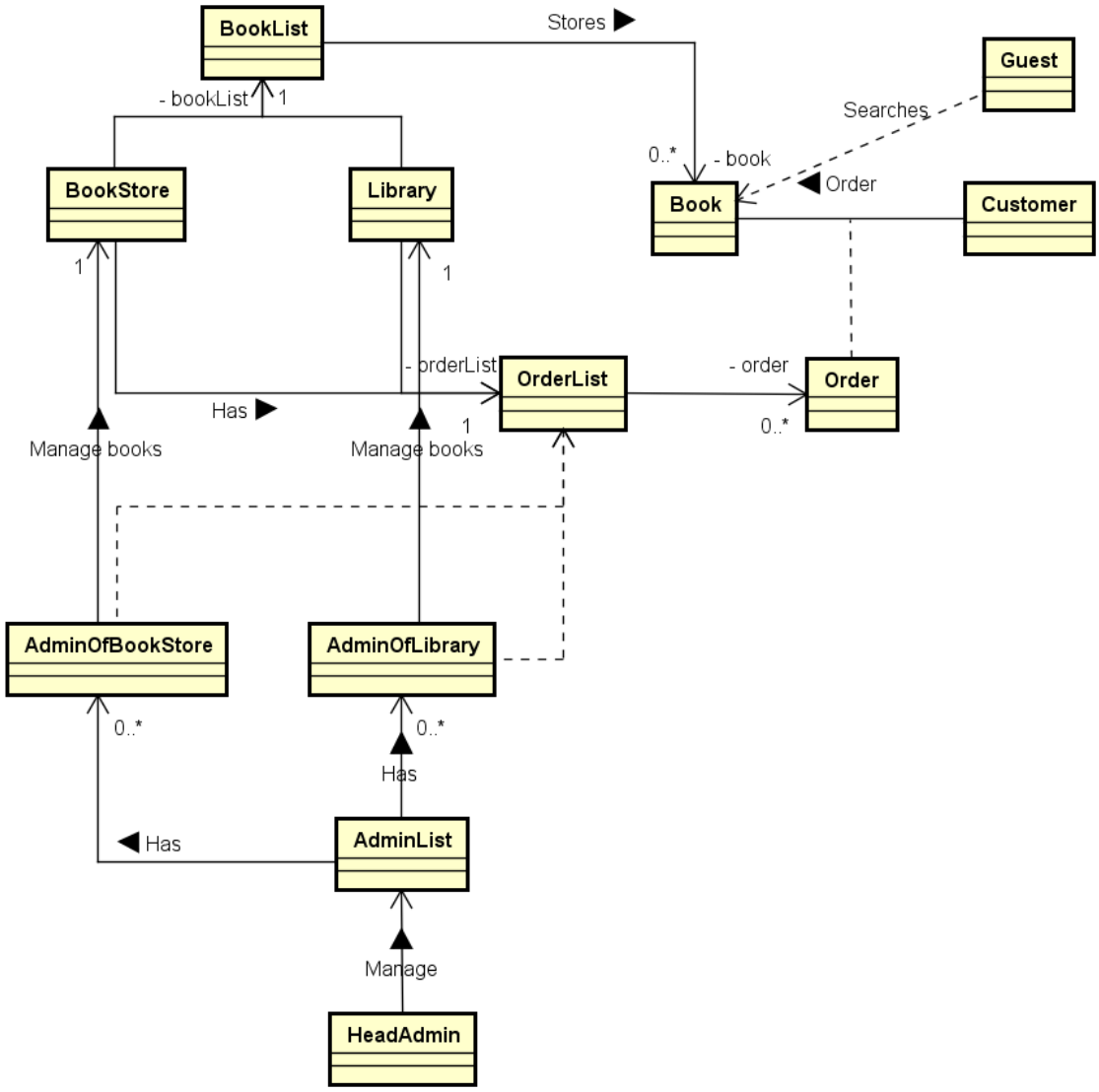
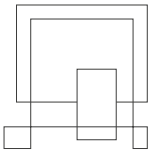


Figure 1

It can be seen that the Book is in the center of the domain and everything happens around it. The Library and the BookStore own many books (they are storing them in the



BookList). The Customer is ordering books and the orders are going to the OrderList which is managed by Administrators of the Library and the BookStore. Administrators are also managing books in the institutions. The Guest can only search for books. Moreover, the HeadAdministrator (Administrator of the system) is responsible for managing the administrators of institutions and by this, also the institutions.

3.2 Use cases

Having stated the requirements, use cases can be created basing on them. The use case diagram below (Figure 2) presents different ways of using the system and also divides the possible actions between actors. The system has following actors: Guest, Customer, Administrator of the Library. Administrator of the Bookstore, Administrator of the System and Time. Each of them can perform different actions in the system. The result of this part of analysis are the following use cases:

1. Login into an account
2. Logout from an account
3. Order a book
4. Create an account
5. Search for a book
6. See personal information
7. Return a book
8. Manage books
9. Confirm book orders
10. Manage administrators
11. Send remainder

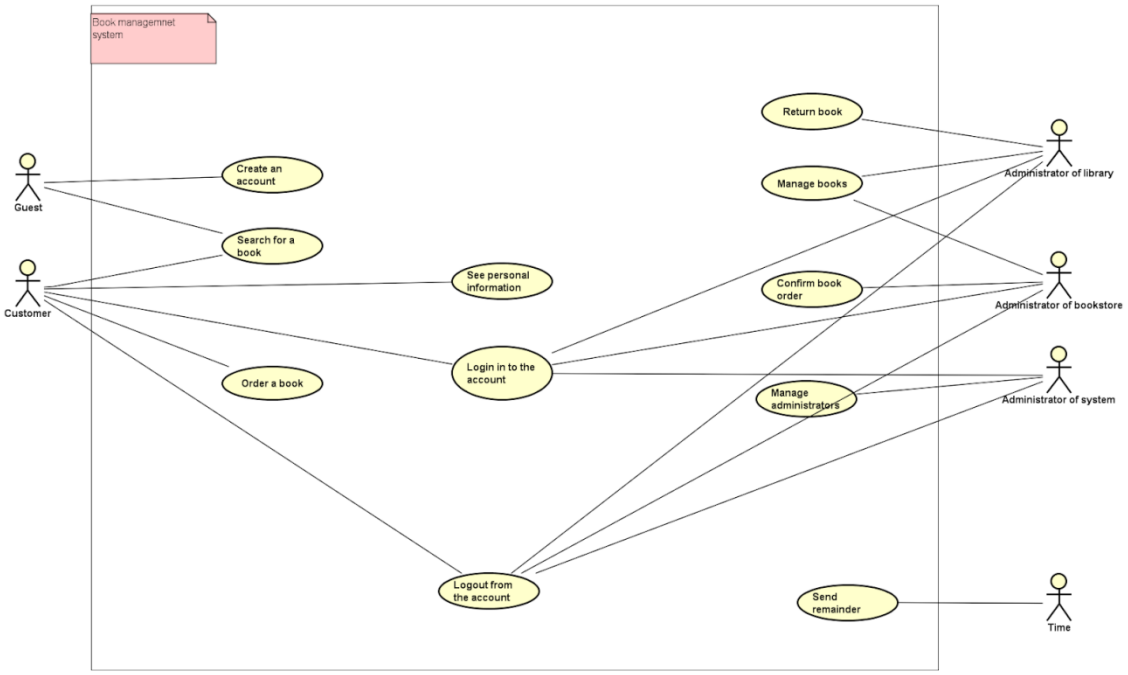
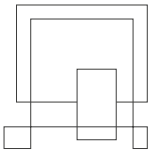


Figure 2

One of the core functionalities of the system is ordering a book, which has been analyzed in the Order a book use case. The use case description can be seen in Figure 3. Other use cases can be found in Appendix C.

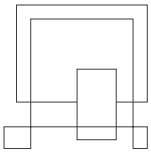


ITEM	VALUE
UseCase	Order a book
Summary	The Customer chooses whether to buy or borrow an already selected book
Actor	Customer
Precondition	The Customer is logged in the system and is seeing book details
Postcondition	The Customer has made an order for a book
Base Sequence	1) The Customer chooses the library and clicks the "Borrow" button next to it. 2) The system makes an order for the book in the selected library. 3) The system displays information that the order was made successfully.
Branch Sequence	1) IF the Customer wants buy, THEN he/she chooses the bookstore and clicks "Buy" button. 2) The system makes an order for the book in selected bookstore. 3) The system displays information that the order was made successfully.
Exception Sequence	1) IF the book is not available in any library THEN instead of list of libraries th e system displays information that the book is not available in any library. 1) IF the book is not available in any book store THEN instead of list of bookst ores the system displays information that the book is not available in any book store. 3.1) IF the book became not available during this proccess, THEN the system displays information "The book is no longer available in this institution. Sorry."
Sub UseCase	Login in to the account
Note	

Figure 3

3.3 Security

The system cannot be successful if it is not safe. Users must be sure, that their personal information is not endangered and that only they have access to their



accounts. The topics such as possible threats and vulnerabilities of the system should be analyzed. The outcome of that analysis is the Threat Model, which enumerates threats that the system can face. Moreover, the goal and means of the attacker are stated to every threat. Last part is concerning the place and the person being able to conduct a specific attack (expressed in EINOO scale).

3.3.1 Thread Model

1) DoS/DDos attack

The Attacker is sending a large number of search requests to the BookService. When the number of requests reaches the limit, the BookService runs out of resources and is unable to process the requests. The BookService crushes and the system is unable to work as it is a key component of the system.

a) Goal of the attacker

The goal of the attacker is Denial of Service and the objective is Availability of the application.

b) Means of the Attacker

Active - Blocking the application

c) EINOO

i) **WHO** - Both Internal and External, because the search request can be done both by a registered user of the system and a guest.

ii) **WHERE** – Online

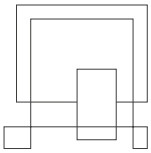
2) Unauthorized access to the Database

The Attacker is able to acquire the password to the database and log into it through SSH. When the attacker has access to the database, he/she can modify the data or erase all data from the database.

a) Goal of the attacker

Elevation of privileges - the attacker can create an administrator account for himself/herself. - Authorization

Information Disclosure - the attacker has access to information that he/she should not have. - Confidentiality



Tampering - the attacker can modify data without being detected - Integrity.

b) Means of the Attacker

Active - Modification

c) EINOO

i) **WHO** - External - the attacker does not have to be a user of the system.

ii) **WHERE** – Online

3) Unauthorized access to an admin's account

The Attacker is able to acquire the password to an admin's account. After that, the attacker is able to modify data in a specific institution, e.g. confirm an order or return a book.

a) Goal of the attacker

Elevation of privileges - Authorization

Information Disclosure - Confidentiality

Tampering - Integrity

b) Means of the Attacker

Active - Modification

c) EINOO

i) **WHO** - Internal, External - if the attacker has access to the admin's computer in the institution.

ii) **WHERE** – Online, Offline - if the attacker has access to the admin's computer in the institution.

4) Unauthorized access to a customer's account

The Attacker is able to acquire the password to a customer's account. After that, the attacker is able to make an order as though he/she was a customer.

a) Goal of the attacker

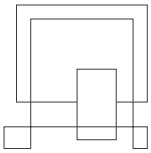
Spoofing Identity - making orders on behalf of other customers - Authentication

Tampering - changing the customer's personal data - Integrity

b) Means of the Attacker

Active - Modification

c) EINOO



i) **WHO** - Internal.

ii) **WHERE** – Online.

5) Replay attack on the library (borrowing a book)

The Attacker intercepts a message with a borrow request sent to the library. Then, the Attacker is able to replay the message and borrow all books with the same isbn as the one in the original message.

a) Goal of the attacker

Denial of Service - the book is not available for customers - Availability

Spoofing Identity - making orders on behalf of other customers - Authentication

Tampering - modification of data - Integrity

b) Means of the Attacker

Active - Modification

c) EINOO

i) **WHO** - Internal.

ii) **WHERE** – Online.

6) Manually shutting down the server

The Attacker breaks into the room with the server and pulls of the plug.

a) Goal of the attacker

Denial of Service - the system is shut down and unavailable - Availability

b) Means of the Attacker

Active - Blocking

c) EINOO

i) **WHO** - External.

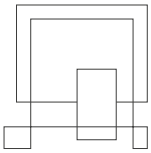
ii) **WHERE** – Offline.

7) Unauthorized access to an admin's computer in the institution

The Attacker breaks into the room with the institution's computer where admin is logged in and interferes the data.

a) Goal of the attacker

Information Disclosure - Confidentiality



b) Means of the Attacker

Active - Modification

c) EINOO

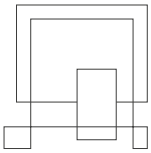
i) **WHO** - External.

ii) **WHERE** – Offline.

3.3.2 Risk assessment model

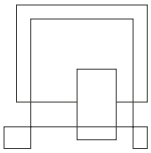
After analyzing the possible threats, one can proceed with creating Risk Assessment. In this part each threat has a frequency, preventive measures, incident prevention, threat effect and corrective measures assigned. Moreover, the parts that are needed for risk calculations are marked with Low, Moderate or High (or Good, Moderate, Bad) indicators, that are rating the solution. The risk is calculated basing on the threat frequency with preventive measures (as risk likelihood), threat effect and corrective measures (as incident consequences). (Table 1)

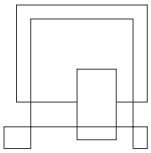
Threat	Threat frequency	Preventive Measures/ Vulnerabilities	Incident Prevention	Threat Effect	Corrective Measures	Risk
DDoS/DoS attack	Low	Use of an external server with unknown configuration High	Buy better server hosting	BookService is down and the system is unable to work. High	Buy a new server hosting and apologize through email to the customers. Bad	Potential loss of customers High
Unauthorized access to DB	Low	Posting password to DB on GitHub High	Storing the password in a secure way; Frequent changes of the password; Making frequent backups	Loss or misuse of data or addition of data High	Restore backups; Change of password Good	Potential loss or modifications of books or customers data; potential inconsistency between the system and reality Moderate



Unauthorized access to an admin's account	Low	Encrypted passwords in DB Low	Better authorization; Frequent changes of passwords; Making frequent backups; Digital signature	Loss or misuse of data or addition of data High	Restore backups; Change of passwords Good	Potential loss or modification of books or customers data; potential inconsistency between the system and reality Low
Unauthorized access to a customer's account	Moderate	Encrypted passwords in DB Low	Digital signature and encrypting passwords	Undesired actions taken on behalf of a customer High	Change of password Moderate	Potential loss of the customer's money Moderate
Replay attack on the library (borrowing a book)	Moderate	SSL configured on the server Low	Use of session keys and nonces or ssl (sequence numbers and nonces), Frequent backups	Books data has changed Hard to detect Moderate	Restore backup Good	Potential unavailability of a book Moderate
Manually shutting down the server	Low	Server owned by a third party Moderate	Buy own server	System is shut down and unable to work High	Contact server owners Moderate	System is not available for some time Moderate
Unauthorized access to the admin's computer in the institution	Low	Computer located in restricted area Moderate	Decrease session key's lifetime; Hire guards to guard the computer; Frequent backups	Undesired actions taken on behalf of an administrator High	Change the password; Restore backup Moderate	Potential loss of data; Time spent to restore Moderate

Table 1





3.4 Choose of GUI

For the GUI of the system, the decision to use a website was made. This followed the discussion about different ways of making the system easily interactive with users. Website, compared to other GUI forms is entirely cross-platform and due to its design flexibility, it can provide an excellent User Experience.

3.5 Database

The business entities have been derived from the functional requirements. Those entities are: Book, Library, BookStore and Customer. The domain diagram and logical model describing them more in depth can be found in Appendix D.

All of them are strong entities of the system. In order to model the relations between them, weak entities are also needed. For example, when the Book is possessed by the Library, this information is represented by the LibraryStorage. Other relation entities are: BookStoreStorage, BookStoreOrder and LibraryOrder. BookStoreAdministrator and LibraryAdministrator are dependent on the BookStore and Library respectively. Those entities are responsible for the Institution Administrator. The relations between the entities can be seen in Figure 4. The ER diagram is also attached in Appendix C.

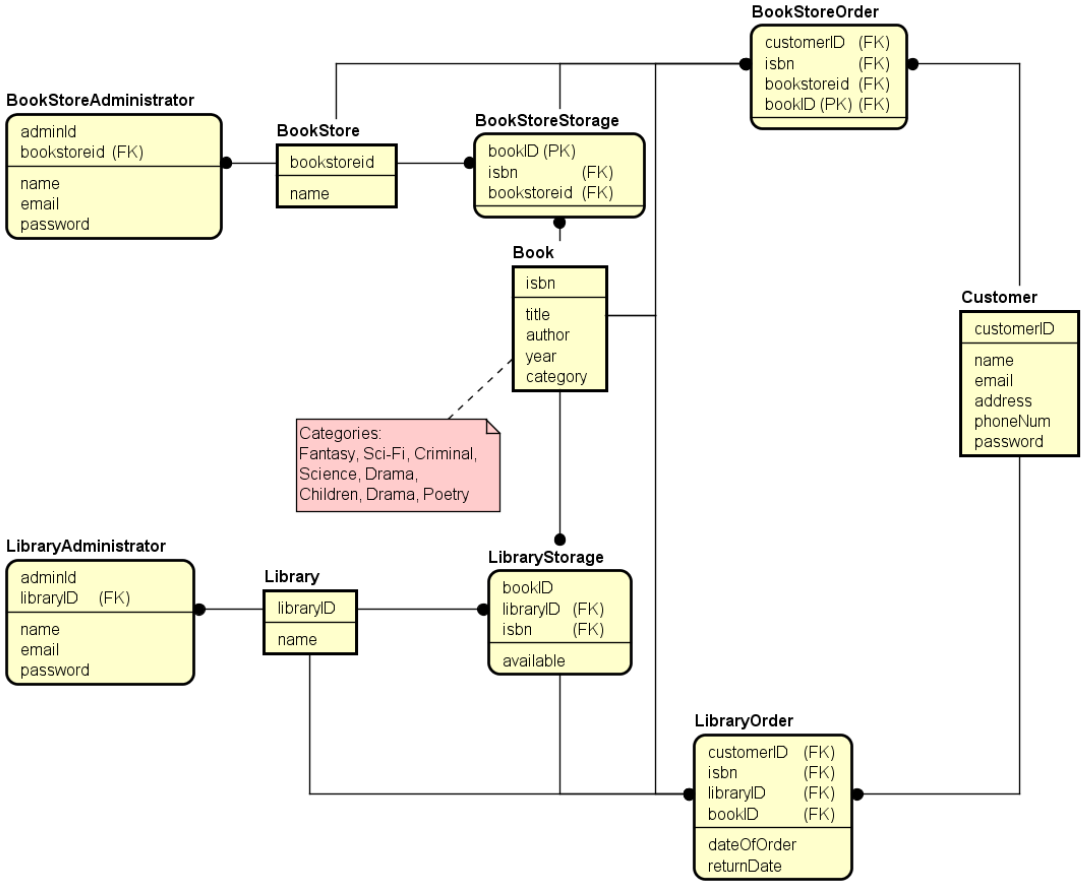
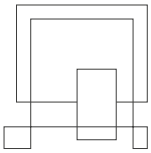

BookStoreOrder
customerID (FK)
isbn (FK)
bookstoreid (FK)
bookID (PK) (FK)
BookStoreAdministrator
adminId
bookstoreid (FK)
name
email
password
BookStore
bookstoreid
name
BookStoreStorage
bookID (PK)
isbn (FK)
bookstoreid (FK)
Book
isbn
title
author
year
category
Customer
customerID
name
email
address
phoneNum
password
Categories:
Fantasy, Sci-Fi, Criminal,
Science, Drama,
Children, Drama, Poetry
LibraryAdministrator
adminId
libraryID (FK)
name
email
password
Library
libraryID
name
LibraryStorage
bookID
libraryID (FK)
isbn (FK)
available
LibraryOrder
customerID (FK)
isbn (FK)
libraryID (FK)
bookID (FK)
dateOfOrder
returnDate

Figure 4



4 Design

Having done analysis, one can proceed with designing the system. This is the part where specific classes are distinguished and relations between them are being specified. In this part of the working process, the analysis of the problem statements and requirements are converted into an overview of the whole system. This is the last step before formulating the code.

4.1 Architecture

The base of a successful system is a well-designed architecture, as it is a skeleton for all features of the system. The 3-Tier architecture pattern has been chosen, as the system is a distributed system. This pattern allows dividing the responsibilities in the system to different components. Another advantage of choosing this pattern is easy scalability. There are 5 components of the system. Starting from the 1st tier, which is responsible for interacting with the user, the website has been chosen. On this tier, to extend the system, a mobile application can easily be created, basing on the 2nd tier. The topic of this choice is developed in the Website section of design. The 2nd tier focuses on the business logic of the system. There are 3 components on this tier. BookStore and Library are the services responsible for the actions taken by the Administrators of those institutions. The BookService is a service that takes care of the general functionalities (e.x. searching for books, Customers registrations, making orders for books). Moreover, the system can be extended by adding more Library and BookStore components, representing real institutions. Lastly, the 3rd tier is responsible for storing to and accessing data from the database. The component that is responsible for this is the DBServer. The architecture of the system is presented on the diagram below. (Figure 5)

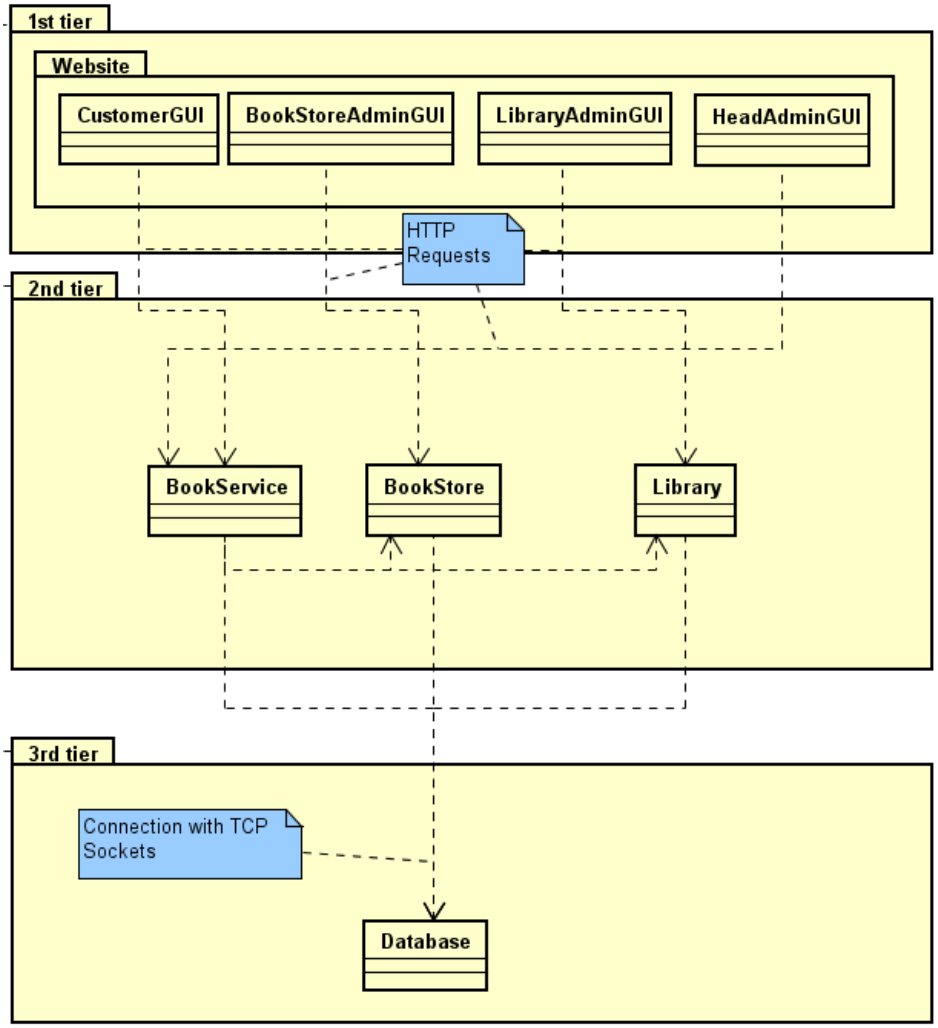
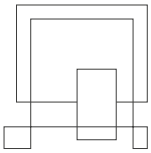


Figure 5

4.2 Book Service and Bookstore Service

The book service consists of three main parts: Requests controllers (API), Model and Controller. The model is an independent part which contains classes describing the business domain regarding functionalities of the customer (in the case of Book service) and orders (in the case of Bookstore). The Controller represents a bridge between requests and other components in the system. The request package in Book service consists of requests such as Make order, Create customer, Book details, Login in and



Search. Book store is handling Order request and Search which is identical as in the book service.

4.3 Library Service

As other services, the Library service also consists of three main parts such as Requests controllers(API), Model and LibraryControllers.

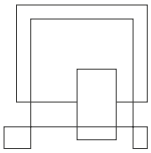
Requests coming to the Library service are handled by 4 controllers in the following way:

1. **SearchController** - handles two GET requests (Search, AdvancedSearch) by passing optional parameters from the request to the LibraryController class. Moreover, it handles another GET request (BookDetails). In this case it receives the “isbn” parameter from the URI and passes it to the LibraryController class.
2. **BooksController** - is responsible for two methods regarding books, which are add (POST) and delete (DELETE) book.
3. **OrdersController** - handles GET request (GetOrders) and DELETE request (ReturnBook)
4. **LogOutController** - handles only one DELETE request LogOut that clears local cache and also sends DELETE request to the BookService to clear its own cache.

The Models namespace contains the basic Book class and the Category enum that holds all possible book categories in the system. Other classes in the Library model namespace are not needed due to the fact that the Library is a service that only consumes requests from the Library administrators and forwards them through Controllers Connections namespace to the DBServer component that is operating in the 3rd tier.

The Controllers namespace consists of the LibraryController singleton class, SessionKeyManager class and the namespace Connections.

The LibraryController singleton is a bridge between Requests (API) and Connections namespace, which is responsible for Socket connection to the DBServer. In the same namespace there is located another important class called SessionKeyManager. As the Library service can be contacted only by library administrators, all of the requests



are sent through the authorization process that is handled in the SessionKeyManager. If the received session key from the request is not present in the local cache of the Library, the HTTP request is sent from the SessionKeyManager to the BookService in order to get the expiration date from the original cache storage. Afterwards, it is needed to check the expirationDate and perform the request if the session key has been evaluated as valid.

The Connections namespace contains one main interface IDatabaseProxy that provides all possible functions that can be called on DBServer through a Socket connection. The detailed analysis of the Socket connection between DBServer and Services can be found in the section 4.7 - Communication.

What is more, in the Library service can also be found the Resources namespace. In this namespace located is the ConfigurationLoader singleton class that is responsible for loading data from the configuration text file. This data is needed to either open a socket connection to the DBServer (host and port) or make a HTTP request to the BookService(url). Moreover, the specific id for the Library service is stored in this configuration file. The whole class diagram of the Library can be found in Appendix C.

4.4 DBServer – Database access

Database entities are accessed by Repositories where each entity has its own Repository. The diagram in Figure 6 shows connections and dependencies between the repositories. The base of the structure are the repositories responsible for CRUD on basic entities like Book, Customer, Library and BookStore. Those repositories are marked with the blue color. Higher in hierarchy are repositories for managing storages in Library and BookStore and Orders, marked with orange. Beside the CRUD operations, they are also exposing methods for searching for books in particular institutions, getting storages by isbn and institution's ID. BooksStorageRepo, marked with the green color, has the highest place in the hierarchy. It is responsible for retrieving information from other repositories and combining them. It is used to get the overall view of the book in the system (i.e BookDetails). Repository manager is a facade to the repositories for the Controller.

Each of the repositories has its own interface. That gives a possibility to change the implementation of the specific repository without interfering with other repositories.

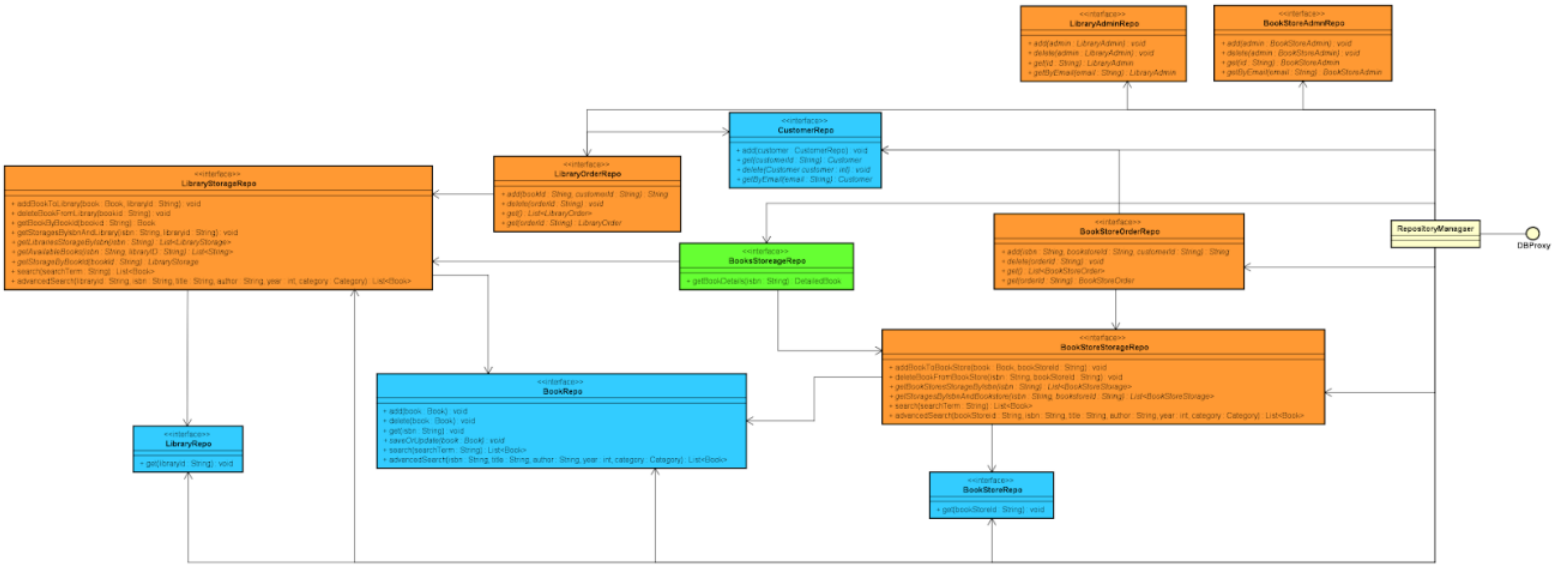


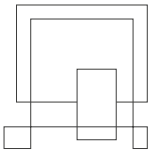
Figure 6

[Figure 6 - full size image can be found in Appendix C]

4.5 GUI

The goal of the First Tier architecture (GUI) in the system is to provide means for the user to be able to interact with the system's underlying business logic hidden in the Second Tiers' servers. The GUI is to be a place where no business logic will be implemented, but where the data will begin it's flow through the system's architecture. In other words, the GUI is to query the business logic servers with data, and display the data to the user.

As stated in Analysis, the choice has been made to use website for above purposes, but similar design could refer also for other GUI choices. The term website and GUI will be used interchangeably further below.



4.5.1 Basic Principles

The GUI should “not know” how the internal business logic works. All the internal server operations should be entirely disassociated with data and logic contained in the GUI structure. This not only simplifies the system’s architecture - but also provides more security to the system by giving the attackers no easy means to manipulate any data. One may find it helpful to think of this first-second-tier relationship as if where the second tier behaves as a “black box” for the first tier. The latter one can send data or ask for some data contained in the “black box”, but does not know how this data is generated or used inside it.
As with most websites, the decision has been made to use HTTP protocol requests to fetch data from/to 2nd Tier servers.

4.5.2 Technologies

Frontend developers environment gives a variety of tools for developers to use to make websites look and work well. The decision was made to use React Javascript framework as a primary tool for making the website modern, up to date with today’s market as well as easy to maintain in the future.

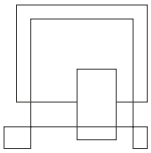
4.6 Security

This section contains descriptions of proactive and reactive mechanisms designed in order to prevent or react to threats stated in security analysis section 3.3 - Security.

4.6.1 Proactive mechanisms

4.6.1.1 Authentication protocol for Book service

A user who has an account in the database may log in to the system. When a customer or administrator wants to log in to the system, he/she must fill a login form with his/her email and password.



This information is passed to the database to authenticate the user. If the information is correct, the database returns the response which contains url and the user type. Otherwise it returns an error message. Then the book service generates a session key which is kept in the memory of the bookservice. The session key's life time is one hour and after that time the user will not have access to the system's functionalities and he/she has to log in again.

When the BookService completes the authentication, it sends the session key, url and user type to the user. The user now may use the system's functionalities depending on the user's type. The whole process is shown in Figure 7.

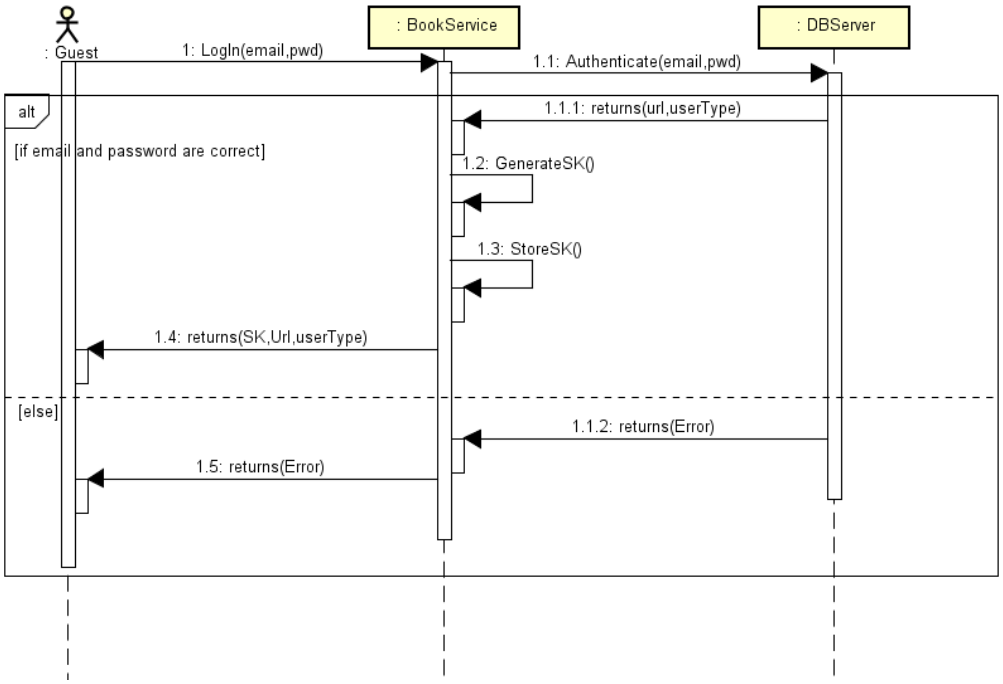
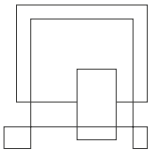


Figure 7

4.6.1.2 Authorization protocol for Library

When the LibraryAdmin makes a request to the Library service, he/she passes his/her own session key within the request as a cookie. Firstly, the Library looks for the



session key in its own cache. If passed key is not in the cache, then the Library sends a HTTP request to the BookService in order to get the expiration date of the given session key. The BookService can respond either with the expiration date if given session key exists in the BookService cache or with an error message (“Unauthorized”) if there is no such session key in the BookService cache.

After the Library receives the expiration date from the BookService, it stores it in its own cache. Then the Library checks the expiration date for the given session key and if the current time is before the expiration date than the user (LibraryAdmin) is authorized and his/her request can be processed in the Library. Otherwise, the Library deletes the session key with the expired expiration date from its own cache and the user gets an “Unauthorized” response. The whole process can be seen in Figure 8.

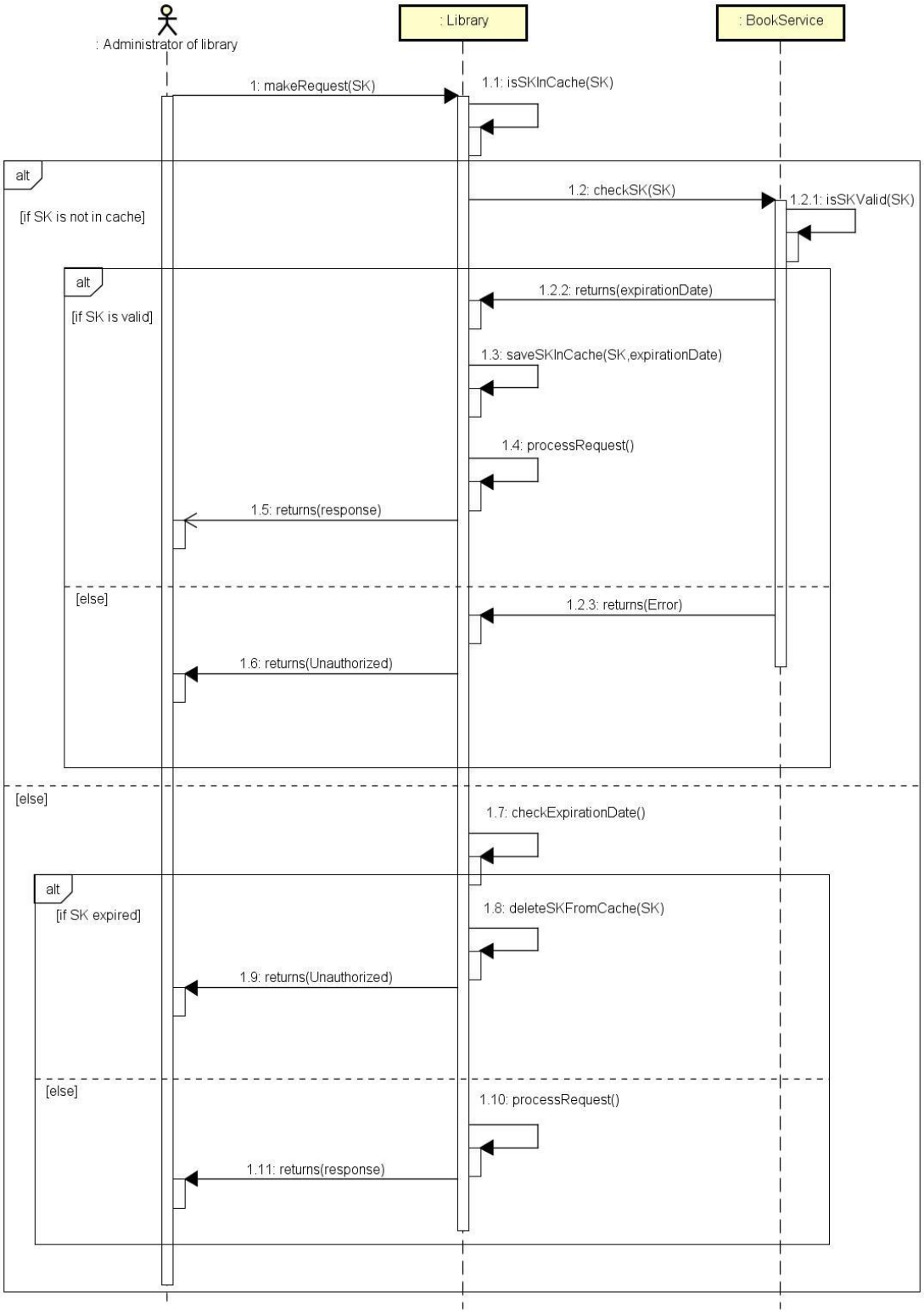
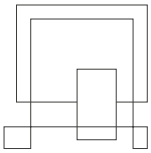

: Administrator of library
: Library
: BookService
1: makeRequest(SK)
1.1: isSKInCache(SK)
alt
[if SK is not in cache]
1.2: checkSK(SK)
1.2.1: isSKValid(SK)
alt
[if SK is valid]
1.2.2: returns(expirationDate)
1.3: saveSKInCache(SK,expirationDate)
1.4: processRequest()
1.5: returns(response)
[else]
1.2.3: returns(Error)
1.6: returns(Unauthorized)
[else]
1.7: checkExpirationDate()
alt
[if SK expired]
1.8: deleteSKFromCache(SK)
1.9: returns(Unauthorized)
[else]
1.10: processRequest()
1.11: returns(response)

Figure 8



4.6.1.3 Password encryption

During creating a new user, the password is encrypted using the SHA-256 hash function. In that form it is later stored in the database. When the user wants to log in, the password that he/she has entered is also encrypted with the same method. To check if the password is correct the two hashes are compared. The power of the attacker with breaking the password depends on what he/she already has access to. If the attacker does not have access to any part of the system, then his/her power is only the brute force attack, as the encryption happens on the server's site. He/She does not know the ciphertext. If the password from the database would have leaked, then the attacker could know the ciphertext, but the method that has been used to encrypt the password is still unknown. In this situation the power of the attacker is the Ciphertext-Only Attack.

4.6.1.4 SSL

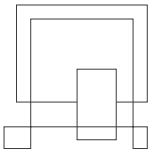
For protecting the information and having a secure connection from a web server to a browser, a SSL protocol will be used. The SSL is encrypting messages and connection so that the third party cannot see the communication between the client and the server. Thanks to using nonces and sequence numbers the reply attacks and man in the middle attacks are prevented.

4.6.1.5 Backup

In order to protect data from loss, the system makes database backup each hour by the automated script on the database server machine and sends it to a safe place.

4.6.1.6 Digital signature

To prevent spoofing identity for each request which needs a proof of authorization the user will need to sing requests with a digital signature. For example, when a customer is ordering a book, or when an administrator is confirming an order.



4.6.1.7 Frequent changes of password

In order to prevent stealing identity. The system will ask for a new password each week. The password must contain numbers and has to be at least 8 characters long.

4.6.2 Reactive mechanisms

4.6.2.1 Restore backup

This solution restores the system to the state before the attack after an unauthorized access to the database or the admin's accounts.

4.6.2.2 Change password

The change of password secures the account that has been attacked.

4.6.2.3 Change to new server provider / buy own server

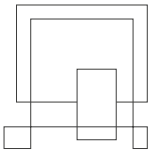
In order to prevent further security problems concerning having a server owned by a third party, the dedicated server should be bought. That would allow controlling and configuring the access to the server machine.

4.6.3 Conclusion

After analyzing threats and calculating the risk, following solutions have been used in the system. The SSL has been added to all services to secure the connection with the website. To authorize the user during the session, the Session Key checking mechanism has been implemented. To secure the password, the SHA-256 hash function has been used.

4.7 Communication

As the system is distributed and consists of many components, in order to connect their functionalities, there must be a communication amongst them. There are two types of



communication technologies used in the system. Between the 1st and the 2nd tier the website is making HTTP requests which are handled by REST requests controllers in the 2nd tier components.

The communication with the Database Server is accomplished by the TCP sockets connection. Services application and the Database Server are exchanging messages in the form of JSON strings containing a Request or a Response.

The Request message has two attributes, operation and arguments. The operation describes the type of the request and arguments supplies necessary information for processing the request by the database.

The response message has two attributes, status and content. The status describes if the requested operation has ended with success or an error occurred. The content of the response contains the result of the request. If the request operation is not recognized by the Database Server, it returns a response with an Error status and content “Wrong operation”. Examples of the messages and their JSONs can be found in Appendix E.

However, the base of the communication is the communication protocol. In the case of this system, the protocol is very simple and works in a request-response manner. The protocol has been represented in the diagram below in the example of searching for a book request. (Figure 9)

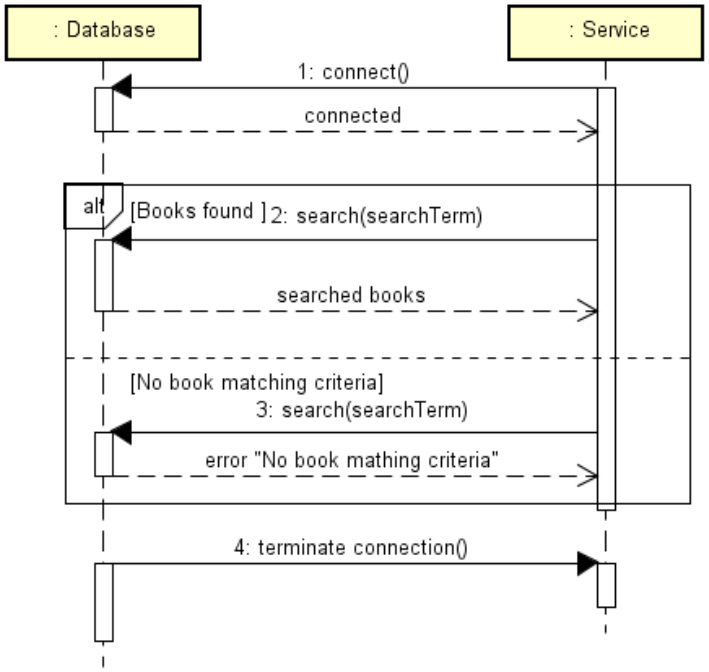
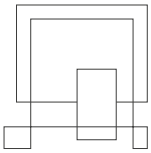

: Database
: Service
1: connect()
connected
alt
[Books found]
2: search(searchTerm)
searched books
[No book matching criteria]
3: search(searchTerm)
error "No book mathing criteria"
4: terminate connection()

Figure 9



5 Implementation

Basing on analysis and design code can be formulated. The following part shows the most interesting parts of the implementation. However, the whole source code can be found in Appendix F.

5.1 Authentication

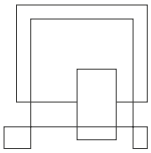
One of the security mechanisms implemented in the system is session key validation. This approach requires sending a session key, acquired during login, with requests that should be accessible only by a specific type of users of the system.

For generating and controlling the session keys responsible is the SessionKeyManager. The generateSK() method generates a new session key using UUID and together with its expiration date it puts it to the HashMap. (Figure 10)

```
public static String generateSK() {
    String sessionKey = UUID.randomUUID().toString();
    Calendar expirationDate = GregorianCalendar.getInstance(TimeZone.getDefault());
    expirationDate.add(Calendar.HOUR, amount: 1);
    sessionKeys.put(sessionKey, expirationDate);
    return sessionKey;
}
```

Figure 10

The method checkSessionKey() takes a session key in parameter and checks if the date of the session key is before the expiration day. If the session key's expiration time has expired the method will throw a SessionKeyIsNotValidExeption and the user's activity will be stopped. (Figure 11)



```
public static void checkSessionKey(String sessionKey) throws SessionKeyIsNotValidException {
    Calendar expDate = sessionKeys.get(sessionKey);
    if (expDate == null) {
        throw new SessionKeyIsNotValidException("The session key is not valid");
    }
    Calendar now = GregorianCalendar.getInstance(TimeZone.getDefault());
    System.out.println(now.getTime());
    if (!now.before(expDate))
        throw new SessionKeyIsNotValidException("The session key is not valid");
}
```

Figure 11

Before each request the checkSessionKey() method is called in order to validate the session key. If the session key is not out of its expiration date, the request is executed. Otherwise, an exception is thrown and it is caught by an advice class (advice classes are described in detail in section 5.2 - BookService and Bookstore - search) and the Unauthorized status is returned. (Figure 12)

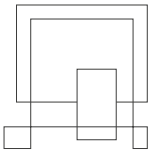
```
@RequestMapping("/search")
public List<Book> search(@RequestParam(value = "searchTerm") String searchTerm, @CookieValue("sessionKey") String sessionKey) {
    SessionKeyManager.checkSessionKey(sessionKey);

    Controller controller=context.getBean(Controller.class);
    return controller.search(searchTerm);
}
```

Figure 12

5.2 BookService and Bookstore - search

Those components have been written in Java and are REST web services which are handling requests from the website. The technology that is used to implement the REST web service in Java is the Spring MVC Framework. This solution has been chosen over Java EE because the Spring Framework comes in a Spring Boot Framework which is providing also other functionalities like Dependency Injection that has been used in the project. It is also easier to use than Java EE. Another advantage is that the Spring Framework is currently the most popular web services framework used in Java and learning it will benefit the team members in the future.

The implementation of the BookService and BookStore is in many places similar, as they are using the same technology and as there is not much business logic in the



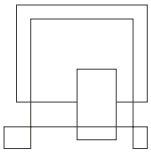
system. That is why processing a request looks similar. A request is received by the RestController and then passed to the Controller. The Controller sends the request to the DBServer through the DatabaseConnection and returns the response depending on the DBServer's response.

The search Request in the BookStore will be used as an example. (Figure 13)

```
@RestController
public class Search{

    private SessionKeyManager sessionKeyManager;
    private Controller controller;

    @Autowired
    public Search(SessionKeyManager sessionKeyManager, Controller controller) {
        this.sessionKeyManager = sessionKeyManager;
        this.controller = controller;
    }

    @RequestMapping("/search")
    public List<Book> search(@RequestParam(value = "searchTerm") String searchTerm,
                             @CookieValue("sessionKey") String sessionKey)
            throws SessionKeyManager.SessionKeyInvalidException {
        sessionKeyManager.isSessionKeyValid(sessionKey);

        return controller.search(searchTerm);
    }
```

Figure 13

The class with requests, Search, is marked with annotation @RestController, which allows the Spring Framework to map it with requests. The constructor is marked with @Autowired annotation. That means that the parameters should be injected by Dependency Injection and Spring will take care of them. The method search(), that is a request handler, is marked with @RequestMapping annotation and the parameter for annotation is a route to the request. The parameters of the method are marked with two more annotation. The @RequestParam is specifying that this parameter should be in the URL with name searchTerm. The session key is passed in the cookies, so it is



marked with @CookieValue annotation. The first action is checking if the session key is valid. If not, an exception is thrown and caught by the exception advice and the 401 (Unauthorized) status code is returned. The implementation of the advice class can be seen in the Figure 14. After ensuring that the session is valid, the searchTerm is passed to the Controller which processes the request and prepares the response. All responces and requests to BookService and BookStore can be found in Appendix H.

```
@ControllerAdvice
public class SessionKeyExceptionAdvice {

    @ResponseBody
    @ExceptionHandler(SessionKeyManager.SessionKeyIsNotValidException.class)
    @ResponseStatus(HttpStatus.UNAUTHORIZED)
    String seesionInvalidHandler(SessionKeyManager.SessionKeyIsNotValidException ex) {
        return ex.getMessage();
    }
}
```

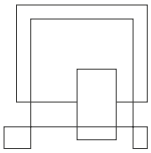
Figure 14

5.3 Library - Orders

The ASP.NET Web API is used as a framework to handle HTTP requests to the Library service. All requests and responses to the Library can be found in Appendix H. The implementation of the flow through the LibraryService on getting the GetOrders request can be found below.

As it is stated in the design section 4.3 - Library Service, there are 4 request controllers in the Library service. The OrdersController is the one responsible for handling requests regarding orders.

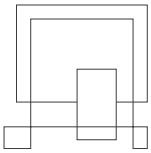
As the Library service is supposed to be reachable only by the corresponding admin, all requests on this service need to be authorized. In Figure 15 it can be seen that the GET request is handled by the GetOrders() method. The first step is checking the session key from the request. This is handled in the CheckSessionKey private method that gets cookies from requests and the SessionKeyManager is used in a static way in order to validate the session key. Only if the session key is validated then the request is processed and passed further to the _libraryController and then to the



DatabaseProxy class where the corresponding request is sent to the DBServer through the socket connection. Otherwise, if the session key is not validated, an Unauthorized response with a status code 401 is sent back to the consumer of the Library service.

```
private bool CheckSessionKey()
{
    var sessionKeyFromClient = Request.Cookies["sessionKey"];
    try
    {
        if (sessionKeyFromClient != null &&
            SessionKeyManager.IsSkValid(sessionKeyFromClient))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (SessionKeyInvalidException e)
    {
        return false;
    }
}

// GET orders
[HttpGet]
[Route("orders")]
public ActionResult<string> GetOrders()
{
    if (CheckSessionKey())
    {
        return _libraryController.GetOrders();
    }
    else
    {
        return Unauthorized();
    }
}
```

Figure 15

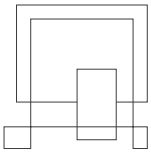


In Figure 16 can be observed how the authorization process is implemented in the Library service, more precisely in the SessionKeyManager class.

```
public class SessionKeyManager
{
    private static readonly string Url = ConfigurationLoader.GetInstance().BookServiceURL;
    private static Dictionary<string, DateTime?> _sessionKeys = new Dictionary<string, DateTime?>();
    private static readonly string LIBRARY_ID = ConfigurationLoader.GetInstance().LibraryID;

    public static bool IsSkValid(string sessionKey)
    {
        DateTime? expirationDate;
        try
        {
             expirationDate = _sessionKeys[sessionKey];
        }catch(Exception e)
        {
            expirationDate = CheckInBookService(sessionKey);
            _sessionKeys.Add(sessionKey, expirationDate);
        }
        return Nullable.Compare(expirationDate, DateTime.Now) > 0;
    }

    private static DateTime CheckInBookService(string sessionKey)
    {
        try
        {
            var response = MakeRequest(Url + "checkSK/" + sessionKey + "/" + LIBRARY_ID,null);
            var date = DateTime.ParseExact(response, "yyyy MMM dd HH:mm:ss", null);
            return date;
        }
        catch (ArgumentNullException ae)
        {
            Console.Write(ae);
            throw new SessionKeyInvalidException("Session can not be authenticated");
        }
        catch (FormatException fe)
        {
            Console.Write(fe);
            throw new SessionKeyInvalidException("Invalid expiration data format");
        }
    }
```

Figure 16

From each of the request controllers, the static method IsSKValid(sessionKey) is called before processing the request. In this method, firstly is checked whether the desired session key is present in the library local cache - in this case it is a Dictionary<string,DateTime> field that stores session keys of all library admin accounts that are using this library currently and the session key's expiration date as a value in this Dictionary. The situation of no data corresponding to the session key in the library local cache, is handled in the catch block by calling the CheckInBookService(sessionKey) private static method. In this method



MakeRequest(url,cookie) method is called from where the HTTP request is sent to the BookService. After that the response containing the expiration date is parsed and returned. Finally, the expiration date is checked in IsSKValid method and a boolean expression is returned back to the request controller.

5.4 DBServer – accessing the database

DBServer is written in Java. Hibernate ORM framework has been used for database manipulating and quarrying.

The class responsible for starting the Hibernate connection with the database is the HibernateAdapter. This class is also providing some CRUD operations in the database for Repositories, because adding, updating and deleting is implemented in the same way for every kind of object. The method addObject() is shown in Figure 17. In order to add an object to DB using Hibernate a session must be open. The next step is to open a transaction and perform a save. The last step is to commit the transaction.

```
public static void addObject(Object obj) {
    Transaction tx = null;
    try (Session session = sessionFactory.openSession()) {
        tx = session.beginTransaction();
        session.save(obj);
        tx.commit();
    } catch (HibernateException e) {
        if (tx != null) tx.rollback();
        e.printStackTrace();
    }
}
```

Figure 17

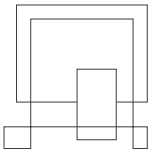
For more advanced queries, a custom query must be created. One of such queries is to get a BookStoreStorage by Book isbn and BookStore ID. The process is presented in Figure 18. First, the session is opened and the transaction has begun. Then, the

query is created with the use of Hibernate Query Language. To use additional parameters in the query, the setParameter method is used, which associates a parameter in the query with a variable. The tolist() method executes the query and the result list is retrieved. Then the result is returned.

```
@Override
public List<BookStoreStorage> getStoragesByIsbnAndBookstore(String isbn, String bookstoreId) {
    Transaction tx = null;
    try (Session session = sessionFactory.openSession()) {
        tx = session.beginTransaction();
        List<BookStoreStorage> storages = session.createQuery( s: "FROM BookStoreStorage as s where " +
                "s.book.isbn like :isbn and " +
                "s.bookstore.bookstoreid like :bookstoreid")
                .setParameter( s: "isbn", isbn)
                .setParameter( s: "bookstoreid", bookstoreId)
                .list();
        tx.commit();
        return storages;
    } catch (HibernateException e) {
        if (tx != null) tx.rollback();
        e.printStackTrace();
    }
    return new LinkedList<>();
}
```

Figure 18

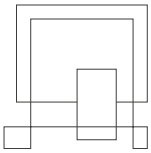
In order to allow Hibernate to map the desired class to the corresponding table, JPA annotations has been used. @Entity annotation marks the class as an Entity to be managed by Hibernate. @Table specifies the table's name. @Column annotation maps the field to a column. The example for the Book class is shown in Figure 19.



```java
@Indexed
@Entity
@Table(name = "books", schema = "public")
public class Book implements Serializable {

    @Field(name = "bookIsbn")
    @Id
    @Column(name = "isbn")
    private String isbn;

    @Field
    @Column(name = "title")
    private String title;

    @Field
    @Column(name = "author")
    private String author;

    @Field
    @FieldBridge(impl = IntegerBridge.class)
    @Column(name = "year")
    private int year;

    @Field(bridge=@FieldBridge(impl=EnumBridge.class))
    @Column(name = "category")
    @Enumerated(EnumType.STRING)
    private Category category;
```

Figure 19

Another functionality of the DBServer is providing searching for books. It is accomplished by the Hibernate Search library using the Apache Lucene index. By



adding annotations @Indexed to the class it is marked to be indexed by Lucene. That allows writing a special kind of search queries that perform full text searching on specified fields, marked with @Field annotations. The example of such query is presented in Figures 20, 21 and 22.

```
public static List executeQuery(String searchTerm, Class classObj, String... fields) {
    SessionFactory sessionFactory = HibernateAdapter.getSessionFactory();
    Session session = sessionFactory.getCurrentSession();
    Transaction transaction = session.beginTransaction();
    EntityManager em = session.getEntityManagerFactory().createEntityManager();
    FullTextEntityManager fullTextEntityManager =
            org.hibernate.search.jpa.Search.getFullTextEntityManager(em);
    em.getTransaction().begin();
```

Figure 20

To perform a search, first a session must be opened and the FullTextEnittyManager object must be created. The next step is to open a transaction.

```
QueryBuilder qb = fullTextEntityManager.getSearchFactory()
        .buildQueryBuilder().forEntity(classObj).get();
org.apache.lucene.search.Query luceneQuery = qb.keyword() TermContext
        .onFields(fields) TermMatchingContext
        .ignoreFieldBridge() TermMatchingContext
        .matching(searchTerm) TermTermination
        .createQuery();
Lucene query in a javax.persistence.Query
javax.persistence.Query jpaQuery =
        fullTextEntityManager.createFullTextQuery(luceneQuery, classObj);
```

Figure 21

Then, using the QueryBuilder, the luceneQuery is created with proper configuration. The Class to be searched in must be specified and fields as well as the searchTerm must be provided. After that, the query is wrapped into the JPA query. The last step is to execute the query, commit the transaction and return the result.

```
// execute search
        List result = jpaQuery.getResultList();
        em.getTransaction().commit();
        em.close();
        System.out.println(result);
        transaction.commit();
        return result;
    }
```

Figure 22

5.5 Website

This part focuses on how the website was implemented, how the queries were handled and how the React components were structured.

The basic structure of the React components looks like in the diagram shown below. (Figure 23)

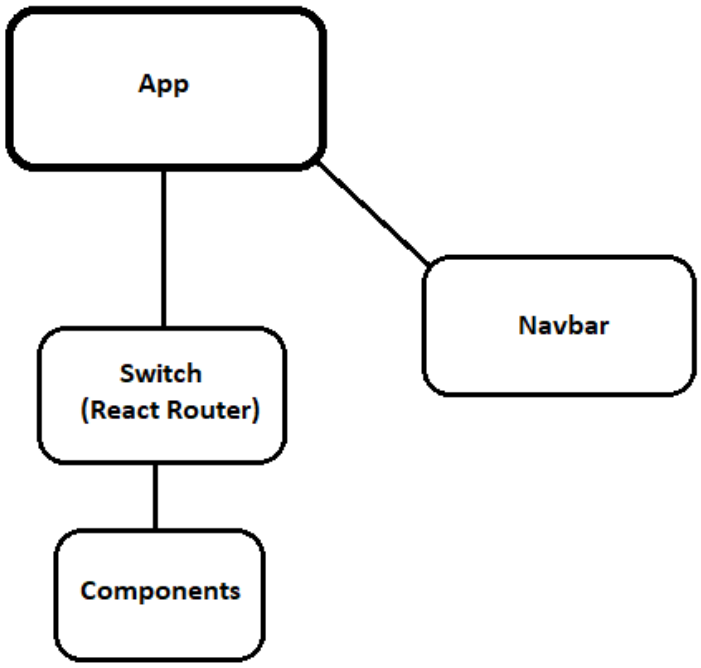
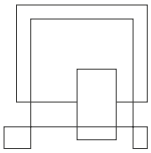


Figure 23

The App component encapsulates both the React Router components as well as the Navbar separately, as the Navbar is always displayed on the page. The React Router serves the role of changing the ‘root’ of the page regarding the specified URL. This also provides a static-page functionality as the page will not reload the whole window, but just the React Router component to display various data.



```
render() {
  return (
    <BrowserRouter>
      <div>
        <Navbar
          loggedIn={this.state.loggedIn}
          name={this.state.name}
          accountType={this.state.accountType}
        />

        <Switch>
          <Route exact path="/" component={Home} />
          <Route path="/books" component={BookList} />
          <Route
            path="/search/:search_term"
            customerId={this.state.customerId}
            render={props => <BookList customerId={this.state.customerId} />}
          />
          <Route
            path="/advancedSearch/:title?/:author?/:year?/:isbn?/:category?"
            customerId={this.state.customerId}
            render={props => (
              <BookList {...props} customerId={this.state.customerId} />
            )}
          />
          <Route path="/bookstore_orders" component={BookstoreOrders} />
          <Route path="/library_orders" component={LibraryOrders} />
          <Route path="/registration" component={Registration} />
```

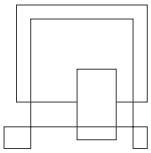
Figure 24

The above code snippet (Figure 24) shows how indeed Routes are defined inside the Switch component, leaving the Navbar outside.

The App component also stores the vital data for the website like:

- cookies
- data of the user

These data are then passed to different components on call and used in queries to the APIs (2nd Tier servers), which will be covered further.



```
class App extends Component {
  state = {
    name: "",
    loggedIn: false,
    accountType: "",
    customerId: ""
  };

  handleLogIn = (name, accountType, sessionKey, customerId) => {
    this.setState({ name: name });
    this.setState({ accountType: accountType });
    this.setState({ loggedIn: true });
    Cookies.set("sessionKey", sessionKey + "");
    this.setState({ customerId: customerId });
  };
```

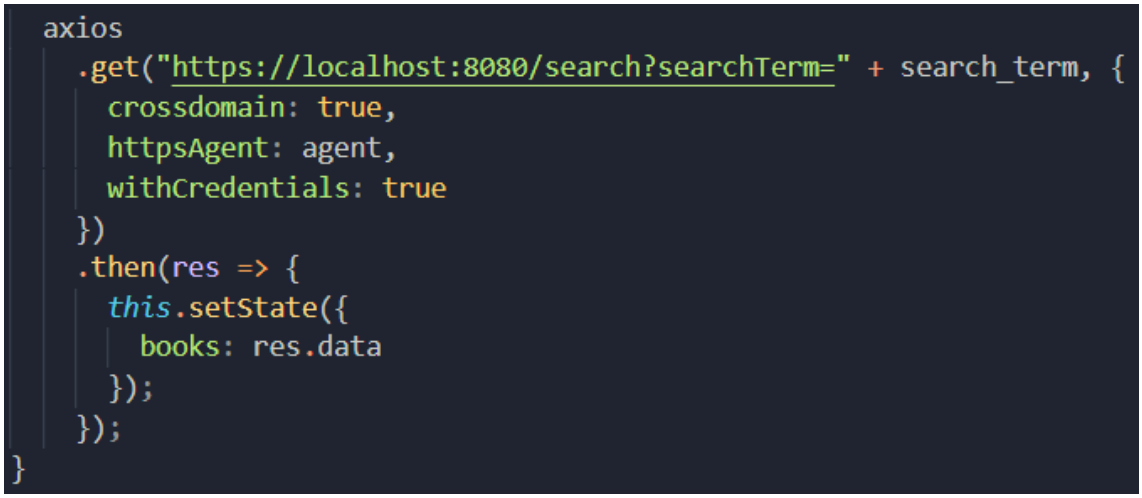
Figure 25

The above code snippet (Figure 25) shows how the state of the data is changed on the call from *handleLogIn* function.

This architecture simplifies the structure of the data saved on the website, as all of the data are stored in one place, namely the *App* component.

Different queries in the website are handled in different places, before loading the page, after loading the page, but also are triggered by the user by for example clicking a button.

Consuming the API was done using a library called *axios*, which simplified and improved the quarrying process.

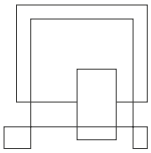


```
  axios
    .get("https://localhost:8080/search?searchTerm=" + search_term, {
      crossdomain: true,
      httpsAgent: agent,
      withCredentials: true
    })
    .then(res => {
      this.setState({
        books: res.data
      });
    });
}
```

Figure 26

The above snippet (Figure 26) shows how axios does the query, and how it can update the state of a component. Axios sends a https request to the URI with CORS header enabled (crossdomain: true), and also using cookies stored in the App component, by passing withCredentials option. Then if the response is received - it can work with the acquired data and in this case change the state of the component.

After the state is changed, the corresponding functions are called, so that different data is displayed to the user. This is shown in the code snippet below (Figure 27), where in this particular example, on each change of the book's state, the mapping function is called and the layout is changed.

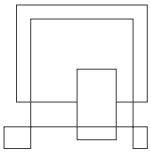


```
const { books } = this.state;
const booksList =
  this.state.books.length > 0 ? (
    books.map(b => {
      return (
        <div key={b.isbn} className="card">
          <div className="card-body">
            <h5 className="card-title">
              <Link to={"/details/" + b.isbn}>{b.title}</Link>
            </h5>

            <div className="card-subtitle text-muted">
              {b.author} ({b.year}) /{" "}
              <span className=" text-danger">{b.category}</span>
            </div>
            <p />
          </div>
        </div>
      );
    })
```

Figure 27

5.6 Communication

As stated in design in the section 4.7 - Communication, the communication with DBServer is accomplished by TCP sockets. This requires classes which are receiving and sending messages. On the DBServer side, the Server side is responsible for listening for client connection and passing the request to the Controller, which is processing the request. When the Response is ready, it is sent back to the client. When the connection is established, the client socket is passed to the HandleClient class, which in a new thread receives a message and after the request is processed, sends back the response. The implementation is shown in the Figure 28.

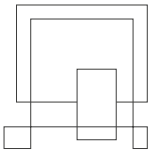


```java
@Override
public void run() {
    String clientIP;
    String response, request;
    try (BufferedWriter out = new BufferedWriter(new OutputStreamWriter(client.getOutputStream()));
         BufferedReader in = new BufferedReader(new InputStreamReader(client.getInputStream()))) {

        clientIP = client.getInetAddress().getHostAddress();
        //without ready reading is not working properly
        in.ready();
        request = in.readLine();
        System.out.println("Request from " + clientIP + ": " + request);

        response = controller.handleRequest(request);
        out.write( str: response + "\n");
        out.flush();
        System.out.println("Response to " + clientIP +": " + response);

    } catch (IOException e) {...}
}
```

Figure 28

On the other side of the connection can be the BookService, the BookStore or the Library and the class responsible for connection is the DatabaseConnection. In all those components it is implemented similarly. It is implementing the DatabaseProxy interface, which contains methods that are corresponding to the available requests to the DBServer.

The method that is responsible for sending and getting messages to and from DBServer is sendMessage() method. It is opening the connection with the DBServer, sending the message and receiving the response. If the connection could not be established, it throws a ServerOfflineException which is cough in the higher parts of the system. The implementation is shown in Figure 29.



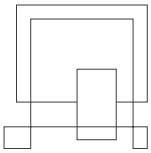
```
private String sendMessage(Request request) throws ServerOfflineException {
    try {
        Socket server = openConnection();

        try (BufferedWriter out = new BufferedWriter(new OutputStreamWriter(server.getOutputStream()));
             BufferedReader in = new BufferedReader(new InputStreamReader(server.getInputStream()))) {

            System.out.println("Sending request to db: " + request);
            out.write( str: request.toJson() + "\n");
            out.flush();
            in.ready();
            return in.readLine();
        } catch (IOException e) {...}
    } catch (IOException e) {
        throw new ServerOfflineException("Could not connect to server");
    }
}
```

Figure 29

As the messages are sent in the JSON Strings format, they must be deserialized and translated to objects. The getResponseStatus and getContent methods are responsible for deserializing the two parts of the response. An interesting method is getContent (Figure 30). As the content can be in many forms and represent many formats, the method is generic and is able to deserialize any object. It is accomplished by the ose of the Gson library for serializing and deserializing JSON. By Type object and passing it to the fromJson method of gson object, the JSON is properly deserialized into the required object. The type T of the method is specified during the method call. In the case of the library, as it is written in C#, the NewtonSoft Json library has been used.

```
private static <T> T getContent(String contentJson) {
    JsonParser parser = new JsonParser();
    JsonElement element = parser.parse(contentJson);
    JsonObject obj = element.getAsJsonObject(); //since you know it's a JsonObject
    String content = obj.get("content").toString();
    Type type = new TypeToken<T>() {}.getType();
    return gson.fromJson(content, type);
}
```

Figure 30



6 Test

Having implemented the system, one has to conduct tests. Tests are needed not only to check whether the system is working and has been implemented correctly, but also to ensure that all requirements have been fulfilled and the system provides the end user with all the desired functionalities. Bearing that in mind, two kinds of tests have been executed: whitebox tests, in this case unit tests, and blackbox test, in this case tests following Requirement Test Descriptions. The results of the tests are shown further in this paragraph.

6.1 Unit test

In the system, unit test has been used to test fragile parts of the system in separation. They are also used to test the business logic. As the project does not have complicated business logic, the main target of the unit test was especially the DBServer and access to the database and searching mechanism. This is due to many problems with correct saving to, reading from, and searching in the database during the development. Moreover, that part of the system is very fragile for changes and must be monitored. To solve this problem, a suite of unit test have been made, to serve as a regression test. When a new bug has been discovered, after fixing it, a new test case has been written to detect that bug in the future. An example of unit test can be seen in Figure 31. The test is testing the Customer Repository and it is checking if looking for the Customer by his/her ID is without errors. Similar tests have been written to each method form repositories. The JUnit framework for unit test in Java and the XUnit framework for unit test in C# has been used for writing tests. Other parts have not been tested by unit test, as they do not have any logic and they are mostly calling other functions. The code of all unit test can be found in the source code (Appendix F, Folder Java\DBServer\src\test and Folder Library\Tests).

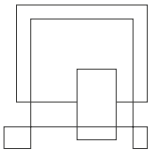
```
@Test
void getCustomerTest() {
    //setup
    Customer customer = new Customer( id: "id",  name: "name",  email: "email",  address: "address",  phoneNum: 111,  password: "password");
    try {
        customerRepo.add(customer);
    } catch (CustomerRepository.CustomerEmailException e) {
        fail("Customer can not be added, email already in use");
    }

    //test
    try {
        assertEquals(customer, customerRepo.get("id"));
    } catch (CustomerRepository.CustomerNotFoundException e) {
        fail("No customer");
    }

    //clenup
    customerRepo.delete(customer);
}
```

Figure 31

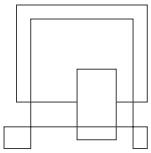
6.2 Requirement Test Descriptions

For this type of testing requirement test descriptions have been formulated, i.e. scenarios of what output the system should give after entering specific input. The descriptions and results are shown in the table below (Table 2).

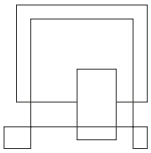
No	Requirement Test Description	Passed (✔/X)	If failed, what?
Login in to the account			
1	An email of an administrator of library with a matching password is entered and the login button is clicked. Afterwards, the system displays a main page.	✔	-
2	An email of an administrator of bookstore with a matching password is entered and the login button is clicked. Afterwards, the system displays a main page.	✔	-
3	An email of a customer with a matching password is entered and the login button is clicked. Afterwards, the system displays a main page.	✔	-



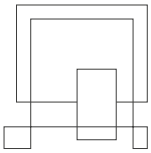
4	A wrong email is entered. The system displays a message: “wrong username or password, try again”.	✔	-
Search for a book			
5	A customer account is being used. Customer enters a term to the search bar. The system displays a list of books matching the search criteria. The customer clicks see details. The system displays detail page of selected book.	✔	-
6	A guest (user without account) enters to the page. Guest enters a term to the search bar. The system displays a list of books matching the search criteria. The guest clicks see details. The system displays detail page of selected book.	✔	-
7	A customer account is being used. Customer clicks button “advanced search”. Then he fills the searching fields. The system displays a list of books matching the search criteria. The customer clicks see details. The system displays detail page of selected book.	✔	-
8	A guest is on main page and clicks button “advanced search”. Then he fills the searching fields. The system displays a list of books matching the search criteria. The guest clicks see details. The system displays detail page of selected book.	✔	-
9	A customer account is being used. Customer enters a term to the search bar. If there is no book matching the search criteria, system displays “There is no book matching the search criteria”	✔	-
10	A guest is on the main page. Guest enters a term to the search bar. If there is no book matching the search criteria, system displays “There is no book matching the search criteria”	✔	-
Order a book			
11	A customer account is being used. Customer is on the book’s detail page where he choose the library. Then he clicks a borrow button. The system displays message “The order was made successfully”	✔	-



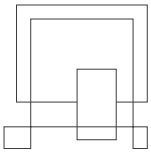
12	A customer account is being used. Customer is on the book's detail page where he choose the bookstore. Then he clicks a buy button. The system displays message "The order was made successfully"	✔	-
13	A customer account is being used. Customer is on the book's detail page. If the book is not available in the library, the system displays message "Not available" and there is no borrow button.	✔	-
14	A customer account is being used. Customer is on the book's detail page. If the book is not available in the bookstore, the system displays message "Not available" and there is no buy button.	✔	-
Create an account			
15	A guest enters the website and clicks Sign up button on the navbar. The system displays registration page with registration form. The guest fills all fields (name, email, address, phone number). Then he clicks confirm button. The system creates a new account and redirect to the previous page.	✔	-
16	A guest enters the website and clicks Sign up button on the navbar. The system displays registration page with registration form. The guest doesn't fill all fields and clicks button confirm. The system displays an alert "All fields must be filled"	✔	-
17	A guest enters the website and clicks Sign up button on the navbar. The system displays registration page with registration form. The guest uses an email which is already in the system and click confirm button. The system displays an alert "This email is already used"	✔	-
18	A guest enters to the website and clicks Sign up button on the navbar. The system displays registration page with registration form. The guest fills the phone number field with letters. The system displays an alert "The phone number cannot contain letter"	✔	-



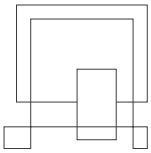
Manage books			
19	The administrator of library is on the main page and clicks button 'Home'. The system displays form for adding a new book. The administrator of library fills the form with title, author, year, isbn, category and then confirms it by pressing save button. The system displays a success alert.	✔	-
20	The administrator of bookstore is on the main page and clicks button 'Home'. The system displays form for adding a new book. The administrator of bookstore fills the form with title, author, year, isbn, category and then confirms it by pressing save button. The system displays a success alert.	✔	-
21	The administrator of library is on the main page and clicks button 'Home'. The system displays form for adding a new book. The administrator of library doesn't fill all fields in the form. The system displays a message "All fields must be filled"	✔	-
22	The administrator of bookstore is on the main page and clicks button 'Home'. The system displays form for adding a new book. The administrator of bookstore doesn't fill all fields in the form. The system displays a message "All fields must be filled"	✔	-
23	The administrator of library is on the main page and clicks button 'Home'. The system displays form for adding a new book. The administrator of library uses letters in the year field. The system displays a message "Year can not contain letters"	✔	-
24	The administrator of bookstore is on the main page and clicks button 'Home'. The system displays form for adding a new book. The administrator of bookstore uses letters in the year field. The system displays a message "Year can not contain letters"	✔	-
25	The administrator of bookstore is on the main page and clicks button 'Home'. Then he/she enters term to the search bar. The system displays a list of books matching the search criteria with the option delete. The administrator of bookstore clicks a delete button of chosen book. The system displays alert message "Are you sure you want to delete this	✔	-



	book". The administrator of bookstore clicks confirm button. Then the system updates the state of book and this book won't be on the list.		
26	The administrator of bookstore is on the main page and clicks button 'Home'. Then he/she uses advanced search. The system displays a list of books matching the search criteria with the option delete. The administrator of bookstore clicks a delete button of chosen book. The system displays alert message "Are you sure you want to delete this book". The administrator of bookstore clicks confirm button. Then the system updates the state of book and this book won't be on the list.	✔	-
27	The administrator of library is on the main page and clicks button 'Home'. Then he/she enters term to the search bar. The system displays a list of books matching the search criteria. The administrator of library clicks see details button. The system displays a details page of chosen book with option delete. The administrator of library clicks a delete button. The system displays alert message "Are you sure you want to delete this book". The administrator of library clicks confirm button. Then the system updates the state of book and this book won't be on the list.	✔	-
28	The administrator of library is on the main page and clicks button 'Home'. Then he/she uses advanced search. The system displays a list of books matching the search criteria. The administrator of library clicks see details button. The system displays a details page of chosen book with option delete. The administrator of library clicks a delete button. The system displays alert message "Are you sure you want to delete this book". The administrator of library clicks confirm button. Then the system updates the state of book and this book won't be on the list.	✔	-
29	The bookstore administrator is on the main page and clicks button 'Home'. Then he/she enters term to the search bar. The system displays a list of books matching the search criteria with the option delete. The bookstore administrator clicks a delete button of chosen book. The system displays alert message "Are you sure you want to delete this book". The bookstore administrator clicks exit button. The system closes the alert and shows list of books matching the search.	✔	-

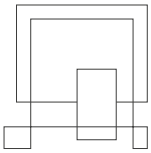


30	The administrator of library is on the main page and clicks button 'Home'. Then he/she enters term to the search bar. The system displays a list of books matching the search criteria. The administrator of library clicks see details button. The system displays a details page of chosen book with option delete. The administrator of library clicks a delete button. The system displays alert message "Are you sure you want to delete this book". The administrator of library clicks exit button. The system closes alert and displays s details page of chosen book.	✔	-
31	The administrator of bookstore is on the main page and clicks button 'Home'. Then he/she enters term to the search bar. If there is no book matching the search criteria, system displays a message "No book matching searching criteria"	X	No results are shown after search and no message is displayed
32	The administrator of library is on the main page and clicks button 'Home'. Then he/she enters term to the search bar. If there is no book matching the search criteria, system displays a message "No book matching searching criteria"	X	The system does not display any message. There is no book shown in the list
Return book			
33	The administrator of library clicks orders button in the navbar. The system displays list of borrowed books. The administrator of library clicks return button on the chosen book. The system updates status of book and displays a message "The book has been returned successfully". Then system shows list of borrowed books.	✔	-
34	The administrator of library clicks orders button in the navbar. There are no books for returning, system displays a message "There is no book to return".	✔	-
35	The administrator of library clicks orders button in the navbar. The system displays list of borrowed books. The administrator of library clicks return button on the chosen book, but session key has expired. The system cannot process the request and displays a message "Something went wrong".	✔	-



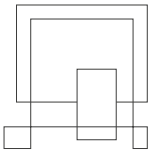
Confirm book order			
36	The administrator of bookstore clicks orders button in the navbar. The system displays list of unfinished orders. The administrator of bookstore clicks confirm button on the chosen book. The system updates status of order and displays a message "The order has been confirmed successfully". Then system sends confirmation letter to the customer and shows list of unfinished orders.	✔	-
37	The administrator of bookstore clicks orders button in the navbar. There are no orders for confirming, system displays a message "There are no unfinished orders".	✔	-
38	The administrator of bookstore clicks orders button in the navbar. The system displays list of unfinished orders. The administrator of bookstore clicks confirm button on the chosen book, but session key has expired. The system cannot process the request and displays a message "Something went wrong".	✔	-
Logout from the account			
39	An administrator of library account is being used. The administration of library clicks a log out button on the nav bar. The administrator of library becomes a guest.	✔	-
40	An administrator of bookstore account is being used. The administrator of bookstore clicks a log out button on the nav bar. The administrator of bookstore becomes a guest.	✔	-
41	A customer account is being used. The customer clicks a log out button on the nav bar. The customer becomes a guest.	✔	-

Table 2



7 Results and Discussion

The final outcome of the project can be stated as positive. Out of 42 test cases, only 2 are failing. Moreover, the failure is minor and easy to fix. That means that the project is functional and after minor fixes ready to deploy for real institutions and clients. In Appendix G can be found a User Guide presenting how to use the system.



8 Conclusions

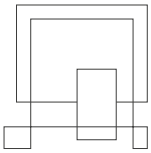
The purpose of the system was to improve the accessibility of books. In order to do that a list of steps was followed.
The first one was to analyze the background of the problem. Meaning not only the need of people, being the future customers, but also stating questions to be answered in order to be able to work with the area being bookstores and libraries, analyzing the questions in depth and stating how to get reliable answers to them, creating a time schedule and assessing the possible risks. Basing on that, it was decided that the project is worth doing.

The following step was to analyze the needs, not of the customers anymore, but of the system. Firstly a domain model and list of requirements were created. Build upon was a use case diagram, which distinguished the functionalities and divided them between specific actors. In this phase also the needed security of the system was taken under consideration and probable threats were stated.

Having successfully done the analysis, it was possible to proceed with designing the system. This chapter included diagrams based strictly on the previous stage, which are a blueprint of the implementation to come. Those include selecting the architecture design pattern, which in this case was the 3-tier architecture, next selecting main parts needed in each tier and the technologies, in which it would be beneficial to implement them and taking a closer look on each of the main parts, making it more and more specific, making it at last clear on how to implement it. What is more, is defining the communication in the system and designing the flow of it. In order to accomplish that, a communication protocol was constructed. Last but not least, was the final design of the needed security of the system.

Having designed the blueprint of the system, implementation have started. The system was implemented according to its design, containing all the needed parts of it.

To prove the success of the implementation, which would prove the success of the design, which would prove efficacious analysis, requirement test descriptions were created. They demonstrated that the system fulfills all requirements and contains almost no bugs. Moreover, the implementation was tested with the use of unit tests, which positively validated the code.

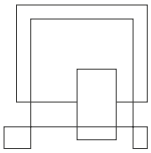


9 Project future

The major part of tasks was completed and it can be said that the system is fully working. For the future there are some features left that can be implemented like:

- sending remainder emails before the due return date to the customers
- creation of new administrators handled by one administrator of the whole system
- a section with the customer's profile containing the customer's personal information
- pagination of search requests
- buying a domain

Moreover, in order to deploy the system, an agreement with real libraries and bookstores would have to be established.



10 Sources of information

Jamil Zaki, 2011. What, Me Care? Young Are Less Empathetic. Scientific American. [e-journal] https://www.scientificamerican.com/article/what-me-care/

Christopher Ingraham, 2016. The long, steady decline of literary reading. The Washington Post. [e-journal] https://www.washingtonpost.com/news/wonk/wp/2016/09/07/the-long-steady-decline-of-literary-reading/?noredirect=on&utm_term=.7935d603d3c3

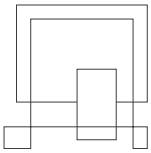
Dennis J. Sumara, 2002. Why Reading Literature in School Still Matters: Imagination, Interpretation, Insight. 1st ed. New York: Routledge

Putai Jin, 1992. Efficacy of Tai Chi, brisk walking, meditation, and reading in reducing mental and emotional stress. Journal of Psychosomatic Research. [e-journal] https://linkinghub.elsevier.com/retrieve/pii/002239999290072A

Gery Deer, 2016. Improve your vocabulary through reading. GLD Enterprises Communications, Ltd. [online] http://geryldeer.com/improve-your-vocabulary-through-reading/

Michael Kozlowski, 2018. Reading books is on the decline. GoodEReader. [online] https://goodereader.com/blog/bookselling/reading-books-is-on-the-decline

Hanho Jeong, 2012. A comparison of the influence of electronic books and paper books on reading comprehension, eye fatigue, and perception. emerald insight. [online] https://www.emeraldinsight.com/doi/pdfplus/10.1108/02640471211241663



11 Appendices

Appendix A Project Description – ProjectDescription.pdf

Appendix B Data glossary – DataGlossary.pdf

Appendix C Diagrams as Astah projects – Diagrams.zip

Appendix D Domain Table, Logical data model – DomainTable_LogicalModel.pdf

Appendix E Socket communication – Communication.pdf

Appendix F Source code – Code.zip

Appendix G User guide – UserGuide.pdf

Appendix H Web Services – WebServices.pdf