Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer

Development of an application for a personal library and expense management

Nazrin Orujaliyeva

Supervisor: Ing. Pavel Náplava, Ph.D.
Field of study: Software engineering and technologies
May 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Rakhimova**	Jméno: **Aiya**	Osobní číslo: **499327**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Zadávající katedra/ústav:	**Katedra počítačů**		
Studijní program:	**Softwarové inženýrství a technologie**		

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webová aplikace pro darování

Název bakalářské práce anglicky:

Web platform for donation

Pokyny pro vypracování:

The project aims to create a platform for donations to sick people who need material assistance. As part of this work:
- analyze existing applications and platforms for donations available on the market,
- design a user interface for the application to motivate future users to use it,
- analyze technologies for the implementation of front-end, back-end, and database parts of the application,
- design application architecture with a focus on the back-end part of it,
- implement a proof of concept version of the platform,
- design test cases,
- perform appropriate testing of a proof of concept and automate testing of the core scenarios.

Seznam doporučené literatury:

Richards, Mark. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020.
Walls, Craig. Spring Boot In Action. Shelter Island, NY: Manning Publications, 2016.
Tidwell, Jenifer, Charlie Brewer, and Aynne Valencia. Designing Interfaces: Patterns for Effective Interaction Design. Beijing: O'Reilly, 2020.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Kyrylo Bulat   katedra počítačů   FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.02.2023**   Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Kyrylo Bulat podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.

Datum převzetí zadání   Podpis studentky

Acknowledgements

I want to express my sincere gratitude to my supervisor, Ing. Pavel Náplava, Ph.D for his expert direction, insightful counsel, encouragement, and the time he invested in me and this bachelor thesis.

Declaration

By the Guideline for adhering to ethical principles when constructing a final academic thesis, I confirm that the thesis provided here is entirely my own work and that I have properly cited all relevant sources of information.

Prague, May 2023

Abstract

The main goal of this bachelor thesis is to analyze, design, successfully implement and test the "Libget" application. The application supports book organization, expense management, and the ability to add quotes and notes.

Keywords: library, book organization, expense management, quotes, budget, statistics, Android, Kotlin

Supervisor: Ing. Pavel Náplava, Ph.D.

Abstrakt

Hlavním cílem této bakalářské práce je analyzovat, navrhnout, úspěšně implementovat a otestovat aplikaci "Libget". Aplikace podporuje organizaci knih, správu výdajů a rozpočtu a také možnost přidávat citáty a poznámky.

Klíčová slova: knihovna, organizace knih, správa výdajů, citaty, rozpočet, statistika, Android, Kotlin

Contents

Figures

Tables

Chapter 1

Introduction

People have grown accustomed to using computers and computer programs in the modern information and communication systems era. Mobile applications are run on a small handheld device that is portable, user-friendly, and accessible from anywhere. Nowadays, many individuals use mobile applications to communicate with friends, access the internet, manage files, create and handle documents, and run businesses.[1] If you enjoy reading books, there's a good chance your library is quite extensive. And if you enjoy reading actual books, they are likely beautifully displayed on your home's bookshelves. What about when you're away from home? Walking up to your books and seeing what you have and don't is simple when you're at home. When you're at the library or a bookshop looking for your next book, keeping track of the books you've read or bought is easier when you have a virtual library. Also, remembering your current expenses on books before buying a new one can help manage your financial goals.

I have not always been a bookworm, but recently I developed a passion for books. I have discussed the idea for the application with my friend, who has been reading books her whole life. She has a lot of friends who share a passion for books and present the idea for them. They agreed that such an application would be highly beneficial. Later, when I was in a bookstore, considering buying two new books and considering my finances, I thought an application like this would be handy.

This bachelor thesis aims to develop a mobile application called "Libget" that can help book lovers organize their libraries more efficiently and economically. The core idea of the "Libget" application combines three main concepts:

helping stay within budget while buying books, keeping track of the read books, and saving quotes.

1.1 The goals of the thesis

The goals of this thesis are:

- to analyze and compare existing applications that help with organizing a personal library;
- to create the analysis of key features and use cases of the "Libget" application;
- using the results of the analysis to design and implement a simplified version of the "Libget" mobile application that will support controlling book expenses, book tracking, and saving quotes and notes;
- to conduct user testing to evaluate the application's functionality and usability.

The structure of this bachelor thesis corresponds to these goals.

Chapter 2

"Libget" application: definitions and usage

Chapter 1 introduced the main concepts of the "Libget" application, and this chapter delves into the definitions of key concepts and examples of practical application in real-world scenarios.

2.1 Definitions of the key terms

Below are the key terms that my application is based on, along with their explanations:

- book - a written work published in printed or electronic form.
- library - a collection or group of collections of books and/or other print or nonprint materials organized and maintained for use (reading, consultation, study, research, etc.).
- reader - a person who reads or is fond of reading. Later referred to as "user".

2.2 The concept of using the "Libget" application

The first tool to make organizing a personal library easier is keeping track of books. If your home library is extensive, memorizing every book you own is hard. There are more efficient approaches than relying on a paper-based list for keeping track of books. Such lists can be excessively lengthy and time-consuming to create, not to mention having a separate list for books you want to purchase and the challenge of finding specific books from among the many others. The simplest and most efficient method to improve something today is to have a smart application on the device that can speed up and simplify everyday tasks. So instead of creating lengthy and time-consuming lists, users can open the mobile application and easily find a specific book or add a book by filling in the designed fields. To add a book to the "Wish List," users can simply check the box labeled "Wish List." It's a quick and easy process.

Another valuable tool for book lovers is keeping track of their finances. It's crucial to stay aware of the various categories of products and services one spends money on monthly and keep track of them. One of those categories is books. It's common to feel tempted to purchase new books frequently. However, it can be beneficial to monitor your expenses and establish a budget for yourself.

Finding quotes or notes in a particular book can take some time. There is a more convenient way to keep them in one place. To locate quotes from a book, one can easily use the filter in the dedicated "Quotes" section and choose the book's title.

And the above-described data is available for the user at all times and in all places where the user has the mobile phone with them.

Chapter 3

Analysis of existing applications

It is crucial to consider the existing applications for keeping track of personal books to confirm that the new "Libget" application's development is reasonable and to gain inspiration from the existing apps' functionality and innovative software solutions.

3.1 Existing applications comparison

There are many library organizing applications available in Google Play. The applications were searched in Google Play using the “book tracker” keyword.[2]The search was restricted to Android applications as "Libget" is an Android application. The reasons for choosing Android are listed in Chapter 5.1. The further analysis is based on reviews and the application's description wherever it was available on Google Play. Based on the reviews of an application, advantages, and disadvantages were determined. Unfortunately, the StoryGraph application reviews were not available on Google Play. Another criterion was a rating of over 4 out of 5 and over 100 000 downloads. This analysis compares the five most popular among them and identifies the presence of key features. The applications are listed in decreasing rating order. The applications have paid premium versions or additional in-app payments. This is described in the last column, "Price," in Table 2.1.

The standards for applications to create analysis and comparison are:

- an app must help organize a personal library;
- an app must have an expense management possibility;
- an app must have a "Quotes" section, where a user can add quotes and notes;

In agreement with the supervisor, the six most popular applications were compared - Storygraph, Read More: A Reading Tracker, Bookly: Book Reading Tracker, Bookshelf - Your virtual library, My Library, and Handy Library. These applications meet the criteria, such as a rating of over 4 out of 5 and over 100 000 downloads.

3.1.1 Storygraph

StoryGraph is currently the most popular application with a 4,9/5 rating and over 100 000 downloads on Google Play[3][4]. The application allows users to search books by mood and create a graph (see Figure 3.1). Users can create lists to share with friends or the public, create their own reading challenges, or join others'. The recommendation algorithm uses a machine-learning AI that understands users' reading preferences and recommends books.

User reviews: reviews are not available on Google Play.

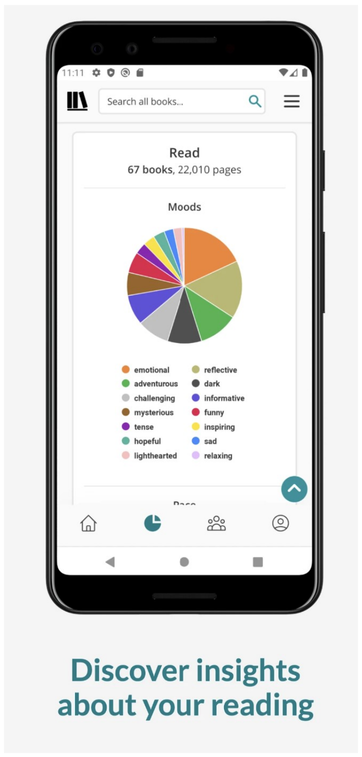


Figure 3.1: An example of statistics based on user's mood.

The app suffices only one of the three main standards, book tracking, and does not support expense management and creating quotes.

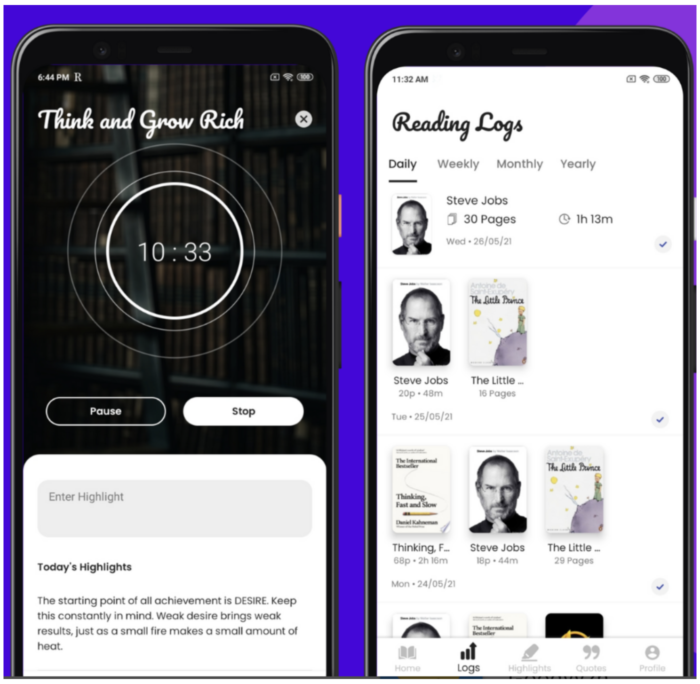
3.1.2 Read More: A Reading Tracker

This application has a 4,8/5 rating on Google Play and over 100 000 downloads[5]. Read more helps users to get out of a reading slump. It allows users to set a daily reading goal, follow their reading history, track reading time, and create quotes. An example of the reading timer feature is presented in Figure 3.2.

User reviews:

- Advantages: design, timer, reading speed calculation, convenient trackers, the ability to convert the audio format into pages, and set goals for yourself for the day, month, and year.

- Disadvantages: only sometimes saves the entered number of pages read, a problem with adding a cover to the book.



The starting point of all achievement is DESIRE. Keep this constantly in mind. Weak desire brings weak results, just as a small fire makes a small amount of heat.

Figure 3.2: Timer feature.

The app has two main standards, book tracking and creating quotes, but it does not support expense management.

3.1.3 Bookly: Book and Reading Tracker

This application has a 4,8/5 rating on Google Play and over 100 000 downloads[6] [7]. Bookly is for book tracking and reading; the actual reading is done with a book, e-book, or audiobook. It also allows users to set a reading goal, play ambient sounds while reading, track reading time, see different kinds of statistics, and create quotes. An example of the reading timer feature and infographics is presented in Figure 3.3.

User reviews:

- Advantages: great to track the amount of time it takes to read a book and break it down into sessions and time per page, and the features to add thoughts and quotes from the book.

- Disadvantages: only English interface, ads appear too often and take over the entire screen, a user can't add more than ten books and can't delete books, and simple, easily features locked behind a paywall.

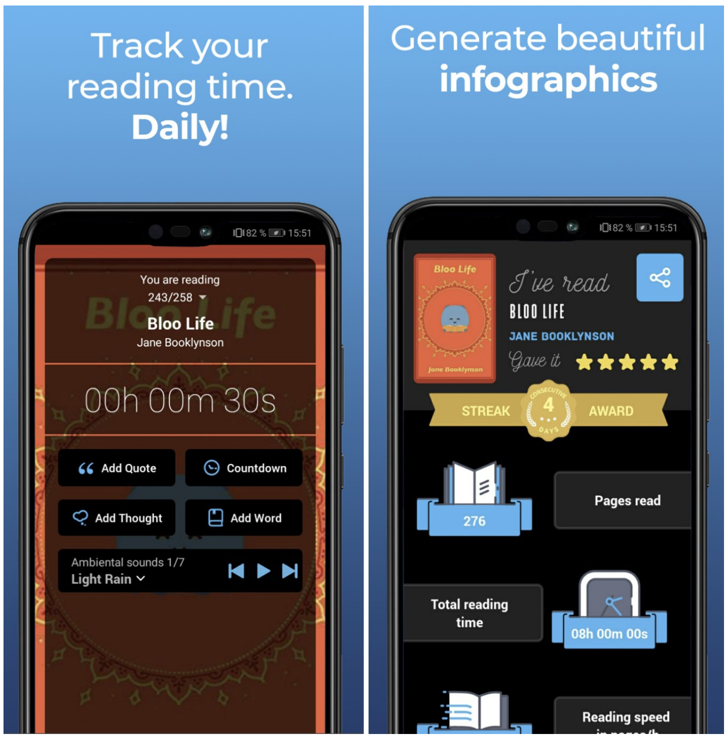


Figure 3.3: Timer, infographics.

The app has two of the three main standards, book tracking and creating quotes, but does not support expense management.

3.1.4 Bookshelf

Bookshelf has a 4,7/5 rating on Google Play and over 100 000 downloads[8] [9]. The application supports book tracing, building statistics for a specific time, and how many books were read in a month. Figure 3.4 is an example of a bookshelf and explore pages.

User reviews:

- Advantages: design, writing notes, a lot of filter options, finds books on the internet.
- Disadvantages: sometimes, the system may encounter challenges in identifying books through their ISBN codes.

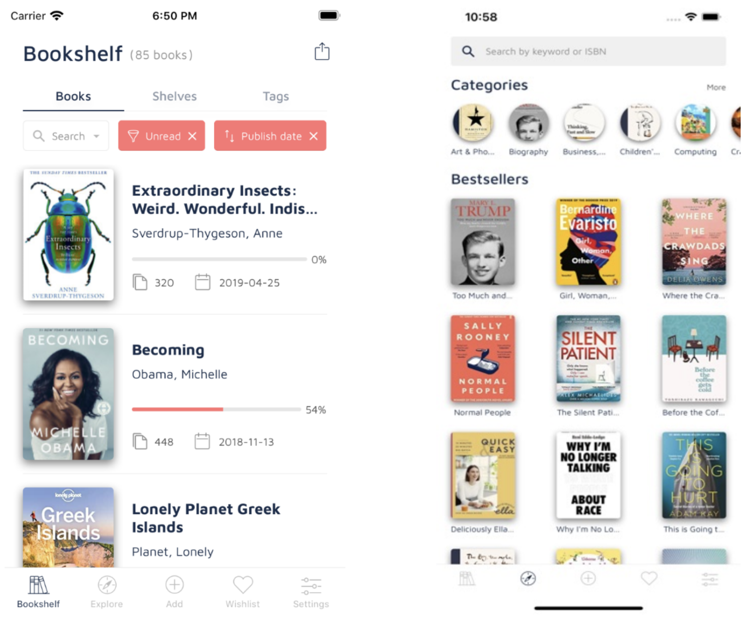


Figure 3.4: An example of a bookshelf and explore pages.

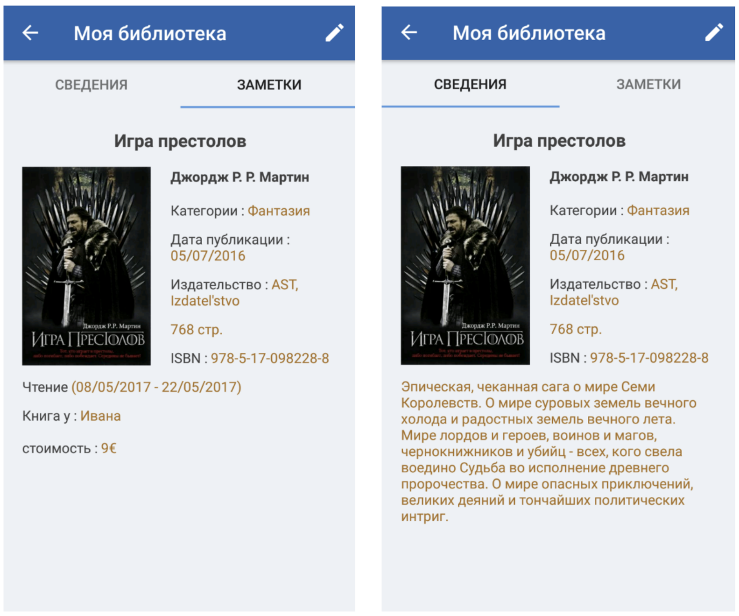
The app suffices only one of the three main standards, book tracking, and does not support expense management and creating quotes.

3.1.5 My library

The application has a 4,5/5 rating on Google Play and over 100 000 downloads[10]. An example of a book details page and a short description page is shown in Figure 3.5. In the details, apart from the name of the book title and name of the author, there is also information about the publisher, date of publishing, number of pages, ISBN code, the cost of the book, who borrowed the book, and the start and end of the reading.

User reviews:

- Advantages: ad-free, ease of use, ability to upload covers, extensive search function by title, genre, and any word that was crammed into the information about the book.
- Disadvantages: sorting only by authors or title, no sorting by series or publisher, no ability to edit the image directly in the application, no way to distribute books into categories (shelves), no rating option.



Игра престолов

Джордж Р. Р. Мартин

Категории : Фантазия

Дата публикации : 05/07/2016

Издательство : AST, Izdatel'stvo

768 стр.

ISBN : 978-5-17-098228-8

Чтение (08/05/2017 - 22/05/2017)

Книга у : Ивана

стоимость : 9€

Моя библиотека

Игра престолов

Джордж Р. Р. Мартин

Категории : Фантазия

Дата публикации : 05/07/2016

Издательство : AST, Izdatel'stvo

768 стр.

ISBN : 978-5-17-098228-8

Эпическая, чеканная сага о мире Семи Королевств. О мире суровых земель вечного холода и радостных земель вечного лета. Мире лордов и героев, воинов и магов, чернокнижников и убийц - всех, кого свела воедино Судьба во исполнение древнего пророчества. О мире опасных приключений, великих деяний и тончайших политических интриг.

Figure 3.5: An example of a book description in a personal library.

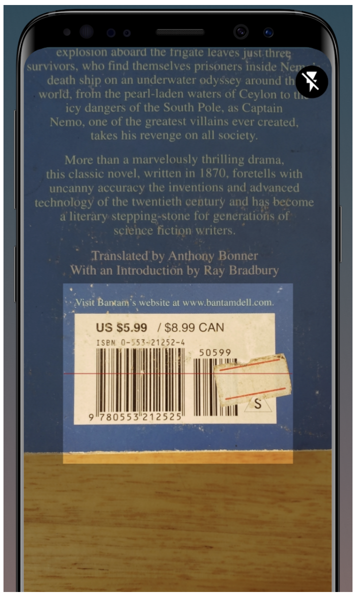
The app suffices only one of the three main standards, book tracking, and does not support expense management and creating quotes.

3.1.6 Handy library

Handy library has a 4,3/5 rating and over 100 000 downloads on Google Play[11]. The application shows how much money was spent and the number of books bought in total in 6 or 12 months, but it does not show which books were purchased and the monthly expenses. Also, it shows what genre of books were purchased the most and a reading status report. Figure 3.6 is an example of scanning a book by its barcode.

User reviews:

- Advantages: the ability to create shelves, many filters, read marks and ratings, and design.
- Disadvantages: translation problems, payment only by Google Payment, which is not available in some countries, a problem with finding books by barcode, finding pictures for a cover on the Internet, no sharing for family members.



explosion aboard the frigate leaves just three
survivors, who find themselves prisoners inside Nem
death ship on an underwater odyssey around th
world, from the pearl-laden waters of Ceylon to th
icy dangers of the South Pole, as Captain
Nemo, one of the greatest villains ever created,
takes his revenge on all society.

More than a marvelously thrilling drama,
this classic novel, written in 1870, foretells with
uncanny accuracy the inventions and advanced
technology of the twentieth century and has become
a literary stepping-stone for generations of
science fiction writers.

Translated by Anthony Bonner
With an Introduction by Ray Bradbury

Visit Bantam's website at www.bantamdell.com.

US $5.99 / $8.99 CAN
ISBN 0-553-21252-4
50599
9 780553 212525
S

Figure 3.6: An example of scanning by a barcode.

The app suffices only one of the three main standards, book tracking, and does not support expense management and creating quotes.

3.2 Conclusion of the analysis

Table 3.1 below compares the selected applications from Google Play based on the critical standards defined in Chapter 2.1.

No	App name	Books list	Expenses	Quotes	Rating	Number of uploads	Price(in USD)
1.	StoryGraph	Yes.	No.	No.	4.9	100 000+	Premium version 4.99.
2.	Read More: A Reading Tracker	Yes.	No.	Yes. Text scanner.	4.8	100 000+	In-app purchase 12.99 per item.
3.	Handy Library	Yes.	Yes.	No.	4.8	100 000+	In-app purchase 1.20 - 7.99 per item.
4.	Bookly: Book Reading Tracker	Yes.	No.	Yes.	4.8	100 000+	A 7-day trial is free, then offers subscription packages: 4.99/month, 19.99/6 months, and 29.99/year.
5.	Bookshelf -Your virtual library	Yes.	No.	No.	4.7	100 000+	In-app purchase 1.49 - 11.99 per item.
6.	My Library	Yes.	No.	No.	4.5	100 000 000+	Free.

Table 3.1: Comparison of applications from Google Play.

Two of the six applications have a "Quotes" section, where users can add quotes and notes(the second and fourth in Table 3.1). But there is a lack of expense management features.

Only one application(the third in Table 3.1) has expense management features. However, there is no "Quotes" section.

The first, fifth, and sixth applications in Table 3.1 meet only one standard - book organization feature, but not the rest.

After analyzing these apps, none simultaneously satisfied all requirements(book organization, expense management, and quotes section). That is why creating the “Libget” application, which meets these requirements, is reasonable. The functionalities of the analyzed applications inspired the functional requirements listed in Chapter 4.1.

3.3 The “Libget” application core

The application’s name comes from combining "library" and "budget." The "Libget" application’s core is stated in this part. The fundamental features are based on the analysis of existing applications and the goals of the thesis, presented in Chapter 1.1. The features include:

- Book organization:
    - see brief content of the book;
    - ability to choose a book cover;
    - rate books;
    - search books by name or author;
    - filter by author, rating, genre, country, read, unread, started;
    - filter by author and genre;
    - sort by price and rating;
    - what books were lent and to whom and whom were borrowed from;
- Expense management:
    - view a graph displaying the number of books purchased in the last month, the past three months, or the entire year, along with the corresponding amount spent;

  - see how much money was spent on books in total in a particular month;
  - set a budget, which specifies the maximum amount you would like to spend on books;

- Quotes:
  - ability to write quotes or notes;

In the next chapter, there is a detailed explanation of the application’s specifications.

Chapter 4

Analysis of the "Libget" application

The study in the earlier chapters shows that the "Libget" application's primary functions are book organization, expense management, and a collection of quotes.

As a result, the fundamental ideas behind the "Libget" application are now clear, and a more thorough analysis of it may begin. This chapter discusses the functional and non-functional requirements, the class diagram, and the use cases of the "Libget" application.

4.1 Functional requirements

The application's specific behavior is described by the functional requirements[12].

According to the goals of this thesis, the "Libget" application should be "simple," which means that it should only have the critical features required to carry out the app's primary purpose. At the same time, this thesis does not anticipate implementing further sophisticated functionalities. The reason for this is that in the beginning, based on a "simple" application, it is necessary to demonstrate both the utility of using such an application through a voice assistant and the demand for the program among consumers. The functional requirements were determined by the application's expectations and agreement with the supervisor described in Chapter 2.3.

These are the "Libget" application's functional requirements:

1. The application enables registering.
2. The application enables logging in or out.
3. The application enables showing all the books the user has bought.
4. The application enables viewing book details (name, price, author, rating, genre).
5. The application shows books and expenses statistics to see how many books the user bought in the last month, three or twelve months, and how much money was spent.
6. The application enables setting and editing a monthly limit.
7. The application allows searching for a book in the library.
8. The application enables showing quotes from a book.
9. The application enables filtering by author, genre, sort by price, and rating.
10. The application enables adding or deleting a book from the Wish List.
11. The application enables adding or deleting a book from the main list of books that were bought.
12. The application enables rating books.
13. The application enables manually writing a quote and adding it to the "Quotes" section.
14. The application enables choosing a book cover.
15. The application enables taking a picture of a text and adding it to the quotes section.
16. The application enables marking a book as TBR(To be read), Started, or Read.
17. The application enables filtering by country, TBR(To be read), Started, or Read.
18. The application enables marking a book as borrowed/unborrowed from someone or to someone (date, lender's name).
19. The application enables searching by ISBN code.
20. The application enables the calculation of reading speed.

21. The application enables setting daily, monthly, and yearly goals.
22. The application enables tracking progress reading books, last read page number (through the interface)
23. The application enables the conversion of the audio format into pages.
24. The application enables a machine-learning AI that understands users' reading preferences and recommends books.
25. The application enables to time the user's reading time.
26. The application enables searching a book by any word crammed into the information about the book.
27. The application enables one to find books on the Internet.
28. The application enables to share of book lists.
29. The application enables synchronizing devices.
30. The application enables the creation of literary forums.
31. The application enables following book news.

The requirements are implemented depending on priority, which is in Table 4.1.

Priority	1	2	3
Name of requirement	FR1-FR14	FR15-FR17	FR18-FR31

Table 4.1: Priority of functional requirements.

In Table 4.1, priority 1 means that these requirements must be implemented. Priority 2 indicates that these requirements will be implemented if there is enough time. Priority 3 means that these requirements will be implemented in future application versions. The priority of requirements is determined by the introductory definition in Chapter 1 and the number of a requirement's occurrences in similar applications.

4.2 Non-functional requirements

The non-functional requirements outline the actions that the application must take.[12]

This application's concept implies the majority of the non-functional requirements. Android has been chosen as the operating system for the "Libget" application. I've solely used Android-based devices in the past. Thus I have more familiarity with Android app development. Furthermore, because the program is intended to be accessible worldwide, English was selected as the primary language of the application.

The complete list of the "Libget" application's non-functional requirements is shown below:

1. The application must have an English interface.
2. The application will run on smartphones with Android version 5.0 (API level 21) and newer.

4.3 Class diagram

The class diagram in Figure 4.1 shows how the system is organized. It describes classes, attributes of the classes, their relationships, and multiplicities[14].

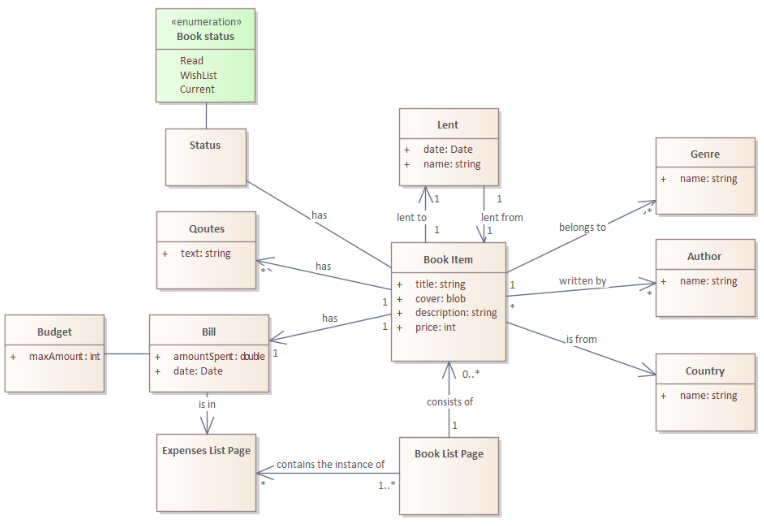


Figure 4.1: Class diagram.

This diagram is based on the concepts described in Chapters 2.1 and 2.2.

The Book List Item class contains six attributes: "title," "cover," "description," and "price."

Bill class contains information about the amount of money a person has spent and the date of purchase. The "maximumAmount" in Budget describes the maximum amount the user would like to spend.

The Book List Page contains the instance of the Expenses List Page so they can interact.

Lent class describes when a book was lent and to whom. Or otherwise, what book a user borrowed and from whom.

Quotes class contains the attribute "text," which presents a quote or a note from a book.

Books have a status such as "Read," "WishList," or "Current." It indicates whether the user has read the book, intends to purchase it, or is currently reading it.

Author, Genre, and Country have only one attribute, "name."

4.4 Use cases

A use case is a set of activities describing an activity the program supports. The method use cases are developed to ensure they address the application's functional needs. The external entity that utilizes or impacts the application is an actor. Relationships between actors and use cases are depicted in use case diagrams[15]

4.4.1 Actors

The "Libget" program has two actors: the User and the Application (see Figure 4.2). The use cases that the user launches are connected to the user. The use cases connected to the application are those that are launched automatically from other use cases.

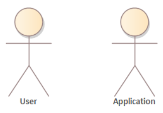


Figure 4.2: Actors.

4.4.2 Use case diagram for user

For use cases of the User actor - see Figure 4.3.

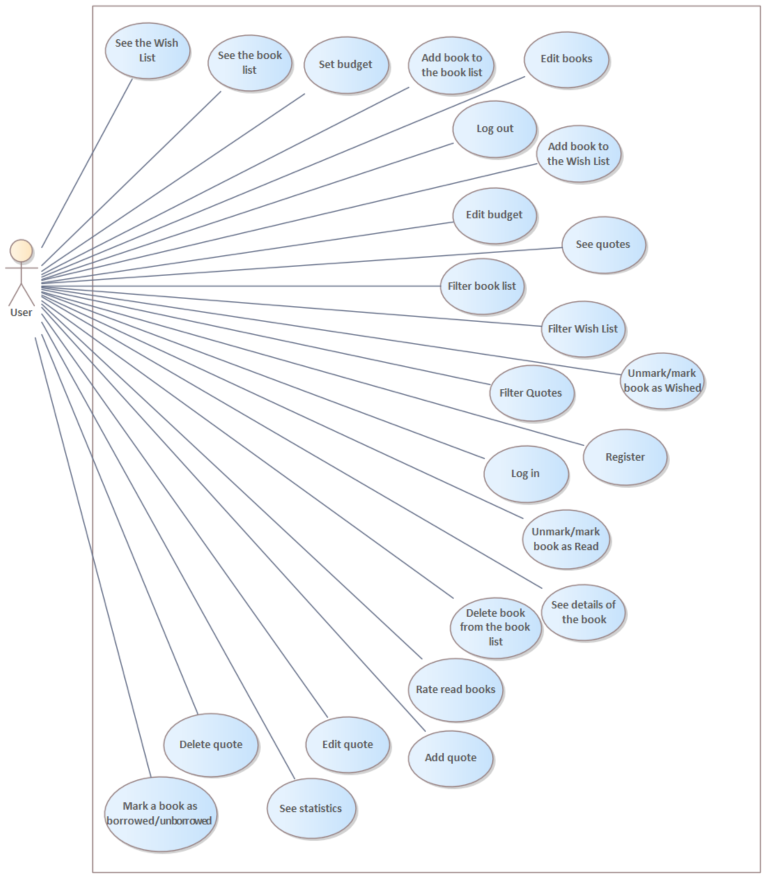


Figure 4.3: Use case diagram for the User actor.

Twenty-one use cases were defined based on the functional requirements listed in Chapter 3.1. Most of them are simple, such as Delete a quote, Delete book from the book list, See quotes, See the book list, See details of the book, See Wish List or See statistics, Edit budget, Edit books, Edit quote, Mark/Unmark book as Read, Mark/Unmark book as WishList, Mark book as borrowed/unborrowed, Set budget. More complex would be Filter books list, Filter Wish List, Filter quotes, Add book to the book list, Add quote, Add book to the Wish List, Register, Log in, Log out, and Rate read books. Some use cases were not implemented due to their complexity and time limitations. These use cases correspond to functional requirements with priority 2 and 3, as indicated in Table 3.1.

Chapter 5

"Libget" application design

Instead of focusing on implementing a solution, design stresses a conceptual one that satisfies the requirements. Designs are ultimately implementable[16]. This chapter explains the "Libget" application's design, as well as the application's wireframes.

5.1 Wireframes

Wireframes were developed to display the application's layout. They adjust and show the fundamental organization of each page.[17] The designs and functionality of other existing applications inspired the design. For expense management, I designed my own wireframe. Additionally, it was discussed and approved by the supervisor. Below are shown three wireframes as they are the main concepts of the "Libget" application. The wireframes are examples of how the application could look like.

There is a home "Libget" page in Figure 5.1. The application opens to this page as the first one that appears. Users can see all the books in their library. A "Filter" button allows users to filter the list by author, genre. Using the "Sort" button, books can be sorted by price and rating. The "Add" button allows you to add a new book to the library. The user can switch to the "Wish List" section, where books that the user wishes to buy, that the user wishes to buy, are displayed. Users can also switch to the "Quotes" section to see all the quotes they added. The search bar at the top of the screen allows users to find a book in their library.

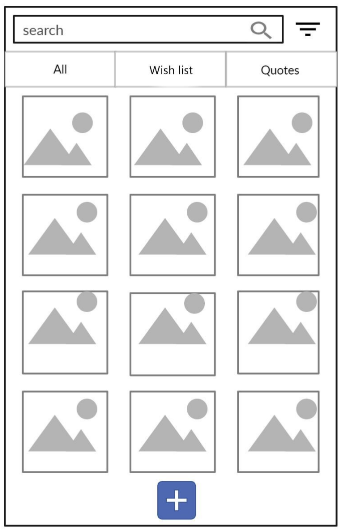


Figure 5.1: The Home page.

In Figure 5.2, there is the Statistics page where users can see how many books they bought and how much money they spent. It is possible to determine the budget - the limit of money the user would like to spend on books.

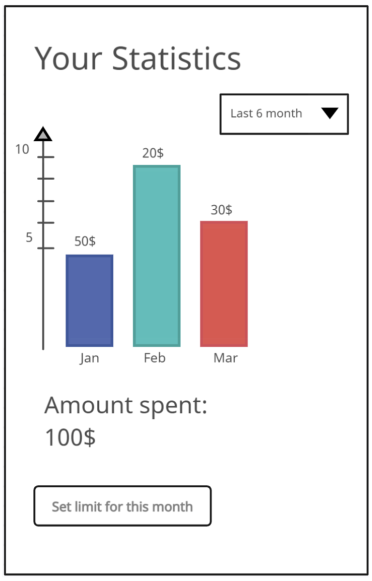


Figure 5.2: Statistics page.

In Figure 5.3, there is the Quotes page where users' quotes and notes are displayed. Under the text, there is also the title of the book.

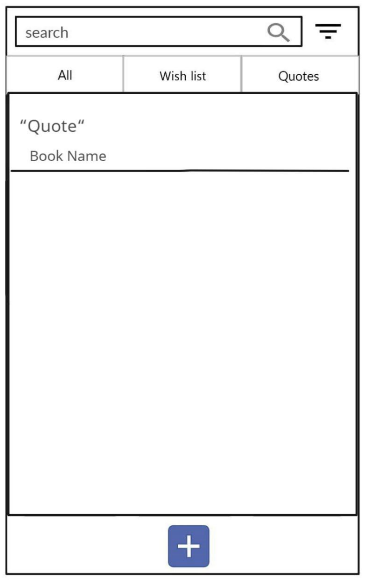


Figure 5.3: Quotes page.

These wireframes were presented to three potential users. Users approved the Home and Quotes pages; they seemed logical and easy to understand and use. However, the "Statistics" wireframe was deemed not telling and inconvenient for tracking expenses.

5.1.1 User interface

The above-described wireframes and the non-functional requirements in Chapter 4.2 were the basis for the "Libget" application's user interface concept. Figure 5.4. shows the high fidelity[35] prototype in Figma[24]. The prototype closely matches the final result of the application's design. Figma is a collaborative web application for interface design; further details of Figma are provided in Chapter 6.1.1.

Due to user testing of the wireframe and the application creation, some changes were made. The Home and Statistics pages differ from the corresponding wireframes. The Home page now features bottom navigation and a "Sort" button for enhanced user experience. The Statistics page has a different graph that is more straightforward than the previous one. The option to set a budget limit has been newly added. The following screenshots describe the content of the implemented application's pages in more detail.

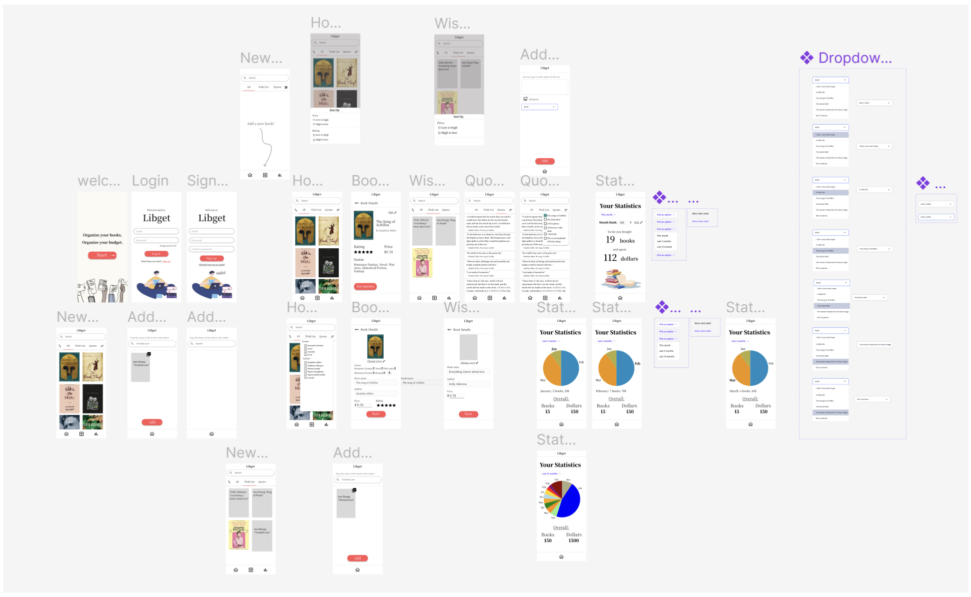


Figure 5.4: Screens map in Figma.

The Home page, as shown in Figure 5.5, displays a complete list of all the books the user owns. The bottom navigation bar facilitates easy switching between the Home page, "Add new book" page, and "Statistics" page. Tabs such as "All," "Wish List," and "Quotes" are also available. The "Filter" and "Sort" allow users to display books according to their preferences. Clicking on a book will reveal its details; the user can edit them, view quotes from the book, or add new ones. A search bar is provided at the top of the page to help users find a specific book in their library.

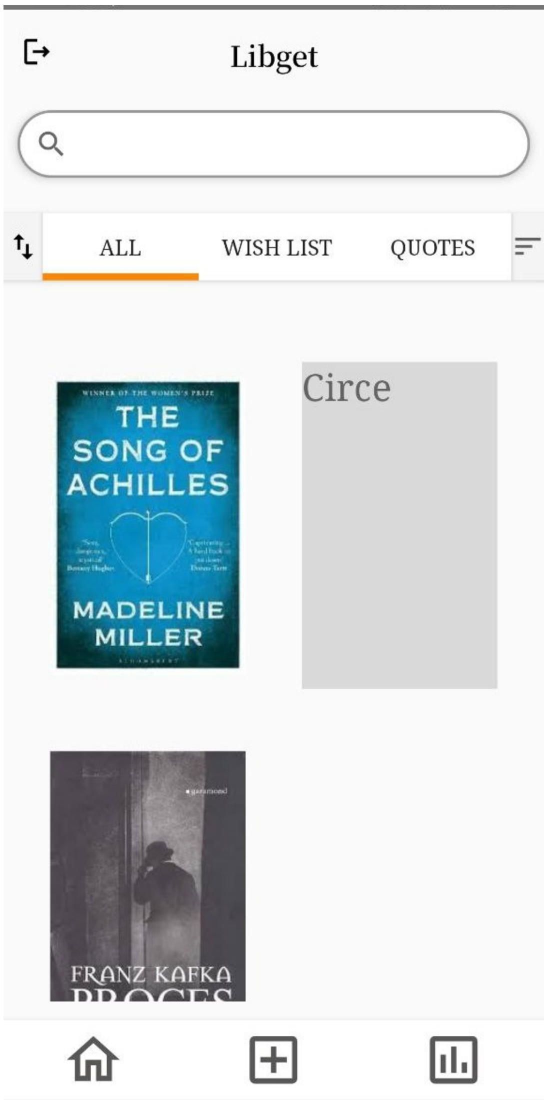


Figure 5.5: Home page.

On the Statistics page, shown in Figures 5.6,5.7, and 5.8, users can track their book expenses and the number of books purchased. They can view this data for the last thirty days(within the application, it is displayed as "1 month"), the past three months, or the entire year. Users also have the option to set a budget limit.

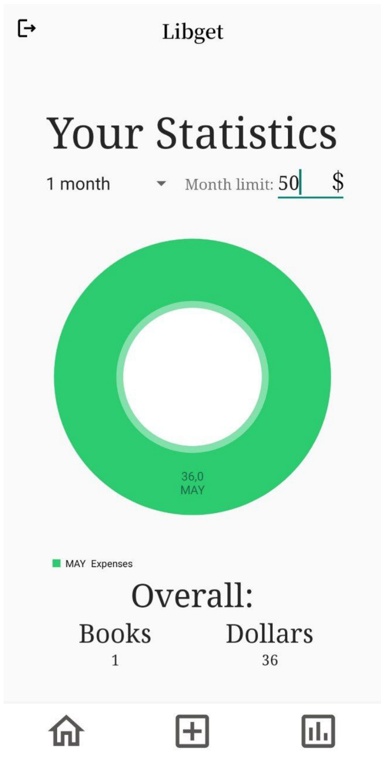


Figure 5.6: Statistics "1 month" page.

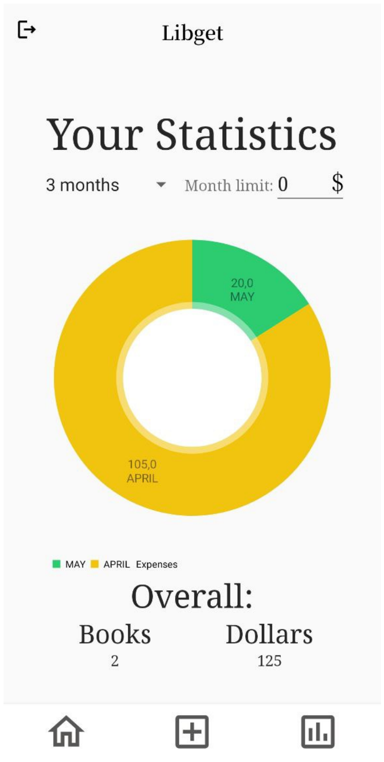


Figure 5.7: Statistics "3 month" page.

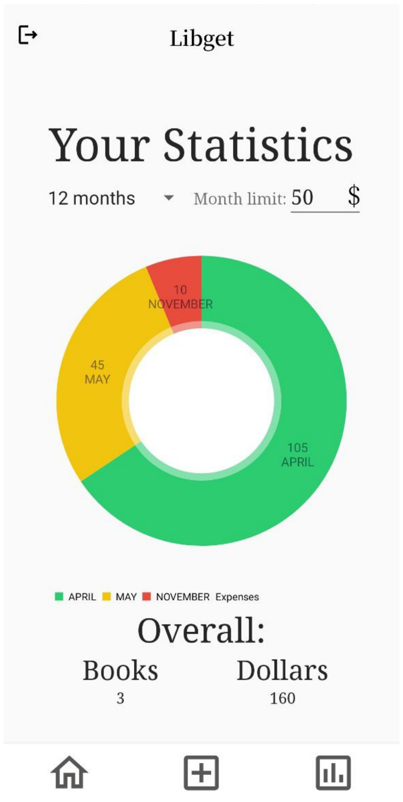


Figure 5.8: Statistics "12 month" page.

The prototype was presented to potential users who had previously tested the wireframes, and they approved it. The pages and functionality were deemed logical, and the interface was user-friendly.

Chapter 6

Development of the "Libget" application

This chapter details the creation of the "Libget" application, including the technologies and architectural decisions used in the development process. Due to limited development experience and time constraints, only a part of the functional requirements from the Class diagram, presented in Chapter 4.3, was implemented as the diagram also contains functional requirements with a different priority than 1. The functional requirements are listed in Chapter 4.1.

6.1 Development process

The development process included these steps:

1. **Planning:** Identified the requirements and goals of the application, including features, user experience, and target audience.
2. **Architecture Design:** The MVVM (Model-View-ViewModel)[27] architecture was chosen to separate concerns and improve code maintainability and testability. ViewModel communicates with the repository to fetch data and update the UI[28].
3. **UI Design:** Wireframes and UI designs were created using design tools and guidelines. The main focus was intuitive navigation, consistent branding, and responsive layouts for different screen sizes.

4. **Database Design:** Implemented data models and DAO (Data Access Object)[33] classes to interact with the database.

5. **Implementation:** Wrote modular and reusable code, adhering to coding best practices and code style guidelines[34].

6.1.1 Used technologies

This describes the technologies used to create the "Libget" application. I chose certain technologies, such as Android Studio, Kotlin, and Figma, because I became very familiar with them during my studies.

- **Operational system.** Implementation took place on Windows 11. The application was also tested on a physical device with Android 13 during development.

- **Integrated development environment (IDE).** Android Studio[18] offers a layout editor, an emulator, and compatibility with the versioned system GIT. It also has an intuitive UI. The requirements for Android development are thus entirely met.

- **Technology Stack Selection:** Kotlin[19] was selected as the primary programming language for Android development due to its modern features and seamless integration with the Android framework. In order to enhance app performance, efficient development was achieved by using Android Jetpack components[30], including LiveData, ViewModel, and Navigation. The alternative for Kotlin is Java[31]. Java and Kotlin are object-oriented languages, but both languages have different purposes.

- **Markup language:** As the application's markup language was selected XML[29]. XML uses human, not computer, language. XML is readable and understandable, even by novices, and no more difficult to code than HTML[28].

- **Build tool.** Gradle[21] was chosen as the preferred build tool since Android[22] has officially endorsed it over Maven[23].

- **Design creation.** Figma[24] was used to create the prototype of the application's design. Figma makes prototyping a seamless and straightforward procedure. It is highly user-friendly, has a short learning curve, and puts a wide variety of tools and functionalities such as smart animations, advanced overlays, transitions, scrolling and hovering interactions, auto layout, and many plugins.

- **Database.** The Firebase Realtime Database[25] is a cloud-hosted database. The advantage of Firebase over other databases is that Firebase quickly syncs data via Android, iOS, and JavaScript SDKs, allowing for expressive queries that scale with the size of the result set. It uses data synchronization—every time data changes, any connected device receives that update within milliseconds.

- **The com.github.PhilJay:MPAndroidChart library** was used to incorporate interactive charts and graphs in the application. It provides a wide range of charting functionalities, including line charts, bar charts, and pie charts. The library offers customization options for the appearance and behavior of the charts, allowing them to be tailored to fit the application's design. By leveraging MPAndroidChart, the application was able to present data in a visually appealing and informative manner, enhancing the overall user experience.

6.2 Architectural Decisions

By following these development steps and making informed architectural decisions, the application achieves better code organization, maintainability, testability, and overall user experience.

1. MVVM Architecture: MVVM was chosen for better separation of concerns, maintainability, and testability. ViewModel acts as an intermediary between the data repository and the UI components, ensuring a clear separation of business logic from the UI.

2. Android Jetpack Components: Leveraged Android Jetpack components like LiveData, ViewModel, and Navigation to simplify development, provide lifecycle-aware components, and improve overall app performance.

3. Repository Pattern[32]: Implemented a repository layer to abstract the data source and provide a unified interface for data access. This allows easy switching between local and remote data sources and promotes code reusability.

```
data class Book(
   var id: String,
   var title: String,
   var author: String,
   var genres: List<String>,
   var coverUrl: String,
   var rating: Float,
   var price: Double,
   val purchaseDate: String,
   val quotes: MutableList<Quote>,
   var inWishList: Boolean,
   val user: String
)
```

Figure 6.1: Book Class properties.

6.3 Book class

The Book class is used in the application to store and manage information about books. It provides a structured representation of a book's details, including its metadata, cover image, rating, price, genres, purchase information, quotes, and user-related data. With the help of this class, the application can handle book-related operations such as displaying book details, managing wish lists, filtering books by genre, and more. The properties of the "Book" class shown in Figure 6.1 include:

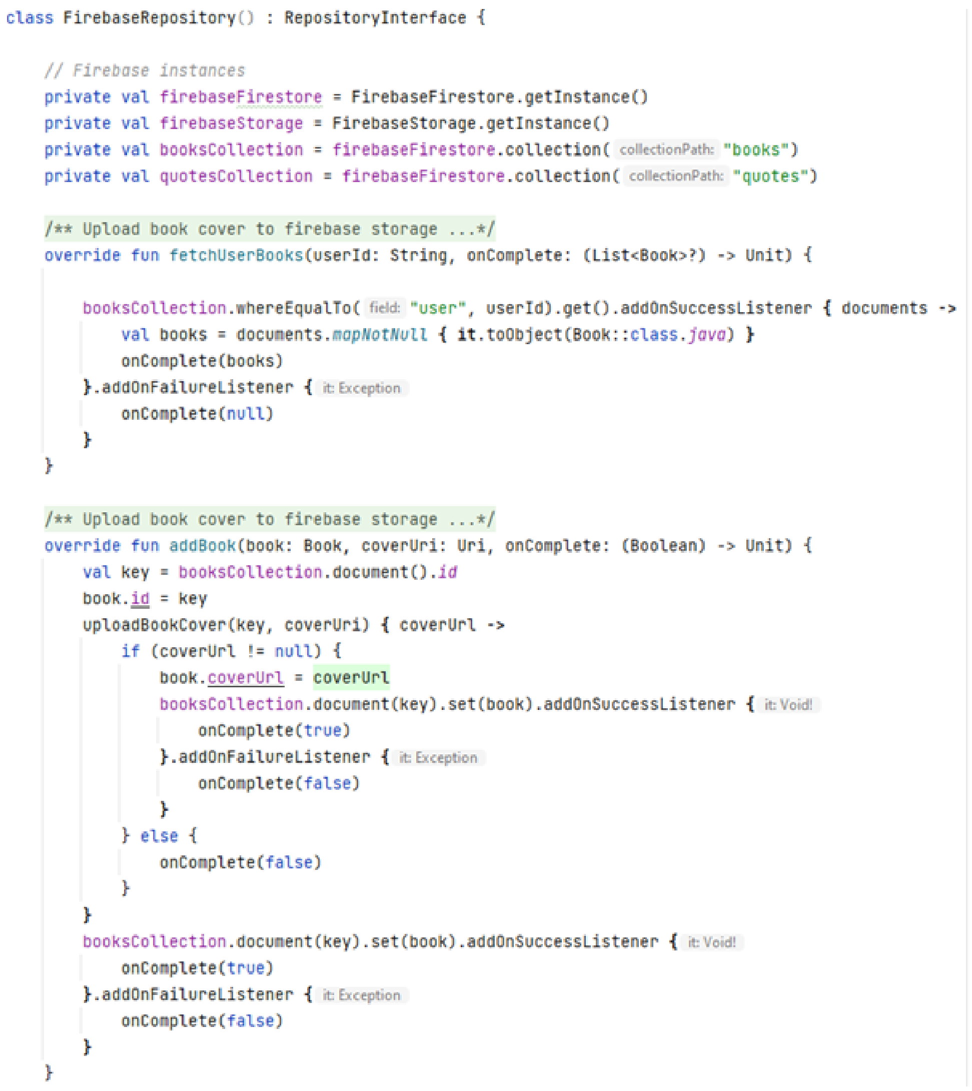
- id: A unique identifier for the book.
- title: The title of the book.
- author: The author of the book.
- genres: A list of genres associated with the book.
- coverUrl: The URL of the book's cover image.
- rating: The book's rating is represented as a floating-point number.
- price: The book's price is represented as a double.
- purchaseDate: The date of purchase for the book.
- quotes: A mutable list of Quote objects associated with the book.
- inWishList: A boolean value indicating whether the book is in the user's wish list.
- user: The user identifier associated with the book.

By defining the "Book" class as a data class, it automatically generates valuable methods such as equals(), hashCode(), and toString() based on the defined properties. This simplifies the process of working with book objects and allows for easy comparison and printing of book information.

6.4 FirebaseRepository class

The FirebaseRepository class uses instances of FirebaseFirestore and FirebaseStorage to interact with the Firebase services. It implements the methods defined in the RepositoryInterface interface to perform operations such as fetching user books, adding a new book with a cover image, updating a book (including a cover image), deleting a book, and fetching user quotes. Each method communicates with Firebase Firestore and Firebase Storage to perform the required actions and invokes the provided onComplete callback to notify the caller about the success or failure of the operation. The FirebaseRepository class in Figure 6.2 contains the following properties:

- firebaseFirestore: An instance of Firebase Firestore is used for accessing the Firestore database.
- firebaseStorage: An instance of Firebase Storage used for accessing the Firebase Storage service.
- booksCollection: A reference to the "books" collection in the
- Firestore database.
- quotesCollection: A reference to the "quotes" collection in the Firestore database.



```kotlin
class FirebaseRepository() : RepositoryInterface {

    // Firebase instances
    private val firebaseFirestore = FirebaseFirestore.getInstance()
    private val firebaseStorage = FirebaseStorage.getInstance()
    private val booksCollection = firebaseFirestore.collection( collectionPath: "books")
    private val quotesCollection = firebaseFirestore.collection( collectionPath: "quotes")

    /** Upload book cover to firebase storage ...*/
    override fun fetchUserBooks(userId: String, onComplete: (List<Book>?) -> Unit) {

        booksCollection.whereEqualTo( field: "user", userId).get().addOnSuccessListener { documents ->
            val books = documents.mapNotNull { it.toObject(Book::class.java) }
            onComplete(books)
        }.addOnFailureListener { it: Exception
            onComplete(null)
        }
    }

    /** Upload book cover to firebase storage ...*/
    override fun addBook(book: Book, coverUri: Uri, onComplete: (Boolean) -> Unit) {
        val key = booksCollection.document().id
        book.id = key
        uploadBookCover(key, coverUri) { coverUrl ->
            if (coverUrl != null) {
                book.coverUrl = coverUrl
                booksCollection.document(key).set(book).addOnSuccessListener { it: Void!
                    onComplete(true)
                }.addOnFailureListener { it: Exception
                    onComplete(false)
                }
            } else {
                onComplete(false)
            }
        }
        booksCollection.document(key).set(book).addOnSuccessListener { it: Void!
            onComplete(true)
        }.addOnFailureListener { it: Exception
            onComplete(false)
        }
    }
```

Figure 6.2: FirebaseRepository class.

fetchUserBooks: Retrieves a list of books associated with a specific user from the Firestore database. It takes a userId parameter to identify the user and an onComplete callback function that will be called with the retrieved list of books or null in case of failure.

addBook: Adds a new book to the Firestore database. It takes a book parameter representing the book object to be added, a coverUri parameter representing the URI of the book cover image, and an onComplete callback function that will be called with a boolean value indicating the success or failure of the operation.

Within the addBook function, a unique ID is generated for the book, and the book's cover image is uploaded to Firebase Storage using the uploadBookCover function (not shown in the provided code). Once the cover image is uploaded, the book object is updated with the cover URL and stored in the Firestore database. The onComplete callback is then called with a boolean value indicating the success or failure of the operation.

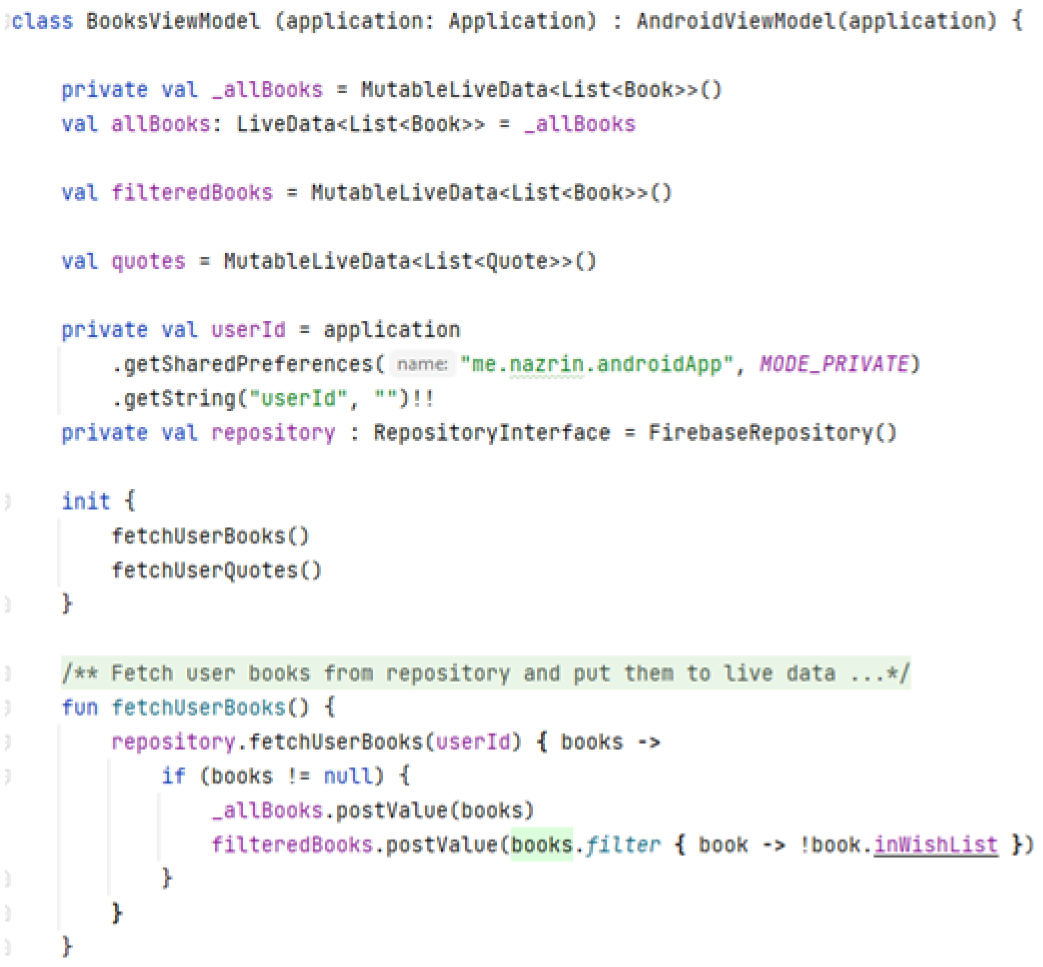
Overall, the FirebaseRepository class encapsulates the logic for interacting with Firebase Firestore and Firebase Storage to perform operations related to books, such as fetching user books and adding new books with cover images. It provides an abstraction layer between the app and the Firebase services, making it easier to manage book data and perform CRUD operations.

6.5 BooksViewModel class

The provided code snippet in Figures 6.3 and 6.4 represents a class called BooksViewModel which extends the AndroidViewModel class. It serves as the ViewModel for managing book-related data and operations in the application. The BooksViewModel acts as an intermediary between the UI components and the repository, providing data and functions for managing books in the application. It allows for observing and updating the book data in a lifecycle-aware manner. The class contains the following properties:

- _allBooks: A private MutableLiveData that holds a list of all books.
- allBooks: A public LiveData that exposes the _allBooks list for observing changes externally.
- filteredBooks: A MutableLiveData that holds a list of filtered books based on certain criteria.
- quotes: A MutableLiveData that holds a list of quotes.
- userId: A string variable representing the user ID obtained from SharedPreferences.
- repository: An instance of the RepositoryInterface (implemented by FirebaseRepository) used for data operations.

In the init block, the fetchUserBooks() and fetchUserQuotes() functions are called to populate the _allBooks initially and quotes LiveData with the respective data from the repository.



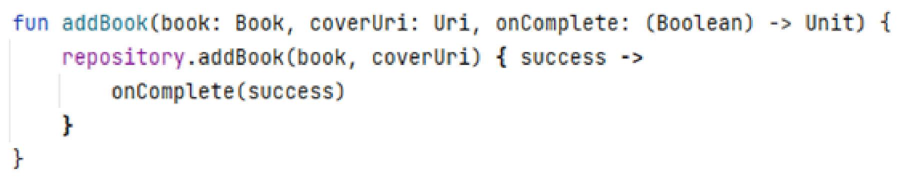
```
class BooksViewModel (application: Application) : AndroidViewModel(application) {

    private val _allBooks = MutableLiveData<List<Book>>()
    val allBooks: LiveData<List<Book>> = _allBooks

    val filteredBooks = MutableLiveData<List<Book>>()

    val quotes = MutableLiveData<List<Quote>>()

    private val userId = application
        .getSharedPreferences( name: "me.nazrin.androidApp", MODE_PRIVATE)
        .getString("userId", "")!!
    private val repository : RepositoryInterface = FirebaseRepository()

    init {
        fetchUserBooks()
        fetchUserQuotes()
    }

    /** Fetch user books from repository and put them to live data ...*/
    fun fetchUserBooks() {
        repository.fetchUserBooks(userId) { books ->
            if (books != null) {
                _allBooks.postValue(books)
                filteredBooks.postValue(books.filter { book -> !book.inWishList })
            }
        }
    }
```

Figure 6.3: BooksViewModel class.

The class provides the following functions:

- fetchUserBooks(): Retrieves the user's books from the repository and updates the _allBooks and filteredBooks LiveData accordingly. It uses the repository.fetchUserBooks() function, passing the userId and handling the response by updating the LiveData with the retrieved books.
- addBook(book: Book, coverUri: Uri, onComplete: (Boolean) -> Unit): Adds a new book to the repository. It uses the repository.addBook() function, passing the book object and cover URI, and handles the response through the onComplete callback.

The BooksViewModel acts as an intermediary between the UI components and the repository, providing data and functions for managing books in



```
fun addBook(book: Book, coverUri: Uri, onComplete: (Boolean) -> Unit) {
    repository.addBook(book, coverUri) { success ->
        onComplete(success)
    }
}
```

Figure 6.4: addBook function.

the application. It allows for observing and updating the book data in a lifecycle-aware manner.

Chapter 7

Testing of the "Libget" application

This chapter describes all the testing conducted when the "Libget" application was being developed.

7.1 Developer tests

Continuous testing was done throughout the development process. The physical device Samsung Galaxy A53 with Android 13 and the virtual Pixel 2 API 30 with Android 11 have been tested. Through this testing method, a significant number of bugs were detected and resolved.

7.2 User tests

User testing was conducted following the conclusion of the primary development phase. It is crucial to involve actual prospective users in the testing of the application to answer important questions:

- Is the application's usage instructions clear?
- Does the application work as the user would expect it to?

- What should be/needs to be included in the application?
- Will the user make use of the program?

The application was tested by three participants who had previously tested the prototype and wireframes. They were given instructions described in Tables 7.1 and 7.2. On the left side of the tables are the steps of the instructions, and on the right are the actual results. The "Libget" application was tested and installed on one mobile phone and tested by the participants individually. It was done in person, and as the participants followed each step, I recorded the actual result and combined the steps and result in Tables 7.1 and 7.2.

7.2.1 Test cases

Tables 7.1 and 7.2 describe two test cases used to find application issues and their corresponding results. The TC1 in Table 7.1 covers functionalities such as registration, logging in and out, adding a new book to "All," adding a new book to "Wish List," viewing the Wish list, changing the book details, adding a quote, viewing the Quotes page, and deleting the quote. The TC2 in Table 7.2 covers functionalities such as sorting books by price and rating, filtering books by genre, opening the statistics page and checking its correction, and setting the limit.

The overall feedback from these potential users was positive, and "Libget" seems to be a valuable tool.

Several bugs that occurred during testing have been resolved:

1. To add a book to the "Wish List" on the application, it was necessary to fill out all the fields. However, only the "name," "author," and "price" fields are essential.
2. When moving a book from the "All" section to the "Wish List" by checking "Wish List" in the book details, an error notification appears, although the book is added to the "Wish List."
3. After changing any detail of a book in the "Wish List," the book moves to the "All" section.

4. It was not possible to move a book from the "Wish List" to the "All" section.
5. Books from the "Wish List" were also included in the statistics, although only bought books must be there.
6. After pressing the "Delete" button on a quote, it only disappears after reopening the "Quotes" section.
7. It was not possible to change the date of the book purchase and the date is not displayed in the book details.
8. After sorting books in the "All" section, books from the "Wish List" section were also displayed, which is not intended.

Due to their complexity, my little development experience, and time limitations, some bugs were unfortunately not eliminated. Those bugs are the following:

1. Sometimes, after making changes in the sections("All," "Wish List," "Quotes"), the changes do not appear at once. Switching to a different section and returning to the previous one is necessary.
2. Two notifications appear consecutively when adding or updating a new book - an error message followed by a success message. Just one of them should occur depending on the situation.
3. In the detail of a book, a small top part of the number in the "price" field is hidden. The number remains easy to understand.
4. There is a glitch when switching from the "All" section to the "Quotes" section - book covers are visible for a quick second and then disappear.
5. When changing a book's cover, there is no preview of the photo.

No	Steps	Actual results
1.	Click the "Start" button.	Registration page opens.
2.	Fill in "Email" naza@cvut.cz and "Password" 123456.	Filled.
3.	Fill in "Confirm password" 123457 and click "Sign up."	Error "Password and confirm password" should be the same" occurred.
4.	Fill in "Confirm password" 123456 and click "Sign up."	Main app page opens.
5.	Click the "+" icon to add a book.	Opens the page for adding a book. Two notifications appeared consecutively - an error message followed by a success message.
6.	Fill in all fields, and select cover. Click two times to add the date and press "Add."	Page with all books opens.
7	Click "Wish List" and then "All."	The new book appeared.
8.	Click on the new book.	All book details are valid.
9.	Write a quote and press "Add quote."	"Quote successfully added" notification appeared.
10.	Change all fields and then click "Save."	Page with all books opens.
11.	Open the book again.	All new details were saved.
12.	Click the "+" icon to add a book.	Opens the page for adding a book.
13.	Fill in the name, author, and price fields, select a genre, check "Wish List," and press "Add."	Error "Fill all field" occurred.
14.	Click "Quotes"	Page with the new quote appeared.
15.	Click the "Delete" icon to delete the quote.	The quote was not deleted.
16.	Click "All" and then on the new.	Page with book details opens.
17.	Click the "Delete" button.	The book was deleted.
18.	Click the "Log out" icon.	Page for login opens.
19.	Log in again using the given email and password.	Login was successful.

Table 7.1: TC1 steps and actual results.

No	Steps	Actual results
1.	Click "Start."	Registration page opens.
2.	Log in using data from the previous test.	Main app page opens.
3.	Add three books with names 1, 2, 3 in different months - in this previous and before previous, with prices 100, 200, 300 and rating 1, 2, 3 and genres Art, Children, Classics.	All book data is valid.
4.	Sort by price ascending.	Books were sorted correctly.
5.	Sort by price descending.	Books were sorted correctly.
6.	Sort by rating ascending.	Books were sorted correctly.
7.	Sort by rating descending.	Books were sorted correctly.
8.	Filter by genre Art.	Only books in the art genre showed.
9.	Filter by genre Children.	Only books in the children's genre showed.
10.	Filter by genre Classics.	Only books in the classic genre showed.
11.	Click the "Delete" icon to delete the quote.	The quote was deleted.
12.	Go to statistics by pressing the "Statistics" icon.	Statistics for the last thirty days(in the application shown as "1 month") displayed.
13.	Set limit.	Limit was set.
14.	Select "3 months".	A diagram for the last three months appeared with the correct statistics.
15.	Select "12 months".	A diagram for the last twelve months appeared with the correct statistics.

Table 7.2: TC2 steps and actual results.

Chapter 8

Improvements and new features in the future

The “Libget” application has all the basic expected features - it helps with book organization, expense management, and saving quotes and notes(functional requirements with priority 1). However, many things can still be improved or added to the application. The list of improvements is based on the functional requirements with priority 2 and 3, listed in Chapter 4.1, and users’ feedback from testing the application.

1. There is an option to edit quotes.
2. The application enables taking a picture of a text and adding it to the quotes section.
3. The application enables marking a book as TBR(To be read), Started, or Read.
4. The application enables filtering by country, TBR(To be read), Started, or Read.
5. The application enables marking a book as borrowed/unborrowed from someone or to someone (date, lender’s name).
6. The application enables search by ISBN code.
7. The application enables the calculation of reading speed.
8. The application enables setting daily, monthly, and yearly goals.

9. The application enables tracking progress reading books, last read page number (through the interface)
10. The application enables the conversion of the audio format into pages.
11. The application enables a machine-learning AI that understands users' reading preferences and recommends books.
12. The application enables to time the user's reading time.
13. The application enables searching a book by any word crammed into the information about the book.
14. The application enables finding books on the Internet.
15. The application enables sharing book lists.
16. The application enables synchronizing devices.
17. The application enables the creation of literary forums.
18. The application enables following book news.

After discussing with potential users, this list of functionalities was deemed sensible. The current version of the application boasts immense potential and showcases a strong and reliable foundation.

Chapter 9

Project management of the thesis

I worked on this thesis for two semesters. The first semester I gathered the needed information and analyzed existing applications, defining the core of the "Libget" application and its functionalities. The next semester was focused on designing, implementing, and testing the application. Creating a highly complex application for the first time proved to be a challenging and demanding task. Although I am not a developer, I put in my best effort to develop a quality application. Once every two weeks, I consulted with my supervisor to discuss the progress and problems along the way in writing the thesis and developing the application.

The approximate number of hours spent on each part of the thesis:

- Analysis and Research - 200 hours.
- Design - 50 hours.
- Implementation - 300 hours.
- Testing - 100 hours.

Based on the breakdown of hours spent, it is evident that my proficiency level in development is not at an advanced level.

Chapter 10

Conclusion

The "Libget" application, which allows for organizing a personal library, budgeting for books, and saving quotes or notes, was the primary focus of this bachelor's thesis. Its analysis, design, implementation, and testing were all undertaken.

Through extensive market research, it became apparent that no existing applications could completely incorporate all necessary concepts. Consequently, "Libget" was created to fill this gap in the market.

The analysis of the "Libget" application included writing functional and non-functional requirements and creating the class diagram and use cases.

The next step was designing and implementing the application. The initial version was created in Kotlin for Android and met all functional requirements with priority 1 and non-functional requirements.

Finally, the testing phase with potential users helped detect various bugs while also receiving valuable feedback to incorporate into future improvements. The user testing revealed that the "Libget" application seemed beneficial for future users. I believe the "Libget" application has a lot of potentials, even though it is still in its early phases, and this is its most basic form. As a result, the "Libget" application can become quite helpful and in high demand. Although I am grateful for this experience, I do not plan on pursuing further development of this application. As I am not a developer, the process of creating and implementing this application proved to be quite challenging for me.

Overall, the thesis successfully achieved its goals and provided a strong foundation for future development. This work was beneficial for me; as a person very new to development, it was an exciting challenge. Also, the application concept is relevant to my life. Furthermore, I became familiar with the application development process and gained knowledge of technologies I have not or rarely used in my studies.

Bibliography

[1] *Mobile application and its Global Impact – Researchgate* [online]. [Accessed 15 January 2023]. Available from: https://www.researchgate.net/profile/Dr-Md-Rashedul-Islam/publication/308022297_Mobile_application_and_its_global_impact/links/5991fbafa6fdcc53b79b606d/Mobile-application-and-its-global-impact.pdf.

[2] *Book tracker – android apps on Google Play. Google* [online]. [Accessed 15 January 2023]. Available from: https://play.google.com/store/search?q=book+tracker&amp;c=apps.

[3] *Storygraph - apps on Google Play (2021). Google* [online]. [Accessed 15 January 2023]. Available from: https://play.google.com/store/apps/details?id=com.thestorygraph.thestorygraph&hl=en&gl=US.

[4] *Plus: The storygraph. Plus | The StoryGraph* [online]. [Accessed 15 January 2023]. Available from: https://app.thestorygraph.com/plus.

[5] *Read more: A reading tracker – apps on Google Play (2018). Google* [online]. [Accessed 15 January 2023]. Available from: https://play.google.com/store/apps/details?id=com.shunan.readmore&hl=en&gl=US.

[6] *Bookly: Book & reading tracker – apps on google play. Google* [online]. [Accessed 15 January 2023]. Available from: https://play.google.com/store/apps/details?id=com.twodoor.bookly.

[7] *GAMES, TwoDoor. Bookly: TBR &; Book tracker. App Store* [online] 7 December 2016. [Accessed 15 January 2023]. Available from: https://apps.apple.com/us/app/bookly-tbr-book-tracker/id1085047737?see-all=reviews.

[8] *Bookshelf-your virtual library - apps on Google Play (2019). Google* [online]. [Accessed 15 January 2023]. Available from: https://play.google.com/store/apps/details?id=com.bookshelf.prod&hl=en&gl=US.

[9] *Appadvice (2022) Bookshelf-your virtual library, AppAdvice* [online]. [Accessed 15 January 2023]. Available from: https://appadvice.com/app/bookshelf-your-virtual-library/1464032274.

[10] *My library - apps on Google Play (2016). Google* [online]. [Accessed 15 January 2023]. Available from: https://play.google.com/store/apps/details?id=com.vgm.mylibrary&hl=en&gl=US.

[11] *Handy Library - book organizer - apps on google play (2018). Google* [online]. [Accessed 15 January 2023]. Available from: https://play.google.com/store/apps/details?id=com.handylibrary.main&hl=en&gl=US.

[12] Functional vs non-functional requirements [updated 2021]. *Enkonix* [online]. [Accessed 15 January 2023]. Available from: https://enkonix.com/blog/functional-requirements-vs-non-functional/.

[13] *Storage* [online]. [Accessed 15 January 2023]. Available from: http://ai2.appinventor.mit.edu/reference/components/storage.html#TinyDB.

[14] Class diagrams. *in UML modeling* [online]. [Accessed 15 January 2023]. Available from: https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams.

[15] Use-case diagrams. *in UML modeling* [online]. [Accessed 15 January 2023]. Available from: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case#:~:text=In%20UML%2C%20use%2Dcase%20diagrams,the%20system%20and%20its%20actors.

[16] *Applying UML and patterns - The University of Texas at Dallas* [online]. [Accessed 12 May 2023]. Available from: https://www.utdallas.edu/~chung/SP/applying-uml-and-patterns.pdf?ref=driverlayer.com/web

[17] *What is wireframing.* [online]. [Accessed 12 May 2023]. Available from: https://www.experienceux.co.uk/faqs/what-is-wireframing/.

[18] Android Developers. *Android Mobile App Developer Tools – Android Developers* [online]. [Accessed 1 March 2023]. Available from: https://developer.android.com/.

[19] *Kotlin Programming Language* [online]. [Accessed 1 March 2023]. Available from: https://kotlinlang.org/.

[20] *XML Essentials - W3C* [online]. [Accessed 1 March 2023]. Available from: https://www.w3.org/standards/xml/core#:~:text=What%20is%20XML%3F,more%20suitable%20for%20Web%20use.

[21] Gradle (2023) *Gradle Build Tool* [online]. [Accessed 1 March 2023]. Available from: https://gradle.org/.

[22] *Configure your build* [online]. [Accessed 10 March 2023]. Available from: https://developer.android.com/studio/build.

[23] Porter, B. *Maven – Welcome to Apache Maven* [online]. [Accessed 10 March 2023]. Available from: https://maven.apache.org/.

[24] *The collaborative interface design tool* [online]. [Accessed 29 April 2023]. Available from: https://www.figma.com/.

[25] *Firebase realtime database Google* [online]. [Accessed 11 May 2023]. Available from: https://firebase.google.com/docs/database.

[26] *How to write test cases (with format & example) (2023) BrowserStack* [online]. [Accessed 12 May 2023]. Available from: https://www.browserstack.com/guide/how-to-write-test-cases#:~:text=Writing%20Test%20Cases-,What%20is%20a%20Test%20Case%3F,necessary%20to%20verify%20a%20feature.

[27] Michaelstonis *Model-view-viewmodel, Model-View-ViewModel / Microsoft Learn* [online]. [Accessed 18 May 2023]. Available at: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm.

[28] *What is User Interface (UI) design? (2023) The Interaction Design Foundation* [online]. [Accessed 18 May 2023]. Available at: https://www.interaction-design.org/literature/topics/ui-design.

[29] *Advantages of XML* [online]. [Accessed 18 May 2023]. Available from: https://www.ibm.com/docs/en/i/7.3?topic=introduction-advantages-xml.

[30] *What is Android Jetpack, and why should we use it? MindOrks* [online]. [Accessed 18 May 2023]. Available at: https://blog.mindorks.com/what-is-android-jetpack-and-why-should-we-use-it/.

[31] *Java.com* [online]. [Accessed 18 May 2023]. Available from: https://www.java.com/.

[32] *Repository pattern: android developers Android Developers* [online]. [Accessed 18 May 2023]. Available at: https://developer.android.com/codelabs/basic-android-kotlin-training-repository-pattern#0.

[33] *Data Access Object Design Patterns: Data Access Object.* [online]. [Accessed 18 May 2023]. Available at: https://www.oracle.com/java/technologies/data-access-object.html#:~:text=The%20Data%20Access%20Object%20(or,to%20a%20generic%20client%20interface.

[34] Carty, D. (2020) *Follow Google's lead with Programming Style Guides: TechTarget, Software Quality* [online]. [Accessed 18 May 2023]. Available at: https://www.techtarget.com/searchsoftwarequality/feature/Follow-Googles-lead-with-programming-style-guides#:~:text=A%20style%20guide%20tells%20a,and%20bad%20programming%20behavior%20explicit.

[35] Babich, N. (2023) *Low fidelity vs. high fidelity: The differences between design prototypes: Webflow blog, Webflow* [online]. [Accessed 23 May 2023]. Available at: https://webflow.com/blog/low-vs-high-fidelity#:~:text=What%20is%20high%20fidelity%3F,the%20final%20product%20as%20possible.

Appendix A

Acronyms

- MVVM - Model-View-ViewModel
- UI - User Interface
- XML - Extensible Markup Language
- DAO - Data Access Object
- IDE - Integrated development environment
- IOS - iPhone operating system
- SDK - A Software Development Kit

Appendix B

Electronic attachments

In the electronic attachments, there are the following files:

- User manual for the "Libget" application: User Manual.pdf
- Source code of the "Libget" application: LibgetApp.zip
- High fidelity prototype in Figma: Prototype.png
- Full text of this bachelor work: Bachelor Thesis Nazrin Orujaliyeva.pdf