Bilkent University

Department of Computer Engineering

Senior Design Project

High-Level Design Report

Project Name: BookClub

Group Members: Bikem Çamli
Mert Osman Dönmezyürek
Barış Eymür
Mahin Khankishizade
Deniz Şen

Supervisor: Assoc. Prof. Selim Aksoy

Jury Members: Prof. Dr. Fazlı Can
Assoc. Prof. Cigdem Gunduz Demir

Innovation Expert: Dr. Haluk Altunel

High-Level Design Report

Dec 31, 2018
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2

Table of Contents

1. Introduction
1.1 Purpose of the system 3
1.2 Design goals 3
1.2.1 Usability 3
1.2.2 Scalability 4
1.2.3 Extensibility 4
1.2.5 Reliability 4
1.2.6 Availability 5
1.2.7 Security 5
1.3 Definitions, acronyms, and abbreviations 5
1.4 Overview 5

2. Current software architecture 6
2.1 ThriftBooks 6
2.2 Amazon 6
2.3 GoodReads 7
2.4 Hepsiburada 7

3. Proposed software architecture 8
3.1 Overview 8
3.2 Subsystem decomposition 9
3.4 Persistent data management 11
3.5 Access control and security 11
3.6 Global software control 12
3.7 Boundary conditions 13
3.7.1 Starting the application 13
3.7.2 Terminating the application 14
3.7.3 Failure in the application 14

4. Subsystem services 14
4.1 Client 14
4.1.1 View 15
4.1.1.1 List Handler 15
4.1.1.2 User Data Handler 16
4.1.1.3 View Manager 17
4.1.2 Connection Manager 17
4.2 Server 17
4.2.1 Controller 18
4.2.1.1 Main Manager 18
4.2.1.2 Service Manager 18
4.2.1.3 Database Manager 19
4.2.1.4 File Manager 19
4.2.1.5 Algorithm Manager 19
4.2.1.6 Connection Manager 19

5. Glossary 19

6. References 20

1. Introduction

1.1 Purpose of the system

Reading is a fun and useful activity: we can learn new things by reading a scientific book or we can find ourselves in a totally new fictional world by reading a novel. Unfortunately, every beautiful thing comes with a price. Wouldn’t it be nice to exchange our books with our friends’ books, instead of paying? If we did that, we would have lessened the price of reading significantly. There are groups of people around the world who have noticed the advantages of book sharing. These people have formed book sharing groups, called book clubs.

The main aim of this project is to create a book trading platform which will serve as a universal “BookClub”. The users of this mobile app will be able to exchange their used books with books of other users and they will also be able to buy second-hand books from other users by paying a reasonable price. The users will specify the books that they desire to give away or sell and the ones they want to read. Using this information, BookClub will match users with the owners of books they are looking for. While doing this, it will find the best possible match for a user by considering user locations, user ratings, book prices and some other attributes. Additionally, it will make personal suggestions to its users by trying to understand their taste of books.

1.2 Design goals

1.2.1 Usability

- The user base of BookClub is very versatile therefore it needs to be usable and understandable for all the different user types.
- The application should include an explicit user manual which consists of information about the usage of BookClub and the details about how matchings and suggestions are made by the system.

1.2.2 Scalability

- As the number of people increases, the application should meet the increased upcoming demands.
- The system should be capable enough to be able to store the books that are newly added by the users.

1.2.3 Extensibility

- We will be taking the users' comments and reviews into consideration to improve and shape the system accordingly.
- The application will constantly get updates and additional features to keep up with the users' wishes and new trends.

1.2.4 Efficiency

- The system should efficiently handle an increasing number of users, thus requests.
- The system can have a high number of books, therefore, the application should match the users in the best way and as quick as possible.
- Book authentication during the procedure of adding a new book should be as minimal as possible.

1.2.5 Reliability

- The matching algorithm must give the best match regardless of the outer factors and users should be able to rely on the matches that BookClub gives.
- Unless there is a match, the system should produce the best suggestions to the user based on their likings and preferences.
- The application must have no pirate books.
- The second-hand book's price should not exceed the price of the first-hand book.

1.2.6 Availability

- BookClub should be downloadable through Google Play Store.

- The system should be available for user access every time.

1.2.7 Security

- The system should protect the personal information of the user such as location, email, password, recent sales and search history.

1.3 Definitions, acronyms, and abbreviations

API: Application Programming Interface
HTTP: HyperText Transfer Protocol
SQL: Structured Query Language
MVC: Model View Controller

1.4 Overview

BookClub is an innovative mobile application which is available on Google Play Store for free. BookClub provides a field where users can exchange books with each other easily thanks to its unique match/suggestion algorithm, unlike other book sharing and exchanging applications/websites. The users log in/register to the app and create lists of the books they are willing to use as trade materials and the books they desire to obtain. Then, the user will be matched with other users thanks to BookClub's matching algorithm. In this situation, by clicking the match screen, they will see the list of matches, the location of the books, the owners and their reviews. They can easily contact the user through the application's chatting platform and arrange a meeting where they will be able to exchange books. This is a remarkable advantage for both sides since the users not worry about the safety of the book or its condition by viewing the owner's review and reaching them. No shipment price, no worries about the reliability of the other user, no pirate versions and no black market. Moreover, the users can sell their books in BookClub, yet, the price has a limit, which is the price of the first-hand price of that book.

There are also cases of not getting the best match, in this case, BookClub's innovative suggestion algorithm offers users new books that they may like, by taking their search history and their book wishlist, into account. The suggestion algorithm works as follows: think of a case where a user does not need any new book, but they have some books that they would like to trade away. In this scenario, BookClub analyzes the user's books that they want to give away and examines their similarities. Later, in the main screen, the user will be served some book suggestions similar to their likings.
Overall, BookClub is a book lovers' platform that makes trading books easier and effortless thanks to its innovative matching and suggestion algorithm. Besides, it will give them a chance to get their desired books for a much cheaper price or even for free. With the original and fast match/suggestion algorithm it will decrease both the black market sales and book piracy, at the same time, making book trading a delightful experience for the users.

2. Current software architecture

In this part of the report, current software architectures similar to BookClub will be explained.

2.1 ThriftBooks [1]

- Users buy both used and new books.
- Users can make review other users.
- Users can make comment on books.
- Users should pay a shipping price.

2.2 Amazon [2]

- Users can sell their used and new books.
- Amazon has a product suggestion system however it is not specific for books.
- Users can comment and rate other users.

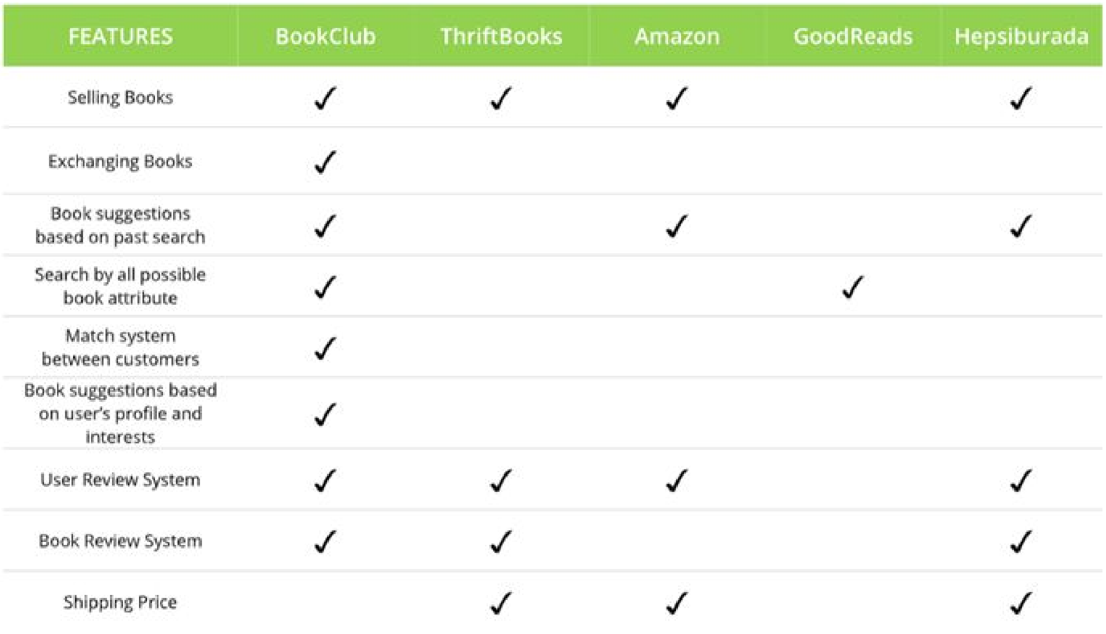
- Users may need to pay a shipping price for some books.

2.3 GoodReads [3]

- GoodReads has a wide range of books for book lovers.
- Users can filtered search for books by different book attributes.
- Users can recommend their friends books.

2.4 Hepsiburada [4]

- Users can sell both their new and second-hand books.
- Users can get product recommendation based on their previous purchases.
- Users can comment and rate both users and products.
- Some products may need a shipping price.



FEATURES	BookClub	ThriftBooks	Amazon	GoodReads	Hepsiburada
Selling Books	✓	✓	✓		✓
Exchanging Books	✓				
Book suggestions based on past search	✓		✓		✓
Search by all possible book attribute	✓			✓	
Match system between customers	✓				
Book suggestions based on user's profile and interests	✓				
User Review System	✓	✓	✓		✓
Book Review System	✓	✓			✓
Shipping Price		✓	✓		✓

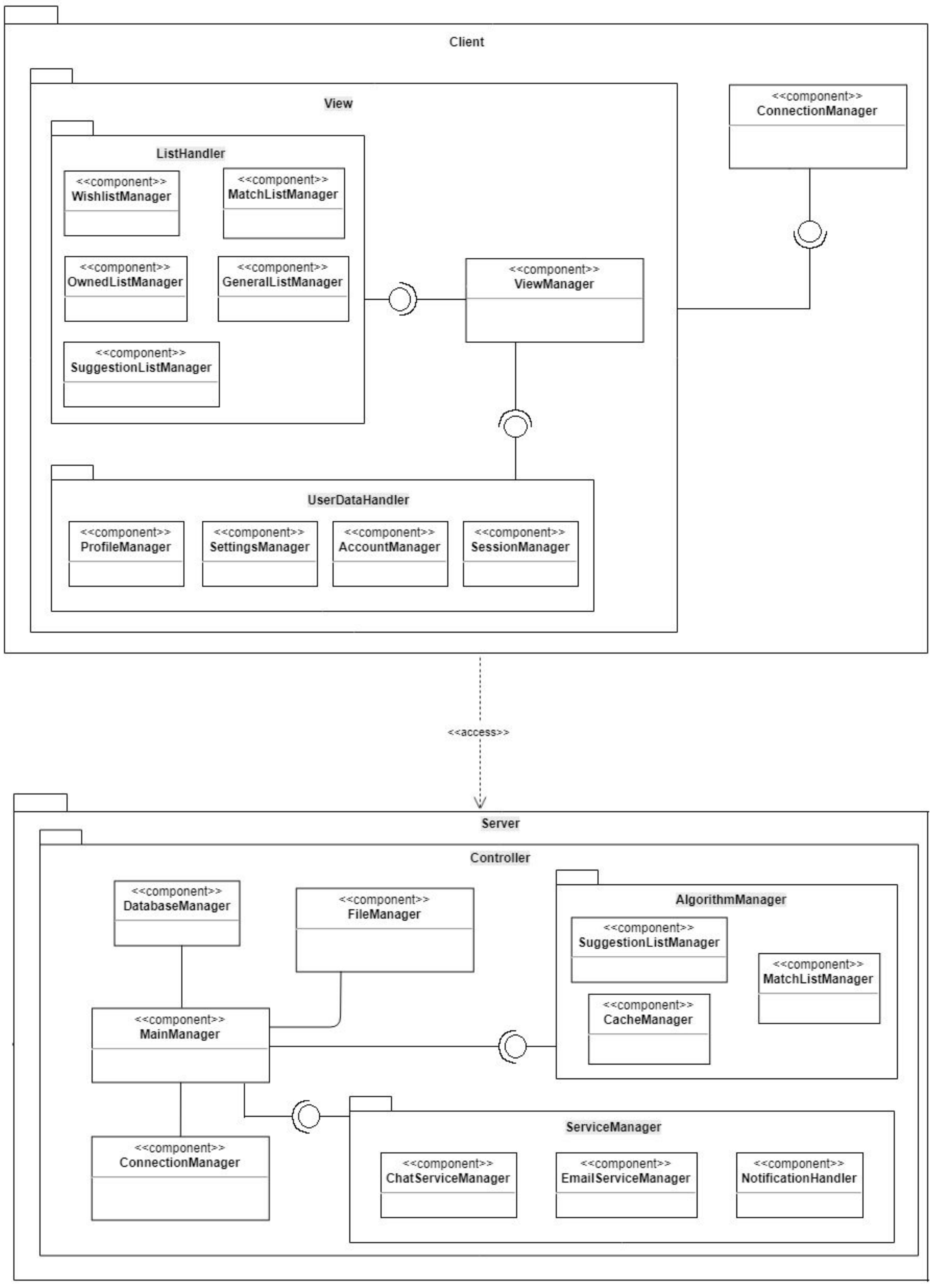
Figure: Comparison between platforms

3. Proposed software architecture

3.1 Overview

In this part of the report, we define the proposed software architecture of the BookClub in a detailed way. In the subsystem decomposition part, the architecture design of the BookClub presented with a decomposition diagram and explained detailly. We showed hardware/ software mapping of the BookClub in the third part. In the persistent data management part, we give information about data storage in BookClub. In access and security part we mention the things we do to keep our system secure. In the sixth part, we explain our global software control which is event-driven control system. In the last part we clarify our boundary conditions in terms of Starting the application, terminating the application and failure in the application.

3.2 Subsystem decomposition



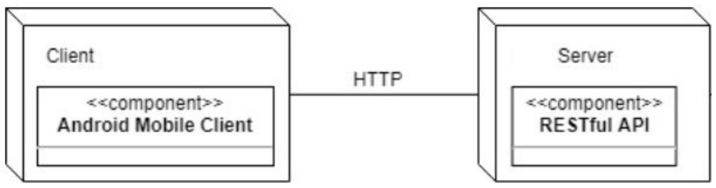
The system architecture of BookClub can be described as a Client / Server architecture. We chose to implement our platform using this architecture because we want BookClub to be able to deal with a large number of users without any deteriorations in performance. Thanks to our server, BookClub will be able to serve many users (clients) in a fast and successful way. We use the server for many purposes. We store data about books, users and user book lists in the server. This will give us a significant opportunity to keep data safe. Also, the user match algorithm will be executed in the server side to generate matches and suggestions for users. Moreover, the server side will be responsible for performing most of the computations. The Android app, which will be the client side, will perform basic operations. It will connect to the server, request relevant information from the server and display it to the user. Then, the users will be able to make their own choices, such as accepting or rejecting a match, etc. The data in the server side will be updated according to the choice being made on the client side. With this architectural structure, BookClub will be able to work in a fast and efficient way and continue to perform its function without any deteriorations, even if the number of users increases significantly.

In BookClub, we also use Model-View-Controller (MVC) design pattern. We distributed parts of MVC between client and server sides. Client-side contains View, which contains classes about user interface and user interaction.

Server-side contains Model (user and book data that is stored in a database) and Controller (which acts as a bridge between Model and View by coordinating and manipulating the user input in the application).

The users of BookClub will interact with the View part of the Client and this will be the only part of the application that the users will be able to directly interact with. The users' input will be taken through the view part. They will be sent to the server including the data and the user’s request via ConnectionManager class in the Client.

3.3 Hardware/software mapping



In our application, the clients are Android Mobile Device users. Since the server uses the location of users to find the best match via matching algorithm, it requires this data from the clients. Clients will get this information by accessing the devices’ GPS features. Obviously, all the data manipulation and data storage are done in server-side so Android device needs the internet connection. The server and the client will communicate over HTTP by utilizing RESTful API.

3.4 Persistent data management

Data storage is one of the significant issues in our application since BookClub is a user-oriented platform which contains a lot of users and their data. Data management is the other side of the data problems since we need to manipulate and manage the data efficiently. To solve these challenges, we will generate a relational database by using MySQL. All users’ private info such as ID, password, email, book interests or genre preferences; their book lists such as wishlist or owned books, and most importantly the data derived from the users’ preferences by the algorithm will be kept in the database.

3.5 Access control and security

The system will be secure in the sense that it will not require too much personal information. To register to BookClub, one has to give a valid e-mail address which they have to confirm through an e-mail verification system and a confidential password. The passwords will be kept in the system’s database and will be encrypted with several hashing mechanisms. If the user forgets their password, the user will be able to change their BookClub password by giving their e-mail address. Then, the system will send an option to change the password to the respective e-mail address. Other than the password, the system will only allow users to change their own information while they are logged in to the system. Furthermore, to log in to the system, the users must use their personal e-mail addresses or usernames along with their passwords.

The system will share personal information with neither other users nor third-parties. The information flow will be done only between the users themselves through the chatting service. Also, the chatting service will only be available for usage in case of a desire of trade between 2 users.

The data manipulation will not be let to the client side as it can produce severe security-related problems. Instead, these manipulations will be done in the server-side which will be explained later in the report. The client application will only be responsible for and sending requests to the server and showing the responses of it.

3.6 Global software control

For global software control in BookClub, we used event-driven control system. Our system is controlled by the user input.
When a user login to the system, the inputs of the user checked by the Database Manager in the server. If the user information is valid, the system allows the user to log in.

When the user creates/changes her/his wishlist View Manager will be notified and from List Handler Package the WishListManager will modify the wishlist of the user and with connection managers, a notification will be created in MainManager of server side. Database Manager will update the database according to this notification. When the wish list of the user changed and a related notification will be created in MainManager, SuggestionListManager and MatchListManager will update the suggestion and match lists.

When the user wants to change his/her profile, a notification will be created in ViewManager. From UserDataHandler the ProfileManager will update the profile view of the user. In conjunction with the view side, with connection managers, a notification will be created in MainManager of server-side and Database Manager will update the database according to this notification. When the user wants to change his/her account settings such as premium options or privacy settings, a notification will be created in ViewManager. From UserDataHandler the AccountManager will update the account settings of the user. If the user wants to change his/her settings a notification will be created in ViewManager. From UserDataHandler the SettingsManager will update the account settings of the user. At the same time with view side, with connection managers, related notifications will be created in MainManager of server-side and Database Manager will update the database according to this notification.

When the user wants to search for a book, a notification will be created in ViewManager. From ListHandler the GeneralListManager will update the page view and return the searched results to the user.

3.7 Boundary conditions

3.7.1 Starting the application

BookClub is an Android application, therefore, it is required to be first downloaded from the Google Play Store and installed to an Android-based device. Then the user has to create an account to be able to use BookClub. They can then sign in using their credentials and start using the application.

3.7.2 Terminating the application

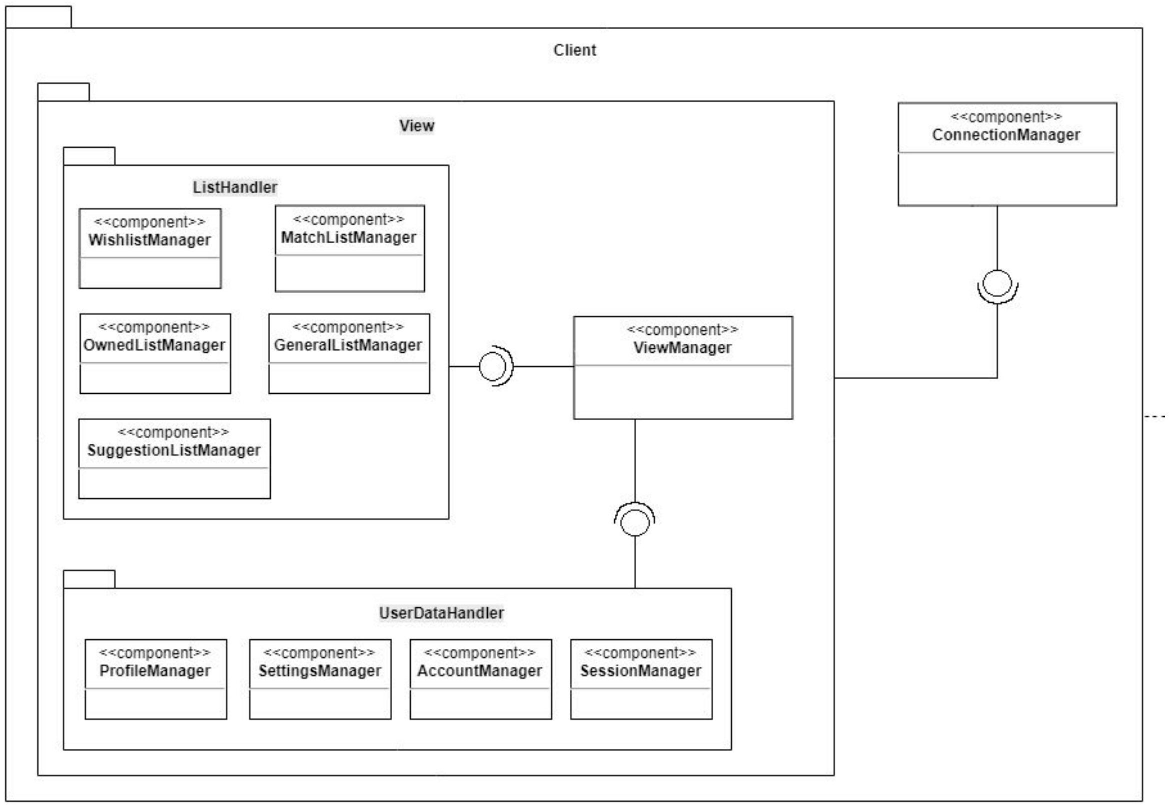
The user can log out from the system by using the “Log Out” button which can be found at the bottom of the settings screen. If the user does not want to log out from the system but also not to use the application they can terminate the application from the Android’s task manager.

3.7.3 Failure in the application

The application strictly requires internet connectivity and it does not have an offline interface. In case of a loss in internet connection while the application is being used, the user will be notified with an error and the application will be terminated.

4. Subsystem services

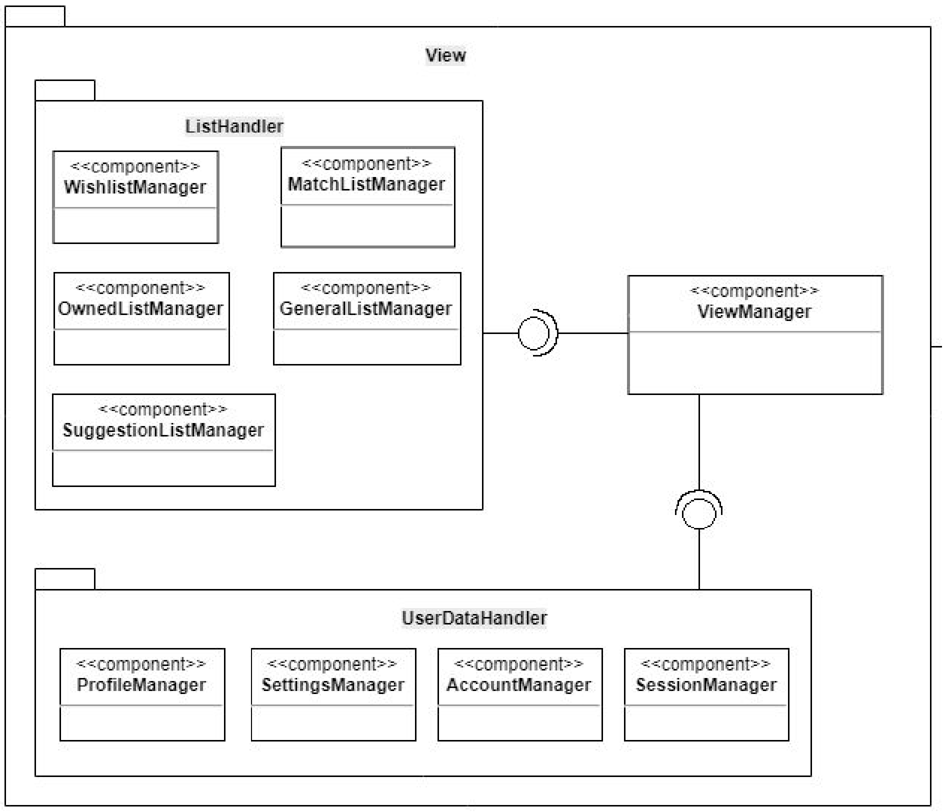
4.1 Client



Client
View
ListHandler
<<component>> WishlistManager
<<component>> MatchListManager
<<component>> OwnedListManager
<<component>> GeneralListManager
<<component>> SuggestionListManager
<<component>> ViewManager
<<component>> ConnectionManager
UserDataHandler
<<component>> ProfileManager
<<component>> SettingsManager
<<component>> AccountManager
<<component>> SessionManager

The client side of the system is responsible for managing the interaction of the BookClub with the user. Since BookClub will be implemented as an Android mobile application the client side corresponds to the mobile application. Client-side is in charge of the presentation of the data in server side to the user. In addition, any changes of data that user will be made, will be done by the connection of client-side with the server side.

4.1.1 View


View
ListHandler
<<component>> WishlistManager
<<component>> MatchListManager
<<component>> OwnedListManager
<<component>> GeneralListManager
<<component>> SuggestionListManager
<<component>> ViewManager
UserDataHandler
<<component>> ProfileManager
<<component>> SettingsManager
<<component>> AccountManager
<<component>> SessionManager

4.1.1.1 List Handler

List Handler layer is responsible for presenting and managing the list of the user. It includes 5 classes that handle 5 different list types in BookClub.

WishListManager: This class manages the wish list of the user. When the user wants to change her/his wishlist this class will notify the connection manager in the client side and the connection manager with the server layer.

MatchListManager: This class eases the adaptation of the match lists on the screen which is done mainly thanks to ViewManager, also notifies the ViewManager to send requests to the ConnectionManager on the outer package.
OwnedListManager: This class eases the adaptation of the owned books' lists on the screen of ViewManager, also notifies the ViewManager to send requests to the ConnectionManager on the outer package.
GeneralListManager: This class eases the adaptation of the owned books' lists on the screen of ViewManager, also notifies the ViewManager to send requests to the ConnectionManager on the outer package.
SuggestionListManager: This class eases the adaptation of the lists of the suggested books on the screen, also notifies the ViewManager to send requests to the ConnectionManager on the outer package.

4.1.1.2 User Data Handler

User Data Handler layer is responsible for the operation that is related to user data such as settings, profile, account, and session.

ProfileManager: This class handles the profile operation of the user such as profile pictures, user information.
SettingsManager: This class handles the operations of the settings. For display and update operations SettingManager is responsible for notifying the related classes.
AccountManager: This class manages the user account and privacy settings of the user. When the user makes a modification related with the account or privacy settings, AccountManager will notify the ConnectionManager to make changes in server side.
SessionManager: When a user logins, this class is responsible for managing the sessions.

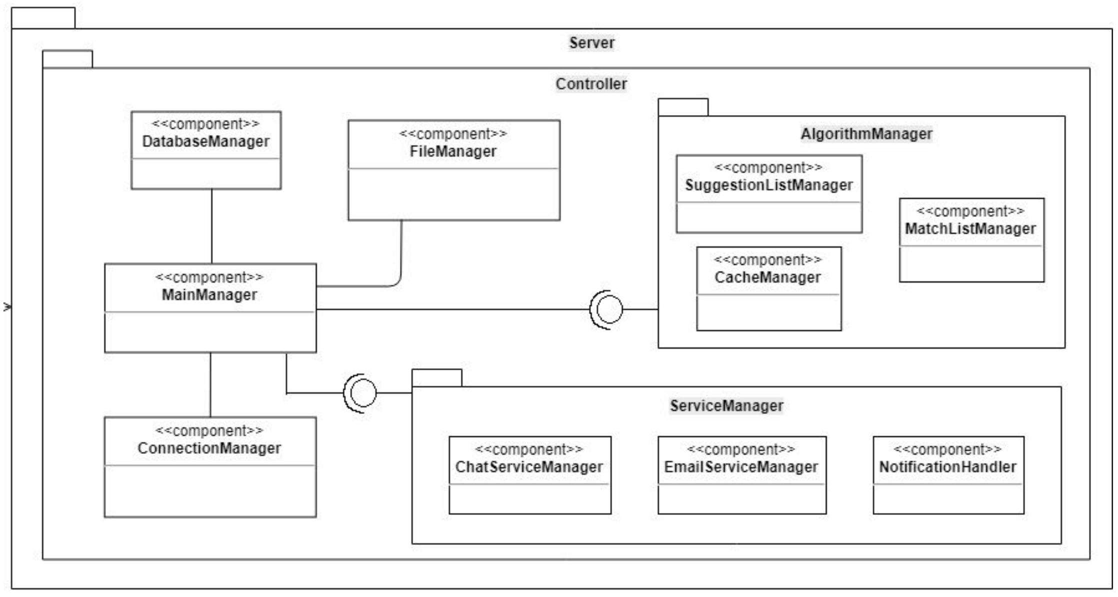
4.1.1.3 View Manager

ViewManager: This class is the main screen adapter of the lists that are mentioned earlier. Since the lists are the main ways of presenting the data on the screen in our application, ViewManager will be an interface between them. Also, it will be the connection between the lists and ConnectionManager as they will not have direct accesses to the outer package.

4.1.2 Connection Manager

Connection Manager class is responsible for connecting the client side to the server. When the user makes a change related to the server side, Connection Manager in the client side will notify the Connection Manage on the server side.

4.2 Server



Server
Controller
<<component>> DatabaseManager
<<component>> FileManager
AlgorithmManager
<<component>> SuggestionListManager
<<component>> MatchListManager
<<component>> CacheManager
<<component>> MainManager
ServiceManager
<<component>> ConnectionManager
<<component>> ChatServiceManager
<<component>> EmailServiceManager
<<component>> NotificationHandler

The responsibility of the server side of the system is to handle different clients, get requests from them, keep data of BookClub users and the application, and manipulate those data by using different components in the Server such as FileManager, AlgorithmManager, etc.. Server package will be implemented in Python on a different machine (a server) to execute the whole package and keep the data of the application.

4.2.1 Controller

This package holds all the operating elements currently in our server system. Omitting this package will not result in any problem at the moment however for further additions to our system, it is necessary to gather the controller-only packages and classes in one package.

4.2.1.1 Main Manager

MainManager: This class is the main director of the requests and responses that are led between client and server, even between server packages and classes themselves. The main reason for this class is to create an interface between the requests and responses which will be helpful for our computations.

4.2.1.2 Service Manager

ChatServiceManager: BookClub provides a chatting service to their users, therefore, this class handles the requests related to chatting between the users. It takes the request from MainManager and processes it, then returns a response according to the request.

EmailServiceManager: BookClub requires e-mail verification and various other e-mail related services. EmailServiceManager formulates e-mails to send to the user’s e-mail address and wait for a response. It again operates through MainManager.

NotificationHandler: This class formulates and sends notifications to the client side to notify the user about various changes and situations.

4.2.1.3 Database Manager

DatabaseManager: This class operates the persistent data on the server to ease the access from the server-side. Besides, depending on the request taken from MainManager, it can access the database for special reasons.

4.2.1.4 File Manager

FileManager: There are some data types that cannot be stored inside a database without serialization, such as images. Therefore the server has to keep some of the data inside the server's storage. File manager will keep track of this data and efficiently organize them.

4.2.1.5 Algorithm Manager

This class manages the algorithm related operations in BookClub. When the user modifies her/his wish list the AlgorithmManager get notification from Connection Manager, execute the algorithm and update the match list and suggestion list.

4.2.1.6 Connection Manager

Connection Manager class is responsible for connecting the server side to the client side. When this class notified by Connection Manager on the client side, it will send notifications to the related classes in server side.

5. Glossary

MySQL: MySQL, is an open source SQL database management system, which is developed by Oracle Corporation. [5]
MVC: The MVC is an architectural design pattern that separates the system into three main components which are model, view and controller. [6]
Client Side: The client side of the system is responsible for managing the interaction of the software with the user. [7]
Server Side: The server side of the system is responsible for system operations and data management. [7]
HTTP: The Hypertext Transfer Protocol is an application protocol which is for collaborative, distributed, hypermedia information systems. [8]
Android Application: Applications that are developed for Android platforms in mobile devices.
Google Play Store: An Android application store which is developed and supported by Google.

6. References

[1] ”New & Used Books from ThriftBooks | Buy Cheap Books Online”. [Online]. Available: https://www.thriftbooks.com/. [Accessed: 23 Dec, 2018].

[2] ”Books at Amazon”. [Online]. Available: https://www.amazon.com/books-used-books-textbooks/b?ie=UTF8&node=283155. [Accessed: 23 Dec, 2018].

[3] ”GoodReads - Share your book recommendations with your friends”. [Online]. Available: https://www.goodreads.com/. [Accessed: 23 Dec, 2018].

[4] ”HepsiBurada - En Çok Satan Kitaplar & Kitap Önerileri”. [Online]. Available: https://www.hepsiburada.com/kitaplar-c-2147483645. [Accessed: 23 Dec, 2018].

[5] "What is MySQL?". MySQL 5.1 Reference Manual. Oracle.[Accessed: 27 Dec, 2018]

[6] “MCV Framework- Introduction” .[Online]. Available: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm. [Accessed: 23 Dec, 2018].

[7] “Client- Server Architecture” .[Online]. Available: https://www.techopedia.com/definition/438/clientserver-architecture . [Accessed: 28 Dec, 2018].

[8] Fielding, Roy T.; Gettys, James; Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee, Tim (June 1999). Hypertext Transfer Protocol – HTTP/1.1. IETF. doi:10.17487/RFC2616. RFC 2616.