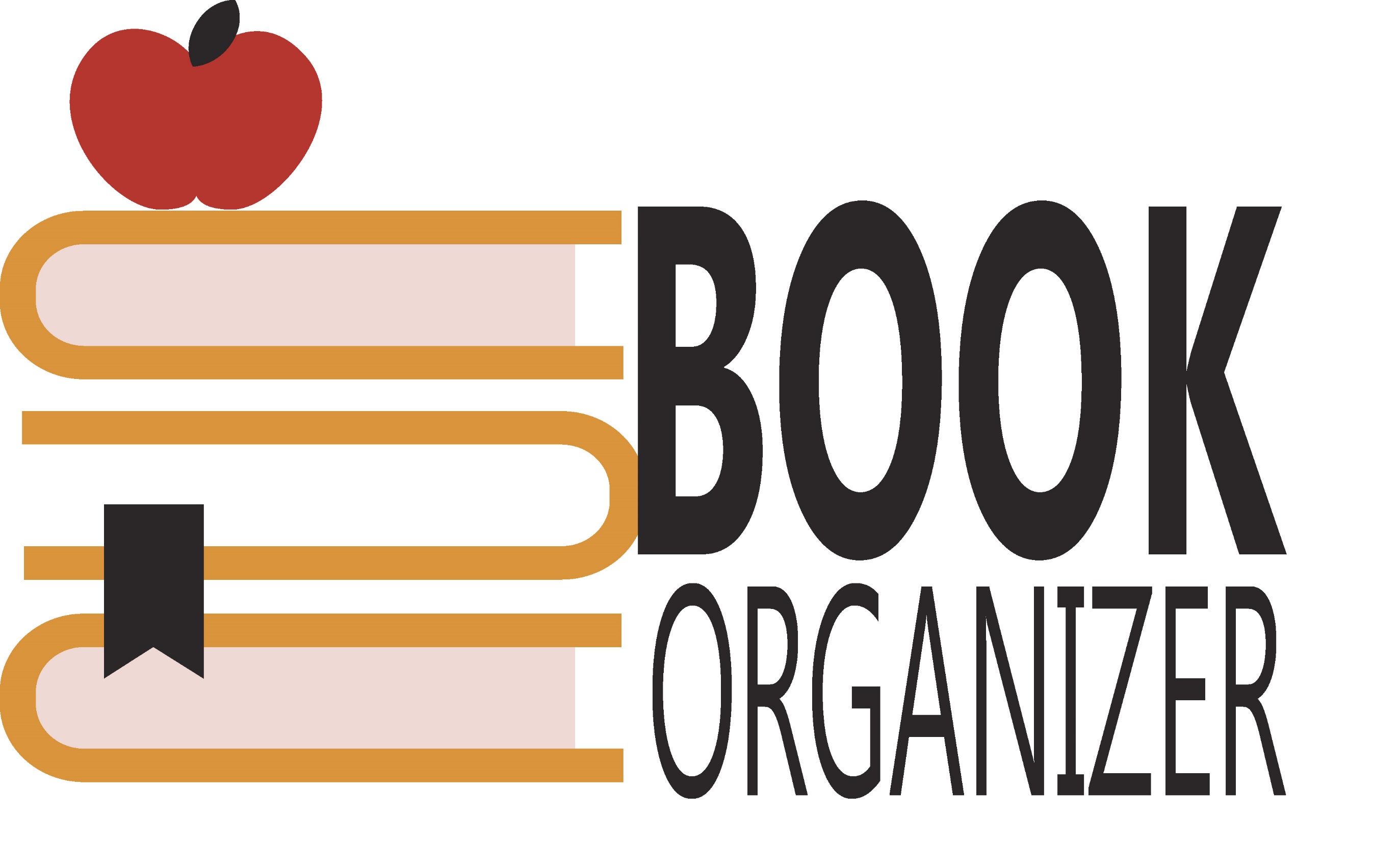
**Arab Open University- Jeddah**



**Faculty of Computer Studies**

**Information Technology and Computing Department**



**Name: Shaima Esmail Alshaebi**

**ID: 201501216**

**TM471: Final Year Project, December 2018   
Supervisor: Ms. Malak Alamir**

**Content**

List of Figures and Tables--------------------------------------------------------------------------------------------- iii

Declaration ---------------------------------------------------------------------------------------------------------- iv

Abstract ----------------------------------------------------------------------------------------------------------------v

Acknowledgments ------------------------------------------------------------------------------------------------- vi

Chapter 1: Introduction --------------------------------------------------------------------------------------------1

Chapter 2: Literature Review--------------------------------------------------------------------------------------2

* **Why Book Organizer? ----------------------------------------------------------** 2

Chapter 3: Requirements and Analysis--------------------------------------------------------------------------6

* **What is Book Organizer? ---------------------------------------------**6

-Project Gantt chart ------------------------------------------------------------------------7

-Project schedule ---------------------------------------------------------------------------8

* **What Book Organizer Does? ------------------------------------------------** 9

-Use case diagram-------------------------------------------------------------------- 9

-Activity diagrams -------------------------------------------------------------------11

-class diagram ------------------------------------------------------------------------14

* **What are the requirements of Book Organizer? ------------------------**15

-Functional requirements----------------------------------------------------------15

-None Functional requirements--------------------------------------------------15

-Software and hardware requirements-----------------------------------------15

Chapter 4: Design, Implementation, and Testing------------------------------------------------------------17

* **How Book Organizer is being designed? ------------------------**17

-Approach used ---------------------------------------------------------------------17

-Justification for choosing approach -------------------------------------------18

-Alternative Approach ------------------------------------------------------------18

-Alterative model is better than waterfall model ----------------------------19

* **What are the changes from analysing to implementation? -**20

-Changes made ---------------------------------------------------------------------20

-Final diagrams ---------------------------------------------------------------------20

-Use case -----------------------------------------------------------------20

-Activity diagrams -------------------------------------------------------22

-Class diagrams ----------------------------------------------------------24

* **What Makes Book Organizer Special? ---------------------------**25

-System Specifications ------------------------------------------------------------25

-Functional requirements -------------------------------------------25

-Non-Functional requirements -------------------------------------25

-Why Book Organizer is Novel --------------------------------------------------26

-Book Organizer Database and interfaces -----------------------------------26

* **How is Book Organizer Being implementing? ------------------**29
* **Testing --------------------------------------------------------------------**32

Chapter 5: Results and Discussion ------------------------------------------------------------------------------37

* **What are the Finding from Developing Book Organizer, Goal Achieved, and Future work? ------------------------------------------------------------------**25

Chapter 6: Conclusions --------------------------------------------------------------------------------------------39

References ---------------------------------------------------------------------------------------------------------------40

Appendices --------------------------------------------------------------------------------------------------------------41

**List of Figurses and Tabels**

-Figure 1.1 Change in reading frequency after adopting mobile reading (West & Ei Chew, 2014)-----------------------------------------------------------------------------------------------------------3

- Figure 1.2 Google play Bpooks and Moon+Reader logos-------------------------------3

- Figure 1.3 Khatmah app logo-------------------------------------------------------------------4

-Figure 1.4 screen shots from Khatmah app-------------------------------------------------4

- Figure 2.1 Gantt chart of the project plan ------------------------------------------------6

- Figure 2.2 Timeline for Gantt chart of the project plan--------------------------------7

- Table 2.3 Project schedual----------------------------------------------------------------------8

-Figure 2.4 use case diagram for Book Organizer -----------------------------------------10

- Figure 2.5 activity diagram for Book Oerginazer user: Reader-----------------------12

- Figure 2.6 activity diagram for Book Oerginazer user: Admin------------------------13

-Figure 2.7 class diagram of Book Organizer -----------------------------------------------14

-Figure 3.1 Iterative model (What is Iterative Model?, 2018) -------------------- ----17

- Figure 3.2 Waterfall model (Govardhan, 2010)-----------------------------------------18

- Figure 3.3 Final use case diagram for Book Organizer ---------------------------------21

- Figure 3.4 Final activity diagram for Book Organizer user: Reader------------------23

-Figure 3.5 Final class diagram of Book Organizer ----------------------------------------24

-Figure 3.6 UML diagram of Book Organizer -----------------------------------------------26

-Figure 3.7 Book Organizer final interfaces ------------------------------------------------27

-Figure 3.8 Book Organizer interfaces2 -----------------------------------------------------28

-Figure 3.9 Login process -----------------------------------------------------------------------29

-Figure 3.10 Register process ------------------------------------------------------------------30

-Figure 3.11 Set time process -----------------------------------------------------------------31

-Figure 4.1 Register exitst username --------------------------------------------------------33

-Figure 4.2 Register nonexistent username ------------------------------------------------34

-Figure 4.3 login nonexistent username ----------------------------------------------------35

-Figure 4.4 Login wrong password -----------------------------------------------------------36

**Declration**

I Shaima Esmail Alshaebi declare that this report of the project entitled “Book Organizer” submitted for “Arab Open University” is written and carried out by my

Name:- shaima Esmail Alshaebi

ID: - 201501216

**Abstruct**

Reading books is a discount ticket to everywhere. Since a trip must be well planned in order to be great and wonderful Book Organizer must be developed. Book Organizer is an Android application that is developed to help people who want to make reading a part of their daily routine and want to enjoy their reading journey by organizing their reading.

Book organizer allows readers to select a book from a list and read that book. The application has features that help readers, such as creating a schedule that contains how many pages must be read in order to finish the book during a specific time period selected by the reader. In addition, the application reminds the reader to read every day in order to help him to make his reading daily.

**Acknolgments**

I would like to thank: my family for supporting;

Ms. Laila Abuljadayel and Ms. Malak Alamir for helping me during project part one and two; Books for inspiring me.

**Intruduction**

As it has been said “today reader tomorrow leader”. Undoubtedly, reading helps in developing the personality. Besides, it helps in acquiring some skills like self-learning skills, analytical thinking skills, and writing skills. Moreover, reading builds an aware and cultured generation that is protected from unawareness. In addition, reading has some health benefits such as mantel simulation and stress reduction.

Due to technology evolution, reading has become easy through portable devices. Also, a lot of systems was developed in order to facilitate reading and find books such as Google Play Books and Moon+Reader. However, there is not a system which helps readers organize their reading and encourage them to read daily. Therefore, the reader may read a lot of books in a short time, but reading will not be efficient if it is not organized and daily.

Book Organizer helps today readers. It is an application that gives the readers the ability to make their reading organized and well planned. It is an application that allows the reader to select books from its list and select a duration to finish reading that book. All this process is done in order to create a schedule that consists of how many pages must be read every day and the date of each day. Besides planning and arranging, Book Organizer aims to make the reader continuously read every day by reminding them to read at a specific time chosen by them. Each day, the reader will receive a notification that reminds them and encourages them to read.

**Literature review**

**Why Book Organizer?**

We all admit the importance of reading in our lives. We cannot deny that reading prevents people and their community from being unlettered. Reading can give you a lot of academic and life skills. For example, Reading enhances self-learning skill. In fact, reading act as a practical practice for the reader that helps them improve self-learning skill. Also, by gaining lots of knowledge through reading, the reader becomes more able to improve analytical thinking skills (Winter-Hébert, 2017). Besides skills, reading can help in mantel simulation by keeping the brain active which prevents it from losing power. Furthermore, reading acts as an exercise for the brain. Also, reading helps in reducing stress since it transports you into other realms (Billington, Dowrick, Hamer, Robinson, & Williams, 2010).

As a result of the development technology, reading books has become easier through mobile devices. According to the report of UNESCO about “Reading in mobile era”, reading in mobile is efficient due to the affordability and convenience especially in places where paper books are not easily available such as developing countries. Also, according to the report, 62 percent of readers read more after adopting mobile reading (as shown in figure 1.1). Consequently, a lot of applications was developed for reading.

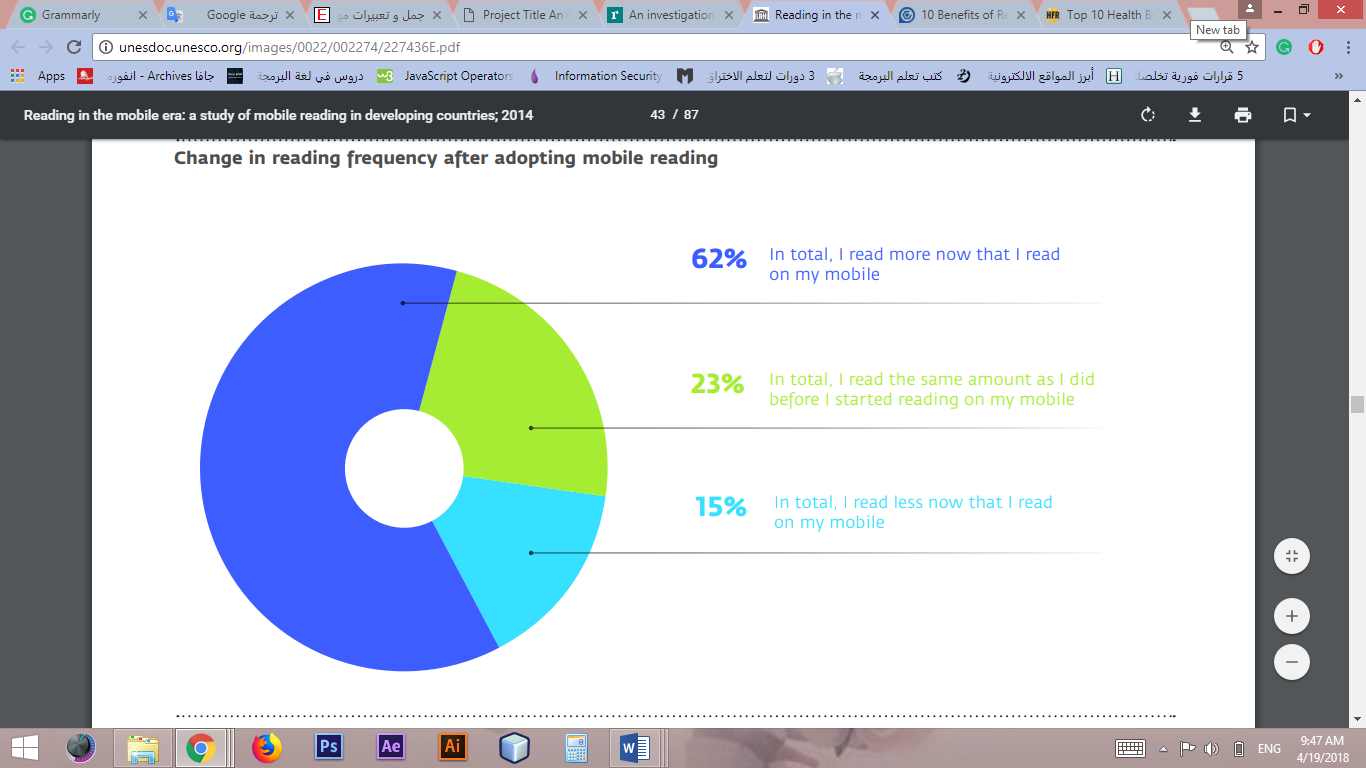


Figure 1.1 Change in reading frequency after adopting mobile reading (West & Ei Chew, 2014)

As an example of these applications is Google Play Books and Moon+Reader (see figure 1.2). These applications enable users to store and read books. In addition, Moon+Reader has some features to customize the interface. Also, it has the possibility of switching between day and night modes automatically. However, these applications do not provide any utilities that help in organizing the reading or even reminding to read as Khatmah application does.

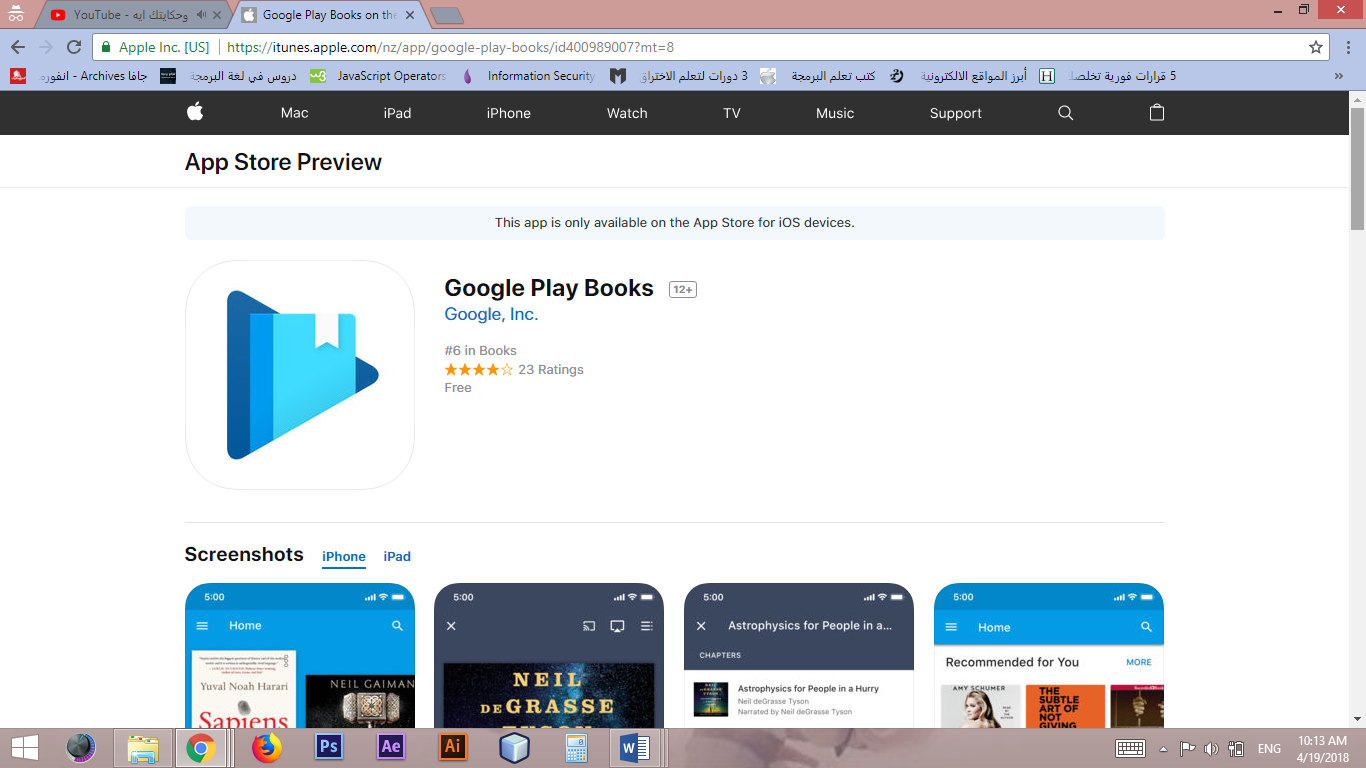
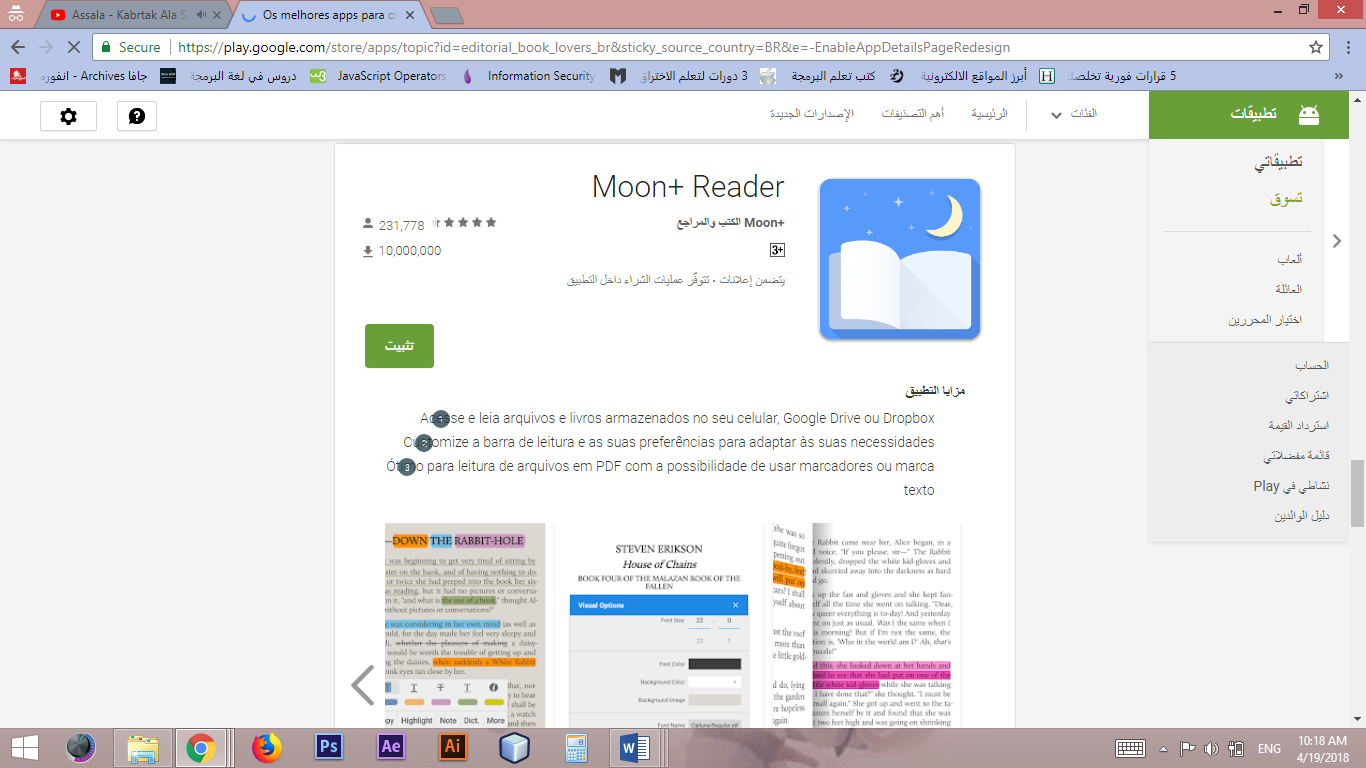
 

Figure 1.2 Google play Bpooks and Moon+Reader logos

Khatmah app helps users to finish reading Quran at a specific time. After the user chooses the duration to finish reading, Khatmah specifies the number of pages that must be read every day. Furthermore, it allows users to set the alarm time for the reminder to read. But, Khatmah is for reading Quran only.(see figures 1.3 , 1.4)

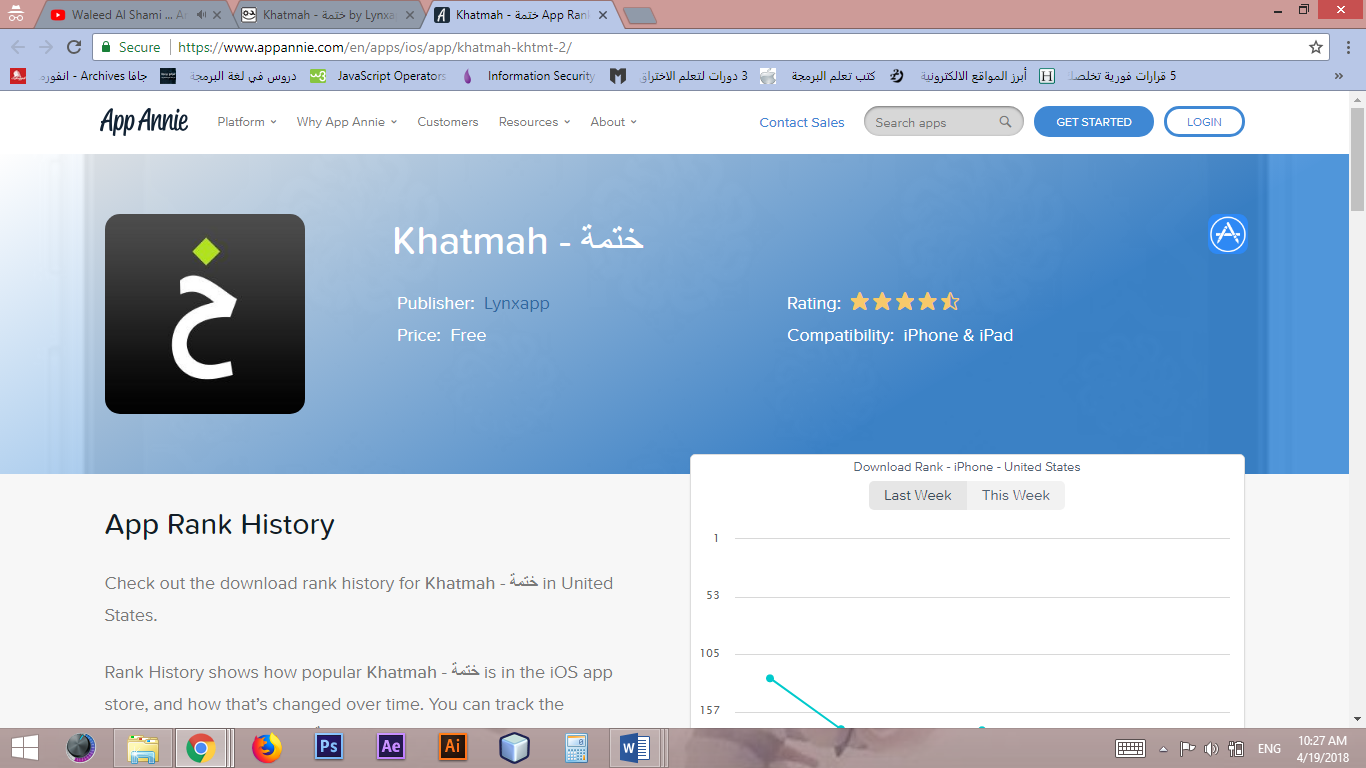


Figure 1.3 Khatmah app logo

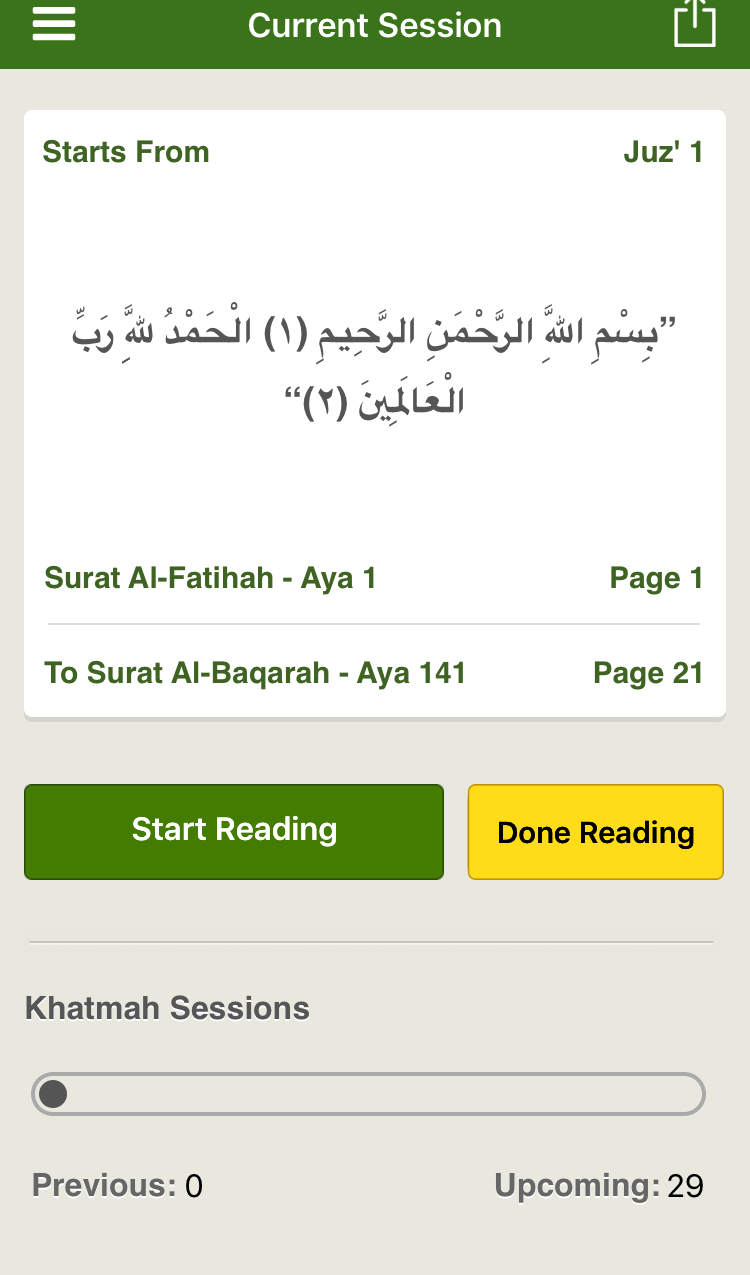
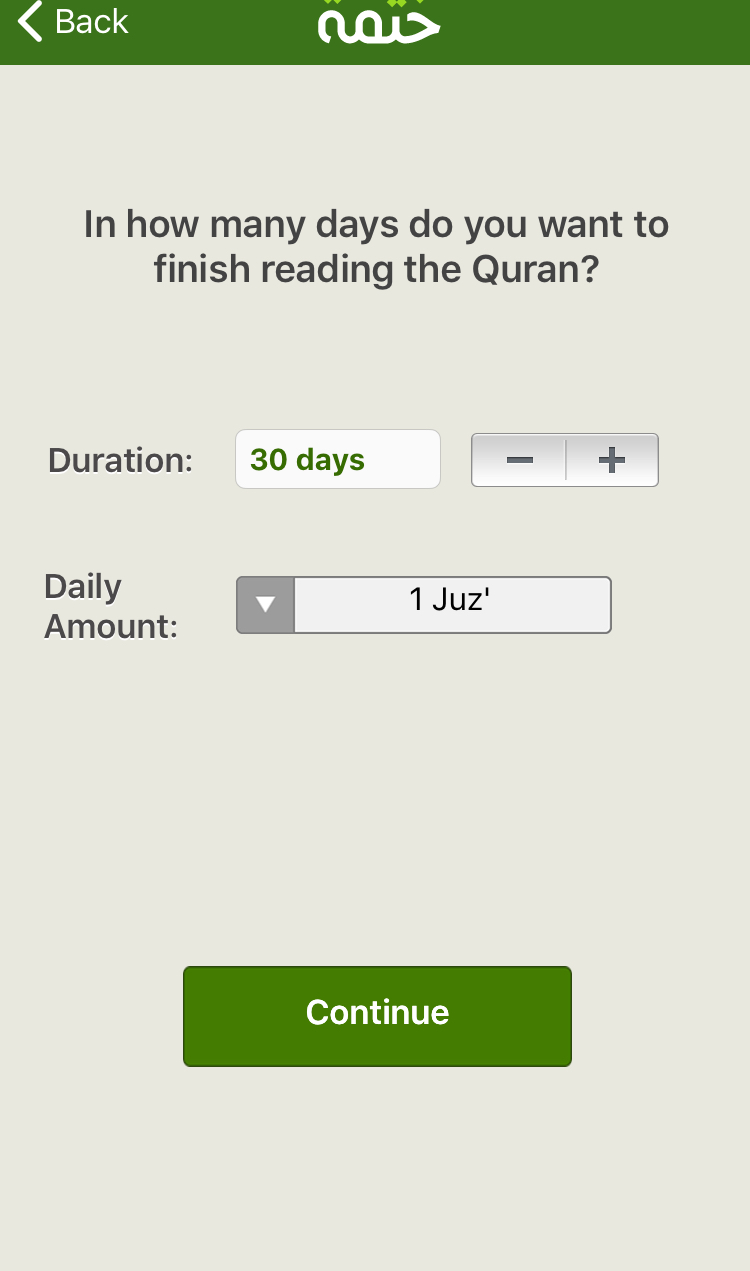
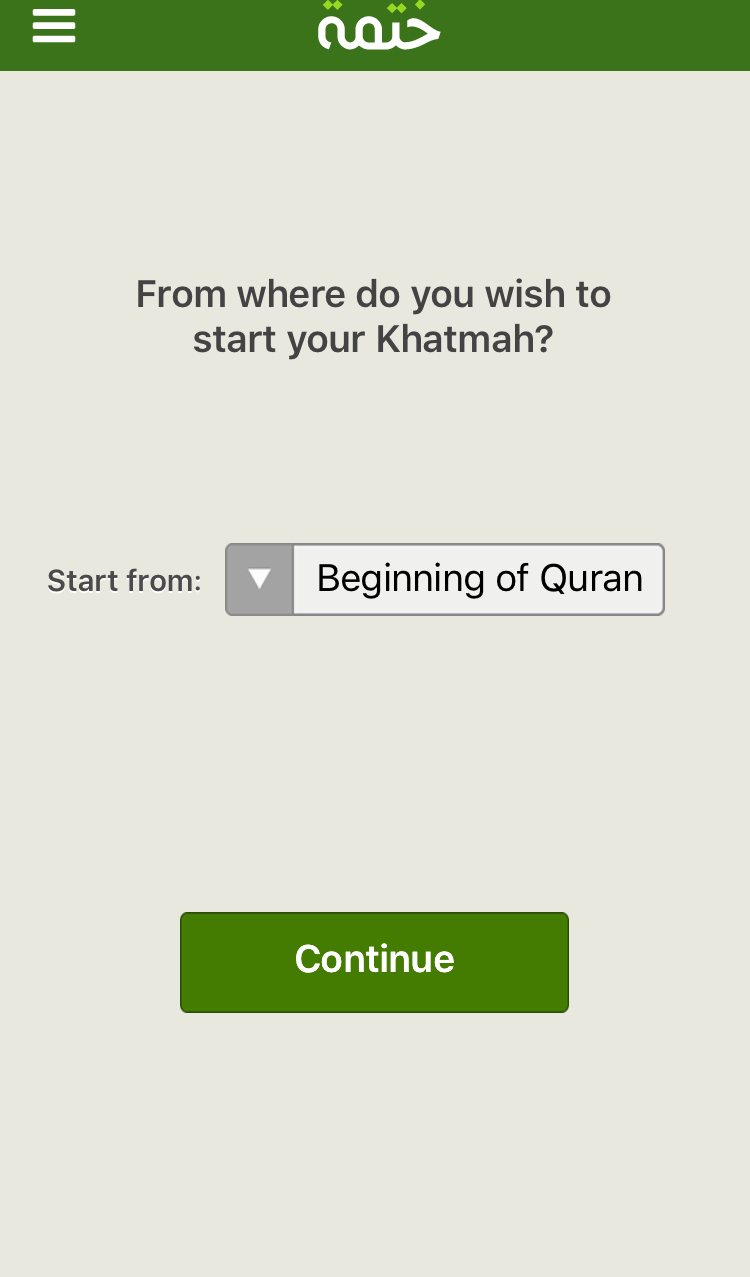


Figure 1.4 screen shots from Khatmah app

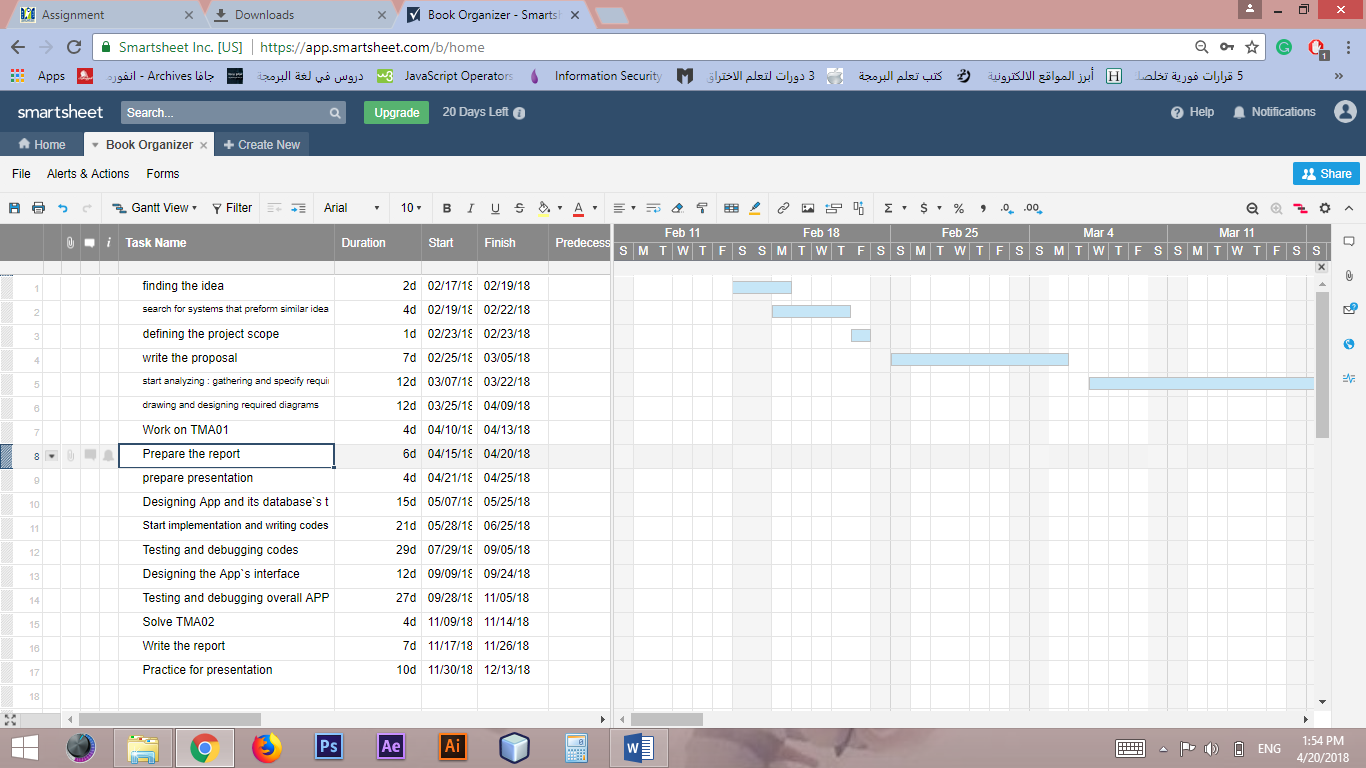
Unlike Khatmah, Book Organizer enables readers to choose from a list of different books. In addition, it creates a schedule based on the number of pages of each book.

**Requirments Analaysis**

**What is Book Organizer?**

Book organizer an application that allows readers to make their reading organized. Also, aims to help them to make their reading continuously. By creating schedules for reading and send a notification for reminding. Book organizing enables readers to register, select book, select time for finishing that book and read the book. Furthermore, it allows readers to set a time for the remainder in order to remind them.  - Project plan and schedule

Before analysing the requirement was started, I did a plan and schedule for the project in order to make tasks clear and organize the work. The figure (2.1), (2.2) and table (2.3) below shows the tasks and the duration of each one.



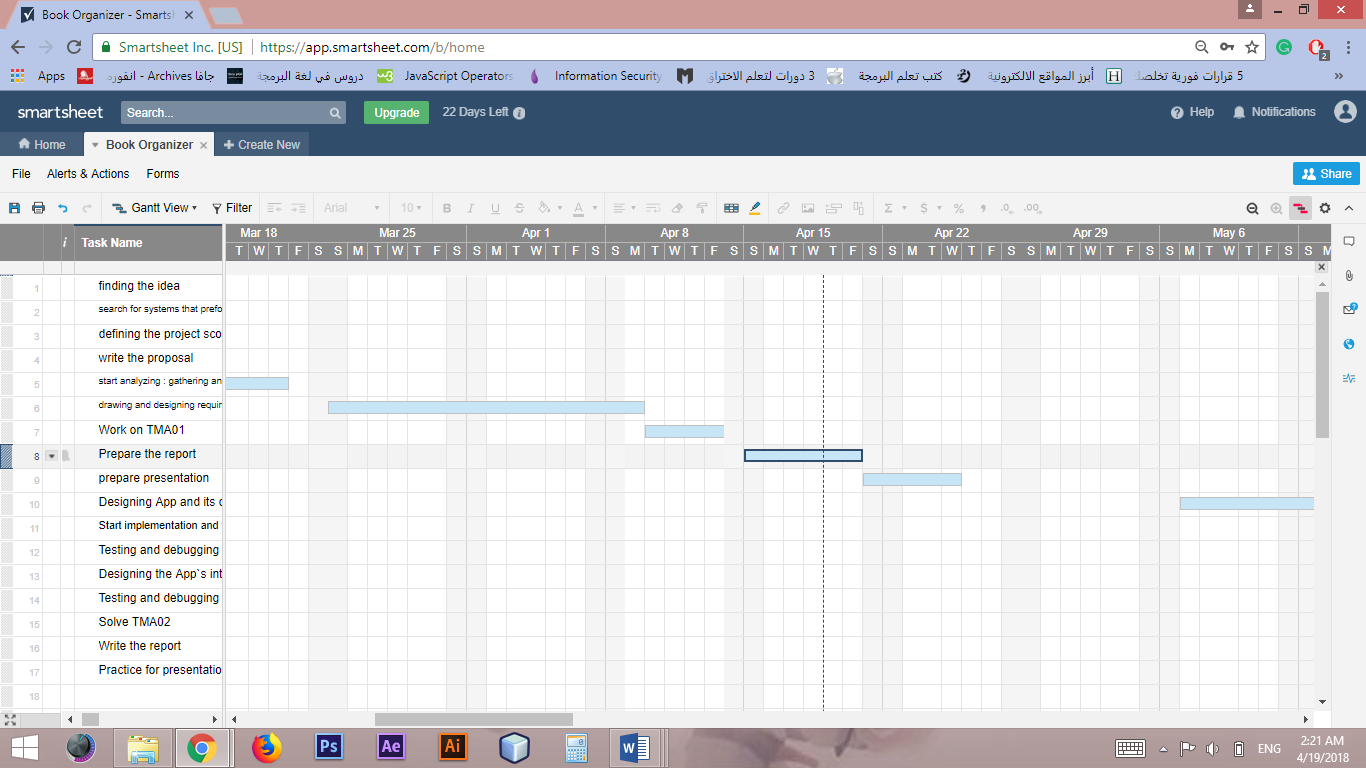
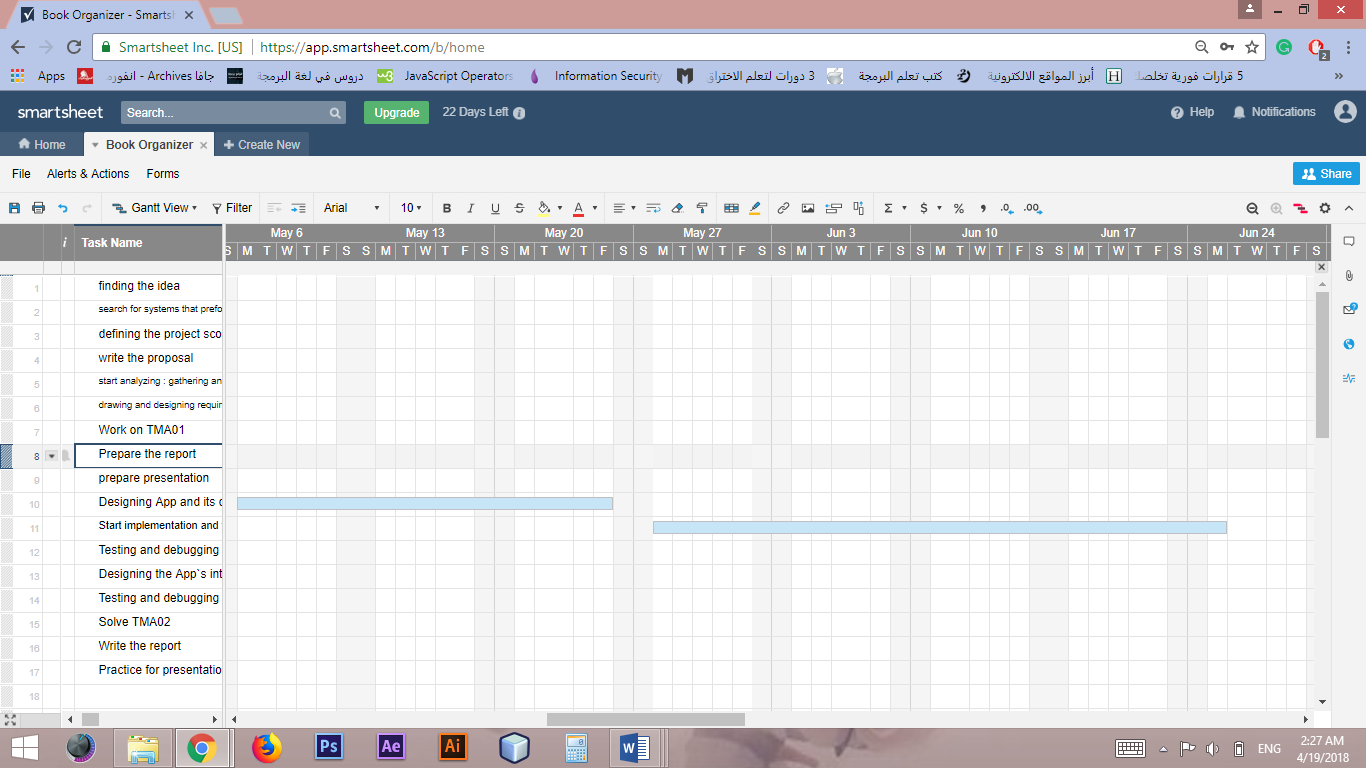
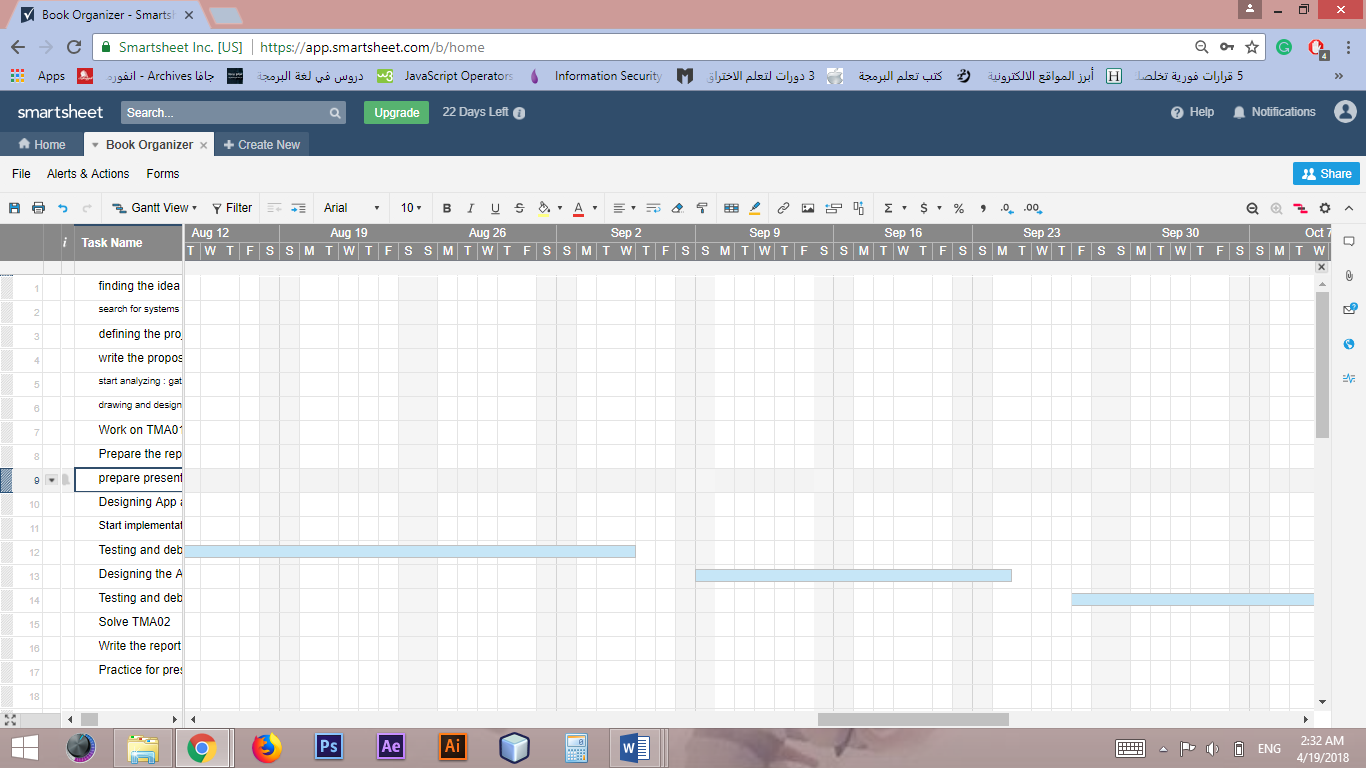
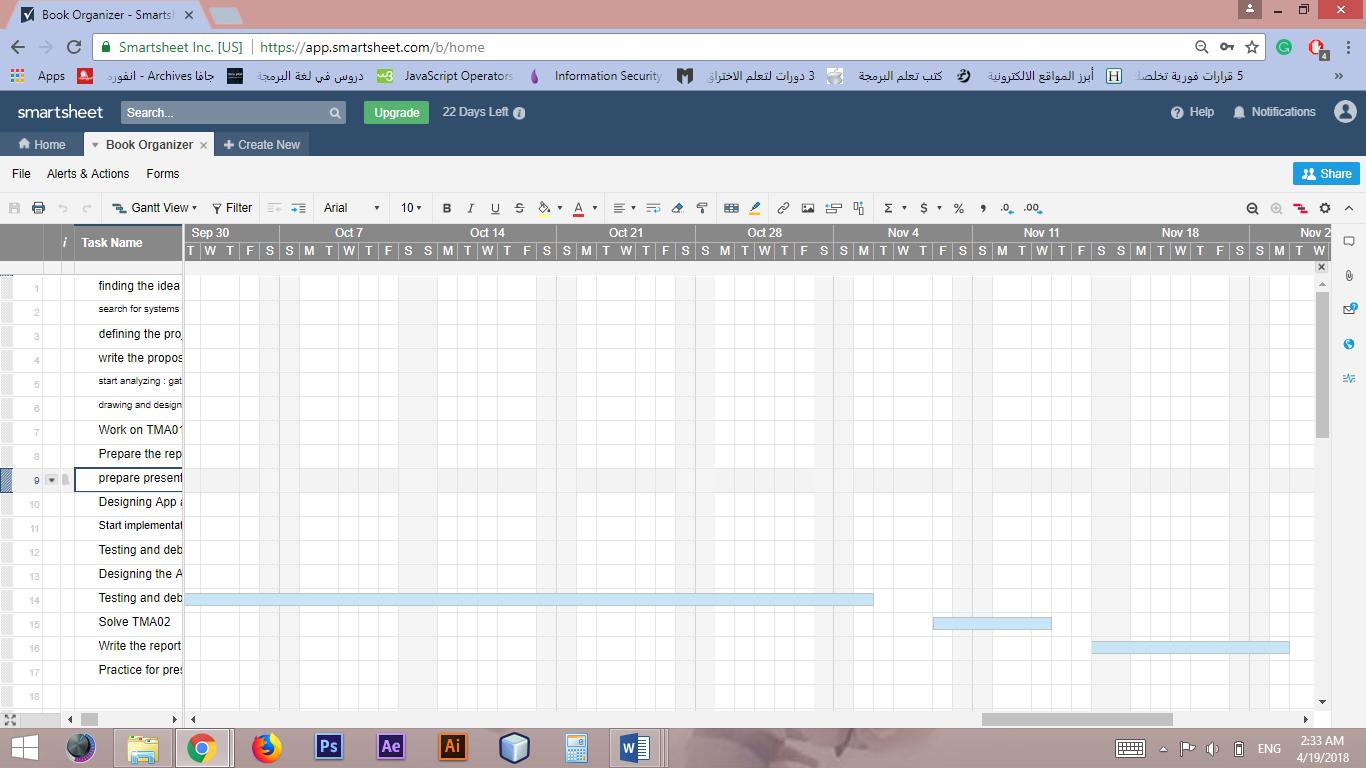


Figure 2.1 Gantt chart of the project plan







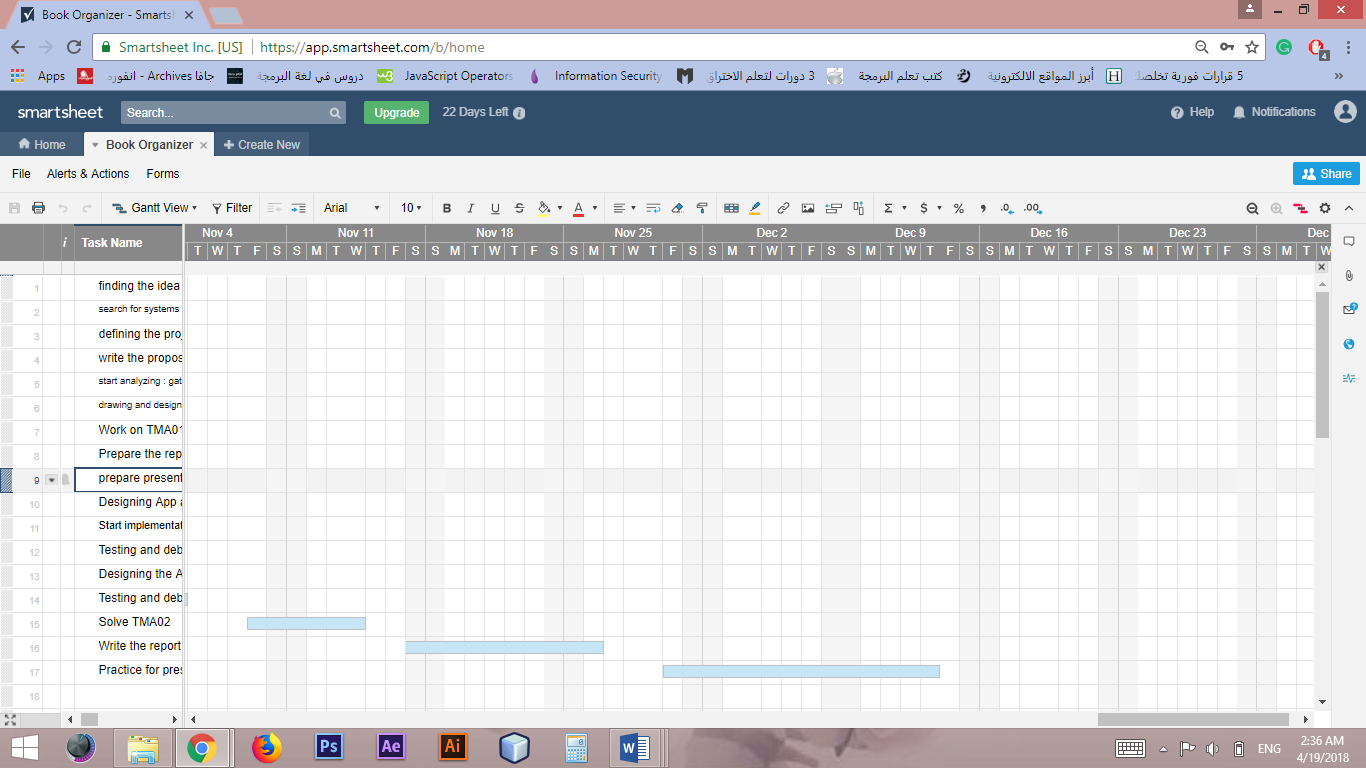


Figure 2.2 Timeline for Gantt chart of the project plan

|  |  |  |  |
| --- | --- | --- | --- |
| Task No. | Task | Required Task  (predecessors) | Estimated duration(in days) |
| 1 | finding the idea | - | 2d |
| 2 | search for systems that preform similar idea for initial understanding requirements and defining resources needed | 1 | 4d |
| 3 | defining the project scope | 2 | 1d |
| 4 | write the proposal | 1,3 | 7d |
| 5 | start analyzing : gathering and specify requirements | 4 | 12d |
| 6 | drawing and designing required diagrams | 5 | 12d |
| 7 | Work on TMA01 | 6 | 4d |
| 8 | Prepare the report | 7 | 6d |
| 9 | Prepare presentation | 8 | 4d |
| 10 | Designing App and its database`s tables and relations | 5,6 | 15d |
| 11 | Start implementation and writing codes | 10 | 21d |
| 12 | Testing and debugging codes | 11 | 29d |
| 13 | Designing the App`s interface | 11 | 12d |
| 14 | Testing and debugging overall APP | 11,13 | 27d |
| 15 | Solve TMA02 | 14 | 4d |
| 16 | Write the report | 15 | 7d |
| 17 | Practice for presentation | 16 | 10d |

Table 2.3 Project schedual

**What Book Organizer Does?**

-use case diagram

As it is shown in the use case diagram in figure2.4, Book Organizer the Android application that enables users to register and get a free account. After registration, the reader will be able to browse books from a list. The app will allow readers to select books and a period of time they wish to finish that book during it. After that, the app will create a schedule that indicates how many pages the user must be read every day in order to finish the book during the selected duration. More than that, the app will enable the reader to read the book and browse the schedule. Moreover, Book Organizer will remind the user to read every day. After the user set a time for reminding, the app will send a notification to them every day for reminding them to read.

The app will be managed by an admin who will be able to add, delete, and update books of the system. However, both reader and admin will not be able to do anything before they log in. After login, the system will verify the password`s user. If the password valid the app will approve login. However, if the password is not valid the login will be rejected. After each session finish, the database will be updated.

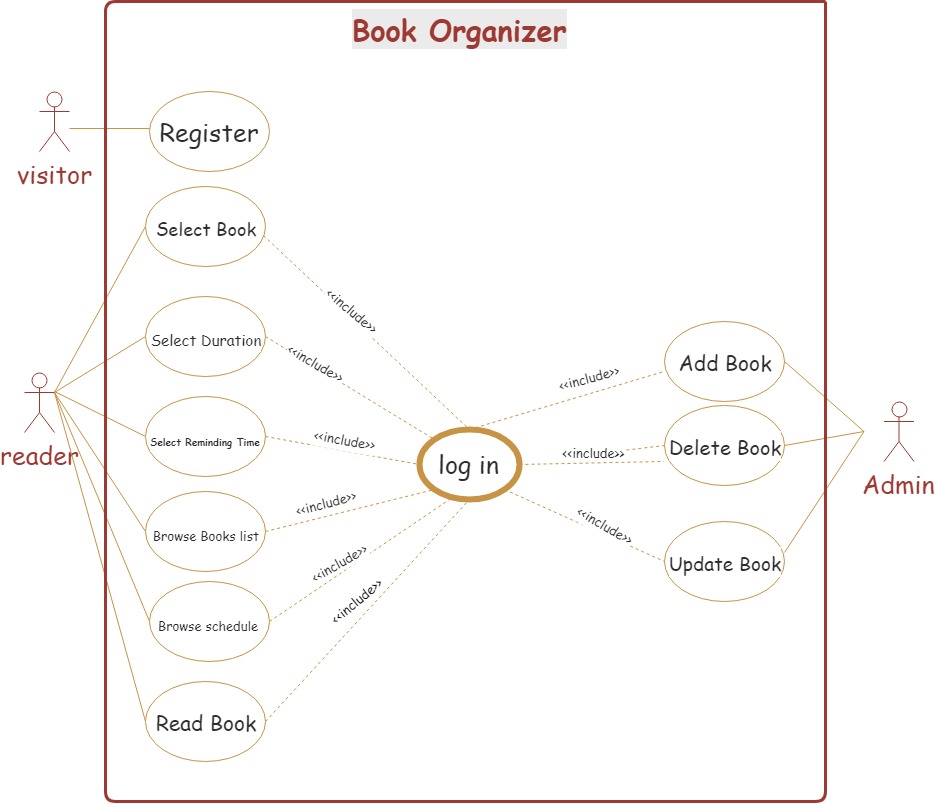


Figure 2.4 use case diagram for Book Organizer

-Activity diagrams

-user: reader

The activity diagram below (Figure 2.5) shows the operation of the app when it is used by the reader. If the reader does not have an account, the app will give them the ability to create one. However, if there is an account, the reader login. Then the app will check the password if it is vailed or not. If the password vailed, the app will check the account if any book was selected. If no book is selected, the app browses the list of books for the user to choose a book and a duration to finish it. After that, the schedule will be created. However, if there is a selected book, the app then check if an alarm is set or not. If the alarm is already set, the app will browse the schedule. Else, the app will ask the reader to set the alarm. Then it will confirm the alarm settings and browse the schedule after that. Next, the app browse the book for the reader to read then exit the app after finish reading. After the reader exit, the app will update the database then prepare for next process.

-user: Admin

On the other hand, if the user is the admin as shown in the figure below (Figure 2.6), the process will be different. After login and verifying password, the admin will be able to add, delete and update books. After that, the app will update the database then prepare for the next process.

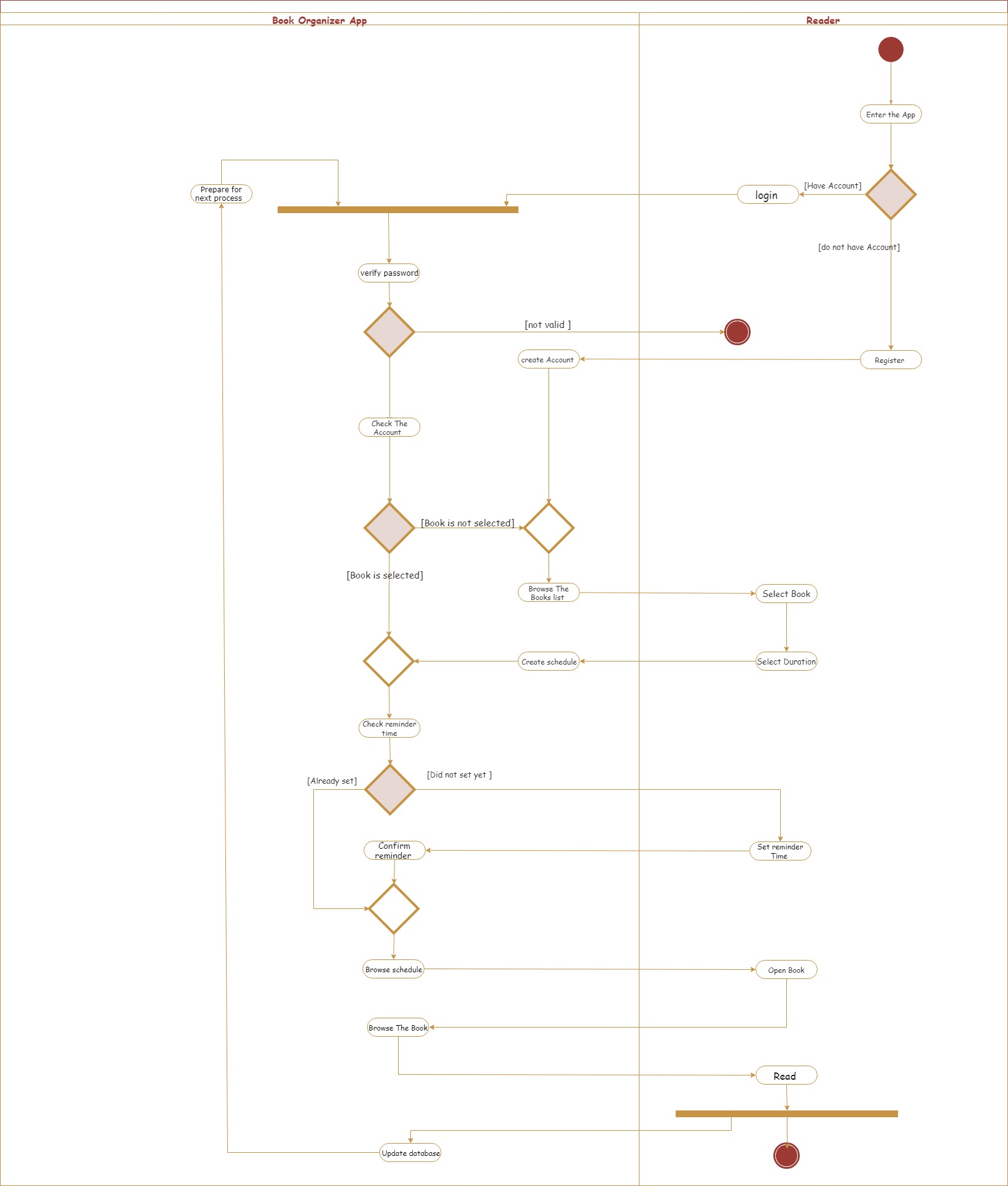


Figure 2.5 activity diagram for Book Oerginazer user: Reader

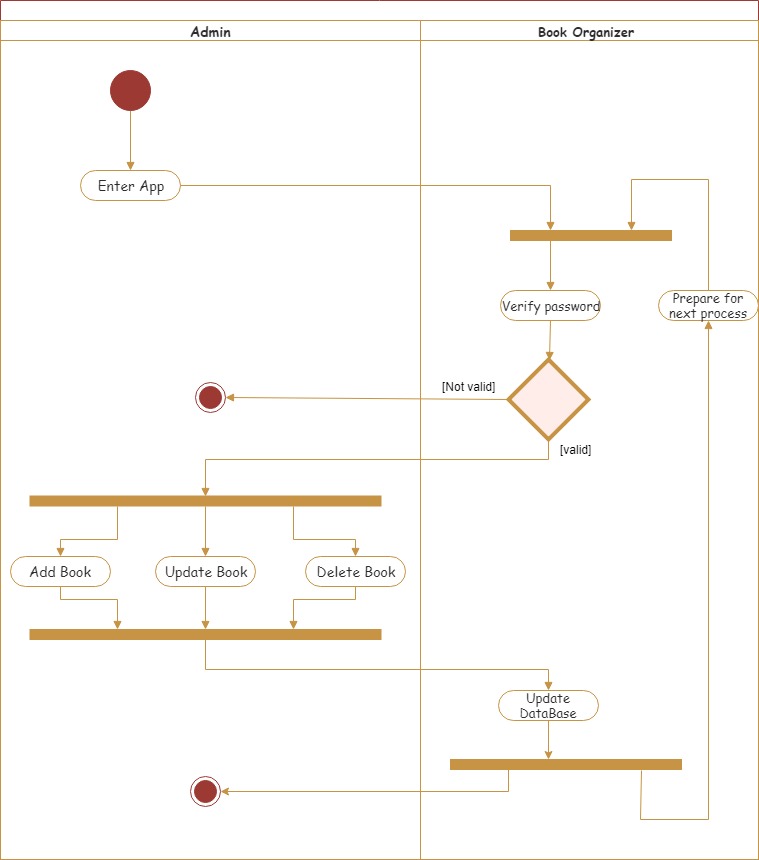


Figure 2.6 activity diagram for Book Oerginazer user: Admin

-class diagram

The class diagram below (figure 2.7) shows the components of the system and relations between them. The reader will be able to read one book at a time. However, they may read a lot of previous books. Books could be selected by zero or more readers and be managed by one admin. The reader can have zero or one schedule and schedule contain one book. In addition, the reader can have zero or more previous schedule.

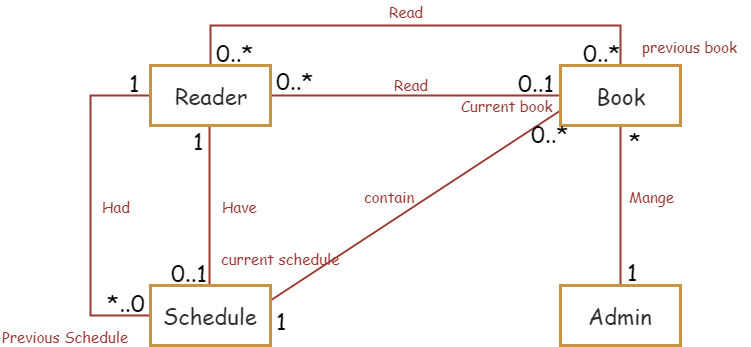


Figure 2.7 class diagram of Book Organizer

**What are the requirements of Book Organizer?**

-Functional requirements

- Book Organizing enables admin to add, delete and update books.

- It gives visitors the ability to create an account.

- Also, the app allows its users to log in.

- It browses the books for the reader.

- It allows the reader to choose a book.

- It allows the reader to specify a duration of time to finish the book

during it.

- The app creates a schedule for the user.

- In addition, it allows readers to set alarm.

- The app sends a notification to the reader for reminding them to read.

- The app enables the reader to read the selected book.

- It also browses the schedule for the reader.

- The app updates the database after each session.

-None Functional requirements

- The app should verify the user`s password (security).

- The system must show the cover picture of each book in the book list (look and feel).

- The system will be easy to use for new users (usability).

- The system will be available for the users any time (availability).

-Software and hardware requirements

- Microsoft Word and Microsoft power point are used and will be used for preparing TMAs, presentations, and reports.

- Smart sheet was used for creating Gantt chart for the project.

- Draw.io was used for drawing project`s diagrams

- Android studio for will be used implementing and coding

- XAMPP server and PhpMyAdmin will be used for implementing the database.

- The hardware that is used: the personal computer.

**Design Implementation and Testing**

**How Book Organizer is being Designed?**

- Approach used: -

Before starting the project, we must choose the appropriate software development life cycle model in order to avoid any costs and efforts that may be needed to change the model after starting to develop. However, some important points must be considered in order to find a model that is suitable for the project, for example, the project context and, business requirements. In Book Organizer, the Iterative model (Figure 3.1) was used to develop it. One of the reasons that Book Organizer is an object-oriented project. Also, the requirements of the project were defined. In iterative model, the system is developed by being divided into small parts. Each part is developed through repeated cycles.

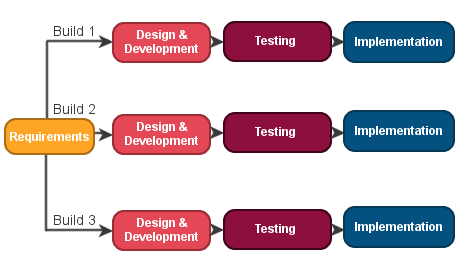


Figure 3.1 Iterative model

- Justification for choosing the approach: -

As it is explained before, requirements were defined in Book Organizer. However, some details were considered as possible features to be added in the future. Therefore, the iterative model was a very appropriate model for developing Book Organizer since it is an adaptable to changes may be needed. Moreover, in iterative model, defects could be discovered at early stages. That is because of the easiness of testing and debugging in the iterative model during small iteration. Another important reason is iterative model supports parallel development.

- Alternative approach: -

The alternative model which was considered to be used as plan B for developing Book Organizer is waterfall model. Waterfall model is a linear-sequential life cycle model (Figure 3.2). The reasons behind choosing waterfall model as an alternative model are that it is an easy model to be used and managed and understand. Also, waterfall model works well for small projects. Another reason, it is appropriate for projects where requirements clear such as Book Organizer.

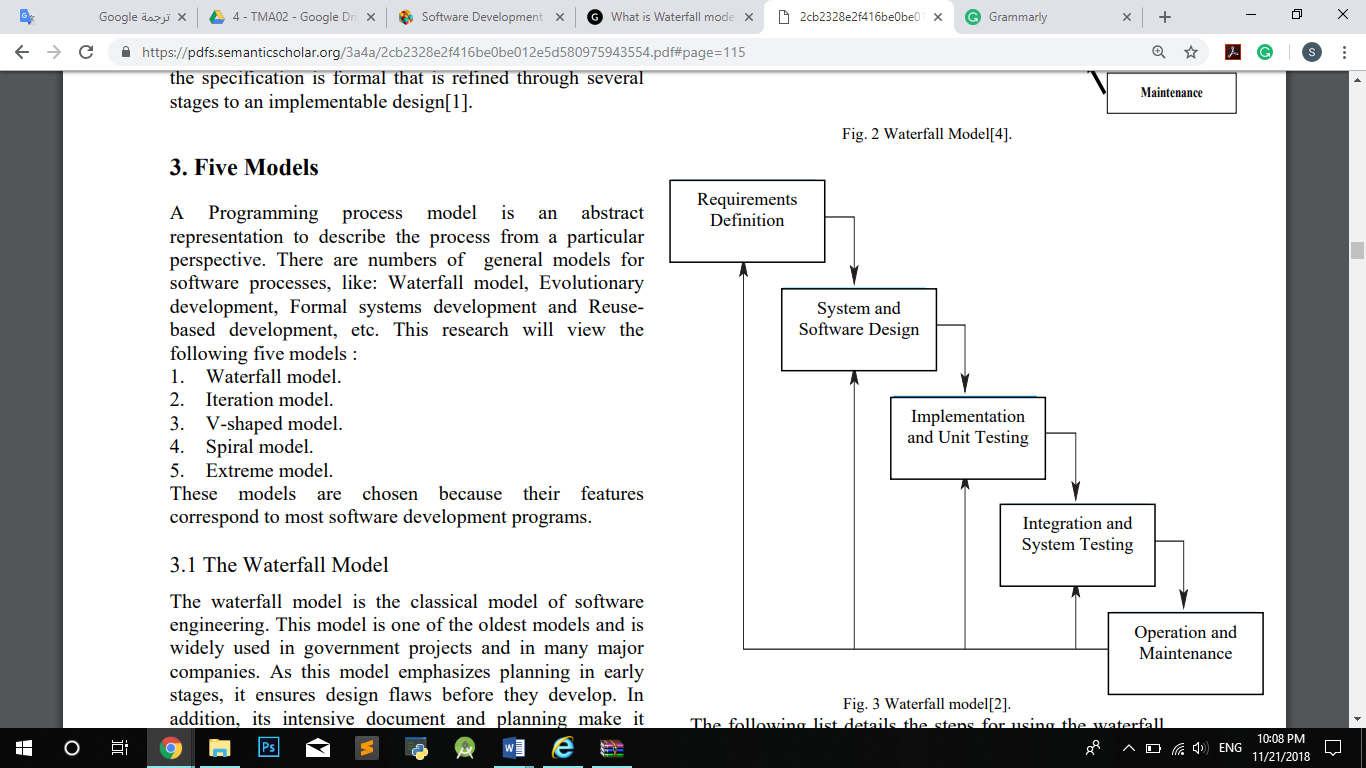


Figure 3.2 waterfall model

- Iterative model is better than waterfall model: -

Iterative model is more flexible than waterfall model where it is difficult to go back to previous stages to make changes. Although waterfall model is an appropriate model for small projects and for projects where requirements are known, project issues could be detected in the test stage. On the other side, in the Iterative model project issues could be detected at early stages.

**What are The Changes from Analysing to Implementation?**

- Changes made: -

During progress from the analysis stage to the implementation stage, some changes are done in order to make Book Organizer recognized, familiar, and more appropriate for solving the problem of scheduling pages. For example, for making the user feel familiar to the app User’s previous schedule list is decided to be added. Also, instate of adding the book by the admin, its link will be added. This change is done in order to make the app more general and allow the availability of books from different resources. Also, to allow books which have not electronic copy to be scheduled. However, the app will still give the user the ability to read electronic books by open them via the browser. Also, since not all books will be existing, the cover picture of the book in books list will not be have appeared.

- Final diagrams: -

- Use Case

As it is shown in (Figure 3.3) below, the reader is able to open his previous sheduales which will be ordered in a list.



Figure 3.3 Final use case diagram

- Activity diagram: reader

Previous activity diagram (Figure 2.5) was very complicated and shows that the reader has to set the reminder for notification. However, setting the reminder is not mandatory it is optional. Also, open book must be proceeded by a condition to check if the book link exists or not (see Figure 3.4).

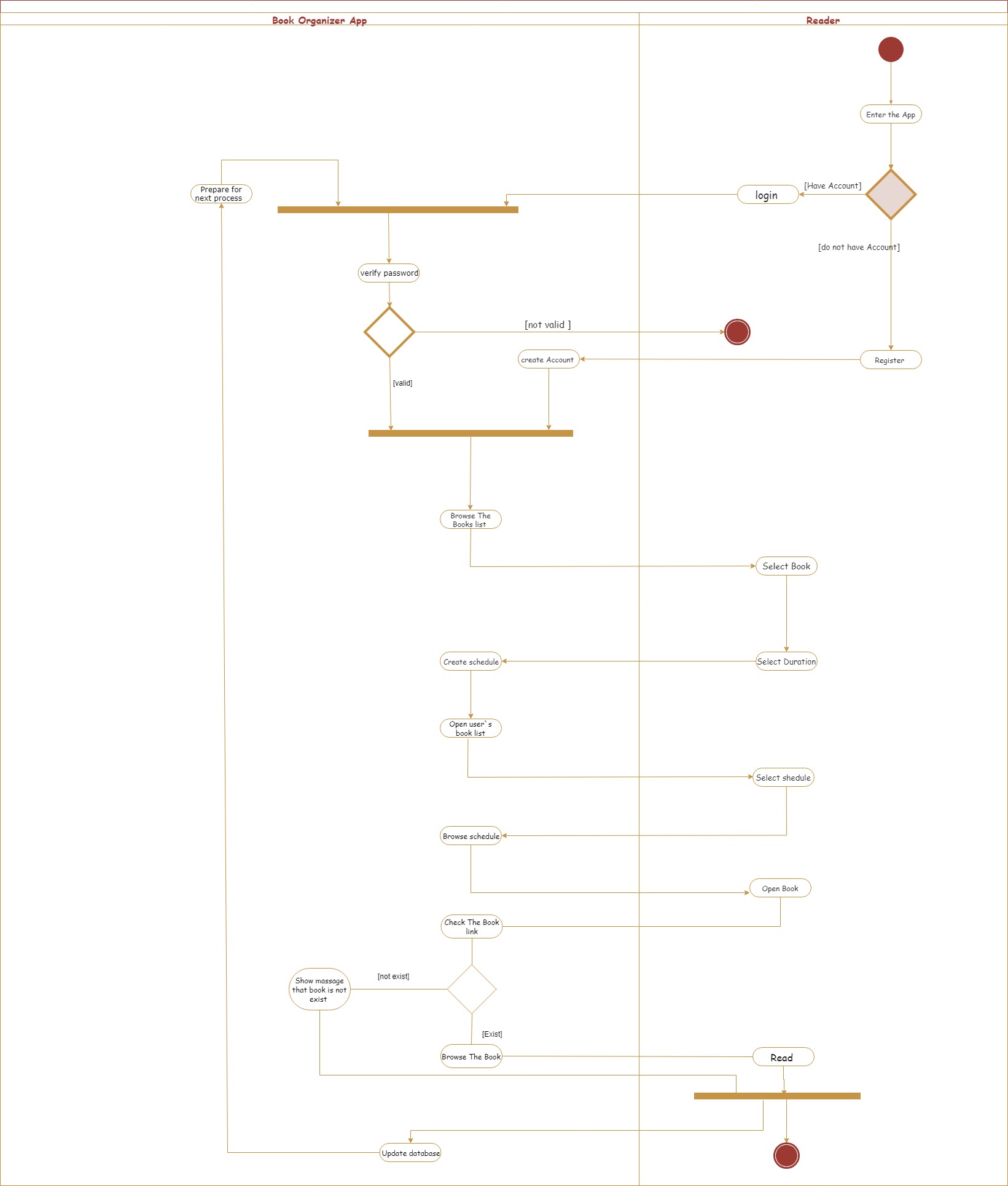


Figure 3.4 Final activity diagram for Book Oerginazer user: Reader

- class diagram:

Class diagram below (see Figure 3.5) shows the component of Book Organizer and the relation between them. In this stage login and registration are added with an inheritance relation between them. Also, there is a composition relationship between login and admin and between log in and admin.

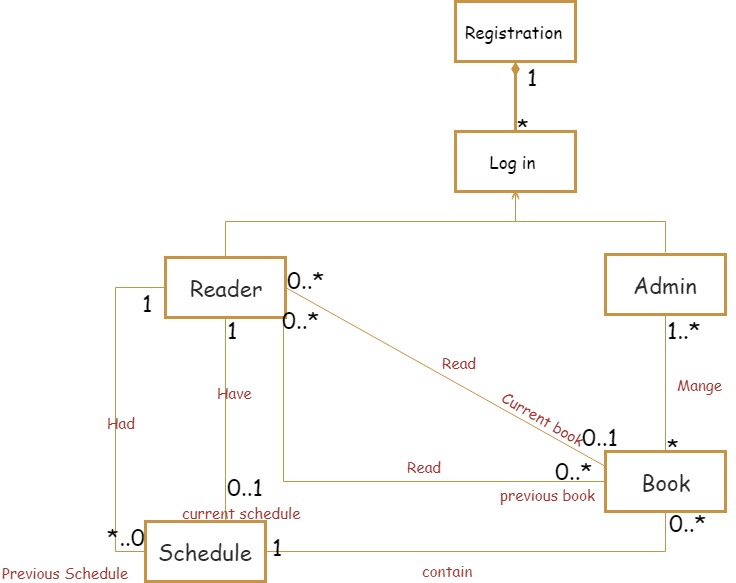


Figure 3.5 Final class diagram

**What Makes Book Organizer Special?**

- System Specifications: -

Specifications of the system considered as the key or the start point of implementation. Specifications determined what makes the system special and novel. In Book Organizer, functional and non-functional requirements were determined as follows: -

-Functional requirements

- Book Organizing allow the visitors to register.

- It browses book list for the reader.

- It allows the reader to choose a book.

- It allows the reader to choose in how many days he wants to finish the book.

- The app creates a schedule for the reader consist of pages must be read every day.

- It gives readers the option to choose their notification time.

- It also browses the schedule for the reader.

- It can browse previous schedules for reader.

- It notifies the reader to read every day.

- It gives the admin the ability to add, delete and update books.

- The app updates the database after each session.

-None Functional requirements

- The app should verify the user`s password (security).

- The system will be easy to use for new users (usability).

- The system will be available for the users any time (availability).

- Why Book Organizer is Novel?

In fact, no applications preform the same functionalities as Book Organizer. It is encouraged reading by reminding to read every day. Also, it schedules the pages for readers which makes reading easy, organized and daily.

- Book Organizing database and interfaces: -

In the figure below (Figure 3.6) you can find tables of Book Organizer and the relations between them.

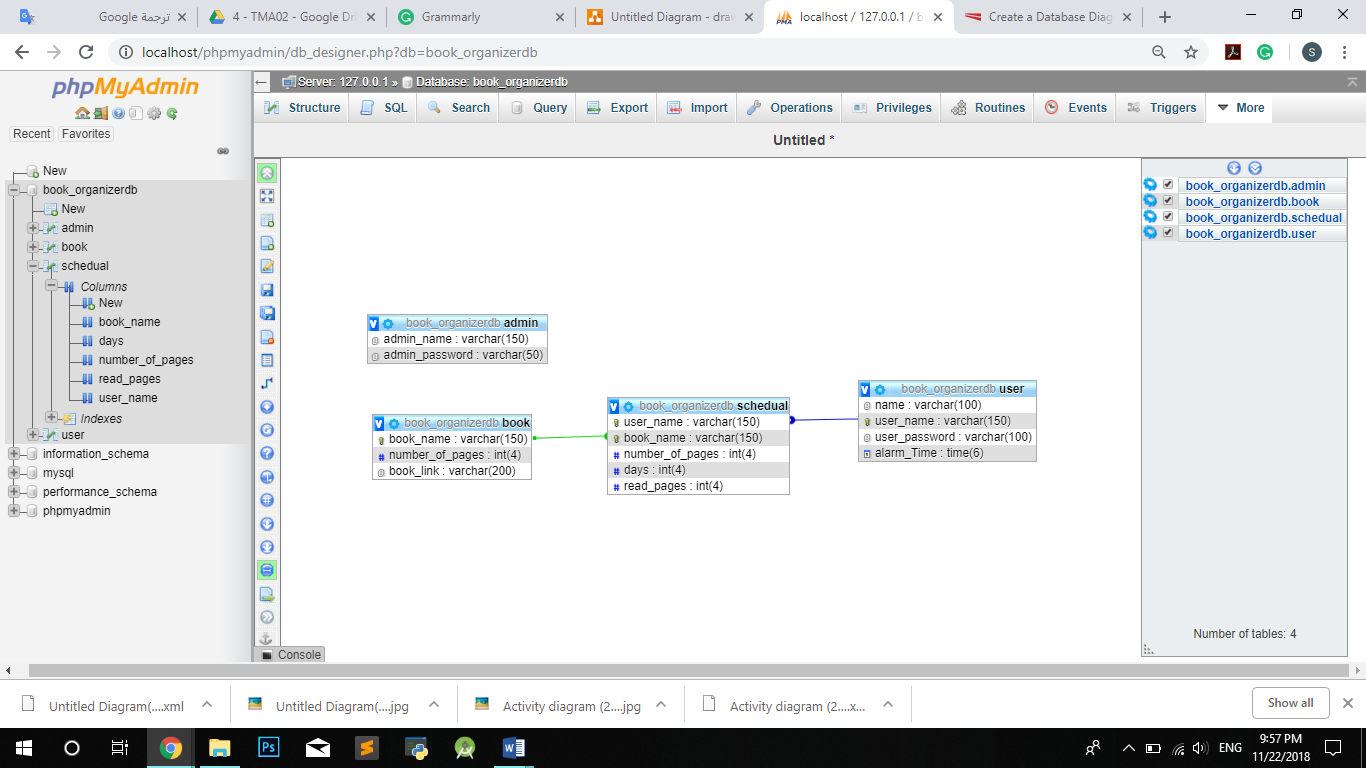
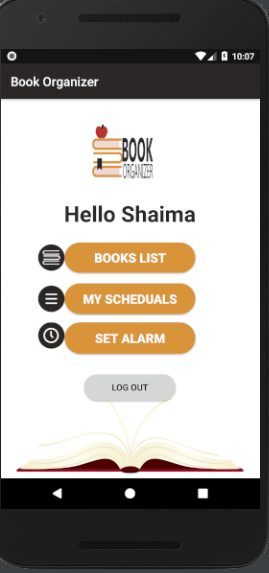
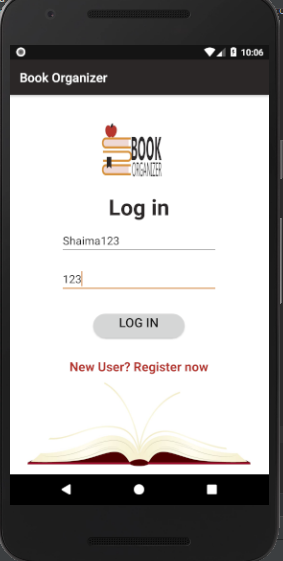
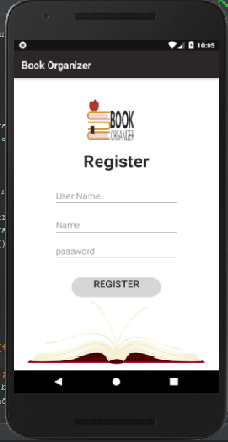
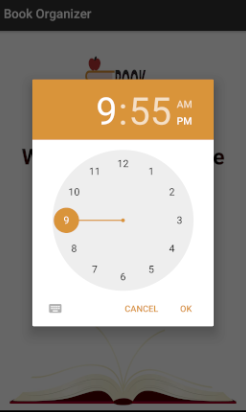


Figure 3.6 Book Organizer Database (UML diagram)

Figure (Figure 3.7) shows interfaces of Book Organizer which are completely designed.

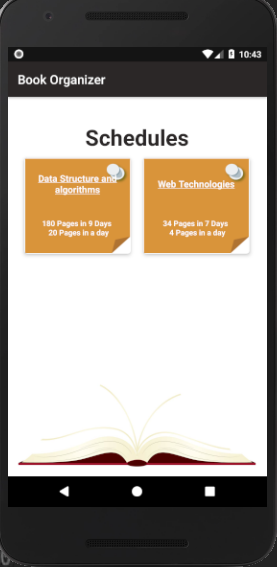
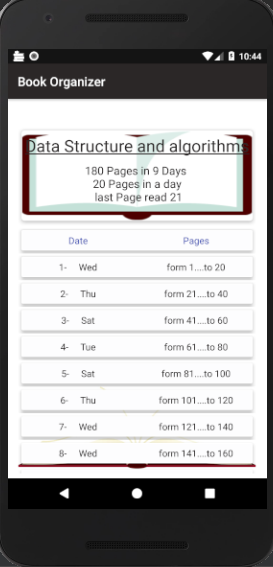


**Register Login Home page**

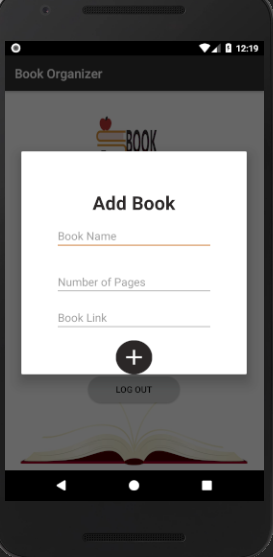
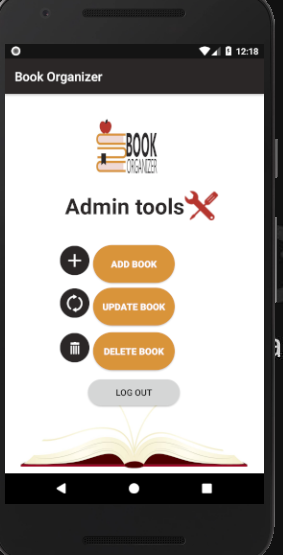


**Set Alarm open clock**

Figure 3.7 Book Organizer fianl interfaces



**Books List Schedule My Schedules**

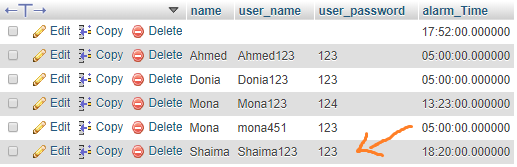


**Admin interface**

Figure 3.8 Book Organizer interfaces 2

**How is Book Organizer Being Implemented?**

After database designing and implementing, the code in Android studio is started written as follows. For each interface, I start writing Its XML code then java code and finally write PHP files which were created as an interface between database and application. I started with register then login etc. Figures below shows how some application pages’ work. (please observe that you can find source code in The attached CD).



**Login Registered users in database**

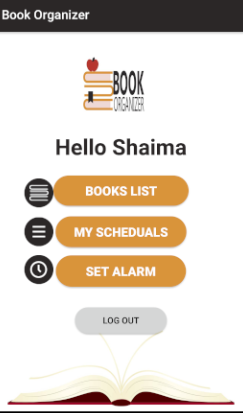
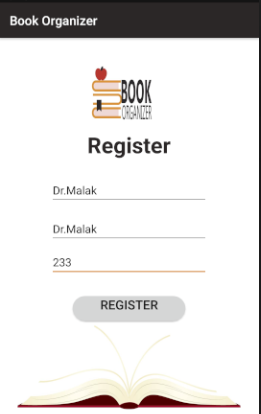
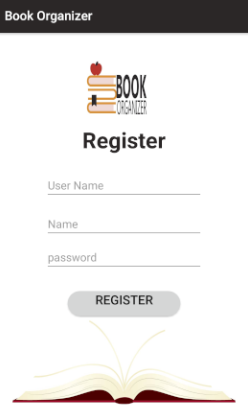
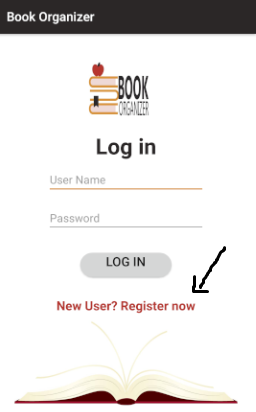


Figure 3.9 Login procces

Figure (Figure 3.9) explains how to log in is done. Once the user inserts its username and password this information is sent by a method called performUserLogin(username, password) to PHP file called login. Login file is retrieving all users in the database (user) table and check whether the username is existing or not and whether the passwords match or not. If the user existing and password is correct login file send a response, username alarm time to the app by getresponse(),getUserName(), and getAlarmTime methods then the app will move to the user`s home page.



**Click new user Register Registration**

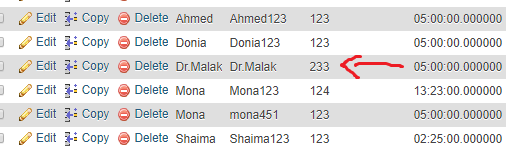
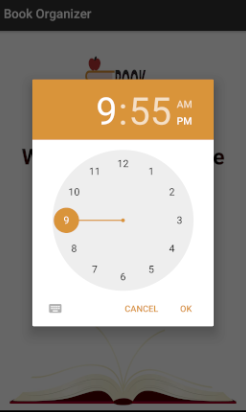
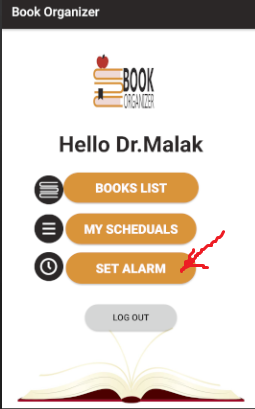
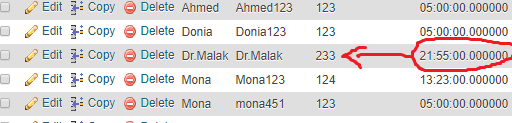
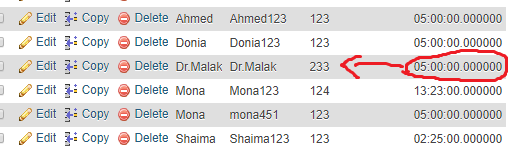


Figure 3.10 Register procces

On the other hand, if the user clicks “new user” text View (Figure 3.10). the app will move to registration fragment. Once a user enters the information, registration fragment sends information to PHP file called Registration throw a method called preformRegisteration(username, name, password). Registration file checks if this username is existing or not. If it is not existing new user will be added to the database (user) table.



**Click set alarm Set alarm page set Time**



**Default alarm time After set Time**

Figure 3.11 Set Time procces

Another example, set notification time once the user chooses the time, it will be sent to a PHP file called set alarm by a method called setAlarm(username, alarmTime). After that, set alarm file set the time to the username in the database that matches the username retrieved from setAlarm class in Android studio.

**Testing**

For debugging Book Organizer, I used an emulator. However, in order to verify users and validate the information entered by users some if statements have written in the code (found in the attached CD). In addition, HTML (found the attached CD in PHP files) files are created in order to check that PHP files are work. Also, toast messages have created in order to check. As an example, if the user registers an existing username message in figure (Figure 4.1) will appear.

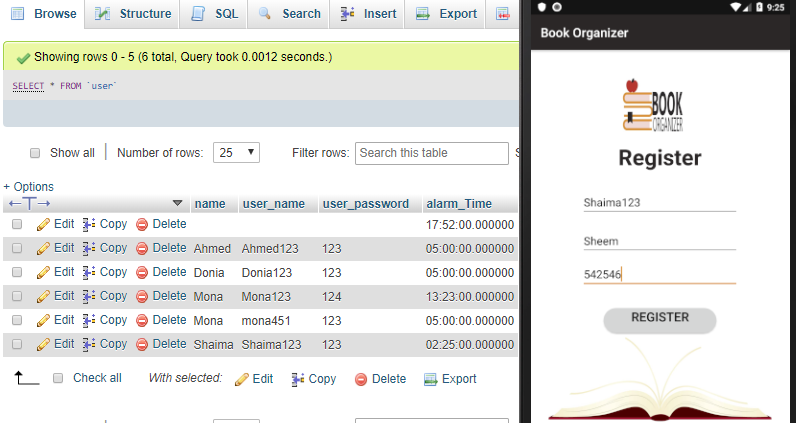
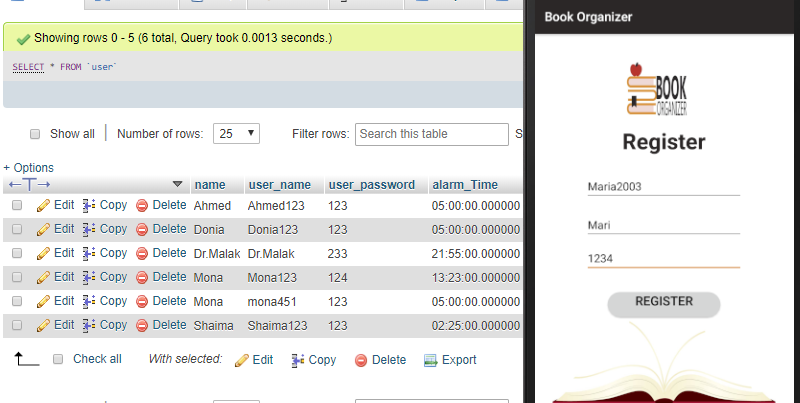
****



Figure 4.1 Register existing username

However, if users insert a valid information such as an example in figure (Figure 4.2) the registration will success.

****

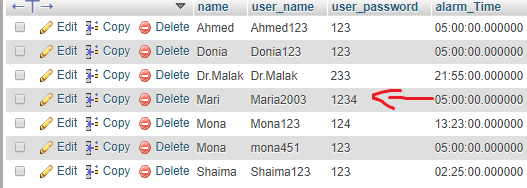
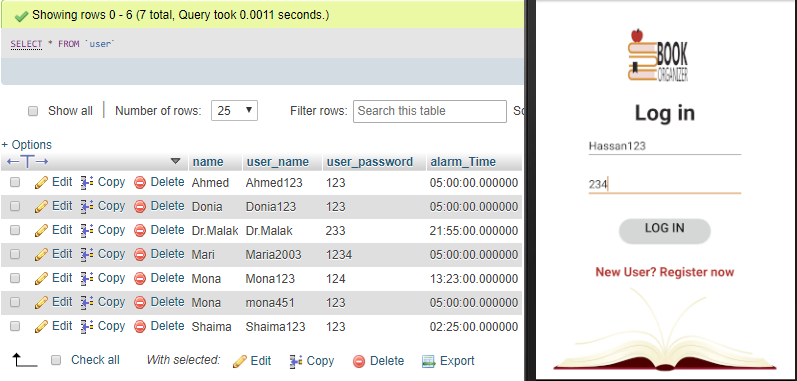


Figure 4.2 Register nonexistent username

On the other hand, figure (Figure 4.3) explains the result if the user tries to login a non-existent username. Also figure (Figure 4.4) shows what happen if the user enters wrong password.



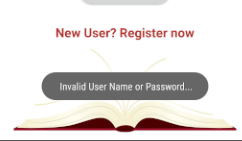
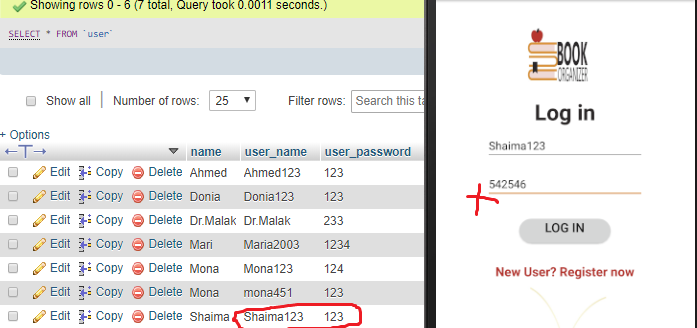


Figure 4.3 Login nonexistent username



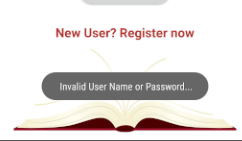


Figure 4.4 Login wrong password

**Result and Discussion**

**What are the Findings from Developing Book Organizer, Goal Achieved, and Future work?**

After a lot of researches and compressions, I found that Book Organizer is a unique system although it may have some similar functionalities with other applications. However, Book Organizer`s main function is not existing in any other system which is organizing book`s pages for readers.

During the journey of developing Book Organizer, much experience has been gained. In addition to developing Android applications, developing Book Organizer give me the opportunity to put all courses that I have learned in practice. For example, designing a database and implement it. Also, moving through the stages of the software development life cycle from the analyzing stage to the test stage. In fact, almost all courses that I have learned were used such as M359, M275, M257, M363, and 215b. In addition, XML language was learned and used to develop the app interfaces. More than that, I have learned how to recognize technical problems and how to search and think professionally in order to find a solution. Further, I have learned how to read the documentation of Android predefined objects.

After this wonderful journey and worthy experience, an Android application called Book Organizer is developed. An application that enables readers to create accounts and login in order to schedule their book`s pages in order to make reading easy as well as daily by sending notifications to the reader. Moreover, it gives the reader the ability to set his notification time.

Although Book Organizer developing almost finished, the journey did not finish. More improvements and features could be added to Book Organizer in the future. For example, Book Organizer could have the ability to extract a graph that explains reading average each month for the reader. Furthermore, an honorary list could be added to Book Organizer that contains reader with the highest reading average in the month. This feature encourages readers to read more hence reading average will increase.

**Conclusions**

It has been said “Today reader tomorrow leader”. Reading importance cannot be denied. Therefore, more attention must be directed towards developing systems that help readers and encourage people to read. Although a lot of applications were developed to enable electronic reading, no applications were developed to make reading wieldier, interesting, and encourage daily reading. Therefore, Book Organizer was developed. An application that makes a schedule for the reader to organize the book`s pages and help in some days specified by the reader. In addition, notify the reader every day to read. Further, it allows new visitors to create accounts and enjoy the journey of reading book in an organized way.

In order to develop Book Organizer professionally, the project plan was made. Also, as any software project, an approach must be chosen in order to develop the project. In Book Organizer, an alternative model was used for developing due to the appropriateness for the requirements and project context.

During developing, a lot of changes in functional and non-functional requirements have done in order to make Book Organizer special, interactive, and easy to be used by readers.

At the end of the journey of developing Book Organizer, I realized that much experience was gained such as developing Android applications, learning XML, and recognizing problems and solving them and much more.

# **References**

Billington, D., Dowrick, P., Hamer, D., Robinson, D., & Williams, D. (2010). *An investigation into the therapeutic benefits of reading in relation to depression and well-being.* Liverpool: LivHIR Institute.

GALLAGHER-MITCHELL, M. (2017, 9). *An investigation into the therapeutic benefits of reading in relation to depression and well-being*. Retrieved from The Reader: https://www.thereader.org.uk/investigation-therapeutic-benefits-reading-relation-depression-well/

*Top 10 Health Benefits of Reading*. (2015, 5). Retrieved from Health fitness Revalution: http://www.healthfitnessrevolution.com/top-10-health-benefits-reading/

West, M., & Ei Chew, H. (2014). *Reading in the mobile era: a study of mobile reading in developing countries.* Paris: UNESCO.

Winter-Hébert, L. (2017). *10 Benefits of Reading: Why You Should Read Every Day*. Retrieved from lifehack: https://www.lifehack.org/articles/lifestyle/10-benefits-reading-why-you-should-read-everyday.html

Govardhan, N. M. (2010). A Comparison Between Five Models Of Software Engineering . In *International Journal of Computer Scienc Issues* (pp. 94-101). www.IJCSI.org.

*What is Iterative Model?* (2018, jan). Retrieved from professionalQA.com: http://www.professionalqa.com/iterative-model

**Appendices**

init.php: - which is the file that create the connection to the database it is required by all PHP files

<?php

$dp\_name= "book\_organizerdb";

$user="root";

$pass="";

$host= "localhost";

$conn = new mysqli($host,$user,$pass,$dp\_name);

?>

Login process code:

Login.XML: -

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingTop= "5dp"  
 android:paddingBottom= "5dp"  
 android:paddingLeft= "5dp"  
 android:paddingRight= "5dp"  
 tools:context=".MainActivity"  
 android:orientation="vertical"  
 android:background="#ffffff">  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:src="@drawable/probackground"/>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="109dp"  
 android:text="Log in"  
 android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"  
 android:textColor="#2A2728"  
 android:textSize="35sp"  
 android:textStyle="bold" />  
  
 <EditText  
 android:id="@+id/user\_name"  
 android:layout\_width="250dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="164dp"  
 android:hint="User Name" />  
  
  
 <EditText  
 android:id="@+id/user\_password"  
 android:layout\_width="250dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="225dp"  
 android:hint="Password" />  
  
 <Button  
 android:id="@+id/login\_btn"  
 android:layout\_width="145dp"  
 android:layout\_height="38dp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="204dp"  
 android:textSize="20sp"  
 android:background="@drawable/buttoun"  
 android:gravity="center\_horizontal"  
 android:text="log in" />  
  
 <TextView  
 android:id="@+id/register\_text"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="145dp"  
 android:text="New User? Register now"  
 android:textColor="#B5362F"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
  
  
</RelativeLayout>

Login.java: -

package com.example.mypc.bookorganizer;  
  
  
import android.app.Activity;  
import android.content.Context;  
import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
  
public class login extends Fragment {  
 private TextView regText;  
 private EditText UserName,UserPassword;  
 private Button Loginbtn;  
 OnLogFromActivityListener logFromActivityListener;  
 public interface OnLogFromActivityListener  
 {  
 public void preformregister();  
 public void preformlogin(String name,String UserName);  
 public void preformAdminlogin();  
 }  
  
  
 public login() {  
 // Required empty public constructor  
 }  
  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view= inflater.inflate(R.layout.*fragment\_login*, container, false);  
 UserName = view.findViewById(R.id.*user\_name*);  
 UserPassword = view.findViewById(R.id.*user\_password*);  
 Loginbtn = view.findViewById(R.id.*login\_btn*);  
 Loginbtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 performLogin();  
 }  
 });  
  
  
 regText = view.findViewById(R.id.*register\_text*);  
 regText.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 logFromActivityListener.preformregister();  
  
 }  
 });  
 return view;  
 }  
  
 @Override  
 public void onAttach(Context context) {  
 super.onAttach(context);  
 Activity activity =(Activity) context;  
 logFromActivityListener= (OnLogFromActivityListener) activity;  
 }  
 public void performLogin()  
 {  
 String username = UserName.getText().toString();  
 String password = UserPassword.getText().toString();  
 Call<User> call= MainActivity.*apiInterface*.preformUserLogin(username,password);  
 call.enqueue(new Callback<User>() {  
 @Override  
 public void onResponse(Call<User> call, Response<User> response) {  
 if (response.body().getResponse().equals("ok"))  
 {  
 logFromActivityListener.preformlogin(response.body().getName(),response.body().getUserName());  
 MainActivity.*prefConfig*.writeLoginStatus(true);  
 }  
  
 else if (response.body().getResponse().equals("failed"))  
 {  
 MainActivity.*prefConfig*.displayToast(" Invalid User Name or Password...");  
 }  
 else if (response.body().getResponse().equals("admin"))  
 {  
 logFromActivityListener.preformAdminlogin();  
 }  
 }  
  
 @Override  
 public void onFailure(Call<User> call, Throwable t) {  
  
 }  
 });  
 UserName.setText("");  
 UserPassword.setText("");  
  
 }  
  
}

Login.php: -

<?php

require "init.php";

$user\_name=$\_GET["user\_name"];

$user\_password=$\_GET["user\_password"];

$sql="select \* from admin where admin\_name='$user\_name' and admin\_password='$user\_password'";

$result = mysqli\_query($conn,$sql);

if (mysqli\_num\_rows($result)> 0)

{

$status="admin";

echo json\_encode(array("response"=>$status));

}

else

{

$sql="select \* from user where user\_name='$user\_name' and user\_password='$user\_password'";

$result = mysqli\_query($conn,$sql);

if(!mysqli\_num\_rows($result)> 0)

{

$status="failed";

echo json\_encode(array("response"=>$status));

}

else

{

$row = mysqli\_fetch\_assoc($result);

$name= $row['name'];

$status="ok";

echo json\_encode(array("response"=>$status,"name"=>$name,"user\_name"=>$user\_name));

}}

mysqli\_close($conn);

?>

LoginTest.html: -

<html>

<titele> login test</titele>

<body>

<form action="http://localhost/loginApp/login.php" method="get" >

<table>

<tr>

<td>User Name:</td>

<td><input type="text" name="user\_name"></td>

</tr>

<tr>

<td>User password:</td>

<td><input type="text" name="user\_password"></td>

</tr>

<tr>

<td><input type="submit" value="Log in"></td>

</tr>

</table>

</form>

</body>

</html>

Register process code:

register.XML: -

?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"  
 android:paddingTop= "5dp"  
 android:paddingBottom= "5dp"  
 android:paddingLeft= "5dp"  
 android:paddingRight= "5dp"  
 android:orientation="vertical"  
 android:background="#ffffff">  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:src="@drawable/probackground"/>  
  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="101dp"  
 android:text="Register"  
 android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"  
 android:textColor="#2A2728"  
 android:textSize="35sp"  
 android:textStyle="bold" />  
  
 <EditText  
 android:id="@+id/text\_name"  
 android:layout\_width="250dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="236dp"  
 android:hint="Name" />  
  
 <EditText  
 android:id="@+id/text\_username"  
 android:layout\_width="250dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="175dp"  
 android:hint="User Name" />  
  
 <EditText  
 android:id="@+id/text\_password"  
 android:layout\_width="250dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="211dp"  
 android:hint="password" />  
  
 <Button  
 android:id="@+id/register\_btn"  
 android:layout\_width="179dp"  
 android:layout\_height="39dp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginBottom="141dp"  
 android:background="@drawable/buttoun"  
 android:gravity="center\_horizontal"  
 android:text="Register"  
 android:textSize="20sp" />  
  
  
  
</RelativeLayout>

register.java: -

package com.example.mypc.bookorganizer;  
  
  
import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.EditText;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
  
*/\*\*  
 \* A simple {****@link*** *Fragment} subclass.  
 \*/*public class registration extends android.app.Fragment {  
 private EditText Name, UserName,UserPassword;  
 private Button Registerbtn;  
  
  
 public registration() {  
 // Required empty public constructor  
 }  
  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view= inflater.inflate(R.layout.*fragment\_registration*, container, false);  
 Name= view.findViewById(R.id.*text\_name*);  
 UserName = view.findViewById(R.id.*text\_username*);  
 UserPassword = view.findViewById(R.id.*text\_password*);  
 Registerbtn = view.findViewById(R.id.*register\_btn*);  
 Registerbtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 performRegistration();  
 }  
 });  
 return view;  
 }  
 public void performRegistration()  
 {  
 String name = Name.getText().toString();  
 String username = UserName.getText().toString();  
 String password = UserPassword.getText().toString();  
 Call<User> call= MainActivity.*apiInterface*.preformRegistration(name,username,password);  
 call.enqueue(new Callback<User>() {  
 @Override  
 public void onResponse(Call<User> call, Response<User> response) {  
 if (response.body().getResponse().equals("ok"))  
 {  
 MainActivity.*prefConfig*.displayToast("Registration success...");  
 }  
 else if (response.body().getResponse().equals("exit"))  
 {  
 MainActivity.*prefConfig*.displayToast("User already exist...");  
 }  
 else if (response.body().getResponse().equals("error"))  
 {  
 MainActivity.*prefConfig*.displayToast("Something went wrong...");  
 }  
 }  
  
 @Override  
 public void onFailure(Call<User> call, Throwable t) {  
  
 }  
 });  
 Name.setText("");  
 UserName.setText("");  
 UserPassword.setText("");  
  
 }  
  
}

register.php: -

<?php

require "init.php";

$name= $\_GET["name"];

$user\_name= $\_GET["user\_name"];

$user\_password= $\_GET["user\_password"];

$sql="select \* from user where user\_name='$user\_name'";

$result = mysqli\_query($conn,$sql);

if(mysqli\_num\_rows($result)> 0)

{

$status="exit";

}

else

{

$sql= "insert into user(name,user\_name,user\_password) values ('$name','$user\_name','$user\_password')";

if (mysqli\_query($conn,$sql))

{

$status ="ok";

}

else

{

$status ="error";

}

}

echo json\_encode(array("response"=>$status));

mysqli\_close($conn);

?>

register\_test.html: -

<html>

<titele> register test</titele>

<body>

<form action="http://localhost/loginApp/register.php" method="get" >

<table>

<tr>

<td>Name:</td>

<td><input type="text" name="name"></td>

</tr>

<tr>

<td>User Name:</td>

<td><input type="text" name="user\_name"></td>

</tr>

<tr>

<td>User password:</td>

<td><input type="text" name="user\_password"></td>

</tr>

<tr>

<td><input type="submit" value="Register"></td>

</tr>

</table>

</form>

</body>

</html>

Set Alarmprocess code:

SetAlarm.XML: -

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".AlarmFragment"  
 android:background="#ffffff">  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentTop="true"  
 android:src="@drawable/probackground" />  
 <TextView  
 android:id="@+id/alarm\_time"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginTop="135dp"  
 android:gravity="center\_horizontal"  
 android:text="When would you like Me to notify you?"  
 android:textColor="#2A2728"  
 android:textSize="35sp"  
 android:textStyle="bold" />  
  
  
 <Button  
 android:id="@+id/clock\_btn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginBottom="248dp"  
 android:background="@drawable/colored\_button"  
 android:text="Open clock"  
 android:textColor="#fdfdfe"  
 android:textSize="20sp"  
 android:textStyle="bold"/>  
  
  
  
</RelativeLayout>

SetAlarm.java: -

package com.example.mypc.bookorganizer;  
  
  
  
  
import android.app.TimePickerDialog;  
import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.support.v4.app.DialogFragment;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.TextView;  
  
import android.widget.TimePicker;  
  
  
*/\*\*  
 \* A simple {****@link*** *Fragment} subclass.  
 \*/*public class AlarmFragment extends Fragment {  
 private Button clockBtn;  
  
  
 public AlarmFragment() {  
 // Required empty public constructor  
 }  
  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view= inflater.inflate(R.layout.*fragment\_alarm*, container, false);  
  
 clockBtn= view.findViewById(R.id.*clock\_btn*);  
 clockBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 DialogFragment clock= new TimeClock();  
 clock.show(getActivity().getSupportFragmentManager(),"Time Picker");  
  
 }});  
 return view;  
 }  
  
  
}

Time clock Object: -

package com.example.mypc.bookorganizer;  
  
import android.app.Activity;  
import android.app.Dialog;  
import android.app.TimePickerDialog;  
import android.os.Bundle;  
import android.support.annotation.NonNull;  
import android.support.annotation.Nullable;  
import android.support.v4.app.DialogFragment;  
import java.util.Calendar;  
  
public class TimeClock extends DialogFragment {  
  
  
 @NonNull  
 @Override  
 public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
 Calendar c = Calendar.*getInstance*();  
 int hour = c.get(Calendar.*HOUR\_OF\_DAY*);  
 int minute = c.get(Calendar.*MINUTE*);  
  
 return new TimePickerDialog(getActivity(), (TimePickerDialog.OnTimeSetListener)getActivity(), hour, minute, android.text.format.DateFormat.*is24HourFormat*(getActivity()));  
 }  
  
 }

SetAlarm.php: -

<?php

require "init.php";

$user\_name= $\_GET["user\_name"];

$alarm\_Time=$\_GET["alarm\_Time"];

$sql="UPDATE user SET alarm\_Time = '$alarm\_Time' WHERE user.user\_name= '$user\_name'";

if (mysqli\_query($conn, $sql))

{

$status ="Alarmset";

}

else

{

$status ="error";

}

echo json\_encode(array("response"=>$status));

mysqli\_close($conn);

?>

SetAlarm\_test.html: -

<html>

<titele> set alarm test</titele>

<body>

<form action="http://localhost/loginApp/setAlarm.php" method="get" >

<table>

<tr>

<td>User Name:</td>

<td><input type="text" name="user\_name"></td>

</tr>

<tr>

<td>Time:</td>

<td><input type="text" name="alarm\_Time"></td>

</tr>

<tr>

<td><input type="submit" value="add"></td>

</tr>

</table>

</form>

</body>

</html>