FINAL REPORT

Daily Bible

Software Engineering CSCE 3444

DEVOTED
GLORIA KIM

Contents

1. System Overview 2

1.1. System description 2

1.2. System Features 2

2. Implementation 3

3. Challenges 5

4. Test Cases 6

5. How to Use? 10

6. Future Works 10

7. Member Contribution 10

8. Appendix: Code .... 오류! 책갈피가 정의되어 있지 않습니다.

1. System Overview

1.1. System description

This project is about developing an app named Daily Bible to encourage people to read the Bible by sharing their notes and having a frequent alarm.

The problem that this project wants to solve is to provide motivation for Christians to read more Bible so that they can know more about God and become more like Him.

1.2. System Features

This project includes mainly four pages to code: (1) FragmentMain, (2) FragmentNote, (3) FragmentAlarm, (4) FragmentRecord. Each page is listed below with the associated features:

(1) FragmentMain
- Show setting
    - Users can change the font size.
- Show credit
    - Each credit name is linked to the original website.
- Show today's date
- Show today's Bible verse
- Show the number of days that read the Bible verse (e.g., Day 3)
- Show the complete button
    - When users finish reading the Bible, they click the complete button to move on to the next Bible verse tomorrow.

(2) FragmentNote
- Show private notes of users
- Create private notes of users
- Edit private notes of users
- Delete private notes of users
- Show public notes of users
- Create public notes of users (Please note that the public note cannot be edited or deleted.)

(3) FragmentAlarm
- Set alarm/notification time
- Set reading method (i.e., 90-day challenge, bible order, history order)

(4) FragmentRecord
- Show the total number of completions to read the entire Bible
- Show the Bible icons which its number is same as the total number of completions

This project has the following functional requirements:

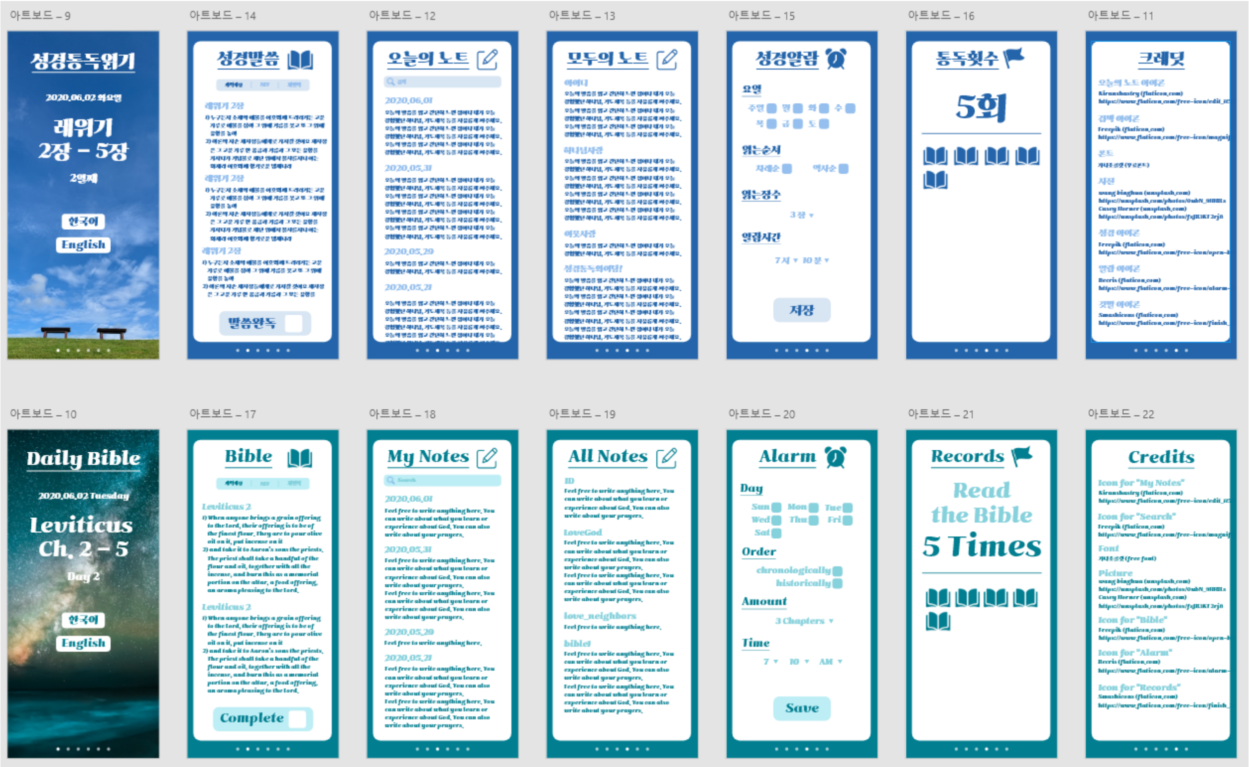
- New users should allow the app to control the alarm system of the mobile phones.
- Users should agree to share their public notes to save in external database.
- Users should agree to save their private notes in their phone memory.
- No users (including administrators) can see the private notes of users
- Users do not need to go through the registration process. They simply need to install the app to use its functions.
- Only administrators can see the statistics of all users.

I was able to implement all functional requirements listed above.

2. Implementation

Design

I used **Adobe XD** to draft my first prototype of the app (as well as design modification in future) as follows:

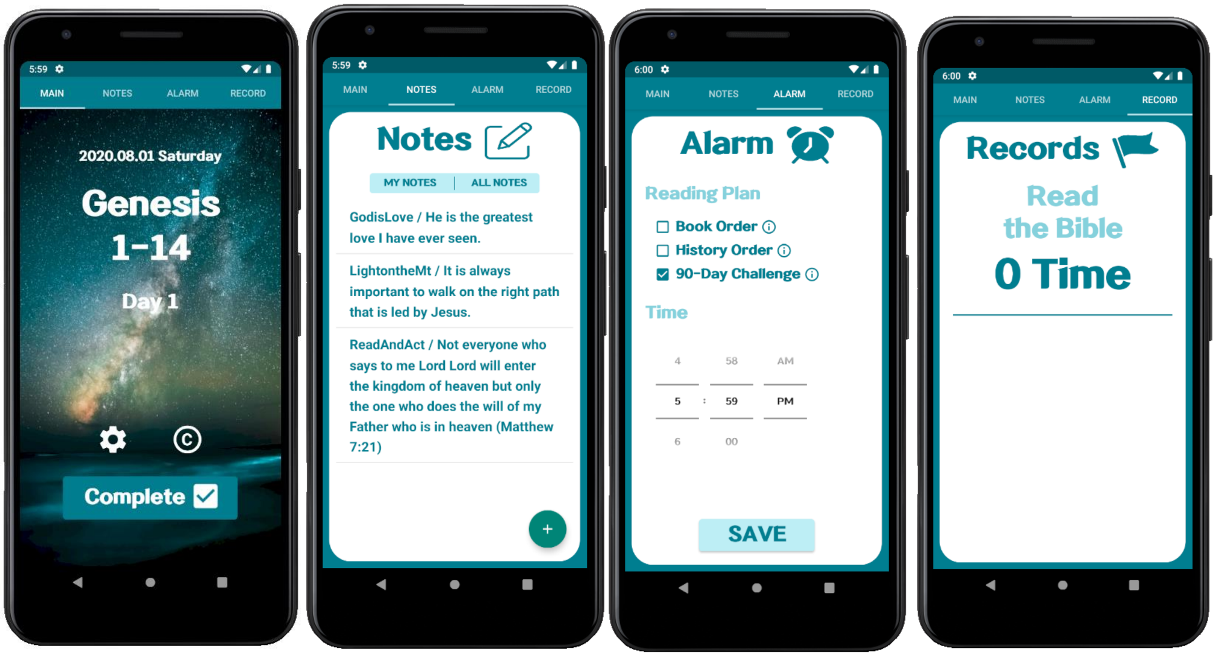


To elaborate the design of the app, I used a number of websites to obtain free icons and photo, which their creditors are listed in the credit page (the credit page can be found in the main page). I also used the online PNG tool to change the color of free icons.

Android App

To create an actual android app, I used my IDEs as **Android Studio**. Using Android Studio, I coded the design aspects in XML files and functional aspects in Java files. I used internal storage to store private notes, and I used Firebase to store public notes. To test my codes, I used several Android emulators, including Pixel 3a API 28, Pixel 2 API 25, and Nexus 5 API 30. Although the minimum required API for this project is 21, I did not test out the app for each and every emulator due to the limited time.

The images of the interfaces are displayed using the android emulator of Pixel 3a as below:



GodisLove / He is the greatest love I have ever seen.

LightontheMt / It is always important to walk on the right path that is led by Jesus.

ReadAndAct / Not everyone who says to me Lord Lord will enter the kingdom of heaven but only the one who does the will of my Father who is in heaven (Matthew 7:21)

Reading Plan

□ Book Order ⓘ
□ History Order ⓘ
☑ 90-Day Challenge ⓘ

Time

```
4    58   AM
5 :  59   PM
6    00
```

SAVE

Read the Bible 0 Time

Implementation

To create a prototype using Adobe XD, I followed the steps below:

(1) Find a good background photo
(2) Extract one main color from that background
(3) Change hue and saturation to pick lighter and darker colors
(4) Search for similar app designs
(5) Create my design
(6) Create interactions if necessary (Because I am the only programmer/designer for this project, I did not create any interactions.)

To implement the project in Android Studio, I followed the steps below:

(1) Create all XML files
(2) Create MainActivity.java, which would be the base to connect fragment pages (i.e., FragmentMain.java, FragmentNote.java, FragmentAlarm.java, FragmentRecord.java).
(3) Create fragments
(4) Create activities (i.e., settings and credit) to connect with FragmentMain.java
(5) Connect XML files to all java files (i.e., fragments and activities)
(6) Create functionalities for one fragment
(7) Test functionalities for that one fragment
(8) Repeat (6) and (7) for all fragments
(9) Create and test functionalities for all activities
(10) Test the app as a whole
(11) Fix any bugs

3. Challenges

What challenges did you face while developing this system? What changes did you have to make to your requirements or system features in response to these challenges?

I have several challenges throughout the development of the app as below:

(1) Refreshing Fragment with *SQLite* (Successfully Resolved)

When I used *SQLite* in Fragment, the result did not show unless the app is restarted due to the nature of fragment. To resolve the issue, I needed to refresh the fragment every time I committed a data using *SQLite*. Somehow, the FragmentManager did not work (I still have not figured out why), so I created OnActivityResult in both fragment and activity to refresh the fragment again.

(2) Listing the retrieved data from Firebase (Partially Resolved)

When I retrieved data from Firebase, I could only retrieve a long string value that contains all data. I could not figure out how to nicely list out each data, so I decided to separate each item by using the fact that the string composed of many data values are separated by comma. However, this creates an issue for users not to use comma, which will then separate their notes.

(3) Having many *AlarmManager* (Removed)

The current *AlarmManager* allows up to two days to set an alarm together. For instance, the *AlarmManager* allows to set an alarm at 7AM for Sunday and Monday, but no for Sunday through Tuesday and so forth. To resolve the issue, I created seven different *AlarmManager*, each associated to a specific day (i.e., Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday). However, the alarm goes on and on whenever I changed the android emulator's date to test out functions, so I simply removed the entire functions to set days, hoping that users are okay to read the Bible daily.

(4) Saving the Alarm (Successfully Resolved)

Whenever I restarted the app, I had issues that the app automatically resets the alarm. I used *SharedPreferences* to save the alarm setting with two different variables, *reading_plan* and *reading_day*. I made reading_plan to save the reading method (i.e., 90-day challenge, bible order, history order), and I made reading_day to count the number of days since the alarm was set. This way, I could allow users to keep track of their reading plan.

(5) Saving the Completion Button (Successfully Resolved)

Again, whenever I restarted the app, I had issues that the app automatically resets the completion button. I created another variable *completed* using *SharedPreferences* to keep track if a user already clicked the button. I also used *Calendar* to allow users to check the completion button only once a day.

(6) Writing Arrays with Bible Verses (Successfully Resolved)

I originally planned to have all Bible verses in Firebase, but I could not find a good tutorial to perform such action. I decided to create arrays in the java files and wrote each Bible verse in the java files. Because there are three different plans, I had to create three different arrays with two different version: one with the string *Day* for notifications (e.g., "Day 1 – Genesis 1 - 4) and the other without the string *Day* for FragmentMain (e.g, “Genesis 1 – 4”). This entire process was very time-consuming, but I managed to complete the functions that I wanted.

(7) Having Copyrighted Materials (Removed)

I found that NIV is not freely available and that KJV is difficult to understand. I simply removed the entire FragmentBible.

(8) Korean version (Removed)

I originally planned to have a Korean version, but due to time limit, I removed that version.

4. Test Cases

Please note that I have tested several times and had the final testing on July 31, 2020 to make sure everything works, so I simply set the test date as July 31, 2020.

Test Case ID	**TU01_ActivityMain**	**Tested by**	**08-05-2020**	**Test Date**	**07-31-2020**
Test Case Description	Check the top navigation tap and view				
Preconditions	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Access to Android Emulator	1. Open the android emulator 2. Check if the tap is titled appropriately 3. Check the tap is clicking 4. Check if the tap is successfully moving to each fragment 5. Check if each fragment’s content is successfully showing	(clicks and views)	Each tap should connect to each fragment and show the fragment’s content.	As Expected	Pass

Test Case ID	**TU02_FragmentMain**	**Tested by**	**08-05-2020**	**Test Date**	**07-31-2020**

Test Case Description	Check (1) the completion button, (2) today's date, (3) today's bible verse, (4) reading day, (5) setting, and (6) credit				
Preconditions	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Access to Android Emulator	1. Open the android emulator 2. Check if the completion button enables once per day 3. Check if the completion button stays clicked once it is clicked 4. Check if the completion button is clicked, (2), (3) and (4) are changed 5. Check if today's date is accurately appearing 6. Check if today's bible verse is accurately appearing based on the chosen reading method 7. Check if reading day's number is accurately appearing 8. Check if setting reflects the changes (i.e., change the font-size in Setting immediately change the font-size of all app) 9. Check if credit page has all successful hyperlinks	(clicks and views)	All content should appear the accurate data based on the reading plan that is set using FragmentAlarm.	As Expected	Pass

Test Case ID	**TU03_FragmentNote**	**Tested by**	**08-05-2020**	**Test Date**	**07-31-2020**

Test Case Description	Check (1) private note can be created, edited, deleted, and displayed and (2) public note can be created and displayed				
Preconditions	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Access to Android Emulator	1. Open the android emulator 2. Check if the floating button is clickable and leads the users to the new page to create notes 3. Check if the note can be saved in private space 4. Check if the saved note in private space can be clicked and be directed to edit or delete the note 5. Check the edit functionality of private note 6. Check the delete functionality of private note 7. Check if the private note is displayed as expected 8. Check if the note can be saved in public space (Firebase) 9. Check if the public note is displayed as expected	For private data, testers can ignore the user ID part. For public data, testers should insert the public ID part.	The notes are created, edited, deleted, and displayed without any errors.	As Expected	Pass

Test Case ID	**TU04_FragmentAlarm**	**Tested by**	**08-05-2020**	**Test Date**	**07-31-2020**
Test Case Description	Check (1) reading plan and (2) notification				

Preconditions	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Access to Android Emulator	1. Open the android emulator 2. Test if different reading plan reflects different notification 3. Test if the different time setting is reflected on the notification time 4. Test if notification still comes in different days 5. Test if notification comes as the set time in different days	(Click the checkboxes and the save button. Use the android emulator's setting to change the date.)	Notification accurately reflects the user's reading plan setting.	As Expected	Pass

Test Case ID	**TU05_FragmentRecords**	**Tested by**	**08-05-2020**	**Test Date**	**07-31-2020**
Test Case Description	Check the number of completion and the number of icons				
Preconditions	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Access to Android Emulator	1. Open the android emulator 2. Change the number of completions manually in java file to see if the number of appeared icon is same as the number of completion 3. Toast messages to check the number of completion	The number of completion = varies (0 to 15)	The number of icon is same as the number of completion, and the number of completion only changes when the user completes	As Expected	Pass

			the entire reading plan		

5. How to Use?

Here is the list of steps with detailed descriptions for users to utilize the app:

(1) Download the app from App Store
(2) Open the app
(3) Set the alarm/notification
(4) Touch the alarm/notification
(5) Check the completion button
(6) Reset the alarm/notification after reading the entire Bible
(7) (Optional) click *Credit* to see the creditors
(8) (Optional) click *Setting* to change the setting

6. Future Works

I would hope to include the future improvements on my application as follows:

(1) More versions to the app (such as Korean version that I originally planned)
(2) Different font sizes
(3) Detailed statistics (rather than having just a simple number of completion)
(4) Many backgrounds (rather than having just one simple photo)
(5) Daily inspiring quotes

7. Member Contribution

Gloria Kim is the sole contributor for this project.