Software Requirements Specifications

Daily Bible

Software Engineering, CSCE 3444, Summer 2020

Devoted

Gloria Kim

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to develop a software requirement specification for the project of creating an android app named Daily Bible.

1.2. Scope

The product is an android app named Daily Bible. The goal of this project is to encourage people to read the Bible by sharing their notes and having a daily alarm so that people know more about God and become more like Him.

1.3. Definitions, acronyms, and abbreviations

In this document, the term “the app” or “the mobile app” refers to Daily Bible.

1.4. Overview

The first section includes introduction, where the project’s purpose, scope, definition, and reference documents are explained. The second section explains the overall description of this project. The description includes product perspective and functions along with user classes and constrains. The third section include specific requirements, which covers external interface, functional and non-functional requirements, and design constraints. The last section shows appendices.

1.5. References

This paper was used resources from Lecture 6 PowerPoint slides.

2. Overall Description

2.1. Product perspective

The product is not a follow-on member of a product family nor a replacement for certain existing system. It is a new, self-contained product that saves shared notes of users to database. Because the product does not require to save any other resources outside of mobile phones, it has only one external database. The diagram is shown in Appendix 4.1.

2.2. Product functions

The mobile app has the following features:

- Set a daily alarm
- Set a daily goal (i.e., number of pages to read)
- Set a way to read (either chronologically or historically)
- Show statistics (i.e., the total number of reading the entire Bible)
- Show users’ notes
- Share users’ notes
- Show shared note of other users

The diagram is shown in Appendix 4.2.

2.3. User classes and characteristics

The intended users are Christians who plan to read Bible daily. Because users should be able to read either English or Korean, the minimum expected age of users is 13 years.

Because this is a mobile app, users are expected to have their own cell phones and are familiar to use mobile apps. Because the app will be released free and available for anyone who is older than 13 years, users are expected to have various backgrounds in terms of education, sex, and income-level.
The diagram is shown in Appendix 4.3.

2.4. Constraints

To legally publish Bible online (which is a very important rule to follow as a Christian), a developer should purchase Bible, which may be very costly. Based on the costs, a developer will decide how many different versions of Bible can be included in the app.

2.5. Assumptions and dependencies

The app Daily Bible is designed to be used in android mobile phones.
The app is tested using an android emulator Pixel 2 API 25. The screen may appear differently in different models.

3. Specific Requirements

3.1. External interface

3.1.1. User interface

The user interface should be simple yet easy-to-navigate. The fonts should be big enough for everyone to clearly read yet small enough to fit in the size of an android phone. The colour-contrast should be different enough for colour-blindness to read the content without worries. The prototype of an app using such rules are shown in Appendix 4.4.

3.1.2. Hardware interface

The required hardware is an android phone.

3.1.3. Software interface

The software requirements are Android Studio as an IDE that uses Java programming language. To develop a prototype, Adobe XD and online PNG (https://onlinepngtools.com/change-png-color) tool are needed. The API known as Firebase is also needed to connect the developed mobile app with the external database. It is also recommended to use OS as Windows to develop the app.

3.1.4. Communication interface

This section is not applicable.

3.2. 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.

3.3. Non-Functional requirements

3.3.1. Usability

All screens can be accessible through the top tabs. 90% of novice users can learn to operate major us cases without outside assistance.

3.3.2. Reliability

The app is expected to have a crash less than 5% (5 fails out of 100 attempts). The MTBF is expected to be at least 2 hours or enough time to write a long note using the app.

3.3.3. Performance requirements

The app should respond the user alarm setting within 2 seconds.
The apps should save the private notes in internal memory within 5 seconds.
The app should save the public notes in external database within 10 seconds.

3.3.4. Safety requirements

The app will be tested using an emulator from Android Studio.

3.3.5. Security requirements

The private notes will be saved in internal memory only and will not be exposed to anywhere else by any mean.
Because the main purpose of this app is to set an alarm that does not deal with sensitive data and that require to respond quickly, it is not necessary to encrypt data, and thus, there is no further security requirements.

3.3.6. Legal

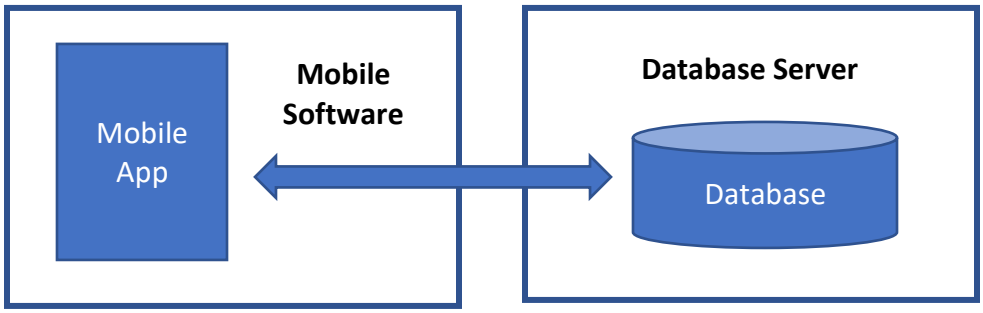
The app serves as a platform to share intellectual property, which its copyright belongs to the original writer. However, when users write public notes, they agree to share their intellectual property for free of charge with anyone.

3.4. Design constraints

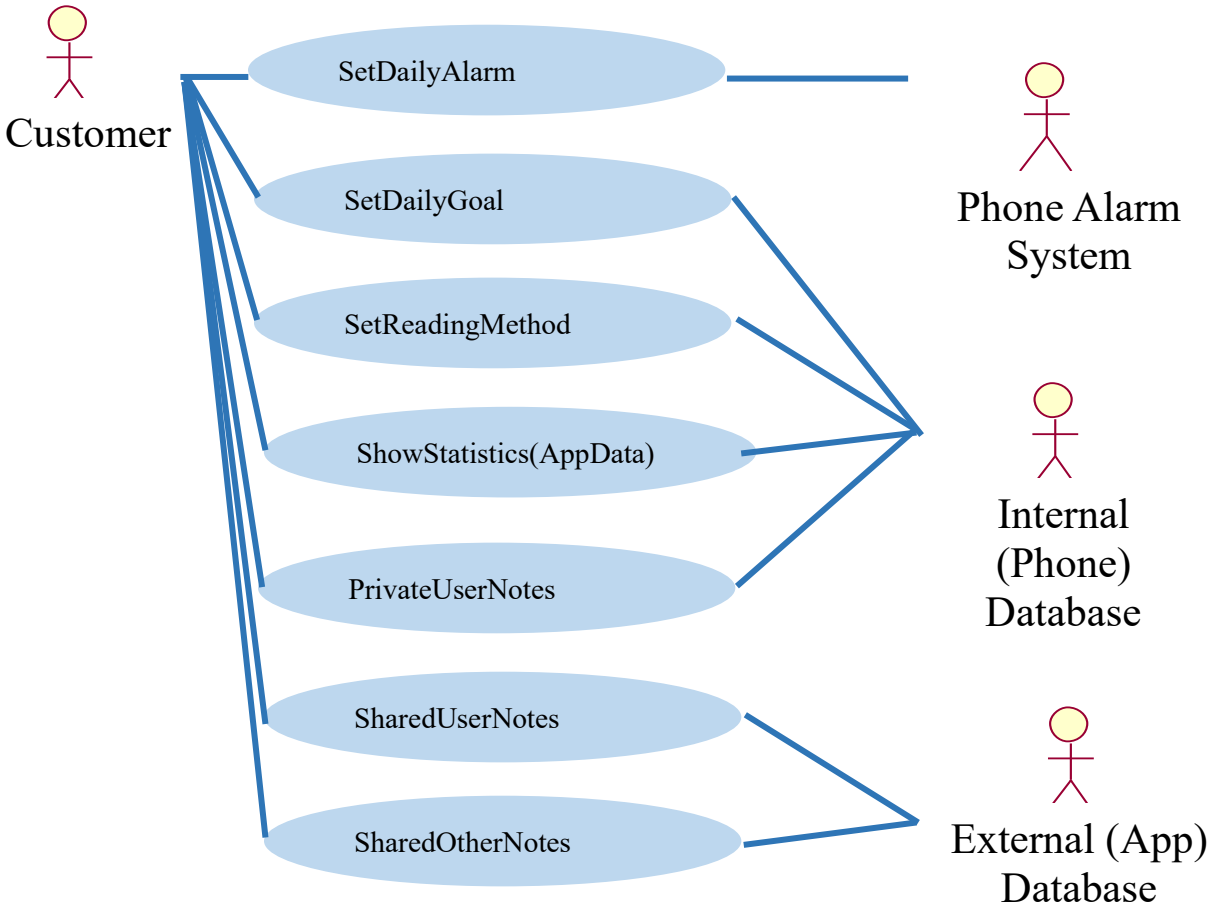
The app is built using Android Studio in Windows. The app is written with Java and XML programming languages. The app is expected to be smaller than 500MB.

4. Appendices

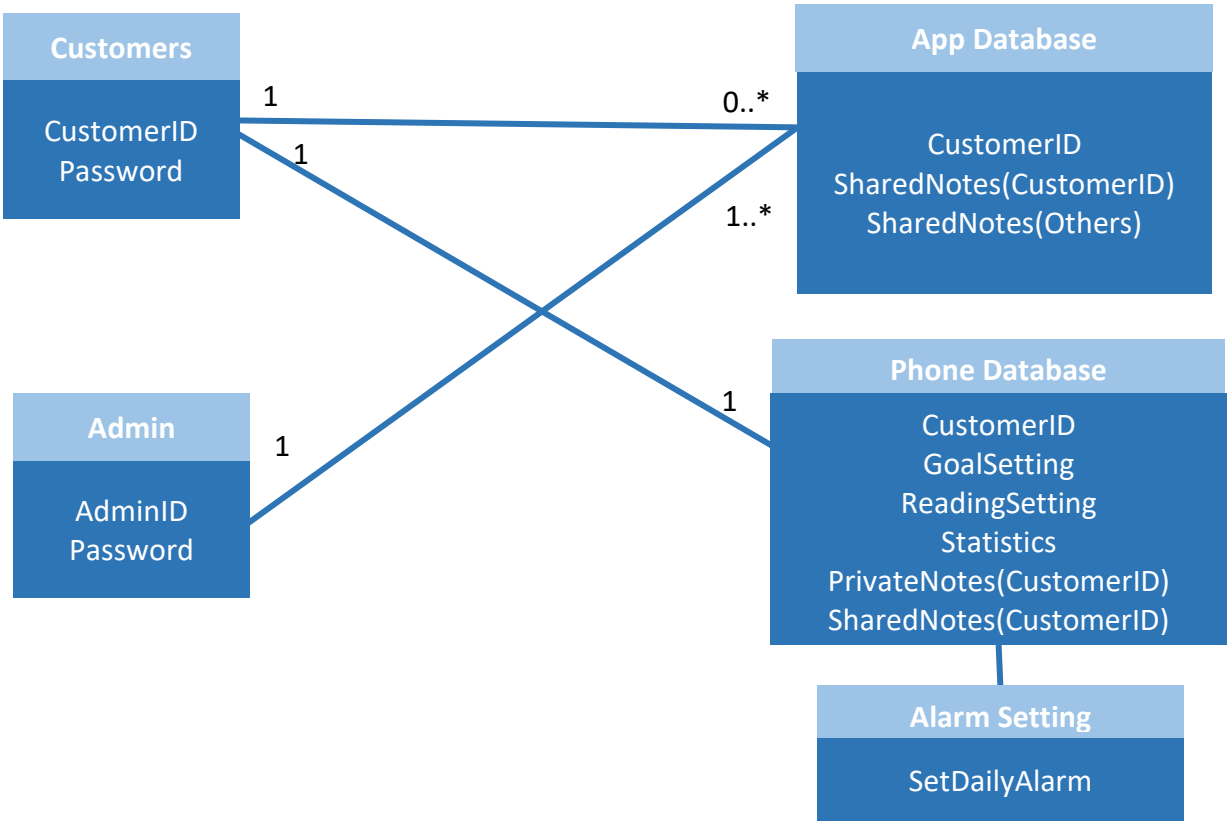
4.1. Product Perspective



4.2. Product Function



4.3. User Classes



4.4. User Interface