Prioritize

Joel Wilhelm
Cody Jones
Shahrukh Rehman
Wajahat Iqbal

Department of Computer Science
University of North Carolina at Greensboro

April 27, 2017

Contents

Abstract

Prioritize is an Android application that incorporates a priority value with a scheduling system to simplify the user's life.

1 Introduction

Prioritize is an Android application that aims to organize the user's life by letting them type in a reminder for something that needs to be done by a certain time, and reminding them in intervals based on priority, the time between creation, and the "due-date." This lets the user periodically be reminded of the event or task over time rather than the user deciding for themselves. Once reminded, the user can choose to have the reminder be stopped, postponed, or simply be reminded at a later time, calculated by the application. These reminders will be synced across the user's Android devices allowing them to be reminded at their convenience.

2 Requirements Specifications

2.1 Functional Requirements

In order for Prioritize to function properly, several requirements must be met. The application must be able to create, store, edit, and delete Reminders. Each Reminder should notify the user at the appropriate time, chosen by the Priority Algorithm.

When creating a Reminder, the user must be able select a date, and have the option of picking a time with that date. The user must also be allowed to pick multiple dates. The user must be able to view the current list of Reminders that have been created, and have the option of deleting them, and editing them. The user should have access to two different types of alerts - an alert, and the more intrusive alarm, which requires the user's input to stop. A Reminder must have the option of being repeated. When a reminder is being alerted, the notification itself must have a "Snooze" option, which tells the app to remind the user again in a little while,a "Remind me Later" option, which sets another reminder based on percentage of time remaining via the priority algorithm, and a "Stop Reminding Me" option; This will cancel the Reminder entirely and leave it on the Reminder List until the date/time has passed.

2.2 User Interface Requirements

The UI must implement a minimalist design. The home page must display a list of the current reminders along with a button at the bottom right for adding a Reminder. Swiping to the right should reveal a calendar that has a mark on the dates that have each reminder due date, along with a mark for when that Reminder is being alerted. The two of these marks can be displayed in a different style for differentiation.

2.3 Usability Requirements

The user must be able to create a Reminder in a substantially small amount of time. These Reminders must be synced automatically with the rest of the users android devices that have the application installed on them under the same Google Account.

2.4 Performance Requirements

The application must have a minimal response time in order for creating a Reminder swiftly, and without being interrupted. That means punctual app transitions, opting for quick GUI animations rather than slower ones. Syncing across android devices must be as quick as the Google Drive API allows.

2.5 Security Requirements

The user's Reminders must not be available for viewing by anyone other than the user and then only through the application itself. The Google Drive API used to syncing android devices uses Google's account system for security in the cloud.

3 System

3.1 Overview of Complete System

When user creates a Reminder a description is set by the user, a date (or dates) are set by the user, a time is optionally set by the user, a priority value is set via a slider, and the user has options for ignoring the priority

system, setting it to be repeatable, and changing from the default alert style to an alarm style notification. Once the reminder is submitted, the priority algorithm takes the priority value, the current date and time, and the due date and time of the Reminder, and calculates a date and time for the user to be reminded based on the percentage of time between the two dates. The information the user entered and the new priority date is saved in a local device database, and is synced with the user's Google Cloud via the Google Drive API. This allows for the user's other android devices that have the application installed under the same account to sync with the Cloud and set the reminder on that system as well. The alerts are set via the Android system which lets the application be closed until the time of activation of the event.

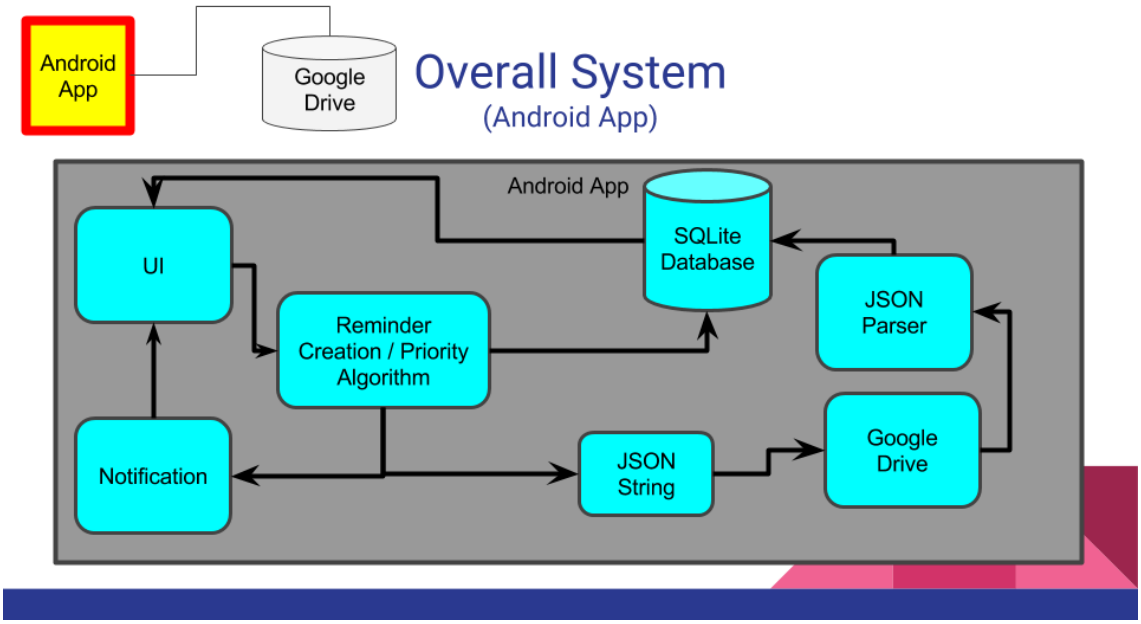

Figure 1: A brief system overview

As shown in Fig. 1, a brief overview of the system is shown. The Reminder Database, implemented in SQLite, is stored locally on the device. Information from the Database is shown in the GUI. The GUI is used to create new Reminders, which are then stored in the Google Drive folder, as well as the SQLite folder. The JSON class will take values needed to create and load a Reminder, and will convert them into a JSON String format, to store

in the Google Drive. It will then parse through that string when loaded, and allow the application to use the information.

4 Subsystems

4.1 Home Activity - Shahrukh Rehman

The Home Activity is a fragment system for activity to display the current list of reminders on one slide. Swiping right reveals a calendar with reminders marked on dates. Clicking the "Add" button will prompt the user to create a new Reminder.

4.2 Notifications - Wajahat Iqbal

The notifications portion of Prioritize reminds the user of the task depending on the priority set by the user. This happens through sending a notification via an alert or an alarm. Notifications will offer Snooze, Remind Me Later, and Stop Reminding Me options.

4.3 SQLite Database - Cody Jones

The SQLite Database manages the user's reminders. The database contains a single table, called Reminder. This database is synced with the cloud so that the user's other Android devices can share all of the same information. Because we are dealing with Reminders as a single entry we opted to avoid using standard SQL databse normalization. All information regarding the Reminder is stored in this local database.

4.4 Priority Algorithm - Cody Jones

The meat behind Prioritize, the Priority Algorithm will select the date and time for the user to be alerted, regarding to the task they created a reminder for. It takes the time between the date of creation and the "due-date" to set a new date based on the percentage of time between the two. The use picks a priority integer ranging from 1 to 5 (low to high priority). Each level corresponds to a specific meaning; 1 - Unimportant, 2 - Semi Important, 3 - Important, 4 - Very Important, 5 - Crucial. The higher the priority level the

sooner the initial alert of the reminder will be. Upon saving the Reminder, the time (or number of days and hours) between the time of creation and the deadline is computed. Based on the priority level and the amount of time between the two dates, we construct an amount of time that signifies when the user should be alerted. That amount of time is incremented into a valid date and time and the notification is then set for that time.

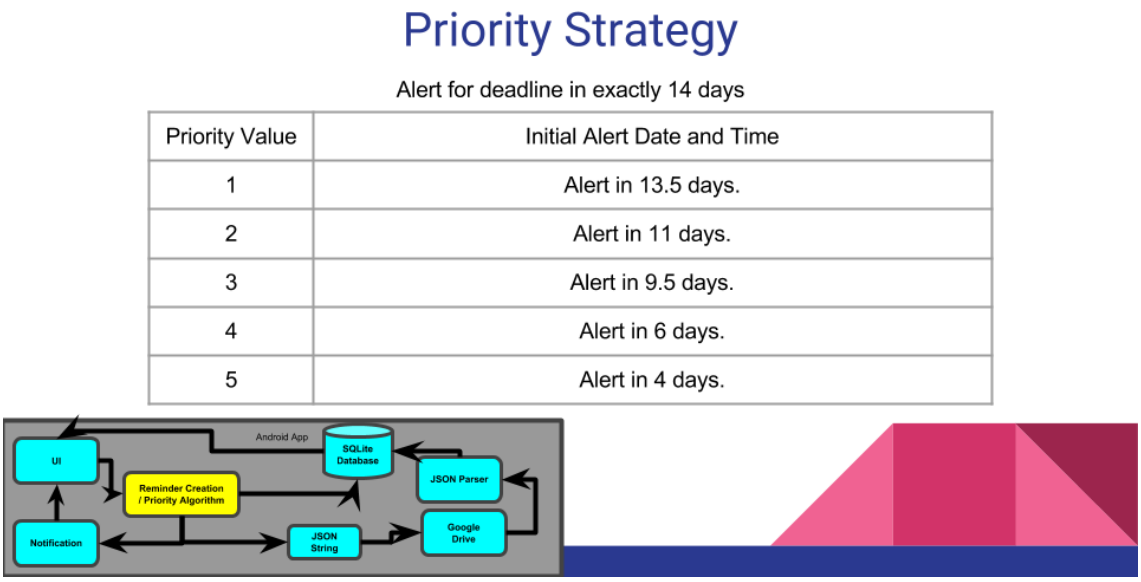

Figure 2: Priority Algorithm calculation example.

4.5 JSON Object and Parser - Joel Wilhelm

In order to take useful data and store it in the Google Drive folder, a method was needed in order to send and receive the necessary information. The JSON data structure was chosen, as it was a good way to both send and receive information in a string based format. In order for this to work properly, a JSON object was created. This object had fields that contained each important piece of information for a particular notification. It also had a constructor that would take all the information from that object, and turn it into a JSON string. This string is what will be stored inside of the database.

In order to use the data stored in the database, a parser was needed. This parser will take the string from the database, and parse through it, creating a new JSON object, whose fields are the values stored by the string constructor. This object is then passed back to the application, who can now use the information in the way it was intended.

This JSON class allows the ability to save notifications, and thusly, is vital to the success of the application.

4.6 Google Drive Syncing - Cody Jones

After considering between using Firebase by Google, we decided to use the Google Drive Android API to store the user's data across their android devices. We felt that security and privacy were crucial for Reminders, using the user's personal drive space provided the most security. Also, being an android application, we know that all users have a Google Account already. The user should only have to log in to their account to view entries that they have previously made. Inside the user's Google Drive space, Prioritize will take advantage of the application data folder, which is inaccessible to anybody except Prioritize. Inside the drive space there is a Reminder folder. This folder will hold all of the JSON representations of the reminders saved in the SQLite Database. When the GDAA hears a change in this folder via the "Change Subscription Listener" which is implemented on it, the device will wake up, parse through the files and add any new entries onto the device and set the notifications accordingly. For handling deletions, the app will again hear a change in the Reminder folder, and will check it's corresponding device folder for any Reminders that need to be deleted. When a device deletes a reminder, it sends a tell to each of the other device folders in the drive space to delete that file as well. This can be seen in Figure 3, as well as later in the report as well.

4.7 GUI - Shahrukh Rehman

Similar to the Home Activity, the GUI encompasses everything the user sees and interacts with. It provides the user with a menu, buttons, options to create a new Reminder, edit old Reminders, a calendar to view when Reminders are due, and so much more. In short, it is the face of Prioritize. Shown in Figure 4 is a visual representation of the GUI dataflow. This is Shahrukh's responsibility.

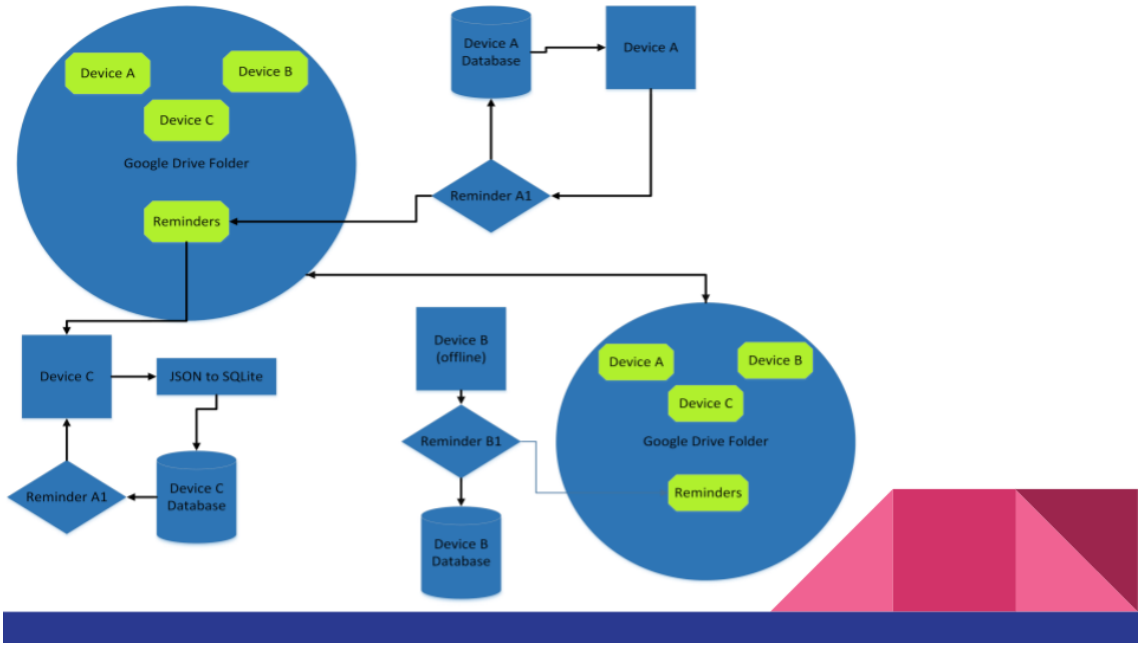

Figure 3: An overview of Synching across multiple devices.

4.8 Adding a Reminder - Shahrukh Rehman

The Add Reminder activity is a system for the user to create a Reminder to be added to the Reminder database. Has field for Reminder description, date picker, time picker, repeatable check, alert/alarm switch, and the option for ignoring priority system. Shown in Figure 5 is a visual representation of the process of creating a new Reminder. When a Reminder is saved it is sent to the local SQLite databse as well as the Google Drive space.

4.9 Documentation - Joel Wilhelm

All documentation for Prioritize was done, or overseen by Joel. He was responsible for creating and overseeing PowerPoint presentations, formal reports and writeups, posters, and user manuals.

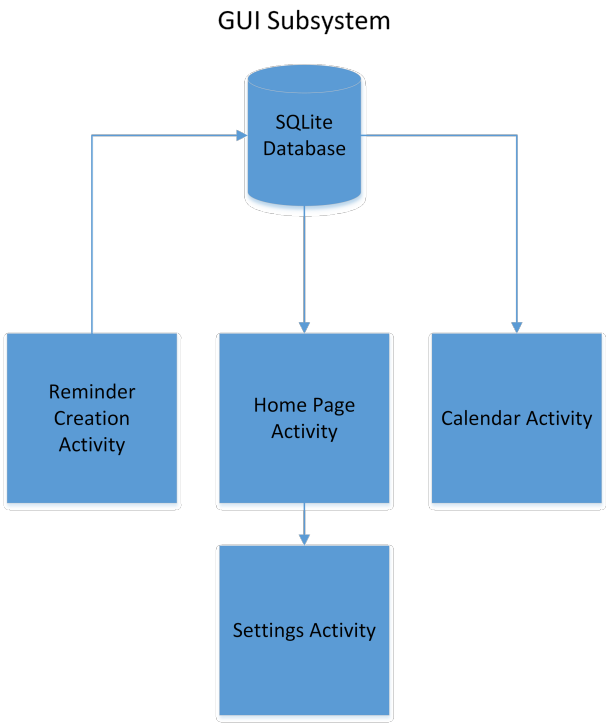

Figure 4: A brief system overview

5 Risk Factors

There are many risk factors involved in any project undertaking. So, it is therefore natural that there are risk factors within Prioritize as well. Listed here are potential risks factors that may have an impact on the success and implementation of Prioritize.

Time is always a factor in any project, big or small. If not enough time is available, the project may become rushed, and will suffer thusly. There is limited time in which to plan, prepare, and implement Prioritize. Thusly, time is a risk factor.

Experience. None of the team has used Android Studio before, or programmed an application designed for a mobile device. This lack of experience is a risk factor, because time investment is mandatory to learn about the environment, and to understand what is necessary to begin implementation.

Group work. The fact is that group work can be difficult. Lack of communication is a major factor in failed projects, so being able to communicate

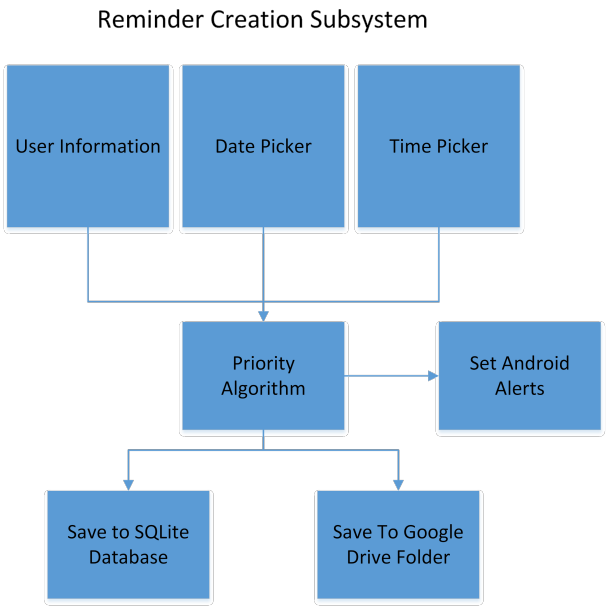

Figure 5: A visual representation of Reminder Creation

is vital. The fact that group members have other classes, jobs, and outside work other than Prioritize means that getting together to work on this project is a Risk Factor.

The Waterfall Method. The Waterfall approach for software engineering is not a very successful method for designing and implementing a piece of software in the real world. While it can be useful in some ways, it is possible to get caught up in the overhead, and not spend enough time working on the project. It is a trade-off between planning, preparation, and documentation, and getting code written. This is a Risk Factor.

Google Drive API. We have chosen to utilize the Google Drive API for our project. This is helpful, as it assists with syncing between devices, security when synching, data storage, and more. However, we must invest time into learning how to use Google Drive API.

6 Data Dictionary

Following is a Dictionary of all pieces and bits of Data within the Prioritize project.

Alarm — More intrusive than a simple alert. Has a noise that will not stop until acknowledged by user.

Alert — A simple message sent to the user's device. Unobtrusive and simple.

Due Date — Day on which the event will be due. Will be a text-based format. DD/MM/YYYY.

Event Description — A String that describes the Event in question.

Ignore Priority — A value that allows the user to enter their own time for a notification, without the use of the Priority Algorithm.

Notification — How the user gets reminded of an upcoming event. Is also an integer value, to signify Alarm or Alert type.

Priority — A value set by the user that signifies how important an event is to them. Will be used inside the Priority Algorithm to determine when the notification will occur.

Repeat — An integer value that will determine how often the user wants to repeat their event.

Reminder — An object that encompasses Due date, Creation Date, Due Time, Creation Time. Event Description, Notification, and Notification Type. Will be used as the object that is used to save and store information. The system depends on Reminder objects as an integral part of Prioritize.

7 Algorithm Analysis

7.1 Priority Algorithm

The Priority Algorithm has three steps to it. It operates on three while loops, getting the number of days, months, and years until the reminder is due. However, these numbers are not large, and the algorithm ends up at O(n). The second step is to calculate when the app will alert the user of their event. This is a simple calculation, and will run in O(1). The third step is to find where the reminder falls on the calendar. This ends up at O(n). Therefore, the Priority Algorithm itself will end up at O(n).

7.2 Database Lookup

The Database lookup has a complexity of O(log(n)).

7.3 JSON Parser

To convert values to and from the JSON format, a parser is generally used. Parsers tend to be O(n), where n is the number of lines to be parsed. Each JSON entry will vary only in the number of dates the Reminder is assigned to have by the user. This means the parser will end up at O(n+d), or O(n).

8 Data Model

We will be using an SQLite database to store all Reminders with all of the Reminders information. We will be using JSON representations of each reminder as a method of transferring that data across the user's Android devices through the Google Drive Android API. Each Reminder stored in the SQLite Database is stored as a single table with the information described below. SQLite doesnt support date and time formats, so we will be storing them as integers in the form ddmmyy and hhmm for date and time respectively.

Reminder Creation Key - String. (Assigned by Device that it was Created on) **Event Description** - String .

Priority — Integer.

Due Date — Integer. 7 of these fields will be present, to represent the possible choices for multiple reminder dates.

Due Time — Integer. 7 of these fields will be present, to represent the possible choices for multiple reminder dates.

Number of Dates — Integer.

Alert Date — Integer.

Alert Time — Integer.

Repeat Every x Days — Integer.

Alarm — Boolean.

Ignore Priority Algorithm — Boolean. (Allows user to set alert for specific time)

8.1 Dataflow

8.2 Reminder Creation

When reminders are created the data must be saved in the SQLite Database on the device it was created on. It must also be saved in JSON format on

the Google Drive Application Data Folder with the Google Drive Android API (GDAA). The GDAA allows access to a folder that is accessible by the app at all times, offline and online. It provides syncing mechanisms for the app and syncs with the Google Drive Application Data Folder on the online cloud whenever there is a connection. It connects to the users personal Drive and uses their data. The diagram examples a situation where Device A and Device C are connected to the internet at the same time. A makes a reminder and C syncs the data to itself. B is offline but the user created a reminder on that device as well. Eventually, once Device B connects to the internet, the to Application Data Folders will sync, and devices A and C will receive the data made by B, and Device B will sync the data created by A.

This can be seen in Figure 6.

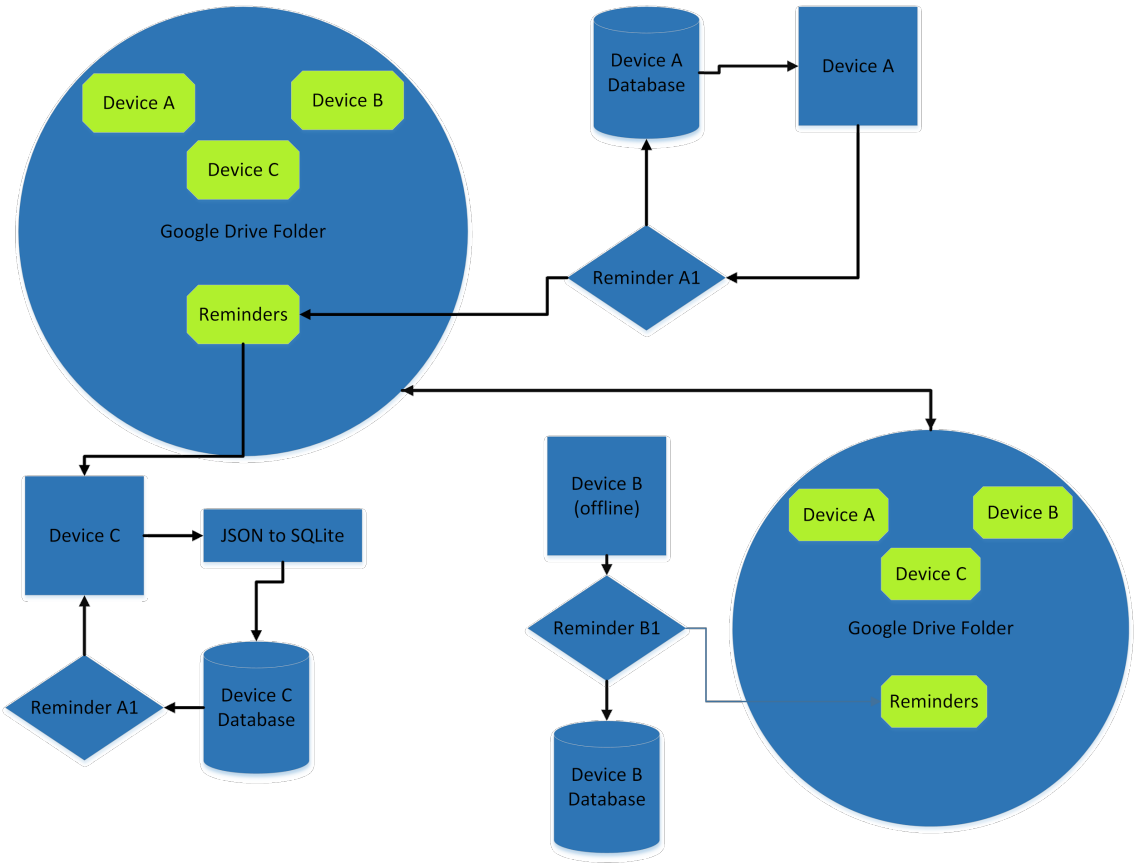

Figure 6: A basic visual representation of Reminder Creation.

8.3 Reminder Deletion

When Reminders are deleted, the device that proceeded with the deletion will delete the reminder from its SQLite Database and from the Application Data Folder, and send tells to the other devices folder telling them to delete the reminders from the local databases. They are told individually to the deletion tells from adding up, any device that is keeping up with deletions more frequently shouldnt be hindered into performing more operations because another device is connected less frequently. When a device receives and performs a delete, the tell that it received from its Application Data Folder is deleted after the operations have been completed. In the figure, Device B is offline and doing nothing, Device A is deleting Reminder A1 and Device C is syncing with the deletion initiated by Device A.

This can be seen in Figure 7.

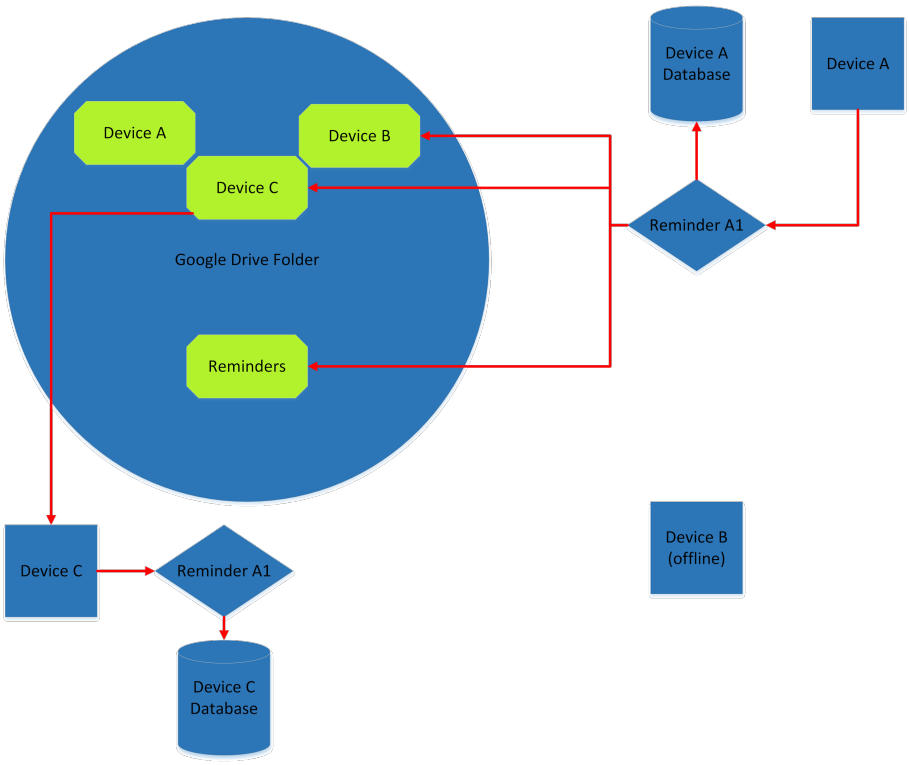

Figure 7: A basic visual representation of Reminder Creation.

9 Design and Controls

Following, in Figure 8 and Figure 9, are screenshots of what the Prioritize Application's User Interface look like. Figure 8 shows the list of reminders currently active, and Figure 9 shows the Date Picker activity.

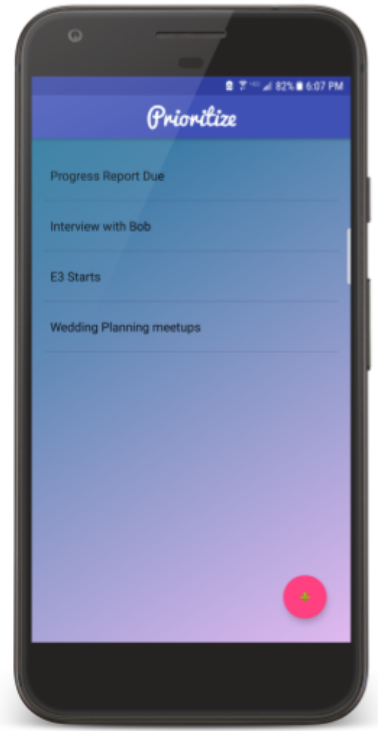

Figure 8: A listing of all the current Notifications the user has in store.

10 Testing

Joel Wilhelm wrote a unit test code for his JSON object creation and parser to make sure it worked properly before sending it to be integrated with the rest of the project code. In order to test this code, a new JSON object was created, and field values were set. The code then took each field, and converted it to a JSON string. When the JSON code needs to unwrap and parse through a string to get the fields, it takes a string. So, the string that was created was passed to a new JSON object, and the fields were set

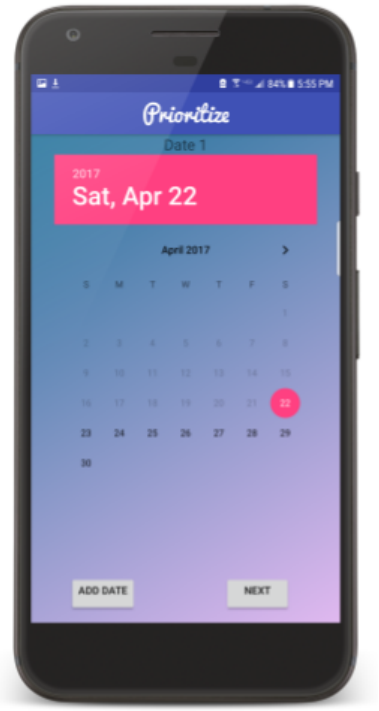

Figure 9: Date Picker. This screen allows the user to pick when their event is due.

to be what the values in the string were. If the two objects contained the same values for each of their fields, both the string creation and the string unpacking were successful. This code was run a few times, and each time, the code functioned properly, the Unit Test was completed successfully. Each time the JSON class needed to be tweaked or changed to fit other bits of code, a new Unit Test was run, to make sure the code still functioned properly, and as planned.

To test Alarms and Alerts, the notifications were set up, and triggered. If all buttons function as desired (clear reminder, alert 1 hour later, or compute priority algorithm for the current time and set a new alert) then the code worked as intended. Alarm must do the same and have a ringtone that does not end until user interaction.

Priority Algorithm testing. Testing this will have an approach that is less mathematical and more based off of opinion. We will use the app frequently and decide whether we've been alerted an appropriate time and at

an appropriate interval.

SQlite Database Testing. The Reminder table is capable of Insertions, Deletions, and Updates for any entry and it is not possible to insert an invalid Reminder.

Device Syncing Testing. Once stable, when creating a reminder on one device under an account, A, another device also using account A should import the new file, if present then everything worked correctly. If account A deletes a reminder on either device, the other device using the account should also delete that entry. If the SQLite database on both devices do not have the reminder anymore, then the test was successful.

11 Fulfillment of Requirements

As of April 11, 2017

There were several functional requirements of the Prioritize Application. First of all, the user needs to be able to create Reminders. Within this goal, several subgoals need to be met. The user needs to be able to input Reminder descriptions, be able to pick time and dates, in some cases multiple dates, designate a priority value, choose between a simple alert and more intrusive alarms, and have the option to repeat the Reminder.

The ability and function to enter Reminder descriptions is complete. The user can pick time and dates, can designate priority levels and values, and has the option to pick what kind of alarm or alert desired. All these functions are complete. Currently in progress is the ability to repeat a Reminder, or to pick multiple dates. To repeat a Reminder, the application will eventually create a new instance of that reminder, just with changed date and times.

The user also must be able to edit or delete reminders after they have been created. This functionality is almost complete, and is still in progress. Each notification should have two settings, a snooze feature, and a feature to end the notification. These features are complete and functional.

The app should be able to sync between devices, this functionality is almost complete, but is going through a debugging phase. Once that has been completed, the functional requirements will have been met.

The User Interface has some requirements as well. They need to be able to see a list of current reminders. This is now possible, and works properly. The UI should be clean and elegant. It is not currently as clean or elegant as it could be, but this is of lesser importance than actual functionality, and

is of lesser priority. One feature that has not been completely added yet is the ability to see a calendar with dates marked for alerts and due dates.

One large requirement of the application is that it is quick and easy to use, and doesnt take too much time to set up. This requirement is satisfied.

12 Training

Please see the User Manual for more information on training, and use of Prioritize.

References

Citations

[1]"Drive — Google developers," in Google Drive APIs, Google Developers. [Online]. Available: https://developers.google.com/drive/. Accessed: Feb. 2017.

[2]"Introducing JSON," in JSON.org. [Online]. Available: http://www.json.org/. Accessed: Feb. 20, 2017.

[3]"Android studio the official IDE for Android," in Android Studio. [Online]. Available: https://developer.android.com/studio/index.html. Accessed: Feb. 25, 2017.