# SchoolPlanner

Ryan, Peter, Sam, and Xuan
Department of Computing Science
Simon Fraser University
British Columbia, Canada
Computing ID: rymei, csp7, kaixuny, xuann

**Abstract--- This paper provides the design details of our hypothetical app created for the group project in CMPT 276 at Simon Fraser University. No significant app featuring a similar set of functionalities was found in the Google Play Store; however, we anticipate that there must exist at least one similar product out of this scope. Our app aims at providing convenience to academic advisors and students at an institution via remote assistance. The main functionalities of the app resemble the web advising applications used at institutions to provide a similar experience to the actual advising sessions. Hypothetically, an advisor using our app should be able to help students out of office hours, without a need to conduct a meeting. The ultimate goal is to allow advisors to assist comparably a larger population of students during busy times of the year.**

## I. Introduction and Motivation

In this modern day and age, the focus and demand on education are growing faster and faster. There are more students and more competition. This leads to an abundance of students with limited access to information. The use of the internet has somewhat helped in this regard. Most universities/colleges have information about different degrees and their requirement, and they have a student center with even more information. This makes accessing the information easy, but many things are not the easiest to find, and there are many things that are not accurate due to updated requirements not being updated on the website. Being unsure of your next step in terms of education is what every student faces at some point in their lives. This is why the advisors are so important. They help students with more than planning out their education. Often, students go to advisors for advice from somebody experienced and knowledgeable since the students themselves don't really know what their options are, or even what they're required to take. Because of this and the fact that there are just so many students compared to the number of advisors, it is often the fact that the students can not get ahold of an advisor very easily or conveniently. This is the case especially during course selection. Often times a student is either too swamped with work from school/job to worry about course selection until the week that they have to pick the courses. By this time, it is very hard to get access to an advisor because they just have so many students coming in at once, and it's not in the interests of the universities to hire more staff just for a short period every semester, so as students, we need to find a solution that will suit all the parties involved. Our solution is to utilize the fact that most students has a smartphone nowadays, and create an app that connects students to advisors. Due to the constraint that we don't have the ability to access the databases of schools to get student information, we have allowed students to "opt-in" using a signup process. Once signed up, students will have access to the features of our app, which is designed to optimize the down time of advisors. By this we mean that we are taking advantage of times such as transit time, time between appointments, and just other moments of free time in general. Our app is aimed at quick and short interactions between the advisor and the student, so that the students can get a relatively quick response when they need it, and advisors can utilize their time to the maximum potential.

**(add stuff regarding related apps)**

## II. Background

### *A.SQLite Database*

SQLite is a free database engine developed by an international team of full-time developers. SQLite implements a self-contained, serverless, zero-configuration, transactional, SQL database engine, and is the most widely deployed database in the world [1]. Android has a built-in library under android.database.sqlite namespace. One of the best advantages of SQLite is that it requires limited memory at runtime (approx. 250 KB) which makes embedding more efficient and provides a simple interface. Since our app does not have access to the real database of SFU, we adopted SQLite to save data in the device's local storage. By default, the database is saved in the directory DATA/data/SchoolPlanner/databases/FILENAME, when it is created. Using the MySQLiteHelper class, we can access the database and manipulate data with a various methods and constructors defined in this class such as add a row, delete a row, update a row, and delete all rows. The database supports the data types String, long, and double in Java. Any type of data can be assigned in a column regardless of the user-defined type. In order to get a specific value for a row, we use the Cursor object. A cursor is the result of a query and points to the specific row of the query in question. Cursor provides various methods such as getCount(), getColumnCount(), getString(columnIndex) and getLong(columnIndex) to retrieve information about the database and the data from columns in each row [2].

### *B. Gitlab*

Gitlab is an open source git repository launched in 2011 that allows multiple developers to work on the same project, and it provides a way to merge differences in code. Generally, git repositories allow users to push, commit, pull, merge their code, and Gitlab is no different. When we first started the project, none of us had prior experience with any kind of git repositories, and although the commands for git were simple, figuring out how the repository worked was a bit different. Once the basics of committing and pushing were figured out, branches and merges were the next step. After various attempts to find solutions on Google, we finally figured out a solution. Aside from our difficulties, Git turned out to be extremely helpful in keeping everyone up to date on the project while allowing everyone to work in tandem.

## III. Activities and Views in Android

For the UI, we made this app simple as possible to minimize the effort to users to accomplish their work. In Android, a single "screen" or a "page" is an activity. Each activity has an auto generated onCreate function where things are usually initialized. Aside from this, there are five other useful functions that activities can take advantage of. These are onStart, onStop, onPause, onResume, and onDestroy. The activity goes through a lifecycle, and when another activity is started, it gets placed on top of the stack and the previous activity is the one just below it until the new activity is done and exited. The MainActivity of each app is automatically started, and to start another activity, startActivity(Intent) is the usual method used.

## IV. Functionalities

### *A. Sign up*

Before the user is granted access to the main activity where the app's functionalities are provided the user needs to create an account. There are two types of users in our app: student, and advisor. Each user type has access to a set of different functionalities, and thus, user differentiation is required. Therefore, we provided a separate *sign up* activity for each user type. As a security measure, registering as an advisor is limited to only those that enter the correct access code during sign up. In reality, our app is currently designed such that the user needs to know the access code prior to sign up. However, there are various ways to provide a randomized access code to a user such as SMS, email verification, and etc.

Signing up as an advisor requires the following information: first and last name, access code, Computing ID, and password. The data are saved as the String data type and are recorded in the ADBAdapter database.

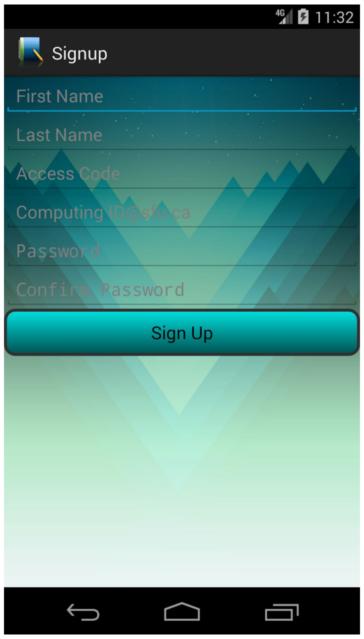


*Figure 1: Advisor Sign Up Activity*

Signing up as a student requires the following information: first and last name, student number, Computing ID, and password. The student number is saved as the long data type and the rest are saved as the String data type in the SDBAdapter database.

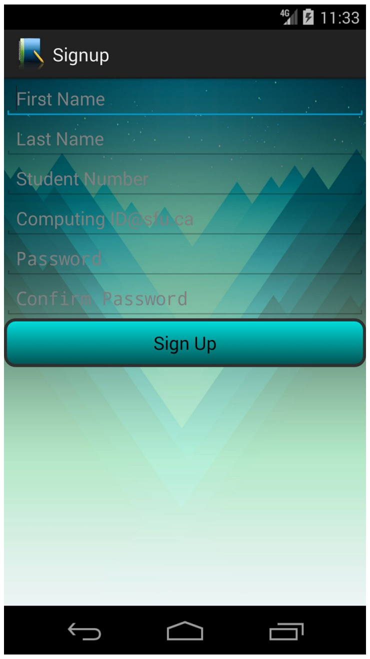


*Figure 2: Student Sign Up Activity*

### *B. Sign in*

The user will sign in using his/her username and password. A successful login will direct the user to the main activity where the system functionalities are organized in a list of buttons. As mentioned previously, the main activity will display a specific set of functionalities depending on the user type. Cursor is used to validate user information by matching user input to data in the database table.

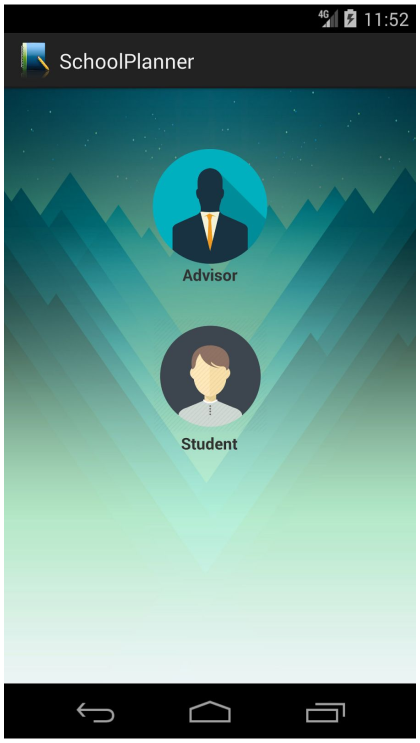


*Figure 3: Startup Screen.*

## C. Menu Screen

### i. Advisor

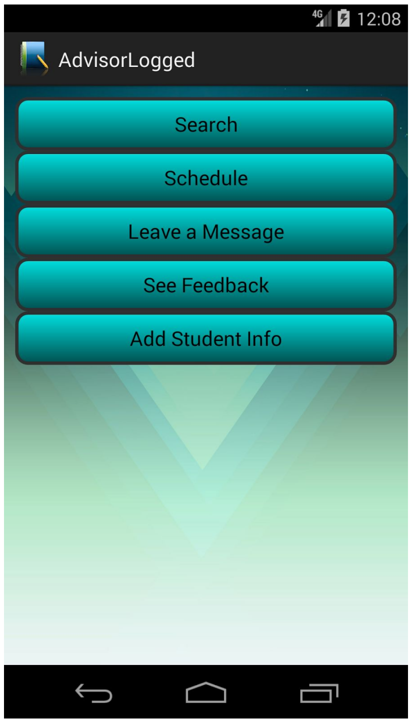


*Figure 4: Advisor Main Activity.*

### Ii. Student

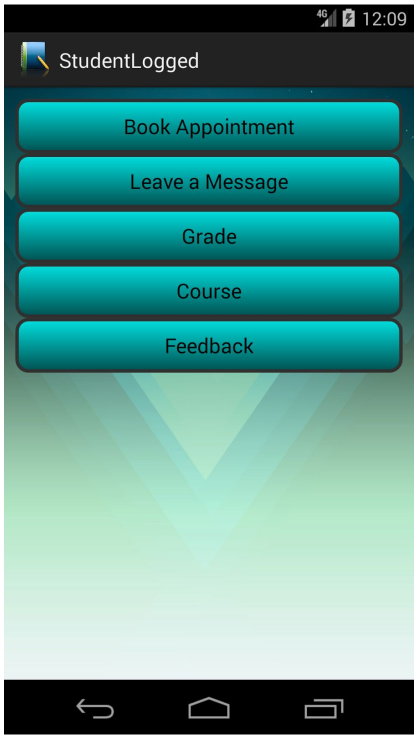


*Figure 5: Student Main Activity.*

## D. Schedule

This activity is only accessible from the advisor's main activity. Using ListView, data are pulled from the Appointments database onto display. Each row consists of the student's name, day, and time. Clicking and holding on a row of data opens up a context menu giving the user two options, Edit, and Delete.

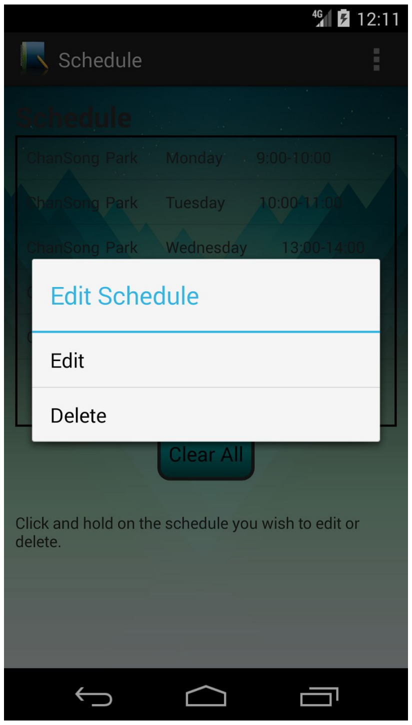

12:11
Schedule
Schedule
ChanSong Park Monday 9:00-10:00
ChanSong Park Tuesday 10:00-11:00
ChanSong Park Wednesday 13:00-14:00
Edit Schedule
Edit
Delete
Clear All
Click and hold on the schedule you wish to edit or delete.

*Figure 6: Edit Schedule Context Menu.*

Clicking on "Edit" starts the Edit_Schedule Activity as a Popup Window. On the Edit_Schedule Activity, user is prompted to enter first name, last name, and choose day and time. Once the user clicks on the submit button after entering all required fields, data are saved using the Intent object. The saved data are then retrieved in the Schedule Activity from the Intent object and the Appointment database is updated using the saved data. When the database is updated, the ListView refreshes with modified values of the data.

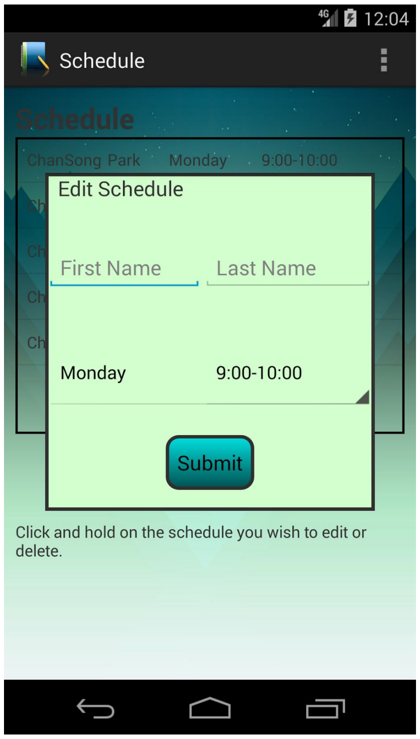

12:04
Schedule
Schedule
ChanSong Park Monday 9:00-10:00
Edit Schedule
First Name
Last Name
Monday
9:00-10:00
Submit
Click and hold on the schedule you wish to edit or delete.

*Figure 7: Edit Schedule Popup Window.*

Clicking on "Delete" in the Context menu removes an entry from the Appointment database and ListView refreshes the new list on display.

Lastly, clicking on the "Clear All" button below the list removes the entire data in the database.

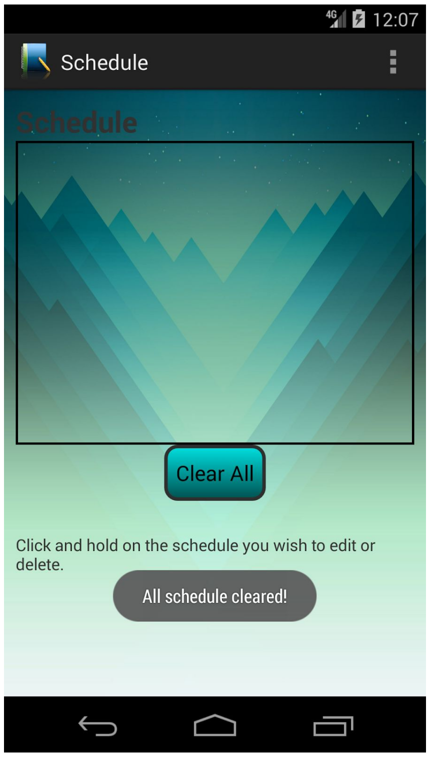


*Figure 8: Clear All Schedule.*

***E. Chat:***

When we thought about the solution of the difficulty in reaching an advisor The first thing that came into our mind was to let advisors and students to communicate in this app. This functionality will let advisors be able to help more students when they have spare time (like before next student come to see them), so we discussed about how to let them talk in this app. And during the discussion, we thought it is better to let students and student or basically all the people using this app to talk in one place so they can ask question, discuss about class, or just chat whatever they want. Therefore, we developed this chat activity that could be a great solution of less advisors.

This chat activity is not a real time chat app, It is more like a thread like reddit. The reason that we developed a thread and not a real time chat is because there are so many students compared with advisors, so if we just develop an activity that let student and advisor talk individually, we are putting too much burden to advisors since students might ask easy question that could be solved by themselves or by other students. However, if we make this activity as a thread students might help each other. I believe that most of the students problem could be solved by themselves or could be solved by other students. Even if students cannot solve their problem without advisor's help, there might be few or many students having the same problem and advisors can help all of them at one time by using the thread. For instance, if there are hundred students wondering about what course they should take for upper level and need advisor's help, advisor can post his or her suggestion to the thread so those hundred people do not need to see advisors anymore, so students and advisors can both save lots of time.

For functionality of this activity, when user enters a message and click the yellow enter button their message will be displayed on the top of the blank space of the app with their name and current time and current date that they send that message. And when all the blanks are full the app will automatically delete the oldest sentence and shift all the existing sentences one by one and add user's latest input to the bottom of the blank. Even if the user close the app or move to another activity, all the messages in the activity will be saved so user will be able to continue their chat anytime.

***F. Booking Appointment:***

By using this functionality, students can book appointment with advisor by selecting the time period and day they want to meet the advisor. The time will be from 9am to 4pm, and the day will be from Monday to Friday. If the selected time period for that day was booked by another student, then there will be an alert comes up, and tell the user to select other time period. Once the time was booked successfully, the student's name and appointment time will be stored in the SQLite database after clicking the book button, and a dialog will pop up telling the user that the appointment was booked successfully. After the user clicks the OK button, it will bring the user back to the menu page.

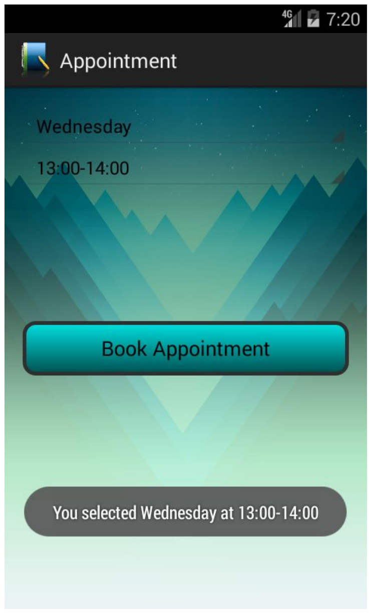


*Figure 9: Book Appointment.*

***G. Searching & display student's information:***

Advisors can see student's information only by searching student's id or computing id. If the id is in the database, then advisor can see student's grade for the previously taken courses, taking courses, and major. Student information will be pull from the student information database. If the student's information is not yet in the database, then nothing will be shown. If the advisor leaves two edit text blank in the searching activity, then an alert will pop out.

***H. Add student's information:***

Since the application cannot connect to the SFU database, so advisor need to add student's information to database manually. Advisor can add student's information to database in this activity by entering student's name, id, major, taken courses and grades and taking courses. Once the information stored successfully, a dialog will pop up.

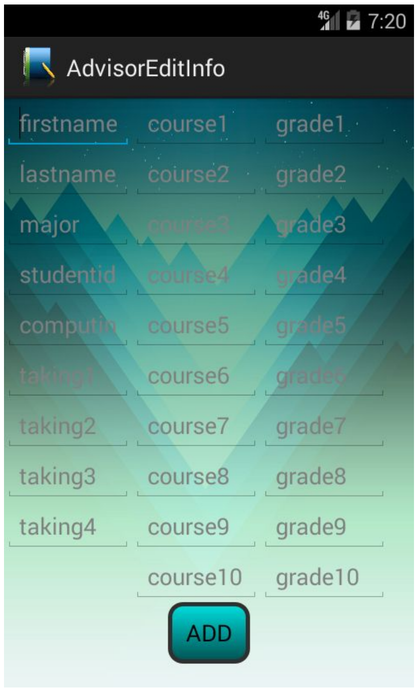


*Figure 10: Add Student Info*

***I. Feedback***

As with most things, our app is not perfect. Therefore, we have added a way for the students to give feedback based on their experiences with the app, or give feedback about how their class is going. Our feedback activity is only accessible by the students, and it provides anonymous feedback about a few things. First, the student may type in whatever they want to say in a text box, and it will be sent to the advisor. Secondly, there are 2 rating bars for the student if they do not have time/ don't want to type out their feedback. These are also sent to the advisor.

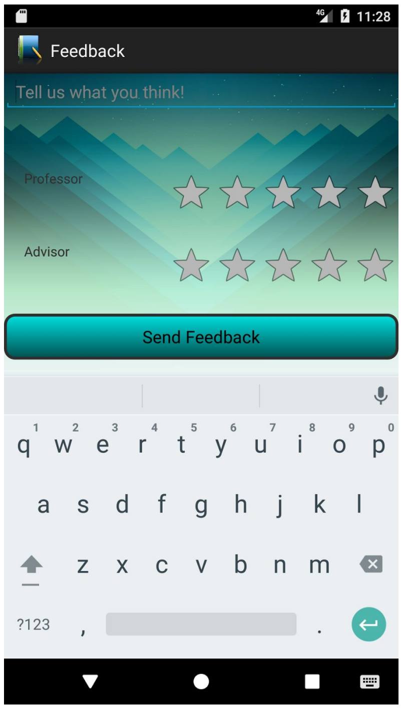
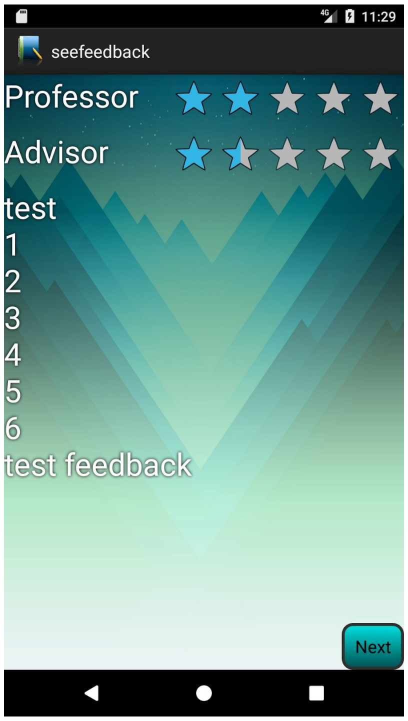


Figure 11: Example of how the feedback activity looks like.



test
1
2
3
4
5
6
test feedback

Figure 12: Example of how advisors can see feedback.

For the advisor, when they view the feedback, they will get a single page of feedback including the comments and the rating bars from the students. If they want to see the next feedback, they can press the next button, and it will bring them to the next set of feedbacks.

When implementing this section, I had to figure out a way to transfer data from one activity to another. Normally, when a variable is created and saved in a single activity, it disappears upon entry to the next activity. So for this purpose, I researched two different methods. The first method is using Android's SharedPreferences. I thought about using this, but these "preferences" only saves locally, and I wanted a way to implement this app so data can be saved onto a server if we can get access to one. So the second method that I ended up using was the Singleton class. Using this, I was able to save the data under the Singleton class that acted as a "DataHolder", which is what I named the file, and it initially was only used to help me with the feedback section. But once the group got together and we helped each other with sections we struggled on, it turned out that this Singleton class was immensely useful for the transfer of information.

## IV. Related Apps

### *A. Know School*

https://play.google.com/store/apps/details?id=com.knowidea.knowschool&hl=en

Know school is an app that lets students sign in and see their school information. This includes schedule, email, and the list of students in their classes. This app is the most similar app to our's in the Google Play store, but there are some issues with an app like this. First, it's a third party app that is not supported by an educational institution such as SFU. This means when a student logs into their account using the app, all their information is at risk of being collected by the owners of the app. This includes some very private information such as the student's grades, personal information, and their computing id and password. As of a recent update, this app also requires the student's facebook to use, which means even more private information is collected from the unsuspecting student.

### *B. Google Calendar (and other calendar apps)*

https://play.google.com/store/apps/details?id=com.google.android.calendar&hl=en

Calendar apps such as Google Calendar have similar scheduling tools as our app, except their calendars are more designed for everyday and general use while our scheduling feature is purely for advisors and students to book appointments. The other calendar apps are generally more developed due to having more time and a more experienced team behind it, and so they usually have a few features such as syncing to other devices that we do not have.

### *C. Messenger*

https://play.google.com/store/apps/details?id=com.facebook.orca&hl=en

Facebook messenger and other similar messaging or SMS apps are like the chat feature that we have. The difference is that the chat system for messaging apps are usually real time, with fun features such as emojis and stickers to enhance the chatting experience. For our app, our chat system is more like a thread, or a message board. Advisors and students can leave messages, but the chat does not update in real-time. Additionally, our app puts all users in one chat room. This means that the chat is not an one-to-one chat. If another student messages the advisor, I would be able to see the message, and vice versa.

### *D. Homework Manager*

https://play.google.com/store/apps/details?id=studios.gr8bit.schoolmanager&hl=en

Homework manager is an app that allows users to enter their courses, and then add homework, tests and quizzes and their dates. Adding courses is similar to our app where the user can add/edit courses and it doesn't rely on fetching information from a school database. However, everything else the app does is vastly different from our app, and features such as adding homework and tests are not part of our app, as our app is targeted at helping students get accessible help from advisors, and not for keeping track of a student's assignment/test due dates.

## V. References

"Stack Overflow," *Stack Overflow*. [Online]. Available: http://stackoverflow.com/. [Accessed: 05-Apr-2017].

“Android Apps on Google Play,” *Google Play*. [Online]. Available: https://play.google.com/store/apps/. [Accessed: 05-Apr-2017].
“Android O Developer Preview,” *Android Developers*. [Online]. Available: https://developer.android.com/index.html. [Accessed: 05-Apr-2017].
“GitLab,” *Wikipedia*, 29-Mar-2017. [Online]. Available: https://en.wikipedia.org/wiki/GitLab. [Accessed: 05-Apr-2017].
tutorialspoint.com, “Java How to Use Singleton Class?,” *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/java/java_using_singleton.htm. [Accessed: 05-Apr-2017].
[1] SQLite, https://www.sqlite.org/about.html, last accessed on Apr 04, 2017
[2] Vogella, http://www.vogella.com/tutorials/AndroidSQLite/article.html, last accessed onApr 04, 2017