getNote

SIMPLE NOTE-SHARING

Matthew Harris

Senior Project

Midterm Documentation

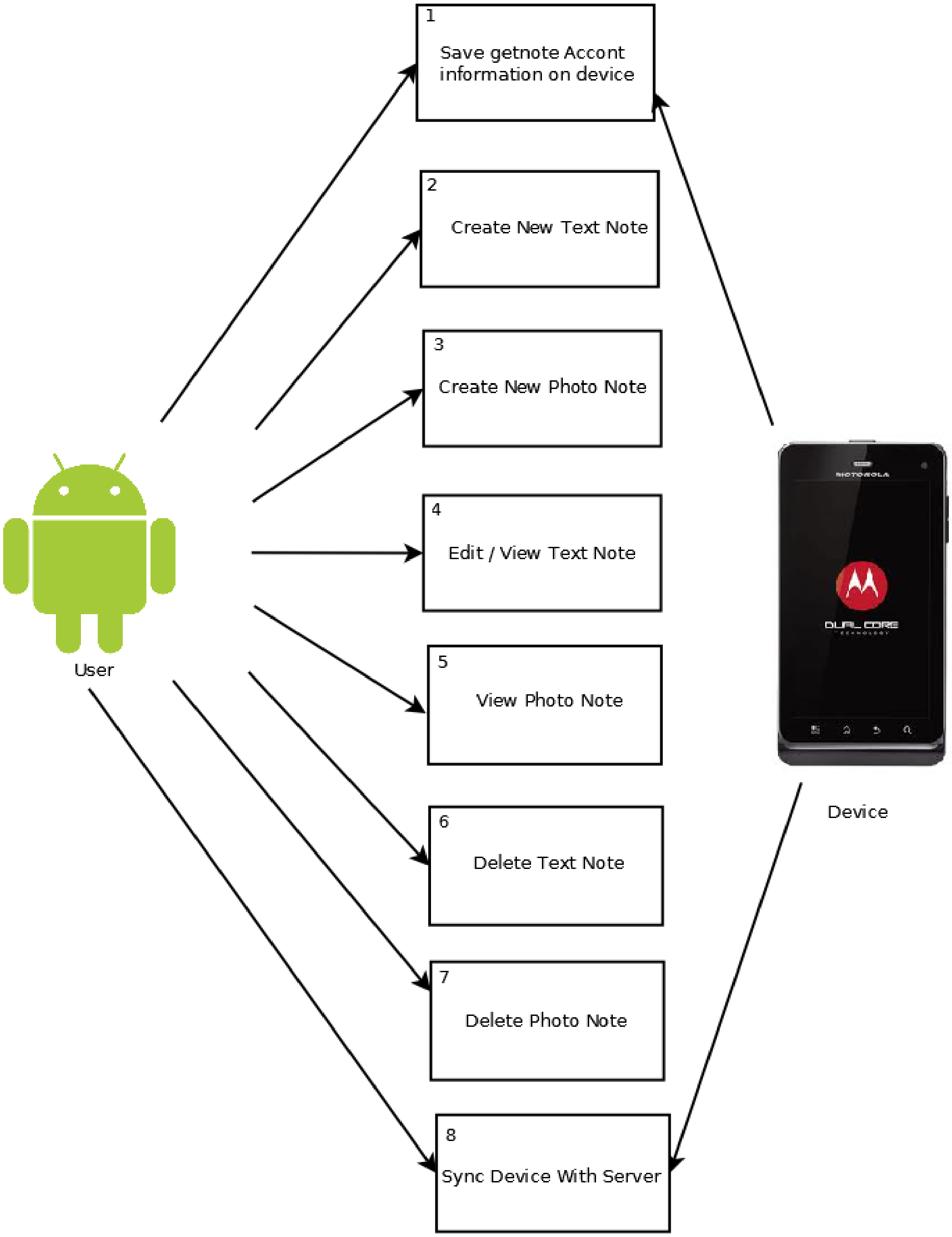
getNote

The Mobile Application

Table of Contents

Student: Matthew Harris
Project: getNote[1] Android[2] application
Collaborators: Dr. Melody Stapleton
Client: getNote
Users: Students studying computer science.

Use Case Diagram



[1] getNote is a registered domain and the Intellectual Property of Matthew Benjamin Harris
[2] Android is a registered trademark of google Inc.

Use Case Scenarios / User Manual

1. Save getNote account information on the device

   - Actor: User

     ✗ The first time the user opens the getNote application, they will be taken to a screen asking them to enter their school email and password (the same as on the server). The user will then press submit, and they will be taken to the *home screen*.

   - Actor: Device

     ✗ Once the user has entered in their username and password, and the device has connection, the device will call the *lookup service* which will pull all the users notes off the server and save them on the device.

2. Create New Text Note

   - Actor: User

     ✗ The user starts on the *home screen*, and will selects the “menu/settings” button on their device, this will bring up the application menu, They will select “new note” and it will open the *class information screen*. The user will then enter the class, class type, and lecture date for their note. They will then press the Text Note button which will open the t*ext editor screen* where the user will type in their note. Once they are done they will press the submit button. At this point the note will be saved on the device.

3. Create New Photo Note
    - Actor: User
        - ✗ The user starts on the *home screen*, and will selects the “menu/settings” button on their device, this will bring up the application menu, They will select “new note” and it will open the *class information screen*. The user will then enter the class, class type, and lecture date for their note. They will then press the Photo Note button, which will call the devices *camera widget* (the pre-installed camera handler). It will ask if they want to select a photo that is on the device or take a new one. Once they have selected or taken a photo, the camera widget will hand back a pointer to the photo and the device will save the photo.
4. Edit / View Text Note
    - Actor: User
        - ✗ The user starts on the *home screen*, and will select (press) one of the pre-existing text notes listed. This will open the t*ext editor screen* where the user will review or edit the note. Once they are done the will press the submit button if changes where made. Or simply press the back button on the device to return to the *home screen*.
5. View Photo Note
    - Actor: User
        - ✗ The user starts on the *home screen*, and will select (press) one of the pre-existing photo notes listed. This will launch the devices gallery widget which will allow the user to see the photo, once they are done they will select the device back button which will return them to the *home screen*.

6. Delete Text Note

   - Actor: User

     ✗ The user starts on the *home screen*, and will select (press) one of the pre-existing .txt notes listed, they will then be taken to the t*ext editor screen.* The user will then press the “Delete Note” button, this will delete the note. The user will be returned to the *home screen* and the note list will be update.

7. Delete Photo Note

   - Actor: User

     ✗ The user starts on the *home screen*, and will select (press) one of the pre-existing .jpg notes listed, they will then be taken to the device's gallery widget . The user will select the “menu/settings” button on their device this will bring up the application menu, the user will then select “Delete”, this will delete the note. The user will be returned to the *home screen* and the note list will be update.

8. Sync Device With Server

   - Actor: User

     ✗ The user starts on the *home screen*, and will selects the “menu/settings” button on their device this will bring up the application menu, the user will then select “refresh”. This will update the list of notes on the *home screen* and/or the server.

   - Actor: Device

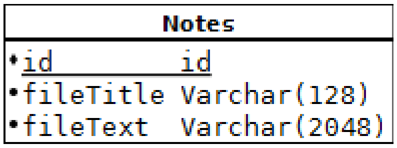
     ✗ The device will call the lookup service every time there is connectivity to sync the device with the server. Pulling and pushing files to and from the server.

UML Class Diagram Introduction

Thus far I foresee creating Six java classes, four activities, one service, and one helper class for the SQLite[3] database. In the Android® SDK classes that extend activity are the screens the user interacts with, while class that extend service run in the background, which are not seen by the user but can be used by a user, an example would be when a user wants to update their list of notes from the server. The screen does not change only updates. My UML Class Diagram is on the next page (page 8).

File System

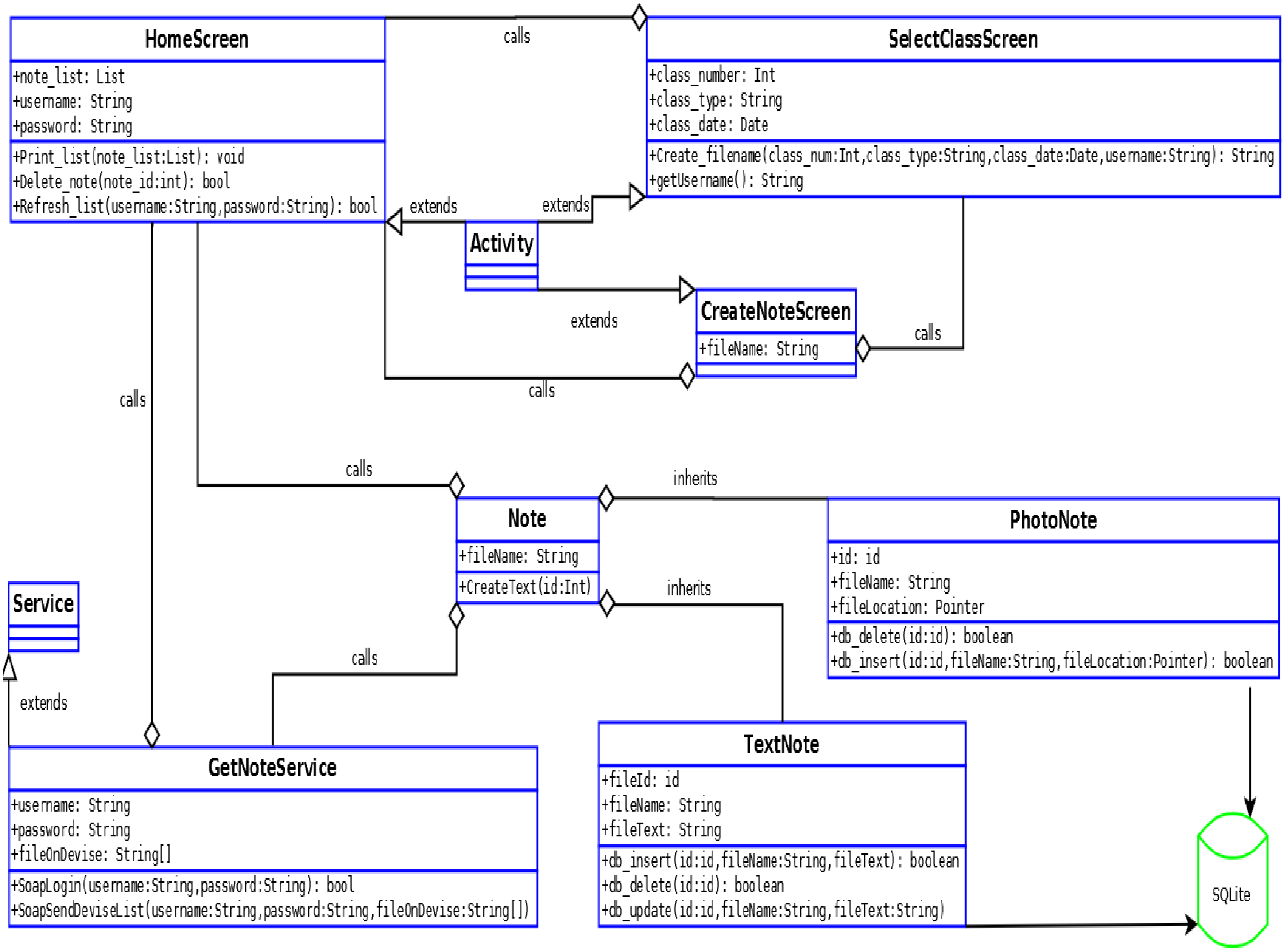
Through out my reading of the Android website[1] and my Android books[2][3] I have learned that SQLite is not a replacement or alternative for a MySQL[4] database[2]. Its a quicker way of fopen a indexed file system[3]. The most common set up for a SQLite® database is one table with three attributes id, file name, and file content[3]. This works well with saving the notes but the little things like the list of class' and course types which are stored in the sites database are stored in the application string file which can be updated dynamically. One down side of this is that I don't have a ER diagram rather a table of index's for my files.



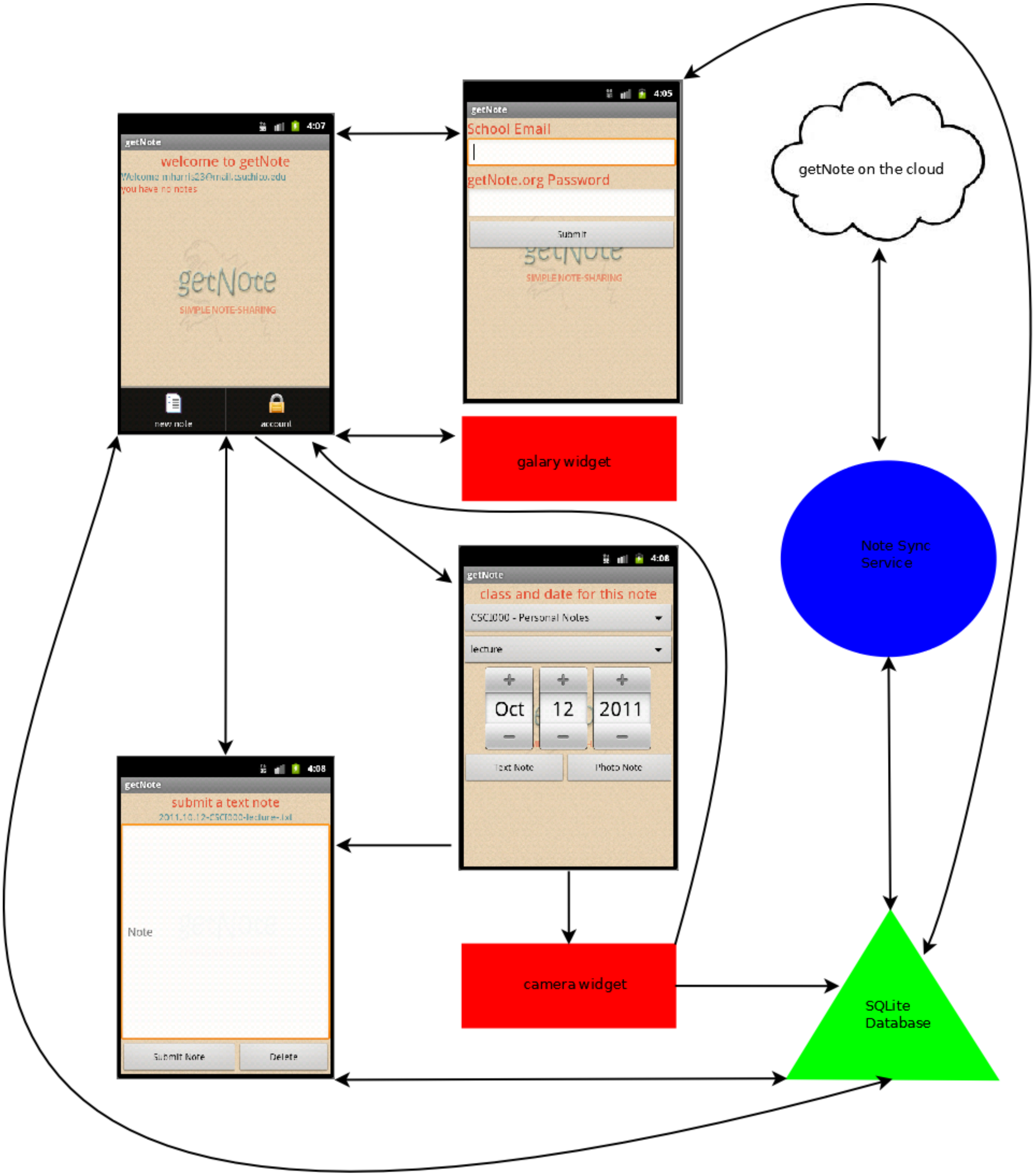
Notes	
•id	id
•fileTitle	Varchar(128)
•fileText	Varchar(2048)

[3] SQLite is in the public domain.
[4] MySQL is in the public domain.

UML Class Diagram


HomeScreen
+note_list: List
+username: String
+password: String
+Print_list(note_list:List): void
+Delete_note(note_id:int): bool
+Refresh_list(username:String,password:String): bool
SelectClassScreen
+class_number: Int
+class_type: String
+class_date: Date
+Create_filename(class_num:Int,class_type:String,class_date:Date,username:String): String
+getUsername(): String
Activity
CreateNoteScreen
+fileName: String
Note
+fileName: String
+CreateText(id:Int)
PhotoNote
+id: id
+fileName: String
+fileLocation: Pointer
+db_delete(id:id): boolean
+db_insert(id:id,fileName:String,fileLocation:Pointer): boolean
Service
GetNoteService
+username: String
+password: String
+fileOnDevise: String[]
+SoapLogin(username:String,password:String): bool
+SoapSendDeviseList(username:String,password:String,fileOnDevise:String[])
TextNote
+fileId: id
+fileName: String
+fileText: String
+db_insert(id:id,fileName:String,fileText): boolean
+db_delete(id:id): boolean
+db_update(id:id,fileName:String,fileText:String)
SQLite
calls
extends
extends
extends
calls
calls
calls
calls
inherits
inherits
calls
extends

Android® Class Diagram



References:

[1] http://developer.android.com/index.html
http://developer.android.com/reference/android/database/sqlite/package-summary.html
Copyright © 2011 google Inc.

[2] Haseman, Chris; Android Essentials http://www.apress.com
Copyright © 2008 by Chris Haseman

[3] Murphy, L Mark; Beginning Android http://www.apress.com
Copyright © 2009 Mark L Murphy