**C868 – Software Capstone Project Summary**

**Task 2 – Section C**

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | http://www.idevnews.com/views/images/uploads/general/wgu_logo.png  Superior Church CRM Mobile Application |
| **Student Name:** | Andrew Merzwski |

Table of Contents

[Task 2 Part C – C868 Software Development Capstone 4](#_Toc43200299)

[Application Design and Testing 4](#_Toc43200300)

[Design Document 4](#_Toc43200301)

[Class Design 4](#_Toc43200302)

[UI Design 5](#_Toc43200303)

[Unit Test Plan 7](#_Toc43200305)

[Introduction 7](#_Toc43200306)

[Purpose 7](#_Toc43200307)

[Overview 7](#_Toc43200308)

[Test Plan 8](#_Toc43200309)

[Items 8](#_Toc43200310)

[Features 8](#_Toc43200311)

[Deliverables 9](#_Toc43200312)

[Tasks 9](#_Toc43200313)

[Needs 10](#_Toc43200314)

[Pass/Fail Criteria 10](#_Toc43200315)

[Specifications 10](#_Toc43200316)

[Procedures 11](#_Toc43200317)

[Results 11](#_Toc43200318)

[C4. Source Code 12](#_Toc43200319)

[C5. Link to Live Version 12](#_Toc43200320)

[User Guide 13](#_Toc43200321)

[Introduction 13](#_Toc43200322)

[Installation and Using the Application 13](#_Toc43200323)

[Login Screen 14](#_Toc43200324)

[Home Screen 14](#_Toc43200325)

[Log out/switch users 15](#_Toc43200326)

[My Groups 15](#_Toc43200327)

[Events 16](#_Toc43200328)

[Member List 17](#_Toc43200329)

[My Contributions 19](#_Toc43200330)

[Group List 20](#_Toc43200331)

[Reports 20](#_Toc43200332)

[Guest Access 21](#_Toc43200333)

[Administrator Guide Addendum 22](#_Toc43200334)

[DB Browser for SQLite 22](#_Toc43200335)

[View device database(s) on DB Browser 22](#_Toc43200336)

[Perform SQL Transactions on Application Database 24](#_Toc43200337)

[Perform SQL Transactions on Authentication Database 24](#_Toc43200338)

[Adding Sample Data 25](#_Toc43200339)

# Task 2 Part C – C868 Software Development Capstone

# Application Design and Testing

# Design Document

## Class Design

Figure 1 (below) shows a complete Entity Relationship Diagram (ERD) for the project classes. The application data will be distributed 6 distinct classes, each of which represent a logical or physical object that data will be stored for.

Two classes on the left side of the image, LoginEntity and EventEntity, are stand-alone classes – they are not dependent on nor do they define any other class. The LoginEntity is implemented in a separate database and is used for user authentication.

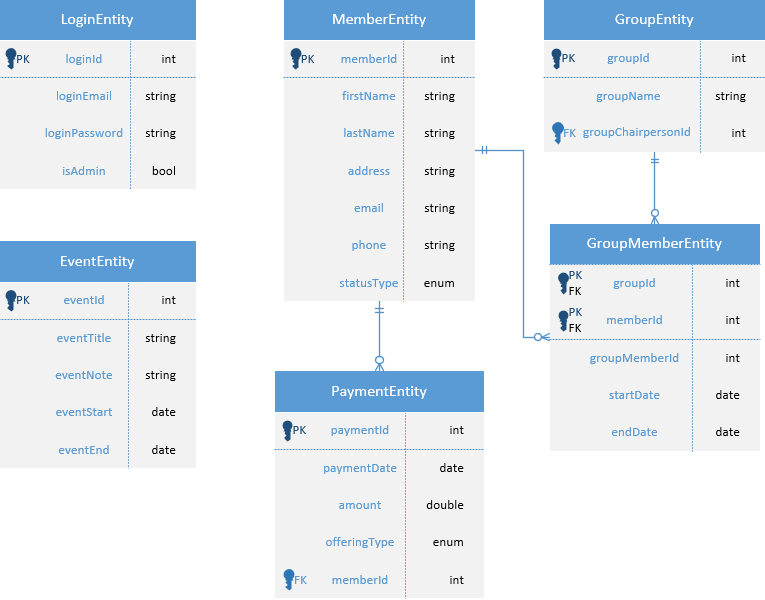


Figure : ERD Diagram

The “main” project class is MemberEntity, which represents the application user. Users can make payments (donations) to the church and can be members of intraorganizational groups/committees. Due to the many-to-many relationship between members and groups, the GroupMemberEntity associative class was created to implement the relationship in the RDBMS. Furthermore, the member class serves as a foreign key to the group class, which represents the chairperson/primary contact of a church subgroup or committee.

## UI Design

The initial entry point to the application is the login screen ([Figure 2](#fig_2)), which will serve to authenticate users and determine if the user has admin privileges. Admin users will be able to perform insert/update/delete actions on all entities (Note: Login entities will not be able to be modified through the application itself).

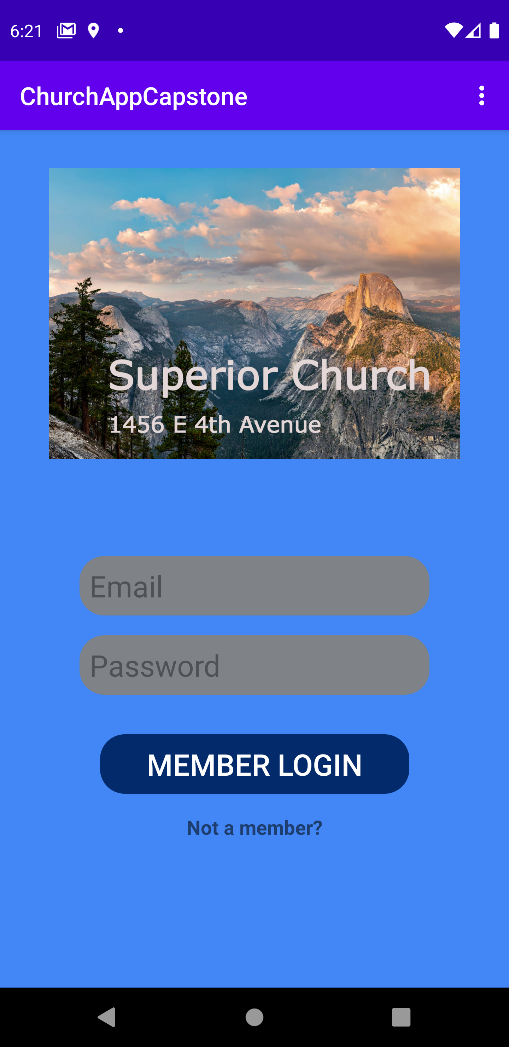


Figure : High-fidelity login screen

Another feature of the application is to provide limited guest (non-authenticated user) access to the church’s event calendar as well as enable guests to contact church staff with questions. Guest access is provided by clicking the “Not a member?” text area at the bottom of the login screen.

Figure 3 shows a low-fidelity representation of the main hub of the application, the home screen, which is shown to the user after login authentication. From this screen, users will be able to click on a button to navigate to the category of their choosing. A sample category display is shown in Figure 4. An outline of various user tasks (color-coded per Figure 4) are as follows:

* Clicking a line item will take users to a screen that provides more detail regarding the specific item. Admin users will additionally be able to modify or delete a list item on the item detail screen (Figure 5).
  + Clicking the search icon will allow users to search for specific events by entering a keyword that will be queried against the event title and the event note/description.
    - Clicking the filter icon will allow users to filter the displayed list of events by week or by month (default is all events).
      * Clicking the add icon (admin only) will allow a user to add an event to the database.

Figure 3: Low-fidelity home screen

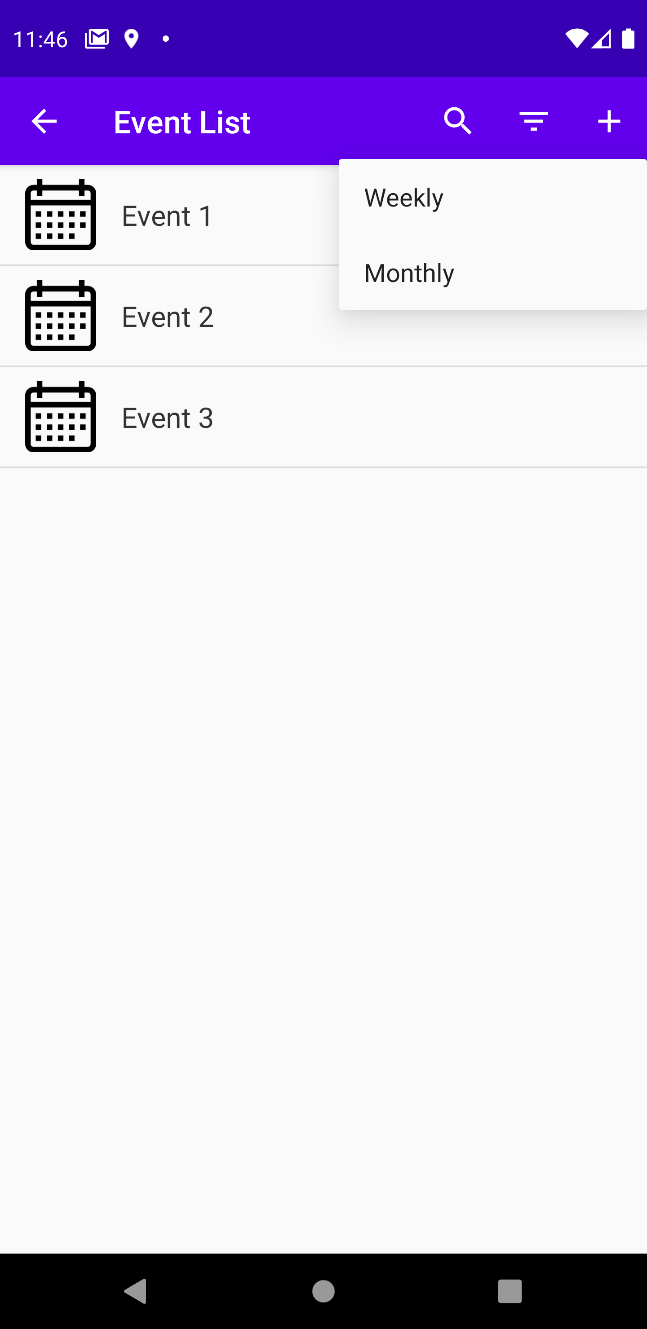
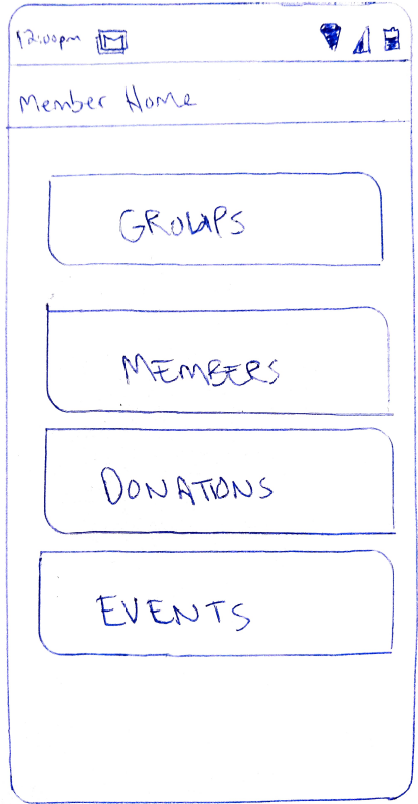


Figure 4: High-fidelity event list screen

The application layout will involve lists that display data and lead to more specific detail. The primary actions for each screen will be displayed on the action bar on the top of the screen.

# 

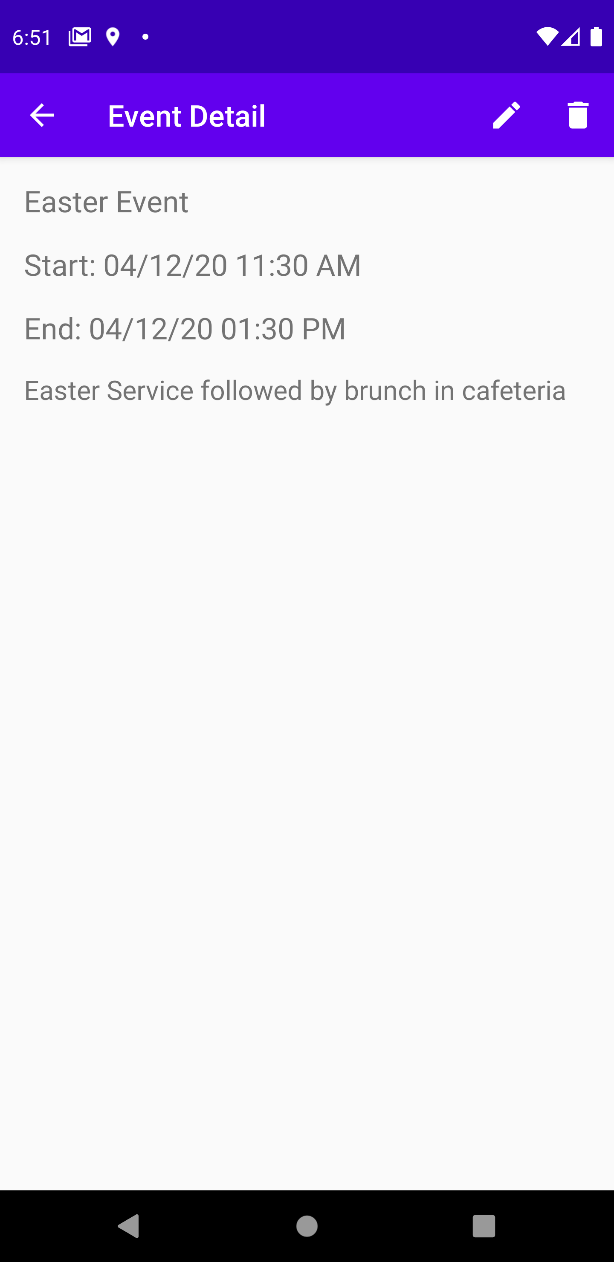


Figure 5: Item detail screen

# Unit Test Plan

## Introduction

### Purpose

Testing will consist of a combination of unit tests, functional tests, and accessibility tests. The purpose of testing is to discover areas of the program that do not function as intended or desired and will be performed throughout the implementation and testing phases of the application lifecycle.

### Overview

The unit tests are primarily concerned with how the application communicates with the databases that are holding the login credentials as well as the rest of the application class data. The unit tests are created by the developer, are automated when using Android Studio, and are located in the “androidTest” module outside of the main application directory. The unit tests take sample data and perform SQL operations on the database using the Room-generated data access objects (DAOs). One major category of the unit tests involves making inserts/updates/deletes on class entities and comparing that entity with the corresponding row in the database using code assertions. This testing is performed for all of the classes listed in the ERD (Figure 1) to the extent they will be used by the end user. Another test involves making sure that database primary keys are generated automatically and correctly by the database.

Throughout the code development stage, regression testing and functional testing is performed to discover bugs and verify that the requirements are being met. This testing involves a semi-formal battery of simulated user actions: sending an email to another member, editing the title of an upcoming event, attempting to login with and without valid credentials, attempting to gain unauthorized access as a guest user, etc. As each code section is developed, the program is run on an emulator and the parts of the program that are affected by the new code are tested to ensure they are functioning correctly.

## Test Plan

### Items

The program needs to be compiled on Android Studio IDE. The tests need to be run using either an actual device in debugging mode, or through an emulated device configured in the IDE.

### Features

Each test will first create a non-persistent database for storing temporary test data, followed by a call to the SampleData class to return a premade list of objects of the tested type. Depending on the test, a method in the entity’s DAO class is called to perform a single SQL command – either insert, update, or delete. At this point, a JUnit assertion method is called to verify the results of the SQL transaction. For example, in the method testing the deletion of an entity in the database, the integer size of the sample data list would be compared to the row count in the database for the corresponding table. If they are not equal, the test will pass. If they are equal after the delete transaction is called on, the test will fail.

### Deliverables

Each test will display a visual result in the Logcat section of the bottom toolbar of the IDE, using the tag “app\_test”.

### Tasks

A new public Java class was created in the androidTest folder. Both the class and the methods were defined with JUnit annotations. Private instances of both the authentication and the application database were defined using the Room.inMemoryDatabaseBuilder() method and corresponding DAO members were constructed for each of the classes outlined in Figure 1.

The test methods were defined, with the “@Test” annotation, according to the functionality of the application. Because user login entities can only be queried by a SQL select statement from the application, tests for selecting and inserting rows into the database was created. For the main application entities, 3 methods were created to test insert, update, and delete statements for all entity classes.

The general framework for each test method is as follows: use the DAO member to call method that performs SQL transaction, call the JUnit assertion method, and finally log a message to the console depending on test failure/success.

### Needs

A computer capable of running Android Studio (current version 4.0) is required to run the tests. All unit tests were developed using the JUnit4 (androidx.test.ext.junit.runners.AndroidJUnit4) framework.

### Pass/Fail Criteria

The pass/fail criteria for each test was whether or not the data in the database matched the expected/original data from the SampleData list, using the JUnit assertion methods. If the assertion returns true, the test is considered to have passed. If false, then the test failed.

Successful tests print a simple confirmation message in the Logcat window. Failed tests throw an AssertionError and the error message is displayed in both the Logcat and Run output windows. For failed tests, a root cause would be determined, the problem code would be modified, and the tests would be run again to verify success. When failed tests could not be immediately debugged, a comment would set in the source code above the test method, including the error message as well as steps taken to resolve the issue up to that point.

## Specifications

Figure 6 shows the test module code to determine if insert transactions to the user authentication database are performing as intended.

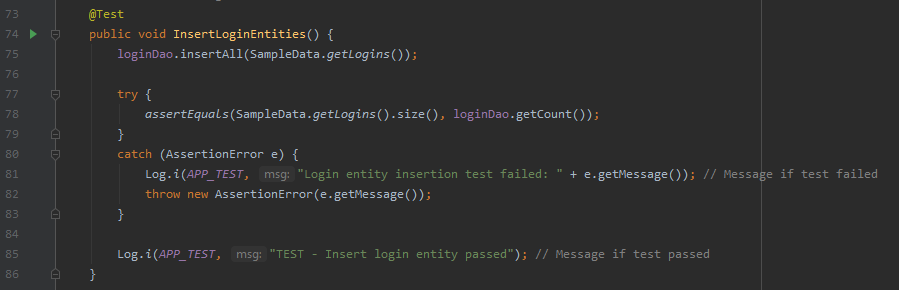


Figure 6: Sample test code

## Procedures

To run the full test suite, in the module directory, right-click either the androidTest folder for the application package or the DatabaseTest class located in that folder and select either Run or Debug from the options menu. To see the output messages, click on the logcat window and type “app\_test”, without the quotes, in the search bar. Single tests may be performed by performing the same action in the method code of the DatabaseTest.java class.

Pass/fail results for all tests are sent to the Logcat window. Failure is also indicated in the Run window as an execution error message.

Testing iterations were (and should be) performed any time the database or entity classes were changed.

## Results

Figures 7 & 8 show the Run and Logcat output for the full test suite, respectively.

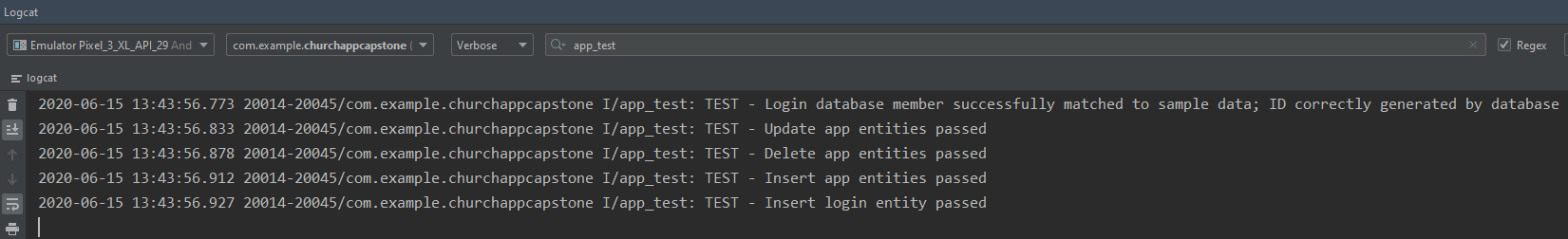


Figure 8: Logcat output of successful test

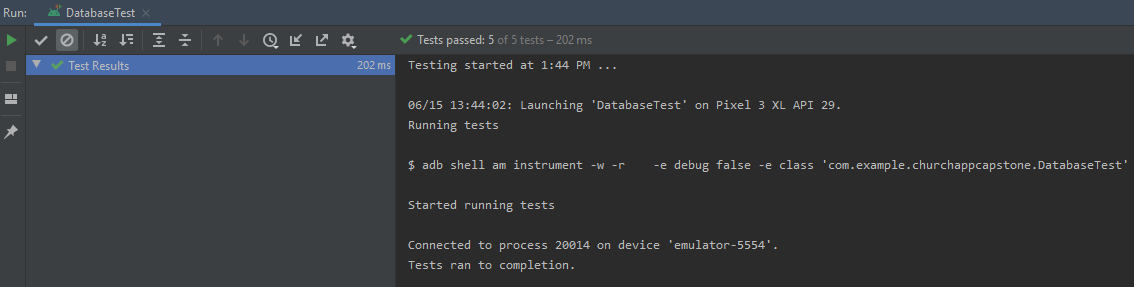


Figure 7: Run output of successful test

The Run output shows the actual results of the tests, whereas the Logcat output shows the statements printed by the execution of the method code itself.

Figure 9 shows the Run output of a test failure. In this example, the email address for a row in the authentication database was compared with the expected original email address and did not match. Note that the message at the top of the window shows both the expected and actual values being tested.

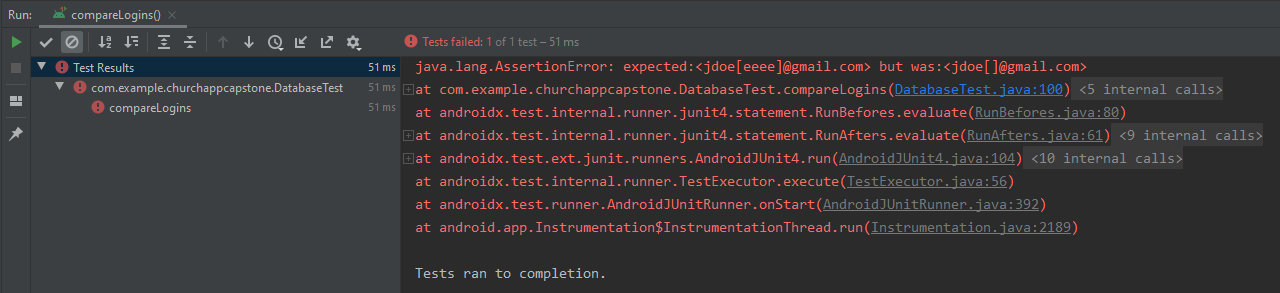


Figure 9: Run output of failed test

# C4. Source Code

All source code can be found in the submitted .zip file “ChurchAppCapstone.zip”.

# C5. Link to Live Version

Not applicable for this project.

# User Guide

## Introduction

This user guide will include how to install, log in, and use all of the functions of the application from an admin perspective (normal users have less, not different, features) and from a guest user perspective.

## Installation and Using the Application

To install, simply download the .apk file, found in the root directory of the submitted .zip file, onto your device and open the file. If using an emulator, the file can be dragged and dropped onto the emulator screen. The application may be launched by clicking the appropriate icon in the device application list after installation is complete.

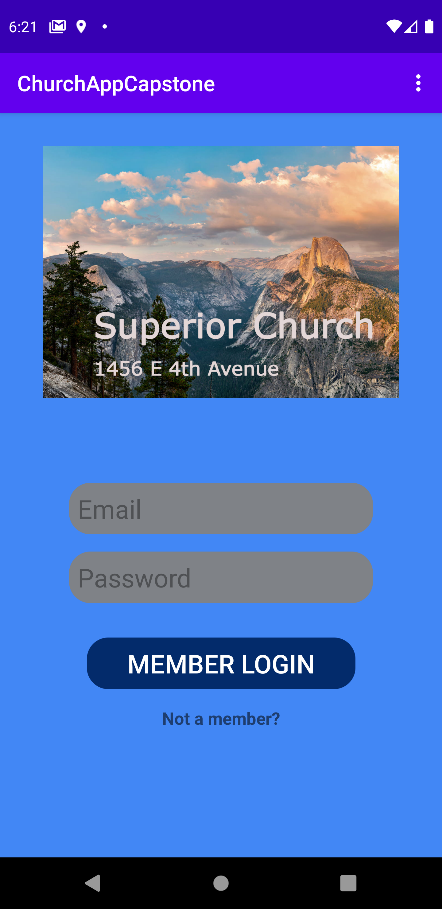


Figure 10: Login screen

## Login Screen

1. Enter email address and password associated with account and click the “Member Login” button underneath ([Figure 10](#fig_10)). The program will display a toast notification when entered fields are invalid or incorrect.
2. If a user does not have an account, they still have [guest access](#guest_access) to limited information.
3. Pressing the device “back” button while on this screen will exit the program.

## Home Screen

(*Note: All edit/add entity screens are similar in layout and functionality and will only be visually displayed in this guide once. Each list in the user guide has a corresponding callout in the related figure, where applicable)*

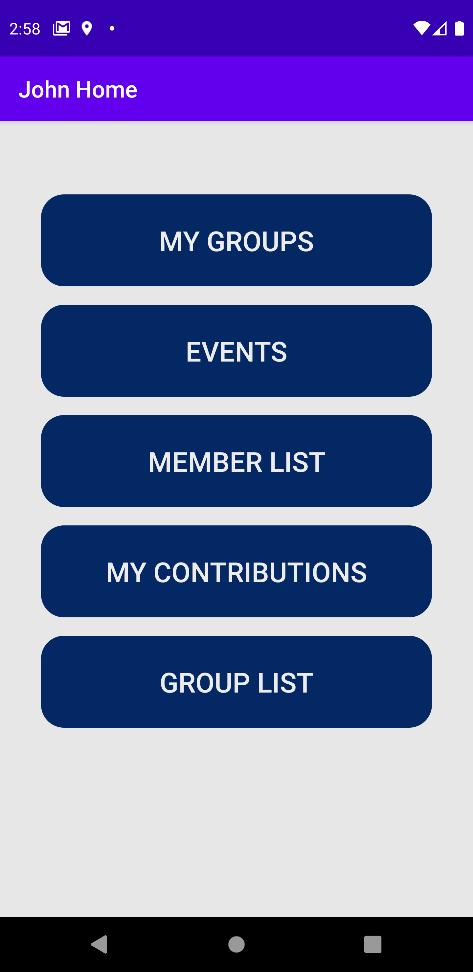


Figure 11: Admin home screen

### Log out/switch users

Press the device back button twice in rapid succession. Pressing it once will display a toast informing the user the button needs to be pressed once more to successfully log out.

### My Groups

Shows a list of church sub-groups that the user is a member of ([Figure 12](#fig_12_13)).

#### Group List Screen

1. Selecting a list item will display details for the selected group ([Figure 13](#fig_12_13))

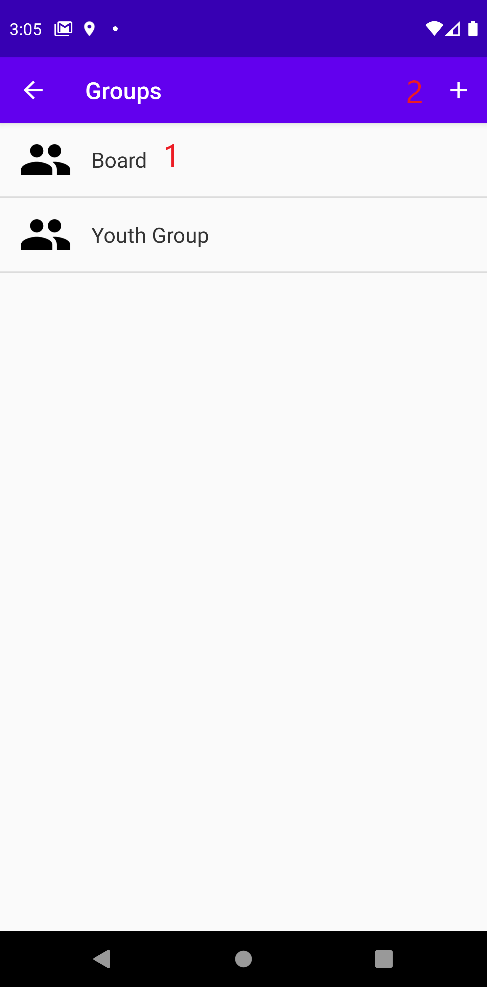


Figure 12: Group list screen

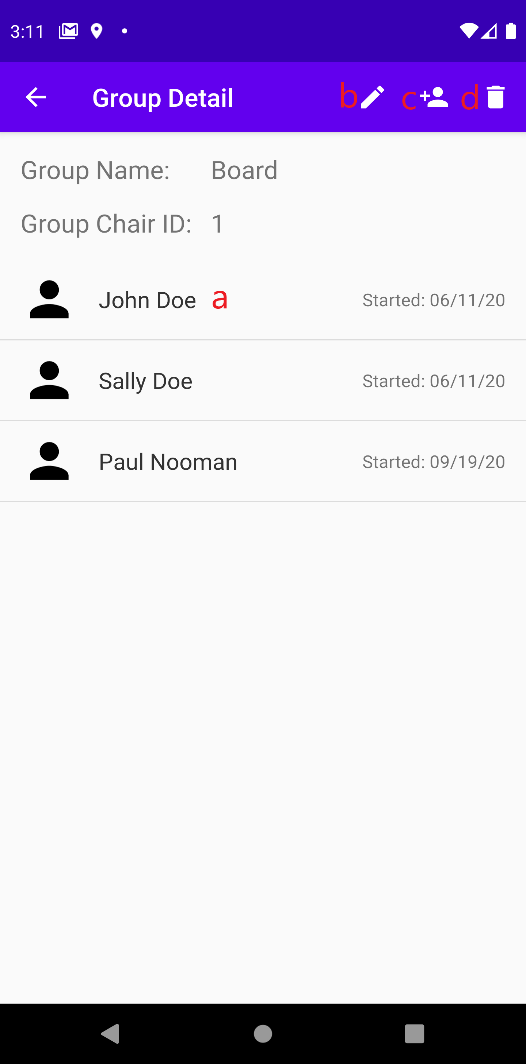


Figure 13: Group detail screen

##### Group Detail Screen

* 1. Clicking on a group member from the list will display details for that group member and enable the admin to edit or remove the member from the group.
  2. Clicking the edit icon will allow the admin to edit the group name and chairperson (similar to the [“add new group” screen](#add_new_group) below).
  3. Clicking the “add member” icon will allow admin to add a member to the group.
  4. Clicking the delete icon will remove the group from the database.

1. Clicking the “Plus” icon will allow admin to add a new group
   1. After entering a (mandatory) name and selecting a group chairperson from a dropdown list consisting of all members, pressing the check mark icon will insert the group into the database. Not entering a value for the group name will display an on-screen toast.
   2. Pressing the “X” icon on the upper-right side of the screen at any time will cancel the transaction and return the user to the group list screen.

### Events

Shows church events ([Figure 14](#fig_14_15)).

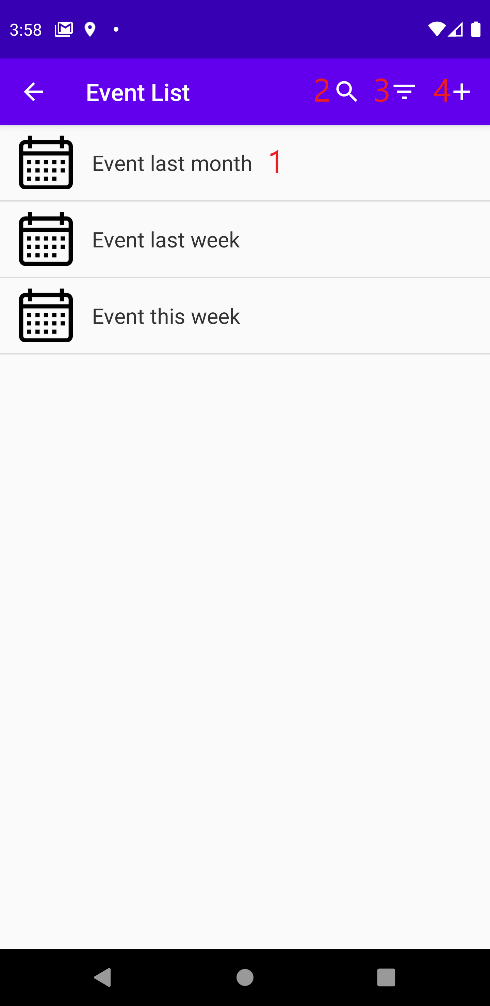


Figure 14: Event list screen

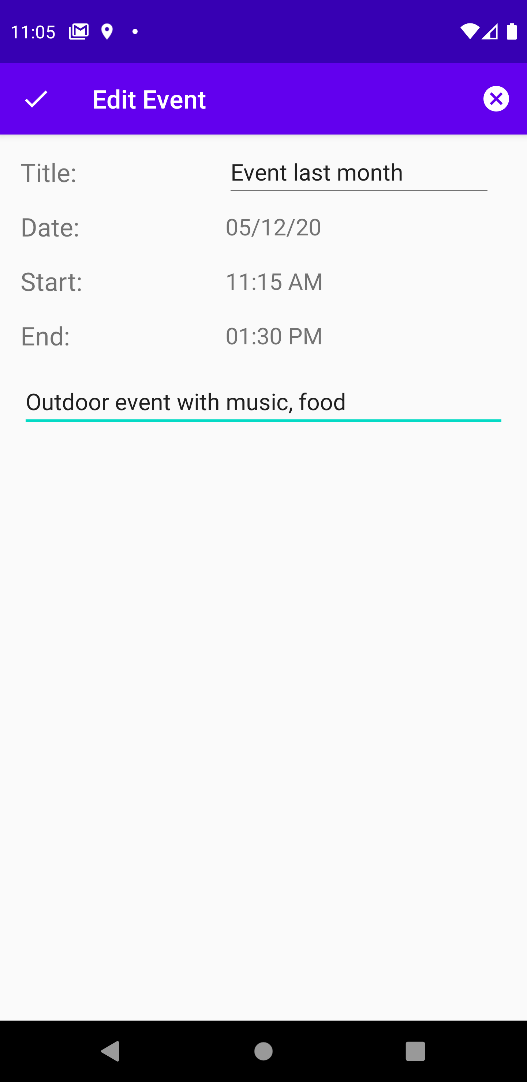


Figure 15: Edit event screen

#### 

#### Event List Screen

1. Clicking a list item will display event details and enable the admin to edit or remove the event ([Figure 15](#fig_14_15)).
2. Clicking the magnifying glass will allow the user to search for event(s) by keyword. A text area in the application bar and the device keyboard are shown and accept user input. The search is automatically performed when the user enters input. This specific search functionality will query the entered search text against the event title and event note for each event in the list.
3. The filter button will enable a report functionality for events. By default, all events are shown. Clicking the filter button will display a submenu, where the user can select an option to display either this week’s or this month’s events. Clicking the filter button again will display all events.
4. Clicking the “Plus” icon will allow an admin user to add a new event, with similar layout and validation rules for [adding a new group](#add_new_group). This screen also includes date and time picker fields that are initiated by clicking on the corresponding text field on the screen.

### Member List

Shows a list of all church members ([Figure 16](#fig_15_16)).

#### Member List Screen

1. Clicking list item will display member details and enable an admin user to edit or remove the member ([Figure 17](#fig_15_16)).

##### Member Detail Screen

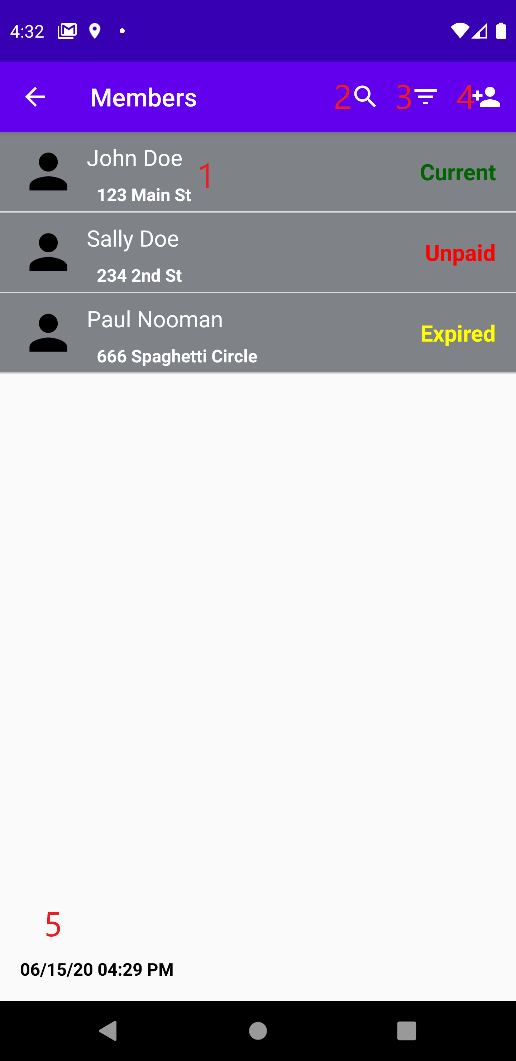


Figure 16: Member list screen

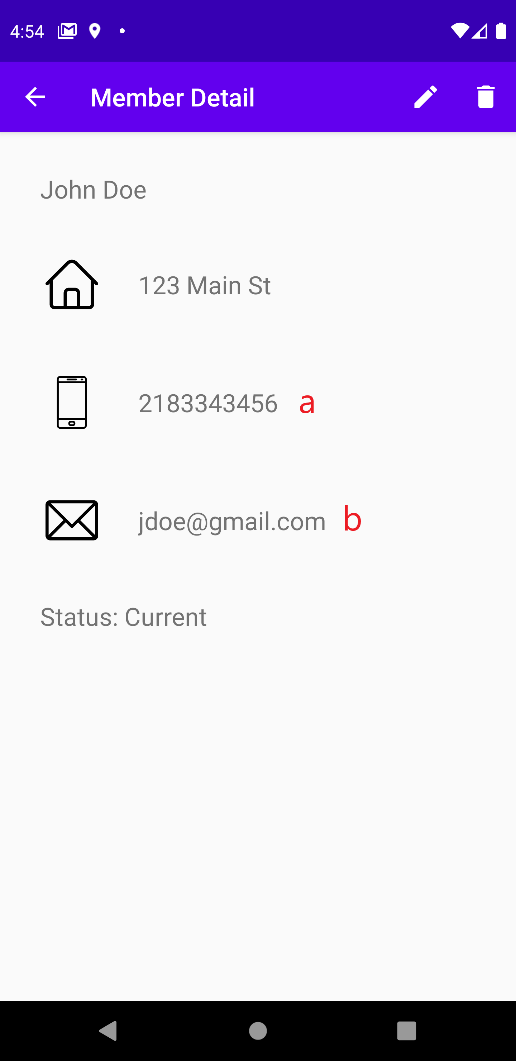


Figure 17: Member detail screen

1. Clicking on the phone icon/text will send the user to the device’s phone app and will allow an authenticated user to make a call or send a text to the specific member.
2. Clicking on the email icon/text will send the user to the device’s connected email app and will allow an authenticated user to send an email to the specific member. If there is no email app connected to the device, the user will be prompted to select an app from a list of device options.
3. Clicking the search icon will provide functionality similar to [searching for events](#search_functionality). Search text will be queried against member first or last names.
4. Clicking the filter will provide reporting functionality similar to [filtering events](#filter_functionality). An admin user can choose to view membership status by “current”, “unpaid” (meant for newly added members), and “expired” (non-new, unpaid members). See also [report timestamp](#timestamp).
5. Clicking the add button will allow an admin user to add a new member, with similar layout and validation rules for [adding a new group](#add_new_group).
6. A timestamp showing the date and time of the most recent filter application is shown at the bottom of the screen.

### My Contributions

Shows a list of contributions to the church made only by the user ([Figure 18](#fig_17)).



Figure 18: Contribution list screen

#### Contribution List Screen

1. Clicking list item will display payment details and enable the admin to edit or remove the payment.
2. Clicking the add button will allow admin to add a new payment, with similar layout and validation rules for [adding a new group](#add_new_group).
3. Clicking on the infinity symbol will send the admin to a new screen displaying contributions from all members. Edit and delete actions may be performed on items from this list, as well.

### Group List

Shows a list of all church groups (admin only), with the same functionality as [My Groups](#_My_Groups).

## Reports

The reporting feature is described above for [filtering members](#filter_members) and [filtering events](#filter_functionality). Each screen contains a title, multiple data points for each list item. The member reporting feature also includes a [timestamp](#timestamp) displaying the date and time the report was generated. [Figure 19](#fig_19) shows an example report for all members currently in “expired” status.



Figure 19: Example member report

## Guest Access

Guests can click on the bottom text of the [Login](#login_screen) screen that states, “Not a member?” to be redirected to a modified version of the [Events](#_Event_List_Screen) screen. In the top menu on the following screen, there is an option to send an email to the church for more info/account creation by clicking the “Contact Us” button, similar to [emailing a member](#email_functionality). This email will be addressed to the church email account, with “Membership/Info Request” in the subject line ([Figure 20](#fig_20)). Account creation will be restricted to church staff/admin personnel through direct database access.

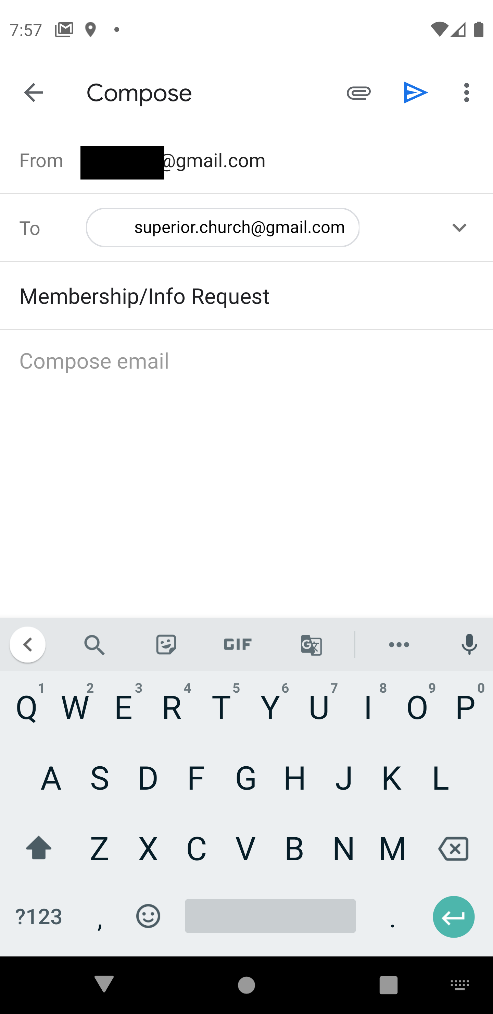


Figure 20: Guest email draft

## Administrator Guide Addendum

The purpose of this section is to provide church staff/admin users information regarding application maintenance and backend database access. Please note that, when making manual database updates, the app cache needs to be cleared on the device prior to running again in order for any changes to be in effect.

### DB Browser for SQLite

DB Browser is a free software that allows you to access and make changes to SQLite databases. It is required to access the user authentication database, as this database cannot be modified through the application itself. The download website is <https://sqlitebrowser.org/dl/>. After the download, run the executable file and follow all instructions to install on your computer.

### View device database(s) on DB Browser

You can view a copy of the application databases by selecting the View 🡪 Tool Windows 🡪 Device File Explorer in Android Studio ([Figure 21](#fig_21_22)).

* On the sidebar that pops up, navigate to the “data/data” path and find the application ([Figure 22](#fig_21_22)).
* If necessary, right click on the application folder and click synchronize.
* Right click on the databases subfolder, click Save As, and save the files in the directory of your choice on the computer.
* Next, with DB Browser running, click Open Database from the toolbar, and find the database copy you saved.

From this point, you will be able to view the structure of all tables and see table contents. If you want to run specific queries, click on the Execute SQL tab, enter the SQL syntax, and click the run/execute button ([Figure 23](#fig_23)).

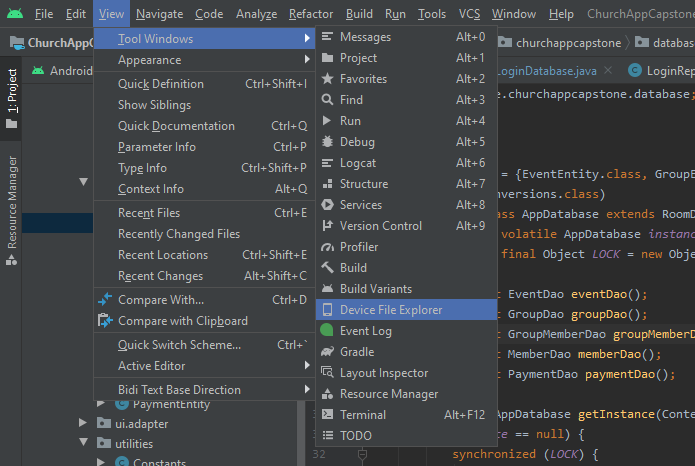


Figure 21: Showing Device File Explorer

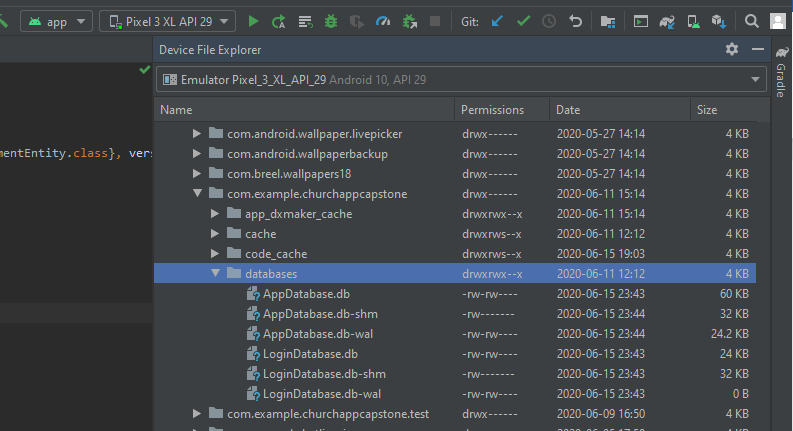


Figure 22: Locating device app data

### Perform SQL Transactions on Application Database

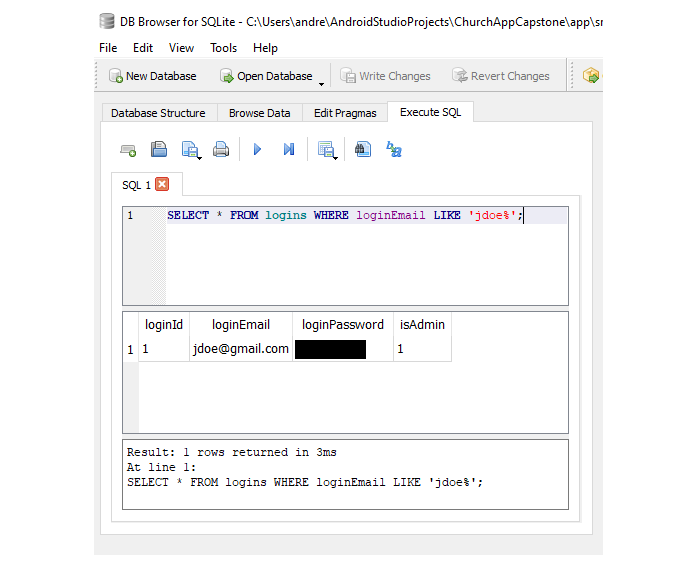


Figure 23: DB Browser SQL query example

All SQL transactions on the main application database can be performed through the application itself. Please refer to the sections detailing the [home screen](#_Home_Screen) above.

### Perform SQL Transactions on Authentication Database

(*Note: App cache needs to be cleared manually before each transaction will be reflected in the application*)

To perform SQL transactions on the login/authentication database, open DB Browser and click Open Database from the toolbar. Navigate to the project source code on the computer and select LoginDatabase.db from the \app\src\main\assets directory. The following instructions refer to [Figure 24](#fig_24) below.

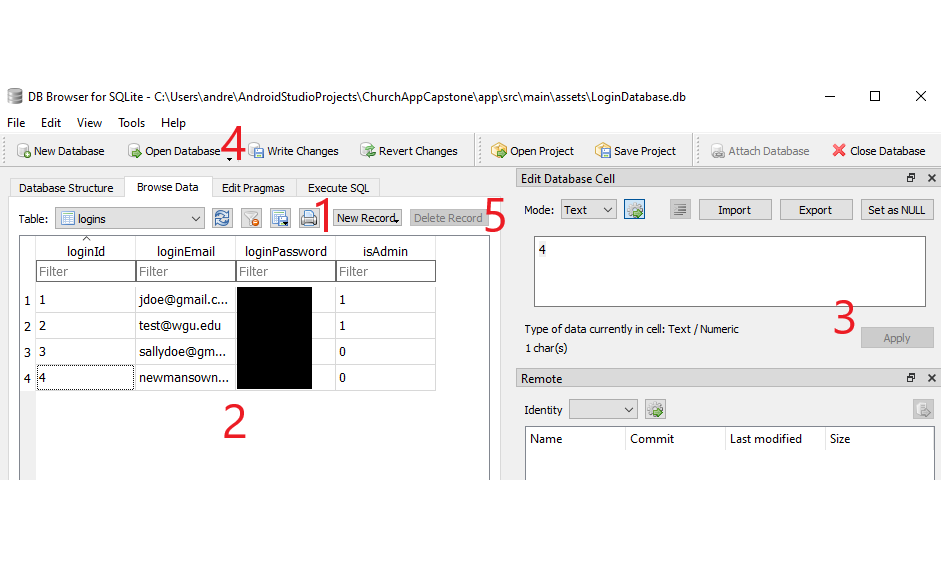
#### Insert a User Login Record

* Click New Record (1)
* Enter data for fields using the browser (2) or the edit cell section (3)
* When finished with insertions, click Write Changes (4)

#### Update a User Login Record

* Select and overwrite the cell(s) to be updated in browser (2) or edit section (3)

Figure 24: Using DB Browser to insert/update/delete records



* When finished, click Write Changes (4)

#### Deleting a User Login Record

* Select the record to be deleted in browser (2)
* Click Delete Record (5)
* When finished, click Write Changes (4)

### Adding Sample Data

To add sample data for system testing, start the application and click the menu icon on the application bar ([Figure 25](#fig_25)). In this menu, click add\_sample\_data. This will populate the main application (not authentication) database with the records in the SampleData.java class.

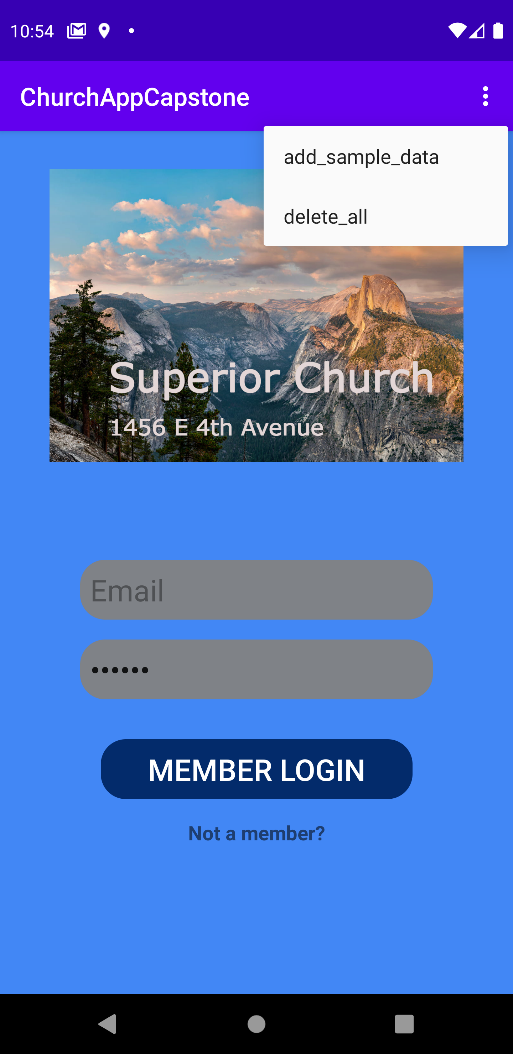


Figure 25: Adding sample data