Arcan Lasso Products

Bill Me Bro

Helping You Keep Track of Spending

CSCI 448: Mobile Applications - Dr. Hoff, Dr. Paone

Chris Butler, Nico Pampe, John Spielvogel

Contents

1. Project Proposal

1.1 Introduction

Working with teams of people can sometime be a stressful process. Many times, there can be miscommunications leading to stressful times which could easily be avoided. These issues can manifest themselves until a team or group falls apart. To proactively counteract the breakdown of a team, it is important to set up a solid basis for communication. This can be done with emails, phone calls, and even texts. But sometimes, the project requires a list of important information that needs to be shared with the team: e.g. purchases, materials, and cost. As the scope of the project develops, a tool to keep track of this can clear up their communications. Arcan Lasso Products has developed an application called “Bill Me Bro” to do just that. With this application a user can easily store their receipts in an organized fashion categorized by type (work, school, personal, etc..) and see exactly where funds have been allocated.

1.2 Overview of Components

Bill Me Bro is an application to log who bought what, when, where, how much it cost, and what else is needed. The application should be easy to set up with a low install-to-use turn around time. The app will keep track of what the project needs, display receipts once items are purchased. For example, a group of roommates may want to track their household spending in order to ensure that costs are being divided fairly. When someone purchases an item, a picture of the receipt is saved and synced across the group’s devices. Once all the items have been purchased, the total cost and cost-breakdown by item and group member are presented. The Bill Me Bro creates an easy environment to track financial of a project and smooth communications between team members.

Bill Me Bro requires several key features to be implemented. First, a database is required for the application to use. This database must keep track of team members, items and whether or not they are still needed, receipts, and contributions. After the database and a basic interface are setup, we can work on other features such as

sending notifications when the database is updated, and using the camera to take pictures of receipts. A picture acts as documentation towards the cost of the project. 2D graphics will be utilized for highlighting sections on the receipt can help indicate important sections such as the total amount or tax added to purchase. Bill Me Bro will utilize touch events to draw on the receipt images to draw attention to important parts of the receipt. Bill Me Bro will have additional smaller features, but the database, notifications, camera, and touch events are the priority components.

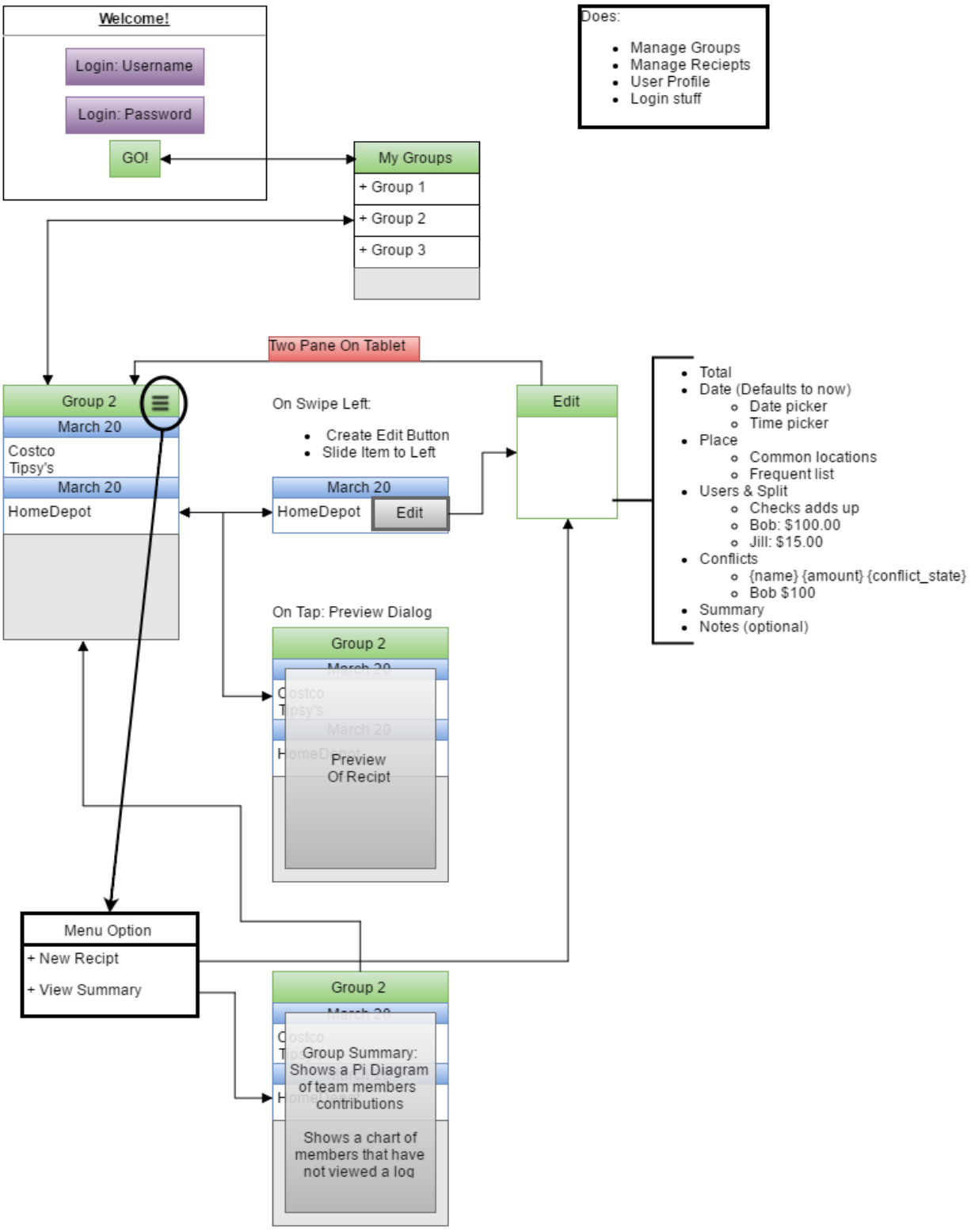
1.3 Expected Challenges

- Drawing on bitmaps might be nontrivial
- I expect anything involving the internet to be frustrating
    - Syncing databases
    - Update notifications
    - Etc.
- Where is everything saving? Can we save data in the cloud with their google account, so that the local copy of the database isn't a master record, but instead always pulled from a master database.
- Highlighting sections of receipt may be difficult
    - Analyzing image for specific sections of data may be difficult
        - requires optical character recognition and image manipulation
    - Relevant xkcd 1425

1.4 Potential Third-Party Libraries

1. Butterknife
    1. Reduce boilerplate
    2. Never use findViewById() again!
2. LeakCanary: mages are HUGE and we'll need to watch for leaks
3. More from Must Have Libraries
    a. Image Processing!

2. Application Storyboard



3. Alpha Release Notes

3.1 Known Bugs

- DataBase does not actually get written to.
    - Causes the receipt list to never update.
    - Receipts don't change.

3.2 Unimplemented Features

- Button to take picture doesn't start activity
- Button to open datepicker
- Button to open timepicker

4. Beta Release Notes

There are always three receipts that are added to show multiple date group. They act as an example.

4.1 Summary

Bill Me Bro is an application to log who bought what, when, where, and how much it cost. The application should be easy to set up with a low install-to-use turnaround time. The app will keep track of what the project needs. Each user will be able to login by using (due to lack of time) solely their username. At this point the user will be able to create different groups of receipts they wish to keep track of. For instance, a user may want a group type of “personal” to store all personal receipts, or a group type of “school” to track all school related expenses. Once a group has been created the user will see a list of receipts containing three example receipts which portray the way the UI will look when fleshed out with receipts. The list of receipts screen menu contains functionality to clear the entire database excluding the 3 example receipts which are hard coded. Once all the items have been purchased, the total cost and cost-breakdown by item and group member are presented. The Bill Me Bro creates

an easy environment to track financial of a project and smooth communications between team members.

Bill Me Bro requires several key features to be implemented. First, a database is required for the application to use. We intended on using a cloud database, but due to time constraints could not deploy a stable version using this. This would have been a feature we would implement before the next release to ensure our intentional goal of syncing across devices would be met. Therefore, we created a local database This database must keep track of team members, items and whether or not they are still needed, receipts, and contributions. A picture acts as documentation towards the cost of the project. Bill Me Bro will utilize touch events to navigate the app and view new groups and receipts. Bill Me Bro will have additional smaller features, but the database, notifications, camera, and touch events are the priority components.

4.2 Featured Components

The following major components were utilized within our application and will be discussed in greater detail in the technical breakdown:

- SQLite Database TODO
- Touch events
- Camera
- Custom Views

4.3 Challenges Encountered

Our main difficulty was getting the database working correctly. We have tables for the users, groups, and receipts. In addition, we made all our views fragments to be useful for the activities.

4.4 Building and Running the Application

Simply open the project in Android Studio's

Let gradle build the project

Run the project, should build on device

4.5 Navigating the Application

4.5.1 Welcome Screen

Opening screen shows a username
Simply put in a username and click Log In
If a user exists, the data is pulled from the database.
Otherwise, a new user is created.

4.5.2 Group Manager Screen

The next activity shows a list of groups the user is currently in
Groups can be added and have the following:
Group Name
Group Type: description of type of group (school, work, etc).
Group Members: user can add members in the group.
Clicking on a group navigates to the receipt list

4.5.3 Edit Group Screen

Accessible by -> Flinging a group viewholder left shows an "EDIT" button.
Swipe left or right while in this view to see adjacent groups

4.5.4 Receipt Manager

In the next activity, two fragments are visible, depending on the orientation. Both have a menu selection for several features

4.5.4.1 Menu Items

1. Summarization: opens a dialog of the total amount of the receipts. In addition, contains a pie chart of the breakdown of the receipts. Pie Chart is a custom view.
2. New Receipt: adds a receipt
3. Delete Receipt: deletes a receipt

4. Remove All Receipts: removes all the receipts from the group

4.5.4.2 Portrait View

An expandable recycle view of the receipts

Each receipt is grouped by Date (Day, Month, Year)

Single tap touch event shows a preview

Swiping left touch event displays an Edit button

    Edit navigates to the `ReceiptEditFragment`

Swiping right touch event hides the Edit button

4.5.4.3 Landscape View

Two-Pane activity with callbacks

Left is a recyclerview of receipts

    A recyclerview of the receipts

    Each receipt is grouped by Date (Day, Month, Year)

    Single tap touch event opens the `ReceiptEditFragment`

Right is the Edit receipt fragment

    Contains: Title, picture, total, date, contributors, conflicts

4.5.4 Receipt Editor

`ReceiptEditFragment` edits the currently selected receipt.

Swipe left or right while in this view to see adjacent groups

Method to take picture using the Camera.

Title: editable text field

Total: onClick -> opens price picker

Date: onClick -> opens datepicker

Contributors: list of users in the group

4.6 Beta Notes

1. There are always three receipts that are added to show multiple date group. They act as an example.

4.7 Known Bugs

1. Rotating device from the AddGroupActivity caused the current activity to fail. App is still running.
2. When the database is cleared, you have to back out of the activity to get a refresh.
3. (Landscape mode only) If you attempt to collapse an expandable recyclerview list item (in the List of Receipts screen) after displaying the detail_fragment view it will crash the application.
4. Entering new username should NOT bring up previous users receipts

4.8 Features To Add Before Next Release

1. A more secure authentication system including passwords.
2. A cloud database

a. To sync across multiple devices

3. A deeper analysis of receipts (in a group) and analysis of group summary.
4. List of contributing members (in “Edit Receipt” screen) does not populate from database as intended

4.9 Easter Egg

There is one easter egg hidden in the app. A person with no name should be able to find it.

5. Application Component Breakdown

5.1 SQLite Database

We used the the stock SQLite database available through the Android API. Our database schema consists of four tables: users, groups, groups_list, and receipts. Each table and its columns are described in the following tables. All columns are nonnull.

The users table list all of the data associated with a user. Right now, that's a UUID and a login username. We don't have any passwords, password salts, emails, or other data per user. All further internal usage of our database uses the user_id column to pair data to users. username exists because expecting user to remember something like 123e4567-e89b-12d3-a456-426655440000 is unreasonable. user_id is used as the primary key in this table; It must be unique.

The groups table list all of the groups users create for their receipts. Each entry gets a UUID and the collection of user-facing fields to describe and keep track of groups. The only constraint on the table is the group_id, the primary key.
The groups_list table consists of pairs of users and groups (in the form of user_id's and group_id's). This table should be queried to get a list of groups a particular user has permissions toward. Future additions might add another column to further describe the permissions the user has. Examples could include read-only, collaborator, and owner allowing users to see, but not modify, see and modify, or control others access - respectively.

The fourth table, receipts, contains the meat of the application. Every table thus far exists to supplement this one. Each receipt has a UUID, receipt_id. Tied to every receipt is a single group, group_id. The remaining fields are all user-facing details of the purchase including the dollar total, total, a user-supplied name, title, and the date of the transaction, date. The column day_of_year is used to help segment the receipts when displaying them in a list by grouping same-day receipts together under a header.

users

Column	Constraint
user_id	Primary key
username	Unique

groups_list

Column	Constraint
user_id	Foreign key
group_id	Foreign key

groups

Column	Constraint	Description
group_id	Primary key	UUID
name		User identifier
type		TODO: I have no idea
date_created		Unix Time

receipts

Column	Constraint	Description
receipt_id	Primary key	UUID
group_id	Foreign key	Owning group
title		User label
date		Date of purchase
day_of_year		Used for displaying
total		$$ total

Figure 1 - Database Table Schemas

5.2 Custom View

We created our own custom view while working on the summarize dialog of a group's project. The summarized dialog shows the user what is the total amount from all the receipts in the project, how much each receipt cost, and a chart of distribution of prices.This dialog is meant to show a user a quick summarized overview of the project.

The custom view is used to create the pie chart. We create a class call `MyGraphview` which extends android's `View` class. Our new class is responsible for painting a set of 2D graphics. The class takes the context of the current activity and an array list of values. The list is comprised of doubles, which are converted into an array of degrees. Finally, the predefine `onDraw` method is overridden to let our class draw a pie chart. We loop over the array of degrees and draw an arc using the `RectF` class to set the size of each slice. The following figure shows the dialog of a summarized project.

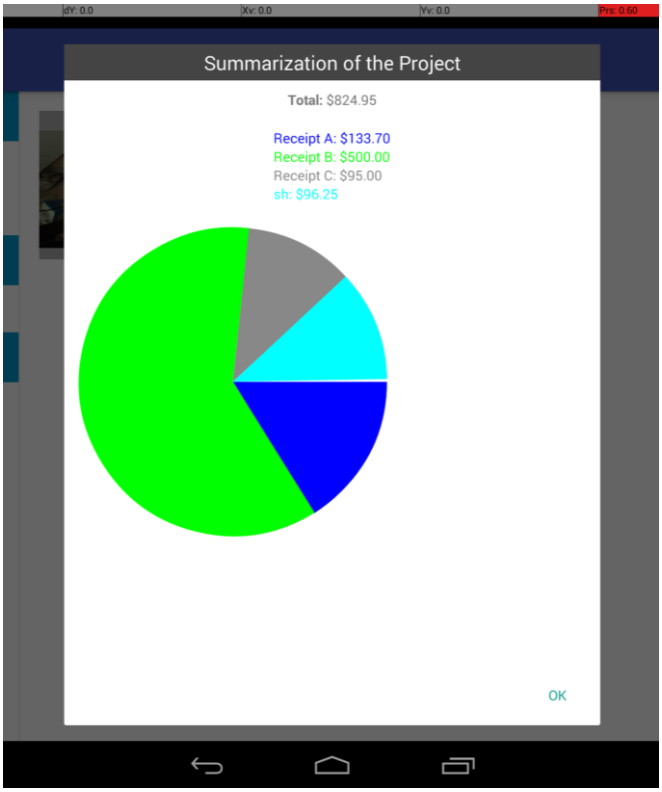


Figure 2 - Project/Receipt Graph Custom View

5.3 Touch Events

There are two areas in the application which utilize touch events in an identical way; the ReceiptListFragment class (List of Receipts screen) and the GroupListFragment class (Group Manager screen). The recycler views in both activities contain view holders which display the data related to the corresponding item you are looking at whether it be a group or a receipt. These view holders have an onTouchEventListener attached which is listening to a gesture detector. Within this gesture detector we have overridden the onFling and onSingleTapUp methods to distinguish between when a user simply taps on a list item or flings it to the left or to the right. When the user flings a group/receipt list item to the left a new “Edit” button will appear within that list item’s view holder. If the user flings the item to the right (after the edit button is visible) the edit button disappears. When the user simply taps on the list item, a new activity is started respective to the item selected (group or receipt). One thing we would have liked to improve upon was the way the fling is executed. We would prefer it to be a smoother action which slowly shows the options coming into the view instead of having a button magically appear. This proved to be more difficult and time consuming to implement and due to our time

constraints we decided it is better to have a simple working implementation rather a “pretty” non-working implementation.

5.4 Camera

Bill Me Bro uses the simple camera feature provided from android devices. They vies contains two objects, an ImageView to display the photo and a Button that takes a new photo. Images are stored as an external storage since they are too large to be placed in a SQLite database. In addition, we simple fire a camera intent rather than trying to access the camera's hardware. The operating system already handles a camera intent for use when the correct action is called. Using the MediaStore class, the public interfaces used in Android can start up the camera intent. Finally, we retrieve the photo from the device’s file system.

5.5 Miscellaneous

The application has implemented quite a few features. One component we felt added a nice feel to the application was the expandable recycler list view which can be seen on the “List of Receipts” page. This component was very difficult to implement at first. After wrapping our minds around the fact that it is simple a nested recycler list view we could begin to see what we should be doing and where. Another component allowing the user to move smoothly through the application are the pager activities found in both the edit receipt and edit group screen. This allows the user to simply swipe to the left or to the right in order to easily access the adjacent group/receipt in the list. This minimizes the amount of time a user has to spend going back and forth, pressing buttons, allowing them to spend more time accomplishing their goal (editing a receipt/group).