GitHub Username: bboccato

BillSMS

Description

BillSMS provides you with a straightforward way to manage your credit card expenses so you will not be surprised at the end of the month. You can set a limit and it will keep track of all your expenses based on the SMSs you get when you swipe your card. We'll let you know when it is time to stop.

Intended User

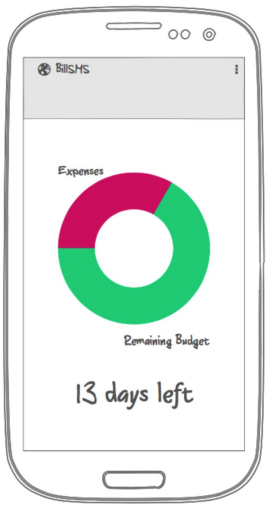
Anyone looking for an easy way to keep track of their credit card expenses and make sure you are still on your budget.

Features

- Set the expire date of your bill
- Automatically add expenses from SMSs received from credt card company
- Set a budget
- Issue a notification if the target budget is reached.

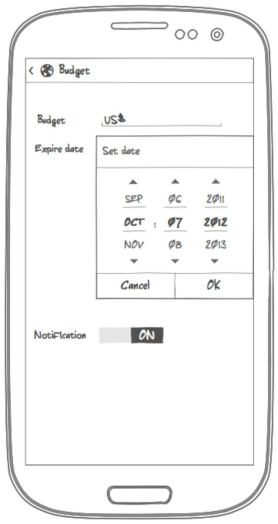
User Interface Mocks

Screen 1



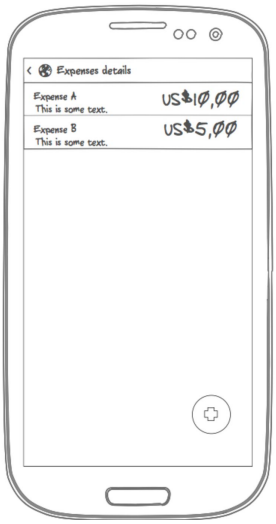
Main activity - Summary of current month expenses vs. remaining budget.

Screen 2



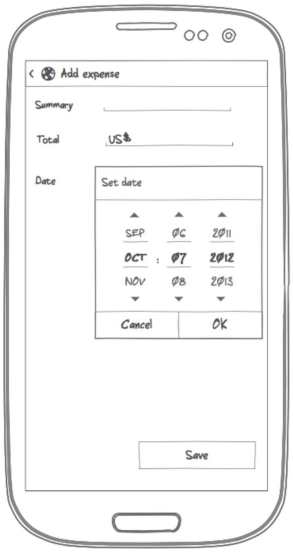
Budget settings - set up monthly budget, credit card expire date and whether or not to send a notification upon reaching the limit.

Screen 3



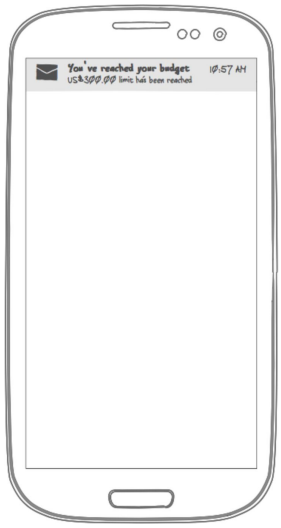
Expense details - a list with all the expenses for the current month - summary, total, and date. FAB button to add expense manually

Screen 4



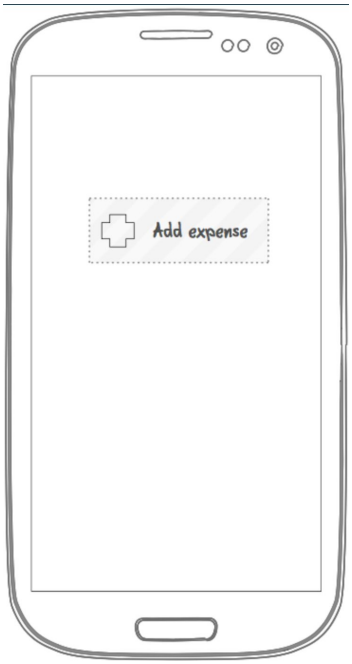
Add expense - add an expense manually - summary, total, and date.

Screen 5



Notification - Notification for when the budget is reached.

Screen 6



Widget - Shortcut to add new expense manually.

Key Considerations

How will your app handle data persistence?

App will collect data from SMS Content provider and persist data on a Content Provider.

Describe any edge or corner cases in the UX.

If there is no budget set, the app will display the expenses only.

Describe any libraries you'll be using and share your reasoning for including them.

Use a lib to draw the pie chart on summary activity, still did not pick up one among ChartView, AnimatedPieView, plain-pie.

Describe how you will implement Google Play Services or other external services.

Use Google Drive service to backup expenses to the cloud so that it is possible to recover data on different devices.
AdMob service to display ads on MainActivity.

Technical considerations.

- App is written solely in the Java Programming Language.
- App utilizes stable release versions of all libraries, Gradle, and Android Studio.
- App keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.
- App uses Loader to load data from Content Provider to activities.
- App includes support for accessibility. That includes content descriptions, navigation using a D-pad.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

- Configure project architecture
- Configure libraries

Task 2: Implement UI for Each Activity

- Build UI for MainActivity
- Build UI for budget setting activity
- Build UI for expense details activity
- Build UI for add expense activity

Task 3: Implement UI functionality

- Add functionality to buttons and menus for each activity
- Add pie graph to MainActivity

Task 4: Implement interface with SMS content provider

- Add functionality to get SMS from Android content provider
- Read SMS info and save it to local DB
- Make requests to Content Provider via an AsyncTask

Task 5: Implement integration with Google Play Services

- Integrate to Google Play Service to backup expenses registered to Google Drive
- Integrate to Google Play Service Admob to show ads on Main Activity

Task 6: Implement notification

- Add notification when target budget is reached

Task 7: Implement widget

- Implement widget that provides a shortcut to add expense manually

Task 8: Implement widget

- Implement widget that provides a shortcut to add expense manually