Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Master's Thesis

Mobile Application for Group Expenses and Its Deployment

Bc. David Vávra

Supervisor: Ing. Zdeněk Míkovec, Ph.D.

Study Programme: Open Informatics

Field of Study: Software Engineering and Interaction

May 9, 2012

Aknowledgements

I would like to thank following people:

- Ing. Michal Kutil, Ph.D. for supporting me in the original idea during semestral work in a class 'Combinatoric Optimization'
- Christine Ebert for helping me to choose a name 'Settle Up' and initial feedback
- Bc. Vladislav Skoumal and the company Bioport Software Labs for efforts with iPhone version, better graphics and marketing
- Ing. Lukáš Barták for his support, detailed feedback and ongoing valuable discussions
- Ing. Zdeněk Míkovec, Ph.D. for supervising my work and new ideas

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Řevnice on May 11, 2012 ....

Abstract

This thesis is about project Settle Up - a mobile application for organizing group expenses. It contains a description of Android implementation, but the main focus is on activity after the release such as feedback, localization, cooperation with iPhone developers, marketing, measurements of user behaviour and monetization. The goal is to provide valuable advice to other mobile developers about how to succeed in a global market.

Abstrakt

Tato práce se zabývá projektem Dlužníček - mobilní aplikace pro správu výdajů ve skupině. V práci je popsána implementace Android verze, ale hlavní důraz je na všechno, co se dělo po vydání aplikace - tedy podpora, lokalizace, spolupráce na vývoji iPhone verze, marketing, měření chování uživatelů a monetizace. Cílem práce je poskytnout ostatním mobilním vývojářům cenné rady jak uspět na globálním trhu.

Contents

1 Introduction **1**
- 1.1 Goals 2

2 Design **3**
- 2.1 Minimal viable feature set 3
- 2.2 Additional features 3
- 2.3 User experience 5
- 2.4 Algorithm for settling debts 5
  - 2.4.1 Basic algorithm 6
  - 2.4.2 Optimal algorithm 7
  - 2.4.3 Transaction tolerance 7
- 2.5 Synchronization 7
  - 2.5.1 Synchronization principles 7

3 Implementation **9**
- 3.1 Overall architecture 9
- 3.2 Algorithm for settling debts in Java 9
  - 3.2.1 Runtime experiments 9
- 3.3 Synchronization API on AppEngine 10
  - 3.3.1 AppEngine 10
  - 3.3.2 API capabilities 11
  - 3.3.3 Tips for API developers 11
- 3.4 Tips for Android developers 12
- 3.5 Evolution of the user interface 13
  - 3.5.1 Screenshots from UI evolution 13

4 Deployment **17**
- 4.1 Google Play Store 17
  - 4.1.1 Publishing an app 17
  - 4.1.2 Tips 18
- 4.2 Competition 18
  - 4.2.1 Conmigo 18
    - 4.2.1.1 Where is Conmigo better 18
    - 4.2.1.2 Where is Settle Up better 19
  - 4.2.2 Tricount 19

4.2.2.1 Where is Tricount better . . . 19
4.2.2.2 Where is Settle Up better . . . 19
4.3 Feedback from users . . . 20
4.3.1 Face to face feedback . . . 20
4.3.2 Problem reports . . . 20
4.3.2.1 Google Groups . . . 20
4.3.2.2 Reports from the app . . . 21
4.3.3 Feature requests . . . 21
4.3.3.1 GetSatisfaction . . . 21
4.3.3.2 Google Play comments . . . 22
4.3.4 Expert advice . . . 22
4.3.5 Communication tips . . . 22
4.3.5.1 Always reply, but know what to refuse . . . 23
4.3.5.2 Be a number one user . . . 23
4.3.5.3 Value active users . . . 23
4.4 Localization . . . 24
4.4.1 GetLocalization . . . 24
4.4.2 Translated languages . . . 24
4.5 Cooperation . . . 25
4.5.1 iPhone version . . . 25
4.6 Marketing and user growth . . . 26
4.6.1 Social networks . . . 26
4.6.1.1 Viral experiment . . . 27
4.6.2 Contests and awards . . . 27
4.6.2.1 AppParade . . . 27
4.6.2.2 aDevCamp . . . 27
4.6.2.3 Paypal X Developer Challenge . . . 28
4.6.2.4 Křišťálová Lupa . . . 28
4.6.3 Media coverage . . . 28
4.6.3.1 April 20 - July 19, 2011 . . . 28
4.6.3.2 July 20 - October 19, 2011 . . . 28
4.6.3.3 October 20, 2011 - January 19, 2012 . . . 28
4.6.3.4 January 20 - April 30, 2012 . . . 29
4.6.4 Advertising . . . 29
4.6.5 Overall statistics . . . 30

5 Evaluation **33**
5.1 Google Analytics for Mobile . . . 33
5.1.1 Pageviews . . . 33
5.1.2 Events . . . 33
5.1.3 Custom variables . . . 34
5.1.4 Tips for developers . . . 34
5.2 Analysis of Settle Up . . . 34
5.2.1 How is Settle Up used in time? . . . 34
5.2.1.1 Interpretation . . . 35
5.2.2 What are the most and least popular functions? . . . 35

5.2.2.1 Interpretation and impact . . . 35
5.2.3 What tasks take users long time? . . . 36
5.2.3.1 Interpretation and impact . . . 36
5.2.4 What's different about online and offline users? . . . 37
5.2.4.1 Interpretation and impact . . . 37
5.2.5 What's different about paid users and free users? . . . 38
5.2.5.1 Interpretation and impact . . . 38
5.2.6 What's different about new users and experienced users? . . . 38
5.2.6.1 Interpretation and impact . . . 38
5.2.7 What currencies are being used? . . . 38
5.2.7.1 Interpretation and impact . . . 39
5.2.8 What payment purposes are being used? . . . 39
5.2.8.1 Interpretation and impact . . . 39
5.2.9 Can UI handle common screen resolutions? . . . 39
5.2.9.1 Interpretation and impact . . . 40
5.2.10 Can UI handle average text input? . . . 40
5.2.10.1 Interpretation and impact . . . 40
5.2.11 What screens should I optimize for landscape mode? . . . 41
5.2.11.1 Interpretation and impact . . . 42
5.2.12 What devices should I test on? . . . 42
5.2.12.1 Interpretation and impact . . . 42

6 Monetization **43**
6.1 Expenses . . . 43
6.2 Donations . . . 43
6.3 Free version with advertising . . . 44
6.3.1 House ads . . . 44
6.3.2 Evaluation . . . 45
6.4 Paid version . . . 45
6.4.1 The price experiment . . . 46
6.4.2 Premium-first functionality . . . 46
6.4.2.1 Evaluation . . . 46
6.4.3 Impact of important events on revenue . . . 47
6.4.3.1 Interpretation . . . 47
6.5 Totals . . . 48

7 Conclusion **51**
7.1 Goal fulfilment . . . 51
7.2 Future work . . . 51

Bibliography **53**

A List of Abbreviations **57**

B Source code of algorithm for settling debts **59**

C Detailed Google Analytics reports **63**

D Contents of attached CD **67**

List of Figures

2.1 Initial mockup of the user interface 4

3.1 Deployment diagram of the whole Settle Up project 10
3.2 Evolution of the app icon 13
3.3 Evolution of the main screen 14
3.4 Evolution of the New Payment screen 14
3.5 Evolution of the Payments Log screen 15
3.6 Evolution of the Settle Debts screen 15

4.1 Screenshots from biggest competition of Settle Up 19
4.2 Problem report e-mail generated by the app 21
4.3 Screenshots from the iPhone version developed by Bioport Software Labs 26
4.4 Active user growth from April 20 to July 19, 2011 29
4.5 Active user growth from July 20 to October 19, 2011 30
4.6 Active user growth from October 20, 2011 to January 19, 2012 31
4.7 Active user growth from January 20, 2011 to April 30, 2012 31

5.1 Data for the question: 'How is Settle Up used in time?' 35
5.2 Data for the question: 'What are the most and least popular functions?' 35
5.3 Data for the question: 'What tasks take users long time?' 37
5.4 Data for the question: 'What currencies are being used?' 39
5.5 Data for the question: 'What payment purposes are being used?' 40
5.6 Data for the question: 'Can UI handle common screen resolutions?' 41
5.7 Data for the question: 'What devices should I test on?' 42

6.1 AdMob advertisement at Settle Up main screen 44
6.2 House ad promoting paid version of Settle Up 45
6.3 Statistics of ad revenue from July 2011 to April 2012 45
6.4 Daily purchases of paid version from November 29, 2011 to February 21 2012 47
6.5 Daily purchases of paid version from February 22 to May 3 2012 48
6.6 Monthly revenue from the paid version 49

C.1 Data for the question: 'What is different about online and offline users?' . . . 64
C.2 Data for the question: 'What is different about paid users and free users?' . . 65
C.3 Data for the question: 'What is different about new users and advanced users?' 66

List of Tables

4.1 Distribution of languages among Settle Up active users, April 20, 2012 25

5.1 Percentage of individual screens in landscape orientation 41

Chapter 1

Introduction

Settle Up was started as a semestral project for class Combinatorical Optimization (A4M33KO)[1] in April 2011. We were told to come up with an algorithm which uses principles taught in the class for some real-world problem. I was thinking about areas of my life that need better optimization. I had just come back from a trip with friends and bills for gas, accommodation, food etc. were a mess. Lots of bills for lots of people, some bills just for a subset of people and I had to spend time in Excel to figure out who owes whom how much. So I proposed an idea of a 'debt settlement' algorithm and mentioned on the side that it might be nice to integrate it into a mobile application (app) because I had previous mobile development experience. Ing. Michal Kutil, Ph.D. accepted this idea and said I should focus on the mobile app and how to improve the experience for people during a trip. I created the first prototype version in 3 weeks and published it on Android Market. I won some app contests with the early version and it gained initial traction in the Czech Republic.

One year later, April 2012, Settle Up[2] has about 36,000 downloads, Android, iPhone and web version, it is localized to 18 languages, it is making some money and the future looks bright. How did I get here? It is quite easy to dive into the code and produce a bare-bone app for an idea in a few weeks. But my goals were higher. Just a few years ago producing a successful app was only possible for large companies, but not any more. With general use of app stores and social networks, a single developer like me can now directly reach global users. There are no initial expenses except for the developer's time. One person can perform multiple roles - developer, designer, marketer, user support and project manager. And sometimes it is more effective than a whole team. This interesting trend is called 'solopreneurship'. In this thesis I will describe what I was doing step by step - focusing on the hard part which took most of the time - the process after the first release. It might be useful for other students and developers who are thinking about following a similar path.

1.1 Goals

I would like to achieve these goals in this thesis:

- Design a mobile application for group expenses.
- Describe synchronization with a cloud server, open application interface (API) and related cooperation with third-parties.
- Show ways of gathering feedback from users related to the app evolution.
- Evaluate marketing efforts related to the user growth.
- Use measurements of user behaviour for the app improvements.
- Propose various business models and evaluate their profitability.

Chapter 2

Design

I am a supporter of 'release early, release often' approach. While designing a new product, the developer cannot know how people will use it and what functions they will prefer. So I decided to release a bare-bone application presenting my idea, quickly iterate and add functionality based on feedback of users. Yet I had to think about future features and make the user interface (UI) and the code easily extendible.

2.1 Minimal viable feature set

I had a lot of ideas, but I knew I had to strip them down to the basics. It really helps to draw individual screens on a piece of paper and iterate. This way you can design first UI and understand priority of features. Figure 2.1 shows my initial mockup. These features were present at release date April 19, 2011:

- Define a group of people
- Add payments - who paid, for who, how much, what purpose
- Who should pay next?
- Payments log
- Settle debts - proposed transactions
- Settle debts - details about how much each person owes
- Export payments and debts via e-mail

2.2 Additional features

User feedback drove adding new features. Read more about feedback interpretation in section 4.3. These features were added later:

- Editing, filtering and deleting payments

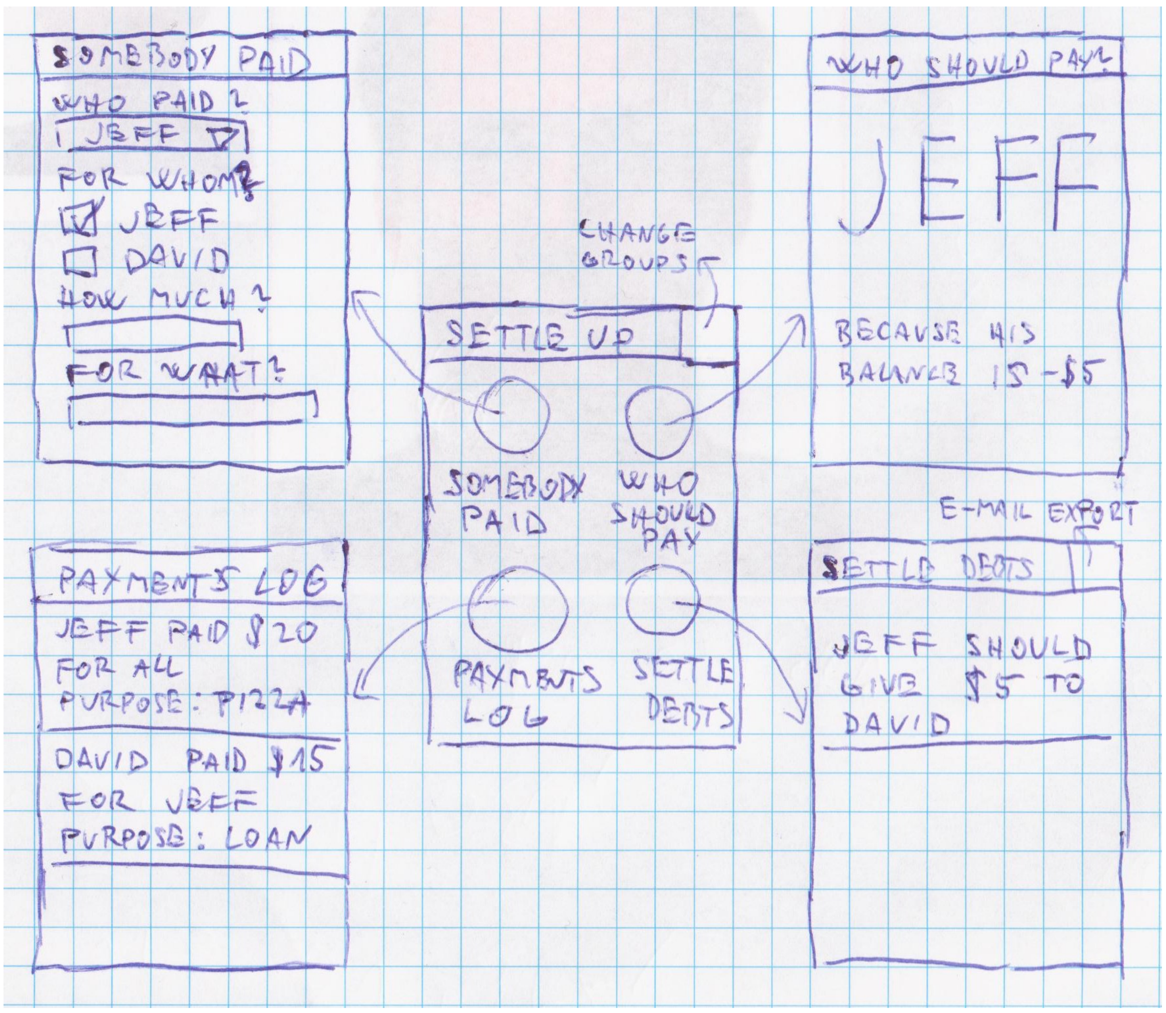


Figure 2.1: Initial mockup of the user interface

- Adding, editing and deleting members
- Mark debt as settled
- Support for multiple currencies and automatic conversion to group currency
- More groups and switching between groups
- Attaching picture of the receipt to the payment
- Synchronizing user's groups with multiple users and devices
- Uneven split
- More people can pay one payment
- Default weights for members (useful for couples or families)

- Repay your debts directly via PayPal
- Import/export to SD card
- Translations to various languages

2.3 User experience

User experience is critical these days. Google Play Store is flooded with apps offering terrible user experience. Users give the app just a few minutes and if they don't understand it, they just uninstall or post a negative comment. Well-thought UI and interaction can really make a difference and make your users love your app.

I used these principles while designing Settle Up:

- Be simple for novice users, hide everything advanced under menu
  - For example usage of a dashboard pattern[3] for the main screen. Novice users see clearly that the app has four major functions: New Payment, Who Should Pay, Payments Log and Settle Debts. They will discover everything else later.
- Do things automatically, but let user change it.
  - Defaults which make sense are important. I don't ask user "Do you want to enable automatic synchronization?" I just enable it every hour, but user can change it in settings. Every pop-up like that might look like some error and users automatically press No/Cancel.
- Make first start as quick as possible.
  - First impression is important - it needs to be quick and clear. No annoying pop-ups like pop-up about change-log or pop-up about paid version. In Settle Up, users need to create a group first - I make it super easy with auto-complete from contacts - they just need to type a few letters from person's name and that is it. It also saves each person's e-mail for later sharing.
- Follow guidelines, don't invent own concepts.
  - Android is very open and developer has a free hand in designing UI. That can result in many errors. For example many developers just take design from iOS and duplicate it on Android. iOS and Android have different interaction patterns and the resulting app is confusing. Google has published a great Android Design[4] guidelines, all developers should read it. In Settle Up, an example would be usage of ActionBar pattern[5] for the top bar.

2.4 Algorithm for settling debts

Algorithm for settling debts is a core part of Settle Up. It is hard to calculate transactions between members by hand for multiple members and Settle Up makes it easy. The problem might look trivial, but it is actually a non-deterministic polynomial(NP) problem.

2.4.1 Basic algorithm

I was researching this problem and I got inspired by some StackOverflow questions[6][7][8]. First you need to forget about payments - the only numbers you need are balances of members. Member's balance is positive if money should be received and negative if owed. Payments are used only to calculate balances. There is a pretty good polynomial algorithm for solving this:

Sort members from based on their balance. The "richest" member pairs with the "poorest" and the transaction is made between them. Always one or both ends with zero balance = can be left out from further calculations. The algoritm is recursively launched on the subset of members. When all debts are settled, the algorithm ends. The algorithm will produce in the worst case N-1 transactions (N is number of members). It also minimizes total amount of money transferred, as opposed to other approaches (like chaining members without sorting).

Some people on StackOverflow were claiming that this is an optimal solution. But I found a counter-example:

Name	Balance
Alice	+$8
Bob	-$4
Charlie	-$5
David	+$7
Eve	-$6
Filip	+$3
Gary	-$3

Algorithm would solve it like this:

From	How much	To
Eve	$6	Alice
Charlie	$5	David
Bob	$3	Filip
Gary	$2	David
Bob	$1	Alice
Gary	$1	Alice

That is six transactions, which is N-1. But the problem can be solved with less transactions:

From	How much	To
Gary	$3	Filip
Eve	$6	Alice
Charlie	$5	David
Bob	$2	Alice
Bob	$2	David

2.4.2 Optimal algorithm

The basic algorithm is not optimal. I observed that it is not optimal in cases when you find subgroups of members which can settle debts between them (total balance is 0). When a subgroup is found, we can run basic algorithm on it and save one transaction. So we need to find these subgroups programatically.

Finding subgroups with sum of 0 is Subset Sum Problem[9]. It is NP-complete, therefore whole algorithm is NP-complete. There are some faster approaches to Subset Sum problem, but in my case groups will not have too many members. Trying 2-combinations, then 3-combinations etc. will be good enough.

2.4.3 Transaction tolerance

This algorithm can be easily extended with a transaction tolerance - usually people don't care about settling debts of small value (like less than $1). With a tolerance set, algorithm can produce even less required transactions. Modification is easy - we will not look for groups where a total balance is 0, but when the total balance is less than the tolerance. Also members with the balance less than the tolerance are removed from further calculations.

2.5 Synchronization

Synchronization(sync) is a major feature of the app, it helps the app stand out from competition. Ability to input payments to multiple phones in a group is very convenient. Synchronization also deals with problem of automatic backup in case you loose your phone. But it is not an easy task to design it. I wanted to follow standards and be open as much as possible.

2.5.1 Synchronization principles

- The app needs to be functional offline.
  - Reason: Sometimes there is no connection.
- No own user accounts, use Google or Facebook authentication.
  - Reason: Creating account on mobile device is painful and discourages users.
- Be open and use the public API for myself.
  - Reason: I cannot cover all platforms, let third-party developers to do it.
- Server will store only changes, not a current state.
  - Reason: Avoids errors with inconsistent states of data, nothing can be lost, even deletion is just a change.
- Use standards for modern APIs - REST and JSON as a format.
  - Reason: It is compact and easy to read, developers are used to it.

Chapter 3

Implementation

Implementation started as a quick prototype of offline-only Android app. But it grew over time with open API and synchronization to various platforms. This chapter describes the most interesting bits from implementation.

3.1 Overall architecture

Figure 3.1 shows a diagram of overall architecture. I have chosen AppEngine for the server-side (see reasons why in subsection 3.3.1). AppEngine server hosts a static project website and dynamic web interface which communicates with NoSQL datastore. It also contains RESTful API which communicates with the datastore and individual clients. One of the clients is the original Android version with own SQLite database, therefore it can operate completely offline until connection is available.

3.2 Algorithm for settling debts in Java

The algorithm is described in section 2.4. First I needed combination generator, I used available implementation in Java[10]. It uses Knuth's algorithm for generating combinations[11]. Full source code of the algorithm is available in Appendix B.

3.2.1 Runtime experiments

Because of exponential growth, I had to guarantee reasonable runtime. I have run the algorithm on Nexus One (1 GHz processor) with these results:

Members	Runtime
5	3 ms
10	82 ms
15	953 ms
18	8 s
20	55 s

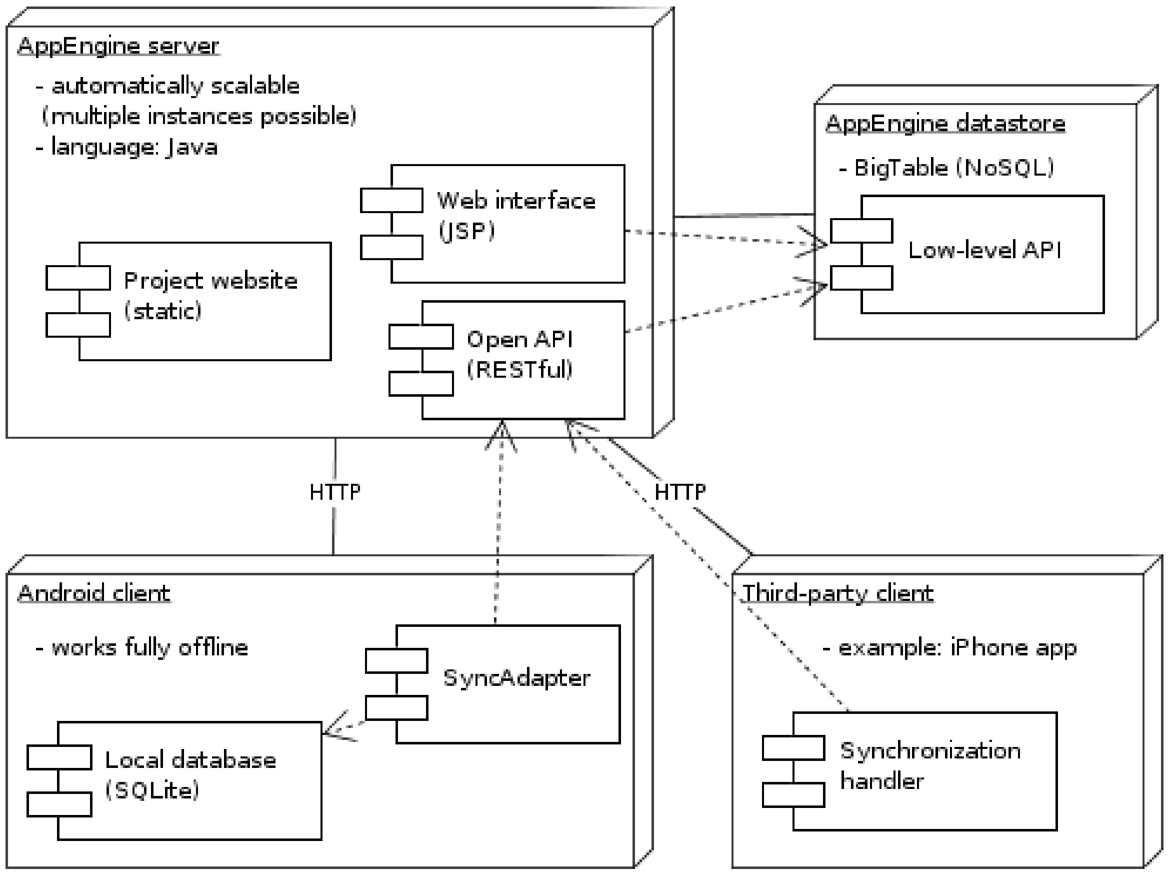


Figure 3.1: Deployment diagram of the whole Settle Up project

The reasonable runtime is for groups with less than 15 members. Therefore I use optimal algorithm for less than 15 members and basic algorithm for more members.

3.3 Synchronization API on AppEngine

Synchronization was and still is the most tricky part of the whole project. It took me a while to design it and I had to modify it several times after first release. But reliability of sync is very important - sensitive financial data is synchronized, people can lose money and stop trusting application as a whole. I have used principles from section 2.5 during development. Full documentation for third-party developer is available on the project website[12].

3.3.1 AppEngine

I chose to host the server on AppEngine[13] mainly because it has unlimited scalability. If my app suddenly becomes very popular, AppEngine will handle it. It is free until your project is successful - I didn't have to pay until December 2011 (7 months after launch). It is very easy to setup and deploy - just one button click from Eclipse. It supports Java, which is great for Android developers - same data objects can be used both on server and client. It is little bit tricky to get used to their scalable datastore - there are no joins between tables and other advanced database concepts. But it is possible to save 1 GB of data for free every day. I don't like weekly minimal charge - my server costs just a few cents a day, but I have to pay $2.10 a week (more about expenses in section 6.1).

3.3.2 API capabilities

API can be used for full access to user's groups. Android app doesn't use anything more than public API. API can be used for creating compatible apps on other plaforms such as iOS, Windows Phone or Blackberry or as a Chrome extension, Excel export etc. API has the following features:

- Create group
- List user's group
- Synchronization - list of changes in payments or members since last synchronization
- Grant permission to the group
- Remove permission to the group
- Delete group

3.3.3 Tips for API developers

- One of the most challenging parts was about identificators(IDs) of the groups, members and payments. Should the server generate them? Should client use own IDs? What if user uses app 2 months offline and then turns on sync? The best solution I found was to generate IDs locally as `hash(local_id + installation_id)` where installation ID is randomly generated during a first start. That ensures uniqueness across various platforms and installations. Same ID for members and payments in different groups actually doesn't matter. But for group IDs, the server must check for uniqueness and handle (unlikely but possible) collisions.
- Another challenging part was conflict handling - what if two people edit the same payment? I believe that asking users to resolve conflicts like that would be annoying - synchronization is running in the background without user interaction. So I am using timestamp from client to resolve conflicts automatically - user who edited payment last wins.
- REST principles proved valuable - it is easy to read and understand the API. For example HTTP error codes are great way for reporting errors. Both server and client frameworks handle them well.
- Authentication is always hard to do. First I used Google Accounts only, since every Android phone has a Google Account. But iPhone developers were correct that it might discourage some users on other platforms - so I added login via Facebook. I also implemented verification of additional e-mails with your primary (Facebook or

Google) account. User can specify own e-mails and groups shared with those e-mails are automatically shared with the primary account(after verification).

3.4 Tips for Android developers

I have learned a lot during Settle Up development. I would like to share most interesting recommendations:

- Use library ActionBarSherlock[14] for implementing ActionBar pattern[5] and tabs. It is a great library with the same API as native components. It uses native components on newest versions of OS and emulates them from Android 2.0.
- Android Asset Studio[15] is a great tool for creating graphics like icons. It is very easy to use and it follows guidelines. You don't need a person for graphics in the beginning.
- Use ContentProviders[16] for managing your local data. It might look complicated at first, but it nicely separates your data and logic. It solves problems with two threads accessing the database at the same time.
- If you decide to use Dashboard pattern for the main screen, use code from Google IO 2011 app[17].
- Always load data from database in a background thread. AsyncTask[18] is a nice way to do that. It makes your UI responsive and fast.
- Save some data needed by multiple Activities in a class which extends class Application. Load them in method onCreate - it is loaded always when your first activity starts.
- Use Intents as much as possible. Intents are one of the best features of Android. Let other apps handle functionality for you. Examples: scanning QR codes, sending e-mail, rating in Play Store, opening website, picking image from camera/gallery, ...
- Separate handling of network connections from Activities. Create abstract activity for that and all activities that use network should inherit from it. You can handle tasks like authentication, common network-related errors etc. at one place.
- Use swipe gestures for swiping between lists - users are familiar with it and expect it.
- Switching of languages is tricky. Handle it in your parent activity in onCreate method. Compare current and saved Locale and switch it if needed.
- Use official SyncAdapter[19] for background sync. Don't sync in own service, it just drains battery and users cannot control it.
- GSON library[20] is a nice library for handling JSON data. It transforms JSON into Java objects and vice versa with ease.

3.5 Evolution of the user interface

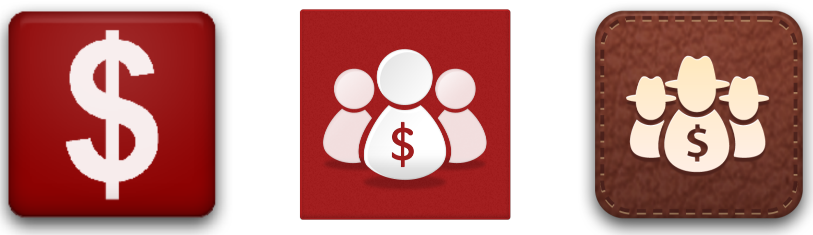
Graphic design is an important part of your app because it bonds users with your app emotionally. However I don't think you need a graphic designer from the very beginning. You can create simple yet effective first design just by following Android guidelines[4]. When your app gains initial traction, you should think about redesign with the help of a graphic designer. In Settle Up, I made 4 big graphic updates:

- April 2011 - initial graphics with royalty-free icons[21]
- May 2011 - new icon and tweeks with help of a graphic designer[22]
- November 2011 - major redesign with help of company Bioport[23], author of the iPhone version
- February 2012 - ActionBar redesign to match with Android 4.0 design

3.5.1 Screenshots from UI evolution

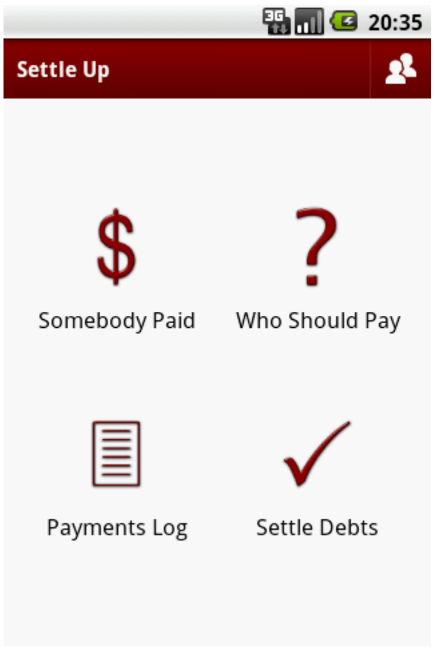
Cooperation with iPhone developers had a positive impact on UI evolution. See how iPhone version looks in Figure 4.3. iPhone developers allowed me to use their graphics and even created some Android-specific elements for me.

Main app icon is an important part of graphic design. See its evolution in Figure 3.2. While redesigning main screen, I focused on keeping simplicity and main usage patterns - see Figure 3.3. New payment screen had to handle new payment options like uneven split or attaching picture of the receipt. I tried to be minimalistic - see Figure 3.4. Payments Log screen didn't change much - see Figure 3.5. Settle Debts screen is now more clear and usable - see Figure 3.6.

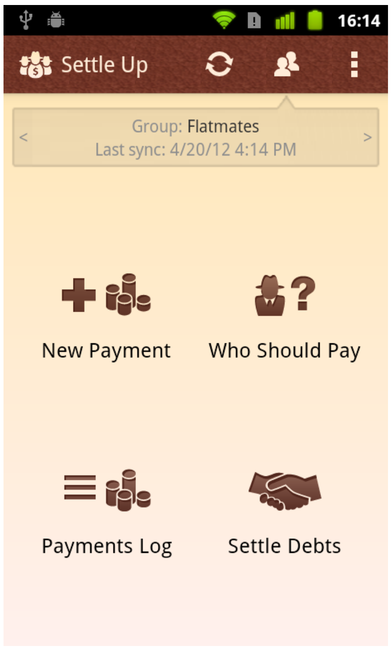


(a) April 2011 (b) May 2011 (c) November 2011

Figure 3.2: Evolution of the app icon

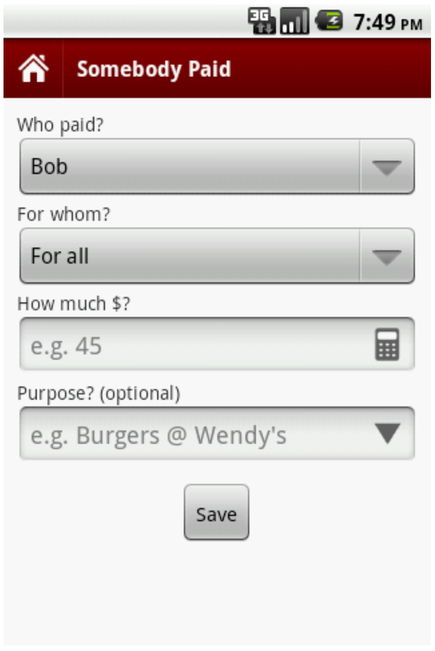


(a) May 2011



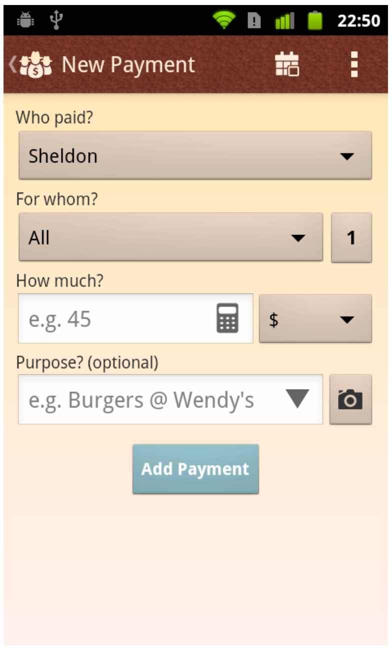
(b) May 2012

Figure 3.3: Evolution of the main screen



Who paid?
Bob
For whom?
For all
How much $?
e.g. 45
Purpose? (optional)
e.g. Burgers @ Wendy's

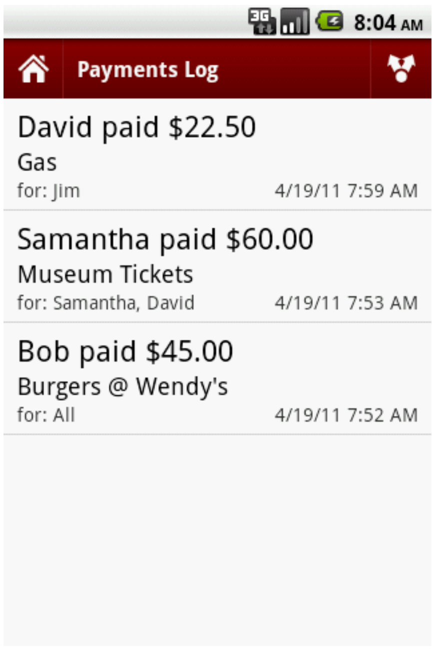
(a) May 2011



Who paid?
Sheldon
For whom?
All 1
How much?
e.g. 45 $
Purpose? (optional)
e.g. Burgers @ Wendy's

(b) May 2012

Figure 3.4: Evolution of the New Payment screen


8:04 AM

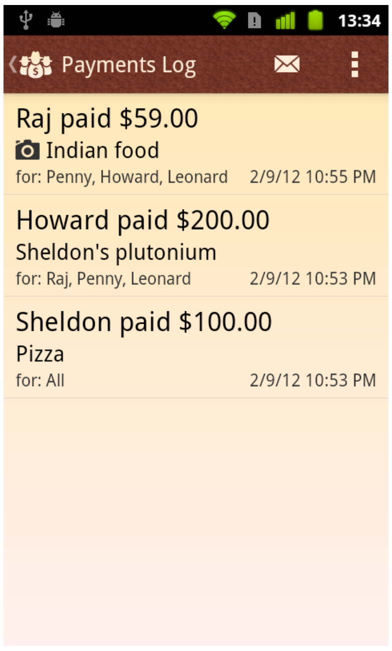
Payments Log

David paid $22.50
Gas
for: Jim 4/19/11 7:59 AM

Samantha paid $60.00
Museum Tickets
for: Samantha, David 4/19/11 7:53 AM

Bob paid $45.00
Burgers @ Wendy's
for: All 4/19/11 7:52 AM

(a) May 2011


13:34

Payments Log

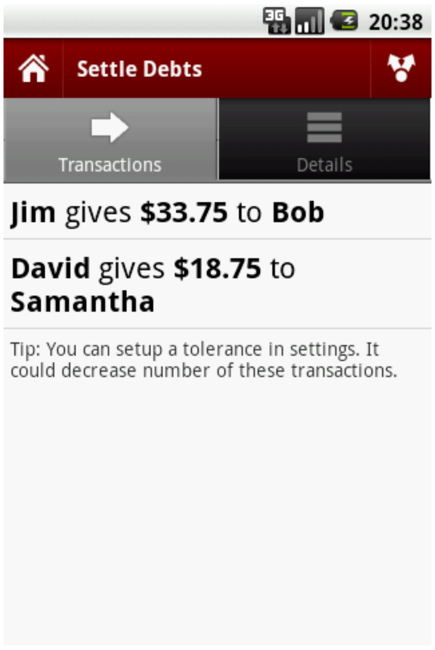
Raj paid $59.00
Indian food
for: Penny, Howard, Leonard 2/9/12 10:55 PM

Howard paid $200.00
Sheldon's plutonium
for: Raj, Penny, Leonard 2/9/12 10:53 PM

Sheldon paid $100.00
Pizza
for: All 2/9/12 10:53 PM

(b) May 2012

Figure 3.5: Evolution of the Payments Log screen


20:38

Settle Debts

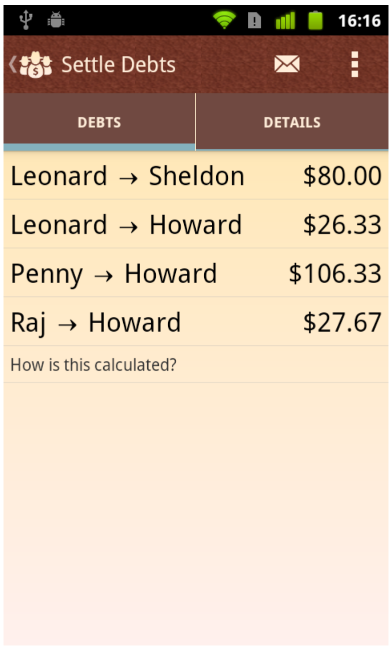
Transactions | Details

Jim gives **$33.75** to **Bob**

David gives **$18.75** to **Samantha**

Tip: You can setup a tolerance in settings. It could decrease number of these transactions.

(a) May 2011


16:16

Settle Debts

DEBTS | DETAILS

Leonard → Sheldon	$80.00
Leonard → Howard	$26.33
Penny → Howard	$106.33
Raj → Howard	$27.67

How is this calculated?

(b) May 2012

Figure 3.6: Evolution of the Settle Debts screen

Chapter 4

Deployment

Implementation is not the hard part. The most time-consuming work starts after the release. This chapter describes various tasks I had to perform for a year following release.

4.1 Google Play Store

Google Play Store (formerly Android Market) is a distribution channel for all Android apps. Access to Developer Console[24] costs one-time $25. Then anyone can publish apps without limitations.

4.1.1 Publishing an app

This is what you need for publishing an app:

- APK file signed by your certificate. There is tool for that in Eclipse.
- Few screenshots of the app. You can capture them using Android SDK or newer phones capture screenshots by pressing power and volume down buttons.
- High resolution icon - use Android Asset Studio[15] to generate.
- Feature graphic (optional) - a big banner promoting your app. You will probably need somebody with graphic skills for this. Don't add too much detail, it should scale to small sizes.
- Description - I suggest short pitch in the beginning, then more detailed list of features. It is good to explain why you need permissions here. You can add some tags to improve visibility in search.
- Website - you should have some basic micro-site, make sure it is mobile friendly.
- E-mail - subsubsection 4.3.2.1 talks about that.

4.1.2 Tips

- Check errors section regularly - you can see reported crashes with whole Java stack - useful for debugging. Encourage your users to report crashes and post their e-mail address into description - then you can contact them about particular bug.
- Read comments regularly - only Developer Console shows all the comments (public Google Play shows comments only in your language). Comments are a good indicator of problems and what users miss the most. Unfortunately you cannot contact people who post comments.
- Translate description to other languages - it has a positive effect on your position in search.
- Create YouTube video about the app and add it there - people watch it.
- Don't forget to fill Recent changes after each release - people read it.
- Watch your statistics(stats) and correlate with promotions you have done. Identify spikes in the user count and find out the reason. It is also useful to see how quickly users update (for Settle Up 33% of users never update).
- Install app Andlytics[25] to your phone - it notifies you when there is a new comment and gives you access to the stats on the go.

4.2 Competition

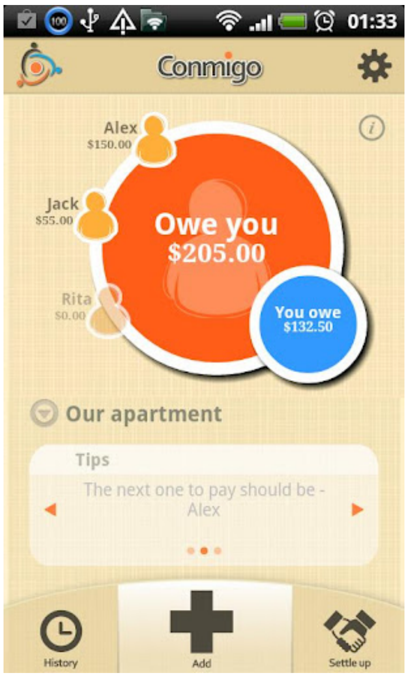
Watching competition is important. It is common that there will be several similar apps like yours in Google Play. You should discover competition before you start coding - maybe there is a superior one and it will not be worthwhile to compete with them. But it is not usually the case - there are lot of apps, but very few high quality ones. I have found about 15 apps which have very similar functionality to Settle Up. Most of them are no longer developed or have a terrible UI. I consider these two apps my biggest competition:

4.2.1 Conmigo

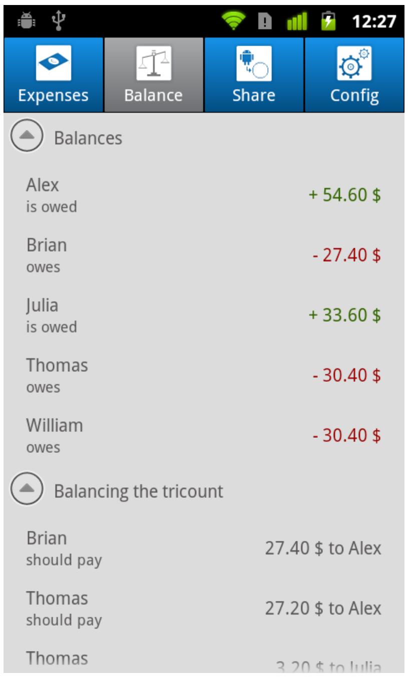
Conmigo[26] was launched after Settle Up and I suspect they were heavily influenced by Settle Up. They even call their "settle debts" function "settle up".

4.2.1.1 Where is Conmigo better

- Visual design is nice and clean. See Figure 4.1. I was afraid that this might be a deal breaker and redesigned Settle Up in November 2011.
- They have a first-time tutorial and lots of small tutorials explaining functions and gestures. Maybe I should work on "first time wizard" as well.



(a) Conmigo



12:27

Expenses | Balance | Share | Config

Balances

Alex is owed	+ 54.60 $
Brian owes	- 27.40 $
Julia is owed	+ 33.60 $
Thomas owes	- 30.40 $
William owes	- 30.40 $

Balancing the tricount

Brian should pay	27.40 $ to Alex
Thomas should pay	27.20 $ to Alex
Thomas	3.20 $ to Julia

(b) Tricount

Figure 4.1: Screenshots from biggest competition of Settle Up

4.2.1.2 Where is Settle Up better

- They promise sync, but they are unable to deliver it for 6 months. Their development is slower.
- They don't have an iPhone version and web version, just promises.

4.2.2 Tricount

Tricount[27] started even before Settle Up as a web service. Now they have Android and iPhone app so I follow them more closely.

4.2.2.1 Where is Tricount better

- This app is the only app with sync. They are more focused on the website, Settle Up has just a basic web interface.
- They have a "sample group" which shows the app with some meaningful data. I like that approach.

4.2.2.2 Where is Settle Up better

- UI of their Android version is not very nice and it doesn't follow Android guidelines.

- They are too focused on French users - even screenshots on Google Play are in the French language.

4.3 Feedback from users

Reacting to user feedback is a crucial part of the after-release process. Users will tell you things you would never imagine and help you understand your app better. You should make giving feedback easy. You should not rely only on one communication channel - give users a choice, because every user is different and uses different tools.

4.3.1 Face to face feedback

I released the app on the day of an event "Beer with Google". Lot of geeks and early adopters were there, ideal opportunity to get some early feedback. I walked from person to person asking them to download the app and play with it. Feedback was very valuable. Mobile apps have advantage of always being in your pocket. I show the app regularly to my friends and acquaintances and ask for realtime feedback. Here are some examples of improvements which came from realtime feedback:

- 'Who Should Pay' should also show the reason (balance of that user).
- Save buttons should not be labeled 'Save' but more concrete like 'Create Group'.
- People were unsure about suggested transactions - so I added 'How was this calculated?' help text.
- My friend was entering multiple beers as an expense. It was hard for him to calculate 'how much 5 beer cost'. He said that it would be great if the app had a little calculator. So I added a calculator right in the New Payment screen.

4.3.2 Problem reports

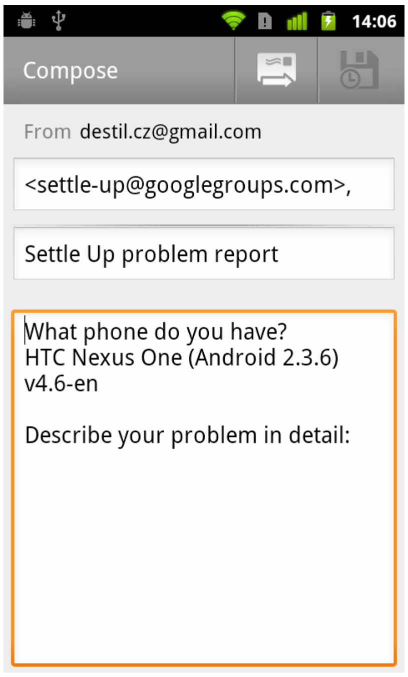
Fixing bugs should always be an absolute priority. Users could live without some functionality, but if the app crashes they will just uninstall and write a negative comment on Google Play. Android itself is becoming fragmented and you cannot anticipate that it will work on all phones and configuration. Bugs will happen and you need to handle them correctly. Google Play records app crashes, as described in subsection 4.1.2.

4.3.2.1 Google Groups

You should create some kind of mailing list with a web interface for problem reports. I use Google Groups[28] for this purpose. You need to provide some e-mail address to your Google Play listing. I suggest using yourapp@googlegroups.com. The biggest advantage is that all problem reports are logged and accessible on the website. Anyone who has the same problem can do a Google search and figure out the answer without bothering you. While setting Google Groups, make sure you set it than anyone can write to the group and that replies go to the author of the message. Users don't have to use Google Groups interface at all. They can use e-mail like they are used to.

4.3.2.2 Reports from the app

Give your users an easy way to write problem reports when they need it most - directly from the app. Good place is in the About screen. I was surprised how many users are using this option. It is launched from your code, so you can customize the e-mail a little and save yourself one useless message asking them what phone and app version they have. See how it looks like in Settle Up in Figure 4.2.



From destil.cz@gmail.com

<settle-up@googlegroups.com>,

Settle Up problem report

What phone do you have?
HTC Nexus One (Android 2.3.6)
v4.6-en

Describe your problem in detail:

Figure 4.2: Problem report e-mail generated by the app

4.3.3 Feature requests

You will receive a lot of feature requests. But every new feature makes the app more complex and harder to understand for new users. You need some way of sorting features to implement just the most wanted ones.

4.3.3.1 GetSatisfaction

Google Group is not a good place for feature requests. They will get lost among other messages. I use an online tool GetSatisfaction[29] and I am satisfied. It has very intuitive interface. People suggest features and I can manage them by tagging them ’not planned’, ’under consideration’, ’implemented’ etc. Others can vote and comment on the features. Only downside is the cost - $19 per month. Here are some features I implemented based on feedback from GetSatisfaction:

- Web interface (21 votes)
- Homescreen widget for New Payment (11 votes)
- Take picture of the receipt (10 votes)
- Uneven payment split (9 votes)
- Filter Payments Log (9 votes)
- Backup to SD card (8 votes)

And these are top feature requests I am considering now:

- Recurring payments (12 votes)
- Feed of changes in the group (9 votes)
- Managing payments in the web interface (5 votes)

4.3.3.2 Google Play comments

Not everyone is willing to find and use GetSatisfaction. Comments are also a valuable source of user feedback. I have done an analysis of comments and discovered top feature requests from comments:

- Improve currency handling (5 people)
- Feed of changes in the group (1 person)
- Single email export for both payments and debts (1 person)
- Add things, not just money (1 person)

GetSatisfaction is not always covering all the needs of users. Feature request 'Improve currency handing' doesn't have much votes there, but it is highly mentioned in Google Play comments.

4.3.4 Expert advice

It is very valuable to ask professionals with a lot of Android development experience for an opinion. Android Developer Advocates from Google[30] organize a "Friday Review" every Friday. It is quite easy to get your app reviewed. Everyone can vote on the suggestions so you just need a few friends who will vote for you. Settle Up was reviewed on May 4, 2012. Whole review is recorded on the video[31]. I received the following valuable advice:

- They generally think the app is useful and the design is clear.
- I should use standard Ice Cream Sandwich (ICS) form elements.
- I should make the advertisement (ad) bigger on tablets.
- In Payments Log, a click should go directly to the editing, loose edit/delete pop-up.
- I should consider using In-app Billing for ad-free version.
- The dashboard pattern is not a best bet, mainly because it does not look good on tablets. I am already thinking about a major redesign with tablets in mind.

4.3.5 Communication tips

I recommend following these tips regarding to communication with users:

4.3.5.1 Always reply, but know what to refuse

It is very important to build users' trust. A great way is to always reply to their feedback. For most users, it is easier just to uninstall the app. You should respect the time of those who contacted you. It doesn't take much time if you label all the related e-mails and reply to them once a day quickly. Users will not have a feeling that they are writing into a 'black hole' and they will more likely recommend the app to their friends. Google Play comments benefit greatly from this approach. A typical scenario: User: "I have some missing payments in synchronization", me: "What did you do? Can you reproduce it?". That user may not bother to investigate further. But he will not post a negative comment into Google Play. Replying always saved me a lot of negative comments.

But it is also important NOT to implement everything users want. Your app cannot serve all groups of users, otherwise it would became too complicated and it would serve nobody. You should be clear about your target group and politely decline feature requests which are outside your app's focus. Examples of features I declined:

- Record what you borrow/lend to others (things, not expenses)
- Expense tracking for individuals, not only groups
- Allow commenting on payments (a little social network)

4.3.5.2 Be a number one user

I have seen many projects where developers or managers were not using the product they were building and it failed. Using it yourself gives you motivation and it makes the app better for everyone. Even big companies like Google use this approach - they call it 'dogfooding'. Using it in real-time situations brings a lot of clarity into what users want and it can persuade friends around you to use it as well and give you valuable feedback. By using it, you will quickly notice any important bugs and it will make you fix those little things in design and usability. You will see little flaws every day and it will bother you - just like it bothers users. Here are some features I implemented as result of using it in a real life:

- Ice Cream Sandwich design - I was one of the first who had a Galaxy Nexus phone with ICS and Settle Up felt old compared to system apps.
- Button for making one-to-one payments with one click.
- When all debts were settled, there was a debt of $0.01 because of real number accuracy. I have fixed that.

4.3.5.3 Value active users

You should always appreciate users who give you some feedback. Majority of people don't bother. They will just uninstall if they find a problem. I took it to the next level in April 2012. I started giving the paid version to the most active users for free.

In the 'Remove Ads' dialog, anyone can click 'I Want It For Free' and send me an e-mail with their reasons. I list following plausible reasons:

- Did you spread a good word about the app on your blog/social networks?
- Did you help with translations?
- Did you help me to hunt down some tricky bug?
- Do you have some other strong reason?

After two weeks of this program, I granted paid version to 13 people. Most of them are friends who give me feedback on regular basis or most active translators. Translators were really grateful and activity on the translations increased. One user wrote me that he could help with my English, some wording wasn't exactly how a native speaker would say it. So he made a tiny changes across whole language file and it makes a difference. There were several reports about missing payments in sync, but I wasn't able to reproduce it. I always asked users for more testing and they didn't answer back. But now with a promise of paid version, one user finally helped me to debug it and I have fixed the important bug. I wish I had started this initiative sooner. It is really beneficial.

4.4 Localization

In today's global market, it is important to think locally. Localization is not hard, crowd-sourced translations are cheap and offer a good quality. Localization could bring you vast amounts of users from non-English markets.

4.4.1 GetLocalization

I am using a service GetLocalization[32] for all Settle Up translations. I am very satisfied with the service. It is free and it brought me many translated languages. I am encouraging users to translate from Settle Up website[33]. I even use the service myself for Czech translations (my mother-tongue). It supports Android string resources and other formats like Play Store text and website texts. Translators can interact with each other and me, vote on translations and remove bad ones. The community is active and nobody has complained about quality of the translations.

4.4.2 Translated languages

By May 2011, Settle Up has been translated into the following 18 languages: English, Czech, German, Spanish (Spain), Spanish (Latin America), French, Turkish, Catalan, Portuguese (Portugal), Portuguese (Brazil), Chinese (China), Dutch, Dutch (Belgium), Danish, Swedish, Polish, Slovak and Russian.

Effects of the localization can be seen in Play Store statistics - Table 4.1.

Table 4.1: Distribution of languages among Settle Up active users, April 20, 2012

Language	Users	%
Czech	5065	36 %
Spanish (Spain)	3313	23.5 %
English (United States)	1780	12.6 %
English (United Kingdom)	802	5.7 %
German	452	3.2 %
Spanish (United States)	340	2.4 %
French	302	2.2 %
Italian	177	1.2 %
Slovak	165	1.1 %
Others	1675	11.9 %

4.5 Cooperation

Although I am a supporter of the 'solopreneurship' idea, I cannot do everything. It is not in my power to develop clients for every mobile platform. Open API allows me to independently cooperate with third-parties on this. API documentation[12] specifies the terms of that cooperation:

- Third-parties need to request API key by contacting me and briefly describing their intentions.
- I am not responsible for any server outages or even service shutdown.
- We can cooperate on branding, graphics and marketing.

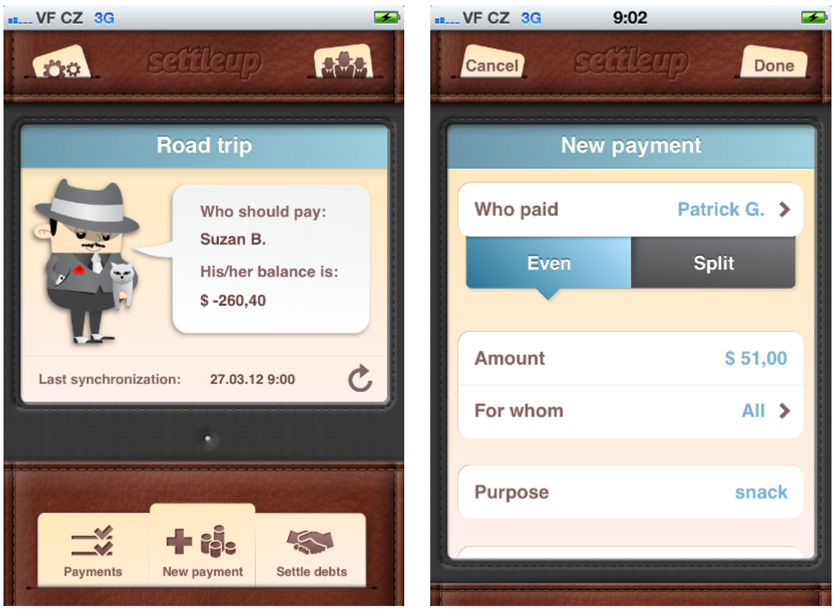
4.5.1 iPhone version

So far 7 developers were granted API access for different purposes like custom reports or intranet integration. But the most successful cooperation is with company Bioport Software Labs[23] on development of a compatible iPhone version. Our cooperation is mutually beneficial:

- Advantages for me:
  - My users can synchronize with friends on iPhone.
  - Bioport helped me with new graphics and redesign.
  - Bioport redesigned the project website.
  - Bioport launched marketing efforts for the whole project increasing my user base and revenue from Android app.
  - They gave me a lot of feedback.
- Advantages for Bioport:
  - They can use the same name, UI flow etc.

- They save money on developing own synchronization, no need to maintain server side.
- Their users can synchronize with friends on Android.
- They have independent revenue from iPhone app.
- We share GetLocalization account, Android users translate also to the iPhone version.

The iPhone version doesn't have all features of the Android version, but synchronization is fully compatible. I admire the visual design of the iPhone version - see Figure 4.3. They also made an amazing video promoting iPhone app and the whole project[34].



(a) Main screen (b) New payment screen

Figure 4.3: Screenshots from the iPhone version developed by Bioport Software Labs

4.6 Marketing and user growth

Google Play Store is full of applications of various quality. Apps need to have some marketing efforts, otherwise they will not get noticed by users.

4.6.1 Social networks

Social networks are important for every new project. They allow me to be in a direct contact with the users. Big news about the app can spread virally. My activities with social networks:

- I post updates about the app to Facebook[35], Twitter[36] and Google+[37].

- I reply to all comments and encourage people to share.
- I monitor mentions about the app and reply to them.
- I monitor the web using Google Alerts[38].

4.6.1.1 Viral experiment

Social networks are great for viral marketing. We did an experiment with a company Bioport after the review at MobilMania.cz and introduction of a new promotion video on April 24, 2012. We asked our friends to start sharing our content at the same time. That started a wave and other people joined the sharing. Results of the experiment:

- Google+: 46 reshares, 52 +1s, our post was no.1 in "What's hot in the Czech Republic" for a few hours.
- Facebook: 26 shares, 21 likes
- Twitter: 12 retweets, 2 favorites
- It had a great impact on downloads - 310 new downloads of Android version for that day.

4.6.2 Contests and awards

App contests are a great way to get noticed. I applied for a few of them in the beginning and I think it is the main reason for the initial traction.

4.6.2.1 AppParade

AppParade is a app contest for Czech developers held every 6 months in Prague. This one happened on May 10, 2011. I had to create a quick pitch[39] and I was surprised by being ranked third and winning special price from Vodafone CZ[40]. Vodafone CZ promoted my app for free on their web, magazine and social networks[41]. AppParade was also really valuable for connections. I met Vladislav Skoumal from Bioport Software Labs and we discussed iPhone version for the first time.

4.6.2.2 aDevCamp

aDevCamp[42] was a developer conference held in Prague on May 21, 2011. An app contest was a part of the conference. I won first prize - Samsung Galaxy Tab. I was surprised again - the app wasn't too mature at that time. My advantage was that the app was simple and the idea easy to understand.

4.6.2.3 Paypal X Developer Challenge

Paypal announced an Android app contest on March 7, 2011[43]. Their goal was to promote use of Paypal mobile APIs. Settle Up was an ideal candidate for this integration. I have added PayPal support and applied. I didn't win anything, but integrating PayPal payments was useful.

4.6.2.4 Křišťálová Lupa

Křišťálová Lupa is an annual contest for Czech web services. This year they announced a new category for mobile solutions. Settle Up was nominated into the top 10 mobile apps which were subject to vote. This nomination had a positive impact on new users. Settle Up finished at 9th place in the contest[44].

4.6.3 Media coverage

It is important to promote your app on various websites and blogs. Most websites about Android apps have a form where you can recommend an app. Sometimes you can pay for a review. I have tried both options.

4.6.3.1 April 20 - July 19, 2011

First three months were mainly about promotion in the Czech Republic. Most significant reviews were for SvetAndroida.cz[45] and Mobilmania.cz[46]. See graph of user growth in Figure 4.4.

4.6.3.2 July 20 - October 19, 2011

Over the summer the app got really popular in Spain. Two big Spanish servers El Android Libre[47] and XatakaAndroid[48] wrote reviews about the app. I tried to get more interest from English-speaking users by paying $27 for a review at AndroidAppLog[49]. I published the app into KatalogAplikaci.cz[50] - nice catalog of Czech Android apps. In the end Google Play Store changed statistics radically - apparently some updates were counted as active installations. Even data from Google itself doesn't have to be 100% accurate. See graph of user growth in Figure 4.5.

4.6.3.3 October 20, 2011 - January 19, 2012

The biggest user boost in autumn was from Vodafone CZ promotion. As AppParade winner, they promoted the app on 8th of December in "Velikonoční ošatka" campaign. Also iPhone version was released right before Christmas. See graph of user growth in Figure 4.6.

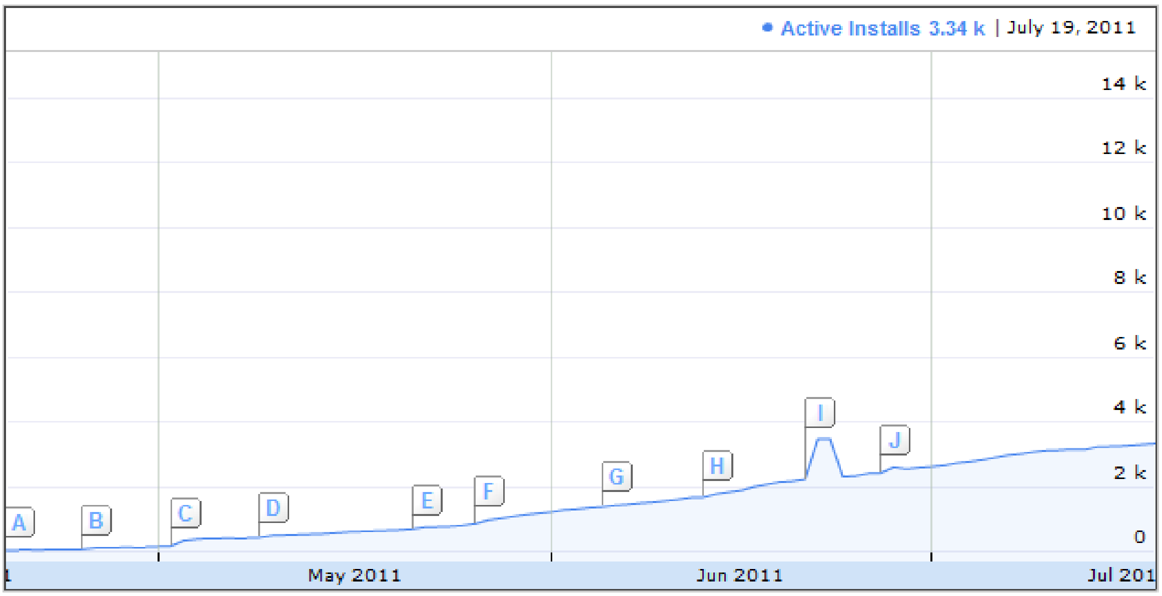


Figure 4.4: Active user growth from April 20 to July 19, 2011
A - Release at #gpivo, B - Twitter mention from @tomucha, C - Review at SvetAndroida.cz, D - Pitch & price at AppParade, E - Pitch & price at aDevCamp, F - Review at Dotekomania.cz & Mobilmania.cz, G - YouTube promo video, H - Review at MojAndroid.sk, I - Play Store stats error, J - Online sync released

4.6.3.4 January 20 - April 30, 2012

I did a lot of updates in the spring and saw an interesting trend. Some users uninstall after each update. It is not the case that the update makes the application worse, but user gets notified about the app and realizes no need for it anymore. I focused on propagation in English speaking countries. There was a notable review from AndroidZoom[51]. At the end of the month Bioport launched marketing efforts in the Czech Republic for both apps. A review at MobilMania.cz[52] and related viral campaign(subsubsection 4.6.1.1) brought me a lot of new users. See graph of user growth in Figure 4.7.

4.6.4 Advertising

Advertising might be a good way to promote the app. I was thinking that ads on the mobile device would be most efficient, because users can quickly install it. I had some experience with AdMob, so I launched an AdMob campaign. I have spent $50 - the minimum for starting a campaign. My goal is to gain popularity in the United States, so I targeted the ad only for United States. Here are the results:

- The campaign was running from April 25, 2012 to May 2, 2012.
- It resulted in 428,280 impressions, 1,584 clicks (0.37% Click-through rate (CTR)) and average $0.03 Cost per click (CPC).
- Installs from US were 6.6 users/day before campaign and 6.5 users/day during campaign.

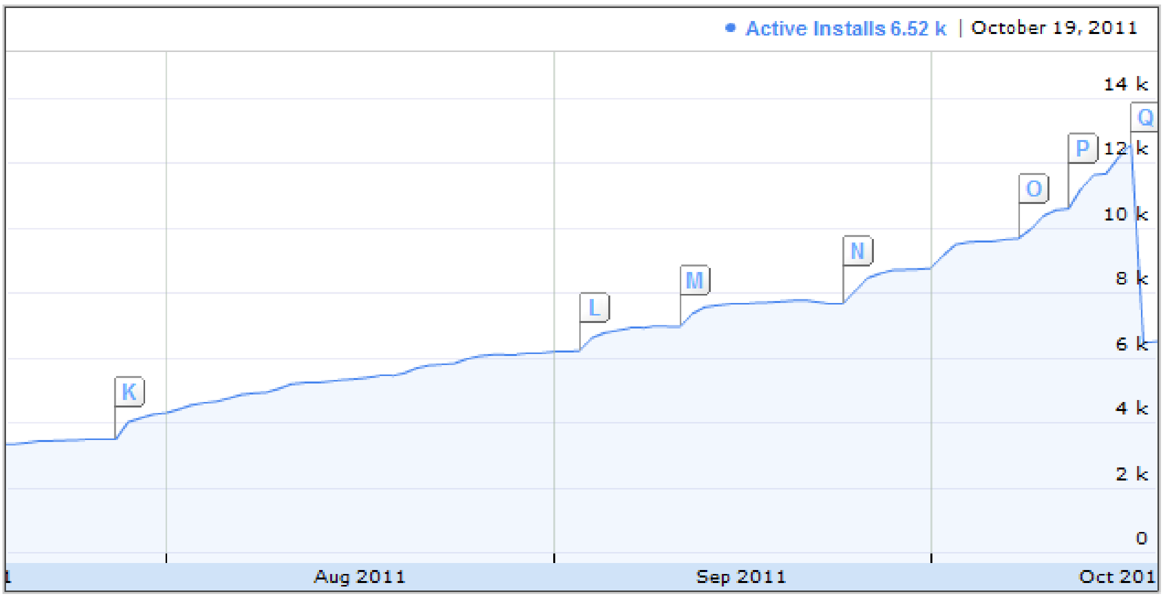


Figure 4.5: Active user growth from July 20 to October 19, 2011
K - review at El Android Libre, L - review at AndroidAppLog, M - Review at XatakaAndroid and Linuxlinks.com, N - Uneven split released, O - published to katalogaplikaci.cz, P - nominated to "Křišťálová Lupa", Q - Google Play Store correction

- Advertising was not worth it. I have lost $50.

4.6.5 Overall statistics

On May 4 2012 Settle Up had:

- 35,943 total installs
- 14,949 active device installs
- 14,293 active user installs
- Top 3 devices: Samsung Galaxy S2 (17%), Samsung Galaxy S (7%), Samsung Galaxy Ace (5%)
- Top 3 countries: Czech Republic (38%), Spain (24%), United States (8%)
- 70% of users have latest or second latest version, others almost never update.

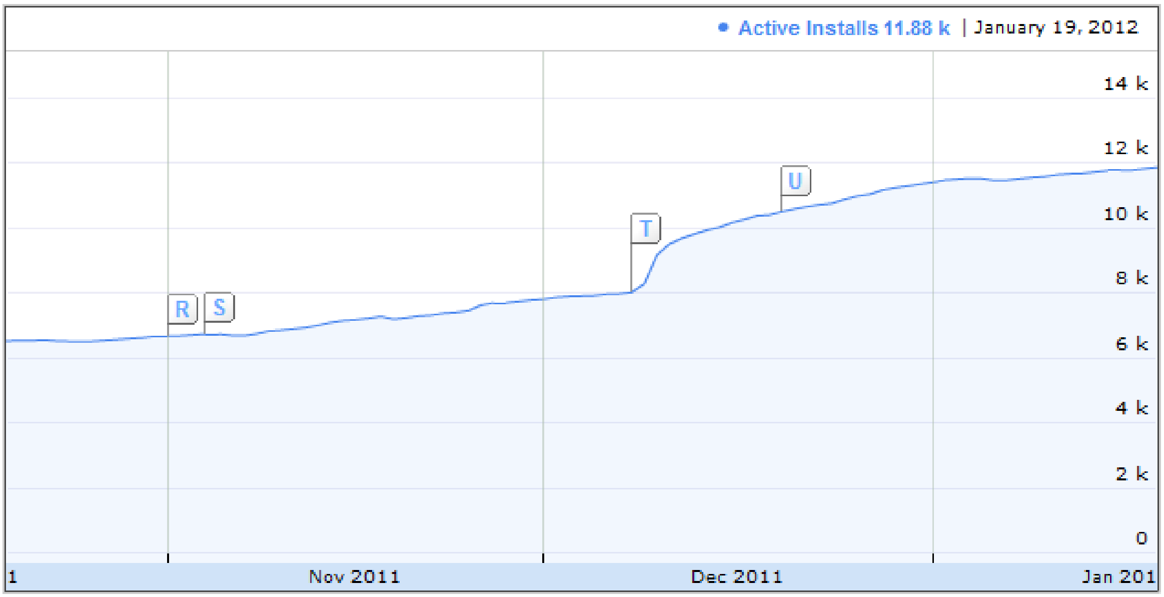


Figure 4.6: Active user growth from October 20, 2011 to January 19, 2012
R - UI redesign, S - featured on getlocalization.com, T - Vodafone CZ promotion, U - iPhone version released

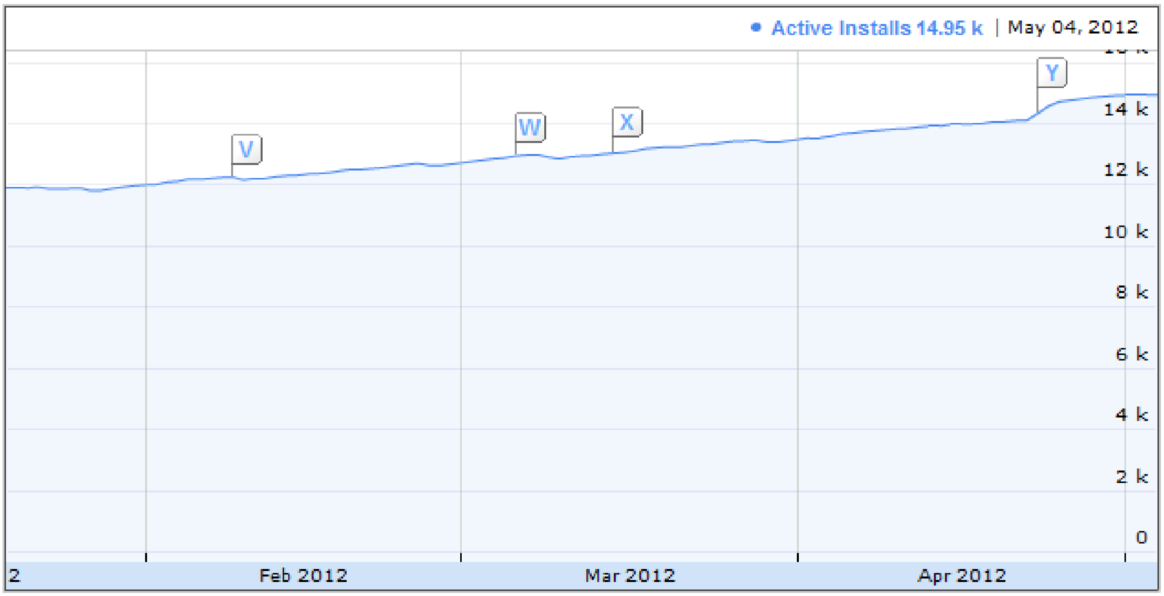


Figure 4.7: Active user growth from January 20, 2011 to April 30, 2012
V - ICS redesign, W - Web interface released, X - review at AndroidZoom, Y - review at MobilMania.cz

Chapter 5

Evaluation

Testing could take various forms. In Settle Up, I chose testing on a large scale by measuring behaviour of all users automatically. That allows me to take user experience improvements to the next level. There are many ways to perform remote evaluation. One would be to create own logs, send them to the server and analyze them. But I chose a more mature solution - Google Analytics for Mobile.

5.1 Google Analytics for Mobile

Google Analytics is a well known service for measuring website usage. But not many people know that it can also measure native mobile apps. In Settle Up, I use Google Analytics SDK for Android[53]. Websites and mobile apps are very similar, that is why it is possible to use the same concepts and full power of the service. You can track the following interaction types with the SDK:

5.1.1 Pageviews

Page in a mobile app is a screen - in Android's terms Activity. There is a library called EasyTracker[54] which makes integrating Analytics SDK and tracking pageviews of all Activities very easy.

5.1.2 Events

Event is an act of interaction on a single page like clicking a button. I am not tracking every click, but individual features of the app. This way I can see which features are more or less popular. You can record following data for every event:

- Category - Can be used for organizing events. I am using Activity name. Example: 'home'.
- Action - Contains an action name. I am using feature name. Example: 'who-should-pay'.

- Label(optional) - Some text related to the event. Example: For event 'change-currency' I track selected currency code. It allows me to measure popularity of currencies.
- Value(optional) - Contains some number related to the event. Analytics web interface shows only total of all values and average of all values. So it is not good for distinct values. Example: For event 'add-member', I track number of characters of the name. Then I can see average number of characters for a member.

5.1.3 Custom variables

You can filter stats based on many pre-defined criteria like time, location, language etc. Custom variables allows you to add more custom criteria for filtering. You can record the following data for every variable:

- Index - Number 1 to 5. You can specify only 5 variables.
- Name - Example: Variable 'paid' allows me to differentiate users with a paid and a free version.
- Value - String value for the variable, Example: 'true' and 'false'.
- Scope - One of the three options:
  - Visitor - set for one user for infinite time
  - Session - set only for one interaction with your app
  - Page - set for particular page in your app

5.1.4 Tips for developers

- EasyTracker[54] is a very useful library, but it doesn't support Events and Custom Variables. I suggest extending it.
- Don't dispatch events to the server immediately, it will drain the user's battery. I am using a batch dispatch once a minute.
- You should notify users that you are using tracking and give them a way to turn it off.

5.2 Analysis of Settle Up

I have measured user behaviour from February 27, 2012 to April 27, 2012. Every subsection is about one question I tried to answer with the measurements. It shows the data, interpretation and impact.

5.2.1 How is Settle Up used in time?

Google Analytics display pageviews over time nicely. See Figure 5.1.

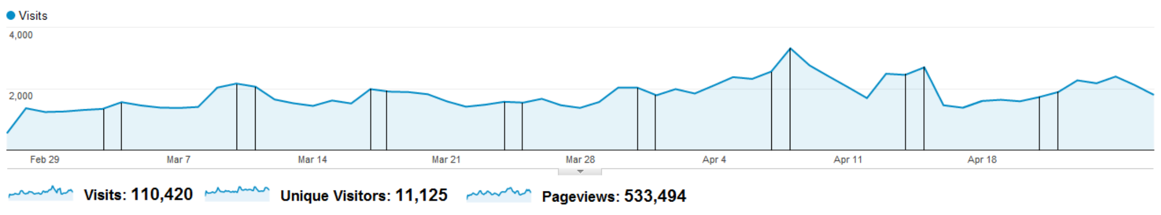


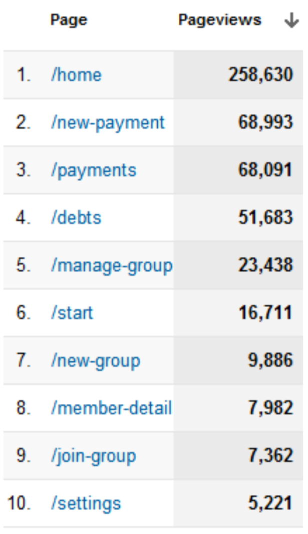
Figure 5.1: Data for the question: 'How is Settle Up used in time?'. Vertical bars show weekends.

5.2.1.1 Interpretation

- The app is used mostly for weekend activities. The strongest days are Friday, Saturday and Sunday, the weakest days are Tuesday, Wednesday and Thursday.
- During the day, the app is used mostly around 12:00 and 18:00 - times for lunch and dinner.

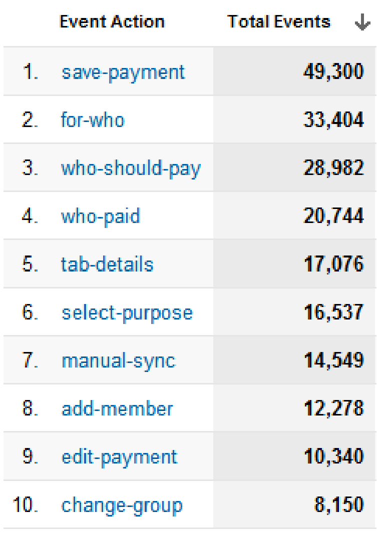
5.2.2 What are the most and least popular functions?

I have tracked an Event for every function and PageView for every screen. Figure 5.2 shows collected data.



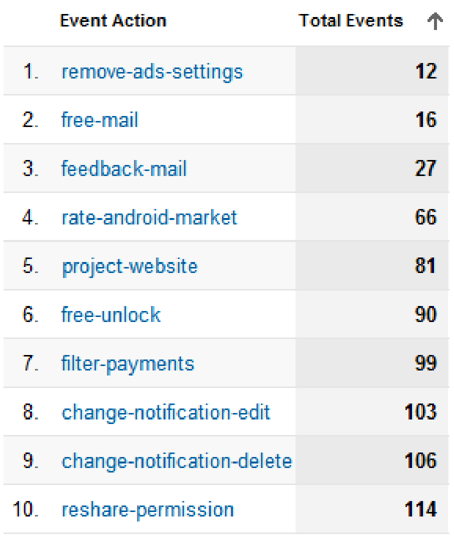
	Page	Pageviews ↓
1.	/home	258,630
2.	/new-payment	68,993
3.	/payments	68,091
4.	/debts	51,683
5.	/manage-group	23,438
6.	/start	16,711
7.	/new-group	9,886
8.	/member-detail	7,982
9.	/join-group	7,362
10.	/settings	5,221

(a) Most popular screens



	Event Action	Total Events ↓
1.	save-payment	49,300
2.	for-who	33,404
3.	who-should-pay	28,982
4.	who-paid	20,744
5.	tab-details	17,076
6.	select-purpose	16,537
7.	manual-sync	14,549
8.	add-member	12,278
9.	edit-payment	10,340
10.	change-group	8,150

(b) Most popular features



	Event Action	Total Events ↑
1.	remove-ads-settings	12
2.	free-mail	16
3.	feedback-mail	27
4.	rate-android-market	66
5.	project-website	81
6.	free-unlock	90
7.	filter-payments	99
8.	change-notification-edit	103
9.	change-notification-delete	106
10.	reshare-permission	114

(c) Least popular features

Figure 5.2: Data for the question: 'What are the most and least popular functions?'

5.2.2.1 Interpretation and impact

- Who should pay is used more then expected.

  - It looks like putting this feature on the main screen was a good idea (I wasn't sure at first). I should work on it more.
  - In May update, I introduced Who should pay improvement - "who should pay next if the person cannot pay" .
  - In May update, I introduced another Who should pay improvement - if a user inputs new payment after Who should pay, that person is preselected.

- People use Payments Log screen more than expected. Similar usage with new payments screen suggests that people check Payments Log after adding payment.
  - I should work more on Payments Log screen.
  - Maybe it would be a good idea to redirect user to Payments Log after entering a payment. The Visitor Flow diagram showed me that 43% users go Payments Log after adding payment. So I guess this should be only optional.
  - In April update, I introduced a feature for filtering Payments Log.

- All the least popular features are minor features hidden in UI on purpose except rating the app in Google Play.
  - I should promote rating in Google Play more. I don't like annoying popups, but maybe some small reward for people who rated.

- People edit 25% of payments they enter.

5.2.3 What tasks take users long time?

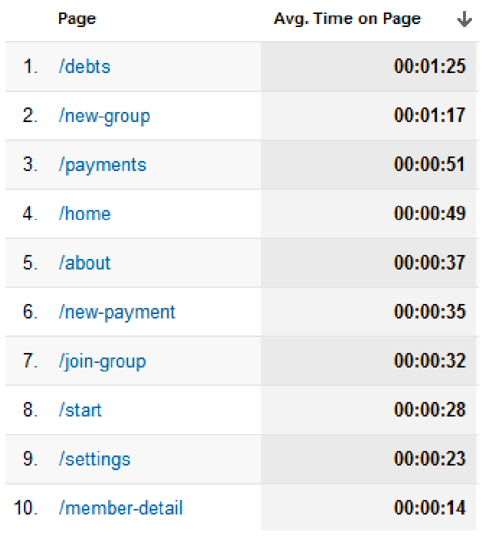
Google Analytics provide data about time spent on each page. Average visit duration for the whole app is 3 minutes 2 seconds and it is decreasing (2 minutes 59 seconds for the last month). Average time for each page is shown in Figure 5.3

5.2.3.1 Interpretation and impact

- Spending a lot of time in the debts screen is reasonable. People are discovering who owes whom and why.
  - In April update, I added help text "How is this calculated?" to the screen to make it more clear.

- Amount of time spent for group creation is alarming.
  - I am not sure why it takes so long. I should observe some users while creating a group and find the weak spots.

- Time for a new payment input is pretty good
  - It is a proof I have designed the form well.



	Page	Avg. Time on Page
1.	/debts	00:01:25
2.	/new-group	00:01:17
3.	/payments	00:00:51
4.	/home	00:00:49
5.	/about	00:00:37
6.	/new-payment	00:00:35
7.	/join-group	00:00:32
8.	/start	00:00:28
9.	/settings	00:00:23
10.	/member-detail	00:00:14

Figure 5.3: Data for the question: 'What tasks take users long time?'

5.2.4 What's different about online and offline users?

I have used session-level custom variable 'online' with values 'true' and 'false'. It marks a particular visit online or offline based on whether the user was working with an online or offline group. Figure C.1 shows differences in usage. Some interesting numbers:

- There were 3,013 unique online users and 6,123 unique offline users.
- There were 48,526 online visits and 39,827 offline visits.

5.2.4.1 Interpretation and impact

- There are more offline users, but online users are more active. I am surprised that there is more usage of online groups than offline groups.
  - I should prioritize bugfixing of online synchronization, since it is more widely used.
- Online users are more experienced. They use more advanced features like calculator and they switch groups more often.
  - In April update, I made group switching easier using a swipe gesture.
- Users from Spain, US and Germany use online groups much more often than the ones in the Czech Republic.
  - I should focus on marketing online groups in the western countries, while I should make sure Czech users know they can use the app fully offline.

5.2.5 What's different about paid users and free users?

I have used user-level custom variable 'paid' with values 'true' and 'false' based on whether the user purchased ad-free version. Figure C.2 shows collected data.

5.2.5.1 Interpretation and impact

- Paid users are usually online users as well. They are using similar functions as online users (join-group, manual-sync etc.). They are more advanced about using functions like swipe to change groups. They trust more calculated transactions because they check details tab less often.
  - Focus on the online functionality and priority bugfixing will bring more paying users.
- English and German speaking users are most willing to pay. On the other hand Czech and Spanish users are not so willing to pay.
  - I should focus my marketing efforts in the English and German speaking countries.
- There are 213 active paid users in the last 2 months while according to Google Play stats, there are 232 purchases and 212 active users of the paid version. That means piracy of the paid version is not happening.
  - I don't have to invest more time into fighting piracy.

5.2.6 What's different about new users and experienced users?

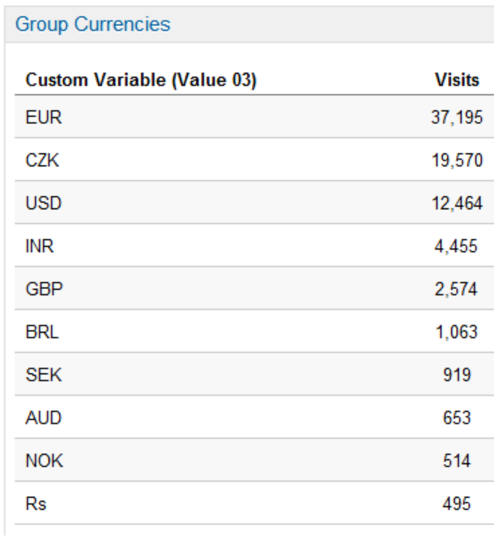
Google Analytics automatically categorizes users as 'New' or 'Returning'. There is also a value 'Count of visits'. I have marked experienced users as users with count of visits larger than 8. Figure C.3 shows collected data.

5.2.6.1 Interpretation and impact

- Most of the behaviour is expected - experienced users spend more time with Debts and Payments Log before they have more data there. I was surprised that many people go into settings and use the calculator during the first usage.
- Group creation takes 1 minute 26 seconds for the new users.
  - It is too long, since every user is required to create some group after first start. I should seriously consider creation of a pre-populated group as mentioned in subsubsection 4.2.2.1.

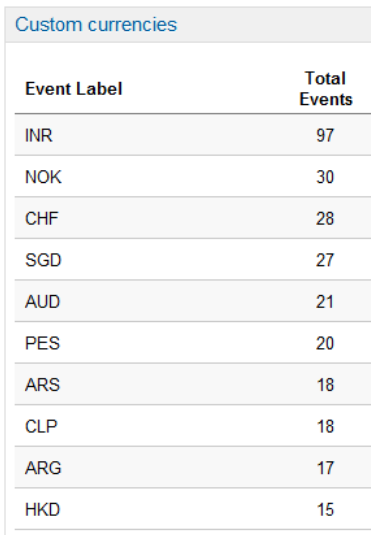
5.2.7 What currencies are being used?

I have used a session-level custom variable 'currency' with currency code for user's group. The Event for creating custom currency saves the currency code into its label. Figure 5.4 shows collected data.


Group Currencies

Custom Variable (Value 03)	Visits
EUR	37,195
CZK	19,570
USD	12,464
INR	4,455
GBP	2,574
BRL	1,063
SEK	919
AUD	653
NOK	514
Rs	495

(a) Most common group currencies


Custom currencies

Event Label	Total Events
INR	97
NOK	30
CHF	28
SGD	27
AUD	21
PES	20
ARS	18
CLP	18
ARG	17
HKD	15

(b) Most common custom currencies

Figure 5.4: Data for the question: 'What currencies are being used?'

5.2.7.1 Interpretation and impact

- Currencies INR, AUD, NOK and CHF are not present in Settle Up by default.
  - In May update, I added those currencies to make it easier for users from those countries.

5.2.8 What payment purposes are being used?

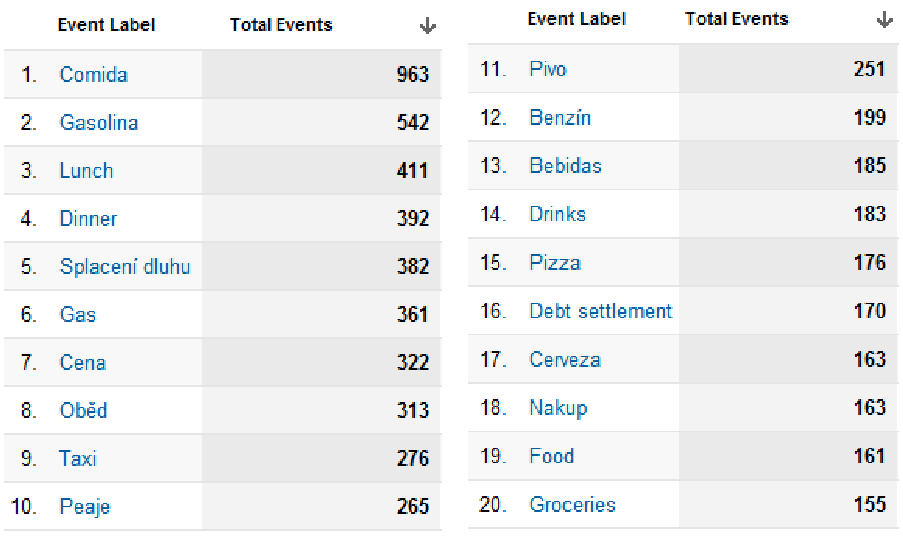
An Event 'save-payment' saves the purpose into its label. The most popular ones are listed in Figure 5.5.

5.2.8.1 Interpretation and impact

- Purposes 'Taxi', 'Toll', 'Pizza', 'Groceries', 'Rent', 'Parking' 'Breakfast' and 'Loan' weren't in the default purposes.
  - In May update, I added those default purposes.

5.2.9 Can UI handle common screen resolutions?

Google Analytics tracks screen resolutions automatically. Figure 5.6 shows the most common screen resolutions.



	Event Label	Total Events
1.	Comida	963
2.	Gasolina	542
3.	Lunch	411
4.	Dinner	392
5.	Splacení dluhu	382
6.	Gas	361
7.	Cena	322
8.	Oběd	313
9.	Taxi	276
10.	Peaje	265

(a) First 10 purposes

	Event Label	Total Events
11.	Pivo	251
12.	Benzín	199
13.	Bebidas	185
14.	Drinks	183
15.	Pizza	176
16.	Debt settlement	170
17.	Cerveza	163
18.	Nakup	163
19.	Food	161
20.	Groceries	155

(b) Second 10 purposes

Figure 5.5: Data for the question: 'What payment purposes are being used?'

5.2.9.1 Interpretation and impact

- Lowest resolution is 240x320 - a screen of HTC Wildfire.
  - I have tested the app on HTC Wildfire and modified a few minor UI features to fit nicely.
- Tablet resolutions (like 1280x752) are used by ~3% of users.
  - February redesign made the app usable on tablets. But it is not ideal. The app could use available space better with Fragments. But for such a small user group, it is not worth it (yet).

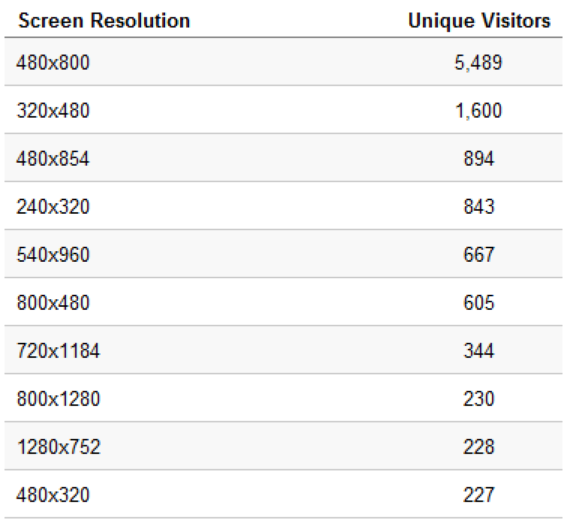
5.2.10 Can UI handle average text input?

I save length of the text for events 'save-payment', 'add-member-later', 'change-member-name', 'save-new-group' and 'rename group'. Google Analytics compute the average lengths:

- Average group name length: 12 characters
- Average member name length: 7 characters
- Average purpose length: 8 characters

5.2.10.1 Interpretation and impact

- I have tried to enter sample data with little more than average length on a phone with a minimal resolution 240x320 (see subsubsection 5.2.9.1). Group name and purpose



Screen Resolution	Unique Visitors
480x800	5,489
320x480	1,600
480x854	894
240x320	843
540x960	667
800x480	605
720x1184	344
800x1280	230
1280x752	228
480x320	227

Figure 5.6: Data for the question: 'Can UI handle common screen resolutions?'

were okay, but member name was wrapping in a Who should pay dialog, which did not look good.

- In March update, I lowered font size in Who should pay dialog and now it handles average length properly.

5.2.11 What screens should I optimize for landscape mode?

I have used page-level custom variable 'orientation' with values 'portrait' and 'landscape'. Google Analytics didn't give me the report per page I needed, so I have to do some more processing. Results are displayed in Table 5.1.

Table 5.1: Percentage of individual screens in landscape orientation

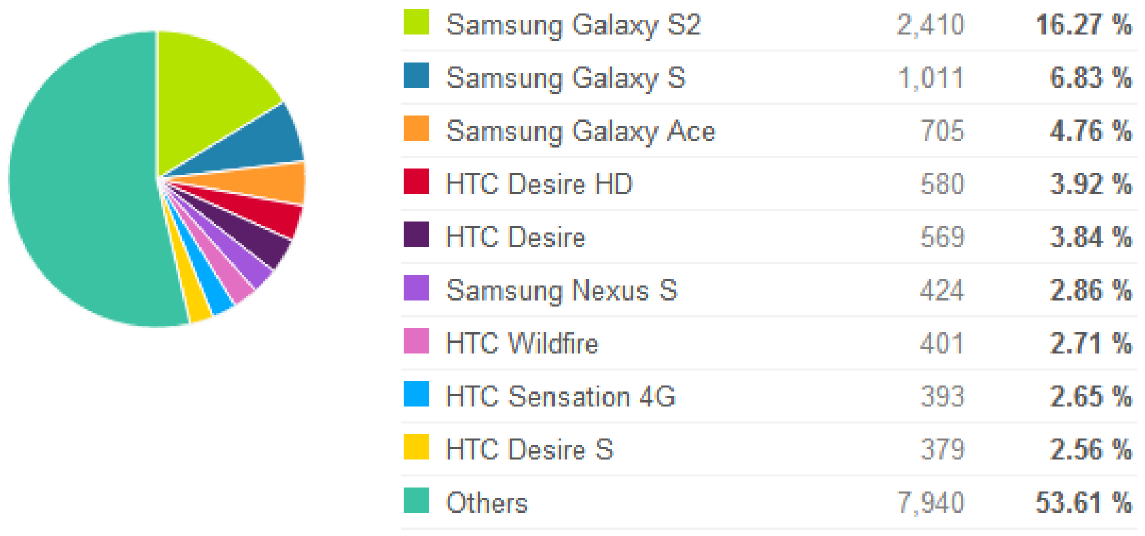
Page	Fraction of landscape pageviews
/new-group	8.34 %
/start	8.15 %
/new-payment	7.33 %
/member-detail	7.24 %
/manage-group	6.12 %
/join-group	6.0 %
/home	5.98 %
/payments	5.83 %
/debts	5.34 %
/settings	4.58 %

5.2.11.1 Interpretation and impact

- I expected higher landscape usage for the New Payment screen, but not for New Group and Start screen.
  - I have been optimizing New Payment for landscape. I should have a look at New Group and Start screen.

5.2.12 What devices should I test on?

Google Analytics track devices automatically, but not very well. The majority of devices have value '(not set)'. Google Play stats are a better source of information about devices. Figure 5.7 shows the most popular devices among active users.



🟩	Samsung Galaxy S2	2,410	**16.27 %**
🟦	Samsung Galaxy S	1,011	**6.83 %**
🟧	Samsung Galaxy Ace	705	**4.76 %**
🟥	HTC Desire HD	580	**3.92 %**
🟪	HTC Desire	569	**3.84 %**
🟪	Samsung Nexus S	424	**2.86 %**
🟪	HTC Wildfire	401	**2.71 %**
🟦	HTC Sensation 4G	393	**2.65 %**
🟨	HTC Desire S	379	**2.56 %**
🟩	Others	7,940	**53.61 %**

Figure 5.7: Data for the question: 'What devices should I test on?'

5.2.12.1 Interpretation and impact

- Dominance of Samsung Galaxy SII is surprising, because it is a high-end phone.
  - Minimal set of testing devices is Samsung Galaxy SII and HTC Wildfire (because of resolution, see subsubsection 5.2.9.1). I am regularly testing on both devices.

Chapter 6

Monetization

Successful monetization is a goal of many developers. It's important to be rewarded for your work, however aggressive monetization can annoy users and damage the reputation of the whole project. In Settle Up, I am trying to be unobtrusive. People should pay because they like and use the product, not because they are forced. I believe this approach might earn less in the short term, but more in the long term. This chapter will describe different business models I am using and evaluate their profitability.

6.1 Expenses

Developing an app like Settle Up is pretty cheap. However there are some expenses worth mentioning:

- AppEngine - after 6 months of free traffic the quota 'Datastore reads' was exceeded. I started to pay $2.10 per week, which is a minimal charge. In total $35.10 for the first project year.
- GetSatisfaction - management options for this service are not free. I was paying $13 monthly for the first 6 months and $19 is a regular price. In total $135.
- Paid review - my experiments with paid review (subsubsection 4.6.3.2) cost $27.
- AdMob - my experiments with advertising (subsection 4.6.4) cost $50.
- Total expenses for the first project year are $247.

6.2 Donations

Donations are great for free or open-source products. Purchase of the paid version could be considered a donation, since supporting development is one of the top reasons listed (more in section 6.4).

I am really impressed by project Flattr[55]. The idea behind Flattr is simple - people would put a Flattr button on the project website and a single click means a micro-donation.

User specifies a fixed amount per month for donations and this money is divided among flattred projects. If lots of people use it, it could have a huge impact. Today it's used mostly by early adopters. I put a Flattr button on the Settle Up website and promised users via social networks that at least 3 microdonations will result in lowering the price to $0.99 for two days. It happened and I kept my promise (it was also an experiment about app pricing, see subsection 6.4.1). In total 6 users clicked the button and my revenue so far is 3.53 EUR.

6.3 Free version with advertising

Advertising is one of the most common ways of Android monetization. It can be really profitable for some developers - most famous example is Angry Birds, earning $1 million each month from advertising[56]. In Settle Up, I am using the ad network AdMob[57]. There are a lot of other ad networks on the market, but I was forbidden to use other networks because of my internship at Google (conflict of interest).

The advertisement is displayed on the main screen - see Figure 6.1. It is not intended to make most of the revenue from ads, but rather drive purchases of a paid version which removes the ads.

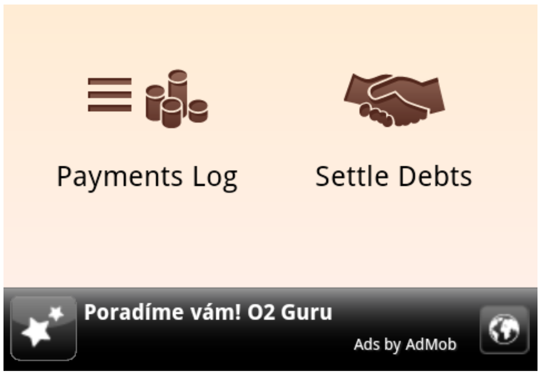


Figure 6.1: AdMob advertisement at Settle Up main screen

6.3.1 House ads

House Ads are a very interesting feature of AdMob. With House Ads, you can define your own advertisements which are shown in your apps instead of network-generated ads. See Figure 6.2 for an example of the house ad. I have carried out these house ad campaigns:

- Ads for Settle Up in my other Android app 'GPS Averaging'. It helped to gain some first users after the release.
- Ads in Settle Up promoting the paid version. The campaign ran for 14 days beginning October 31, 2011.
  - Localized Czech ad resulted in 129 clicks with 2.08% CTR.
  - Generic English ad resulted in 116 clicks with 0.89% CTR.

  - I have sold 15 paid versions, that is a revenue of \$3.20 per day. It was \$2.26 per day before the campaign.

- Ads in Settle Up promoting the iPhone version. The campaign ran 7 days beginning April 23, 2012 only in the Czech Republic.
  - It resulted in 61 clicks with 1.08% CTR.

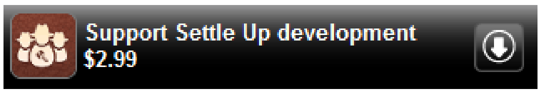
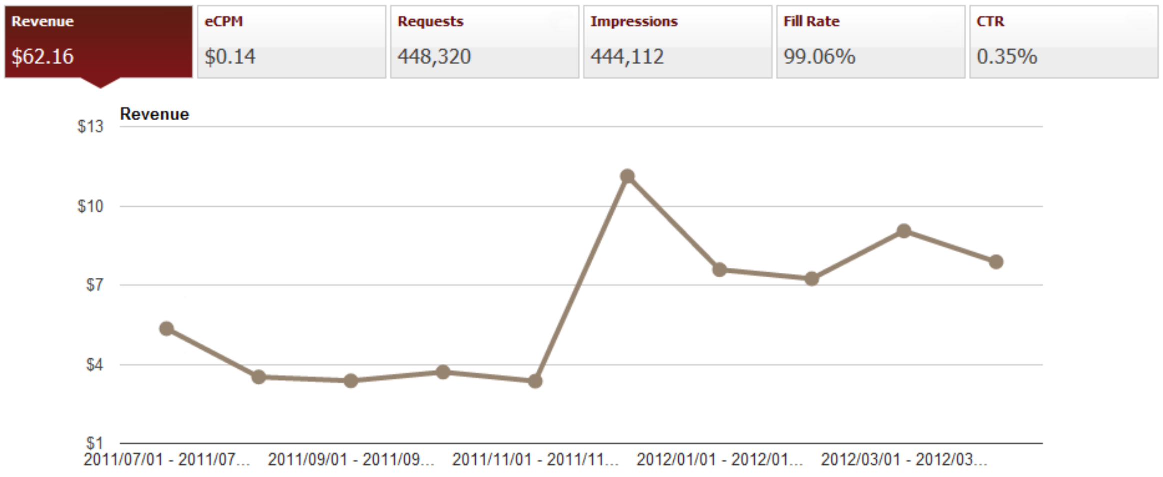


Figure 6.2: House ad promoting paid version of Settle Up

6.3.2 Evaluation

Figure 6.3 shows overall revenue and revenue growth for each month. The growth in December was caused by Vodafone CZ promotion (subsubsection 4.6.3.3). In conclusion the revenue from advertising is small - 13% of revenue from paid version (section 6.5). However it's a good idea to have a free version - every user can try it, use it and spread the good word about the app. Removing ads is the main reason for buying a paid version.



Revenue	eCPM	Requests	Impressions	Fill Rate	CTR
\$62.16	\$0.14	448,320	444,112	99.06%	0.35%

Figure 6.3: Statistics of ad revenue from July 2011 to April 2012
The graph shows revenue for each month. Legend: eCPM - revenue per thousand impressions, Fill Rate - percentage of impressions compared to requests, CTR - click-through rate

6.4 Paid version

The paid version is technically another app called "Settle Up: No Ads". Users still need a free version, this app serves as a key for unlocking paid functionality. It's better than having a full-featured paid version, because both paid and free users are using the same app and contribute to the statistics. It would be better to solve this by In-app Billing[58], but I have read many opinions that implementing In-app Billing on Android is time-consuming and frustrating. I would focus on it in case of a high piracy, but subsubsection 5.2.5.1 showed that piracy of Settle Up is almost non-existent.

6.4.1 The price experiment

One of the hardest parts is how to set a price for your app. I am not forcing users to buy it, paid version is usually bought by users who use it for some time and are satisfied. My thoughts are that these users are willing to pay more than minimal price $0.99. So I set the price for $2.99.

But I was not sure if a lower price would generate higher revenue and more happy users. I did an experiment on February 14, 2012. I lowered the price to $0.99 and promoted this discount on social networks. Seven users bought the paid version, resulting in revenue of $6.93, that's $3.46 per day. In a week before promotion I sold 15 paid versions for $2.99, that's $6.40 per day. The experiment didn't prove that lower price is better. But it might be worth to lower price for a longer period and gather more accurate data.

6.4.2 Premium-first functionality

I was thinking how to make the paid version more attractive. For a long time users would get only ads removed and a good feeling about supporting development. Premium functionality was an obvious choice, but I didn't want to turn Settle Up into a trial version. That would negatively impact the reputation. I couldn't make already released features premium-only so I got idea about giving new features a month early to the paid users. I call it a 'premium-first' approach. Paid users will get something extra and I will test new features on a limited group of people until it's released for everybody. Not every feature is suitable for this. I have looked into GetSatisfaction (subsubsection 4.3.3.1) and selected most wanted distinct features.

6.4.2.1 Evaluation

- 5 paid versions were sold in a week before first premium feature, that's $2.13 per day.
- I launched a feature 'Import/Export to SD card' in 'premium-first' on March 29, 2012.
  - 7 paid versions were sold in the following week, that's $2.99 per day.
- I launched feature 'Filter Payments Log' in 'premium-first' on April 13, 2012.
  - 9 paid versions were sold in the following week, that's $3.84 per day.

Although I expected higher purchases, the experiment was successful and I will continue to roll out features for paid users first. It allows me to promote the same features twice. It grows the revenue, doesn't hurt the reputation and I have fixed a few bugs in those features before I released them to everyone.

6.4.3 Impact of important events on revenue

Features which are used by paid users are described in subsubsection 5.2.5.1. Improving them will generate more revenue. I was curious about the revenue effects of individual updates and other important events in the project. Google Play offers data about daily downloads only from November 29 2011 - that's why I don't include earlier reports. Figure 6.4 shows a graph of daily purchases from November 2011 to February 2012 and Figure 6.5 shows similar graph from February to May 2012.

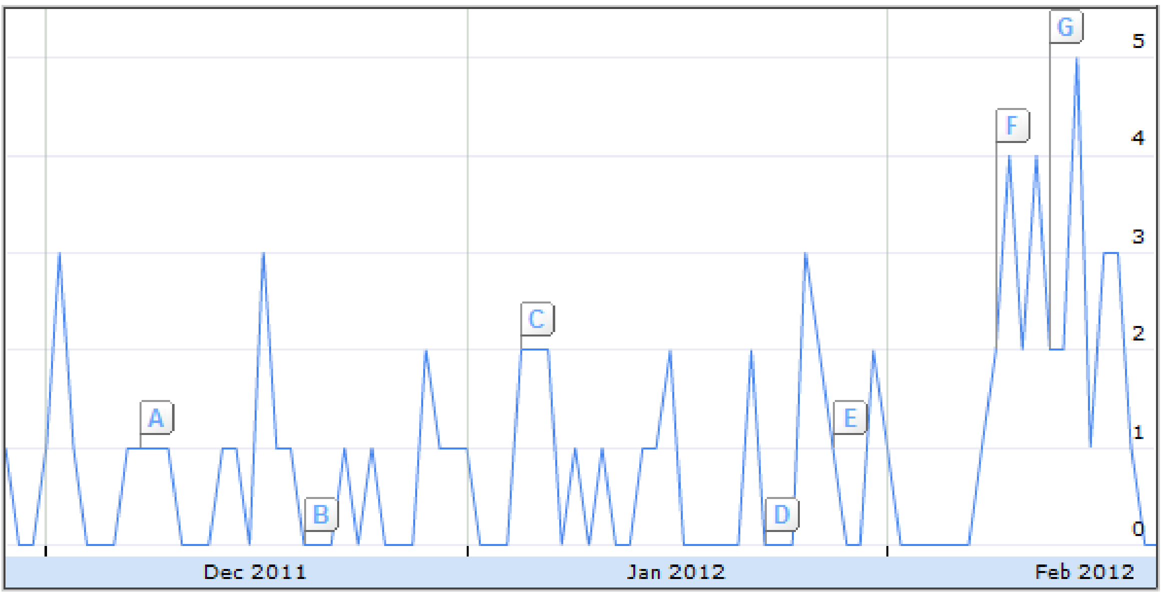


Figure 6.4: Daily purchases of paid version from November 29, 2011 to February 21 2012
A - Vodafone CZ promotion, B - iPhone version, C - v3.7(bugfixes, translations), D - v3.8(easier group sharing), E - v3.8.5(Bugfixes), F - v4.0(ICS redesign), G - v4.0.4(Bugfixes)

6.4.3.1 Interpretation

- Promotions like the one from Vodafone CZ or AndroidZoom.com didn't have a big effect on purchases. It confirms that people use the free version first and purchase if they are satisfied.
- ICS redesign was a big success confirmed by increased purchases. Important updates have positive effect on purchases.
- Not all updates result in a purchase growth like the update about easier group sharing.
- I was afraid that an introduction of Google Analytics tracking would slow down purchases, but it didn't happen.
- Premium-first features drive purchases as analyzed in. subsection 6.4.2.

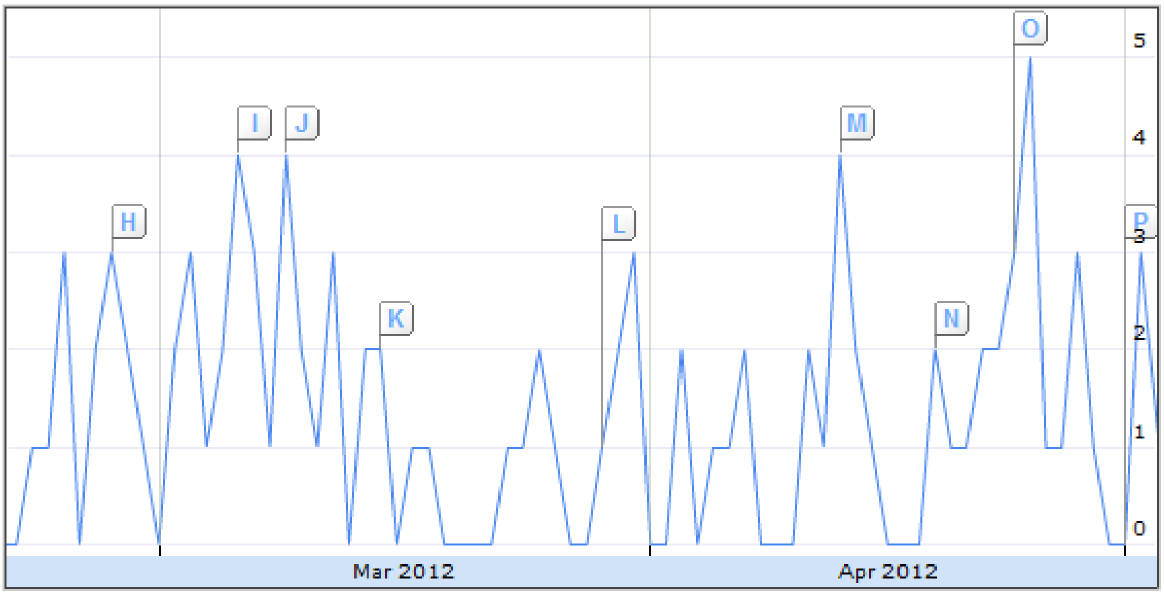


Figure 6.5: Daily purchases of paid version from February 22 to May 3 2012
H - v4.1(Google Analytics), I - web interface, J - v4.2(lot of tweaks), K - Review at AndroidZoom.com, L - v4.5(premium-first: export/import), M - v4.5(premium-first: filter payments), N - website redesign, O - Review at MobilMania.cz, P - v4.7(everyone: import/export)

6.5 Totals

When calculating revenue, one must subtract a 30 % transaction fee for Google. Figure 6.6 shows monthly revenue from the paid version after subtracting the fee. I have earned the following in a first project year:

- $462 for the paid version and $62 for ads
- Expenses were $247
- Profit was $277

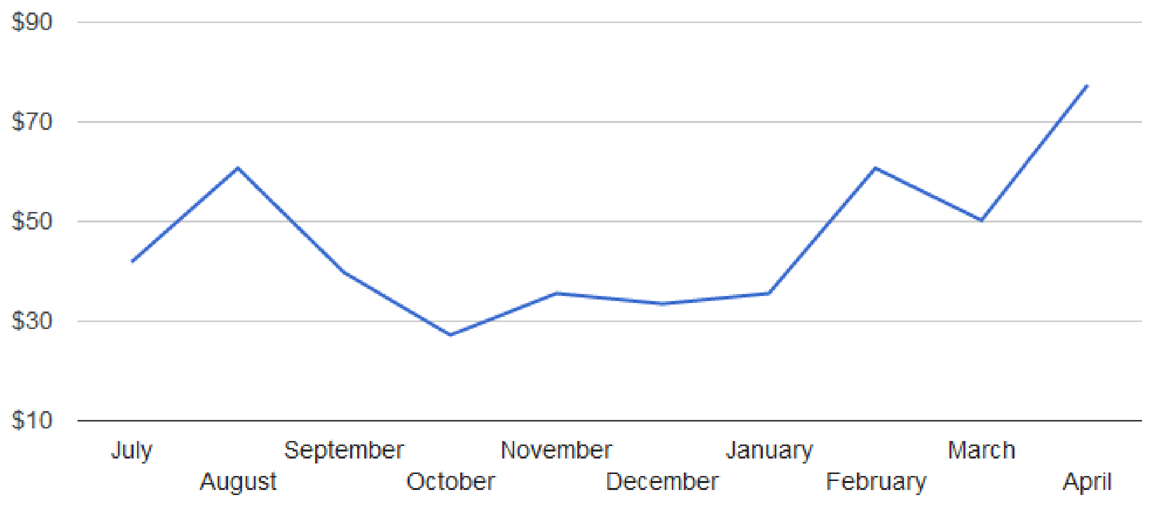


Figure 6.6: Monthly revenue from the paid version

Chapter 7

Conclusion

It has been an interesting year with Settle Up. I have learned a lot about Android and user behaviour. I have found that the process after release is hard and time-consuming, but it can be managed by a single developer like me. I hope that other developers and marketers will benefit from my experience. The revenue from monetization didn't compensate my time (section 6.5), but the project is getting more popular and data suggests that the revenue will grow if I continue working on the project. The future of Settle Up[2] looks bright.

7.1 Goal fulfilment

Let's look at the goals from section 1.1:

- I designed a mobile application for group expenses (chapter 2).
- I described synchronization with a cloud server (section 2.5), open API (section 3.3) and related cooperation with third-parties (section 4.5).
- I showed ways of gathering feedback from users related to the app evolution (section 4.3).
- I evaluated marketing efforts related to the user growth (section 4.6).
- I used measurements of user behaviour for the app improvements (chapter 5).
- I proposed various business models and evaluated their profitability (chapter 6).

7.2 Future work

I am planning to keep managing the project and improving it based on user feedback. I will release new features premium-first, which will boost the revenue (subsection 6.4.2). Here is my to-do list for a next few months:

- Improve currency support because it's a number one suggestion in Google Play comments (subsubsection 4.3.3.2).

- Implement a feed of changes from synchronization since it's one of the top requests at GetSatisfaction (subsubsection 4.3.3.1). The measurements showed that paid users use online synchronization heavily (subsubsection 5.2.5.1).
- Prepare a demo group for the new users to improve first-time experience which takes too much time according to subsubsection 5.2.3.1.
- Remove a dashboard pattern on the main screen and redesign whole app for tablets and phones as suggested by Reto Meier in subsection 4.3.4.

In the more distant future Settle Up will serve as a great sample of my work and it will help me to get hired as an Android developer. There are three options for the project afterwards: continue working on it as my passion project, open-source it or sell it.

Bibliography

[1] Combinatorical Optimization - A4M33KO.
http://www.fel.cvut.cz/education/bk/predmety/12/58/p12581804.html.

[2] Settle Up - Google Play.
https://play.google.com/store/apps/details?id=cz.destil.settleup.

[3] UI Design Pattern - Dashboard.
http://www.androiduipatterns.com/2011/02/ui-design-pattern-dashboard.html, February, 26 2011.

[4] Android Design guidelines.
http://developer.android.com/design/index.html.

[5] Android Design - ActionBar.
http://developer.android.com/design/patterns/actionbar.html.

[6] Algorithm to share/settle expenses among a group.
http://stackoverflow.com/questions/974922, June 2009.

[7] Who owes who money optimization problem.
http://stackoverflow.com/questions/4554655, December 2010.

[8] What algorithm to use to determine minimum number of actions required to get the system to 'zero' state?
http://stackoverflow.com/questions/877728, May 2009.

[9] Cormen Thomas H.; Leiserson Charles E.; Rivest Ronald L.; Stein Clifford;. *Introduction to Algorithms, 35.5: The subset-sum problem.* MIT Press, 2nd edition, 2001.

[10] Michael Gilleland. Combination Generator.
http://www.merriampark.com/comb.htm.

[11] Kenneth H. Rosen. *Discrete Mathematics and Its Applications, pages 284–286.* NY: McGraw-Hill, 2nd edition, 1991.

[12] David Vávra. Settle Up API Documentation.
http://www.settleup.info/?api.

[13] AppEngine - Developer's Guide.
https://developers.google.com/appengine/docs/.

[14] Jake Wharton. ActionBarSherlock.
http://actionbarsherlock.com/.

[15] Android Asset Studio.
http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html.

[16] Content Providers - Android documentation.
http://developer.android.com/guide/topics/providers/content-providers.html.

[17] Dashboard Layout implementation.
http://code.google.com/p/iosched/source/browse/android/src/com/google/android/apps/iosched/ui/widget/DashboardLayout.java.

[18] AsyncTask - Android documentation.
http://developer.android.com/reference/android/os/AsyncTask.html.

[19] Sample Sync Adapter.
http://developer.android.com/resources/samples/SampleSyncAdapter/index.html.

[20] Google GSON Library.
http://code.google.com/p/google-gson/.

[21] Royalty Free Icons.
http://icons.mysitemyway.com/.

[22] Michal Acler - Mobile Designer.
http://www.linkedin.com/in/agilek.

[23] Bioport Software Labs.
http://bioport.cz/.

[24] Google Play Developer Console.
https://play.google.com/apps/publish.

[25] Andlytics.
https://play.google.com/store/apps/details?id=de.betaapps.andlytics&hl=en.

[26] Conmigo.
https://play.google.com/store/apps/details?id=com.conmigo.share.

[27] Tricount.
http://tricount.com/en/organizing-group-expenses-among-friends/.

[28] Google Group for Settle Up.
https://groups.google.com/forum/?fromgroups#!forum/settle-up.

[29] GetSatisfaction for Settle Up.
https://getsatisfaction.com/settle_up.

[30] Android Developers - Google+.
https://plus.google.com/108967384991768947849.

[31] The Friday App Review (May 4, 2012).
http://www.youtube.com/watch?v=nBYz-YlVtAc.

[32] Get Localization - crowdsourced translations of Settle Up.
http://www.getlocalization.com/settle_up/.

[33] Settle Up website.
http://www.settleup.info.

[34] Bioport Software Labs. Video for Settle Up promotion.
http://www.youtube.com/watch?v=kOn3yB9-tqY.

[35] Facebook page for Settle Up.
http://www.facebook.com/settleup.dluznicek.

[36] Twitter account for Settle Up.
https://twitter.com/#!/settle_up.

[37] Google+ page for Settle Up.
https://plus.google.com/101747049863114967912.

[38] Google Alerts.
http://www.google.com/alerts.

[39] AppParade #3 - Pitch 06 - David Vavra.
http://www.youtube.com/watch?v=qZ7YlXYyqzE.

[40] Třetí přehlídku AppParade ovládly aplikace pro Android. Zvítězil Night Walker Tomáše Zvěřiny.
http://www.mediar.cz/treti-prehlidku-appparade-ovladly-aplikace-pro-android-zvitezil-night-walker-tomase-zveriny/, May 12 2011.

[41] Vodafone CZ. Aplikace Dlužníček vám udělá pořádek ve financích.
http://www.facebook.com/note.php?note_id=10150276484900638.

[42] mobileDevCamp.
http://www.mdevcamp.cz/.

[43] Third PayPal X Developer Challenge is all about Android.
http://www.intomobile.com/2011/03/07/third-paypal-x-developer-challenge-all-android/.

[44] Křišťálová Lupa 2011 - výsledky.
http://kristalova.lupa.cz/2011/vysledky-2011/.

[45] Svět Androida. Dlužníček – mějte finance pod kontrolou.
http://www.svetandroida.cz/dluznicek-%E2%80%93-mejte-finance-pod-kontrolou-201106.

[46] MobilMania. Android Apps 6: ČR/SR speciál.
http://dotekomanie.blog.mobilmania.cz/2011/05/android-apps-6-crsr-special/.

[47] El androide libre. Gestiona los gastos compartidos de viajes y cenas con Settle Up y MissPlitty.
http://www.elandroidelibre.com/2011/07/gestiona-los-gastos-compartidos-de-viajes-y-cenas-con-settle-up-y-missplitty.html.

[48] Xataka Android. Settle up, la mejor forma de gestionar pagos en grupos.
http://www.xatakandroid.com/aplicaciones-android/settle-up-la-mejor-forma-de-gestionar-pagos-en-grupos.

[49] Android App Log. Settle Up.
http://www.androidapplog.com/2011/09/android-app-reviews/settle-up-app-review/.

[50] Katalog Aplikací. Dlužníček.
http://www.katalogaplikaci.cz/Aplikace-887-Dluznicek-pro-Android.

[51] AndroidZoom. Settle Up.
http://www.androidzoom.com/android_applications/finance/settle-up_xlrh.html.

[52] iPhoneMania. Dlužníček: iPhone a Android vyřeší společné výdaje každé party.
http://iphonemania.mobilmania.cz/Dluznicek+iPhone+a+Android+vyresi+spolecne+vydaje+kazde+party.

[53] Google Analytics SDK for Android.
https://developers.google.com/analytics/devguides/collection/android/devguide.

[54] Google Analytics Sample Code.
http://code.google.com/p/analytics-api-samples/.

[55] Settle Up - Flattr.
https://flattr.com/thing/492336/Settle-Up.

[56] Angry Birds Shows Advertising Model For Games Works.
http://thenextweb.com/eu/2010/12/08/michael-hed-angry-birds/.

[57] AdMob.
http://www.google.com/ads/admob/.

[58] In-app Billing Overview.
http://developer.android.com/guide/market/billing/billing_overview.html.

Appendix A

List of Abbreviations

ad advertisement

API Application Interface

app Application

CPC Cost per click

CTR Click-through rate

HTTP Hypertext Transfer Protocol

ICS Ice Cream Sandwich (Android 4.0)

ID Identificator

NP Non-deterministic polynomial

sync Synchronization

stats Statistics

REST Representational State Transfer

UI User Interface

Appendix B

Source code of algorithm for settling debts

```
package cz.destil.settleup.utils;

import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

import cz.destil.settleup.data.Member;

/**
 * Debt calculator - calculates who should pay how much to who with optional
 * tolerance value.
 *
 * @author David Vavra
 *
 */
public class DebtCalculator {

  public static int MAX_MEMBERS_FOR_OPTIMAL = 15;

  /**
   * Main algorithm, calculates who should send how much to who It optimizes
   * basic algorithm
   *
   * @param members
   *            List of members with their credit and debts
   * @param tolerance
   *            Money value nobody cares about
   * @return List of Hashmaps encoding transactions
   */
  public static List<HashMap<String, Object>> calculate(List<Member> members
      , double tolerance) {
    tolerance += 0.009; // increasing tolerance due to double precision
              // issues
```

```java
List<HashMap<String, Object>> results = new LinkedList<HashMap<String, Object>>();
// remove members where debts are too small (1-pairs)
Iterator<Member> iterator = members.iterator();
while (iterator.hasNext()) {
  Member member = iterator.next();
  if (Math.abs(member.getBalance(true)) <= tolerance) {
    iterator.remove();
  }
}
// safety check
if (members.size() == 1)
  return results;

// for 18 and more members alghoritm would take long time, so using just
// basic alghoritm
if (members.size() <= MAX_MEMBERS_FOR_OPTIMAL) {

  // find n-pairs, starting at 2-pairs, deal with them using basic
  // algorithm and remove them
  int n = 2;
  while (n < members.size() - 1) {
    CombinationGenerator generator = new CombinationGenerator(members.size(), n);
    boolean nPairFound = false;
    while (generator.hasMore()) {
      double sum = 0;
      int[] combination = generator.getNext();
      for (int i = 0; i < combination.length; i++) {
        sum += members.get(combination[i]).getBalance(true);
      }
      if (Math.abs(sum) <= tolerance) {
        // found n-pair - deal with them
        List<Member> pairedMembers = new LinkedList<Member>();
        for (int i = 0; i < combination.length; i++) {
          pairedMembers.add(members.get(combination[i]));
        }
        List<HashMap<String, Object>> values = basicDebts(pairedMembers, tolerance);
        results.addAll(values);
        members.removeAll(pairedMembers);
        nPairFound = true;
      }
      if (nPairFound)
        break;
    }
    if (!nPairFound)
      n++;
  }
}
// deal with what is left after removing n-pairs
List<HashMap<String, Object>> values = basicDebts(members, tolerance);
results.addAll(values);
// Sort debts according to who should pay
Collections.sort(results, new DebtComparator());
return results;
```

```java
}

/**
 * Not-optimal debts algorithm - it calculates debts with N-1 transactions
 *
 * @param members
 *            List of members with their credit and debts
 * @param tolerance
 *            Money value nobody cares about
 * @return List of Hashmaps encoding transactions
 */
private static List<HashMap<String, Object>> basicDebts(List<Member>
    members, double tolerance) {
  List<HashMap<String, Object>> debts = new LinkedList<HashMap<String,
      Object>>();
  int resolvedMembers = 0;
  while (resolvedMembers != members.size()) {
    // transaction is from lowes balance to highest balance
    Collections.sort(members);
    Member sender = members.get(0);
    Member recipient = members.get(members.size() - 1);
    double senderShouldSend = Math.abs(sender.getBalance(true));
    double recipientShouldReceive = Math.abs(recipient.getBalance(true));
    double amount;
    if (senderShouldSend > recipientShouldReceive) {
      amount = recipientShouldReceive;
    } else {
      amount = senderShouldSend;
    }
    sender.spent -= amount;
    recipient.paid -= amount;
    // create transaction
    HashMap<String, Object> values = new HashMap<String, Object>();
    values.put("from", sender);
    values.put("amount", amount);
    values.put("to", recipient);
    debts.add(values);
    // delete members who are settled
    senderShouldSend = Math.abs(sender.getBalance(true));
    recipientShouldReceive = Math.abs(recipient.getBalance(true));
    if (senderShouldSend <= tolerance)
      resolvedMembers++;
    if (recipientShouldReceive <= tolerance)
      resolvedMembers++;

  }
  // limit transactions by tolerance
  Iterator<HashMap<String, Object>> iterator = debts.iterator();
  while (iterator.hasNext()) {
    HashMap<String, Object> debt = iterator.next();
    if ((Double) debt.get("amount") <= tolerance)
      iterator.remove();
  }
  return debts;
}

```

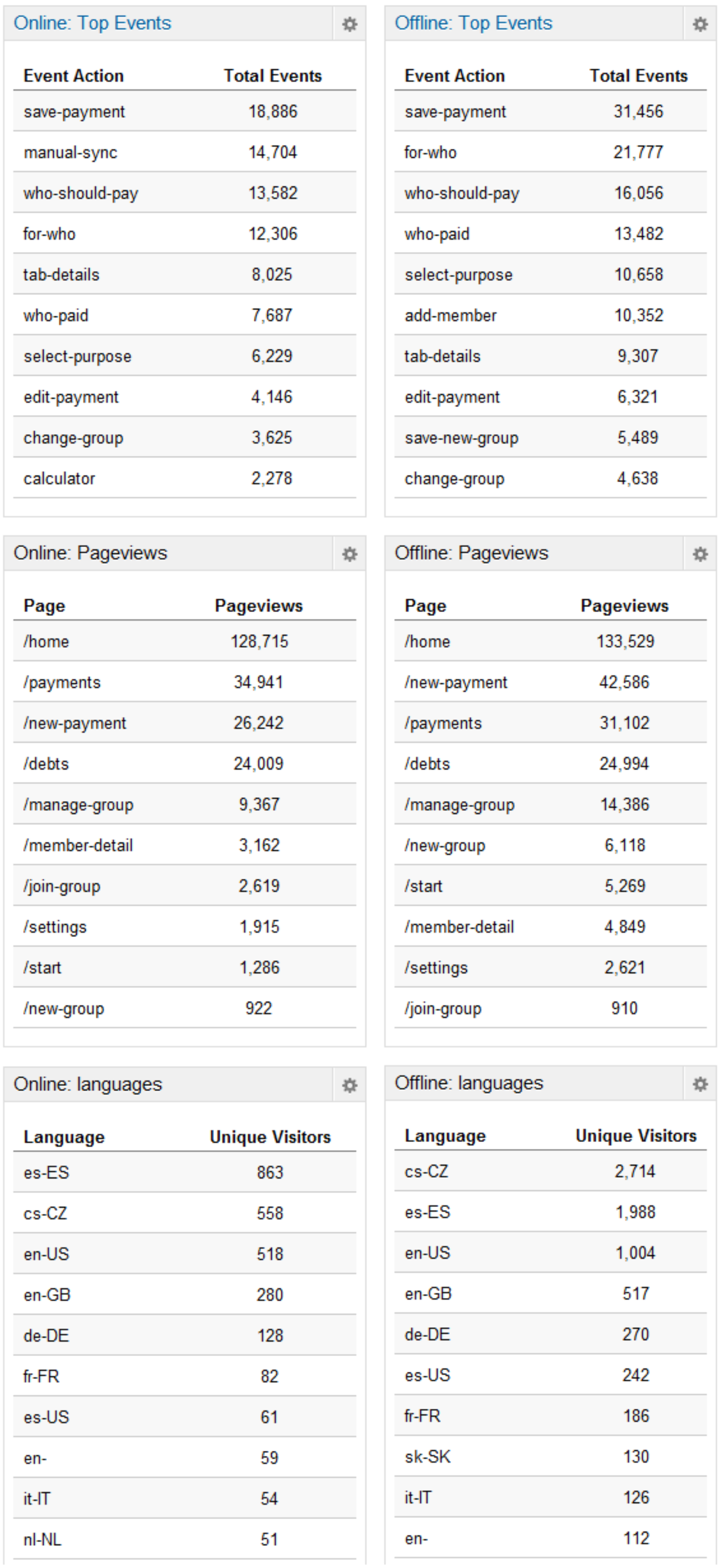
```java
  static class DebtComparator implements Comparator<HashMap<String, Object>>
      {

    @Override
    public int compare(HashMap<String, Object> lhs, HashMap<String, Object>
        rhs) {
      return ((Member) lhs.get("from")).name.compareTo(((Member) rhs.get("
          from")).name);
    }

  }
}
```

Appendix C

Detailed Google Analytics reports



Online: Top Events

Event Action	Total Events
save-payment	18,886
manual-sync	14,704
who-should-pay	13,582
for-who	12,306
tab-details	8,025
who-paid	7,687
select-purpose	6,229
edit-payment	4,146
change-group	3,625
calculator	2,278

Offline: Top Events

Event Action	Total Events
save-payment	31,456
for-who	21,777
who-should-pay	16,056
who-paid	13,482
select-purpose	10,658
add-member	10,352
tab-details	9,307
edit-payment	6,321
save-new-group	5,489
change-group	4,638

Online: Pageviews

Page	Pageviews
/home	128,715
/payments	34,941
/new-payment	26,242
/debts	24,009
/manage-group	9,367
/member-detail	3,162
/join-group	2,619
/settings	1,915
/start	1,286
/new-group	922

Offline: Pageviews

Page	Pageviews
/home	133,529
/new-payment	42,586
/payments	31,102
/debts	24,994
/manage-group	14,386
/new-group	6,118
/start	5,269
/member-detail	4,849
/settings	2,621
/join-group	910

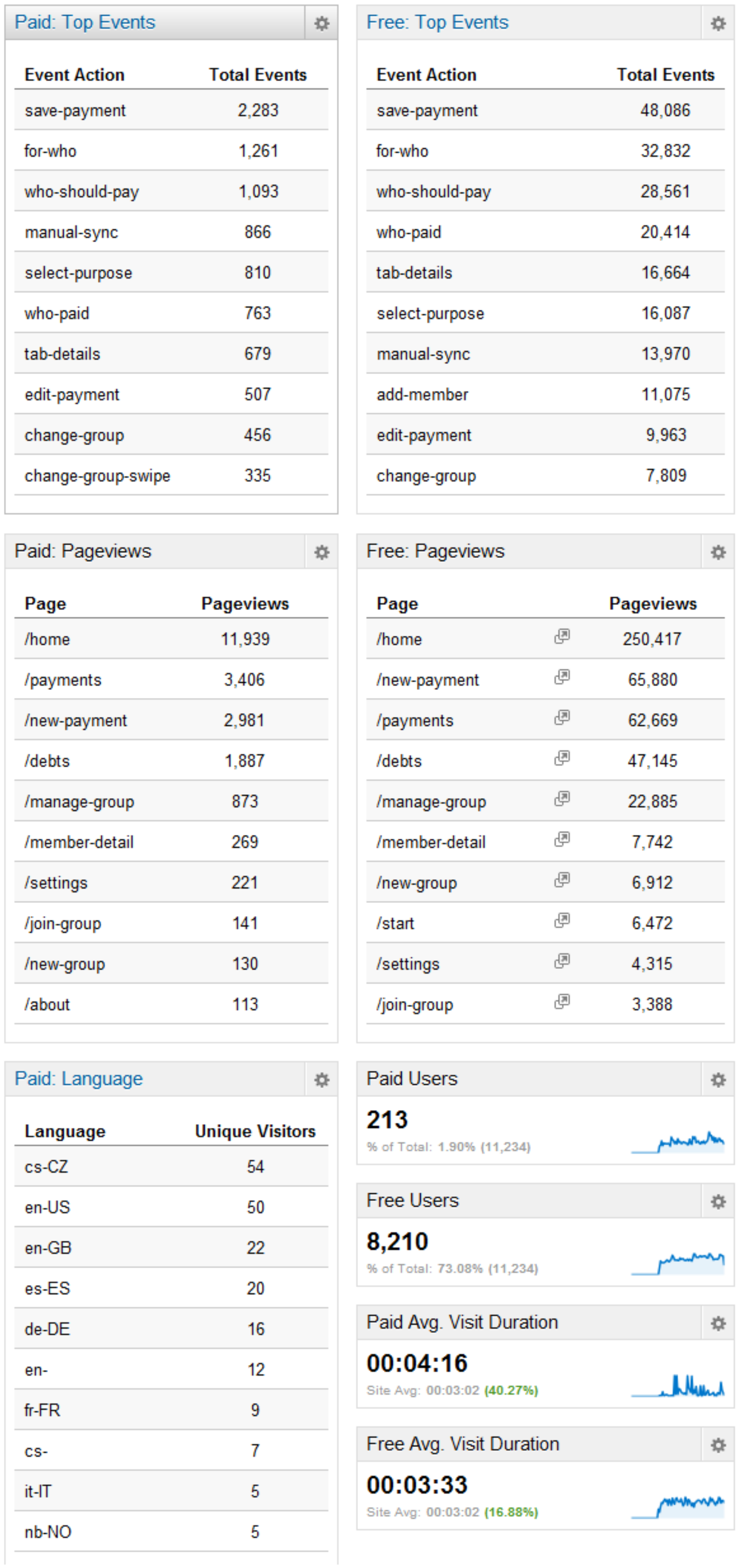
Online: languages

Language	Unique Visitors
es-ES	863
cs-CZ	558
en-US	518
en-GB	280
de-DE	128
fr-FR	82
es-US	61
en-	59
it-IT	54
nl-NL	51

Offline: languages

Language	Unique Visitors
cs-CZ	2,714
es-ES	1,988
en-US	1,004
en-GB	517
de-DE	270
es-US	242
fr-FR	186
sk-SK	130
it-IT	126
en-	112

Figure C.1: Data for the question: 'What is different about online and offline users?'



Paid: Top Events

Event Action	Total Events
save-payment	2,283
for-who	1,261
who-should-pay	1,093
manual-sync	866
select-purpose	810
who-paid	763
tab-details	679
edit-payment	507
change-group	456
change-group-swipe	335

Free: Top Events

Event Action	Total Events
save-payment	48,086
for-who	32,832
who-should-pay	28,561
who-paid	20,414
tab-details	16,664
select-purpose	16,087
manual-sync	13,970
add-member	11,075
edit-payment	9,963
change-group	7,809

Paid: Pageviews

Page	Pageviews
/home	11,939
/payments	3,406
/new-payment	2,981
/debts	1,887
/manage-group	873
/member-detail	269
/settings	221
/join-group	141
/new-group	130
/about	113

Free: Pageviews

Page	Pageviews
/home	250,417
/new-payment	65,880
/payments	62,669
/debts	47,145
/manage-group	22,885
/member-detail	7,742
/new-group	6,912
/start	6,472
/settings	4,315
/join-group	3,388

Paid: Language

Language	Unique Visitors
cs-CZ	54
en-US	50
en-GB	22
es-ES	20
de-DE	16
en-	12
fr-FR	9
cs-	7
it-IT	5
nb-NO	5

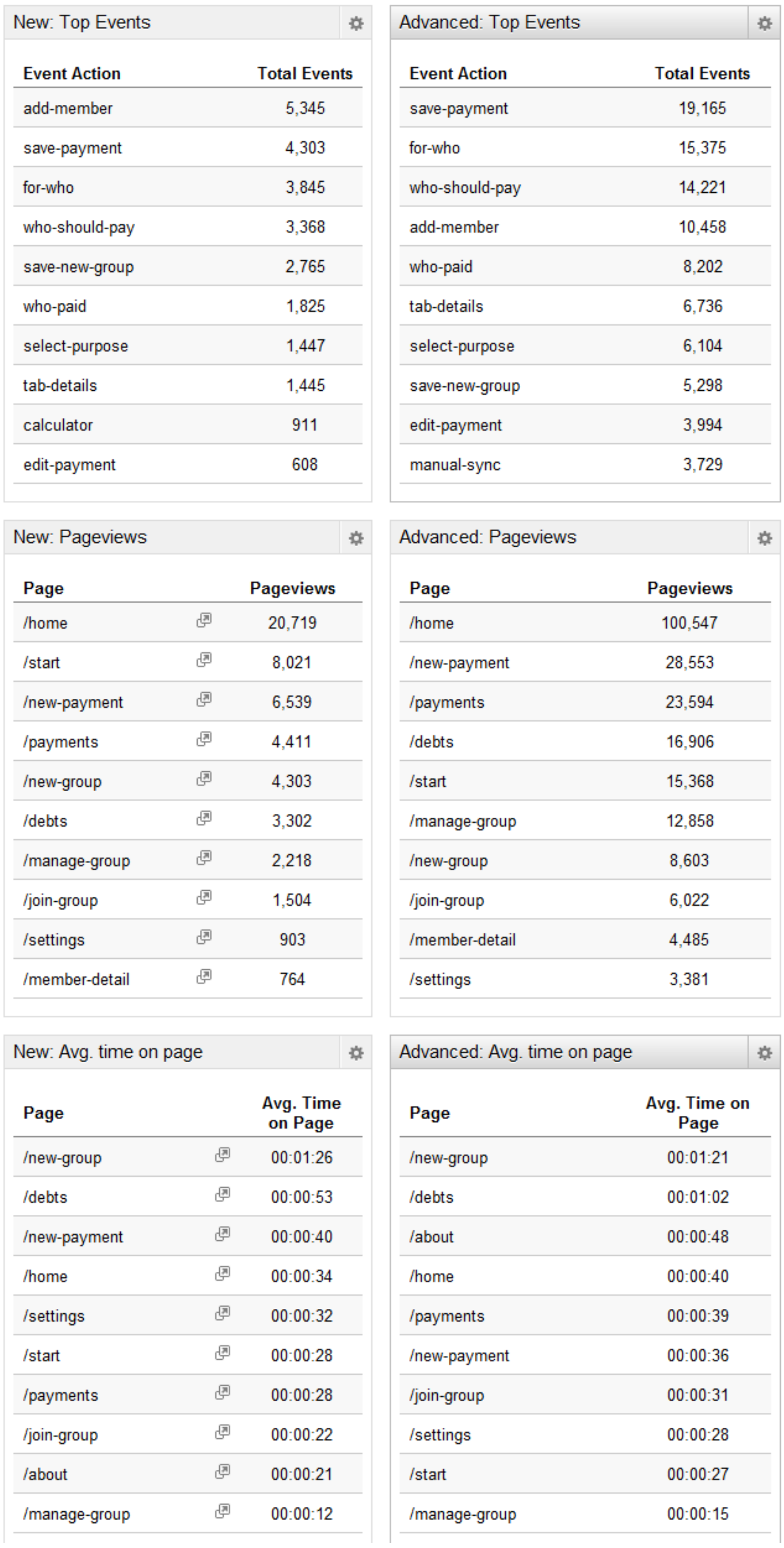
Paid Users
213
% of Total: 1.90% (11,234)

Free Users
8,210
% of Total: 73.08% (11,234)

Paid Avg. Visit Duration
00:04:16
Site Avg: 00:03:02 (40.27%)

Free Avg. Visit Duration
00:03:33
Site Avg: 00:03:02 (16.88%)

Figure C.2: Data for the question: 'What is different about paid users and free users?'



New: Top Events

Event Action	Total Events
add-member	5,345
save-payment	4,303
for-who	3,845
who-should-pay	3,368
save-new-group	2,765
who-paid	1,825
select-purpose	1,447
tab-details	1,445
calculator	911
edit-payment	608

Advanced: Top Events

Event Action	Total Events
save-payment	19,165
for-who	15,375
who-should-pay	14,221
add-member	10,458
who-paid	8,202
tab-details	6,736
select-purpose	6,104
save-new-group	5,298
edit-payment	3,994
manual-sync	3,729

New: Pageviews

Page	Pageviews
/home	20,719
/start	8,021
/new-payment	6,539
/payments	4,411
/new-group	4,303
/debts	3,302
/manage-group	2,218
/join-group	1,504
/settings	903
/member-detail	764

Advanced: Pageviews

Page	Pageviews
/home	100,547
/new-payment	28,553
/payments	23,594
/debts	16,906
/start	15,368
/manage-group	12,858
/new-group	8,603
/join-group	6,022
/member-detail	4,485
/settings	3,381

New: Avg. time on page

Page	Avg. Time on Page
/new-group	00:01:26
/debts	00:00:53
/new-payment	00:00:40
/home	00:00:34
/settings	00:00:32
/start	00:00:28
/payments	00:00:28
/join-group	00:00:22
/about	00:00:21
/manage-group	00:00:12

Advanced: Avg. time on page

Page	Avg. Time on Page
/new-group	00:01:21
/debts	00:01:02
/about	00:00:48
/home	00:00:40
/payments	00:00:39
/new-payment	00:00:36
/join-group	00:00:31
/settings	00:00:28
/start	00:00:27
/manage-group	00:00:15

Figure C.3: Data for the question: 'What is different about new users and advanced users?'

Appendix D

Contents of attached CD

An attached CD has the following structure:

```
|-- data : Exported data about downloads from Google Play
|   |-- free-version : Data about free version
|   |-- paid-version : Data about paid version
|-- doc : API documentation
|-- exe : Executable file of Android version from May 2012
|-- screenshots : Screenshots of Android app
|   |-- czech : Czech screenshots
|   |-- english : English screenshots
|-- src : Source code of 'Settle debts' algoritm. Other source code is not published.
|-- text : The thesis
|   |-- latex : LaTeX sources
|   |-- master-thesis-david-vavra.pdf : PDF output
|-- video : Videos about Android version
|-- index.html : Project overview and related links
|-- install.txt : How to install Android app
|-- readme.txt : Files overview
```