Payback

Final Report

Universidade do Porto

Faculdade de Engenharia

FEUP

Mestrado Integrado em Engenharia Informática e Computação

Sistemas Distribuídos

T6G02:
Duarte Duarte - 201109179
Francisco Maciel - 201100692
Luís Cleto - 201104279
Miguel Marques - 201109178
Ruben Cordeiro - 201109177

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal
1 Junho 2014

Index

Introduction

In this project, we created a mobile application / mobile first website to help users manage their debts with friends and other users. Most of the proposed features are now included in the final solution, with a few exceptions. Our main app features are fully operational, but we ended up using only Facebook and Google for our login and signup, since time constraints did not allow us to further our authentication module.

We prioritized Facebook and Google because they are the most common and most important out of the ones we suggested, leaving out Github login and Twitter. Additionally, Twitter's Oauth API does not provide a primary e-mail address, which was not suited for our business logic, see the following forum thread for added reference: https://dev.twitter.com/discussions/15589.

For the same reason, we did not use the reCaptcha API, but worked hard to use Gravatar and Open Exchange Rates for bullet proof fairness in debt values. Working with other APIs was not an easy task but allowed us to gain respectable knowledge in OAuth authentication standard, implementing our own authentication in our node-js server.

Finally, time constraints also kept us from implementing email notifications, with an extra reason being the application serves the purpose of notifying the user.

The solution is described in the following document, starting with the general overview of features and following to the solution description.

All the elements of the group have contributed for the project in an equal fashion.

Application Overview

In our daily lives and in groups of friends or coworkers, it is easy to lose track of all the money that gets borrowed between people. For snack machines, coffee and even lunch, people often need to borrow some money. However, after a couple of "loans", its difficult to remember to who we owe money or who is in our debt.

Payback is a financial tool/application that allows the user to manage and collect their debts in an easy and intuitive fashion. With it, users can keep records of all personal debts and loans and easily transfer their debts to someone who owes the user money, keeping things well organized. It also facilitates splitting costs for expenses shared by groups of people, such as group meals or other activities, by allowing groups of users to take on a shared debt.

Main features

The Payback app offers the following features:

1. Borrow/lend money
2. Transactions have support for preset and custom currencies
3. View by debt direction or debtors
4. Automatic debt synchronization
5. Partial debt payments
6. Automatic debt management
7. Keep track of your friends and past debtors/creditors
8. Secure financial records with encrypted data
9. Allow group-owned debts for cost splitting

Web Services

3rd party web services (APIs)

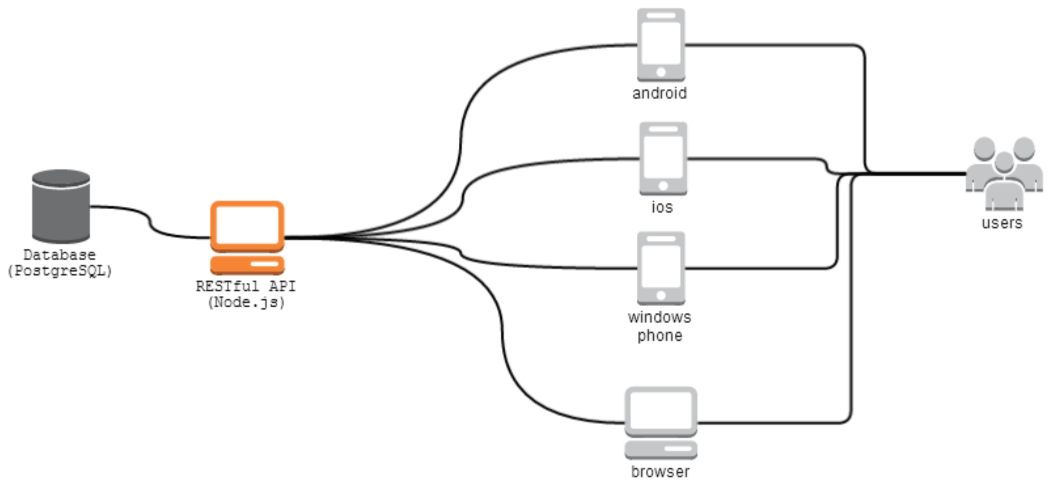
- Authentication providers ("*Login with..*")
    - Facebook
    - Google
- Others
    - Gravatar (User avatars)[1]
    - Fixer.io (Currency conversion)[2]

Provided services

[1] https://en.gravatar.com/
[2] http://fixer.io/

We provide a RESTful API server implemented in Node.js backed by a PostgreSQL database.

System architecture:



API specification: https://payback-app.herokuapp.com/api.html

Target Platforms

The web application can be deployed as a native application for iOS and Android thanks to Apache Cordova[3]. This build tool does not allow the deployment of the application for Windows Phone, however, the web application runs in the most recent browsers with no issues.

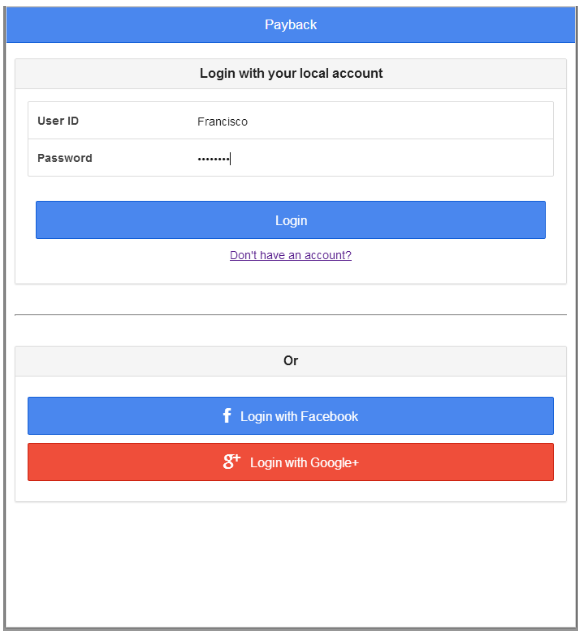
- Server
  - RESTful API server - Node.js
  - Database - PostgreSQL
- Client applications
  - Mobile applications
    - Android
    - iOS
  - Website

The mobile application

[3] http://cordova.apache.org/

The application was developed as a Single Page Application[4] using the Ionic Framework[5]. Ionic is an Hybrid application framework comprised of a variety of HTML/CSS components and widgets for the UI views. Additionally, this framework uses AngularJS as a MVC framework, decoupling the presentation logic from the backend models.

The application is fairly intuitive and simple. The user is initially exposed to the login and signup screens. The user can either login/signup locally or authenticate with his Facebook/Google account.



Payback

Login with your local account

User ID	Francisco
Password	••••••••

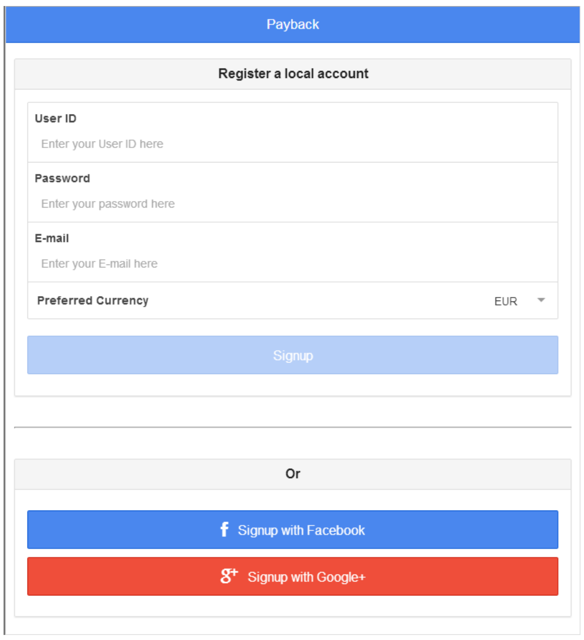
Login

Don't have an account?

Or

Login with Facebook

Login with Google+



Payback

Register a local account

User ID	Enter your User ID here
Password	Enter your password here
E-mail	Enter your E-mail here
Preferred Currency	EUR

Signup

Or

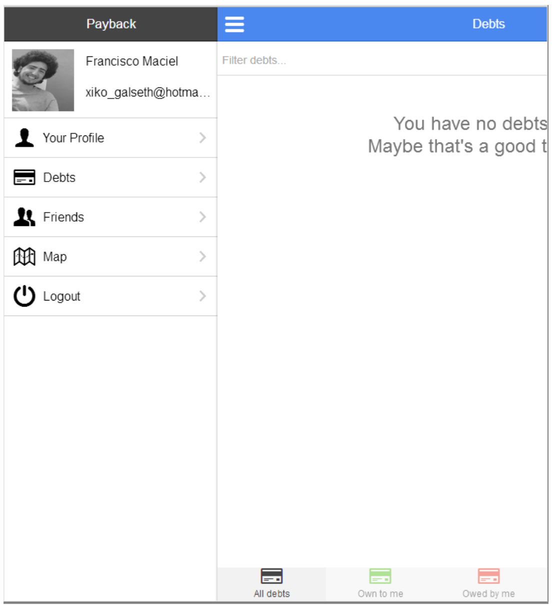
Signup with Facebook

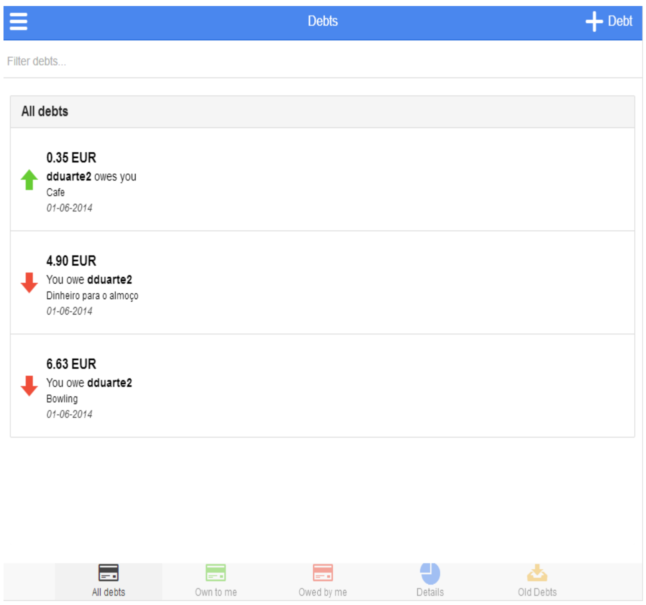
Signup with Google+

After logging in, the user can start navigating the application. The first panel is the debts panel, where the user has several tabs that correspond to his current debts and

[4] http://en.wikipedia.org/wiki/Single-page_application
[5] http://ionicframework.com/

detailed information. To navigate to other parts of the application, the user can quickly tap the top left corner to show the sidebar navigation menu:


Payback
Francisco Maciel
xiko_galseth@hotma…
Your Profile
Debts
Friends
Map
Logout
Debts
Filter debts...
You have no debts
Maybe that's a good t
All debts
Own to me
Owed by me


Debts
Debt
Filter debts...
All debts

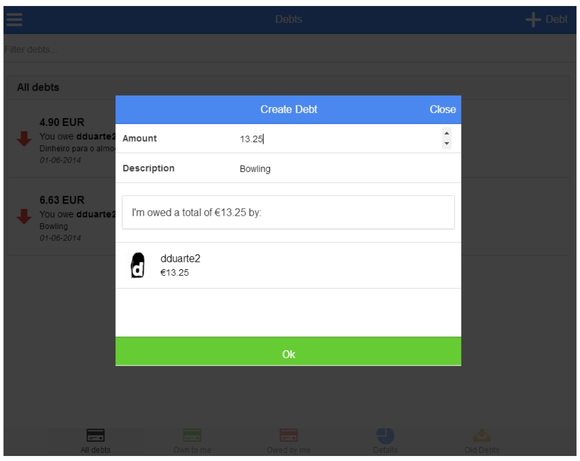
0.35 EUR
dduarte2 owes you
Cafe
01-06-2014

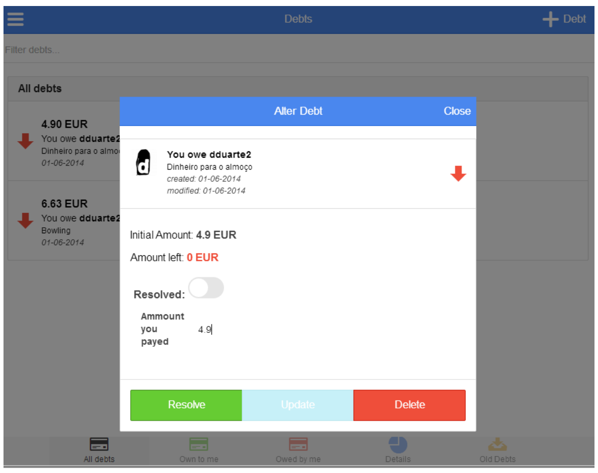
4.90 EUR
You owe dduarte2
Dinheiro para o almoço
01-06-2014

6.63 EUR
You owe dduarte2
Bowling
01-06-2014

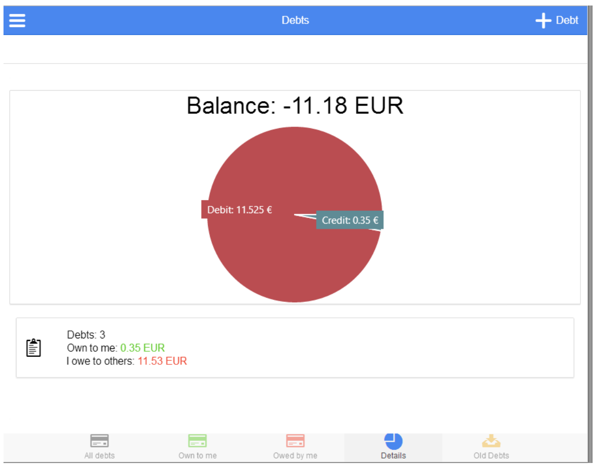
All debts
Own to me
Owed by me
Details
Old Debts

In the debts menu, the user can access create and update debts, and view the overall details and history:


Create Debt
Close
Amount 13.25
Description Bowling
I'm owed a total of €13.25 by:
dduarte2
€13.25
Ok


Alter Debt
Close
You owe dduarte2
Dinheiro para o almoço
created: 01-06-2014
modified: 01-06-2014
Initial Amount: 4.9 EUR
Amount left: 0 EUR
Resolved:
Ammount you payed 4.9
Resolve
Update
Delete

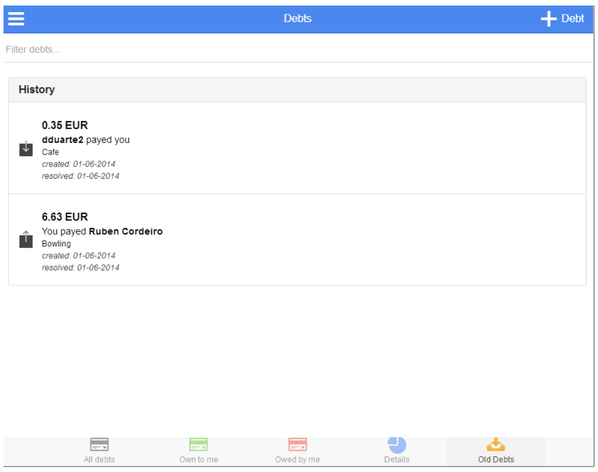
Example of switching tabs:


Debts

Debt

Balance: -11.18 EUR

Debts: 3
Own to me: 0.35 EUR
I owe to others: 11.53 EUR


Debts

Debt

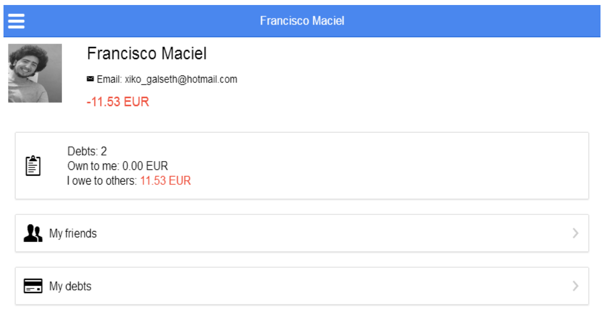
Filter debts...

History

0.35 EUR
dduarte2 payed you
Cafe
created: 01-06-2014
resolved: 01-06-2014

6.63 EUR
You payed **Ruben Cordeiro**
Bowling
created: 01-06-2014
resolved: 01-06-2014

The user can also access his own profile and list of friends:

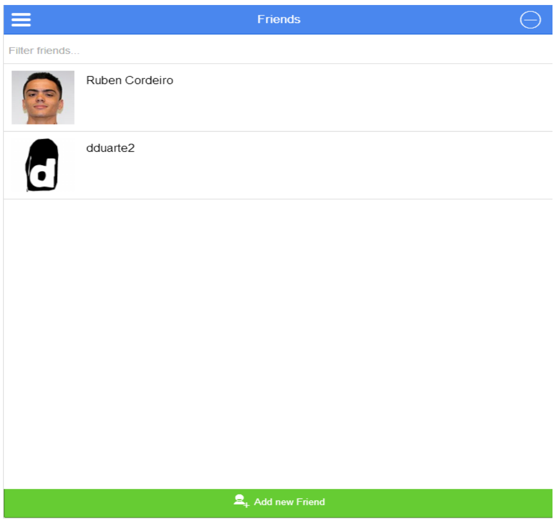

Francisco Maciel

Francisco Maciel

Email: xiko_galseth@hotmail.com

-11.53 EUR

Debts: 2
Own to me: 0.00 EUR
I owe to others: 11.53 EUR

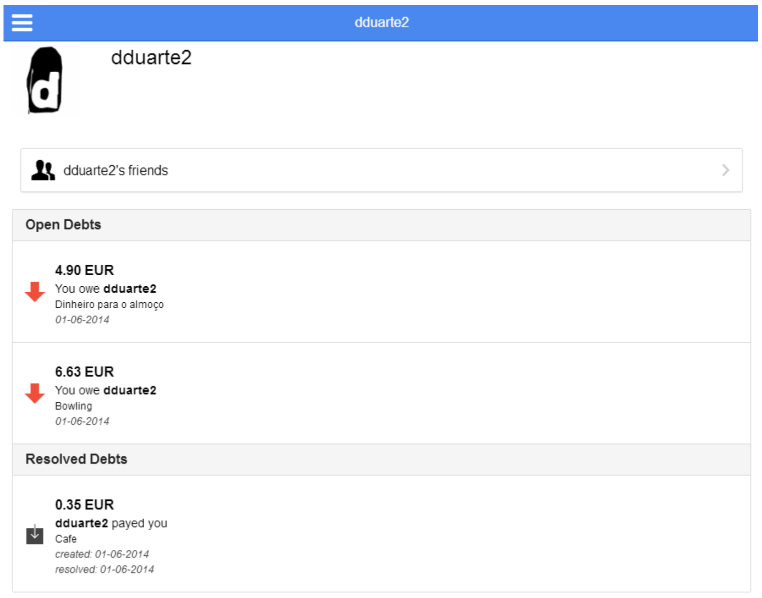

Friends

Filter friends...

Ruben Cordeiro

dduarte2

A user can tap a friend to view his/her profile, filtering the debts that concern both of them.



Open Debts

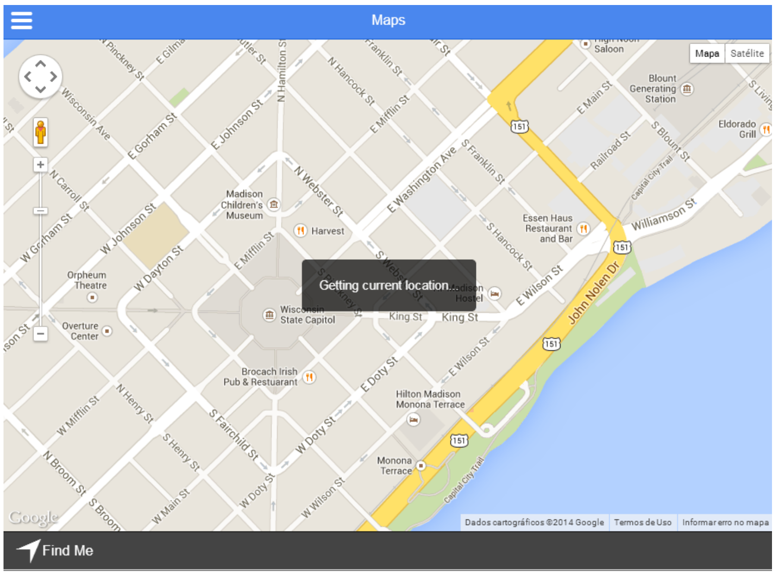
4.90 EUR
You owe **dduarte2**
Dinheiro para o almoço
01-06-2014

6.63 EUR
You owe **dduarte2**
Bowling
01-06-2014

Resolved Debts

0.35 EUR
dduarte2 payed you
Cafe
created: 01-06-2014
resolved: 01-06-2014

The application also provides a geolocation module. The “Map” tab allows the user to track his current location via the Google Maps API.



Specification

Back-end

One of the main development challenges was the development of a truly stateless REST API. Most of our previous experience in web development was fairly limited, since all of our previous solutions provided a simple server side authentication flow with user sessions.

With our REST API, the authentication endpoints return a unique access token for each user with a set expiration date. After the initial authentication flow, the access token gives the client access to the resources of the server.

The generation of the tokens follows the *Json Web Token* standard, a *JWT* is split into three parts, separated by periods. *JWT's* are URL-safe, although, in our implementation, all the access tokens are sent in the HTTP header of each request. The first part of a *JWT* is an encoded string representation of a simple JavaScript object which describes the token along with the hashing algorithm used. In our case, each token was generated with the hash of the authenticated user id and a client secret using the HMAC SHA-256 specification.

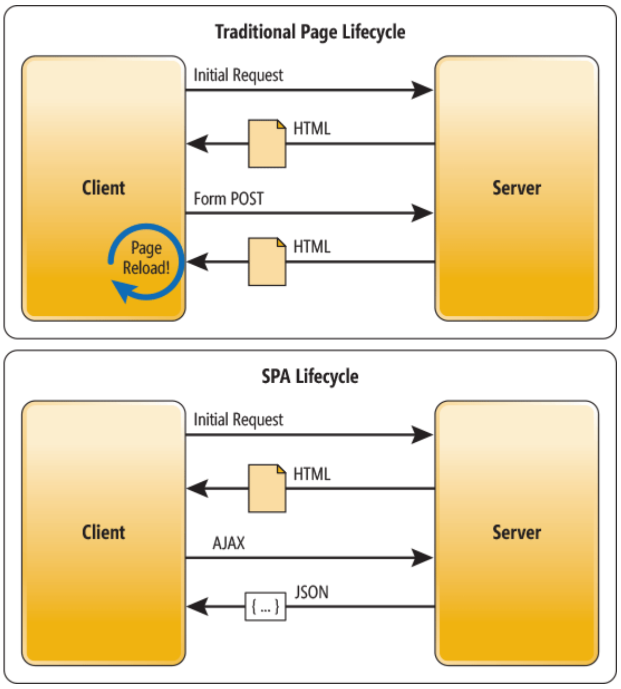
All the requests to the API server are made via HTTPS for added security. We also implemented a checksum to be required in every request using HMAC-SHA1 algorithm to verify the data integrity, however this feature is now disabled because we are serving the API through SSL/TLS, which already uses a checksum internally.

The exchange rates are provided by an external API, fixer.io, which is a free JSON API for foreign exchange rates, based on the daily feed of the European Central Bank. Our server acts as a proxy between the client and this external API. The exchange rates change every 24 hours at 3 A.M CET. Despite the 99.99% of uptime advertised by this API, the latest exchange rate values are cached in the server through a Nedb[6] (a subset of MongoDB) implementation. Additionally, a scheduler job is being run in the server in order to update the exchange rates.

[6] https://github.com/louischatriot/nedb

Front-end

Instead of serving static HTML pages, our web application is a single page application with multiple views and routes.



The average payload of each request was dramatically reduced, since, after the initial payload with all the HTML/CSS and Javascript assets, the client only exchanges *JSON* data with the server via the API calls.

The current user session data is safely encrypted and safely stored in the session cookies.

Conclusion

Looking back on our work, we are very satisfied with our final solution. We soon realized that we were overly ambitious in our proposal because we had to use and learn so many technologies, protocols and frameworks. Integrating Facebook and Google proved to be a good challenge.

Moreover, we believe that we definitely met the expectations for developing a complex Distributed System, and we learned from the development hiccups that arose from this project.

Additionally, we feel like we developed a fairly secure and efficient implementation, both server and client side. Node.js has proven to be quite fast.

In the future, we look forward to improving and publishing our application on the App Store (iOS) and Play Store (Android). We would very much like to see our application fully developed, using Paypal API to pay debts using our application, and completing our geolocation module to find nearby debtors.