Concept

1. Concept of the System's Economy

The system is based on points serving as a virtual currency. There is no exchange of real-world currency involved and no initial or recurring membership fee – signing up and participating is free of cost for everyone. Since people have a very clear notion of the value of money but will most likely have difficulties assigning a value to their work using an unfamiliar system of points, one point is intended to have roughly the same value as one unit of the respective country's real-world currency. This will make it easy for people to understand the points concept, come up with realistic price estimations for their own offers and requests and put other people's prices in relation to that.

Everybody who signs up freshly starts out with an account balance of 0 points. Both negative and positive balances are allowed only to a certain limit. This will avoid the system to establish massive imbalances where some people accumulate too much credit without spending enough and other people go into enormous debt by constantly taking from the community but not giving anything back to it. When a user first signs up, the initial limit for negative balances is 50 points and the limit for positive balances is 250 points. This means users could initially earn up to 250 points before they would be forced to also spend some of their points if they want to be able to generate more earnings again. On the other hand, they are initially allowed to spend only 50 points before they would be blocked on spending until they have given something back to the community. Setting the initial negative limit relatively tight is intended to avoid large-scale exploits of the community through people who sign up once to have somebody do something for them and then just silently keep their accounts without ever giving anything back to the community. While this might still happen, the 50 points limit for first-timers will ensure that such exploits remain small and can't harm the system in profound way. Active participants who regularly give to and take from the community in a relatively balanced manner will get their credit and debt limits gradually increased. Similarly to the concept of credit ratings for bank accounts, those users have proven their trustworthiness to the community and are rewarded by being allowed bigger scales of transactions.

There is no interest associated with account balances – this means that there is no benefit from saving points and also no disadvantage from being in debt. Having a negative account balance should not have the same negative connotation that it has in the real world. In face, negative account balances are completely acceptable and even necessary for the system to even out – nobody could ever have a positive balance if there weren't at least some people with a negative balance. The objective for every participant should be to balance out their account as good as they can, taking a negative account balance as an incentive to apply themselves more by doing something (perhaps actively browsing nearby requests and see if there is something that they could take on) while a positive account balance should motivate them to spend some of their points. If all accounts within the community would be summed up, the outcome will always be exactly 0 points.

The system is generally based on the assumption that every person will be able to contribute something to the community according to their individual talents, interests and capabilities. Some people might have very specific skills like knowing how to fix a car or design a business card while others might prefer to take on tasks that are rather time-based than skill-based such as doing somebody's grocery shopping or walking somebody's dog.

However, all this only applies to a world full of able-bodied, mentally and physically healthy people. This is far away from reality. In our society, and in our direct neighborhood, there are people who rely on the help of others without being able to give anything back other than love and gratitude. Gladly, in most cases these people are supported by a net of social services and health care. But the social and political institutions backing this system only work up to a certain extent. They will do what is necessary, but often enough there is no money to go any further than that. Especially older people who have little or no relatives to help and support them often fall into poverty and loneliness.

The platform has great potential to establish some structures to care for and support people in need. With every transaction that two users engange in, both of them will be asked if they want to donate a share of the agreed points to the needy. If a service would be compensated with 50 points, the performer of the service can chose whether they want to receive the full 50 points or whether they would be willing to receive a smaller amount, for example 48 points, with the difference going directly into a virtual donation pot. Simultaneously, the receiver of the service has the option of donating by increasing the sum booked from their account by one or more points. Incorporating the donation call into the transaction process makes a lot of sense as people are already spending or receiving points anyway and will be much more likely to say "Sure, one or two points for the good cause, why not?" than proactively making an unprompted donation.

The donation pot will be used to compensate people for performing tasks for people in need. The people performing such tasks may of course choose to do them for free, but offering them at least a small amount of compensation for their efforts could turn out to be a highly effective way to give people an incentive to help others. Obviously, this aspect of the community will require working closely together with social institutions and aid organizations to figure out where help is needed most. In the best-scenario case of the community growing into a flourishing economy with many transactions taking place every day, this could turn out to be a powerful contribution to relieving social issues.

1. Technical Concept
   1. Front-end

The final platform should be available both as a regular web version and as an app for smartphones. In an actual release version, both client sides (website and app) would be built in HTML5, CSS3 and JavaScript to fulfill the critierium of universal availability. Building the website in Flash could be problematic as it requires an additional browser plug-in that not all potential users might have installed. Especially less experienced users might not be able or willing to install anything in order to view the site. Furthermore, Flash is not supported by iOS devices and therefore there would be no way of accessing the website from an iPad. HTML5 has become the de-facto standard technology for building modern websites and will ensure that the platform is accessible by a broad audience.

The final mobile app is intended to be a hybrid solution, using PhoneGap as a wrapper to deploy a standalone app. Choosing a hybrid over a native or web-based approach is also attributed to the criterium of universal availability. The app should be available to all smartphone users, regardless of their operating system. To achieve this with a native solution would require the app to be developed from the ground up for every supported platform, resulting in significantly higher costs of initial development and maintenance. While a web app would be instantly available to all web-enabled devices, it lacks the ability to access some essential device capabilities that may be important for the app, for instance the possibility to send push notifications to the user. It would also be harder for users to access the app, as they have to go through their browser and type in the address (or use a bookmark) every time they want to use it. All of the presented issues can be solved by a hybrid approach: It reduces the costs of development and maintencane, even enables re-use of some of the code that is used for building the website, it offers an interface to access specific device features and it makes the app discoverable in app stores and market places and easily be accessed through the device's home screen.

The decision of choosing PhoneGap over similar frameworks like Appcelerator Titanium and Adobe AIR is based on the fact that apps developed with PhoneGap can utilize HTML5/CSS3 standards and access native device capabilities through a JavaScript SDK. Appcelerator Titanium uses a different approach: Using the Titanium JavaScript SDK, the entire application will be written in JavaScript, which will be compiled down to native code for deployment. This accounts for better performance, but is usually only relevant and noticable for highly CPU-intensive apps. Apps developed with Adobe AIR use Flash an dActionScript 3 as the underlying technologies. Using PhoneGap has the clear advantage of allowing the developer to re-use code across the desktop web platform and the mobile app and would therefore be the framework of choice.

* 1. Back-end

The back-end should be designed for high security and easy scalability. Known as one of the most secure operating systems on the market, OpenBSD will serve as the web server's underlying platform. Apache will be the web server of choice as it is highly configurable and proven as a viable solution in many large-scale projects. Coming with a free licence and being known for outstanding reliability and consistency, the platform's database will be PostgreSQL. The backend API will be built in Python using the Django framework. Due to its clean design and syntax, Python is known to encourage developers to write efficient code that is easy to read and maintain. Therefore it is an excellent language to develop a flexible and highly scalable back-end that allows for quick changes without having to recode large parts of the system.

In order to minimize bugs and hence also minimize security threats, the development process should include thorough unit testing. Once the system is live, extensively monitoring and analyzing traffic and performance (by using Nagios or a similar tool) will be crucial to identifying potential bottlenecks and predict when it will be necessary to take steps to scale up the system. Depending on where the bottleneck is located, appropriate measures to cope with increasing server load can be upgrading hardware (scaling up) or creating a cluster by adding more web or database server nodes and a load balancer to distribute requests (scaling out).

* 1. Prototype and Design

In the scope of this thesis, a prototype version of the app will be developed to demonstrate the basic functionality as well as the look and feel of the platform's mobile version. There are many different approaches to prototyping, the two main categories of which being rapid prototyping and evolutionary prototyping. In rapid prototyping (also called "throwaway prototyping"), the goal is to quickly produce a simple working version to gather feedback on the requirements and functionality. The code will eventually be fully discarded and will not become part of the codebase for the actual release product. In evolutionary prototyping, the core of a system is developed thoroughly to outline the most basic functionality. The system will then be constantly refined and extended with new features. The difference here is that the prototype code will eventually turn into the final system instead of being thrown away.

As the thematic priority of this project is set on the design of user interface and user experience, a rapid prototyping approach focussing on front-end development and usability optimization will be applied. In order to save the time and effort that it would undoubtedly take to build a functioning back-end while still being able work with real data instead of just mock objects, the app will hook into a back-end based on Parse, a BaaS provider that offers an extensive free plan. Using this solution, a full data model will be created with the associated database hosted in the cloud. Adding, modifying and deleting data as well as running queries to fetch data will be possible using a RESTful API provided by Parse.

As outlined above, the rapid prototyping approach will cause the prototype to be discarded before development on the actual production version commences. Since the prototype is intended to demonstrate the system's functionality regardless of the underlying technology, it is acceptable and even advisable to develop the prototype using a language that is different from the one chosen for the production version. This will prevent developers to give into the temptation of re-using chunks of code from the prototype for the actual development and ensure that the prototype will really be "thrown away".

The prototype for this project will be developed as a native iOS application using Objective-C and the Cocoa Touch framework. This will allow for a relatively fast generation of visible and functional output without the need of tying together various frameworks to support different platforms. The main purpose of the prototype is to demonstrate the overall interaction concept and show how the application will function and feel from a user perspective. It will implement core functionality and set its focus on creating a rich and intuitive user experience. Internal aspects that would be crucial to an actual production version but are mostly invisible to the user, such as performance optimization, thorough exception handling or security measures will be largely disregarded. The goal is to create a prototype that is functional and polished to the point of being ready to be handed to a group of test users and having them verify that the basic idea is conveyed in a clear way, that the app is easy and fun to use and its underlying core user flows are coherent. With the prototype, test users will be able to create their own accounts and log into the system, browse through existing offers and requests or make specific searches, contact and be contacted by other users to engange in a deal with them and create, edit or delete offers and requests of their own.

In addition to the mobile app prototype, the layout and visual design for the web platform will be drafted in the form of wireframes with some sample screens being designed in greater detail to demonstrate what the web version will look like and how it will work without actually implementing any functionality. All graphics and assets required for the mobile app as well as the screen designs for the website will be created using Adobe Photoshop and Illustrator.