WalleTx

Software Research Specification

Prestige Worldwide

Group Members:
Arron Solano
Brian Howell
Daniel Carroll
Michael Danko

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to define the requirements for Bitcoin WalleTx. The intended audience of this document includes the CEN4021 instructional staff at FSU and the end users of the WalleTx. It is also intended that this document serve as the principle point of reference for the Prestige Worldwide team throughout the development of WalleTx.

Bitcoin WalleTx (WalleTx) is a bitcoin wallet tracker tool for Android that assists users in monitoring their bitcoin balance, transaction history, and spending trends across multiple wallets. It is common for bitcoin users to possess numerous wallets, yet there do not exist many services that are capable of aggregating wallet information in order to provide the user with an overall picture of their bitcoin funds. Bitcoin WalleTx solves this problem by allowing users to group or categorize their bitcoin wallets, as well as tag their transactions with real-world information related to how their bitcoins are being spent. Charts and graphs help to identify trends in both single wallet spending and across wallet groups, a useful feature that is essential for enabling users to integrate their bitcoin finances with a more traditional budget.

1.2 Scope

1.2.1 Product

Bitcoin WalleTx (referred to as WalleTx)

1.2.2 Scope of Work

WalleTx is an android application that enables users to import their bitcoin wallets by public key and tag their bitcoin transactions in order to obtain an aggregated overview of their bitcoin finances. WalleTx shall allow users to manage their public keys; manage wallet groups; manage transaction tags; tag individual transactions; and receive feedback regarding trends via charts and graphs. A full listing of functional requirements for WalleTx is located in the Functional Requirements section of this document.

1.2.3 Out of Scope

WalleTx shall not provide any functionality allowing users to spend or receive bitcoins since only public keys shall be stored on the device of the user.

1.2.4 Application

Bitcoin is a new technology and the ecosystem is currently undergoing major infrastructure development. There currently do not exist any tools that allow users to label their bitcoin transactions in order to identify trends in their bitcoin spending behavior. WalleTx aims to bring this functionality to the Bitcoin space.

1.3 Definitions

- **WalleTx** - Shorthand for Bitcoin WalleTx
- **Bitcoin** - a tradeable virtual asset existing on the ledgers of a distributed and synchronized network of ledgers
- **bitcoin** - a tradeable virtual asset unit, divisible into 100 million units (also known as a Satoshi)
- **Blockchain** - A blockchain is made of a blocks, linked to the block before and block after, that contain information pertinent to the underlying system. In the case of Bitcoin these blocks contain transaction information.
- **Public key** - A hash of a wallet's public key that allows other users on the Bitcoin network to send bitcoins to the Public Key.
- **Tx** - A transaction that moves bitcoins from one address to another, usually in exchange for goods or services.

1.4 Acronyms

BTC Common unit of Bitcoin currency

1.5 References

- Bitcoin: A Peer-to-Peer Electronic Cash System
  `https://bitcoin.org/bitcoin.pdf`

- Blockchain Data API
  `https://blockchain.info/api/blockchain_api`

1.6 Overview

Description of the remainder of the SRS. The remainder of the document outlines the functionality and operation of the WalleTx software. It outlines a general description of who, what, why, when, and how individuals will use our software. It provides the functional and non-functional constraints our software will incur. The document will also serve to illustrate software functionality from a high level architectural view to minute details about every user scenario the software may encounter. The appendix provides information and graphs pertinent to many of these details.

2 General Description

WalleTx is an internet connected, bitcoin connected software that is used as a financial management tool. With this software you are able to review a history of spending with the option to categorize individual transactions.

The bitcoin network is a decentralized peer to peer network of computer systems that utilizes proof of work to create and maintain a database of virtual coins. Transferring these coins is done by authenticating with the system utilizing a type of public/private key authentication. The public key can also be used by other users in the network to as an address if other users with to 'send' you bitcoins. There is no physical ownership of bitcoins, it is merely the ability to sign transactions with your private keys that enable you to control and send bitcoins to other users. The database of transactions between bitcoin accounts is maintained in a database known as a blockchain. The blockchain contains the history of every bitcoin transaction from day 0 to today. Bitcoin miners are responsible for updating the database. On average, a block is mined every 10 minutes and any transactions that happen in that 10 minutes become added to the blockchain forever. In order to mine a block the miners must brute force reverse a hashing problem, in return for their brute force hashing, energy consumption, and proof of work the miner is rewarded with a 25 bitcoins.

2.1 Product Perspective

WalleTx is dependent on the bitcoin blockchain to source information regarding transaction amounts, times, and dates. It is possible that manual data entry can be implemented for those transactions however it is assumed the blockchain access will be available at all times. If there is not internet, you cannot spend your bitcoin. Accessing this software will be completed through mobile smartphones with Android operating systems.

2.2 Product Functions

Product functions include accessing the bitcoin blockchain to download transactions, allowing users to tag and categorize transactions into which categories they wish. And then to analyze data by association to come to their own conclusions with regard to spending habits. Users will be able to add multiple wallets to allow for spending and saving 'wallets'.

2.3 User Characteristics

Users are from a broad spectrum of every day life. It is WalleTx's intention to be available to people of all ages from every corner of the globe. Bitcoin is a global network, with many users from young to old and WalleTx will be usable by the same demographic. Anyone who is capable of utilizing bitcoin technology to transact will be able to use WalleTx to manage their financial records.

2.4 General Constraints

- Regulatory policies - the software currently reads data from a publicly available database known as the blockchain. The software does not transmit or receive bitcoin transactions and is therefore not subject to any monetary regulatory policy.
- Hardware limitations; for example, signal timing requirements - there are no hardware limitations aside from being used on a smartphone with the Android Operating System
- Interface to other applications - the software will interface with other API's to integrate with the bitcoin blockchain to download information.

- Signal handshake protocols - communication with third party applications will take place over TCP/IP.
- Criticality of the application - the application will hold data that is critical to the user and their personal financial planning. Loss of functionality will not result in a loss of access to their funds.
- Safety and security considerations - a loss of the smartphone or exposure of data will result in persons being able to associate a bitcoin address with an individual and potentially exposing their financial history. Loss will not result in a loss of funds.

2.5 Assumptions and Dependencies

The assumption is that the application will be used on a mobile device. The next assumption is that the mobile device is utilizing the Android operating system. At this time the application will only be able to run on the Android operating system.

3 Functional Requirements

3.1 The system shall allow users to manage their imported wallets or public keys

3.1.1. Users shall be able to add new single address wallets into the system

3.1.2. Users shall be able to edit previously existing wallets

3.1.3. Users shall be able to delete existing wallets from the system

3.1.4. Users shall be able to name individual wallets and assign them to a group

3.1.5. The system should be designed to allow for future integration with other wallet types such as deterministic seed wallets or service based wallets such as Coinbase and Circle

3.2 The system shall allow users to customize wallet groups

3.2.1. System shall designate a default wallet group

3.2.2. Users shall be able to add new wallet groups by name

3.2.3. Users shall be able to edit previously existing group names

3.2.4. Users shall be able to delete existing wallet groups, an action that should reassign all wallets associated with the group to the default group

3.3 The system shall allow users to manage transaction tag categories

3.3.1. User shall be able to define new tag categories by name

3.3.2. User shall be able to edit existing tag categories

3.3.3. User shall be able to delete existing tag categories, an action that should remove the tag from any transactions it has been assigned to

3.4 The system shall allow users to manage tags for individual transactions

3.4.1. User shall be able to add a tag to an individual transaction

3.4.2. User shall be able to remove a tag from an individual transaction

3.5 System shall provide an aggregate view of all imported wallets that contains the total balance and number of transactions

3.6 System shall provide a summary view for each wallet group that provides an overview of the group balance, transactions, and charts

3.7 System shall provide a summary view for individual wallets that provides an overall summary of its balance, transactions, and charts

3.8 System shall provide a transactions detail view for each wallet group that lists the transactions associated with the group

3.9 System shall provide a transactions detail view for each individual wallet that lists its transactions

3.10 System shall provide graphical charts for identifying transaction trends

3.10.1. System shall provide pie charts views breaking down transaction percentage by tag for all wallet groups and for individual wallets

3.10.2. System should provide additional trend identifying charts for wallet groups and individual wallets

3.11 The system should provide a view containing individual transaction information

3.12 The system should be able to sync with the blockchain every 10 minutes to verify new blocks (possibly use web socket)

4 Non-Functional Requirements

4.1 Query and reporting times - this will be dependent on the end users RAM and processor speed as well as database placement and architecture. Must account for moderate processor speeds and run on little as RAM as possible. Android Framework provides tools to test and profile CPU and RAM usage. Querying

the blockchain will be done via the Blockchain API and speed to return a query will be dependent on outside services and network connection speed.

4.2 Storage - this will be dependent on the end users storage memory on the phone. Must have small footprint on phone's storage. Need to compare application size for similar available applications to scope size. Records will be stored locally to prevent repeated connections to the blockchain network for duplication information. Records and their added tagging information will probably be measured in hundreds of bytes meaning that the storage footprint will be very reasonable.

4.3 Response time (between activities) - this will be dependent on the end users RAM and design of UI. Must account for low RAM and optimize UI for quick activity. This evaluation will fall under query and reporting time.

4.4 Screen Resolution - dependent on end users phone. Must find resolution to work on most phones. (font, layout tweaks, image changes)

4.5 Versions - Need to specify and target revision to reach optimal user targets. Evaluated by most popular revision available or with best compared resources.

4.6 No system downtime - All updates to the app will be done through Android updates pushed through Google Play Store. Standard update process for life of application on Android platform.

4.7 Battery Usage - this will be dependent on the end users battery size, installed applications, and screen size. Must try to minimize battery drain. Need to evaluate against controls to test battery usage.

5 System Architecture

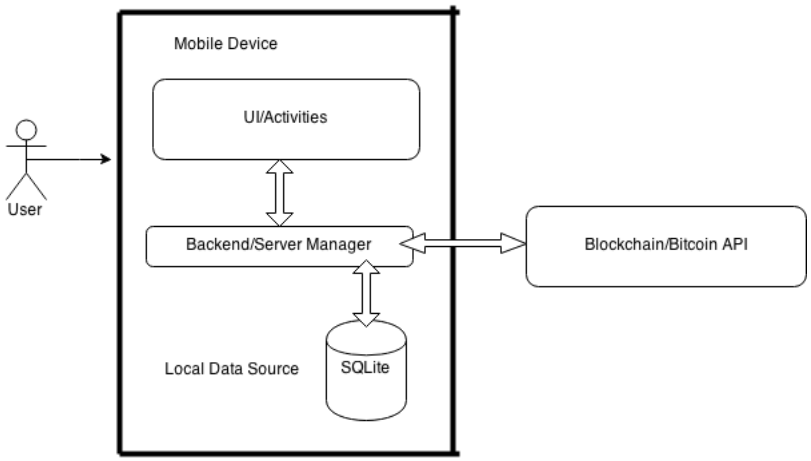


Figure 1: System Architecture UML

Architecture Overview:
The application will be accessed and stored on an Android phone. The user will interact with application using touch screen.

- Environment is mobile, can be used anywhere
- Users will have existing bitcoin address(es)
- Main functions consist of storing and organizing bitcoin wallet(s) information and displaying data
- Input is users bitcoin address(es), output is data organization and customization

Logical Overview:

- Utilizing Android OS on mobile phones (minimum Android SDK v4.0 Ice Cream Sandwich), developing in Android SDK framework using JAVA
- Data storage will be local, utilizing ActiveAndroid ORM

- Data gathered form bitcoin address(es) using blockchain and bitcoin API
- IDE will Google's Android Studio

UI/Activities:

- Will be designed using Android SDK standards in XML mark-up
- Purpose provides interaction with end user
- Interactions include end user input and data binding with back-end/server manager

Back-end/Server manager:

- Will be designed using Android SDK standards in Java
- Purpose provides interaction between database, blockchain, bitcoin API, and UI/Activities
- Data binding will occur in this area
- Interactions include data management, activities controller, and external data management

ActiveAndroid ORM:

- Will be designed using ActiveAndroid ORM utilizing sql queries and statements
- Purpose provides data storage and management for the application
- Interactions include Back-end/Server communication for queries and storage

Blockchain/Bitcoin:

- Bitcoin interaction in application will be limited to API interaction and blockchain interaction
- Will not need to be designed. Will be managed by Back-end/Server and BitcoinJ libraries.
- Purpose is to provide data and information from blockchain and user's existing bitcoin account(s)
- Interactions will occur with designated API services.

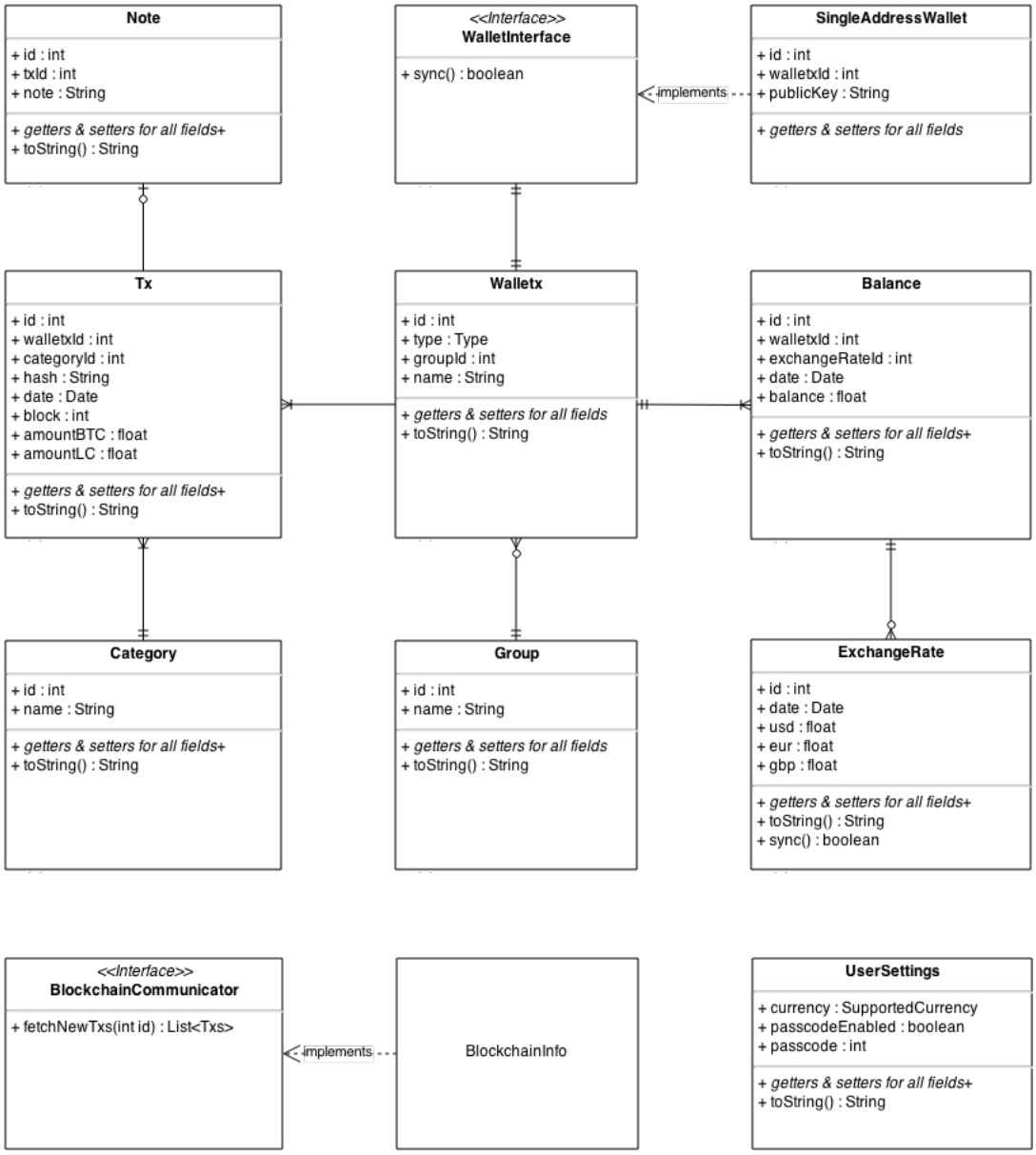


Figure 2: Class Models

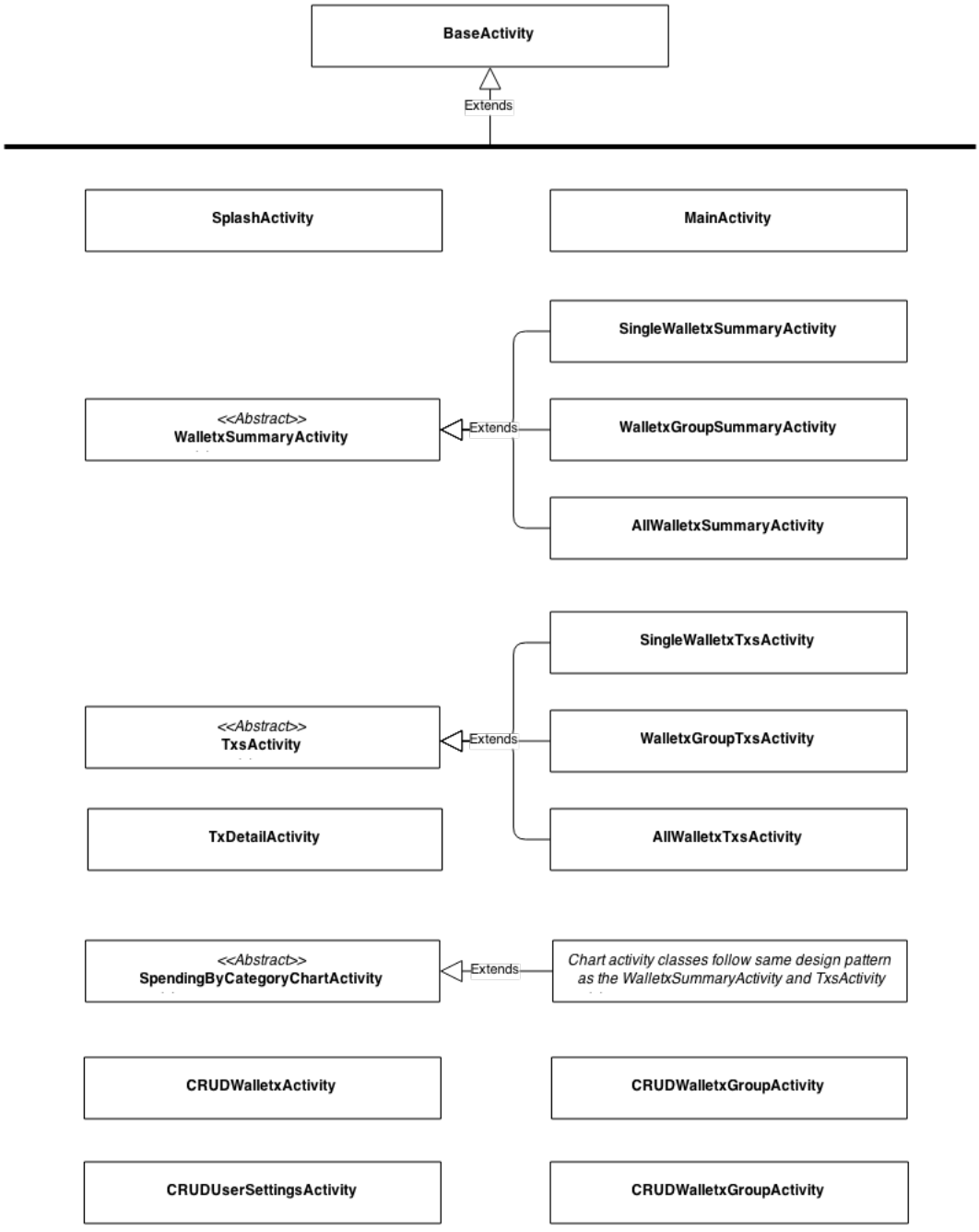


Figure 3: Class Activities

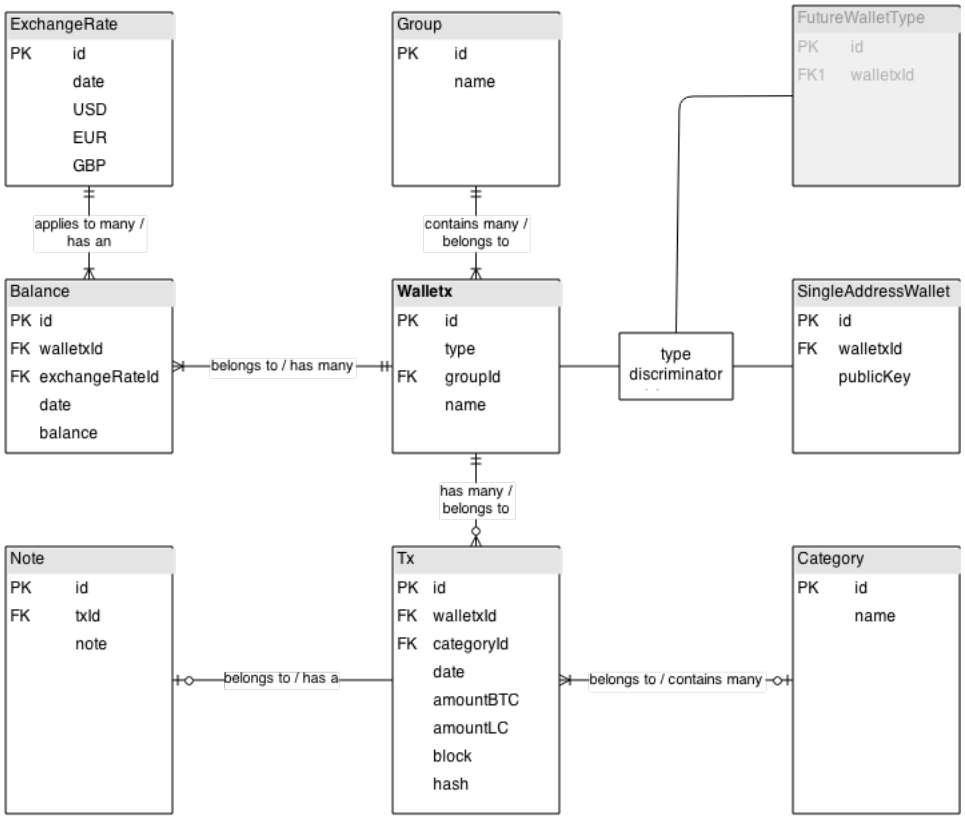


Figure 4: Database Schema

6 System Model

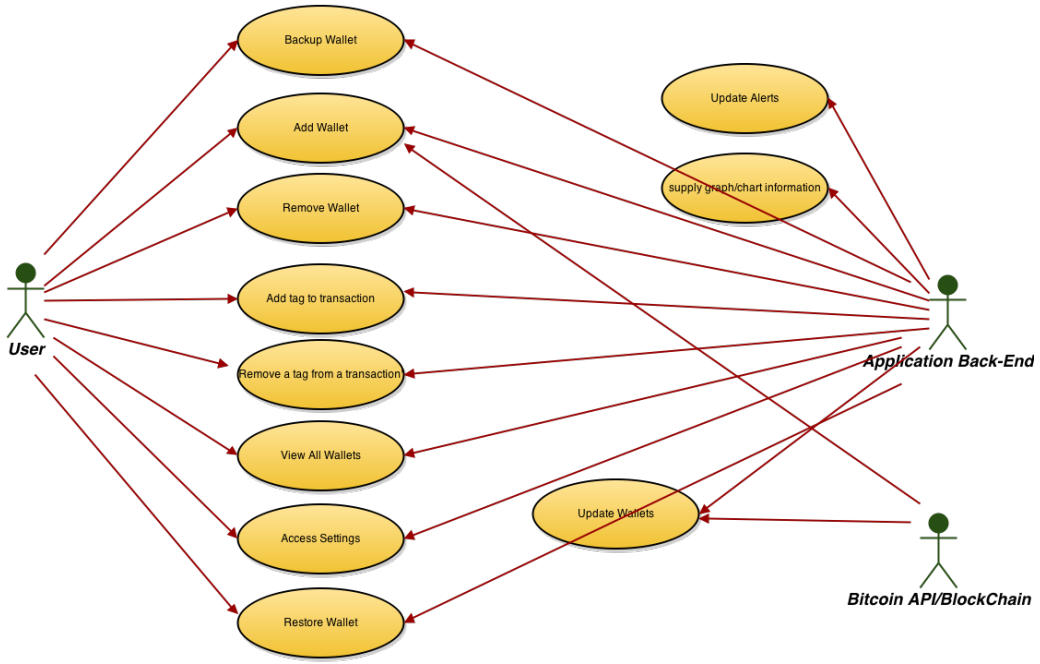


Figure 5: Master Use Case Diagram

7 Appendices

7.1 Data Dictionary

7.1.1 Actor Descriptions

App users will be the primary actors of the application and they will generally fit into the same category. Users will want to interface with the application to allow it to pull their transaction history from the blockchain so that they can categorize their data. Professional financial management entities could conceivably use data created from our software and if time permits we will implement data formatting to work in conjunction with popular financial planning systems and software.

Users will supply their information, such as bitcoin addresses and spending categories, this will allow them to graph their data, maintain balances, and users will also be able to delete all of their own data.

Individuals who own and spend bitcoin will be most interested in using our app. Financial management and planning is an important part of financial responsibility and a ledger to report spending habits will be of use to many members of the bitcoin community.

Support and maintenance of the system is mostly the users responsibility. Maintaining the hardware is the responsibility of the user, maintaining the operating system and related libraries will be the responsibility of Google and the phone manufacturer. Maintaining the software will primarily be done by Prestige Worldwide while any sub libraries and related software support will be maintained by their software development teams.

The system will use blockchain APIs provided by a number of providers. Many of the APIs are free and there is even an option to run our own server, utilizing open source software to interface with the blockchain and provide API services to our app users. Final design and implementation will be decided on time to implement and performance to the user metric.

No other systems will be dependent on our app. Therefore no functionality is critical for any downstream applications.

7.1.2 Use Case Descriptions

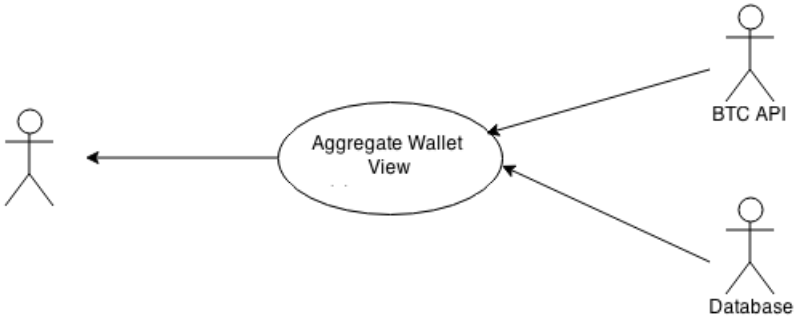


Figure 7.6: Managing User Wallets

WalleTx: Aggregate Wallet View	
Actors	User, SQLite Database, Blockchain API
Description	User will view an aggregate view of all imported wallets that contains the total balance and number of transactions
Data	all imported wallets, transaction hashes, and total balance of combined wallets
Stimulus	User initiated option
Response	Summary view for all wallets will be displayed showing a list of all imported wallets, total balance and number of transactions.
Comments	
Figure 7.7	

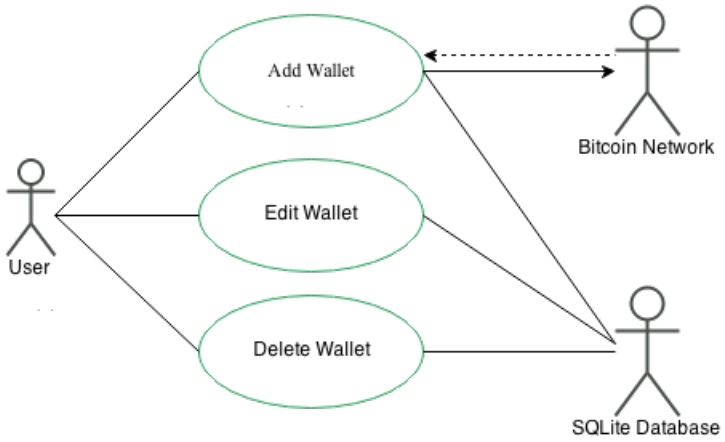


Figure 7.7: Managing User Wallets

WalleTx: Add Wallet	
Actors	User, SQLite Database, Blockchain API
Description	User will add an existing wallet public key to the application to enable mapping of transactions to the wallet public address
Data	public wallet address, transaction hashes, tags associate with transactions
Stimulus	User initiated option
Response	Additional view will open up allowing the user to manual type their wallet address, paste from the clipboard, or scan a qr code. After a valid bitcoin public address is entered the application will contact the Blockchain.info API to pull transaction information putting that information into the database. View will return to the main wallet view.
Comments	
Figure 7.8	

WalleTx: Modify Wallet	
Actors	User, SQLite Database
Description	User will be able to edit the name of the wallet.
Data	Public Wallet Address associated with database entries
Stimulus	User initiated option
Response	View will open allowing the user to change the name of the wallet.
Comments	Changing the wallet address would only serve to change all the underlying transactions, therefore it may be a better idea to only allow adding or removing public wallet addresses to manipulate public wallet address data in the database. Other features not implemented by the Bitcoin blockchain, such as currency pair, wallet nickname, etc. can be modified here.

Figure 7.8

WalleTx: Remove Wallet	
Actors	User, SQLite Database
Description	User will remove an existing wallet public key from the application and have corresponding data removed from the SQLite database.
Data	public wallet address
Stimulus	User initiated option
Response	Dialog will pop up allowing the user to confirm (possibly have a text input required to continue) and when confirmed the data will be purged from the database.
Comments	

Figure 7.8

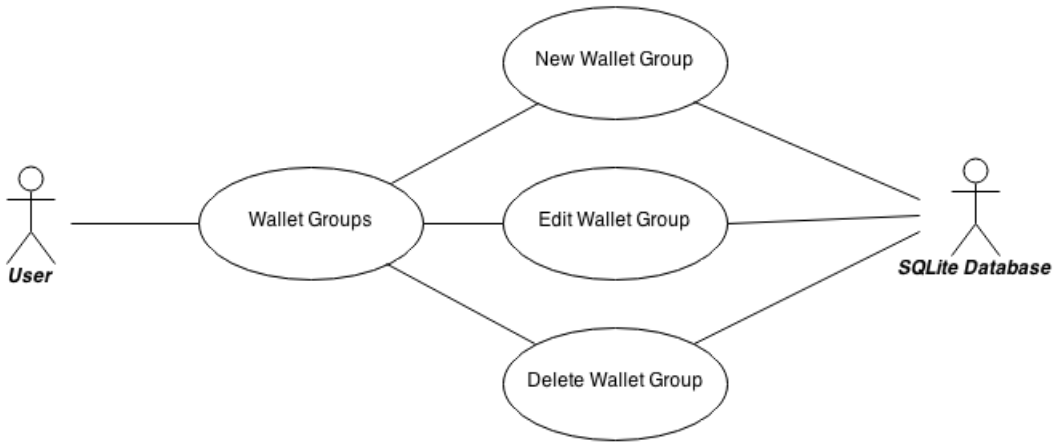


Figure 7.8: Managing Wallet Groups

WalleTx: View all transaction tag categories	
Actors	User, SQLite Database
Description	User is presented with a list view of wallet groups with options to: (1) Add a new wallet group, (2) Edit a wallet group, and (3) Delete a wallet group
Data	Wallet group categories
Stimulus	User initiated option. User selects Wallet Groups from the menu
Response	Application opens CRUDGroupActivity and populates the list view with all the groups in the group table
Comments	This is entry point for user to add, edit and delete wallet groups
Figure 7.9	

WalleTx: Add wallet group category	
Actors	User, SQLite Database
Description	User adds a new wallet group (by name) to the application
Data	Wallet group name
Stimulus	User initiated option
Response	A dialog will open containing: (1) a text field for entering the new wallet group, (2) a Cancel button, (3) and an Add button. If Add is selected, user entered wallet group name is inserted into the Wallet Group table and user is returned to the CRUDGroupActivity. If cancel is selected, user is returned to the CRUDGroupActivity.
Comments	Response should validate for empty wallet group name. CRUD-GroupActivity UI must be updated upon successful insertion of a new tag category
Figure 7.9	

WalleTx: Edit wallet group category	
Actors	User, SQLite Database
Description	User edits an existing wallet group
Data	Wallet group name
Stimulus	User initiated option. User selects edit button from wallet group view.
Response	A dialog will open containing: (1) an edit text field for updating wallet group, (2) a Cancel button, (3) and an Edit button. If Edit is selected, the wallet group is updated in the Wallet Group table and user is returned to the CRUDGroupActivity. If cancel is selected, user is returned to the CRUDGroupActivity.
Comments	Response should validate for empty wallet group name. CRUD-GroupActivity UI must be updated upon successful update of a tag category
Figure 7.9	

WalleTx: Delete Wallet Group	
Actors	User, SQLite Database
Description	User deletes an existing wallet group
Data	Wallet group name
Stimulus	User initiated option. User selects delete button from wallet group list view.
Response	A dialog will open confirming if user wishes to delete the wallet group. If confirmed, the wallet group must be removed then reassign all wallets associated with deleted group to the default group and then removed from the Wallet Group table. User is then returned to the CRUDGroupActivity.
Comments	CRUDGroupActivity UI must be updated upon successful update of the wallet groups
Figure 7.9	

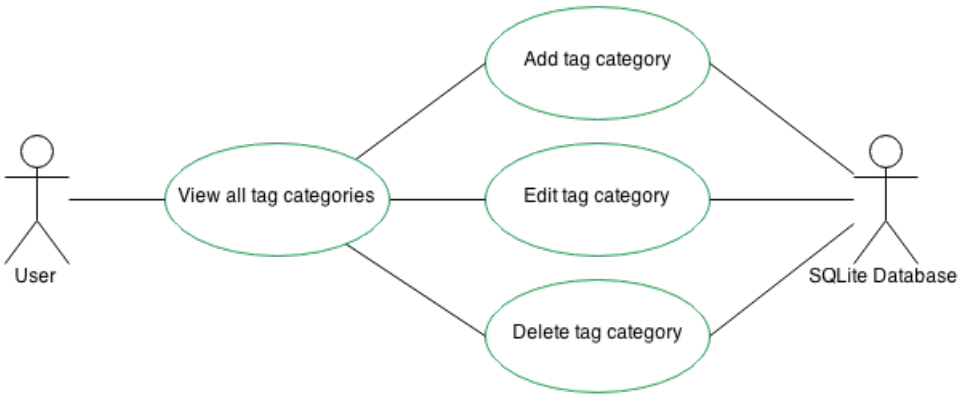


Figure 7.9: Managing Transaction Tag Categories

WalleTx: View all transaction tag categories	
Actors	User, SQLite Database
Description	User is presented with a list view of all tags with options to: (1) Add a new tag, (2) Edit a tag, and (3) Delete a tag
Data	Tags categories
Stimulus	User initiated option. User selects Tx Tags from the menu
Response	Application opens CRUDTagsActivity and populates the list view with all tags in the Tag table
Comments	This is entry point for user to add, edit and delete tags
Figure 7.10	

WalleTx: Add transaction tag category	
Actors	User, SQLite Database
Description	User adds a new tag category (by name) to the application
Data	Tag category name
Stimulus	User initiated option
Response	A dialog will open containing: (1) a text field for entering the new tag category, (2) a Cancel button, (3) and an Add button. If Add is selected, user entered tag category is inserted into the Tag table and user is returned to the CRUDTagsActivity. If cancel is selected, user is returned to the CRUDTagsActivity.
Comments	Response should validate for empty tag name. CRUDTagsActivity UI must be updated upon successful insertion of a new tag category
Figure 7.10	

WalleTx: Edit transaction tag category	
Actors	User, SQLite Database
Description	User edits an existing tag category
Data	Tag category name
Stimulus	User initiated option. User selects edit button from tags list view.
Response	A dialog will open containing: (1) an edit text field for updating tag category, (2) a Cancel button, (3) and an Edit button. If Edit is selected, the tag is updated in the Tag table and user is returned to the CRUDTagsActivity. If cancel is selected, user is returned to the CRUDTagsActivity.
Comments	Response should validate for empty tag name. CRUDTagsActivity UI must be updated upon successful update of a tag category
Figure 7.10	

WalleTx: Delete transaction tag category	
Actors	User, SQLite Database
Description	User deletes an existing tag category
Data	Tag category name
Stimulus	User initiated option. User selects delete button from tags list view.
Response	A dialog will open confirming if user wishes to delete the tag. If confirmed, the tag must be removed from any transactions onto which it is applied and then removed from the Tags table. User is then returned to the CRUDTagsActivity.
Comments	CRUDTagsActivity UI must be updated upon successful update of a tag category
Figure 7.10	

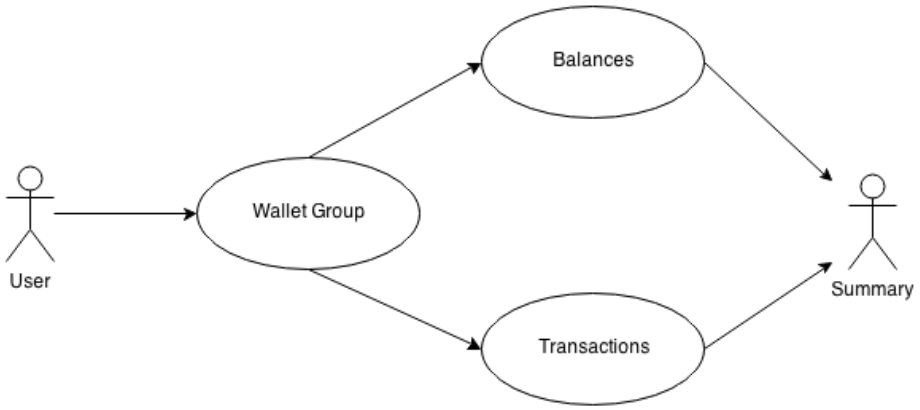


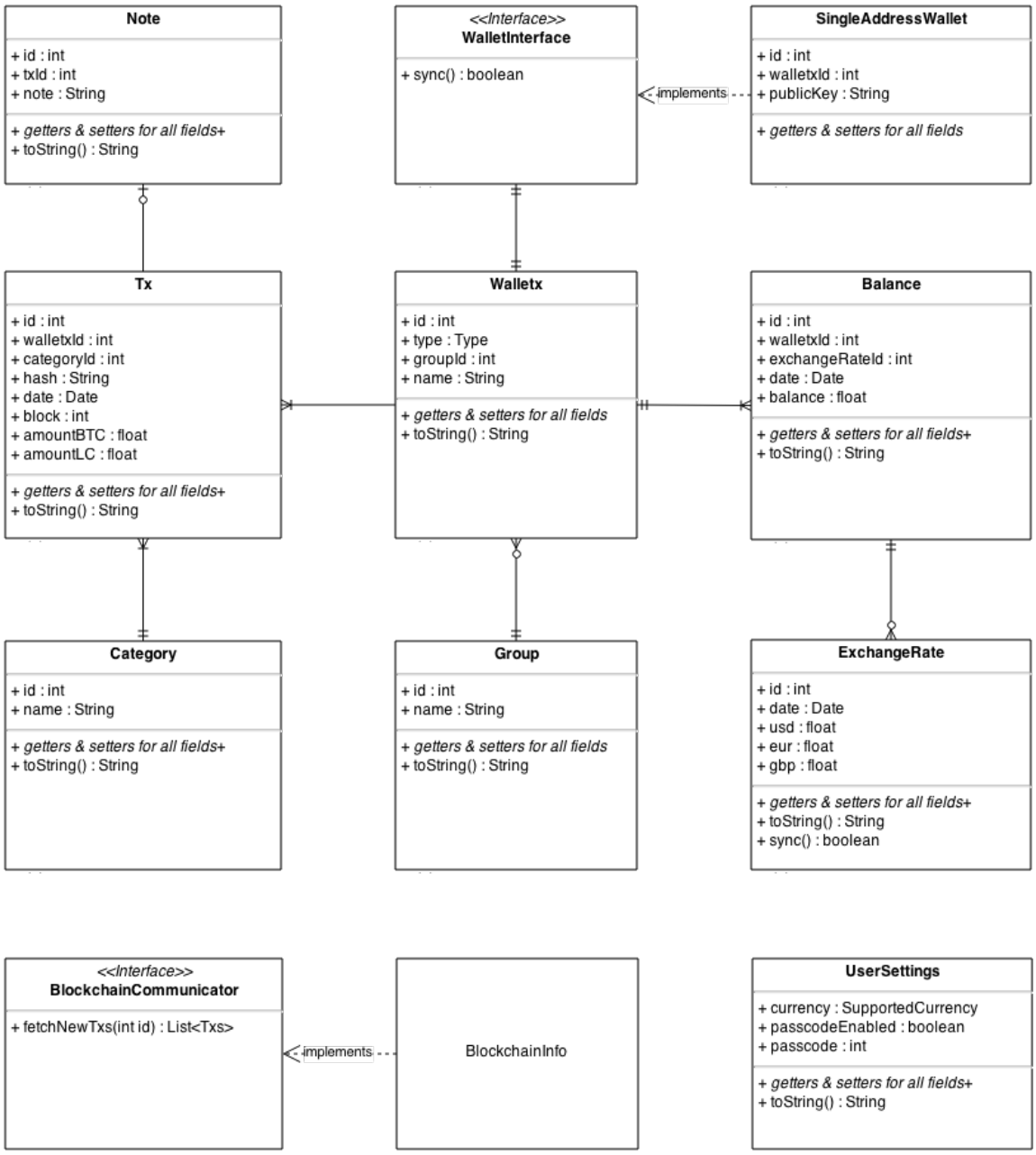
Figure 7.10: Wallet Summary View

WalleTx: Wallet Group Summary View	
Actors	User
Description	A user will be able to view a summary of all balances and transactions from created groups.
Data	transactions and current balances within the wallet groups.
Stimulus	User initiates summary view.
Response	A detailed summary is presented to the user.
Comments	The details can be refreshed as often as requested by the user.
Figure 7.12	

WalleTx: Individual Wallet Summary View	
Actors	User
Description	A user will be able to view a summary of all balances and transactions from individual wallets.
Data	transactions and current balances within the wallets.
Stimulus	User initiates summary view.
Response	A detailed summary is presented to the user.
Comments	The details can be refreshed as often as requested by the user.

Figure 7.12

7.1.3 Class Descriptions



7.1.4 Attribute Descriptions

7.2 Raw Use Case Point Analysis

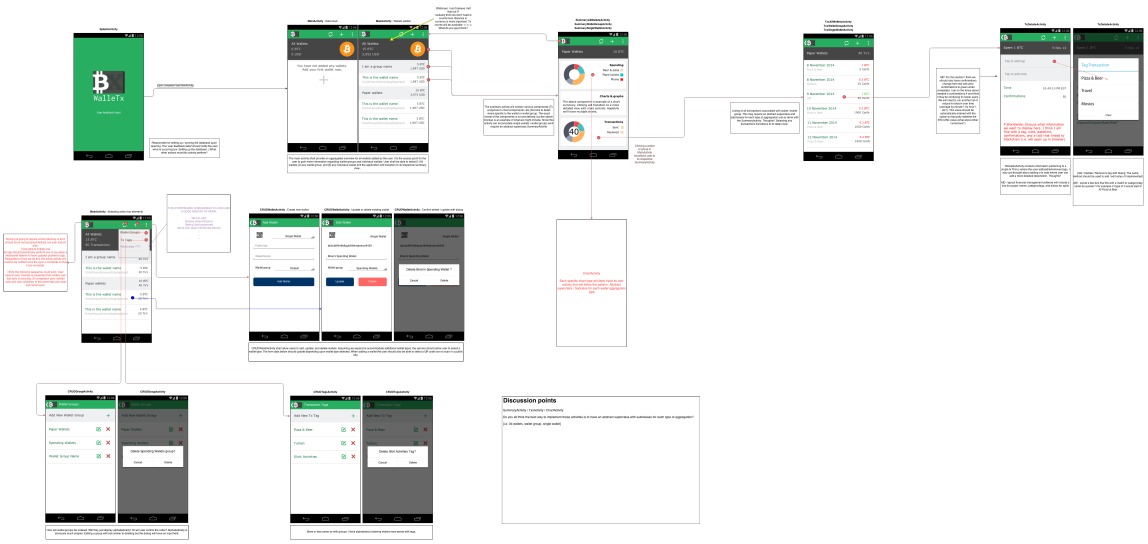
7.2.1 Actor Summary Table

Actor	Summary
Customer/End User	This actor will be the end user. They will interact with WalleTx, providing it with wallet information, tagging transactions, and viewing aggregated wallet information
Development Team/Maintenance	This actor will be interacting via the support and maintenance of the application. They will maintain DB organization, revision handling, and overall maintenance of the application
External Bitcoin Wallets	External Bitcoin Wallets will be providing our application with vital information. All external wallets' totals and transactions will aggregated in WalleTx.
SQLite Database	This actor is a database stored on each end user's phone. This will contain data from wallets, transactions, and display information. It will interact solely with the application via code.
Bitcoin Blockchain	The bitcoin blockchain will provide transaction specific data for all included wallets. This actor will interact with the application via database back-end.

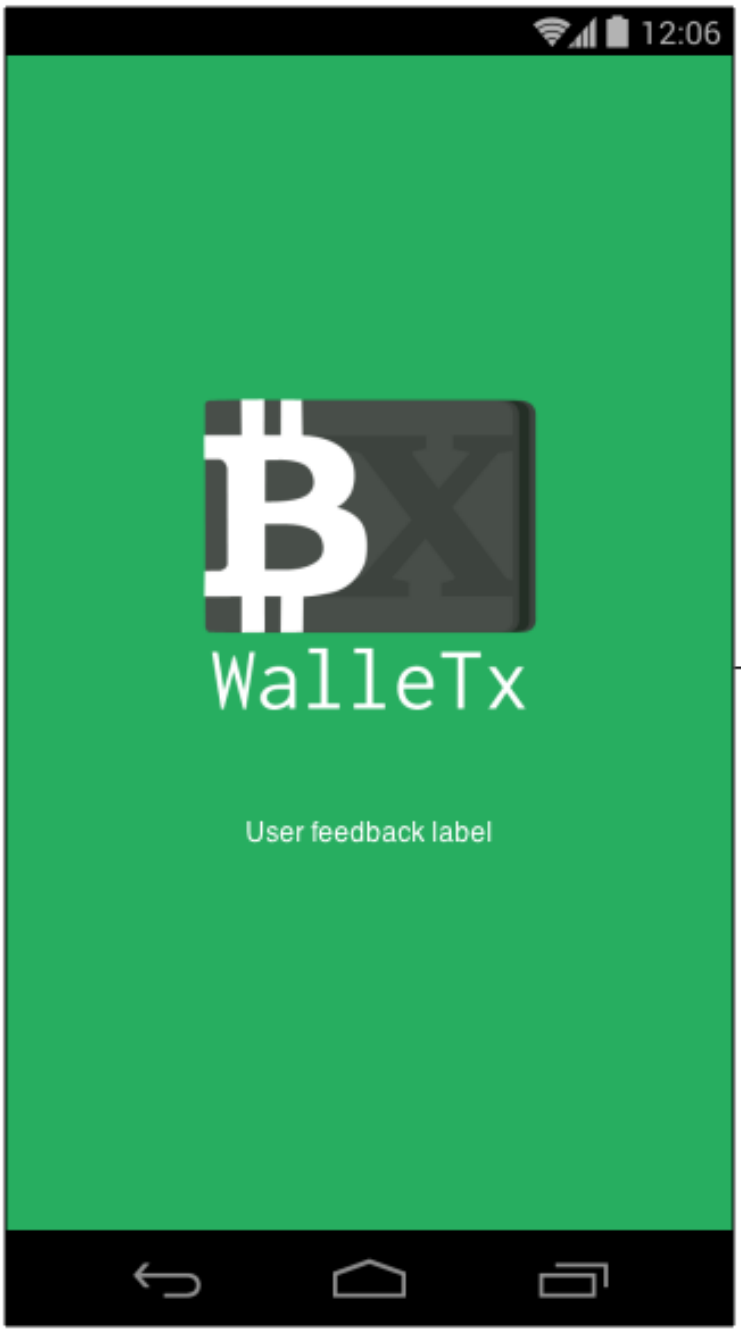
7.2.2 Use Case Summary Table

Actor	Use-Case Summary
Customer/End User	1. adding a wallet 2. removing a wallet 3. adding a tag to a transaction 4. removing a tag from a transaction 5. viewing all wallets 6. accessing application settings
Development Team/Maintenance	1. update revisions 2. maintain database structure
External Bitcoin Wallets	1. update added wallet information 2. provide public key 3. provide transactions made by wallets
SQLite Database	1. store wallet data 2. wallet data deleted 3. wallet data added 4. provide data for alerts 5. supply data for graphs/charts
Bitcoin Blockchain	1. update wallet/DB transactions 2. provide data for new wallet added

7.2.3 Screens and Reports with Navigation Matrix

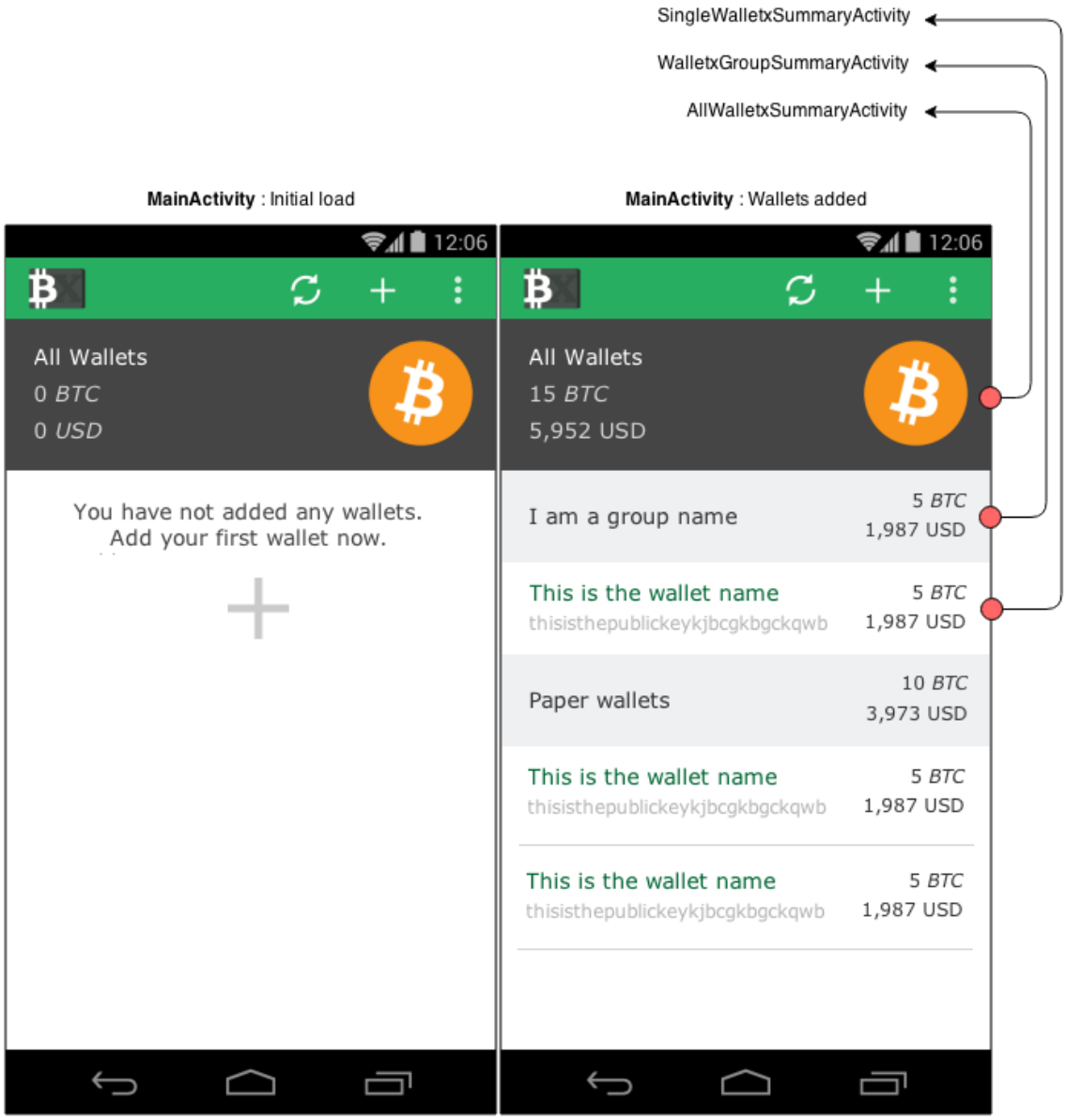


SplashActivity

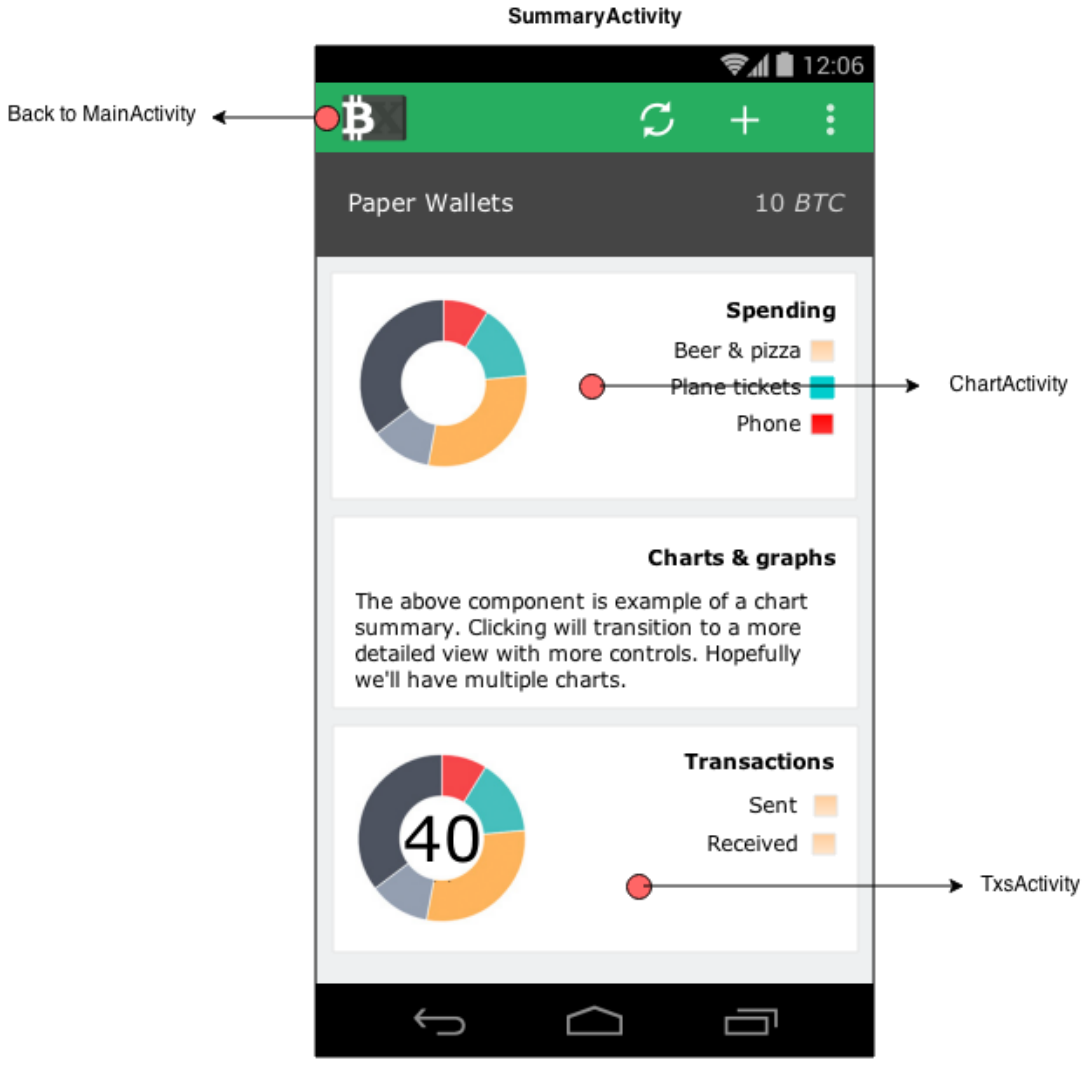


To MainActivity upon completion

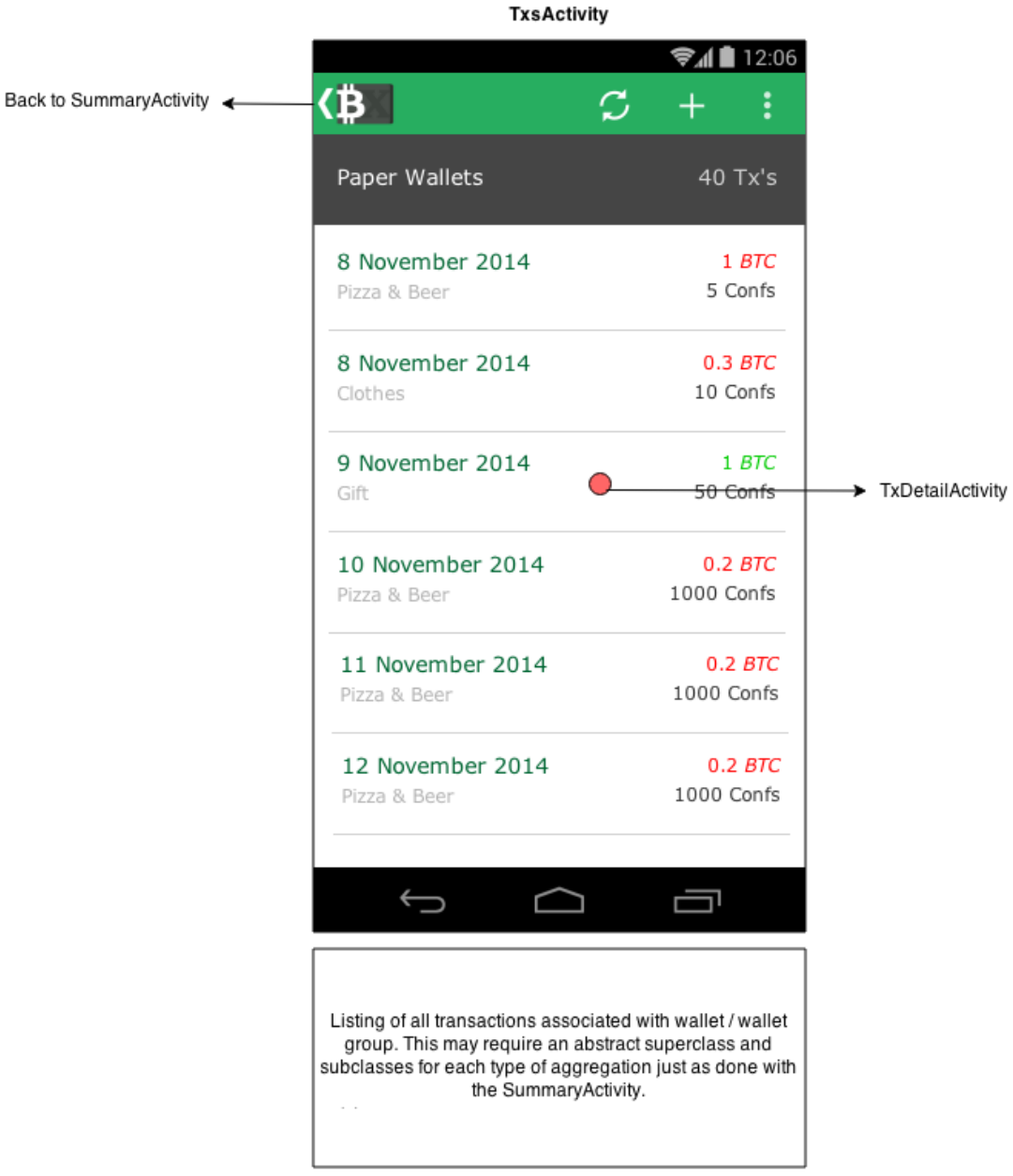
Sets up and syncs the database upon opening. '*User feedback label*' shall notify the user regarding what is occurring [*ex: Setting up the database...*]

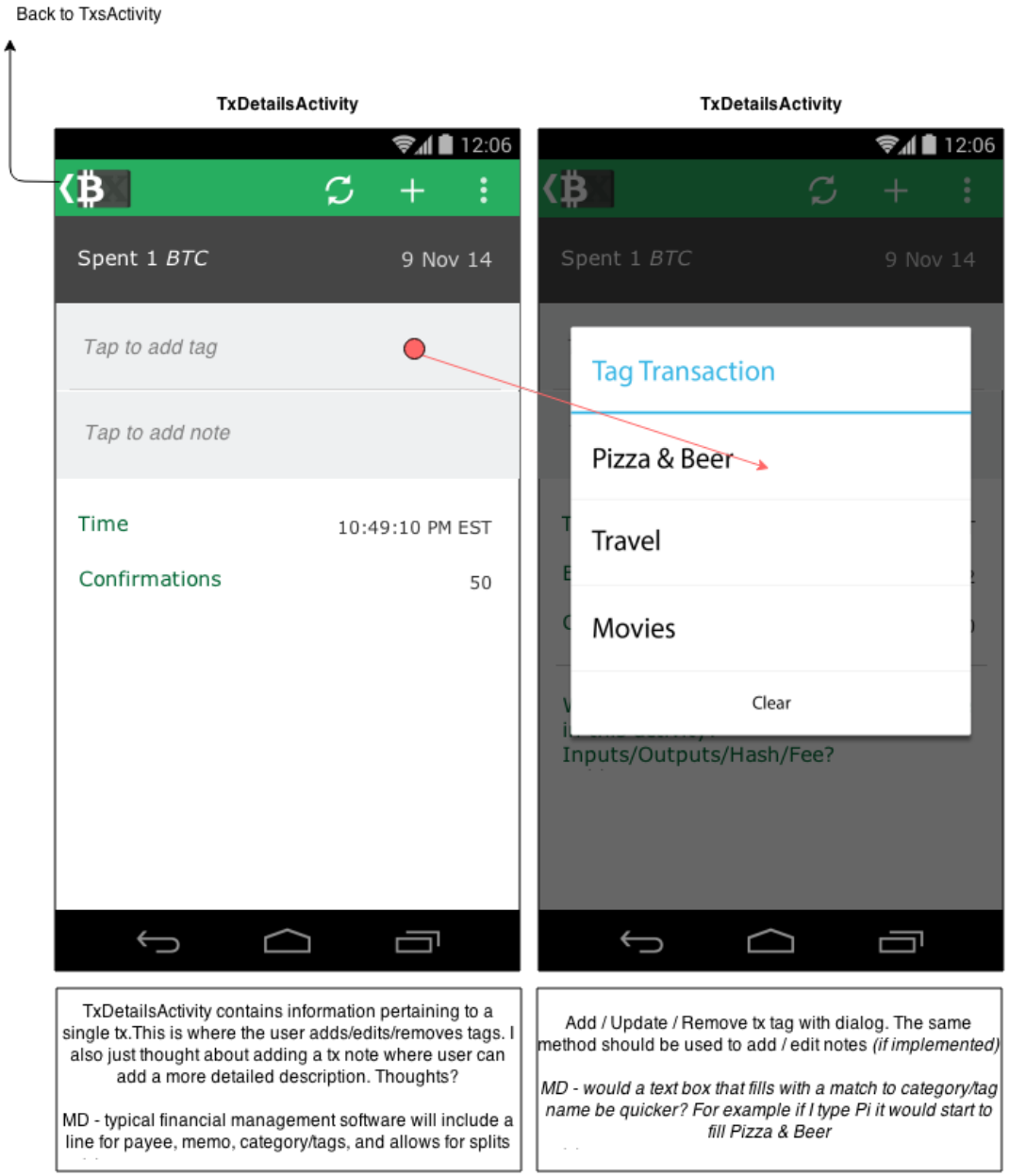


The main activity shall provide an aggregated overview for all wallets added by the user. It is the access point for the user to gain more information regarding wallet groups and individual wallets. User shall be able to select [1] All wallets, [2] any wallet group, and [3] any individual wallet and the application will transition to its respective summary view..

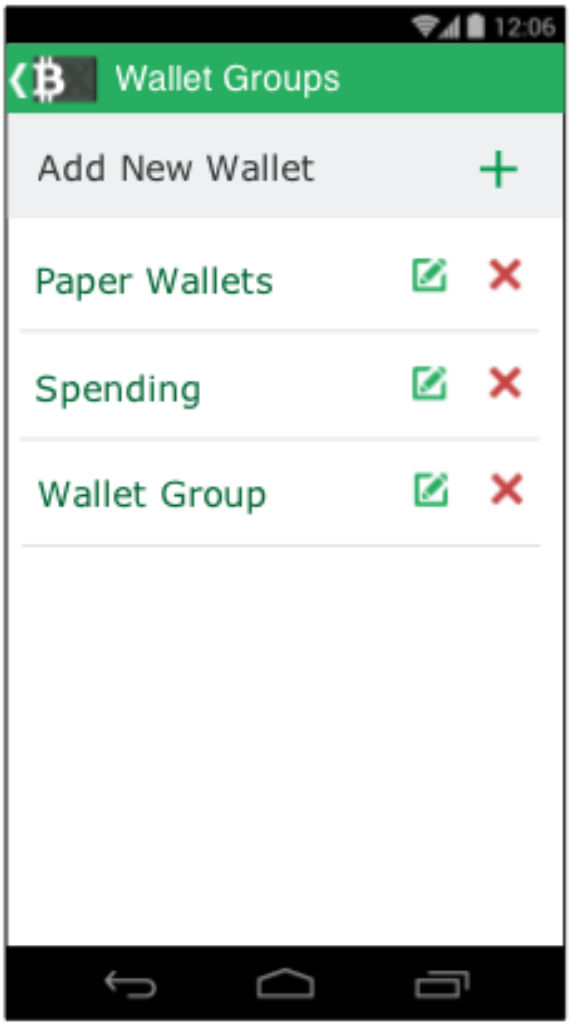


The summary activity will contain various components *[Tx component / chart components / etc.]* that link to detail views specific to the wallet or wallet group. The exact format of the components is not yet defined, but the above mockup is an example of what we might include. Since this activity can accomodate single wallets / wallet groups we'll require an abstract superclass SummaryActivity

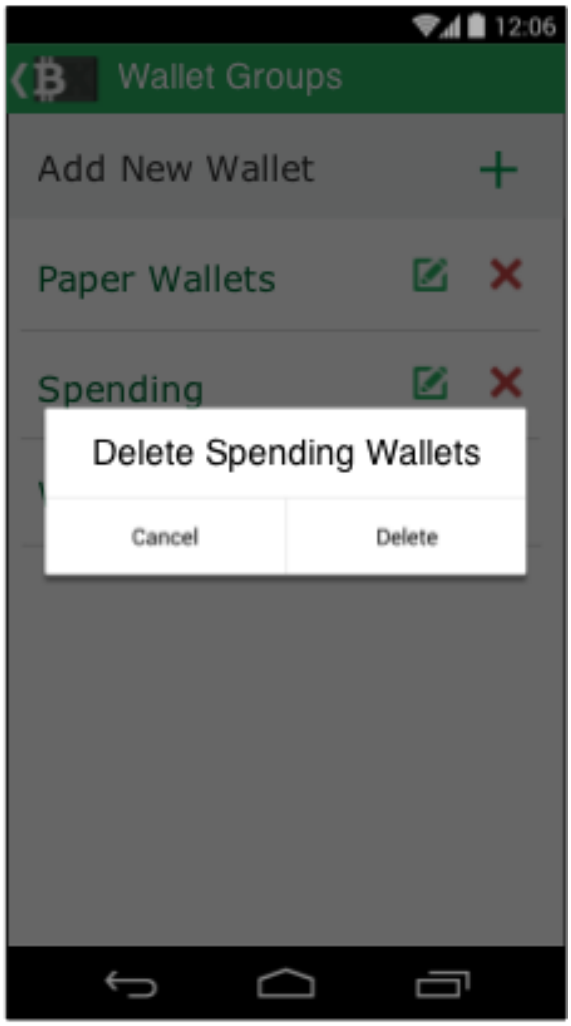

TxsActivity
12:06
Back to SummaryActivity
Paper Wallets
40 Tx's
8 November 2014
Pizza & Beer
1 BTC
5 Confs
8 November 2014
Clothes
0.3 BTC
10 Confs
9 November 2014
Gift
1 BTC
50 Confs
TxDetailActivity
10 November 2014
Pizza & Beer
0.2 BTC
1000 Confs
11 November 2014
Pizza & Beer
0.2 BTC
1000 Confs
12 November 2014
Pizza & Beer
0.2 BTC
1000 Confs
Listing of all transactions associated with wallet / wallet group. This may require an abstract superclass and subclasses for each type of aggregation just as done with the SummaryActivity.

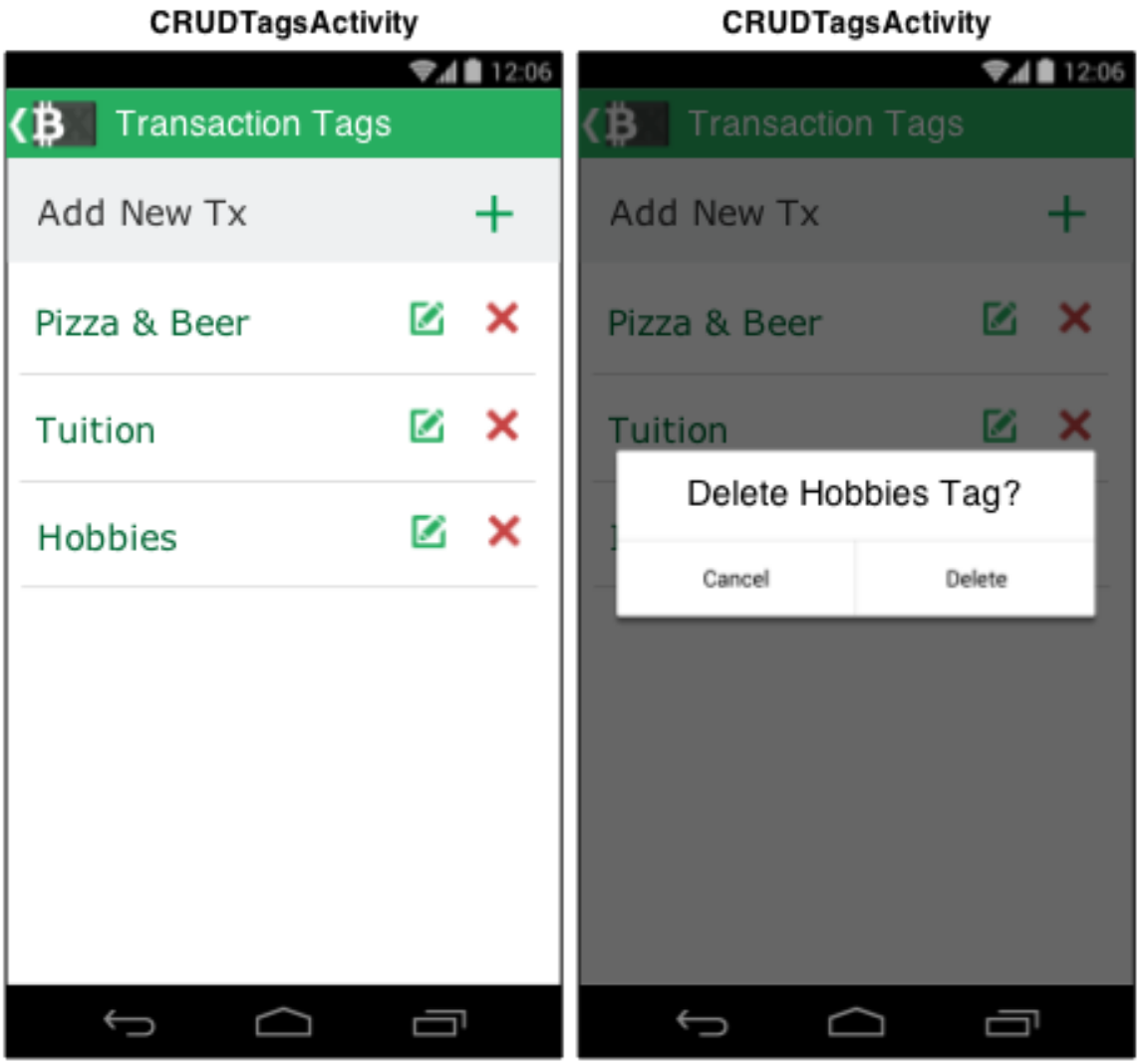

Back to TxsActivity
TxDetailsActivity
TxDetailsActivity
12:06
Spent 1 BTC
9 Nov 14
Tap to add tag
Tap to add note
Time
10:49:10 PM EST
Confirmations
50
Tag Transaction
Pizza & Beer
Travel
Movies
Clear
Inputs/Outputs/Hash/Fee?
TxDetailsActivity contains information pertaining to a single tx.This is where the user adds/edits/removes tags. I also just thought about adding a tx note where user can add a more detailed description. Thoughts?
MD - typical financial management software will include a line for payee, memo, category/tags, and allows for splits
Add / Update / Remove tx tag with dialog. The same method should be used to add / edit notes (if implemented)
MD - would a text box that fills with a match to category/tag name be quicker? For example if I type Pi it would start to fill Pizza & Beer

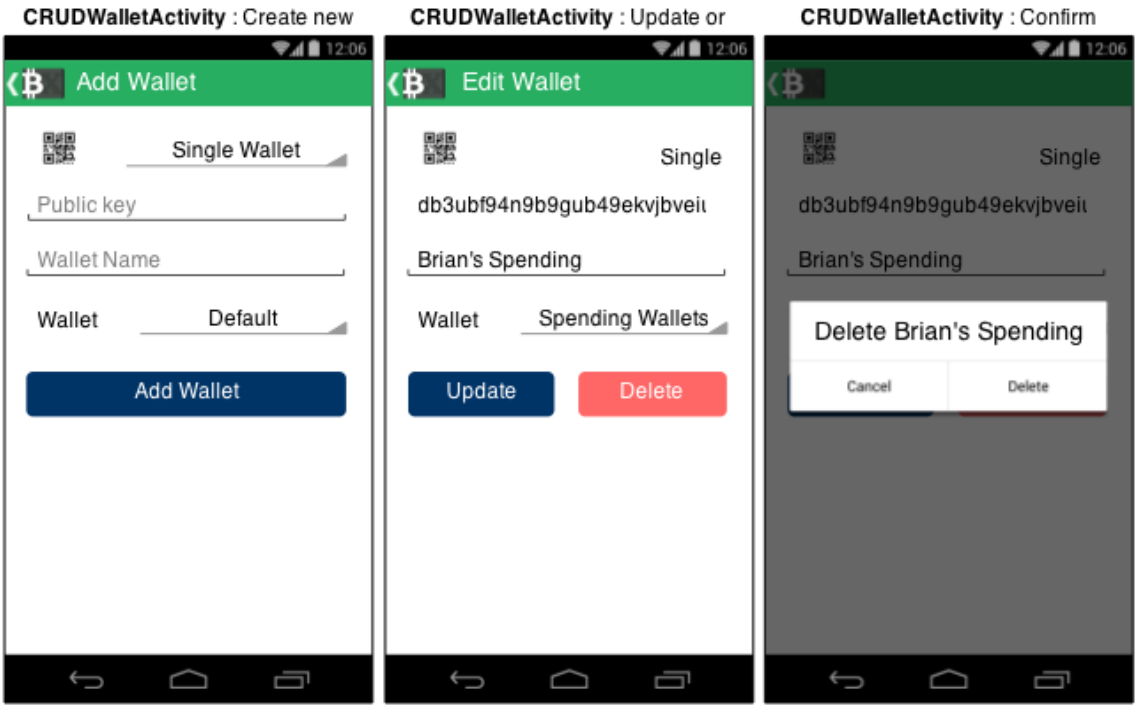
CRUDGroupActivity



CRUDGroupActivity




CRUDTagsActivity
12:06
Transaction Tags
Add New Tx
Pizza & Beer
Tuition
Hobbies
CRUDTagsActivity
12:06
Transaction Tags
Add New Tx
Pizza & Beer
Tuition
Delete Hobbies Tag?
Cancel
Delete



CRUDWalletActivity shall allow users to add, update, and delete wallets. Assuming we expand to accommodate additional wallet types, the spinner should allow user to select a wallet type. The form data below should update depending upon wallet type selected. When adding a wallet the user should also be able to select a QR code icon to scan in a public key.

7.3 Other Appendices

Reserved for future additions and amendments.