http://sce.uhcl.edu/helm/rationalunifiedprocess/applet/images/artfc_w.gif [Artifacts](http://sce.uhcl.edu/helm/rationalunifiedprocess/process/artifact/ovu_arts.htm) > http://sce.uhcl.edu/helm/rationalunifiedprocess/applet/images/artfc_y.gif [Requirements Artifact Set](http://sce.uhcl.edu/helm/rationalunifiedprocess/process/artifact/ars_req.htm) > http://sce.uhcl.edu/helm/rationalunifiedprocess/applet/images/additional.gif [{More Requirements Artifacts}](http://sce.uhcl.edu/helm/rationalunifiedprocess/process/artifact/arp_req_add.htm) > http://sce.uhcl.edu/helm/rationalunifiedprocess/applet/images/ar_srs.gif [Software Requirements Specification](http://sce.uhcl.edu/helm/rationalunifiedprocess/process/artifact/ar_srs.htm) > http://sce.uhcl.edu/helm/rationalunifiedprocess/applet/images/ie.gif SRS w/ Use-Cases

**<Testudo>**

**Software Requirements Specification**

**For <Cryptocurrency wallet>**

**Version <1.0>**

*[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]*

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <3/26/2018> | <1.0> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1.          Introduction](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#1.                  Introduction)

[1.1      Purpose](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#1.1               Purpose)

[1.2      Scope](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#1.2               Scope)

[1.3      Definitions, Acronyms and Abbreviations](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#1.3               Definitions, Acronyms and Abbreviations)

[1.4      References](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#1.4               References)

[1.5      Overview](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#1.5               Overview)

[2.          Overall Description](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#2.                  Overall Description)

[2.1      Use-Case Model Survey](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#2.1               Use-Case Model Survey)

[2.2      Assumptions and Dependencies](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#2.2               Assumptions and Dependencies)

[3.          Specific Requirements](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#3.                  Specific Requirements)

[3.1      Use-Case Reports](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#3.1               Use-Case Reports)

[3.2      Supplementary Requirements](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#3.2               Supplementary Requirements)

[4.          Supporting Information](http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_srsuc.htm#4.                  Supporting Information)

**Software Requirements Specification**

**1.**                  **Introduction**

*[The introduction of the****Software Requirements Specification (SRS)****should provide an overview of the entire document. It should include the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the****Software Requirements Specification****.]*

*[Note: The****Software Requirements Specification****captures the complete software requirements for the system, or a portion of the system.  Following is a typical****Software Requirements Specification****outline for a project****using use-case modeling****. This artifact consists of a package containing use cases of the use-case model and applicable Supplementary Specifications and other supporting information.  For a template of an****Software Requirements Specification******not****using use-case modeling, which captures all requirements in a single document, with applicable sections inserted from the Supplementary Specifications (which would no longer be needed), see rup\_srs.dot.]*

*[Many different arrangements of a****Software Requirements Specification****are possible.  Refer to [IEEE830-1998] for further elaboration of these explanations, as well as other options for a****Software Requirements Specification****organization.]*

* 1. **Purpose**

*[Specify the purpose of this****Software Requirements Specification****. The****Software Requirements Specification****should fully describe the external behavior of the application or subsystem identified. It also describes nonfunctional requirements, design constraints and other factors necessary to provide a complete and comprehensive description of the requirements for the software.]*

*The wallet should create public and private keys to access the blockchain network, so you can place coins in your wallet. That was the main purpose, however it is equally important to keep track of how much coin you currently have in your account. Along with a ledger that keeps track of the transaction history with your wallet, all requirements of a signing-only wallet have been described.*

**1.2**               **Scope**

*[A brief description of the software application that the****Software Requirements Specification****applies to; the feature or other subsystem grouping; what Use-case model(s) it is associated with; and anything else that is affected or influenced by this document.]*

*The software should behave like a normal wallet in that it should be able to tell the user how much they currently have in holdings as well as keep it safe.*

**1.3**               **Definitions, Acronyms and Abbreviations**

*[This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the****Software Requirements Specification****.  This information may be provided by reference to the project Glossary.]*

*Private key - The private portion of a keypair which can create*[*signatures*](https://bitcoin.org/en/glossary/signature)*that other people can verify using the*[*public key*](https://bitcoin.org/en/glossary/public-key)*.*

*Public key - The public portion of a keypair which can be used to verify signatures made with the private portion of the keypair.*

*Wallet - Software that stores private keys and monitors the block chain (sometimes as a client of a server that does the processing) to allow users to spend and receive satoshis.*

*Signatures - A value related to a public key which could only have reasonably been created by someone who has the private key that created that public key. Used in Bitcoin to authorize spending satoshis previously sent to a public key.*

*Parent key - In HD wallets, a key used to derive child keys. The key can be either a private key or a public key, and the key derivation may also require a chain code.*

*HD Protocol - The Hierarchical Deterministic (HD) key creation and transfer protocol (BIP32), which allows creating child keys from parent keys in a hierarchy. Wallets using the HD protocol are called HD wallets.*

**1.4**               **References**

*[This subsection should provide a complete list of all documents referenced elsewhere in the****Software Requirements Specification****.  Each document should be identified by title, report number (if applicable), date, and publishing organization.  Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

[*https://bitcoin.org/en/developer-guide#offline-wallets*](https://bitcoin.org/en/developer-guide#offline-wallets)

*Wallet section, Signing-only/Offline subsection*

**1.5**               **Overview**

*[This subsection should describe what the rest of the****Software Requirements Specification****contains and explain how the document is organized.]*

*The rest of this document describes the basic features of a wallet and how it should work from a user perspective.*

**2.**                  **Overall Description**

*[This section of the****Software Requirements Specification****should describe the general factors that affect the product and its requirements.  This section does not state specific requirements.  Instead, it provides a background for those requirements, which are defined in detail in****Section 3****, and makes them easier to understand. Include such items as product perspective, product functions, user characteristics, constraints, assumptions and dependencies, and requirements subsets.]*

*The product should be downloadable from the internet and then should be able to run offline and keep track of a person’s personal bitcoin holdings. The user should have a good feel for the user interface since it will be showing them how much bitcoin they currently have in this wallet and their previous transactions. A main factor into how the wallet works with the bitcoin blockchain relies on the implementation of the public key api from bitcoin. This will allow all transactions that the user does through the blockchain with their public key which was generated from our wallet, to be considered real transactions on the public ledger.*

**2.1**               **Use-Case Model Survey**

*[If using use-case modeling, this section contains an overview of the use-case model or the subset of the use-case model that is applicable for this subsystem or feature.  This includes a list of names and brief descriptions of all use cases and actors, along with applicable diagrams and relationships.  Refer to the****Use-Case-Model Survey Report****, which may be used as an enclosure at this point.]*

*User Interface – should allow the user to see previous transactions, create public and private keys, transfer their holdings to cold storage, and show the total holdings of bitcoin.*

**2.2**               **Assumptions and Dependencies**

*[This section describes any key technical feasibility, subsystem or component availability, or other project related assumptions on which the viability of the software described by this****Software Requirements Specification****may be based.]*

*We’re going to need to use a cryptography java package which is included on eclipse. It will help us hash the users public keys, to avoid hacking of their private key.*

**3.**                  **Specific Requirements**

*[This section of the****Software Requirements Specification****should contain all the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements and testers to test that the system satisfies those requirements.   When using use-case modeling, these requirements are captured in the use cases and the applicable supplementary specifications.  If use-case modeling is not used, the outline for supplementary specifications may be inserted directly into this section.]*

*The User Interface should have options to create a public and private key. It should also have a way to transfer the users private key to a printed paper along with the bitcoin holdings on the users account. In turn, this should erase the account entirely; this is putting your private key and holdings into a cold storage. The User Interface should also have a way to create public and private keys which are basically other wallets. The creation of these public keys should also incorporate a way to hash them which requires some cryptography implementation.*

**3.1**               **Use-Case Reports**

*[In use-case modeling, the use cases often define the majority of the functional requirements of the system, along with some non-functional requirements.  For each use case in the above use-case model, or subset thereof, refer to, or enclose, the use-case report in this section.  Make sure that each requirement is clearly labeled.]*

*User Interface – should allow the user to see previous transactions, create public and private keys, transfer their holdings to cold storage, and show the total holdings of bitcoin.*

**3.2**               **Supplementary Requirements**

*[Supplementary Specifications capture requirements that are not included in the use cases.  The specific requirements from the Supplementary Specifications, which are applicable to this subsystem or feature, should be included here and refined to the necessary level of detail to describe this subsystem or feature.  These may be captured directly in this document or referred to as separate Supplementary Specifications, which may be used as an enclosure at this point. Make sure that each requirement is clearly labeled.]*

*Creation of keys – access to bitcoin blockchain/public ledger, hashing of public keys.*

**4.**                  **Supporting Information**

*[The supporting information makes the****Software Requirements Specification****easier to use.  It includes:*

*•           Table of Contents*

*•         Index*

*•         Appendices*

*These may include use-case storyboards or user-interface prototypes. When appendices are included, the****Software Requirements Specification****should explicitly state whether or not the appendices are to be considered part of the requirements.]*

GUIDELINES FOR REQUIREMENTS

|  |  |  |
| --- | --- | --- |
| Checkpoints:  Software Requirements Specification  * The following basic issues should be addressed:   + Functionality: What is the software supposed to do?   + External interfaces: How does the software interact with people, the system's hardware, other hardware, and other software?   + Performance: What is the speed, availability, response time, recovery time of various software functions, etc.?   + Attributes: What are the portability, correctness, maintainability, security, etc. considerations?   + Design constraints imposed on an implementation: Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environments, etc.? * Are any requirements specified that are outside the bounds of the SRS? This means the SRS   + Should correctly define all of the software requirements,   + Should not describe any design or implementation details,   + Should not impose additional constraints on the software. * Does the SRS properly limit the range of valid designs without specifying any particular design? * Does the SRS exhibit the following characteristics?   + Correct: Is every requirement stated in the SRS one that the software should meet?   + Unambiguous     - Does each requirement have one, and only one, interpretation?     - Has the customer's language been used?     - Have diagrams been used to augment the natural language descriptions?   + Complete     - Does the SRS include all significant requirements, whether related to functionality, performance design constraints, attributes, or external interfaces?     - Have the expected ranges of input values in all possible scenarios been identified and addressed?     - Have responses been included to both valid and invalid input values?     - Do all figures, tables and diagrams include full labels and references and definitions of all terms and units of measure?     - Have all TBDs been resolved or addressed?   + Consistent     - Does this SRS agree with the Vision document, the use-case model and the Supplementary Specifications?     - Does it agree with any other higher level specifications?     - Is it internally consistent, with no subset of individual requirements described in it in conflict?   + Ability to Rank Requirements     - Has each requirement been tagged with an identifier to indicate either the importance or stability of that particular requirement?     - Have other significant attributes for properly determining priority been identified?   + Verifiable     - Is every requirement stated in the SRS verifiable?     - Does there exist some finite cost-effective process with which a person or machine can check that the software product meets the requirement?   + Modifiable     - Are the structure and style of the SRS such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style?     - Has redundancy been identified, minimized and cross-referenced?   + Traceable     - Does each requirement have a clear identifier?     - Is the origin of each requirement clear?     - Is backward traceability maintained by explicitly referencing earlier artifacts?     - Is a reasonable amount of forward traceability maintained to artifacts spawned by the SRS?   Reference: [[IEEE93](http://sce.uhcl.edu/helm/rationalunifiedprocess/process/referenc.htm#IEEE93)]  [Copyright  © 1987 - 2001 Rational Software Corporation](http://sce.uhcl.edu/helm/rationalunifiedprocess/copyrite/copyrite.htm) |  | [[Display Rational Unified Process using frames](javascript:loadTop();)](javascript:loadTop();) |

|  |
| --- |
| Rational Unified Process   http://sce.uhcl.edu/helm/rationalunifiedprocess/_borders/rupversion.gif |