**SpartaGold:**

**an SJSU Cryptocurrency**

|  |
| --- |
| A Project Report Presented to  The Faculty of the Computer Engineering Department |
| San Jose State University In Partial Fulfillment Of the Requirements for the Degree Bachelor of Science in Software Engineering and  Bachelor of Science in Computer Engineering |

|  |
| --- |
| By |
| Art Tucay Jr., Benjamin Roppiyakuda,  Mojdeh Keykhanzadeh, Paul Portela, Phi Le |
| 09/2014 |

|  |
| --- |
| Copyright © 2014 |
| Art Tucay Jr., Benjamin Roppiyakuda,  Mojdeh Keykhanzadeh, Paul Portela, Phi Le |
| ALL RIGHTS RESERVED |

|  |
| --- |
| APPROVED FOR THE COLLEGE OF ENGINEERING |
|  |
| Dr. Thomas Austin, Project Advisor |
|  |
| Professor Keith Perry, Instructor |
|  |
| Dr. Xiao Su, Computer Engineering Department Chair |

**ABSTRACT**

**SpartaGold: an SJSU Cryptocurrency**

By Art Tucay Jr, Ben Roppiyakuda, Mojdeh Keykhanzadeh, Paul Portela, Phi Le

A digital currency has to establish authenticity, security, and non-repudiation in order for it to be considered a viable currency. Credit cards, debit cards, and PayPal transactions are all considered widely used centralized digital currencies. As digital currencies expanded, decentralized currencies connected through peer-to-peer networks emerged, such as Litecoin, Primecoin, and the very popular Bitcoin. These decentralized digital currencies need cryptographic user-authenticated protocols for transactions to provide reduced transaction costs in comparison to centralized digital currencies.

The objective of this project is to build a functioning decentralized digital cryptocurrency for the San Jose State University campus. This cryptocurrency, dubbed “SpartaGold”, will be tradable and spendable among SpartaGold users. SpartaGold will utilize digital signature techniques to provide high security for user transactions.

Cryptocurrencies currently in circulation conduct transactions with high end security protocols. The SpartaGold project values security first and foremost as well. The SpartaGold project has the ability to increase the public’s knowledge concerning security and peer-to-peer networking, primarily within the SJSU campus.

Table of Contents

**Chapter 1 Introduction**

* 1. Project Goals and Objectives
  2. Problem and Motivation
  3. Project Application and Impact
  4. Project Results and Deliverables
  5. Market Research
  6. Project Report Structure

**Chapter 2 Background and Related Work**

* 1. Background and Used Technologies

2.2 State-of-the-art

* 1. Literature Survey

**Chapter 3 Project Requirements**

3.1 Domain and Business Requirements

3.2 System (or Component) Functional Requirements

3.3 Non-function Requirements

3.4 Context and Interface Requirements

3.5 Technology and Resource Requirements

**Chapter 4 System Design**

4.1 Architecture Design

* 1. Interface and Component Design
  2. Structure and Logic Design
  3. Design Constraints, Problems, Trade-offs, and Solutions

**Chapter 5 System Implementation**

5.1 Implementation Overview

5.2 Implementation of Developed Solutions

5.3. Implementation Problems, Challenges, and Lessons Learned

**Chapter 6 Tools and Standards**

6.1. Tools Used

6.2. Standards

**Chapter 7 Testing and Experiment**

7.1 Testing and Experiment Scope

7.2 Testing and Experiment Approach

7.3 Testing and Experiment Results and Analysis

**Chapter 8 Conclusion and Future Work**

**References**

List of Figures

Figure 1. Send Signed Message .............................................................. Page 24

Figure 2. Receive Message and Give Goods ........................................... Page 25

Figure 3. Send Message .......................................................................... Page 26

Figure 4. Validate Message ...................................................................... Page 26

Figure 5. See Balance .............................................................................. Page 27

Figure 6. Block Chain ............................................................................... Page 42

Figure 7. SpartaGold Wallet Architecture ............................................. Page 45

Figure 8. Send Message GUI Design ....................................................... Page 46

Figure 9. Receive Message GUI Design .................................................. Page 47

Figure 10. Transactions GUI Design ………............................................. Page 47

Figure 11. Peer-to-Peer Component Diagram ......................................... Page 48

Figure 12. Send Gold Sequence Diagram ............................................. Page 52

Figure 13. Receive Gold Sequence Diagram ......................................... Page 53

Figure 14. Validate Transactions Sequence Diagram .............................. Page 53

Figure 15. Class Diagram ......................................................................... Page 54

Figure 16. Control Flow Diagram For Sending Gold............................... Page 55

Figure 17. Context Diagra………….......................................................... Page 56

Figure 18. MWV Data Flow Diagram ….................................................... Page 57

Figure 19. V Testing Model .……….......................................................... Page 82

Figure 20. Test Plan ...…….……….......................................................... Page 83

List of Tables

Table 1. System Functional Requirements ............................................... Page 28

Table 2. Non-functional Requirements: Usability ..................................... Page 30

Table 3. Non-functional Requirements: Reliability .................................... Page 30

Table 4. Non-functional Requirements: Performance .............................. Page 30

Table 5. Non-functional Requirements: Supportability ............................. Page 31

Table 6. Non-functional Requirements: Security ...................................... Page 31

Table 7. Hardware Requirement for SpartaGold Wallet ....................... Page 33

Table 8. Hardware Requirement for Mining .............................................. Page 34

Table 9. Software Requirement for Users Computer ............................... Page 35

Table 10. Software Requirement Traceability Matrix ................................ Page 37

Table 11. Use Case 1 ............................................................................... Page 49

Table 12. Use Case 2 ............................................................................... Page 49

Table 13. Use Case 3 ............................................................................... Page 50

Table 14. Testing Levels .......................................................................... Page 84

Table 15. Testing Cases ........................................................................... Page 86

**Chapter 1:  Introduction**

**1.1 Project Goals and Objectives**

The goal of the SpartaGold project was to provide San Jose State University with a digital currency that allowed students and staff to conduct transactions with economic and competitive transaction fees.  By simply downloading the SpartaGold Wallet, students and staff are able to exchange SpartaGold with others whenever they need to buy, sell, or give products.  A SpartaGold transaction fee also makes small transactions convenient as the fee no longer diminishes business profits.  Once SpartaGold proves to be a convenient and secure form of currency, the SpartaGold project will shift its goal to promoting the SpartaGold network to on-campus businesses and vendors.

**1.2 Problem and Motivation**

Today, nearly all transactions on SJSU are processed through cash and credit cards.  Cash may be unclean and unsecure when in transport due to physical exchanges.  Credit card companies charge transaction fees which cause an inconvenience to businesses trying to sell items at a low price.  In addition, credit card companies are vulnerable to credit card information leaks due to their inherent single point of failure with centralized banks, leaving customers with a hesitant sense of security.  SpartaGold addresses these problems by utilizing a decentralized peer-to-peer network. A decentralized P2P network removes the centralized bank, creating a network of peers who collectively track all balances and transactions. SpartaGold also removes charges many credit companies impose on users, making it a strong contender against cash and credit cards. Of course, in order to establish a P2P network, SpartaGold needed users.

**1.3 Project Application and Impact**

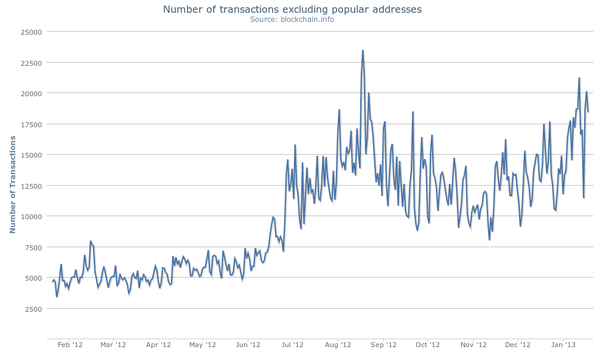
SpartaGold is a digital decentralized cryptocurrency targeted towards the SJSU community.  Students are able to sell and buy items from one another without worrying about transaction vulnerabilities and high transaction fees.  For example, students often raise money for their respective clubs by selling food and beverages on campus to the SJSU community.  With SpartaGold, the community can now sell and buy goods without cash or credit.  In addition, students would also be able to sell and buy books from each other using SpartaGold without worrying about actual cash handling during transactions. These examples, along with many other general applications, have provided SpartaGold users with a cheap and easy process for the exchange of goods.  In short, SpartaGold is applicable to any transactions between multiple peers, providing the SJSU community with low transaction costs.

**1.4 Project results and deliverables**

At the end of May of 2014, the SpartaGold project report was completed and submitted to the professor in charge of senior project planning.  The document contains the project scope, schedule, requirements, tools, and a test plan that was executed during the implementation phase.  A prototype was also delivered to the project advisor Dr. Thomas Austin during the summer to show the proof of concept.  These two deliverables guided the project towards the final steps of implementation of the SpartaGold Wallet with functional GUI, network connectivity, and cryptographic security. After strict alpha and beta testing with users, the SpartaGold project reached completion in December of 2014. Users of SpartaGold have generated their own Gold through the use of the SpartaGold Wallet, and have successfully transferred funds among each other.

**1.5 Market Research**

Many digital currencies compete for popularity and users, with Bitcoin at the top.  Bitcoin is currently the most popular digital currency with more than two million users.  Bitcoins have slowly been gaining acceptance.  Reddit, one of the largest social news networks, recently allowed Bitcoins as payment for their services and goods.  There are other digital cryptocurrencies trying to compete, each used to their own extent within their own niches.  The goal of this project was to reduce student costs and increase purchasing privacy by implementing a digital currency for SJSU student and faculty niche. The following chart shows the popularity of Bitcoin by graphing the number of transactions with Bitcoin by date:



**1.6 Project Report Structure**

This report documents the background of decentralized cryptocurrencies and how they stand in today’s economy and technological business, the needs and scope of the SpartaGold project, the architecture and implementation of the SpartaGold Wallet and network, and the tools used to complete these tasks. Each section is numerically labeled, with each subsection numerically labeled through an appendage to the super section. Tables and Figures are labeled and titled, and can be referred to through the Table of Contents. A collection of terms used throughout the document is listed at the end of Chapter 3: Project Requirements.

**Chapter 2: Background and Related Work**

**2.1 Background and Used Technologies**

Cryptocurrencies are digital funds highly encrypted with various forms of information security, and are used to conduct online transactions among peers.  The concept is simple at first glance: User-A wants to buy goods from User-B and agrees to pay User-B 10 units of currency.  User-B accepts the payment and reciprocates by sending User-A the desired purchase.  This is an average transaction done with physical currencies around the world, using paper money or metal coins as the valued commodity.  However, when conducting online transactions, physical currencies are not available or easily provided.  Problems arise in sender authentication, receiver authentication, or third-party attacks.  User-A must authenticate that they are indeed sending their 10 units of currency to User-B, and User-B must authenticate that User-A is the person sending 10 units of currency.  User-B must also validate that User-A has 10 units of currency to spend, and User-A and User-B must reassure that a malicious attack from User-C cannot disrupt their transaction.  These safety protocols are implemented using cryptography, the practice of securely communicating with another entity while in the presence of other parties.

Peer-to-peer (often abbreviated as “P2P”) networks are a very important factor in cryptocurrencies that do not rely on a central authority.  Peer-to-peer file sharing is a data sharing process which connects users to other users, also known as “peers,” instead of users to a single server.  Instead, each user acts as both a client and a server, synonymously. Specific software is necessary to connect a peer to other peers.  Users send requests for downloads of specific files through this software.  Once a connection is established with other peers on the internet running similar software, the software will securely transport required files and messages from one peer’s computer to another, and then proceed with cryptographic protocols.  This is the backbone of current cryptocurrencies, as it allows users to assure funds are legitimate, and to transfer currencies without a central authority, or, in the case of online file sharing, a single server.

The development team of SpartaGold has utilized knowledge from a series of courses at SJSU to produce SpartaGold and the SpartaGold Wallet:

|  |  |
| --- | --- |
| **Course** | **Application** |
| Java Data Structures | Base code written in Java |
| Object-Oriented Design | GUI implementation |
| Information Security | Security for transactions |
| Computer Networks | Basis for the P2P network |
| Project Management | Team work allocation and timeline |

**2.2 State-of-the-art**

The most widely known and used decentralized cryptocurrency in the online market is, subjectively, Bitcoin.  Bitcoin’s boom in the virtual market began in 2009, and many users found the use of a purely digital, highly anonymous currency very alluring for online transactions.  Unlike some physical currencies, such as the US dollar or the European Euro, Bitcoin is a deflationary cryptocurrency; it gradually and naturally grows in value due to its limited supply.  Since Bitcoin has a limit of 21 million bitcoins, the supply is limited, so the price per coin rises.  While inflation is a typical problem among currencies with printable money, deflation also causes some problems within an economy.  When a currency deflates, exchange rates begin to fall in relation to the deflated currency.  This causes holders of a deflated currency to hold onto their money instead of spending it.  In addition, those in debt with the current deflated currency will be hit with an even harder debt to fill, as the amount needed to pay off a debt rises with the value of the currency.

Other cryptocurrencies, such as Primecoin and Dogecoin, are still in circulation online.  The most common difference between digital currencies on the current online market is the proof-of-work, or the algorithm used to verify that a user has put the most effort in validating a transaction.  Primecoin uses prime chains as its proof-of-work system, while Dogecoin and Litecoin use a scrypt function.  These and other different digital currencies (also known as “Altcoins”), are briefly described in section 2.3 Literature Survey.

All decentralized cryptocurrencies rely on a peer-to-peer network.  Some support multiple peer-connectivity programs which store, send, and receive coins from other users. These programs come in multiple forms also known as “wallets”: a software wallet, a mobile wallet, a web wallet, or a paper wallet.  Web wallets store coins on a cloud and are accessed normally through web browsers, while paper wallets are physical print-outs of the encrypted string (the code behind each coin), generally condensed into a QR code.  Software wallets are stored on desktops or laptops and provide basic transaction functions using the peer-to-peer network, among other functionalities.  Mobile wallets are smaller versions of a software wallet used for mobile devices such as smartphones.  Mobile wallets handle transactions using QR code scans and phone-to-phone transfers, likely through Bluetooth.

**2.3 Literature Survey**

Decentralized cryptocurrencies exist in many different forms. The following lists brief examples of past and currently existing cryptocurrencies and their unique traits:

**Bitcoin**

Bitcoin is the basis of many current digital currencies, and is the model of how a cryptocurrency can work without a central server.  The popularity of Bitcoin has expanded over time, and the value per coin has skyrocketed.  On October 5, 2009, one US dollar had an equivalent worth of 1,309.03 bitcoins, or BTC.  As of March 23, 2014, one US dollar has an equivalent worth of about 0.001803 BTC.  To put that in a better scale, 1 BTC equals roughly 554 USD (this price fluctuates by the second).  Bitcoin uses a peer-to-peer network as a solution to double spending by maintaining a list and creating a chain of transactions.  A bitcoin, at its core, is a chain of digital signatures created from the hash of the previous chain and the receiver’s public key, which is then digitally signed and added to the end of the chain.  The longest chain proves that a miner has spent the most amount of computing power on the hash-based proof-of-work, and thus they would be rewarded with a specified amount of bitcoin, after a SHA-256 proof-of-work function is done and broadcasted to all peers.  [6]

**Peercoin**

Peercoin, also known as PPCoin, is a cryptocurrency based on Bitcoin and its architecture.  PPCoin is currently the third largest cryptographic currency, with more than 21 million coins in circulation.  It uses many of the same components as Bitcoin, such as a peer-to-peer network, hashing functions, and a proof-of-work.  However, what makes PPCoin unique is its implementation of a newer transaction validation system titled “proof-of-stake”.  The idea behind proof-of-stake is to have miners validate transactions based on the amount of coins they are holding, instead of the amount of computational power they have, like Bitcoin’s proof-of-work.  For example, if a miner is holding 1% of the currency, then they will validate 1% of all transactions.  This has the effect of making it very expensive for dishonest miners to validate fraudulent transactions, easy for users to validate, and environmentally friendly as it does not require much power to validate.  The purpose of this system is to avoid a case where there are not enough miners connected to the network.  PPCoin also implements an unlimited and constantly growing amount of coins circulating through the system.  [5] [7]

**Primecoin**

Primecoin was created by the same creator(s) of Peercoin, under the pseudonym of “Sunny King.”  Primecoin’s proof-of-work sets it apart from Bitcoin, using prime chains as it’s proof-of-work system.  Primecoin accepts three types of prime chains as proof-of-work: Cunningham chain of the first kind, Cunningham chain of the second kind, and bi-twin chain.  Much if its architecture mimics that of its sister, Peercoin.  [5]

**Ripple**

Ripple is an online currency exchange system used to trade currencies with other currencies.  Ripple is a math-based cryptocurrency with secure and reliable transactions.  A unit of Ripple currency, dubbed “XRP”, is purchasable and tradable, much like other cryptocurrencies.  Peer-to-peer payments are also incorporated into the system, providing the same decentralized benefits of other cryptocurrencies.  Transactions are direct and distributed among all users of the Ripple protocol, and provides no counterparty risk or fees to any system administrator.  Ripple boasts zero inflation due to a set amount of XRP.  [2]

**Litecoin**

Litecoin is a peer-to-peer digital currency created by Charles Lee on October 13, 2011.  According to current trading markets, Litecoin now is the second most valuable digital currency in the world.  In general, Litecoin uses the same protocol as Bitcoin, with the exception of two improvements.  Instead of using SHA-256 to hash in the proof-of-work algorithm as Bitcoin, Litecoin uses scrypt.  This proof-of-work algorithm requires a large amount of memory allocated towards computational processing.  Litecoin also implements a shorter transaction confirmation time, so transactions complete at a faster rate. According to its development team, Litecoin’s transaction confirmation time is only 2.5 minutes (on average) in comparison to Bitcoin’s 10 minutes.  Litecoin is a limited currency, capped at a total of 84 million coins.  [9]

**Dogecoin**

Dogecoin is based on three cryptocurrencies: Bitcoin, Litecoin, and Luckycoin.  Dogecoin is named after the popular “doge” meme, and is used in a similar fashion as Bitcoin.  However, the creator, Billy Markus, strove to keep Dogecoin apart from Bitcoin’s reputation as a Silk Road currency.  Dogecoin uses the scrypt function instead of Bitcoin’s SHA-256, but shares similarities in its encryption protocol. Dogecoin uses hashes of public keys as addresses, all beginning with the letter D.  There is no maximum amount of dogecoins, with approximately 5.2 billion coins created every year, making it an inflationary currency.  It is currently often used as a tipping system for generally well received content on social media sites.  [12]

**Namecoin**

Namecoin is a good example of the Bitcoin protocol’s adaptability.  Based on Bitcoin, Namecoin is not a currency in the traditional sense.  Instead, Namecoin acts as an alternative Domain Name System (DNS), controlling the bit domain outside the realm of the Internet Corporation for Assigned Names and Numbers (ICANN), which oversees all the top-level domains.  Users simply purchase bit domains for one Namecoin apiece, and the domain itself is added to the so-called “block chain,” or public ledger that keeps track of who owns which namecoins/domains.  Ownership of a name is based on the ownership of a coin, which is, in turn, based on public key cryptography.  The Namecoin network reaches consensus every few minutes as to which names have been reserved or updated.  Because Namecoin has value outside of the currency itself (in the form of domain name ownership), it may one day prove to be stronger and more valuable than other cryptocurrencies.  [1]

**Scrypt**

Scrypt is a cryptographic function used to make hardware attacks difficult.  Password-based key derivation functions, or password-based KDFs, were designed to be computationally costly and take a significant amount of time.  However, possible breaches still exist, such as parallel computing for brute force attacks.  Scrypt is a variation of a password-based KDF, but it requires an increased use of memory, making it memory-hard as well as password-based.  Cryptocurrencies such as Litecoin and Dogecoin use scrypt as a proof-of-work, requiring all work to be done within a system’s memory instead of the GPU, where other cryptocurrencies allocate work.  [10]

**Cybercash**

Cybercash was an electronic payment service designed to support e-commerce transactions.  The company was founded in 1994 by Daniel C. Lynch, William N. Melton, Steve Crocker, and Bruce G. Wilson.  Similar to PayPal, Cybercash was designed to ensure secure online transactions; it acted as a third party from a seller to a buyer.  Cybercash is based off of the Cybercash Wallet program; buyers had to install a software on their computer before any transaction could take place.  [2]

**Disadvantage of SpartaGold**

Disadvantages of using SpartaGold exist in the literal concept of a digital currency.  Any new currency is only as valuable as it is perceived.  If the majority of users of SpartaGold believe that the worth of a single SpartaGold is 1/100th of a US Dollar, then its value will drop.  This is why the Bitcoin market is so volatile; Bitcoins can be sold for over 500 US Dollars one minute, then 200 US Dollars the next.

**Chapter 3: Project Requirements**

**3.1 Domain and Business Requirements**

**3.1.1 Process Summary Diagram**

|  |
| --- |
| **send message.png** |
| **Figure 3.1 Send Signed Message** |

|  |
| --- |
| Receive Message.png |
| **Figure 3.2 Receive Message and Give Merchandise** |

**3.1.2 Process Decomposition Diagram**

|  |
| --- |
| **send_message.png** |
| **Figure 3.3 Send Message** |

|  |
| --- |
| validate_transaction.png |
| **Figure 3.4 Validate Transaction** |

|  |
| --- |
| see_balance.png |
| **Figure 3.5 See Balance** |

**3.2 System Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **F-01** | The user shall be able to see their pending and past transactions. | Medium |
| **F-02** | The user shall be able to see their available balance. | High |
| **F-03** | The user shall be able to receive Gold from another user. | High |
| **F-04** | The user shall have the option to validate transactions. | Medium |
| **F-05** | The user shall have a ledger of all transaction records that have occurred in the SpartaGold network. | High |
| **F-06** | The user should be able to sort their transaction record by date, address, amount, sent, received, or pending. | Low |
| **F-07** | The user shall be able to send Gold to another user by inputting the receiver’s address and the amount in a text field. | High |
| **F-08** | The user shall be charged a transaction fee in SpartaGold which will be awarded to the validator. | High |
| **F-09** | The user shall be able to generate new random addresses for transactions. | High |
| **F-10** | The user shall be able to sign a message with their private key. | High |
| **F-11** | The user shall be able to verify the signature of a received message. | High |
| **F-12** | The user should be able to generate a QR code of an address. | Low |
| **F-13** | The system shall provide anonymity to users’ personal information. | High |
| **F-14** | The system shall implement a cryptographic protocol to ensure secure one-on-one conversations. | High |
| **F-15** | The system shall implement a proof-of-work function to deter attacks and phony validations. | High |
| **F-16** | The system’s wallet shall request password verification for access to it’s features. | Medium |
| **F-17** | The system’s wallet shall provide a new address when requested. | Medium |
| **F-18** | The system should allow users to transport their wallet addresses or private keys through print or mobile devices. | Low |
| **F-19** | The system shall allow users to mine through a set of console commands or buttons. | Medium |
| **F-20** | The system shall broadcast pending transactions to others connected to the network. | High |
| **F-21** | The system’s wallet shall still operate with limited functionality when offline by disabling transaction sending, transaction receiving, and ledger synchronization. | Medium |
| **F-22** | The system shall provide validators with a specified amount of SpartaGold when a block chain is discovered. | Medium |
| **F-23** | The system shall provide validators with a specified amount of Gold allocated from the transaction as a transaction fee when a block chain is discovered. | Medium |

**Table 1. System Functional Requirements**

**3.3 Non-functional Requirements**

**Usability**

|  |  |  |
| --- | --- | --- |
| **N-01** | The system shall let users complete a transaction within 3 clicks. | Medium |
| **N-02** | The system wallet’s UI shall be visually appealing (tested with a focus group), efficient by strictly following Fitts’ Law, and provide fast navigation. | High |
| **N-03** | The system wallet should not provide more than 3 buttons per page (File and Settings bar, window bar not included). | Low |
| **N-04** | The wallet shall support standard English QWERTY keyboard. | High |

**Table 2. Non-functional Requirement: Usability**

**Reliability**

|  |  |  |
| --- | --- | --- |
| **N-05** | The system shall update users’ ledger automatically when connected to the network. | High |
| **N-06** | The system shall connect users using a peer-to-peer network during the wallet’s startup in less than 10 seconds. | High |
| **N-07** | The system should check for internet connectivity every 10 seconds when offline. | Low |
| **N-08** | The system should not complete a transaction process and cancel a broadcast to the network when an error (ex: lost connection, power failure) occurs mid-transaction. | High |

**Table 3. Non-functional Requirement: Reliability**

**Performance**

|  |  |  |
| --- | --- | --- |
| **N-09** | The system shall broadcast transaction information to the network in less than 5 seconds after a user initiates a transaction. | Medium |
| **N-10** | The SpartaGold network should be accessible as long as an internet connection is live. | Low |
| **N-11** | The system shall send accurate and unaltered information across the network when connected to the internet. | High |
| **N-12** | The system shall listen for updates to the ledger at most every 10 minutes. | High |

**Table 4. Non-functional Requirement: Performance**

**Supportability**

|  |  |  |
| --- | --- | --- |
| **N-13** | The system should support updates without introducing new bugs (tested with regression testing). | Medium |
| **N-14** | The system shall provide the same support and functionality of previous updates when using the most recent updated system. | High |

**Table 5. Non-functional Requirement: Supportability**

**Security**

|  |  |  |
| --- | --- | --- |
| **N-15** | The system shall not provide private key information to anyone who is not the private key’s original owner. | High |
| **N-16** | The system shall be protected from Man-in-the-Middle attacks at all times. | High |

**Table 6. Non-functional Requirement: Security**

**3.4 Context and Interface Requirements**

**3.4.1 Development**

The system was designed to support users’ needs.  SpartaGold was developed strictly based on functional requirements.  The primary focus was to provide users safe, secure, and efficient transactions.  In addition, the system was developed with simplicity in mind, as SpartaGold’s target user base spans all branches of education.  Developing this system became a multi-stage process.  First, the main focus involved implementing the requirements and using them as guidelines in prototyping a system.  This was done by developing simple components via functions to carry out individual tasks such as identifying users, carrying out a transaction between two parties, signing a transaction, and more. Stage two in the development process focused on testing each component to verify they functioned as intended.  After each component was tested, the top level design was assembled and made available for university students.

**3.4.2 Testing**

Initial prototypes of the system were tested by the developers.  The developers recruited beta users and users then scouted for changes and made it such that the system was usable to the public.  All testing was conducted by the development team, SJSU Students, and SJSU faculty. Suggestions were submitted via email or communicated directly to the team for further improvement.

**3.4.3 Deployment**

Since the system operates on a peer-to-peer network, all users were required to download the SpartaGold Wallet in order to perform transactions.  The SpartaGold Wallet, after being rigorously tested by the developing team, was deployed throughout San Jose State University’s campus, for students and faculty members to use.  Once popularity grows, implementation and marketing will proceed for on-campus vendor trade and exchange.

**3.4.4 Interface**

An initial prototype interface was implemented alongside the system.  Basic GUI design principles were taken into account in order to design an interface that was efficient for users.  This prototype was tested for usability; once finished, it became an initial version.  Once the entire initial version of the system was made available for the users, we obtained feedback and made changes accordingly.

**3.5 Technology and Resource Requirements**

**3.5.1 Hardware Requirements**

***SpartaGold Wallet***

The SpartaGold Wallet does not require high level and expensive hardware.  Most desktop systems have enough processing power to operate the wallet and conduct transactions, assuming an internet connect is established. The minimum requirements to run the SpartaGold Wallet are as follows:

|  |  |
| --- | --- |
| **Component** | **Minimum Requirement** |
| CPU | 1GHz 32-bit (x84) or 64-bit (x64) |
| RAM | 1GB (DDR, DDR2, or DDR3) |
| HDD/SSD | 20GB (ATA or SATA) |

**Table 7. Hardware Requirement for SpartaGold Wallet**

***Mining***

CPUs and GPUs are used to solve a proof-of-work process in order to verify the amount of computational power used during validation.  However, CPU's are not used for mining anymore since they have relatively slow computational speeds compared to GPU, FPGA, and ASIC devices.  GPU's have the ability to perform fast computations with their multiple ALU's but come with a high cost.  The minimum requirements to mine for SpartaGold are as follows:

|  |  |  |
| --- | --- | --- |
| **Component** | **Minimum Requirement** | **Notes** |
| CPU | 1 GHz 32-bit (x84) or 64-bit (x64) | Needed to run the OS, not mining. |
| RAM | 1 GB (DDR, DDR2, or DDR3) | Needed to run the OS, not mining. |
| HDD/SSD | 20 GB (ATA or SATA) | Needed to store data of the OS, wallet software, and the block chain. |
| Motherboard | At least 1 PCI-E slot (4X, 8X or 16X) | More slots are recommended to increase the mining speed by using more than one card. |
| GPU | 1 GB RAM, 667 MHz clock speed | Clock speed is more important than the amount of memory. |
| PSU | ATX. Input 110/220VAC. Output +3.3, +5, ±12VDC. Power 750W | The system runs 24/7, so it is important to have a reliable PSU that meets all graphic card requirements. |
| Internet connection | 1.5Mbps | An internet connection must be stable. |

**Table 8. Hardware Requirement for Mining**

The introduction of FPGAs and ASIC devices, which offer much faster mining speeds in comparison to CPUs/GPUs, has put an end to CPU/GPU mining.  These devices consist of integrated circuits that can be configured by the customer.  They are programmed to do only one job, which can be allocated to mining for Gold, and thus provide a very fast mining speed.

**3.5.2 Software Requirements**

|  |  |
| --- | --- |
| **Software** | **Required for** |
| Windows OS, Mac OS, Linux OS | Environment for development |
| Netbeans IDE, Eclipse IDE | Development of SpartaGold software |
| Java 7 | Creation, compilation, and execution of SpartaGold code |
| Internet Explorer, Firefox, Chrome, Safari | Browser view and testing of an optional website for SpartaGold |
| SpartaGold-developed wallet | Execution of transactions and ledger updates |
| Android SDK, XCode | Mobile development and testing |
| TCP/IP , HTTP ,HTTPS , FTP | Network System |

**Table 9. Software Requirement for Users Computer**

**3.5.3 Dependencies**

Dependencies determine the order in which each task is performed and they expose bottlenecks in the entire development process.  The initial dependency for implementation was the design of the cryptographic protocol. Development started by creating different versions of the protocol until one was decidedly secure and efficient.  The next dependency was the design of the proof-of-work.  Different types of proof-of-work functions were analyzed from established cryptocurrencies to help determine a properly scoped proof-of-work for SpartaGold.  Another dependency was the need for a powerful computer. This computer helped during tests of SpartaGold’s proof-of-work function and the overall network.  The last dependency was the need for users.  Users were a critical necessity to the P2P network, as they provided feedback and overall interest and participation.  Users were needed to help test the UI and the network, and were recruited from students of San Jose State University.  These dependencies were dealt with at the beginning of the design process.

**3.5.4 Traceability Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirements** | **Test Case ID** | **Design Element**  **(Figure ID)** | **Implementation**  **(For future Use)** |
| **F-01** | TC-04, TC-05 | 4.3 |  |
| **F-02** | TC-03 | 3.5, 4.3, 4.4, 4.5 |  |
| **F-03** | TC-02 | 3.2, 4.8 |  |
| **F-04** | TC-08 | 4.9 |  |
| **F-05** | TC-08 | 4.1, 4.6, 4.9 |  |
| **F-06** | TC-05 | 4.4 |  |
| **F-07** | TC-01 | 3.1, 4.7, 4.11 |  |
| **F-08** | TC-08 | 4.9 |  |
| **F-09** | TC-06, TC-07 | 4.4 |  |
| **F-10** | TC-01 | 4.1, 4.6, 4.7, 4.8 |  |
| **F-11** | TC-02 | 4.1, 4.6, 4.7, 4.8 |  |
| **F-12** | TC-07 | 4.4 |  |
| **F-13** | TC-01, TC-06, TC-07 | 4.1 |  |
| **F-14** | TC-01, TC-02 | 4.1 |  |
| **F-15** | TC-01, TC-08 | 4.9 |  |
| **F-16** | TC-01, TC-02, TC-03, TC-04, TC-06, TC-08 | 4.4, 4.5, 4.6 |  |
| **F-17** | TC-06, TC-07 | 4.1, 4.4 |  |
| **F-18** | TC-06, TC-07 | none |  |
| **F-19** | TC-08 | 4.9 |  |
| **F-20** | TC-01, TC-02, TC-08 | 4.6 |  |
| **F-21** | TC-01, TC-02, TC-03, TC-04, TC-05, TC-06, TC-07, TC-08 | 4.1 |  |
| **F-22** | TC-03, TC-08 | 4.6 |  |
| **F-23** | TC-03, TC-08 | 4.6 |  |

**Table 10. Functional Requirements Traceability Matrix**

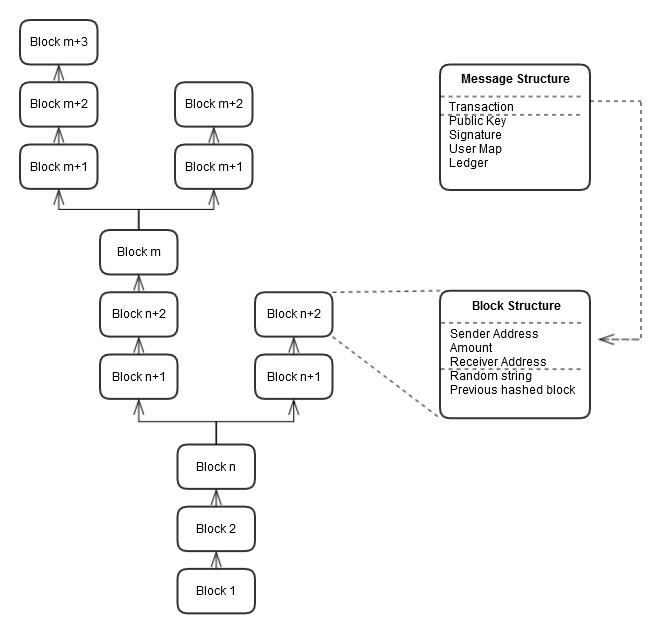
**3.5.5 Definitions, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| *address* | A randomly generated string of characters used to locate a specific connection in a peer-to-peer network |
| *Android SDK* | The Android SDK(software development kit) is a set of development tools used to develop application for Android |
| *ALU* | Arithmetic Logic Unit |
| *ASIC* | Application Specific Integrated Circuits |
| *ATA* | Advanced Technology Attachment |
| *ATX* | Advanced Technology eXtended |
| *block chain* | A long string of characters continuously edited and appended after each validation is finished |
| *Gold* | A unit of currency for SpartaGold |
| *CPU* | Central Processor Unit |
| *DDR* | Double Data Rate |
| *FPGA* | Field Programmable Gate Array |
| *GPU* | Graphics Processing Unit |
| *GUI* | Graphical User Interface |
| *HDD* | Hard Disk Drive |
| *IDE* | An integrated development environment |
| *ledger* | A document housing all previous transactions and block chains |
| *Message* | Communication between users in a network |
| *Mbps* | Mega-Bit Per Second |
| *mining/mine* | The process of validating transactions using a proof-of-work which awards validators with a specified amount of SpartaGold |
| *OS* | Operating System |
| *PCI-E* | Peripheral Component Interconnect Express |
| *private key* | A secret string of characters not shared with others, used to sign messages to prove authenticity |
| *PSU* | Power Supply Unit |
| *public key* | A public string of characters shared with other users, used to encrypt data and unlockable only through the use of the user’s complimentary private key |
| *RAM* | Random Access Memory |
| *SATA* | Serial ATA |
| *SpartaGold* | A digital cryptocurrency |
| *SSD* | Solid State Drive |
| *transaction* | An exchange of SpartaGold over the peer-to-peer network |
| *validator* | A user who commits processing time to complete a proof-of-work and validate a transaction |
| *wallet* | Software used to send, receive, store, and log Gold transmitted in relation to the user |

**Chapter 4: System Design**

# **4.1 Architecture design**

SpartaGold transactions are built around the block chain architecture. This block chain grows by one every time another transaction is completed. The longest chain is fundamentally shared among all validators, or “miners”, as a mining winner is decided based on the length of their chain.



**Figure 4.1:** The SpartaGold block chain architecture

A **block** contains string of five key parts: a sender’s address, an amount, a receiver’s address, a randomly generated string, and the previous hashed block in the chain. These parts are hashed together using SHA-1 and concatenated to the first 20 bytes in length.

The **block chain**, also known as the **ledger**, is a collection of sequential blocks. As branches of this chain emerge, discrepancies among who validated the transaction first arises. To relieve this contest, the validator with the longest block chain is awarded.

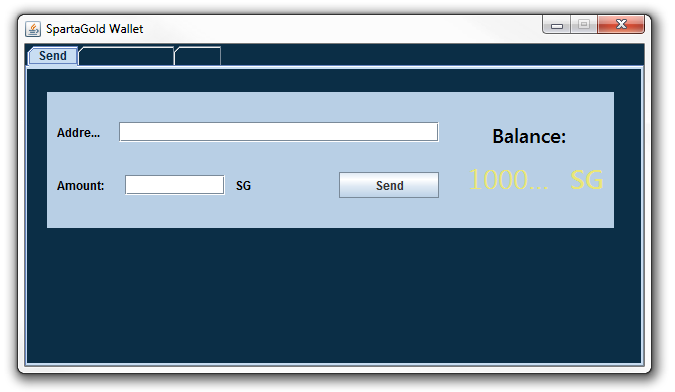
A **message** is a batch of files containing key parts for block chain processing, proof-of-work, and user balance updates. The files included in the message are the transaction file, a public key file, a signature file, a user map file, and a ledger file. The transaction file holds the sender’s address, the amount, and the receiver’s address in readable text. The public key and signature files are used in conjunction with the transaction file to validate the sender and receiver. The user map contains a map of all users in the system and their balances. Any tampering of this map can easily be overwritten by the copies stored on every other node in the network. The ledger file holds the block chain.

The **SpartaGold Wallet** automatically generates a user’s address based on their IP address. The Wallet is also the hub of all transaction activity and processing. Validation and proof-of-work is done within the “Mine” page, and Gold sending is done within the “Send” page.

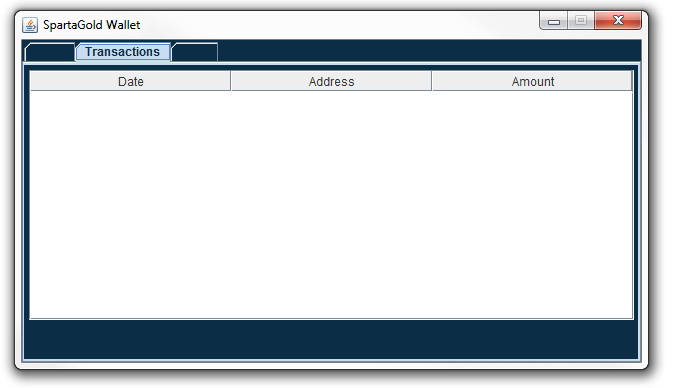
# **4.2** **Interface and component design**

## 4.2.1 Interface Mock Ups

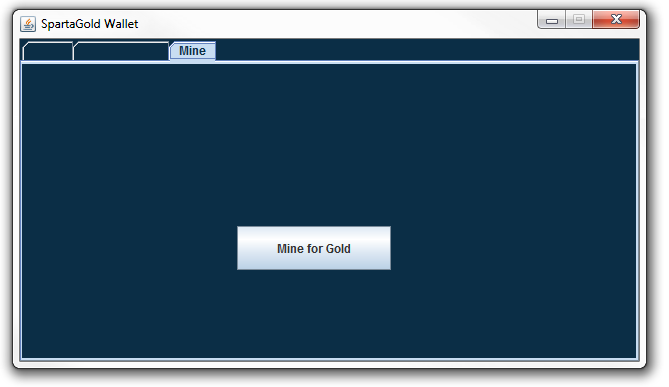
The SpartaGold Wallet has three main pages: “Send”, “Transactions”, and “Mine”. The “Send” page allows users to send Gold to specific users. The “Transactions” page allows users to view all past transactions. The “Mine” page contains the “Mine for Gold” button. The design is simple and concise, with minimal buttons. Each button is labeled with their function. The total balance of the user is displayed on the “Send” page of the Wallet.



**Figure 4.3:** “Send” page of the SpartaGold Wallet. This Send page includes a form with fields for the receiver’s address and the amount to send.



**Figure 4.4:** “Transactions” page of the SpartaGold Wallet. The Transactions page displays all past transactions as a list by date, address, and amount.



**Figure 4.5:** “Mine” page of the SpartaGold Wallet. The “Mine for Gold” button begins the validation process by listening for broadcasted messages and performing the proof-of-work. This process ends once the Wallet is closed.

## 4.2.2 Peer-to-Peer Component Diagram

|  |
| --- |
|  |

**Figure 4.6:** A component diagram of the peer-to-peer network. Users are all interconnected, as there is no central server or authority to designate transactions.

# **4.3 Structure and logic design**

The structure and logic design of the SpartaGold system can be conveyed by the UML diagrams: use cases, sequence diagram, class diagram, state diagram, package diagram and control flow diagram.

## 4.3.1 Use Cases

The use cases below walks us through the main steps of the three main processes of the SpartaGold system. Each use case consists of participating actors, flow of events, entry conditions, exit conditions, and quality requirements. These use cases help us define and validate our requirements which is a great first step towards a successful project.

|  |  |
| --- | --- |
| **Use case name** | **SendGold** |
| Participating actors | Initiated by Bob  Communicates with Alice |
| Flow of events | |  | | --- | | 1. Bob gets the address from Alice. | | 2. Bob states in the message the amount of Gold he has, the change he wants to receive, and the transaction fee. | | 3. Bob hashes the message, appends a timestamp to the resulting hash, and signs with his private key. | | 4. Bob gets the original message, the results from step 3, the timestamp, and sends everything as a single message encrypted with Alice’s public key. | | 5. Alice receives the message and decrypts the message with her private key. | |
| Entry condition | * Bob and Alice must have public and private keys |
| Exit conditions | * Bob has received an acknowledgement and his change from the transaction OR * Bob has received an explanation indicating why the transaction could not be passed. |
| Quality requirements | * Bob’s message is acknowledged within 10 minutes. * The selected message arrives no later than 5 seconds after it is sent Bob. |

Table 11. Use Case 1

|  |  |
| --- | --- |
| **Use case name** | **ReceiveGold** |
| Participating actors | Initiated by Alice  Communications with Bob and Miner |
| Flow of events | |  | | --- | | 1. Alice gives Bob her address. | | 2. Bob prepares the message and sends it to Alice. | | 3. Alice receives the message, decrypts it, and broadcasts it to the peers in the SpartaGold network for validation. | | 4. The Miner receives the message of Alice, validates it, updates the public ledger, and notifies everybody by broadcasting it. | | 5. Alice’s total amount of Gold gets updated and she then completes the transaction with Bob. | |
| Entry condition | * Alice must have an address * SpartaGold network must have people available to validate each transactions. |
| Exit conditions | * The public ledger has been updated. * Alice has received an acknowledgement regarding the validation. |
| Quality requirements | * Alice’s broadcasted message is acknowledged within 10 minutes. |

Table 12. Use Case 2

|  |  |
| --- | --- |
| **Use case name** | **ValidateTransaction** |
| Participating actors | Initiated by a Miner  Communications with Alice and the SpartaGold network |
| Flow of events | |  | | --- | | 1. The Miner is told about the broadcasted pending transaction of Bob and Alice. | | 2. The Miner begins the validation process by trying to solve a proof-of-work puzzle. | | 3. Around 10 minutes later, the Miner finds the answer to the puzzle and broadcasts it to the SpartaGold network along with the transaction message. | | 4. All peers in the SpartaGold network update and record a copy of the ledger. | | 5. The Miner is awarded the transaction fee and newly generated Gold from the SpartaGold network. | |
| Entry condition | * The Miner has a copy of the pending transactions and ledger. |
| Exit conditions | * The Miner’s total amount of Gold is updated to include the awarded Gold and transaction fee. |
| Quality requirements | * Proof-of-work puzzle must be computationally difficult to solve. |

Table 13. Use Case 3

## 4.3.2 Sequence Diagrams

The sequence diagrams below tie our use cases with our objects. They explain how the behavior of a use case is assigned among its participating objects. The first sequence diagram in figure 4.7 shows how the user will send an amount of SpartaGold to someone in the SpartaGold network. The second sequence diagram in figure 4.8 shows how the user will receive SpartaGold from someone in the SpartaGold network. Last but not least, in figure 4.9, the sequence diagram shows how a user will mine Gold in the SpartaGold network.

|  |
| --- |
| Send Bucks.jpg |

**Figure 4.7:** Send Gold through the network by entering information into the wallet.

|  |
| --- |
| Receive Payment1.jpg |

**Figure 4.8:** Receive Gold from another user by creating a personal address.

|  |
| --- |
| validate.PNG |

**Figure 4.9:** Validate Transactions

## 4.3.3 Class Diagram

In figure 4.10, the class diagram shows the essential classes and the relationships among them. For example, a user is associated with a ledger because he/she reads from it, and also, the user is associated with messages because he/she writes them. The same idea goes for the miner. The miner reads and writes to the ledger. The miner also creates the blocks of transactions that make up the ledger by validating them before writing them to the ledger.

|  |
| --- |
| class diagram.png |

**Figure 4.10:** Class Diagram

## 

## 

## 4.3.4 State Machine Diagram

The figure below shows the state diagram of the SpartaGold node as it listens for a connection. At first, the node is listening for any nodes trying to connect to it. Once it notices that there is a node in the network trying to connect, it creates a connection and it is ready to handle incoming messages. When a message is received, it calls the appropriate methods and then it closes the connection.

|  |
| --- |
| uml_node_listening.png |

**Figure 4.11:** State Diagram - SpartaGoldNode Object

## 4.3.5 Package Diagram

The classes used to build SpartaGold are organized by using packages. The figure below shows the hierarchy of the six packages.

|  |
| --- |
| uml_packages.png |

**Figure 4.12:** Package Diagram - Hierarchy of Class Packages

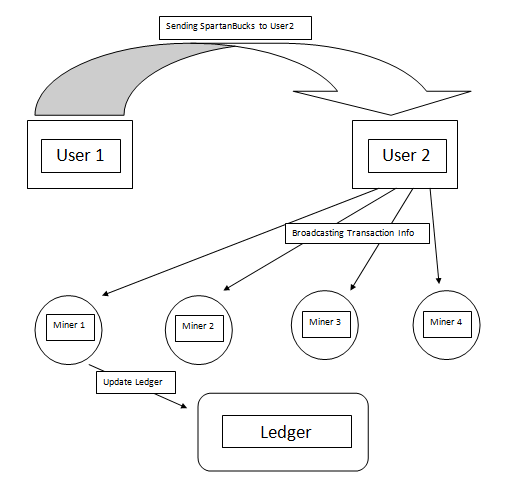
## 

## 4.3.6 Control Flow Diagram: Sending SpartaGold

|  |
| --- |
|  |

**Figure 4.13:** Control flow diagram of the process of sending SpartaGold.

## 4.3.7 Context Diagram

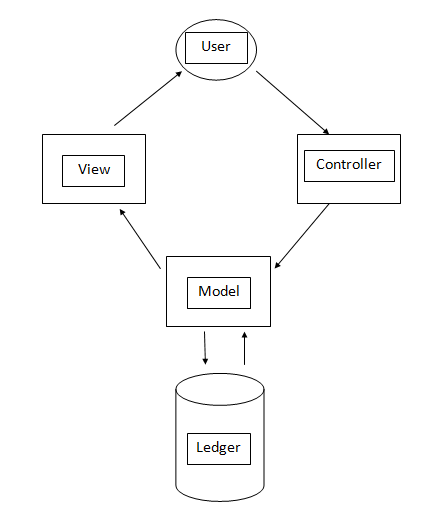


**Figure 4.14:** Context diagram for SpartaGold.

## 

## 4.3.8 Data Flow Diagram

For the SpartaGold wallet we will use the Model View Controller (MVC) pattern for our implementation. Below is the data flow diagram of SpartaGold wallet.



**Figure 4.1:** MVC data flow diagram for SpartaGold Wallet.

# **4.4 Design Constraints, Problems, and Trade-Offs and Soluti**o**ns**

## 4.4.1 Design Constraints

**Society**

The system is intended for San Jose State University Students. Therefore, the system is available for such individuals for the purpose of this project. However, since the system is completely online and dependent on internet access, privileged users are able access their wallet and perform transactions anywhere.

**Environment**

As this system is purely software oriented, there are no environmental constraints. Privileged users are able to access their account and perform transactions anywhere around the world as long as they have an internet connection.

**Economic**

From a financial perspective, it was important to reduce the amount of overhead to implement the system. Overhead depends on the amount of computing power, as well as the size of each transaction. For example, assuming each transaction takes up 1KB as opposed to a transaction taking up 2KB, users’ hard drive could possibly fill up twice as fast, requiring more frequent hard drive purchases. In the case of the system, the cost will only increase as users will require an increasing amount of space to store transaction data.

Economically, the P2P network reduces the need for a central authority, which greatly reduces the costs of conducting transactions. Central authorities collect transaction fees to mitigate costs for services, insurances, and interests. With a decentralized network, the service and insurance is provided by the users for the users. Transaction fees are implemented only as incentive for users to validate quickly and efficiently.

**Hardware**

As the program itself is relatively small, each user of the network is required to keep track of all transactions. As transactions are rapidly accumulating, there is no definite requirement of space to hold all transactions. The amount of space will always be increasing.

* No environmental constraints as the system is implemented via online.
* Hardware and software constraints (the amount of data to be stored is only going to increase).

**Software**

Software-wise, SpartaGold is compatible with Linux, Windows, and MAC operating systems. Most users have one of the three operating systems as these are the most common to the public. The program is composed of Java code, so the latest JDK or JRE is highly recommended.

## 4.4.2 Design Problems and Challenges

**Economic**

Deflation or inflation was one of the economic design problems. Cryptocurrencies are subject to the law of supply and demand. Putting a cap on the number of SpartaGold circulating causes deflation by increasing the value since the users keep hoarding the Gold instead of spending them. Placing a cap also puts another challenge of exhausting money one day in our way. Thereby, SpartaGold designers decided to not put a cap. However, this approach is vulnerable to inflation since every time more money is pumped into the system, the value of a SpartaGold decreases. Considering all factors, SpartaGold does not put a limit on the amount of Gold that may be in circulation; this is to provide a more economically secure system since the supplies never end and it is more appealing to the users.

Also, as money supply growth is deterministic, changes in money demand are reflected in the exchange value of the currency, raising or lowering the cost of producing the next coinbase as the protocol adjusts the difficulty up or down in response to the entry or exit of hashing power. So the exchange value of the mining determines the marginal costs rather than the other way round.

**Resources**

The SpartaGold system runs across a large network of computers and requires a considerable amount of electricity for miners to validate transactions. Therefore, users and miners need computing resources to help keep the system up and running. The miner is paying for electricity today for an award he will get at some point in future.

**Society and Environment**

The development of SpartaGold, like other digital currencies, will be an ongoing tug of war with public power and the domination system, driven by users' needs and the changing times. Leading resistance to digital currencies are banks with concerns that one day it might eliminate the existence of central banks.

**Hardware/Software**

For software, designing a secure protocol that is not vulnerable to double spending, DOS attacks and man-in-the-middle attacks was a very challenging problem. Also, using a cryptographic hash function for a digital wallet and the proof-of-work was another design constraint. An alternative implementation was to store coins in hardware wallets, but that was too challenging since it involved hardware which costs money.

**Safety**

The first challenge of safety was how to leverage the deflation and inflation of SpartaGold to assure the user that their money is secure. The first consideration was how much money the miners pump into the system each time they validate to assure the stability of the system. Also, designers considered how to prevent coordination of large pools of miners who were collaborating together to mine because it can be a security issue if the pool is too large.

**Reliability**

It is important for SpartaGold to be reliable so users can feel confident that their money will not go to waste and thus increase the success of the system. One important characteristic is making sure that the system is up for most of the time. This is crucial since we want users to feel confident that their transactions will be sent across the network and validated as well. Also, having the system cryptographically secure will increase its reliability. The system was implemented with cryptographic standards to assure components are not vulnerable to attacks.

## 4.4.3 Design Solutions and Trade-Offs

**De-centralized Authority**

Instead of implementing a system based around a centralized authority, a peer-to-peer network was integrated. With a central authority, transactions would require a bank to conduct its transactions. In addition, the user does not need to keep track of their history of transactions as it would be stored through the bank. However, a major dilemma to this approach is non-anonymity; the bank will always be monitoring each customers’ funds - especially when they use a debit or credit card for a purchase. Also, it puts more overhead and responsibility on the people in charge as they will be responsible for keeping track of all the transactions. This increases the amount of space and computing power needed in order to keep everything running efficient for the rest of the users on the network. There is a much larger risk to a centralized authority as well. If the central server goes down in a worst case scenario, the entire network falls with it, becoming unavailable for users. As this approach is traditional, the problematic aspects revealed a decentralized network to be the optimal choice for SpartaGold. In a decentralized network, all data and validation of transactions are processed and saved by every user in the system. In order to achieve this, SpartaGold was designed to be a peer-to-peer network. A peer-to-peer network will allow for high anonymity for every user by hashing transactions and establishing a public/private key for any encryption needed. This removes responsibility from what would be a central source and distributes the responsibility to everyone in the network.

**Processor Speed**

As mentioned earlier, mining for SpartaGold efficiently requires a powerful computer with extreme computing ability. A user has multiple options, but this comes down to the user’s budget. Assuming that money and computing speed are directly proportional, more money invested in a faster computer will yield more efficient mining due to faster processing speed which directly correlates to more SpartaGold in a shorter period of time.

**Economic Solution**

There were two solution options: either put a cap on the amount of SpartaGold that can be found/generated in the economy, or allow for an infinite amount of Gold to be in circulation. A cap would gradually cause deflation on the currency, due to the limited availability of Gold. Just as real-world gold goes up in value due to a finite amount, SpartaGold will also increase in value. On the other end, creating a capless currency would consistently inflate the value of each Gold.

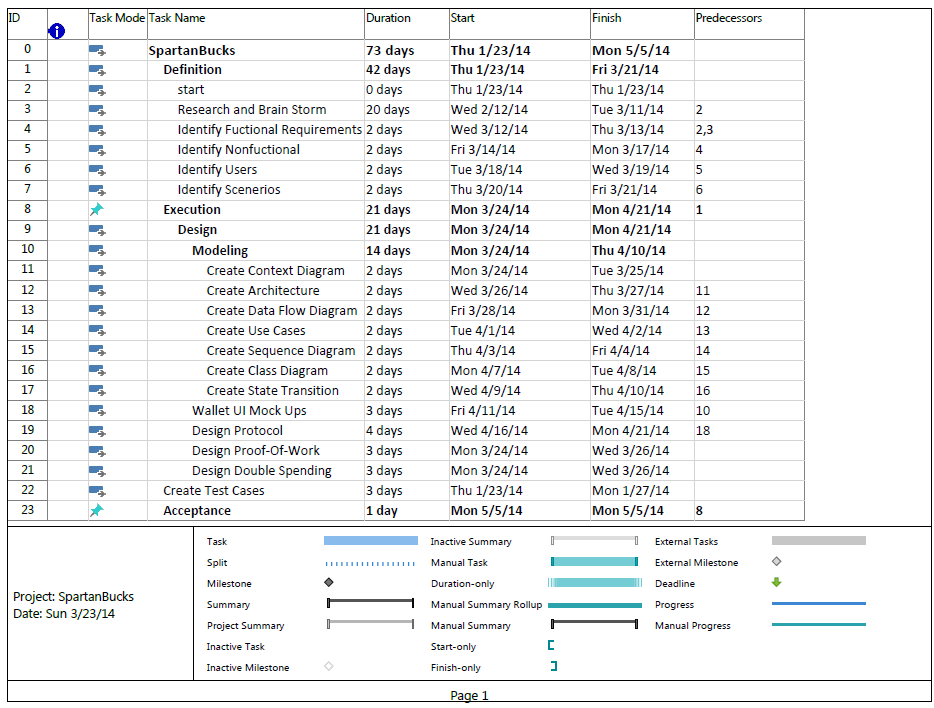
After some deliberation, the latter solution was implemented into the system. No cap was integrated, and the currency is expected to inflate over time. However, this is not necessarily a problem, as the amount of currency allowed in the system can be controlled to ease any difficulties with inflationary currency. In addition, avoiding the inherent problems with a capped currency was in our best interest. Capped currencies grow in value per unit, but this causes many holders of this currency to spend less, causing stagnation in the economy. Volatility arises from this limitation, and each unit of a capped currency explodes and diminishes in frightening spikes.

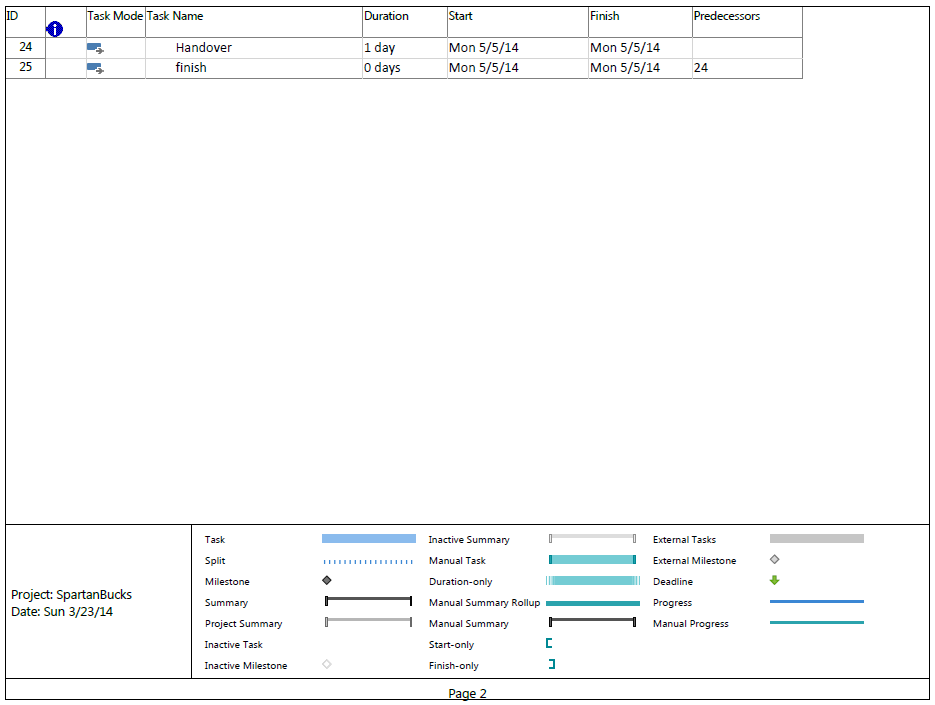
**Chapter 5: Project Plan and Schedule**

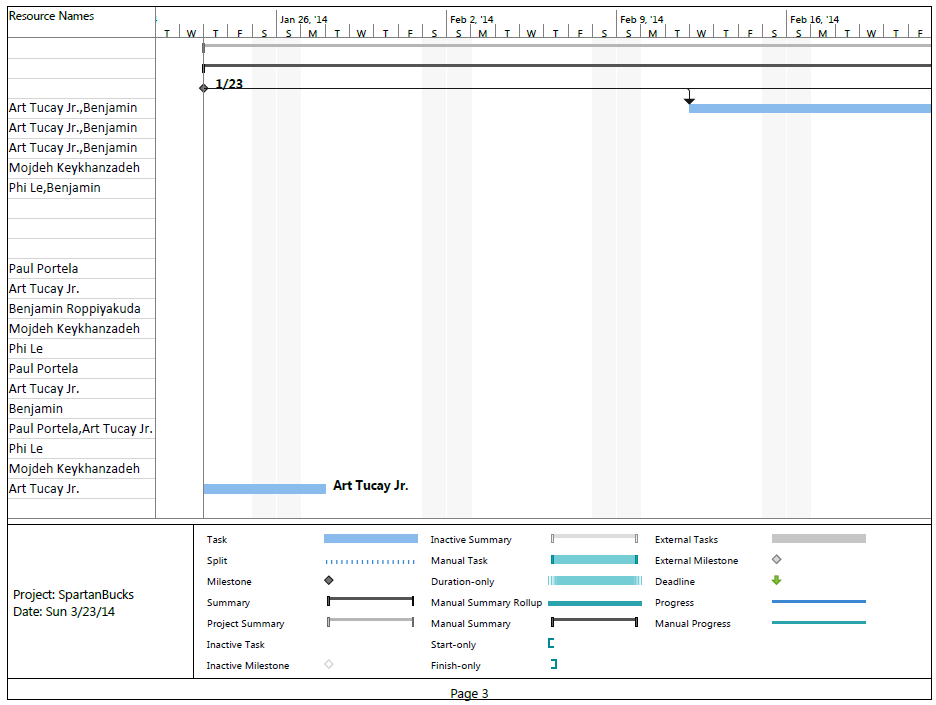
**5.1    Project team**

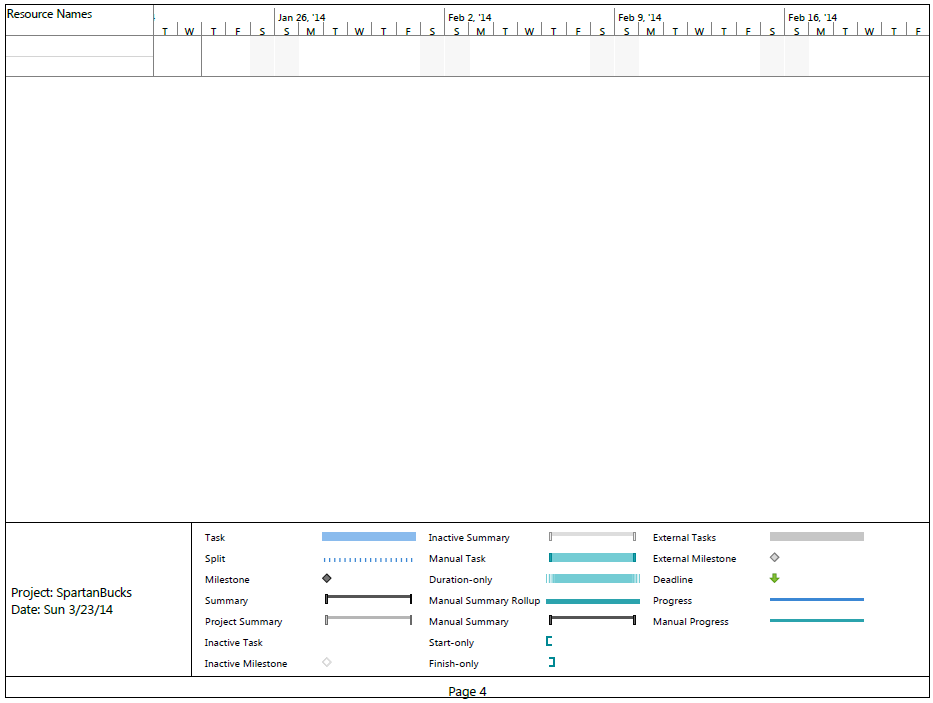
Our team consists of five San Jose State University students. Art Tucay Jr., Paul Portela, and Mojdeh Keykhanzadeh are working towards their degree in software engineering, and Benjamin Roppiyakuda and Phi Le are working towards their degree in computer engineering. Our project adviser is Dr. Thomas Austin, and our project manager is rotated among the entire team.

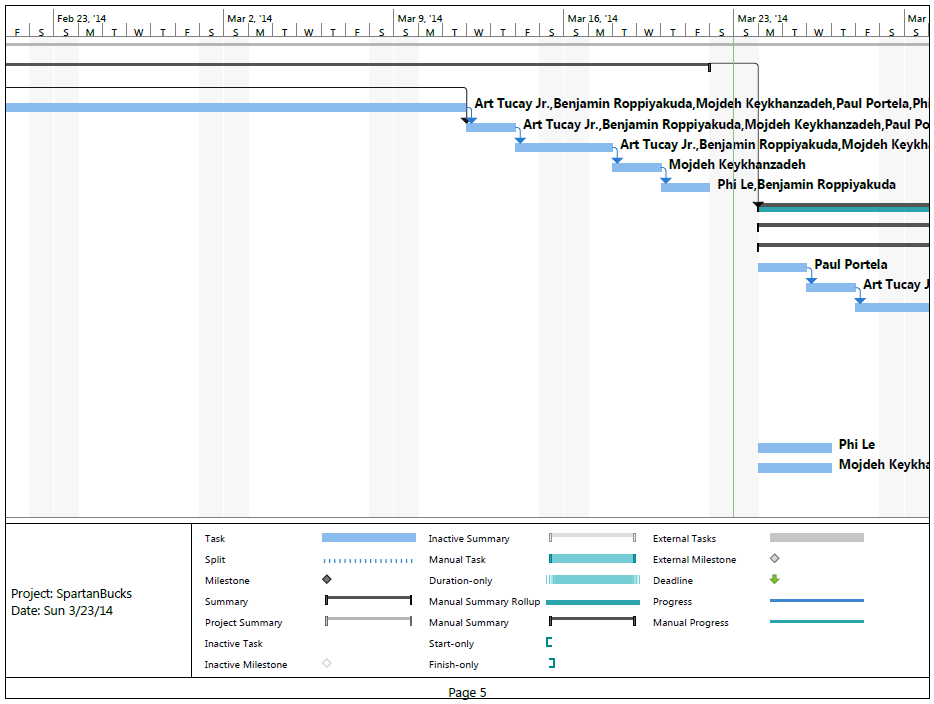
**5.2    Project tasks and schedule**

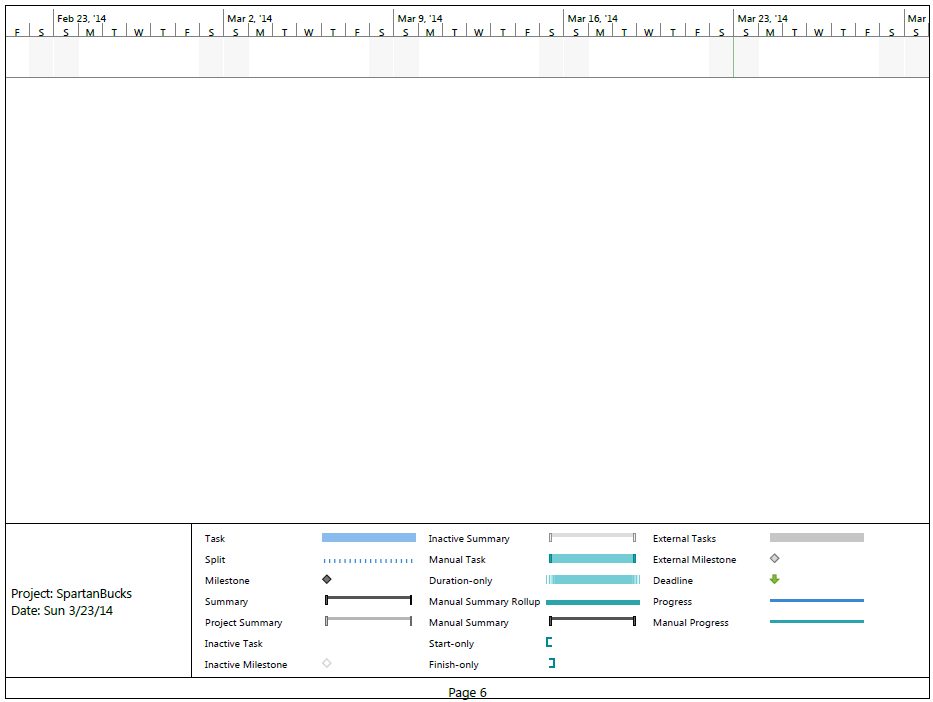


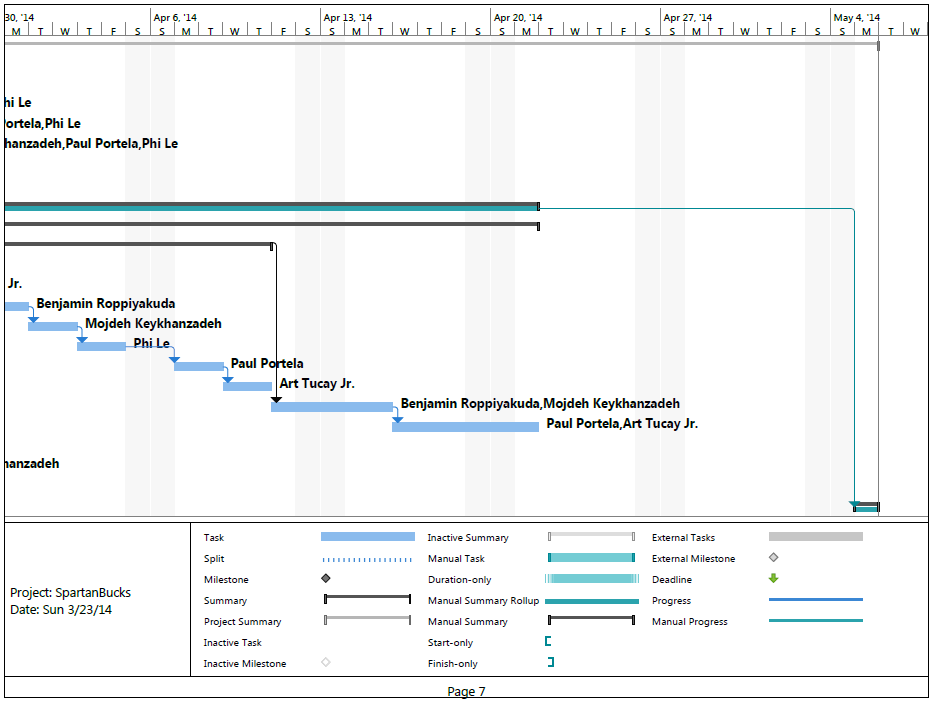


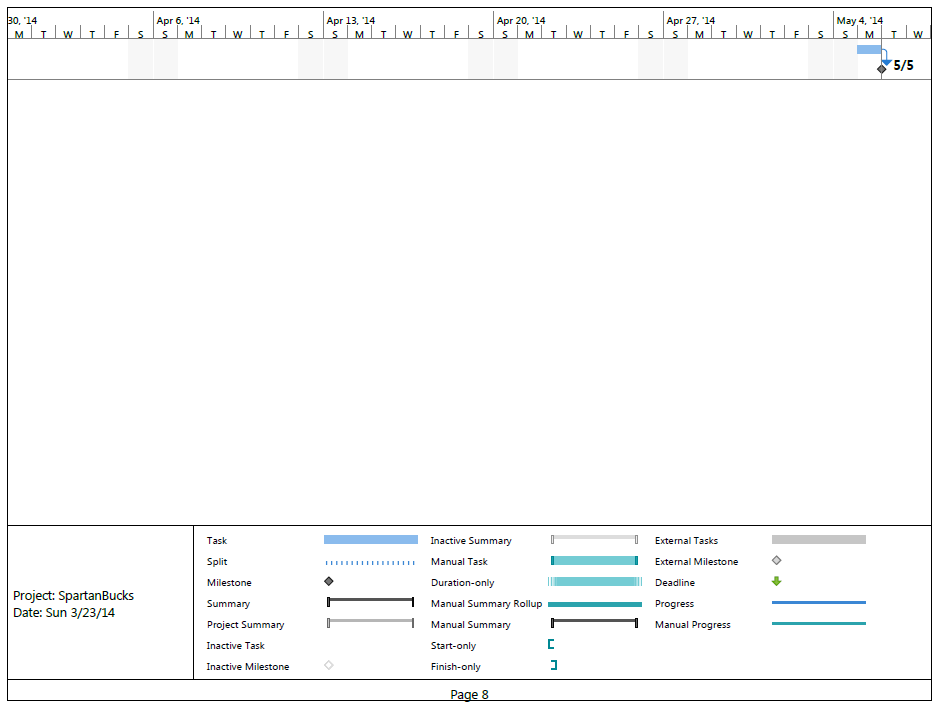












**Chapter 6: Tools and Standards**

**6.1.   Tools Used**

All software will be written in Java using Eclipse IDE and Oracle Java development kits.

**6.2.   Coding Standards**

Coding standards focus specifically on how us as developers are writing our code. The following standards listed are to be used for the scope of this project and are important in assuring an accurate and desirable product.

**6.2.1   Coding Indentation**

Indentation is essential to code organization. It drastically improves readability toward the developers. The purpose for our team indenting is to:

-    Distinguish one or more control statements (i.e. any kind of loop or recursion)

-    Distinguish one or more condition statements (i.e. if statement or )

-    Distinguish a new scope (function/class/struct, etc.)

A standard indentation shall be a tab. Multiple tabs shall be used when nesting control statements or conditions. As a tab is a consistent amount of space, it will simplify implementing the organization of our code as we will not have to repeatedly hit the spacebar to reach a proper indent.

**6.2.2   Inline Commentation**

Inline comments explain a particular section of code. Comments could be used for explaining function definitions and how they work, particular algorithms, or any part of the code that may not appear clear and concise.

**6.2.3   Structured Programming**

Our group shall use structured programming. The system shall be organized into modules. Each module is a top level view of major functionalities. All modules shall be further divided into sub-functionalities. Each sub-functionality shall contain classes and functions pertaining to its operation.

**6.2.4   Classes, Functions, and Methods**

Our classes, functions, and methods are to be reasonably sized to avoid complexity. They are reasonably sized in the sense that our functions and methods are not long to make it easier for us to fix bugs. It is to our advantage to have small functions and split up methods to smaller parts since it enhances our readability

**6.3   Coding Guidelines**

We adhere to java coding standards. We aim to complete a line of source code from reaching more than one line. Our code shall implement proper spacing within the code itself.When a line of code exceeds the width of a single line of coding space, we use wrapping line techniques.

**6.3.1   Line Length**

As our group understands simplicity as an essential factor in undertaking this project, we shall make it a point to always aim to complete a line of source code from reaching more than one line. In some situations we will not be able to meet the requirement, so we shall deal with this situation accordingly (see section 6.3.3).

**6.3.2   Spacing**

Our code shall implement proper spacing within the code itself. An example is as follows:

#include <iostream>  
int main()  
{

     int a, b, c;

    a = 1;

    b = 2;

    std::cout << "Hello World!";

    c = a + b;  
}

Multiple variable declarations on a single line shall be separated by a space. Variable assignments on each side of the assignment operator shall be separated by a space. Any operation involving variables, functions, and/or classes shall have spaces in between each operator.

**6.3.3   Wrapping Lines**

When a line of code exceeds the width of a single line of coding space, we deal with the situation accordingly:  
-    Begin a new line after a comma

-    Begin a new line after an operator symbol

After starting a new line, it shall be indented to the opening of a parenthesis, an assignment operator, or a comment. An example is as follows:

*#include <iostream>*

*int functionA (int a, int b, int c, int d,*

*int e, int f, int g);  
int main()*

*{*

*int a, b, c, d, e, f, g;             // There are unused variable declarations but this      // example should demonstrate our standards*

*a = 5;*

*b = 10;*

*c = a + b + a + b + a + b + a + b + a +*

*b + a + b + a;*

*}*

**6.3.4   Variable Declarations**

Variable declarations occupying multiple lines shall always be preceded by a data type.

**6.3.5   Program Statements**

Individual programing statements shall be limited to one per line. This includes anything from variable declarations to function / class definitions.

**6.3.6   Use of Parenthesis**

In order to ensure user friendly readability for our group and for others looking into our code, parenthesis shall be included involving any arithmetic operation to distinguish the order in which it executes, even if it would execute correctly without them.

**6.3.7   Inline Commentation**

As mentioned earlier, inline comments improve program readability. This would be particularly beneficial for someone who is not up to date with particular functions or sections of the project to get up to speed quickly. Our comments shall provide an overview of a particular section of code when required. The inline comments shall take up no more than 25% of our actual code; any higher would make finding the actual code difficult.

We use Javadoc guidelines for inline commenting. In general, Javadoc standards

for any multi line comments is (“\\*....\*\”). For single line comment, “\\” is used.

**6.3.8   Coding for Efficiency and Coding for Readability**

Given the purpose of our project and what it entails, we shall code our project such that it runs as efficiently as possible. In addition, we predict there will be many updates and bugs needed to be addressed as we go along, so our code shall also be relatively simple and easy to maintain. This includes minimizing the amount of objects needed to be changed when a minor modification needs to be made.

**6.3.9   Reasonably Sized Functions and Methods**

Our program  won’t contain an excessively large number of lines of code.Our is

written in the way to perform specific tasks. If they become too long, then chance

are the task being performed can be broken down into subtasks which can be

handled by new routines or methods.