BLOCKCHAIN TRAINING ALLIANCE

Kris Bennett, Mike Rieger, Ernesto Lee

Certified Blockchain Solution Architect (CBSA)

Official Exam Study Guide

This exam study guide covers exam objectives, including Blockchain technology, landscape, and design patterns. The guide includes how to choose the appropriate blockchain system for various use cases.

Exam guide includes:
- Eight Chapters
- Key Terms
- Exam Practice Questions

Certified Blockchain Solution Architect (CBSA), Official Exam Study Guide
By: Kris Bennet, Mike Rieger, Ernesto Lee

Book is published by Blockchain Training Alliance, Inc.

Limit of Liability / Disclaimer of Warranty / Terms of Use
While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. There are no warranties which extend beyond the descriptions contained in this paragraph. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not apply or be suitable for your situation. You should consult with a professional where appropriate. The accuracy and completeness of the information provided herein and the opinions stated herein are not guaranteed or warranted to produce any particular results, and the advice and strategies contained herein are not suitable for every individual. By providing information or links to other companies or websites, the publisher and the author do not guarantee, approve or endorse the information or products available at any linked websites or mentioned companies, or persons, nor does a link indicate any association with or endorsement by the publisher or author. This publication is designed to provide information with regard to the subject matter covered. It is offered or sold with the understanding that neither the publisher nor the author is engaged in rendering legal, accounting, investment, or other professional service. If legal advice or other expert assistance is required, the services of a competent professional should be sought. This publication is no guarantee of passing this exam or other exam in the future. Neither the publisher or the author shall be liable for any loss or loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

Table of Contents

Chapter 1: What is Blockchain?

In this chapter we cover what Blockchain is, the core concepts that Blockchain technology is built upon, and the high-level mechanics of Blockchain itself.

The Basic Principles

To understand Blockchain from a human perspective it is important to review the people who inhabit Yap Island and their unique currency called Rai Stones. These stones could not be physically traded so the Yapese people used mental ledgers where all tribe members kept a copy of the ledger in their head.

Everyone knew who owned which Rai stone at any time on the island. When two parties wished to transact, they would announce their transaction to the tribe. When a transaction was announced, all tribe members updated their mental ledgers and the tribe went about their daily lives.

Centralized vs Decentralized

The Yapese could have kept a single ledger, just like a bank, where one tribe member would keep an account or ledger of all transactions for the tribe. This person would have to be very trustworthy as the entire tribe would have to trust them as the record keeper. This is known as a centralized ledger or system. If the single copy of the ledger were changed by any means, wealth would be lost (or gained) unfairly. Another important point with a centralized system is that if the record keeper is ill or not present, no transactions can take place. This creates a single point of failure in the system.

With a decentralized ledger, nobody has to trust anyone else, a trustless environment is assumed from the beginning as all tribe members have a copy of the same ledger regardless if they know one another. This is known as a decentralized ledger or system. Some members may have corrupt or incomplete data, this is okay as the majority will not accept it. Some members may not be present, that's okay. They can get caught up when they come back online.

This all works with a decentralized ledger as the whole tribe must reach consensus on the truth in order to update the ledger. This is called "Group Consensus". The truth is assumed to be the version of the ledger that 51% or more of the tribe members present agree on.

One day a ship carrying a new stone back to the island sank in the harbor. The tribe decided to add it to the ledger and trade it just like any other stone even though it was at the bottom of the harbor and untouchable physically. The Yapese established a key principle of Blockchain, possession does not equal ownership.

The Mechanics of Blockchain

To understand the mechanics of Blockchain we can look at what makes a bank in a modern economy successful. A bank takes deposits and issues credits which also decouples possession and ownership. A bank provides a trustable ledger to all parties, recording all deposits and credits. A bank also acts as a trust broker when two parties who don't trust each other want to trade.

A decentralized ledger provides the same benefits as a bank!

What is a "block"?

Let's say all transactions are recorded on paper and each sheet of paper has 25 lines. When a sheet is filled up (25 transactions), the tribe will "validate" the transactions on this current page. Essentially, does everyone agree with the data on the page?

If the majority of the tribe agree that the 25 transactions are all the same it is validated via group consensus. Once the page has been validated, it is added to a stack of previously validated sheets. Each sheet on the stack can be assumed to be trustworthy because once a sheet is validated it can't be changed by linking the sheets together.

How are blocks "chained" together?

To link our sheets together we embed information from the previous sheet of paper into the new, recently validated sheet. In Blockchain, our sheet of paper is equal to a block. The act of embedding a previous block of information into the current block of information is called chaining. Hence, the name Blockchain.

To chain blocks together today, all data in a block is run through a special function called a "cryptographic hash". Cryptographic hashes create a unique output for a specific input. Therefore, the hash of each block will always be unique based upon the inputs.

To link or chain blocks of data together the header of the current block contains the hash of the last (validated) block. Changing the data on any block in a Blockchain will result in a completely different hash and the new hash will not match the hash in the next block header thus breaking the Blockchain and invalidating all blocks linked to where the change was made. This give Blockchain its property of immutability (can't be changed) and makes it highly censorship-resistant.

Blockchain Is…

So what is Blockchain? Blockchain began as an idea documented in a whitepaper by the anonymous Satoshi Nakamoto. The ideas outlined in this whitepaper lead to the world's first and largest Blockchain – Bitcoin. At its heart, Blockchain is a record keeping system that can record the transfer of "tokens" or "coins" monetary wealth . Bitcoin and other cryptocurrencies such as Ether, LiteCoin, and Monero are current examples of this. Blockchains provide a successful enabling platform for cryptocurrencies by providing a digital immutable ledger that is widely distributed and peer-validated. As each of the above mentioned cryptocurrencies exists on its own Blockchain, they cannot be directly exchanged for one another. It is critically important to note that a Blockchain does not require a currency and many exciting and compelling use cases exist which require no special currency, coin, or token.

Blockchain is a record keeping system that can record the transactions of importance that are non-monetary. Transfer of ownership, an update to a medical record, capturing a training certification and recording important single-party announcements are examples of this.

The three types of Blockchain transactions are:

- Two or more parties, exchange of monetary value such as cryptocurrency
- Two or more parties, but no exchange of monetary value such as updates to medical records, notary services
- One party announcing an important event such as supply chain management, business process automation, creation/auditing of financial records.

Blockchain can also be an event tracking system where announcements mark events and events can be actionable through the use of Smart Contracts/Chaincode which is nothing more than software programmed events.

By using Smart Contracts/Chaincode, Blockchain can also be a workflow platform by writing rules around events. In order for a client app to communicate with a Smart Contract, the public address of that contract must be known by the user or the application the user is invoking contract functionality from.

Chapter 1 Study Points

Below is the list of topics of importance for this chapter, additional research should be conducted prior to taking the CBSA exam:

- What is a centralized ledger?
- What is a decentralized ledger?
- Differences between a centralized and decentralized ledger?
- What is a block?
- The difference between possession and ownership
- Double vs. Triple Entry accounting
- What is Group Consensus?
- How are blocks chained together?
- 3 types of Blockchain transactions

Chapter 1 Key Terms

Below is the list of key terms for this chapter:

- Group Consensus
- Triple Entry Accounting
- Blockchain
- Event tracking
- Centralized Ledger
- 51%
- Possession
- Block
- Record keeping
- Decentralized Ledger
- Double Entry Accounting
- Ownership
- Chaining

Chapter 1 Quiz

1. Blockchain is:

a. A record keeping system.
b. An event tracking systems.
c. A workflow platform.
d. a and b
e. All of the above.

2. Double-entry accounting is:

a. A method for tracking information
b. A record for tracking debits and credits
c. A record for tracking debits, credits, and an immutable link to all past debits and credits
d. The ledger for an investor's portfolio that includes the current price of their investment

3. Select which statement is true about Blockchain:

a. Changing the data on any block will result in a different hash
b. A block is made up of 25 lines or records
c. Chain refers to the program that physically links blocks together

4. Group Consensus is reached when how many members agree?

a. 10%
b. 51% or more
c. 100%
d. 49% or more

5. Blocks in Blockchain are "chained" together by:

a. Adding a pointer in the ledger to the previous block.
b. By taking a picture of the block being added with the block behind it.
c. By embedding the genesis block into the header of the new block.
d. By hashing the previous block and embedding that hash into the new block's header.

6. Which of the following is a type of Blockchain transaction?

a. Two or more parties, exchange of monetary value such as cryptocurrency.
b. Two or more parties, but no exchange of monetary value such as updates to medical records.
c. One party announcing an important event such as supply chain management, business process automation, creation/auditing of financial records.
d. All of the above are a type of Blockchain transaction.

7. A decentralized ledger acts as a trust broker, like a bank.

a. True
b. False

8. Blockchain was documented and released via a whitepaper by:

a. Bill Gates
b. Satoshi Takamoto
c. Larry Ellison
d. Satoshi Nakamoto

9. Blockchain is the same thing as Bitcoin.

a. True
b. False

10. Blockchain provides the same services as a traditional bank, except for:

a. Taking deposits and issuing credits
b. Acting as a trust broker in exchanges
c. Decoupling possession and ownership
d. Providing Certificates of Deposit

11. Hyperledger and Ethereum both went live in which year?

a. 2008
b. 2009
c. 2012
d. 2015
e. None of the above

Chapter 1 Solution for the Quiz:

Q1: E
Q2: B
Q3: A
Q4: B
Q5: D
Q6: D
Q7: A
Q8: D
Q9: B
Q10: D
Q11: D

Chapter 2: How Does Blockchain Work?

In this chapter we cover what the high-level benefits of Blockchain are as well as the drawbacks or challenges of Blockchain. A deep dive into cryptography; cryptographic hashing is covered. A deep analysis of the various Blockchain group consensus mechanisms are explored with a focus on Proof of Work (PoW) and Proof of Stake (PoS).

Benefits of Blockchain

There are a number of benefits to Blockchain solutions which include being publicly verifiable, secure, transparent and cost effective. Blockchain also provides tokenization that can enable organizations of all sizes to create tradable tokens backed by real-world value, provide fractional ownership solutions and create opportunities to decrease processing times and remove middlemen.

Some of the primary benefits of Blockchain are that it leverages a decentralized infrastructure, is a completely trustless environment, and provides immutability by cryptographically linking all blocks together. All blocks on the Blockchain are indexed using a Merkle Tree. A Merkle Tree is a lightweight digital fingerprint of all the transactions within a block.

Peer-to-Peer data sharing, hosting hardware owned by many not a few, extremely high fault tolerant, and security are the key tenets of a decentralized system.

Knowing the difference between decentralized, distributed and centralized systems is critical to understanding Blockchain.

Drawbacks of Blockchain

As with any technology there are a number of drawbacks to Blockchain. Starting with how new Blockchain is, to the stigma of its use originating in the Dark Web. The creation of Blockchain is also a mystery that tends to put people on edge. ICO/ITO scams and the misperception that Blockchain is just another name for cryptocurrency are also drawbacks that aren't technical but do impact the adoption of Blockchain itself.

More tangible challenges with Blockchain today include the fact that Blockchain technology is still changing and evolving, best practices and recommended patterns for implementation are still being formed. There are not very many trained resources and therefore, the cost for trained resources is high.

Finally, scalability is a core concern when it comes to Blockchain. Blockchain prioritizes security over speed. Therefore, solutions that require high transaction speeds are not good candidates for Blockchain. Different group consensus methods beyond Proof of Work are currently being proposed to overcome current scalability limitations. Today, most major public Blockchains are able to process 10-20 transactions per second worldwide.

Cryptography

Cryptography is used in Blockchain to address the issues and concerns of privacy. Cryptography is the study of how to send information back and forth securely in the presence of adversaries. A cryptographic function is a function for encoding or encrypting data to protect the contents from others. The following components are the basis of a cryptographic function:

- The Secret – The data which we are trying to protect
- The Key – A piece of data used for encrypting and decrypting the secret
- The Function – The process or function used to encrypt the secret
- The Cipher – The encrypted secret data, output of the function
- The Secret and the Key are passed into the Function to create a Cipher

Public Key Cryptography

In common Blockchain design, identity and transaction approval is made possible through Public key Cryptography. In Public Key Cryptography there are two keys, the Public Key and the Private Key. The Public Key is used to verify the digital signature of a given key pair. The Private Key is used to sign/approve any transaction/action that might be made by the holder of the key pair. All transactions submitted to the Blockchain are signed using the user's private key and are verified on the Blockchain using the public key.

Cryptographic Hashing

A cryptographic hash function is a one-way function that encrypts information that CANNOT be decrypted. Any data that is passed into a cryptographic hash function will create a unique, fixed-length, hashed output. Therefore, the only way to validate data that has been hashed is to enter the exact same (identical) inputs into the same cryptographic hash function. If there's any difference between what should be and what is, it's easy to identify as the hashed outputs will be completely different.

Blockchain Consensus

All announcements/transactions are recorded in blocks in Blockchain. When a block is filled up, it is validated through group consensus before it can be added to the chain of previously validated blocks. There are a number of Blockchain consensus mechanisms but regardless of the consensus type used, it is important to note that all transaction data on a chained block is assumed to be trustworthy and the chained data has not been tampered with due to the validation of data by group consensus.

Proof of Work (PoW) Consensus

Bitcoin implemented Byzantine Fault Tolerance through a validation system called Proof of Work. In Proof of Work consensus, when a block is full each node competes to solve a guessing game problem to validate the block of data. This problem is non-computational and random guesses are most efficient. Nodes are called Miners and they have to guess the "nonce" to succeed in validating a block. All block data plus the current guess (nonce) are run through a cryptographic hash, if the result matches the current level of "difficulty", the miner has guessed the right answer. Difficulty is adjusted by the network to correspond to load.

A nonce is the random data that is combined with the block data which will produce a hash output matching the current difficulty level of the Blockchain. Any miner who thinks they have the correct answer will share it with all other miners. Miners will confirm the answer is correct by using the nonce with their block data to try to get a result that matches the difficulty setting. If 51% or more of the miners agree with the proposed nonce,

the transactions on the winner's block are considered to be correct and the miner with the correct answer will be rewarded (usually the reward is given in platform tokens). If the majority of miners do not agree with the nonce, no reward is given and the work performed is a sunk cost as no validation equals no reward. Any nodes that do not have the correct block data will reconcile by copying the validated block from neighboring nodes. Proof of Work consensus creates a game theory incentive for each node to behave accurately and honestly; any dishonest participants will incur real-world costs in guessing the nonce for a zero percent chance of being rewarded with a payout.

In Proof of Work consensus, it is important to note that adding a node to the network increases security by 1/N (N = number of nodes on network) AND it increases transaction time by 1/N
ALL nodes must validate hence, the prioritization of security over speed in PoW validation.

Proof of Stake (PoS) Consensus

Proof of Stake is a newer Blockchain consensus system that has been proposed as an alternative to Proof of Work consensus to overcome the scalability and cost concerns in PoW. Proof of Stake removes the guessing game from the validation of blocks so mining no longer requires powerful and specialized hardware, therefore, it requires less energy for processing.

Proof of Stake consensus uses a system where "Validator" nodes each give or pay a stake in order to validate transactions. When it's time for group consensus, all who wish to participate lock up funds in a stake. A random node is selected and the hash of that node's block data is shown to all other participants. All other nodes wager on the validity of the block transactions. If the majority agree with the proposed block, the random node is rewarded as are all who wagered on that node. If the majority disagree, the random node loses their stake, gets no reward, and a new node is randomly selected to share their block data. The game theory incentive toward honesty and accuracy is maintained, only the mechanics of how it's enforced are changed.

The key difference with this consensus is that no computing is ever performed during consensus, only wagering and any kind of device can wager, regardless of computing power. One potential vulnerability of the Proof of Stake is the Nothing at Stake problem, where a validator node approves all transactions on both sides of a ledger after a hard fork has occurred.

Proof of Work vs Proof of Stake

PoW	PoS
Work for a reward	Make a safe bet (stake) for a reward
Slow transaction speed	Fast transaction speed
High energy consumption	Low energy consumption
Proven	New
Capital spent on hardware	Capital spent on staking funds

Other Consensus Mechanisms

Proof of Activity – is a hybrid of PoW and PoS. Empty template blocks are mined (PoW) then filled with transactions which are validated via PoS.

Proof of Burn – is where coins are "burned" by sending them to an address where they cannot be retrieved. The more coins burned, the better the chances of being selected to mine the next block.

Proof of Capacity – is where hard drive space is staked to participate. The most space 'staked', the better the odds of being selected to mine the next block. The consensus algorithm here generates large data sets called 'plots' which consume storage.

Proof of Elapsed Time – was created by Intel to run on their trusted execution environment. It is similar to PoW but far more energy efficient. The concern is this requires trust in Intel and can be viewed as a central authority.

Proof of Authority - uses a set of "authorities" which are nodes that are explicitly allowed to create new blocks and secure the Blockchain. This is a replacement for PoW but only for Private Blockchains. Nodes have to earn the right to become a validator/authority.

The Lifecycle of a Public Blockchain Transaction

1. User uses Dapp/web3 to start transaction
2. User signs the transaction with their private key
3. Transaction validated on locally running node
4. Transaction broadcast to entire network
5. Miners choose to accept or pass on the transaction
6. Miner takes each accepted transaction and writes it to the current block

Chapter 2 Study Points

Below is the list of topics of importance for this chapter, additional research should be conducted prior to taking the CBSA exam:

- Benefits of Blockchain
- What is Cryptography
- Public Key Cryptography
- Cryptographic Hashing
- What is Blockchain Consensus
- Proof of Stake Consensus
- Entropy (randomness) is used to generate account IDs and prevent duplication
- Blockchains can store any type of data, not just financial transactions
- Pending transactions on the Ethereum Blockchain are always ordered by the highest fee paid to lowest, and then written to the block in that order.
- Drawbacks for Blockchain
- Key components of Cryptography
- Differences between Public & Private Keys
- Proof of Work Consensus
- PoW vs. PoS
- The concept of "trustless" environments, applications, and architectures
- Cryptographic hashing used to protect anonymity
- The Nothing at Stake problem

Chapter 2 Key Terms

Below is the list of key terms for this chapter:

- Decentralization
- Public Key Cryptography
- Cryptographic Function
- Cryptographic Hashing
- Proof of Activity
- Proof of Elapsed Time
- Entropy / Randomness
- Cryptographic Hashing
- Trustless environment
- Secret & Key
- Private Key
- Proof of Work
- Proof of Burn
- Proof of Authority
- Hard forks
- Merkle Tree
- Cipher
- Public Key
- Proof of Stake
- Proof of Capacity
- Trustless
- Anonymity

Chapter 2 Quiz

1. What is not a benefit of Blockchain?

a. Trust
b. Security
c. Decentralization
d. Immutability
e. The Urkel Tree

2. In Proof of Work consensus what happens when you add another node to the network?

a. Security time is increased by 1/N and transaction time is increased by 1/N where N equals the number of nodes on network
b. Security time is increased by 1/N and transaction time is decreased by 1/N where N equals the number of nodes on network
c. Security time is decreased by 1/N and transaction time is increased by 1/N where N equals the number of nodes on network
d. Security time is decreased by 1/N and transaction time is decreased by 1/N where N equals the number of nodes on network
e. None of the above

3. The Private Key does what?

a. Used to verify the digital signature of a given key pair
b. Used to create a one-way encoded hash of a block of data
c. Used to sign any transaction that might be made by the holder of the key pair
d. Used to confirm the identity of an application

4. Cryptographic hashing is a one-way function that encrypts information that can be decrypted.

a. True
b. False

5. Proof of Stake consensus costs less, is faster and more secure than Proof of Stake because of the use of a “nonce”.

a. True
b. False

6. A 'hard fork' occurs when:

a. New Blockchain software is released which breaks or modifies existing rules.
b. Some nodes decide to keep a different version of the ledger then others
c. Node hardware is updated to newer models
d. A & B
e. None of the above

7. Proof of Stake consensus aims to do all of the following except:
   a. Improving transaction capacity
   b. Lower energy consumption
   c. Removing the need for specialized hardware
   d. Making Smart Contracts easier to develop

8. On a public Blockchain such as Ethereum, transactions are validated before they're added to the block.
   a. TRUE
   b. FALSE

9. The Merkle Tree serves which important function in Blockchain?
   a. It acts as an index, allowing transactions to be found quickly regardless of their location on the Blockchain
   b. It reduces the overall size of the Blockchain
   c. It increases speed in Proof of Work consensus
   d. All of the Above
   e. None of the Above

10. All of the following are key terms in Cryptography except:
   a. The Secret
   b. The Function
   c. The Root Hash
   d. The Cypher
   e. The Key

11. Public Blockchains like Ethereum can only store financial information.
   a. TRUE
   b. FALSE

Chapter 2 Solution for the Quiz:

Q1: E
Q2: A
Q3: C
Q4: B
Q5: B
Q6: D
Q7: D
Q8: B
Q9: A
Q10: C
Q11: B

Chapter 3: Types of Blockchain

In this chapter we cover what creates the chain in Blockchain, who the participants are in a Blockchain solution, the different types of Blockchains available and how to classify them. Smart Contracts, Blockchain based tokens/coins and how gas works in Ethereum are also covered.

Before diving into the various types of Blockchain, it is imperative that one understands the mechanics of the chaining or linking of the blocks of data together in Blockchain. The purpose of chaining blocks together is immutability which essentially establishes permanence of our chained data and ensures that it cannot be changed without everyone noticing.

Chaining blocks together starts with a new block's header. This header contains information about the block itself including platform version, timestamp, difficulty level, nonce as well as the hash output of the previous block data it is being linked too. Note that all data from the current block, including its header is hashed and the hash output for the current block will be stored in the header of the next block.

Through this chaining process all blocks are linked together and hence, changing the data in any block anywhere in the Blockchain will result in the hash of that block not matching the hash value stored in the header of the next block that references it. Anyone wishing to change or alter a record stored on a Blockchain will be forced to revalidate every subsequent block on a majority of nodes on the network. On all but the very smallest networks doing so would require a large and very impractical amount of computing power relative to the potential payoff or gain an attacker could realize.

Another key point with Blockchain is that it is Append-only, as opposed to conventional data stores which also provide update and delete functionality. Data on the Blockchain cannot be deleted or edited, only additions can be made! This means all data written to a Blockchain is permanent - an important consideration to keep in mind when evaluating a potential Blockchain solution. This append-only mechanism provides a detailed history of ALL events, not just a snapshot of the current state and this is what makes Blockchain such an interesting technology.

Many people don't realize that there are actually many different types of Blockchain technology available and selecting which one to use is an important decision that can have long term effects.

Public vs Private Blockchains

When it comes to selecting the right Blockchain technology, one of the first questions or decisions to determine is who can write data to the Blockchain itself? A simple question but one that will dictate which Blockchain options one will choose from. There are only two options to select from:

- Public Blockchain – where everyone can add a record
- Private Blockchain – where only certain participants can write data

In general, private Blockchains are preferable in scenarios where companies and private organizations need to build secure and permissioned applications where a user's identity is known and performance is important. Private Blockchain networks are much smaller than their public counterparts, planning for and providing proper uptime of all components is a critical consideration.

Public Blockchains can contain any kind of data, although they are primarily used today to record multi-

party exchanges of financial value. A block on a public Blockchain can be filled with a variety of non-related transactions, financial or otherwise.

Open vs Closed Blockchains

The second question or decision for selecting the right Blockchain technology is who can read data from the Blockchain? There are only two options to select from as well:

- Open Blockchain – everyone can read Blockchain data
- Closed Blockchain – only certain participants can read data

It is important to note that when discussion public vs private and open vs closed we are discussing variable points on a solution matrix, not absolute or binary values. A great way to determine what type of Blockchain one needs is to determine if all participants are considered equal or should some have abilities or permissions that others do not? Answering this will help guide the solution to use a permissioned or permission-less Blockchain technology. An example of a permissioned Blockchain is when an election chairperson can add candidates to an election. Digital currency, which can be exchanged and traded by all, is an example of a permission-less Blockchain.

Open Source

Most public Blockchains are open sourced projects, and any developer or organization can use this open source code base to create their own private instances. These instances can duplicate existing functionality from their public cousins, or may contain additional logic and changes implemented after the source code was forked and modified by private developers.

Smart Contracts

Smart Contracts, also known as chaincode, are a way to program rules and decision points into transactions and processes on a Blockchain. For those from a development background, a Smart Contract can be thought of a class in traditional programming terms. Smart Contracts are published to the Blockchain directly and allow one to automate transactions and ensure they all follow the same rules. Each Smart Contract, along with the transactions it performs, exist as records or transactions on the Blockchain. Therefore, Smart Contracts also live as permanent entities on the Blockchain – this is an important point to keep in mind when evaluating a Smart Contract as a potential solution component.

Smart Contracts provide:

- Autonomy: Smart Contracts can be developed by anyone, no need for intermediaries such as lawyers, brokers, or auditors
- Backup: A Blockchain and Smart Contracts deployed to it can provide a permanent record, allowing for auditing, insight, and traceability even if the creator is no longer in business
- Efficiency: Removing process intermediaries often results in significant process efficiency gains
- Accuracy: Replacing human intermediaries with executable code ensures the process will always be performed the same
- Cost Savings: Replacing intermediaries often provides significant cost reduction

Tokens/Coins

Some platforms offer built-in token architectures that be used to create one's own token or coin on Blockchain. Monetary value can be represented in a Blockchain system by these coins or tokens. Tokens can be backed by real-life assets, but this is not required. Issuing tokens or coins is available for both public and private Blockchains.

By issuing one's own token/coin, fractional ownership of assets is a creative use or option. Some key benefits to issuing a token/coin on Blockchain include:

- Ownership of tokens is tracked on the Blockchain
- Trading and exchanging of tokens is managed on the Blockchain

There are two types of tokens within a Blockchain solution:

- Equity tokens – which represent ownership in some real-world asset
- Utility tokens – which represent usage credits on the solution platform itself

Specific to Ethereum Blockchains is the Ethereum Token Standards which are the most widely accepted and used Blockchain standards to date. The Ethereum Token Standards are:

- ERC20 – the most common standard. Currently the only accepted ERC standard.
- ERC223 – proposed new token standard, will be backward compatible with ERC20. Improves smart contract transfers and will be safer, quicker and cheaper.
- ERC721 – non-fungible tokens, can be used for asset tokenization. Can be non-tradable, track certifications, accomplishments, professional milestones

Gas in Ethereum

Specific to the Ethereum Blockchain is the concept of gas. This concept was born out of a limitation the developers of Ethereum saw with Bitcoin, specifically its programming language called Bitcoin Script. One of the major limitations of Bitcoin Script is the inability to perform loops or iterations in the language. This severely limits the types of functions developers can create in Bitcoin Script. Ethereum developers introduced the concept of gas to allow for functionality lacking in Bitcoin Script in order to provide developers Turing complete Smart Contract development languages such as Solidity and Viper.

Gas is simply how users pay for the cost of a transaction to be processed or validated on the Ethereum Blockchain. Gas is a separate reward given to all miners independently of the consensus mining reward. Gas is used to compensate all nodes on the network for the cost incurred in recording a single transaction. Every transaction (write) on the Ethereum Blockchain must be submitted with gas, any unused gas is returned to the user. Note that reading data from the Blockchain is not considered a

transaction, and therefore does not incur a gas cost. The concept of gas not only pays for the cost of recording a transaction on their copy of the ledger, but it also prevents infinite loops and closes certain security vulnerabilities.

It is important to note that gas is only consumed when data is written to the Blockchain. Reading from the Blockchain consumes no gas. A function (in a Smart Contract), which runs out of gas, will be terminated and no gas will be returned to the user. It is also important to note that the management of gas is handled at the protocol level – the protocol itself will remove Ether from a user's wallet, convert it to the requested amount of gas, and return any unspent gas to the wallet after converting it back into Ether. The user does not need to intervene or even be aware that this occurring. This also means developers only need to consider the gas costs of the transactions in their functions; they do not need to worry about managing the conversion of currency into gas and gas back into currency.

Finally, calculating the amount of gas needed to process a transaction is possible but a general rule of thumb is, the more write operations in a Smart Contract there are, the more gas required. Gas is usually decoupled from tokens or coins so that real gas price costs remain constant while token or coin prices are volatile. If NOT using an Ethereum Blockchain, it is important to determine how one will implement their own gas/fee/incentive system, or otherwise compensate nodes for the act of recording new individual transactions.

Gas is also tied to the type of operation being performed – more complex operations will require more gas than simple ones. Online calculators, such as Eth Gas Station, are available to help developers and architects estimate gas costs based on the operations being performed.

Blockless Solutions

Some platforms like IoTA are known as 'blockless' solutions. These platforms are unique in that each individual transaction is validated by a small number of peer nodes instead of being grouped together in blocks and validated by the entire network. Blockless solutions can offer much greater performance and transaction capacity that traditional Blockchain solutions, but with a diminished level of security, compared to traditional blocked solutions.

Chapter 3 Study Points

Below is the list of topics of importance for this chapter, additional research should be conducted prior to taking the CBSA exam:

- Types of Blockchains
- Public vs. Private
- Open vs. Closes
- What is a Smart Contract?
- What is Chaincode?
- How does Gas work?
- Distributed Ledger Technology (DLT)
- Type of Blockchain Tokens/Coins
- Differences between token types
- What is ERC?
- 3 main ERC token standards
- What is Gas in Blockchain?
- Blockless Blockchains?

Chapter 3 Key Terms

Below is the list of key terms for this chapter:

- Open / Closed
- Public / Private
- Blockless
- IoTA
- ERC20
- ERC721
- Gas
- Smart Contracts
- ERC223
- Equity Token
- Utility Token
- Chaincode

Chapter 3 Quiz

1. A public closed Blockchain allows which of the following?

a. Many people can write, only a few can read
b. Many people can read and write data
c. Only a few people can write data, many can read
d. All of the above
e. None of the above

2. Blockless platforms offer which advantage?

a. Lower cost
b. Greater transaction processing capacity
c. Guaranteed smaller network size
d. Greater storage space on the ledger

3. Each block in a Blockchain is linked to what?

a. The block that occurred after it
b. The preceeding block
c. The first block
d. The current block

4. Users must explicitly purchase gas before using a Blockchain solution

a. TRUE
b. FALSE

5. The ERC20 standard defines

a. A consensus mechanism
b. How hard forks are managed
c. A coin / token standard
d. None of the above

6. Which of the following is not one of the three main Ethereum token standards?

a. ERC20
b. ERC223
c. ERC508
d. ERC721

7. A private / open Blockchain would be a good choice for situations where only a few people should be able to write data, but a large number of people should be able to consume that data.

a. TRUE
b. FALSE

8. Which of the following is an example for a 'blockless' platform?
   a. IoTA
   b. Swarm
   c. Ripple
   d. Litecoin

9. If a function call runs out of gas, the gas submitted by the user is returned and the function rolls-back
   a. TRUE
   b. FALSE

10. Which of the following token standards is used for non-fungible, non-transferrable assets on the Ethereum Blockchain?
   a. ERC20
   b. ERC721
   c. ERC508
   d. ERC1014

11. An open Blockchain architecture should be used in cases where public verification is important.
   a. TRUE
   b. FALSE

Chapter 3 Solution for the Quiz:

Q1: A	Q5: C	Q9: B
Q2: B	Q6: C	Q10: B
Q3: B	Q7: A	Q11:A
Q4: B	Q8: A	

Chapter 4: How is Blockchain Different Than What We Have Today?

In this chapter we cover how Blockchain is different from the other technologies in use today. This chapter goes into detail on the network, application and data comparisons of a Blockchain versus non-Blockchain platform.

Type of Networks

There are three primary network architectures when building a technology solution: centralized, distributed and decentralized. In a centralized system, both the data and the solution components are owned by a single entity, resources are concentrated in a single location or system and are easy to maintain. Centralized systems are a single point of failure, provide low fault tolerance and low or fixed scalability.

In distributed systems, the data is owned by a single entity, but resources are distributed across multiple locations, data centers, and systems are not owned by the solution provider – this is akin to modern cloud computing and "as a service" solutions. Thus, distributed systems provide higher fault tolerance and more scalability. Most centralized systems evolve over time to distributed systems.

Decentralized systems have no single owner of either data or network hardware resources as ownership and upkeep is shared amongst all participants. Decentralized systems provide extremely high fault tolerance and are infinitely scalable as resources are independent entities. Decentralized systems are the most difficult to maintain as no one entity owns it, it is shared amongst the peers or participants of the system.

Decentralized systems, including Blockchain, often run on a Peer-to-Peer (P2P) network architecture. P2P is a network of equally privileged participants where tasks and workloads are partitioned across all participants. Participants are called nodes and every node is a consumer and a supplier of resources. This is opposite of a centralized or distributed system where there are defined servers (suppliers) and clients (consumers). In a true P2P system all nodes are both clients and servers simultaneously. Requests for processing in a decentralized system are served by the peers around you, not by a central resource.

Blockchain is a decentralized (P2P) system, it is not a distributed or a centralized one. Other P2P systems include:

- Mesh Networking where network connectivity and network resources is shared amongst participating nodes/systems
- Multimedia content distribution
- Bittorrent file and data trading platforms

- Content based addressing (IPFS) which is known as *Hypermedia Distributed File System*

Blockchain systems, to varying degrees, always prioritize security over speed. This means that scaling up, or adding more nodes increases security but not performance, as every node must validate all transactions on the block. Adding more nodes makes things safer, but not faster. This is an important distinction to make as the opposite is usually true on distributed systems.

Software vs Firmware

With the advent of the Internet era, software developers have become accustomed to being able to patch and upgrade their applications whenever they want or need to. Releases can occur on a regular or irregular basis depending on when and what functionality is being deployed. Agile development methodologies can be adopted both pre and post release.

Firmware developers do not usually have this luxury as firmware is software that is embedded into the hardware itself. To update or patch firmware based software requires special hardware and the physical machine or access itself is typically required. For example, if your oven needs a firmware upgrade, it's not practical to drag it to the local appliance store to be re-flashed.

Due to their permanent nature on the Blockchain, Smart Contracts are like firmware, not software. Once a Smart Contract is deployed to the Blockchain it is permanent and can never be changed, just like any other transaction stored on a Blockchain. New versions of the contract can be released to sit alongside old ones, but old ones can never be deleted.

Although the published Smart Contract cannot be changed the code can be updated and a new Smart Contract can be created and published but the old one will remain forever on the Blockchain.

If the Smart Contract developer desires or requires, they can add in a special function to any Smart Contract called a kill or self-destruct. This special function enables a Smart Contract to be "killed". A killed Smart Contract will refuse any new transactions, although it still continues to exist. This function does NOT delete the contract. Smart Contract developers should think through the choice to implement this function carefully and great caution should be taken to ensure it can only be called by the right people or circumstances. Several Smart Contracts have been implemented by developers and designers who did not properly think through the implementation of their kill function, allowing malicious attackers to kill their contracts, sometimes while the contract was holding millions of dollars of other people's money. DO NOT take the decision to implement a kill or self-destruct function lightly!

Blockchain vs Database

There are many factors to consider when deciding on what technology to implement and whether Blockchain is the right fit or not. A key component of any solution is where the data for your solution will reside. Traditional systems today leverage database technology that has matured over decades. When examining Blockchain as an option, a traditional database system will be a better fit over Blockchain when:

- Performance is important
- Large number of trained resources is important
- Highly confidential data

- Don't need a history, just a snapshot of data
- Ease of maintenance
- Application logic expected to change frequently
- Maintaining centralized control of resources is important
- Would a distributed database or other technology be adequate?

Blockchain will be a better fit over a traditional database system when:

- Requires transparency and public validation
- Needs extreme fault tolerance
- Needs to be infinitely scalable
- Full data history and lifecycle is important to capture
- No single authority can or should be entrusted with the data

Data Sovereignty is another factor to consider when comparing Blockchain solutions versus traditional ones. In a centralized system, all data is owned by the system owner. In scenarios where one must demonstrate they own and control the data as well as demonstrate where it is and is not stored, Blockchain may not be a good solution (although private Blockchains can still be a viable option here).

Chapter 4 Study Points

Below is the list of topics of importance for this chapter, additional research should be conducted prior to taking the CBSA exam:

- What are the different types of networks?
- What is a distributed system?
- What is P2P?
- What is Hypermedia Distributed File System?
- What is firmware development?
- What is the Kill function?
- Differences between Blockchain vs. Databases?
- What is a centralized system?
- What is a decentralized system?
- How does P2P work?
- What is IPFS?
- Differences between software and firmware development.
- How does the Kill function work?
- What is Data Sovereignty?

Chapter 4 Key Terms

Below is the list of key terms for this chapter:

- Centralized System
- Peer-to-Peer
- Hypermedia Distributed File System
- Databases
- Distributed System
- Decentralized System
- P2P
- IPFS
- Firmware
- Software
- Smart Contract Kill function
- Data Sovereignty

Chapter 4 Quiz

1. Blockchain Smart Contracts are analogous to:

a. Software
b. Firmware
c. Natural Language contracts
d. C++ solutions

2. Databases provide greater fault-tolerance than public Blockchain networks.

a. TRUE
b. FALSE

3. Public Blockchains are ideal solutions when data sovereignty is a concern.

a. TRUE
b. FALSE

4. Which of the following is true of Smart Contracts?

a. They must be less than 100kB in size
b. They are written in English
c. They are translated to hex format when compiled
d. They exist as permanent records on the Blockchain once deployed

5. Just like conventional networks, Peer-to-Peer networks contain both clients and servers.

a. TRUE
b. FALSE

6. Blockchain will be a better fit over a traditional database system when:

a. Public validation is required
b. Infinite scalability is needed
c. No single authority can or should own the data
d. All of the above
e. None of the above

7. When a Smart Contract is killed it…

a. No longer accepts new transactions but remains on the Blockchain forever
b. Is removed from the Blockchain completely
c. Is removed from the Blockchain and replaced with a “bookmark” or pointer
d. Is moved to the test Blockchain network for historical purposes

8. IPFS stands for:

a. Internet Protocol File System
b. Inter Planetary File System
c. Inner Planetary File System
d. Internet Public File System

9. Hypermedia Distributed File System is …

a. A mesh network
b. A multimedia content distribution system
c. A content-based addressing system
d. A content delivery network

10. What is NOT a primary network architecture?

a. Decentralized
b. Meshed Filenet
c. Distributed
d. Centralized

11. Blockchains support CRUD operations, just a like a database.

a. TRUE
b. FALSE

Chapter 4 Solution for the Quiz:

Q1: B
Q2: B
Q3: B
Q4: D
Q5: B
Q6: D
Q7: A
Q8: B
Q9: C
Q10: B
Q11: B

Chapter 5: What Does a Blockchain App Look Like?

In this chapter we cover what the standard Blockchain application architecture looks like, various components and tools for Blockchain application development and Integrated Development Environments or IDE.

The majority of developing Blockchain applications is not much different than developing non-Blockchain applications with the exception of the reading and writing to a Blockchain. Blockchain developed applications are commonly referred to as a Decentralized App, DApp, dApp, dapp or ĐApp.

Blockchain Application Architecture

Developing Blockchain applications resembles typical full stack web application development which is usually referred to as a multi-tiered application. Typical Blockchain applications include a user interaction layer, a middle/interface layer and a Blockchain layer.

The user interaction layer includes a presentation layer or layers comprised of HTML/CSS and mobile technologies. No special or new requirements are needed in this layer when developing a Blockchain application. The user experience for a Blockchain application should seek to make users understand and value any new or different features that Blockchain provides.

The middle/interface layer is the gateway to the Blockchain itself and provides all the communication to and from the Blockchain. Server-side code written in Node.js is the technology that provides all of the application logic as well as the contract interfaces for a Blockchain application. This code brokers communication between the user and the Blockchain itself.

All communications to and from Blockchain is in bytecode. The middle layer abstracts the translation of bytecode away from the user and the middle layer by using JavaScript libraries. This layer should contain all application validation and exception handling code and some developers will also implement business logic here as well, but some will implement this logic in their Smart Contracts in the Blockchain layer too.

The Blockchain layer is where both Smart Contracts (or Chaincode) and their transactions are published, stored and validated. Smart Contracts are converted to bytecode, published to the Blockchain and validated just like any other Blockchain transaction.

Smart Contracts can call other Smart Contracts. Smart Contracts do not have access to external data by default but this limitation can be overcome through the use of an Oracle. Oracles are trusted data feeds that can send information into a Smart Contract. Oracles are usually supplied by third parties but can also be developed independently.

Some other tools and frameworks to be familiar with include:

- Web3.js frameworks
- Ganache (formerly known as TestRPC) – Allows for testing and development with a local Blockchain simulation
- Truffle – a tool and release Blockchain application management system that allows for deployment and testing on Ethereum test and production Blockchains.

Integrated Development Environment (IDE)

Since the majority of a Blockchain application is the same as other applications, many of the mainstream Integrated Development Environments, or IDE for short, can be used when building a Blockchain application. Any HTML, CSS and JavaScript editor or IDE will work for the initial layers of development for a Blockchain application including most Open Source code editors such as Visual Studio Code as well as some professional/paid editors. Many of these IDEs have a plug-in architecture that enables the IDE to be extended or updated for developing Blockchain code. Most IDEs have plugins already available for both Ethereum and Hyperledger.

Given the decentralized nature and that everything in Blockchain occurs at the protocol level, Blockchain application and Smart Contract development does require some special tooling. The Smart Contract tools needed will be derived from the type of Blockchain technology used. If using an Ethereum Blockchain then you will be developing and interfacing with the Blockchain via the Solidity (SOL) language.

One of the best ways to get started in developing Ethereum Smart Contracts is using Remix (http://remix.ethereum.org/). Remix is an online integrated development environment for Smart Contract development for Ethereum Blockchains. Remix focuses on the development and deployment of Solidity Smart Contracts. Remix is a good solution if you intend to:

- Develop Smart Contracts (remix integrates a solidity editor).
- Debug a Smart Contract's execution.
- Access the state and properties of an already deployed Smart Contract.
- Debug already committed transaction.
- Analyze solidity code to reduce coding mistakes and to enforce best practices

If using a Hyperledger Blockchain then you will be most likely developing and interfacing with the Blockchain via Hyperledger Composer. Hyperledger Composer allows you to model your business network and integrate existing systems and data with your Blockchain applications.
All functions in Hyperledger Fabric are known as either "Intake" or "Deploy" transactions. Deploy transactions deploy a new asset to the Blockchain, while intake functions invoke functionality provided by those assets. Client apps communicate with chaincode via the use of an SDK.

One interesting architecture pattern in Blockchain solutions is called a DAO, or Decentralized Autonomous Organization. A DAO is a solution comprised of a number of Smart Contracts which aim to replace the functions provided by humans in many organizations. The DAO architecture pattern allows companies and enterprises to form, they are leaner and more geographically-distributed than their traditional counterparts.

Chapter 5 Study Points

Below is the list of topics of importance for this chapter, additional research should be conducted prior to taking the CBSA exam:

- What is a Decentralized App?
- What is a typical Blockchain application architecture?
- What is user interaction layer?
- What is data layer?
- What is a Blockchain layer?
- What is Remix?
- What is Hyperledger Composer?
- What is a DAO?
- How does an Oracle work?
- What are the various acronyms for a Decentralized App?
- What are the application layers of a Blockchain application?
- What is user middle/interface layer?
- What is an IDE?
- What is Solidity?
- When is remix a good option for us?
- What are "Intake" and "Deploy" transactions?
- What is an Oracle?

Chapter 5 Key Terms

Below is the list of key terms for this chapter:

- Decentralized Application
- Middle/Interface Layer
- Blockchain Application Layers
- Intake transaction
- Web3.js
- TestRPC
- DAO
- DApp
- Data Layer
- Oracles
- Deploy transaction
- Truffle
- Remix
- User Interface (UI) Layer
- Blockchain Layer
- Hyperledger Composer
- Solidity
- Ganache
- IDE

Chapter 5 Quiz

1. DApp stands for

a. Decentralized Application
b. Directed Acyclic Parity Principle
c. Direct Autonomous Purchasing Power
d. Downstream Application of Powerful Provisions

2. Remix is a browser-based IDE for editing Smart Contracts on which platform?

a. Bitcoin
b. Monero
c. Hyperledger
d. Ethereum

3. By default, Smart Contracts cannot access data outside the Blockchain.

a. TRUE
b. FALSE

4. The middle layer in a Blockchain application contains the user interface.

a. TRUE
b. FALSE

5. Developing a user interface for a public Blockchain application requires developers to learn new skills.

a. TRUE
b. FALSE

6. What are the two types of transactions in the Hyperledger Fabric?

a. Deploy and Intake
b. Deploy and Invoke
c. Publish and Call
d. Copy and Evoke

7. To develop in Hyperledger, you use the following tool:

a. Solidity
b. Remix
c. Composer
d. Eclipse

8. DAO stands for:

a. Decoupled Application Overture
b. Decentralized Autonomous Organization
c. Distributed Autonomous Organization
d. Decentralized Autonomous Offer

9. IDE is short for:

a. Independent Development Environment
b. Integrated Design Environment
c. Independent Design Environment
d. Integrated Development Environment

10. Smart Contracts by default have no access to external data. To overcome this, you can use a/an _____.

a. API
b. Web Service
c. REST Endpoint
d. Oracle

11. Other tools and frameworks to be familiar with when building Blockchain applications include:

a. Ganache
b. Truffle
c. Souffle
d. Web3.js
e. a,c
f. a, b and d
g. All of the above

12. When developing Blockchain applications which of the following is NOT used?

a. HTML
b. JavaScript
c. NodeJS
d. J#

Chapter 5 Solution for the Quiz:

Q1: A
Q2: D
Q3: A
Q4: B
Q5: B
Q6: A
Q7: C
Q8: B
Q9: D
Q10: D
Q11: F
Q12: D

Chapter 6: How Do I Design a Blockchain App?

In this chapter we cover how one would approach the design of a Blockchain application or solution. The designing of a Blockchain application should be approached just like any other enterprise project for an organization starting with the defining of the guiding principles of the solution itself.

Guiding Principles

Guiding principles should be a series of yes/no questions that you, your team, and your organization must agree on before starting solution design. The answers will define the default or assumed view or behavior of the solution. This doesn't mean your solution can't or won't do the opposite, just that if doing to opposite is needed, it requires justification.

Some examples of guiding principles include:

- Will our solution be feature-heavy or feature-light? In other words, do we provide as many features as possible, or do we aim to provide a simple application?
- Which is more important, collaboration or security? In other words, do we favor open and frictionless collaboration, or do we want to ensure the highest levels of security?
- Will support be centralized or decentralized? In other words, if users have a question or need help will they come to us directly, or should they seek out help from a broad community?
- Will our solution for our users have consistency or specialization? In other words, will our solution treat all users the same and grant them the same access and abilities or will this solution allow users to specialize and grow in areas they desire?

Guiding Principles will help to define personas.

Personas

Personas are the types of users who will use your Blockchain application or solution. Each person can have multiple personas based on mood, time of day, expectations, etcetera. A persona is not a person.

Example Persona #1: Rosa is a busy accounting clerk who is frustrated at the amount of paper involved in her current business processes. She feels she could accomplish much more if the processes she's involved in would simply adopt modern toolsets. She often becomes discouraged and frustrated at work and wonders if other organizations are doing a better job than hers. The expense reimbursement process is particularly painful for her.

Example Persona #2: Rosa loves gourmet cooking and on weekends she loves to try preparing new dishes and sharing them with friends. She likes to experiment by deviating from popular recipes to add her own touch. Often, she comes up with new dishes she thinks are quite good. She wishes there was an easier way to share these creations and collaborate on cooking ideas with other amateur chefs.

Personas will help to create user stories.

User Stories

User stories are short stories about a persona and the desired interaction they will have with your Blockchain application or solution. User stories should define how each persona interacts with your solution in a particular use case, and should describe the expected outcome.

Example Persona: Rosa is happy the new Blockchain expense reimbursement system in her company has gone live. Rosa begins her day by checking all the submitted expenses from the last day. She has to approve or deny them all. She can quickly review electronic records and approve or deny with the click of a button. Any denied expenses can be sent back to submitter for changes or can be deleted. All approved expenses are sent to accounting without Rosa having to do anything extra. Rosa feels this is a far better system than the old paper-based system because of the enormous efficiency gains she has experienced since go-live.

User stories will help to create functional requirements.

Functional Requirements

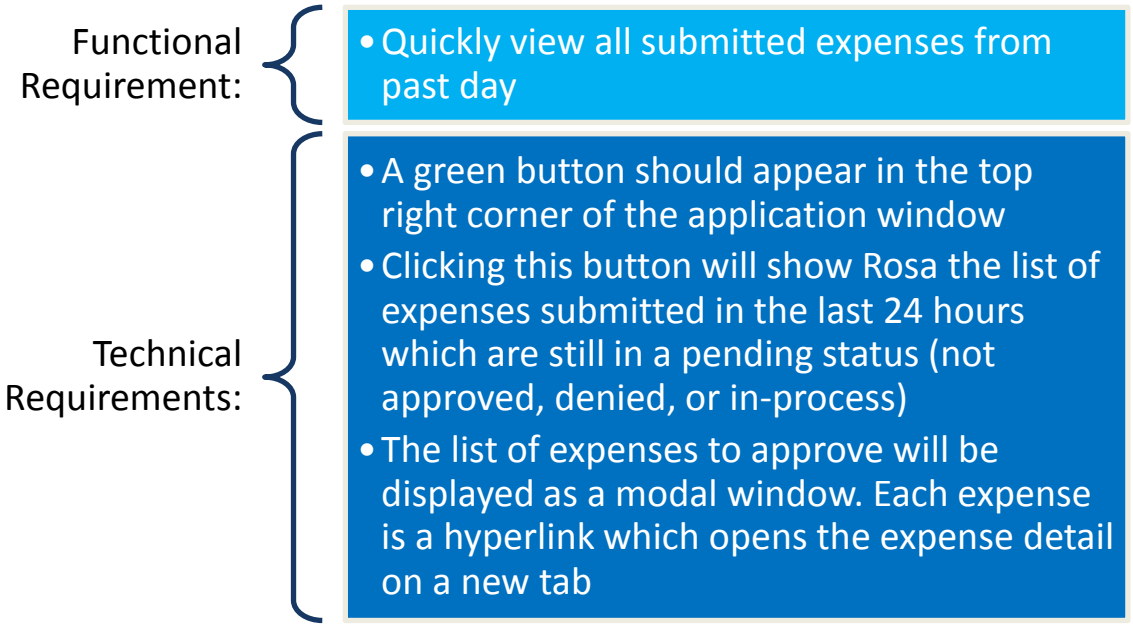
A functional requirement describes what the solution should do without focusing on how that thing should be done. From the previous user story, we can extract these functional requirements:

- Quickly view all submitted expenses from past day
- One click approval or denial of each expense
- A denied expense can be send back to the submitter or can be deleted
- Approved expenses are routed to accounting for pay-out

Functional requirements will drive technical requirements.

Technical Requirements

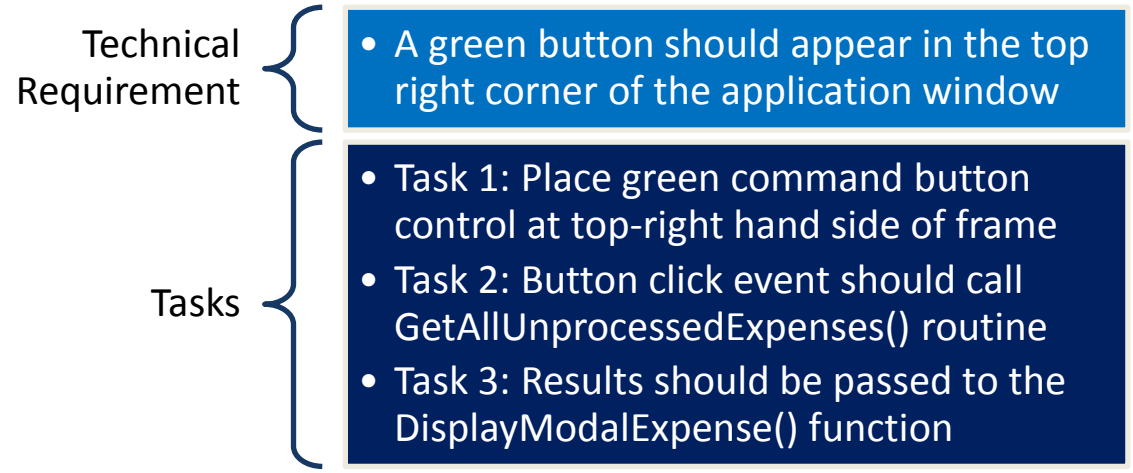
Technical requirements explain how a functional requirement will be fulfilled. In our example the functional requirement can be broken into these technical requirements:



Technical requirements will define development tasks.

Tasks

Tasks are the explicit steps that must be taken to fulfill a technical requirement. In our example technical requirements can be broken into these tasks:



Once you get to the task level, providing estimates for each task is achievable. Each task should also be evaluated to determine the skillset needed to complete it. In general, if a task cannot be estimated or skillset cannot be identified, the task needs to be further broken down into sub-tasks.

In our example tasks can be estimated and matched to a skillset:

Task	Estimated Time to complete	Skillset/Resource
Place green command button control at top-right hand side of frame	1 hour	User Interface Designer
Button click event should call GetAllUnprocessedExpenses() routine	1 hour	Back-end Developer
Results should be passed to the DisplayModalExpense() function	1 hour	Back-end Developer

Blockchain Architecture – Fundamental Questions

When a project or solution is proposed, good architects will work hard to ensure they, their team, their organization and the proposed project itself are positioned for success. To properly position a project for success and to determine if Blockchain is the right technology, ask the following questions:

1. What does this solution need to let users do?
2. Will the proposed solution reduce or remove the problems and pain points currently felt by users?
3. What should this solution prevent users from doing?
4. Do you need a solution ready for heavy use on day 1?
    a. No = Blockchain, yes = traditional
5. Is your solution idea enhanced by the use of Blockchain? Does the use of Blockchain create a better end-user experience? If so, how?
    a. No = traditional, yes = Blockchain
6. Has your business developed custom software solutions before?
    a. YES
        i. Do you have the bandwidth to train developers on Blockchain tech, either directly or indirectly?
        ii. Do you already have a healthy dev ops process & practice?
        iii. Will you outsource development, project management, engagement management, architecture, or UX design?
    b. NO
        i. Do you want to develop in house or outsource?
        ii. Do you have a single solution idea, or do you envision creating multiple solutions?
7. What level of support are you going to need?
8. How big is the developer community?
9. Does your vision of the future align with the project or platform's vision of the future?

10. Does the platform aim to make new and significant contributions to the development space, or is it an efficiency / cost play? Which is more important to you?
11. If private is preferred, how will membership and access be granted, controlled, and regulated?

12. Should the solution be an open or closed Blockchain?
13. Who needs to see the data? Who should NOT see the data?
14. How fast does it need to be?
    a. Greater speed (private Blockchain) = lower trust-ability (smaller network size)
    b. Lower speed (public Blockchain) = higher trust-ability (larger network size)

Will your solution use or require Smart Contracts or their equivalent? If so, create a plan for contract updates and changes. Remember, once a contract is deployed, it's permanent unless its Kill/Self-Destruct method is implemented and called. If using a Kill function, make sure this can't be called maliciously or indirectly. If a contract is updated, a new version is deployed alongside the old contract. Due to the permanence of Blockchain make sure you know how your application will know to use the proper or updated contract.

Finally, Blockchain is not a binary solution or technology! Hybrid solutions, ones that use both conventional data stores and systems as well as Blockchain are possibly the best option.

In the end, always be asking:

- Do my personas align to my guiding principles? If not, is there a justifiable reason why not?
- Do my user stories align to my guiding principles? If not, is there a justifiable reason why not?
- Do my functional requirements align to my guiding principles? If not, is there a justifiable reason why not?
- Do my technical requirements align to my guiding principles? If not, is there a justifiable reason why not?
- Do my tasks align to my guiding principles? If not, is there a justifiable reason why not?

Chapter 6 Study Points

Below is the list of topics of importance for this chapter, additional research should be conducted prior to taking the CBSA exam:

- What are Guiding Principles?
- What are User Stories?
- What are Technical Requirements?
- How do you size Tasks?
- What are some fundamental Blockchain architecture questions?
- What are Personas?
- What are Functional Requirements?
- What are Tasks?
- Where and how do you identify resources?

Chapter 6 Key Terms

Below is the list of key terms for this chapter:

- Guiding Principle
- Functional Requirement
- Work Item
- Resource identification
- Persona
- Technical Requirement
- Project sizing
- User Story
- Task
- Project estimating

Chapter 6 Quiz

1. Guiding Principles can be violated if a sufficient business reason exists.
   a. TRUE
   b. FALSE

2. A good solution can and often does incorporate Blockchain along with more conventional technologies.
   a. TRUE
   b. FALSE

3. Which of the following design artifacts describes what a solution should do without focusing on how it should be done?
   a. Tasks
   b. Technical Requirements
   c. Personas
   d. Functional Requirements

4. Which of the following is true regarding a contract's Kill function?
   a. Security is provided for this method by the platform, no additional concerns exist
   b. A killed contract can be revived
   c. Funds can be extracted from a killed contract
   d. None of the above

5. Guiding Principles will be the same for every type of Blockchain solution.
   a. TRUE
   b. FALSE

6. A user can have multiple personas.
   a. True
   b. False

7. User stories will help to create:
   a. Guiding Principles
   b. Technical Requirements
   c. Functional Requirements
   d. Use Cases

8. Which of the following design artifacts describes how a functional requirement will be fulfilled?
   e. Task
   f. Persona
   g. Guiding Principle
   h. Technical Requirement

9. What are the base columns you should have in your Task estimation worksheet?

a. Task Name
b. Task Owner
c. Estimated Time to Complete
d. Skillset Required/Role

10. What is NOT a Blockchain architecture fundamental question?

a. Does the use of Blockchain create a better end-user experience?
b. Do you need a solution ready for heavy use on day 1?
c. Who needs to see the data? Who should NOT see the data?
d. What color would you like your help icon be for your end users UI?

11. To maximize efficiency, a project team should be as large as possible.

a. TRUE
b. FALSE

12. Public Blockchains offer lower transaction speed than private Blockchains.

a. TRUE
b. FALSE

Chapter 6 Solution for the Quiz:

Q1: A
Q2: A
Q3: D
Q4: D
Q5: B
Q6: A
Q7: C
Q8: D
Q9: B
Q10: D
Q11:B
Q12:A

Chapter 7: How Do I Develop a Blockchain App?

In this chapter we cover some of the fundamental concepts to keep in mind as you move from the design phase into actual construction of a Blockchain solution. As you work to develop your solution, you and your development team should be consistently evaluating these points to make sure you're not following bad patterns or creating a solution with unnecessary security vulnerabilities. Here are some important and highly-recommended things to keep in mind.

Use caution calling external contracts, favor your own
If given the choice between calling an external contract developed in-house and one created by a third party, you should favor the in-house contracts. This is due to the fact that any contracts developed in-house will have been subjected to the same security, review, testing, and acceptance criteria as the rest of your solution – in other words these contracts will meet the same quality and safety standards you've set for the rest of your solution components.

If this is not possible, be sure to set aside extra time for you and your team (including security professionals) to audit and review the source code of any external contracts. You want to make sure that any contracts you call meet the standards you have in place for the rest of your application. Third parties who offer external contract functionality may have never imagined the use cases you're pursuing with your application, so it's critical to ensure you're not inheriting any security flaws created by external parties. Keep the old adage in mind – *a chain is only as strong as its weakest link*.

Handle all errors; catch all exceptions in external calls
This paradigm is important in any kind of software development scenario and certainly not unique to Blockchain. However, in Blockchain applications where you could potentially be dealing with other people's money this becomes even more important. EVERY function in your Smart Contracts should be wrapped in a try/catch statement, and ALL expected exceptions should be dealt with explicitly no matter how small you think the chances of it being thrown actually are. Any exceptions which you do not explicitly catch and manage should be caught and managed generically, and no exception should be allowed to bubble-up to the calling function or contract. If possible, the contract should gracefully deal with exception and attempt to proceed without user intervention. If this is not possible, the error should be reported to the user gracefully along with suggestions to the user on how to resolve the error or otherwise move forward. If you are calling external contracts that were not created in-house, do not ever assume another developer has taken these same precautions and be prepared to deal with any exceptions that bubble back up into your contract as a result. Understand that any unexpected or cryptic errors presented to the end-user will cause them to view the application as 'broken'.

Favor Pull Payments over Push Payments
When constructing a Smart Contract that deals with coins, tokens, or cryptocurrencies you will most likely have one or more functions that are called to pay out to a user or return funds to a user. In such scenarios you should avoid a 'push payment' - try to avoid directly 'pushing' funds into a user's wallet. Rather than taking this approach, setup a 'pull' payment system instead. A pull payment breaks the payment process

up into two distinct steps. The first step informs the user there are funds in the Smart Contract they can withdraw or pull out. In the second step, the user acts on this notification and pulls the funds out to an address they specify. Breaking this process up into two parts allows your contract two different places to verify the correct user is withdrawing the funds, and it allows the user to potentially specify a new wallet address if the original address is no longer valid or has been compromised. This leads to a more secure and robust application.

On-chain data is public! Is it protected?
Remember that all data on a public Blockchain such as Ethereum is viewable to anyone via the use of a Blockchain explorer such as Etherscan.io. All transaction data written to a public Blockchain is stored in hex-encoded plain text or decimal format unless your contract layer or middle layer (or both) perform encryption and decryption services on it. If you are writing sensitive data to a public Blockchain, or do not wish your transaction details to viewable to all, make sure you are encrypting or otherwise protecting your data before writing it to the Blockchain. Alternatively, consider whether a Blockchain is the best place to store data of this type, or if the data is better served by conventional technology such as a relational database.

Always move from local testing, to test networks, then to production
When developing and testing Smart Contracts make sure you perform extensive testing in a local dev environment. On the Ethereum public Blockchain, make sure to use a tool such as Ganache (TestRPC) to test Smart Contracts locally before pushing to a test or production Blockchain. As all records on the Blockchain are permanent (append-only ledger) and Smart Contracts live as transactions on the Blockchain, old contracts never go away. Performing the majority of your testing locally allows you to act as a good steward of the test resources and not unnecessarily to the test networks such as Rinkeby or Ropsten. Only once you're confident that your contracts are ready for production use should you deploy them to a test network for bug bounties and final validation. Remember that once deployed to the production Blockchain costs become real, so make sure you've done plenty of testing first to avoid unnecessary expenses.

Using an Agile approach pre-release is fine, but doesn't work well after releasing to production.

When developing a Blockchain solution, many wonder if Agile methodologies can still be embraced. The answer to this is in the nature of Smart Contracts and the Blockchain. Keep in mind every Smart Contract is a permanent record on the Blockchain and cannot be removed once deployed. Killing or calling a contract's self-destruct function (if implemented) comes with serious considerations, and therefore it is not recommended to continue an Agile cycle of continuous release and development once a contract is pushed to production. In a pre-production release, Agile can still be embraced, but teams should plan for longer testing cycles to ensure as few as possible releases occur after the first push to production.

During design and development, take extra caution to ensure alignment from Guiding Principles down to Tasks.

- Guiding principles drive Personas
    - Ensure Personas align with Guiding Principles
- Personas drive User Stories
    - Ensure User Stories align with Guiding Principles

- User Stories drive Functional Requirements
    - Ensure Functional Requirements align with Guiding Principles
- Functional Requirements drive Technical Requirements
    - Ensure Technical Requirements align with Guiding Principles
- Technical Requirements drive Tasks
    - Ensure Tasks align with Guiding Principles

Make technology decisions last!

Many design and development projects make the common mistake of choosing a technological platform before they've taken the time to discuss and define Guiding Principles, Personas, User Stories, and Requirements. Leading with a technology decision up front is never recommended as it leads you to design a solution that's compatible with the feature set of the technology platform rather than designing a solution which aligns with the needs of the end users. To avoid this trap, make sure decisions about which technologies to use are delayed until design decisions have been made, documented, and are well understood by major stakeholders in the project.

Be careful to only call contracts you've written, or can trust implicitly!

When linking contracts to share functions, take great caution when calling into contracts not developed in-house. All third party contracts should be reviewed to ensure they meet the same standards of quality and security as the contracts produced by your own in-house development teams. You do not want the quality or safety of your solution to suffer because of inherited defects and vulnerabilities.

Monolithic vs Modular

When determining whether to use a series of contracts which call each other (modular), or whether to keep all code and logic in one and only one contract, it is generally advisable to take a modular approach. A monolithic architecture may be the best bet when security is a paramount concern though. Remember to keep the following points in mind:

- Modular is more efficient
- Modular allows for code reuse
- Modular introduces additional security concerns
- Modular is generally preferable, but not always
- Monolithic == potentially greater security

Sandwich complexity model

When designing a three-tier decentralized application, try to concentrate as much business logic and complexity as possible in the middle layer. Think of your application as a sandwich; the UI (top bread layer) and the smart contract layer (bottom bread layer) should remain as simple as possible, and when it makes sense complexity and business logic should live in the middle layer (the meat of the sandwich).

- Keep Smart Contract layer (bottom) as simple as possible
- Keep presentation layer (top) as simple as possible
- Keep complexity in the middle of the application

Chapter 7 Study Points

Below is the list of topics of importance for this chapter, additional research should be conducted prior to taking the CBSA exam:

- Differences between Smart Contracts
- Best practices to Smart Contract Development
- Basic error handling concepts
- Types of software development process and when to use them with Blockchain
- Sandwich Complexity Model
- In-house vs. 3rd Party Smart Contracts
- Agile vs. Waterfall
- Pull vs. Push
- Exception handling concepts
- How to align a project to your Guiding Principles
- Modular vs. Monolithic

Chapter 7 Key Terms

Below is the list of key terms for this chapter:

- 3rd Party Smart Contract
- In-house Smart Contract
- Agile
- Waterfall
- Push Payment/Transaction
- Pull Payment/Transaction
- Error handling
- Exception handling
- Project alignment
- Modular application
- Monolithic application
- Sandwich Complexity

Chapter 7 Quiz

1. According to the Sandwich Complexity Model, the bulk of the application logic should live at which layer?
   a. The validation nodes
   b. The middle layer
   c. The User Interface
   d. The Smart Contract layer
   e. None of the above

2. Unless the technology decisions are made up-front, it is hard to design a good Blockchain solution.
   a. TRUE
   b. FALSE

3. A modular design pattern for Smart Contracts is generally preferable unless which feature is desired?
   a. Security
   b. Scalability
   c. Simplicity
   d. None of the above

4. Push payments are the desired pattern for giving funds to a user.
   a. TRUE
   b. FALSE

5. An Agile development methodology is desirable post-release.
   a. TRUE
   b. FALSE

6. What are some of the best practices for testing a Blockchain application?
   a. Always go from local testing, to a test network and then to production network.
   b. Use a local Blockchain tool like Ganache when doing development and initial testing.
   c. Both a and b
   d. None of the above

7. When developing a Blockchain application always align _____ to your guiding principles.
   a. Users
   b. Trouble tickets
   c. Personas
   d. Infrastructure

8. What is the risk of using a Monolithic architecture for an application or Smart Contract?
   a. It provides reusable code
   b. There is a single attack surface or single point of failure
   c. In introduces additional security attack surfaces
   d. There are no risks to a Monolithic architecture

9. All Smart Contract function calls should be wrapped in a:
   a. Try / Catch statement
   b. A Contract declaration
   c. A For statement
   d. A ForEach Statement

10. If a 3rd party contract has never been hacked, it is safe to trust.
   a. TRUE
   b. FALSE

11. Data on a public Blockchain is automatically encrypted.
   a. TRUE
   b. FALSE

12. Which of the following is an Ethereum test network?
   a. Franklin
   b. Ropey
   c. Ropsten
   d. Zoe

Chapter 7 Solution for the Quiz:

Q1: B	Q5: B	Q9: A
Q2: B	Q6: C	Q10: A
Q3: A	Q7: C	Q11: B
Q4: B	Q8: B	Q12: C

Chapter 8: How Do I Test a Blockchain App?

In this chapter we cover how one would approach testing a Blockchain application or solution. The testing of a Blockchain application should be approached just like the testing of any other enterprise project for an organization. A successful testing philosophy of Blockchain applications is that testing is not an expense, it is a risk-reduction strategy.

Testers should be part of all the phases of your Blockchain project as letting testers be a part of the entire lifecycle of the project (architecture, design and development) will give them a better idea of the project vision and their tests will align to your product or application vision. This is called "Shift-Left Testing".

On the development side of the project, using Test Driven Development (TDD) has been a significant help in Blockchain application development given the restrictions and permanence of Blockchain. At its core, TDD is simply writing your test cases first and then your developers writing their functions that can pass the predefined test cases second. Many projects do testing after development is complete or near complete. Testing should be done during development, test cases should be tested after every software build.

Drawing back to designing and architecting Blockchain applications, personas should be the start of your test cases where you create a set of test cases for each persona. Making good test cases from the beginning makes development go faster!

For testing Blockchain applications, it is recommend that you plan on 5x to 10x traditional application testing time. Remember, you will be testing a lot of firmware not just software-based functionality that cannot be changed once it is on the Blockchain. A target of 100% test coverage of an application is often not practical but is the ideal standard.

It is highly recommended that testing tools are used when it comes to testing Blockchain applications. Note, testing tools will not improve your test cases, they will help you manage them. Automated testing tools allow you test faster but not better.

The different types of testing include:

- Unit Testing
- Developer level testing
- Configuration & Environment Testing – Testing of code in a specific environment (Test, Staging, Production)
- Load & Performance Testing
- Volume/Stress testing
- Regression Testing – as testing of all code deployed, even code not changed. This protects against the reintroduction of old bugs and catch any new bugs introduced from the new code

Given the speed of change within Blockchain, new vulnerabilities are found often. It is strongly recommended to hire a cybersecurity expert for solution review. Offering bounties to developers (internal and external) who find flaws, security holes, and exploits in your code can help provide coverage and speed to your testing.

Blockchain Testing Do's & Don'ts

Do's:

- Test your solution under a pre-defined set of conditions, for example:
  - Can our solution handle 10 concurrent users averaging one transaction every 10 seconds?
  - Can our solution handle 75 concurrent users averaging one transaction every 2 seconds?
  - Can our solution handle 223 concurrent users averaging one transaction every .5 seconds?
  - When a password complexity rule is updated, are existing users with weak passwords notified they should be updated?
  - Test the "Happy Path" – what you expect most users will do most of the time
  - Then wander off the "Happy Path" – your users will
- Separate your development and test environments
- Test your documentation and support materials – users who can't find support or information about your solution will perceive it as 'broken'
- Test a function multiple times from the POV of multiple users
- Test on every device your users will use
- Explicitly state any untested platforms are not supported!

Don'ts:

- Do not let developers test their own code – developers will subconsciously test the code according to the mental model they used to create it
- Do not let developers touch the test environment – developers will forget to document config steps and other requirements. Not letting them touch the test environment will surface these missing pieces.
- Don't test on the latest iPhone, test on ALL iPhones!!

Bug Classifications

It is important to classify your bugs. The different types of bugs incude:

- Business logic – something isn't right according to business requirements
- Security – the code is vulnerable to some security exploits
- Regression – some code updates caused existing features to break
- Performance – the code is slow or some actions execute extra functions
- Accessibility – the code doesn't meet spec for accessibility (Americans with Disabilities Act)
- UI bugs – user interface doesn't meet the design specification
- Integration – two or more components don't work together as expected

When logging bugs, they should be linked back to the persona and business value. Train your testers to log bugs using multi-media - a picture (or video) is worth a thousand (million) words!

Load Testing

When testing load on an application, be sure to take into account both the expected number of concurrent users and the average number of transactions expected per user. Ensure your application can scale beyond these expectations and does not become a victim of its own success.

Key Blockchain Architecture Testing Questions

Some key questions that Blockchain Architects will ask regarding testing include:

- Does the application suit user need(s) today?
- Will the application suit user need(s) tomorrow?
- How much of your application do you plan to test?
- Will you be using any testing tools?
- What about automated testing?
- How will you track and manage your testing?
- Will you let end users participate in testing?

Chapter 8 Study Points

Below is the list of topics of importance for this chapter, additional research should be conducted prior to taking the CBSA exam:

- Testing philosophies
- What is Test Driven Development?
- When should test cases be written?
- Types of testing
- What is load testing?
- Test Automation tools
- What are the Blockchain testing recommendations?
- What is Shift-Left testing?
- What are test cases?
- What drives test cases?
- What is a bug bounty?
- What are the various bug classifications?
- Key Blockchain Architecture Testing Questions

Chapter 8 Key Terms

Below is the list of key terms for this chapter:

- Cost vs. Risk Mitigation
- Test Driven Development
- Unit Testing
- Load Testing
- Regression Testing
- Business logic Bug
- Performance Bug
- Integration Bug
- Shift-Left Testing
- Test Case
- Configuration Testing
- Performance Testing
- Bug Types
- Security Bug
- Accessibility Bug
- Testing Tools
- TDD
- Persona
- Environment Testing
- Volume/Stress testing
- Bug classifications
- Regression Bug
- UI Bug
- Test Automation

Chapter 8 Quiz

1. All of the following are types of bugs except:

a. Business Logic
b. Security
c. Integration
d. Contract Size

2. TDD refers to:

a. Taking Down Data
b. Test Driven Development
c. Testing During Development
d. Testing Direct Duration

3. Multimedia bug reports add little value relative to their cost.

a. TRUE
b. FALSE

4. Traditional testing time should be scaled up how much in the Blockchain world?

a. 2x
b. 5x – 10x
c. 20x
d. 100x

5. Testing tools lead to better test cases.

a. TRUE
b. FALSE

6. Shift-Left Testing is…

a. Where your developers test their own code before deploying it to the next environment.
b. Involving your testers later in the project lifecycle.
c. Involving your testers earlier in the project lifecycle.
d. Where your developers write the test cases and your testers test them.

7. Which of the following is NOT a type of testing?

a. Evolution
b. Unit
c. Configuration
d. Stress
e. Regression

8. _____ should be the start of your test cases.

a. Technical Requirements
b. Actors
c. Personas
d. User stories

9. What is the ideal but not realistic target for test coverage of a Blockchain application?

a. 100%
b. 92%
c. 85%
d. None of the above

10. What is a bug bounty?

a. Hiring an outside expert to review and update your Blockchain architecture and code.
b. Offering a non-fungible token to someone who writes up all of the risks to your code.
c. Posting a notice to a community that hacking is not allowed on your code.
d. Offering a reward to others who find flaws, security holes or exploits in your code.

11. What is NOT a Blockchain Testing best practice?

a. Separate your development and test environments
b. Test a function multiple times from the point of view of multiple users
c. Only create support materials if you are using a public Blockchain
d. Explicitly state any untested platforms that are not supported by your Blockchain application.

12. Blockchain developers should test their own code before publishing it to production?

a. True
b. False

Chapter 8 Solution for the Quiz:

Q1: D
Q2: B
Q3: B
Q4: B
Q5: B
Q6: C
Q7: A
Q8: C
Q9: A
Q10: D
Q11: C
Q12: B