Arman Khondker
EID: aak2464

EE 379K – Solution and Deployment Blueprints Milestone #3

Part 1 – Deriving Deployment Blueprints

Deployment Blueprint #1

Satisfaction of domain function and I/O by solutions

SB Solution Component	BB Functions Satisfied
Server authentication module	• Authenticate and Authorize User • Authenticate Valid Email
Server seller profile maintenance module	• List an item • Edit an item listing • Add description to item • Attach images to listing • Create a shipping label • Remove item listing
Server buyer profile maintenance module	• Buy an item • Check recently viewed items • Add an item to watchlist • View shipping status • Search for items • Filter search for items • Sort Search for items • Submit cancellation request • View all order history • View order details • View size chart • Contact seller • Report listing • Edit preferences • Browse suggestions • Pay for an item • Make a return request
Server payment module	• Issue refund • Process Payment
Server shipping module	• Track shipping Label • Create return label

SB Solution Component	BB I/O Satisfied
Server authentication module	• Username • Password

	• Email address • Street Address
Server seller profile maintenance module	• Pictures of items • Description of items • Cost of items • Condition of items • Seller history • Listing history
Server buyer profile maintenance module	• Viewed items history • Watchlist of items • Purchase history • Search History • Email preferences • Item preferences • Buyer has reported listing • Account alerts updated
Server payment module	• Transaction Cost • Payment Method • Buyer ID • Seller ID
Server shipping module	• Transaction ID • Buyer address • Seller address • Shipping label generated

Allocation of solutions to deployment components

DB Component	**SB Components Allocated to DB Component**
Web Server Cluster	Server authentication module
Web Server Cluster	Server seller profile maintenance module
Web Server Cluster	Server buyer profile maintenance module
Web Server Cluster	Server payment module
Web Server Cluster	Server shipping module
Database	N/A
Web Browser	N/A

Arman Khondker
EID: aak2464

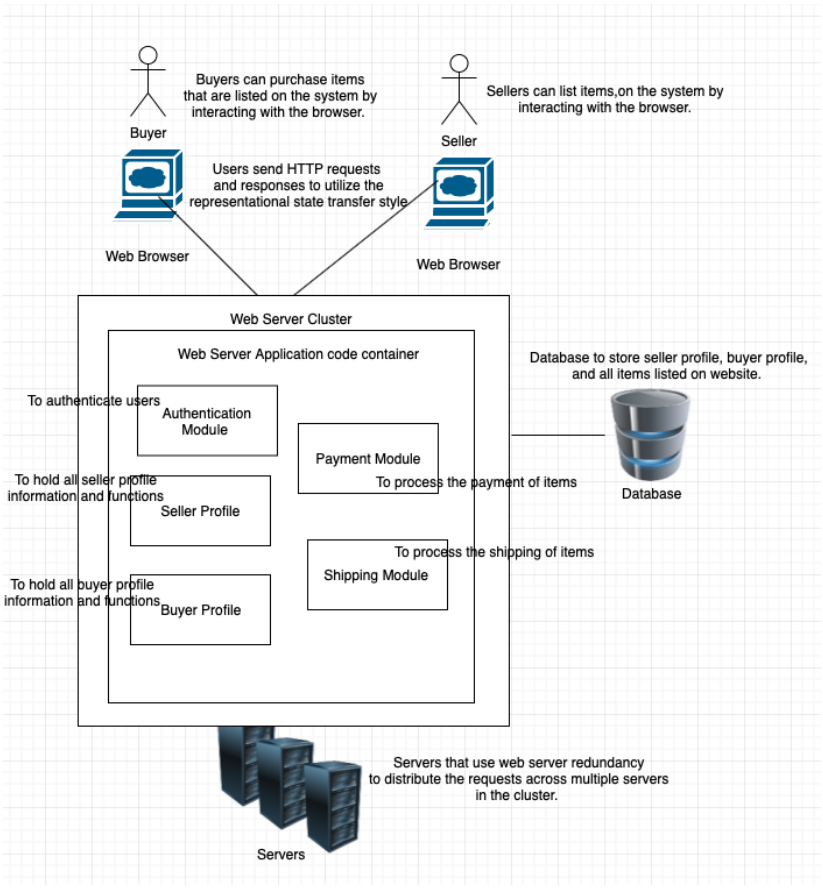
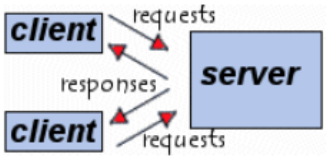


Figure 1: Graphical depiction of Deployment Blueprint #1

Similar Style Reference

> **Client/Server system operation**
>
> A client/server system operates as outlined in the following diagram:
>
> 
>

- ***"*client computers** (computers forming part of the network) contact a **server**, generally a very powerful computer in terms of input/output, which provides services to the client computers."
- https://ccm.net/contents/152-client-server-environment

Arman Khondker
EID: aak2464

Rationale

- Satisfaction of stakeholder qualities (with priorities):
    - 1. **Reliable** – the web server redundancy will allow our system to be reliable because it will be run across multiple different servers and ensure that our system has necessary protection in place. Additionally, the utilization of representational state transfer will ensure that all of our API calls will have a uniform way of calling them and thus make them reliable to use. The inclusion of replication into our system will also allow our system to be more reliable because we will be able to use previous aspects of our system that we know are reliable and reuse them.
    - 2. **Optimized** – our system will become optimized by utilizing distributed computing which will allow us to utilized multiple computer systems to tackle separate problems our system is trying to solve. Additionally, our system will be optimized through the use of representational state transfer which will allow our API calls to be set within uniform standards that allow our calls to maximize our data exchange.
    - 3. **Scalable** – our system will be scalable because we will use the replication design principle. An architecture without replication will suffer from the database server having a low throughput; however, if we implement replication for scalability we can cache response on multiple servers and thus the cache can respond to multiple read requests. This will allow our application to scale to any proportion. Additionally, our client server architecture will allow us to have scalable network to move or add clients without affecting the operation of the network.
    - 4. **Flexible** – our system will be flexible with the inclusion of web server redundancy which will allow our system to have service requests distributed throughout multiple servers that are in clusters. These service requests and redundancy allow us to potentially change our modules in the future and have the ability to be flexible in our system design.
    - 5. **Secure** – our system will be secure through the use of the client-server architecture which will allow our system to not be dependent on the number of entry points giving access to our data. Additionally, representational state transfer will allow us to have a uniform and safe way to transfer data through API calls which will make our system more secure.
    - 6. **Accessible** – our system will be accessible on a wide variety of devices and to all potential users by using the client-server architecture which will allow clients to access all centralized resources in the system easily. We will also use distributed computing to make our system more distributed across multiple servers which allow our components to be accessed at more places.
    - 7. **Fast** – our system will be fast by using distributed computing which will allow our system to solve specific problems on individual servers. Additionally, we will take advantage of representational state transfer to allow our API calls to be faster and more consistent to use.

- 8. **Redundant** – our system will have inherent redundancy of all of our data through the use of web server redundancy which will allow our system to have request distributed across multiple servers which will be in clusters. Additionally, we will use the client-server architecture which will give us access to centralized rescues which will have redundant issue of useful data.
- 9. **Clear Navigation** – our system will have clear navigation features through the use of the replication design principle which will our best UI components to proliferate throughout our system and promote the idea of having a clean and easy to use user interface for our users.
- 10. **Consistent** – the system will be consistent and follow certain principles through the use of representational state transfer which allows us to have a uniform way of sending and receiving data through our APIs. Additionally, our system will have centralized resources from the client server architecture which will allow all of our users to manage the same resources.

There are a variety of architectural styles/design principles that provide the basis for my Deployment Blueprint. The following is a list of four key styles in my deployment blueprint that provided the inspiration for its structure.

- Web server redundancy*:* the requests of the server will be distributed across multiple different servers which will be put together in clusters
- Client-server Architecture: client's computers will be able to connect with a server which will provide all of our system services to the user.
- Replication design principle: the replication design principle will allow us to replicate components and parts of system so that we can take advantage of client-side caching which will increase our system scalability.
- Representational State Transfer: REST will be used to ensure that the client can run our APIs and expect certain results that behave under uniform rules. Additionally, REST will help maintain our client-server architecture and promoting its use.
- Distributed Computing: we will use multiple computer systems to tackle different problems are system is attempting to solve.

The potential conflicts that arise with this given deployment blueprint and architectural styles are that potentially distributed computing could lead to an overuse of resources for our system. Additionally, the use of representational state transfer will require heavy development cost to create solid APIs with great documentation for future use of our protocols. Lastly, a potential conflict can arise with our systems dedication to the replication design principle because we may not fully stay true to our stakeholder need of building an optimized system rather we would be building a more robust system.

The Deployment Blueprint represents a small refactoring of the original Business Blueprint Components. This may sacrifice the Business blueprints vision because it may confuse the intended logic of the overall system by trying to make it more complex. Additionally, this may cause our system to lose sight of our overall stakeholder needs defined in assignment 2; however, this refactoring is justified because it allows us to optimize for those aforementioned

stakeholder needs by implanting ways to incorporate key architecture styles and design principles.

References

- Web server redundancy: https://www.itpro.co.uk/server-storage/34505/what-is-server-redundancy
- Client-server architecture: https://www.techopedia.com/definition/438/clientserver-architecture
- Replication design principle: https://www.cs.cmu.edu/~charlie/courses/15-214/2014-spring/slides/24-distributed-systems-part-4.pdf
- Representational State Transfer: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- Distributed computing: https://www.techopedia.com/definition/7/distributed-computing-system

Arman Khondker
EID: aak2464

Deployment Blueprint #2

Satisfaction of domain functions and I/O by solutions *– Part Manual Solution*

SB Solution Component	**BB Functions Satisfied**
Application Staff	• Authenticate and Authorize User • Authenticate Valid Email • Issue refund • Process Payment • Track shipping Label • Create return label
Server Seller profile maintenance module	• List an item • Edit an item listing • Add description to item • Attach images to listing • Create a shipping label • Remove item listing • View all listings
Server buyer profile maintenance module	• Buy an item • Check recently viewed items • Add an item to watchlist • View shipping status • Search for items • Filter search for items • Sort Search for items • Submit cancellation request • View all order history • View order details • Sign up for alerts • View FAQ • View size chart • Contact seller • Report listing • Edit preferences • Browse suggestions • Pay for an item • Make a return request

SB Solution Component	**BB I/O Satisfied**
Application Staff	• Username • Password

	• Email address • Street Address • Logged into website • Email validated • Transaction Cost • Payment Method • Buyer ID • Seller ID • Transaction ID • Buyer address • Seller address • Shipping label generated
Server Seller profile maintenance module	• Pictures of items • Description of items • Cost of items • Condition of items • Seller history • Listing history
Server buyer profile maintenance module	• Viewed items history • Watchlist of items • Purchase history • Search History • Email preferences • Item preferences • Buyer has reported listing • Account alerts updated

Allocation of solutions to deployment components

DB Component	**SB Components Allocated to DB Component**
Application Staff	Application Staff
Web Cluster	Server Seller profile maintenance module
Web Cluster	Server buyer profile maintenance module
User account system	N/A
File System	N/A

Arman Khondker
EID: aak2464

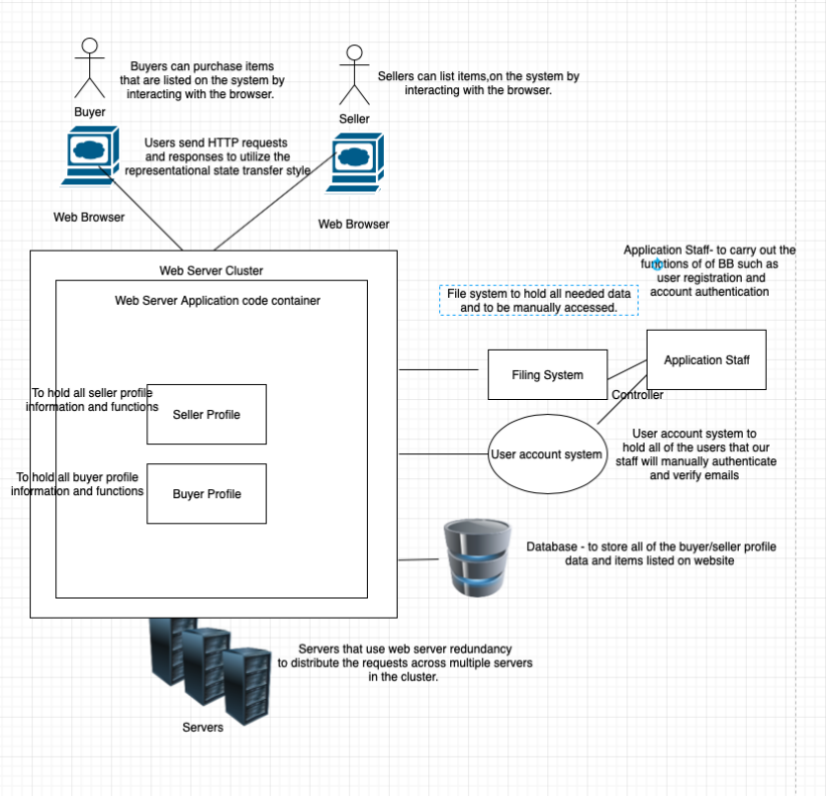
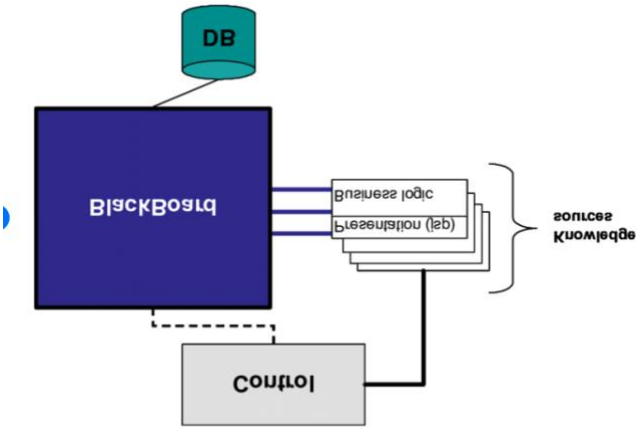


Figure 2: Graphical depiction of Deployment Blueprint #2

Similar Style Reference



- "a well-established style for solving the problem of control, communication and collaboration in a system"
- https://www.researchgate.net/figure/Components-of-the-Blackboard-architecture_fig3_239766921

Arman Khondker
EID: aak2464

Rationale

- Satisfaction of stakeholder qualities (with priorities):
    - 1. **Flexible** – the event driven architecture will allow us to have a flexible system that will be able to be modified based on the given events that we feed into the system. Additionally, our system will be flexible with the inclusion of web server redundancy which will allow our system to have service requests distributed throughout multiple servers that are in clusters.
    - 2. **Accessible** – our system will be accessible on a wide variety of devices and to all potential users by using the client-server architecture which will allow clients to access all centralized resources in the system easily.
    - 3. **Consistent** – our system will be consistent by the use of domain realities which allow us to stay close to our stakeholder needs. Additionally, our use of web server redundancy will allow us to have our system be consistent across multiple different servers.
    - 4. **Reliable** - the web server redundancy will allow our system to be reliable because it will be run across multiple different servers and ensure that our system has necessary protection in place. Additionally, the use of blackboard will allow us to have a controller which will determine what needs to execute and what our system more reliable.
    - 5. **Secure** - our system will be secure through the use of the client-server architecture which will allow our system to not be dependent on the number of entry points giving access to our data.
    - 6. **Scalable** - our client server architecture will allow us to have scalable network to move or add clients without affecting the operation of the network. Additionally, the use of an event driven architecture will allow to scale the system in regard to possible events.
    - 7. **Fast** –our system will be faster through the use of blackboard architecture which will allow us to optimize our speed through a controller component which will increase total run time.
    - 8. **Optimized** – the domain realities that we chose to derive our system will help us make our system more optimized by keeping stakeholder primary needs in mind. Additionally, the use of an event driven architecture will allow us to optimize for potential events.
    - 9. **Redundant** – our system will have inherent redundancy of all of our data through the use of web server redundancy which will allow our system to have request distributed across multiple servers which will be in clusters. Additionally, we will use the client-server architecture which will give us access to centralized rescues which will have redundant issue of useful data.
    - 10. **Clear Navigation** – our system will have clear navigation features though the use of an architecture built on domain realities which will allow us to be more aligned with our primary users work flow.

Arman Khondker
EID: aak2464

There are a variety of architectural styles/design principles that provide the basis for this specific deployment blueprint. The following is a list of four key styles in my deployment blueprint and a description of how these styles address stakeholder needs.

- Client-server Architecture: client's computers will be able to connect with a server which will provide all of our system services to the user.
- Domain Realities: Deployment components closely mirror business blueprint components, this will allow us to maintain a close relationship with the stakeholder needs that we identify as important
- Blackboard: we will have a controller component that determines which module we want to execute and in which order, our blackboard will be all of our available data
- Event driven architecture: we will have portions of our architecture be driven by key events that happen in the system, this will allow us to address stakeholder needs by having events that cater to specific qualities
- Web server redundancy*:* the requests of the server will be distributed across multiple different servers which will be put together in clusters.

The potential conflicts that arise that an arise with this given deployment blueprint and architectural styles are that the combination of the manual solution with the web clusters could lead to a hit in our applications performance. Additionally, the system could suffer from a lack of cohesion because we are incorporating architectural styles that conflict and don't have the best synergy.

The Deployment Blueprint represents a major refactoring of the original Business Blueprint Components. This may sacrifice the Business blueprints vision because we have combined the authenticator, payment processing, and shipping component functionalities into one singular solution which could cause us to lose sight of our top stakeholder needs; however, this refactoring is justified because we can take advantage of all of the perks of the architecture style we have used. This will allow our system to be more flexible to change and more closely aligned with our domain reality.

References

- Client-server architecture: https://www.techopedia.com/definition/438/clientserver-architecture
- Domain Reality: https://www.win.tue.nl/~aserebre/2IW80/2013-2014/A2%20-%20DSSA%20Arch%20patterns.pdf
- Blackboard: http://www.dossier-andreas.net/software_architecture/blackboard.html
- Event Driven Architecture: https://microservices.io/patterns/data/event-driven-architecture.html
- Web server redundancy: https://www.itpro.co.uk/server-storage/34505/what-is-server-redundancy

Arman Khondker
EID: aak2464

Part 2 – Evaluating Solution Blueprint Compliance

Deployment Blueprint #1

Component, c	Technology, t	*CompFuncCoeff(c,t)*	*CompDataCoeff(c,t)*	*CompFuncBoundaryError(t)*
Authenticator	Server authentication module	|2|/2 = **1**	|4|/4 = **1**	(1-1) + 0 = **0**
Seller	Server seller profile maintenance module	|6|/7 = **0.85**	|6|/6 = **1**	(1-0.85) + 0 = **0.15**
Buyer	Server buyer profile maintenance module	|6|/7 = **0.85**	|6|/6 = **1**	(1-0.89) + 0 = **0.11**
Payment Processing	Server payment module	|2|/2 = **1**	|4|/4 = **1**	(1-1) + 0 = **0**
Shipping	Server shipping module	|2|/2 = **1**	|3|/3 = **1**	(1-1) + 0 = **0**

Deployment Blueprint #2:

Component, c	Technology, t	*CompFuncCoeff(c,t)*	*CompDataCoeff(c,t)*	*CompFuncBoundaryError(t)*
Authenticator	Application Staff	|2|/2 = **1**	|4|/4 = **1**	(1-1) + (1+1) = **2**
Seller	Server seller profile maintenance module	|6|/7 = **0.85**	|6|/6 = **1**	(1-0.85) + 0 = **0.15**

Arman Khondker
EID: aak2464

Buyer	Server buyer profile maintenance module	|6|/7 = **0.85**	|6|/6 = **1**	(1-0.89) + 0 = **0.11**
Payment Processing	Application Staff	|2|/2 = **1**	|4|/4 = **1**	(1-1) + (1+1) = **2**
Shipping	Application Staff	|2|/2 = **1**	|3|/3 = **1**	(1-1) + (1+1) = **2**

Arman Khondker
EID: aak2464

Part 3 – ATAM Utility Trees

Deployment Blueprint #1 Quality Attribute Utility Tree

- Utility – we will aggregate quality assessment to obtain an overall "goodness" of the system by ensuring that the metrics defined in the level 5 sublevel of our ATAM Utility tree align with stakeholder needs and can be executed with our system given resources. Additionally, we will ensure that all our objectives are properly aligned with our overall stakeholder goal and reflect all possible aspects as those aforementioned qualities. We will normalize all of our results by giving priority to the qualities which have been identified as higher priority and lower priority for those which are considered less important.
    - 1. Reliable
        - Ensure that the website is always running and have potential backup plans
            - Objective: Ensure that no web server cluster will go down with a degree of 99% or greater certainty (H, M)
                - Metric: Execute scenario "sell an item" and scenario "buy an item" and check to see if the web server goes down. Repeat this process with automated testing of the key scenarios and record the amount of time that the server goes down. Ensure that the amount of times the server goes down is negligible and that we have 99% or more certainty that the server is reliable. (M)
    - 2. Optimized
        - Minimize server response time for key operations such as buying or selling
            - Objective: Perform the "Buy an Item" key scenario and make sure the response time is under 1 second under ideal conditions (M, M)
                - Metric: Record the response times for 'Buy an item' request by using a test script on multiple virtual machines with 20 web clients processes in each. Ensure that the response time is as desired. (H)
    - 3. Scalable
        - Maximize our system's ability to grow and scale
            - Objective: Show that our web server cluster will scale by performing function 'list an item' under progressively large loads (M, M)
                - Metric: Record response times for 'list an item' while using a test script from 10 machines with incrementing web client processes in each. Then, conduct tests with multiple different web servers and compare the response time as the load increases (H)

- 4. Flexible
  - Allow our system to change easily as our stakeholder needs change
    - Objective: Minimize the coupling between key components so that we can potentially change components easily in the future (M, M)
      - Metric: Measure the coupling of our key components by using the metric in milestone #2 where we calculate the # of dependencies between components" (M)
- 5. Secure
  - Ensure transaction integrity
    - Objective: Ensure that all of our system transactions are secure and can't be accessed without proper credentials and clearance (M, L)
      - Metric: Ensure that our systems data is properly encrypted with modern security standards and that our system is properly protected from DDOS attacks (M)
- 6. Accessible
  - Allow our system to work on a variety of devices and browsers
    - Objective: Ensure that our system works on multiple system with different constraints and that our system works on different browsers (M, L)
      - Metric: Run an automated testing script on virtual machines on the top 10 most popular browsers used on the internet. Ensure that our system works on all of these devices with different configurations (L)
- 7. Fast
  - Minimize application running time
    - Objective: Ensure that the application run time of key scenarios is below a threshold of 5 seconds adjusted for user response times (M, M)
      - Metric: Perform key scenario "Sell an item" and measure the amount of time the application takes to run and ensure that it is under the aforementioned threshold (M)
- 8. Redundant
  - Create redundant copies of our data and hold them in a secure place
    - Objective: Use web server redundancy to ensure that that requests of the server will be distributed across multiple different servers (M, M)
      - Metric: Measure how well we have utilized web server redundancy by running an automated testing script that will test if a given request such as "Buy an Item" is distributed across multiple servers (M)

- 9. Clear Navigation

- Ensure easy user interface navigation
  - Objective: Ensure that our UI for the application is easy to use for our primary stakeholders and satisfies the needs of our stakeholders completely (M, M)
    - Metric: Run A/B tests on different user interfaces and gain feedback from potential customers and users of our application. Determine what the best UI is an implement it into our system (M)

- 10. Consistent
  - Ensure item listing integrity
    - Objective: Ensure that the items on the client's page are updated as item data changes (M,M)
      - Metric: Conduct a test where you take an item listing and change the price of the item on a certain client, then you should check if another clients web page shows updates that reflect the changes on the other client's item (M)

Arman Khondker
EID: aak2464

Deployment Blueprint #2 Quality Attribute Utility Tree

- Utility – To obtain overall system "goodness" we will aggregate quality assessments such that the most important qualities are weighed more heavily in our assessment of the system. Additionally, we will pay close attention to how our objectives align with the strengths of the architectural design styles that we have chosen.
    - 1. Flexible
        - Allow our system to change easily as our stakeholder needs change
            - Objective: Minimize the coupling between key components so that we can potentially change components easily in the future (M, M)
                - Metric: Measure the coupling of our key components by using the metric in milestone #2 where we calculate the # of dependencies between components" (M)
    - 2. Accessible
        - Allow our system to work on a variety of devices and browsers
            - Objective: Ensure that our system works on multiple system with different constraints and that our system works on different browsers (M, L)
                - Metric: Run an automated testing script on virtual machines on the top 10 most popular browsers used on the internet. Ensure that our system works on all of these devices with different configurations (L)
    - 3. Consistent
        - Ensure item listing integrity
            - Objective: Ensure that the items on the client's page are updated as item data changes (M,M)
                - Metric: Conduct a test where you take an item listing and change the price of the item on a certain client, then you should check if another clients web page shows updates that reflect the changes on the other client's item (M)
    - 4. Reliable
        - Ensure that the website is always running and have potential backup plans
            - Objective: Ensure that no web server cluster will go down with a degree of 99% or greater certainty (H, M)
                - Metric: Execute scenario "sell an item" and scenario "buy an item" and check to see if the web server goes down. Repeat this process with automated testing of the key scenarios and record the amount of time that the server goes down. Ensure that the amount of times the server goes down is negligible and that we have 99% or more certainty that the server is reliable. (M)
    - 5. Secure

- Ensure transaction integrity
  - Objective: Ensure that all of our system transactions are secure and can't be accessed without proper credentials and clearance (M, L)
    - Metric: Ensure that our systems data is properly encrypted with modern security standards and that our system is properly protected from DDOS attacks (M)

- 6. Scalable
  - Maximize our system's ability to grow and scale
    - Objective: Show that our web server cluster will scale by performing function 'list an item' under progressively large loads (M, M)
      - Metric: Record response times for 'list an item' while using a test script from 10 machines with incrementing web client processes in each. Then, conduct tests with multiple different web servers and compare the response time as the load increases (H)
- 7. Fast
  - Minimize application running time
    - Objective: Ensure that the application run time of key scenarios is below a threshold of 5 seconds adjusted for user response times (M, M)
      - Metric: Perform key scenario "Sell an item" and measure the amount of time the application takes to run and ensure that it is under the aforementioned threshold (M)
- 8. Optimized
  - Minimize server response time for key operations such as buying or selling
    - Objective: Perform the "Buy an Item" key scenario and make sure the response time is under 1 second under ideal conditions (M, M)
      - Metric: Record the response times for 'Buy an item' request by using a test script on multiple virtual machines with 20 web clients processes in each. Ensure that the response time is as desired. (H)
- 9. Redundant
  - Create redundant copies of our data and hold them in a secure place
    - Objective: Use web server redundancy to ensure that that requests of the server will be distributed across multiple different servers (M, M)
      - Metric: Measure how well we have utilized web server redundancy by running an automated testing script that will test if a given request such as "Buy an Item" is distributed across multiple servers (M)

- 10. Clear Navigation
  - Ensure easy user interface navigation
    - Objective: Ensure that our UI for the application is easy to use for our primary stakeholders and satisfies the needs of our stakeholders completely (M, M)
      - Metric: Run A/B tests on different user interfaces and gain feedback from potential customers and users of our application. Determine what the best UI is an implement it into our system (M)