# Homework #2: Architectural Design

**Team: Caliware**

|  |  |  |
| --- | --- | --- |
| **Timothy Cioffi-Dinkel** | **Anthony Farina** | **Joanna Hang** |
| timothycioffi@csu.fullerton.edu | afarina@csu.fullerton.edu | johang@fullerton.edu |
|  |  | sig.jpg |

|  |  |
| --- | --- |
| **Lourdes Lopez** | **David Sullivan** |
| lourdeslopez@csu.fullerton.edu | dsullivan@fullerton.edu |
| Untitled.png |  |

Date Submitted: May 4, 2016

CPSC545: Software Design Architecture

Spring 2016

Dr. Chang-Hyun Jo

# Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Description** | **Member** |
| 03/03/2016 | Vision Statement | Lourdes, Anthony, David, Joanna, Timothy |
| 03/03/2016 | System Context Diagram | Joanna and Lourdes |
| 03/08/2016 | Domain Knowledge | Joanna |
| 03/08/2016 | Technical Environment | Anthony |
| 03/08/2016 | Use Cases | David and Timothy |
| 03/08/2016 | Quality Attributes | Anthony, David, Joanna, Lourdes, and Timothy |
| 3/19/2016 | QAW | Anthony, David, Joanna, Lourdes, and Timothy |
| 3/26/2016 | Tactics | Anthony, David, Joanna, Lourdes, and Timothy |
| 4/2/2016 | ADRT | Anthony, David, Joanna, Lourdes, and Timothy |
| 4/2/2016 | Utility Tree | Anthony, David, Joanna, Lourdes, and Timothy |
| 4/9/2016 | ADD Iteration 1 | Anthony, David, Joanna, Lourdes, and Timothy |
| 4/16/2016 | ADD Iteration 2 | Anthony, David, Joanna, Lourdes, and Timothy |
| 4/23/2016 | Documentation | Anthony, David, Joanna, Lourdes, and Timothy |

# Table of Contents

[Homework #2: Architectural Design 1](#_Toc450159247)

[Revision History 2](#_Toc450159248)

[Table of Contents 3](#_Toc450159249)

[Exercise 1 6](#_Toc450159250)

[Vision 6](#_Toc450159251)

[Vision Statement 6](#_Toc450159252)

[Business Goals 6](#_Toc450159253)

[Target Market 6](#_Toc450159254)

[Users 6](#_Toc450159255)

[Stakeholders 6](#_Toc450159256)

[System Features 6](#_Toc450159257)

[Functional Requirements 6](#_Toc450159258)

[Nonfunctional Requirements 7](#_Toc450159259)

[Software Requirements 7](#_Toc450159260)

[Hardware Requirements 7](#_Toc450159261)

[System Context Diagram 8](#_Toc450159262)

[Technical Environment 9](#_Toc450159263)

[Domain Knowledge 11](#_Toc450159264)

[System Features and Use Cases 13](#_Toc450159265)

[List of System Features 13](#_Toc450159266)

[Use Cases for System Features 13](#_Toc450159267)

[Quality Attributes 22](#_Toc450159268)

[Availability 22](#_Toc450159269)

[Interoperability 22](#_Toc450159270)

[Modifiability 23](#_Toc450159271)

[Performance 23](#_Toc450159272)

[Security 24](#_Toc450159273)

[Testability 24](#_Toc450159274)

[Usability 25](#_Toc450159275)

[Portability 25](#_Toc450159276)

[Monitorability 26](#_Toc450159277)

[Exercise 2 28](#_Toc450159278)

[Quality Attribute Workshop 28](#_Toc450159279)

[0. Logistics for QAW, target system, QAW team, facilitator. 28](#_Toc450159280)

[1. QAW Presentation and Introductions 28](#_Toc450159281)

[2. Business/Mission Presentation 29](#_Toc450159282)

[3. Architectural Plan Presentation 29](#_Toc450159283)

[4. Identification of Architectural Drivers 31](#_Toc450159284)

[5. Scenario Brainstorming 31](#_Toc450159285)

[6. Scenario Consolidation 34](#_Toc450159286)

[7. Scenario Prioritization 37](#_Toc450159287)

[8. Scenario Refinement 40](#_Toc450159288)

[Find Tactics 46](#_Toc450159289)

[Quality Attribute Utility Tree 47](#_Toc450159290)

[Architectural Design Relation Table (ADRT) 48](#_Toc450159291)

[Exercise 3 51](#_Toc450159292)

[Iteration One 51](#_Toc450159293)

[Step 0 51](#_Toc450159294)

[Step 1: Confirm that there is sufficient requirements information 56](#_Toc450159295)

[Step 2: Choose an element of the system to decompose 56](#_Toc450159296)

[Step 3: Identify candidate architectural drivers 56](#_Toc450159297)

[Step 4 (Design Step): Choose a design concept that satisfies the architectural drivers. 60](#_Toc450159298)

[Step 5 (Design Step): Instantiate architectural elements and allocate responsibilities. 65](#_Toc450159299)

[Step 6 (Design Step): Define interfaces for instantiated elements. 68](#_Toc450159300)

[Step 7: Verify and refine requirements and make them constraints for instantiated elements. 70](#_Toc450159301)

[Iteration Two 70](#_Toc450159302)

[Step 2: Choose an element of the system to decompose 70](#_Toc450159303)

[Step 3: Identify candidate architectural drivers 70](#_Toc450159304)

[Step 4 (Design Step): Choose a design concept that satisfies the architectural drivers. 71](#_Toc450159305)

[Step 5 (Design Step): Instantiate architectural elements and allocate responsibilities. 75](#_Toc450159306)

[Step 6 (Design Step): Define interfaces for instantiated elements. 79](#_Toc450159307)

[Step 7: Verify and refine requirements and make them constraints for instantiated elements. 80](#_Toc450159308)

[ADD Rationale 82](#_Toc450159309)

[Software Architecture Documentation 84](#_Toc450159310)

[Stakeholder View Table 84](#_Toc450159311)

[Document Package 85](#_Toc450159312)

[Lessons Learned 93](#_Toc450159313)

[Timothy Cioffi-Dinkel 93](#_Toc450159314)

[Anthony Farina 93](#_Toc450159315)

[Joanna Hang 93](#_Toc450159316)

[Lourdes Lopez 93](#_Toc450159317)

[David Sullivan 94](#_Toc450159318)

[Group Experience 94](#_Toc450159319)

[Reference List 95](#_Toc450159320)

[Appendix 97](#_Toc450159321)

[Appendix A - Quality Attribute Workshop Presentation 97](#_Toc450159322)

[Team Charter 105](#_Toc450159323)

[Team Evaluation 111](#_Toc450159324)

# Exercise 1

## Vision

### Vision Statement

For online shoppers, who want to purchase products at discount prices on the go, the Fullerton eShopping system provides access to purchase products from anywhere and anytime. Unlike Amazon.com, our product specializes in sellers interested in high volume sales at lower than typical retail seller prices. The more products the customers purchase, the lower the unit cost.

### Business Goals

* Generate revenue - Achieve positive cash flow
* Increase ROI - Achieve 35% return on investment within 6 months
* Increase sales volume - Increase sales volume per product by 15%

### Target Market

Our target market is online shoppers interested in volume shopping and discount prices.

### Users

The term Users for the system includes two individuals accessing the system. Buyers, or for this project just users, are those wishing to purchase any of the available items from the system. They have access to an account created by them which has all personal information as well as credit card information and tracking options.

Users also include Seller's or individuals who are placing products onto the system. They too have access to an account but will have uploading abilities and personal/banking information will be listed in their account.

### Stakeholders

* Customer
* Seller
* Internal Departments: Customer Service, Accounting, Shipping, Warehouse, Marketing
* Project Team: Development Manager, Product Owner, Architect, Developer, QA and Tester
* Management Team
* Company Executives

## System Features

### Functional Requirements

The eShopping System shall have a set of functional requirements such as:

1. eShopping System shall allow users to sign-up for an account
2. eShopping System shall authenticate user and seller accounts via email confirmation and password
3. eShopping System shall allow users to login/logout of accounts
4. eShopping System shall allow users to update profile
5. eShopping System shall allow users to safely purchase items from the store
6. eShopping System shall record accesses to the system via audit trails for potential hackers
7. eShopping System shall store credit card information and bank account information for both purchases and selling.
8. eShopping System shall allow sellers to post items for sale.

### Nonfunctional Requirements

The eShopping System shall have a set of nonfunctional requirements as well, such as:

1. eShopping System shall allow users to place items into their virtual cart within less than a second of selecting the option.
2. eShopping System shall place a seller’s item onto its own page within fifteen seconds of the seller’s verification.
3. eShopping System shall be available 99.99%; it shall only be offline for eight hours a year for maintenance or system issues.
4. eShopping System shall flag and track illegal access’ to the system.
5. eShopping System shall have a monthly update with new features and additions.
6. eShopping System shall allow an average of 3,000,000 transactions per minute.
7. eShopping System is web based and shall be allowed on desktop and portable web browsers.
8. eShopping System shall allow for credit card transactions for the time allotted by the bank's transaction speed.

### Software Requirements

The eShopping System shall be available for Windows, OSX, or Linux operating systems. The main focus will be on Windows and full access by OSX or Linux is not guaranteed to be as updated.

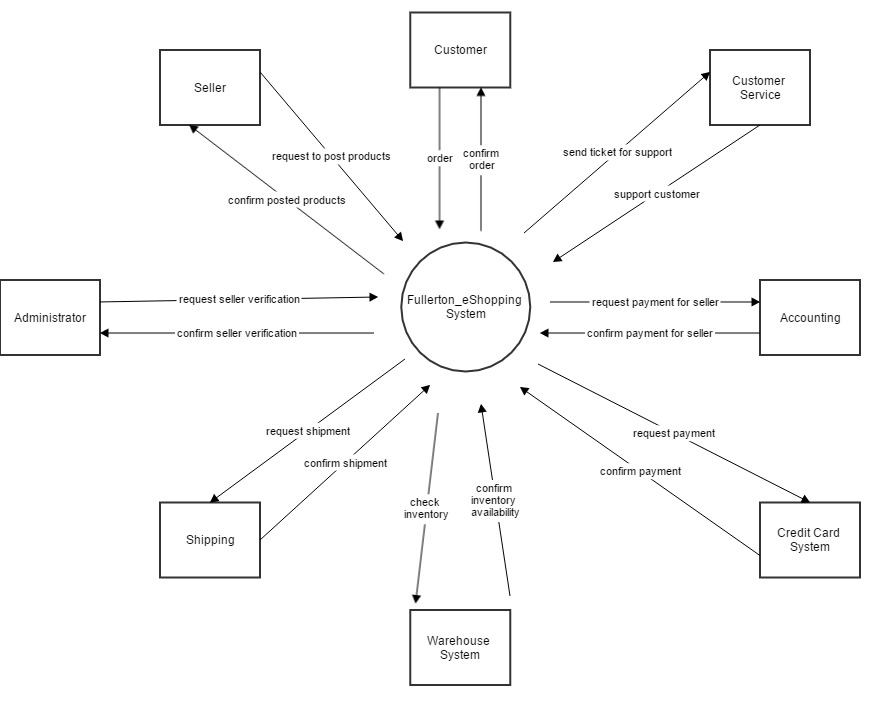
The eShopping System shall be able to be used on any of the major web browsers. This team does not see an issue with using non-major web browsers, but issues could possibly arise. Major web browsers include Google Chrome, Mozilla Firefox, and Internet explorer/Edge.

The eShopping System shall include mobile applications available for use with IOS or Android mobile devices.

### Hardware Requirements

The eShopping System is a web-based system, therefore the hardware requirement is for a computer to have the ability to access the internet. This requires a working computer and access to a LAN or WIFI source.

## System Context Diagram



## Technical Environment

When planning on building an ecommerce system, there are a variety of standards, regulations, practices, and engineering techniques that need to be considered before design and construction begins. The technical environment for an ecommerce system has a lot in common with other systems but it also has some unique characteristics.

Starting with regulations and standards, e-commerce is unique in that it has to abide by government standards set by the FTC. These standards and regulations can be confusing because they span many different acts. One of the main acts that needs to be understood is the CAN-SPAM Act. The CAN-SPAM Act as stated on the FTC website, *“requires the Commission to issue regulations “defining the relevant criteria to facilitate the determination of the primary purpose of an electronic mail message.” The CAN-SPAM Act applies almost exclusively to “commercial electronic mail messages”.”* (n.d.-b). Another act that must be followed is the Mail, Internet, or Telephone Order Merchandise Rule. This rule “*requires that when you advertise merchandise, you must have a reasonable basis for stating or implying that you can ship within a certain time. If you make no shipment statement, you must have a reasonable basis for believing that you can ship within 30 days.”* (n.d.-a).It also states how to handle an order when you can’t ship in the time that was stated previously. The FTC has several other rules and acts that must be considered and followed when selling products online which range from guidelines to customer privacy and advertising. An overview of all of these acts/rules/regulations can be found at the FTC website (n.d.-a).

Another unique characteristic for e-commerce is that the system must be PCI compliant. These standards require that the system must be able to securely process, transmit, and store credit card information (Apple, n.d.). Since credit card payments are a major part of an e-commerce system, these standards will be critical when designing and constructing the system.

Security and privacy are very important to an e-commerce system. There are plenty of industry standards and practices which aim at ensuring security and privacy such as encryption or HTTPS/SSL.

The system will also include mobile applications. Mobile applications bring about their own industry design standards. The two main mobile application environments/OS’s are Android and iOS. Google and Apple provide design standards for developing applications in their respective ecosystems. These standards can be found at <http://developer.android.com/design/index.html> for Android and <https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/> for iOS.

There are plenty of other industry practices and standards that must be considered when developing an ecommerce system. Different types of web client frameworks have their own practices and standards. There are different types of databases which could fulfill the system’s needs. Scalability will need to be considered along with cloud based technologies.

## Domain Knowledge

Domain knowledge for e-commerce systems include the following: shopping cart solution, shipping and fulfillment, payment gateway, SSL (Secure Sockets Layer), storefront, and PCI compliance (Payment Card Industry Data Security Standard).

**Shopping Cart Solution**

The shopping cart solution tracks what the customer wants to purchase. The customer adds products he/she wishes to purchase into his/her shopping cart. When the customer checks out, the customer shall be taken to the payment page.

**Shipping and Fulfillment**

Shipping and fulfillment estimates the shipping cost for the packages. Shipping cost can be determined by a few factors: speed, departing location, destination, weight, content, size, insurance, and tracking (Lazazzera, n.d.). Shipping is typically fulfilled in a business partnership through one of these mail carriers: USPS, UPS, or FedEx.

**Payment Gateway**

In e-commerce systems, a payment gateway is needed to facilitate and authorize payments. The gateway processes credit card information and sends the information to the processor to determine if the sale is authorized. Payment can be directly entered onto the server or redirected to a hosted payment page (E-commerce Platforms, 2014).

**Secure Sockets Layer (SSL)**

SSL (Secure Sockets Layer) establishes an encrypted link between the server and client; it encrypts data sent to and from the web site (Digicert, n.d.). An SSL certificate is crucial to an e-commerce site as customers will be able to determine if the site should be trusted if secure.

**Storefront**

The storefront pertains to the online store. It may include the shopping cart solution, is integrated with payment gateways, tracks and updates inventory, sends notification to the sellers with a warning of low inventory, and also generates sales reports and traffic data (Pair Network, n.d.).

**PCI Compliance (Payment Card Industry Data Security Standard)**

An e-commerce system must be PCI compliant, which ensures the system meets the data security standards for payments to process, store, transmit credit card information (PCI Security Standards Council, 2016). This ensures the protection and safety of data. According to the standards, thee-commerce solution must maintain a secure environment. The standard also sets the technical and operational requirements (PCI Security Standards Council, 2016). The e-commerce system must be validated payment software. It shall not store any cardholder data, the network must contain firewall rules, and the system must encrypt transmission of cardholder data (PCI Security Standards Council, 2015).

## System Features and Use Cases

### List of System Features

* FE-1 The system allows customers to create accounts.
* FE-2 The system allows sellers to create seller accounts.
* FE-3 The system allows sellers to post product for sale.
* FE-4 The system allows sellers to define a pricing scale for their products.
* FE-5 The system allows customers to place items into a shopping cart.
* FE-6 The system allows customers to make a purchase (check-out).
* FE-7 The system keeps track of inventory.
* FE-8 The system allows the customer to track shipping on a purchase.
* FE-9 The system allows tracks seller transactions for processing seller payments.
* FE-10 The system track issues reported by customers for customer service.

### Use Cases for System Features

#### UC-1 Customer Account

|  |  |
| --- | --- |
| ID | UC-1 |
| Name | Customer Account |
| Description | Customer registers for an account |
| Actors | Customer, eShopping System |
| Trigger | Customer initiates a request to create an account |
| Preconditions | The customer does not have an account |
| Postconditions | The customer account is created |
| Main | 1. The customer enters First Name, Last Name, Email Address and Confirm Email 2. The email address will be used as the customer’s login ID 3. The eShopping system will display the customer's information. 4. The eShopping system will send a verification email to the customer’s email address with a customer ID and instructions to verify the account 5. The Customer will click on the link in the verification email (with the customer ID encoded) and register an account by entering Password, and Confirm Password 6. The customer agrees to terms and conditions. |
| Alternate | none |
| Exception | The email address is already registered |

#### UC-2 Seller Registration

|  |  |
| --- | --- |
| ID | UC-2 |
| Name | Seller Registration |
| Description | Seller registers to be allow to sell products |
| Actors | Seller, Administrator, eShopping System |
| Trigger | Seller initiates an application to sell products |
| Preconditions | The Seller is not registered |
| Postconditions | * The Seller account is created |
| Main | 1. The Seller enters First Name, Last Name, Business Name, Tax ID, Email Address, Confirm Email Address 2. The eShopping system will display a receipt of seller information and the seller information will be processed in 2-5 business days 3. The eShopping system emails the Administrator(s) of a new Seller registration record 4. An Administrator will verify seller information - UC-5 5. The eShopping system will send an approval email to the seller email address with a seller ID and instructions to register an account 6. The Seller will click on the link in the approval email (with the seller ID encoded) and register an account by entering Password, and Confirm Password 7. The seller agrees to and submits the Seller Contract. |
| Alternate | The Administrator denies the Seller Registration and an email is sent to the Seller with a denial reason. |
| Exception | Registration is denied |

#### UC-3 Post Product for Sale

|  |  |
| --- | --- |
| ID | UC-3 |
| Name | Post Product for Sale |
| Description | Seller posts products through their account for buyers to place in shopping cart. |
| Actors | Sellers, eShopping System |
| Trigger | Seller uploads product information to the eShopping system |
| Preconditions | Seller has a product they wish to sell |
| Postconditions | Seller defines pricing for product |
| Main | 1. Seller uploads images, enters product description and enters price 2. Confirmation email is sent to buyer for which they have to click on link to verify product uploading 3. Product page is created with the product’s information posted as well as Seller’s information |
| Alternate | none |
| Exception | Confirmation email is not verified. |

#### UC-4 Define Pricing for Product

|  |  |
| --- | --- |
| ID | UC-4 |
| Name | Define Pricing for Product |
| Description | Sellers define a pricing scale for their products |
| Actors | Sellers, eShoping System |
| Trigger | Seller defines price for product |
| Preconditions | Seller finished uploading product information |
| Postconditions | Seller’s product in posted on website for buyers to buy |
| Main | 1. The Seller enters the amount per item 2. For discount prices, the seller enters the quantity range and amount per item 3. Seller clicks post to site |
| Alternate | none |
| Exception | none |

#### UC-5 Shopping Cart

|  |  |
| --- | --- |
| ID | UC-5 |
| Name | Shopping Cart |
| Description | Customer will place wanted items into a virtual shopping cart before purchasing items. This acts as a storage area for wanted items in the case of multiple purchases |
| Actors | Customer, eShopping System |
| Trigger | Customer clicks on the “add to cart” button in the product’s page |
| Preconditions | Customer wants to buy an item listed in the eShopping System |
| Postconditions | Customer has item in cart, displaying the amount within the shopping cart in icon |
| Main | 1. Customer sees item they wish to buy 2. Customer clicks on the “add to cart” button listed on the page with the quantity of 1 in drop-down menu 3. Quantity can be increased in the drop-down menu 4. eShopping System shall copy the product into the user’s cart for viewing and checkout 5. eShopping System will send email to user of their product they sent to cart 6. Items in cart will stay in cart as long as the buyer is signed into the customer account... 7. If the user is not signed in, the items will stay in the cart as long as the buyer does not clear the browser cookies. |
| Alternate | none |
| Exception | none |

#### UC-6 Checkout

|  |  |
| --- | --- |
| ID | UC-6 |
| Name | Checkout |
| Description | Customer purchases items in shopping cart |
| Actors | Customer, eShopping System, Accounting System, Shipping System |
| Trigger | The customer clicks the checkout button |
| Preconditions | The customer has items for purchase in shopping cart |
| Postconditions | * The customer has been charged * The seller payment has been confirmed * The order has been submitted * The customer has the Order Number and display “Order completed” * A confirmation email has been sent to the Customer |
| Main | 1. The user confirms the items in the shopping cart that have been selected 2. The user enters Billing Address: First Name, Last Name, Email Address, Confirm Email Address, Address Line 1, Address Line 2, City, State, selects Country, and enters Zip Code 3. The customer selects checkbox “Is Shipping Address” or if not, enters the required Shipping Address information with the same required address fields as above 4. Shipping method is selected: 1) 5-day shipping 2) 2-day shipping 3) overnight shipping 5. The user enters Credit Card and security code 6. The Credit Card System will confirm payment 7. The Accounting System will confirm payment for the seller 8. The eShopping system will display an Order Number and an “Order completed” message. 9. The eShopping system will submit the order to be processed by the Shipping system 10. The eShopping system will send a confirmation email to the Customer. |
| Alternate 1 | The user does not complete the order. |
| Alternate 2 | The user already has a registered account:   1. The user confirms the items in the shopping cart that have been selected 2. The user confirms First Name, Last Name, Email Address, Confirm Email Address, Address Line 1, Address Line 2, City, State, selects Country, and enters Zip Code 3. The user selects checkbox “Is Shipping Address” or if not, enters the required Shipping Address information with the same required address fields as above 4. Shipping method is selected: 1) 5-day shipping 2) 2-day shipping 3) overnight shipping 5. The user confirms Credit Card 6. The Credit Card System will confirm payment 7. The Accounting System will confirm payment for the seller 8. The eShopping system will display an Order Number and an “Order completed” message. 9. The eShopping system will submit the order to be processed by the Shipping system 10. The eShopping system will send a confirmation email to the Customer. |
| Exception | The system will display an error if the billing information or shipping information is invalid. |

#### UC-7 Track Inventory

|  |  |
| --- | --- |
| ID | UC-7 |
| Name | Keep track of inventory |
| Description | System keeps track of inventory |
| Actors | eShopping System, Accounting System, Inventory staff, Shipping System |
| Trigger | The customer completes a purchase and deduces the inventory count |
| Preconditions | The customer completed a purchase |
| Postconditions | Inventory count lowers. |
| Main | 1. The available inventory to customer purchase count will be deduced when orders are completed. 2. When the inventory reaches below 10, the item will post a message on its product page “Only X amount available” |
| Alternate 1 | The customer cancels an order   1. Items are re-added back to the inventory automatically |
| Alternate 2 | The customer returns an order   1. Items are inspected for damage and/or if items have been open. 2. If items have not been opened or are undamaged, the inventory staff will add the inventory back. |
| Exception | none |

#### UC-8 Track Shipping

|  |  |
| --- | --- |
| ID | UC-8 |
| Name | Track Shipping |
| Description | User can track items that are being shipped |
| Actors | User, eShopping System |
| Trigger | User accesses the tracking page within their account |
| Preconditions | User buys an item |
| Postconditions | User knows the location of their purchase |
| Main | 1. eShopping System shall have a button in a user’s account to track package 2. eShopping System shall gain the tracking information from the shipping carrier 3. eShopping System shall display the location of the package 4. eShopping System shall notify the user through email and displayed in the tracking page when the package has been delivered |
| Alternate | none |
| Exception | none |

#### UC-9 Seller Payments

|  |  |
| --- | --- |
| ID | UC-9 |
| Name | Seller Payments |
| Description | eShopping System will pay sellers the amount for the product minus the System’s fees |
| Actors | eShopping System, Seller |
| Trigger | User pays for an item |
| Preconditions | User’s payment is received and processed |
| Postconditions | Seller is paid |
| Main | 1. eShopping shall alert the seller through email that their product has been purchased 2. eShopping shall receive payment from bank and withhold pre-determined amount for profitability 3. eShopping shall pay remaining balance to seller through direct deposit |
| Alternate | none |
| Exception | 1. Payment does not go through from buyer 2. seller’s bank account information is not correct |

#### UC-10 Track Issues for Customer Server

|  |  |
| --- | --- |
| ID | UC-10 |
| Name | Track issues for Customer Service |
| Description | Ticketing system that tracks issues from users, tracks returns, and verifies price matching |
| Actors | Users, eShopping System’s customer service department (CSD) |
| Trigger | User calls into CSD |
| Preconditions | User has an issue they would like to report |
| Postconditions | The issue is resolved |
| Main | 1. eShopping System shall have a call center for incoming calls 2. eShopping System shall submit each call as a ticket for tracking purposes 3. eShopping System shall have access to seller and user accounts in order to resolve issues 4. eShopping System shall submit code issues to a bug tracking system |
| Alternate | Online ticketing system where issues are submitted from the user’s account |
| Exception | none |

## Quality Attributes

### Availability

We Ping / Echo at intervals to detect when the Server is unresponsive. The Communication Channel under normal operation would respond by logging the fault. Then the Communication channel switches to a redundant active server. Server is available 99.99%.

|  |  |
| --- | --- |
| **Source** | Ping / Echo |
| **Stimulus** | Server Unresponsive |
| **Artifact** | Communication Channels |
| **Environment** | Normal Operations |
| **Response** | Log fault  Fix by switching to Active Redundancy |
| **Response Measure** | Availability 99.99% |

### Interoperability

Our eShopping System submits an order with the credit card payment information. The credit card payment system is known prior to runtime. The information is exchanged and a notification is sent if the request was rejected.

|  |  |
| --- | --- |
| **Source** | eShopping System |
| **Stimulus** | Submit Order |
| **Artifact** | Credit Card Payment System |
| **Environment** | Known prior to runtime |
| **Response** | Information is exchanged  Notification is sent if the request is rejected |
| **Response Measure** | Our information included correctly 99.99% of the time |

### Modifiability

The seller requests added functionality for new reports to be created in the user interface. At design time, the functionality is added and tested in 2 weeks.

|  |  |
| --- | --- |
| **Source** | End User - Sellers |
| **Stimulus** | Requests added functionality for new report |
| **Artifact** | User Interface |
| **Environment** | Design Time |
| **Response** | Functionality added and tested |
| **Response Measure** | 2 weeks |

### Performance

The CPU Monitor periodically reads the CPU usage statistics during normal operation. The eShopping System continues to operate with no interruption. The CPU measure should be below 80% usage 99.99% of the time.

|  |  |
| --- | --- |
| **Source** | CPU Monitor |
| **Stimulus** | Periodic |
| **Artifact** | CPU |
| **Environment** | Normal Operations |
| **Response** | Continues to operate |
| **Response Measure** | Below 80% usage 99.99% of the time |

### Security

An unauthorized person attempts to access credit card information by accessing data within the system during normal operation. An audit trails traces the user IP address and account and an analysis of the compromised data is completed in 48 hours.

|  |  |
| --- | --- |
| **Source** | Unauthorized person |
| **Stimulus** | Attempts to access credit card information |
| **Artifact** | Data within the system |
| **Environment** | Normal Operations |
| **Response** | Audit trail |
| **Response Measure** | Analysis of compromised data is completed within 48 hours |

### Testability

Upon completion of a unit of source code, an integration tester completes an integration test and captures the results. The integration tester completes the integration test 1-8 hours depending on the complexity of the change.

|  |  |
| --- | --- |
| **Source** | Integration Tester |
| **Stimulus** | Unit of Code is complete |
| **Artifact** | Source Code |
| **Environment** | Integration time |
| **Response** | Capture results |
| **Response Measure** | 1-8 hours to complete integration test depending on complexity of change |

### Usability

The customer learns to place an order in the system at runtime provides the user with intuitive shopping cart feature for placing an order. The average time to make a purchase after the user is ready for checkout is within 2 minutes.

|  |  |
| --- | --- |
| **Source** | Customer |
| **Stimulus** | Learns how to place an order |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | eShopping system provides user with intuitive shopping cart feature for placing an order |
| **Response Measure** | Average first time purchase can be made within 2 minutes once customer is ready for checkout |

### Portability

#### General scenario:

|  |  |
| --- | --- |
| **Source** | End User, Developer |
| **Stimulus** | run on different platform, move component to a different platform, access software from a different platform |
| **Artifact** | User interface, application code or components |
| **Environment** | Normal Operation, design time, development time, deployment time, run time |
| **Response** | use, modify, deploy |
| **Response Measure** | Calendar Time  Number of new defects introduced  Number of affected artifacts |

#### Concrete scenario:

The user accesses the software on multiple mobile devices at run time and the features continue to operate and 99.9% of the features work as designed.

|  |  |
| --- | --- |
| **Source** | User |
| **Stimulus** | Accesses application |
| **Artifact** | Multiple mobile devices |
| **Environment** | Run time |
| **Response** | Features continues to operate |
| **Response Measure** | 99.9% of features work as designed |

### Monitorability

#### General scenario:

|  |  |
| --- | --- |
| **Source** | Operations staff |
| **Stimulus** | Incorrect response, incorrect timing, crash |
| **Artifact** | Processors, communication channels, persistent storage, processes, queue lengths, transaction processing |
| **Environment** | Normal Operations, repair mode, degraded operation, overloaded operation |
| **Response** | log, notify |
| **Response Measure** | Time to detect fault |

#### Concrete scenario:

An operations staff monitors transactions processing for timing under normal operation and sends a notification within 1 minute when the transaction processing increases 20% or more from the average processing time.

|  |  |
| --- | --- |
| **Source** | Operations staff |
| **Stimulus** | Monitors transaction processing for timing |
| **Artifact** | Transaction processing |
| **Environment** | Normal Operations |
| **Response** | Notify |
| **Response Measure** | Detects increase of 20% or more from average transaction processing time and notifies within 1 minute |

# Exercise 2

## Quality Attribute Workshop

The Quality Attribute Workshop (QAW) is a facilitated method that engages stakeholders to discover the driving quality attributes such as performance, security, availability, etcetera of a software-intensive system. The QAW discovers, documents, and prioritizes the system’s quality attribute, and the QAW is completed before creating the software architecture.

### 0. Logistics for QAW, target system, QAW team, facilitator.

**Target System**

The target system is the Fullerton eShopping System. For online shoppers, who want to purchase products at discount prices on the go, the Fullerton eShopping system provides access to purchase products from anywhere and anytime. Unlike Amazon.com, our product specializes in sellers interested in high volume sales at lower than typical retail seller prices. The more products the customers purchase, the lower the unit cost.

**QAW Team**

* Timothy Cioffi-Dinkel, developer
* Anthony Farina, seller
* Joanna Hang, customer
* Lourdes Lopez, Lead developer
* David Sullivan, CEO

**QAW Facilitator**

* Timothy Cioffi-Dinkel

### 1. QAW Presentation and Introductions

The QAW facilitator described the motivation for a QAW and detailed the steps. A slideshow presentation was used to facilitate the QAW (Appendix A). The purpose was to discuss the overall target architecture of the commercial e-commerce system. The goals were to identify important quality attributes and tradeoffs and to clarify the system requirements by defining scenarios.

The steps to the QAW are:

1. Introductions
2. Business Mission
3. Architectural Plan
4. Architectural Drivers
5. Scenario Brainstorming
6. Scenario Consolidation
7. Scenario Prioritization
8. Scenario Refinement

### 2. Business/Mission Presentation

The stakeholders presented the system’s business drivers. From this, the facilitator used the business mission to capture any relevant information that may define or affect the quality attribute drivers.

The business drivers were identified as:

1. The mission of Fullerton eShopping is to offer customers a unique online and mobile shopping experience anywhere, anytime and is secure.
2. Our product specializes in sellers interested in selling high volume sales and discount prices.
3. Our customer benefits from finding good prices at a lower unit costs for large purchases.
4. We will generate a profit from providing the services and a fair return on our investment from a percentage of the sales.
5. We want people to search to be able to easily find our seller’s products and get a good discount when ordering in bulk.
6. We want to generate revenue and achieve a positive cash flow.
7. We want to achieve 35% return on investment within six months.
8. We want to increase sales volume per product by 15%.
9. We want our customers to be happy with our product that is efficient, performs well, and reliable.
10. We want our system to keep up with technology and implement new changes at low costs.

### 3. Architectural Plan Presentation

The architectural plan presentation plans strategies to how the business/mission requirements will be satisfied. In addition, key technical requirements and constraints are presented with any context diagrams.

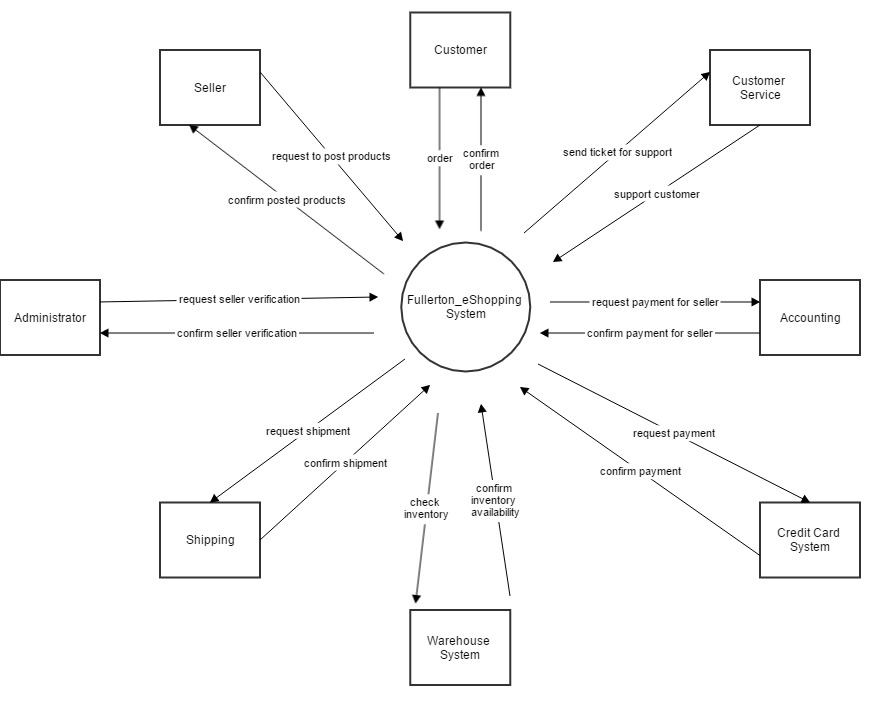
**Key Technical Requirements**

* The eShopping System must be available 99.99% of the time.
* The eShopping System must be secured against attacks.
* The eShopping System shall interface with a credit card payment system for credit card transactions.
* The eShopping System shall meet FTC standards.
* The eShopping System shall be PCI compliant.

**Key Constraints**

* The eShopping System shall be able to be used on any of the major web browsers. (Google Chrome, Mozilla Firefox, and Internet Explorer/Edge)
* The eShopping System shall include mobile applications available for use with IOS or Android mobile devices.
* The eShopping System shall run on Windows, OSX or Linux operating systems

**System Context Diagram**



### 4. Identification of Architectural Drivers

Following steps 2 and 3, the architectural drivers were first identified.

1. Mobile device compatibility
2. Easy to use interface in order to facilitate purchases
3. Increasing customer base
4. Increasing sellers
5. Wide variety of seller product categories available
6. Security - PCI compliant infrastructure
7. Standards - Meet Federal Trade Commission standards
8. High Availability - 99.99%
9. Interface with credit card payment system
10. Performance

Then, these architectural drivers were clarified and additional drivers were added.

1. Portability - Mobile device compatibility
2. Usability - Easy to use interface in order to facilitate purchases
3. Usability - Increasing customer base
4. Usability - Increasing sellers
5. Availability - Wide variety of seller product categories available
6. Security - PCI compliant infrastructure
7. Security - Standards - Meet Federal Trade Commission standards
8. Security - Secure credit card information
9. High Availability - 99.99%
10. Interoperability - Interface with credit card payment system
11. Performance
12. Monitorability
13. Testability
14. Modifiability

### 5. Scenario Brainstorming

The following general brainstorm scenarios were created:

|  |  |
| --- | --- |
| **Scenario #** | **Description** |
| 1 | **Portability - Mobile device compatibility**  Source: End User  Stimulus: Run on multiple platforms  Environment: Normal Operation  Response: Continue to operate |
| 2 | **Usability - Easy to use interface in order to facilitate purchases**  Source: End User  Stimulus: User learns to place an order  Environment: Normal Operations  Response: Provide user with intuitive way to place an order in a shopping cart |
| 3 | **Usability - Increasing customer base**  Source: End User, customers  Stimulus: User makes purchase  Environment: Normal Operations  Response: User friendly purchase interface |
| 4 | **Usability - Increasing Sellers**  Source: End Users, seller  Stimulus: Post items for sale  Environment: Normal Operations  Response: Easy to use intuitive interface for sellers to post items |
| 5 | **Availability - Wide variety of seller product categories available**  Source: End Users, seller  Stimulus: Post items for sale  Environment: Normal Operations  Response: Easy to use intuitive interface for sellers to post items |
| 6 | **Security - PCI compliant infrastructure**  Source: system  Stimulus: Payment is made  Environment: Normal Operations  Response: Payment is secure and meets PCI compliance |
| 7 | **Standards - Meet Federal Trade Commission standards**  Source: end user  Stimulus: entering personal information  Environment: Normal Operations, offline, runtime  Response: data is stored within federal trade commission standards |
| 8 | **High Availability - 99.99%**  Source: ping/echo  Stimulus: server unresponsive  Environment: normal operations  Response: log fault, notify |
| 9 | **Interoperability - Interface with credit card payment system**  Source: eShopping System  Stimulus: Submit Payment  Environment: Known prior to runtime  Response: Information is exchanged, notification is sent if the request is rejected |
| 10 | **Performance**  Source: CPU Monitor  Stimulus: periodic  Environment: Normal Operations  Response: Continues to operate |
| 11 | **Security**  Source: Unauthorized person Stimulus: Attempts to access credit card information Environment: Normal Operations Response: Audit trail |
| 12 | **Monitorability** Source: Operations staff Stimulus: Monitors transaction processing for timing Environment: Normal Operations Response: notify |
| 13 | **Testability** Source: Integration Tester Stimulus: Unit of Code is complete Environment:Integration time Response: Capture results |
| 14 | **Modifiability** Source: End User - Sellers Stimulus: Requests added functionality for new report Environment: Design Time Response: Functionality added and tested |

### 6. Scenario Consolidation

In step 6, scenarios are consolidated if the scenarios were similar in content. Scenario 2-4 as these scenarios all regard usability and having an intuitive user interface. Scenario 5 was removed since increasing the number of product categories will come with increasing the number of sellers. Scenario 6 was merged with Scenario 11 as these scenarios refer to security standards for the system.

|  |  |
| --- | --- |
| **Scenario #** | **Description** |
| 1 | **Portability - Mobile device compatibility**  Source: End User  Stimulus: Run on multiple platforms  Environment: Normal Operation  Response: Continue to operate |
| 2 | **Usability - Easy to use interface in order to facilitate purchases**  Source: End User  Stimulus: User learns to place an order  Environment: Normal Operations  Response: Provide user with intuitive way to place an order in a shopping cart |
| ~~3~~ | **~~Usability - Increasing customer base~~**  ~~Source: End User, customers~~  ~~Stimulus: User makes purchase~~  ~~Environment: Normal Operations~~  ~~Response: User friendly purchase interface~~ |
| ~~4~~ | **~~Usability - Increasing Sellers~~**  ~~Source: End Users, seller~~  ~~Stimulus: Post items for sale~~  ~~Environment: Normal Operations~~  ~~Response: Easy to use intuitive interface for sellers to post items~~ |
| 5 | **Availability - Wide variety of seller product categories available**  Source: End Users, seller  Stimulus: Post items for sale  Environment: Normal Operations  Response: Easy to use intuitive interface for sellers to post items |
| ~~6~~ | **~~Security - PCI compliant infrastructure~~**  ~~Source: system~~  ~~Stimulus: Payment is made~~  ~~Environment: Normal Operations~~  ~~Response: Payment is secure and meets PCI compliance~~ |
| 7 | **Standards - Meet Federal Trade Commission standards**  Source: end user  Stimulus: entering personal information  Environment: Normal Operations, offline, runtime  Response: data is stored within federal trade commission standards |
| 8 | **High Availability - 99.99%**  Source: ping/echo  Stimulus: server unresponsive  Environment: normal operations  Response:log fault, notify |
| 9 | **Interoperability - Interface with credit card payment system**  Source: eShopping System  Stimulus: Submit Payment  Environment: Known prior to runtime  Response: Information is exchanged, notification is sent if the request is rejected |
| 10 | **Performance**  Source: CPU Monitor  Stimulus: periodic  Environment: Normal Operations  Response: Continues to operate |
| 11 | **Security**  Source: Unauthorized person Stimulus: Attempts to access credit card information Environment: Normal Operations Response: Audit trail |
| 12 | **Monitorability** Source: Operations staff Stimulus: Monitors transaction processing for timing Environment: Normal Operations Response: notify |
| 13 | **Testability** Source: Integration Tester Stimulus: Unit of Code is complete Environment:Integration time Response: Capture results |
| 14 | **Modifiability** Source: End User - Sellers Stimulus: Requests added functionality for new report Environment: Design Time Response: Functionality added and tested |

### 7. Scenario Prioritization

The scenarios were prioritized as voted by each stakeholder. Each stakeholder received votes equal to 30% from the total number of scenarios generated. With nine scenarios left and five stakeholders, each stakeholder received 2.7 votes for each round, rounded up to 3, and a total of 15 votes. Half of the stakeholder votes can be allocated per each round.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario #** | **Description** | **Number of Votes**  **Round 1** | **Number of Votes**  **Round 2** | **Priority** |
| 1 | **Portability - Mobile device compatibility**  Source: End User  Stimulus: Run on multiple platforms  Environment: Normal Operation  Response: Continue to operate | 0 | 2 | 5 |
| 2 | **Usability - Easy to use interface in order to facilitate purchases**  Source: End User  Stimulus: User learns to place an order  Environment: Normal Operations  Response: Provide user with intuitive way to place an order in a shopping cart | 0 | 1 | 6 |
| 3 | **High Availability - 99.99%**  Source: ping/echo  Stimulus: server unresponsive  Environment: normal operations  Response:log fault, notify | 3 | 0 | 1 |
| 4 | **Interoperability - Interface with credit card payment system**  Source: eShopping System  Stimulus: Submit Payment  Environment: Known prior to runtime  Response: Information is exchanged, notification is sent if the request is rejected | 2 | 0 | 4 |
| 5 | **Performance**  Source: CPU Monitor  Stimulus: periodic  Environment: Normal Operations  Response: Continues to operate | 2 | 0 | 3 |
| 6 | **Security**  Source: Unauthorized person Stimulus: Attempts to access credit card information Environment: Normal Operations Response: Audit trail | 3 | 0 | 2 |
| 7 | **Monitorability** Source: Operations staff Stimulus: Monitors transaction processing for timing Environment: Normal Operations Response: notify | 0 | 1 | 7 |
| 8 | **Testability** Source: Integration Tester Stimulus: Unit of Code is complete Environment:Integration time Response: Capture results | 0 | 1 | 8 |
| 9 | **Modifiability** Source: End User - Sellers Stimulus: Requests added functionality for new report Environment: Design Time Response: Functionality added and tested | 0 | 0 | 9 |

After voting, the scenarios were prioritized as follows:

1. Availability
2. Security
3. Performance
4. Interoperability
5. Portability
6. Usability
7. Monitorability
8. Testability
9. Modifiability

### 8. Scenario Refinement

The scenarios were further refined with the stimulus, response, source of stimulus, environment, artifact stimulated, response measure. Business/mission goals were identified by the scenario, and stakeholders ask any questions or raise any issues.

1. Availability

|  |  |
| --- | --- |
| **Quality Attribute** | Availability |
| **Business Goals** | The mission of Fullerton eShopping is to offer customers a unique online and mobile shopping experience *anywhere, anytime* and is secure. |
| **Source** | Ping / Echo |
| **Stimulus** | Server Unresponsive |
| **Artifact** | Communication Channels |
| **Environment** | Normal Operations |
| **Response** | Log fault  Fix by switching to Active Redundancy |
| **Response Measure** | Availability 99.99% |
| **Questions** | 1. Are there any SLAs about the availability of the system? 2. Is there a load balancer in place? 3. Is there a hot backup available? 4. What are the maintenance windows? |
| **Issues** | Will need to establish recovery time objective |

2. Security

|  |  |
| --- | --- |
| **Quality Attribute** | Security |
| **Business Goals** | The mission of Fullerton eShopping is to offer customers a unique online and mobile shopping experience anywhere, anytime and is *secure*. |
| **Source** | Unauthorized person |
| **Stimulus** | Attempts to access credit card information |
| **Artifact** | Data within the system |
| **Environment** | Normal Operations |
| **Response** | Audit trail |
| **Response Measure** | Analysis of compromised data is completed within 48 hours |
| **Questions** | 1. Are critical risks to web and mobile application avoided through training? 2. Are account activities being audited? 3. Are roles well defined - what are the roles and permissions? 4. PCI compliance testing 5. Do we have adequate plan for patching software for platform security holes? 6. Are we scanning the attacks? 7. What is the response for an attack scenario?    1. Access control    2. Strong encryption    3. Encryption    4. Procedures for restoring important data after damage |
| **Issues** | 1. Firewall 2. Having security architecture in software 3. (SSL certificates, encrypted data - in the event of an attack important data is not readable) |

3. Performance

|  |  |
| --- | --- |
| **Quality Attribute** | Performance |
| **Business Goals** | We want our customers to be happy with our product that is *efficient, performs well*, and reliable. |
| **Source** | CPU Monitor |
| **Stimulus** | Periodic |
| **Artifact** | CPU |
| **Environment** | Normal Operations |
| **Response** | Continues to operate |
| **Response Measure** | Below 80% usage 99.99% of the time |
| **Questions** | Are the hardware resources sufficient for the load?  What will be the expected peak load time? |
| **Issues** | How to respond when part of a system is performing poorly. |

4. Interoperability

|  |  |
| --- | --- |
| **Quality Attribute** | Interoperability |
| **Business Goals** | We want our customers to be happy with our product that is efficient, performs well, and reliable. |
| **Source** | eShopping System |
| **Stimulus** | Submit Order |
| **Artifact** | Credit Card Payment System |
| **Environment** | Known prior to runtime |
| **Response** | Information is exchanged  Notification is sent if the request is rejected |
| **Response Measure** | Our information included correctly 99.99% of the time |
| **Questions** | Which vendor?  Will credit card information be stored? |
| **Issues** | Must pass PCI compliance audit |

5. Portability

|  |  |
| --- | --- |
| **Quality Attribute** | Portability |
| **Business Goals** | The mission of Fullerton eShopping is to offer customers a unique online and *mobile* shopping experience anywhere, anytime and is secure. |
| **Source** | Customer |
| **Stimulus** | Accesses application on mobile device |
| **Artifact** | System |
| **Environment** | Normal Operation |
| **Response** | Continue to operate |
| **Response Measure** | 99.9% of features work as designed |
| **Questions** | Will we deploy to Windows Phone?  What OS versions will we support?  What browsers and browser versions will be supported? |
| **Issues** | Need to deploy separately for iOS and Android |

6. Usability

|  |  |
| --- | --- |
| **Quality Attribute** | Usability |
| **Business Goals** | We want people to search to be able to *easily* find our seller’s products and get a good discount when ordering in bulk |
| **Source** | Customer |
| **Stimulus** | Learns how to place an order |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | eShopping system provides user with intuitive shopping cart feature for placing an order |
| **Response Measure** | Average first time purchase can be made within 2 minutes once customer is ready for checkout |
| **Questions** | Will we generate our own search engine?  How will items be categorized? |
| **Issues** | Need several iterations of design and feedback to generate an intuitive UI |

7. Monitorability

|  |  |
| --- | --- |
| **Quality Attribute** | Monitorability |
| **Business Goals** | We want our customers to be happy with our product that is *efficient, performs well, and reliable.* |
| **Source** | Operations staff |
| **Stimulus** | Monitors transaction processing for timing |
| **Artifact** | Transaction processing |
| **Environment** | Normal Operations |
| **Response** | Notify |
| **Response Measure** | Detects increase of 20% or more from average transaction processing time and notifies within 1 minute |
| **Questions** | What will be our action if transaction times are increased?  How will the UI respond to a timed out transaction? |
| **Issues** |  |

8. Testability

|  |  |
| --- | --- |
| **Quality Attribute** | Testability |
| **Business Goals** | We want our customers to be happy with our product that is efficient, performs well, and *reliable*. |
| **Source** | Integration Tester |
| **Stimulus** | Unit of Code is complete |
| **Artifact** | Source Code |
| **Environment** | Integration time |
| **Response** | Capture results |
| **Response Measure** | 1-8 hours to complete integration test depending on complexity of change |
| **Questions** | What will be the expected response time for an unsuccessful integration test?  Will integration testing cover most scenarios? |
| **Issues** | Documentation will be necessary for testers to understand code |

9. Modifiability

|  |  |
| --- | --- |
| **Quality Attribute** | Modifiability |
| **Business Goals** | We want our system to keep up with technology and implement new changes at low costs. |
| **Source** | End User - Sellers |
| **Stimulus** | Requests added functionality for new report |
| **Artifact** | User Interface |
| **Environment** | Design Time |
| **Response** | Functionality added and tested |
| **Response Measure** | 2 weeks |
| **Questions** | What is the impact of the modification on factors such as risk and cost? |
| **Issues** | Following design patterns will enable a modifiable architecture |

## Find Tactics

The two top prioritized quality attributes are availability and security. Here are the listed tactics and description for each.

**Availability Tactics**

1. Fault detection: before system takes action regarding a fault, the fault must be anticipated
2. Fault recovery: refined into 2 parts
   1. Preparation-and-repair-tactics: retrying computation or introducing redundancy
   2. Reintroduction tactics: reintroduce failed component back after it has been corrected into normal operation.
3. Fault prevention: prevent faults from initially occurring

**Security Tactics**

1. Detect: finding attacks
   1. Detect intrusion
   2. Detect service denial
   3. Verify message integrity
   4. Detect message delay
2. Resist: preventing and limiting attacks
   1. Identify actors
   2. Authenticate actors
   3. Authorize actors
   4. Limit access
   5. Limit exposure
   6. Encrypt data
   7. Separate entities
   8. Change default settings
3. React: responses to attacks
   1. Revoke access
   2. Lock computer
   3. Inform actors
4. Recover: permit restoration of services
   1. Audit trail

## Quality Attribute Utility Tree

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Quality Attribute** | **Attribute Refinement** | **Priority** | **ASR Scenarios** |
| **Utility** | Availability | Fault Detection | (H,H) | Server responses faults are logged and are fixed by switching to active redundancy. Server availability 99.99%. |
| Interoperability | Payment | (H,M) | User purchases item with credit card and the system sends information to and from banks. Our information included correctly 99.99% of the time |
| Modifiability | Source Code Changes | (L,H) | User submits a request for a change, the change is attempted, tested and released in update. 2 week timeframe |
| Performance | CPU Monitoring | (H,M) | Monitoring system should monitor CPU if usage falls below 80%. |
| Security | Authentication | (H,M) | Unauthorized access should be audited and trailed, completed within 48 hours |
| Testability | QA | (L,L) | Source code changes should be tested fully and documented by the QA team within a 1-8 hour work period |
| Usability | User Awareness | (H,M) | System provides user with intuitive shopping cart feature for placing an order, Average first time purchase can be made within 2 minutes once customer is ready for checkout |
| Portability | Mobile Architecture | (H,M) | Accesses application on mobile device, 99.9% of features work as designed |
| Monitorability | Process Monitoring | (M,L) | Monitors transaction processing for timing. Detects increase of 20% or more from average transaction processing time and notifies within 1 minute |

## Architectural Design Relation Table (ADRT)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Quality Attributes** | **Tactics** | | **Design Patterns** | **Architectural Patterns** | **Implementation Tech & Concerns** |
| Availability | Detect Faults | Ping / Echo | Publish / Subscribe | Client-Server | SSL, TSL, SSL Certification |
|  | Recover from Fault | Active Redundancy | Active Object | Client-Server | High availability groups, synchronous state |
|  | Prevent Faults | Exception Prevention | MVC | Layered | NLog |
| Security | Detect Attacks | Detect Intrusion | Asynchronous Cryptography | Client-Server | SSL certification |
|  | Resist Attacks | Authenticate Actors | Shared Data | Client-Server | ADFS authentication |
|  | React to Attacks | Inform Actor | Shared Data | Client-Server | Windows Server Performance Monitor |
|  | Recover from Attacks | Restore | Shared Data | Layered | VMWare, NetApp |
| Performance | Manage Resources | Introduce Concurrency | Active Object, Proxy | SOA, Client-Server | Load balancing F5 switch |
|  | Manage Resources | Maintain Multiple Copies of Data | Shared Data | Client-Server | Java J2EE Platform |
| Interoperability | Manage Interfaces | Orchestrate | Broker | SOA, Client-Server | REST |
| Portability | Modules | Split Module | MVC | Layered | XCode, Android Studio |

# Exercise 3

## Iteration One

### Step 0

In Step 0, the stakeholders prioritized the requirements according to business and mission goals. These requirements are collected to be the inputs to ADD v.2 when we begin Step 1. After our initial interviews with the stakeholders we have defined the requirements and categorized them as functional requirements, design constraints, and quality attributes.

#### Inputs

##### Functional Requirements

FR1. eShopping System shall allow users to safely purchase items from the store

FR2. eShopping System shall store credit card information and bank account information for both purchases and selling.

FR3. eShopping System shall allow sellers to post items for sale.

FR4. eShopping System shall allow users to sign-up for an account

FR5. eShopping System shall authenticate user and seller accounts via email confirmation and passwording

FR6. eShopping System shall allow users to login/logout of accounts

FR7. eShopping System shall allow users to update profile

FR8. eShopping System shall record accesses to the system via audit trails for potential hackers

##### Design Constraints

DC1. The eShopping System shall include mobile applications available for use with IOS or Android mobile devices.

DC2. The eShopping System shall be able to be used on any of the major web browsers. This team does not see an issue with using non-major web browsers, but issues could possibly arise. Major web browsers include Google Chrome, Mozilla Firefox, and Internet explorer/Edge

DC3. The eShopping System shall be available for Windows Server, or Linux operating systems.

DC4. The eShopping System shall be a web based system, therefore the hardware requirement is for a computer to have the ability to access the internet. This requires a working computer and access to a LAN or WIFI source.

DC5. The system shall be developed using Java as the programming language.

##### Quality Attributes

|  |  |
| --- | --- |
| **Quality Attribute** | Availability |
| **Source** | Ping / Echo |
| **Stimulus** | Server Unresponsive |
| **Artifact** | Communication Channels |
| **Environment** | Normal Operations |
| **Response** | Log fault  Fix by switching to Active Redundancy |
| **Response Measure** | Availability 99.99% |

|  |  |
| --- | --- |
| **Quality Attribute** | Security |
| **Source** | Unauthorized person |
| **Stimulus** | Attempts to access credit card information |
| **Artifact** | Data within the system |
| **Environment** | Normal Operations |
| **Response** | Audit trail |
| **Response Measure** | Analysis of compromised data is completed within 48 hours |

|  |  |
| --- | --- |
| **Quality Attribute** | Performance |
| **Source** | CPU Monitor |
| **Stimulus** | Periodic |
| **Artifact** | CPU |
| **Environment** | Normal Operations |
| **Response** | Continues to operate |
| **Response Measure** | Below 80% usage 99.99% of the time |

|  |  |
| --- | --- |
| **Quality Attribute** | Interoperability |
| **Source** | eShopping System |
| **Stimulus** | Submit Order |
| **Artifact** | Credit Card Payment System |
| **Environment** | Known prior to runtime |
| **Response** | Information is exchanged  Notification is sent if the request is rejected |
| **Response Measure** | Our information included correctly 99.99% of the time |

|  |  |
| --- | --- |
| **Quality Attribute** | Portability |
| **Source** | Customer |
| **Stimulus** | Accesses application on mobile device |
| **Artifact** | System |
| **Environment** | Normal Operation |
| **Response** | Continue to operate |
| **Response Measure** | 99.9% of features work as designed |

|  |  |
| --- | --- |
| **Quality Attribute** | Usability |
| **Source** | Customer |
| **Stimulus** | Learns how to place an order |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | eShopping system provides user with intuitive shopping cart feature for placing an order |
| **Response Measure** | Average first time purchase can be made within 2 minutes once customer is ready for checkout |

|  |  |
| --- | --- |
| **Quality Attribute** | Monitorability |
| **Source** | Operations staff |
| **Stimulus** | Monitors transaction processing for timing |
| **Artifact** | Transaction processing |
| **Environment** | Normal Operations |
| **Response** | notify |
| **Response Measure** | Detects increase of 20% or more from average transaction processing time and notifies within 1 minute |

|  |  |
| --- | --- |
| **Quality Attribute** | Testability |
| **Source** | Integration Tester |
| **Stimulus** | Unit of Code is complete |
| **Artifact** | Source Code |
| **Environment** | Integration time |
| **Response** | Capture results |
| **Response Measure** | 1-8 hours to complete integration test depending on complexity of change |

|  |  |
| --- | --- |
| **Quality Attribute** | Modifiability |
| **Source** | End User - Sellers |
| **Stimulus** | Requests added functionality for new report |
| **Artifact** | User Interface |
| **Environment** | Design Time |
| **Response** | Functionality added and tested |
| **Response Measure** | 2 weeks |

### Step 1: Confirm that there is sufficient requirements information

The primary stakeholders and architects reviewed the requirements and determined that the requirements were complete. All the requirements have quality attribute scenarios. The requirements are sufficient to start the first iteration of architectural design and were prioritized.

### Step 2: Choose an element of the system to decompose

Since this is the first iteration, the entire system will be decomposed.

### Step 3: Identify candidate architectural drivers

To identify candidate architecture drivers, we added the prioritized quality attributes that we captured from the QAW workshop to a table in order to rank them in addition to the functional requirements and design constraints. This gave us a list of requirements that affect system. For each requirement, we indicated H, M, L (High, Medium, Low) to indicate the importance to the stakeholders and their potential impact on the architecture. The priority is ranked again by the impact on the architecture. We created the **Table: Quality Attribute Utility Tree** to rank the requirements based on the quality attributes and then prioritized the top 5 drivers in the **Architecture Driver Priorities** table. The Architecture Driver Priorities table represents the top five candidate architectural drivers.

**Table: Architectural Driver Importance to Stakeholders and Impact on Architecture**

|  |  |  |
| --- | --- | --- |
| **Architectural Driver** | **Importance to Stakeholders** | **Impact on Architecture** |
| FR1. eShopping System shall allow users to safely purchase items from the store | H | H |
| FR2. eShopping System shall store credit card information and bank account information for both purchases and selling. | H | M |
| FR3. eShopping System shall allow sellers to post items for sale. | H | H |
| FR4. eShopping System shall allow users to sign-up for an account | H | L |
| FR5. eShopping System shall authenticate user and seller accounts via email confirmation and passwording | H | M |
| FR6. eShopping System shall allow users to login/logout of accounts | H | M |
| FR7. eShopping System shall allow users to update profile | L | M |
| FR8. eShopping System shall record accesses to the system via audit trails for potential hackers | H | M |
| DC1. The eShopping System shall include mobile applications available for use with IOS or Android mobile devices. | M | H |
| DC2. The eShopping System shall be able to be used on any of the major web browsers. This team does not see an issue with using non-major web browsers, but issues could possibly arise. Major web browsers include Google Chrome, Mozilla Firefox, and Internet explorer/Edge | H | M |
| DC3. The eShopping System shall be available for Windows Server, or Linux operating systems. | H | H |
| DC4. The eShopping System shall be a web-based system, therefore the hardware requirement is for a computer to have the ability to access the internet. This requires a working computer and access to a LAN or WI-FI source. | H | M |
| DC5. The system shall be developed using Java as the programming language. | M | L |
| Scenario 1 Availability, Server responses faults are logged and are fixed by switching to active redundancy. Server availability 99.99%. | H | H |
| Scenario 2 Security, Unauthorized access should be audited and trailed, completed within 48 hours. | H | M |
| Scenario 3 Performance, Monitoring system should monitor CPU if usage falls below 80%. | H | M |
| Scenario 4 Interoperability, User purchases item with credit card and the system sends information to and from banks. Our information included correctly 99.99% of the time. | H | M |
| Scenario 5 Portability, 99.9% of features work as designed when accessing application on mobile device | H | M |
| Scenario 6 Usability, System provides user with intuitive shopping cart feature for placing an order, Average first time purchase can be made within 2 minutes once customer is ready for checkout. | H | M |
| Scenario 7 Monitorability, Monitors transaction processing for timing. Detects increase of 20% or more from average transaction processing time and notifies within 1 minute. | M | L |
| Scenario 8 Testability, Source code changes should be tested fully and documented by the QA team within a 1-8 hour work period. | L | L |
| Scenario 9 Modifiability, User submits a request for a change, the change is attempted, tested and released in update. 2 week timeframe | L | H |

#### Table: Quality Attribute Utility Tree

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Quality Attribute** | **Attribute Refinement** | **Priority** | **ASR Scenarios** |
| **Utility** | Availability | Fault Detection | (H,H) | Server response faults are logged and are fixed by switching to active redundancy. Server availability 99.99%. |
| Interoperability | Payment | (H,M) | User purchases item with credit card and the system sends information to and from banks. Our information included correctly 99.99% of the time |
| Modifiability | Source Code Changes | (L,H) | User submits a request for a change, the change is attempted, tested and released in update. 2 week timeframe |
| Performance | CPU Monitoring | (H,M) | Monitoring system should monitor CPU if usage falls below 80%. |
| Security | Authentication | (H,M) | Unauthorized access should be audited and trailed, completed within 48 hours |
| Testability | QA | (L,L) | Source code changes should be tested fully and documented by the QA team within a 1-8 hour work period |
| Usability | User Awareness | (H,M) | System provides user with intuitive shopping cart feature for placing an order, Average first time purchase can be made within 2 minutes once customer is ready for checkout |
| Portability | Mobile Architecture | (H,M) | Accesses application on mobile device, 99.9% of features work as designed |
| Monitorability | Process Monitoring | (M,L) | Monitors transaction processing for timing. Detects increase of 20% or more from average transaction processing time and notifies within 1 minute |

3. Below are the five high-priority requirements that were chosen as the candidate architectural drivers.

#### Table: Architecture Driver Priorities

|  |  |  |
| --- | --- | --- |
| **Architectural Driver** | **Importance to Stakeholders** | **Impact on Architecture** |
| Scenario 1: Availability, 99.99% | H | H |
| Scenario 2: Security, Analysis of compromised data is completed within 48 hours | H | M |
| Scenario 3: Performance, Below 80% usage 99.99% of the time | H | M |
| Scenario 4: Interoperability, When submitting orders to the Credit Card Payment system our information shall be included correctly 99.99% of the time | H | M |
| Scenario 5: Portability, 99.9% of features work as designed when accessing application on mobile device | H | M |

### Step 4 (Design Step): Choose a design concept that satisfies the architectural drivers.

The goal of Step 4 is to choose the major architectural elements and the relationships among them. By creating a matrix of the Pros and Cons of the patterns associated with the drivers in the ADRT, we selected the patterns and tactics that could be utilized to satisfy the architectural drivers.

1. Design concerns of the candidate architectural drivers were identified.
2. Then, alternative patterns were identified to address each concern, and the value was estimated for the patterns’ discriminating parameters.
3. Patterns were selected to satisfy the candidate architectural drivers
4. Examined the relationships between the selected patterns.
5. Describe the selected patterns and architectural views
6. Evaluate and resolve inconsistencies

**Table: Architectural Design Relation Table (ADRT)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Quality Attributes** | **Tactics** | | **Design Patterns** | **Architectural Patterns** | **Implementation Tech & Concerns** |
| Availability | Detect Faults | Ping / Echo | Publish/Subscribe | Client-Server | SSL, TSL, SSL Certification |
| Recover from Fault | Active Redundancy | Active Object | Client-Server | High availability groups, synchronous state |
| Security | Detect Attacks | Detect Intrusion | Asynchronous Cryptography | Client-Server | SSL certification |
| Resist Attacks | Authenticate Actors | Shared Data | Client-Server | ADFS authentication |
| React to Attacks | Inform Actor | Shared Data | Client-Server | Windows Server Performance Monitor |
| Recover from Attacks | Restore | Shared Data | Layered | VMWare, NetApp |
| Performance | Manage Resources | Introduce Concurrency | Active Object, Proxy | SOA, Client-Server | Load balancing F5 switch |
| Maintain Multiple Copies of Data | Shared Data | Client-Server | Java J2EE Platform |
| Interoperability | Manage Interfaces | Orchestrate | Broker | SOA, Client-Server | REST |
| Portability | Cross Platform | Multi-platform | MVC | Layered | XCode, Android Studio |

**Table: Discriminating Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Quality Attribute** | **Pattern Selected** | **Discriminating parameters** | **Values for Discriminating Parameters** |
| **Availability** | Ping / Echo | Amount of time for response to server nodes | 100ms |
| Active Redundancy (hot spare) | Time to failover to spare server node | 1 second |
| **Security** | Detect Intrusion | Known malicious behaviors | Amount of false positives |
| Authenticate Actors | Password strength | Strong password (combination of characters and length of password) |
| Inform Actor | Amount of time to inform the actor | Less than 30 minutes |
| Restore | Amount of time to restore data or system component | 2 hours after event |
| **Performance** | Introduce Concurrency | Reduced blocked time | 100 ms |
| Maintain Multiple Copies of Data | Access speeds | 5 ms read \ write |
| **Interoperability** | Locate | Discovering service | Ability to connect to service through known location points |
| **Portability** | Multi-Platform | Amount of developers for different platforms | 5 developers |

**Rationale**

The Ping/Echo and Active Redundancy tactics for availability are ideal for rapid detection and failing over for maximum availability. The system is expected to be running 24/7 and it is important be aware of and react to issues that will prevent us from meeting our availability target.

The Detection Intrusion, Authenticate Actors, Inform Actor, and Restore are tactics for security we chosen to detect and recover from failure events and be able to restore it to the previous state. These tactics cover all areas in the security general scenario because of the high risks of security breaches in all internet connected software systems.

The Introduce Concurrency, and Maintain Multiple Copies of Data tactics for performance will prepare our system for handling the load during peak usage periods.

The Locate tactic for Interoperability was used because the Credit Card service is a known location point and is the simplest approach to use.

The Multi-Platform tactic for Portability was chosen to partition the development for different platforms among 5 available developers: Android development and iOS development.

**Table: Matrix to Evaluate Candidate Patterns**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Client-Server** | | **Layered** | | **SOA** | |
|  | **Pro** | **Con** | **Pro** | **Con** | **Pro** | **Con** |
| **Availability** | Centralized control of resources and services | Single point of failure |  |  | Asynchronous messaging, dynamic reconfiguration leads to no system interruption | ESB adds additional point of failure |
| **Security** | Centralized control of resources and services |  |  |  | Enterprise service bus |  |
| **Performance** | Multiple distributed servers | Centralized control of resources and services, performance bottleneck |  | Layering adds performance penalty to a system by adding overhead |  | ESB lowers performance. SOA creates performance overhead on middleware |
| **Interoperability** | Centralized the control of resources and services |  |  |  | Orchestration server, service providers and consumers run on different platforms. SOA integrates different types of systems. SOA allows interaction with external services. |  |
| **Portability** |  |  | Incremental development,  Implementing systems on top of virtual machines | Strict ordering relation, layer bridging | Service provider and consumer components can use different implementation languages and platforms |  |

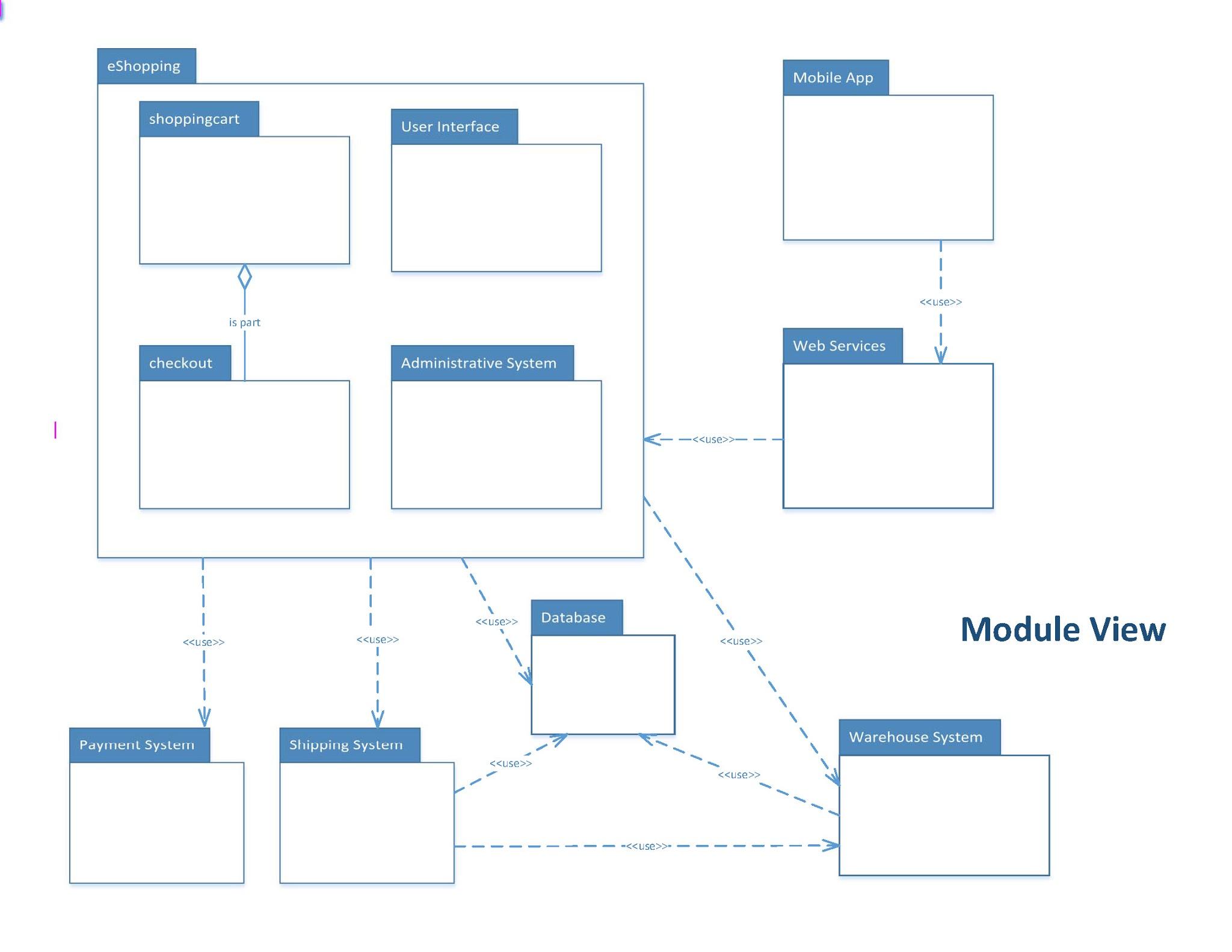
Our overall architectural design will use Client-Server architecture to support both web and mobile clients communicating over HTTP to application servers.

### Step 5 (Design Step): Instantiate architectural elements and allocate responsibilities.

To design the architecture for the eShopping system, we analyzed the architecture using three views: the Module view to decompose the child components, the Component-and-Connector view to represent the behaviors and interfaces of the child components, and the Allocation view to outline the behavior of the non-software elements of the system.

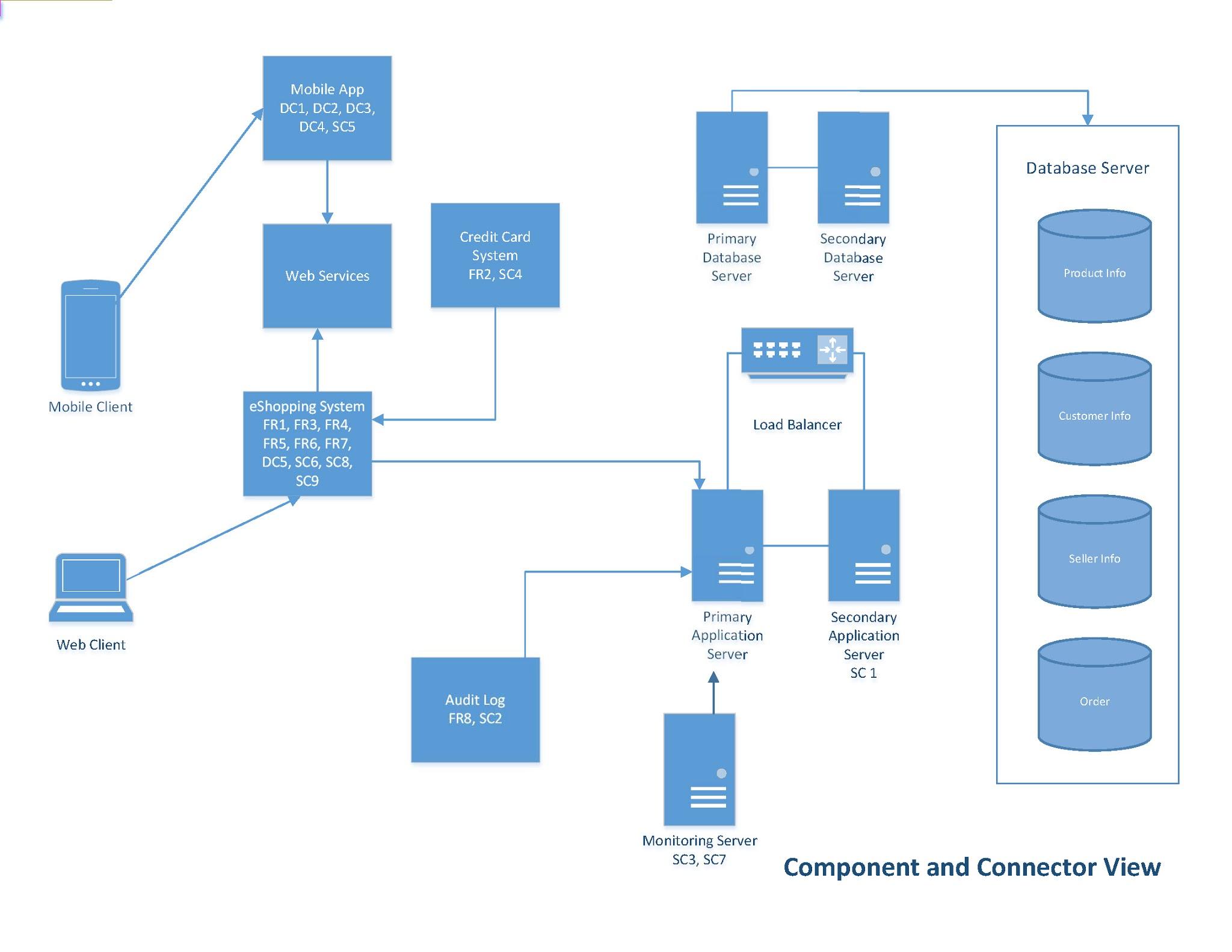
#### Module View

The module view is the composite view consisting of the elements of the system and their relations and describes the high level abstraction of the entire system.

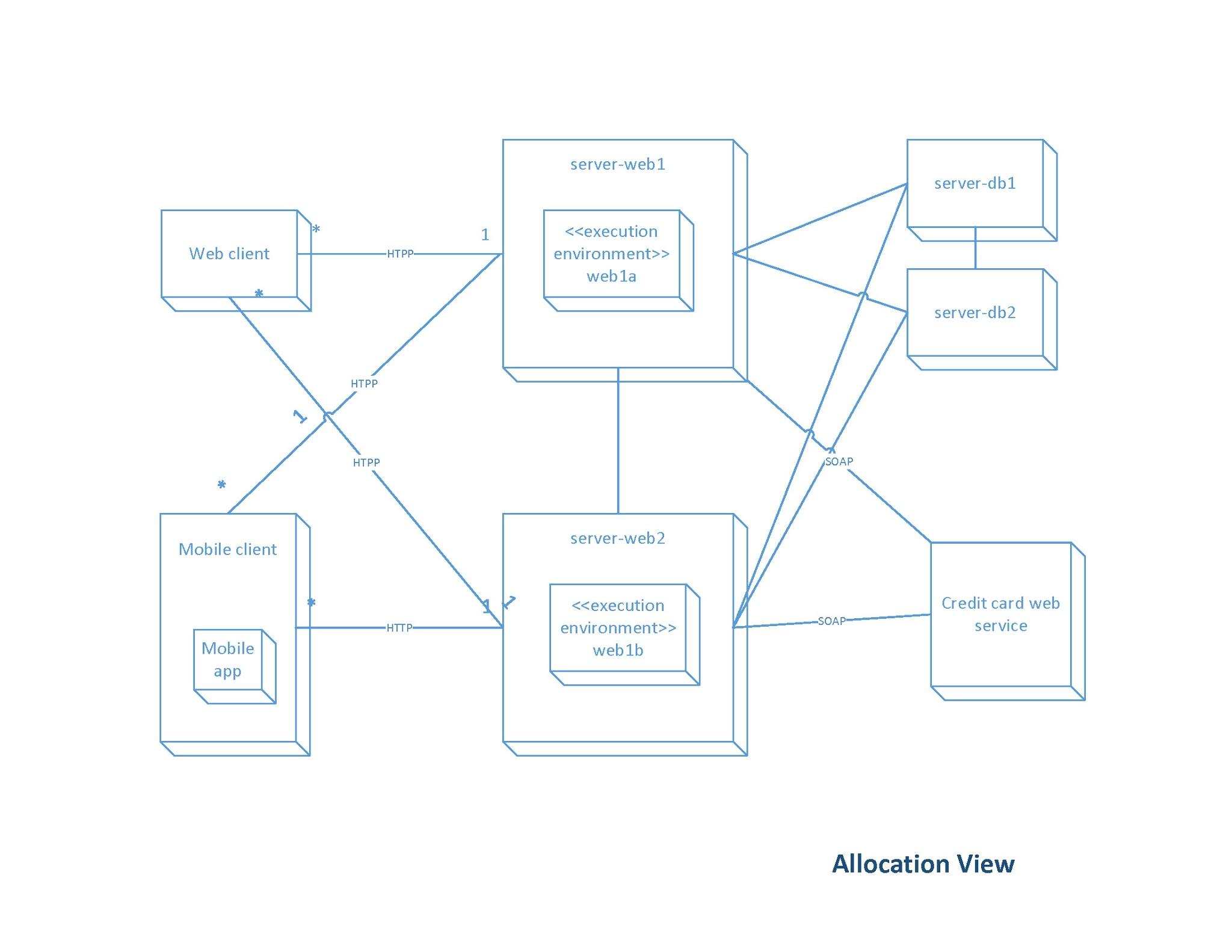


#### Component-and-Connector View

In the component and connector view diagram below, the runtime behavior of the components and the interaction are presented.



#### Allocation View

The deployment view outlines how the software elements relate to the non-software structure in its environment. 

### Step 6 (Design Step): Define interfaces for instantiated elements.

|  |  |
| --- | --- |
| Interface | Make Payment |
| From Element | eShopping System |
| To Element | Payment System |
| Information Exchanged | Transaction is sent with payment information to the credit card system. |
| Response | Acknowledgement is expected |
| Error Handling | Declined transaction error message, add to error queue |

|  |  |
| --- | --- |
| Interface | Mobile Services |
| From Element | Mobile App |
| To Element | Web Services |
| Information Exchanged | Bidirectional |
| Response | none |
| Error Handling | Error log |

|  |  |
| --- | --- |
| Interface | Update Inventory |
| From Element | Shipping System |
| To Element | eShopping System |
| Information Exchanged | Submitted Orders |
| Response | Update inventory of items |
| Error Handling | Error log |

|  |  |
| --- | --- |
| Interface | Fulfill Order |
| From Element | Shipping system |
| To Element | Warehouse System |
| Information Exchanged | Fulfilled Orders |
| Response | Update inventory |
| Error Handling | Error log |

### Step 7: Verify and refine requirements and make them constraints for instantiated elements.

For iteration one, the broad system was designed using functional requirements, design constraints, and quality attributes for which were selected by the stakeholders and the designers.

The functional requirements (as listed above) were refined during step 6 as the elements chosen were chosen instantiated. The mobile application and payments system allows for the requirements such as user interfaces and account creation or management. The mobile application also satisfies the design constraints for which the application would be cross platform and essentially accessible from all means by the user. A web only application is also designed and therefore satisfies the last of the design constraints. The payment system designed satisfies the last of the requirement as it allows users to actually purchase the product they desire.

## Iteration Two

### Step 2: Choose an element of the system to decompose

For iteration two, the team chose **FR1. The eShopping System shall allow users to safely purchase items from the store**, as the element to decompose. This element was selected based on business criteria as it is the highest functional requirement for the majority of the stakeholders.

### Step 3: Identify candidate architectural drivers

For the selected driver, we indicated H, M, L to indicate the importance to the stakeholders and their potential impact on the architecture. The priority is ranked again by the impact on the architecture. We created the **Table: Quality Attribute Utility Tree** to rank the architectural drivers based on the quality attributes and then prioritized the top 5 drivers in the **Architecture Driver Priorities** table.

#### Table: Architecture Driver Priorities

|  |  |  |
| --- | --- | --- |
| **Architectural Driver** | **Importance to Stakeholders** | **Impact on Architecture** |
| Scenario 1 Availability, 99.99% | H | H |
| Scenario 2 Security, Analysis of compromised data is completed within 48 hours | H | M |
| Scenario 3 Performance, Below 80% usage 99.99% of the time | H | M |
| Scenario 4 Interoperability, When submitting orders to the Credit Card Payment system our information shall be included correctly 99.99% of the time | H | M |
| Scenario 5 Portability, 99.9% of features work as designed when accessing application on mobile device | H | M |

### Step 4 (Design Step): Choose a design concept that satisfies the architectural drivers.

The goal of Step 4 is to choose the major architectural elements and the relationships among them. By creating a matrix of the Pros and Cons of the patterns associated with the drivers in the ADRT, we selected the patterns and tactics that could be utilized to satisfy the architectural drivers.

FR1. eShopping System shall allow users to safely purchase items from the store, concerns the following quality attributes:

**Table: Architectural Design Relation Table (ADRT)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Quality Attributes** | **Tactics** | | **Design Patterns** | **Architectural Patterns** | **Implementation Tech & Concerns** |
| Availability | Detect Faults | Ping / Echo | Publish/Subscribe | Client-Server | SSL, TSL, SSL Certification |
| Recover from Fault | Active Redundancy | Active Object | Client-Server | High availability groups, synchronous state |
| Security | Detect Attacks | Detect Intrusion | Asynchronous Cryptography | Client-Server | SSL certification |
| Resist Attacks | Authenticate Actors | Shared Data | Client-Server | ADFS authentication |
| React to Attacks | Inform Actor | Shared Data | Client-Server | Windows Server Performance Monitor |
| Recover from Attacks | Restore | Shared Data | Layered | VMWare, NetApp |
| Performance | Manage Resources | Introduce Concurrency | Active Object, Proxy | SOA, Client-Server | Load balancing F5 switch |
| Maintain Multiple Copies of Data | Shared Data | Client-Server | Java J2EE Platform |
| Interoperability | Manage Interfaces | Orchestrate | Broker | SOA, Client-Server | REST |
| Portability | Cross Platform | Multi-platform | MVC | Layered | XCode, Android Studio |

**Table: Discriminating Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Quality Attribute** | **Pattern Selected** | **Discriminating parameters** | **Values for Discriminating Parameters** |
| **Availability** | Ping / Echo | Amount of time for response to server nodes | 100ms |
| Active Redundancy (hot spare) | Time to failover to spare server node | 1 second |
| **Security** | Detect Intrusion | Known malicious behaviors | Amount of false positives |
| Authenticate Actors | Password strength | Strong password (combination of characters and length of password) |
| Inform Actor | Amount of time to inform the actor | Less than 30 minutes |
| Restore | Amount of time to restore data or system component | 2 hours after event |
| **Performance** | Introduce Concurrency | Reduced blocked time | 100 ms |
| Maintain Multiple Copies of Data | Access speeds | 5 ms read \ write |
| **Interoperability** | Locate | Discovering service | Ability to connect to service through known location points |
| **Portability** | Multi-Platform | Amount of developers for different platforms | 5 developers |

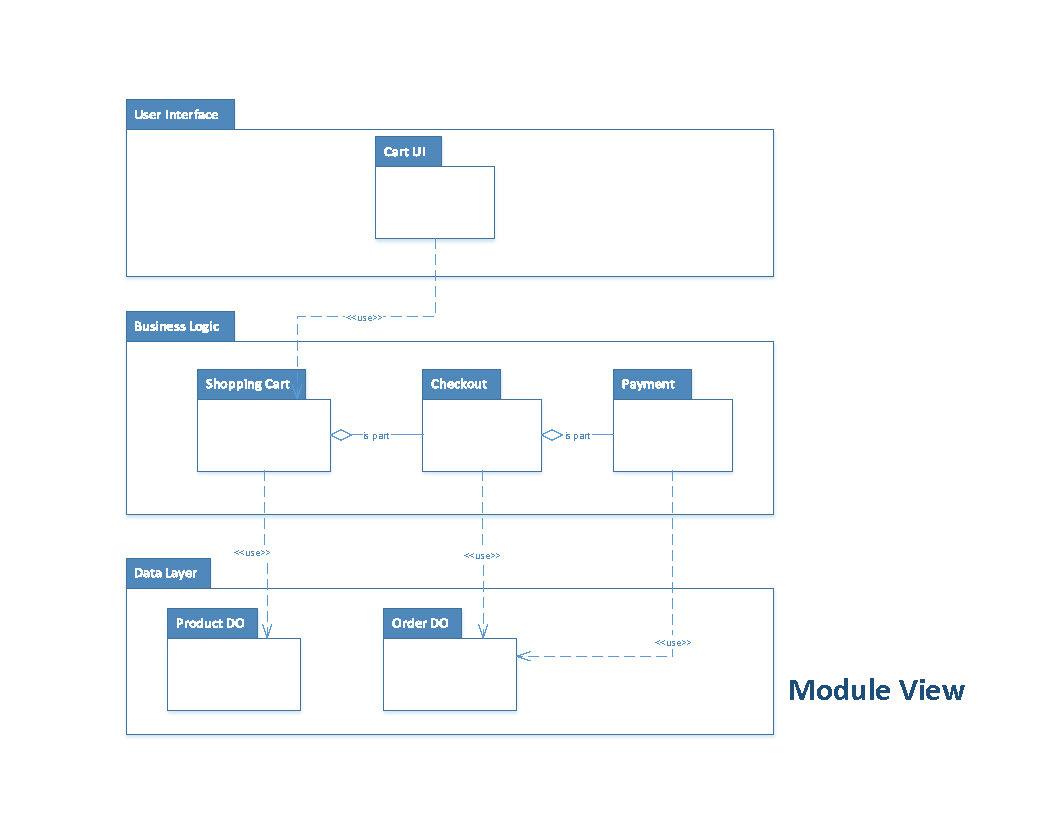
**Table: Matrix to Evaluate Candidate Patterns**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Client-Server** | | **Layered** | | **SOA** | |
|  | **Pro** | **Con** | **Pro** | **Con** | **Pro** | **Con** |
| **Availability** | Centralized control of resources and services | Single point of failure |  |  | Asynchronous messaging, dynamic reconfiguration leads to no system interruption | ESB adds additional point of failure |
| **Security** | Centralized control of resources and services |  |  |  | Enterprise service bus |  |
| **Performance** | Multiple distributed servers | Centralized control of resources and services, performance bottleneck |  | Layering adds performance penalty to a system by adding overhead |  | ESB lowers performance. SOA creates performance overhead on middleware |
| **Interoperability** | Centralized the control of resources and services |  |  |  | Orchestration server, service providers and consumers run on different platforms. SOA integrates different types of systems. SOA allows interaction with external services. |  |
| **Portability** |  |  | Incremental development,  Implementing systems on top of virtual machines | Strict ordering relation, layer bridging | Service provider and consumer components can use different implementation languages and platforms |  |

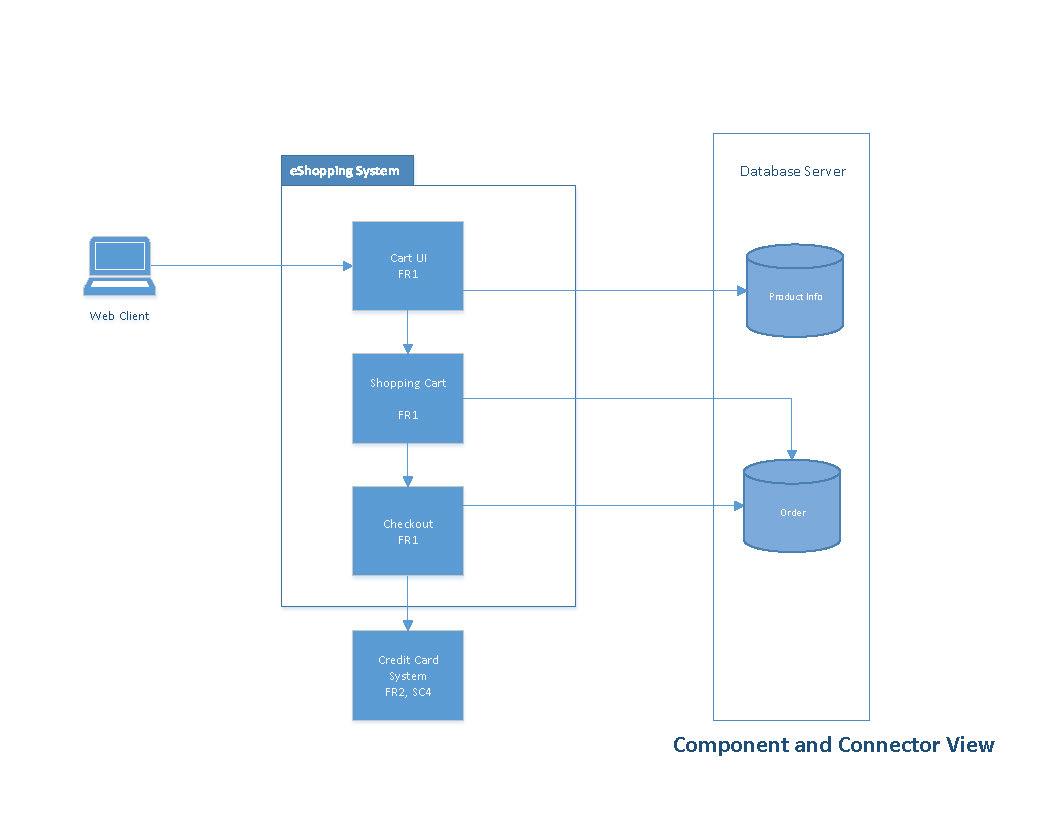
### Step 5 (Design Step): Instantiate architectural elements and allocate responsibilities.

To design the architecture for the eShopping system, we analyzed the architecture using three views: the Module view to decompose the child components, the Component-and-Connector view to represent the behaviors and interfaces of the child components, and the Allocation view to outline the behavior of the non-software elements of the system.

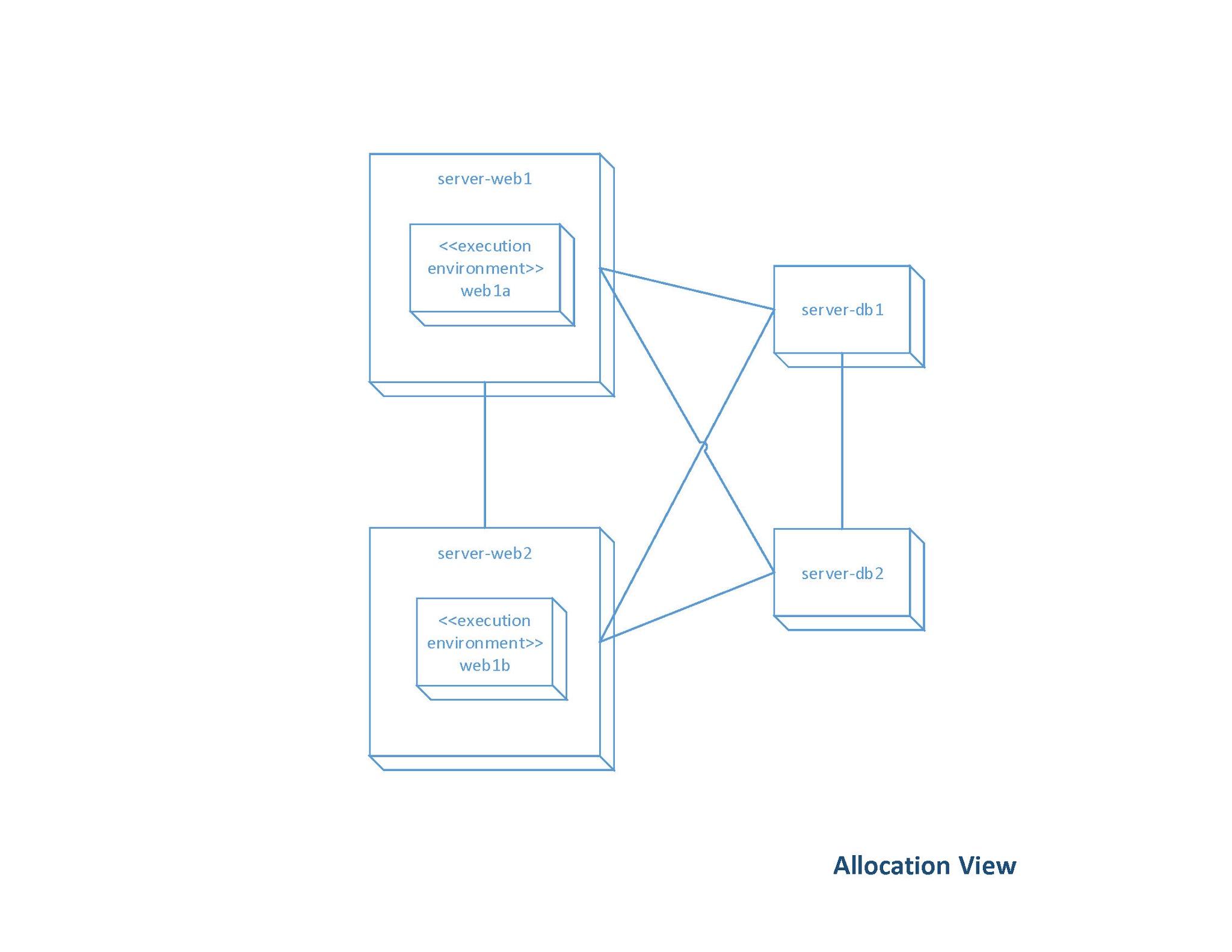
#### Module View



#### Component and Connector View



#### Allocation View



### Step 6 (Design Step): Define interfaces for instantiated elements.

|  |  |
| --- | --- |
| Interface | Update Cart |
| From Element | Cart UI - User Interface Layer |
| To Element | Shopping Cart - Business Logic Layer |
| Information Exchanged | Product Items with Quantity Ordered |
| Response | Shopping Cart Items updated |

|  |  |
| --- | --- |
| Interface | Checkout |
| From Element | Shopping Cart - Business Logic Layer |
| To Element | Checkout - Business Logic Layer |
| Information Exchanged | Product Items with Quantity |
| Response | Order updated |

|  |  |
| --- | --- |
| Interface | Get Products |
| From Element | Shopping Cart - Business Logic Layer |
| To Element | Product DO - Data Object Layer |
| Information Exchanged | Product Items |
| Response | Product Details selected |

|  |  |
| --- | --- |
| Interface | Update Order |
| From Element | Check Out - Business Logic Layer |
| To Element | Order DO - Data Layer |
| Information Exchanged | List of items |
| Response | Update Database |

|  |  |
| --- | --- |
| Interface | Payment |
| From Element | Check Out - Business Logic Layer |
| To Element | Order DO - Data Layer |
| Information Exchanged | Completed order after payment made |
| Response | Update Database |

### Step 7: Verify and refine requirements and make them constraints for instantiated elements.

For the second iteration, the architecture for the “shopping cart” was designed. The same steps for which iteration one took place, iteration two looks at a smaller area of the system and decomposes an element fully as opposed to the general system. This decomposition of the element still must satisfy the quality attributes, for which this design does.

The shopping cart would be available for all systems, as dictated by the first iteration, and will be available via an application. The performance of the shopping cart would adhere to the performance of the system, for which was dictated in iteration one. The security of the products placed into the shopping cart would inherit the security of the system which was also dictated in iteration one. The two quality attributes that were inherently designed by iteration 2 was interoperability and portability. The shopping cart would have a checkout feature for which the shopping cart would allow for the payment of the product. With this the call to the payment system, developed in a different iteration, a connection to an external system would be needed, thus satisfying this attribute. The shopping cart feature would also have to be added to every rendition of the system, this includes the mobile app. With the mobile app created in other iterations, the shopping cart feature would allow for the user to select an item to be stored in their account as well as pay for the product from any mobile device, thus also satisfying this attribute.

## ADD Rationale

**Prioritize the requirements** - The stakeholders gathered together and prioritized which requirements needed to be included in the system and in what order the requirements need to be fulfilled. Requirements consisted of functional requirement, design constraints and quality attributes. Overall, the highest priority for the eShopping System revolved around the system’s availability and security.

**Confirm there is sufficient requirements information** - Once the stakeholder’s completed the list of prioritized requirements, the designers then reviewed the requirements to determine which requirements will impact the systems architecture the most. This allows for the designers to understand what the stakeholder’s wanted in the software and how the requirements the stakeholder’s desire will impact the system as a whole.The highest priority feature is FE1, eShopping System shall allow users to safely purchase items from the store.

**Choose an element of the system to decompose** - During the first iteration, the element chosen was the entire system. For the second iteration, the shopping cart functional requirement was selected as the element to decompose as it is the highest priority requirement as prioritized by the stakeholders.

**Identify candidate architectural drivers** - Once the element was selected, the candidate architectural drivers were selected. The requirements were ranked by the stakeholders for significance and then once more by the designers. The second ranking was determined by potential impact to the system. H, M, L (High, Medium, Low) were the values used for ranking. Architectural drivers for each iteration are listed within the corresponding architectural drivers table.

**Choose a design concept that satisfies the architectural drivers** - During the first iteration design concepts were selected that satisfied the identified architectural drivers. During the second iteration the design concepts for the overlapping architectural drivers were reviewed and further refined. Tactics were selected carefully and reviewed for impact as an addition of one tactic for one quality attribute may negatively impact another quality attribute.

Tactics selected for availability include Ping/Echo and active redundancy. The system is required to be have 99.99% availability. Tactics for security include Detect Intrusion, Authenticate Actors, Inform Actor and Restore.

**Instantiate architectural elements and allocate responsibilities** - Once the tactics were selected, the elements were instantiated and the elements were assigned responsibilities. In both the first and the second iteration the architecture was captured using three views; Model View, Component and Connector View and Allocation View.

**Define interfaces for instantiated elements** - Once the elements were instantiated we used the Module view, component and connector view as well as the allocation view to define interfaces.

**Verify and Refine Requirements -** At the end of each iteration we reviewed the decomposed element to ensure that functional requirements, quality attribute requirements and design constraints were met. Requirements were refined where necessary.

## Software Architecture Documentation

### Stakeholder View Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Stakeholder** | **Module View 1** | **Module View 2** | **C &C View 1** | **C & C View 2** | **Allocation View 1** | **Allocation View 2** |
| 1 | Developers | H | H | H | H | M | M |
| 2 | Testers | H | H | H | H | M | M |
| 3 | User | N | N | O | O | N | N |
| 4 | Management | M | M | M | M | M | M |
| 5 | Executives | O | O | O | O | O | O |

N = None

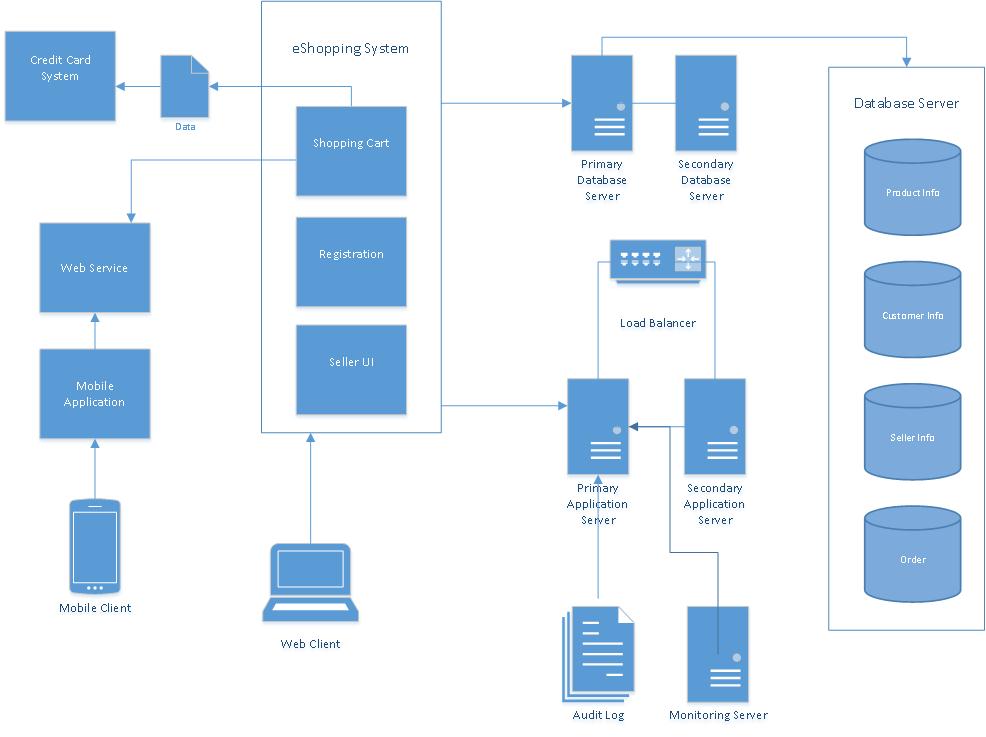
O= Overview Only

M = Moderate Detail

H = High Detail

### Document Package

#### Section 1. Primary Presentation



Key



#### Section 2. The Element Catalog

##### Section 2.A. Elements and Their Properties

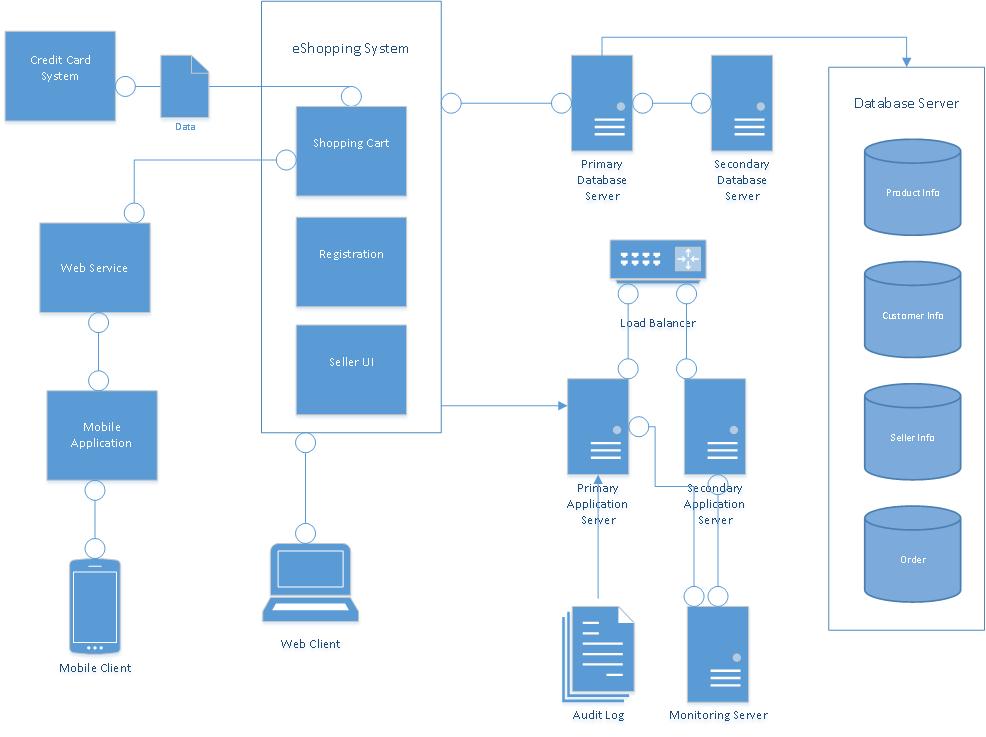
1. **Audit Log**The audit log contains a log of events documenting system access.
2. **Credit Card System**An external system to the eShopping Application.
3. **eShopping System**Main application for the eShopping System.
4. **Load Balancer**Device that distributes application traffic across a number of servers.
5. **Mobile Application**  
   Application designed for mobile device.
6. **Mobile Client**  
   Mobile device used to access the eShopping System through the mobile application.
7. **Monitoring Server**A server that contains an application that will monitor the usage of the primary application server.
8. **Primary Application Server**  
   Primary web server hosting the eShopping application
9. **Primary Database Server**  
   Primary database server containing multiple databases and has read/write access.
10. **Secondary Application Server**Secondary web server hosting the eShopping application
11. **Secondary Database Server**  
    Redundant database server for failover when primary is down.
12. **Web Client**  
    Client browser application UI
13. **Web Service**REST application service for external mobile clients.

##### Section 2.B. Relation and Their Properties

User Session is not depicted. Upon login, a session cookie is created on the client browser. The user session has a 20 minute timeout.

Secure authorization token is created upon login from mobile application login to REST service and is saved to the mobile device. The authorization token expires 1 hour from login time.

##### Section 2.C. Element Interfaces



Key



**Credit Card System**

The eShopping System interfaces with the Credit Card System upon completion of an order.

**Load Balancer**

The load balancer communicates with both the primary application server and the secondary application server in order to direct traffic.

**Mobile Application**

The mobile application interfaces with the web services in order to place orders and checkout and purchase the completed orders the eShopping System.

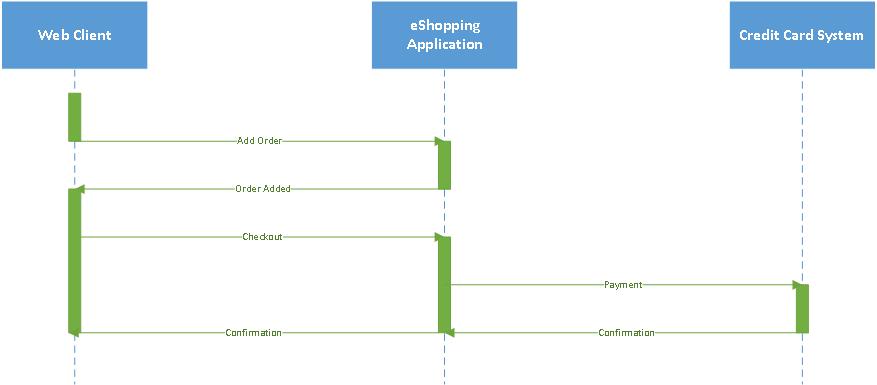
**Monitoring Server**

The Monitoring Server contains an application that communicates with both the primary and secondary application servers in order to monitor CPU Usage.

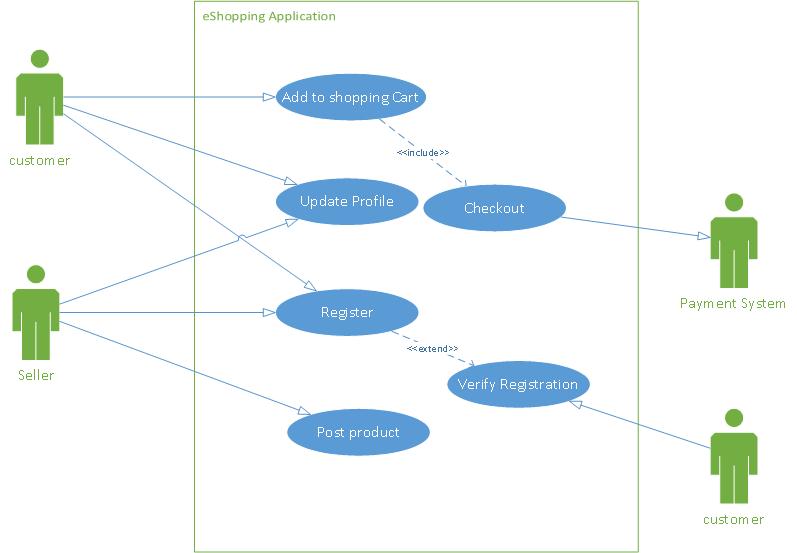
##### Section 2.D. Element Behavior

###### Sequence Diagram

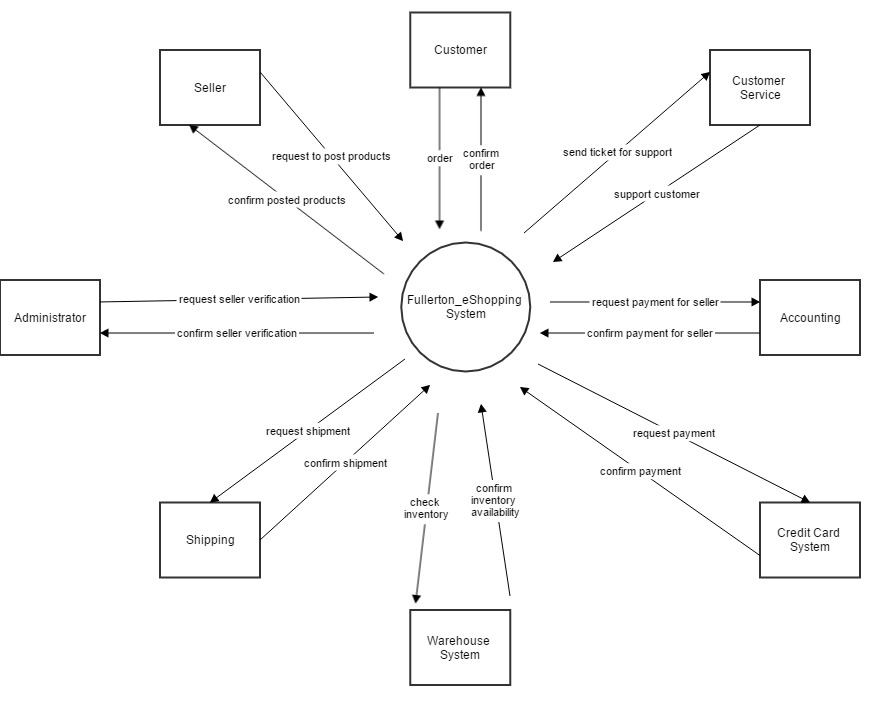
Client connecting to eShopping application for placing an order and processing a payment transaction.



###### Use-Case Diagram



#### Section 3. Context Diagram



#### Section 4. Variability Guide

Depending on volume additional application servers may be introduced to the system and configured with the load balancer.

The eShopping System can be interfaced with various payment methods based on different vendors.

.

#### Section 5. Rationale

This documentation encapsulates the architecture for the Fullerton eShopping System. In this document package, we have the primary view, the element catalog, context diagram, and variability guide.

The primary view captures the elements in the system that address the highest priority architectural concerns which are availability, security, and performance. These quality attributes were chosen as the architectural drivers as they were selected by the stakeholders by using the Quality Attribute Workshop. In addition, the architectural views were designed as a result of performing two iterations of Attribute-Driven Design version 2.0 in which the whole system was decomposed in the first iteration and the shopping cart was decomposed in the second iteration.

From ADD v.2, the module view (not shown in documentation package), component and connector view, and allocation view (not shown in documentation package) were designed. The component and connector view shows the client/server architectural pattern and show runtime system quality attributes. In our component and connector view, the runtime system quality attributes include performance and availability. This pattern shows where the client makes the request to the server. The module view, which partitions the application into groups, shows the layered architectural pattern, and satisfying the portability quality attribute. The allocation view structures the deployment and satisfies the performance, security, and availability quality attributes.

In the element catalog, the elements and element properties were defined. The elements were defined from the primary view and explain the different parts for the Fullerton eShopping System. The relationships and element interfaces display the interactions between the elements, and the element behavior is also shown through the sequence diagram and use-case diagram.

The context diagram shows the scope of the view.

The variability guide details in any variation points or alternatives regarding the architecture. In the Fullerton eShopping System, variability can be found two areas: 1) in the hardware / web servers with a dependability on the load balancers and the credit card payment system which will direct payment according to credit card types and authorizations.

# Lessons Learned

## Timothy Cioffi-Dinkel

This project allowed me to get an in depth look and allow me to practice with a real architectural design method. It taught me how to structure large scale applications and how to design in pieces when the system I am working on is larger than a snippet of code for which I currently work on. ADD and the QAW showed me how an architect receives requirements from stakeholders, and how the prioritization of these requirements is formulated. From these priorities it was easier to dissect the system into parts for which individual design could take place through the iterative process for which ADD is built upon. Lastly, ADD showed me the importance of verification of the satisfaction of requirements before moving onto other areas of the project, thus ensuring a completed piece which provides the greatest satisfactions for those who place the greatest risk into its development. Overall, learning this will benefit me in the future if I land a position as an architect because I will have the tools needed to design a great software, and until then, I have a greater understanding of the work the head engineers do to allow me to provide more input in future meetings.

## Anthony Farina

This exercise has taught me a great deal about software architecture design and practice. For example, throughout this process domains I have learned how to go from beginning to end in software architecture design. The QAW taught me how to define, prioritize, and refine quality attributes. I learned how important quality attributes can be to a software architecture. I have also learned how to perform ADD v2 which I believe will be very important in my professional career since it consisted of several important topics such as patterns and views. In the future, I would like to apply this process to current and future projects that I personally am working on. I would also like to learn about other types of design approaches.

## Joanna Hang

I learned about architectural design and practiced ADD and how to document an architecture system. From the QAW, I learned how to prioritize and refine quality attributes. I also learned how to write quality attribute scenarios and how to differentiate quality attributes from one another. I learned the tactics for each quality attribute. From ADD v2, I learned how to decompose a system, select patterns and instantiate the architecture. In addition, I learned how to create various views (module, component and connection, and allocation views). These views were helpful in detailing different aspects of the architecture for different stakeholders. Overall, I learned how quality attributes can drive the design of the architecture and can be used as architectural drivers in addition to functional requirements and design constraints.

## Lourdes Lopez

Throughout this exercise I learned the importance of capturing quality attributes and using the quality attributes to design the architecture of a system. A QAW session can be an effective way to begin communication with the stakeholders in terms of quality attributes and also to begin prioritization of quality attributes. I learned how to use ADD to iteratively design the architecture by decomposing an element per iteration and finding the architectural drivers for the selected element. I learned how to select tactics and how to analyze the tradeoff on quality attributes when selecting a tactic. With exercise three, I also learned how to document views as well as combine view based on the level of usage by each stakeholder type. I most definitely will use what I learned from this class in my professional work and would like to learn more in detail about tactics and patterns.

## David Sullivan

I learned from this experience that there are many aspects to architecture design. I could see how the QAW is a useful way to meet with the stakeholders and focus on quality attributes rather just gathering the functional requirements. By working from the quality attributes to the architecture view, I can understand how to iteratively create a software architecture that adheres to the business objectives and priorities. The ADD v.2 method decomposes the software by analyzing the quality attributes of the system requirements and the quality attributes are critical to the success of the project.

## Group Experience

Throughout this exercise we learned a great deal in regards to designing and documenting an architecture. Most importantly, we learned how to begin communication with stakeholders in order to gather requirements that impact the systems architecture. In Exercise two, we practiced how to conduct a quality attribute workshop. By practicing the QAW several times, we learned what type of questions need to be asked in order to bring about conversations that lead to capturing quality attribute requirements. Once the requirements were captured, we learned about combining, prioritizing, refining and documenting the quality attribute requirements. Once requirements were gathered we moved on to practicing ADD version two. With ADD we learned how to take the requirements that we captured and begin designing the system through an iterative process. We practiced this by completing two iterations. In the first iteration we learned how to decompose the system as a whole and in the second iteration we learned how to decompose one element. As a team, one thing that we would all like to learn in more detail is how to implement some of the tactics that we selected.

# Reference List

Android. (n.d.). Up and running with material design. Retrieved from <http://developer.android.com/design/index.html> on April 2016.

Apple, Inc. IOS human interface guidelines: Designing for ios. Retrieved from <https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/> on April 2016.

Bachmann, F., Bass, L., Clements, P., Garlan, D., Ivers, J., Little, R., Nord, R., Stafford, J. (2002, January). Documenting software architecture: Documenting behavior. Carnegie Mellon University. Retrieved from <https://www.sei.cmu.edu/reports/02tn001.pdf>

Bachmann, F., Bass, L., Clements, P., Garlan, D., Ivers, J., Little, R., Nord, R., Stafford, J. (2002, June). Documenting software architecture: Documenting interfaces. Carnegie Mellon University. Retrieved from <http://resources.sei.cmu.edu/asset_files/TechnicalNote/2002_004_001_13973.pdf>

Digicert. (n.d.). What Is SSL (Secure Sockets Layer) and What Are SSL Certificates?. Retrieved from <https://www.digicert.com/ssl.htm> on April 2016.

E-commerce Platforms. (2014). How to choose a payment gateway for your ecommerce store. Retrieved April 2016, from <http://ecommerce-platforms.com/ecommerce-selling-advice/choose-payment-gateway-ecommerce-store>

Federal Trade Commision. (n.d-a). Business guide to the FTC's mail, internet, or telephone order merchandise rule. Retrieved April 2016, from <https://www.ftc.gov/tips-advice/business-center/guidance/business-guide-ftcs-mail-internet-or-telephone-order>

Federal Trade Commission. (n.d.-b). CAN-SPAM rule. Retrieved April 2016, from <https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/can-spam-rule>

Federal Trade Commission. (n.d.-c). Online advertising and marketing. Retrieved April 2016, from <https://www.ftc.gov/tips-advice/business-center/advertising-and-marketing/online-advertising-and-marketing>

Lazazzera, R. (n.d.) The beginner's guide to ecommerce shipping and fulfillment – Shopify. (n.d.). Retrieved from <https://www.shopify.com/blog/14069585-the-beginners-guide-to-ecommerce-shipping-and-fulfillment> on April 2016.

Pair Network. (n.d.). E-Commerce - e-commerce basics - pair networks. (n.d.). Retrieved April 2016, from <https://www.pair.com/support/knowledge_base/e-commerce/e-commerce_basics.html>

PCI Security Standards Council. (2016). Official PCI security standards council site - verify PCI compliance, download data security and credit card security standards. Retrieved April 2016, from <https://www.pcisecuritystandards.org/>

PCI Security Standards Council. (2013, January). PCI DSS E-commerce Guidelines. Retrieved from <https://www.pcisecuritystandards.org/pdfs/PCI_DSS_v2_eCommerce_Guidelines.pdf> on April 2016.

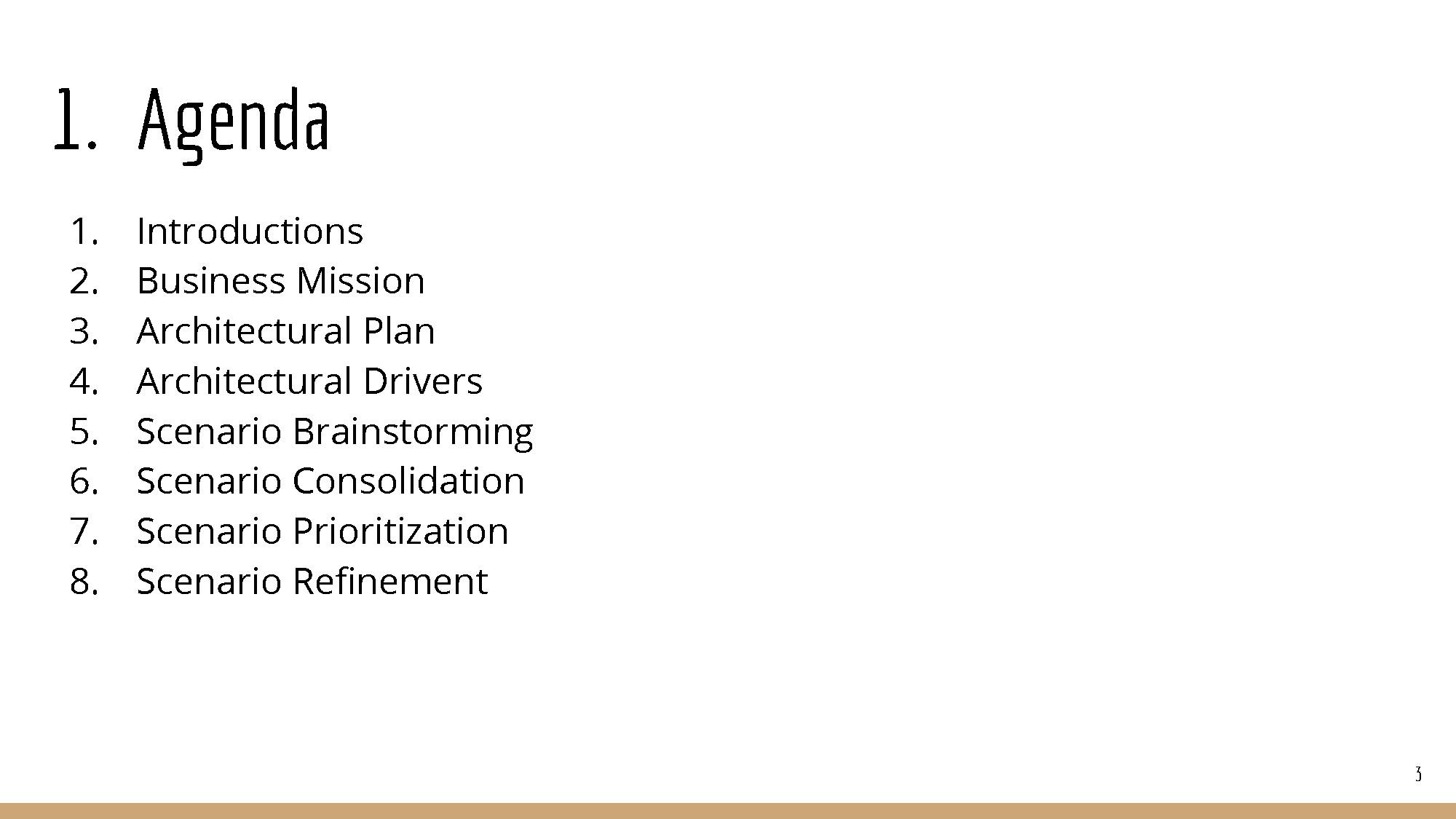
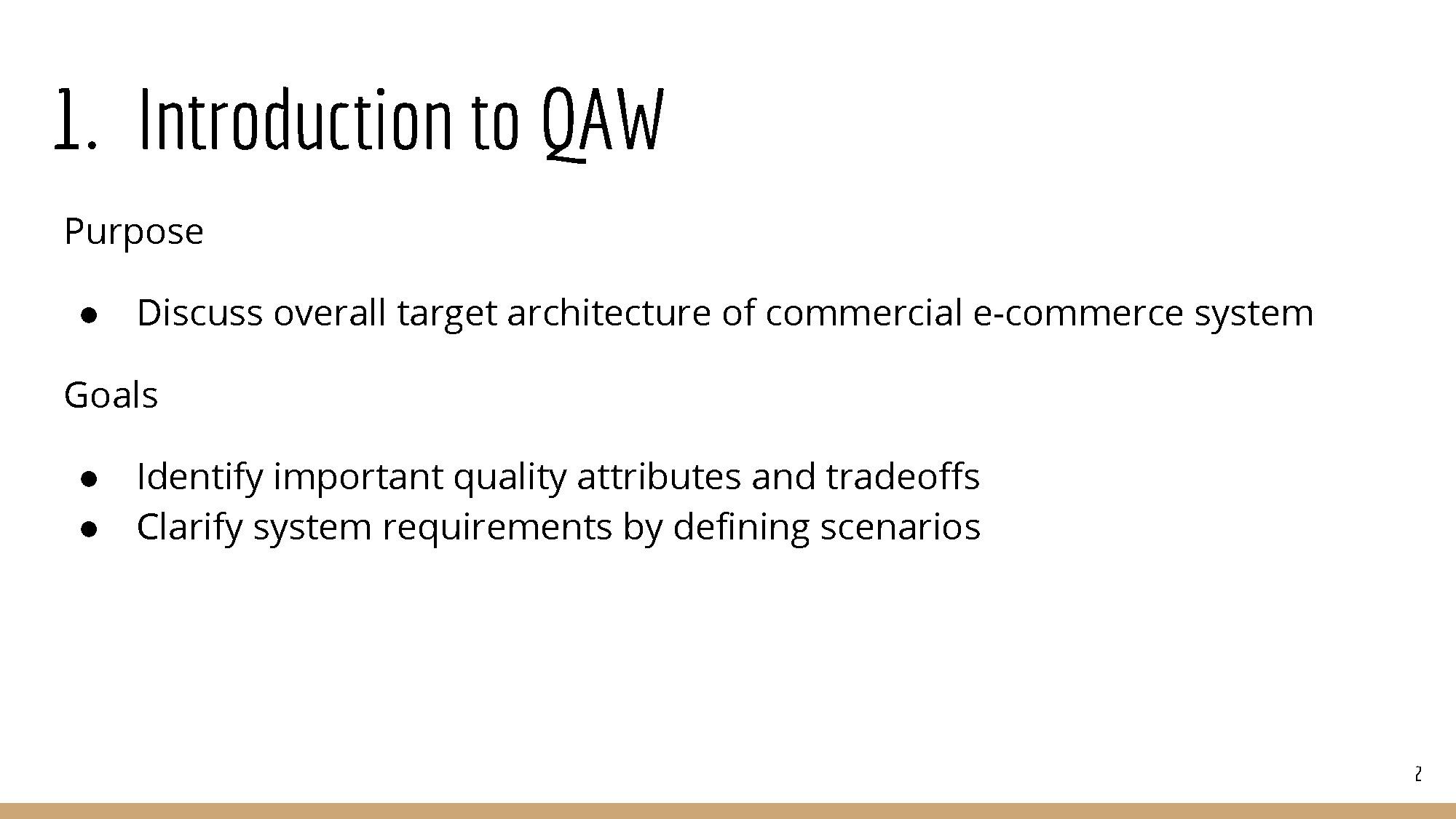
PCI Security Standards Council. (2015). Requirements and security assessment procedures, download data security and credit card security standards. Retrieved from <https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-1.pdf> on April 2016

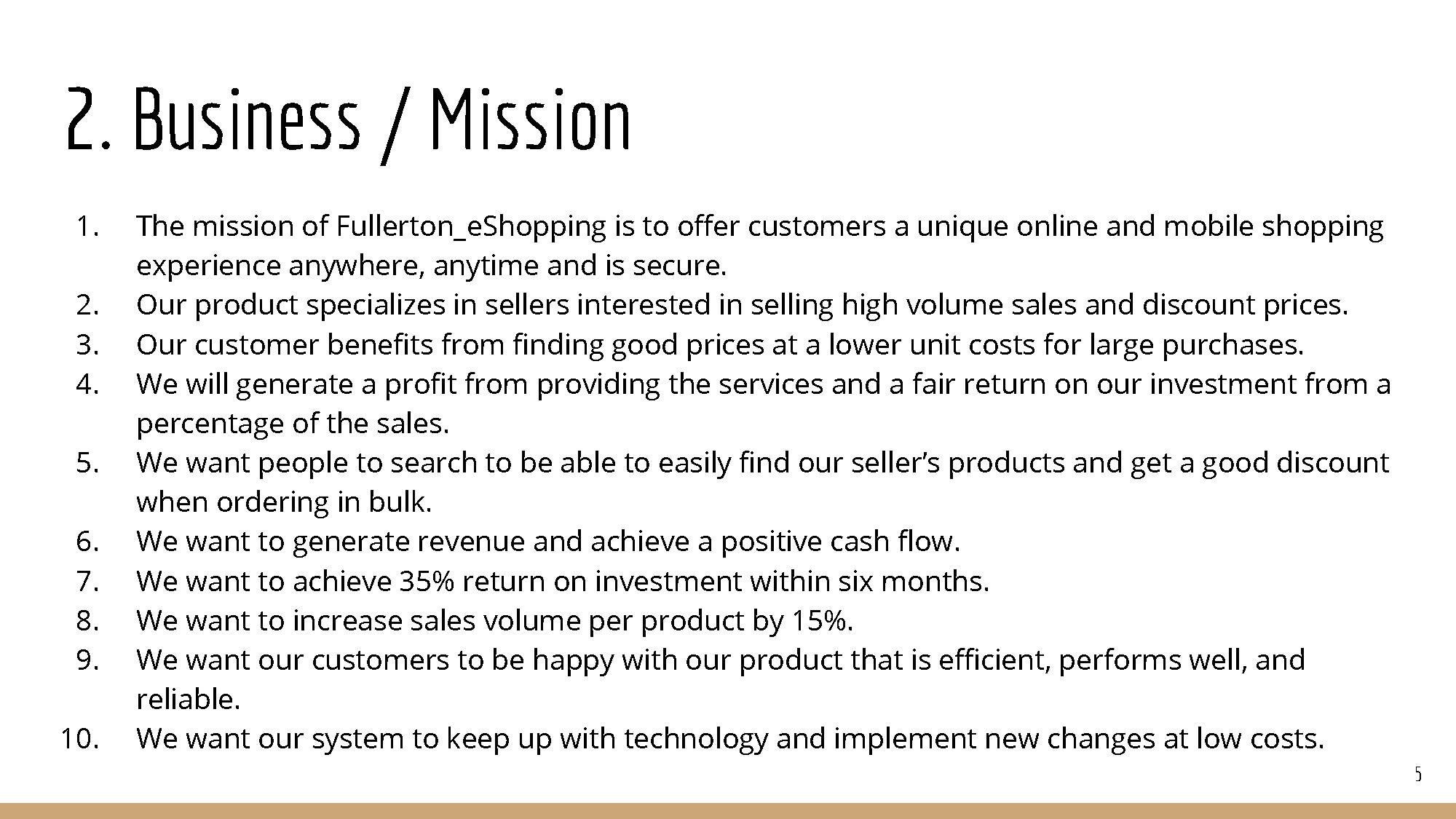
*The Module Viewtype*. (2016). Retrieved April 2016, from <http://ece.ut.ac.ir/classpages/S85/SoftwareArchitecture/08-The%20Module%20Viewtype.pdf>

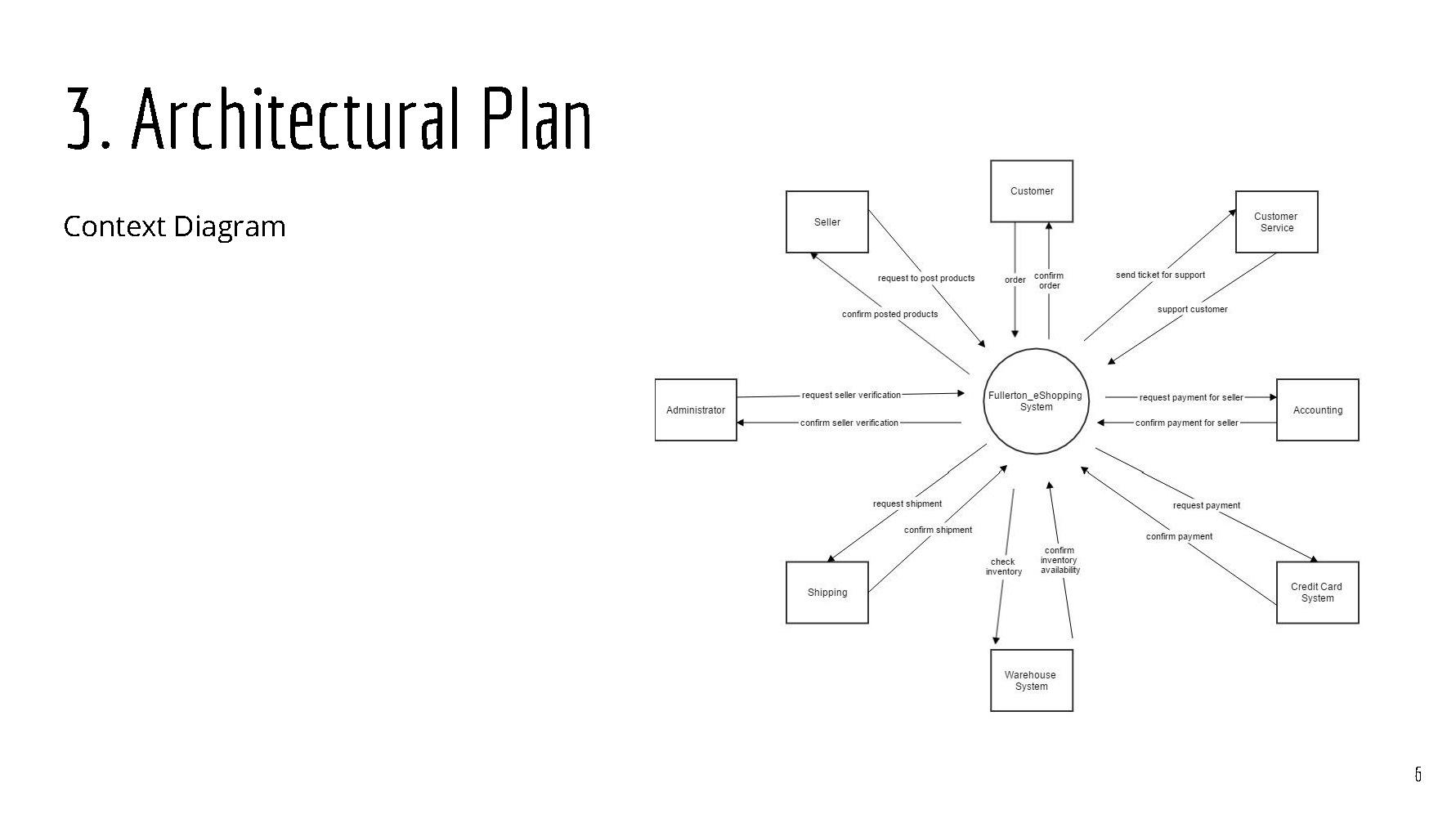
The University of British Columbia: Electrical and Computer Engineering. (n.d). Designing the architecture. Retrieved April 29, 2016, from <http://www.ece.ubc.ca/~matei/EECE417/BASS/ch07lev1sec2.html>

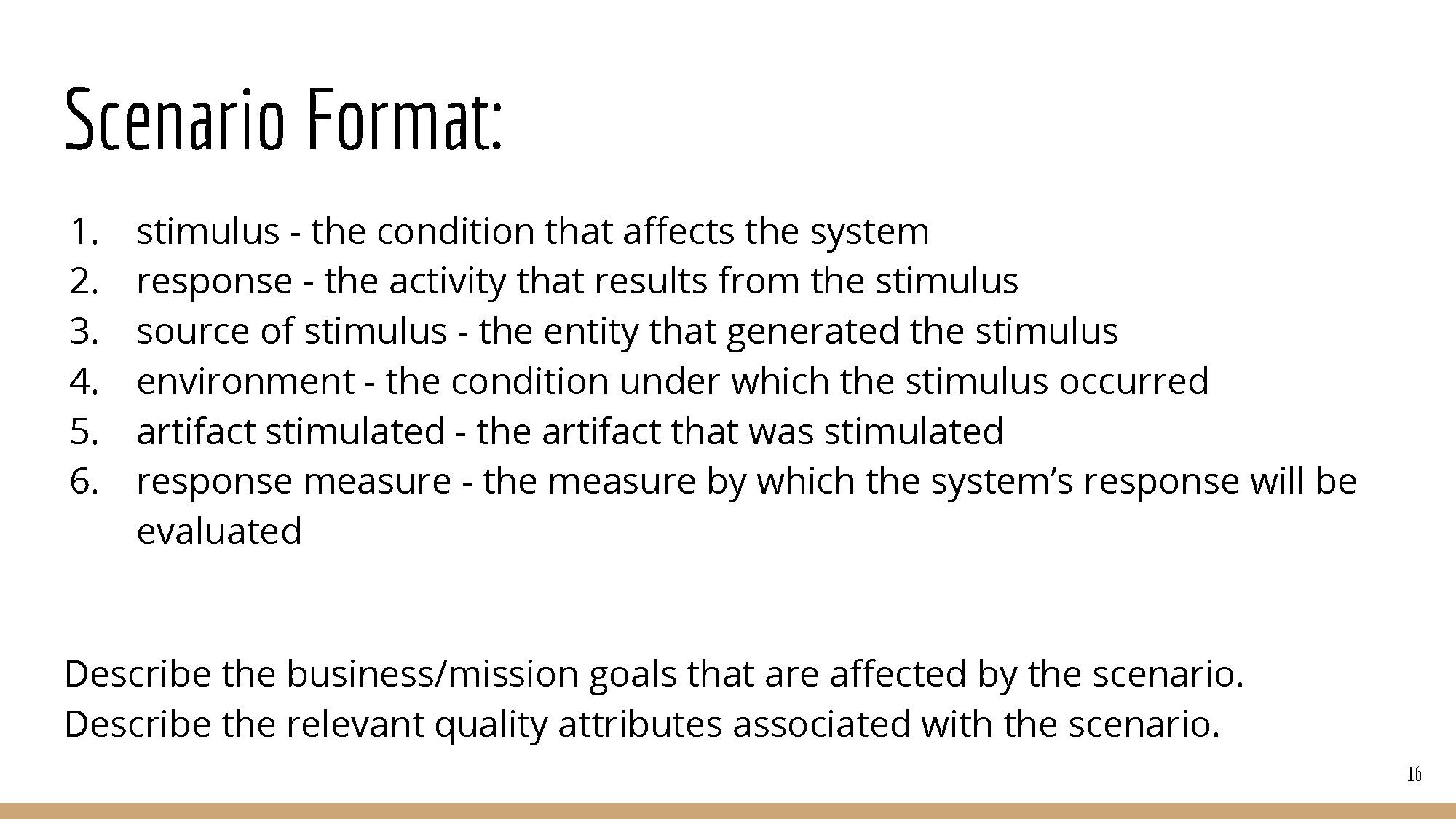
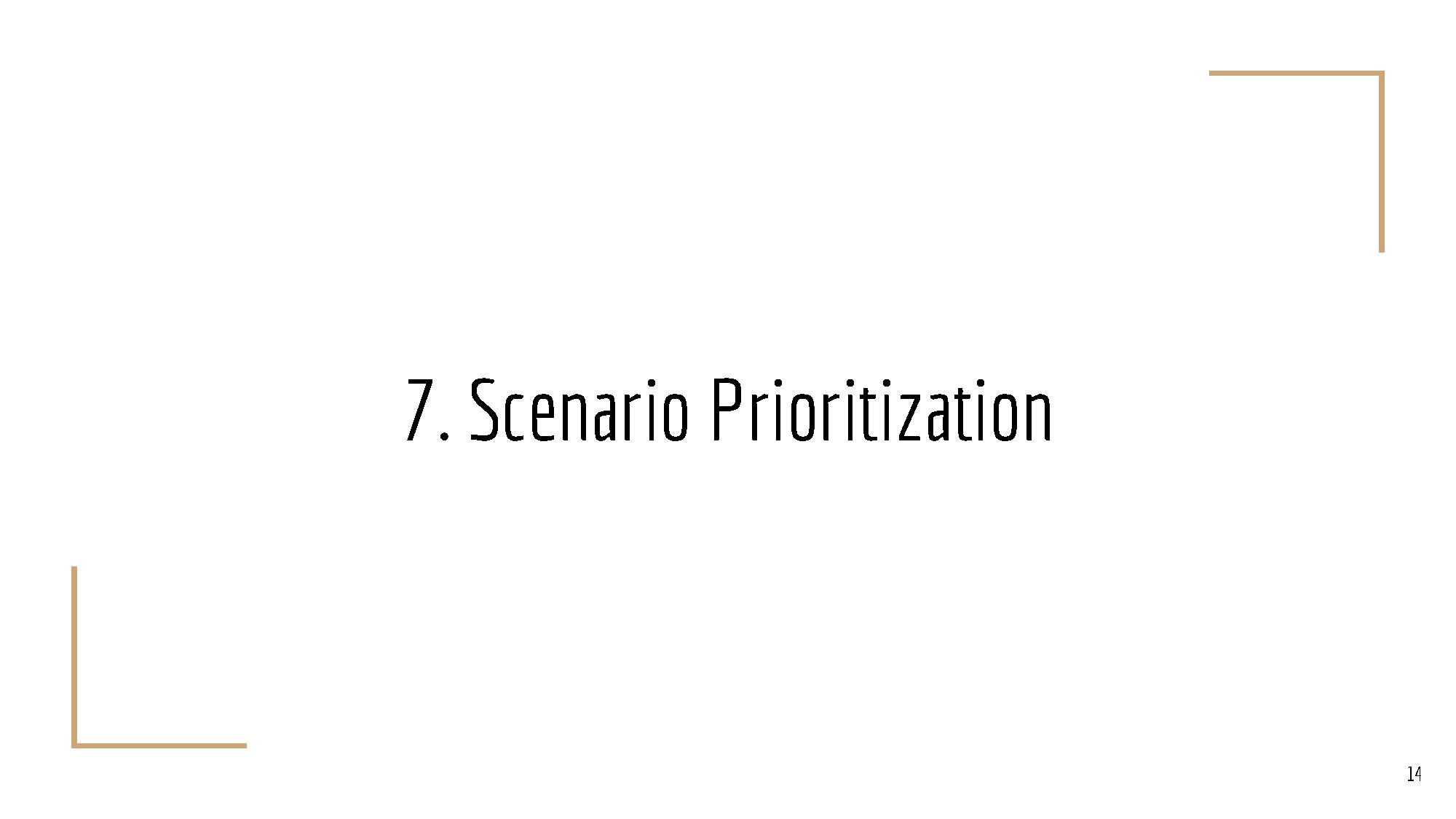
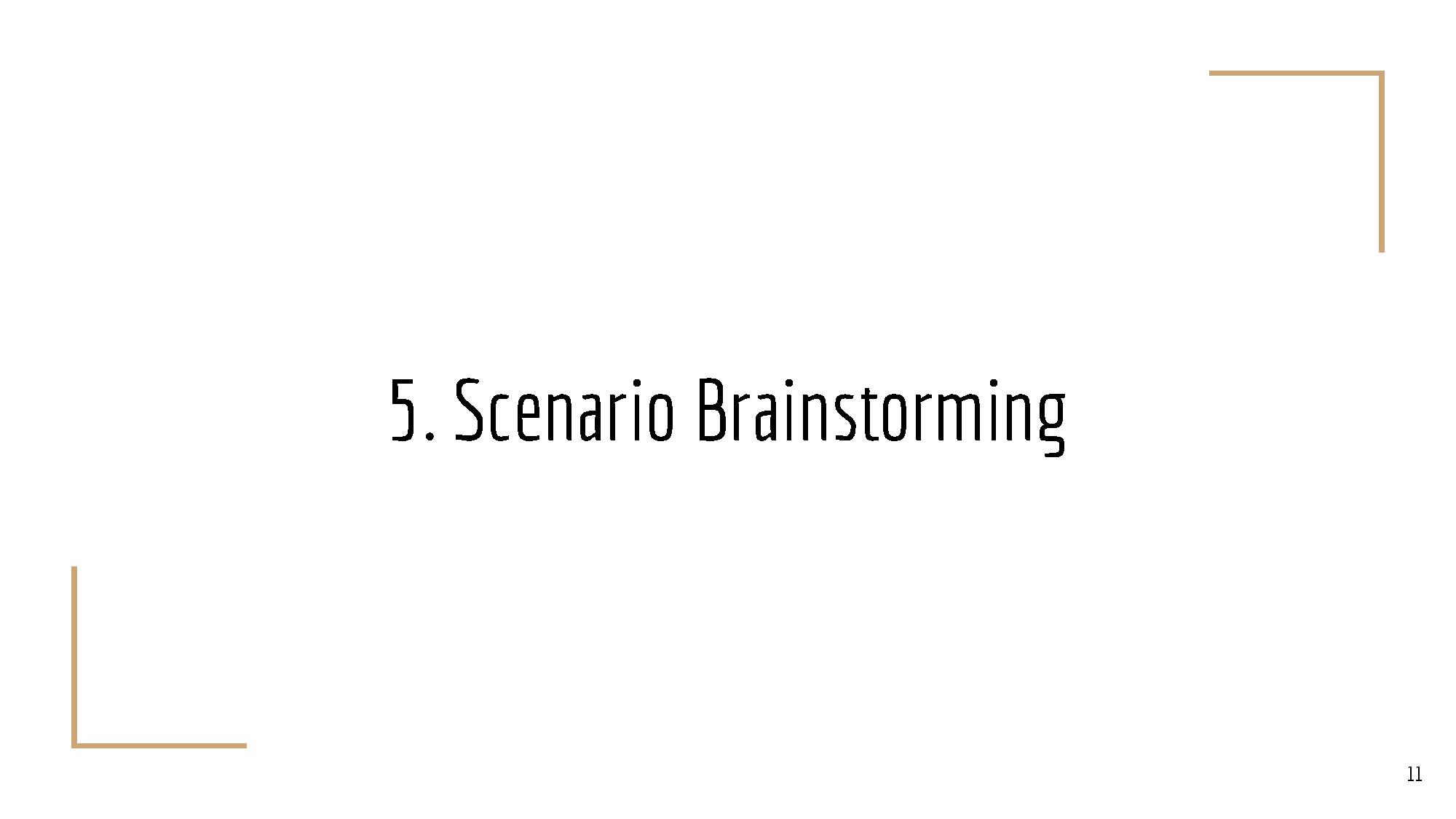
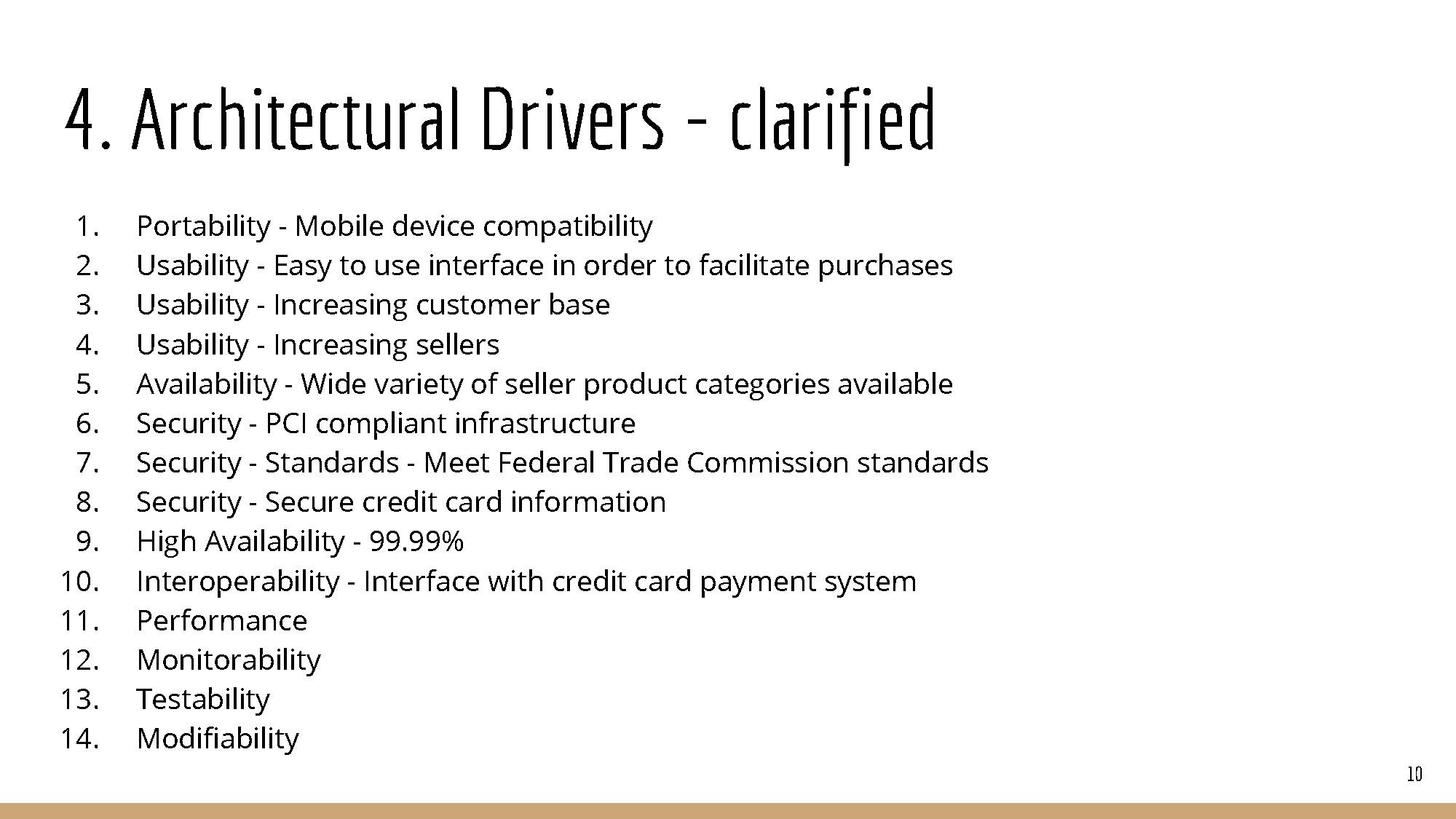
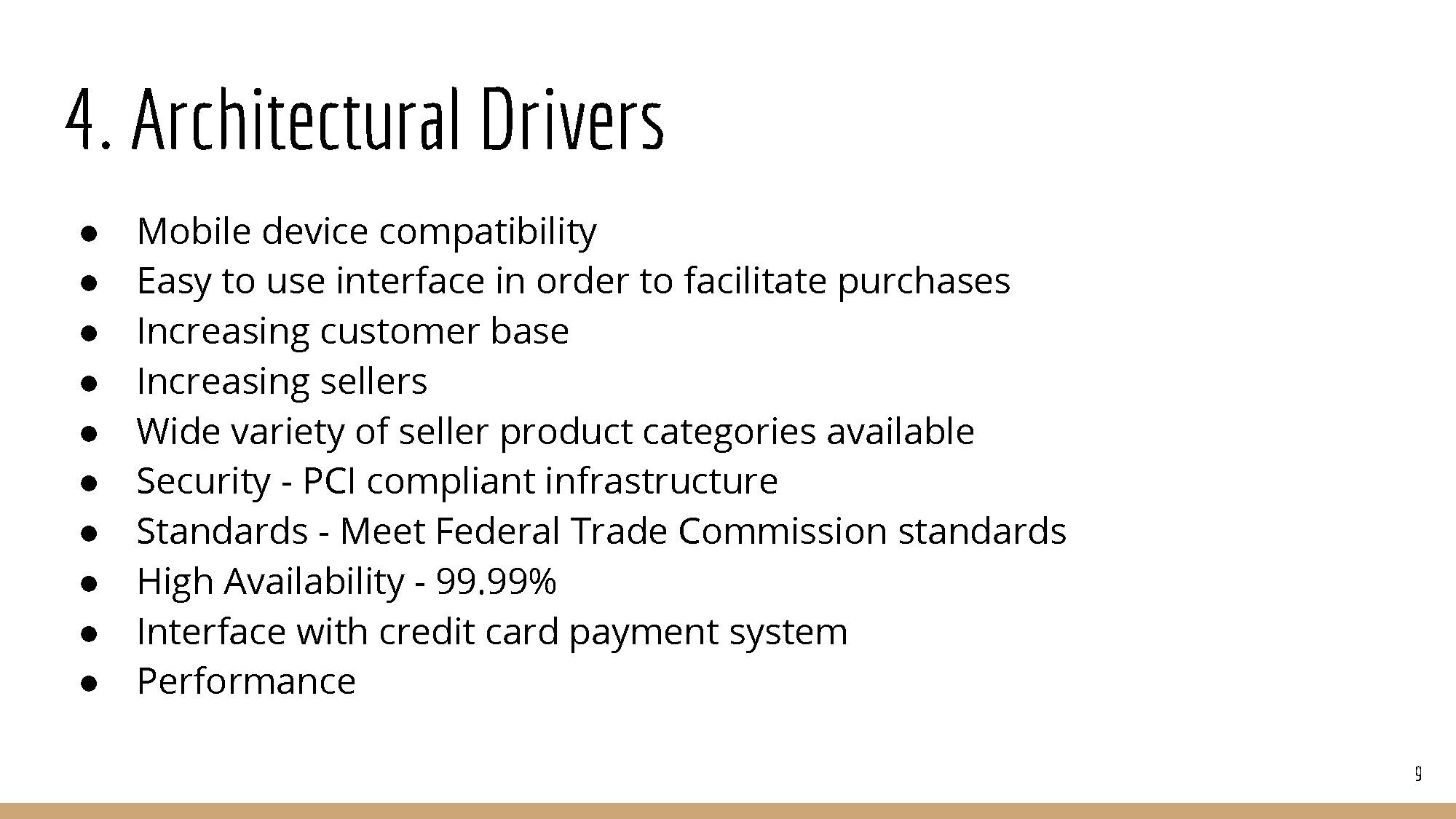
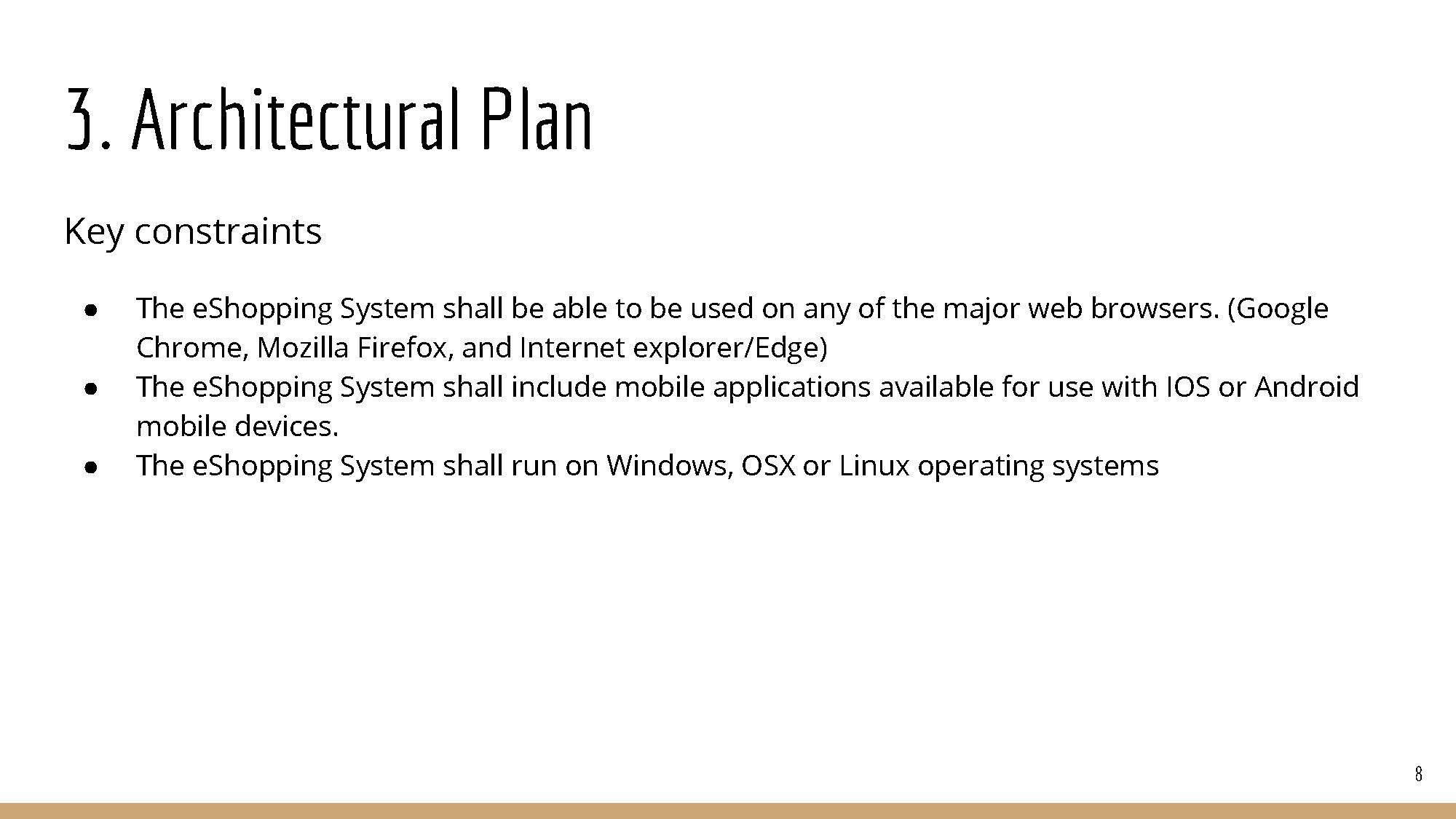
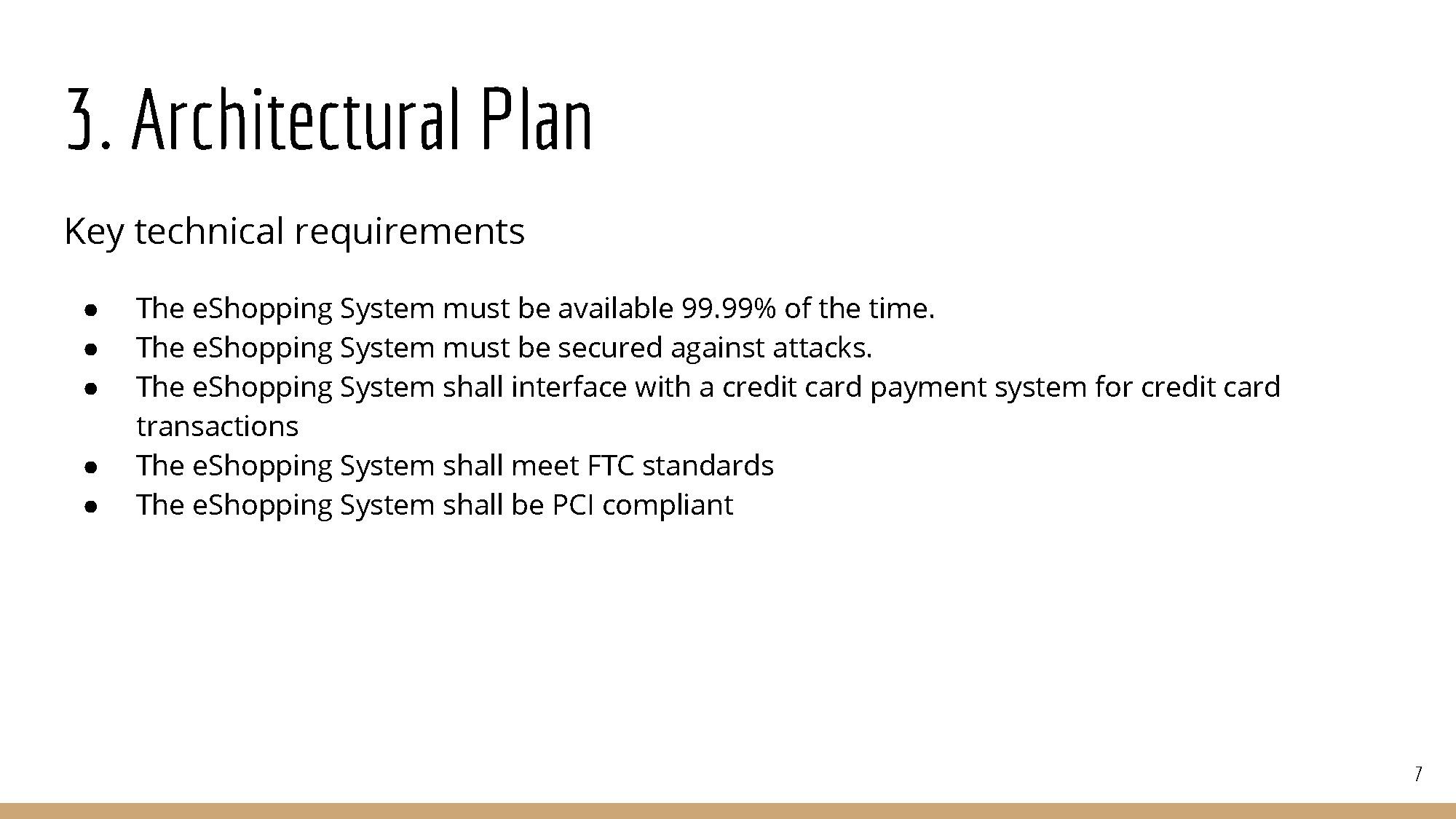
# Appendix

## Appendix A - Quality Attribute Workshop Presentation









# Team Charter

|  |  |  |
| --- | --- | --- |
| **Course Title** | CPSC 545 Software Design & Architecture | All team members participated in the creation of this charter and agree with its content.  **Date** 01/31/2016 |
| **Instructor** | Dr. Chang-Hyun Jo |
| **Course Dates** | 01/24/2016 – 05/14/2016 |

**Team Members** (Contact Information)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Address (city, state, country) | Phone | Cell | Email |
| Timothy Cioffi-Dinkel | San Diego, CA | 951-775-6150 | 951-775-6150 | timothycioffi@csu.fullerton.edu |
| Anthony Farina | Bear, DE | 302-299-5495 | 302-299-5495 | afarina@csu.fullerton.edu |
| Joanna Hang | Fullerton, CA | 714-686-7401 | 714-686-7401 | johang@fullerton.edu |
| Lourdes Lopez | Turlock, CA | 209-205-0581 | 209-205-0581 | lourdeslopez@csu.fullerton.edu |
| David Sullivan | Fullerton, CA | 714-686-9488 | 714-686-9488 | dsullivan@fullerton.edu |

**Team Member Skill Inventory** (Areas individual members can contribute)

|  |  |
| --- | --- |
| Timothy Cioffi-Dinkel | * MySQL * Java/C++ * XML, HTML/CSS, PHP * Photoshop * Game Development, Unity/C# |
| Anthony Farina | * MySQL - Database administration and design * Java * Android Development * Microsoft Office * Google Web Toolkit |
| Joanna Hang | * SQL Server - Database administration and design * SQL Reporting Services * HTML, CSS, JavaScript, jQuery, Bootstrap, and Less * Microsoft Office * Adobe Photoshop, InDesign, Digital Publishing Suite |
| Lourdes Lopez | * Data Analytics - Complex queries using T-SQL or PL-SQL * Business Intelligence tools - Business Objects Web Intelligence, Business Objects Universe Design, Microsoft SSRS and Crystal Reports * Web Application Development with C# ASP.NET and JavaScript * Experience with agile development using Scrum methodologies * Microsoft Visio for documenting business process |
| David Sullivan | * iOS Development * Android Development * Java * C#, ASP.NET, JavaScript, Bootstrap * Microsoft Web API |

**Team Goals** (Project goals, team process goals, quality goals, etc.)

|  |
| --- |
| * Combine the unique skill sets of each of our team members, so that we can produce a successful project as well as learn from each other's areas of expertise. * Learn about the software process by reviewing course material as well as experiencing the process as we work towards our course assignments. * Improve upon our existing knowledge of software development processes so that we will develop software in a more organized manner. * Keep track of our weekly assignments and goals so that we meet our deadlines and are well prepared for upcoming assignments. * Establish and maintain strong communication between teammates in order to facilitate success. |

**Team Roles** (Define roles of each member to achieve goals)

|  |  |
| --- | --- |
| Lourdes Lopez  Team Lead | * Act as the team liaison which includes communicating with the professor and/or class regarding the team’s progress and/or asking questions * Turn in assignments, take a screenshot of submitted assignment, and e-mail team members * Collaborate with Facilitator on agendas * Manage project timeline |
| Joanna Hang  Recorder | * Take meeting notes and put them in Google Drive * Document ideas and key decisions from meetings * Ensure team members know the purpose and goals for assignments |
| Timothy Cioffi-Dinkel  Facilitator | * Inform team of any changes to meetings * Assist team lead with agenda before meetings * Coordinate video calls/start video calls * Begin/end conference calls with a check-in with each team member * Ensure all team members participate and ask everyone for input. * Cover topics on the agenda * Manage meeting times |
| David Sullivan  System Architect | * Ensure the project is within the scope of technical requirements * Review project through a technical standpoint |
| Anthony Farina  Quality Assurance | * Ensure process meets users’ quality expectations * Extract collaborative assignments from Google Drive and convert documents to appropriate document type (Word, Excel, PDF, etc.) * Perform final review by checking format and reviewing assignments * Upload final copy to Google Drive and inform Team Lead that assignment is ready to turn in |

**Ground Rules** (Meeting schedule/locations, attendance expectations, agenda, assignment completion, communication methods, etc.)

|  |
| --- |
| * All team members will check Titanium daily. * All team members will check emails at least once per day and reply within 24 hours. * All team members will meet on Tuesday’s at 6:00 PM PST using Google Hangouts. * Google Hangouts will be the main tool for communication between team members. * Google Hangouts will be installed on each member’s mobile phone. * All team members will check Google Hangouts at least twice per day. * All team member will complete collaborative assignments using Google Drive. * All team members must be respectful, honest, and clear to all other team members at all times. * No plagiarism or cheating. Each team member will proof read each other’s work to prevent this. * Group will rotate team leaders each semester. * All team members will communicate and ask questions. * All team will equally contribute, show effort, and will not leave the work to a few individuals. * All team members will be supportive, willing to help and assist other team members. * Each member must be open to constructive criticism from each other team member. * Each member is responsible for their tasks. Tasks must be completed by the deadlines set by the professor and or the group. * If emergency arises that may affect deadlines, meetings, etc., team member will at least inform the team leader or if possible the entire group via Google Hangout about the situation so the group can plan accordingly. |

**Time Commitments/Availability** (Pacific Time)

|  |  |
| --- | --- |
| Timothy Cioffi-Dinkel | * M-F 5pm-8pm * Sat-Sun all day |
| Anthony Farina | * M-F 5pm-8pm * Sat-Sun all day |
| Joanna Hang | * M-F 5pm-8pm * Sat-Sun all day |
| Lourdes Lopez | * M-F 5pm - 8pm * Sat-Sun all day |
| David Sullivan | * M-F 5pm - 8pm * Sat-Sun all day |

**Conflict Management** (What are potential conflicts that might arise among or between team members during this course? How will team members deal with these and other conflicts?)

|  |
| --- |
| * In order to avoid conflict among team members, roles and responsibilities should be equally designated. * If conflicts arise, concerns should be submitted to the agenda and will be addressed to the whole team to settle conflicts. |

**Risk Management** (What are potential barriers to the achievement of these goals?)

|  |
| --- |
| * Identify risk factors within team (meeting time conflicts , external commitments to work and family) and within the software process (training, organizational policies, configuration) * Prioritize tasks in order of importance and communicate to stakeholders * When problems arise, communicate with team and stakeholders * Actively monitor progress and reevaluate requirements |

**Team Evaluation Criteria** (List evaluation criteria that will be used to evaluate team members objectively.)

|  |
| --- |
| * Evaluate each individual team member on their own work. * Evaluate each individual team member on their participation in group discussions. * Evaluate each individual team member on their ability to collaborate on the group project. * Evaluate each individual team member on their ability to produce material in a timely matter. * Evaluate each individual team member on their ability to fulfill role. |

# Team Evaluation

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Members** | Timothy Cioffi-Dinkel | Anthony Farina | Joanna Hang | Lourdes Lopez | David Sullivan | **Total** | **Comments on Your Evaluation on Team** |
|
| **Evaluators** |
| Timothy Cioffi-Dinkel | 100 | 100 | 100 | 100 | 100 | **500** | Great work, solid team |
| Anthony Farina | 100 | 100 | 100 | 100 | 100 | **500** | Everyone did their part with no trouble. |
| Joanna Hang | 100 | 100 | 100 | 100 | 100 | **500** | Great work! |
| Lourdes Lopez | 100 | 100 | 100 | 100 | 100 | **500** | Great job team! |
| David Sullivan | 100 | 100 | 100 | 100 | 100 | **500** | Everyone stepped up and worked hard |
| **Total** | 500 | 500 | 500 | 500 | 500 | 2500 |  |
| **Max** | 500 | 500 | 500 | 500 | 500 | 500 |  |
| **Average** | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 500.00 |  |
| **Percent** | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% |  |
| **Signature** |  |  | sig.jpg | Untitled.png |  |  | Work was divided amongst all team members and exceeded our expectations. |
| **Comments on Your Score Earned from Team** | Fair enough | I'm happy. | Great! | Fair | I'm happy |  |  |