Artisans Coop North System Upgrade

System Specification

Prepared For:

Artisan’s Coop North’s Developer

Prepared By:

Jack’s Technologies

December 3, 2017

**Table of Contents**

PAGE

**EXECUTIVE SUMMARY………………………………………………………………………………………………………………………3**

**I. INTRODUCTION………………………………………………………………………………………………………………………………4**

Problem Statement and Project Vision………………………………………………………………………………………………4

System Services…………………………………………………………………………………………………………………………………4

Nonfunctional Requirements and Design Constraints………………………………………………………………………4-5

System Evolution………………………………………………………………………………………………………………………………5

Document Outline…………………………………………………………………………………………………………………………….5

**II. STRUCTURAL MODEL…………………………………………………………………………………………………………………….6**

Introduction……………………………………………………………………………………………………………………………………..6

Class Diagram……………………………………………………………………………………………………………………………………6

Metadata………………………………………………………………………………………………………………………………………….7

**III. ARCHITECTURE DESIGN……………………………………………………………………………………………………………….21**

Introduction……………………………………………………………………………………………………………………………………..21

Infrastructure Model………………………………………………………………………………………………………………………..22

Deployment Diagram 1: Architecture Model……………………………………………………………………………………22

Deployment Diagram 2: Nodes and Artifacts…………………………………………………………………………………..23

Hardware and Software Requirements…………………………………………………………………………………………….24

Security Plan…………………………………………………………………………………………………………………………………….26

**IV. USER-INTERFACE…………………………………………………………………………………………………………………………27**

User-Interface Requirements and Constraints………………………………………………………………………………….27

Window Navigation Diagram……………………………………………………………………………………………………………28

Forms: Screen/User-Interaction Design……………………………………………………………………………………………29

Reports “Printed Output” Design……………………………………………………………………………………………………..31

**V. APPENDICES…………………………………………………………………………………………………………………………………32**

Bibliography and References…………………………………………………………………………………………………………….32

Supporting Documentation………………………………………………………………………………………………………………33

**Executive Summary**

Ms. Elaine Weltz, a business facilitator at Artisan’s Coop North, hired Jack’s Technologies to develop multiple applications for a system named Artisan’s Coop North Upgrade Project. This system will allow Artisan’s Coop North to go fully digital by using a database for inventory tracking and check-in, completing business transactions via Credit Card readers, and viewing artwork over ArtisansCoopNorth.com. In later versions, we’ll add security features to Artisan’sCoopNorth.com as well as making the system applicable over other operating systems such as iOS.

We’ve finished an analysis of the feasibility, requirements, and system evolution of Artisan’s Coop North Upgrade Project. We have also planned system structure, concluded on the necessary software and hardware, and created a user-interface design. We are ready to develop the project and will have the first version released in three months.

This document contains five sections. The Introduction in the coming section will give an overview of the project; the Structural Model will discuss the system’s classes; Architecture Design will list all the software and hardware required for the project; User-Interface will show website layout and navigation systems of ArtisansCoopNorth.com; and the Appendices will give credit to sources.

**I. Introduction**

**Problem Statement and Project Vision**

Artists at Artisan’s Coop North are using inefficient methods to complete their work. These methods, such as using written receipts and Excel documents for inventory tracking, are considered obsolete in the digital world and can drastically reduce productivity. Artisan’s Coop North Upgrade Project addresses these issues by adding an online database, adding support for Credit Card readers, and allowing customers to view artwork online.

The stakeholders include craftsman at Artisan’s Coop North, who will use the system to improve their work efficiency and make it easier for customers to view their work; Ms. Elaine Weltz of Artisan’s Coop North as well as any other investors; the development team at Jack’s Technologies, who’s finances and reputation depend on the outcome of the project; and art customers, who will now have more convenient access to artwork from Artisan’s Coop North’s craftspeople.

**System Services**

Artisan’s Coop North’s Upgrade Project’s functional requirements are as follows:

1. Users can register and create accounts with a unique username and password of their choosing. They can then login to ACoopNorth.com using their username and password (see use-case 1- Register for account.)

2. Users, after logging into their account, must be able to input and view information about their artwork in the database. The user must also be able to track inventory and sales using the database. The database must be able to categorize information based on fields requested by the user (see use-cases 4-Store artwork and 5- Submit information to the database).

3. Customers must be able to access ACoopNorth.com and use the viewing feature to access pictures of artwork that have been uploaded. The customer must also be able to zoom-in and rotate the artwork as a 3D model. If the customer wants to purchase the artwork, they can do so using a credit or debit card (Visa or Mastercard only). (see use-cases 6- View artwork online and 7- Pay via credit/debit card).

4. Ms. Weltz’s account must be able to make payments directly into the bank accounts of employees via E-banking (see use cases 6- View artwork online and 6.2 Pay artist).

*\*Use case diagram and all use-case descriptions can be found in Section 5.0 Requirements Model of the System Proposal*

**Nonfunctional Requirements and Design Constraints**

There are some limitations to consider when developing Artisan’s Coop North Upgrade System.

* Each artist must only be able to view inventory and sales of artwork that belongs to them.
* Customers must only be able to view artwork from the warehouse.
* Only Ms. Weltz must be able to view all artists’ banking information and make payments to them.
* Only Ms. Weltz can access all information stored in database by all artists.
* Only Ms. Weltz can upload pictures onto ACoOpNorth.com for customer access.
* Only Ms. Weltz can check-in artwork that arrive at the warehouse.
* Only the artist who created the artwork must be able to modify information about the artwork in the database.
* The database must support at-least 1000 users.
* The database should respond to user requests fast.
* The system should retain a 98% uptime.
* The system can’t be down for no longer than 30 minutes for maintenance during peak-time, and two hours for off-time.
* The mobile and web application must have an easy learning curve and come with a walkthrough to remove the need for training.
* The mobile application must be available on Android, iOS, and Windows.
* The mobile application must be available for free for the users.
* The web application must be available on Google Chrome, Microsoft Edge, Mozilla Firefox, and Internet Explorer.

**System Evolution**

In the first version of Artisan’s Coop North’s Upgrade System, we will deliver the core components of the system: A functional database for information storage and inventory and sales tracking, a viewing feature for customers to view artwork on ArtisansCoopNorth.com, support for credit card readers, and register option for users to create accounts on ArtisansCoopNorth.com. In the next version, we will add upgrade security for ArtisansCoopNorth.com as well as make the system usable on other operating systems such as iOS.

**Document Outline**

This document includes four main sections:

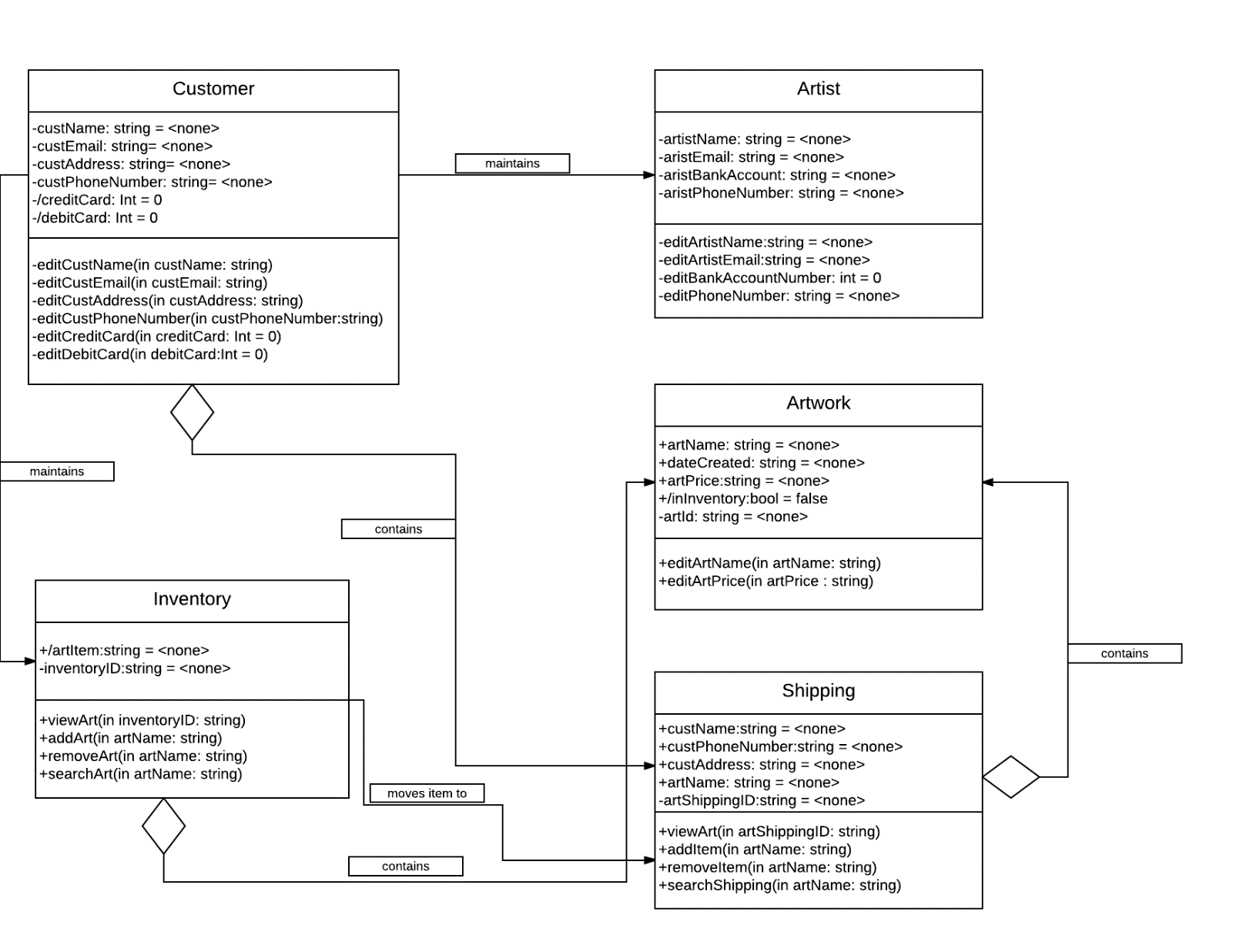
1. Structural Model: relationships between objects and classes in the system.
2. Architecture Design: the necessary hardware and software required by the system.
3. User-Interface: the screen components of the system and how the user will navigate through it.
4. Appendices: Definitions for terminology, resources, and supporting documentation.

**II. Structural Model**

**Introduction**

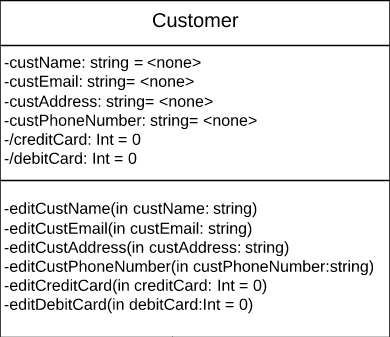
This section contains a Class Diagram and Metadata subsections. The Class Diagram provides an overview of the system classes and the relationships between the classes. The Metadata provides more detail about the attributes and operations of each class. The Metadata section is organized into tables and processing outlines.

**Class Diagram**



**Metadata**

Customer Class



Description: Represent a customer

Visibility: Private

Is Abstract: No

Attributes:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Data Type | Is Derived | Read Only | Visibility | Multiplicity | Default Value |
| custName | Customer Last, First | String | No | No | Private | 0…..\* | <none> |
| custEmail | Customer email address | String | No | No | Private | 1 | <none> |
| custAddress | Customer physical address | String | No | No | Private | 1 | <none> |
| custPhoneNumber | Customer phone number | String | No | No | Private | 1 | <none> |
| -/creditCard | Customer credit card number | Int | No | Yes | Private | 1 | 0 |
| -debitCard | Customer debit card number | Int | No | Yes | Private | 1 | 0 |

Operations:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Return Type | Parameters | Visibility | Scope | Is Query | Is Polymorphic |
| editCustName | Allows customer to change name | <none> | custName  Direction: in String  Default: None | Private | Instance | No | No |
| editCustEmail | Allows customer to change email | <none> | custEmail  Direction: in String  Default: None | Private | Instance | No | No |
| editCustAddress | Allows customer to change physical address | <none> | custAddress  Direction: in String  Default: None | Private | Instance | No | No |
| editCustPhoneNumber | Allows customer to change phone number | <none> | custPhoneNumber  Direction: in String  Default: None | Private | Instance | No | No |
| editCreditCard | Allows customer to edit credit card number | 0 | creditCard  Direction: in Integer  Default: 0 | Private | Instance | No | No |
| editDebitCard | Allows customer to edit debit card number | 0 | debitCard  Direction: in Integer  Default: 0 | Private | Instance | No | No |

Processing outlines:

editCustName

User inputs new customer name

Save new name

Erase previous name

Display “Saved message

editCustEmail

User inputs new customer email

Check email address for validity

IF Valid

Save new email address

Erase previous email address

Display “Saved message

ELSE

Reply “Invalid Email Address, try again”

editCustAddress

User inputs new address

Checks address for validity

IF Valid

Save new address

Erase previous address

Display “Saved message”

ELSE

Reply “Invalid address, try again”

editCustPhone

User inputs new phone number

Checks phone number for validity

IF Valid

Save new phone number

Erase previous phone number

Display “Saved message”

ELSE

Reply “Invalid phone number, try again”

editCreditCard

User inputs new credit card number

Check credit card number for validity

IF Valid

Save new credit card number

Erase previous credit card number

Display “Saved message”

ELSE

Reply “Invalid credit card number, try again”

editDebitCard

User inputs new debit card number

Check debit card number for validity

IF Valid

Save new debit card number

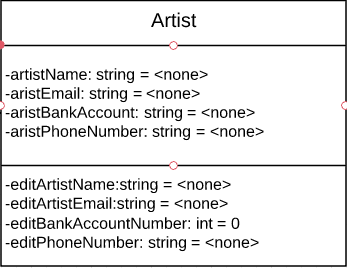
Erase previous debit card number

Display “Saved message”

ELSE

Reply “Invalid debit card number, try again”

Artist Class



Description: Represents artists

Visibility: Public

Is Abstract: No

Attributes:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Data Type | Is Derived | Read Only | Visibility | Multiplicity | Default Value |
| artistName | Artist Last, First | String | No | No | Private | 1 | <none> |
| artistEmail | Artist email address | String | No | No | Private | 1 | <none> |
| artistBankAccount | Artist physical address | String | No | No | Private | 1 | <none> |
| aristPhoneNumber | Artist phone number | String | No | No | Private | 1 | <none> |

Operations:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Return Type | Parameters | Visibility | Scope | Is Query | Is Polymorphic |
| editArtistName | Allows artist to change name | <none> | artistName  Direction: in String  Default: None | Public | Instance | No | No |
| editArtistEmail | Allows customer to change email | <none> | artistEmail  Direction: in String  Default: None | Privat | Instance | No | No |
| editBankAccount | Allows customer to change bank account | <none> | artistBankAccount  Direction: in String  Default: None | Private | Instance | No | No |
| editPhoneNumber | Allows customer to change phone number | <none> | artistPhoneNumber  Direction: in String  Default: None | Private | Instance | No | No |

Processing outlines:

editArtistName

User inputs new name

Save new name

Erase previous name

Display “Saved message”

editArtistEmail

User inputs new email address

Check email address for validity

IF Valid

Save new email

Erase previous email

Display “Saved message”

ELSE

Reply “Invalid email address, try again”

editBankAccount

User inputs new bank account number

Check bank account number for validity

IF Valid

Save new bank account number

Erase previous bank account number

Display “Saved message”

ELSE

Reply “Invalid bank account number, try again”

editPhoneNumber

User inputs new phone number

Check phone number for validity

IF Valid

Save new phone number

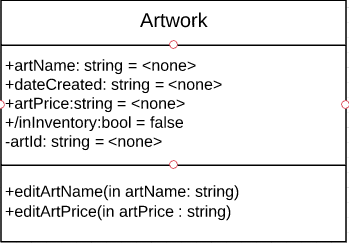
Erase previous phone number

Display “Saved message”

ELSE

Reply “Invalid phone number, try again”

Artwork Class



Description: Represents artwork

Visibility: Public

Is Abstract: No

Attributes:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Data Type | Is Derived | Read Only | Visibility | Multiplicity | Default Value |
| artName | Art name | String | No | No | Public | 0…..\* | <none> |
| dateCreated | Art date created | String | No | No | Public | 1 | <none> |
| artPrice | Art price | String | No | No | Public | 1 | <none> |
| inInventory | Whether art is in inventory | Bool | Yes | Yes | Public | 1 | false |
| artID | Unique identifier | String | No | Yes | Private | 1 | <none> |

Operations:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Return Type | Parameters | Visibility | Scope | Is Query | Is Polymorphic |
| editArtName | Allows user to change art name | <none> | artName  Direction: in String  Default: None | Public | Instance | No | No |
| editArtPrice | Allows user to change art price | <none> | artPrice  Direction: in String  Default: None | Public | Instance | No | No |

Processing outlines:

editArtName

User inputs new name

Save new name

Erase previous name

Display “Saved message

editArtPrice

editPhoneNumber

User inputs new price

Check price for validity

IF Valid

Save new price

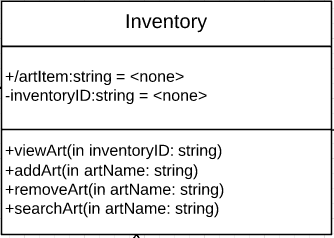
Erase previous price

Display “Saved message”

ELSE

Reply “Invalid price format, try again”

Inventory Class



Description: Represents inventory of artwork

Visibility: Public

Is Abstract: No

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Data Type | Is Derived | Read Only | Visibility | Multiplicity | Default Value |
| artItem | Name of artwork | String | No | No | Public | 1 | <none> |
| inventoryID | Unique identifer | String | No | No | Private | 1 | <none> |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Return Type | Parameters | Visibility | Scope | Is Query | Is Polymorphic |
| viewArt | Displays list of artwork in inventory | <none> | inventoryID  Direction: in String  Default: None | Public | Instance | Yes | Yes |
| addArt | Adds artwork to inventory | <none> | artItem  Direction: in String  Default: None | Public | Instance | No | Yes |
| removeArt | Removes artwork from inventory | <none> | artItem  Direction: in String  Default: None | Public | Instance | No | Yes |
| searchArt | Allows user to search for a specific artwork in inventory | <none> | artItem  Direction: in String  Default: None | Public | Instance | No | No |

Processing outlines:

viewArt

Load art items in Inventory

Display art items in Inventory

addArt

User inputs new art item

User saves new art item

Display “Saved” message

removeArt

User deletes art item from inventory

Display “Removed” message

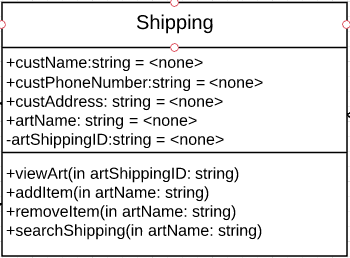
searchArt

User enters input

Load list of art items that contain input in name

Display list of art items

Shipping



Description: Represents shipping of artwork

Visibility: Public

Is Abstract: No

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Data Type | Is Derived | Read Only | Visibility | Multiplicity | Default Value |
| custName | Customer Last, First | String | Yes | No | Public | 0….\* | <none> |
| dateCreated | MM/DD/YY date of art created | String | No | No | Public | 1 | <none> |
| artPrice | Price of art | String | No | No | Public | 1 | <none> |
| inInventory | Whether art is still in inventory | Bool | No | No | Public | 1 | False |
| artID | Unique identifer | String | No | Yes | Private | 1 | none |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Return Type | Parameters | Visibility | Scope | Is Query | Is Polymorphic |
| viewArt | Displays list of artwork for shipping | <none> | inventoryID  Direction: in String  Default: None | Public | Instance | Yes | Yes |
| addItem | Adds artwork to shipping | <none> | artItem  Direction: in String  Default: None | Public | Instance | No | Yes |
| removeItem | Removes artwork from shipping | <none> | artItem  Direction: in String  Default: None | Public | Instance | No | Yes |
| searchShipping | Allows user to search for a specific artwork in shipping | <none> | artItem  Direction: in String  Default: None | Public | Instance | No | No |

Processing outlines:

viewArt

Load art items in shipping

Display art items in shipping

addItem

User inputs art item into shipping

User saves art item in shipping

Display “Saved” message

removeItem

User deletes art item from Shipping

Display “Removed” message

searchShipping

User enters input

Load list of art items from shipping that contain input in name

Display list of art items in shipping

**III. Architecture Design**

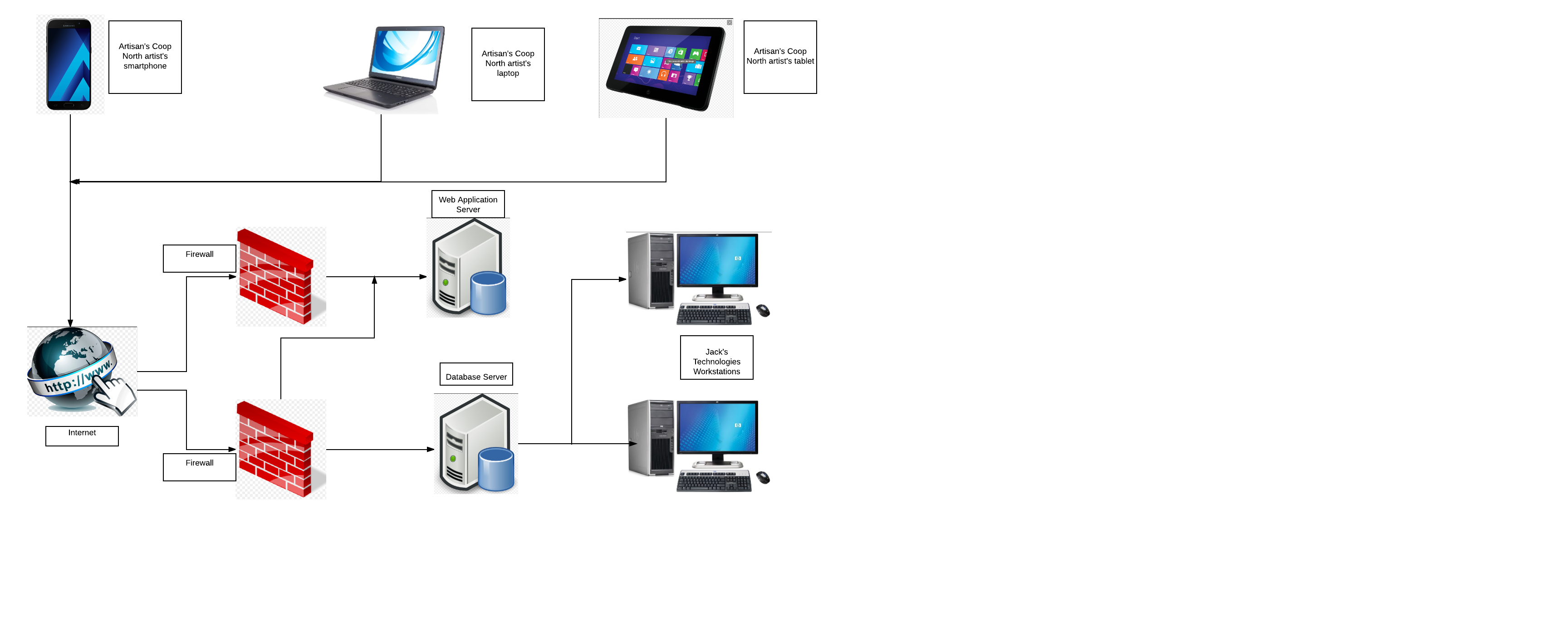
**Introduction**

This section has three subsections. The Infrastructure model displays two Deployment Diagrams, which explains how the hardware will be connected as well as what artifacts will reside on which devices. The Hardware and Software Requirements will list the necessary hardware and software that are needed to maintain Artisan’s Coop North. The Security Plan will describe potential threats to hardware, software, people, and servers, as well as how to address those threats.

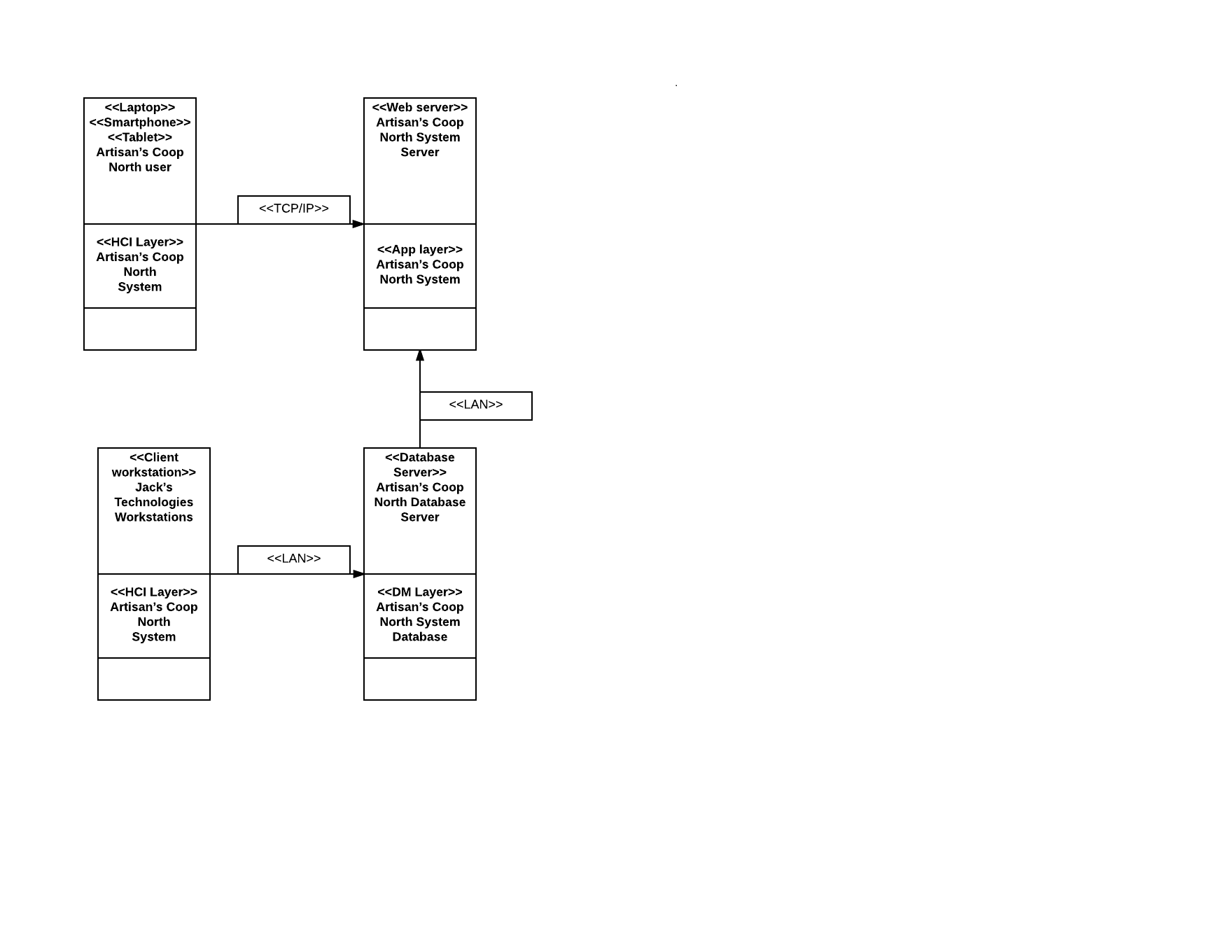
Artisan’s Coop North Upgrade System will be a three-tier client-server system, using a thin model. It will require the purchase of two main server as well as a backup server stored in another location. Employees at Jack’s Technologies should purchase workstations if they don’t have them already. Jack’s Technologies will also need a modem, router, and security-related software such as anti-spyware and anti-virus software.

**Infrastructure Model**

Deployment Diagram 1: Architecture Overview



Deployment Diagram 2: Nodes and Artifacts



**Hardware and Software Requirements**

Required Hardware Components

* Servers
* Database, Web Application, and Backup Servers must be purchased to store Artisan’s Coop North’s data and application logic. One main database server along with one web application server will be located at Jack’s Technologies. Jack’s Technologies recommend that a backup server with both database and application logic backups should be located in a secure location that has minimal risk of fire or other disasters.
* Jack’s Technologies Employee Workstations
* If Jack’s Technologies don’t have workstations, they should purchase workstations for this project. Workstations will be requirement to maintain Artisan’s Coop North’s System.
* Motorola MB7420 Modem
* If Jack’s Technologies isn’t connected to the internet via a modem, they should purchase a modem. We recommend the Motorola MB7420 because it’s the best choice for high internet and download speeds (over 50Mbps).
* Security Cameras and monitors
* Security cameras and monitors are useful in protection against theft. We recommend purchasing this hardware from Authorized Premier Provider (ADT) as they are rated one of the best security companies in the business.
* Router or Ethernet
* A router or Ethernet cable is recommended at Jack’s Technologies to maintain Artisan’s Coop North’s System. Either a router or Ethernet cable will work, although we recommend router since it’s wireless.
* Equipment Locks and Bolts
* Locks and Bolts are useful in protect against theft. They should be used to lock down servers, and computer locks should be used to secure workstations.
* Key Cards
* Key Cards are recommended as protect against equipment threat. This way, only authorized personnel will have access to the building, server rooms, and work stations. Again, we recommend ADT for the purchasing and installation of this hardware.

Required Software Components

* Windows 10, Linux, MacOS
* Jack’s Technologies should be equipped with the latest operating systems. They should also run frequent updates to make sure they’re software is running at best possible capacity.
* Android, iOS
* Jack’s Technologies should also be equipped with the latest operating systems for smartphones. Again, we recommend that they run frequent updates on these operating systems.
* Anti-Virus Software
* Employee workstations must have anti-virus software installed, to prevent damage or loss of data. We recommend purchasing this software from McAfee, Norton, or Kaspersky.
* Password Software
* Jack’s Technologies workstations should be equipped with strong password software to prevent unauthorized access. We recommend purchasing this software from LastPass or KeePass.
* Anti-Spyware Software
* Jack’s Technologies should have installed Anti-Spyware software on all work stations to prevent theft of protected data. We recommend McAfee or Norton for the purchasing of this software.
* Data Encryption Software
* Data Encryption Software is required for Artisan’s Coop North’s System to protect sensitive data such as customer and craftspeople information. We recommend purchasing this software from Folder Lock or AxCrypt Premium.

**Security Plan**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Threats/Components | Fire | Flood | Power Loss | Virus | Theft | Unauthorized Access |
| Client Workstations and other hardware | 1, 2 | 1 | 1, 3 | 10,11,14 | 5,6,7 | 4,5,6,7,8,9,13,14 |
| Servers | 1, 2, 12 | 1, 12 | 1, 3, 12 | 10, 11, 13, 14 | 5, 6, 7 | 4, 5, 6, 8, 9, 13, 14 |
| People | 1, 2 | 1 | 1 | N/A | N/A | 1 |
| Software | N/A | N/A | 12 | 10, 12, 14 | 8, 9, 12, 13, 14 | 8, 9, 12, 13, 14 |

**Controls:**

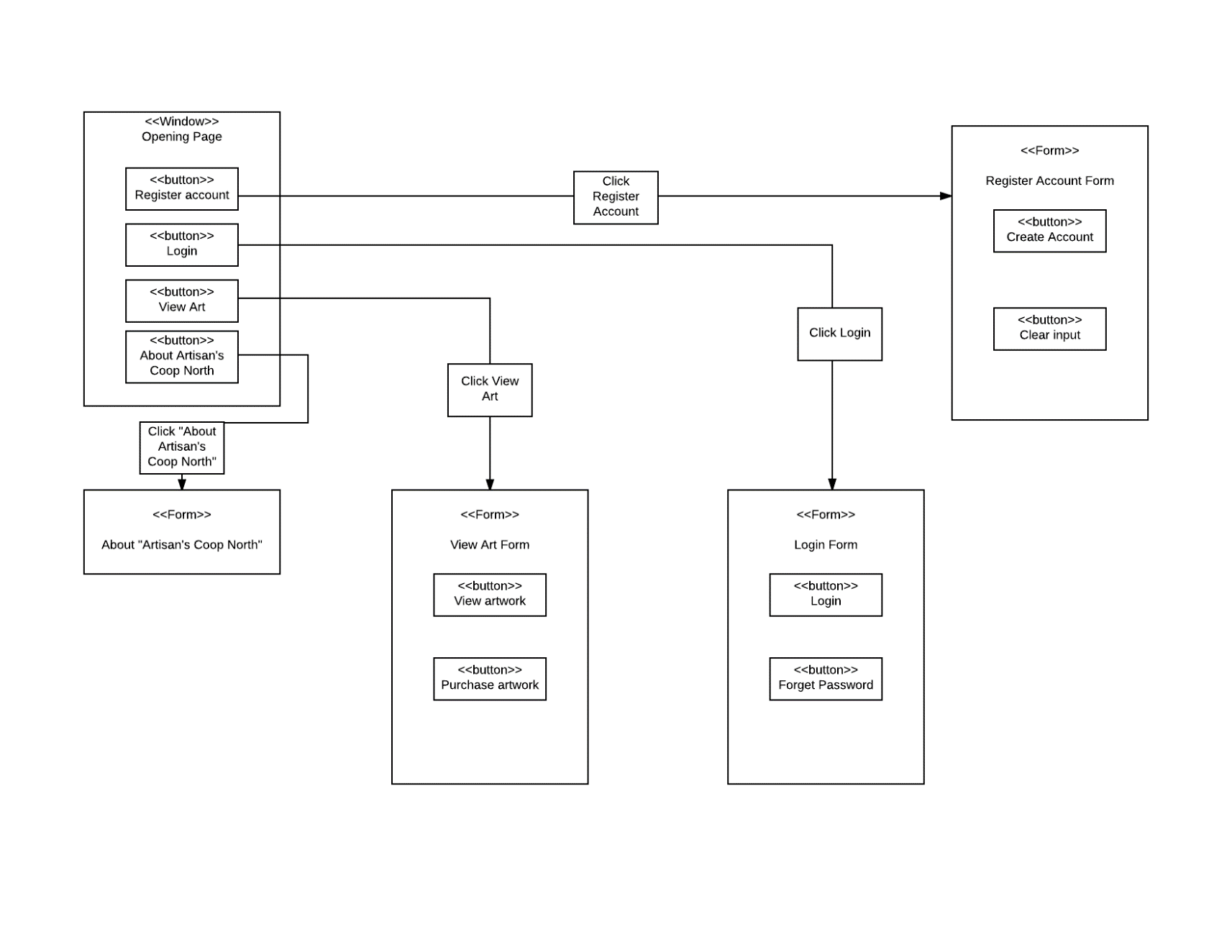
1. Disaster recovery plans
2. Fire alarms and sprinklers in each room of building (that doesn’t have hardware that can be affected by water, such as computers and servers)
3. UPS (Uninterrupted Power Supply)
4. Key card access
5. Lock and Bolts equipment
6. Security Cameras and monitors
7. Insurance
8. Data encryption
9. Password Software
10. Anti-Virus software
11. Anti-Spyware Software
12. Backup Server
13. Firewalls around application and database servers
14. User training

**User-Interface**

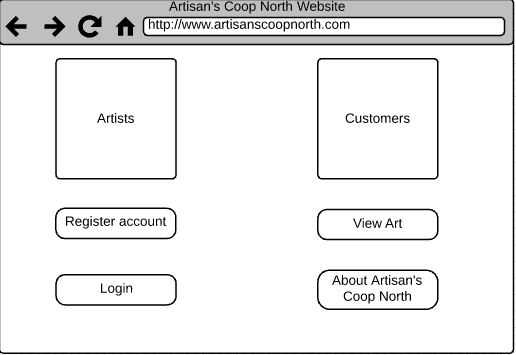
**User-Interface Requirements and Constraints**

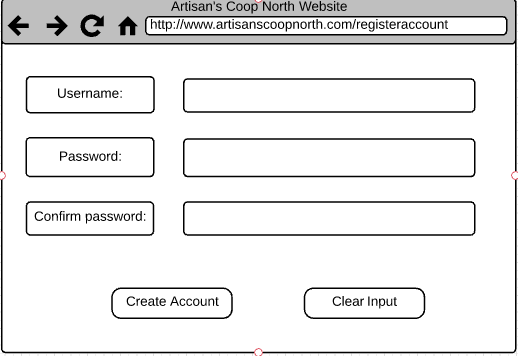
This section gives an outline for user-interface design. A Windows Navigation Diagram, which shows how to navigate between screens, and a Forms: Screen/User-Interaction Design, which shows Artisan’s Coop North’s System visual components and screens. The Reports: Printed Output Design section will show an example of a sale transaction completed by this application.

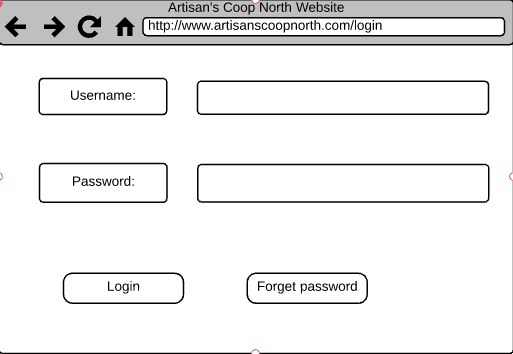
**Windows Navigation Diagram**

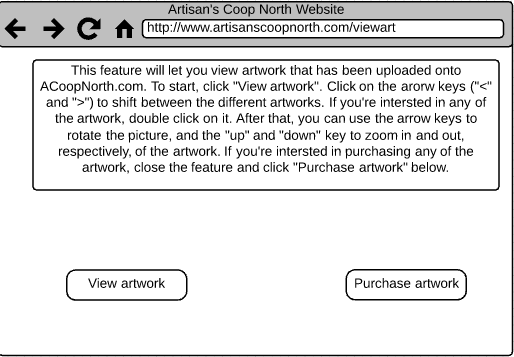
****

**Forms: Screen/User-Interaction Design**









**4.3 Reports: “Printed Output” Design**

Artisan’s Coop North

<Report Name>

<Time Period of Report>

Customer Report:

Output Name (Last, First)

Artwork Report

Artwork name; price

Artwork name; price

Total Profit: $price

Customer Information

Name

Email

Address

Phone number

Artist Information

Name

Phone Number

Email

**V. Appendices**

**Bibliography/References**

Big Data Management Software, www.capterra.com/sem-compare/big-data-software?headline=Big Data Management Software&utm\_source=bing&utm\_medium=cpc&utm\_term=%2Bdatabase %2Bsoftwares&gclid=CJjPwv7E9NcCFcydfgodwKMHEA&gclsrc=ds.

Dennis, Alan, Barbara Haley Wixom, David Tegarden, and Elaine Seeman. Systems Analysis & Design: An Object-Oriented Approach with UML. 5th ed. Hoboken, NJ: Wiley, 2015. Print.

Khandelwal, Swati. “Best Password Manager - For Windows, Linux, Mac, Android, iOS and Enterprise.” The Hacker News, 30 July 2016, thehackernews.com/2016/07/best-password-manager.html.

Pfeiffer, William Sanborn. Pocket Guide to Technical Communication. 5th ed. Upper Saddle River, NJ: Prentice Hall, 2011. Print.

Shipley, Renee. “Best Encryption Software 2017 - Encrypt Files on Windows PCs.” TopTenReviews, 13 Mar. 2017, www.toptenreviews.com/software/security/best-encryption-software/.

**Supporting Documentation**

**Server**



A server is a computer that provides data to other computers. It can serve data through a local area network (LAN) or a wide area network (WAN) over the internet. There are multiple types of servers (web servers, file servers, etc.).

**Router**



A router is piece of hardware that routes data from a local area network to another network location. Only allows authorized machines to connect to other computer systems, and keeps log files of all local area network activity.

**Modem**



A modem is a piece of hardware that allows a computer or other devices, such as a router, to connect to the internet. It converts an analog signal from a cable wire to a digital signal, and converts outgoing digital data from a computer to an analog signal.