**Software Requirements Specification (SRS)**

By Brandon Gibson, Nhan Nguyen, Michael Glum, Nhi Nguyen, and Ryan Cavanaugh

**Table of Contents**

*Introduction………………………………………………………………………………………….pg. 3*

*Overview…………………………………………………………………………………………….pg. 3*

*Pages………………………………………………………………………………………………. pg. 4*

*Database……………………………………………………………….………………………….. pg. 4*

*Stored Procedures……………………………………………………….……………………….. pg. 5*

*Session Variables…………………………………………..…………………………………….. pg. 7*

*Use Case Diagram……………………..………………………………………………………… pg. 8*

*Class Diagram…………………………………………………………………………………….. pg. 9*

*State Chart Diagram……………………………………………………………………………… pg. 10*

*Sequence Diagram…………………………………..…………………………………………… pg. 11*

**Introduction**

**Purpose**

The purpose of this project is to create an online marketplace that users can effectively sell and buy items from each other. Many people have items they want to sell, and many people wish to buy used items for much less money than new items.

**Scope**

The scope of this project includes making an interface for users to buy, sell, and login to accounts. The project will also create and use a session to keep users logged in throughout the website and use a SQL Server database to store all information.

**Requirements**

1. Allow users to Create Account
2. Returning users can log in to account
3. Sellers can make listings for products
4. Sellers can add up to 5 pictures for each listing
5. Admins can review listings and approve or deny them before being visible to other users
6. Admins can promote other users to an admin status
7. Users can view their account information that is stored
8. Users can view all approved listing on a “buy” page
9. User has a cart that they can review and remove items from
10. User has a wishlist that they can review and remove items from, but are not intending on buying as of yet
11. Users can filter listings on the “buy” page by price
12. Users can filter listings on the “buy” page by category

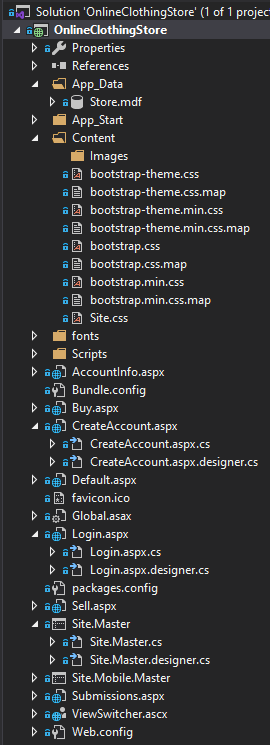
**Overview**

**The Project**

The product should be a website that can function as an online marketplace. This website should allow users to create accounts, sell items, buy items, view cart, and view Wishlist. The website also should have admins that approve listings before being displayed as public.

**The Specifications**

This project will be written in ASP.NET framework, and in the Visual Studio 2019 IDE. This utilizes standard HTML5 and CSS for the website and comes with bootstrap imported. For website functionality, ASP.NET is C# and uses this to add website functionality instead of JavaScript. The database is written in SQL Server but accessing the database will be done through stored procedures.

**Pages**

Each page is listed with the extension .aspx. This file contains all the HTML to create the webpage it is for. Each .aspx file represents a webpage on the website. Each .aspx file contains a dropdown the includes an .aspx.cs file. This file is the C# functionality. In the content folder, there is a folder for images that contain the listing’s images and bootstrap themes. The Store database is also located in the App\_Data folder. The current pages in the project are:

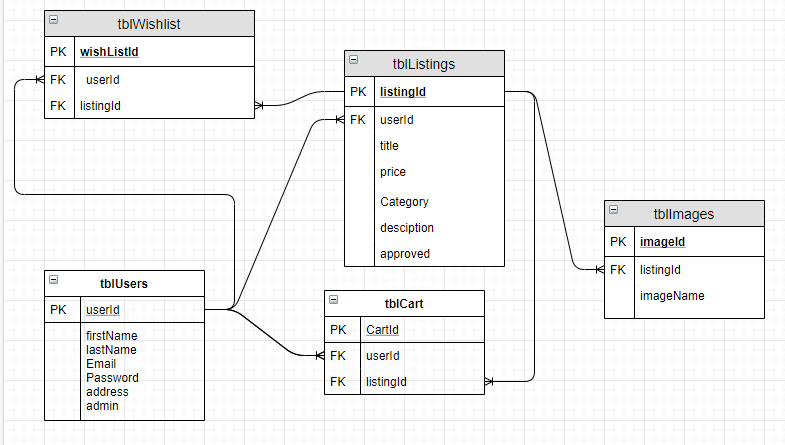
* **AccountInfo** – This is what is shown in place of create account page once the user is logged in
* **Buy** – This is the page for the buy page to display all the listings
* **CreateAccount** – This page is for the create account page
* **Default** – This is the home page
* **Login** – The login page for existing users
* **Sell** – The sell page to make a listing
* **Site.Master** – This page covers all pages, so things such as the footer and page select hotbar at the top of the page are covered here.
* **Submissions** – This is the page to review submissions if the user is an admin.
* **Cart** – Page to show the cart of the logged in user

**Database**

The database in written in SQL server. The database is in the APP\_DATA folder in the project. To view the project tables and schema, double click on the Store.mdf file and it will open it up in a tab called “Server Explorer.” You can then view the tables and procedures of the database. The tables in the database are

* tblCart – represents the cart(s) of users
* tblImages – images for listings
* tblListings – all listings for items
* tblUsers – users for the website
* tblWishList – represents the wishlist(s) of users

The schema of the database is as follows:



**Stored Procedures**

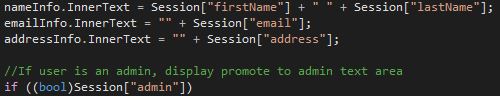
Almost all interaction to the database inside the program is done through stored procedures. These are prewritten procedures that add, remove, update, or return information from inside the database to be used in the program. The stored procedures of our database and their input values are as follows:

* **spAddListing** – adds a listing to the database
  + userId – INT
    - userId of author creating listing
  + title – varchar(100)
  + price – INT
  + category – VARCHAR(50)
  + description (optional) – VARCHAR(1000)
  + success (output) – INT (returns listingId if listing was created in database, or 0 if failed)
* **spAddRemoveCart** – adds or removed item from cart
  + userId – INT
    - Can leave this empty or 0 if removing item from cart
    - userId of cart owner
  + listingId – INT
    - listingId of listing being added to cart
* **spAddRemoveImages** – adds or removes images
  + imageId – INT
    - imageId of image to remove
    - Set to 0 if adding image
  + listingId – INT
    - listingId of listing adding images to
  + imageName – VARCHAR(50)
    - filename of image to set as img source in program
* **spAddRemoveWishlist** – adds or removed item from wishlist
  + userId – INT
    - Can leave this empty or 0 if removing item from wishlist
    - userId of wishlist owner
  + listingId – INT
    - listingId of listing being added to wishlist
* **spApproveRemoveSubmission** – approve or delete listing upon admin review
  + listingId – INT
    - listindId of listing being reviewed
  + approved – BIT
    - 1 for approved
    - 0 for denied
* **spCreateAccount** – creates user in the database
  + firstName – VARCHAR(100)
  + lastName – VARCHAR(100)
  + email – VARCHAR(100)
  + password – NVARCHAR(100)
  + address – VARCHAR(100)
  + Success (output) – BIT
    - 1 for success
    - 0 for failed to create account
* **spGetCart** – gets cart contents of a user
  + userId
    - userId of user logged in
* **spGetImages** – gets images for specific listing
  + listingId – INT
    - listingId of listing to get images for
* **spGetListings** – Gets all approved listings
  + returns a select statement
* **spGetSubmissions** – Gets first unapproved listing to be reviewed
* **spGetWishlist** – gets wishlist contents of a user
  + userId
    - userId of user logged in
* **spPromoteToAdmin** – makes a user an admin in database
  + email – VARCHAR(100)
    - email of user being promoted
  + success (output) – BIT
    - 1 for successful
    - 0 if failed
* **spValidateLogin** – validate if credentials match user in database
  + email – VARCHAR(100)
  + password – NVARCHAR(100)
  + success (output) – BIT
    - 1 for successful
    - 0 if failed

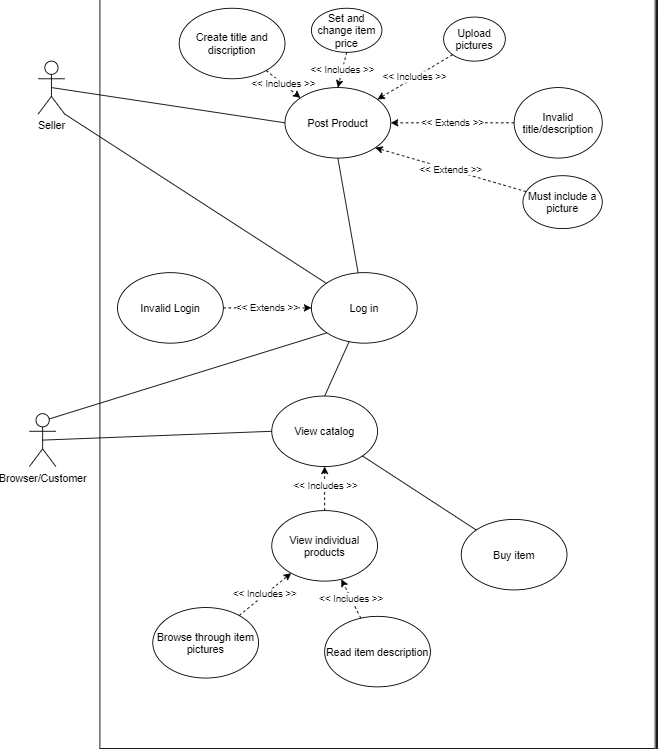
**Session Variables**

Session variables are in the program that keeps variables defined in the user’s session. So once a user logs in, there account information is saved into session variables. If no user is logged in, these variables are null. Session variables are not defined as a specific type of data type, so knowing what data is stored in each variable is necessary to use them. The session variables in our program, and their datatype, are as follows:

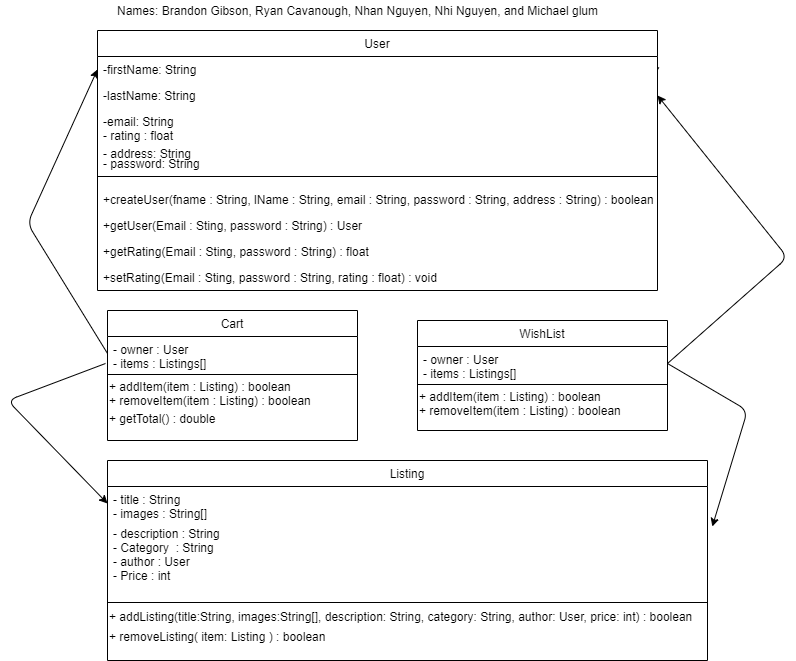
* Session["userId"] – int
  + userId of user that is logged in
* Session["email"] – string
  + Email of user logged in
* Session["firstName”] – string
  + First name of user logged in
* Session["lastName”] – string
  + Last name of user logged in
* Session["Address"] – string
  + Address of user logged in
* Session["admin"] – Boolean
  + If user logged in is admin or not

To use these session variables, simple just put the Session[“”] with the session variable name inside the quotations. Since these variables can be any datatype, when using them, you must cast them to the appropriate datatype as listing above. An example of using these in the program are:

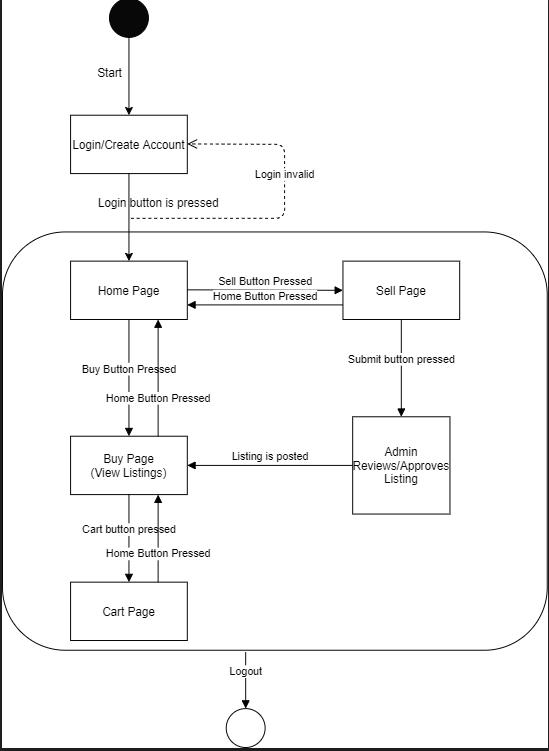
**Use Case Diagram**



**Class Diagram**



**State Chart Diagram**

****

**Sequence Chart Diagram**

