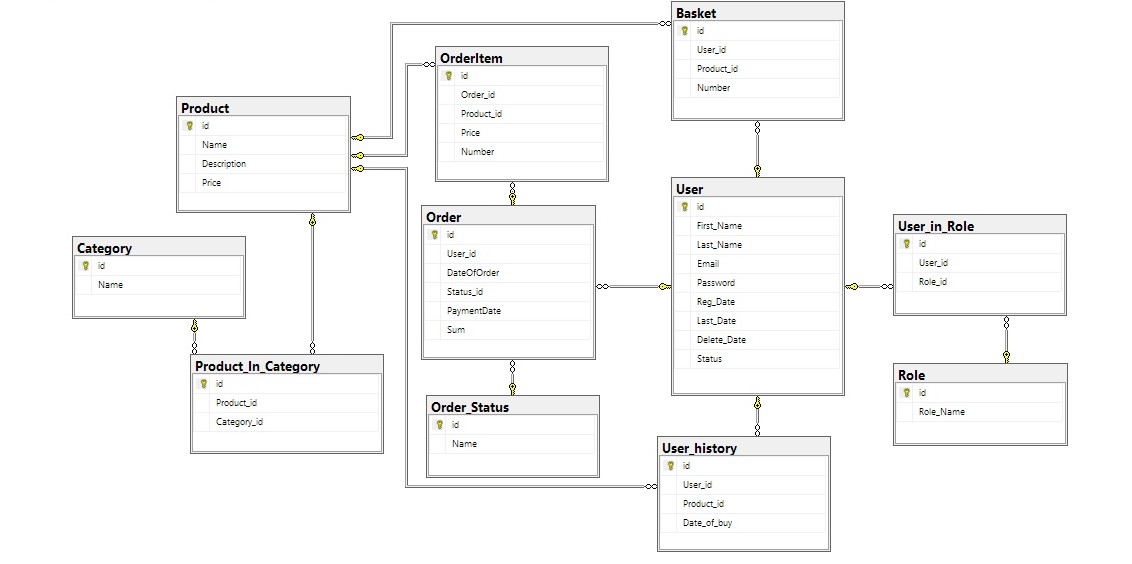
**Documentation for WebStore**

1. **Design Documentation**
   1. **DB Scheme:**



**User** table is used for storing user data

**Role** table store a list of user roles that used in app

**User\_in\_Role** table stores data for role matching to the user

**Basket** table is used to save current user cart

**User\_history** table where stores for storing products from completed orders

**Product** tablestore information about products

**Category** table that contain products categories

**Product\_in\_Category** stores data for category matching to the product

**Order** table that store main data about user’s oders

**OrderItem** table that store information about products for each order

**Order\_Status** table is used for storing different order statuses

* 1. **Application modules description**
     1. **DB Access Layer**

Access to the database Is provided by classes described in the test.cs file in Models directory. For exchange of data with the database is used Linq. These classes consist of properties that repeat the database scheme and they are labelled with a special attributes to correlate these models with tables. Also in this file contains DbContext class that is inherited from DataContext class from Linq. This class is used for making a connection to database and also there are properties for tables in the database contains.

.

* + 1. **Authentication and authorization**

For authentication is used class Auth that is described in Auth.cs in Models directory. This class implements IAuth interface and contains four methods Login, Logoff, SetCookie and curUser. Login method is used to check user availability with the transmitted data in the database. If user in base for the user is formed FormsAuthenticationTicket and then method SetCookie is called. This method created Cookie file and add it to the request of server. Method Logoff is used to log out from the site, it’s delete the cookie file for the current user and update the date of last visit for user. Method curUser is used to receive id of current user from AuthCookie file.

For authorization functions in app used class called CustomAuthorizeAttribute. This class is inherited from AuthorizeAttribute. This class contains bool property AllowAnonymous for anonymous user definitions.

In this class two methods are defined. This method I used for definition of user role and as a result of work it returns bool value depending on whether the role of the current user is relevant to the role required to access a particular application module. Also in this class defined method that process requests from unauthorized users to access any forbidden parts of site, this method redirect users request on login page. As the help class for this method is defined class Roles. There are two methods there first one called GetRoles for receiving a list of roles from database and the second one is used to determine whether the current user has a certain role.

* + 1. **Controllers**

There are four controllers in app. AuthController is the controller that is used for authorization and authenticationfunctions. For methods in this controller defined access level, that allow every user to access this controller. Methods of this controller includes Login method, when user is trying to login into app controller return view for login page. Another Login method is calling when there is a post request. This method receive LoginViewModel object as argument, validate received model, add model errors if any fields don’t responding to the model requirements and then Login method from Auth class is calling, after that controller’s method check if login success calling curUser method which in case of success redirect user to index view, else it add model error, that entered data has mistakes.

The next method of AuthController when it is called it check if current user is not anonymous and after that call Logout method from Auth class and redirect user to the Index view.

There is Registration method in this controller that return registration view. Also there is Registration method that is called by post request, it is receive a RegViewModel object, check if the model is valid, after that it create new User with the received data, the role for this user Is defined as User. After that is calling CreateUser method from Admin class with the created User as argument. After that method Login is calling from this controller for user that was created.

Another controller is AdminController it contains methods for administrator to manage users and their roles.

There is method AdminPanel that receives all users from the database and return AdminPanel view for the received users. Also controller contain method for AdminPanel that received post requests with the user id and redirect administrator to the method of editing selected user called Edit. This method create user with current id in the database and return View for editing him. There is Edit method that receives post requests from Edit views with the AdminPanelViewModel as argument. This method call method SaveChanges in the Admin class and redirect to the AdminPanel view. Also there is method that allow to administrator to freeze user account. Freeze method receives post request with the user id and after that call Freeze method from Admin class and after that it makes redirect to AdminPanel view. Also there is Restore method in controller that allow administrator to restore user account if it was freeze. This method receives user by id and call Restore method from Admin class with it user and redirect to AdminPanel view. Also administrator has opportunity to create user, for there are two methods CreateNewUser, first of the just return CreateNewUser view, the second one receives AdminCreateUserViewModel object by post request and call method CreateNewUser from Admin class.

HomeController has the next methods. Index method is used for displaying information about products on the main page of the store. It receives data from GetItems method of UserMethods class and return View with this data. There is another Index method that receives post requests from the view with additional product id parameter, to this method has access only registered users with User role. It is call AddToCart metod from UserMethods class. To see users cart is exist Busket method in the controller, when it is called it is call ViewBusket method from UserMethods class. To make an order there is MakeOrder method that call method in UserMethods MakeAnOrder. History method is used to see list of products that user has bought ever, it is receives data from ViewHistory method in UserMethods class and return View with this data. To see current users orders there is Orders method that use ViewOrder method from to receive information and after that it return view. To change user data is used ChangeUserData method, which model of current user by id and after that return view with this user. Also there is ChangeUserData that receives post request with the User model, it is call ChangeUserData with the received model as parameter.

SalesController implements functions of the seller. SellerePanel method is used for display all the products from the database on view. There is SellerPanel method that receives post requests it is used to receive id of selected product and after that it is called Edit method that receives the product model of selected product and after that it is calling Edit method with this product. Edit method return the view for this model. Edit method that receives post requests is used for calling SaveChanges method from SalesPerson class. AddNewItem methods Is used for adding new product to the database first of them returns view, the second one processes the post request and call method AddItem that adds item to the database. Seller has an opportunity to add new categories to the database. For it there are two methods, one receives the list of all product categories from database and return the view for them, the second one is processes post requests and call method AddCat with received model. The last function of seller is to see and change the status of users orders. It is provided by two methods. ViewOrders method is used for displaying all the orders and order items. ChangeStatus method receives get requests and with the string of new status and id of product for which this status should be set and after that it is call ChangeStatus method with these parameters.

ErrorControler contains methods for calling error views.

* + 1. **Classes to Provide different roles function**

In app defined classes to provide different role functions. Methods of this classes called from controllers.

Methods for authentication and authorization that used in Auth controller is described in Auth class, they were described in paragraph 1.2.2.

**Methods for main functions of webstore is described in UserMethods class.**

* GetItems method is used to receive all products from db and return them as IEnumearable<Product> object.
* AddToCart method receives id of product and add this product to the cart of the current user in db. If this item is already in user cart, it just increases the value Number by 1.
* ViewBasket method is used receive a List of BusketViewModel objects. It is receive a Basket objects from db for current user and after that for every received object it is create a BusketViewModel and add it to the list.
* MakeAnOrder method Is used to place an order. This method receives all products from db and add new order in Order table in database and all the products that relate to this order add to the OrderItem table. After that items in cart is cleared.
* ViewHistory method is used to receive products that current user had ever bought. It is receives data from UserHistory table in database. Then it is formed HistoryViewModel objects and add them to the list and return this list.
* ViewOrders method is need to see all the orders that is not completed yet. It is received all user orders and order items and formed from them OrderViewModel object and return it.
* ChangeUserData method Is used for changing user data by user. It is receives User model as parameter and update information in db for current user according to the received model.

**Methods that provides administrator functions is described in Admin class.**

* ReceiveUsers method is used to receive all users from database and form AdminPanelViewModel object for each of users and add it to the list that is returning by method.
* ReceiveUserById method receive parameter user id and find the user in with current id in database, form AdminPanelViewModel object for this user and return it.
* RecieveUserByEmail method is used to find and return user by transmitted to this function email string.
* CreateUser method has two variants. First one takes two parameters User model and id of the role for him, it is insert User model to the User table in database and create a record in table UserInRole with user id and role id. Another variant of this method takes AdminCreateUserViewModel object as parameter, after that it is create User object by received model and create UserInRole object after that it insert them to the appropriate tables in db.
* Freeze method is used to freeze user with id, this method takes as parameter. It is find user by id and change value of Status field on 1.
* Restore method is used by admin to restore user account, it is receives user id, find him in database and change Status value for him on 0.
* SaveChanges method takes User model, find user with the same id in db and update all the fields according to received model and save changes in db.
* ClearBusket method is used to clear cart for current user in database table Basket.
* Restore method takes RestoreViewModel parameter and used for restore password and user account by user, it is update password and user status in database with data from received model.

**Methods to provide seller function described in SalesPerson class.**

* AddItem method is used to add new product to the database. It takes AddItemViewModel object and create new objects Product and Product\_in\_Category by the data in the received model and add data to the Product table and ProductInCategory table in the dstsabase.
* SaveChanges takes as argument SellerPanelViewModel object. After that it is find product in the database that should be changed and change fields in the model according to the received model and save changes in database.
* GetItems method is used to recieve all products from database. It is create a SellerPanelViewmodel objects by the all products in db and add them to the list. In the result this method return list of SellerPanelViewModel objects.
* GetItemById method is need to receive product with certain id. It is find a product in database, form SellerPanelViewModel object and send it as a result.
* AddCat method is used to add new categories to the database. It takes name of category as parameter and after that create Category object and insert it to the database.
* ViewCategories method is used to receive all the categories from the database and it is return IEnumerable object of Category as result.
* GetOrders method is used to receive to all the user’s orders and its items. It create OrdersViewModel object from orders and order items and receive this object as result.
* ChangeStatus method is need to update the status of order, it is receives the name of status to which it should be changed and id of order for which this status should be set. It is find order by id in database and set for it appropriate status.
  + 1. **Models**

Instead of the models that used to work with the database, there are special models that used for displaying and exchanging of data with views.

* LoginViewModel model that is used in the process of user’s logon. This class contain two property. For each property defined attributes that set limitation on the data in these properties.
* RegViewModel model that defined properties for registration of new users. It has attributes for verification of entered data.
* AddItemViewModel contain properties for adding of new products to the database.
* AdminCreateUserViewModel model contained properties for adding new users by administrator to the db.
* AdminPanelViewModel is model that contain properties that is used when admin interact with user’s data (update, edit) in app.
* BusketViewModel contains properties to see products in cart.
* HistoryViewModel is used to see products in user history.
* OrdersViewModel is a model that contains two properties first one of OrderView class.

OrderView contains properties that is need to see or update data about orders. The second one is OrderItemView, this class contain properties to for order items.

* SellerPanelViewModel is used when seller interact with items in database.
  + 1. **Views**
* AdminPanel is used for displaying all user accounts. Also it is form requests to the admin controller for edit, freeze and restore accounts. View based on AdminPanelViewModel.
* CreateNewUser is view with the fields that is need to create a new user by administrator.

It is based on AdminCreateUserViewModel.

* Edit is view for editing users, it is based on AdminPanelViewModel.
* Login is view that is used for logon users to the app. It is based on LoginViewModel.
* Registration is view for registration of new users, based on RegVewModel.
* RestorePassword is view where user can change the password; this view is based on RestoreViewModel.
* AddNewCategory is view for adding new categories to the db by seller. This view is based on IEnumerable<Category> object
* Edit view is used for editing product by seller. View based on SellerPanelViewModel.
* SellerPanel view that is used for displaying all the products in database and also from this form send requests on edit of products. Based on IEnumerable<SellerPanelViewModel >
* ViewOrders view for displaying of all user’s orders. Also this form sent requests on updating of order status. This view based on OrdersViewModel.
  1. **Architecture scheme**

App could be divided on four main parts. First one is database level. This part contain database, db context class and db tables models.

The next part is a part that describes models that used inside the app for more convenient exchange between parts of program, classes that describes such models has names ends with ViewModel.

Next app level is controllers and classes that contain methods for controllers. App contain HomeController, AuthController, AdminController and SalesController and corresponding them help classes UserMethods, Auth, Admin and SalesPerson classes. This level is used for exchange data between database and views. Every controller has different access level.

The last part is presentation level. It has views for different app functions, views call this functions via methods in controllers.

* 1. **Security**

User accesses to different functions are defined in controllers by CustomAuthorizeAttribute with parameters.

AdminController is available only to users with Admin role.

**AuthController** available for anonymous users.

**HomeController** has one method that available to anonymous users; it is used for loading products on the main page of the site. Other methods of the controller available to users whose role is User.

**SalesController** is available for users with role Seller.

* 1. **Mail sender**

Mail sender Is a module that is used for informing users that they have not been in there account for a long time. This module work automatically. Every day it is looking for users whose last visit was over 30 days ago and sent them an email. This module use System.Net.Mail. For sending mail used gmail with email address: testwebstore000@gmail.com and password: webstorepassword. It is starts with the starting of app.

* 1. **Account Disable**

AccountDisable module that used for automatic freezing of accounts that don’t used for two weeks after sending inform email. This module starts every day, it is search users whose last visit was over 44 days ago and change their account status on 1 (Freeze). User has option to restore the account at the last login to the system. It is starts with the starting of app.

* 1. **Logger and error handling**

For handling known exceptions in app used try-catch construction. It is used in methods which use database. If error happens method return null value and make a record in Log file. Controllers methods that receives null value from methods that they used for loading data returns an error message to the view, that the user see.

For handling unknown exceptions or http errors there is Application\_Error method in Global app class. When error happens it is defines the class to which the error belongs. If it is error 404 Not found or 403 Access denied, it is returns relevant views NotFound and AccessDenied. If occurs any unknown errors in app method returns Unknown view, where will be some additional information for further analysis. If take place any http errors except 404 and 403 appears method calls relevant View HttpError. Every error that processed in this method records to the Log file.

Logging in app is provided by Nlog package. Logs main settings are located in Nlog.config file. Log files are created every day, in Log directory. It contains the date of error, level of error, method where this error was appeared, error message, in some methods it contain additional information for example id of user for which error was appeared and the stack trace. It has an opportunity to save log archives for 30 days.

In app writing of logs is organized directly in the classes where error handling occurs.