ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет програмування та комп'ютерних
і телекомунікаційних систем

Кафедра інженерії програмного забезпечення

Лабораторна робота № 6

з дисципліни «Програмування Інтернет»
на тему:

«Авторизація та аутентифікація в MVC 4. Аутентифікація Windows. Аутентифікація форм. Налаштування використання SimpleMembershipProvider. Використання універсальних провайдерів..

Виконав:

студент 3 курсу, групи ІПЗ-18-1 _____ В.В.Охота
(підпис) (Ініціали, прізвище)

Перевірив: _____ О.М. Яшина
(підпис)

Хмельницький – 2020 р.

Мета. Отримати навики розробки власної логіки валідації моделі.

Завдання. Надбати основні поняття розробки механізму авторизації і аутентифікації.

Хід роботи

У даній роботі було допрацьовано проєкт який був розроблений у лабораторній роботі №5.

1. Було створено три нових моделі
   1.1. AccountViewModels

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace CarShop.Models
{
    public class ExternalLoginConfirmationViewModel
    {
        [Required]
        [Display(Name = "Адрес электронной почты")]
        public string Email { get; set; }
    }

    public class ExternalLoginListViewModel
    {
        public string ReturnUrl { get; set; }
    }

    public class SendCodeViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }
        public string ReturnUrl { get; set; }
        public bool RememberMe { get; set; }
    }

    public class VerifyCodeViewModel
    {
        [Required]
        public string Provider { get; set; }

        [Required]
        [Display(Name = "Код")]
        public string Code { get; set; }
        public string ReturnUrl { get; set; }

        [Display(Name = "Запомнить браузер?")]
        public bool RememberBrowser { get; set; }

        public bool RememberMe { get; set; }
    }

    public class ForgotViewModel
    {
        [Required]
        [Display(Name = "Адрес электронной почты")]
        public string Email { get; set; }
    }

    public class LoginViewModel
```

```
    {
        [Required]
        [Display(Name = "Адрес электронной почты")]
        [EmailAddress]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [Display(Name = "Запомнить меня")]
        public bool RememberMe { get; set; }
    }

    public class RegisterViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Адрес электронной почты")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "Значение {0} должно содержать не менее {2}
символов.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Подтверждение пароля")]
        [Compare("Password", ErrorMessage = "Пароль и его подтверждение не совпадают.")]
        public string ConfirmPassword { get; set; }
    }

    public class ResetPasswordViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Адрес электронной почты")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "Значение {0} должно содержать не менее {2}
символов.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Подтверждение пароля")]
        [Compare("Password", ErrorMessage = "Пароль и его подтверждение не совпадают.")]
        public string ConfirmPassword { get; set; }

        public string Code { get; set; }
    }
```

1.2. IdentityModels

```
using System.Data.Entity;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;

namespace CarShop.Models
{
```

```
    // В профиль пользователя можно добавить дополнительные данные, если указать больше
свойств для класса ApplicationUser. Подробности см. на странице
https://go.microsoft.com/fwlink/?LinkID=317594.
    public class ApplicationUser : IdentityUser
    {
        public async Task<ClaimsIdentity>
GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)
        {
            // Обратите внимание, что authenticationType должен совпадать с типом,
определенным в CookieAuthenticationOptions.AuthenticationType
            var userIdentity = await manager.CreateIdentityAsync(this,
DefaultAuthenticationTypes.ApplicationCookie);
            // Здесь добавьте утверждения пользователя
            return userIdentity;
        }
    }

    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext()
            : base("DefaultConnection", throwIfV1Schema: false)
        {
        }

        public static ApplicationDbContext Create()
        {
            return new ApplicationDbContext();
        }
    }

}
```

1.3. ManageViewModels

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNet.Identity;
using Microsoft.Owin.Security;

namespace CarShop.Models
{
    public class IndexViewModel
    {
        public bool HasPassword { get; set; }
        public IList<UserLoginInfo> Logins { get; set; }
        public string PhoneNumber { get; set; }
        public bool TwoFactor { get; set; }
        public bool BrowserRemembered { get; set; }
    }

    public class ManageLoginsViewModel
    {
        public IList<UserLoginInfo> CurrentLogins { get; set; }
        public IList<AuthenticationDescription> OtherLogins { get; set; }
    }

    public class FactorViewModel
    {
        public string Purpose { get; set; }
    }

    public class SetPasswordViewModel
    {
        [Required]
        [StringLength(100, ErrorMessage = "Значение {0} должно содержать символов не
менее: {2}.", MinimumLength = 6)]
        [DataType(DataType.Password)]
```

```
        [Display(Name = "Новый пароль")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Подтверждение нового пароля")]
        [Compare("NewPassword", ErrorMessage = "Новый пароль и его подтверждение не
совпадают.")]
        public string ConfirmPassword { get; set; }
    }

    public class ChangePasswordViewModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Текущий пароль")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "Значение {0} должно содержать символов не
менее: {2}.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Новый пароль")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Подтверждение нового пароля")]
        [Compare("NewPassword", ErrorMessage = "Новый пароль и его подтверждение не
совпадают.")]
        public string ConfirmPassword { get; set; }
    }

    public class AddPhoneNumberViewModel
    {
        [Required]
        [Phone]
        [Display(Name = "Номер телефона")]
        public string Number { get; set; }
    }

    public class VerifyPhoneNumberViewModel
    {
        [Required]
        [Display(Name = "Код")]
        public string Code { get; set; }

        [Required]
        [Phone]
        [Display(Name = "Номер телефона")]
        public string PhoneNumber { get; set; }
    }

    public class ConfigureTwoFactorViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }
    }
}
```

2. Також було створено два контролера

2.1. AccountController

```
[Authorize]
    public class AccountController : Controller
    {
        private ApplicationSignInManager _signInManager;
        private ApplicationUserManager _userManager;

        public AccountController()
```

```
        {
        }

        public AccountController(ApplicationUserManager userManager,
ApplicationSignInManager signInManager )
        {
            UserManager = userManager;
            SignInManager = signInManager;
        }

        public ApplicationSignInManager SignInManager
        {
            get
            {
                return _signInManager ??
HttpContext.GetOwinContext().Get<ApplicationSignInManager>();
            }
            private set
            {
                _signInManager = value;
            }
        }

        public ApplicationUserManager UserManager
        {
            get
            {
                return _userManager ??
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
            }
            private set
            {
                _userManager = value;
            }
        }

        //
        // GET: /Account/Login
        [AllowAnonymous]
        public ActionResult Login(string returnUrl)
        {
            ViewBag.ReturnUrl = returnUrl;
            return View();
        }

        //
        // POST: /Account/Login
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }

         // Сбои при входе не приводят к блокированию учетной записи
         // Чтобы ошибки при вводе пароля инициировали блокирование учетной записи,
замените на shouldLockout: true
         var result = await SignInManager.PasswordSignInAsync(model.Email,
model.Password, model.RememberMe, shouldLockout: false);
         switch (result)
         {
             case SignInStatus.Success:
                 return RedirectToLocal(returnUrl);
             case SignInStatus.LockedOut:
                    return View("Lockout");
```

```
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new { ReturnUrl = returnUrl,
RememberMe = model.RememberMe });
                case SignInStatus.Failure:
                default:
                    ModelState.AddModelError("", "Неудачная попытка входа.");
                    return View(model);
            }
        }


        //
        // GET: /Account/Register
        [AllowAnonymous]
        public ActionResult Register()
        {
            return View();
        }

        //
        // POST: /Account/Register
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Register(RegisterViewModel model)
        {
            if (ModelState.IsValid)
            {
                var user = new ApplicationUser { UserName = model.Email, Email =
model.Email };
                var result = await UserManager.CreateAsync(user, model.Password);
                if (result.Succeeded)
                {
                    await SignInManager.SignInAsync(user, isPersistent:false,
rememberBrowser:false);

                    // Дополнительные сведения о включении подтверждения учетной записи
и сброса пароля см. на странице https://go.microsoft.com/fwlink/?LinkID=320771.
                    // Отправка сообщения электронной почты с этой ссылкой
                    // string code = await
UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
                    // var callbackUrl = Url.Action("ConfirmEmail", "Account", new {
userId = user.Id, code = code }, protocol: Request.Url.Scheme);
                    // await UserManager.SendEmailAsync(user.Id, "Подтверждение учетной
записи", "Подтвердите вашу учетную запись, щелкнув <a href=\"" + callbackUrl +
"\">здесь</a>");

                    return RedirectToAction("Index", "Home");
                }
                AddErrors(result);
            }

            // Появление этого сообщения означает наличие ошибки; повторное отображение
формы
            return View(model);
        }
        //
        // POST: /Account/LogOff
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LogOff()
        {
            AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);
            return RedirectToAction("Index", "Home");
        }

        //
        // GET: /Account/ExternalLoginFailure
```

```
[AllowAnonymous]
public ActionResult ExternalLoginFailure()
{
    return View();
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        if (_userManager != null)
        {
            _userManager.Dispose();
            _userManager = null;
        }

        if (_signInManager != null)
        {
            _signInManager.Dispose();
            _signInManager = null;
        }
    }

    base.Dispose(disposing);
}

#region Вспомогательные приложения
// Используется для защиты от XSRF-атак при добавлении внешних имен входа
private const string XsrfKey = "XsrfId";

private IAuthenticationManager AuthenticationManager
{
    get
    {
        return HttpContext.GetOwinContext().Authentication;
    }
}

private void AddErrors(IdentityResult result)
{
    foreach (var error in result.Errors)
    {
        ModelState.AddModelError("", error);
    }
}

private ActionResult RedirectToLocal(string returnUrl)
{
    if (Url.IsLocalUrl(returnUrl))
    {
        return Redirect(returnUrl);
    }
    return RedirectToAction("Index", "Home");
}

internal class ChallengeResult : HttpUnauthorizedResult
{
    public ChallengeResult(string provider, string redirectUri)
        : this(provider, redirectUri, null)
    {
    }

    public ChallengeResult(string provider, string redirectUri, string userId)
    {
        LoginProvider = provider;
        RedirectUri = redirectUri;
        UserId = userId;
    }
```

```
            public string LoginProvider { get; set; }
            public string RedirectUri { get; set; }
            public string UserId { get; set; }

            public override void ExecuteResult(ControllerContext context)
            {
                var properties = new AuthenticationProperties { RedirectUri =
RedirectUri };
                if (UserId != null)
                {
                    properties.Dictionary[XsrfKey] = UserId;
                }

context.HttpContext.GetOwinContext().Authentication.Challenge(properties,
LoginProvider);
            }
        }
        #endregion
    }
}
```

2.2. ManageController

```
  [Authorize]
    public class ManageController : Controller
    {
        private ApplicationSignInManager _signInManager;
        private ApplicationUserManager _userManager;

        public ManageController()
        {
        }

        public ManageController(ApplicationUserManager userManager,
ApplicationSignInManager signInManager)
        {
            UserManager = userManager;
            SignInManager = signInManager;
        }

        public ApplicationSignInManager SignInManager
        {
            get
            {
                return _signInManager ??
HttpContext.GetOwinContext().Get<ApplicationSignInManager>();
            }
            private set
            {
                _signInManager = value;
            }
        }

        public ApplicationUserManager UserManager
        {
            get
            {
                return _userManager ??
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
            }
            private set
            {
                _userManager = value;
            }
        }

        //
```

```
        // GET: /Manage/Index
        public async Task<ActionResult> Index(ManageMessageId? message)
        {
            ViewBag.StatusMessage =
                message == ManageMessageId.ChangePasswordSuccess ? "Ваш пароль успішно
змінено."
                : message == ManageMessageId.SetPasswordSuccess ? "Пароль задан."
                : message == ManageMessageId.SetTwoFactorSuccess ? "Настроен поставщик
двухфакторной проверки подлинности."
                : message == ManageMessageId.Error ? "Произошла ошибка."
                : message == ManageMessageId.AddPhoneSuccess ? "Ваш номер телефона
добавлен."
                : message == ManageMessageId.RemovePhoneSuccess ? "Ваш номер телефона
удален."
                : "";

            var userId = User.Identity.GetUserId();
            var model = new IndexViewModel
            {
                HasPassword = HasPassword(),
                PhoneNumber = await UserManager.GetPhoneNumberAsync(userId),
                TwoFactor = await UserManager.GetTwoFactorEnabledAsync(userId),
                Logins = await UserManager.GetLoginsAsync(userId),
                BrowserRemembered = await
AuthenticationManager.TwoFactorBrowserRememberedAsync(userId)
            };
            return View(model);
        }
        //
        // GET: /Manage/ChangePassword
        public ActionResult ChangePassword()
        {
            return View();
        }

        //
        // POST: /Manage/ChangePassword
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> ChangePassword(ChangePasswordViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var result = await
UserManager.ChangePasswordAsync(User.Identity.GetUserId(), model.OldPassword,
model.NewPassword);
            if (result.Succeeded)
            {
                var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null)
                {
                    await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
                }
                return RedirectToAction("Index", new { Message =
ManageMessageId.ChangePasswordSuccess });
            }
            AddErrors(result);
            return View(model);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing && _userManager != null)
            {
                _userManager.Dispose();
```

```
                _userManager = null;
            }

            base.Dispose(disposing);
        }



        #region Вспомогательные приложения
        // Используется для защиты от XSRF-атак при добавлении внешних имен входа
        private const string XsrfKey = "XsrfId";

        private IAuthenticationManager AuthenticationManager
        {
            get
            {
                return HttpContext.GetOwinContext().Authentication;
            }
        }

        private void AddErrors(IdentityResult result)
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError("", error);
            }
        }

        private bool HasPassword()
        {
            var user = UserManager.FindById(User.Identity.GetUserId());
            if (user != null)
            {
                return user.PasswordHash != null;
            }
            return false;
        }

        private bool HasPhoneNumber()
        {
            var user = UserManager.FindById(User.Identity.GetUserId());
            if (user != null)
            {
                return user.PhoneNumber != null;
            }
            return false;
        }

        public enum ManageMessageId
        {
            AddPhoneSuccess,
            ChangePasswordSuccess,
            SetTwoFactorSuccess,
            SetPasswordSuccess,
            RemoveLoginSuccess,
            RemovePhoneSuccess,
            Error
        }

#endregion
    }
}
```

3. Та створено відповідні представлення

    В папці Account

3.1. Register

```
@model CarShop.Models.RegisterViewModel
@{
    ViewBag.Title = "Реєстрація";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-
horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Створення нового профілю</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Реєстрація" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

3.2. Login

```
@using CarShop.Models
@model LoginViewModel
@{
    ViewBag.Title = "Виконати вхід";
}

<h2>@ViewBag.Title.</h2>
<div class="row">
    <div class="col-md-8">
        <section id="loginForm">
            @using (Html.BeginForm("Login", "Account", new { ReturnUrl =
ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role = "form"
}))
            {
                @Html.AntiForgeryToken()
                <h4>Використайте вже зареєстроаний профіль.</h4>
                <hr />
                @Html.ValidationSummary(true, "", new { @class = "text-danger" })
                <div class="form-group">
                    @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label"
})
                    <div class="col-md-10">
```

```
                        @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
                        @Html.ValidationMessageFor(m => m.Email, "", new { @class =
"text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-
label" })
                    <div class="col-md-10">
                        @Html.PasswordFor(m => m.Password, new { @class = "form-control"
})
                        @Html.ValidationMessageFor(m => m.Password, "", new { @class =
"text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <div class="checkbox">
                            @Html.CheckBoxFor(m => m.RememberMe)
                            @Html.LabelFor(m => m.RememberMe)
                        </div>
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <input type="submit" value="Виконати вхід" class="btn btn-
default" />
                    </div>
                </div>
                <p>
                    @Html.ActionLink("Реєстрація нового користувача", "Register")
                </p>

            }



@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

В папці Manage

3.3. ChangePassword

```
@model CarShop.Models.ChangePasswordViewModel
@{
    ViewBag.Title = "Змінити пароль";
    Layout = "~/Views/Shared/_Layout2.cshtml";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("ChangePassword", "Manage", FormMethod.Post, new { @class =
"form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Форма изменения пароля</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.OldPassword, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.OldPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.NewPassword, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
```

```
            @Html.PasswordFor(m => m.NewPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Змінити пароль" class="btn btn-default" />
        </div>
    </div>
}
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

3.4. Index

```
model CarShop.Models.IndexViewModel
@{
    ViewBag.Title = "Керування";
    Layout = "~/Views/Shared/_Layout2.cshtml";
}

<h2>@ViewBag.Title.</h2>

<p class="text-success">@ViewBag.StatusMessage</p>
<div>
    <h4>Змінити параметри облікового запису</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>Пароль:</dt>
        <dd>
            [
            @if (Model.HasPassword)
            {
                @Html.ActionLink("Зміна паролю", "ChangePassword")
            }
            else
            {
                @Html.ActionLink("Створити", "SetPassword")
            }
            ]
        </dd>


    </dl>
</div>
```

Скріншоти програмного виконання

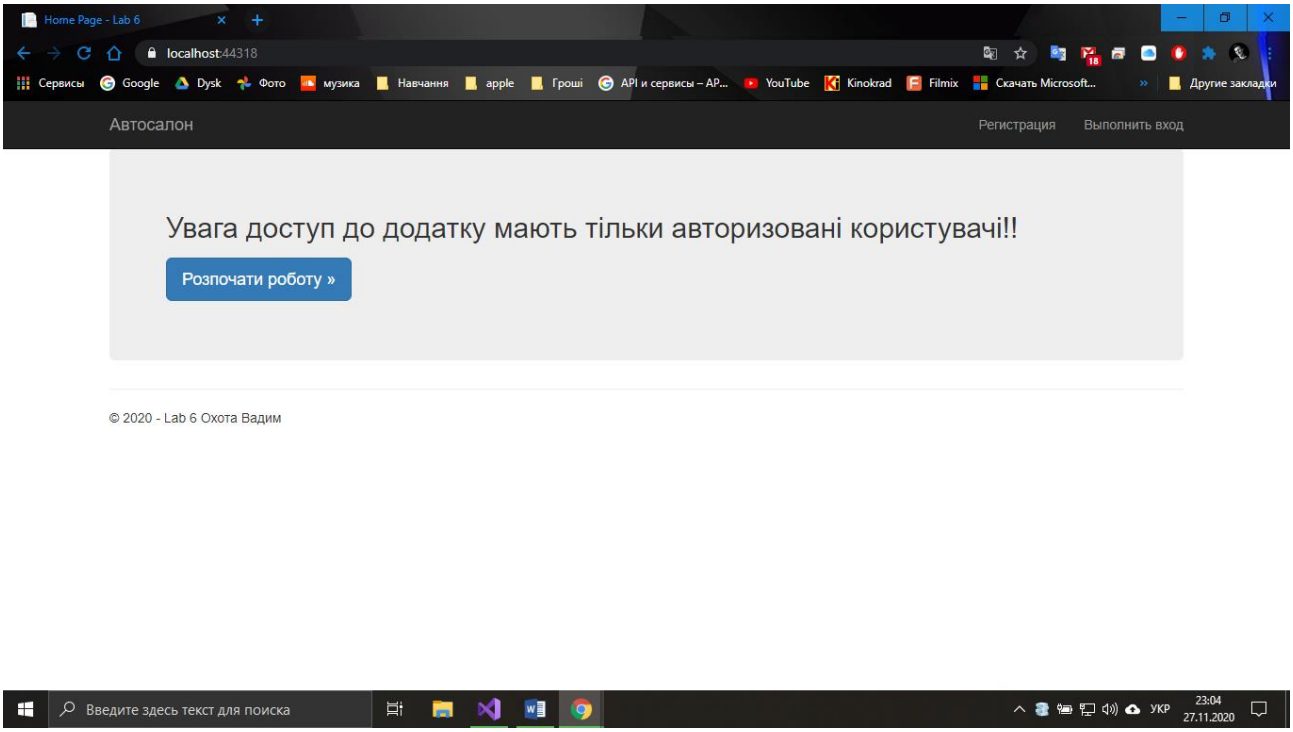

Рисунок 1 Головна сторінка

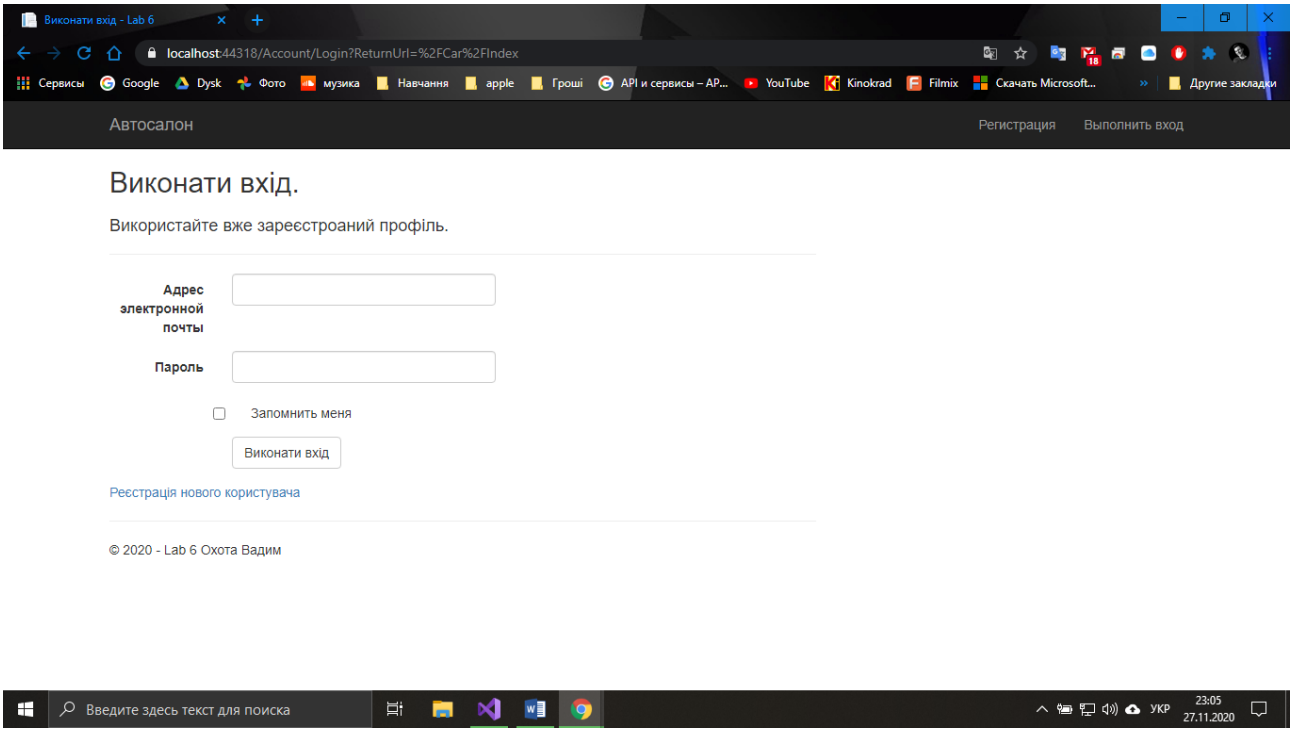

Рисунок 2 Заборона доступу для неавторизованого користувача. Форма входу

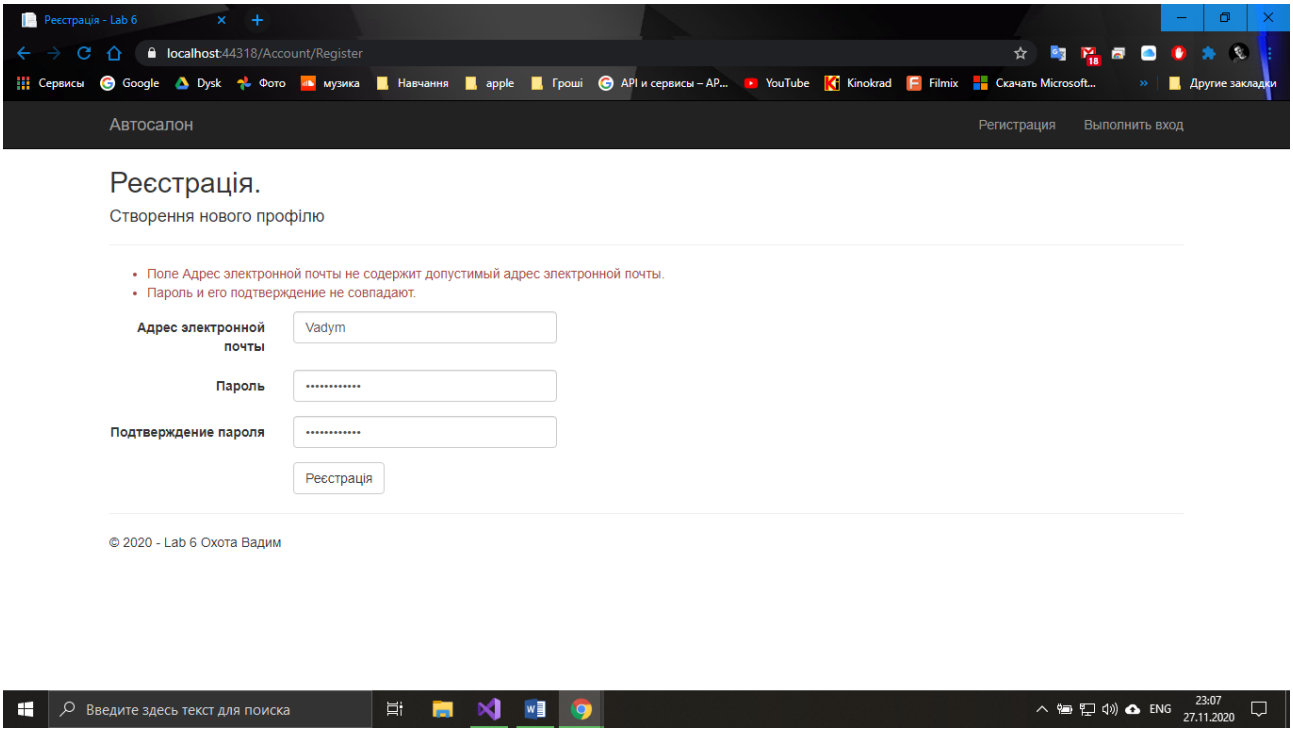

Рисунок 3 Форма реєстрації та валідація

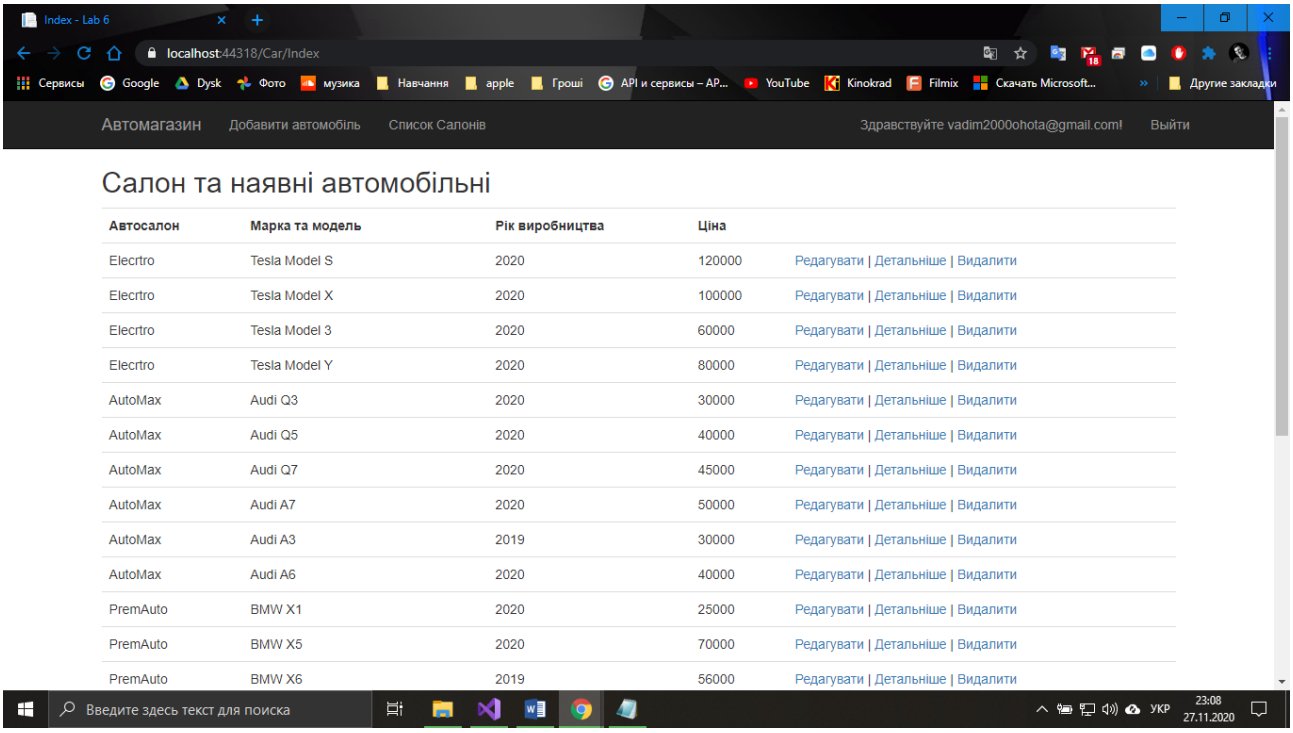

Рисунок 4 Дозволений доступ з авторизованого профілю

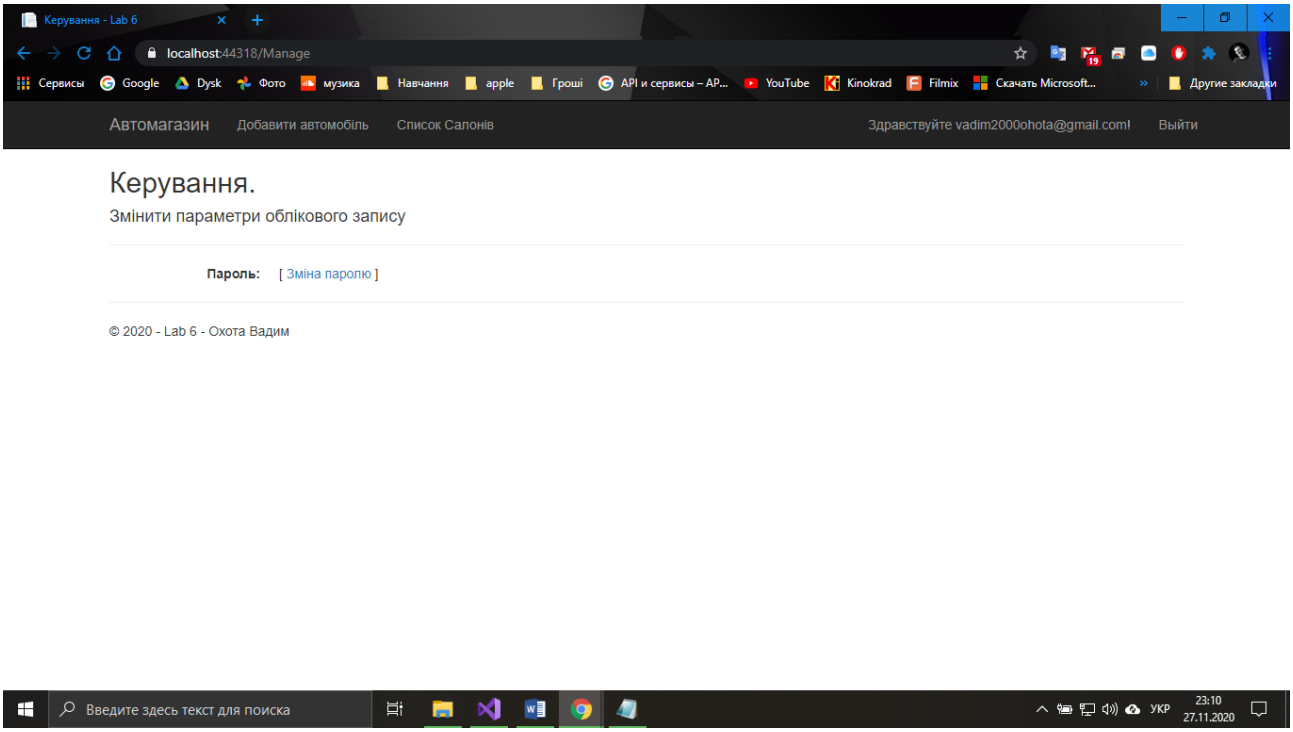

Рисунок 5 Керування обліковим записом

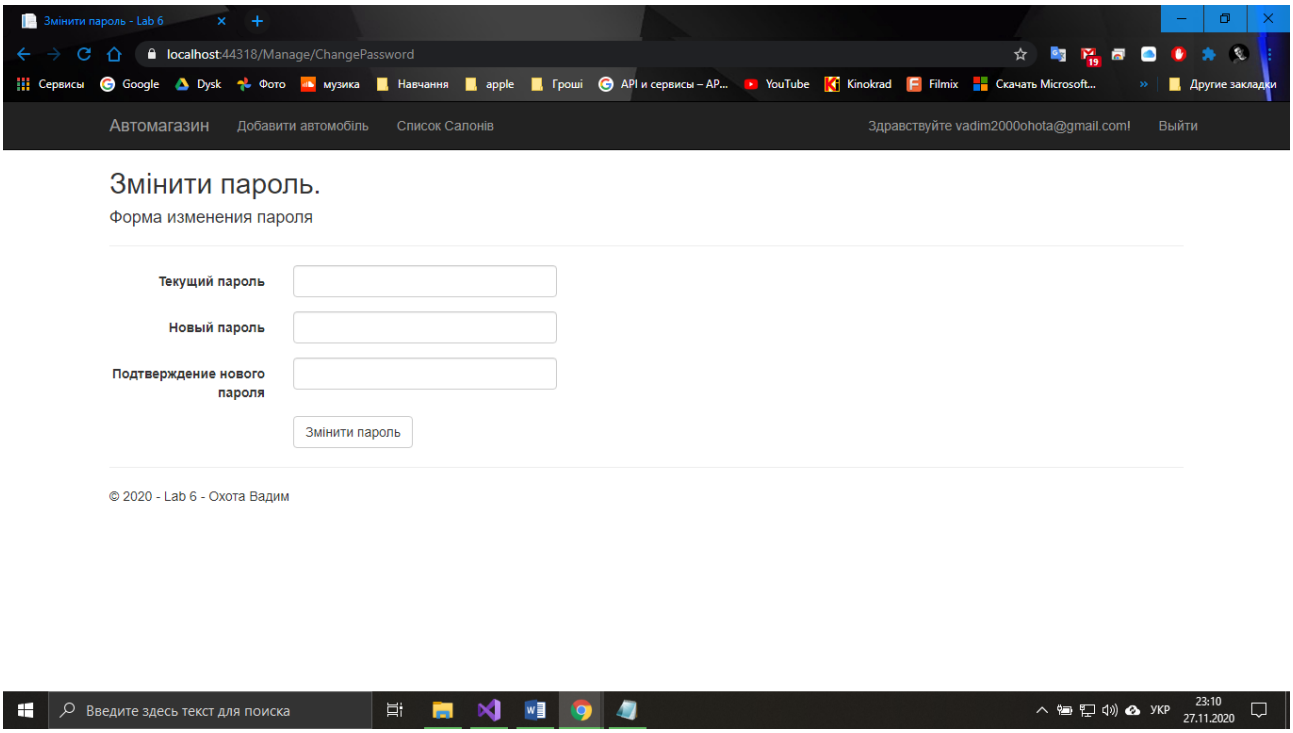

Рисунок 6 Заміна пароля

Контрольні питання

1. **Механізми авторизації і аутентифікації**

   Вони дозволяють розмежувати доступ для різних груп користувачів, а також ідентифікувати користувачів.

2. **Авторизація в MVC**

Авторизація в MVC опирається на три ключові області, які допомагають керувати користувачами в системі. Це членство (membership), ролі і профілі.

3. **Управління користувачами**

   Управління користувачами, членством і ролям проводиться за допомогою провайдерів членства і ролей.

4. **Провайдер SimpleMembershipProvider**

   SimpleMembershipProvider покликаний спростити роботу з користувачами і ролями, володіючи 124 більшою гнучкістю і розширюваністю в порівнянні з традиційним провайдером членства.

5. **Універсальні провайдери**

   Інший підхід до авторизації та аутентифікації представляють універсальні провайдери членства і ролей. Хоча в цілому вони надають все той же функціонал, що і SimpleMembershipProvider і SimpleRolesProvider

6. **Аутентифікація Windows**

   Аутентифікація Windows представляє один із способів аутентифікації користувача в системі. При використанні цієї моделі аутентифікації при зверненні користувача до ресурсів веб-додатка разом з HTTP-запитом надсилається і токен безпеки Windows, який і верифікує користувача.

7. **Налаштуння додатка на Аутентифікацію Windows**

   Для цього перейдемо до властивостей проекту і встановимо для властивості Anonymous Authentication значення Disabled, а для властивості Windows Authentication значення Enabled

8. **Налаштування авторизації**

   Застосовуючи фільтри авторизації, ми можемо обмежити доступ до дій контролера або до контролера в 121 цілому для певних ролей (використовуючи групи в Windows) або користувачів.

9. **Аутентифікація форм**

   Ще один способів аутентифікації користувачів є аутентифікація форм. Вона більш гнучка в порівнянні з аутентификацией Windows, хоча, можливо, і трохи більш складна. Вона грунтується на видачу аутентифікованим користувачеві кукинаборів, за якими він надалі верифікується.

10. **Файл конфігурації web.config**

    Файл Web.config. Файл конфігурації програми, який знаходиться в кореневій теці програми

11. **Метод Login контролера AccountController**

Вся логіка аутентифікації укладена в методах Login контролера AccountController, які здійснюють вхід користувача в систему.

12. **Валідація користувача**

    Валідація користувача: це робить метод Membership.ValidateUser. І якщо користувач знаходиться в нашій базі даних, то далі додаток переходить до другого етапу.

13. **Аутентифікаційний тікет**

    Деякий квиток безпеки, за яким веб-додаток буде впізнавати користувача. Цей тікет додаток встановлює для браузера у вигляді куки-набору на ім'я .AUTHPATH за допомогою методу FormsAuthentication.SetAuthCookie.

14. **Створення куки-набору**

    Важливо відзначити, що при установці куки-наборів їх вміст шифрується за допомогою машинних ключів, яку автоматично створює IIS на сервері.

15. **Налаштування аутентифікації**

    Використовуючи атрибути вузла forms у файлі конфігурації, ми можемо налаштувати параметри аутентифікації.

16. **Таблиці за замовчуванням в ASP.NET MVC 4**

    При першому зверненні до БД, якщо її не існувало, вона буде створена і автоматично заповнена таблицями з певними полями.