

# Assignment 1 – Internet Applications and Web Programming (CPIS-358)

**Project Title:** EyeTune

**Group Number:** 11

**Members:** Faris al-Ghamdi (2236815), Fawaz Abdulhadi (2343406)

## 1. Introduction

### Problem Statement

Today, most people spend long hours in front of screens—computers, tablets, and mobile devices. This constant exposure leads to eye strain, headaches, dryness, and long-term discomfort. Many users are unaware of healthy screen habits and do not follow recommended guidelines for safe screen usage.

Our group aims to solve this problem by creating a simple, accessible web tool that analyzes basic screen usage behavior, educates users about eye health, and encourages better habits.

### Project Solution

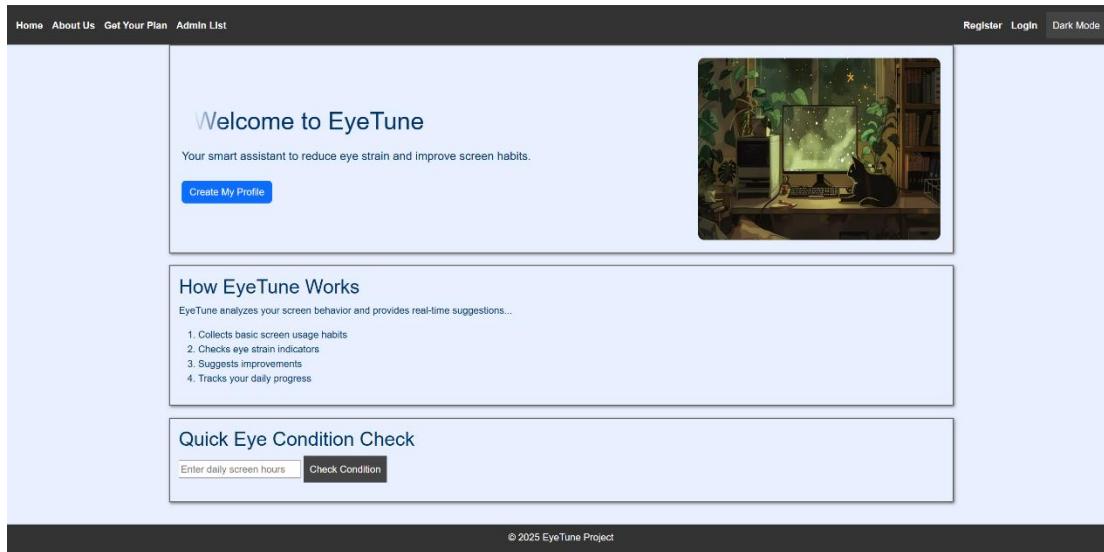
Our project, **EyeTune**, is a web-based system designed to help users reduce eye strain through awareness, guidance, and interactive features. EyeTune provides:

- A friendly interface to educate users on eye care
  - Dynamic “Eye Condition Check” tool that displays real-time feedback
  - Signup/Login system for users
  - Visual elements such as mapped images, tables, lists, and dynamic content
- Through HTML, CSS, and JavaScript, EyeTune delivers an interactive, informative, and user-friendly solution to support better digital wellness.

## 2. Project Pages Overview

### Home Page

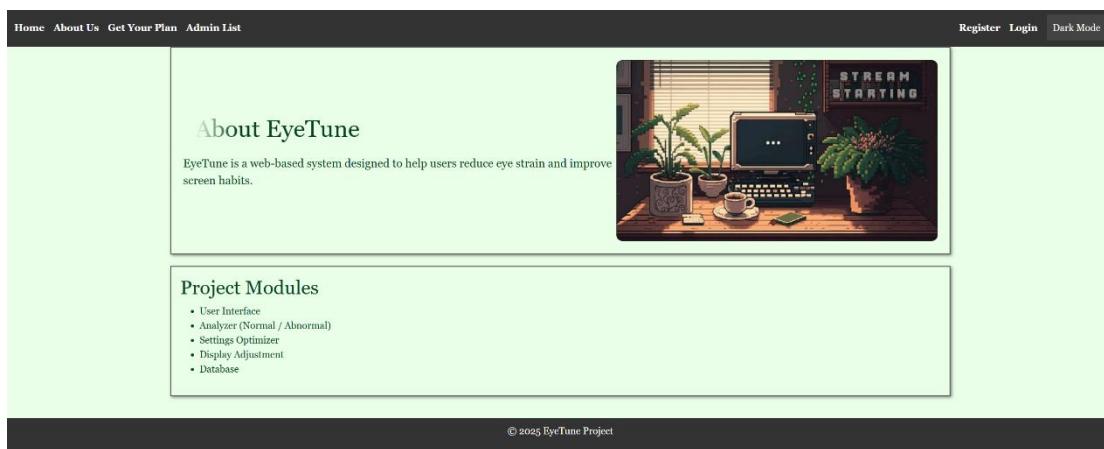
Includes a welcome message, project description, image mapping, internal linking, lists, table of eye-care tips, and dynamic JavaScript features. (style.css + internal css + inline css + some bootstrap)



The screenshot shows the EyeTune home page. At the top, there's a navigation bar with links for Home, About Us, Get Your Plan, Admin List, Register, Login, and Dark Mode. The main content area has a "Welcome to EyeTune" heading, a sub-headline "Your smart assistant to reduce eye strain and improve screen habits.", and a "Create My Profile" button. To the right is a decorative illustration of a cat sitting at a desk with a computer monitor displaying a night sky. Below this section is another box titled "How EyeTune Works" with a list of four steps: 1. Collects basic screen usage habits, 2. Checks eye strain indicators, 3. Suggests improvements, and 4. Tracks your daily progress. At the bottom is a "Quick Eye Condition Check" section with input fields for "Enter daily screen hours" and a "Check Condition" button.

### About Us Page

Provides project background, mission, and modules used in EyeTune. Contains images and styled sections. (style.css + internal css + inline css + some bootstrap)



The screenshot shows the EyeTune about us page. The top navigation bar and footer are identical to the home page. The main content area has a "About EyeTune" heading and a paragraph describing EyeTune as a web-based system designed to help users reduce eye strain and improve screen habits. To the right is a decorative illustration of a desk setup with a computer monitor displaying the text "STREAM STARTING". Below this is a "Project Modules" section with a bulleted list: User Interface, Analyzer (Normal / Abnormal), Settings Optimizer, Display Adjustment, and Database.

## Signup Page

Includes a registration form with JavaScript password-strength validation. (style.css + some bootstrap)

The screenshot shows a registration page titled "Register". On the left, there is a section for creating a new account with three input fields: "Email", "Password", and "Confirm Password". Below these fields is a blue "Register" button. On the right, there is a section titled "Use another service to register." with a note stating: "There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services.](#)" At the bottom of the page, there is a footer bar with the text "© 2025 EyeTune Project".

## Login Page

Allows users to log in with JavaScript-based validation and error handling. (style.css + some bootstrap)

The screenshot shows a login page titled "Log in". On the left, there is a section for using a local account to log in with two input fields: "Email" and "Password". Below these fields is a checkbox labeled "Remember me?". Below the "Email" field is a blue "Log in" button. To the right, there is a section titled "Use another service to log in." with a note stating: "There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services.](#)" At the bottom of the page, there is a footer bar with the text "© 2025 EyeTune Project".

## Create Eye profile

Collect some data from users so the data can be used to provide the best tips and recommendation on how to improve their time behind screen and lower eye fatigue.  
(style.css + some bootstrap)

The screenshot shows a web application interface. At the top, there is a dark navigation bar with links: Home, About Us, Get Your Plan, Admin List, Register, Login, and Dark Mode. The main content area has a title "Create Your Eye Profile" and a sub-instruction "Please enter your details so we can generate your eye care tips." Below this is a blue "Generate My Plan" button. The form itself is titled "Create" and has a section labeled "Eyes". It contains several input fields and dropdown menus:

- Age: An input field.
- Gender: A dropdown menu with the placeholder "Select Gender".
- ScreenTimeHoursPerDay: An input field.
- GlassesType:
  - None (I don't wear glasses): A radio button.
  - HasEyeDisease: A checkbox.
- EyeDiseaseDetails: An input field.
- None: A radio button.
- FamilyHistoryOfEyeDisease: A checkbox.

All pages have their same dark mode theme (from style.css)

The screenshots illustrate the consistent dark mode theme applied to various sections of the EyeTune project:

- Create Your Eye Profile:** A form for users to input details about their eye health and habits. It includes fields for Age, Gender, ScreenTimeHoursPerDay, GlassesType (with options like "None" or "Wear glasses"), and EyeDiseaseDetails.
- Register:** A form for creating a new account. It requires Email, Password, and Confirm Password, with a "Register" button.
- Log in:** A form for logging in with a local account. It includes fields for Email and Password, a "Remember me?" checkbox, and a "Log In" button. It also links for "Forgot your password?", "Register as a new user?", and "Resend email confirmation".
- Welcome to EyeTune:** The main landing page featuring a welcome message, a "Welcome to EyeTune" heading, a sub-headline "Your smart assistant to reduce eye strain and improve screen habits.", and a "Create My Plan" button.
- About EyeTune:** Informational page detailing the project's purpose ("EyeTune is a web-based system designed to help users reduce eye strain and improve screen habits."), its modules ("User Interface", "Analytics Engine (Backend)", "Setting Optimizer", "Display Adjustment", and "Database"), and a screenshot of a computer setup with a "STREAM STARTING" message on the screen.

### 3. Technologies Used

#### HTML

- Pages: Home, Login, Signup, About Us
- Forms, lists, tables, headings, paragraphs
- Internal and external linking
- Image mapping
- Sections and articles

#### CSS

- Inline, internal, and external styles
- General classes, shadows, borders
- Responsive layout
- Menus and page structure
- Layering (z-index) (*you will add this*)

#### JavaScript

- Functions and events
- Dynamic images and text
- Form validation (Login + Password strength)
- Control structures (loops, if-else)
- Changing HTML and CSS dynamically

#### Razor Views (.cshtml)

- Pages: Converted static HTML to dynamic Views (Home, Login, Signup, Admin, Eye Check).
- **Razor Syntax:** Used @Html.BeginForm, @Html.TextBoxFor, and @Url.Action to bind data to the server.
- Master Page Layout (\_Layout.cshtml) for consistent navigation.

#### ASP.NET MVC (Backend)

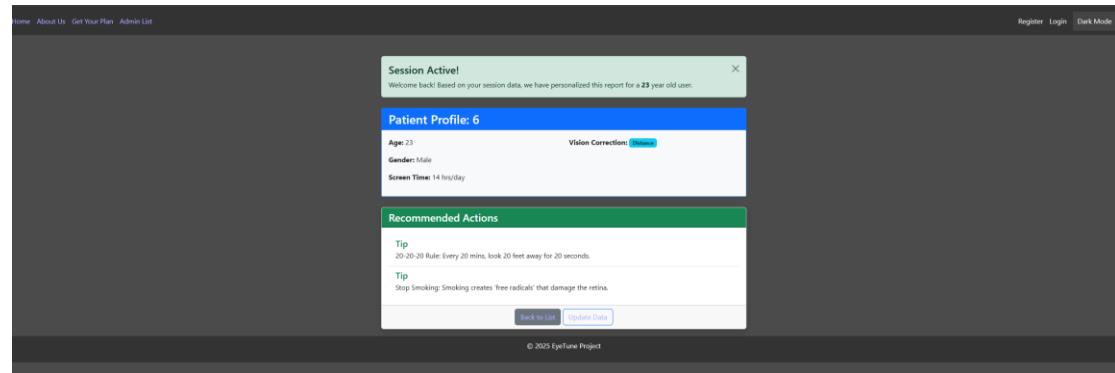
- **Controllers:** Created DefaultController for navigation and UserController for account management.
- **Models:** Designed UserControllerClass and EyeCheckModel to handle data structure.
- **State Management:**
  - **Sessions:** Used to keep users logged in securely.
  - **Cookies:** Implemented "Remember Me" functionality to persist login state.

## Database (SQL Server)

- **LocalDB:** Used Microsoft SQL Server LocalDB for storage.
- **ADO.NET:** Used SqlConnection and SqlCommand to connect code to the database.
- **CRUD Operations:** Implemented INSERT for Sign Up and SELECT for Login and Admin Search.

## 4. Sessions, cookies and security

The session was added to use the age of the user and welcome them. Setting the system to personalize the report to the users age.



For cookies we follow the rubric in make it save user state and we apply it in Dark-Mode function that will save dark mode even when coming back

A screenshot of the Chrome DevTools Application tab. On the left, there's a sidebar with sections for 'Manifest', 'Services', 'Storage' (with 'Local...', 'Session...', 'External...', 'Index...', and 'Cookies'), and 'HTTP'. The 'Cookies' section is currently selected. On the right, a table lists cookies with columns for Name, Value, P..., Ex..., Size, H..., S..., P..., and C... (which is partially cut off). One cookie, 'PreferredTheme', has its value 'Dark' highlighted with a red rectangle. Below the table, a code editor shows a JavaScript function named 'darkModeToggle()'. The function toggles the 'dark-mode' class on the body element and then saves the choice to local storage using 'localStorage.setItem("theme", "dark")' if the class is present, or 'localStorage.setItem("theme", "light")' if it's not. The code editor also shows '0 references'.

For security we authorized that deleting function is only functioning by admin

```
[Authorize]
0 references
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var eyes = await _context.Eyes.FindAsync(id);
    if (eyes != null)
    {
        _context.Eyes.Remove(eyes);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

1 reference
private bool EyesExists(int id)
{
    return _context.Eyes.Any(e => e.Id == id);
}

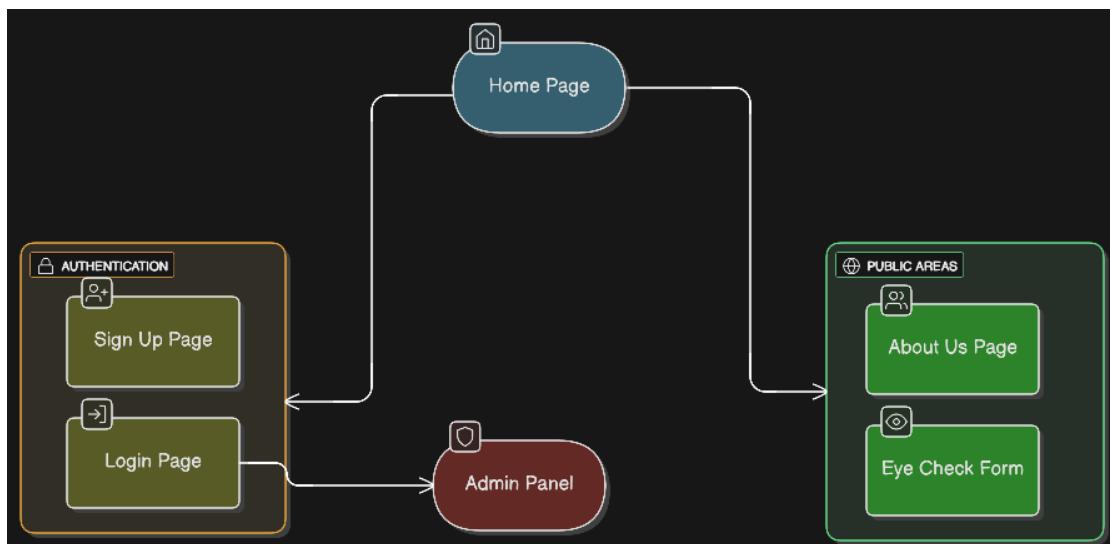
[Authorize]
0 references
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var eyes = await _context.Eyes
        .FirstOrDefaultAsync(m => m.Id == id);
    if (eyes == null)
    {
        return NotFound();
    }

    return View(eyes);
}
```

## 5. Sitemap

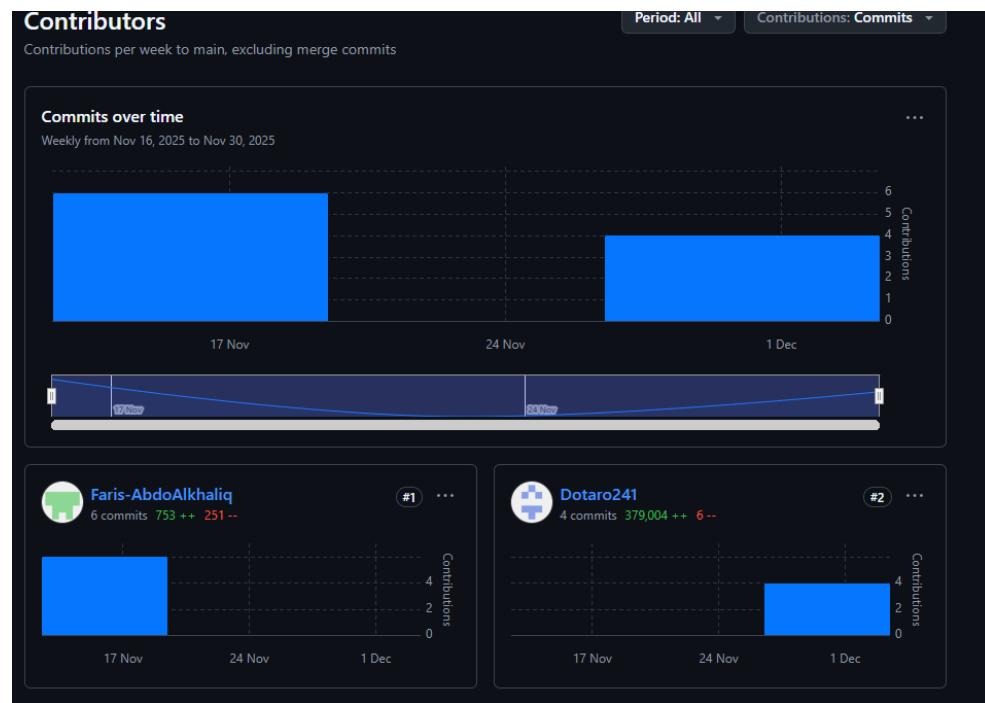
Below is a simple visual sitemap showing the structure of the entire project:



### Sitemap Explanation

- **Home** is the main entry point of the website.
- Every page links back to every other page using the top navigation menu.
- All images on each page map to the **About Us** page.
- Internal link inside Home page (#tipsTable) provides intra-page navigation.
- **Public Pages:** The Home, About Us, and Eye Check forms are accessible to all visitors.
- **Member Pages:** The **Login** and **Sign Up** pages allow users to create accounts and authenticate.
- Admin Panel (Secured): Only accessible when logging in
- Database Integration
- Image Mapping

## 6. GitHub Usage



## 7. Code Snippets

### Sign-in

```
[HttpGet]
public ActionResult Signup()
{
    return View();
}

[HttpPost]
public ActionResult Signup(UserControllerClass newUser)
{
    if (ModelState.IsValid)
    {
        using (SqlConnection sqlCon = new SqlConnection(connectionString))
        {
            sqlCon.Open();

            string query = "INSERT INTO Users (Username, Email, Password, Age) VALUES (@Username, @Email, @Password, @Age)";

            SqlCommand sqlCmd = new SqlCommand(query, sqlCon);

            sqlCmd.Parameters.AddWithValue("@Username", newUser.Username ?? "");
            sqlCmd.Parameters.AddWithValue("@Email", newUser.Email ?? "");
            sqlCmd.Parameters.AddWithValue("@Password", newUser.Password ?? "");
            if (newUser.Age == null)
                sqlCmd.Parameters.AddWithValue("@Age", DBNull.Value);
            else
                sqlCmd.Parameters.AddWithValue("@Age", newUser.Age);
            sqlCmd.ExecuteNonQuery();
        }
        TempData["SuccessMessage"] = "Account created! Please login.";
        return RedirectToAction("Login");
    }
}
```

```

        return View(newUser);
    }
}



## Admin Page


[HttpGet]
public ActionResult Admin(string search)
{
    List<UserControllerClass> userList = new List<UserControllerClass>();

    using (SqlConnection sqlCon = new SqlConnection(connectionString))
    {
        sqlCon.Open();
        string query = "SELECT * FROM Users";

        if (!string.IsNullOrEmpty(search))
        {
            query += " WHERE Username LIKE @Search";
        }

        SqlCommand sqlCmd = new SqlCommand(query, sqlCon);
        if (!string.IsNullOrEmpty(search))
        {
            sqlCmd.Parameters.AddWithValue("@Search", "%" + search + "%");
        }

        SqlDataReader reader = sqlCmd.ExecuteReader();
        while (reader.Read())
        {
            userList.Add(new UserControllerClass
            {
                Id = Convert.ToInt32(reader["Id"]),
                Username = reader["Username"].ToString(),
                Email = reader["Email"].ToString(),
                Age = reader["Age"] == DBNull.Value ? 0 : Convert.ToInt32(reader["Age"])
            });
        }
    }
}

```

```
});  
}  
}  
return View(userList);
```