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

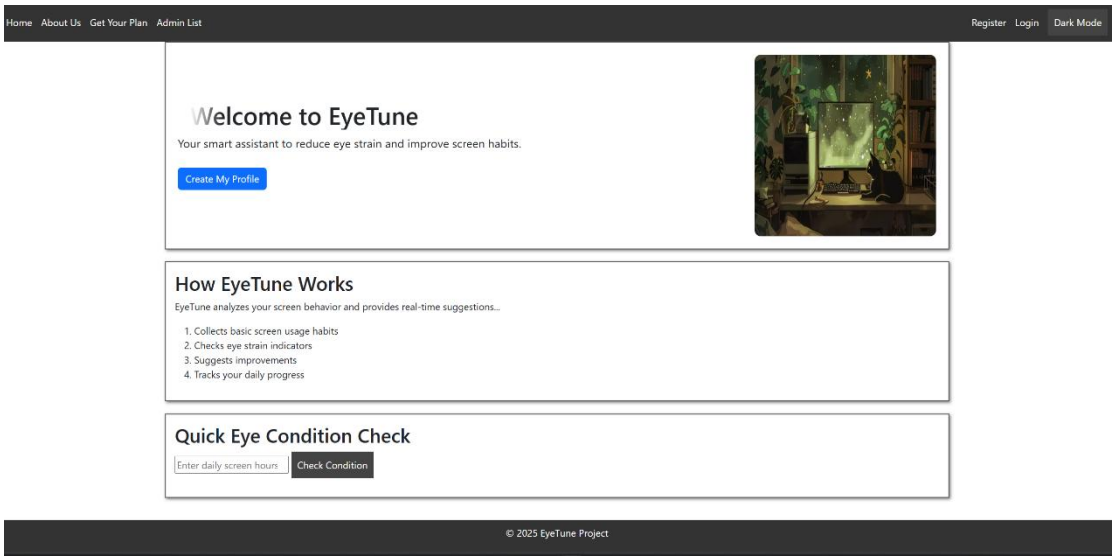
Our project, **EyeTune**, is a web-based system designed to help users reduce eye strain through awareness, guidance, and interactive features. EyeTune provides:

- 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

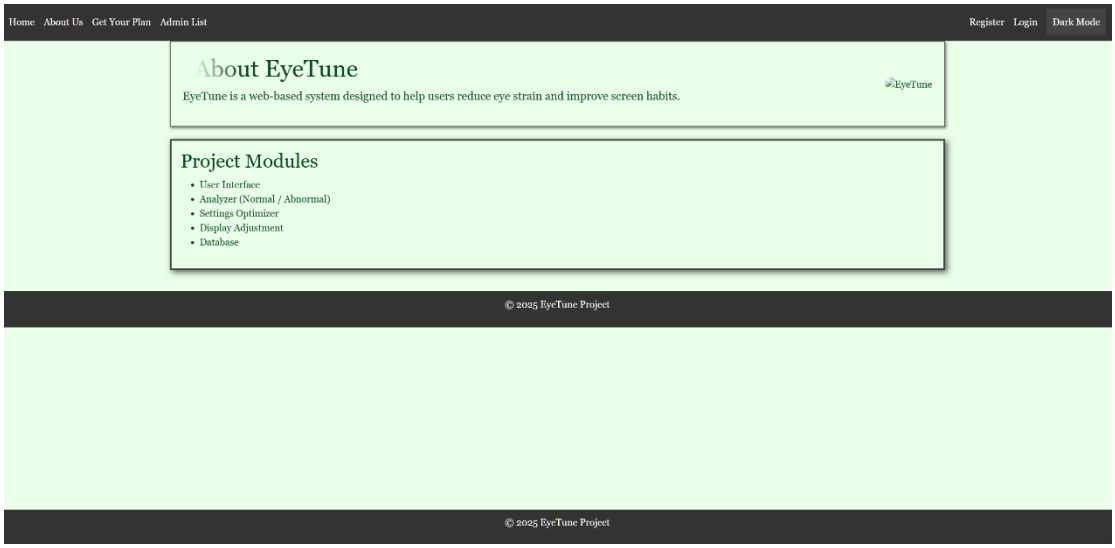
Home Page

Includes a welcome message, project description, image mapping, internal linking, lists, table of eye-care tips, and dynamic JavaScript features.



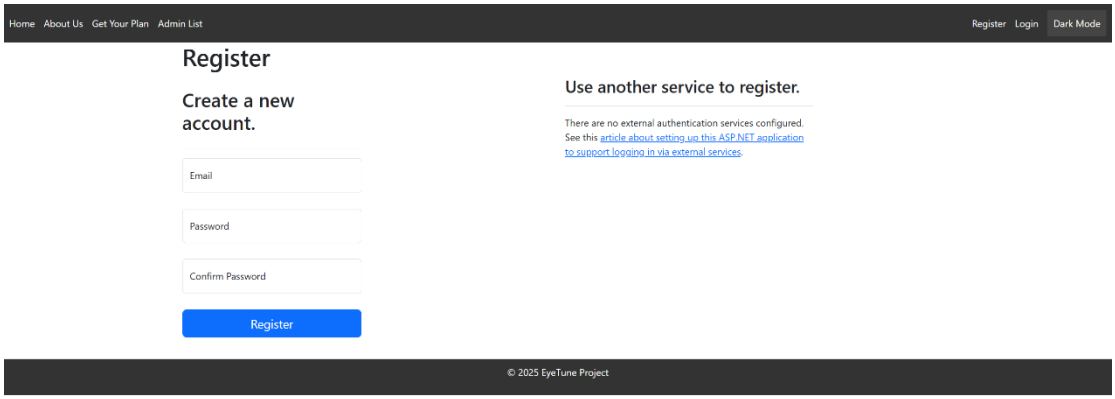
About Us Page

Provides project background, mission, and modules used in EyeTune. Contains images and styled sections.



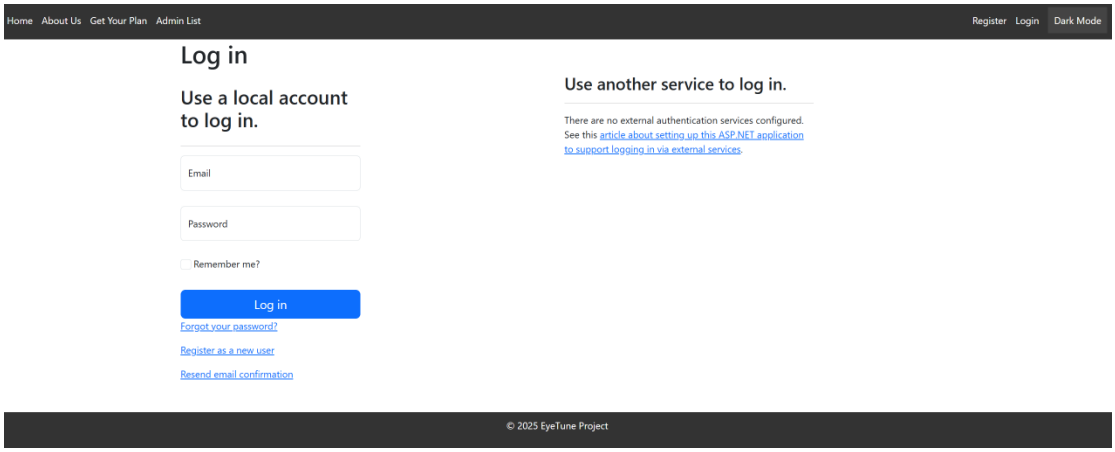
Signup Page

Includes a registration form with JavaScript password-strength validation.



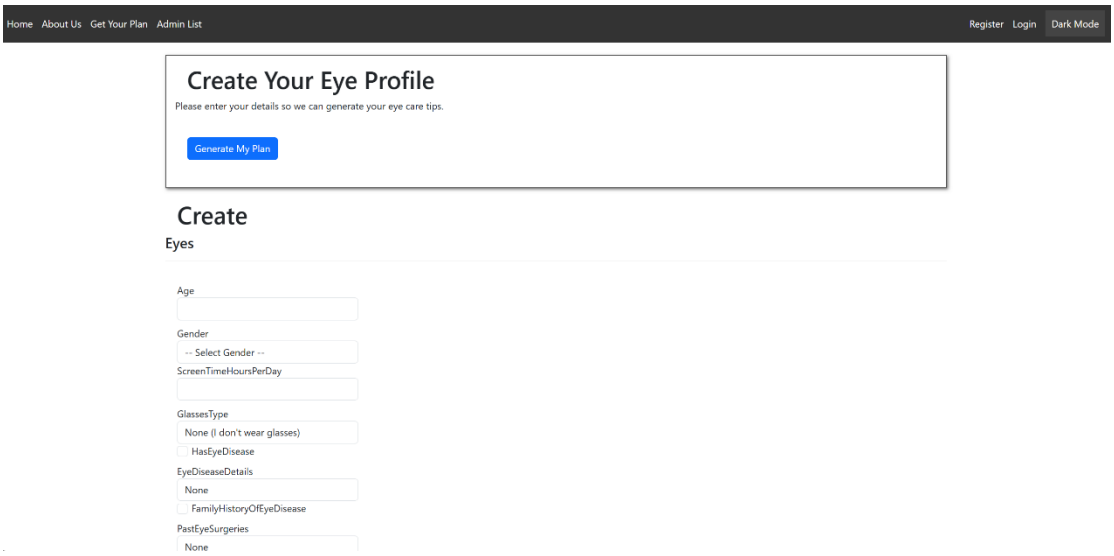
Login Page

Allows users to log in with JavaScript-based validation and error handling.



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.



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)

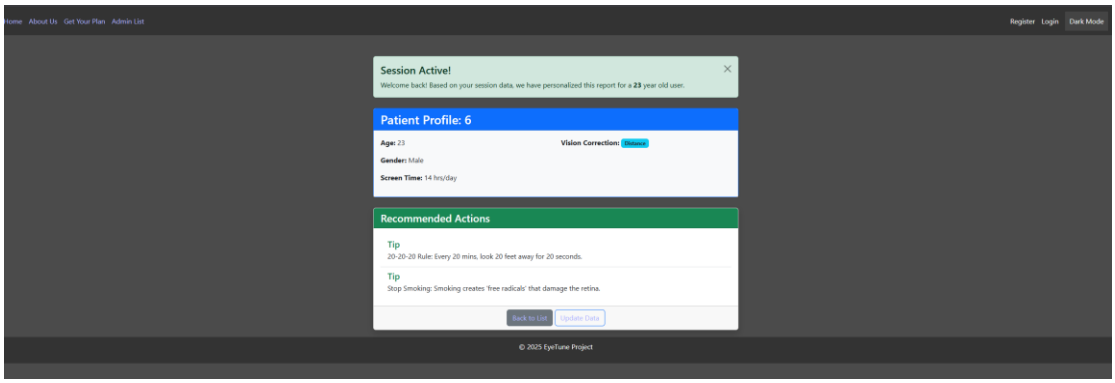
- **Controllers:** Created DefaultController for navigation and UserController for account management.
- **Models:** Designed UserControllerClass and EyeCheckModel to handle data structure.
- **State Management:**
    - **Sessions:** Used to keep users logged in securely.
    - **Cookies:** Implemented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

Database (SQL Server)

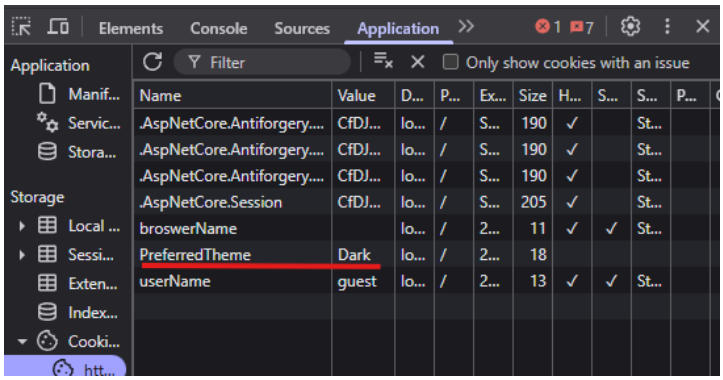
- **LocalDB:** Used Microsoft SQL Server LocalDB for storage.
- **ADO.NET:** Used SqlConnection and SqlCommand to connect code to the database.
- **CRUD Operations:** Implemented INSERT for Sign Up and SELECT for Login and Admin Search.

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




```
0 references
function darkModeToggle() {
    let body = document.body;
    // 2. Toggle the class
    body.classList.toggle("dark-mode");

    // 3. Save the choice
    if (body.classList.contains("dark-mode")) {
        localStorage.setItem("theme", "dark");
    } else {
        localStorage.setItem("theme", "light");
    }
}
```

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);
}
```

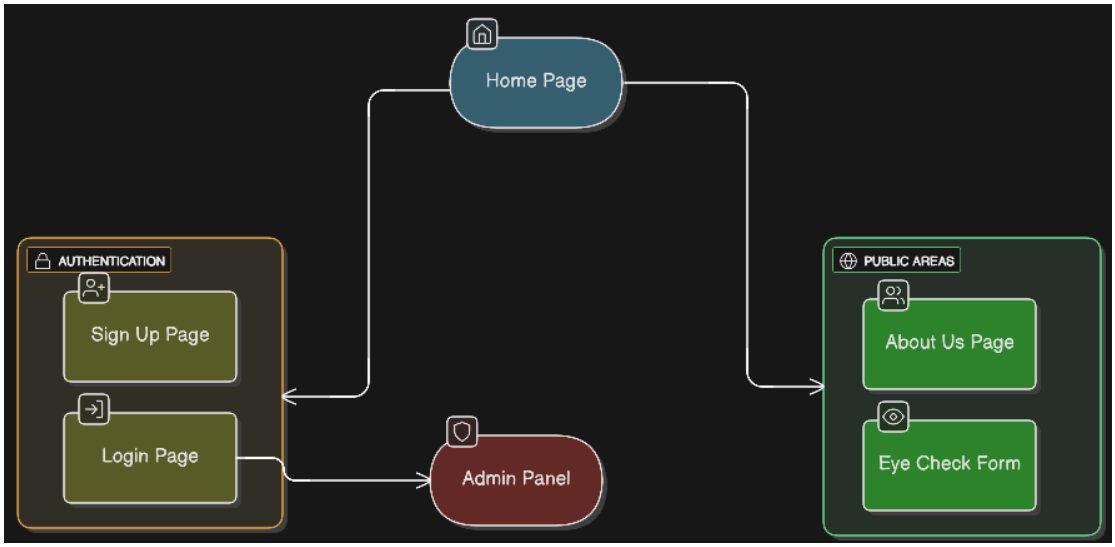
```
[Authorize]
0 references
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    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:

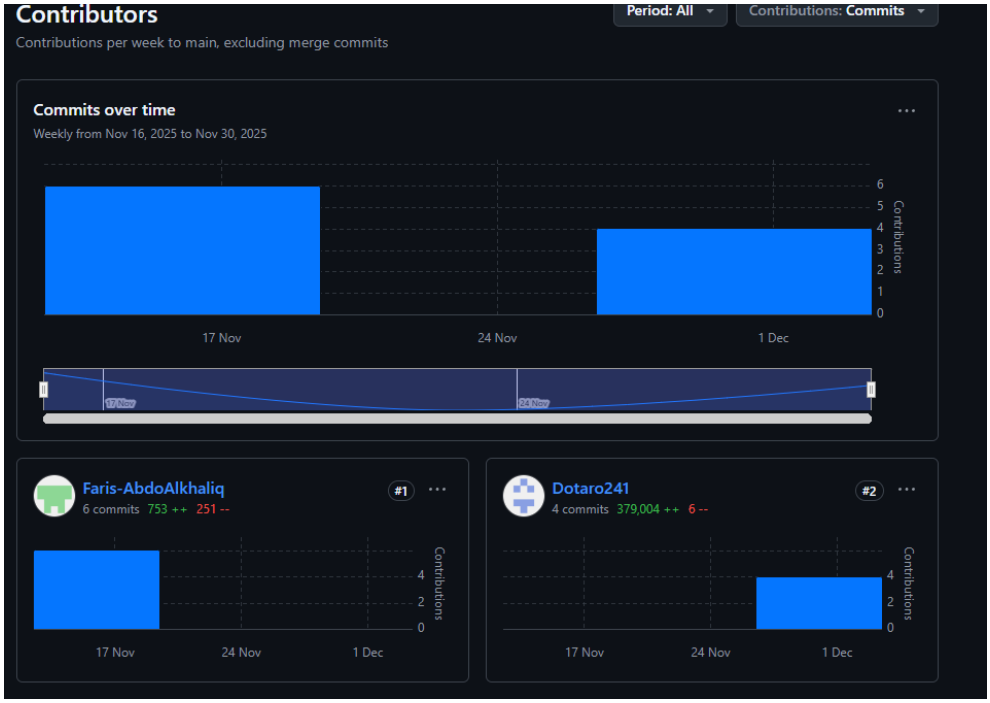


Sitemap Explanation

- **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");
    }
    else
    {
        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);
```