Fife College

Web Development:
Advanced Web Scripting Concepts

HL9W 35

Assessment
LO 2,3

Student	Kamil Milej						
I.D.	2264629						
Date	16/05/2025						
	Pass		**Fail**		**Remediation**		
Tutor							

Design Document

1. Introduction

This project is a simple web application that requires users to log in with a username and password stored in a server-side database. After logging in, users can access several pages: Login page, Topic Information page, and a Frequently Asked Questions (FAQ) page and Homepage page. Users can log out at any time, and access to protected pages is blocked after logging out.

- ✓ The goal of the application is to demonstrate key web development skills, including:
- ✓ DOM Programming (dynamic elements and events) - used for dynamically showing/hiding content, handling user interactions e.g. toggling FAQ answers, and rendering templates.
- ✓ AJAX (loading data without refreshing the page) - used to fetch data from the server.
- ✓ HTML use the native HTML <template> tag combined with the Fetch API to dynamically display topic information
- ✓ Use of the MVC (Model-View-Controller) design pattern, which divide application into:

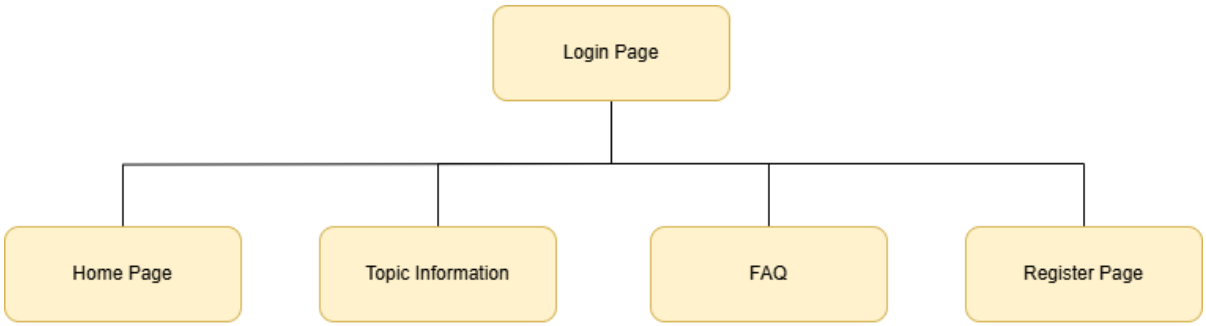
Model - backend/data layer, responsible for managing data (e.g. user accounts, topic information, FAQ content). It interacts with the database and returns data to the controller.

View - frontend layer, represents the user interface, built with HTML, CSS, and templates. It displays the data to the user and handles basic interactions.

Controller - connects the view and model, containing the main application logic. It handles user actions (e.g. login, logout, loading topic info), makes AJAX requests, updates the DOM, and manages routing.

- ✓ Web application security (SQL injection protection and session management) - login form uses prepared statements (PDO) to protect against SQL Injection. Session management ensures that users cannot access protected content without logging in.
- ✓ Deployment to a live server

2. Site Structure Diagram

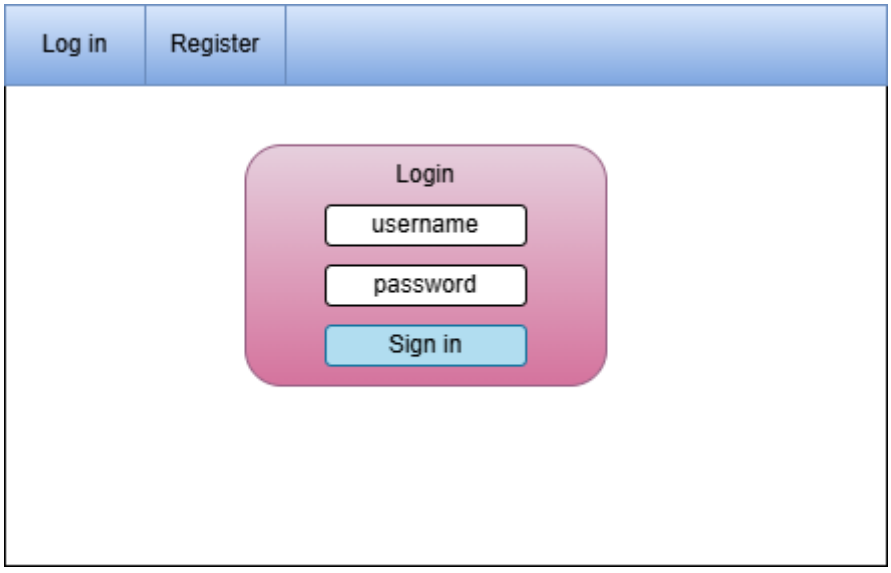


This diagram shows the structure of my website, which includes the following:

- The login page is the first page the user sees.
- From the login page, the only available action is to navigate to the registration page.
- After logging in, the navigation bar displays the remaining page options along with a Logout button.

3. Wireframe design for each screen

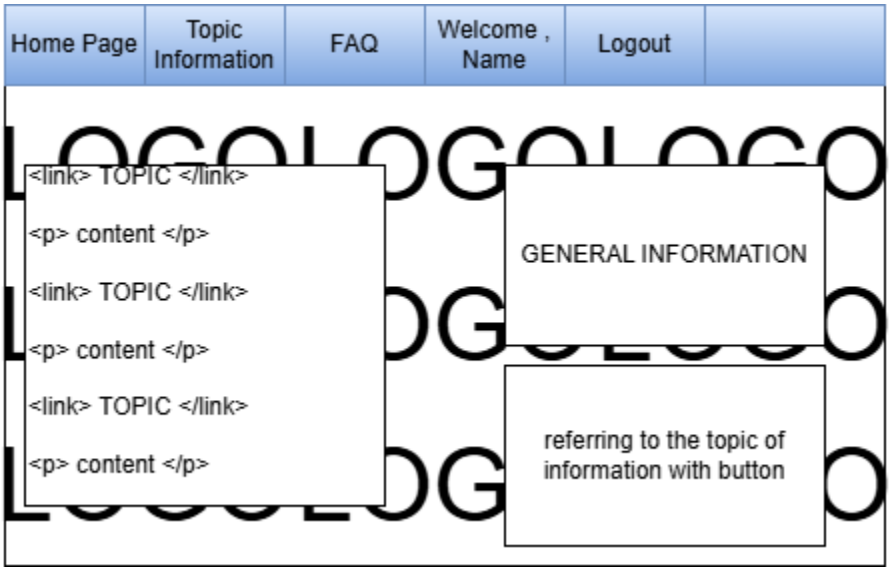
a) Login Page



The login page will consist of a navbar that will appear on every page and will include two options: "Log In" and "Register". Additionally, there will be a window that allows the user to log in to the site. According to the requirements, the user will not be able to perform any actions on the site without being logged in. After logging in, the user will be automatically redirected to the homepage, which serves as the main page after login. The login process will use the following prepared statement: $stmt = $dbConnect->prepare("SELECT user_id, username, password, role FROM Users WHERE username = ?"); which means it uses placeholders, which protects against SQL injection.

Login Page – 1st main screen	
DOM Programming	login.js – (show password)
Templating System	Include (Header and footer)
MVC Pattern	The application follows the Model-View-Controller (MVC) architecture. All files related to the login page are isolated from other pages.
AJAX	-
Security	Input validation (server-side), PDO with prepared statements
Deployment	Everything show correctly

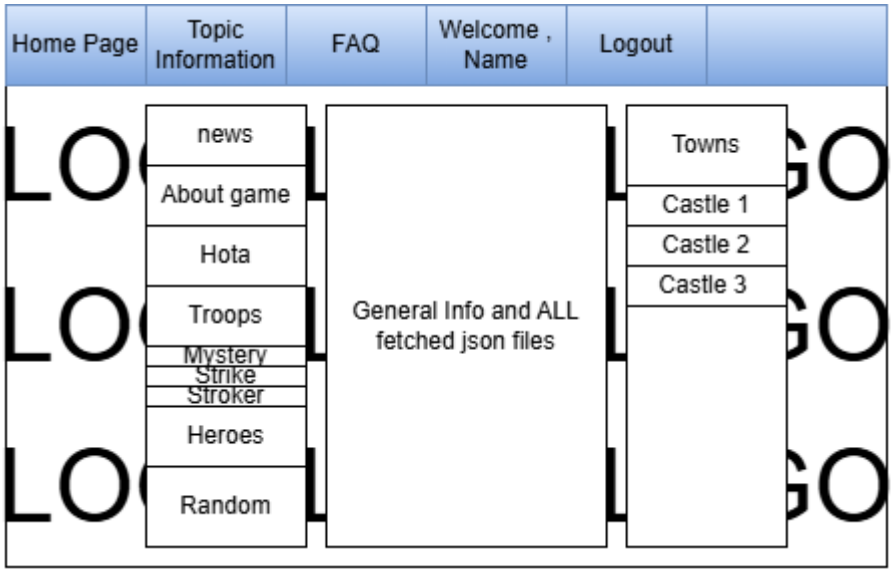
b) Home Page



The homepage is the main landing page that contains links to other pages and some general information. It also includes a navbar, which changes dynamically after logging in, additional options appear, while the "Register" and "Login" buttons disappear and are replaced with a "Logout" option. Additionally, the logo is part of the visual design. The page is divided into two sections, left and right, which contain general content and a button that redirects to the "Topic Information" section.

Homepage Page	
DOM Programming	-
Templating System	Include (Header and footer)
MVC Pattern	The application follows the Model-View-Controller (MVC) architecture. All files related to the login page are isolated from other pages.
AJAX	-
Security	session check for logged-in users – if user is not log in cant use this page.
Deployment	Everything show correctly

c) Topic Information Page



Topic Information is our main page where many features have been added, such as DOM Programming, AJAX, and a Templating System.

After entering the page, we also have a navbar, a logo, and the main section which is divided into three columns. In the left column, there will be buttons that change the content in the middle column, and one of the buttons will have a dropdown that expands and shows 3 categories which must be added to the assignment - these are 3 different pieces of information about dragons.

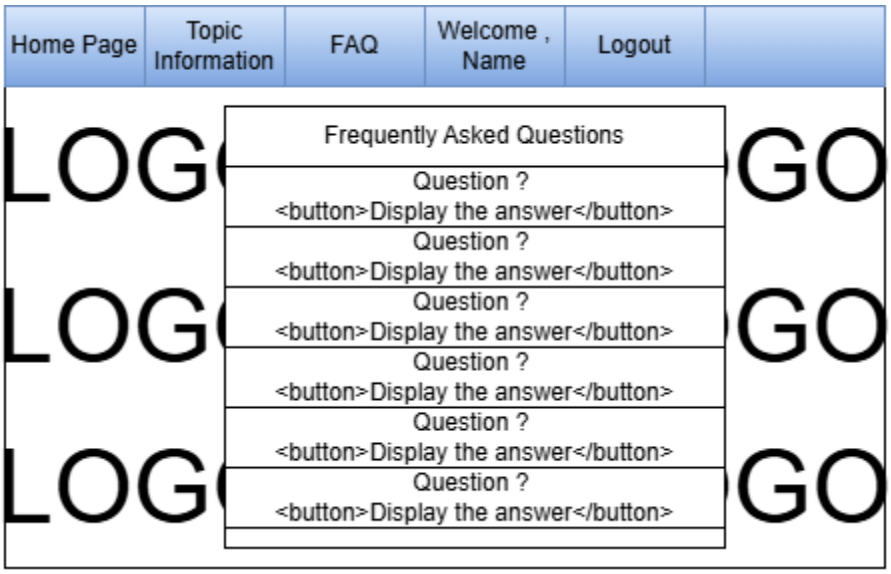
I already tried to use the method from lesson : Click here to load employees and const file = loadEmployees.getAttribute("href"); to put href into te variable but from my attempts, I found out that this cannot be used with buttons. That's why in the left column I will use data-json, which perfectly shows the data even with buttons. The school example I will use in the right column. Also, in this column, AJAX with JSON is used, that is, asynchronous JavaScript, whose main purpose is usually to handle the page without reloading it and to fetch or send data in the background, such as JSON or XML. Whenever I use fetch() to

asynchronously communicate with a server without reloading the page, I am using the AJAX technique, because AJAX is a concept, not a specific technology.

In the middle column, all data from both side columns will be displayed.
In the right column, there will be "Towns" and a "Show More" button, which will trigger a function when clicked that displays additional information below, in this case, about castles in Heroes 3.
After clicking "Show More", the castles will appear in the centre using: <li> Castle and thanks to getAttribute("href"), it is assigned to a variable which is then used to fetch the JSON file. The data is then displayed using appendChild. Also, <template id="castleTemplate"> </template> will be used here to define a reusable HTML structure. It is not rendered by default, but allows JavaScript to clone and populate it dynamically.

Topic Information Page	
DOM Programming	Extensive DOM manipulation including dynamic content injection, template cloning, visibility toggling, and event handling for multiple UI components.
Templating System	Include (Header and footer)
MVC Pattern	The application follows the Model-View-Controller (MVC) architecture. All files related to the login page are isolated from other pages.
AJAX	Multiple asynchronous data fetches from external JSON files for dragon types and town information. Data is processed and injected into the page without reloading.
Security	session check for logged-in users – if user is not log in cant use this page.
Deployment	Everything show correctly

d) FAQ Page



The FAQ page also includes a background, a navigation bar, and a central content section. In the center section, the Frequently Asked Questions are displayed.
Each question is shown individually, with the option to expand or collapse the answer.

Topic Information Page	
DOM Programming	Dynamic answers toggling using JavaScript
Templating System	Include (Header and footer)
MVC Pattern	The application follows the Model-View-Controller (MVC) architecture. All files related to the login page are isolated from other pages.
AJAX	No but, static JavaScript array used.
Security	session check for logged-in users – if user is not log in cant use this page.
Deployment	Everything show correctly