THE MINISTRY OF EDUCATION AND RESEARCH
“1 DECEMBRIE 1918” UNIVERSITY OF ALBA IULIA
THE FACULTY OF INFORMATICS AND ENGINEERING
COMPUTER SCIENCE
FULL-TIME DEGREE PROGRAM

BACHELOR THESIS

COORDINATOR: **Prof. Arpad Incze**
GRADUATE: **Paul-Andrei Viziteu**

ALBA IULIA
2025

THE MINISTRY OF EDUCATION AND RESEARCH
“1 DECEMBRIE 1918” UNIVERSITY OF ALBA IULIA
THE FACULTY OF INFORMATICS AND ENGINEERING
COMPUTER SCIENCE
FULL-TIME DEGREE PROGRAM

WEBPAGE FOR VIDEO SHARING

COORDINATOR: **Prof. Arpad Incze**
GRADUATE: **Paul-Andrei Viziteu**

ALBA IULIA
2025

Table of Contents

INTRODUCTION

In the current digital era, online streaming platforms have become a cornerstone of how people consume video content. However, many of these platforms are bloated with excessive features, aggressive advertising, and complex interfaces that can overwhelm the average user. This often leads to a frustrating experience, especially for those who prefer a simple, clean, and efficient way to watch or share videos. This dissatisfaction with mainstream platforms served as the primary motivation behind the development of my thesis – a minimalist, user-friendly webpage for video sharing.

The idea for this application emerged during the second year of my university studies, where I acquired foundational knowledge in web development technologies. Using basic but effective tools such as HTML, CSS, JavaScript, PHP, and MySQL, I was able to design and implement a functional and accessible video-sharing platform. Although these technologies are considered introductory, they remain widely applicable and provide sufficient power to build robust applications when used thoughtfully.

General Objectives

The general objectives of this thesis and its associated application are:

- To design and develop a video-sharing webpage with minimalistic and intuitive user interface.
- To ensure core functionality such as video upload, playback, search, and browsing are implemented efficiently and without unnecessary features.
- To demonstrate that even basic web technologies can be used to build a functional streaming platform accessible to a wide range of users.
- To reinforce and apply the knowledge acquired during academic coursework in a practical, real-world scenario.

Research Methodology

The methodology used for this project is based on iterative, incremental development. The process began with an analysis of existing streaming platforms and identifying common user pain points. Based on these observations, I drafted functional requirements for a simplified video platform.

The development process followed standard software engineering practices:

- Requirements analysis and prototyping.
- Frontend and backend design using standard web technologies.
- Implementation of core features (video upload, display, search).
- Local testing, debugging, and deployment.
- Create adjustments based on feedback from peers.
- Managed version control using Git and collaborated on code through GitHub.

Possible Limitations

Despite the functional nature of the application, there are certain limitations to acknowledge:

- Scalability: Despite its functional nature, the current implementation of the webpage is primarily suitable for small-scale usage, such as personal projects or internal departmental applications, rather than high-traffic public environments. This limitation stems from several factors. Firstly, the database, a single MySQL/MariaDB instance, while robust for basic operations, could become a significant bottleneck under heavy concurrent user access or a rapidly growing video library. Future optimizations would necessitate strategies such as database sharding to distribute data across multiple servers or implementing read replicas to offload read operations. Secondly, the application runs on a single Apache server via XAMPP. For high-traffic scenarios, this architecture lacks the necessary redundancy and load-balancing capabilities. Implementing a reverse proxy or a load balancer would be crucial to distribute incoming requests across multiple application servers, enhancing both performance and availability. Furthermore, for efficient global video delivery, integrating a

Content Delivery Network (CDN) would be essential to reduce latency and bandwidth consumption by caching video content closer to the users.

- Feature Set: The intentional omission of advanced features such as user analytics, monetization options, and recommendation algorithm was a deliberate design choice to uphold the core principle of simplicity and counter the 'feature bloat' prevalent in mainstream streaming platforms. This focused approach allowed for the efficient development and implementation of essential functionalities like video upload, playback, search, and ensuring a straightforward and accessible user experience.

Future work

To evolve this minimalistic platform into a more comprehensive service, several key areas for future development could be explored:

- Server-Side Video Processing: For broader compatibility and optimized streaming, implementing server-side video encoding and transcoding to support various formats and adaptive bitrates would be essential. This would also involve exploring Content Delivery Networks (CDNs) which is a group of connected servers that cache and deliver content over the Internet [7].
- Expended Search and Filtering: Enhancing the search functionality to include filters by genre, release year, director would provide users with more precise content discovery tools.
- Enhanced Scalability and Performance: Implementing database sharding and replication would be vital for handling increased data volumes and user traffic. Integrating a load balancer and potentially containerizing the application using technologies like Docker would enable seamless horizontal scaling.

CHAPTER 1: THEORETICAL FOUNDATIONS AND RELATED WORK

1.1. Core Web Technologies for Dynamic Websites

Modern dynamic websites are built upon a foundation of core web technologies, primarily HTML, CSS, and JavaScript. HTML (Hypertext Markup Language) provides the structural framework of web pages, enabling developers to define headings, paragraphs, images, links, and multimedia content. CSS (Cascading Style Sheets) enhances this structure with design elements such as colors, fonts, spacing, layout positioning. JavaScript adds interactivity and client-side logic, allowing animations and asynchronous data loading.

These technologies work together within the client-server model, where the client (typically a browser) sends requests to a web server, which then processes those requests and returns appropriate responses. The browser renders the results using the aforementioned technologies, often incorporating additional scripts or styles dynamically.

1.2. Server-Side Scripting with PHP

PHP (Hypertext Preprocessor) is a widely-used open-source server-side scripting language.It is embedded within HTML code and executed on the server before the page is sent to the client.PHP plays a critical role in web development by enabling dynamic content generation, form handling, user authentication, session management, and database communication.

By integrating with Apache or Nginx web servers, PHP processes user inputs and generates HTML output, thereby acting as a mediator between the client and server. Its integration with MySQL databases allows for the creation of interactive, data-driven web applications.

PHP’s syntax is relatively easy to learn is well-suited for most kind of applications, which makes it ideal for educational and personal projects like the video-sharing webpage developed in this thesis.

1.3. Relational Database Management Systems: MySQL and MariaDB

MySQL and MariaDB are relational database management systems (RDBMS) widely used in web development due to their robustness, speed, and compatibility with server-side technologies. They use Structured Query Language (SQL) for defining, manipulating, and querying data.

These systems organize data into tables composed of rows and columns, supporting relationships between different data entities. In the context of a video-sharing platform, databases can be used to store metadata such as video titles, descriptions, thumbnails, video location, and user information. MariaDB, a fork of MySQL, offers improved performance and is fully compatible with its predecessor, making it a suitable replacement in most web environments.

1.4. Video Content on the Web: Formats and Delivery

Delivering video content on the web involves several considerations, including format compatibility, streaming method, and user accessibility. Common video formats include MP4(H.264/AAC), WebM, and Ogg. Mp4 is the most universally supported format due to its balance of compression and quality, which is why I decided to use it in my project.

The <video> HTML element embeds a media player which supports video playback into the document. It can be used for audio content as well [4]. The tag also can specify multiple sources within it which improved compatibility across browsers, letting me test for most of them.

For simple use cases, direct file hosting and embedding work well, though more complex platforms often require transcoding, adaptive streaming, and content delivery networks (CDNs).

1.5 Overview of Existing Video Sharing Platforms

Video sharing platforms like YouTube and Vimeo have transformed how users access and distribute media content. YouTube offers powerful recommendation algorithms, monetization options, and a vast content library. Vimeo, while more niche, is known for high-quality uploads and privacy controls.

Despite their popularity, these platforms include features such as targeted ads, subscriptions (which were meant to notify you about new videos from the content creator that you subscribed to, however they changed it and added another button that only appears after you subscribe and even that one doesn't automatically toggle notifications, you have to select to receive notifications, honestly this features bloat is ridiculous), and deep integration with user data, which can distract from a straightforward viewing experience. This project aims to focus on simplicity and ease-of-use, avoiding unnecessarily complex and bloated features commonly seen on these platforms.

CHAPTER 2: SYSTEM ANALYSIS AND DESIGN

2.1. Project Requirements

This project aims to deliver a lightweight video-sharing platform that enables users to view and download videos in a minimalist and user-friendly environment. The focus is on simplicity, reliability, and functionality without the use of heavy frontend frameworks or unnecessary features.

These requirements served as the foundational guidelines during the design, development, and testing phases, ensuring the final product aligns with the project's objectives and are split in two main categories: functional and non-functional requirements.

Functional requirements describe what the system must do to meet the user's needs. They detail the specific behaviors and the functionalities of the application.

The video-sharing platform shall provide the following core functionalities:

- User Authentication and Authorization:
  - The system shall allow new users to register for an account by providing a unique username and password.
  - The system shall enable registered users to log in to their accounts using their credentials.
  - The system shall provide a logout mechanism for authenticated users.
  - The system shall offer a 'remember me' option, utilizing cookies to keep user credentials remembered for a period of 30 days, upon user consent.
- Content Management and Display:
  - The system shall display individual pages for movies, allowing users to browse available film content.
  - The system shall display individual pages for television series, allowing users to browse episodic content.
  - The system shall feature a section for popular videos from any source, accessible from the main page.
  - The system shall support the playback of video files (e.g. MP4)
- Video Interaction:

- The system shall allow logged-in users to comment on individual movie and TV series videos.
- Each comment shall display the comment text, the username of the comment author, and the date and time the comment was posted.
- The system shall enable users to download videos.

- Search Functionality:
  - The system shall provide a search functionality that allows users to find movies and TV series by their title.
- User Video Upload:
  - The system shall allow authenticated users to upload video files.
  - Users shall be able to provide a title (required) and an optional description for their uploaded videos.
  - Users shall be able to upload a thumbnail image for their video.
  - The system shall support video uploads in '.mp4' and '.webm' formats, with a maximum files size of 20MB.
  - The system shall support thumbnail uploads in '.jpeg', '.jpg', and '.png' formats, with a maximum file size of 1MB.
  - Uploaded videos and thumbnails shall be stored on the local host in designated directories (videos/ and images/ respectively).
  - Metadata for user-uploaded videos (title, description, video URL, thumbnail URL and associated 'user_id') shall be stored in the 'featured_vidoes' table in the database.

Non-functional requirements describe how the system performs a particular function. They relate to quality attributes such as usability, performance, security, and compatibility.

The video-sharing platform shall adhere to the following non-functional requirements:

- Usability (User Experience – UX):
  - The User Interface (UI) shall be simple and intuitive, ensuring ease of navigation for all users.
  - All primary navigation links (e.g Home, TV Series, Movies, Login/Logout, Search) shall be consistently accessible at the top of each page.
- Performance:

- Pages shall load quickly, achieved through minimal use of the external dependencies and optimized code (Within 1 second on a broadband connection).

- Security:
  - The system shall implement robust input sanitization to prevent common web vulnerabilities.
  - The system shall utilize prepared statements to effectively prevent SQL injection attacks.
  - All user passwords shall be stored securely using one-way hashing algorithms (e.g 'password_hash()' in PHP) to protect user credentials. Password hashing plays a central role in the design of secure systems. We store a password hash in lieu of a password for authentication purposes [15].
  - The system shall limit file upload sizes to prevent dos (denial-of-service attacks) through application level checks [14].
- Compatibility and Responsiveness:
  - The webpage shall be responsive, providing a pleasant and consistent user experience across various devices, including phones and tablets, through the use of media queries and flexible layouts.
- Reliability:
  - The system shall handle common errors gracefully, providing informative feedback to the user without crashing.

2.2. Technology Stack Justification

The chosen technology stack prioritizes simplicity, ease of deployment, and compatibility with standard development environments:

- PHP is a used since it is a popular scripting language for server-side logic. It is easy to integrate with HTML and provides robust features for handling forms, file uploads, and sessions.
- MySQL/MariaDB offers a reliable, fast, and relational database model well-suited for storing structured data like users and videos.

- Apache via XAMPP allows rapid setup of local server that supports PHP and MySQL out of the box, simplifying development and testing.
- HTML/CSS/JavaScript represent the fundamental technologies for frontend structure, design, and interactivity. Their simplicity makes them easy to maintain and modify.
- These technologies are foundational for web development and were covered during coursework, reinforcing practical understanding.

2.3. System Architecture

The system is designed to follow a three-tier architecture:

- Client Layer: Runs in the user's web browser, using HTML/CSS/JavaScript to render pages and collect user inputs.
- Application Layer: Handled by Apache and PHP scripts, responsible for routing requests, validating inputs, and interacting with the database.
- Data Layer: Managed by MySQL/MariaDB, where user accounts and video metadata are stored.

Component Interactions:

- The client sends a request (e.g. to log in).
- PHP scripts validate and process the request.
- Validated data is inserted or retrieved from the database.
- A response is generated and sent back to the client (e.g. confirmation or error message).

This layered approach supports code separation, better maintainability, and even future scalability.

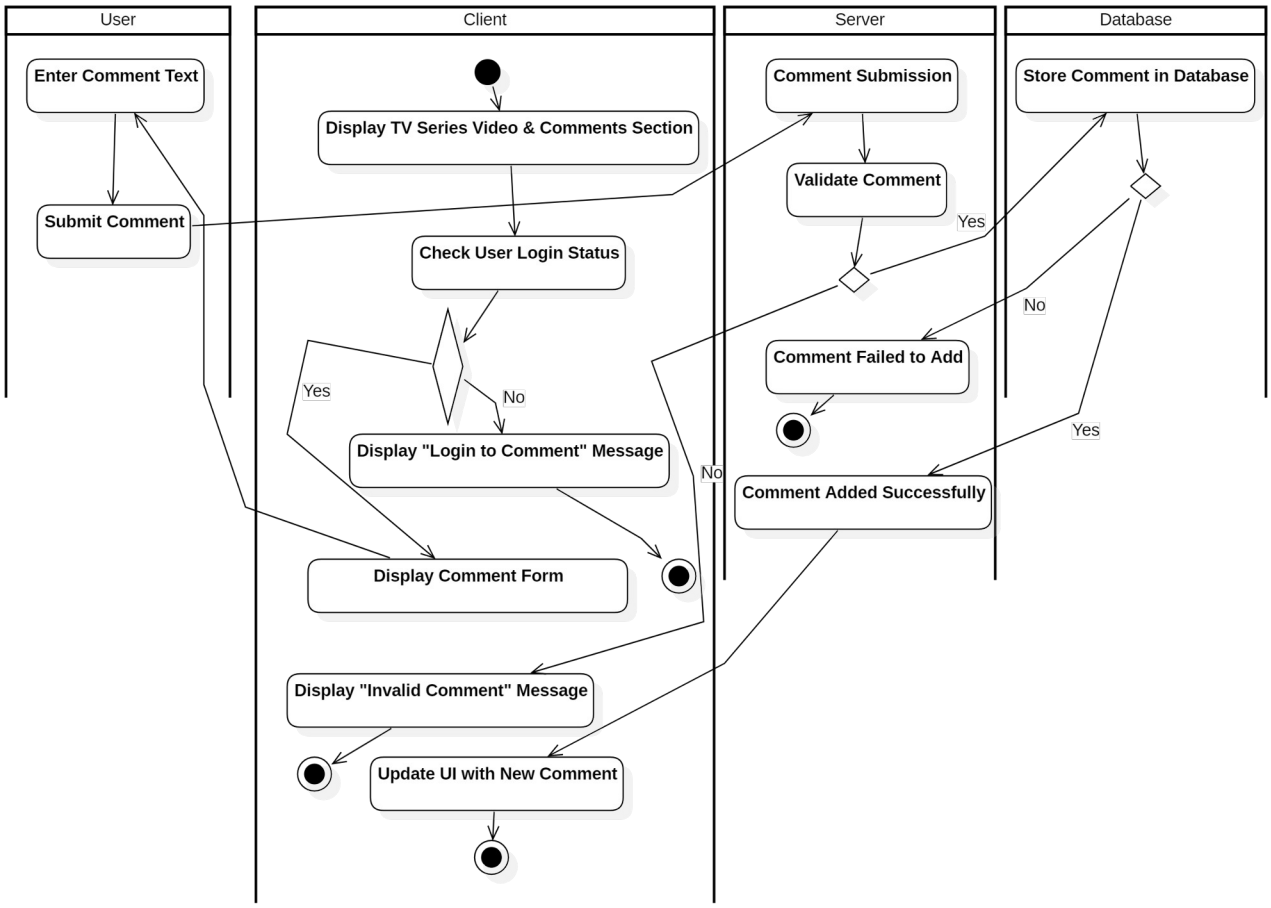

Figure 1: Activity Diagram of User commenting on a Movie/TV series

2.4. Database Design

The database serves as the persistent storage layer for the project, responsible for securely managing user accounts, organizing video content, and storing user comments. The chosen database management system for this project is MySQL/MariaDB, a widely used open-source relational database, which integrates seamlessly with PHP through the XAMPP environment. The design prioritizes data integrity, consistency, and efficient retrieval of the information.

The database schema is structured around several key tables, each serving a specific purpose:

- ‘users’ Table:

- Purpose: Stores information about registered users of the platform.
- Attributes:
  - ‘id’ (INT, Primary Key, AUTO_INCREMENT): A unique identifier for each user.
  - ‘username’ (VARCHAR(255), UNIQUE): The user’s chosen username, enforced to be unique to prevent duplicates.
  - ‘password’ (VARCHAR(255)): Stores the hashed password of the user. Hashing ensures that plain-text passwords are never stored, significantly enhancing security.
  - ‘registration_date’ (TIMESTAMP): The timestamp of when the user account was created.
- Indexing: An index on ‘username’ allows for fast lookups during login.

- ‘movies’ Table:
  - Purpose: Stores detailed information about movie content available on the platform.
  - Attributes:
    - ‘id’ (INT, Primary Key, AUTO_INCREMENT): A unique identifier for each movie.
    - ‘title’ (VARCHAR(255)): The title of the movie.
    - ‘description’ (TEXT): A brief synopsis of the movie.
    - ‘director’ (TEXT): The director of the movie.
    - ‘rating’ (DECIMAL): The rating of the movie.
    - ‘genre’ (VARCHAR(100)): The genre of the movie.
    - ‘release_year’ (INT): The year the movie was released.
    - ‘video_url’ (VARCHAR(255)): The URL or path to the movie’s video file.
    - ‘thumbnail_url’ (VARCHAR(255)): The URL or path to the movie’s thumbnail file.
  - Indexing: Indexes on ‘id’ and ‘title’ support efficient retrieval and search operations.
- ‘tv_series’ Table:
  - Purpose: Stores detailed information about TV series content available on the platform.
  - Attributes:
    - ‘id’ (INT, Primary Key, AUTO_INCREMENT): A unique identifier for each TV series.
    - ‘title’ (VARCHAR(255)): The title of the TV series.
    - ‘description’ (TEXT): A brief description of the TV series.
    - ‘creator’ (VARCHAR(255)): The creator of the TV series.

        - ‘genre’ (VARCHAR(100)): The genre of the TV series.
        - ‘video_url’ (VARCHAR(255)): The URL or path to the TV series’ video file.
        - ‘thumbnail_url’ (VARCHAR(255)): The URL or path to the TV series’ thumbnail file.
    - Indexing: Similar to ‘movies’, indexes on ‘id’ and ‘title’ ensure efficient data access.
- ‘comments’ Table:
    - Purpose: Stores user-submitted comments associated with specific videos.
    - Attributes:
        - ‘id’ (INT, Primary Key, AUTO_INCREMENT): A unique identifier for each comment.
        - ‘video_id’ (INT): The ID of the video (either from ‘movies’ or ‘tv_series’) the comment belongs to.
        - ‘video_type’ (VARCHAR(50)): Specifies whether the ‘video_id’ refers to a move or a TV series. This design choice allows for associating comments with content from different tables.
        - ‘user_id’ (INT): The ID of the user who posted the comment.
        - ‘username’ (VARCHAR(50): The name of the user who posted the comment.
        - ‘comment’ (TEXT): The actual text content of the comment.
        - ‘comment_date’ (TIMESTAMP): The timestamp when the comment was posted, automatically set upon insertion.
    - Indexing: Indexes on ‘video_id’ and ‘user_id’ facilitate quick retrieval of comments for a specific video or by a specific user.
- ‘featured_videos’ Table:
    - Purpose: Stores a list of user uploaded videos to be displayed on the homepage.
    - Attributes:
        - ‘‘id’ (INT, Primary Key, AUTO_INCREMENT): A unique identifier for each featured video.
        - ‘title’ (VARCHAR(255)): The title of the featured video.
        - ‘description’ (TEXT): A brief description of the featured video.
        - ‘category’ (VARCHAR(100)): The category of the featured video.
        - ‘video_url’ (VARCHAR(255)): The URL or path to the featured video file.

- ‘thumbnail_url’ (VARCHAR(255)): The URL or path to the featured video thumbnail file.
- ‘created_at” (TIMESTAMP): The timestamp when the video was uploaded, automatically set upon insertion.
- ‘user_id’ (INT): The identifier of the user who uploaded the video.

- ‘remember_me’ Table:
  - Purpose: Stores tokens for ‘remember me’ functionality, allowing users to stay logged in across sessions.
  - Attributes:
    - ‘id’ (INT, Primary Key, AUTO_INCREMENT): A unique identifier for each ‘remember me’ token.
    - ‘user_id’ (INT): The ID of the user associated with the token.
    - ‘token’ (VARCHAR(255)): The unique token used for authentication.
  - Indexing: A unique index on ‘token’ ensures that each token is distinct, and an index on ‘user_id’ allows for efficient lookup of tokens belonging to a specific user.

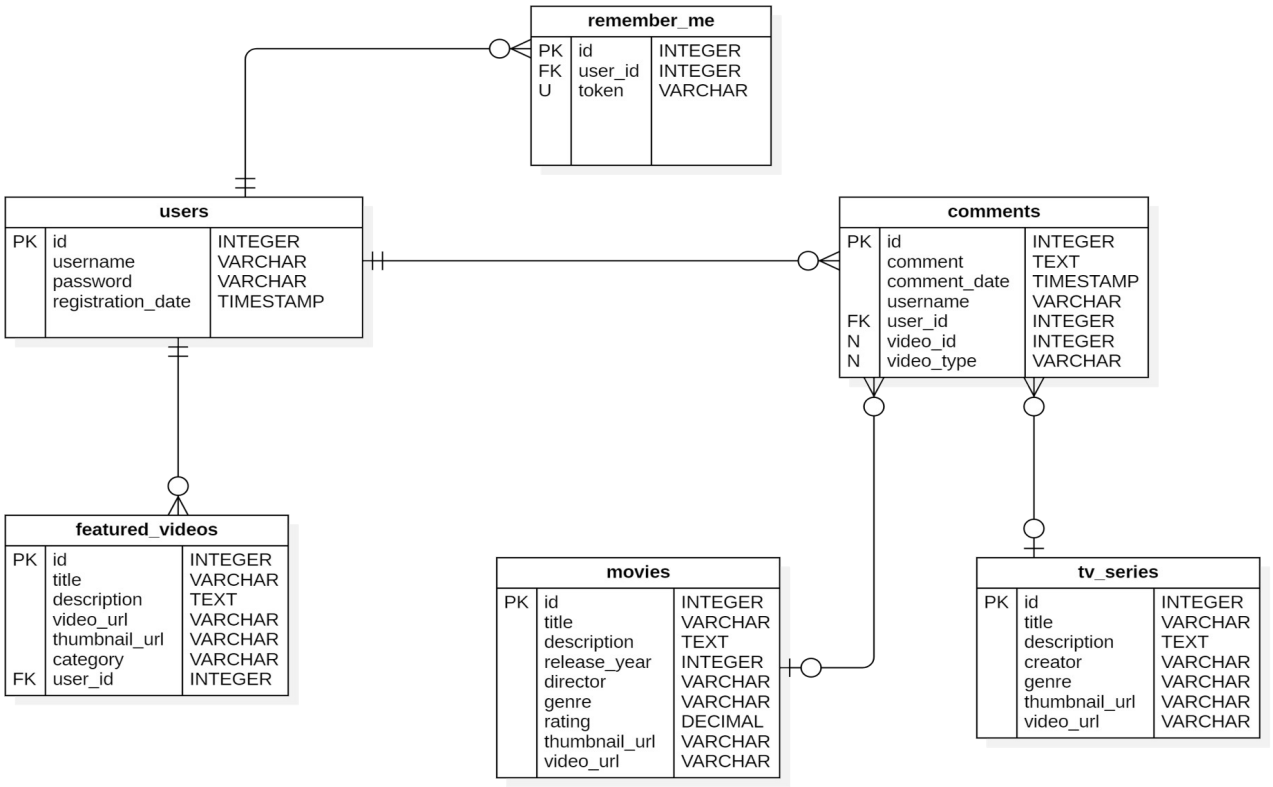

Figure 2: Entity-Relationship Diagram

2.5. User Interface (UI) and User Experience (UX) Design Considerations

The user interface is intentionally minimal to reduce distractions and emphasize functionality such that even the most non-tech user can manage to navigate the website.

The registration and login page have a clean form layout with only the required fields, the featured videos page has six videos shown in two groups of three videos each, the television series and movies pages each have similar layout with the featured videos however they use just the thumbnails that redirect to the actual video once clicked, the actual video page where movies and television series thumbnails redirect to contains the video in big picture with two buttons at the bottom to go to the previous or next video (if it exists) and below is the comment section which is separate for each video and can be seen by a user but comments can only be created by authenticated users.

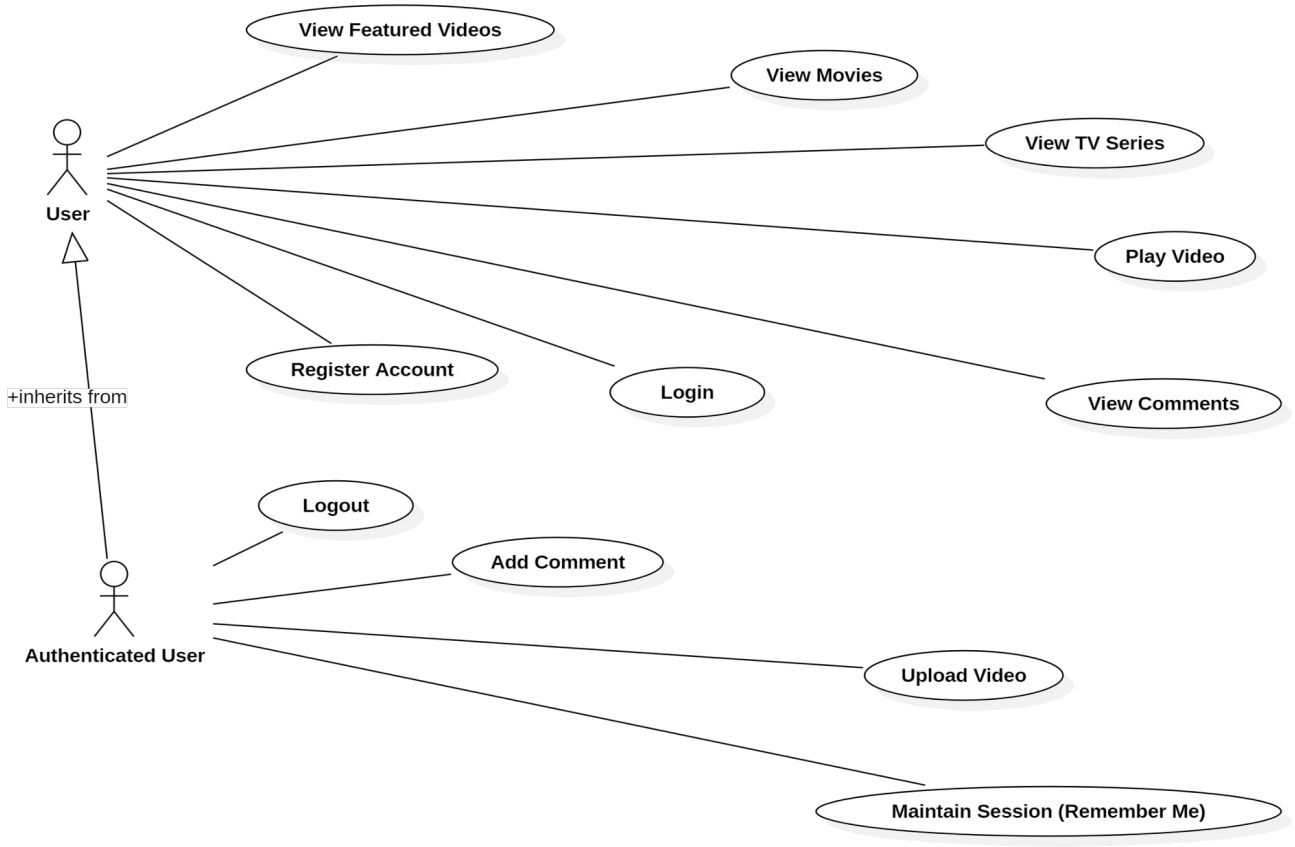

Figure 3: Use Case Diagram

CHAPTER 3: IMPLEMENTATION DETAILS

3.1. Development Environment Setup

The project was developed using XAMPP, a cross-platform package that includes Apache, MariaDB, and PHP.

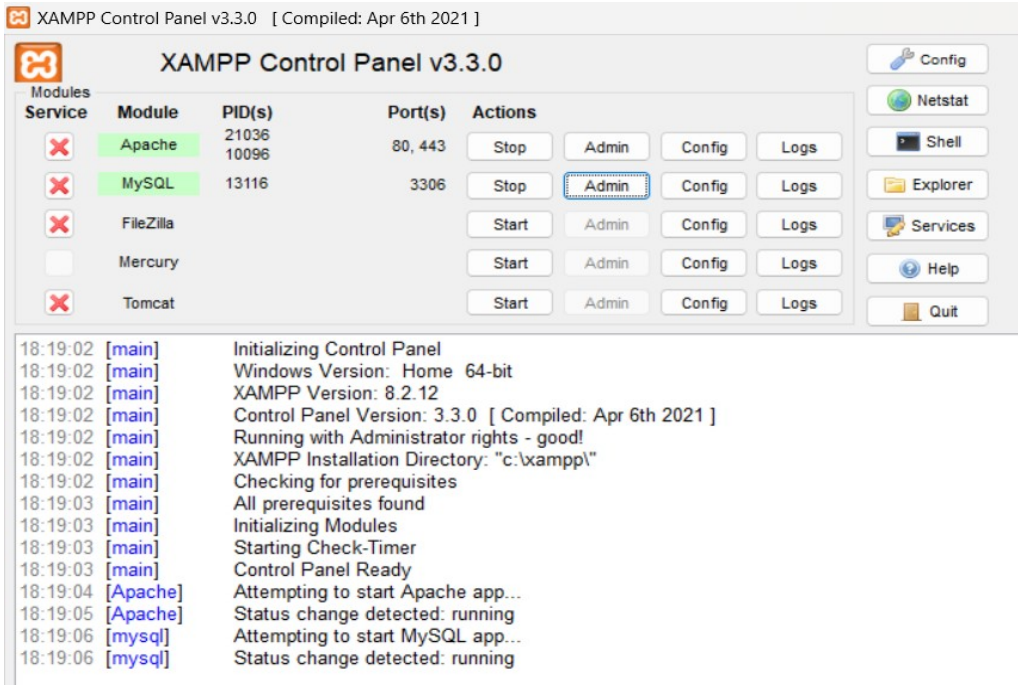

Figure 4: XAMPP Control Panel

The server was hosted locally during development to ensure a controlled environment and not incur obscene hosting costs. The database was managed with the ‘phpmyadmin’ web-tool to create, read, update, and delete records and tables.

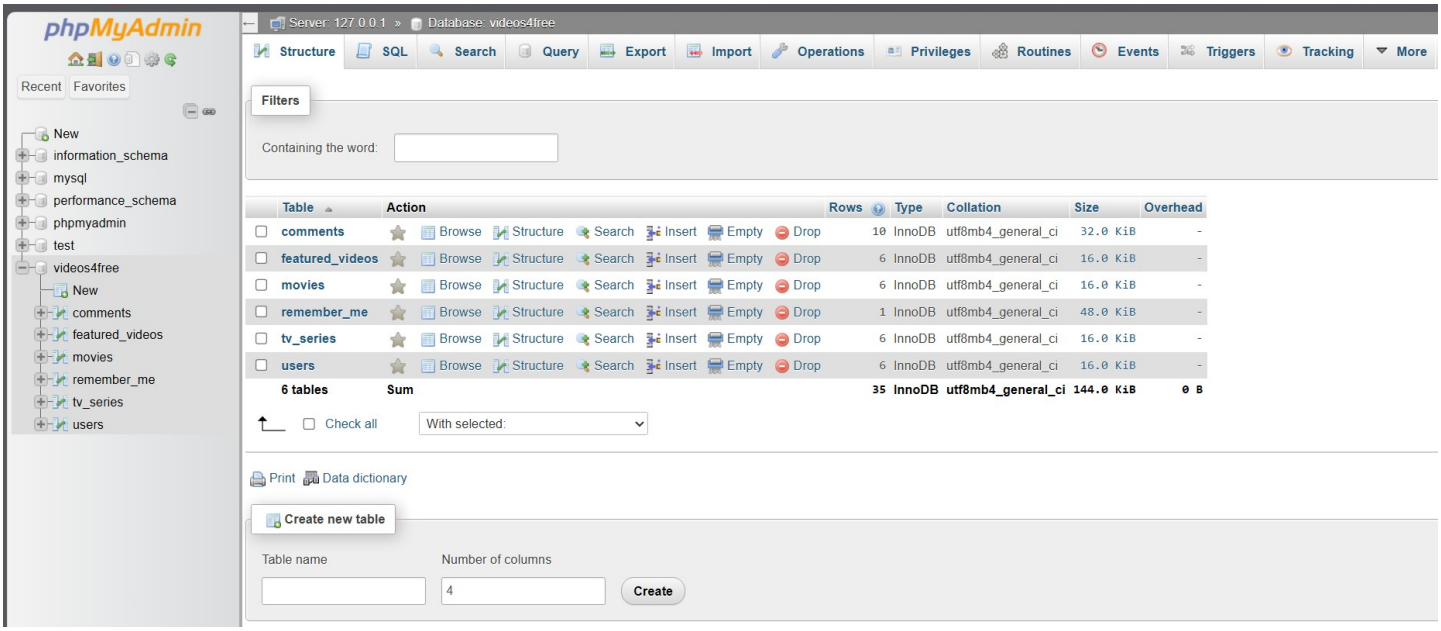

Figure 5: Database Management on Localhost through Phpmyadmin

The project files are organized under the ‘htdocs’ directory with subfolders for images, videos and php/css/js files.

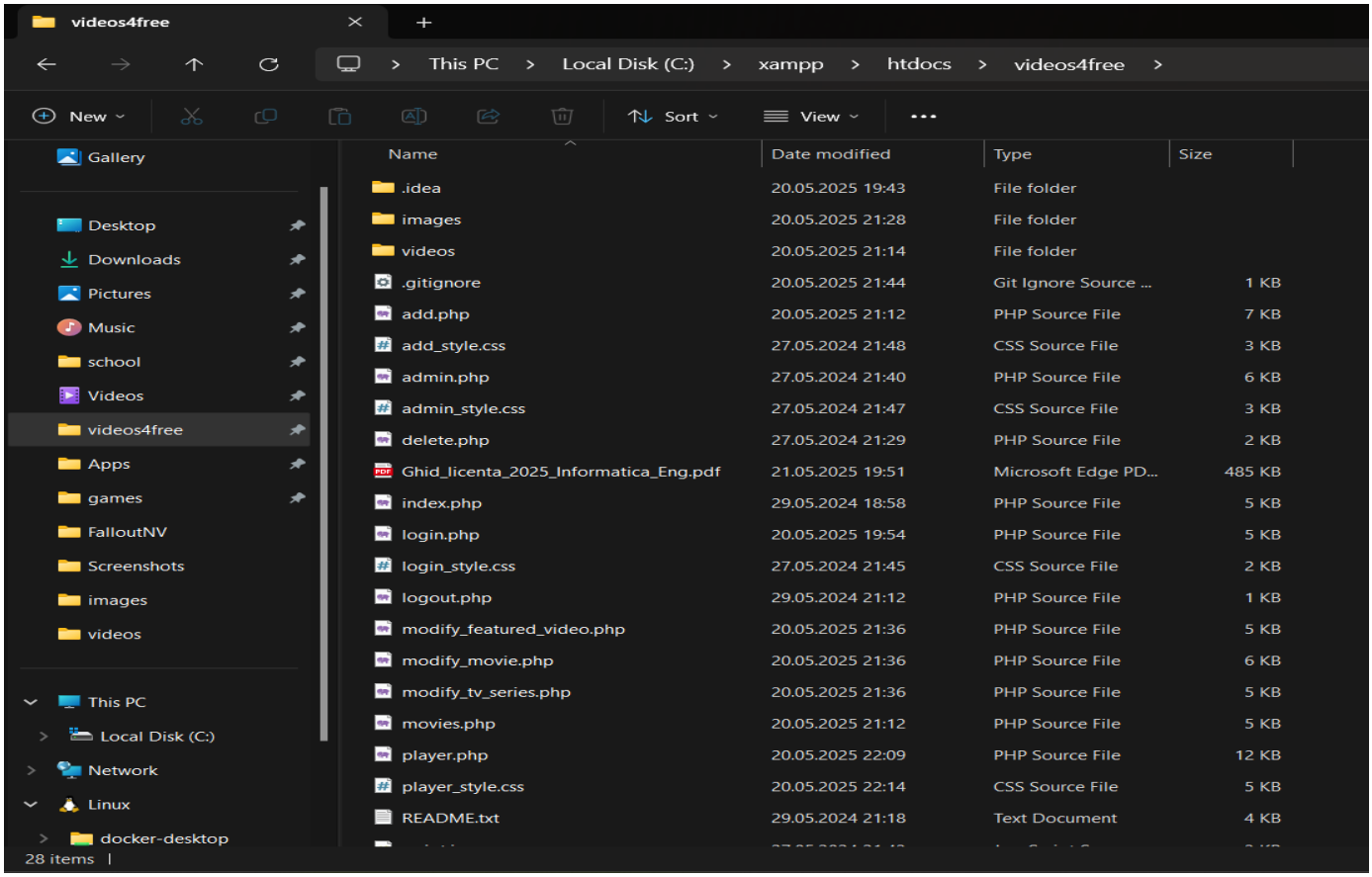

Figure 6: Project Directory and Subfolders Structure

3.2. Backend Implementation with PHP and MySQL/MariaDB

The backend consists of PHP scripts responsible for managing user actions, processing uploads, and interfacing with the database. PHP was chosen due to its simplicity and widespread use in academic projects and because it was taught in my second year of university quite extensively, making me very familiar with it's logic and syntax.

Example of PHP backend usage to show featured videos on main page in Visual Studio Code:

```
<?php
session_start();
```

```
?>

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="images/icon1.png" type="image/x-icon">
  <title>Webpage for video sharing</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <header>
    <a href="index.php"><img src="images/logo.png" alt="Logo" id="logo"></a>
    <img src="images/menu.png" alt="Menu" id="menu">
    <nav class="desktop_menu">
      <ul>
        <li><a href="index.php">Home</a></li>
        <li><a href="tv_series.php">TV Series</a></li>
        <li><a href="movies.php">Movies</a></li>
        <?php
        if (isset($_SESSION['username'])) {
          echo "<li><a href='upload.php'>Upload Video</a></li>";
          echo "<li><a href='logout.php'>Logout</a></li>";
        }
        else {
          echo  "<li><a href='login.php'>Login</a></li>";
          }
        ?>
        <li><a href="search.php">Search</a></li>
      </ul>
    </nav>
    <nav id="mobile_menu">
      <img src="images/close.png" id="Close" alt="404closenotfound">
      <ul>
        <li class="mobile_ui"><a href="index.php">Home</a></li>
        <li class="mobile_ui"><a href="tv_series.php">TV Series</a></li>
        <li class="mobile_ui"><a href="movies.php">Movies</a></li>
        <li class="mobile_ui"><a href="search.php">Search</a></li>
        <?php
        if (isset($_SESSION['username'])) {
          echo "<li class='mobile_ui'><a href='upload.php'>Upload Video</a></li>";
```

```
            echo "<li class='mobile_ui'><a href='logout.php'>Logout</a></li>";
        } else {
            echo "<li class='mobile_ui'><a href='login.php'>Login</a></li>";
        }
        ?>
      </ul>
    </nav>
</header>
<main>
    <section>
      <h1>Welcome to our Video Sharing/Streaming Website</h1>
      <p>We provide high-quality content for your viewing pleasure free of charge!</p>
    </section>
    <section>
      <h2>Featured Videos</h2>
      <p>Check out some of our latest user uploads:</p>
      <article>
        <?php
        // Connect to your database
        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = "webpage_for_video_sharing";

        $conn = new mysqli($servername, $username, $password, $dbname);

        // Check connection
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

        // Get the latest 3 videos (general featured videos)
        $sql_featured = "SELECT * FROM featured_videos ORDER BY created_at DESC LIMIT 3";
        $result_featured = $conn->query($sql_featured);

        echo "<ul class='featured-videos' id='general_featured_videos'>";
        // Show the videos
        if ($result_featured->num_rows > 0) {
            while ($row = $result_featured->fetch_assoc()) {
                echo "<li class='slide'><video controls poster='" . $row["thumbnail_url"] . "'>";
                echo "<source src='" . $row["video_url"] . "' type='video/mp4'>";
                echo "<source src='" . $row["video_url"] . "' type='video/webm'>";
                echo "Your browser does not support the video tag.";
                echo "</video><small>" . htmlspecialchars($row["title"]) . "</small></li>";
            }
```

```
            } else {
                echo "<p>No featured videos available.</p>";
            }
            echo "</ul>";

            // Your latest video uploads (user-specific)
            if (isset($_SESSION['user_id'])) { // Check for user_id in session
                echo "<p>Your latest video uploads:</p>"; //
                $user_id = $_SESSION['user_id'];
                    $sql_user_uploads = "SELECT * FROM featured_videos WHERE user_id = ? ORDER BY created_at DESC
LIMIT 3"; //
                $stmt_user_uploads = $conn->prepare($sql_user_uploads);
                $stmt_user_uploads->bind_param("i", $user_id);
                $stmt_user_uploads->execute();
                $result_user_uploads = $stmt_user_uploads->get_result();

                echo "<ul class='featured-videos user-uploads' id='user_uploads_videos'>";
                if ($result_user_uploads->num_rows > 0) {
                    while ($row = $result_user_uploads->fetch_assoc()) {
                        echo "<li class='slide'><video controls poster='" . $row["thumbnail_url"] . "'>";
                        echo "<source src='" . $row["video_url"] . "' type='video/mp4'>";
                        echo "<source src='" . $row["video_url"] . "' type='video/webm'>";
                        echo "Your browser does not support the video tag.";
                        echo "</video><small>" . htmlspecialchars($row["title"]) . "</small></li>";
                    }
                } else {
                    echo "<p>You haven't uploaded any videos yet.</p>";
                }
                echo "</ul>";
                $stmt_user_uploads->close();
            }

            $conn->close();
            ?>
            <div id="buttons">
                <button id="previous"><img src="images/play.png" alt="p" id="previous_play"></button>
                <button id="next"><img src="images/play.png" alt="p" id="next_play"></button>
            </div>
        </article>
    </section>
</main>
<script src="script.js"></script>
</body>

</html>
```

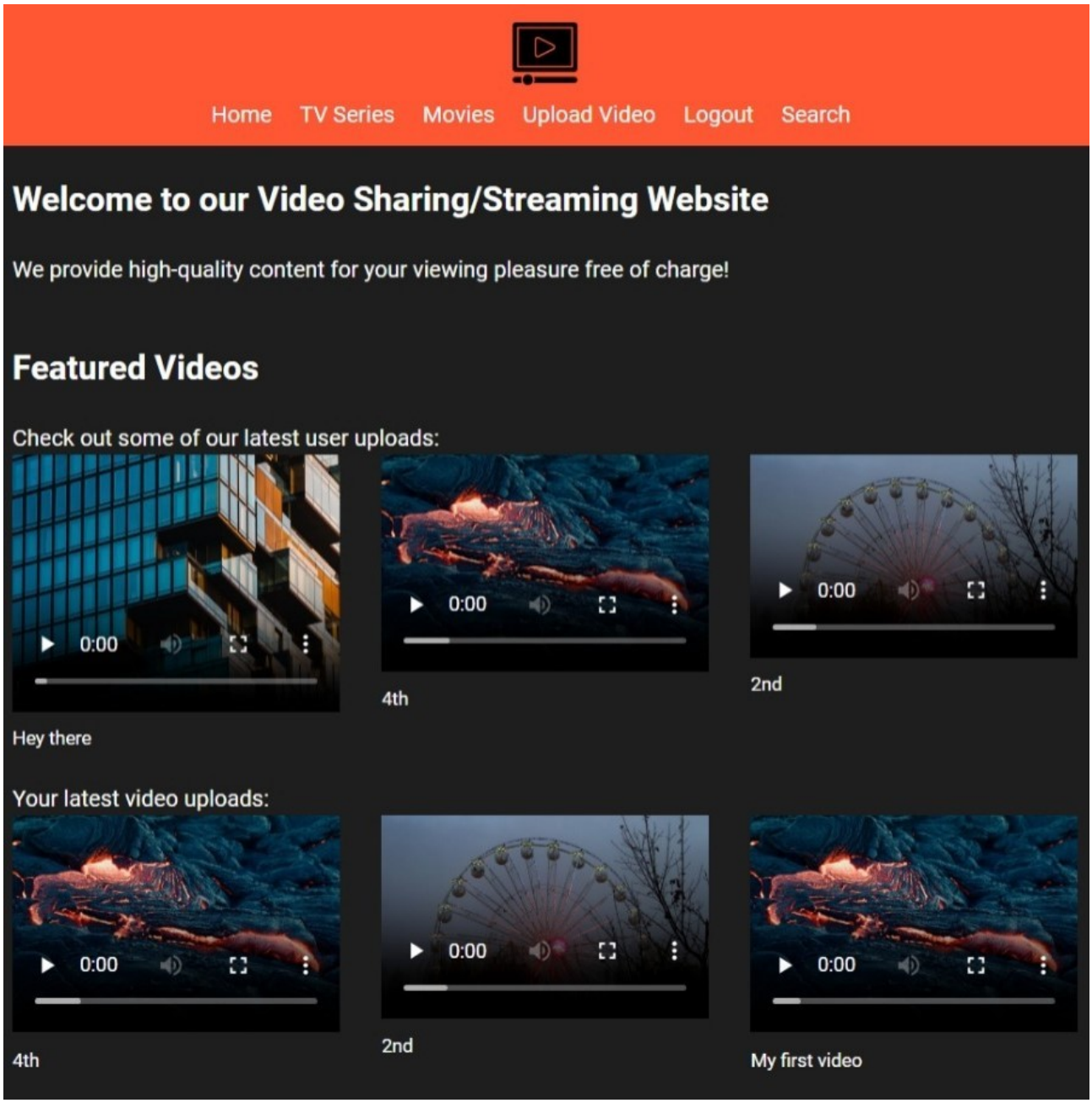

Figure 7: Main Page with Featured Videos

3.3. User Authentication and Session Management

An authentication system allows users to register and log in. Passwords are hashed using ‘password_hash()’ and stored in the database. During login, the provided password is verified against the stored hash using ‘password_verify()’ to confirm the user’s identity.

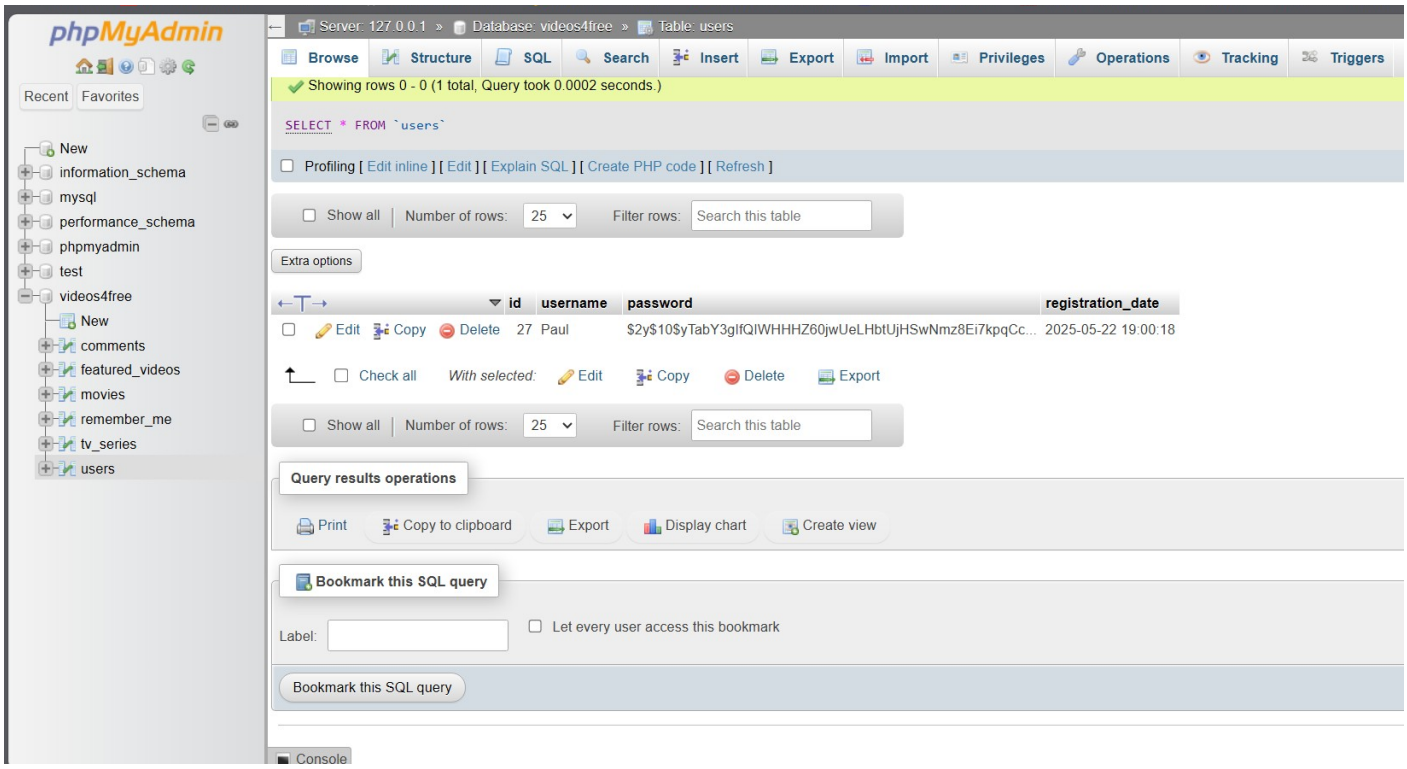

Figure 8: User Password Hashed in Database after Registering

Sessions are created using ‘session_start()’ which looks for a session file if one was previously created on any page otherwise it creates an empty session file on the server for this new session and stores data from arrays like ‘$_SESSION[‘username’]’ until the user ends the session (e.g. user logs out, closes browser).

3.4. Frontend Implementation with HTML, CSS and JavaScript

The frontend uses HTML and CSS for layout and design, and JavaScript is used for mobile menu interactions:

- HTML plays a fundamental role in defining the structure and content of the page. The <head> tag contains meta-information about the HTML document, not directly displayed on the page such as the document type, the character encoding, the primary language, and the

title. The <section> and <article> tags are used to group related content thematically within the document, similar to chapters in a book.

- CSS is used to maintain a consistent look across all pages. Flexbox and classes are used provide structure and spacing.
- JavaScript is like an ‘engine’ in this project, making parts of static HTML pages move, react, and change without needing a full page reload from the server. It handles two main functionalities:
  - Controlling the opening and closing of mobile navigation menu.
  - Assisting in addition of new videos by hiding or showing certain fields depending on the type of video uploaded by the admin.

3.5. Streamlined Content Management System

The administrative hub, ‘admin.php’, serves as the central dashboard for the admin. It presents a consolidated view of all managed content, neatly organized into three sections: Featured Videos, Movies, and TV Series. For each content type, it displays key details in a tabular format, such as title, description, and category-specific information like director or creator. Crucially, each entry is accompanied by ‘Edit’ and ‘Delete’ options, allowing administrators to manage individual items directly. The page also features a prominent ‘Add New Content’ button, streamlining the process of introducing new material to the platform.

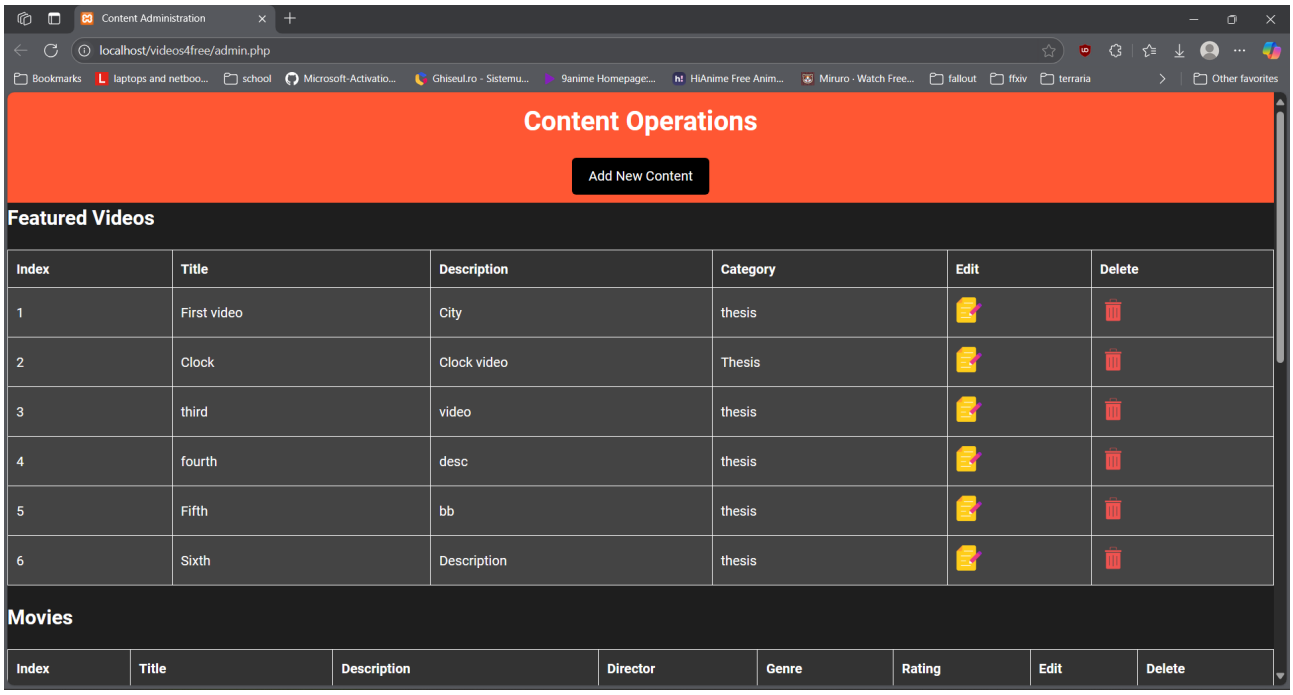

Figure 9: Admin Page for Content Management

The ‘add.php’ script provides the functionality for introducing new content. It presents a dynamic form where the administrator first selects the type of content to be added (Featured Video, Movie, or TV Series). Based on this selection, the form intelligently reveals the relevant fields. Common fields like ‘Title’ and ‘Description’ are always present. For ‘Featured Videos’, specific fields such as ‘Video URL’, ‘Thumbnail URL’, ‘Category’, and an ‘Is Featured’ checkbox appear. Similarly, ‘Movies’ have fields for ‘Release Year’, ‘Director’, ‘Rating’, and ‘Genre’, in addition to ‘Thumbnail URL’ and ‘Video URL’. ‘TV Series’ require ‘Thumbnail URL’, ‘Video URL’, ‘Creator’, and ‘Genre”. Upon submission, the script securely inserts new data into the corresponding database table, employing prepared statements to safeguard against SQL injection vulnerabilities.

Content modification is handled by three specialized scripts: ‘modify_featured_videos.php’, ‘modify_movie.php’, and ‘modify_tv_series.php’. When an administrator clicks an ‘Edit’ link on the ‘admin.php’ page, they are directed to the appropriate modification script, with the specific content item’s ID passed along. These scripts first retrieve the existing data for the selected item from the database. This information is then used to pre-populate a form, allowing the administrator to easily view and alter the details. Fields available for editing mirror those in the ‘add’ functionality, including title, description, URLs, and other metadata specific to the content type.

Once changes are submitted, the script updates corresponding record in the database, again utilizing prepared statements for secure data handling. A success message is displayed, along with a link to return to the main content administration page.

The ‘delete.php’ script is responsible for removing content entries. When a ‘Delete’ link is activated in ‘admin.php’, this script is called with the ID of the item to be deleted. It then executes deletion queries across all three content tables (featured videos, movies, and tv series) for the provided ID. This approach ensures that if an ID somehow exists in multiple tables the is removed. The script then redirects the administrator back to the ‘admin.php’ page reflecting the updated content list.

CHAPTER 4: TESTING AND EVALUATION

This chapter details the systematic approach taken to ensure the quality, functionality, and performance of the developed video-sharing webpage. It covers the testing methodologies employed, the types of tests conducted, the tools utilized, and the evaluation of the system's compliance with the defined requirements and objectives.

4.1. Testing Approach

The testing of the project application was conducted using a combination of manual and automated testing techniques, following an iterative and incremental approach, consistent with the development methodology outlined in Chapter 3. This approach allowed for continuous feedback and refinement throughout the development lifecycle, ensuring that defects were identified and addressed early.

The primary objectives of the testing phases were to:

- Verify that all implemented features function correctly according to the specified requirements.
- Ensure the application is robust and handles erroneous inputs or unexpected scenarios gracefully.
- Assess the user interface's intuitiveness and responsiveness across different devices and browsers.
- Evaluate the overall performance and the efficiency of the application.
- Identify and document any bugs or areas for improvement.

4.2. Testing Methodologies

Given the scope and nature of this project, a Black-Box Testing approach was primarily adopted. This methodology focuses on testing the application's functionality without knowledge of its internal code structure. Testers (mostly colleagues) interact with the user interface and various

functionalities (e.g. user registration, video playback, search) to verify expected outputs based on given inputs.

Complementing black-box testing, Ad-Hoc Testing was also performed, particularly during the early stages of development (also mostly with the help of colleagues). This informal testing allowed for rapid identification of obvious defects and usability issues, fostering quick iterations and improvements.

4.3. Types of Tests Conducted

The following types of testes were systematically performed to cover different aspects of the application's quality:

- Functional Testing:
  - Unit Testing (Informal): While formal unit testing frameworks were not explicitly used for every small code segment, individual PHP functions (e.g. password hashing, database interactions) and JavaScript client-side scripts were tested in isolation to ensure they performed their intended operations correctly.
  - Integration Testing: This involved testing the interactions between different modules and components of the application. Key integration points tested included:
    - User registration and login (interaction between frontend forms, PHP backend, and MySQL database).
    - Video playback (interaction between 'player.php' script and the stored video URLs/Thumbnails).
    - Comment submission and display.
    - Search functionality (frontend search bar with backend database queries).
  - System Testing: This focused on testing the complete and integrated software system to evaluate the system's compliance with its specified requirements. All user flows, from a new registering to an existing user watching and commenting on a video, were tested end-to-end.

- Regression Testing: As new features were added or bugs were fixed, regression tests were conducted to ensure that existing functionalities were not negatively impacted by the changes. This involved re-running a set of core tests.

- Non-Functional testing:
  - Usability Testing: The user interface (UI) was assessed for its intuitiveness, ease of navigation, and overall user experience (UX). Feedback was informally gathered during personal trials and adjustments were made to improve clarity and simplicity, aligning with the thesis's minimalist design philosophy.
  - Compatibility Testing: The application was tested across different web browsers (e.g. Google Chrome, Mozilla Firefox, Microsoft Edge) to ensure consistent rendering and functionality. Basic responsiveness was also tested on various screen sizes to verify the mobile menu and layout adjustments.
  - Security Testing: Focus as placed on common web vulnerabilities relevant to the technologies used:
    - Input Validation: Forms (login, signup, comment submission) were tested with various inputs to ensure proper data validation and sanitization (e.g. use of 'htmlspecialchars' and prepared statements to prevent XSS and SQL injection).
    - Authentication: The password hashing mechanism ('password_hash()', 'password_verify()'), was verified to ensure secure storage and comparison of credentials.
    - Session Management: Session handling for logged-in users was checked to ensure proper session creation, destruction, and prevention of unauthorized access to the restricted pages.
  - Performance Testing (Informal): Although formal load testing tools were not used, the application's responsiveness and page load times were informally observed under typical usage scenarios. The choice of lightweight technologies contributed to generally fast loading times, as highlighted in Chapter 3.

CONCLUSION

This project successfully developed a minimalist, user-friendly webpage for video sharing, demonstrating the practical application of foundational web development technologies. Through its design and implementation, I've gained a comprehensive understanding of basic full-stack development, integrating client-side technologies like HTML, CSS, and JavaScript with server-side scripting (PHP) and database management (MySQL/MariaDB) to create a functional web application.

One of the most challenging, yet rewarding, aspects of this development was implementing the featured video section for mobile devices. This involved dynamically displaying only one video at a time from two distinct video carousels and synchronizing their navigation. Achieving this required careful coordination between CSS media queries for responsive layout and intricate JavaScript logic to manage two separate video arrays and a unified navigation system, where a single set of 'next' and 'previous' buttons controlled both carousels simultaneously. This particular challenge significantly deepened my problem-solving skills and my ability to create interactive and responsive user experiences, showcasing the power and flexibility of core web technologies in addressing complex interface requirements.

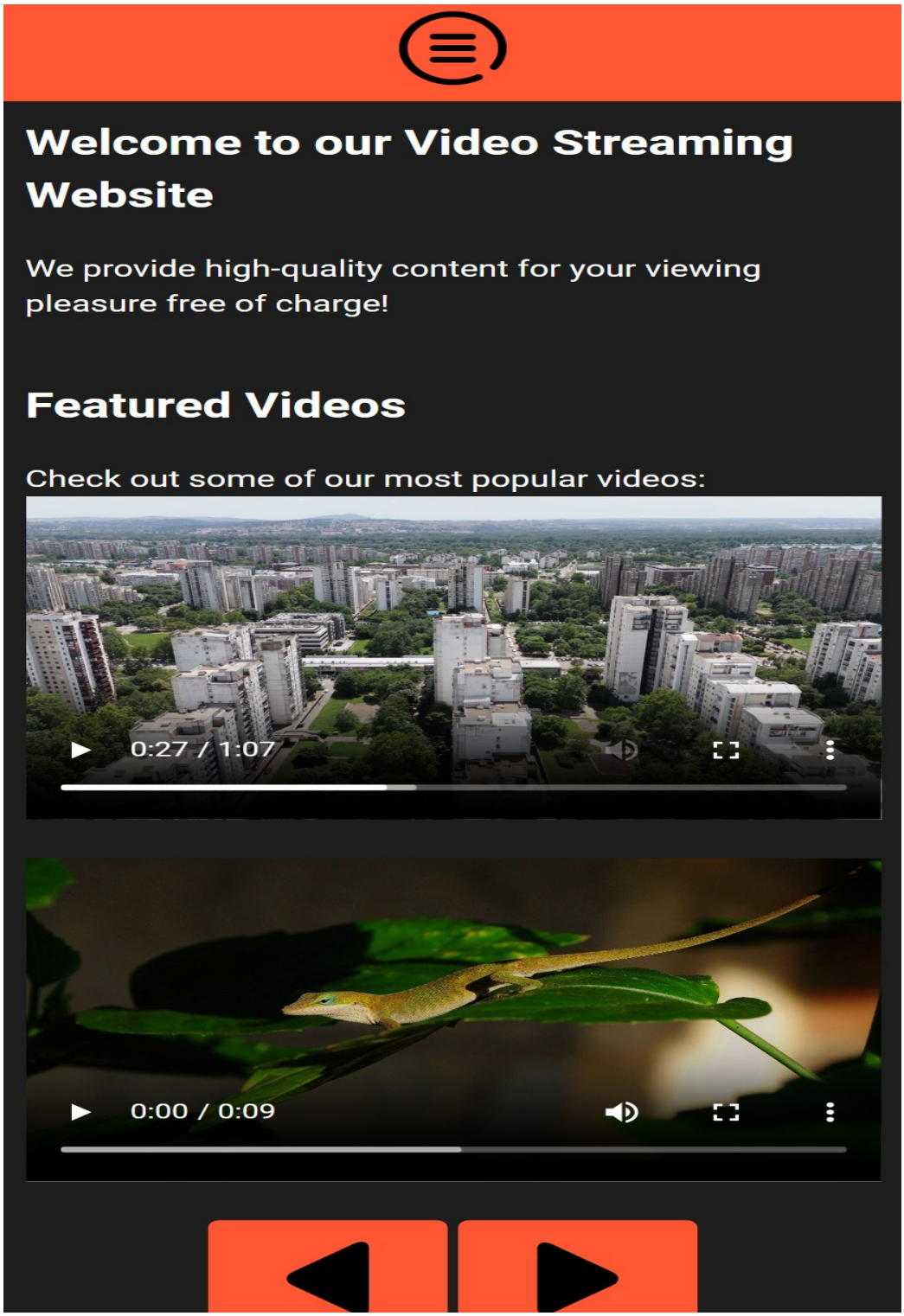

Figure 10: Featured Videos in Mobile Menu

Beyond the application's core functionality, the development process itself served as a significant learning experience in professional software practices. Utilizing GitHub for version

control was instrumental in managing code changes and maintaining a historical record of the project's evolution. While leveraging GitHub was ultimately beneficial, the initial integration with Visual Studio Code presented its own set of difficulties. As this was my first time directly linking a local development environment to a remote repository, much of the online documentation and guides available proved outdated, leading to struggles with authentication methods and workflow setup. Overcoming these hurdles required persistent research and experimentation, significantly improving my understanding of Git commands, repository management, and the crucial role of version control in individual development projects.

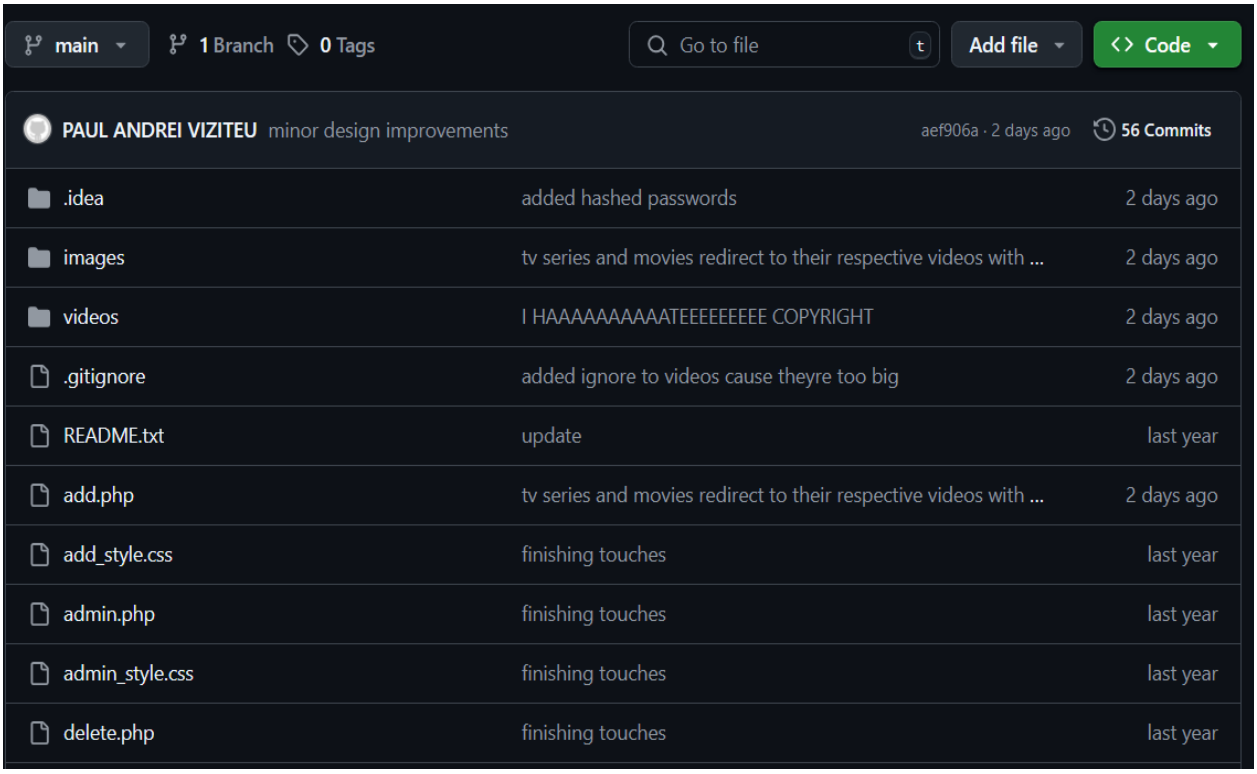

Figure 11: GitHub Repository for Version Control

The iterative development process, from requirements analysis to implementation and testing, provided valuable experience in software engineering practices. The project not only reinforced theoretical knowledge acquired during academic coursework but also offered practical insights into building a robust, albeit small-scale, video sharing/streaming platform focused on simplicity and essential functionalities.

BIBLIOGRAPHY

[1] Git, “About Git,” [Online]. Available: https://git-scm.com/about [Accessed: May 2024].

[2] MariaDB Foundation, “MariaDB Documentation,” [Online]. Available: MariaDB Server Documentation - MariaDB Knowledge Base [Accessed: May 2024].

[3] Apache Friends, “XAMPP Apache + MariaDB + PHP + Perl,” [Online]. Available: XAMPP Installers and Downloads for Apache Friends [Accessed: May 2024].

[4] Mozilla Developer Network (MDN), “HTML Video element,” [Online]. Available: <video>: The Video Embed element - HTML: HyperText Markup Language | MDN [Accessed: May 2024].

[5] Vimeo, “What is Vimeo?” [Online]. Available: About Vimeo [Accessed: May 2024].

[6] YouTube, “YouTube Features Overview,” [Online]. Available: https://www.youtube.com/features [Accessed: May 2024].

[7] Cloudflare, “What is a video CDN?” [Online]. Available: What is a video CDN? | CDN video streaming | Cloudflare [Accessed: May 2024].

[8] Pixabay, “Official website,” [Online]. Available: https://pixabay.com [Accessed: May 2024].

[9] Flaticon, “Official website,” [Online]. Available: https://www.flaticon.com/ [Accessed: May 2024].

[10] Freepik, “Official website,” [Online]. Available: https://www.freepik.com/ [Accessed: May 2024].

[11] GitHub, “Official website,” [Online]. Available: Get started with GitHub documentation - GitHub Docs [Accessed: May 2024].

[12] StarUML, “StarUML documentation,” [Online]. Available: https://docs.staruml.io/ [Accessed: May 2025].

[13] Visual Studio Code, “Documentation for Visual Studio Code,” [Online]. Available: https://code.visualstudio.com/docs [Accessed: May 2025].

[14] Inspector.dev, “Ultimate Guide to PHP File Upload Security,” [Online]. Available: https://inspector.dev/ultimate-guide-to-php-file-upload-security/ [Accessed: June 2025].

[15] Farshim, P., Tessaro, S. (2021). Password Hashing and Preprocessing. In: Canteaut, A., Standaert, FX. (eds) Advances in Cryptology – EUROCRYPT 2021. EUROCRYPT 2021. Lecture Notes in Computer Science(), vol 12697. Springer, Cham.