Nosferatu Cinema
Web Site Documentation

Raul Istrat
Computer Science Departament at
West University of Timisoara
raul.istrat03@e-uvt.ro

1 Introduction

Nosferatu Cinema is a fictional cinema for which I built a website for. The homepage displays movies that will be screened in the coming week. Visitors can check all screening schedules on the `movies` page, and they can also filter by selecting a specific movie. Although creating an account and logging in doesn't provide additional features on the site, an `admin` user gains access to the `dashboard` page, where they can add or remove movies and screenings from the database. The project makes use of data from the TMDB API, which provides movie data such as titles, runtimes, and other information.

2 Project Requirments

Following are the requirments that the website needs to meet. Along with my justification for how I met said requirments in red.

1. HTML
   - The website must contain a minimum of 4 and a maximum of 8 web pages.
     The website contains the following pages: `home`, `movies`, `about`, `sign-in`, `sign-in`, `dashboard`. Making 6 pages in total. Although I'm not sure if the `dashboard` page is considered a single page, since it has 2 separate subpages for screenings and movies.
2. CSS
   - Styling must be done in separate files.
     The following **`.css`** files are defined:
     → style.css — for all pages, defines the base styling and the styling for the header.
     → about.css, dashboard.css, home.css, movies.css — each defines the custom styling for the page with the same name.
     → form.css — shared between the `sign-in` and the `sign-up` pages.
   - The site must be responsive.
     Each page's style was carefuly crafted to fit to all aspect ratios using css media queries, some example of responsiveness:
     → As the width of the browser gets narrower, the header separates into 2 rows.
     → In the `dashboard` page, posters will only be displayed if there is enough room in the window for them.
   - The layout must be created using tables and frames.
     Not so sure about this one. The layout of each page was created using grids, occasionally some flex/inline/flexbox elements can be found. I don't believe I understood this requirment, since the suggested layouts are pretty outdated.
   - It must contain a drop-down menu made with CSS.
     In the `dashboard` page, the admin can select using a dropdown menu the order the data is displayed in.
   - CSS transformations must be used.
     I believe "invert()" and "hue-rotate()" are considered CSS transformations, which are used for the delete buttons on the `dashboard` page.

3. JavaScript/jQuery Elements

   – Modifying the style of an element or a group of elements.
   An onclick event for the dropdown menu was used to call the "toggle_dropdown()" function with JavaScript. Modifying the style of the dropdown content to either show on the page or not.

   – Using functions in forms.
   The forms for adding new entries in the `dashboard` page are calling JavaScript functions that asynchronously send requests to the server such that elements can be added without reloading the page.

   – Using mouse and keyboard events.
   There are multiple onclick events, such as the one mentioned above.

   – Dynamic modification of the position of an element.
   JavaScript functions have been implemented to add or delete rows from tables.

4. PHP+MySQL

   – It must contain at least one registration and login form in PHP.
   There are dedicated `sign-in` and `sign-up` pages. Which send requests via PHP.

   – At least one MySQL table must be used to store and update information.
   There are three MySQL tables defined:
   → screenings — ID(INT), movie_id(INT), screening_date(DATE), screening_time(TIME)
   → films — ID(INT), title(VARCHAR 255), description(TEXT), duration(INT), director(VARCHAR 255), poster_url(VARCHAR 255)
   → users — username(VARCHAR 255), password(VARCHAR 255)

   – A page with dynamic content must be generated using PHP and information taken from a MySQL table, defining the table with data (more than 50 entries) taken from CSV files and imported into MySQL using the PHPMyAdmin interface in XAMPP/ or JSON or other databases alternatives.
   The `home`, `movies`, and `dashboard` pages all use PHP to display information from the `screenings` and `films` tables. The data in the `films` table was defined using a Python script implemented by me and the TMDB API. Another Python script was used for randomly generating a lot of screening data to insert in the database. Although the data was imported directly using queries, a script to generate a CSV would have been as easy to implement, but the process was more automatic this way.

3 Functionalities of the website

In the