Total marks:

_____/60

_____/20%

Marking Rubrics

Group name:	P5_G5	Title: Movie Ticketing System
Names and ID:	Leon Siow Yi Hong	2204403
	Quak Jing	2205378
	Angel Yap Yoon Ying	2205499
	Koh Khai Jeck	2304740

Criteria	Poor	Below Average	Average	Good	Excellent	Marks
Construct static web pages (15 marks) [CO1]	**0-3** Little or no understanding of HTML and CSS; pages lack structure and design; visual elements are poorly aligned and unattractive.	**4-6** Basic understanding of HTML and CSS; pages are mostly structured and visually appealing, but with some issues.	**7-9** Good understanding of HTML and CSS; pages are well-structured and visually appealing, with some minor improvements possible.	**10-12** Excellent understanding of HTML and CSS; pages are highly structured and visually appealing, with a clear understanding of design principles.	**13-15** Exceptional understanding of HTML and CSS; pages are expertly structured and visually stunning, with a mastery of design principles.	

Build dynamic web pages using event-driven client-side scripts (15 marks) [CO2]	**0-3** Little or no understanding of client-side scripting languages; pages are static and lack interactivity.	**4-6** Basic understanding of client-side scripting languages; pages have some interactivity and respond to user input, but with some issues.	**7-9** Good understanding of client-side scripting languages; pages are highly interactive and respond well to user input, with minor improvements possible.	**10-12** Excellent understanding of client-side scripting languages; pages are very interactive and highly responsive to user input, with a clear understanding of event-driven programming techniques.	**13-15** Exceptional understanding of client-side scripting languages; pages are expertly interactive and highly responsive to user input, with a mastery of event-driven programming techniques.	
Develop server-side and database-driven web applications (15 marks) [CO3]	**0-3** Little or no understanding of server-side scripting languages and databases; web applications are non-functional, insecure, and not scalable.	**4-6** Basic understanding of server-side scripting languages and databases; web applications are functional, but with some security and scalability issues.	**7-9** Good understanding of server-side scripting languages and databases; web applications are functional, secure, and scalable, with some minor improvements possible.	**10-12** Excellent understanding of server-side scripting languages and databases; web applications are highly functional, secure, and scalable, with a clear understanding of validation and error-handling mechanisms.	**13-15** Exceptional understanding of server-side scripting languages and databases; web applications are expertly functional, secure, and scalable, with a mastery of validation and error-handling mechanisms.	

Report (10 marks)	**0-2** Poor documentation and presentation, difficult to understand.	**3-4** Basic documentation, lacking clarity or organization.	**5-6** Good documentation and presentation, with some areas needing improvement.	**7-8** Well-documented and clear presentation, easy to follow with minor improvements possible.	**9-10** Exceptionally well-documented and professionally presented, clear and highly informative.	
Peer Evaluation (5 marks)	**1** No contribution.	**2** Contributions are minimal or not relevant to project goals.	**3** Contributions are somewhat relevant but could be improved.	**4** Contributions are generally helpful and relevant to project goals.	**5** Contributions significantly enhance project goals and outcomes.	

Contents

Introduction

Absolute Cinema is a web-based movie ticketing system designed to provide users with a seamless and engaging experience for browsing, purchasing, and managing movie tickets. The system also allows the users to manage their profile and tickets.

The key feature of the website includes browsing current and upcoming movies, viewing movie details, wish listing movies, and purchase tickets. The system also allows users to manage their Wishlist, view purchased tickets and access their booking details.

The system is developed using PHP, JavaScript, and MySQL, ensuring robust backend functionality and dynamic front end interactions. The purpose of Absolute Cinema is to deliver a user-friendly and visually stunning interface for users from browsing the movie listings to purchasing the tickets.

Website Structure

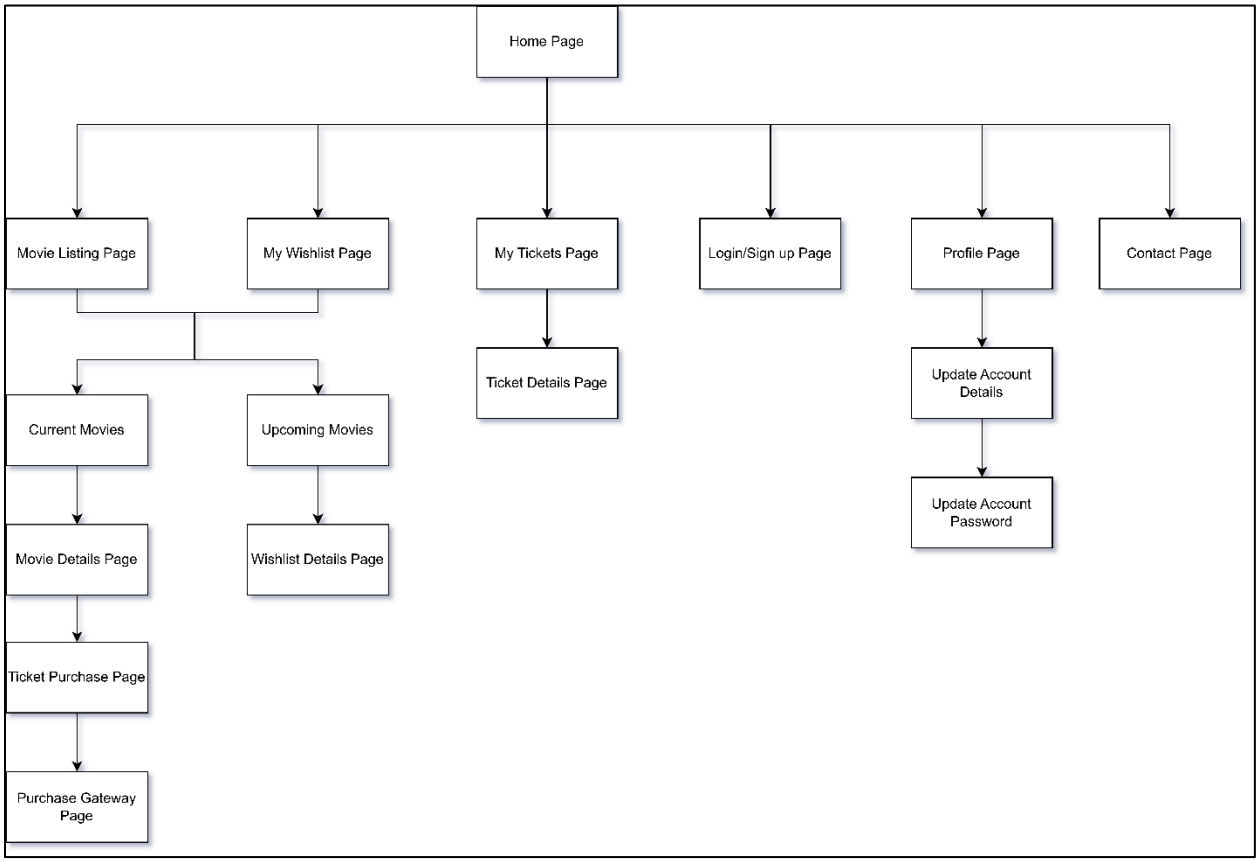


Figure 1. Website Structure

Key Code Snippets with Explanation

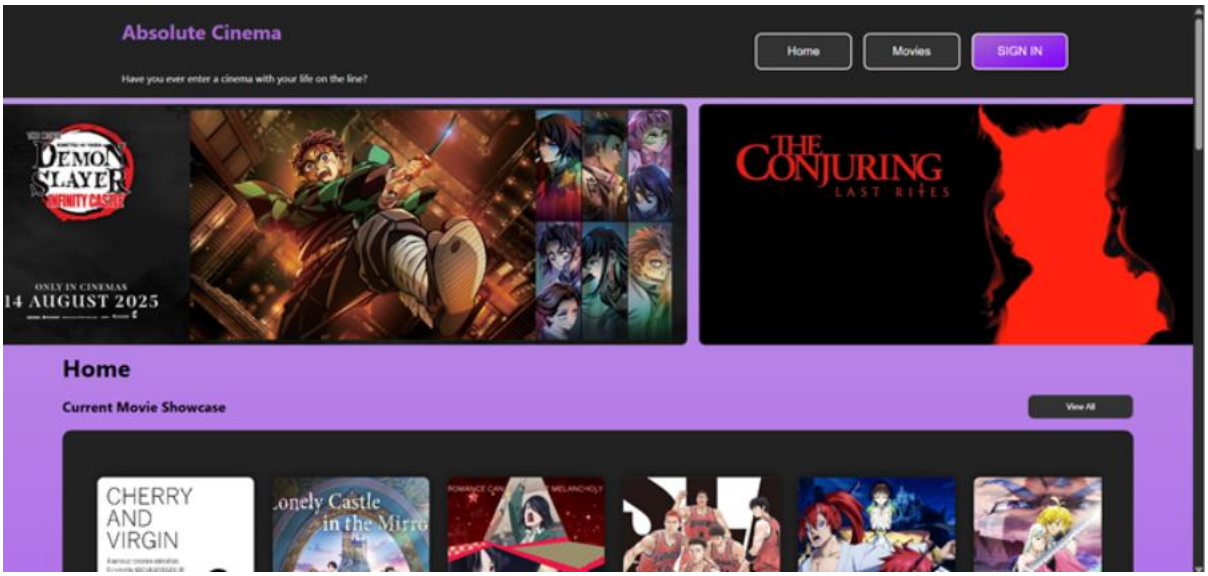
Db.sql

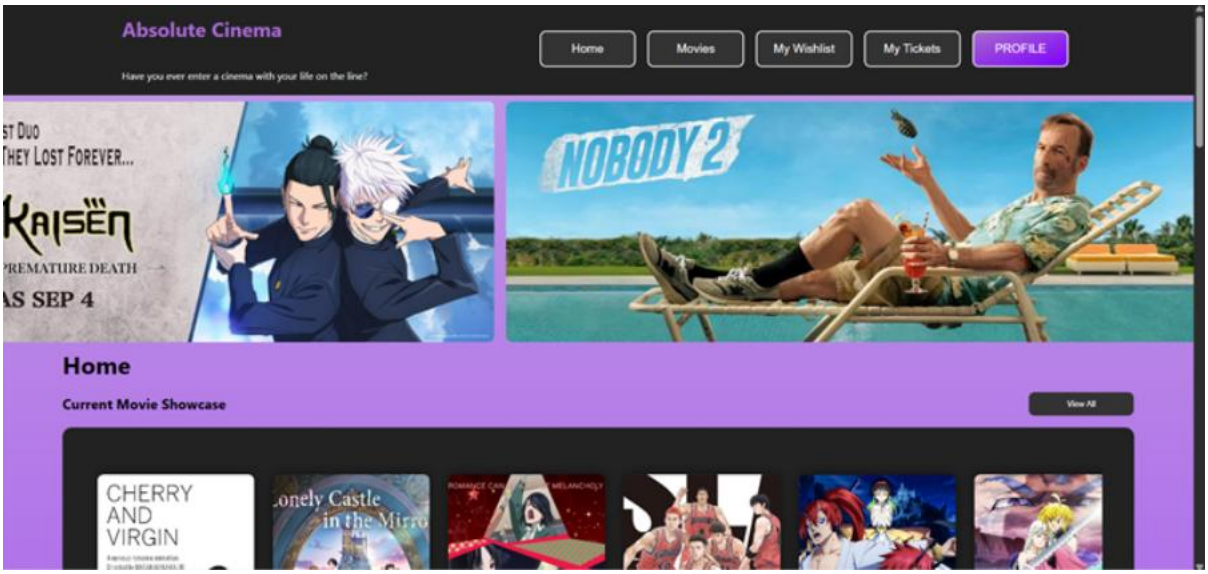
```sql
-- Creating the database
DROP DATABASE IF EXISTS movie_ticketing;
CREATE DATABASE IF NOT EXISTS movie_ticketing CHARACTER SET UTF8MB4;

USE movie_ticketing;

-- Creating the user table
DROP TABLE IF EXISTS user;
CREATE TABLE IF NOT EXISTS user (
  userID INT AUTO_INCREMENT NOT NULL,
  userName VARCHAR(50) NOT NULL UNIQUE,
  userPassword VARCHAR(50) NOT NULL,
  userEmail VARCHAR(50) NOT NULL,
  userPhoneNo VARCHAR(50) NOT NULL,
  PRIMARY KEY (userID)
);

-- Creating the whishlist table
DROP TABLE IF EXISTS upcomingWishlist;
CREATE TABLE IF NOT EXISTS upcomingWishlist (
    userID INT NOT NULL,
    movieID VARCHAR(50) NOT NULL,
    PRIMARY KEY (userID, movieID),
    FOREIGN KEY (userID) REFERENCES user(userID)
);

DROP TABLE IF EXISTS currentWishlist;
CREATE TABLE IF NOT EXISTS currentWishlist (
    userID INT NOT NULL,
    movieID VARCHAR(50) NOT NULL,
    PRIMARY KEY (userID, movieID),
    FOREIGN KEY (userID) REFERENCES user(userID)
);

-- Creating the Transaction Table
DROP TABLE IF EXISTS transactions;
CREATE TABLE IF NOT EXISTS transactions (
    transactionID INT AUTO_INCREMENT NOT NULL,
    userID INT NOT NULL,
    movieID VARCHAR(50) NOT NULL,
    totalPrice FLOAT(10) NOT NULL,
    PRIMARY KEY (transactionID),
    FOREIGN KEY (userID) REFERENCES user(userID)
);
```

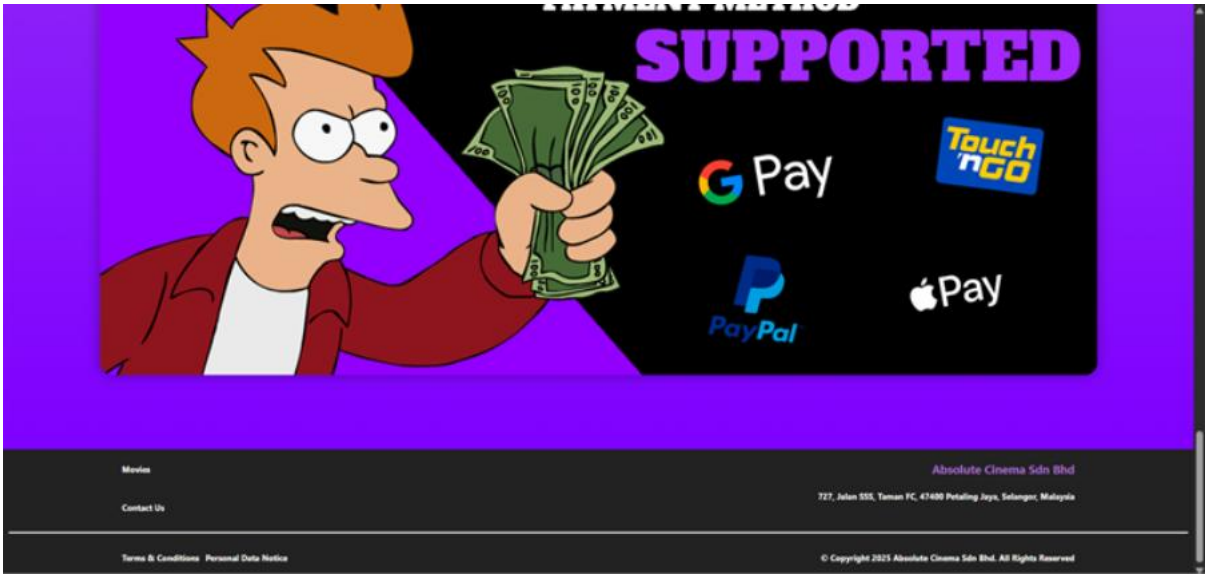
```sql
-- Creating the Ticket Table
DROP TABLE IF EXISTS ticket;
CREATE TABLE IF NOT EXISTS ticket (
    ticketID INT AUTO_INCREMENT NOT NULL,
    userID INT NOT NULL,
    transactionID INT NOT NULL,
    PRIMARY KEY (ticketID),
    FOREIGN KEY (userID) REFERENCES user(userID),
    FOREIGN KEY (transactionID) REFERENCES transaction(transactionID)
);

-- Creating the Contact Table
DROP TABLE IF EXISTS userMessage;
CREATE TABLE IF NOT EXISTS userMessage (
    name VARCHAR(50) NOT NULL,
    email VARCHAR(50) NOT NULL,
    message VARCHAR(300) NOT NULL
);
```

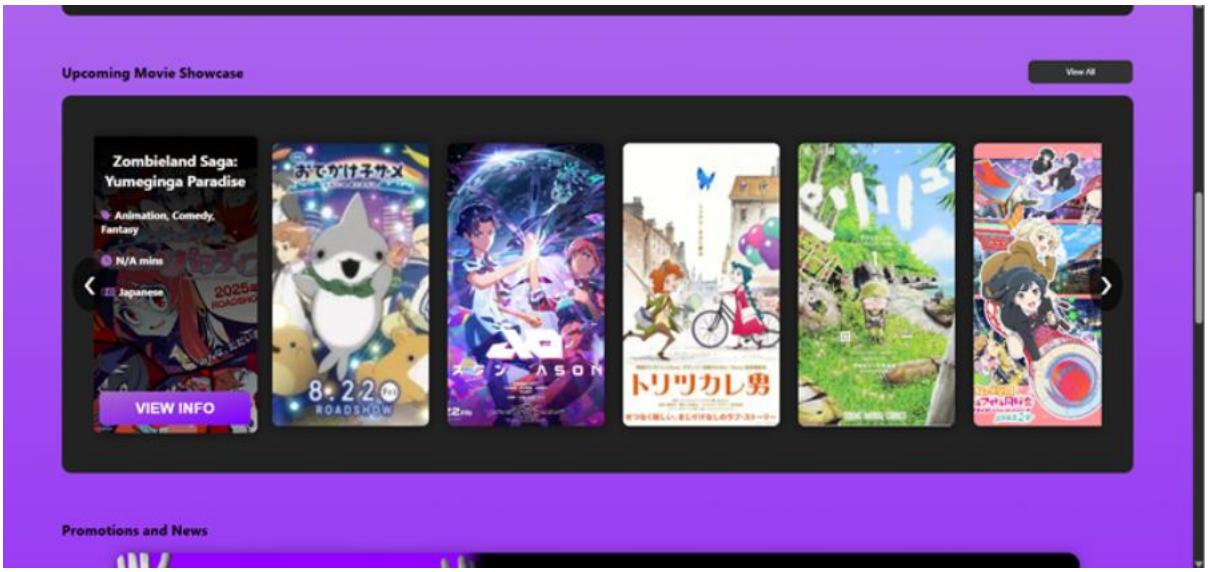
- Db.sql contains SQL queries to create all the necessary tables in mySQL. The tables contain information that is crucial for the website to function.

Home Page


Absolute Cinema
Have you ever enter a cinema with your life on the line?
Home
Movies
SIGN IN
DEMON SLAYER
INFINITY CASTLE
ONLY IN CINEMAS
14 AUGUST 2025
THE CONJURING
LAST RITES
Home
Current Movie Showcase
View All
CHERRY AND VIRGIN


Absolute Cinema
Have you ever enter a cinema with your life on the line?
Home
Movies
My Wishlist
My Tickets
PROFILE
NOBODY 2
Home
Current Movie Showcase
View All
CHERRY AND VIRGIN


SUPPORTED
G Pay
Touch 'n GO
PayPal
Pay
Movies
Contact Us
Absolute Cinema Sdn Bhd
727, Jalan SS5, Taman FC, 47400 Petaling Jaya, Selangor, Malaysia
Terms & Conditions Personal Data Notice
© Copyright 2025 Absolute Cinema Sdn Bhd. All Rights Reserved


Upcoming Movie Showcase
View All
Zombieland Saga: Yumeginga Paradise
Animation, Comedy, Fantasy
N/A mins
Japanese
VIEW INFO
8.22
ROADSHOW
トリツカレ男
Promotions and News


Promotions and News
ABSOLUTE ALL TIME LOW
ONLY
RM 15.00
before tax (10%)
PER TICKET
ORIGINAL PRICE
OF RM 20.00

Index.php

```
<title>Absolute Cinema</title>
<link rel="stylesheet" href="styles.css"/>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css">
<style>
    .promo {
        align-self: center;
        margin-bottom: 2%;
        width: 90%;
        height: auto;
        border-radius: 20px;
        box-shadow: 0 0 1em rgba(0, 0, 0, 0.35);
        transition: transform 0.4s ease;
    }

    .promo:hover {
        cursor: pointer;
        transform: scale(1.05);
    }

    .detailButton {
        font-size: small;
        width: 13em;
    }
```

```
<main>
    <div class="home-movie-row-wrapper">
            <div class="home-movie-row-wrapper" id="banners"></div>
    </div>

    <div class="container">
        <h1 style="margin: 0;">Home</h1>
        <div class="homeInfo">
            <h3>Current Movie Showcase</h3>
            <button class="detailButton" onclick="window.location.href='movie/'">View All</button>
        </div>
        <div class="home-movie-row-wrapper" style="border-radius: 15px; background-color: #222222;">
            <button class="arrow left" onclick=scrollCurrentMovies(-1)>&#10094;</button>
            <div class="home-movie-row-wrapper" style="padding: 3%; width:94%" id="currentMovieContainer"></div>
            <!-- JS will populate movie cards here -->
            <button class="arrow right" onclick=scrollCurrentMovies(1)>&#10095;</button>
        </div>
        <br>
        <br>

        <div class="homeInfo">
            <h3>Upcoming Movie Showcase</h3>
            <button class="detailButton" onclick="window.location.href='movie/?state=upcoming'">View All</button>
        </div>
        <div class="home-movie-row-wrapper" style="border-radius: 15px; background-color: #222222">
            <button class="arrow left" onclick=scrollUpcomingMovies(-1)>&#10094;</button>
            <div class="home-movie-row-wrapper" style="padding: 3%; width:94%" id="upcomingMovieContainer"></div>
            <!-- JS will populate movie cards here -->
            <button class="arrow right" onclick=scrollUpcomingMovies(1)>&#10095;</button>
        </div>
        <br>
        <br>

        <h3>Promotions and News</h3>
        <img class="promo" onclick="window.location.href='info/promotion1'" src="images/promotions/promotion1" alt="ticket price">
```

```
        <img class="promo" onclick="window.location.href='info/promotion2'" src="images/promotions/promotion2" alt="payment method">
    </div>
</main>

<?php
    include("include/footer.php");
?>
<script src="script.js"></script>
<script>
    window.onload = async () => {
        displayBanner();
        await displayCurrentMovie();
        await displayUpcomingMovie();
    }
```

- Serves as the **homepage** for the "Absolute Cinema" website, showcasing movies and promotions.
- A container (#banners) at the top for displaying featured content, populated by the displayBanner() JavaScript function.
- Current Movies
  - Displays a row of movies labeled "Current Movie Showcase".
  - Includes left/right arrow buttons to scroll horizontally through the movie cards.
  - The displayCurrentMovie() function populates the #currentMovieContainer.
  - A "View All" button links to movie/ to see all current movies.
- Upcoming Movies
  - Displays a row of movies labeled "Upcoming Movie Showcase".
  - Also has arrow buttons for horizontal scrolling, controlled by scrollUpcomingMovies().
  - The displayUpcomingMovie() function populates the #upcomingMovieContainer.
  - A "View All" button links to movie/?state=upcoming to filter for upcoming movies.
- Promotion Section
  - Displays clickable promotional images (promo1, promo2).
  - Each image has a hover effect (scale up) and redirects to a specific info page (e.g., info/promotion1) when clicked.
- calls three functions to populate the page: displayBanner(), displayCurrentMovie(), displayUpcomingMovie()

script.js

```
//For HomePage Move Shotimes fetch dt
// fetch movie detail json using IMDB ID
async function fetchMovieDetail(imdbId) {
    try {
        const url = `https://api.imdbapi.dev/titles/${imdbId}`;
        const response = await fetch(url);
        const data = await response.json();
        console.log(data);

        return data;
    } catch (error) {
        console.error("Error fetching movie:", error);
        return null;
    }
}

// fetch Array of Movie json from IMDB ID
async function fetchMovieArray(url) {
  try {
        const response = await fetch(url);
        const data = await response.json();
        console.log(data);
        return data.titles;

    }
    catch (error) {
        console.error("Error fetching movie:", error);
        return null;
    }
}
```

- Banner Display:
  - displayBanner(): Populates the banner section by creating a scrolling row of images.
  - Uses a predefined array bannerFiles to display each banner image twice in a loop.
  - Applies a fixed max-height and object-fit style to the images.

```
async function displayCurrentMovie() {
    const container = document.getElementById("currentMovieContainer");
    container.innerHTML = '';

    let counter = 0;
    let rowDiv = document.createElement("div");
    rowDiv.className = "movie-card-row";
    rowDiv.id = "current-movie-card-row";
    rowDiv.classList.add("hidden");

    let url = `https://api.imdbapi.dev/titles?types=MOVIE&genres=Animation&languageCodes=ja&endYear=`
      + (new Date().getFullYear() - 3) + `&sortBy=SORT_BY_RELEASE_DATE&sortOrder=DESC`;
    const NEWEST_SHOWS = await fetchMovieArray(url);

    for (const movieData of NEWEST_SHOWS) {
        if (movieData.primaryImage === undefined || !movieData) {
          continue;
        }
        const movie = await fetchMovieDetail(movieData.id);

        const movieDiv = document.createElement("div");
        movieDiv.className = "movie-card";
        movieDiv.style = "margin: 0 5px";
        movieDiv.innerHTML = `<img src="${movie.primaryImage.url}" alt="${movie.originalTitle}">`;
        const movieOverlay = document.createElement("div");
        movieOverlay.className = "movie-overlay";

        movieOverlay.innerHTML = `
        <div style="height: 365px">
          <h3>${movie.primaryTitle}</h3>
          <h5><i class="fa-solid fa-tag" style="color: #A76BCE; padding-right: 1%"></i> ${movie.genres?.slice(0,3).join(', ') || 'N/A'}</h5>
          <h5><i class="fa-solid fa-clock" style="color: #A76BCE; padding-right: 1%"></i> ${movie.runtimeSeconds ? movie.runtimeSeconds / 60 : 'N/A'} mins</h5>
          <h5><i class="fa-solid fa-language" style="color: #A76BCE; padding-right: 1%"></i> ${movie.spokenLanguages && movie.spokenLanguages.length > 0 ? movie.spokenL
        </div>
        <div style="display: flex; justify-content: center;">
          <a href="movie/movie-detail.php?movieID=${movie.id}"><button>BUY TICKET</button><a>
```

```
<div style="display: flex; justify-content: center;">
  <a href="movie/movie-detail.php?movieID=${movie.id}"><button>BUY TICKET</button><a>
</div>
`;
movieDiv.appendChild(movieOverlay);

rowDiv.appendChild(movieDiv);

counter++;

if (counter === 20) {
  container.appendChild(rowDiv);
  rowDiv.classList.add("fade-in");

  rowDiv.classList.remove("hidden");
  rowDiv.classList.add("fade-in");
  break;
}
```

- Current Movies Section:
  - displayCurrentMovie(): Fetches and displays "Current" movies.
  - Constructs a specific API URL to fetch movies sorted by release date
  - For each movie, creates a movie card with its image and an overlay containing: Title, top 3 genres, runtime
  - A "BUY TICKET" button linking to a detail page (movie-detail.php) with the movie's ID.
  - Limits display to the first 20 valid movies (with images).
  - Applies a fade-in animation after populating the row.

```
async function displayUpcomingMovie() {
    const container = document.getElementById("upcomingMovieContainer");
    container.innerHTML = '';

    let counter = 0;
    let rowDiv = document.createElement("div");
    rowDiv.className = "movie-card-row";
    rowDiv.id = "upcoming-movie-card-row";
    rowDiv.classList.add("hidden");

    let url = `https://api.imdbapi.dev/titles?types=MOVIE&genres=Animation&languageCodes=ja&endYear=`
      + (new Date().getFullYear()) + `&sortBy=SORT_BY_RELEASE_DATE&sortOrder=DESC`;
    const NEWEST_SHOWS = await fetchMovieArray(url);

    for (const movieData of NEWEST_SHOWS) {
        if (movieData.primaryImage === undefined || !movieData) {
          continue;
        }
        const movie = await fetchMovieDetail(movieData.id);

        const movieDiv = document.createElement("div");
        movieDiv.className = "movie-card";
        movieDiv.style = "margin: 0 5px";
        movieDiv.innerHTML = `<img src="${movie.primaryImage.url}" alt="${movie.originalTitle}">`;
        const movieOverlay = document.createElement("div");
        movieOverlay.className = "movie-overlay";

        movieOverlay.innerHTML = `
        <div style="height: 365px">
          <h3>${movie.primaryTitle}</h3>
          <h5><i class="fa-solid fa-tag" style="color: #A76BCE; padding-right: 1%"></i> ${movie.genres?.slice(0,3).join(', ') || 'N/A'}</h5>
          <h5><i class="fa-solid fa-clock" style="color: #A76BCE; padding-right: 1%"></i> ${movie.runtimeSeconds ? movie.runtimeSeconds / 60 : 'N/A'} mins</h5>
          <h5><i class="fa-solid fa-language" style="color: #A76BCE; padding-right: 1%"></i> ${movie.spokenLanguages && movie.spokenLanguages.length > 0 ? movie.spokenL
        </div>
        <div style="display: flex; justify-content: center;">
```

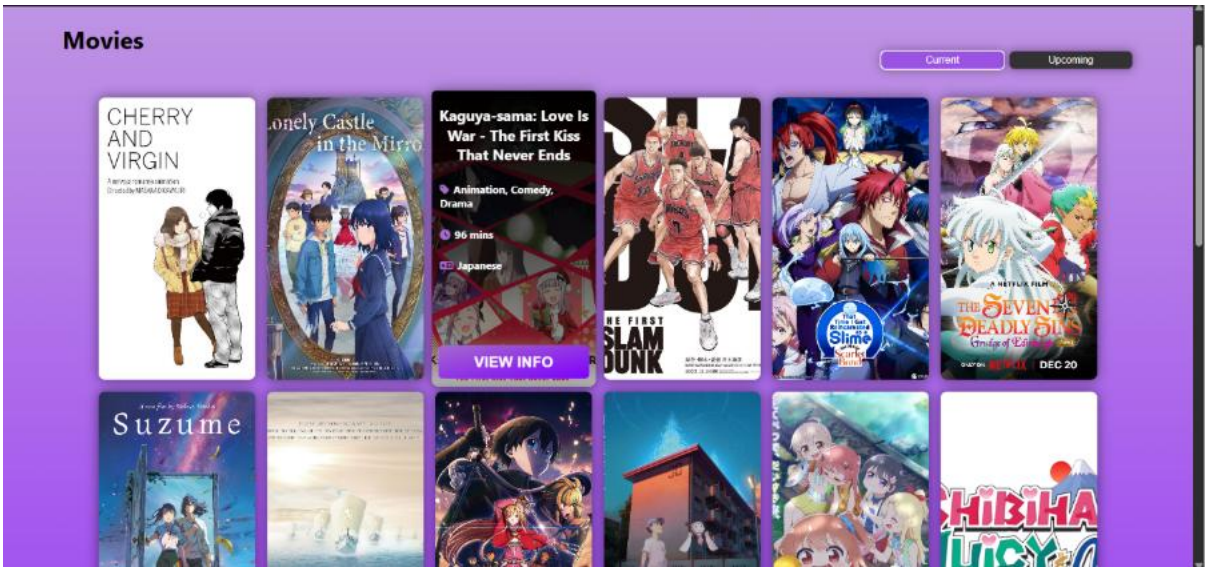
```
          <a href="myWishlist/wish-detail.php?movieID=${movie.id}"><button>VIEW INFO</button><a>
        </div>
        `;
        movieDiv.appendChild(movieOverlay);

        rowDiv.appendChild(movieDiv);

        counter++;

        if (counter === 20) {
          container.appendChild(rowDiv);
          rowDiv.classList.add("fade-in");

          rowDiv.classList.remove("hidden");
          rowDiv.classList.add("fade-in");
          break;
        }
```

- displayUpcomingMovie(): Fetches and displays "Upcoming" movies.
- Constructs an API URL to fetch movies from the current year.
- Creates movie cards similar to the current section, but with a "VIEW INFO" button linking to a wishlist detail page (wish-detail.php).

```
function scrollCurrentMovies(direction) {
    const row = document.querySelector("#current-movie-card-row");
    if (!row) return;
    row.scrollBy({ left: 300 * direction * 2, behavior: "smooth" });
}

function scrollUpcomingMovies(direction) {
    const row = document.querySelector("#upcoming-movie-card-row");
    if (!row) return;
    row.scrollBy({ left: 300 * direction * 2, behavior: "smooth" });
}
```

- scrollCurrentMovies(direction): Smoothly scrolls the current movies row horizontally by a calculated amount (600px * direction).
- scrollUpcomingMovies(direction): Smoothly scrolls the upcoming movies row horizontally.

Movies Page



- The movies page shows the list of movies to user. The current and upcoming button will allow user to toggle between current available movie and upcoming movie list. When hover over a movie it will show a button to direct user to movies details

Index.php

```
<!--Body-->
<main>
    <div class="container">
        <div class="containerHeader">
            <h1>Movies</h1>
            <div style="padding-top: 4%; padding-bottom: 2%;">
                <button id="currentBtn" class="selectButton movie-filter-button active" onclick="handleButtonClick(displayCurrentMovie)">Current</button>
                <button id="upcomingBtn" class="selectButton movie-filter-button" onclick="handleButtonClick(displayUpcomingMovie)">Upcoming</button>
            </div>
        </div>
        <div id="movieContainer"></div>
    </div>
</main>

<!--Footer-->
<?php
    include("../include/footer.php");
?>
<script src="script.js"></script>
<script>
    window.onload = () => {
        const params = new URLSearchParams(window.location.search);
        const state = params.get("state");

        if(state) {
            handleButtonClick(displayUpcomingMovie);
        } else {
            handleButtonClick(displayCurrentMovie);
        }

        if (window.location.search) {
            history.replaceState(null, '', window.location.pathname);
        }
    }
</script>
```

```
<script src="script.js"></script>
<script>
    window.onload = () => {
        const params = new URLSearchParams(window.location.search);
        const state = params.get("state");

        if(state) {
            handleButtonClick(displayUpcomingMovie);
        } else {
            handleButtonClick(displayCurrentMovie);
        }

        if (window.location.search) {
            history.replaceState(null, '', window.location.pathname);
        }
    }
</script>
```

- When the page load it will get the state value, to decide whether to display upcoming or current movie. Using an if else statement

```
// fetch movie detail json using IMDB ID
async function fetchMovieDetail(imdbId) {
    try {
        const url = `https://api.imdbapi.dev/titles/${imdbId}`;
        const response = await fetch(url);
        const data = await response.json();
        console.log(data);

        return data;
    } catch (error) {
        console.error("Error fetching movie:", error);
        return null;
    }
}

// fetch movie preview video json using IMDB ID
async function fetchMoviePreview(imdbId) {
     try {
        const url = `https://api.imdbapi.dev/titles/${imdbId}/videos`;
        const response = await fetch(url);
        const data = await response.json();
        console.log(data);

        return data;
    } catch (error) {
        console.error("Error fetching movie:", error);
        return null;
    }
}

// fetch Array of Movie json from IMDB ID
async function fetchMovieArray(url) {
  try {
        const response = await fetch(url);
        const data = await response.json();
        console.log(data);
        return data.titles;

    }
    catch (error) {
        console.error("Error fetching movie:", error);
        return null;
    }
}
```

- The fetchMovieDetail function fetch a specific movie data using the IMDb API by passing the imdbId
- The fetchMoviePreview function fetches the movie's video preview using the IMDb API.
- The fetchMovieArray function gets an array of movies from the IMDb API.

```
// display the movies selection
async function displayCurrentMovie() {
    const container = document.getElementById("movieContainer");
    container.innerHTML = '';

    let url = `https://api.imdbapi.dev/titles?types=MOVIE&genres=Animation&languageCodes=ja&endYear=`
      + (new Date().getFullYear() - 3) + `&sortBy=SORT_BY_RELEASE_DATE&sortOrder=DESC`;
    const NEWEST_SHOWS = await fetchMovieArray(url);

    for (const movieData of NEWEST_SHOWS) {
        if (movieData.primaryImage === undefined || !movieData) {
          continue;
        }
        const movie = await fetchMovieDetail(movieData.id);

        const movieDiv = document.createElement("div");
        movieDiv.className = "movie-card";
        movieDiv.innerHTML = `<img src="${movie.primaryImage.url}" alt="${movie.originalTitle}">`;
        const movieOverlay = document.createElement("div");
        movieOverlay.className = "movie-overlay";

        movieOverlay.innerHTML = `
        <div style="height: 365px">
          <h3>${movie.primaryTitle}</h3>
          <h5><i class="fa-solid fa-tag" style="color: #A76BCE; padding-right: 1%"></i> ${movie.genres?.slice(0,3).join(', ') || 'N/A'}</h5>
          <h5><i class="fa-solid fa-clock" style="color: #A76BCE; padding-right: 1%"></i> ${movie.runtimeSeconds ? movie.runtimeSeconds / 60 : 'N/A'} mins</h5>
          <h5><i class="fa-solid fa-language" style="color: #A76BCE; padding-right: 1%"></i>
          ${movie.spokenLanguages && movie.spokenLanguages.length > 0 ? movie.spokenLanguages[0].name : 'N/A'}</h5>
        </div>
        <div style="display: flex; justify-content: center;">
          <a href="movie-detail.php?movieID=${movie.id}"><button>VIEW INFO</button><a>
        </div>
        `;
        movieDiv.appendChild(movieOverlay);
        container.appendChild(movieDiv);

        movieDiv.classList.add("fade-in");
    }
}
```

```
// display the movies selection
async function displayUpcomingMovie() {
    const container = document.getElementById("movieContainer");
    container.innerHTML = '';

    // Fetch Series
    const url = `https://api.imdbapi.dev/titles?types=MOVIE&genres=Animation&languageCodes=ja&endYear=`+
      new Date().getFullYear() + `&sortBy=SORT_BY_RELEASE_DATE&sortOrder=DESC`;
    const UPCOMING_SERIES = await fetchMovieArray(url);

    for (const movieData of UPCOMING_SERIES) {
        if (movieData.primaryImage === undefined || !movieData) {
          continue;
        }
        const movie = await fetchMovieDetail(movieData.id);

        const movieDiv = document.createElement("div");
        movieDiv.className = "movie-card";
        movieDiv.innerHTML = `<img src="${movie.primaryImage.url}" alt="${movie.originalTitle}">`;
        const movieOverlay = document.createElement("div");
        movieOverlay.className = "movie-overlay";

        movieOverlay.innerHTML = `
        <div style="height: 365px">
          <h3>${movie.primaryTitle}</h3>
          <h5><i class="fa-solid fa-tag" style="color: #A76BCE; padding-right: 1%"></i> ${movie.genres?.slice(0,3).join(', ') || 'N/A'}</h5>
          <h5><i class="fa-solid fa-clock" style="color: #A76BCE; padding-right: 1%"></i> ${movie.runtimeSeconds ? movie.runtimeSeconds / 60 : 'N/A'} mins</h5>
          <h5><i class="fa-solid fa-language" style="color: #A76BCE; padding-right: 1%"></i>
          ${movie.spokenLanguages && movie.spokenLanguages.length > 0 ? movie.spokenLanguages[0].name : 'N/A'}</h5>
        </div>
        <div style="display: flex; justify-content: center;">
          <a href="/Movie-Ticketing-System/myWishlist/wish-detail.php?movieID=${movie.id}"><button>VIEW INFO</button><a>
        </div>
        `;
        movieDiv.appendChild(movieOverlay);
        container.appendChild(movieDiv);

        movieDiv.classList.add("fade-in");
    }
}
```

- The displayCurrentMovie and displayUpcomingMovie function similarly where it fetch movie from the imdb API.

```
async function handleButtonClick(asyncMethod) {
    if (asyncMethod == displayUpcomingMovie) {
      upcomingBtn.classList.add("active");
      currentBtn.classList.remove("active");
    } else {
      currentBtn.classList.add("active");
      upcomingBtn.classList.remove("active");
    }

    // Disable both buttons
    const buttons = document.querySelectorAll('.selectButton');
    buttons.forEach(b => b.disabled = true);

    try {
        await asyncMethod();
    } finally {
        buttons.forEach(b => b.disabled = false);
    }
}
```

- This function manages the state of the buttons so that there it disables both buttons to avoid multiple clicks and run the displayUpcomingMovie or displayCurrentMovie functions. The buttons are re-enabled once function is finish running.

Movie Details and Wishlist Details Page

- This page shows the details of selected movie and allows users to add movie to wish list and buy ticket for the movie. Users must be logged in to add movie to wishlist or buy ticket. If users have not logged in the button to add movie to wishlist and buy ticket will redirect users to log in page

index.php

```
<script src="script.js"></script>
<script>
    window.onload = () => {
        const params = new URLSearchParams(window.location.search);
        const imdbId = params.get("movieID");
        displayPreview(imdbId);
        displayDetails(imdbId);
    }
</script>
```

- When the page load, it will retrieve movieID as imbID and display the preview and details of the movie with the specific movieID

```
async function displayPreview(imdbId) {
  const container = document.getElementById("vid");

  const video = await fetchMoviePreview(imdbId);

  if (!video || !video.videos || video.videos.length === 0) {
    container.innerHTML = "Video Preview Not Available";
  } else {
    container.innerHTML = `
      <iframe
        id="moviePreview"
        src="https://www.imdb.com/video/imdb/${video.videos[0].id}/imdb/embed"
        frameborder="0"
        allowfullscreen>
      </iframe>
    `;
  }
}
```

```
// display movie detail
async function displayDetails(imdbId) {
  const containerLeft = document.getElementById("movieInfo");
  const containerTitle = document.getElementById("title");

  const movie = await fetchMovieDetail(imdbId);

  containerTitle.innerHTML = `
    <h2>${movie.primaryTitle}</h2>
  `;

  containerLeft.innerHTML = `
    <img src="${movie.primaryImage.url}" alt="${movie.originalTitle}">
    <div id="movieDetail">
      <h5><i class="fa-solid fa-tag"></i> ${movie.genres?.join(', ') || 'N/A'}</h5>
      <h5><i class="fa-solid fa-clock"></i> ${movie.runtimeSeconds ? Math.floor(movie.runtimeSeconds / 60) : 'N/A'} mins</h5>
      <h5><i class="fa-solid fa-language"></i> ${movie.spokenLanguages?.[0]?.name || 'N/A'}</h5>
    </div>
    <button id="wishButton" class="detailButton"></button>
    <button id="buyButton" class="detailButton"></button>
  `;
```

- Both of these functions display the movies information, displayPreview display the preview video of the movie using iframe and display Details display the detaiils of the movie.

```
async function isWishlisted(imdbId) {
  const userID = sessionStorage.getItem("loggedUserID");

  const response = await fetch(`isWishlisted.php?userID=${userID}&movieID=${imdbId}`);
  const data = await response.json();

  return data.length > 0;
}

async function addWishlist(imdbId) {
  const userID = sessionStorage.getItem("loggedUserID");
  const response = await fetch('addWishlist.php', {
    method: 'POST',
    headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
    body: `userID=${encodeURIComponent(userID)}&movieID=${encodeURIComponent(imdbId)}`
  });
  const result = await response.json();
  return result.success;
}

async function removeWishlist(imdbId) {
  const userID = sessionStorage.getItem("loggedUserID");
  const response = await fetch('removeWishlist.php', {
    method: 'POST',
    headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
    body: `userID=${encodeURIComponent(userID)}&movieID=${encodeURIComponent(imdbId)}`
  });
  const result = await response.json();
  return result.success;
}
```

- These codes use to manage the wishlist of user. IsWishlisted checks if user's wishlist have the specific movie, addWishlist adds movie to user's wishlist by sending a post request to addWishlist.php, removeWishlist removes the specific movie from the user's wishlist.

```php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "movie_ticketing";
$userID = $_POST['userID'];
$movieID = $_POST['movieID'];

$conn = mysqli_connect($servername, $username, $password, $dbname);

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

- Connects to the MySQL database. Using mysqli_connect to establish connection and if connection fails an error message is displayed.

```php
$sql = "INSERT INTO currentWishlist (userID, movieID) VALUES (?, ?)";
$stmt = mysqli_prepare($conn, $sql);
mysqli_stmt_bind_param($stmt, "is", $userID, $movieID);

if (mysqli_stmt_execute($stmt)) {
    echo json_encode(['success' => true]);
} else {
    echo json_encode(['success' => false, 'error' => mysqli_stmt_error($stmt)]);
}

mysqli_stmt_close($stmt);
mysqli_close($conn);
```

- Insert Data into currentWishlist. If the insertion success, a success message is returned, otherwise, an error message is returned.

```php
$sql = "SELECT * FROM currentWishlist WHERE userID = $userID AND movieID = '$movieID'";
$result = mysqli_query($conn, $sql);

$wishlist = [];
while($row = mysqli_fetch_assoc($result)) {
    $wishlist[] = $row;
}
echo json_encode($wishlist);

mysqli_close($conn);
```

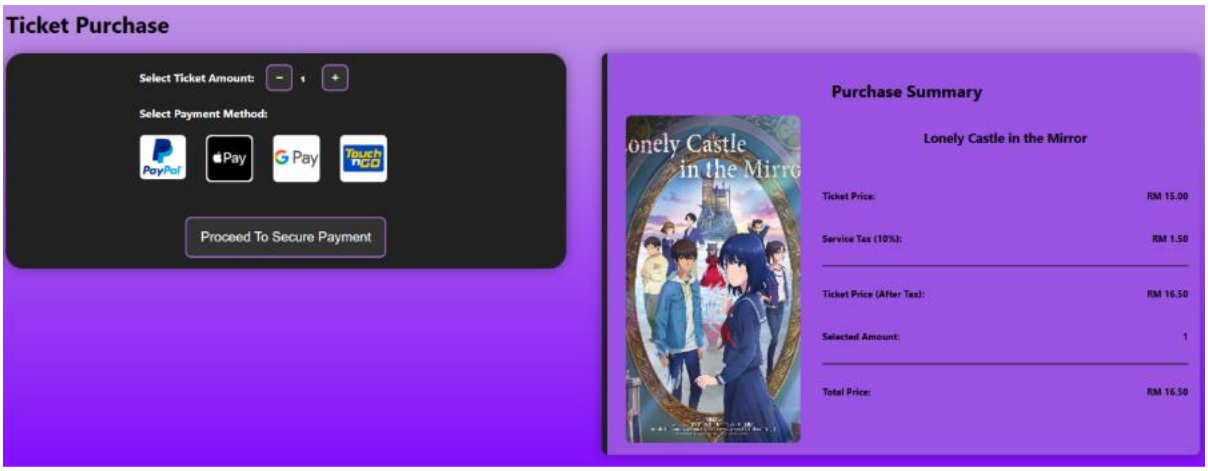
- Checks if specific data already exists inside the user wishlist

```php
$sql = "DELETE FROM currentWishlist WHERE userID=? AND movieID=?";
$stmt = mysqli_prepare($conn, $sql);
mysqli_stmt_bind_param($stmt, "is", $userID, $movieID);

if (mysqli_stmt_execute($stmt)) {
    echo json_encode(['success' => true]);
} else {
    echo json_encode(['success' => false, 'error' => mysqli_stmt_error($stmt)]);
}

mysqli_stmt_close($stmt);
mysqli_close($conn);
```

- Remove a specific movie from user wishlist

Purchase Page

This page is the checkout page for user to buy ticket. User can choose payment method and proceed to pay.



(purchase.php)

```php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "movie_ticketing";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $userID = htmlspecialchars($_POST["userID"]);
    $totalPrice = htmlspecialchars($_POST["ticketAmount"] * 16.5);
    $movieID = htmlspecialchars($_POST["movieID"]);

    makeTransaction($conn, $userID, $totalPrice, $movieID);
}

function makeTransaction($conn, $userID , $totalPrice, $movieID) {
    $stmt = mysqli_prepare($conn, 'INSERT INTO transactions (userID, movieID, totalPrice) VALUES (?, ?, ?)');
    mysqli_stmt_bind_param($stmt, 'isd', $userID, $movieID, $totalPrice);

    if (mysqli_stmt_execute($stmt)) {
        $transactionID = mysqli_insert_id($conn);

        $ticketAmount = $_POST["ticketAmount"];

        $ticketStmt = mysqli_prepare($conn, 'INSERT INTO ticket (userID, transactionID) VALUES (?, ?)');
        for ($i = 0; $i < $ticketAmount; $i++) {
            mysqli_stmt_bind_param($ticketStmt, 'ii', $userID, $transactionID);
            mysqli_stmt_execute($ticketStmt);
        }
        mysqli_stmt_close($ticketStmt);

        header("Location: gateway.php");
        exit;
    } else {
        echo "Error: " . mysqli_stmt_error($stmt);
    }

    mysqli_stmt_close($stmt);
}
```

- Connect into movie_ticketing database.
- The code checks if the request method is POST, and calls makeTransaction() function to process the transaction using userID, movieID and totalPrice. Then the SQL generate a TransactionID
- Using the transactionID, the script proceeds to insert tickets into the ticket table. It inserts the specified number of tickets and associates them with the user and the transaction.
- Once the transaction and tickets are successfully recorded, the user is redirected to a payment gateway, gateway.php to proceed with the payment.

(index.php) for purchase page

```
    <h1>Ticket Purchase</h1>
</div>
<div class="purchaseContainer">
    <div id="purchaseInfo">
        <div id="paymentMethod" style="display: flex; min-width: 25vw;">
            <form method="POST" action="purchase.php" onsubmit="return validatePaymentOption()" style="padding: 0 2em">
                <input type="hidden" id="userID" name="userID">
                <input type="hidden" id="movieID" name="movieID">
                <div>
                    <label for="ticketAmount" style="font-weight: bolder; padding-right: 3%;">Select Ticket Amount:</label>
                    <button type="button" class="selectButton" onclick="changeAmount(-1)">-</button>
                    <input type="number" id="ticketAmount" name="ticketAmount" value="1" min="1" readonly>
                    <button type="button" class="selectButton" onclick="changeAmount(1)">+</button>
                </div>
                <div style="padding-top: 5%">
                    <label for="gateway" style="font-weight: bolder; ">Select Payment Method:</label>
                    <div class="payment-options">
                        <label class="payment-option">
                            <input type="radio" name="gateway" value="paypal">
                            <img src="https://upload.wikimedia.org/wikipedia/commons/9/93/PayPal_Logo2014.png" alt="PayPal">
                        </label>

                        <label class="payment-option">
                            <input type="radio" name="gateway" value="applepay">
                            <img src="https://cdn-icons-png.freepik.com/512/5968/5968630.png" alt="Apple Pay">
                        </label>

                        <label class="payment-option">
                            <input type="radio" name="gateway" value="googlepay">
                            <img src="https://cdn-icons-png.flaticon.com/512/6124/6124998.png" alt="Google Pay">
                        </label>

                        <label class="payment-option">
                            <input type="radio" name="gateway" value="tng">
                            <img src="https://images.seeklogo.com/logo-png/53/3/touch-n-go-logo-png_seeklogo-534257.png" alt="TNG">
                        </label>
                    </div>
                    <div style="color: red; padding-top: 2%; font-weight: 1000;" id="optionError"></div>
                    <div style="display: flex; padding-top: 10%; justify-content: center;">
                        <input class="makePayment" type=submit value="Proceed To Secure Payment">
                    </div>
                </div>X
            </form>
        </div>
    </div>
    <div id="paymentSummary"></div>
        <div class="foreground" style="flex-direction: column">
            <h2 style="text-align: center;">Purchase Summary</h2>
            <div id="paymentInfo" style="display: flex;"></div>
        </div>
```

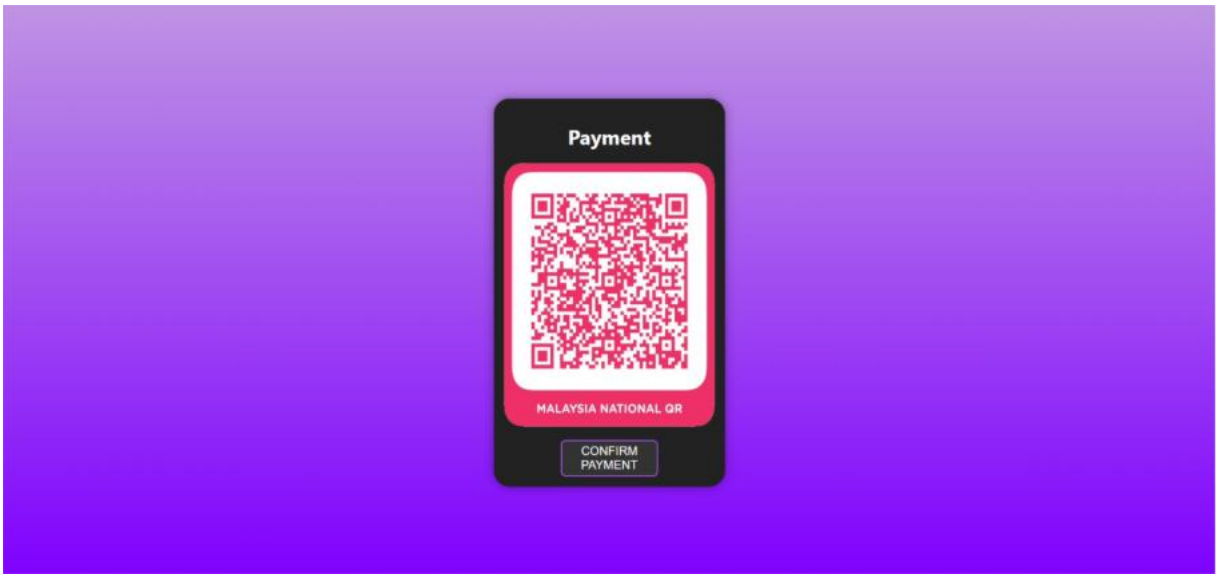
```
<script src="script.js"></script>
<script>
    document.addEventListener("DOMContentLoaded", () => {
        const userID = sessionStorage.getItem("loggedUserID");
        if (userID) {
            document.getElementById("userID").value = userID;
        }
    });

    window.onload = () => {
        const params = new URLSearchParams(window.location.search);
        const imdbId = params.get("movieID");

        displayInfo(imdbId);
        document.getElementById("movieID").value = imdbId;
    }
</script>
```

- HTML code for the content of web page
- Once html loaded it will retrieve the userID from the sessionStorage using sessionStorage.getItem("loggedUserID") and fill the hidden input field with the ID

userID with this value.

- When window fully loaded it will retrieve movieID from the url and use it to call displayInfo to display information about the movie.

(gateway.php)




```
<!DOCTYPE html>
<html>
    <head>
        <title>Absolute Cinema</title>
        <link rel="stylesheet" href="../styles.css"/>
    </head>
    <body>
        <main>
            <div class="gatewayContainer">
                <div id="gatewayInfo">
                    <h1>Payment</h1>
                    <img src="" id="QR" width="400px" height="500px"><br>
                    <a href="..\myTicket\"><button id="paymentButton">CONFIRM PAYMENT</button></a>
                </div>
            </div>
        </main>
        <script>
            const images = [
                "../images/QR/Angel.png",
                "../images/QR/Jeck.png",
                "../images/QR/Leon.png",
                "../images/QR/Quak.png",
            ];
            let randomIndex = Math.floor(Math.random() * images.length);
            console.log(randomIndex);
            document.getElementById("QR").src = images[randomIndex];

            document.getElementById("paymentButton").addEventListener("click", () => {
                window.alert("Your Ticket Purchase Was Successful!!!\nSee You At Absolute Cinema!!!")
            })
        </script>
    </body>
</html>
```

- This code is to display the qr. code for the payment and let user confirm their payment.

(script.js) for payment page

```
// fetch movie detail json using IMDB ID
async function fetchMovieDetail(imdbId) {
    try {
        const url = `https://api.imdbapi.dev/titles/${imdbId}`;
        const response = await fetch(url);
        const data = await response.json();
        console.log(data);

        return data;
    } catch (error) {
        console.error("Error fetching movie:", error);
        return null;
    }
}
```

- This function is used to fetch movie details

```
function changeAmount(amount) {
        const input = document.getElementById('ticketAmount');
        let current = parseInt(input.value);
        if (isNaN(current)) current = 1;
        const newValue = Math.max(1, current + amount);
        input.value = newValue;

        const ticketSelected = document.getElementById("ticketSelected");
        ticketSelected.textContent = `${newValue}`

        const totalPrice = document.getElementById("totalPrice");
        totalPrice.textContent = `RM ${(newValue * 16.5).toFixed(2)}`
    }
```

- This function is used to adjust the number of tickets and price by retrieving the current ticket amount from the input field and the updated ticket count is displayed in the ticketSelected element, and the total price is recalculated (newValue * 16.5), reflecting the updated ticket amount. There is validation done to ensure the number of tickets is valid.

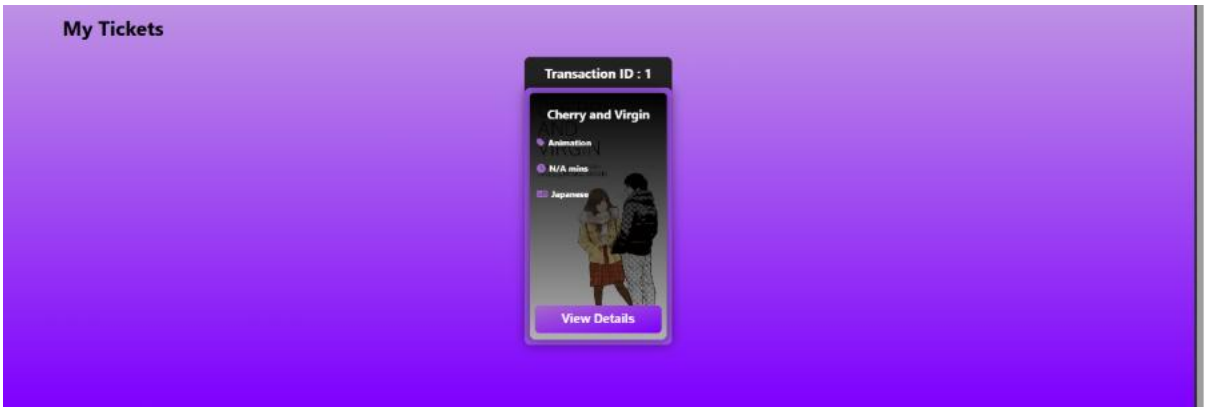
```
async function displayInfo(imdbId) {
    const container = document.getElementById("paymentInfo");
    const amount = parseInt(document.getElementById('ticketAmount').value);

    const movie = await fetchMovieDetail(imdbId);
    container.innerHTML = `
        <img src="${movie.primaryImage.url}" alt="${movie.originalTitle}">
        <div style="flex: 1; padding-left: 4%">
            <h3 style="text-align: center;">${movie.primaryTitle}</h3>
            <br>

            <div style="display: flex; justify-content: space-between;">
                <h5>Ticket Price:</h5>
                <h5>RM 15.00</h5>
            </div>

            <div style="display: flex; justify-content: space-between;">
                <h5>Service Tax (10%):</h5>
                <h5>RM 1.50</h5>
            </div>

            <hr>

            <div style="display: flex; justify-content: space-between;">
                <h5>Ticket Price (After Tax):</h5>
                <h5>RM 16.50</h5>
            </div>

            <div style="display: flex; justify-content: space-between;">
                <h5>Selected Amount:</h5>
                <h5 id="ticketSelected">${amount}</h5>
            </div>

            <hr>

            <div style="display: flex; justify-content: space-between;">
                <h5>Total Price:</h5>
                <h5 id="totalPrice">RM ${(amount * 16.5).toFixed(2)}</h5>
            </div>
        </div>
    `;
}
```

- This function Displays detailed movie information and the ticket purchase summary. It updates the page dynamically with the details of the selected movie and ticket information.

```
function validatePaymentOption() {
    const selected = document.querySelector('input[name="gateway"]:checked');
    const errorDiv = document.getElementById("optionError");
    if (!selected) {
        errorDiv.textContent = "PLEASE SELECT A PAYMENT METHOD!!!";
        return false;
    }
    errorDiv.textContent = "";
    return true;
}
```

- This function ensure user selects a payment method before submiting the form

MyTicket Page



(getTickets.php, getTransaction.php)

```php
$sql = "SELECT * FROM ticket WHERE transactionID = $transactionID ORDER BY ticketID";
$result = mysqli_query($conn, $sql);

$tickets = [];
while($row = mysqli_fetch_assoc($result)) {
    $tickets[] = $row;
}
echo json_encode($tickets);

mysqli_close($conn);
```

```php
$sql = "SELECT * FROM transactions WHERE userID = $userID ORDER BY transactionID DESC";
$result = mysqli_query($conn, $sql);

$transactions = [];
while($row = mysqli_fetch_assoc($result)) {
    $transactions[] = $row;
}
echo json_encode($transactions);

mysqli_close($conn);
```

- (getTickets.php) Select all from ticket with the specific transaction ID where all the ticket is order by their id. This code fetches the ticket information from database
- (getTransaction.php) Select all from transactions table with current user id and order all transaction by their id. This code fetches the transaction information from the database

(script.js) for **myTicket page**

```
// fetch movie detail json using IMDB ID
async function fetchMovieDetail(imdbId) {
    try {
        const url = `https://api.imdbapi.dev/titles/${imdbId}`;
        const response = await fetch(url);
        const data = await response.json();
        console.log(data);

        return data;
    } catch (error) {
        console.error("Error fetching movie:", error);
        return null;
    }
}
```

- FetchMovieDetail fetches the movie retrieves detailed movie information from the IMDb API using the imdb Id passed to it.

```
async function displayTransactions() {
    const container = document.getElementById("movieContainer");
    const userID = sessionStorage.getItem("loggedUserID");

    const response = await fetch(`getTransactions.php?userID=${userID}`);
    const data = await response.json();

    if (data.length === 0) {
        container.innerHTML = "<h3>No Ticket(s) Found.</h3>";
        return;
    }

    const movieDetails = await Promise.all(
        data.map(ticket => fetchMovieDetail(ticket.movieID))
    );
```

```
    data.map((transaction, i) => {
        const movie = movieDetails[i];

        const card = document.createElement('div');
        card.className = "ticket-card";

        const movieDiv = document.createElement("div");
        movieDiv.className = "movie-card";
        movieDiv.innerHTML = `<img src="${movie.primaryImage.url}" alt="${movie.originalTitle}">`;

        const movieOverlay = document.createElement("div");
        movieOverlay.className = "movie-overlay"
        movieOverlay.innerHTML = `
        <div style="height: 365px">
            <h3>${movie.primaryTitle}</h3>
            <h5><i class="fa-solid fa-tag" style="color: #A76BCE; padding-right: 1%"></i> ${movie.genres?.slice(0,3).join(', ') || 'N/A'}</h5>
            <h5><i class="fa-solid fa-clock" style="color: #A76BCE; padding-right: 1%"></i> ${movie.runtimeSeconds ? movie.runtimeSeconds / 60 : 'N/A'} mins</h5>
            <h5><i class="fa-solid fa-language" style="color: #A76BCE; padding-right: 1%"></i>
            ${movie.spokenLanguages && movie.spokenLanguages.length > 0 ? movie.spokenLanguages[0].name : 'N/A'}</h5>
        </div>
        <div style="display: flex; justify-content: center;">
            <a href="ticketInfo?movieID=${transaction.movieID}&transactionID=${transaction.transactionID}"><button class="viewTicket">View Details</button></a>
        </div>
        `;
        movieDiv.appendChild(movieOverlay);
        card.appendChild(movieDiv);

        const ticketTransaction = document.createElement("div");
        ticketTransaction.className = "ticketTransaction";
        ticketTransaction.innerHTML = `
        <h5 class="ticketH5">Transaction ID : ${transaction.transactionID}</h5>
        `;
        ticketTransaction.appendChild(card);

        container.appendChild(ticketTransaction);
    });
}
```

- The displayTransaction function fetches transactions data for a user and displays all transactions along with its movie details. It fetches all transactions according to userID from sessionStorage. It calls the displayMovieDetail for each transaction's movieID. If no transaction found it display no tickets found message.

```
async function displayTickets() {
    const container = document.getElementById("ticketContainer");
    const params = new URLSearchParams(window.location.search);
    const transactionID = params.get("transactionID");

    const response = await fetch(`getTickets.php?transactionID=${transactionID}`);
    const data = await response.json();

    const movie = await fetchMovieDetail(params.get("movieID"))

    data.map(ticketData => {
        const ticket = document.createElement("div");
        ticket.className = "ticketInfo";
        ticket.innerHTML = `
            <div style="flex: 1">
                <img src="ticketQR.png" alt="QR code" style="width: 90%; height: auto;">
            </div>
            <div style="flex: 4;">
                <h3>${movie.primaryTitle}</h3>
                <h4>Ticket ID: ${ticketData.ticketID}</h4>
            </div>
        `;

        container.appendChild(ticket);
    });
}
```

- This function fetches ticket details and movie information based on the transactionID and movieID, and then updates the page to display the ticket information.

MyWishlist Page

This page shows the user wishlist. It shows the movies that users added to their wishlist.

(getCurrentWishlist.php, getUpcomingWishlist.php)

```php
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT * FROM currentWishlist WHERE userID = $userID";
$result = mysqli_query($conn, $sql);

$tickets = [];
while($row = mysqli_fetch_assoc($result)) {
    $tickets[] = $row;
}
echo json_encode($tickets);

mysqli_close($conn);
```

```php
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT * FROM upcomingWishlist WHERE userID = $userID";
$result = mysqli_query($conn, $sql);

$tickets = [];
while($row = mysqli_fetch_assoc($result)) {
    $tickets[] = $row;
}
echo json_encode($tickets);

mysqli_close($conn);
```

- Both of these codes retrieve data from the wishlist (current or upcoming) of the current user.

(script.js) for **My Wishlist Page**

```
async function displayUpcomingWishlist() {
  const container = document.getElementById("movieContainer");
  container.innerHTML = '';
  const userID = sessionStorage.getItem("loggedUserID");


  const response = await fetch(`getUpcomingWishlist.php?userID=${userID}`);
  const data = await response.json();

  if (data.length === 0) {
      container.innerHTML = "<h3>No Item(s) Found.</h3>";
      return;
  }

  const movieDetails = await Promise.all(
      data.map(wishlist => fetchMovieDetail(wishlist.movieID))
  );

  data.map((wishlist, i) => {
    const movie = movieDetails[i];

    const card = document.createElement('div');
    card.className = "wishlist-card";

    const movieDiv = document.createElement("div");
    movieDiv.className = "movie-card";
    movieDiv.innerHTML = `<img src="${movie.primaryImage.url}" alt="${movie.originalTitle}">`;

    const movieOverlay = document.createElement("div");
    movieOverlay.className = "movie-overlay"
    movieOverlay.innerHTML = `
    <div style="height: 365px">
      <h3>${movie.primaryTitle}</h3>
      <h5><i class="fa-solid fa-tag" style="color: #A76BCE; padding-right: 1%"></i> ${movie.genres?.slice(0,3).join(', ') || 'N/A'}</h5>
      <h5><i class="fa-solid fa-language" style="color: #A76BCE; padding-right: 1%"></i> ${movie.spokenLanguages && movie.spokenLanguages.length > 0 ? movie.spokenLanguages[0].name : 'N/A'}</h5>
    </div>
    <div style="display: flex; justify-content: center;">
      <a href="wish-detail.php?movieID=${movie.id}"><button>VIEW INFO</button><a>
    </div>
    `;
    movieDiv.appendChild(movieOverlay);
    card.appendChild(movieDiv);
    container.appendChild(card);
  });
}
```

```
async function displayCurrentWishlist() {
  const container = document.getElementById("movieContainer");
  container.innerHTML = '';
  const userID = sessionStorage.getItem("loggedUserID");


  const response = await fetch(`getCurrentWishlist.php?userID=${userID}`);
  const data = await response.json();

  if (data.length === 0) {
      container.innerHTML = "<h3>No Item(s) Found.</h3>";
      return;
  }

  const movieDetails = await Promise.all(
      data.map(wishlist => fetchMovieDetail(wishlist.movieID))
  );

  data.map((wishlist, i) => {
    const movie = movieDetails[i];

    const card = document.createElement('div');
    card.className = "wishlist-card";

    const movieDiv = document.createElement("div");
    movieDiv.className = "movie-card";
    movieDiv.innerHTML = `<img src="${movie.primaryImage.url}" alt="${movie.originalTitle}">`;

    const movieOverlay = document.createElement("div");
    movieOverlay.className = "movie-overlay"
    movieOverlay.innerHTML = `
    <div style="height: 365px">
      <h3>${movie.primaryTitle}</h3>
      <h5><i class="fa-solid fa-tag" style="color: #A76BCE; padding-right: 1%"></i> ${movie.genres?.slice(0,3).join(', ') || 'N/A'}</h5>
      <h5><i class="fa-solid fa-language" style="color: #A76BCE; padding-right: 1%"></i> ${movie.spokenLanguages && movie.spokenLanguages.length > 0 ? movie.spokenLanguages[0].name : 'N/A'}</h5>
    </div>
    <div style="display: flex; justify-content: center;">
      <a href="../movie/movie-detail.php?movieID=${movie.id}"><button>VIEW INFO</button><a>
    </div>
    `;
    movieDiv.appendChild(movieOverlay);
    card.appendChild(movieDiv);
    container.appendChild(card);
  });
}
```

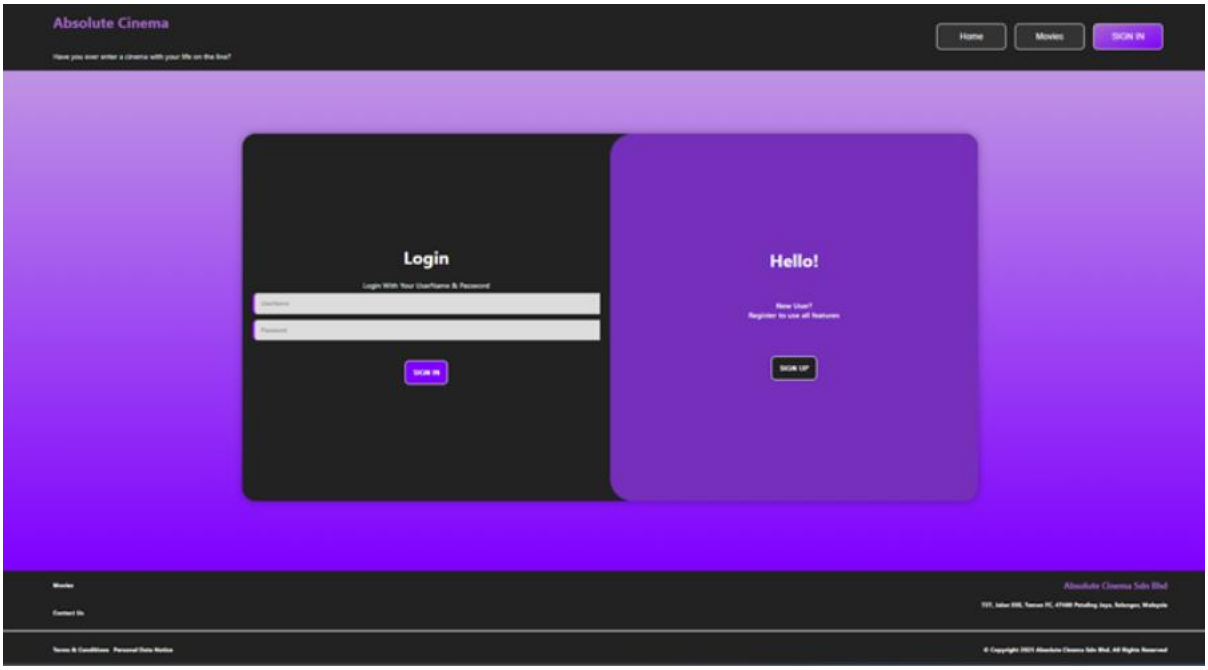
- fetchMovieDetail(imdbId), fetchMoviePreview(imdbId) will fetch the movie detail and preview video using IMDb id using the api. fetch(url) makes an asynchronous HTTP

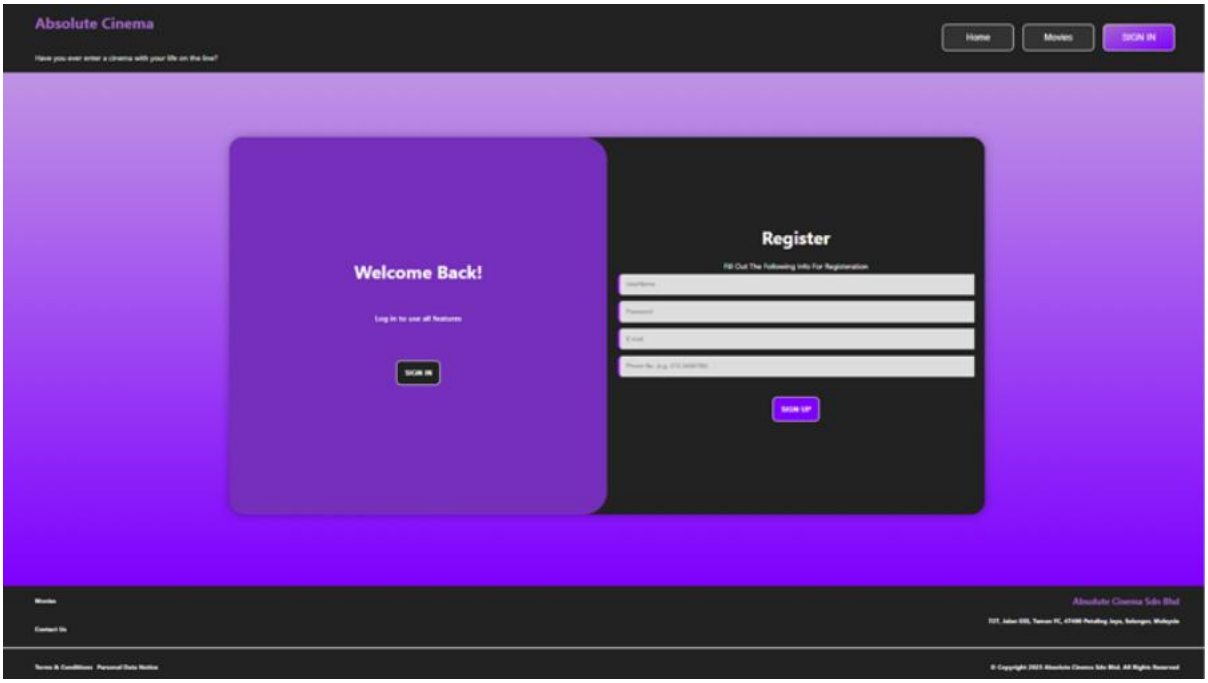
request to the API.

- displayPreview(imdbId), displayDetails(imdbId) will display movie preview and display. The displayDetails also include the option to add/remove the movie from the user's wishlist based on their login status.
- displayUpcomingWishlist(), displays the user's upcoming wishlist by fetching the movie details and displaying each movie's title and image. It also provides a "VIEW INFO" button for more details.
- displayCurrentWishlist() is like displayUpcomingWishlist(), but for the current wishlist.
- handleButtonClick(asyncMethod) manages the active state of the "Current" and "Upcoming" buttons. It disables both buttons while the data is being fetched, preventing multiple clicks, and re-enables them after the data is displayed.

Login Page

The login and registration page allows users to either sign in or create a new account through toggleable forms; when signing up, the system collects username, password, email, and phone number, then sends the data to `dbFunction.php` where `registerUser()` inserts it into the database and returns the new user ID to be stored in `sessionStorage` before redirecting to the homepage, while signing in sends the entered credentials to `validateUser()` which checks them against existing records and, if valid, returns the user ID for session storage and redirects to the homepage, otherwise displaying an error message.





(dbFunction.php) for **Login page**

```php
switch ($operationType) {
    // Sign in to Account
    case ('signin'):
        $userName = htmlspecialchars($_POST["userName"] ?? '');
        $userPassword = md5(htmlspecialchars($_POST['userPassword'] ?? ''));
        validateUser($conn, $userName, $userPassword);
        break;

    // Sign up new Account
    case ('signup'):
        $userName = htmlspecialchars($_POST['userName'] ??'');
        $userPassword = md5(htmlspecialchars($_POST['userPassword'] ??''));
        $userEmail = htmlspecialchars($_POST['userEmail'] ??'');
        $userPhoneNo = htmlspecialchars($_POST['userPhoneNo'] ??'');
        registerUser($conn, $userName, $userPassword, $userEmail, $userPhoneNo);
        break;
}
```

For **signin**:

- Retrieves userName from POST data and sanitizes it with htmlspecialchars.
- Retrieves userPassword, sanitizes it with htmlspecialchars, and hashes it
- Calls the validateUser() function, passing the database connection, username, and hashed password.

For **signup**:

- Retrieves userName.

- Retrieves and hashes userPassword
- Retrieves userEmail.
- Retrieves userPhoneNo.
- Calls the registerUser() function, passing the database connection and all four user details.

```php
// Validate User Details : Return userID if Valid, else return null
function validateUser($conn, $userName , $userPassword) {
    // Retrieve All Accounts
    $sql = 'SELECT userID, userName, userPassword FROM user';
    $result = mysqli_query($conn, $sql);

    // Validate User Name & Pasword (Case Sensitive)
    if (mysqli_num_rows($result) > 0) {
        while ($row = mysqli_fetch_assoc($result)) {
            if ($row['userName'] == $userName && $userPassword == $row['userPassword']) {
                echo json_encode(['success' => true, 'userID' => $row['userID']]);
                exit();
            }
        }
    }
    echo json_encode(['success' => false, 'error' => "INVALID USERNAME/PASSWORD!"]);
}

// Register New Account : Return new userID, else throw error
function registerUser($conn, $userName , $userPassword, $userEmail, $userPhoneNo) {
    // Prepare & Execute Query
    $stmt = mysqli_prepare($conn, "INSERT INTO user (userName, userPassword, userEmail, userPhoneNo) VALUES (?, ?, ?, ?)");
    mysqli_stmt_bind_param($stmt, "ssss", $userName, $userPassword, $userEmail, $userPhoneNo);

    try {
        mysqli_stmt_execute($stmt);

        // Retrieve User ID
        $last_id = mysqli_insert_id($conn);
        echo json_encode(['success' => true, 'userID' => $last_id]);
    }
    catch (Exception $e) {
        echo json_encode(['success' => false, 'error' => "USERNAME ALREADY EXISTS!"]);
    }
```

- Checks if a provided username and password combination exists in the database.
- Executes a SQL query to select all user IDs, names, and passwords from the user table.
- Loops through each row in the result set.
- Performs a case-sensitive comparison of the provided $userName and $userPassword (which is already MD5 hashed) against each record.
- On Success: Immediately outputs a JSON response with success: true and the matching userID, then terminates the script (exit()).
- On Failure: If no match is found after checking all records, outputs a JSON response with success: false and the error "INVALID USERNAME/PASSWORD!".

```php
// Register New Account : Return new userID, else throw error
function registerUser($conn, $userName , $userPassword, $userEmail, $userPhoneNo) {
    // Prepare & Execute Query
    $stmt = mysqli_prepare($conn, "INSERT INTO user (userName, userPassword, userEmail, userPhoneNo) VALUES (?, ?, ?, ?)");
    mysqli_stmt_bind_param($stmt, "ssss", $userName, $userPassword, $userEmail, $userPhoneNo);

    try {
        mysqli_stmt_execute($stmt);

        // Retrieve User ID
        $last_id = mysqli_insert_id($conn);
        echo json_encode(['success' => true, 'userID' => $last_id]);
    }
    catch (Exception $e) {
        echo json_encode(['success' => false, 'error' => "USERNAME ALREADY EXISTS!"]);
    }

    // Close stmt
    mysqli_stmt_close($stmt);
}
```

- Inserts a new user account into the database.
- Prepares an INSERT statement using mysqli_prepare for security, using placeholders (?) for values.
- Binds parameters to the statement, specifying all four values ($userName, $userPassword, $userEmail, $userPhoneNo) as strings ("ssss").
- Tries to execute the prepared statement inside a try-catch block.
- On Success: Retrieves the auto-generated userID of the new account using mysqli_insert_id, and outputs a JSON success response containing this new userID.
- On Failure (Exception): Catches any exception (e.g., a duplicate username violating a unique key) and outputs a JSON error response with success: false and the message "USERNAME ALREADY EXISTS!".
- Closes the prepared statement to free resources.

```
<main>
    <div class="container" style="align-items: center">
        <div class="loginContainer" id="loginContainer">
            <div class="form-loginContainer sign-up">
                <form id="signupForm">
                    <h1>Register</h1>
                    <span>Fill Out The Following Info For Registeration</span>
                    <input name="operationType" value="signup" hidden>
                    <input name="userName" type="text" placeholder="UserName" maxlength="50" required>
                    <input name="userPassword" type="password" placeholder="Password" maxlength="50" required>
                    <input name="userEmail" type="email" placeholder="E-mail" maxlength="50" required>
                    <input name="userPhoneNo" type="text" placeholder="Phone No. (e.g. 012-3456789)" maxlength="50" pattern="01[0-9]{1}-[0-9]{7,8}" required>
                    <errorText id="signupErrorText"></errorText>
                    <button>SIGN UP</button>
                </form>
            </div>
            <div class="form-loginContainer sign-in">
                <form id="signinForm">
                    <h1>Login</h1>
                    <span>Login With Your UserName & Password</span>
                    <input name="operationType" value="signin" hidden>
                    <input name="userName" type="text" placeholder="UserName" maxlength="50" required>
                    <input name="userPassword" type="password" placeholder="Password" maxlength="50" required>
                    <errorText id="signinErrorText"></errorText>
                    <button>SIGN IN</button>
                </form>
            </div>
            <div class="toggle-loginContainer">
                <div class="toggle">
                    <div class="toggle-panel toggle-left">
                        <h1>Welcome Back!</h1>
                        <p>Log in to use all features</p>
                        <button id="login">Sign In</button>
                    </div>
                    <div class="toggle-panel toggle-right">
```

- Login and registration form with a sliding panel toggle effect.

(script.js) for **Login page**

```
// Fetch Sign in
document.getElementById('signinForm').addEventListener('submit', async function(event) {
    event.preventDefault(); // Prevent default form submission

    const form = event.target;
    const formData = new FormData(form);

    fetch('dbFunction.php', {
            method: 'POST',
        body: formData,
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            sessionStorage.setItem('loggedUserID', data.userID);
            window.location.href = '../';
        }
        else {
            document.getElementById("signinErrorText").textContent = data.error;
        }
    })
    .catch(error => {
        console.error(error);
    });
});
```

- Handles the submission of the login form.
- Event Listener: Attaches to the submit event of the form with the ID signinForm.
- event.preventDefault() stops the browser's default form submission behavior, preventing a page reload.
- Creates a FormData object from the submitted form to package the input values (userName, userPassword, operationType) for sending.
- Sends POST request to dbFunction.php.
- On success:
  - Saves userID to sessionStorage.
  - Redirects to home page.
- On failure:
  - Displays error message.
  - Catches and logs errors.

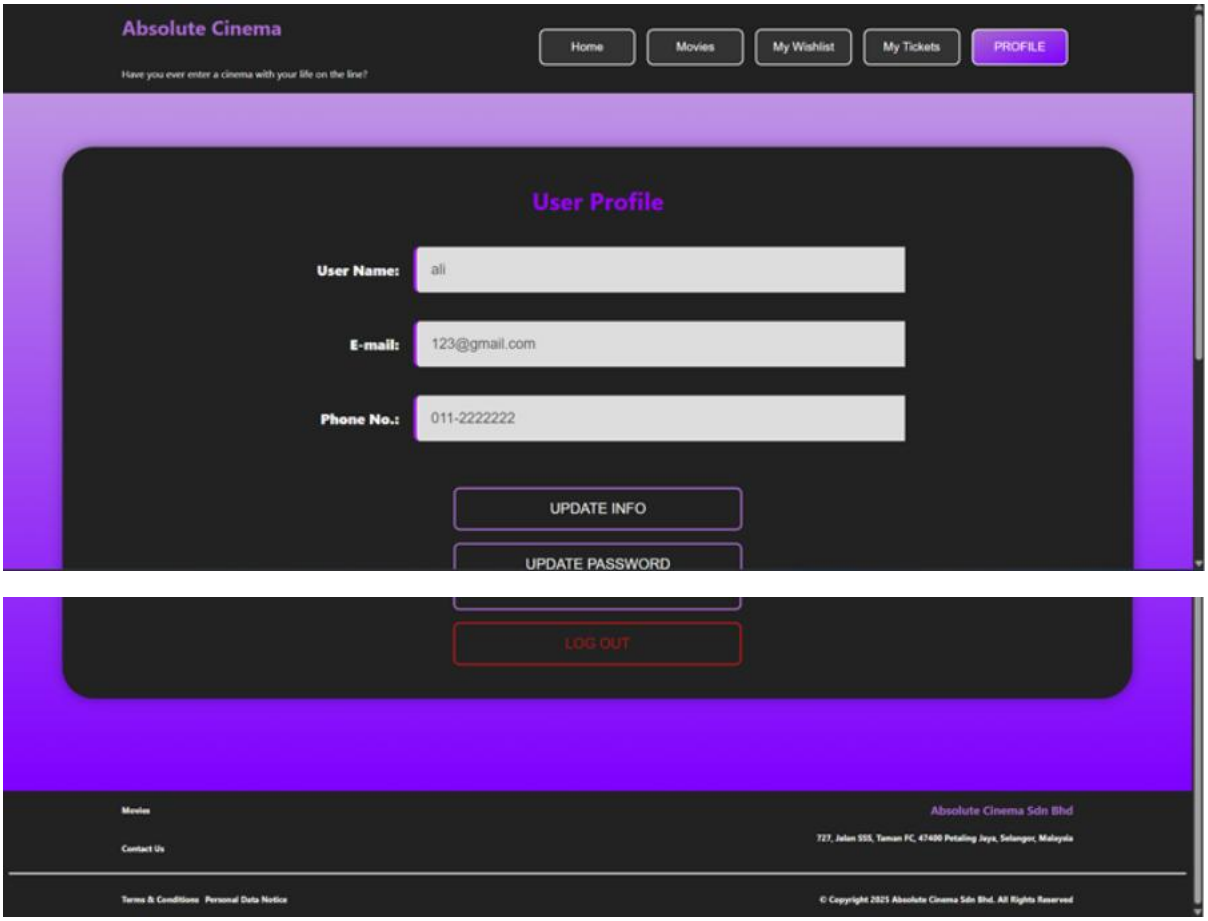
```
// Fetch Sign up
document.getElementById('signupForm').addEventListener('submit', async function(event) {
    event.preventDefault(); // Prevent default form submission

    const form = event.target;
    const formData = new FormData(form);

    fetch('dbFunction.php', {
        method: 'POST',
        body: formData,
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            sessionStorage.setItem('loggedUserID', data.userID);
            window.location.href = '../';
            window.alert("Account Registered Successfully");
        }
        else {
            document.getElementById("signupErrorText").textContent = data.error;
        }
    })
    .catch(error => {
        console.log(error);
    });
});
```

- Adds an event listener to the **signup form** so that this code runs when the user submits the form.
- Prevents the default browser behavior of reloading the page when the form is submitted.
- Stores a reference to the form element that triggered the event (the signup form).
- Creates a FormData object that automatically gathers all input values from the signup form (userName, userPassword, userEmail, userPhoneNo, and operationType).
- Sends the signup form data to PHP via POST.
- Converts the PHP response into JSON.
- Checks the JSON result:
- If data.success is true:
    - Save userID in sessionStorage.
    - Redirect to homepage (../).
    - Show alert “Account Registered Successfully”.
- If data.success is false:
-

- Show error message inside #signupErrorText

Profile Page

The profile page loads the logged-in user's details (name, email, phone) from the database using fetchUser() and displays them in read-only fields, while providing options to update info, update password, or log out. Clicking Update Info opens a form where changes are submitted to dbFunction.php (updateInfo), which updates the database and redirects back to the profile. Similarly, Update Password opens a form that validates the new password and updates it in the database (updatePassword) if valid. The Logout button clears the session and redirects the user to the login page, ensuring secure exit.



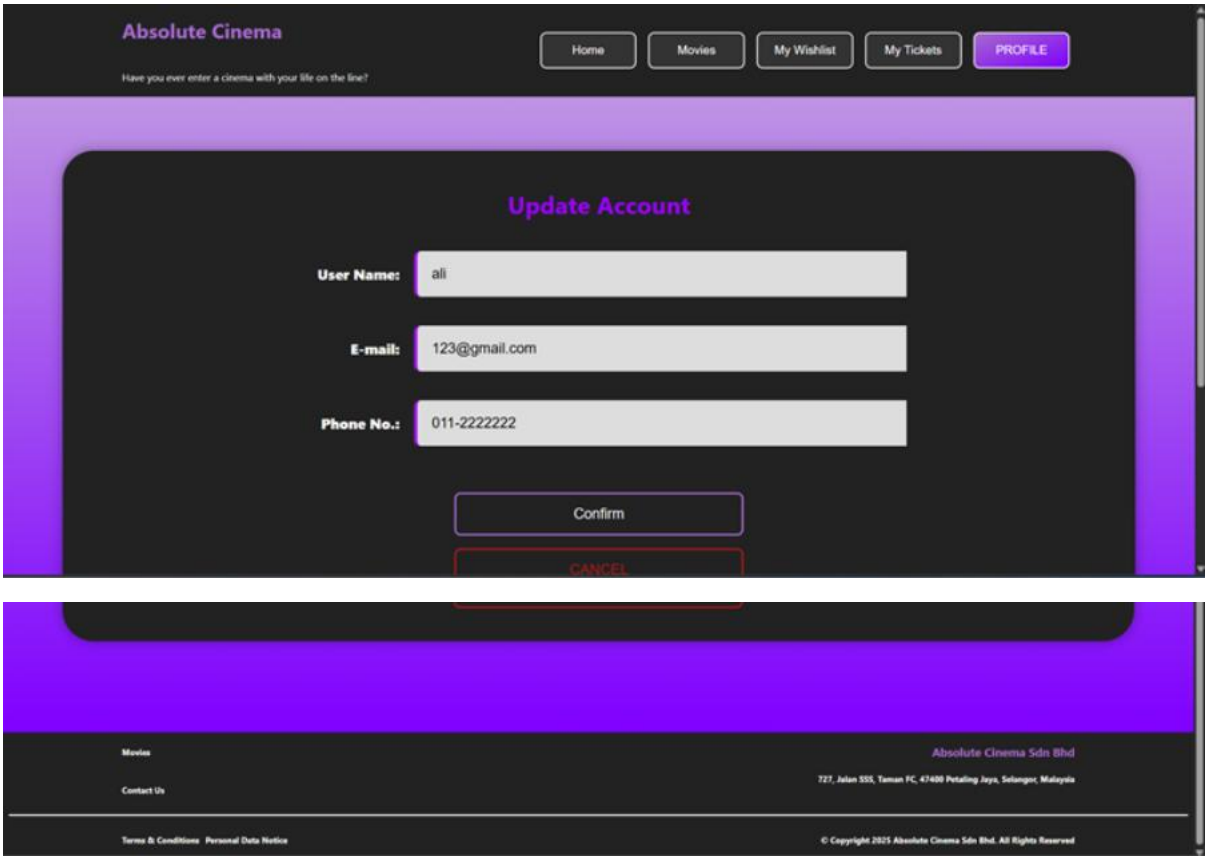
(dbfunction.php) for **profile page**

```php
// Fetch User By ID
if (isset($data['type'])) {
    $userID = $data["userID"] ?? 0;
    getUserByID($conn, $userID);
}
else {
    $operationType = $_POST['operationType'];
    switch ($operationType) {
        // Update Account Info
        case 'updateInfo':
            $userID = htmlspecialchars($_POST['userID'] ??'');
            $userName = htmlspecialchars($_POST['userName'] ??'');
            $userEmail = htmlspecialchars($_POST['userEmail'] ??'');
            $userPhoneNo = htmlspecialchars($_POST['userPhoneNo'] ??'');
            updateInfo($conn, $userName, $userEmail, $userPhoneNo, $userID);
            break;
        // Update Account Password
        case 'updatePassword':
            $userID = htmlspecialchars($_POST['userID'] ??'');
            $userPassword1 = md5(htmlspecialchars($_POST['userPassword1'] ??''));
            $userPassword2 = md5(htmlspecialchars($_POST['userPassword2'] ??''));
            updatePassword($conn, $userID, $userPassword1, $userPassword2);
            break;
```

- Fetches user details by calling getUserByID($conn, $userID) using the provided userID.
- If no user type is found, assumes the request is a form submission.
- Reads the operationType from the submitted form data.
- Based on the operationType value, performs one of two operations:
    - UpdateInfo: Calls updateInfo(userID, username, userEmail, userPhoneNo) to update the user's account details.
    - UpdatePassword: Executes a function to update the user's password.

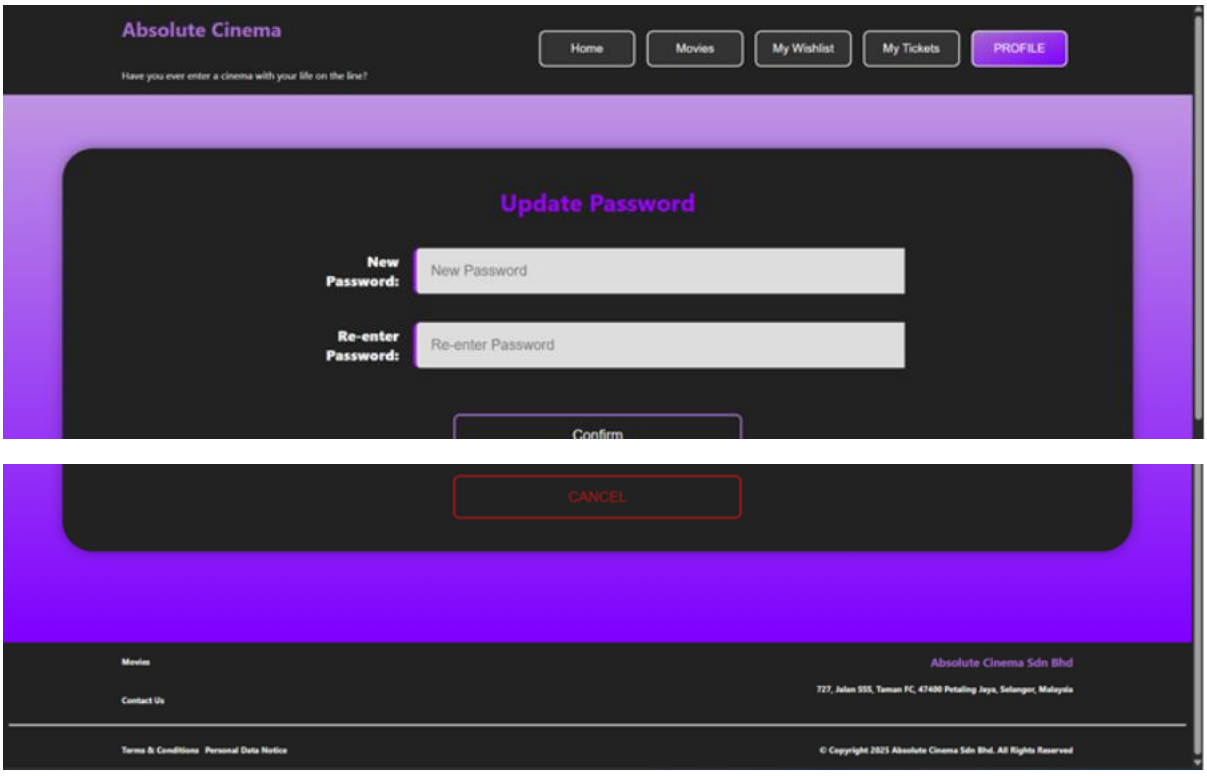
```php
// Fetch User By ID
function getUserByID($conn, $userID) {
    // Search User By ID Query
    $sql = "SELECT * FROM user WHERE userID = '". $userID ."'";
    $result = mysqli_query($conn, $sql);

    // Return Result
    if (mysqli_num_rows($result) > 0) {
        echo json_encode(['success' => true, 'userData' => mysqli_fetch_assoc($result)]);
    }
    else {
        echo json_encode(['success' => false, 'error' => "USER DOES NOT EXISTS!"]);
    }
}
```

- getUserByID retrieves a user from a database by their ID.



```php
// Update Account Info
function updateInfo($conn, $userName, $userEmail, $userPhoneNo, $userID) {
    // Prepare & Execute Querry
    $stmt = mysqli_prepare($conn, "UPDATE user SET userName = ?, userEmail = ?, userPhoneNo = ? WHERE userID = ?");
    mysqli_stmt_bind_param($stmt, "ssss", $userName, $userEmail, $userPhoneNo, $userID);

    // Successfully Updated Account
    try {
        mysqli_stmt_execute($stmt);
        echo json_encode(['success' => true]);
    }
    catch (Exception $e) {
        echo json_encode(['success' => false, 'error' => mysqli_stmt_error($stmt)]);
    }

    // Close stmt
    mysqli_stmt_close($stmt);
}
```

- Updates a user's account information (name, email, phone) in the database.
- Updates a user's account information (name, email, phone) in the database.



```php
// Update Account Password
function updatePassword($conn, $userID, $userPassword1, $userPassword2) {
    // Different Password
    if ($userPassword1 != $userPassword2) {
        echo json_encode(['success' => false, 'error' => "BOTH PASSWORDS ARE NOT EQUAL!"]);
    }
    else {
        // Search If Password is Same As Current
        $sql =  "SELECT userPassword FROM user WHERE userID = " . $userID;
        $result = mysqli_query($conn, $sql);
        if (mysqli_num_rows($result) > 0) {
            $data = mysqli_fetch_assoc($result);
            // Same Password As Current
            if ($userPassword1 == $data['userPassword']) {
                echo json_encode(['success' => false, 'error' => 'NEW PASSWORD IS EQUAL TO CURRENT PASSWORD!']);
            }
            else {
                // Prepare & Execute Querry
                $stmt = mysqli_prepare($conn, "UPDATE user SET userPassword = ? WHERE userID = ?");
                mysqli_stmt_bind_param($stmt, "ss", $userPassword1, $userID);
                try {
                    mysqli_stmt_execute($stmt);
                    echo json_encode(['success' => true]);
                }
                catch (Exception $e) {
                    echo json_encode(['success' => false, 'error' => mysqli_stmt_error($stmt)]);
                }

                // Close stmt
                mysqli_stmt_close($stmt);
            }
        }
        else {
            echo json_encode(['success' => false, 'error' => "USER ACCOUNT DOES NOT EXISTS!"]);
        }
```

- Updates a user's password after validating new credentials.
- Checks if the two new password inputs ($userPassword1, $userPassword2) match.
  - If not, returns an error: "BOTH PASSWORDS ARE NOT EQUAL!".
- If passwords match, fetches the current password from the database for the given userID.
  - If no user is found, returns an error: "USER ACCOUNT DOES NOT EXISTS!".
- Compares the new password with the current one from the database.
  - If they are the same, returns an error: 'NEW PASSWORD IS EQUAL TO CURRENT PASSWORD!'.
- If all checks pass, uses a prepared statement to securely update the password in the database.
- Returns a JSON success message on update, or an error message if any check fails or the query execution throws an exception.

(index.php) for **profile page**

```
<body onload="fetchUser()">
    <!--Header-->
    <?php
        include("../include/header.php");
    ?>

    <!--Body-->
    <main>
        <div class="container">
            <div class="profileContainer">
                <h1>User Profile</h1>
                <input id="userID" hidden>
                <div class="inputField">
                    <label for="userName">User Name:</label>
                    <input id="userName" disabled>
                </div>
                <div class="inputField">
                    <label for="userEmail">E-mail:</label>
                    <input id="userEmail" disabled>
                </div>
                <div class="inputField">
                    <label for="userPhoneNo">Phone No.:</label>
                    <input id="userPhoneNo" disabled>
                </div>
                <div class="errorField">
                    <errorText id="errorText"></errorText>
                </div>
                <div id="buttonRow">
                    <a href="updateInfo.php"><button>UPDATE INFO</button></a>
                    <a href="updatePassword.php"><button>UPDATE PASSWORD</button></a>
                    <a onclick="logout()"><button id="logout">LOG OUT</button></a>
                </div>
```

- Displays the logged-in user's profile information and provides options to update it.

(script.js) for **profile page**

```
function logout() {
    sessionStorage.removeItem("loggedUserID");
    window.location.href = "../userAuthentication/";
}
```

- Removes loggedUserID from sessionStorage.
-
```
// Return To Main Page
function back() {
    window.location.href = "../profile";
}
```
  Redirects the user to the ../userAuthentication/ directory.
- Redirects the user to the ../profile directory.

```javascript
// Fetch User
function fetchUser() {
    let userID = sessionStorage.getItem("loggedUserID");
    fetch('dbFunction.php', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            type: "fetchByID",
            userID: userID,
        }),
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            document.getElementById('userID').value = userID;
            document.getElementById('userName').value = data.userData.userName ?? "undefined";
            document.getElementById("userEmail").value = data.userData.userEmail ?? "undefined";
            document.getElementById("userPhoneNo").value = data.userData.userPhoneNo ?? "undefined";
        }
        else {
            window.alert("Error: " + data.error + "\nLogging Out of System");
            sessionStorage.removeItem("loggedUserID");
            window.location.href = "../userAuthentication/";
        }
    })
    .catch(error => {
        console.log(error);
    });
}
```

- Uses a prepared statement to securely set new values for a specific user ID.
- Outputs a JSON message indicating success or failure (with an error).
- Contains a syntax error in the SQL query (SEI instead of SET).
- Gets the loggedUserID from sessionStorage.
- Sends a POST request to dbFunction.php with the type fetchByID and the userID.
- On success, populates the HTML fields (userID, userName, userEmail, userPhoneNo) with the fetched user data.
- On failure, shows an alert with the error, removes the loggedUserID, and redirects to the authentication page.

```
// Update User Password
function setPasswordForm() {
    document.getElementById('userID').value = sessionStorage.getItem("loggedUserID");
    document.getElementById('passwordForm').addEventListener('submit', async function(event) {
        event.preventDefault(); // Prevent default form submission

        // Send Data in Form
        const form = event.target;
        const formData = new FormData(form);

        fetch('dbFunction.php', {
            method: 'POST',
            body: formData,
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                window.location.href = "../profile/"
                window.alert("Password Updated Successfully");
            }
            else {
                document.getElementById("errorText").textContent = data.error;
            }
        })
        .catch(error => {
            console.log(error);
        });
    });
}
```

- Sets the value of the hidden userID input field from sessionStorage.
- Attaches an event listener to the form with id passwordForm to handle its submission.
- Prevents the default form submission behavior.
- Sends the form data to dbFunction.php via a POST request.
- On success, redirects to ../profile/ and shows a success alert.
- On failure, displays the error message in the element with id errorText.

```
// Update User Info
function setUpdateForm() {
    document.getElementById('updateForm').addEventListener('submit', async function(event) {
        event.preventDefault(); // Prevent default form submission

        // Send Data in Form
        const form = event.target;
        const formData = new FormData(form);

        fetch('dbFunction.php', {
            method: 'POST',
            body: formData,
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                window.location.href = "../profile/"
                window.alert("Account Updated Successfully");
            }
            else {
                document.getElementById("errorText").textContent = data.error;
            }
        })
        .catch(error => {
            console.log(error);
        });
    });
}
```

- Attaches an event listener to the form with id updateForm to handle its submission.

- Prevents the default form submission behavior.
- Sends the form data to dbFunction.php via a POST request.
- On success, redirects to ../profile/ and shows a success alert.
- On failure, displays the error message in the element with id errorText.

(updateInfo.php)

- Provides a form for a user to edit and update their profile details (name, email, phone number).

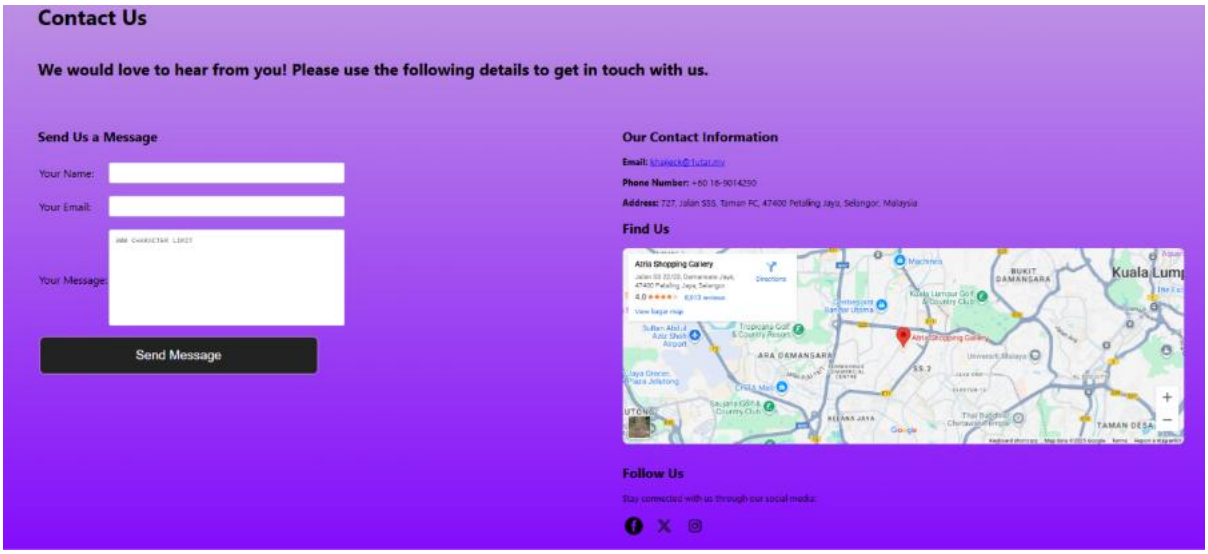
(updatePassword.php)

```
<body onload="setPasswordForm()">
    <!--Header-->
    <?php
        include("../include/header.php");
    ?>

    <!--Body-->
    <main>
        <div class="container">
            <div class="profileContainer">
                <h1> Update Password</h1>
                <form id="passwordForm">
                    <input name="operationType" value="updatePassword" hidden>
                    <input id="userID" name="userID" hidden>
                    <div class="inputField">
                        <label for="userPassword1">New<br>Password:</label>
                        <input id="userPassword1" name="userPassword1" type="password" placeholder="New Password" maxlength="50" required>
                    </div>
                    <div class="inputField">
                        <label for="userPassword2">Re-enter<br>Password:</label>
                        <input id="userPassword2" name="userPassword2" type="password" placeholder="Re-enter Password" maxlength="50" required>
                    </div>
                    <div class="errorField">
                        <errorText id="errorText"></errorText>
                    </div>
                    <div id="buttonRow">
                        <a><button>Confirm</button></a>
                        <a><button type="button" id="logout" onclick="back()">CANCEL</button></a>
                    </div>
                </form>
            </div>
        </div>
    </main>
```

- Provides a form for a user to change their password.

Contact Us Page

This page allows users to submit a message to the application by having a feedback form. It also shows the contact information to the users and the icons below the map direct user to the application social media page.



(process_contact.php)

```php
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "movie_ticketing";
    $name = $_POST['name'];
    $email = $_POST['email'];
    $message = $_POST['message'];

    $conn = mysqli_connect($servername, $username, $password, $dbname);

    if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
    }

    $sql = "INSERT INTO userMessage (name, email, message) VALUES (?, ?, ?)";
    $stmt = mysqli_prepare($conn, $sql);
    mysqli_stmt_bind_param($stmt, "sss", $name, $email, $message);

    if (mysqli_stmt_execute($stmt)) {
        header("Location: index.php?success=1");
        exit;
    } else {
        echo json_encode(['success' => false, 'error' => mysqli_stmt_error($stmt)]);
    }

    mysqli_stmt_close($stmt);
    mysqli_close($conn);
?>
```

- Connect to the movie_ticketing database, return error message if unable to connect
- Insert data into userMessage table.

(index.php) for **contact Page**

```
<script>
if (new URLSearchParams(window.location.search).get('success') === '1') {
    alert('Your message has been sent successfully!');
    // Optionally, remove the query string from the URL:
    history.replaceState(null, '', window.location.pathname);
}
</script>
```

- If url contains the query parameter, success = 1 and shows a success message to let user know message sent successfully and remove the query string success = 1.

Conclusion

In conclusion, the development of Absolute Cinema Movie Ticketing System successfully meets the requirements of the project. We successfully created a web application that integrates static and dynamic elements using client-side and server-side technologies. Absolute Cinema contains all the core functionality pages including home page, movie listing page, Wishlist page, profile page and contact us page. The system provides a user-friendly experience for users to easily view and buy tickets and manage their tickets while interacting with a visually appealing website.

Workload Summary

Name	Contribution
Leon Siow Yi Hong	Code for - Home Page - Style.css - User Login System Report
Quak Jing	Code for - Movies Page - Wishlist Page - Style.css Report
Koh Khai Jeck	Code for - Contact page - Wishlist page - Style.css Report
Angel Yap Yoon Ying	Code for - Home Page - Information Page - Style.css Report

References & Citations

Free IMDB API (no date). https://imdbapi.dev/.

Golden Screen Cinemas Malaysia | Showtimes & Tickets online (no date). https://www.gsc.com.my/.