

OPINIONATE

1 0 . 1 1 . 2 9 . 1 1 9

Jack Schneider, Andrew Doherty, Max Kempler, Rashan Khan, Aidan O'Leary

OPINIONATE

A ONE-STOP MEDIA RATING PLATFORM

Ever had an opinion on a show, movie or book and wanted to connect with others who share your interests?

Well, our platform is a one stop site that allows anyone to rate any media. With our product you're able to easily review your favorite content, see what your friends think, and join groups to explore new media together.

Discover new media content based on what you and others are watching. A spot for entertainment lovers to share, discuss and discover. Contribute to our growing library of reviews by signing up today!

PROJECT TIMELINE

- Milestone 0: Ideation phase
- Milestone 1: Key features, target audience, use cases, mockups/wireframes
- Milestone 2: Functional/Non-functional requirements, Tech stack
- Milestone 3: Development phase, present demo
- Milestone 4: Server setup, documentation, final presentation

FUNCTIONAL/NON-FUNCTIONAL REQUIREMENTS

Functional Requirements

- The ability to sign in and out of your account
- Create an account
- Delete your account
- Create a review
- Browse reviews
- Browse media
- Bookmark media
- Search for media
- Edit a review
- Display media data such as its picture, ratings, and rating descriptions

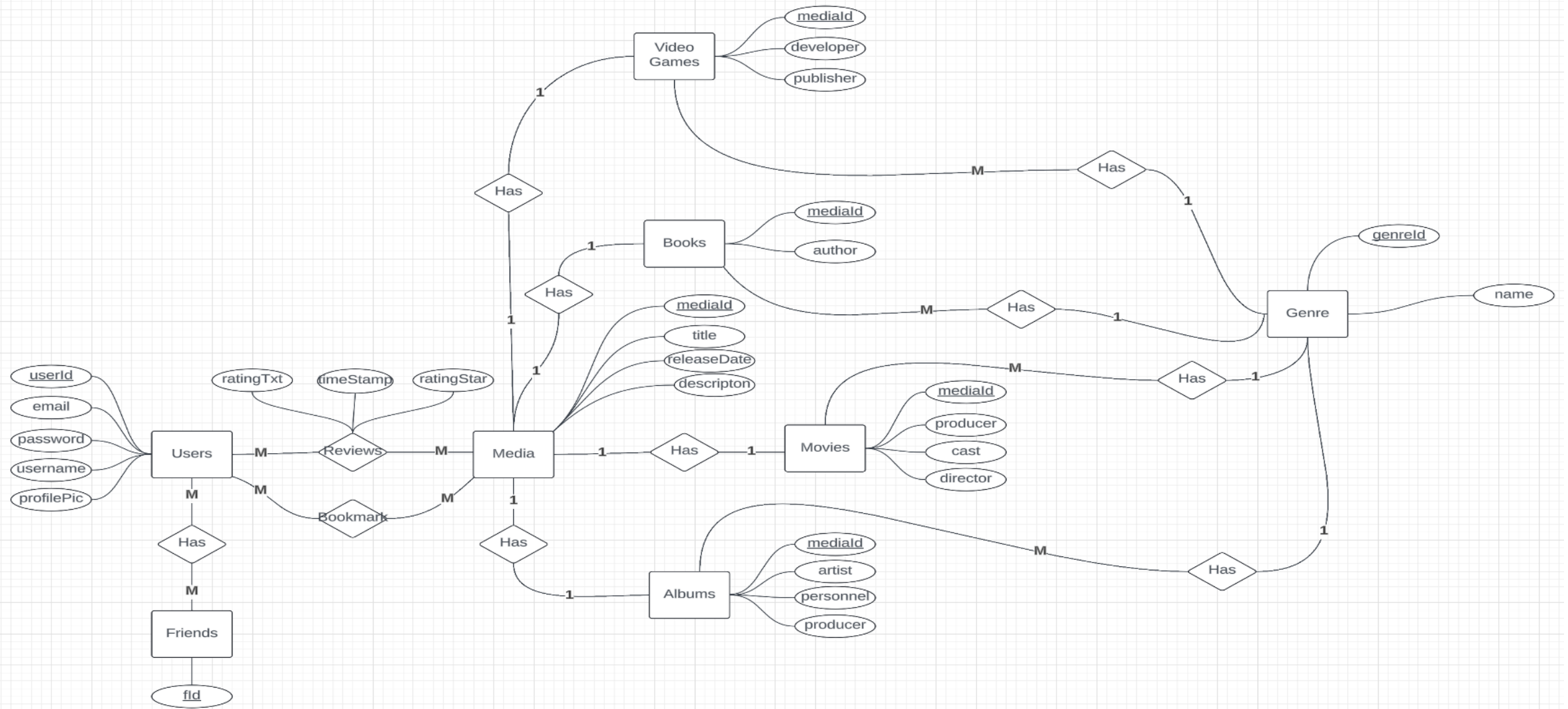
Non-Functional Requirements

- Support multiple users at once
- Display various forms of media
- Passwords will be encrypted by hashing
- The user interface should be straightforward, easy to use/navigate, and understandable
- Desktop site
- Allow users to choose media from a list

TECH STACK

- Node.js
- HTML
- CSS
- JavaScript
- Postgres (SQL)





ER DIAGRAM

DATABASE AND BACKEND COMMUNICATION

- Uses a connection pool to send HTTP requests
 - GET or POST
- Requests send SQL queries to insert or extract information
- Some queries return data as JSON

```
//Connect to the Remote Database.  
const pool = new Pool({  
  user: 'postgres',      // Database user  
  host: '10.11.29.119',  // Remote VM IP address  
  database: 'Opinionate', // Database name  
  password: 'capping2024', // Database password  
  port: 5432,            // PostgreSQL default port  
});
```

POPULATING OUR DATABASE

Fetching Spotify Info

```
const fetchMusicData = async (albumId) => {  
  try {  
    if (!spotifyAccessToken) await fetchSpotifyAccessToken();  
  
    // Fetch album info  
    const albumResponse = await fetch('https://api.spotify.com/v1/albums/${albumId}', {  
      headers: {  
        Authorization: `Bearer ${spotifyAccessToken}`,  
      },  
    });  
  }  
};
```

Json Response

```
spotify_id: '5EaE0Us301MZRicDMUIuqo',  
title: 'Lights And Sounds',  
releaseDate: '2006-01-01',  
coverUrl: 'https://i.scdn.co/image/ab67616d0000b2735c6f9b09bf2035d181e19aac',  
artist: 'Yellowcard',  
tracks: [  
  'Three Flights Up - Instrumental',  
  'Lights And Sounds',  
  'Down On My Head',  
  'Sure Thing Falling',  
  'City Of Devils',  
  'Rough Landing, Holly',  
  'Two Weeks From Twenty',  
  'Waiting Game',  
  'Martin Sheen Or JFK',  
  'Space Travel',  
  'Grey',  
  'Words, Hands, Hearts',  
  'How I Go',  
  'Holly Wood Died'  
]
```


ADDING INFO TO DATABASE

Data is assigned

```
const albumData = {
  spotify_id: album.id,
  title: album.name,
  releaseDate: album.release_date,
  coverUrl: album.images?.[0]?.url || null,
  artist: album.artists[0]?.name || null,
  tracks: album.tracks.items.map(track => track.name), // Track names in a single string
};
```

Data added to our DB

```
const insertMusicData = async (album) => {
  const client = await pool.connect();
  try {
    await client.query('BEGIN');
    const mediaResult = await client.query(
      `INSERT INTO media ("mediaId", title, "releaseDate", description)
      VALUES ($1, $2, $3, $4) RETURNING "mediaId"`,
      [mediaID, album.title, album.releaseDate, album.description || null]
    );

    await client.query(
      `INSERT INTO albums ("mediaID", artist, cover_url, tracks)
      VALUES ($1, $2, $3, $4)`,
      [mediaID, album.artist, album.coverUrl, album.tracks]
    );
    //mediaID++
    await client.query('COMMIT');
    console.log(`Album "${album.title}" inserted successfully`);
  } catch (error) {
    await client.query('ROLLBACK');
    console.error('Error inserting album data:', error);
  } finally {
    client.release();
  }
};
```

JSON -> DISPLAY

```
spotify_id: '5EaE0Us301MZRicDMUIuqo',
title: 'Lights And Sounds',
releaseDate: '2006-01-01',
coverUrl: 'https://i.scdn.co/image/ab67616d0000b2735c6f9b09bf2035d181e19aac',
artist: 'Yellowcard',
tracks: [
  'Three Flights Up - Instrumental',
  'Lights And Sounds',
  'Down On My Head',
  'Sure Thing Falling',
  'City Of Devils',
  'Rough Landing, Holly',
  'Two Weeks From Twenty',
  'Waiting Game',
  'Martin Sheen Or JFK',
  'Space Travel',
  'Grey',
  'Words, Hands, Hearts',
  'How I Go',
  'Holly Wood Died'
]
```



Lights And Sounds

Yellowcard

Release Date: 2006-01-01

Tracks:

- Three Flights Up - Instrumental
- Lights And Sounds
- Down On My Head
- Sure Thing Falling
- City Of Devils
- Rough Landing, Holly
- Two Weeks From Twenty
- Waiting Game
- Martin Sheen Or JFK
- Space Travel
- Grey
- Words, Hands, Hearts
- How I Go
- Holly Wood Died

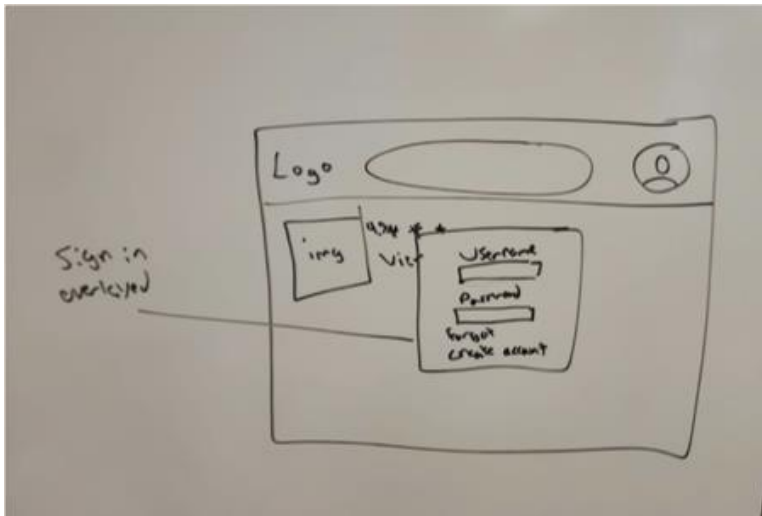
Average Rating: No reviews yet

★★★★★ 0 Stars

Write your review here...

SIGN-IN/CREATE ACCOUNT

Original Design



Final Design

Search for media

Sign In

Username:

Password:

Sign In

[Or click here to create an account](#)

Search for media

Create an Account

Username:

Email:

Password:

Confirm Password:

Create Account

[Already have an account? Sign in here](#)

ACCOUNT SIGN-IN/CREATION

```
// sign in
document.getElementById('login-form').addEventListener('submit', async (event) => {
  event.preventDefault();
  const username = document.getElementById('sign-in-username').value;
  const password = document.getElementById('sign-in-password').value;
  const loginError = document.getElementById('login-error');

  try {
    const response = await fetch('http://localhost:3000/login', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ username, password })
    });
    const result = await response.json();

    if (result.success) {
      window.location.href = 'userProfile.html';
    } else {
      loginError.textContent = result.message;
      loginError.style.display = 'block';
    }
  } catch (error) {
    loginError.textContent = 'An error occurred during login. Please try again.';
    loginError.style.display = 'block';
    console.error('Login request error:', error); // Log client-side errors for debugging
  }
});
```

```
// Handle Account Creation Form Submission
document.getElementById('register-form').addEventListener('submit', async (event) => {
  event.preventDefault();

  const username = document.getElementById('username').value;
  const email = document.getElementById('email').value;
  const password = document.getElementById('password').value;
  const confirmPassword = document.getElementById('confirm-password').value;
  const registerError = document.getElementById('register-error');

  // Check if passwords match
  if (password !== confirmPassword) {
    registerError.textContent = 'Passwords do not match. Please try again.';
    registerError.style.display = 'block';
    return;
  }

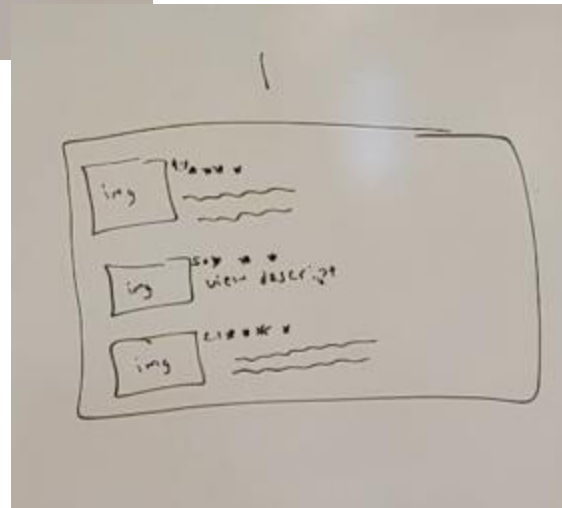
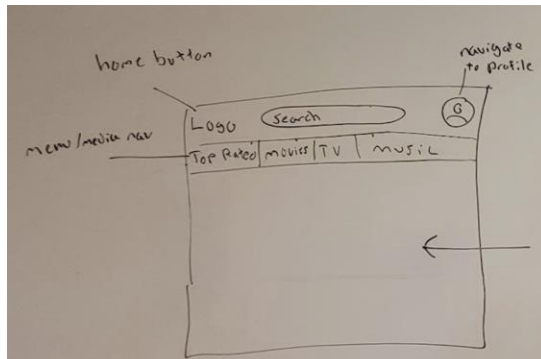
  try {
    // Send request to server to create a new account
    const response = await fetch('http://localhost:3000/register', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ username, password, email })
    });

    const result = await response.json();

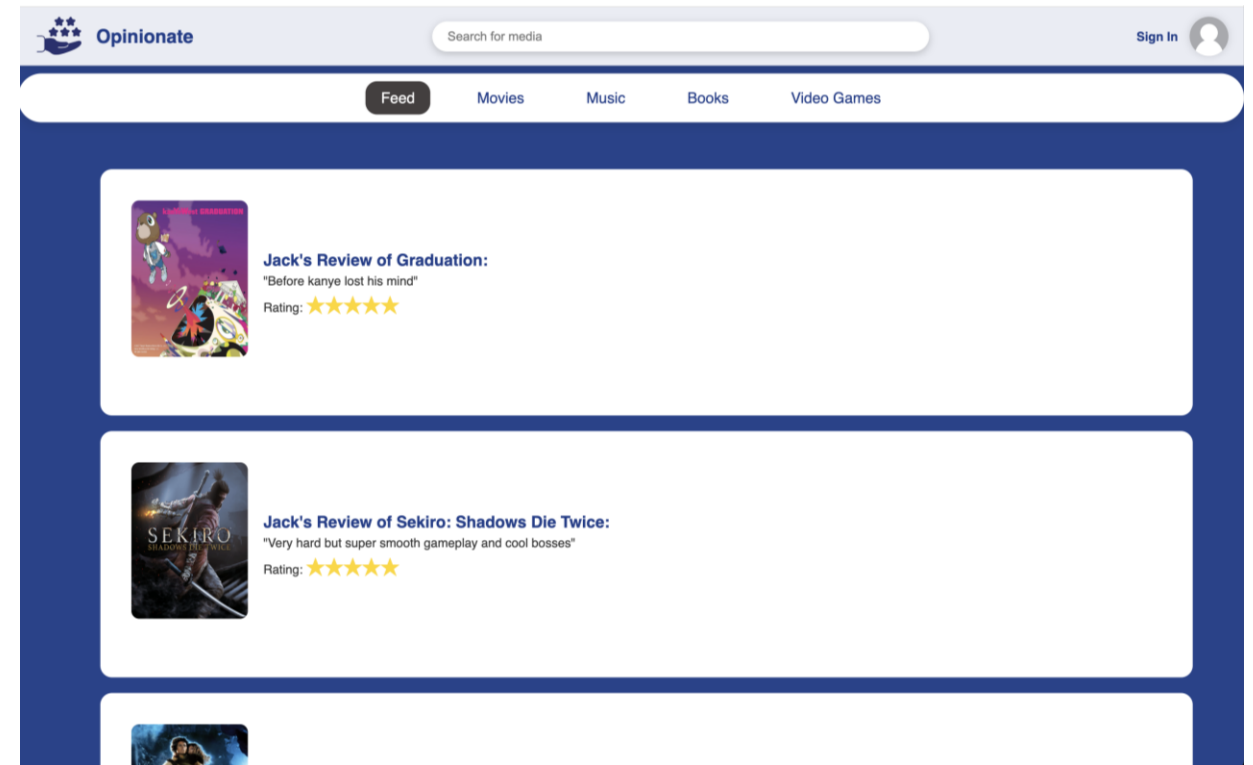
    if (result.success) {
      // Account created successfully; redirect to sign-in or another page
      alert(result.message); // shows success message (do we want this)
      window.location.href = 'userProfile.html';
    } else {
      // Show error message from server response
      registerError.textContent = result.message;
      registerError.style.display = 'block';
    }
  } catch (error) {
    registerError.textContent = 'An error occurred. Please try again.';
    registerError.style.display = 'block';
  }
});
```

LANDING/FEED PAGE

Original Design



Final Design



DISPLAYING REVIEWS AND AVERAGE RATINGS

```
app.get('/getReviews', async (req, res) => {
  try {
    const query = `
      SELECT
        reviews.*,
        media.title,
        users.username,
        COALESCE(movies.poster_url, albums.cover_url, books.cover_url, "videoGames".poster_url) AS cover_url,
        CASE
          WHEN movies."mediaID" IS NOT NULL THEN 'Movie'
          WHEN albums."mediaID" IS NOT NULL THEN 'Album'
          WHEN books."mediaID" IS NOT NULL THEN 'Book'
          WHEN "videoGames"."mediaID" IS NOT NULL THEN 'Video Game'
        END AS mediaType
      FROM reviews
      JOIN media ON reviews."mediaID" = media."mediaId"
      JOIN users ON reviews."userID" = users."userID"
      LEFT JOIN movies ON media."mediaId" = movies."mediaID"
      LEFT JOIN albums ON media."mediaId" = albums."mediaID"
      LEFT JOIN books ON media."mediaId" = books."mediaID"
      LEFT JOIN "videoGames" ON media."mediaId" = "videoGames"."mediaID"
      ORDER BY reviews.review_id DESC;
    `;

    const result = await pool.query(query);
```

```
app.get('/getReviewsForMedia/:id', async (req, res) => {
  const mediaID = req.params.id; // Get mediaID from the query parameters
```

```
app.get('/getAverageRating/:mediaID', async (req, res) => {
  const { mediaID } = req.params; // Extract mediaID from the request parameters

  try {
    const query = `
      SELECT
        AVG(reviews."ratingStar") AS averageRating,
        COUNT(reviews."mediaID") AS reviewCount
      FROM reviews
      WHERE reviews."mediaID" = $1
      GROUP BY reviews."mediaID";
    `;

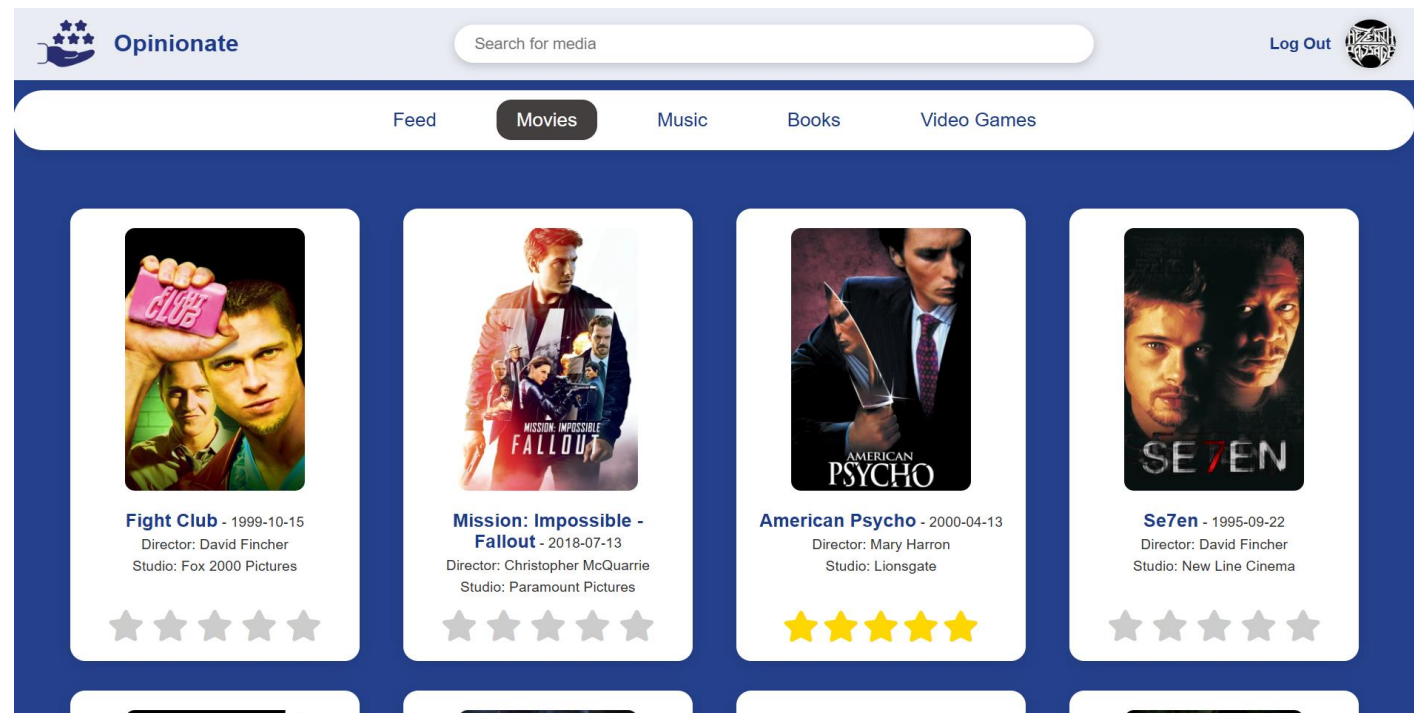
    const result = await pool.query(query, [mediaID]);
```

MEDIA TABS

Original Design

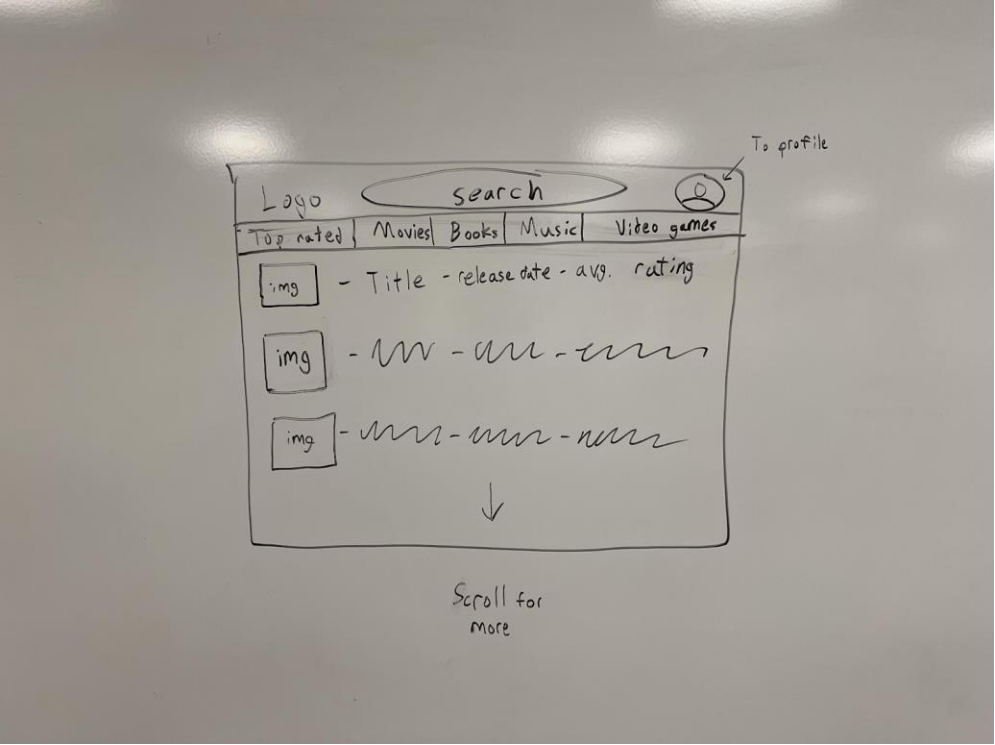


Final Design

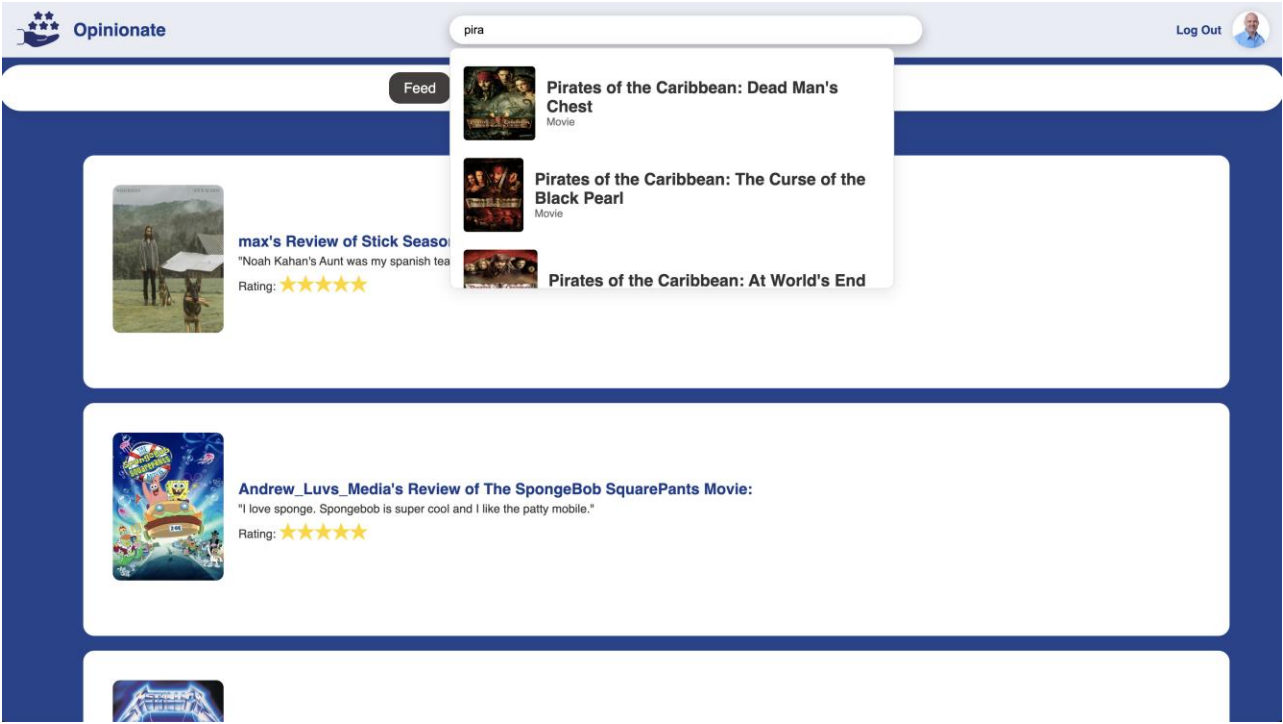


SEARCH FOR MEDIA

Original Design



Final Design



SEARCH FOR MEDIA - CODE

```
// Search functionality
const searchInput = document.getElementById('search-input');
const suggestionsContainer = document.getElementById('search-suggestions');

let searchTimeout;

searchInput.addEventListener('input', function () {
  clearTimeout(searchTimeout);
  const query = this.value.trim();

  if (query.length > 0) {
    searchTimeout = setTimeout(() => {
      fetchSearchSuggestions(query);
    }, 300);
  } else {
    suggestionsContainer.innerHTML = '';
    suggestionsContainer.style.display = 'none';
  }
});

function fetchSearchSuggestions(query) {
  fetch(`/api/search?q=${encodeURIComponent(query)}`)
    .then(response => response.json())
    .then(data => {
      displaySearchSuggestions(data);
    })
    .catch(error => {
      console.error('Error fetching search suggestions:', error);
    });
}
```

```
function displaySearchSuggestions(suggestions) {
  suggestionsContainer.innerHTML = '';

  if (suggestions.length > 0) {
    suggestions.forEach(item => {
      const suggestionItem = document.createElement('div');
      suggestionItem.classList.add('suggestion-item');

      suggestionItem.innerHTML = `

<div>
  <p><strong>${item.title}</strong></p>
  <p>${item.mediaType}</p>
</div>
`;

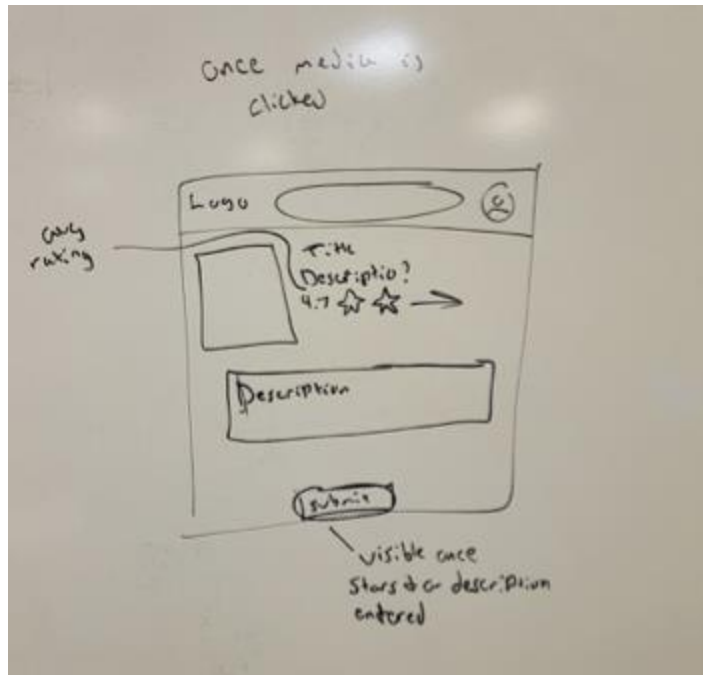
      suggestionItem.addEventListener('click', () => {
        let url = '';
        if (item.mediaType === 'Movie') {
          url = `mediaMovies.html?id=${item.mediaId}`;
        } else if (item.mediaType === 'Album') {
          url = `mediaMusic.html?id=${item.mediaId}`;
        } else if (item.mediaType === 'Book') {
          url = `mediaBooks.html?id=${item.mediaId}`;
        } else if (item.mediaType === 'Video Game') {
          url = `mediaVG.html?id=${item.mediaId}`;
        }
        window.location.href = url;
      });

      suggestionsContainer.appendChild(suggestionItem);
    });

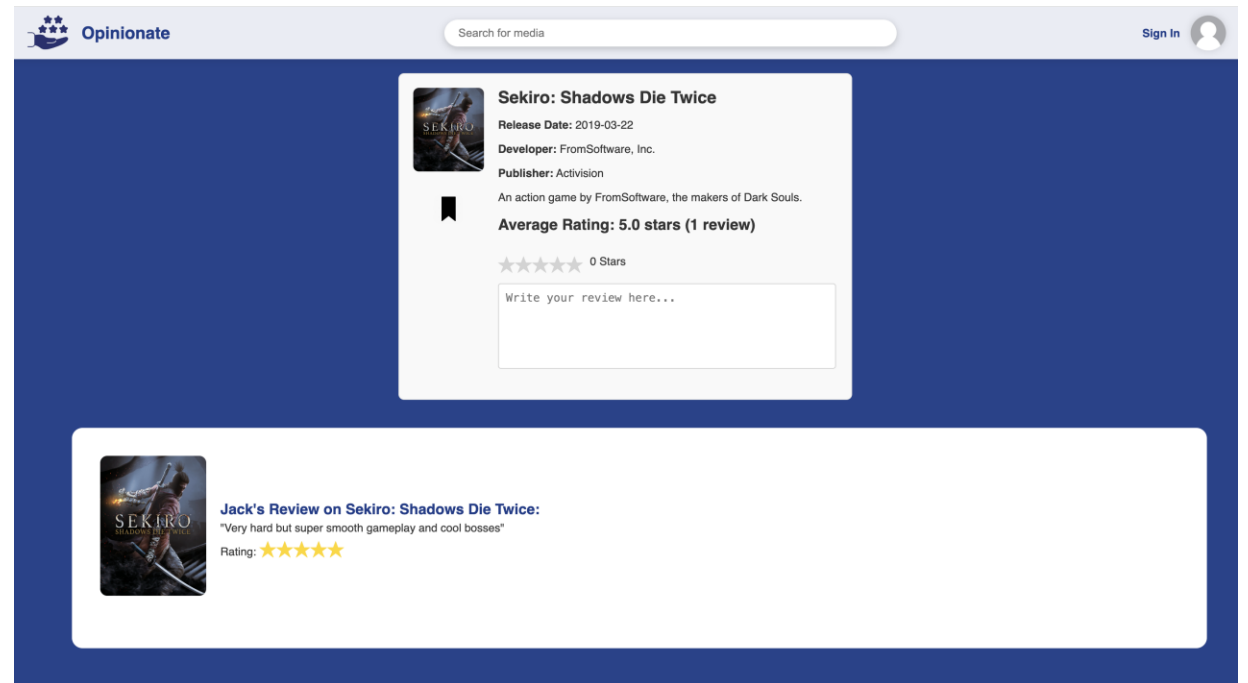
    suggestionsContainer.style.display = 'block';
  } else {
    suggestionsContainer.style.display = 'none';
  }
}
```

RATING PAGES

Original Design



Final Design



SUBMITTING REVIEWS

```
// Route to submit a review
app.post('/submitReview', async (req, res) => {
  // Ensure the user is logged in
  if (!req.session.user) {
    return res.status(401).json({ success: false, message: 'User not logged in' });
  }

  const { mediaID, reviewText, rating } = req.body;
  const userID = req.session.user.id; // Retrieve userID from session

  try {
    const query = `
      INSERT INTO reviews ("userID", "mediaID", "ratingTxt", "ratingStar")
      VALUES ($1, $2, $3, $4)
      RETURNING *;
    `;
    const values = [userID, mediaID, reviewText, rating];
    const result = await pool.query(query, values);

    res.status(200).json({ success: true, review: result.rows[0] });
  } catch (error) {
    console.error('Error inserting review:', error);
    res.status(500).json({ success: false, message: 'Error saving review' });
  }
});
```



Spider-Man

Director: Sam Raimi

Release Date: 2002-05-01

Cast: Tobey Maguire, Willem Dafoe, Kirsten Dunst, James Franco

After being bitten by a genetically altered spider at Oscorp, nerdy but endearing high school student Peter Parker is endowed with amazing powers to become the superhero known as Spider-Man.

Average Rating: 5.0 stars (1 review)

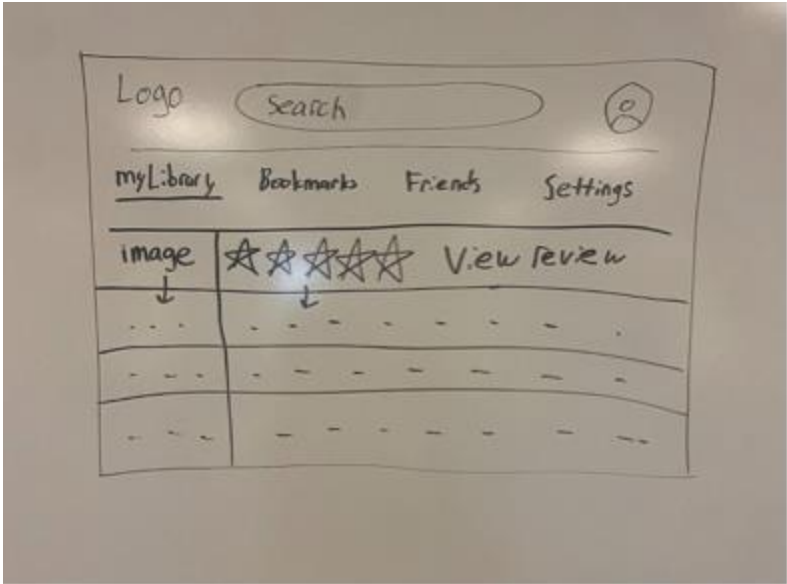
★★★★★ 5 Stars

Great movie!

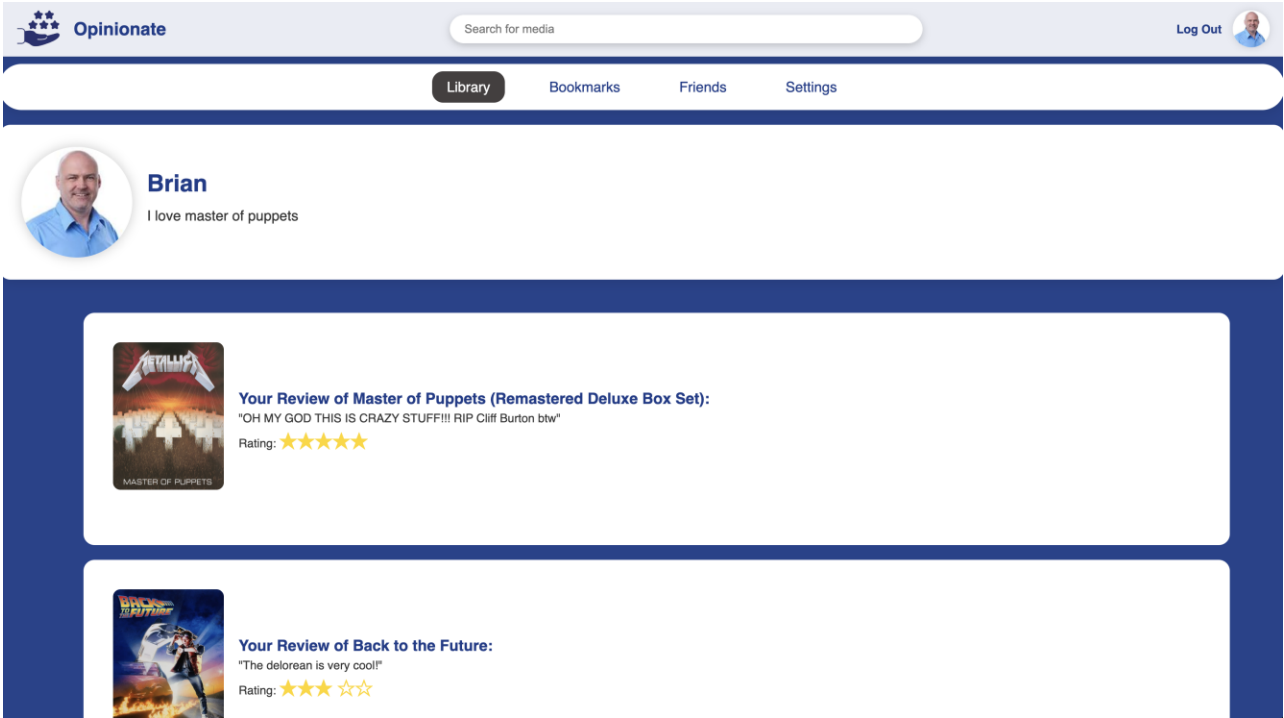
Submit

PROFILE PAGE

Original Design



Final Design



PROFILE BIO DISPLAY

```
<div class="profile-container">
  <div class="profile-picture">
    <img id="profile-picture-display" alt="User Profile Picture">
  </div>
  <div class="user-description">
    <h2 id="username-display"></h2>
    <p id="user-description-text"></p>
  </div>
</div>
```

```
// loads the user profile
async function loadUserProfile() {
  console.log("loading user profile");
  try {
    const response = await fetch('/user-info');
    const data = await response.json();

    if (data.success) {
      // Update bio section
      document.getElementById('username-display').textContent = data.username;
      document.getElementById('user-description-text').textContent = data.description || "No description provided.";
      document.getElementById('email').textContent = data.email || "No email provided.";

      // Update bio profile picture
      if (data.profilePicture) {
        document.getElementById('profile-picture-display').src = `data:image/jpeg;base64,${data.profilePicture}`;
      }

      // Update settings section profile picture
      if (data.profilePicture) {
        document.getElementById('settings-profile-picture-display').src = `data:image/jpeg;base64,${data.profilePicture}`;
      }

      // Update top-right profile icon
      if (data.profilePicture) {
        document.getElementById('top-right-profile-picture').src = `data:image/jpeg;base64,${data.profilePicture}`;
      }
    }
  } catch (error) {
    console.error('Error loading user profile:', error);
  }
}
```

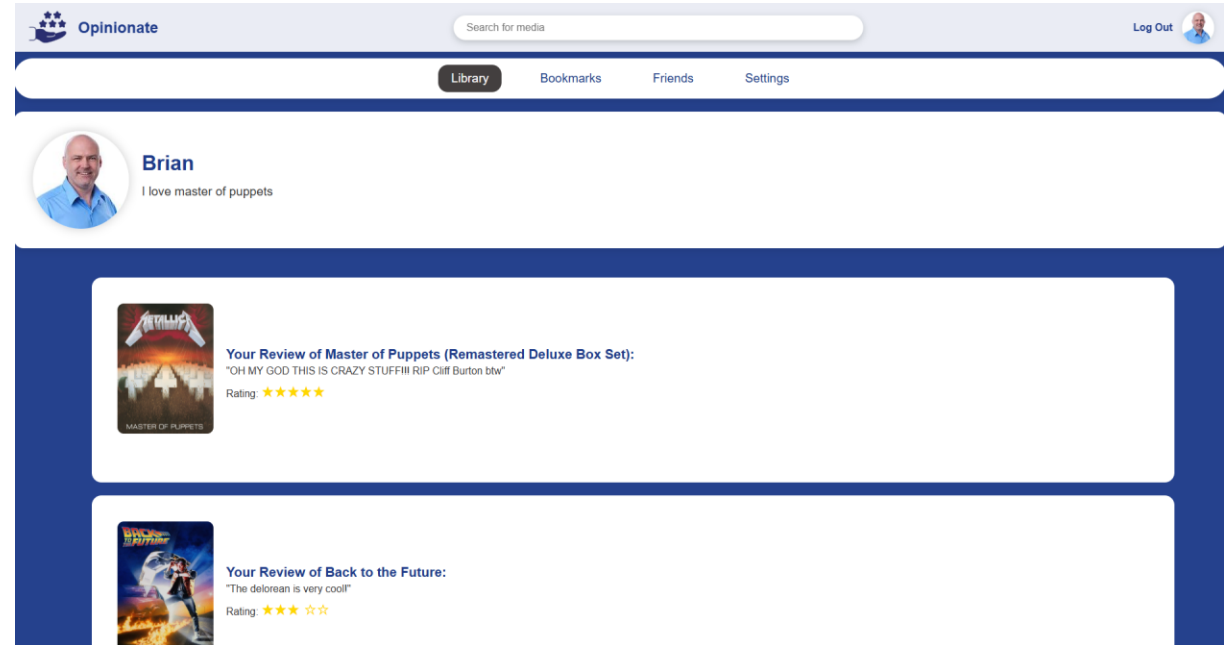
LIBRARY PAGE

```
// Route to fill the user's library tab in their user profile page.
app.get('/getUserReviews', async (req, res) => {
  try {
    // Retrieve userID from the session
    const userID = req.session.user.id;
    if (!userID) {
      return res.status(401).json({ error: 'User not authenticated.' });
    }

    const query = `
      SELECT
        reviews.*,
        media.title,
        users.username,
        COALESCE(movies.poster_url, albums.cover_url, books.cover_url, "videoGames".poster_url) AS cover_url,
        CASE
          WHEN movies."mediaID" IS NOT NULL THEN 'Movie'
          WHEN albums."mediaID" IS NOT NULL THEN 'Album'
          WHEN books."mediaID" IS NOT NULL THEN 'Book'
          WHEN "videoGames"."mediaID" IS NOT NULL THEN 'Video Game'
        END AS mediaType
      FROM reviews
      JOIN media ON reviews."mediaID" = media."mediaId"
      JOIN users ON reviews."userID" = users."userID"
      LEFT JOIN movies ON media."mediaId" = movies."mediaID"
      LEFT JOIN albums ON media."mediaId" = albums."mediaID"
      LEFT JOIN books ON media."mediaId" = books."mediaID"
      LEFT JOIN "videoGames" ON media."mediaId" = "videoGames"."mediaID"
      WHERE reviews."userID" = $1;
    `;

    const result = await pool.query(query, [userID]);

    res.status(200).json({ success: true, reviews: result.rows });
  } catch (error) {
    console.error('Error fetching user reviews with cover URLs:', error);
    res.status(500).json({ error: 'An error occurred while fetching user reviews.' });
  }
});
```



BOOKMARKS PAGE


```
app.get('/getUserBookmarks', async (req, res) => {
  // Ensure the user is logged in
  if (!req.session.user) {
    return res.status(401).json({ success: false, message: 'User not logged in' });
  }


  const userID = req.session.user.id;

  try {
    const query = `
      SELECT
        bookmark.*,
        media.title,
        COALESCE(movies.poster_url, albums.cover_url, books.cover_url, "videoGames".poster_url) AS cover_url,
        CASE
          WHEN movies."mediaID" IS NOT NULL THEN 'Movie'
          WHEN albums."mediaID" IS NOT NULL THEN 'Album'
          WHEN books."mediaID" IS NOT NULL THEN 'Book'
          WHEN "videoGames"."mediaID" IS NOT NULL THEN 'Video Game'
        END AS mediaType
      FROM bookmark
      JOIN media ON bookmark."mediaID" = media."mediaId"
      LEFT JOIN movies ON media."mediaId" = movies."mediaID"
      LEFT JOIN albums ON media."mediaId" = albums."mediaID"
      LEFT JOIN books ON media."mediaId" = books."mediaID"
      LEFT JOIN "videoGames" ON media."mediaId" = "videoGames"."mediaID"
      WHERE bookmark."userID" = $1;
    `;


    const values = [userID];
    const result = await pool.query(query, values);
    console.log(JSON.stringify(result.rows, null, 2));


    res.status(200).json({ success: true, bookmarks: result.rows });
  } catch (error) {
    console.error('Error fetching bookmarks:', error);
    res.status(500).json({ success: false, message: 'Error retrieving bookmarks' });
  }
});
```


 Opinionate


[Log Out](#) 


[Library](#) [Bookmarks](#) [Friends](#) [Settings](#)

 **Brian**
I love master of puppets


Star Wars
Movie

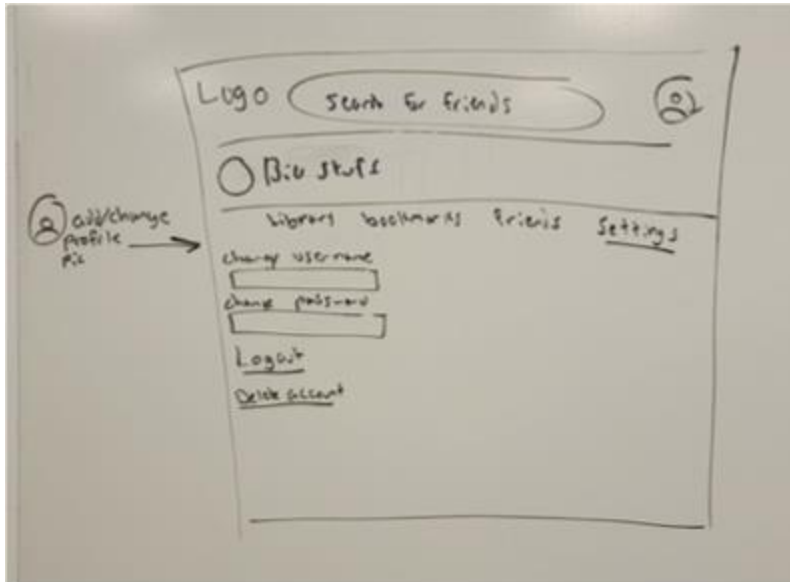

Back to the Future
Movie


Guardians of the Galaxy
Movie


Grand Theft Auto V
Video Game

SETTINGS/UPDATING PROFILE

Original Design



Final Design

The screenshot shows the 'Update profile settings' page in the Opinionate application. The page has a dark blue background. At the top, there is a header with the Opinionate logo, a search bar, and a 'Log Out' button. The main content area is a white card with the title 'Update profile settings'. Below the title, there is a 'Profile Picture' section with a circular profile picture of a man and a 'Choose File' button. Below the profile picture, there are input fields for 'Username' (containing 'Brian'), 'Current Password', 'New Password', and 'Confirm New Password'. Below these fields, there is an 'Email' field (containing 'someMaristEmail@gmail.com') and a 'Description' field (containing 'I love master of puppets'). At the bottom of the card, there is a 'Delete Account' button.

UPDATING PROFILE

```
// Update username
if (hasUsernameChanged) {
  try {
    const response = await fetch('/update-username', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ username })
    });
    const result = await response.json();
    if (result.success) {
      loadUserProfile();
      submitButton.setAttribute('data-original-username', username);
      updateSuccessful = true;
    } else {
      alert(result.message || 'An error occurred while updating the username.');
```

```
// this allows us to update the description. it may have to change once we go to add the others in
async function submitChanges(event) {
  event.preventDefault();

  const description = document.getElementById('description').value;
  const username = document.getElementById('username').value;
  const email = document.getElementById('email').value;
  const profilePictureInput = document.getElementById('profile-picture');
  const currentPassword = document.getElementById('current-password').value;
  const newPassword = document.getElementById('new-password').value;
  const confirmNewPassword = document.getElementById('confirm-new-password').value;

  const submitButton = document.getElementById('submit-changes');
  const successMessage = document.getElementById('update-success-message');

  const originalDescription = submitButton.getAttribute('data-original-description');
  const originalUsername = submitButton.getAttribute('data-original-username');
  const originalEmail = submitButton.getAttribute('data-original-email');

  const hasDescriptionChanged = description !== originalDescription;
  const hasUsernameChanged = username !== originalUsername;
  const hasEmailChanged = email !== originalEmail;
  const hasPasswordChanged = currentPassword && newPassword && (newPassword === confirmNewPassword);
  const hasProfilePictureChanged = profilePictureInput.files.length > 0;

  let updateSuccessful = false;
```

```
<div id="settings" class="section" style="display: none;">
  <div class="form-container" id="sign-in-form">
    <h2>Update profile settings</h2>

    <form oninput="showSubmitButton()">
```

```
<!-- Fields to Change User Settings -->
<div class="form-group">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" data-original="">
</div>
```