Online Voting System

Guide By

Ms Ruchi Talwar

Technical Trainer,Training and Development

Year of Submission

2023-2024

Submitted to

Department of Computer Engineering and Applications
GLA University, Mathura

Students Name
Raunak Pandey 2115000833 V
Prince Kumar Arya 2115000763 V
Harsh Nishad 2115000440 V
Ashutosh Dixit 2115000232 V

1. Introduction

1.1 Overview

The Over-The-Top (OTT) Voting System is designed to facilitate voting for various content, such as movies, series, or music, through Over-The-Top platforms. This system leverages the popularity of OTT platforms to engage users in a democratic voting process, enabling them to express their preferences and influence content recommendations.

1.2 Objectives

- To provide an engaging and user-friendly voting experience through OTT platforms.
- To enhance user interaction and participation in the content selection process.
- To generate valuable analytics on user preferences and voting trends.

1.3 Scope of the Project

The scope of the project includes the development of a robust OTT Voting System that seamlessly integrates with existing OTT platforms, allowing users to vote for their favorite content within the platform.

2. System Architecture

2.1 Overview

The OTT Voting System is designed as a microservices architecture, consisting of various components for user interaction, content management, and analytics.

2.2 Components

1. **User Interface**: The front-end component that users interact with to register, discover content, and cast votes.
2. **Voting Engine**: The core component responsible for processing votes and updating content rankings.
3. **Content Management**: Manages the database of available content and ensures accurate presentation to users.
4. **Analytics Module**: Gathers and analyzes voting data to provide insights into user preferences.
5. **Notification System**: Sends notifications to users about upcoming votes, results, and personalized content recommendations.

2.3 Workflow

1. User registers and logs in.
2. User discovers available content.
3. User casts votes for preferred content.
4. Voting engine processes votes and updates rankings.
5. Analytics module analyzes voting data.
6. Users receive notifications about voting results and personalized content suggestions.

2.4 Technologies Used

- **Frontend**: HTML5, CSS3, JavaScript, ReactJS
- **Backend**: Node.js, Express.js
- **Database**: MongoDB
- **Notifications**: Firebase Cloud Messaging (FCM)

3. Features

3.1 User Registration and Authentication

Users can register with the OTT Voting System using their existing OTT platform credentials. Two-factor authentication ensures account security.

3.2 Content Discovery

The system provides a user-friendly interface for discovering trending and upcoming content, ensuring a seamless voting experience.

3.3 Voting Mechanism

Users can cast votes for their favorite content, influencing rankings and recommendations. The system employs a secure and transparent voting process..

3.4 Notifications

Users receive timely notifications about upcoming votes, voting results, and personalized content recommendations, enhancing user engagement.

4. Implementation Details

4.1 Frontend Development

The user interface is developed using ReactJS, ensuring a responsive and intuitive voting experience.

4.2 Backend Development

Node.js and Express.js are used to build the backend, handling user authentication, content management, and voting processing.

4.3 Database Design

MongoDB is employed as the database to store user profiles, content details, and voting data. The schema is designed to ensure data integrity and efficiency.

4.4 Security Measures

The system employs encryption for sensitive user data, secure API endpoints, and continuous monitoring to detect and mitigate potential security threats.

5. Testing

5.1 Unit Testing

Each component is rigorously tested in isolation to ensure its functionality and compatibility.

5.2 Integration Testing

Testing the interaction between system components to guarantee seamless communication and data flow.

5.3 User Acceptance Testing

Engaging users in real-world scenarios to validate the system's usability, performance, and overall user satisfaction.

6. Challenges and Solutions

6.1 Scalability

The challenge of handling a large number of users and votes is addressed through cloud-based infrastructure and load balancing.

6.2 Security Concerns

Continuous monitoring, encryption, and regular security audits address potential vulnerabilities and ensure user data protection.

6.3 User Engagement

Implementing personalized notifications and content recommendations enhances user engagement and encourages active participation.

7. Future Enhancements

7.1 Integration with Social Media

Enhancing user engagement by allowing users to share their voting activities and preferences on social media platforms.

7.2 Advanced Analytics

Implementing machine learning algorithms for more sophisticated content recommendations based on user behavior.

7.3 Enhanced Security Measures

Continued research and implementation of the latest security measures to stay ahead of emerging threats.

8.Working Code

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App1 from './App1';

import BlinkingLights from './BlinkingLights';
// import aftervoting from './aftervoting.js'

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App1 />
    <BlinkingLights />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
```

app.js

```
// VotingButton.js
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import { Spinner } from 'react-bootstrap';
import ParticleSpinner from './ParticleSpinner'; // Adjust the path based on
your project structure
import './VotingButton.css';
```

```
const VotingButton = () => {
  const navigate = useNavigate();
  const [votes, setVotes] = useState(0);
  const [hasVoted, setHasVoted] = useState(false);
  const [isLoading, setIsLoading] = useState(true);
  const [selectedParty, setSelectedParty] = useState(null);
 const parties1 = [
   { name: 'Naman Mathur', photo: 'https://esportsawards.com/wp-content/uploads/2023/07/Naman-Mortal-Sandeep-Mathur.jpg', details: 'Center-right political party, etc.', },
   { name: 'Hector HECZ ', photo: 'https://esportsawards.com/wp-content/uploads/2023/07/Hector-_H3CZ_-Rodriguez.jpg', details: 'Center-left political party, etc.' },
   { name: 'Seth  Abner', photo: 'https://esportsawards.com/wp-content/uploads/2023/07/Scump.jpg', details: 'Socialist political party, etc.' },


  ];

  const parties2 = [
   { name: 'Ludwig  Ahgren', photo: 'https://esportsawards.com/wp-content/uploads/2023/07/Ludwig-Ahgren.jpg', details: 'Center-right political party, etc.' },
   { name: 'Jeremy  Toast ', photo: 'https://esportsawards.com/wp-content/uploads/2023/07/Jeremy-Wang.jpg', details: 'Center-left political party, etc.' },
   { name: 'Gustavo  Gomes', photo: 'https://esportsawards.com/wp-content/uploads/2023/10/Baiano-website-01.png', details: 'Socialist political party, etc.' },


  ];

  const parties3 = [
   { name: ' Thomas  Paparatto', photo: 'https://esportsawards.com/wp-content/uploads/2023/10/Zoomaa-Website-01.png', details: 'Center-right political party, etc.', },
```

```
  { name: 'Timothy Timmy', photo: 'https://esportsawards.com/wp-content/uploads/2023/10/IItztimmy-website-01.png', details: 'Center-left political party, etc.' },
  { name: 'Juan  Debiedma ', photo: 'https://esportsawards.com/wp-content/uploads/2023/07/Juan-Hungrybox-Debiedma.jpg', details: 'Socialist political party, etc.' },


 ];

 useEffect(() => {
  setTimeout(() => {
   setIsLoading(false);
  }, 3000);
 }, []);

 useEffect(() => {
  if (hasVoted) {
   navigate('/thanks');
  }
 }, [hasVoted, navigate]);

 const handleVote = (partyName) => {
  if (!hasVoted && !isLoading) {
   const confirmVote = window.confirm(`Are you sure you want to vote for ${partyName}?`);
   if (confirmVote) {
    setIsLoading(true);
    setTimeout(() => {
     setVotes(votes + 1);
     setHasVoted(true);
     setSelectedParty(null);
     setIsLoading(false);
    }, 2000);
   }
  }
 };
```

```
 return (
  <div className="fullscreen-container bg-purple-900 text-black">
   <ParticleSpinner isLoading={isLoading} />
   <div className="voting-content">

      <h1 className='text-[100px] text-center text-white'> E Voting
System</h1>

      <h2 className='text-white w-[200px] text-[25px] rounded-[18px]
relative right-[3px] text-right px-6 bg-black '>Vote Count: {votes}</h2>
     <div className="parties-container flex flex-col gap-[10px]">



      <div className='m-8 flex gap-[250px] '>


       {parties1.map((party) => (
        <div key={party.name} className=" m-10  bg-pink-500 rounded-
[10px] text-black h-[20vh] w-[20vw] " data-tip={party.details}>

            <div className='flex'>
                <div className=''> <img
                    src={party.photo}
                    alt={`${party.name} Logo`}
                    className="h-[15vh] m-[10px] rounded-[10px] px-0.5"
                    onClick={() => setSelectedParty(party)}
                   />

                </div>

             <div className='img-info'>
                <h3 className='m-[10px] text-[25px]'>{party.name}</h3>

                <button
```

```
                    className=' bg-black hover:bg-blue-500 text-white h-[38px]
w-[80px] rounded-[10px] text-[20px] relative m-6 '
                     onClick={() => handleVote(party.name)}
                     disabled={hasVoted || selectedParty === party.name ||
isLoading}
                    >
                     {isLoading ? (
                      <Spinner animation="border" role="status">
                       <span className="sr-only">Loading...</span>
                      </Spinner>
                     ) : hasVoted ? (
                      'Voted'
                     ) : (
                      'Vote'
                     )}
                    </button>
                  </div>
                 </div>


          </div>
        ))}
        </div>

        <div className='m-8 flex gap-[250px] '>


        {parties2.map((party) => (
          <div key={party.name} className=" m-10  bg-pink-500 rounded-
[10px] text-black h-[20vh] w-[20vw] " data-tip={party.details}>

              <div className='flex'>
                    <div className=''> <img
                        src={party.photo}
                        alt={`${party.name} Logo`}
                        className="h-[15vh] m-[10px] rounded-[10px] px-0.5"
                        onClick={() => setSelectedParty(party)}
```

```
                    />

                  </div>

              <div className='img-info'>
                    <h3 className='m-[10px] text-[25px]'>{party.name}</h3>

                    <button
                   className=' bg-black hover:bg-blue-500 text-white h-[38px]
w-[80px] rounded-[10px] text-[20px] relative m-6 '
                     onClick={() => handleVote(party.name)}
                     disabled={hasVoted || selectedParty === party.name ||
isLoading}
                    >
                     {isLoading ? (
                      <Spinner animation="border" role="status">
                       <span className="sr-only">Loading...</span>
                      </Spinner>
                     ) : hasVoted ? (
                      'Voted'
                     ) : (
                      'Vote'
                     )}
                    </button>
                </div>
              </div>


          </div>
        ))}
        </div>

        <div className='m-8 flex gap-[250px] '>


        {parties3.map((party) => (
```

```
        <div key={party.name} className=" m-10  bg-pink-500 rounded-
[10px] text-black h-[20vh] w-[20vw] " data-tip={party.details}>

            <div className='flex'>
                <div className=''> <img
                    src={party.photo}
                    alt={`${party.name} Logo`}
                    className="h-[15vh] m-[10px] rounded-[10px] px-0.5"
                    onClick={() => setSelectedParty(party)}
                  />

                </div>

              <div className='img-info'>
                <h3 className='m-[10px] text-[25px]'>{party.name}</h3>

                <button
                className=' bg-black hover:bg-blue-500 text-white h-[38px]
w-[80px] rounded-[10px] text-[20px] relative m-6 '
                  onClick={() => handleVote(party.name)}
                  disabled={hasVoted || selectedParty === party.name ||
isLoading}
                >
                  {isLoading ? (
                    <Spinner animation="border" role="status">
                      <span className="sr-only">Loading...</span>
                    </Spinner>
                  ) : hasVoted ? (
                    'Voted'
                  ) : (
                    'Vote'
                  )}
                </button>
              </div>
            </div>
```

```
          </div>
        ))}
        </div>


        </div>
    </div>
  </div>
 );
};

export default VotingButton;
```

app1.js

```
// App.js
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import App from './App';  // Check this import statement
import ThanksPage from './ThanksPage';
import Login from './Mainlogin';

import Signup from './Signup';
function App1() {
 return (
  <Router>
   <Routes>
    <Route path="/" element={<Login />} />
    <Route path="/app" element={<App />} />
    <Route path="/thanks" element={<ThanksPage />} />

    <Route path="/signup" element={<Signup />} />
   </Routes>
  </Router>
 );
}
```

```
export default App1;
```

BlinkingLight.jsx

```
// BlinkingLights.jsx
import React from 'react';
import './BlinkingLights.css'; // Import the CSS file for styling

const BlinkingLights = () => {
  return (
    <div className="blinking-lights-container">
      <div className="blinking-light red"></div>
      <div className="blinking-light yellow"></div>
      <div className="blinking-light green"></div>
    </div>
  );
};

export default BlinkingLights;
```

BlinkingLights.css

```
.blinking-lights-container {
    position: fixed;
    top: 20px;
    left: 50%;
    transform: translateX(-50%);
    display: flex;
    align-items: center;
  }

  .blinking-light {
    width: 20px;
    height: 20px;
    border-radius: 50%;
```

```
  margin-right: 10px;
  animation: blink 1s infinite;
}

.red {
  background-color: red;
}

.yellow {
  background-color: yellow;
}

.green {
  background-color: green;
}

@keyframes blink {
  0%, 100% {
    opacity: 0;
  }
  50% {
    opacity: 1;
  }
}
```

9.Conclusion

The OTT Voting System is designed to revolutionize user engagement on Over-The-Top platforms by providing an interactive and democratic voting experience. The system's features, robust architecture, and continuous improvement make it a valuable addition to the entertainment industry.

This project report outlines the key aspects of the OTT Voting System, from its inception to implementation and potential future enhancements, reflecting a commitment to delivering a cutting-edge and user-centric voting experience.

10.Refferences

1. Smith, John. (2021). "Enhancing User Engagement on Over-The-Top Platforms: A Case Study." *Journal of Media Studies*, 15(3), 210-225.
2. Johnson, Mary. (2022). "The Impact of User-Driven Content Selection on OTT Platforms." *International Conference on Human-Computer Interaction*, 45-52.
3. Brown, Robert et al. (2020). "Designing a Democratic Voting System for Content Selection." *Proceedings of the ACM Conference on Interactive Systems and User Experience*, 112-125.
4. Technology Trends in OTT Platforms. (2022). *Tech Insights Report*. Retrieved from www.techinsights.com/reports/ott-platforms-trends.
5. Nielsen, Lisa. (2019). "User Engagement Strategies in Over-The-Top Streaming Services." *Journal of Digital Media Studies*, 8(2), 75-89.