DOI 10.31891/2307-5732-2024-343-6-17
УДК 004

KOSTYUCHENKO ARTEM
Zhytomyr Polytechnic State University
https://orcid.org/0009-0000-7606-042X
e-mail: ipzm241_kav@student.ztu.edu.ua
LOKTIKOVA TAMARA
Zhytomyr Polytechnic State University
https://orcid.org/0000-0002-3525-0179
e-mail: dfikt_ltn@ztu.edu.ua
KUSHNIR NADIA
Zhytomyr Polytechnic State University
https://orcid.org/0000-0002-0797-3687
e-mail: kipz_kno@ztu.edu.ua
LYSOGOR IURII
Zhytomyr Polytechnic State University
https://orcid.org/0000-0003-1194-2813
e-mail: lysogor@ztu.edu.ua

RESEARCH OF THE PRINCIPLES OF BUILDING AND DESIGNING A PLATFORM FOR SHARING VIDEO CONTENT

A video content sharing platform can only function effectively if it is designed to meet the diverse needs of users and the specifics of video playback synchronization. The features of such platforms include, in particular, providing simultaneous access to content to several users located in different geographical locations with minimal delay. In addition, interactive user interaction via text and voice chats is important, creating a virtual space for sharing media content. These user needs form the main requirements for the technical solution, including high bandwidth of the data transmission channel, efficient request processing, integration of the interface with the ability to manage rooms and access rights to video playback. The article proposes the development of a platform for sharing video content, which is characterized by such features as continuous playback synchronization, integration of text and voice chat, and the ability to manage access rights to video content. When developing the platform, special attention was paid to the use of modern technologies, namely, the client-server architecture, JavaScript libraries Express and React together with the Vite tool, MongoDB database, as well as WebSocket and WebRTC protocols that minimize delays and ensure the smooth operation of the system were chosen to build and implement the system. The WebSocket protocol synchronizes the actions of all participants in the room, including video and messaging. The WebRTC protocol is used to organize voice communication between participants, which makes the interaction between users even more dynamic and realistic. As a result, an effective web application for organizing joint viewing of video content was created, which significantly increased the level of user interaction and provided maximum comfort during viewing, which was proven during various types of testing, including load testing using the JMeter tool.

Keywords: web platform, video content, interactivity, WebSocket, WebRTC.

КОСТЮЧЕНКО АРТЕМ, ЛОКТІКОВА ТАМАРА, КУШНІР НАДІЯ, ЛИСОГОР ЮРІЙ
Державний університет «Житомирська політехніка»

ДОСЛІДЖЕННЯ ПРИНЦИПІВ ПОБУДОВИ ТА ПРОЕКТУВАННЯ ПЛАТФОРМИ ДЛЯ СПІЛЬНОГО ПЕРЕГЛЯДУ ВІДЕОКОНТЕНТУ

Ефективне функціонування платформи для обміну відеоконтентом можливе лише за умови врахування при її розробці різноманітних потреб користувачів та специфіки синхронізації відтворення відео. До особливостей таких платформ, зокрема, належить надання одночасного доступу до контенту кільком користувачам, які знаходяться в різних географічних точках, з мінімальною затримкою. Крім того, важливою є інтерактивна взаємодія користувачів за допомогою текстових і голосових чатів, що створює віртуальний простір для обміну медіаконтентом. Ці потреби користувачів формують основні вимоги до технічного рішення, включаючи високу пропускну здатність каналу передачі даних, ефективну обробку запитів, інтеграцію інтерфейсу з можливістю управління кімнатами та правами доступу до відтворення відео. У статті пропонується розробка платформи для спільного перегляду відеоконтенту, яка характеризується такими можливостями як безперервна синхронізація відтворення, інтеграція текстового та голосового чату, а також можливість управління правами доступу до відеоконтенту. При розробці платформи особливу увагу було приділено використанню сучасних технологій, а саме для побудови та реалізації системи було обрано клієнт-серверну архітектуру, JavaScript-бібліотеки Express та React спільно з інструментом Vite, СУБД MongoDB, а також протоколи WebSocket та WebRTC, які мінімізують затримки та забезпечують безперебійне функціонування системи. Протокол WebSocket синхронізує дії всіх учасників кімнати, включаючи обмін відео та повідомленнями. Протокол WebRTC використовується для організації голосового зв'язку між учасниками, що робить взаємодію між користувачами ще більш динамічною та реалістичною. У результаті створено ефективний веб-застосунок для організації спільного перегляду відеоконтенту, що значно підвищив рівень взаємодії користувачів та забезпечив максимальний комфорт під час перегляду, що було доведено під час проведеного тестування різного виду, включаючи навантажувальне за допомогою інструменту JMeter.

Ключові слова: веб-платформа, відеоконтент, інтерактивність, WebSocket, WebRTC.

Problem statement

Modern digital platforms play a key role in ensuring communication, consuming entertainment content, and creating social connections. Video sharing has become a popular element of social interaction, which is actively developing due to the rapid growth of online services such as streaming platforms and video conferencing services.

One of the barriers to this trend is the problem of synchronizing the playback of video content for users in different geographical and technical conditions. Differences in Internet connection speeds, delays in data transmission, and different technical capabilities of devices lead to uneven playback, which affects the quality of shared viewing.

There are already platforms that allow users to watch video content together, but these platforms do not sufficiently integrate tools for active interaction among participants. Limitations in the forms of communication, such as insufficient integration of chats and other real-time responses, make it difficult to create a fully interactive environment.

A significant part of the challenges associated with the development of video content sharing platforms relates to ensuring stable and continuous interaction between users in a global network. The issues of system scalability and its ability to support a large number of simultaneous connections without compromising the quality of service remain relevant.

The problems of personalizing interaction and managing access rights also pose a significant challenge that remains insufficiently addressed by modern platforms. The need to accommodate different types of audiences and user scenarios emphasizes the importance of developing flexible architectures that allow users to tailor the interaction process to their needs while ensuring stable content synchronization for all participants.

Analysis of research and publications

Among the most well-known similar platforms are Netflix Party, Watch2Gether, and Scener. Each of them offers unique features and functionality, but also has limitations. Reviewing and analyzing these platforms will help identify their advantages and disadvantages, as well as the needs of users who remain unsatisfied with existing solutions, which, in turn, will create the basis for developing a new platform that meets modern requirements and user expectations.

Netflix Party [1], more recently known as Teleparty, is one of the first platforms that allows users to watch content on Netflix together. Launched in 2020, it quickly gained popularity in the context of the global increase in requests for remote viewing of movies and TV shows during the COVID-19 pandemic. Teleparty provides the ability to synchronize video playback between users, allowing all participants to watch the same content at the same time, regardless of geographic location.

However, despite its popularity, Teleparty has limitations. First, it only works with Netflix content, which limits the choice of video content to watch. Secondly, the platform does not offer interactivity features such as video chat or other elements that would facilitate deeper social interaction between users.

Watch2Gether [2] is an online platform that allows users to watch videos from various sources, such as YouTube, Vimeo, Dailymotion, and others, together. Launched in 2010, the platform is notable for its ease of use and versatility, as it is not limited to just one video streaming service. Users can create private or public viewing rooms, invite friends, and share login links, making it convenient for organizing joint viewing.

Despite its many advantages, Watch2Gether has limitations. First, the platform does not support all streaming services, which can limit the choice of content for users. Although it offers the ability to watch from multiple sources, some popular services, such as Netflix or Disney+, are not integrated into the system. Secondly, the platform does not provide the ability to watch movies or TV series, as this requires appropriate licenses, which may reduce interest in using the platform.

Scener [3] is an online platform that allows users to watch movies and TV shows together in real time. Launched in 2019, Scener offers a unique experience by combining elements of social viewing with interactive features. Users can create “virtual rooms” for viewing by inviting friends to participate through links or social networks.

One of the main advantages of Scener is the ability to integrate with popular streaming services such as Netflix, Hulu, and Disney+, which makes Scener a convenient choice for sharing current movies and TV shows. The platform also offers integrated video chat, which allows users to communicate in real time, increasing the level of social interaction while watching.

However, despite its advantages, Scener has certain limitations. First, the platform depends on the availability of subscriptions to the relevant streaming services, which may limit access to content for some users. Secondly, integration with video chat can create technical problems, such as delays or deterioration in image quality, which can affect the overall viewing experience.

After analyzing existing platforms, we can formulate a number of features that need to be implemented in the newly created platform to meet the needs of users that were not fully covered by existing solutions:

1. Introduce a subscription model to provide access to a wide range of video content. This approach will allow users to watch a variety of video content without having to search for it on different platforms.
2. Integration of text and voice chats to increase the level of social interaction while watching.
3. Playback access control functionality. Giving users the ability to control access rights for other participants will create a more secure and manageable viewing environment.
4. Ensure minimal delay in data transmission.
5. Ability to change video content for viewing without the need to recreate the room.

Formulating the objectives of the article

The purpose of the study is to investigate the development of a platform for sharing video content that will meet modern user requirements, taking into account the limitations of existing solutions.

Summary of the main material

The first stage of any platform development is the choice of architecture, as this choice determines the overall efficiency, reliability, and scalability of the future software product. The choice of an architectural solution becomes key to ensuring optimal resource utilization and maintaining stable system operation under high loads.

Among the available architectural approaches, the client-server architecture [4] deserves special attention, as it is one of the most common solutions in the context of web application development. It provides a clear division of functions between the client and server parts, which allows for a high level of flexibility and scalability. In addition, this architectural approach supports modularity and simplifies the process of deploying and maintaining the system.

A scientific analysis of various architectural approaches shows that the client-server model provides efficient distribution of computing loads, supports various data transfer protocols, and reduces delays in the exchange of information between clients and the server. Based on this, the client-server architecture was chosen to build the platform.

The development of the system's functional structure is the next stage of software development. At this stage, the system is detailed by identifying the main functional elements and defining the interaction between them in accordance with the project requirements.

Fig. 1 shows a diagram of the options for using the developed platform with three user roles: User, Room Owner and Administrator.

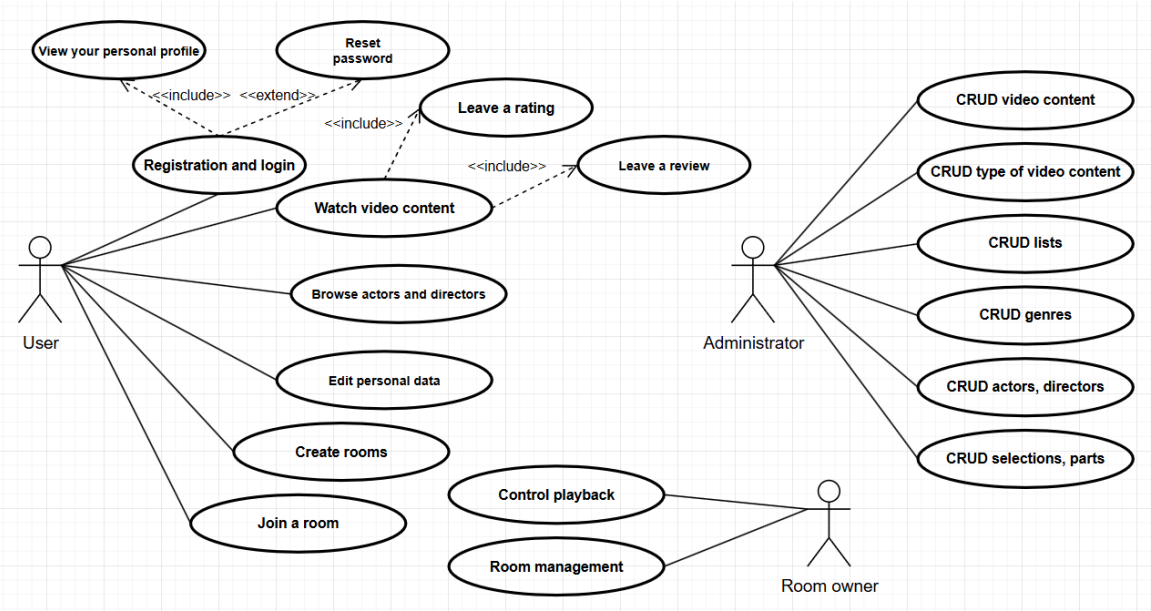

Fig. 1. Diagram of options for using the platform

After determining the use cases and functional interaction of users with the system, the internal structure of the server part is designed. The server part of the system is implemented as a REST API application [5].

As a description of the structure of the server part, a class diagram was built, which clearly displays the key components of the system and their interconnections.

Figure 2 shows a diagram of the classes of the server side of the platform.

One of the most important decisions in the architecture of any modern software system is the choice of a database management system (DBMS). The efficiency of data storage, retrieval, and processing, and thus the overall system performance, depends on its successful choice. For the proposed platform, we have chosen a document-oriented DBMS - MongoDB [6]. This choice was made due to a number of advantages of this system, including high flexibility of the data schema, horizontal scalability, and the ability to efficiently process large amounts of unstructured data.

The platform's database has a modular structure consisting of 15 separate collections, each of which is responsible for storing a certain type of data:

1. Actor - a collection that stores information about actors, their biographies and filmographies.
2. VideoContent - a collection that contains data about video content.
3. Country - a collection that stores information about countries.
4. Director - a collection containing information about directors and their works.
5. Genre - a collection that stores data on the genres of video content, which allows you to classify materials by topic.
6. List - a collection containing information about lists of video content.
7. Message - a collection that stores messages that are sent within rooms, providing communication between users.
8. Part - a collection containing data about parts of video content.
9. Rating - a collection that stores information about the ratings of video content rated by users.

10. Review - a collection containing user reviews.
11. Room - a collection that stores information about the rooms in which video content is viewed.
12. Selection - a collection that contains data on content selections that can be recommended to users.
13. Token - a collection that stores information about tokens for password recovery.
14. TypeContent - a collection that contains information about the types of video content, which allows you to classify it by format.
15. User - a collection that stores data about system users.

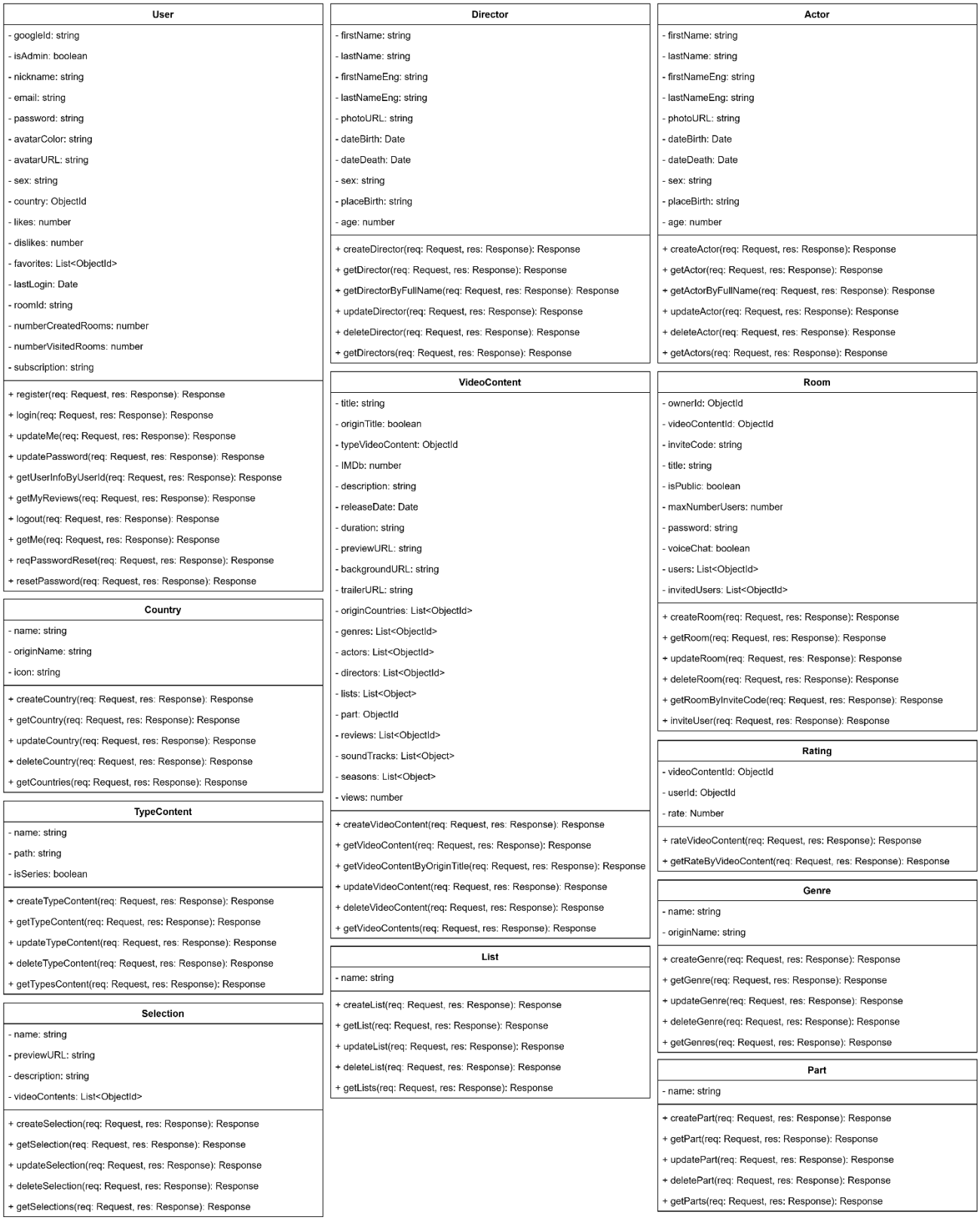

Fig. 2. Diagram of platform classes

Real-time user interaction is based on the use of WebSocket [7] and WebRTC [8] protocols. The WebSocket protocol provides efficient two-way communication between clients, allowing for instant exchange of text messages and metadata. To simplify development and increase reliability, the system uses the Socket.IO library [9], which provides an abstraction over the WebSocket protocol.

The WebSocket protocol is also used to organize synchronous playback of video content. The server uses WebSocket to transmit updates on the playback status (play, pause, search) to all connected clients, thus ensuring consistent playback for all session participants.

The WebRTC protocol is used to implement voice chat and video conferencing. This protocol allows you to establish peer-to-peer connections between clients, providing low latency and high quality audio and video data transmission. The use of the WebRTC and WebSocket protocols is a comprehensive solution for the implementation of interactive functions in the system.

The next stage of the platform development was the implementation of the client side, which provides direct user interaction with the system. The key element of the client part is a user interface designed to visualize data received from the server part and provide intuitive control of the system's functionality.

The technology stack of the platform's client side is based on the popular JavaScript library React [10]. The component-based approach of React.js ensures modularity and code reuse, which contributes to the creation of scalable and efficient interfaces. Reactive updating of component state ensures smooth and intuitive user interaction with the system.

The Vite tool [11] was chosen to optimize the development process and improve application performance.

The final stage of the platform development was the deployment of its components on cloud platforms.

The client side is hosted on the Netlify platform [12]. This platform was chosen due to its powerful capabilities for deploying static websites and web applications. Automation of deployment processes, integration with version control systems, and support for CI/CD processes ensured rapid release of updates and high reliability of the client part of the developed system.

The server part is deployed on the Vercel platform [13]. This platform, specializing in server applications, provided high performance and rapid scaling of computing resources. Optimization of the content release provided by Vercel had a positive impact on the overall user experience.

An additional stage in the development of the web platform was comprehensive performance testing, which included load testing using the Apache JMeter tool [14], which allowed us to determine the platform's ability to handle different levels of user load, assess its resistance to high peak loads, and identify potential bottlenecks in the system architecture.

Fig. 3 shows the interface of the main page of the developed platform. This page serves as an entry point for authorized users, providing access to the main functions and capabilities of the system. The interface is designed taking into account the principles of usability [15], which allows users to easily navigate and perform the necessary actions without any difficulties.

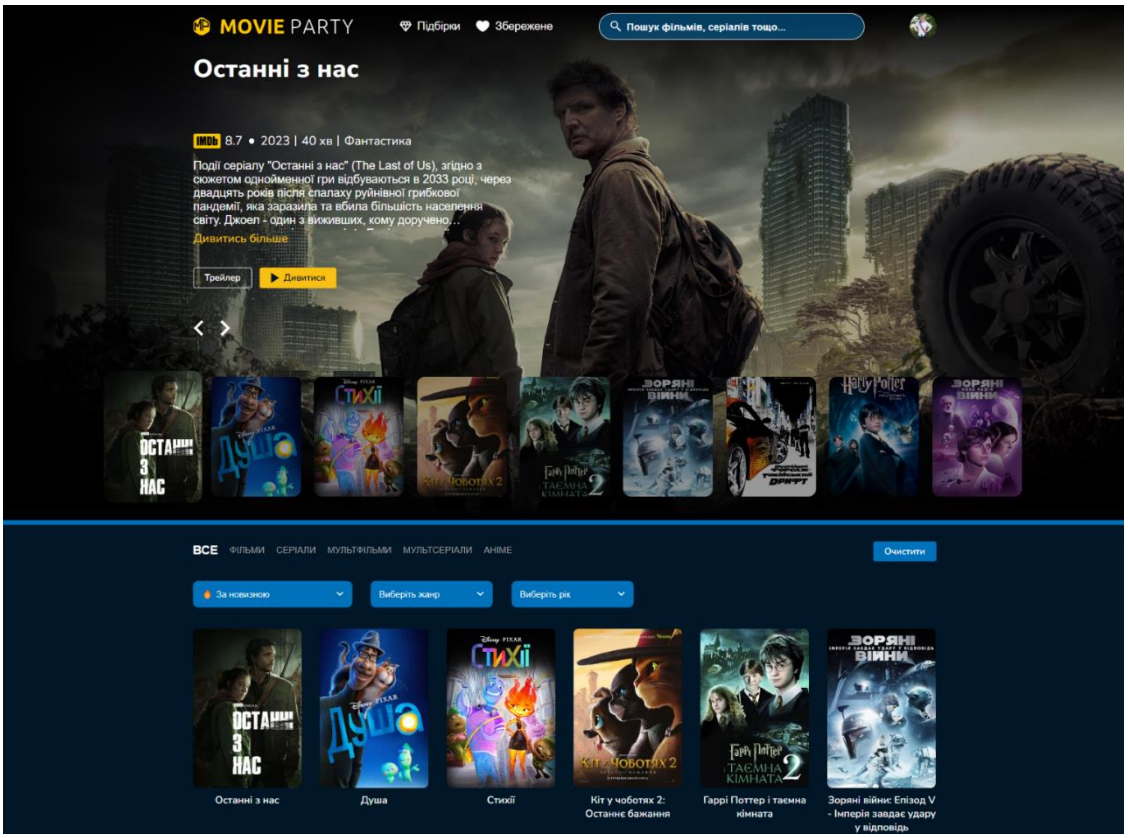

Fig. 3. The main page of the platform

Fig. 4 shows the interface of a virtual room designed to ensure effective communication between users while watching video content together. The room interface includes text and voice chat functionality that allows participants to exchange messages in real time.

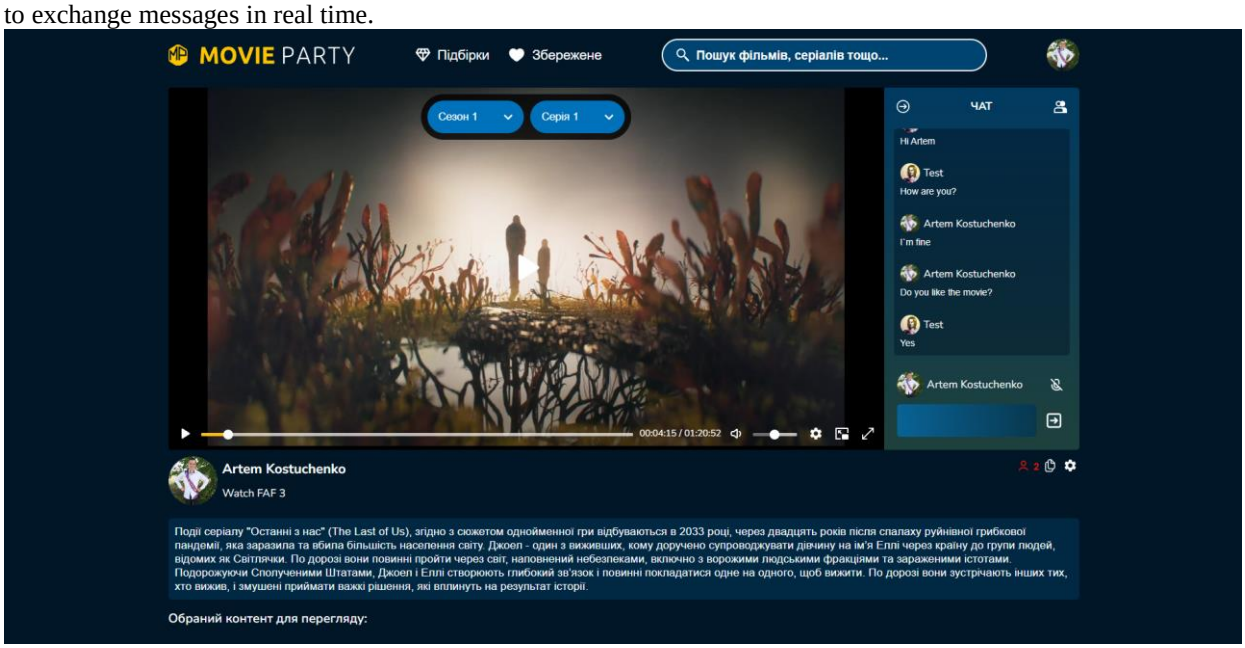

Fig. 4. The virtual room page

Conclusions

The study examined the principles of building and designing a video content sharing platform that takes into account modern requirements for interactivity, flexibility, and scalability. The analyzed existing similar solutions - Netflix Party, Watch2Gether, and Scener - showed their limitations in the ability to synchronize content, communicate between users, and control access to playback functions. The proposed platform solves these problems by introducing innovations such as the use of WebSocket and WebRTC protocols to provide synchronization and voice chat, respectively, which ensured minimal delays in user interaction.

The use of the MongoDB database with a flexible collection structure allowed us to optimize the storage of large amounts of data and ensure the efficient operation of the system. The implementation of the client side based on the React JavaScript library using the Vite tool and the deployment of the platform on Netlify and Vercel cloud services ensured high performance and reliability of the system.

Prospects for further research in this area will focus on expanding the platform's functionality by integrating new means of communication, such as video chats, and introducing artificial intelligence to personalize content according to user preferences. Another important area of focus will be optimizing the system to handle even higher loads, which will allow for simultaneous interaction of thousands of users while maintaining high quality of the stream.

References

1. Netflix Party (Teleparty) [Electronic resource] – Resource access mode: https://www.teleparty.com/.
2. Watch2Gether Product Documentation [Electronic resource] – Resource access mode: https://community.w2g.tv/t/how-to-use-watch2gether/736.
3. Scener [Electronic resource] – Resource access mode: https://www.scener.com/.
4. Client Server Architecture A Complete Guide - 2020 Edition, 2020. – 218 p.
5. Amundsen M. RESTful Web APIs: Services for a Changing World / Amundsen., 2013. – 406 p.
6. Banker K. MongoDB in action / Kyle Banker. – New York, 2011. – 312 p.
7. Lombardi A. WebSocket: Lightweight Client-Server Communications / Andrew Lombardi., 2015. – 141 p.
8. Ristic D. Learning WebRTC / Dan Ristic., 2015. – 186 c.
9. Sidelnikov G. React.js Book: Learning React JavaScript Library From Scratch / Greg Sidelnikov., 2017. – 117 p.
10. Rai R. Socket.IO Real-Time Web Application Development / Rohit Rai., 2013. – 140 p.
11. Vite | Next Generation Frontend Tooling [Electronic resource] – Resource access mode: https://vite.dev/.
12. Vercel: Build and deploy the best web experiences with the Frontend Cloud [Electronic resource] – Resource access mode: https://vercel.com/.
13. Netlify: Scale & Ship Faster with a Composable Web Architecture [Electronic resource] – Resource access mode: https://www.netlify.com/.
14. Erinle B. Performance Testing with JMeter 3 / Bayo Erinle., 2017. – 166 p.
15. Tidwell J. Designing Interfaces: Patterns for Effective Interaction Design / J. Tidwell, C. Brewer, A. Valencia., 2020. – 599 p.