jamk

Moderation panel for the virtual event platform as a micro frontend module

Samson Azizyan

Bachelor's thesis
May 2022
Information and Communication Technology
Degree Programme in Information and Communication Technology

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Description

Azizyan, Samson

Moderation panel for the virtual event platform as a micro frontend module

Jyväskylä: JAMK University of Applied Sciences, May 2022, 35 pages

Information and Communication Technologies (ICT). Bachelor's thesis.

Permission for open access publication: Yes

Language of publication: English

Abstract

The thesis assignment was commissioned by Liveto Group Oy. Liveto specializes in providing tools for event organizers. Some of the main products offered by Liveto include for example a virtual event platform for hosting various virtual events and an online store that can be easily integrated into any web page.

The objective of the thesis was to develop a scalable, comprehensive moderation panel as a micro frontend module for the existing virtual event platform using React framework originally developed by Facebook to facilitate frontend development.

The thesis was carried out as a research implementation project, including multi-method research and software product development. The process began with constructing a comprehensive knowledge base by describing the background of the client company and discussing in depth how the Liveto virtual event platform works. The next phase was the research and examination of various micro frontend frameworks and moderation technologies. Following the research, conclusions were drawn, and most suitable micro frontend framework and moderation features were selected. Lastly, the implementation of the software started with planning and visualising the software architecture, followed by the implementation of the moderation panel's interface and its logic.

As a result, a functional moderation panel was developed as a part of the existing virtual event platform. All the initially planned features were implemented correctly, and the interface styling followed Liveto's guidelines. As the deadline for this project was set earlier than the Liveto's micro frontend framework migration can be executed, the moderation panel could not be developed as a micro frontend module. However, it was implemented as a future-proof and easily detachable component. Further development is required before the implemented software can go into production, as the interface lacks responsive styles for the mobile and tablet devices.

Keywords/tags (subjects)

Web application, Web programming, frontend development, React, programming

Miscellaneous (Confidential information)

jamk

Description

Azizyan, Samson

Moderointipaneeli virtuaalitapahtuma-alustalle ”micro frontend” - moduulina

Jyväskylä: Jyväskylän ammattikorkeakoulu. Toukokuu 2022, 35 sivua

Tietojenkäsittely ja tietoliikenne. Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Verkkojulkaisulupa myönnetty: Kyllä

Julkaisun kieli: englanti

Tiivistelmä

Opinnäytetyön tilaajana toimii Liveto Group Oy, Liveto tarjoa monipuolisia työkaluja erilaisten tapahtumien järjestämistä varten. Liveton päätuotteisiin kuuluu esimerkiksi virtuaalitapahtuma-alusta erilaisten virtuaalitapahtumien järjestämiseen sekä verkkokauppa, joka on helposti integroitavissa mille tahansa verkkosivulle.

Opinnäytetyön tavoitteena oli kehittää skaalautuva ja kattava moderointipaneeli ”micro frontend”-moduulina olemassa olevaan virtuaalitapahtuma-alustaan käyttämällä Facebookin kehittämää React-ympäristöä.

Opinnäytetyö oli toteutettu tutkimuksellisen kehittämistyönä, joka sisälsi monimenetelmätutkimuksen ja ohjelmistotuotekehityksen. Opinnäytetyöprosessin ensimmäinen vaihe oli kerätä kattavan tietopohjan kuvailemalla asiakasyrityksen taustaa ja esittämällä Liveton virtuaalitapahtuma-alustan toimintaa. Seuraava vaihe oli erilaisten mikrokäyttöliittymäkehysten ja moderointiteknologioiden tutkimus. Tutkimuksen seurauksena, johtopäätökset ja valinta sopivimmasta ”micro frontend” - ympäristöstä ja moderointiominaisuuksista oli tehty. Lopuksi ohjelmiston kehitysprosessi alkoi ohjelmistoarkkitehtuurin suunnittelulla ja visualisoinnilla, jota seurasi moderointipaneelin käyttöliittymän ja sen logiikan toteutus.

Lopputuloksen saatiin toimiva moderointipaneeli, joka on kehitetty osaksi olemassa olevaa virtuaalitapahtuma-alustaa. Kaikki alun perin suunnitellut ominaisuudet toteutettiin oikein, käyttöliittymätyylit oli toteutettu Liveton ohjeiden mukaisesti. Tämän projektin takarajan ollessa aikaisempana ajankohtana kuin ajankohta jona Liveton mikrokäyttöliittymäkehyksen siirto voidaan suorittaa, moderointipaneelia ei voitu kehittää mikrokäyttöliittymämoduulina. Moderointipaneeli toteutettiin kuitenkin tulevaisuudenkestävänä ja helposti irrotettavana komponenttina. Lisäkehitystä tarvitaan ennen kuin toteutettu ohjelmisto on tuotantovalmis, käyttöliittymästä puuttuu responsiiviset tyylit mobiili- ja tablettilaitteille.

Avainsanat (asiasanat)

Web-sovellus, Web-ohjelmointi, käyttäliittymäohjelmointi, React, ohjelmointi

Muut tiedot (Salassa pidettävät liitteet)

Contents

Figures

Tables

List of Abbreviations and Acronyms

API:	Application programming interface, a connection between computers or between computer programs.
Asynchronous:	A form of parallel programming that allows a unit of work to run separately from the primary application thread.
Backend:	The server-side of the application
Callback:	A callback is a function passed as an argument to another function.
CSS:	Cascading Style Sheets.
Frontend:	The part of an application that the user interacts with directly
HTML:	Hypertext Markup Language.
JS:	JavaScript.
JSX:	JavaScript XML. JSX allows makes it possible to write HTML in React.
REST:	Representational state transfer, an architecture style for designing networked applications.
Runtime:	A stage of the programming lifecycle.
Sass:	Syntactically Awesome Style Sheets, an extension for CSS.
UI / UX:	User Interface / User Experience.
Webpack:	A tool that is used for compiling JavaScript modules.

1 Introduction

The COVID-19 pandemic spread globally in the first quarter of 2020, resulting in quarantines and isolations. As most physical events were cancelled and no new ones being on the horizon, companies whose business model heavily relied on organizing physical events were forced out of business or had to come up with another direction for the company. Jyväskylä based Liveto Group Oy was one of these companies.

1.1 Background of the Company

A Finnish software company, Liveto Group Oy, offered an all-in-one solution for event organizers to hold physical events. These events could range from single presentations to full festivals. Due to the lack of physical events, Liveto began developing a virtual event platform with the capability of hosting virtual and hybrid events. In addition to virtual event platform, some of the many main event organizing tools offered by Liveto are an online store, that can be easily integrated into any website, an event app that can be used to socialize with event attendees while being on event premises and a management platform, that offers most diverse set of event organizing tools.

1.2 Objective of The Thesis

Considering Liveto chat's versatility, it requires an efficient moderation panel, that can be run by as few as one moderator. The Moderation Panel must be developed as an independent module as part of the transition to the micro frontend architecture. The moderation panel will be the first module to be developed under the new architecture. The aim of this thesis is to create most efficient and user-friendly moderation panel as an independent micro frontend module.

1.3 Research Implementation

The purpose of this thesis is to implement a piece of software that can be used in production, using information from the research. The first core part of the thesis will be research, that consists of two phases: micro frontend framework and moderation. The second core part will be an implementation of the Liveto moderation panel, which is the first ever micro frontend module of the new modular Liveto frontend architecture.

2 Liveto Virtual Event Platform

Liveto virtual event platform (Figure 1) offers event organizers a set of tools that allow them to create the most personalized virtual event possible. These kinds of platforms got common due to pandemic, the main difference between the Liveto virtual event platform and the rest is customizability. The virtual event can be customized to look exactly like the event organizer wants. Customization is made possible with the virtual event platform editor software (Figure 2).

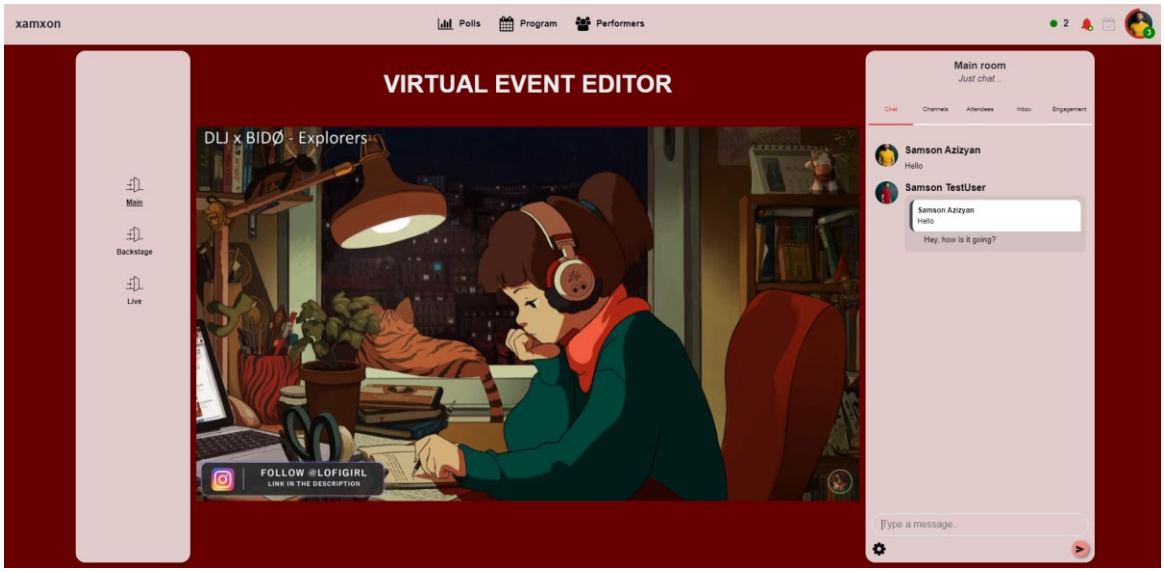

Figure 1. Liveto Virtual Event Platform main view

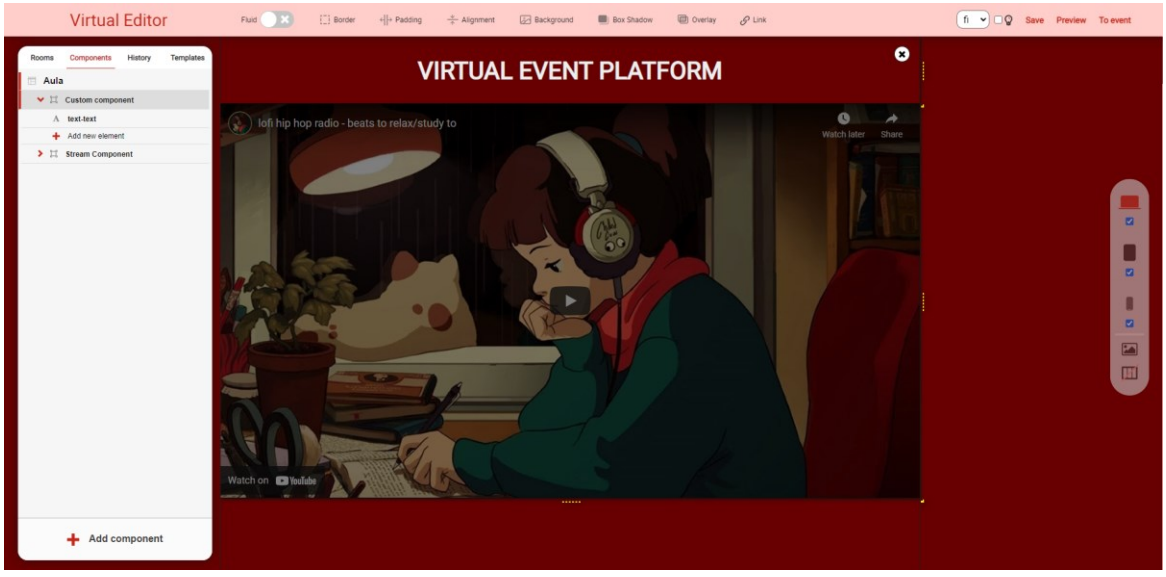

Figure 2. Virtual Event Editor

Chat, video calls, scheduling, event programs, polls, and engagement score are among the main features on the virtual event platform. The feature that is most relevant to this thesis is chat (Figure 3). The in-house developed chat is built to support unlimited amount of chat rooms and holds an unlimited number of participants.

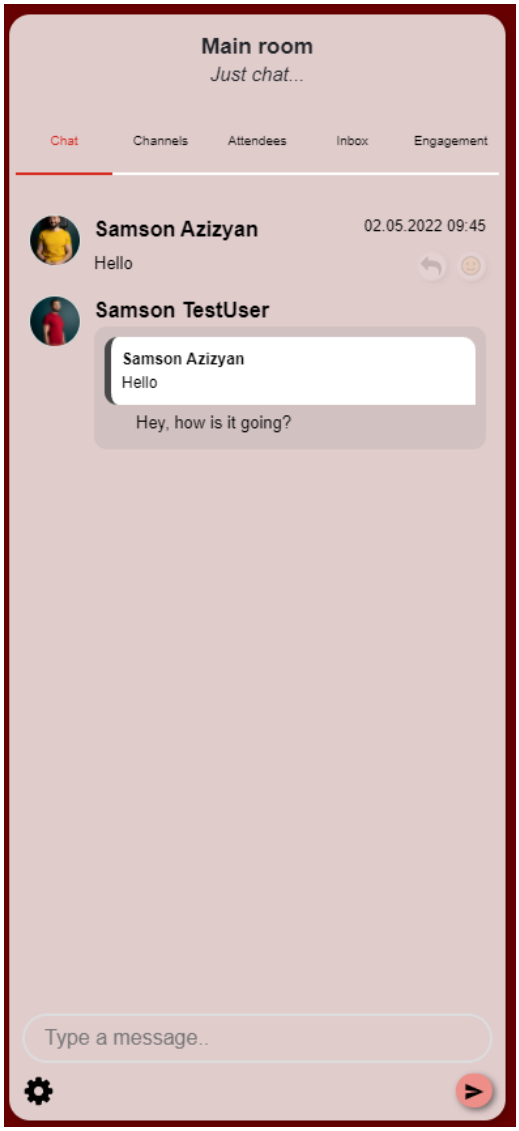

Figure 3. Virtual Event Platform Chat

2.1 Physical / In-person Event

A live event refers to any event in which the performer and the participants are both present in the same space at the same time physically. There are many types of in-person events, from con-

certs to theatre performances. A venue is always required for an in-person event, whether it is indoors or outdoors. There may be several areas or spaces in an in-person event, but the performers will be in the same area as the audience. (In-person Event.)

2.2 Virtual Event

The term 'virtual event' is used to describe a type of online event where people follow content, network, buy, sell, etc. via computers or mobile devices. Unlike in-person events, virtual events are built on a web-based platform with different functionalities that mimic in-person events. Attending a virtual event is easy as you can join it from anywhere and at any time, which in turn reduces the need to travel and saves attendees' time. Virtual events could be anything from seminars to concerts. (Virtual Event.)

2.3 Hybrid Event

Hybrid events combine the best features of virtual and in-person events. In hybrid events, all participants have the opportunity to participate, regardless of their location or method of participation. Event attendees have the option of attending live or online. Hybrid events offer attendees the option of following content, networking, and more in two different environments. (Hybrid Events.)

2.4 Remodelling of the Frontend Architecture

As of right now, Liveto's frontend software architecture consists of monolithic projects which hold a large amount of code and are hard to work on by multiple development teams simultaneously. It is Liveto's vision to separate monolithic projects into individual micro frontend projects in the future.

3 Micro Frontends and Moderation

As to which Micro Frontend framework will be used in the future frontend architecture is not yet decided, the first part of this research will focus on three micro frontend frameworks. These

frameworks are Bit and Webpack's Module Federation. After the framework is defined, the examples for live virtual event moderation will be examined, and conclusions will be drawn as to what set of features will be implemented.

3.1 Micro Frontends

As a monolithically developed project gets more complicated and the team that is working on it increases in size, it gets harder for every developer to know every part of this project. A change to one part of the software might affect or even break other parts. This increases the growth of the technical debt. To solve this problem, projects are broken up into microservices or Micro Frontends. Each Micro frontend can be run by a dedicated developer or a team of developers (Figure 4). These eventually are combined into one application that the end user sees through their browser. Another reason for the micro frontend architecture is reusability, the same module can be used in multiple projects provided it receives the data that it requires. (Geers 2020, Chapter 1.)

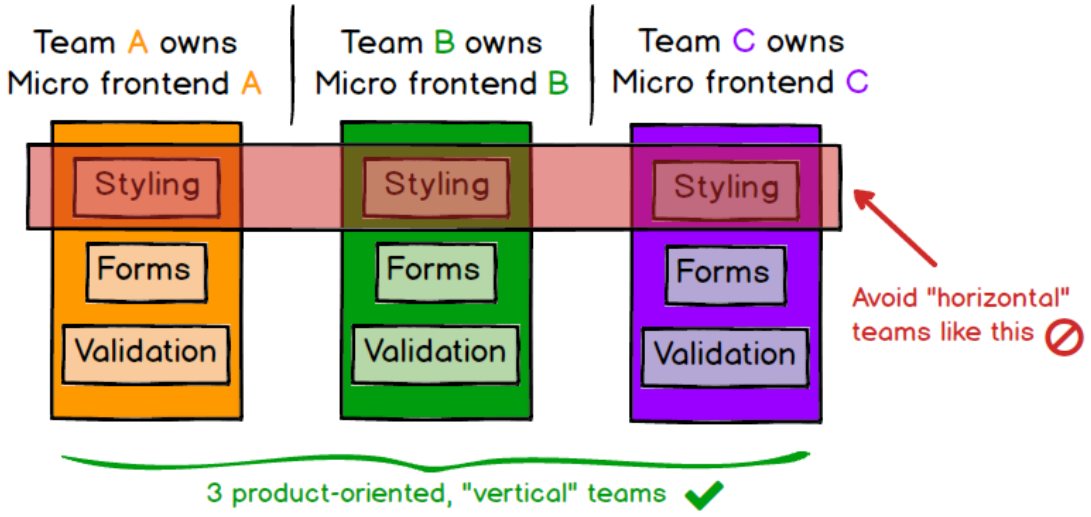

Figure 4. Simple micro frontend architecture (Jackson 2019)

3.1.1 Bit

Bit is a micro frontend framework that like many other similar frameworks lets you break you application down into smaller independent, and reusable components. Components are created and can be assembled. Those components can be of a different kind, apps, UI components, backend services. (What is Bit?)

Bit offers four membership tiers: free, pro, business, and enterprise. The first two are very limited, basically only framework is offered, the hosting and integrations will need to be done manually. Business and Enterprise tiers offer a large set of various tools, some of which are essential for the use of the Bit framework, for example team development, integrations, component registry, certifications. (Plans for teams.)

3.1.2 Module Federation

Module Federation is a plugin for Webpack, which is a static module bundler for modern JS applications. Module Federation's principle is a JavaScript code sharing in runtime. When a JavaScript application consumes a federated module, it simultaneously shares dependencies with that application, meaning if a dependency is missing from the federated code, Webpack will download it from the other application. This method makes the development of federated software seamless. Module Federation is completely free and easy to use.

3.1.3 Micro Frontend Framework Conclusion

There is a reason for Bit being the most popular Micro Frontend framework as it offers a large toolset for the smooth workflow. However, the toolset is only available with paid subscription plans. Module Federation being free and easy to use were the decisive factors in choosing it as a technology for the future of Liveto's frontend development.

3.2 Moderation

In this part of the thesis different examples of moderation will be examined. Key features and methods will be selected, most suitable of those will be implemented into Liveto's virtual event platform's moderation panel.

3.2.1 YouTube

YouTube is an online video sharing platform for anyone to host videos in, YouTube users are also able to stream live videos with live chat being present. The live chat offers a moderation platform with various moderation tools, relevant tools are listed below.

Block messages containing certain words

This feature blocks messages that contain or closely matched predefined words. Moderator can create a collection of blacklisted words from the settings menu. (Moderate live chat.)

Hold potentially inappropriate live chat messages for review

Live chat messages on YouTube can be held for potential inappropriate content. Those live chat messages, which are identified by YouTube's system, will be retained in the chat feed if moderator opts in. Moderator can decide whether these messages should be visible or hidden. (Moderate live chat.)

Slow mode

By setting a time limit between comments, moderator can limit how often each user can comment. Slow mode is already a feature that is implemented on Liveto's virtual event platform's chat. (Moderate live chat.)

3.2.2 Twitch

Twitch is an online video service focused on live streaming, main category of platform's content being gaming and eSports. With gaming community being on the younger side, the moderation for the live chat must be efficient. Key moderation features are listed below.

AutoMod

AutoMod stops potentially harmful messages from appearing in a channel by automatically catching them, therefore they can be reviewed by the channel moderator before appearing publicly. Users who send messages that AutoMod flags as potentially inappropriate will have their messages held until they have been approved or denied by moderators. (Setting Up Moderation for Your Twitch Channel.)

Block Hyperlinks

All links will be blocked in your channel if this setting is enabled. The only people who can post links with this setting enabled are the channel owner and moderators. Moderators can add individual URLs to your permitted terms if channel needs to allow individual hyperlinks while blocking hyperlinks in general. (Setting Up Moderation for Your Twitch Channel.)

Non-Mod Chat Delay

Channel chat messages can be delayed for a short time. With this delay messages can be removed before viewers are able to see them. The setting of 2 seconds is a quality compromise between moderation and viewer experience. (Setting Up Moderation for Your Twitch Channel.)

This is a list of banned chat users that can be reviewed and unbanned. The Unban Request feature allows banned users to submit an unban request to channel owners and moderators in the Chat column to be reviewed. The Unban Request feature can be turned on / off by a channel owner. In addition to enabling the feature, you can set a cooldown period, which prevents banned users from filling the request before the cooldown period runs out.

4 User Interface and User Experience

There is a possibility that the virtual event moderators can be not as tech-savvy as expected; therefore, the user experience for the moderation panel needs to be carefully planned and designed. In addition to optimal user experience, stylistically the moderation panel must follow Liveto style guidelines.

4.1 Interface development approaches

Friedman (2012) has stated: “Usability and the utility, not the visual design, determine the success or failure of a web-site.” The main target of user interface and user experience research is to find correct approaches to developing an easy-to-use moderation panel.

Credibility and quality are important to users. Users are willing to accept advertisements and design compromises if a page provides high-quality content. The main objective of analyzing a

webpage is to find some fixed points that will guide the user through the content. (Fadeyev, Friedman & Mifsud 2012, Chapter 1) In this case the main fixed point would be the moderation panel's navigation system. The average web-user is impatient and expects immediate gratification. The general rule is this: If a website does not fulfill users' expectations, then the designer has not done his job properly and the business loses money. (Fadeyev, Friedman & Mifsud 2012, Chapter 1.)

The web application's design must minimize the amount of question marks, it must be as obvious as possible. According to Friedman (2012), if the navigation and site architecture are not intuitive, the number of questions increases, making it more difficult for users to understand how the system works.

"The human eye is a highly non-linear device, and web-users can instantly recognize edges, patterns, and motions" (Fadeyev, Friedman & Mifsud 2012, Chapter 1). Therefore, the moderation panel might need to be hosted inside a modal with blacked out background, this will make the edges of the panel clear.

A white space's importance cannot be overestimated. As well as reducing the cognitive load for visitors, it makes it easier for them to comprehend the information presented on the screen. Visitors typically first scan the page and divide the content area into digestible chunks when they approach a design layout. (Fadeyev, Friedman & Mifsud 2012, Chapter 1.) The components of the panel must have enough space between them, this will make the panel not overly crowded with content and less confusing for the user.

4.2 User options template

The user settings modal (Figure 5) on Liveto virtual event platform can serve as a perfect template for the moderation panel. The settings modal has proved itself to be intuitive with the virtual event platform users in the past. It includes a side navigation (Figure 5, section 1), a header (Figure 5, section 2), and a main scrollable content section (Figure 5, section 3). The default colors and styles being from Liveto guidelines, but also customizable with virtual platform's "custom CSS" feature.

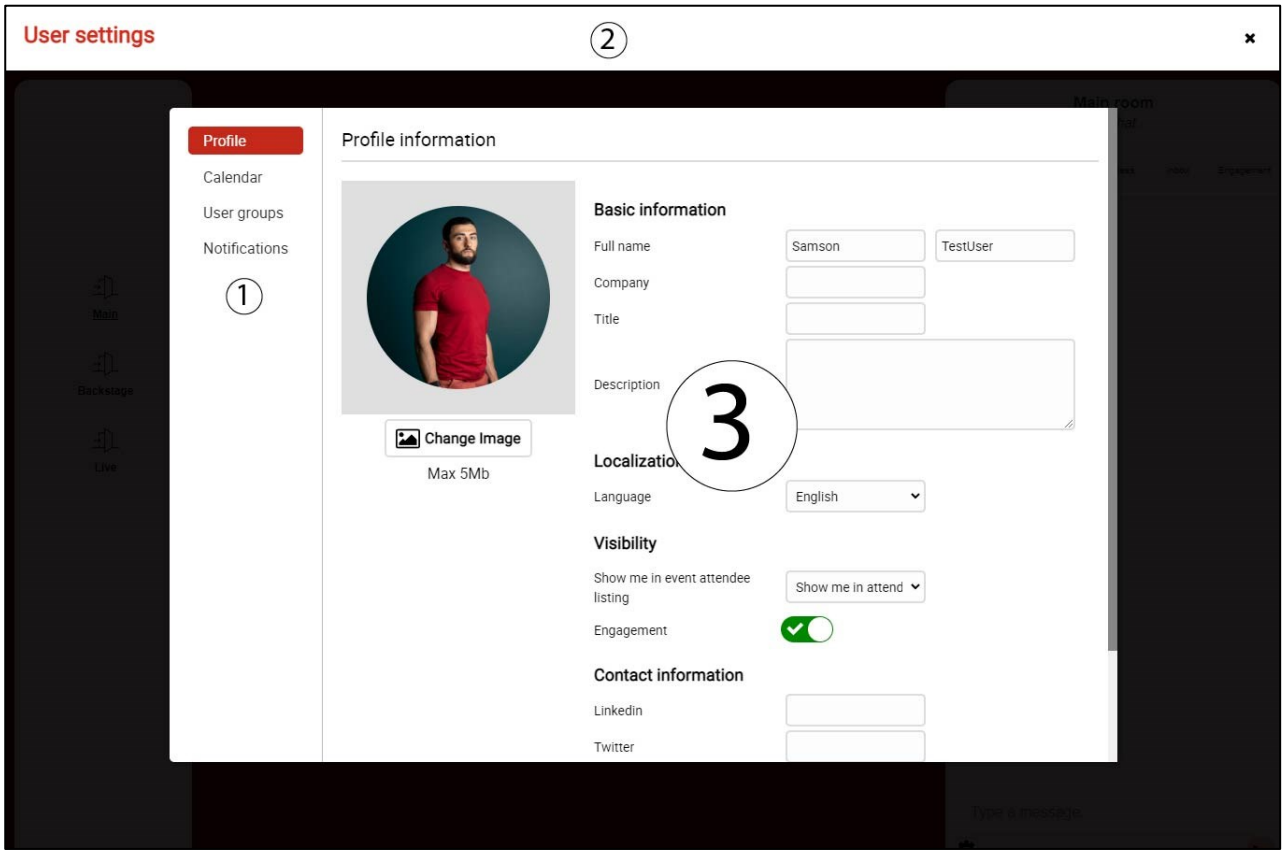

Figure 5. User settings modal

5 Platform architecture

Even though the moderation panel is a sizable module, it is still a small part of the virtual platform architecture. There are multiple moving pieces in play to make the moderation panel work. PostgreSQL database will store all the moderation settings, REST API server will manage http calls and modify the data in the database. REDIS database will store word blacklists, user whitelists and held messages. Signalling service controls the logic for chat and for moderation. Virtual event platform React frontend holds all the user interface logic.

As shown in the architecture diagram (Figure 6), moderation panel is not communicating with the REST API server directly, all communications go through the signalling, that includes signalling client and signalling service. The websocket connection is a two way always open socket connection, these kinds of connections are mainly used for live chats. Signalling service stores the message data into REDIS database. Moderation settings data is a persistent data, signalling service calls REST API and REST API stores the persistent data into PostgreSQL database. React redux store is

used for the locally stored data; redux shared state makes it easier to share data between react components.

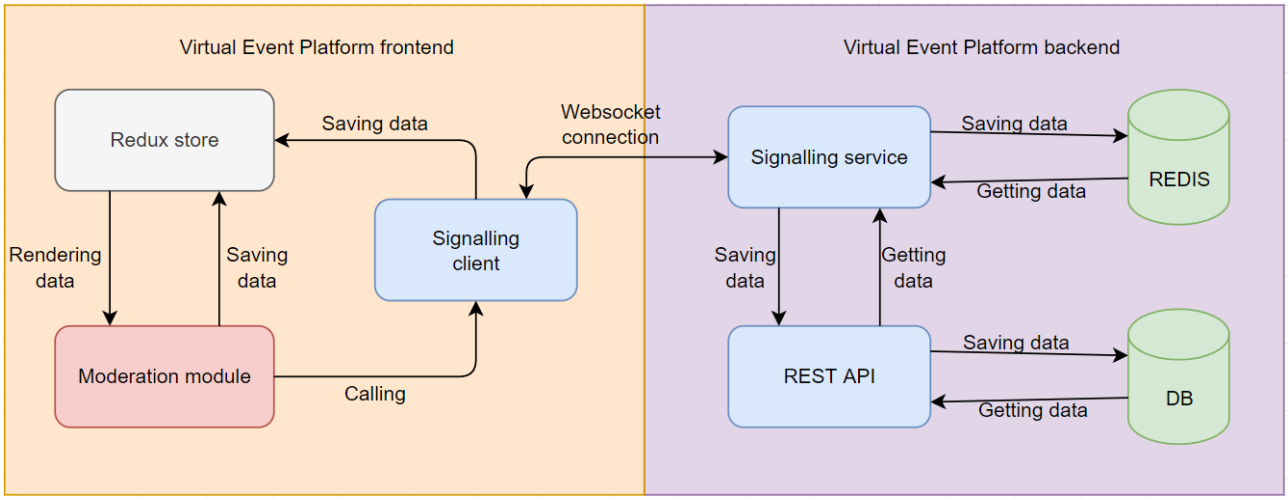

Figure 6. Moderation architecture

6 Technologies

With an addition to Webpack's Module federation there are various technologies used on this project.

6.1 JavaScript

Main programming language that is used for the frontend and some of the backend development at Liveto is JavaScript. JavaScript turns standard HTML websites into interactive and dynamic web applications.

> *"The JavaScript language, working in tandem with related browser features, is a Web-enhancing technology. When employed on the client computer, the language can help turn a static page of content into an engaging, interactive, and intelligent experience. Applications can be as subtle as welcoming a site's visitor with the greeting "Good morning!" when it is morning in the client computer's time zone—even though it is dinnertime where the server is located. Or applications can be much more obvious, such as delivering the content of a slide show in a one-page download while JavaScript controls the sequence of hiding, showing, and "flying slide" transitions while navigating through the presentation. (Goodman & Morrison 2007, Chapter 1.)"*

6.2 Version Control

Managing changes to software code is the goal of version control, also known as source control. Software version control systems help software teams keep track of changes over time to source code. Version control systems have become increasingly important as development environments have become more complex. (What is version control?)

Liveto is using Atlassian's Bitbucket as a version control platform, naturally Bitbucket will be used on this project as well.

6.3 Node.js

Node.js is an asynchronous open-source event-driven JavaScript runtime tool. The following "hello world" (Figure 7) example shows the concurrency of Node.js. A callback is fired for each connection, however, if there is no work to do, Node.js goes to sleep. (About Node.js.)

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Figure 7. Node.js "hello world" example

On this project Node.js is used for the developing of the signalling service for handling the moderation's WebSocket communication. As requirement for moderation is to have a real time live traffic connection, similar to chat's logic.

6.4 React

React is a component-based JavaScript framework, that was developed by engineers at Facebook. React was created for building user interfaces that can work with dynamically changing datasets. Two-way data binding and rendering templates are common features in React applications. Through some daring advances in web development, React changed how Model-View-Controller applications were created. (Gackenheimer 2015, Chapter 1.)

The frontend impelentation of this project will be developed using React as it is the company's main frontend development framework.

7 Implementation

The implementation began with couple of meetings, goals of those meetings were a feature list and an architecture of this project. The list of the possible features has been reviewed and key features were selected for the implementation. For the chat moderation the initial requirement was simply to hold all the messages for the review. The additional features would make the moderator's task easier as the amount of held chat messages can be substantial.

As the deadline for the moderation panel was set for the beginning of June 2022 and the micro frontend migration might not happen in 2022, the decision was made to develop the moderation platform as a part of the virtual event platform monolith. The moderation panel must be developed as an easily detachable component for the micro frontend conversion.

As this project hosts a substantial amount of code, only examples will be shown and explained throughout the implementation process.

7.1 Features

The moderation panel requires multiple features to be implemented as it is a sizable module for the virtual event platform. These features also represent the settings for the chat and chatrooms. All the features would have a dedicated setting. Chatroom setting can be: "on", "off" or "inherit".

The “inherit” setting means that the chatroom is inheriting this setting from the chat. Chat setting can only be “on” or “off”.

Table 1. Moderation features

Feature	Priority	Risks	Estimated time
Hold all messages	High	The moderator might have a hard time keeping up with all the	16h
Hold all guest mes-sages	High	The moderator might have a hard time keeping up with all the	1h
Blocklist words	Medium		4h
Blocklist users	Medium		4h
Hold by blocklisted word	Medium		6h
Hold by blocklisted user	Medium		6h
Slowmode	High		8h

Hold all messages

The ”Hold all messages” feature means that all the messages would be held for the moderator to review. If this setting is set to “on”, all other holding message settings would be overridden. This was the only one feature that was initially required by a client.

Hold all guest messages

This feature will hold all the messages that were sent by the guest users only. This feature is selected for a development because messages that require moderation usually are sent by guest users.

Blocklist words

Any word can be blocklisted, the moderator decides on which words to blocklist. This feature is the essential part of the “Hold by blocklisted word” feature.

Blocklist users

Messages of blocklisted users would automatically be held for the moderator’s review. This feature is the essential part of the “Hold by blocklisted user” feature.

Hold by blocklisted word

Messages that include the blocklisted word would be held for a review automatically. This feature is selected for development as an assistance for the moderation and to add more customizability. It might get too taxing for the moderator if the event has thousands of participants to moderate all the messages.

Hold by blocklisted user

Messages that are sent by a blocklisted user would be held for a review automatically. This feature was selected for development because of the same reason as the “Hold by blocklisted word” feature, just to give a moderator some breathing space.

Slowmode

Slowmode is a time delay that is required before the next chat message can be sent by a single user. This feature would prevent the chat to be spammed with a lot of messages.

Moderation settings

Initially the location for moderation settings was supposed to be in the management platform, but the decision was made to move the settings to the moderation panel as it would be too difficult to change settings, especially mid-event. Management panel can only be accessed by the event organizers and moderators do not have event organizer's access rights; therefore, it would only be logical that the moderation settings can be accessed through the moderation panel.

7.2 Moderation workflow

Chat moderation diagram (Figure 8) shows the relationship flow between the chat feed and the moderation. The life cycle of a message begins with user sending a message, message proceeds to the signalling service which stores the message into the REDIS database. If message does not need to be held for review, the signalling service sends signal to virtual event attendees with the message content and message shows up on the bottom of the chat feed. In case of message being flagged and must be held for review, the signalling service sends a signal to moderators with a flagged message, as a result, the message will show up on the moderation panel as a held message. Moderator can either approve or deny the message, if the message is approved, the signalling service sends a signal to every attendee with the chat message; therefore, the message is visible in the chat feed. In case of a denial of the message, the message gets deleted from the REDIS database by the signalling service.

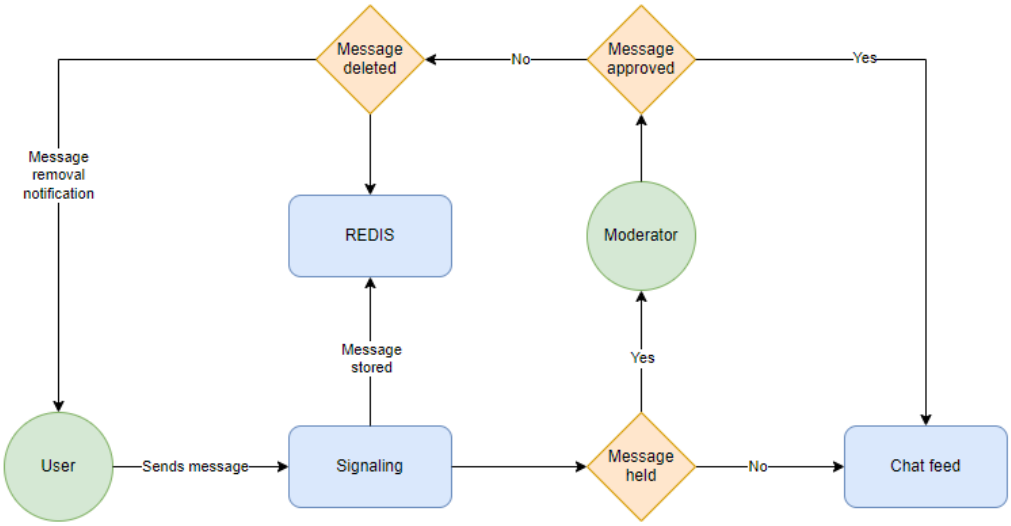

Figure 8. Chat moderation flow

7.3 Component structure

The component structure (Figure 9) of the moderation panel follows the Liveto software development guidelines. The parent folder includes a main component moderation.jsx, sub-folders components, css and utils. “components” folder includes two JSX files chatComponents.jsx and moderationsComponents.jsx, these files consist of smaller react components (Figure 10) that are used in either the main component or within another smaller component. The reason to have all the smaller components within two files is based on efficiency. The current structure makes development easier and faster. A need to search for a file every time one small piece of code needs an adjustment is eliminated. “css” folder includes the SASS style file moderation.scss. “utils” folder includes two files: helpers.js and hooks.jsx, hooks.jsx contains all the custom React hooks used in the moderation panel and helpers.js contains all the helper method used in the moderation panel.

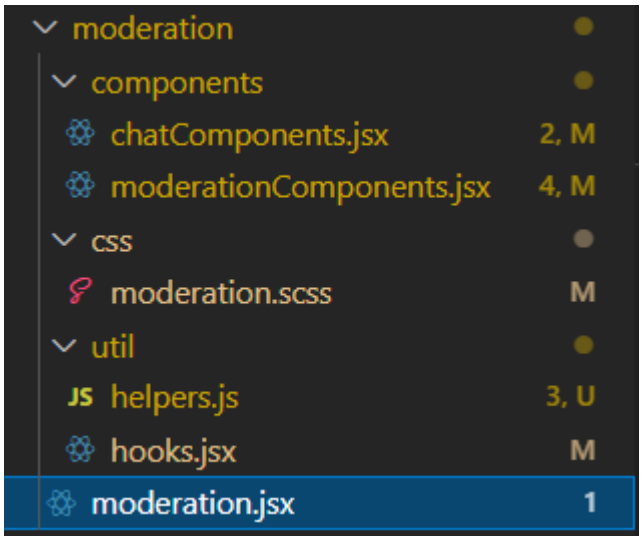

Figure 9. Folder structure

```
export const ModerationViewHeader = (props) => {
  const { currentView = "chat", close } = props
  return <div className="moderation-section-header">
    <Localized id={`moderation-${currentView}-header`} />
    <button className="close-button" onClick={close}><IconTimes /></button>
  </div>
}
```

Figure 10. Small react component

7.4 UI/UX

The development of the user interface for the moderation panel follows the Liveto guidelines. The moderation panel is developed as a pop-up modal, that can be accessed from the main header dropdown (Figure 11). The main header is populated with multiple navigation links and buttons; therefore, the most logical location for the moderation panel button would be inside the profile dropdown component.

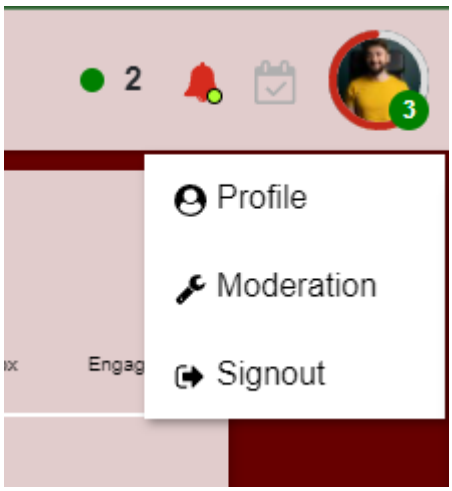

Figure 11. Dropdown menu in the main header component

7.4.1 Moderation modal

The moderation panel modal (Figure 12) includes a side navigation (Figure 12, section 1), a header (Figure 12, section 2), a chatroom filter (Figure 12, section 3), and a main content section (Figure 12, section 4). All components are of a default Liveto virtual event platform style. Navigation tab font size is 16px with active tab being red with white text and inactive tab being white with black text. Header title text is 20px in size and red in colour. The decision to use these specific colours heavily based on Liveto's theme being mostly red and white. All the components with CSS classes are customizable in style with the custom CSS file injection.

Figure 12. The Moderation Panel modal

The header component (Figure 12, section 2) consists of two elements: a section name and a close button. These elements are located on opposite sides of the header component and are centred vertically.

7.4.2 Chat moderation section

Chat moderation section includes a chatroom dropdown filter component (Figure 12, section 3) and a held messages container (Figure 12, section 4). Chatroom filter component is created with an additional react-select library, it offers an easily integrated and customizable dropdown / search component.

A held messages container lists all the held messages as card type component (Figure 13). This style of list components was chosen because cards are easily distinguishable from each other. The held message card contains user's name, chatroom name, timestamp, message, and a moderation button row.

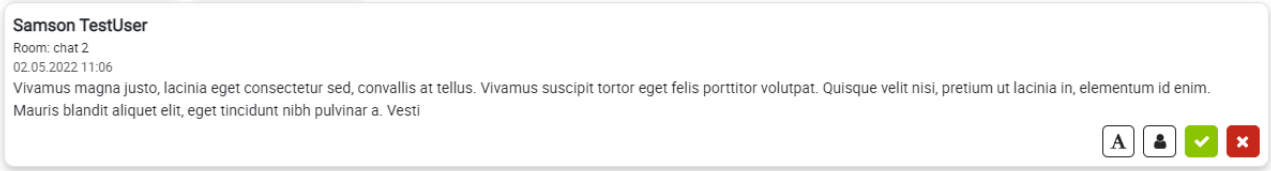

Figure 13. Held message card component

The “Blocklist words” – button is a toggle type button, clicking the button triggers an event that converts held message’s words into buttons and adds a new “Submit” – button into the button row (Figure 14). Blocklisted words show up as red buttons, by clicking the “word” button, the words can be either blocklisted or unblocklisted. Changes are saved by clicking the “Submit” button.

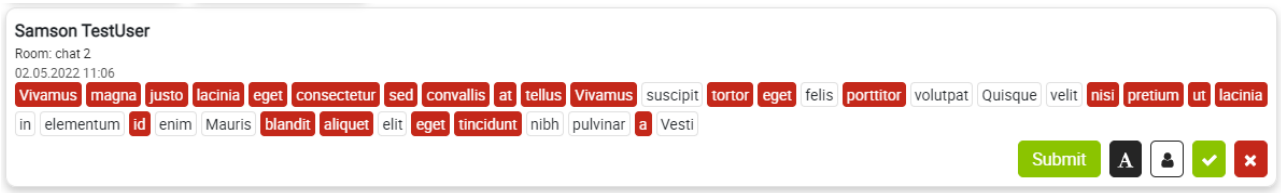

Figure 14. Held message card, blocklist words mode

The “blocklist user” button adds user to / removes user from the blocklist. The “accept message” button accepts the message, as a result, held message card for this message gets removed from the moderation panel and the chat message gets added to the bottom of the chat feed. The “decline message” button declines the message, held message card for this message gets removed from the moderation panel and this message gets deleted from the REDIS database.

7.4.3 Moderation settings section

The moderation settings section (Figure 15) of the moderation panel contains three sub-sections and a “Submit” button. The sub-sections are a “general setting” section, “blocklist words” and “blocklist users”.

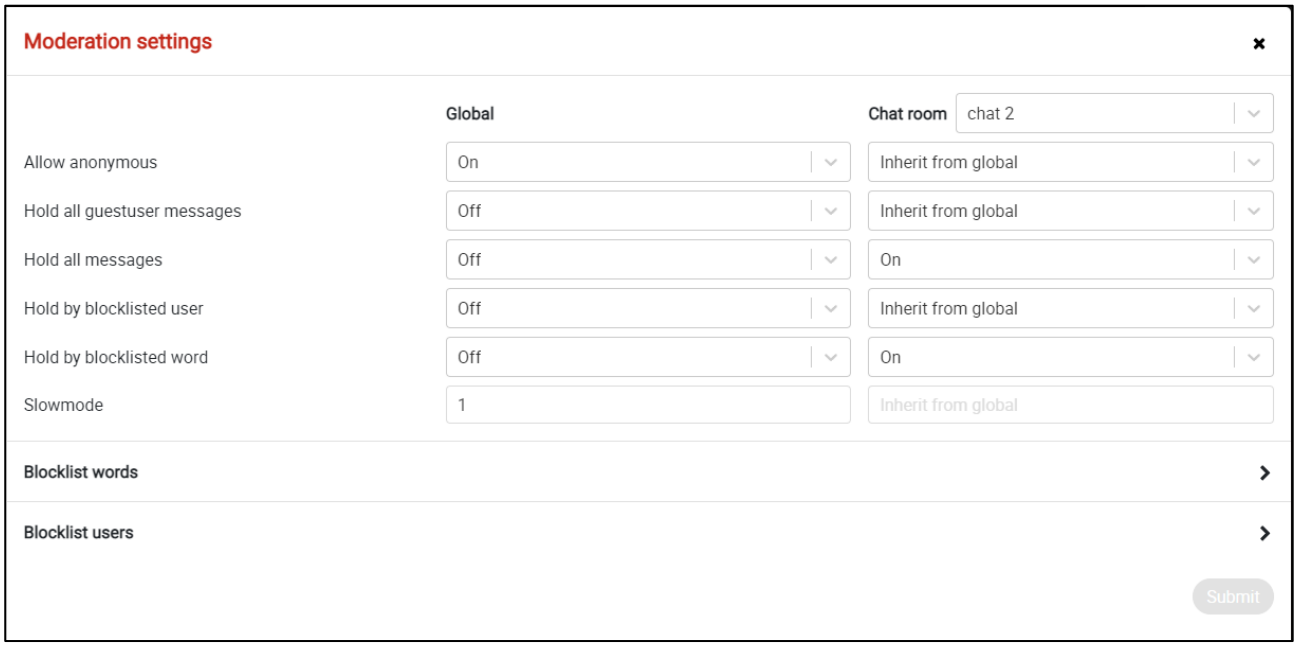

Figure 15. Moderation settings section

The general setting sub-section contains all the settings that can be adjusted globally or for a specific chat room. The chat room dropdown selector lets the moderator select a chat room. The "Submit" button is disabled by default; the button gets enabled if the settings are changed and are not equal to the initial values.

The "blocklist words" (Figure 16) and "blocklist users" (Figure 17) sub-sections are expandable on click and similar in design. Both sub-sections contain two inputs and a blocked content container, the first input filters the blocked content and the second one lets the moderator add a new blocked word / user. In the "blocklist users" case, the second input is a dropdown selector. "Add" button adds a blocked content to the list. Every blocked user / word component includes a "x" button to remove the user / words from the blocklist.

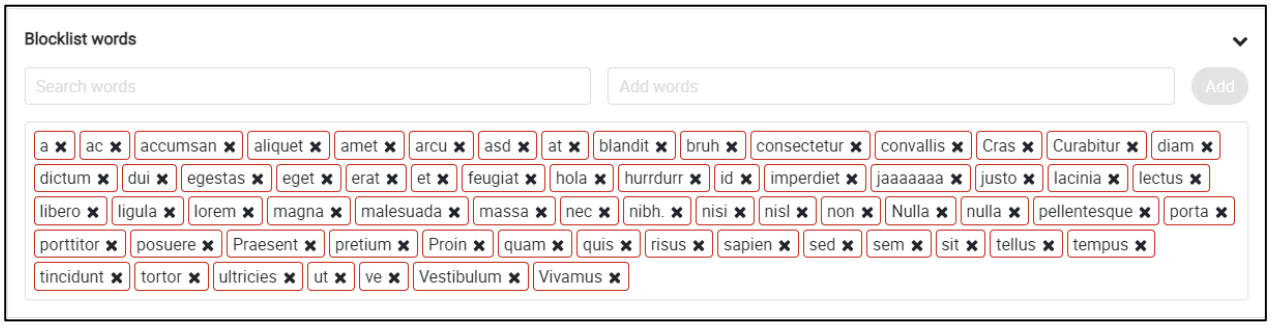

Figure 16. Blocklist words sub-section

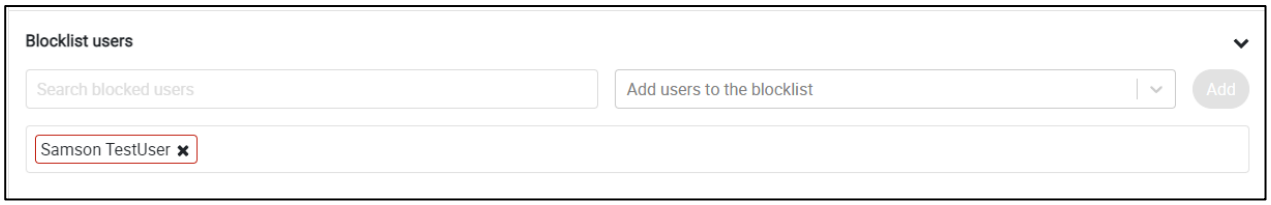

Figure 17. Blocklist users sub-section

The ”Submit” button (Figure 15) saves the settings, blocklists included.

7.4.4 Chat feed

Held for review messages will only show up to the message sender. A message is greyed out and includes an info text below (Figure 18). Once message is approved the greyed-out message is deleted and the message is added to the bottom of the chat feed for everyone to view. If the message is declined, the user receives a push notification about the declination.

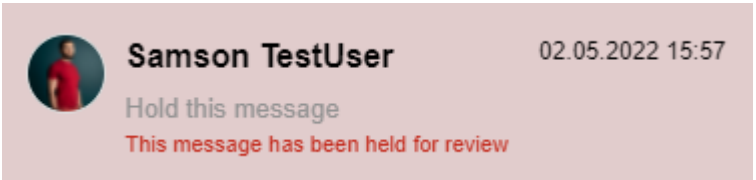

Figure 18. Held message in chat feed

7.5 State Management

With the moderation panel being an easily detachable module, all the moderation data is stored in the local state, using React's useState hook. The data gets requested from the signalling service within the dedicated component inside a React's useEffect hook, useEffect hooks take two parameters in, a callback function and a dependency array. If the dependency array is empty, useEffect hook runs the callback function once when the component is rendered initially. In case if the dependency array containing one or more elements, the callback function is being run every time value of any element inside the dependency array changes. In this case, the initial global moderation settings need to be requested only once; therefore, the useEffect dependency array remains empty. As shown in Figure 19, the settings data is requested from the signalling service by calling chatServiceInvoke method, after that the response data is assigned to the response variable. Lastly, the data gets set to the settings variable in local state by using setSettings method. The request is surrounded with try/catch statement, that can catch an error preventing the application from crashing.

```
useEffect(() => {
  const fetchData = async () => {
    try {
      setFetching(true)
      const response = await chatServiceInvoke('modGetEventSettings', { event_slug })
      setSettings(response)
      chatrooms.forEach(async (room) => {
        const { room_id } = room
        await chatServiceInvoke("modRequestHeldMessages", { room_id })
      });
    } catch (error) {
      console.error("Error fetching global moderation settings: ", error)
    } finally {
      setFetching(false)
    }
  }
  fetchData()
}, [])
```

Figure 19. Moderation data request

Some additional data that is needed to enrich the moderation data is stored in the shared state, in this case React Redux store.

7.6 Custom hooks

React provides the developer with the ability to create custom hooks that get executed on passed parameter change. Creating custom hooks is a quality method to hide a chunk of complicated code that otherwise would have been located inside a dedicated component. Custom hooks are very similar to regular React components

Figure 20 represents the largest custom hook of this project, useChangeModerationSettings hook is responsible for the moderation settings' modification. The hook takes in three parameters: initSettings, setSettings, setModSettings. The initSettings parameter is an object that includes initial global and chatroom settings, setSettings and setModSettings parameters are callback functions that are used for setting the global settings and the chatroom settings into the parent component's local state, respectively. The hook returns an object with five fields: submitButton, inProgress, modify, error and status.

modify

The modify method is used to edit modifiedGlobal and modifiedChatroom local state objects.

submitButton

A button component that is used to fire the submitSettings function, which calls the signalling service to edit moderation settings. The button is disabled if modified setting objects are empty or if the signalling service call is in progress.

inProgress

The inProgress variable is of a boolean type, it used for parent component to display the loading animation while the signalling service call is in progress.

error

The error variable is for storing the errors caught by the try/catch statement.

status

The status variable stores the status of the signalling service call, it can be null, “success” or “failure”, in case of null, the status component is not shown to the end user. The handleStatus function sets the status variable to null after 5 seconds.

```
export function useChangeModerationSettings(initSettings, setSettings, setModSettings) {
    const { global, chatroom } = initSettings
    const [modifiedGlobal, setModifiedGlobal] = useState({})
    const [modifiedChatroom, setModifiedChatroom] = useState({})
    const isGlobal = Object.keys(modifiedGlobal).length > 0 // Global settings modified
    const isRoom = Object.keys(modifiedChatroom).length > 0 // Chat room settings modified
    const [inProgress, setInProgress] = useState(false)
    // Submit button is disabled if there are no changes to the initial value or the request is in progress
    const disabled = (!isGlobal && !isRoom) || inProgress
    const [status, setStatus] = useState(null)
    const [error, setError] = useState(null)
    const event_slug = useSelector(state => state.event.slug)
    const handleStatus = (val) => {
        setStatus(val)
        setTimeout(() => {
            setStatus(null)
        }, 5000)
    }

    // Modifying initial data into new objects
    const modify = (type, key, value) => {
        if (type === "global") {
            let cond
            if (Array.isArray(value)) {
                cond = checkLists(global[key], value)
            } else {
                cond = global[key] === value      You, 5 days ago • Added list type settings to the moderaton setting…
            }
            if (!cond) {
                setModifiedGlobal(prev => {
                    return { ...prev, [key]: value }
                })
            } else {
                const { [key]: value, ...newObj } = modifiedGlobal
                setModifiedGlobal(newObj)
            }
        } else {
            if (chatroom[key] !== value) {
                setModifiedChatroom(prev => {
                    return { ...prev, [key]: value }
                })
            } else {
                const { [key]: value, ...newObj } = modifiedChatroom
                setModifiedChatroom(newObj)
            }
        }
    }
    // Submitting changes
    const submitSettings = async () => {
        try {
            setInProgress(true)
            if (isGlobal) {
                const payload = {
                    event_slug,
                    changes: modifiedGlobal
                }
                const response = await chatServiceInvoke("modChangeEventSetting", payload)
                setSettings(response)
                setModifiedGlobal([])
            }
            if (isRoom) {
                const payload = {
                    room_id: chatroom.room_id,
                    changes: modifiedChatroom
                }
                const response = await chatServiceInvoke("modChangeRoomSetting", payload)
                setModSettings(response)
                setModifiedChatroom([])
            }
            handleStatus("success")
        } catch (err) {
            handleStatus("failure")
            setError(err)
        } finally {
            setInProgress(false)
        }
    }
    const submitButton = <button className="submit-setting-change generic-liveto accept" disabled={disabled} onClick={submitSettings}>
        <Localized id="invitation-submit-invitation-button">Submit</Localized>
    </button>
    return { submitButton, inProgress, modify, error, status }
}
```

Figure 20. useChangeModerationSettings custom hook

7.7 Helper functions

All helper functions are located in helpers.js file, inside util folder. Functions are located inside a separate file; therefore, the main components will not get too large and populated with functions. The second reason is reusability, these functions are very generic and can be reused within different components, methods, and custom hooks.

```
1
2  export const stripFromPunctuations = (word) => {
3      return word.replace(/[.,\/#!$%\^&\*;:{}=\-_`~()]/g, "").trim();
4  }
5  export const sortArray = (list, cond) => {
6      const sorted = list.sort((a, b) => a[cond].toLowerCase() > b[cond].toLowerCase() ? 1 : -1);
7      return sorted
8  }
9  export const sortStringArr = (list) => {
10     return list.sort((a, b) => (a.toLowerCase() > b.toLowerCase()) ? 1 : -1)
11 }
12 export const checkLists = (arr1 = [], arr2 = []) => {
13     const cond1 = arr1.every(a => arr2.includes(a))
14     const cond2 = arr2.every(a => arr1.includes(a))
15     return cond1 && cond2
16 }
```

Figure 21. helpers.js file

The file contains four helper functions: stripFromPunctuations, sortArray, sortStringArr and checkLists.

stripFromPunctuations

This function strips the words from punctuations and whitespaces. Used when converting words into buttons (Figure 14).

sortArray

This function sorts an array of objects by an object's string property alphabetically. Used for sorting users by user's names inside the "blocklist users" sub-section (Figure 17).

sortStringArr

This function sorts an array of strings alphabetically. Used to sort blocklisted words (Figure 16).

checkLists

Checks if two arrays of objects contain the same exact object. Used inside the modify function (Figure 20) to compare the initial settings and the modified settings.

8 Results

As a result of this thesis, a fully functional moderation panel for the virtual event platform was developed. Through the moderation panel, a moderator receives a comprehensive set of tools for reviewing the chat feed messages. Moderator can accept / decline chat messages, add words or users to a blocklist. Two tiers of chat moderation settings can be edited from the moderation panel: global chat settings and chatroom specific settings. Chatroom setting tier can inherit settings from the global chat setting tier. The moderation settings can be global (chat specific) or chatroom specific, this adds to the customizability of the process.

8.1 Further development

Although the moderation panel is fully functional, it is still in the experimental stage, and it needs a further development to be a production ready software. At the end of the development process, two new possible features have emerged as very useful moderation tools, these features are “Send a moderation message from the moderation panel” and “Report a chat message”. These features will be implemented and released in future updates.

8.1.1 Responsiveness

Up until this stage, the moderation panel was developed as desktop only version. The virtual event platform being available on all types of devices, it requires that the moderation panel be also adjusted to be available for these devices. The next logical development step would be to adjust the styles of the moderation panel to be available on mobile and tablet devices in addition to desktop devices.

8.1.2 Testing and buf fixes

After the initial development is completed, the software moves to the testing stage of the development. The testing stage includes automated and manual testing, after the testing is completed by a dedicated Quality Assurance Developer, the possible list of bugs is created, and the software will go back to the development stage. After all the bugs are fixed, the software will go through testing again.

8.1.3 Code review

As soon as all tests are passed, the pull request to the staging environment is created. A senior developer will review the code and give feedback on possible improvements. After the pull request is approved, the new code will be merged into the staging environment and eventually into the production.

8.2 Problems

No substantial problems were faced while working on this project, however one minor problem did present itself. The time travelling of the held for a review chat messages, the message that is held for a review is sent earlier than it is reviewed and possibly added to the chat feed. Two possible solutions were proposed for this problem, first solution was to insert a reviewed message into the chat feed based on time it was originally sent, the second solution was to simply insert the reviewed message at the bottom of the chat feed as the newest message. The solution selected was the later one because in the first solution, if the chat is lively, the message might get hidden and never seen by another user.

9 Conclusions

The initial goal was to develop a moderation panel as an independent micro frontend module, but with the deadline being set on a substantially earlier date and the micro frontend migration is not being even 50% ready by then, the panel was developed as an easily detachable part of the monolithic virtual event platform React application.

The fully functional moderation panel that includes all the initially planned features was developed as a result. Additionally, the moderation module is scalable; therefore, the moderation capabilities can go beyond the chat. The moderation panel component is easily detachable; therefore, it is micro frontend modularity migration ready.

Further development of the software created during this thesis process is needed. The software would not currently be sufficient for comprehensive production use considering it is lacking the responsive styling and quality assurance. It would benefit possible moderators to be instructed on how to use the software before it being introduced into production.

References

Fadeyev, D., Friedman, V., Mifsud, J. 2012. User Experience: Practical Techniques, Volume 1. Freiburg: Smashing Magazine GmbH.

Gackenheimer. C., 2015. Introduction to React. New York: Apress.

Geers, M. 2020. Micro Frontends in Action. Shelter Island: Manning Publications.

Goodman, D., Morrison, M. 2007. JavaScript Bible, Sixth Edition. Indianapolis: Wiley Publishing.

Jackson, C. 2019. Micro Frontends. Accessed on 19 March 2022. Retrieved from https://martin-fowler.com/articles/micro-frontends.html

About Node.js. N.d. Page on Node.js's website. Accessed on 9 April 2022. Retrieved from https://nodejs.org/en/about/

Hybrid Events. N.d. Page on Liveto Group Inc's website. Accessed on 19 March 2022. Retrieved from https://liveto.io/en/hybridevent/

In-person Event. N.d. Page on Liveto Group Inc's website. Accessed on 19 March 2022. Retrieved from https://liveto.io/en/in-person-event/

Moderate live chat. N.d. Page on Google's support website. Accessed on 2 April 2022. Retrieved from https://support.google.com/youtube/answer/9826490?hl=en#zippy=

Plans for teams. N.d. Page on Bit's website. Accessed on 2 April 2022. Retrieved from https://bit.cloud/pricing

Setting Up Moderation for Your Twitch Channel. N.d. Page on Twitch's support website. Accessed on 2 April 2022. Retrieved from https://help.twitch.tv/s/article/setting-up-moderation-for-your-twitch-channel?language=en_US

What is Bit? N.d. Page on Bit documentation's website. Accessed on 2 April 2022. Retrieved from https://bit.dev/docs/quick-start

Virtual Events. N.d. Page on Liveto Group Inc's website. Accessed on 19 March 2022. Retrieved from https://liveto.io/en/virtualevent/

What is version control? N.d. Page on Atlassian’s website. Accessed on 9 April 2022. Retrieved from https://www.atlassian.com/git/tutorials/what-is-version-con