Suleyman Demirel University
Faculty of Engineering and Natural Science

DIPLOMA PROJECT

Social network with workspace

Nurali Ainabekov
Azhar Ilyassova
Nazerke Nazarkulova
Toty Kabasheva
Abay Kokenov

6B06101 - “Information Systems”

Kaskelen, 2022

Suleyman Demirel University
Faculty of Engineering and Natural Science

Dean of Faculty
Assoc. Prof., PhD Azamat Zh.

_____

«__» _____2022y.

DIPLOMA PROJECT

Social network with workspace

6B06101 - “Information Systems”

Superviser: Matayev Sh. MBA
Student: Nurali A.
Azhar I.
Nazerke N.
Toty K.
Abay K.

Kaskelen, 2022

Abstract

The government had to force into lockdown due to the continuous COVID-19 epidemic, and many people faced with the necessity to reduce their social communications dramatically. Only a few studies have taken into consideration the negative psychological effects of enforced social isolation and the ways they can be reduced in real-world conditions due to the unique nature of such scenarios. We saw that Kazakhstan was not ready for distance learning and remote work during the pandemic. Most of us have never heard of online learning and work before and have had to deal with problems working on video platforms. The frequent reason was a complicated design or special limits in functionality. According to this, our team aims to develop a multifunctional video platform where users can chat, hold official meetings or just relax and play with friends, they can also spend time watching movies or listening to music together.

Аңдатпа

Үкімет жалғасып жатқан COVID-19 індетіне байланысты карантин еңгізуге мәжбүр болды және көптеген адамдар қоғамдағы қарым-қатынастарын едәуір төмендету қажеттілігіне тап болды. Бірнеше зерттеулерде мәжбүрлі әлеуметтік оқшаулаудың теріс психологиялық салдары және мұндай сценарийлердің ерекше сипатына байланысты оларды нақты жағдайда қалай азайтуға болатындығы қарастырылды. Біз пандемия кезінде Қазақстанның қашықтықтан білім алуға және қашықтықтан жұмыс істеуге дайын емес екенін көрдік. Біздің көпшілігіміз бұрын-соңды онлайн оқыту және жұмыс туралы естімеген едік және бейне платформаларда жұмыс жасауда қиындықтарға тап болдық. Мұның себебі көбінесе күрделі дизайн немесе функционалдылыққа арнайы шектеулер болды. Осыған сүйене отырып, біздің команда көп функциялы бейне платформаны дамытуды мақсат етті, онда пайдаланушылар чат арқылы сөйлесе алады, ресми кездесулер өткізеді немесе достарымен жай демалып, ойнай алады, сонымен қатар фильмдерді бірге көруге немесе музыка тыңдауға уақыт бөле алады.

Аннотация

Правительство было вынуждено ввести карантин из-за продолжающейся эпидемии COVID-19, и многие люди столкнулись с необходимостью существенно сократить свои взаимоотношения в обществе. Лишь в нескольких исследованиях рассматривались негативные психологические последствия вынужденной социальной изоляции и то, как их можно уменьшить в реальных условиях из-за уникального характера таких сценариев. Мы увидели, что Казахстан оказался не готов к дистанционному обучению и удаленной работе во время пандемии. Большинство из нас никогда раньше не слышали об онлайн обучении и работе и столкнулись с трудностями в работе на видеоплатформах. Причиной зачастую становился сложный дизайн или специальные ограничения в функционале. Исходя из этого, наша команда поставила перед собой цель разработать многофункциональную видеоплатформу, где пользователи смогут общаться в чате, проводить официальные встречи или просто отдыхать и играть с друзьями, также смогут проводить время за совместным просмотром фильмов или прослушиванием музыки.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Once upon a time, there was only verbal and non-verbal communication, but now, there is another more efficient type of communication, which is Online communication. Many individuals have begun to interact through the use of the internet. For real, throughout the quarantine networking had already surged in popularity.

Nowadays, after recent pandemic COVID-19, a good number of people have crossed to online, working, studying and communicating. As expected, a large number of platforms, for online networking, were so penetrated to our lives that we can choose one of them for their cool features. In my opinion, we need only one platform for communication, that can satisfy all human needs here and now.

Saying about our project, a multi-user platform where users can chat, debate, hold official meetings, or just relax and spend time with friends while watching movies or listening to music together. Our platform has a modern and user-friendly design, where users can stream and promote their products, and can also gather for a formal meeting. The platform will have several rooms, which the user himself will add and will navigate using the sidebar. On the main page, user can also find recommendations for channels, topics that interest him.

1.2 Aims and Objectives

The goal of this thesis was to develop a useful and functional platform for all. Our group was successful in creating a website, that perfectly fulfills all current human requirements.

1.2.1 Aims

- What is the major idea of our website?
- What kind of problem 'ChatRoom' can solve?

- Project should make online communication easier

1.2.2 Objectives

- Website that allows people easily communicate.
- Modern Design, that will easily perceived.
- Unlimited user connection.
- Users can stream and promote their products, and can also gather for a formal meeting.
- Provide fast and reliable communication between people.

1.3 Thesis Outline

Chapter 1 Introduction: The reason for building the project, as well as the major goals, are discussed in this chapter.

Chapter 2 Project Management: Analyze and Plan

Chapter 3 Design: This section includes User Experience and User Interface Design, where we performed research and detailed descriptions of the interface.

Chapter 4 Frontend: Explaining about Front-End.

Chapter 5 Backend: Explaining about Back-End.

Chapter 6 Conclusion: Summarizing the job that has been completed.

Chapter 2

Project Management

2.1 Methodology

As a methodology of organization, our group used Agile(see Figure 2.1), which is a set of standards based software development technique. Mentioning pros of this technique, because testing is an important component of the project implementation process, the final product's complete quality is improved. Any project that follows the Agile [6] technique should never fail, in principle. Agile operates in tiny sprints with the goal of delivering value on a continual basis. Even if a certain technique does not go as anticipated, there is always a tiny element that may be saved and used in the present. Our team contains 5 people, two Back-End Developer, Front-End Developer, UI/UX Designer and Project Manager

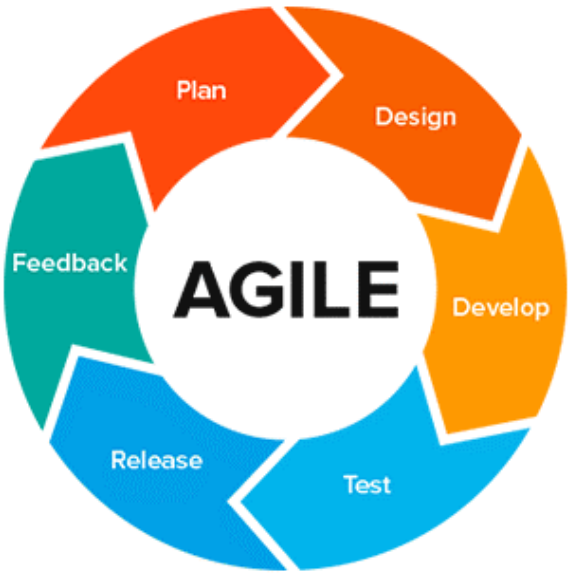

Figure 2.1: Agile diagram

2.2 Workflow

We chose Trello(see Figure 2.2), because, it allows users to interact with their team members and complete work relevant to their projects without having to navigate between apps. Users will be able to view the tasks and who they are allocated to, as well as what has already been done to them, with just one glance. Trello [4] is also highly visual, allowing the system view project phases, team member responsibilities, and assignment due dates. Trello's visibility helps you to focus on handling tasks and finishing tasks rather than wasting time attempting to decipher project specifics.

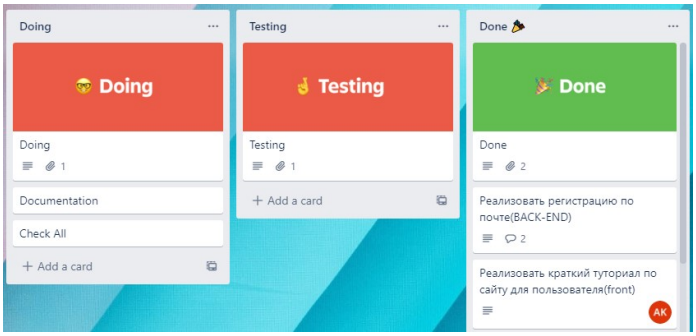

Figure 2.2: Trello

2.3 Analyze

The public of social media has expanded by more than 10 percent in the last year, and there were 3.96 billion [7] users(see Figure 2.3) at the start of July 2020. That is, for the first time, it is possible to assert that people who use social media outnumber those who do not. According to current growth rates, a million individuals establish a social media account for the first time every day, or about 12 new users every second. Although half of the world's population has already been reached, growth looks to have increased in recent months.

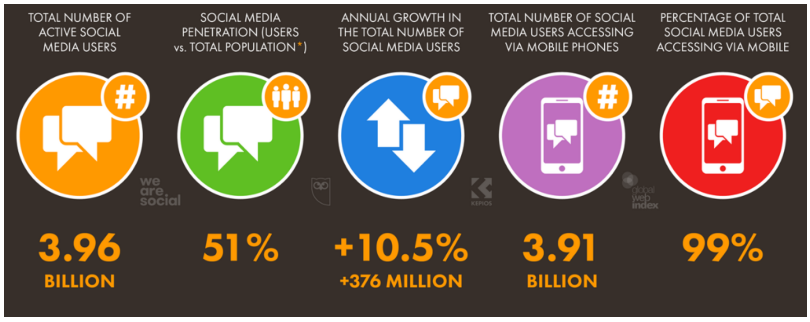

Figure 2.3: Statistics

We conducted a survey(see Figure 2.4) among schoolchildren, adults, students and employees of large companies. As predicted, all of them was using 5-6 online platforms, one for school, one for chatting, one for calling parents and etc.

Figure 2.4: Interview

We came to the conclusion that popular video communication services such as Zoom, Webex and Google Meet are not very convenient to use. For example, to create a video conference on zoom, you need to spend time getting to know the interface, Webex system menu and user interface are not very convenient, it also often freezes, Google Meet lacks a waiting room, you can be thrown out by the platform automatically and screen sharing is limited to one person, you also need to wait when the request is received by the creator of the room, which is not very convenient if he left.

2.4 Team Communication

Due to distance learning, several team members have not been in the same city since the start of the project. All of our talks and forecasts took place over the chat. Members of the team worked on the same duties and were allowed to express themselves. Observations on the project Telegram, Skype, and Google Meet are examples of platforms were utilized for meetings and talks in general.

Chapter 3

Design

3.1 Research

In the process of creating and developing our product, we decided to adhere to the concept of the project life cycle. After studying and deepening into the problem that we faced, having outlined the boundaries of the product functionality, our team proceeded to UX research. This stage lasted quite a long time and the preparation was very thorough. In the course of deepening into the problem and finding a solution to it, various UX tools were applied. The research methods that have led to the most effective solutions are described below. The very first tool that we used was building user personas, studying the target audience of the product. In the following image, you can see the research results. (see Figure 3.1)

Figure 3.1: Summary of target audience

Further, work was carried out with the focus group, the collection of information from potential users of the product. Our request was formulated in this way: "What difficulties do users face when using similar services and how can we improve the user experience in this aspect?" After an hour of interviews with respondents, all collected materials were sorted, and based on the result of this study, points were identified that received the maximum amount of attention.

3.2 UI Kit development and UX patterns

When the flow became more or less clear and the wireframes were rendered, we started developing the UI Kit [5] of the product. You can see some components from UI Kit (Figure 3.2 3.3)

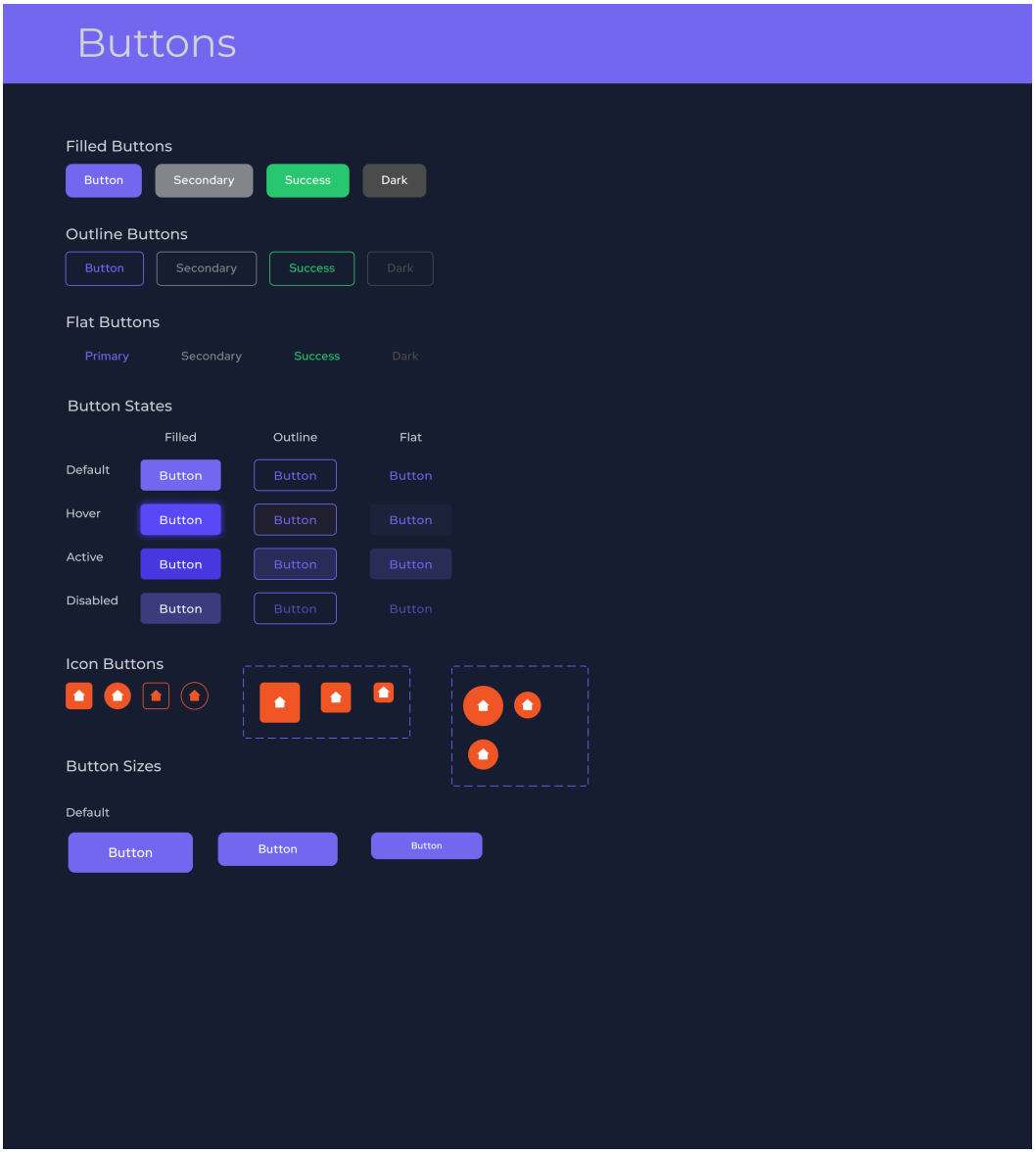

Figure 3.2: Main components

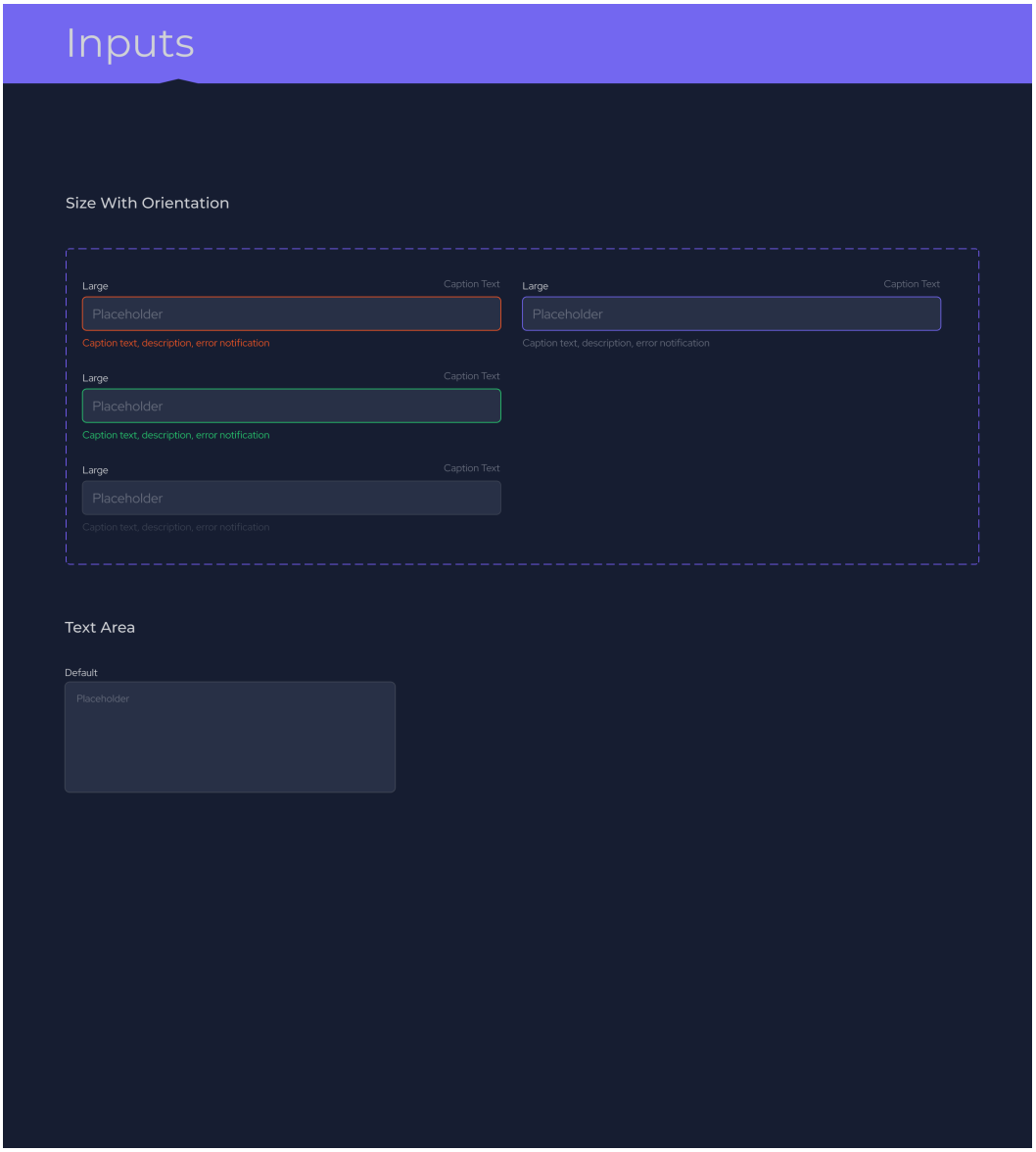

Figure 3.3: Main components

3.3 What is the reason of simplifying the authorization process?

The flow has been simplified as much as possible so that the user does not have difficulty with authorization and quickly gets the opportunity to join a meeting or create a room for it. All the user needs to do is enter their name or nickname and upload their photo.

At the very beginning of development, we, like all similar services, implemented a complex authorization with entering a first name, last name, password, confirmation via SMS and mail, but after that we decided that this would have a bad effect on usability. There are a number of reasons why we have abandoned certain registration methods.

Authorization using social networks. Advantages: fast process, no need to come up with a password, it is possible to learn about user preferences and use this to create promotional mailings, offers. Disadvantages: young audience, for the older generation it may be difficult to use, deleting the account on the social network, the user will not be able to access the site.

User data is stored locally, we save up to 5 profiles, that is, the user, if desired, can log out of his profile, create a new one, or log into an existing one from the device on which he was logged in earlier. When expanding the functionality, simplified authorization will be added to enable access to more service functions.

3.4 How has the development of the Internet affected our design?

Since the renaming of Facebook to Meta, it has become mainstream to have a conversation, to be interested in the metaverses, which is very good, since this is a new stage in the development of the Internet, and everyone understands this. There was a need to hold meetings, to work in a more immersive way. When developing the design, we had the main goal to make such an interface, using which the user will at least get a little closer to the feeling that he is not just writing a text and sending it to the chat members, but also speaking, listening, "moving" around the space.

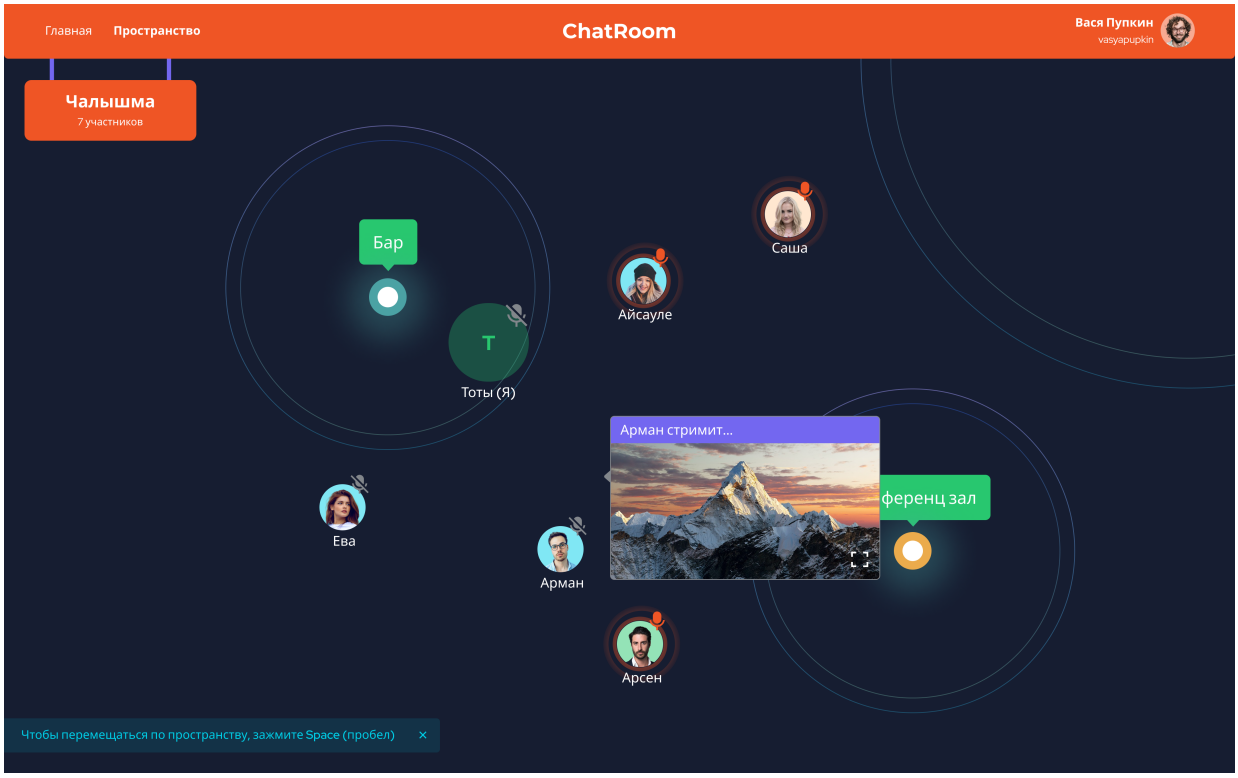

Figure 3.4: User movement design

Chapter 4

Frontend

4.1 Overview

Our main task is a site that will be convenient and easy to use for all ages. Our project has a modern and user-friendly design, with which this task will be easily accomplished. You can quickly register and create a space for working meetings, for studying or for communication with friends in our website. Additional function is you can adjust the sound, by moving your avatar around the room. The role of the frontend is very great to commit and show the final result of the project to the user. The frontend is responsible for the visible part of the project, where the user will interact with the main content.

4.2 Technologies used

Our project uses Typescript with library called React. There are several reasons why. Typescript is more convenient for development, because using it you avoid a lot of errors and bugs, and also save your time. And we used React primarily because React has a huge and strong community. React.js is one of the most popular frontend JavaScript libraries for building web applications.(see Figure 4.1). Github has a React repository with over 1,500 contributors, also users can ask questions in many various discussion forums.

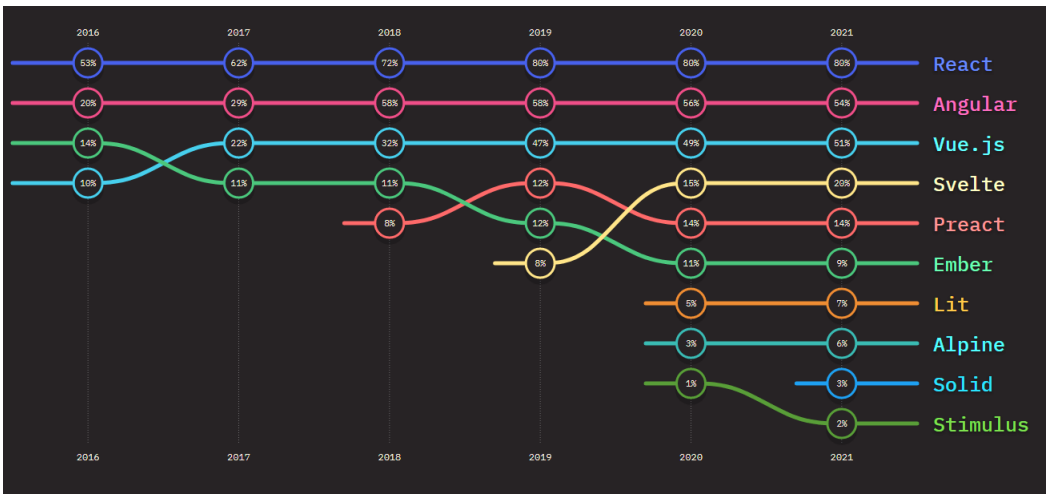

Figure 4.1: The most popular front-end frameworks

Also React consists of reusable components. The structure of the site written in React is based on components. Components are independent functional elements that can be reused wherever you need. It helps easier to develop and maintain your apps. Another advantage is UI-focused design. Design is an important part of our project. Thus, the React is well suited because it is UI-focused. No less significant privilege is react known to be SEO-friendly. React makes it easier for the user to find the right content.

4.3 Packages used

- @chakra-ui/react [1]
  Chakra-UI is a simple, accessible component library to facilitate styling.

- lodash/debounce
  A function is created that is called debounce and it postpones the call until certain milliseconds pass, which we set after this function and wait for the moment of the last call of the canceled function. Debounce is a useful function that takes two parameters. The first parameter accepts a function, subsequently calls the same function, and the second parameter accepts timeout, the time after which this function is called again.

- react-router-dom
  React Router DOM enables you to implement dynamic routing in a web app, thus we can move between components/pages.

- simple-peer
  Peer connects to WebRTC and stores audio and video stream data.

- socket.io-client
  Socket.io helps to connect backend.

- axios
  Axios is a JS library it is an HTTP client with which we can make requests to certain resources.

4.4 Main functions

The main functions of the site are described here.

Authorization

```
const onCreateProfile = useCallback(async (userData: CreateUserData) => {
  let avatarUrl = "";
  if (userData.avatar) {
    const avatarUploadRes = await ApiService.UploadAvatar(userData.avatar);
    avatarUrl = avatarUploadRes.path;
  }

  await userContext.createNewUser(
    {
      name: userData.name,
      avatar: avatarUrl,
    },
    userData.remember
  );

  navigate("/");
}, []);
```

Figure 4.2: Authorization

Creating a room

```
const onCreateRoom: React.FormEventHandler = useCallback(
  async (e) => {
    e.preventDefault();

    await ApiService.CreateRoom({
      id: roomId,
      title: title,
    });

    navigate(`/r/${roomId}`);
  },
  [title, roomId]
);
```

Figure 4.3: Creating a room

Account Selection

```
const onSelectUser = useCallback((id: string) => {
  userContext.setUserById(id).then(() => navigate("/"));
}, []);
```

Figure 4.4: Account Selection

Join Room

```
// Join room
socketRef.current.emit("me:room:join", {
  roomId: roomId,
  user: { ...user, ...userPos },
});
```

Figure 4.5: Join Room

When you call to all in the room

```
// When joined room data recieved
socketRef.current.on("room:data", (data) => {
  setRoomData(data);

  console.log(socketRef.current.id, data);

  // Call all friends in this room
  data.users
    .filter((u: any) => u.socketId !== socketRef.current.id) // except myself
    .forEach((userItem: any) => {
      const peer = new Peer({
        initiator: true,
        trickle: false,
        stream,
      });

      peer.on("signal", (signal) => {
        socketRef.current.emit("me:signal:send", {
          userToSignal: userItem.socketId,
          signal,
        });
      });

      friendsRef.current.push({
        ...userItem,
        peer,
      });

      setPeers((prev) => [...prev, peer]);
    });
});
```

Figure 4.6: Call to all

Handler function when friend calls me

```
// When friend calls me
socketRef.current.on("friend:signal:send", (data) => {
  /*
    {
      signal: payload.signal,
      callerID: socket.id,
    }
  */
  const peer = new Peer({
    initiator: false,
    trickle: false,
    stream,
  });

  peer.on("signal", (signal) => {
    socketRef.current.emit("me:signal:return", {
      signal,
      callerID: data.callerID,
    });
  });

  peer.signal(data.signal);

  friendsRef.current.push({
    peerID: data.callerID,
    peer,
  });

  setPeers((prev) => [...prev, peer]);
});
```

Figure 4.7: Handler function when friend calls me

Handler function when the user you are calling answers the call

```
socketRef.current.on("friend:signal:return", (payload) => {
  /*
    {
      signal: payload.signal,
      id: socket.id,
    }
  */
  const item = friendsRef.current.find(
    (u) => u.socketId === payload.id
  );
  item.peer.signal(payload.signal);
});
```

Figure 4.8: Handler function when the user you are calling answers the call

4.5 WebRTC. How we implemented WebRTC?

The underlying technologies have evolved, but the fundamental remains the same: when we click the button on the site page, JavaScript is run, which asks access

to the microphone and creates a connection with the server - the WebRTC SIP gateway. Furthermore, one client-server leg is a browser-gateway, the second leg can be any length, and the third leg can connect to a mobile or landline phone via the SIP proxy chain. As a result, the browser functions as a softphone and may participate fully in VoIP telephony.

4.5.1 Three fundamental variables are required to create a video chat and screen sharing application.

1. For managing UI, a basic React setup is required.

Basic setup with a join button that calls a backend API, obtains a unique id, and redirects the user to the room (React runs on port 3000) Our backend is listening on port 4000 on localhost, and the response will contain a unique id that will be used as the room id in the next phases.

2. Socket connection support in the backend (Nodejs).

Backend - a simple host setup with a server listening on port 4000 and a router that uses "/connection" to generate a unique id and provide it to the frontend. This uses the uuid package to create a unique string.

3. A peer-to-peer server is a program that allows you to establish and maintain a peer-to-peer connection.

With the disable button enabled, a new component called RoomID is generated, with a container div named "join room" to host our video components.
Now that we have the stream data from our device's camera and microphone, we can utilize the navigator to obtain it. To do this, we may utilize the Connection helper class, which supports all incoming and outgoing streaming data while still maintaining a socket connection to the backend. On the socket connection, the back end should now be listening. Using "socket.io" to make connecting to a socket simple.

We've added a socket connection to the backend to listen for room joins, which will be triggered by a userData containing the roomID and userID from the front end. When a peer connection is established, the user ID is accessible.

The socket then linked the room to the roomID (from the unique id returned in the frontend), allowing us to send a message to all of the users in the room.

Now, socket.to is used (roomID).

We may send a message to all connected users except ourselves by using broadcast.emit('new-user-connect', userData). And because this new user connection is listening on the frontend, all other users in the room will receive the

new user's information.

Chapter 5

Backend

5.1 Overview

For the backend, we also used JavaScript and Socket.io library according to these criteria:

- The Node.js [2]framework, among other things, allows a developer to manage data updates from the front end and create scalable network applications capable of processing several simultaneous user requests.
- Socket.IO, unlike web sockets, lets you deliver messages to all connected clients. For instance, suppose you're building a chat and want to alert all users whenever a new member joins. This is simple to accomplish with only one operation. To complete this work with web sockets, you'll need a list of connected clients and the ability to transmit messages one at a time.
- In web sockets, using proxying and load-balancers is tricky. These technologies are supported out of the box by Socket.IO.
- As previously stated, Socket.IO allows for progressive (graceful) deterioration.
- When a connection is lost, Socket.IO provides automatic reconnection.
- Working with Socket.IO [3]is simpler.

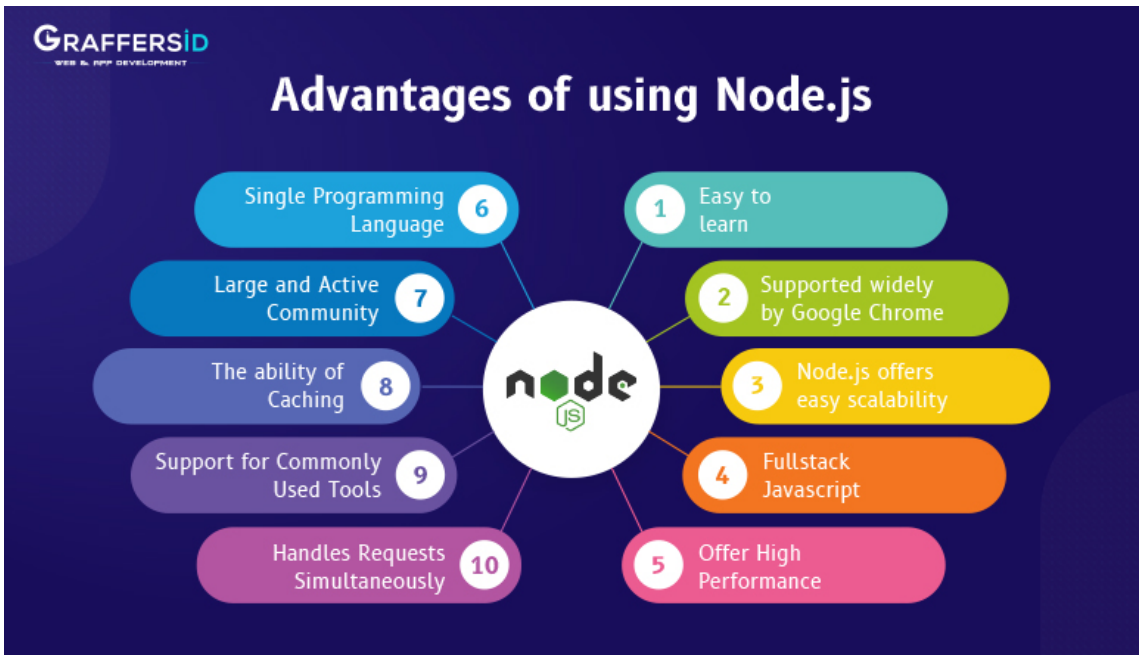

Figure 5.1: Node.js

5.2 Socket.io

Socket.IO is a library that enables low-latency, bidirectional, event-based communication between a client and a server. It uses the WebSocket protocol and has capabilities like HTTP long-polling fallback and automatic reconnection.

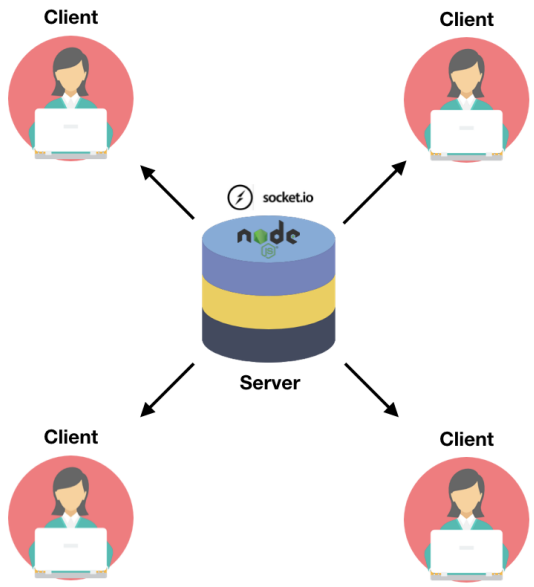

Figure 5.2: Client Server relationship

For designing distributed systems, the client/server paradigm is the most extensively used paradigm. In this method, client programs request services from a server process. As described in the "Basics" section, this implies asymmetry in client-server communication. In this section, we'll look into client-server interactions in further detail, as well as some of the issues that might emerge while developing client-server applications.

Before a service may be offered, the client and server must follow a set of well-known norms (and accepted). A protocol must be implemented on both sides of a connection based on this set of norms.The protocol might be symmetric or asymmetric depending on the scenario. A symmetric protocol allows any party to play master or slave. In an asymmetric protocol, one side is always identified as the master, while the other is always identified as the slave.

5.3 Express.js

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application. It's a layer built on the top of the Node js that helps manage servers and routes. It is used to rapidly and simply design and create web apps. Web applications are applications that may be launched using a web browser. Because Express. js just requires javascript, programmers and developers may quickly and easily create web apps and APIs.

5.3.1 Why did we go with Express.js?

For both frontend and backend development, JavaScript is the only language you'll need to learn. As a result, JavaScript is required, and the backend is created using the same language if Express.js is used. Support for Node.js is one of the most compelling reasons to use Express.js. So that you may create an Express.js backend using all of the NPM packages that make the process easier. You'll be able to debug your application faster than ever before thanks to the debugging mechanism it provides. This simplifies the process of locating faults without spending a lot of time.

5.4 Why we don't use Database?

We decided to leave the database connection at the possible stages of developing our site, we have all this in our MVP. Just such a stage of development will be the connection of the MySQL database. We often use localStorage and sessionStorage store to store session data. In addition, we do not use authorization for our site, so access to user and room data is not required if our streams and meetings are

completed.LocalStorage is a browser-based data storage option. The data is saved as string key/value pairs, and each domain has its own LocalStorage. Make care to transform JavaScript objects to a JSON string before storing them.

5.5 Multer

Multer is a Node. js middleware that handles multipart/form-data and is mostly used for file uploading. For best efficiency, it is written on top of busboy. Multer populates the request object with a body object and a file or files object. The values of the form's text fields are stored in the body object, while the files submitted via the form are stored in the file or files object.

5.6 Postman and Terminal

To keep track of each process of our application when the backend was launched, we used applications such as Terminal and Postman. The Socket instance extends the Node.js EventEmitter class on the server side.

The Socket instance utilizes the component-emitter library's event emitter, which exposes a subset of the EventEmitter functions on the client side.

Two WebSocket servers can be created via the WebSocket echo service. The first is for raw WebSocket connections, while the second is for Socket.IO connections. The Raw WebSocket echo server is a straightforward echo server. It's created with three additional pathways to assist you understand how to use query parameters, headers, and message types in Postman when testing your WebSocket APIs.

The Socket.IO echo server serves as an example of how to use Postman to listen for and handle events from Socket.IO WebSocket servers.

Here you can see examples of how events work: To upload an avatar, responds with an event called "avatar-upload":

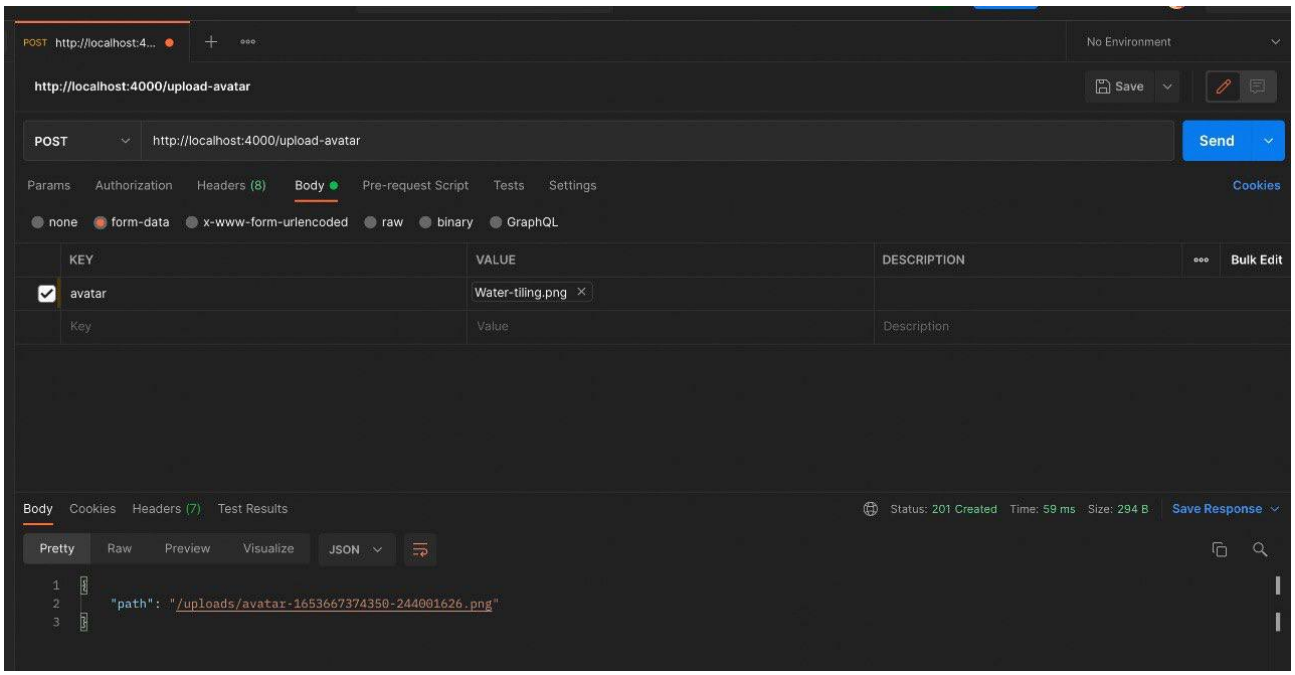

Figure 5.3: Avatar uploading

To create a room we use the event "rooms":

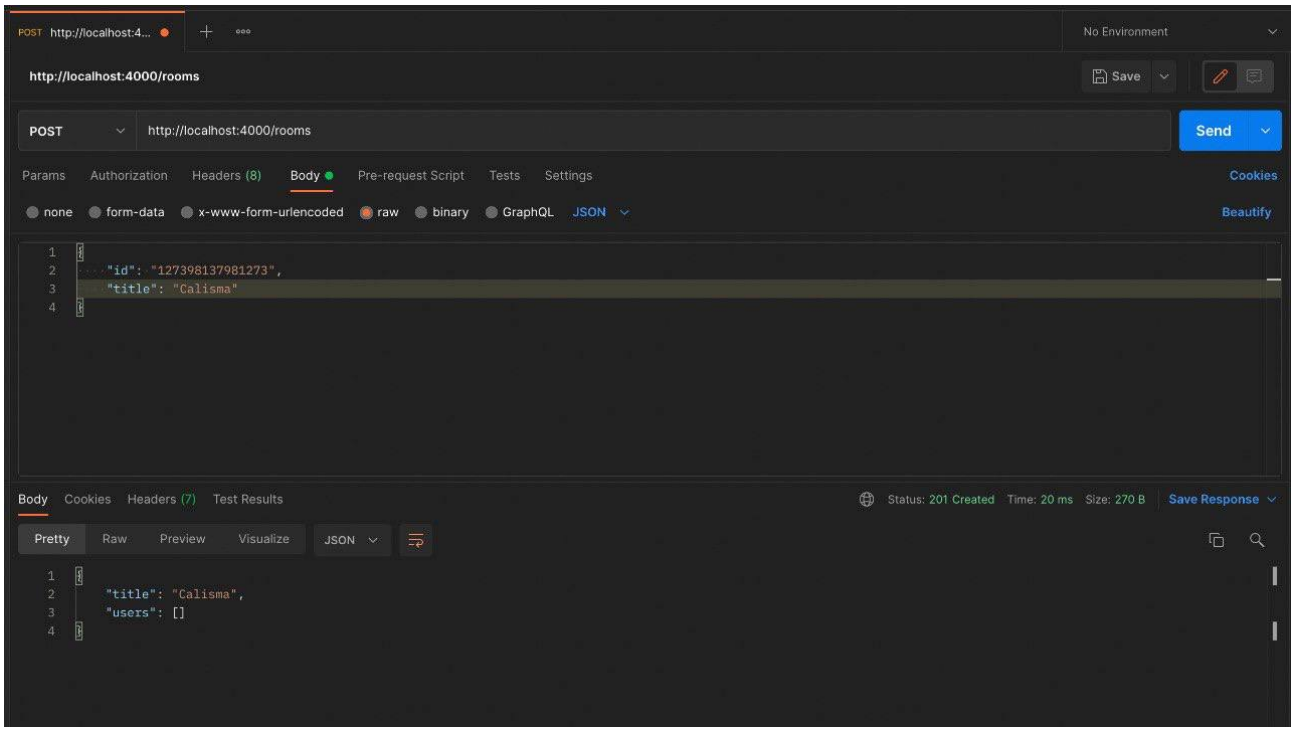

Figure 5.4: Creating room

To get information about a room we use event "rooms" + room id:

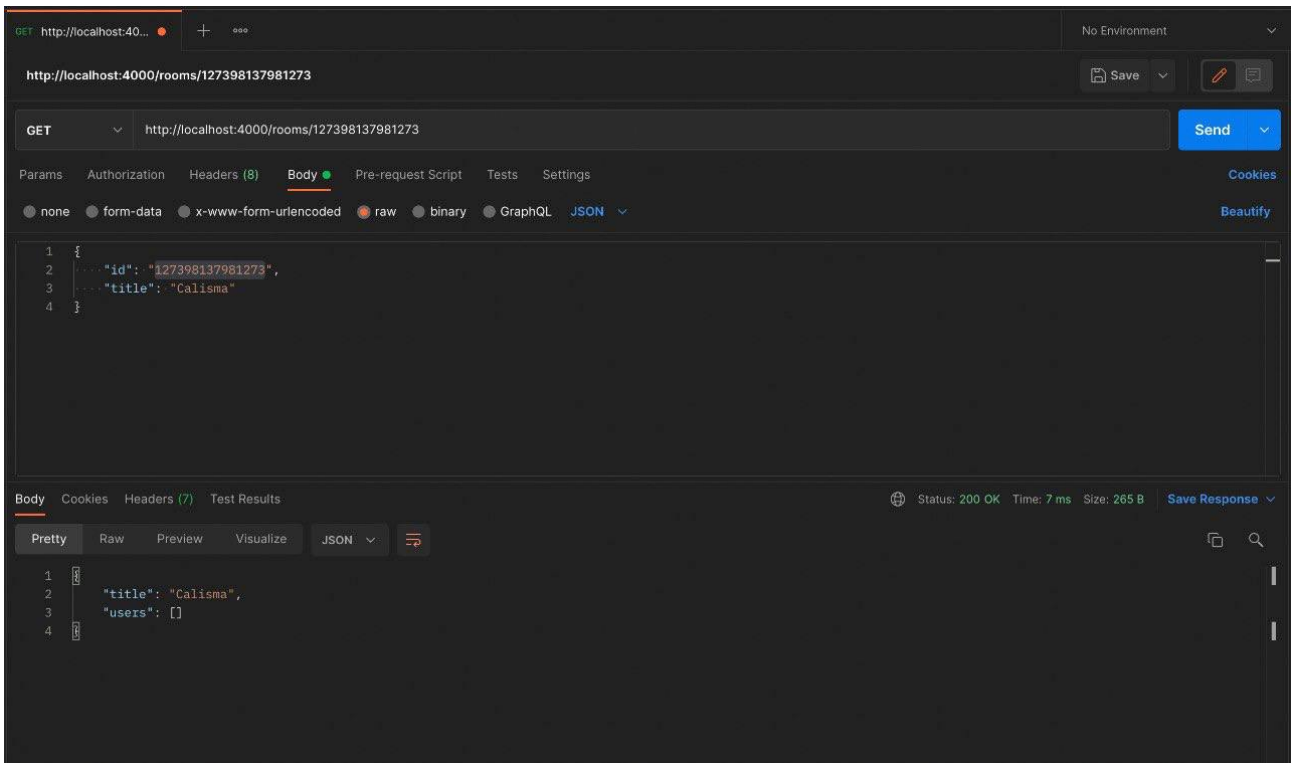

Figure 5.5: Getting information about a room by id

Chapter 6

Conclusion

Finally, based on our expertise, we have successfully designed a comprehensive simple and intelligible platform for online communication. Because our ChatRoom is so informative, beneficial, and handy for everyone, we feel that the market will always want such communication sites.

We thought about the idea and performed research from several angles to see how we might best implement the project and make it more accessible and clear.We detected all of the flaws through testing, started working on rectifying some data, and modified everything we needed in real time, all while using the best available practices that we were taught at the university. As a result, we attempted to design a basic system.

References

[1] *Chakra UI*. URL: https://chakra-ui.com/.

[2] *Node JS*. URL: https://nodejs.org.

[3] *Socket IO*. URL: https://socket.io.

[4] *Trello*. URL: https://trello.com/.

[5] UXPUB. *Создаем библиотеку компонентов в figma*. URL: https://ux.pub/editorial/sozdaiem-bibliotieku-komponientov-v-figma-chast1-tipoghrafika-tsvieta-i-stili-35d0/. (accessed: 11.09.2020).

[6] Майк Кон. *Agile: оценка и планирование проектов*. Альпина Паблишер, 2018. ISBN: 9785961452082.

[7] Юлия Сергеева. *Как COVID-19 изменил интернет и нас*. URL: https://www.web-canape.ru/business/kak-covid-19-izmenil-internet-i-nas-statistika-interneta-i-socsetej-posle-pandemii/. (accessed: 11.09.2020).