- Додавање основи на HTML и CSS основи за изглед на страната

- Sign in, Create Account, Sign in with Google Popup, Log Out имплементација за да може чатот да работи

- Log Out Методот го избира елементот на копчето со ID на "Log Out". Методот addEventListener прикачува слушател на настани на копчето што слуша за настан „click“ Функцијата што се извршува кога ќе се кликне на копчето започнува со signOut. Оваа линија на код го одјавува корисникот од веб-локацијата користејќи го методот Firebase Authentication signOut(). Методот .then() е ветување што се извршува кога операцијата за одјавување е успешна. Ако има грешка при операцијата за одјавување, методот .catch() ја евидентира грешката во конзолата користејќи console.log(error);. Преостанатите линии на код го ажурираат приказот на веб-локацијата со поставување на приказот на копчето „Sign In “ на „block“ и ресетирање на полињата за внесување е-пошта и лозинка на празни низи. Ова осигурува дека информациите за најавување на корисникот не се складираат откако ќе се одјават.

- Sign In (const email = document.getElementById('email').value и const лозинка = document.getElementById('pass').value. Овие линии код ги враќаат вредностите внесени во полињата за внесување e-mail и password. Методот signInWithEmailAndPassword(auth, email, password ) се користи за автентикација на корисникот со неговиот е-маил и password. Овој метод враќа ветување што се извршува кога операцијата за најавување е успешна. Ако операцијата за најавување е неуспешна, се извршува методот .catch() и се најавува порака за грешка во конзолата. Својството error.code се користи за одредување на типот на грешката што се случила, а функцијата ErrorHandler се повикува со соодветна порака за грешка. Пораката за грешка исто така се најавува на конзолата користејќи console.log(error). Ако операцијата за најавување е успешна, се извршува методот .then(). Овој метод го вади корисничкиот објект од објектот userCredential и го крие елементот со ID на „IsLogedIn“ користејќи document.getElementById('IsLogedIn').style.display = none;. Овој елемент обично се користи за да покаже дали корисникот е најавен или не.

- Sing Up ( Функцијата што се извршува кога ќе се кликне копчето започнува со const email = document.getElementById('CreateEmail').value, const password = document.getElementById('CreatePass').value, и const корисничко име = document.getElementById( 'корисничко име').value;. Овие линии на код ги враќаат вредностите внесени во полињата за внесување е-пошта, лозинка и корисничко име, const time = (new Date().getHours()<10?'0':'') + new Date().getHours() + ":" + (new Date().getMinutes()<10?' 0':'') + new Date().getMinutes(); линијата на кодот го поставува тековното време во одреден формат. Методот createUserWithEmailAndPassword(auth, email, password) се користи за креирање нова корисничка сметка со дадената е-пошта и лозинка. Овој метод враќа ветување што се извршува кога операцијата за создавање сметка е успешна. Ако операцијата за создавање сметка е неуспешна, се извршува методот .catch() и се најавува порака за грешка во конзолата. Ако операцијата за создавање сметка е успешна, се извршува методот .then(). Овој метод го враќа корисничкиот објект од објектот userCredential и создава нов запис во базата на податоци Firebase во реално време под јазолот "UserInfo" со е-пошта, лозинка и корисничко име на корисникот користејќи го методот set(). Методот set(ref(base data, 'messages/' + Date.now()),) се користи за додавање нова порака во јазолот „messages“ во базата на податоци Firebase во реално време што покажува дека корисникот се приклучил разговорот. Пораката го вклучува корисничкото име на корисникот, корисничкиот UID и тековното време. Ако има грешка при операцијата за креирање сметка, се извршува методот .catch(). Својството error.code се користи за одредување на типот на грешката што се случила, а функцијата ErrorHandler се повикува со соодветна порака за грешка. Пораката за грешка исто така се најавува на конзолата користејќи console.log(error).

- Sign in with Google Popup( Функцијата signInWithPopup() се повикува со аргументот на давателот, кој е пример на GoogleAuthProvider. Ова отвора прозорец каде што корисникот може да се најави со Google. Ако корисникот успешно се најави, методот .then() се повикува со аргумент за резултат, кој содржи информации за најавениот корисник. Функцијата потоа го извлекува токенот за пристап на корисникот и корисничките информации, вклучувајќи ја неговата е-пошта и име на прикажување, користејќи го методот credentialFromResult() и објектот result.user. Следно, функцијата добива референца за јазолот UserInfo во базата на податоци во реално време на Firebase користејќи го методот firebase.database().ref(). Потоа ги слуша промените во јазолот UserInfo користејќи го методот on() со настанот „value “. Функцијата за повратен повик проверува дали Uid на најавениот корисник веќе постои во јазолот UserInfo. Ако не постои, функцијата ја додава е-поштата на корисникот и името на прикажувањето во јазолот UserInfo користејќи го методот set() со нов единствен клуч генериран со помош на user.uid. Ако има грешка за време на процесот на најавување, методот .catch() се повикува со аргумент за грешка. Функцијата се справува со грешката така што ќе ја најави на конзолата и потенцијално ќе му покаже порака за грешка на корисникот.

- Поврзување со firebase и креирање нова апликација за бесплатно хостување

- Функционалности на горе наведените функции

- Остварување user да остане логиран ако страната се исклучи или рефрешне се додека самиот не се одлогира

- Проверка дали корисникот е моментално најавен со Google, а ако не е, го враќа неговото корисничко име од базата на податоци на Firebase. Првиот дел од кодот, кој започнува со if (user.displayName == null), е if statement кој проверува дали прикажаното име на корисникот е null, што значи дека не се најавени со Google. Ако прикажаното име е null, кодот создава променлива ref што укажува на локацијата на базата на податоци на Firebase каде што е зачувано корисничкото име на корисникот. Методот ref.on("value", function(snapshot) прикачува event listener на локацијата на ref базата на податоци и го враќа корисничкото име на корисникот од базата на податоци. Откако ќе се преземе корисничкото име, тој го ажурира елементот HTML со ID на корисникот со преземеното корисничко име користејќи го document.getElementById('user').innerHTML = username. Вториот дел од кодот, кој започнува со if (user.displayName != null), проверува дали прикажаното име на корисникот не е null, што значи дека тие се најавени со Google. Ако прикажаното име не е null, тој го ажурира елементот со ID на корисникот со прикажаното име на корисникот користејќи ја линијата код document.getElementById('user').innerHTML = user.displayName.

- Функцијата SendMsg() ги враќа внесените пораки од корисникот, името на испраќачот. Ако пораката е празна, функцијата се враќа и не прави ништо. Во спротивно, функцијата создава нова порака во базата на податоци со timestamp, вклучувајќи го името на испраќачот, содржината на пораката, единствениот кориснички ID на испраќачот и времето испратено. Функцијата потоа го брише полето за внесување.

- Функцијата reef.on("child\_added", function(snapshot) {} се извршува секогаш кога се додава нова порака. Таа зема snapshot од новиот child и ја проценува содржината на пораката за да одреди како да се прикаже во прозорецот за разговор. Го инкрементира counter кој понатаму се користи. Следно, кодот ја проценува содржината на пораката користејќи условни изјави. Ако пораката започнува со „data:“, кодот претпоставува дека е порака со слика и ја поставува променливата информации на HTML код за да ја прикаже сликата со името на испраќачот и времето испратено. Ако пораката е известување дека корисникот штотуку се приклучил на собата за разговор, кодот ја поставува променливата информации на HTML код за да се прикаже пораката за известување со името и времето на корисникот. Ако пораката е обична порака, кодот ги поставува информациите во HTML код за да се прикаже пораката со името и времето на испраќачот. По поставувањето информации врз основа на содржината на пораката, кодот го додава во прозорецот за разговор користејќи document.getElementById("chat").innerHTML += info;. Потоа ги повикува функциите LastTime и scrollBottom за да го ажурира времето прикажано до пораката и да се движи до дното на прозорецот за разговор. Генерално, овој код е одговорен за динамичко ажурирање на прозорецот за разговор со нови пораки додека се додаваат во базата на податоци.

- Од кога ќе се сетнат пораките во датабазата има уште една функција која ги чита пораките на промена. Се додадува која листа да слуша за промени, тие имаат опции кои се (child\_added, child\_removed, value, child\_changed. Во овој случај јас користам on(‘child\_added’), на секоја нова порака се чита само последната внесена порака.

- Feature за праќање слики file input кој слуша настан за кликнување на елемент со идентификација „FireInput“. Кога ќе се кликне, тој активира настан за кликнување на елемент со идентификација „fileInput“. Ова се користи за отворање на дијалог прозорецот за избирач на слики за да му се овозможи на корисникот да избере слика за поставување. fileInput.onchange овој код слуша настан за промена на елемент со id „fileInput“. Кога корисникот избира датотека, функцијата се активира. Ја повикува функцијата base64Url() и го чека резултатот користејќи го клучниот збор await. Добиената URL-адреса потоа се доделува на променлива со име url. Изјавата console.log() се користи за евидентирање на URL-то на конзолата за цели на отстранување грешки. Атрибутот src на елемент на слика со id „ModalImage“ е поставен на URL-то. Конечно, настанот за кликнување се активира на елемент со идентификација „ImagePublic“. Овој блок од код слуша настан за кликнување на елемент со id „sub“. Кога ќе се кликне, таа ја повикува функцијата set() од Firebase Realtime Database за да ги складира податоците за подигнатите слики во базата на податоци под нов детски јазол со временски печат како клуч. URL-адресата на сликата е зачувана под клучот „message“, корисничкото име на моментално најавениот корисник е зачувано под клучот „username“, тековното време се складира под копчето „timeSent“ и корисничкиот ID на моментално најавениот корисникот е зачуван под копчето „userUid“. Ако податоците се успешно зачувани во базата на податоци, се кликнува на копчето „dismiss“ за да се затвори модалот и вредноста на елементот за внесување датотека се брише. Ако се појави грешка при складирање на податоците, тие се евидентираат во конзолата.

- Управување со онлајн статусот на корисниците во базата на податоци во реално време на Firebase. Првиот дел од кодот го користи методот set() за запишување податоци во базата на податоци, конкретно во јазолот „OnlineUsers“ под uid-то на тековниот корисник. Го поставува корисничкото име, онлајн статусот и uid во базата на податоци. Следниот дел од кодот ги враќа податоците од јазолот „OnlineUsers“ во базата на податоци и ги слуша настаните што се додадени на детето. За секое додадено дете, проверува дали userUid на детето се совпаѓа со uid на тековниот корисник, а ако не се совпаѓа, го ажурира соодветниот HTML елемент за да го прикаже корисничкото име на детето и зелениот круг емотикони, што покажува дека тие се онлајн. Исто така, ја зголемува променливата OnlineUsersCount. Ако корисникот е на мобилен уред, кодот поставува слушател на настани за промени во состојбата на видливост на уредот. Ако состојбата на видливост е „hidden“, ги отстранува податоците на корисникот од јазолот „OnlineUsers“ во базата на податоци. Ако состојбата на видливост е „visible“, таа ги ажурира податоците на корисникот во базата на податоци со моменталното време и статусот „online“.Ако корисникот не е на мобилен уред, кодот поставува слушател на настани за настанот пред да се вчита, кој се активира кога прозорецот ќе се истовари. Во овој случај, ги отстранува податоците на корисникот од јазолот „OnlineUsers“ во базата на податоци. Кодот слуша за child\_removed настани од јазолот „OnlineUsers“ во базата на податоци. За секое отстрането дете, го ажурира соодветниот HTML елемент за да го прикаже корисничкото име на детето без емотиконите со зелени кругови, ја намалува променливата OnlineUsersCount и ги ажурира елементите на HTML што го прикажуваат бројот на онлајн корисници.

- Се користи set() метод за load-ње на пораките кога странат ќе се отвори прв пат.  
  
- scrollBottom() зема два аргументи, елемент и времетраење t. Го анимира scroll wheel-от на елементот така што тој се движи до дното на неговата содржина. За да се постигне овој ефект се користи методот jQuery .animate(). Функцијата ја зема висината на содржината (користејќи $(елемент)[0].scrollHeight) и го анимира својството scrollTop до таа вредност во одреденото времетраење t.

- LastTime(element) зема низа елементи и ги отстранува сите елементи освен последниот. Функцијата претпоставува дека последниот елемент во низата е најновата порака во разговорот и додава временска ознака на неа. Оваа функција се користи за да се осигура дека само најновата порака во разговор покажува временски печат.

- MessagePoP() додава визуелна „tail“ на најновата порака во разговорот. Оваа опашка помага визуелно да се разликува најновата порака од претходните. Функцијата ги избира сите елементи со id „right“ и ја додава класата „no-tail“ на сите елементи освен последниот. На последниот елемент во списокот му е додадена опашката користејќи CSS стилови.

- Во фајлот PrivateChat.js се содржат истите функции од горенаведените со иста функционалност.

- кодот кој ги селектира сите user-и е за справување со однесувањето кога корисникот ќе кликне на одреден корисник во листата на онлајн корисници исто така и за Групни чатови. Тоа е дел од апликација за разговор каде што корисниците можат да имаат приватни разговори меѓу себе. Прво, кодот додава слушател на настани со кликнување на сите елементи со ID „Корисник“. Кога корисникот ќе кликне на елемент со овој ID, кодот ги извршува следните функционалности:

Ја менува висината на прозорецот за разговор и ги криеира/прикажува соодветните влезни полиња за приватен или групен разговор.

Ги отстранува сите постоечки пораки за разговор од прозорецот за разговор.

Ако корисникот е на мобилен уред, тој активира настан за кликнување на елементот „ShowOnlineUsers“ кој прикажува листа на онлајн корисници.

Го поставува приказот на делот за информации за разговор да го блокира и го пополнува со копче за назад и името на корисникот со кој се разговара.

Го добива UID-от на корисникот со кој се разговара и создава референца на базата на податоци на Firebase за приватниот разговор помеѓу тековниот корисник и избраниот корисник.

Го добива името на тековниот корисник и иницијализира променлива бројач за да го следи бројот на прикажани пораки.

Поставува слушател за нови пораки додадени во референцата за базата на податоци за приватни разговори.

Кога се додава нова порака, кодот го зголемува бројачот, ги добива информациите за пораката и создава HTML елементи за прикажување на пораката во прозорецот за разговор. Пораката може да биде текстуална порака или слика. Елементите на HTML се додаваат во прозорецот за разговор користејќи го својството innerHTML.

Функцијата LastTime() се повикува да го ажурира временскиот печат на последната порака прикажана во разговорот.

Функцијата scrollBottom() се повикува да го прелистува прозорецот за разговор до дното за да се видат најновите пораки.

Ако бројачот е 1 (т.е. се прикажува првата порака), центрифугата за вчитување е скриена.

Функцијата MessagePoP() се повикува да прикаже скокачко известување доколку корисникот прими нова порака додека моментално не гледа во прозорецот за разговор.

- document.querySelector('#send-private').addEventListener('click', SendMsgPrivate) оваа линија на код го избира елементот DOM со ID на "send-private" и додава слушател на настани за кликнување на него, кој ќе ја повика функцијата SendMsgPrivate кога ќе се кликне копчето. Функцијата SendMsgPrivate е дефинирана за да се справи со испраќање приватна порака. Кога корисникот ќе кликне на копчето „send-private “ или ќе го притисне копчето „Enter“, оваа функција ќе се повика. Функцијата прво ја враќа содржината на пораката од полето за внесување со ID на „send-private “, како и името на испраќачот и тековното време. Потоа функцијата проверува дали пораката е празна. Ако е празна, функцијата се враќа и не прави ништо. Ако пораката не е празна, функцијата создава нова порака во базата на податоци на Firebase со timestamp. Ги проверува корисничките ID за да ја одреди патеката за зачувување на пораката, а потоа создава нов запис во базата на податоци „PrivateChats“ со содржината на пораката, името на испраќачот, корисничкиот ID и времето кога е испратена пораката. Кодот проверува дали user.uid е поголем од ChatWithUid, што значи дека тековниот корисник е тој што го иницира разговорот за приватен разговор. Ако ID на тековниот корисник е поголем од ID на другиот корисник, патеката за зачувување на пораката во базата на податоци на Firebase се конструира со користење на ID на тековниот корисник прво проследено со ID на другиот корисник. Ако не, патеката се конструира со користење на ID на другиот корисник прво проследено со ID на тековниот корисник.

- Следниот код додава event listener на елемент со ID на „BackToPublic“. Кога ќе се кликне на овој елемент, функцијата што следи ќе се изврши. Преглед на она што го прави функцијата:

1. document.querySelector('#chat').style.height = window.innerHeight - 150 +'px' oваа линија ја поставува висината на елементот со ID на „chat“ да биде висината на прозорецот минус 150 пиксели.
2. document.querySelector('.parent-input-group').style.display = 'none' оваа линија крие елемент со класа „родител-влезна група“.
3. document.querySelector('.parent-input-private').style.display = 'none' Оваа линија крие елемент со класа „родител-влез-приватно“.
4. document.querySelector('.parent-input').style.display = 'block' оваа линија покажува елемент со класа на "родител-влез".
5. if (document.querySelector('#chat').hasChildNodes()) {} oвој код проверува дали елементот со ID на „chat“ има детски јазли.
6. document.querySelectorAll('#chat .imessage').forEach(element => { element.remove();})Оваа линија ги отстранува сите елементи со класа на „порака“ што се деца на елементот со ID на „chat“.
7. document.querySelectorAll('#chat #selector').forEach(element => { element.remove(); }) оваа линија ги отстранува сите елементи со ID на „selector “ што се деца на елементот со ID на „chat“.
8. firebase.database().ref('messages').on('child\_added', (snapshot) овој код ги слуша новите детски јазли кои се додаваат на патеката на базата на податоци во реално време на Firebase наречена „messages“. Кога ќе се додаде нов детски јазол, ќе се изврши функцијата во функцијата со стрелки.

- Остатокот од кодот во функцијата создава нов HTML елемент врз основа на податоците во новиот детски јазол и го додава на елементот „chat“. Исто така, го прелистува елементот „chat“ до дното и додава дополнителна функционалност на новите елементи што штотуку беа креирани.

- Кодот за додавање нова група започнува со создавање нова променлива која прави стринг од UIDs кои се одделени со , и празно место. Овој код се врти низ сите полиња за избор со #UserGroup ID и го додава UID на секој штиклиран корисник на променливата StringArray. UID на корисникот се извлекува од родителскиот јазол на полето за избор и се додава во StringArray заедно со запирка и празно место. Овој код проверува дали StringArray е празен. Ако е така, функцијата се враќа и не создава нова група. Ако StringArray не е празен, кодот проверува дали е внесено име на група. Ако не е внесено име на групата, променливата GroupName е поставена на „No Name Group“. Во спротивно, GroupName е поставена на вредноста внесена во полето за внесување „Име на групата“.Овој код создава нова група во базата на податоци на Firebase со единствена патека конструирана од споената низа кориснички UID-и складирани во променливата StringArray. Името, корисниците и својствата на креаторот се поставени на името на групата, поврзаната низа на кориснички UID-и и UID на тековниот корисник, соодветно.