**UNIVERSITETI I PRISHTINËS “HASAN PRISHTINA”**

**FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE**

**Departamenti i Inxhinierisë Kompjuterike**



**PUNIM SEMINARIK**

Lënda :**Sisteme të shpërndara**

Tema :**Sistemi për menaxhimin e çerdhës së fëmijëve**

**Grupi 5**

**Prishtinë, korrik 2021**

**Menaxhimi i Dokumentit**

**Mësimdhënësi:** Isak Shabani

**Asistent:** Vigan Abdurrahmani

**Ky dokument është punuar nga:**

1. Lirim Muçolli
2. Natyra Vitija
3. Rrezarta Cokli
4. Samira Jashanica
5. Tahir Temaj
6. Teuta Ukshini
7. Toni Krasniqi
8. Valtrin Shala

**Për krijimin e dokumentit janë përdorur veglat:**

Nodejs

Visual Studio Code

Google Docs

Microsoft Word

**Programi është implementuar me:**

Javascript

Nodejs

Vuejs

Database: **Firebase**

Përmbajtja

[**1. Hyrje**](#_txxuuag8z116) **4**

[**2.Përshkrimi i projektit**](#_78uiv2hkwnn) **5**

[**3. Implementimi i sistemit**](#_kuinjyoz6flm) **6**

[3.1. Dashboard](#_2itn7y110uy) 9

[3.2. Students](#_wimstcflm3z) 10

[3.3. ChatRoom](#_al23belg8utk) 11

[**4. Implementimi i databasës- Firebase**](#_y8k13rdhgpxb) **19**

[**5. Llojet e Komunikimit**](#_o74rd3t96gxb) **24**

[**6. Implementimi në Kod**](#_vkqfu5xltxhr) **25**

[6.1. Serializimi dhe Deserializimi](#_pcowvor6puhz) 30

[**7. Rreth projektit**](#_2sybc9foa9ct) **35**

[**8. Konkluzioni**](#_f9lke2tdjhmr) **35**

# **1. Hyrje**

Një sistem i shpërndarë është një mjedis informatik në të cilin përbërës të ndryshëm përhapen nëpër shumë kompjuterë (ose pajisje të tjera informatike) në një rrjet. Këto pajisje e ndajnë punën, duke koordinuar përpjekjet e tyre për të përfunduar punën në mënyrë më efikase sesa nëse një pajisje e vetme do të kishte qenë përgjegjëse për detyrën.

Sistemet e shpërndara janë një zhvillim i rëndësishëm për IT dhe shkencën kompjuterike pasi një numër gjithnjë e më i madh i punëve të ndërlidhura janë aq masivë dhe kompleksë sa do të ishte e pamundur që një kompjuter i vetëm të merret me to vetëm. Por llogaritja e shpërndarë gjithashtu ofron avantazhe shtesë në krahasim me mjediset tradicionale të llogaritjes.

Sistemet e shpërndara kanë evoluar me kalimin e kohës, por implementimet më të zakonshme të ditëve të sotme janë krijuar kryesisht për të funksionuar përmes internetit dhe, më konkretisht, cloud. Një sistem i shpërndarë fillon me një detyrë, siç është pasqyrimi i një videoje për të krijuar një produkt të gatshëm gati për t'u lëshuar.

Sistemet e shpërndara zvogëlojnë rreziqet që kanë të bëjnë me një pikë të vetme të dështimit, duke forcuar besueshmërinë dhe tolerancën ndaj defekteve. Sistemet moderne të shpërndara zakonisht janë të dizajnuara për t'u shkallëzuar në kohë reale; gjithashtu, mund të rrotullohen burime shtesë informatikë në fluturim, duke rritur performancën dhe duke ulur më tej kohën deri në përfundim.

Sistemet e para të shpërndara ishin të përhapura ne rrjetet e zonës lokale si Ethernet, i cili u shpik në vitet 1970. Sot sistemet e shpërndara gjejnë zbatim pothuajse në cdo sferë duke na mundësuar që shumë procese dhe punë t’i kryejmë më lehtë, shpejt dhe sigurt.

# 

# **2.Përshkrimi i projektit**

Ky raport në lënden “**Sisteme të shpërndara**” është një projekt grupor që ka të bëjë me krijimin e një sistemi për menaxhimin e çerdhes së fëmijëve.

Gjatë punimit të këtij projekti synim ka qenë qasja dhe administrimi në sistem. Ky sistem mundëson që menaxhimi i çerdheve të jetë sa më efikas.Pra punonjësit e çerdheve do kenë mundësi të publikimit të çdo aktiviteti, e edhe prindërit njëkohësisht do të jenë të informuar në çdo qast rreth fëmiut të tyre. Për mundësinë e komunikimit online edukator-prind,është bërë inkorpurimi i chatit përmes socketave në serverë të ndryshëm.

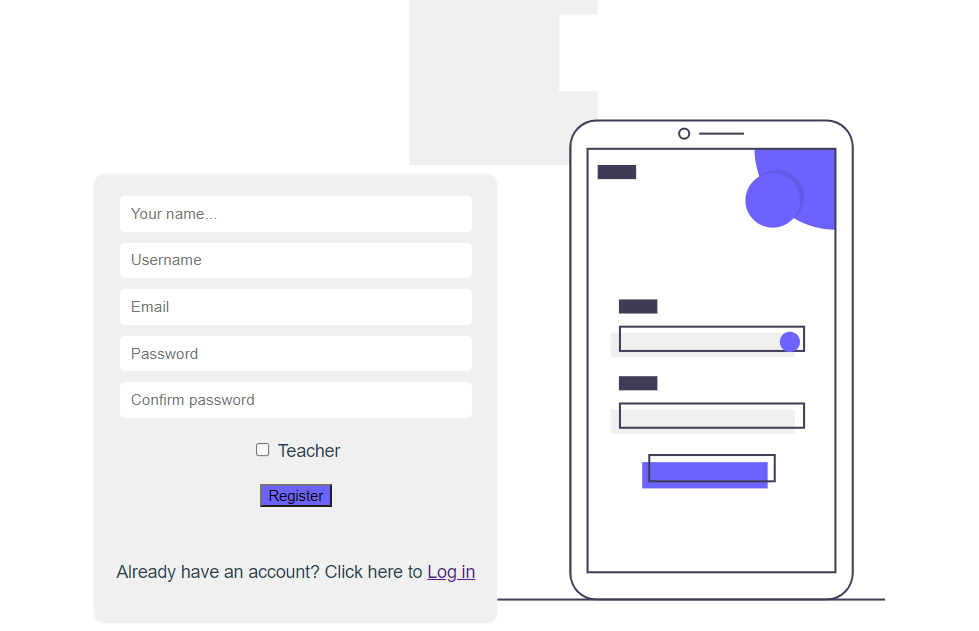
Detyra e dhënë është si vijon:

Të zhvillohet një sistemi i shpërndarë për menaxhimin e çerdhes se fëmijëve. Sistemi duhet të ketë të realizuar dhe inkorporuar edhe një Chat duke përdorur (TCP ose UDP socket-e) Client/Server për komunikim në mes të shfrytëzuesve të sistemit (komunikimi duhet të mundësohet me tekst, zë dhe video). Serializimin dhe deserializimin e objekteve që do të i përdorni në projekt i realizoni përmes teknologjive: XML, JSON, Binary në Java dhe NodeJS.

# **3. Implementimi i sistemit**

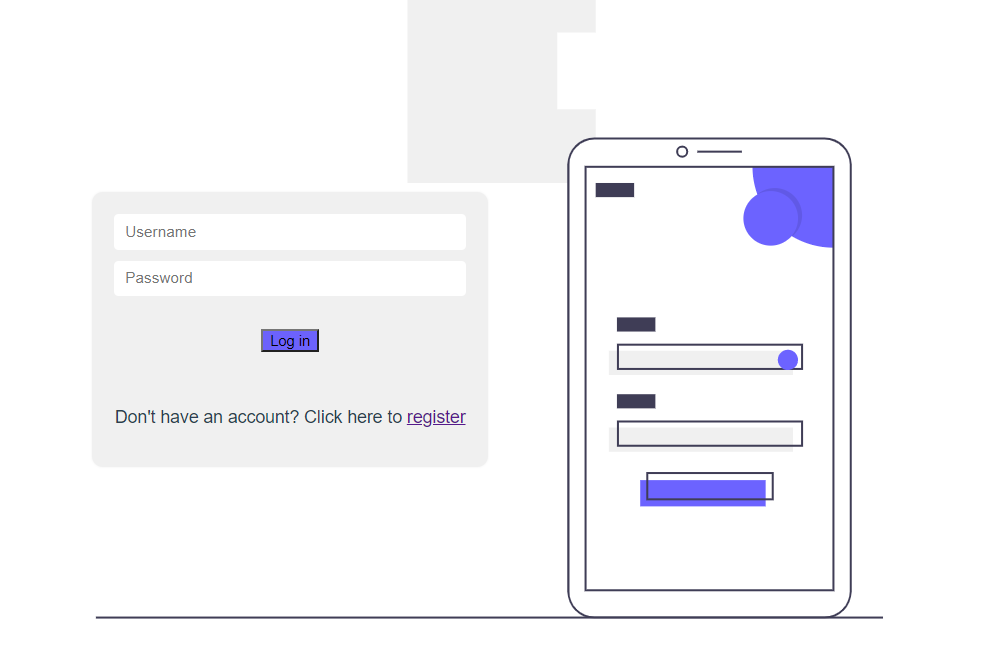
Sistemi i shpërndarë për menaxhimin e çerdhes se fëmijëve është i implementuar në localhost si dhe ne portin 8080 dhe duket si ne vijim:

* Përmban opsionin për regjistrim



*Fig.1. Regjistrimi i usereave në platformë*

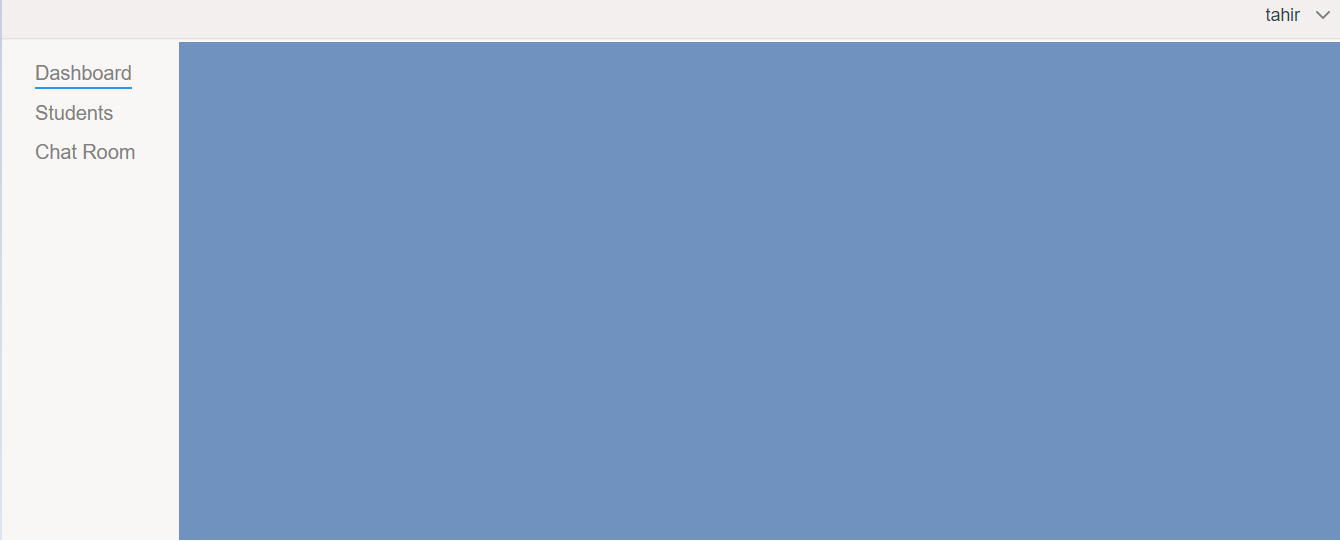
* Si dhe për log-in



*Fig.2. Kycja në sistem përmes loginit*

Në kuadër të kyçjes së userit shfaqet pjesa kryesore home e hostuar në localhost,e cila përmban opsionet si:

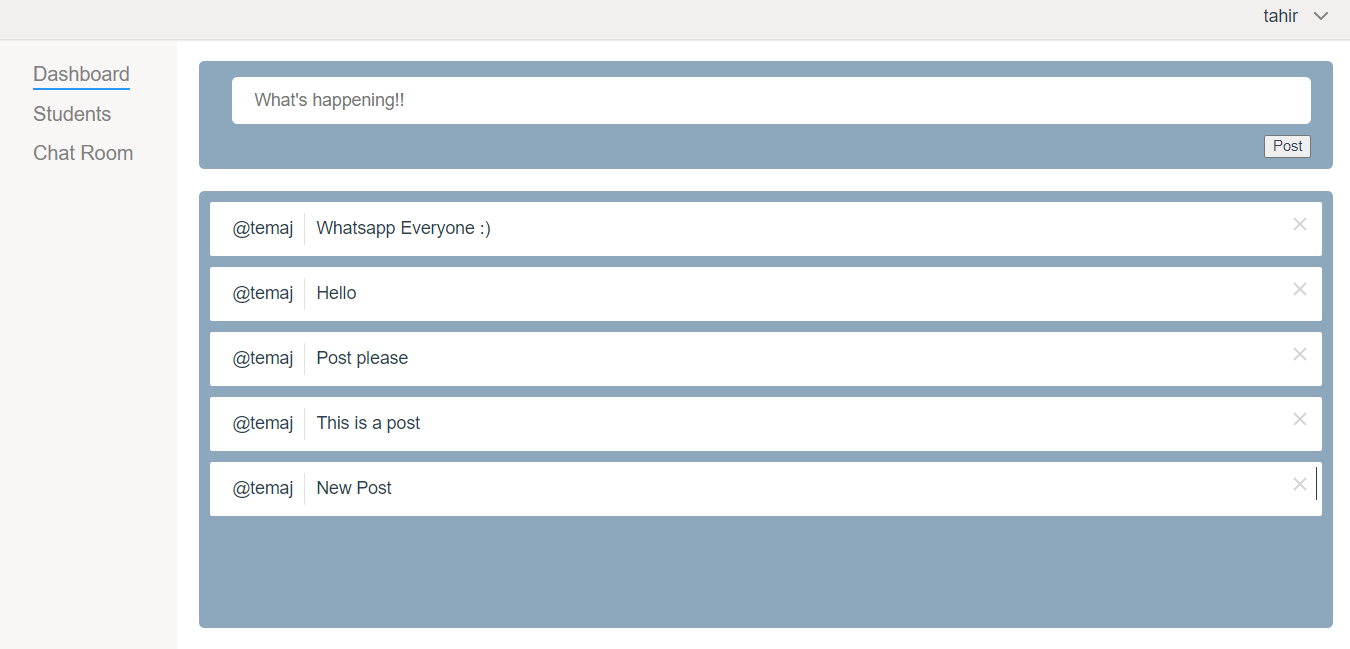
* **Dashboard** - përfshin pjesën e postit, ku prindërit mund të shohin çdo aktivitet që bejnë fëmijët e tyre në bazë të postimeve nga faqja jonë. Secili user i kyçur në platformen tonë ka të drejtën të postoj, gjithçka që i doket atij interesante dhe argëtuese rreth çerdhes sonë
* **Students** - Në kuadër të regjistrimit të userit kemi dhënë mundësinë që useri adekuat të jetë nxënës/student apo mësues, prandaj tek tabela e students shfaqen vetëm userat e regjistruar në opsionin adekuat. Pra tek students regjistrohen të gjithë userat që janë aktualisht pjesë e çerdhes sonë dhe vijojnë klasën e tyre përkatëse, por edhe ata që duan të jenë pjesë e çerdhes sonë (nxënës te rinj).
* **ChatRoom -** përfshin pjesën e bashkëbisedimit edukator - prind për të gjitha aktivitetet që ndodhin gjatë periudhës së qëndrimit të fëmijëve në çerdhen tonë. Komunikimi mundëson një afërsi dhe lehtësim të madh per prindërit sepse bën që edhe ata të jenë te knaqur dhe të sigurt në punën që ne bejmë. Kjo e bën platformën tonë adekuate për të menaxhuar një sistem të tillë.



*Fig.3.Hyrja në sistem në kuadër të home*

## **3.1. Dashboard**

Më poshtë është pjesa në të cilën kemi opsionin për postim, ku shfaqën të gjitha postimet e mëparshme por edhe ato të rejat që sapo krijohen. Opsion tjetër është edhe fshierja e një postimi të bërë më parë, por që nuk na duhet më.

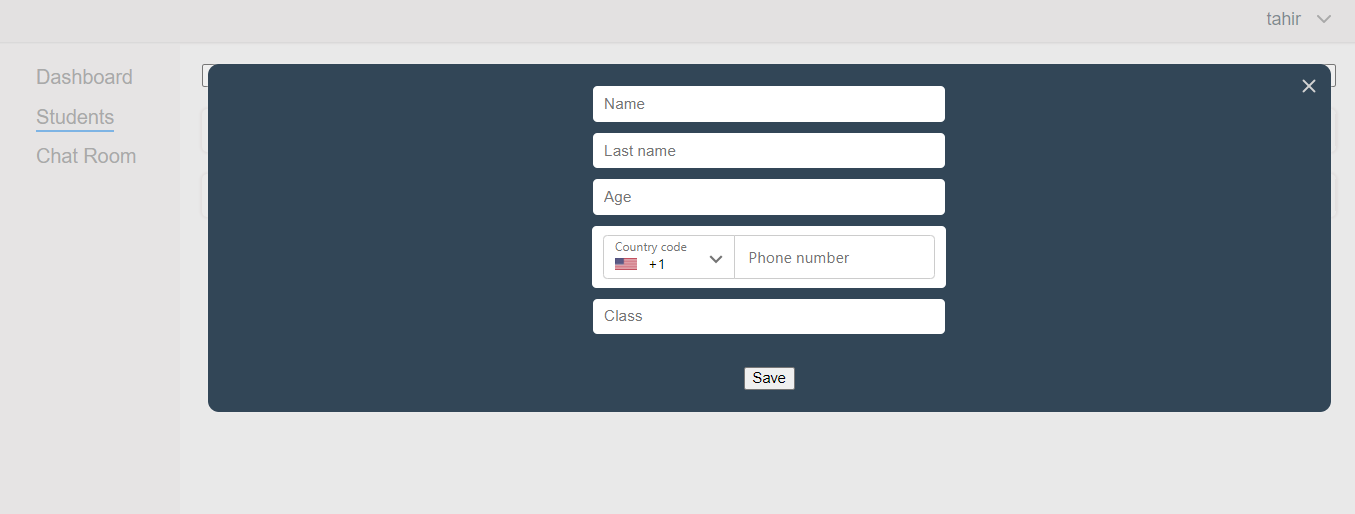


*Fig.4. Përmbajtja e Dashboard-it dhe postimet*

Parakusht për të bërë postime është që përdoruesit të jenë të loguar më parë, ku përdoruesit tjerë shohin emrin e atij që ka postuar dhe përmbajtjen e postimit. Në qoftë se postimi nuk permban material edukativ ose diçka që nuk ka lidhje me platformën, atëherë kemi mundësinë të fshijmë postimin.

## **3.2. Students**

Opsioni Students mundëson shtimin e studentave në platformën tonë dhe menaxhimin e të dhënave të tyre. Nëse studenti për ndonjë arsye vendos të largohet nga çerdhja jonë kemi opsionin për të bërë fshirjen e të gjitha të dhënave për studentin në fjalë. Po ashtu, në qoftë se në kohën e regjistrimit ne kemi marrur të dhenat e studentit, por studenti ka ndryshuar vendbanim ose numër telefoni ose çkado qoftë informatë kemi opsionin e updatimit në lidhje me atë student. Kjo mundëson siguri të të dhënave dhe informim sa më të saktë për studentët tanë.

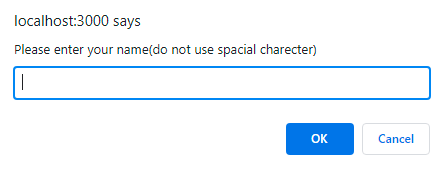


*Fig.5.Shtimi i studentëve në sistem*

## **3.3. ChatRoom**

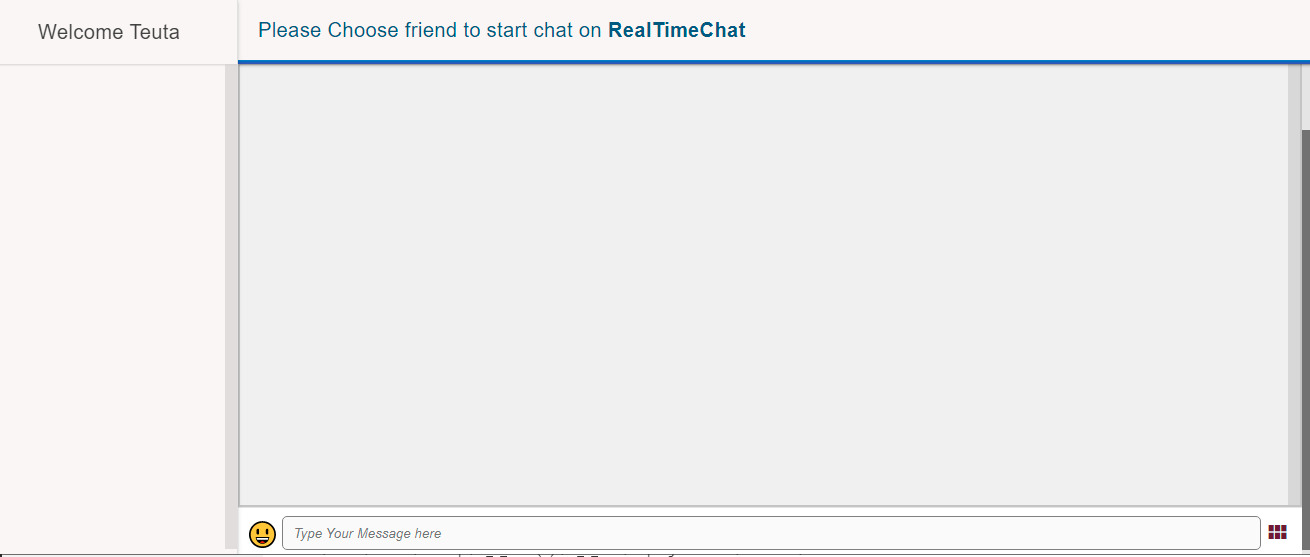
Një sistem i menaxhimit të çerdheve ndihmon për të komunikuar në mënyrë efikase, me mbështetjen e teknologjive. Të gjithë janë të lidhur përmes aplikacioneve për celularë ose aplikacioneve në internet dhe çdo informacion që duhet të kalojë mund të dërgojë shpejt duke përdorur Chat/SMS. Kjo shmang një numër të konsiderueshëm të thirrjeve telefonike dhe gjithashtu kursen kohë.

Chati jonë është i hostuar në localhost me port 3000 dhe është implementuar me anë të socket.io,e cila mundëson komunikim të bazuar në ngjarje bidirectionale(dy drejtime) klient-server në kohë reale. Ky punon në çdo platformë, shfletues(browser) ose pajisje, duke u përqëndruar në të njëjtën mënyrë te besueshmëria dhe shpejtësia. Kjo do të thotë që serveri mund të dërgojë mesazhe te klientët.

Në kuadër të ekzekutimit shfaqet konsola që na mundëson të jemi pjesë e chatit,si në pamjen më poshtë:

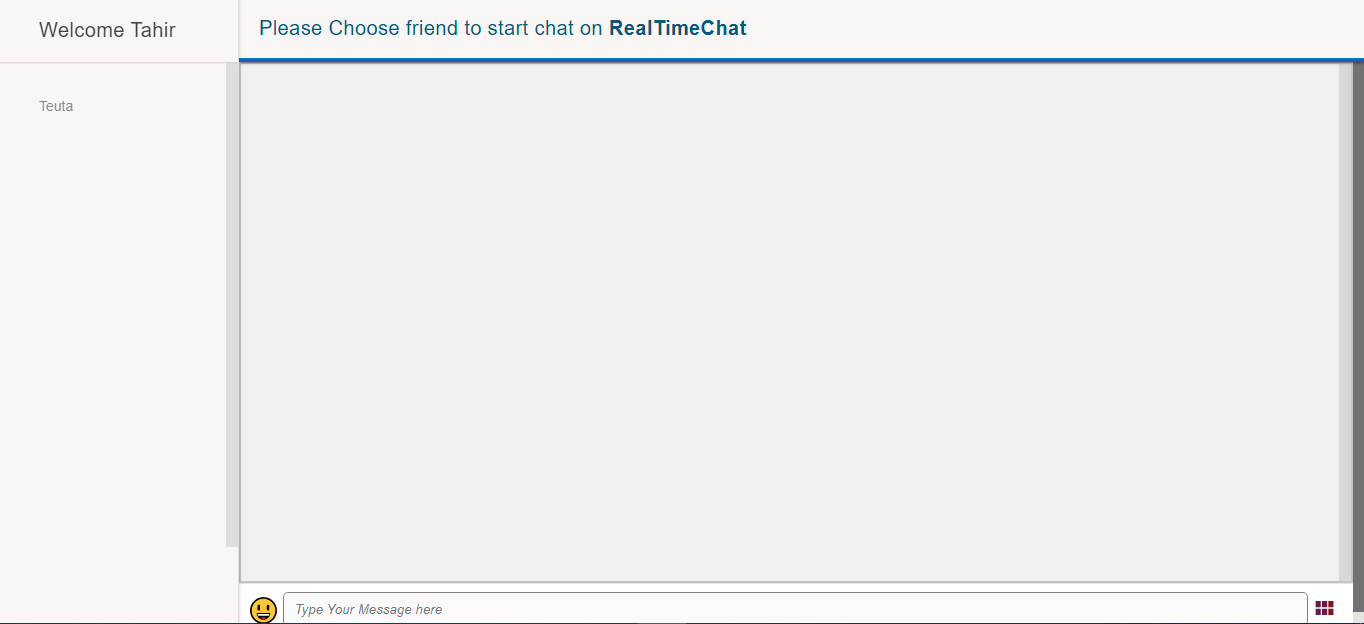
*Fig.6. Konsola në localhost të chatit*

Pasi shënojmë emrin tonë, kështu duket pamja e chatit. Si dhe kërkesa për të zgjedh një mik për bisedë.



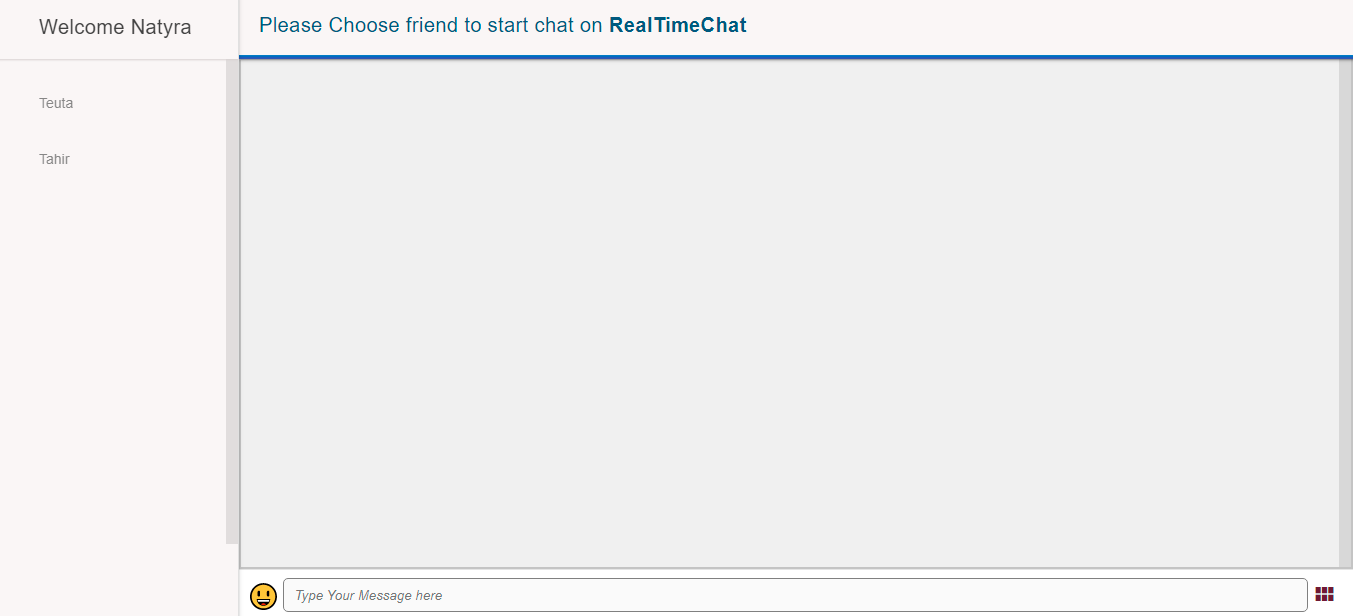
*Fig.7. Pamja e chatit ne RealTime*

Pas zgjedhjes së së bashkëbiseduesit tonë chati i tij do të duket kështu:



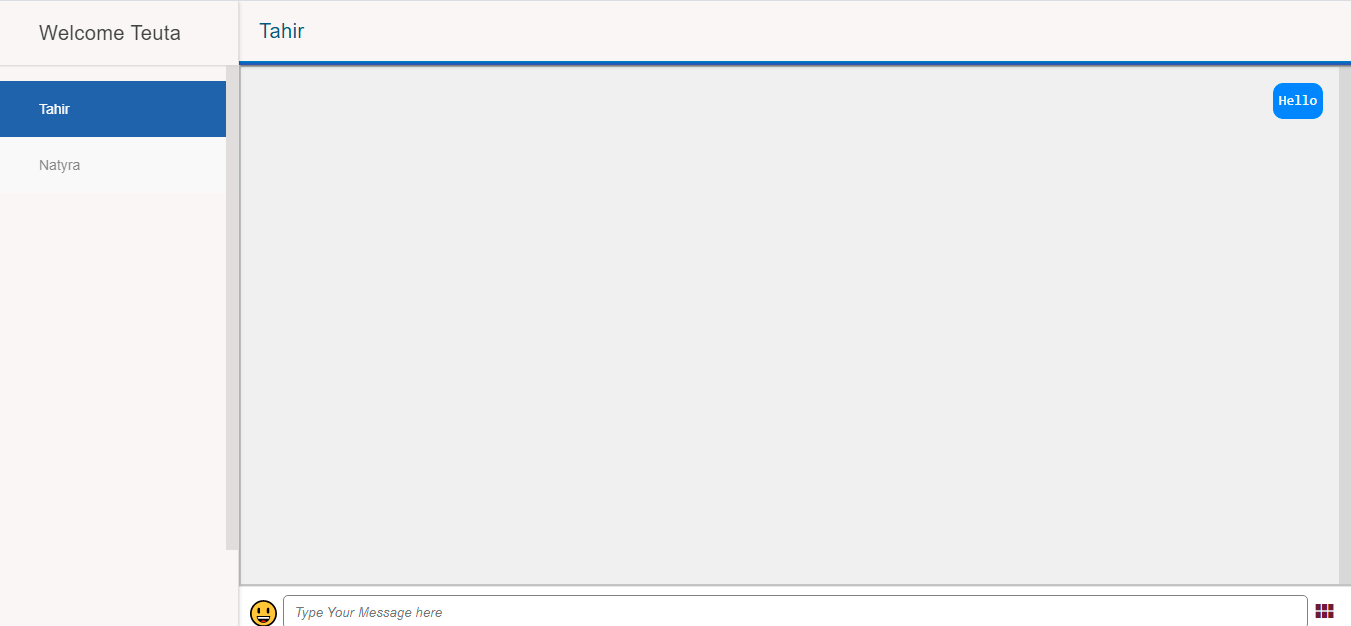
*Fig.8. Pamja nga Chati i Bashkebiseduesit*

Chati mundëson që bëhet komunikimi me më shumë se një person, pra nëse duam te flasim me dy ose më shumë persona njëkohesisht, atëherë si në pamjen në vijim shfaqen emrat e personave që ne duam të komunikojmë.



*Fig.9.Bashkëbisedimi me shumë persona*

Dërgimi i mesazhit bëhet njëjtë për secilin user (përdorues). Kjo varet se cili user do të komunikoj me tjetrin në kohen e caktuar.



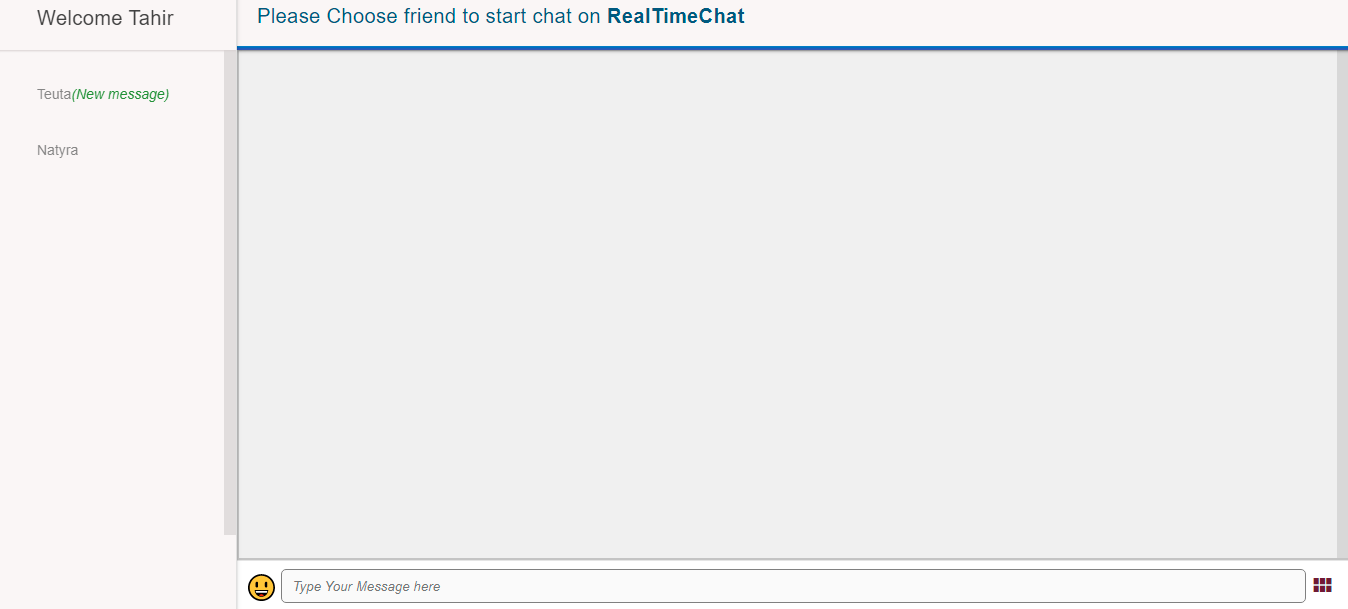
*Fig.10. Dërgimi i mesazhit*

Dërgimi i mesazhit shoqërohet me një ringtone, me anë të shtyrjes së komandës Enter dëgjohet një zhurmë që e informon bashkëbiseduesin tjetër që dikush të ka dërguar një mesazh.



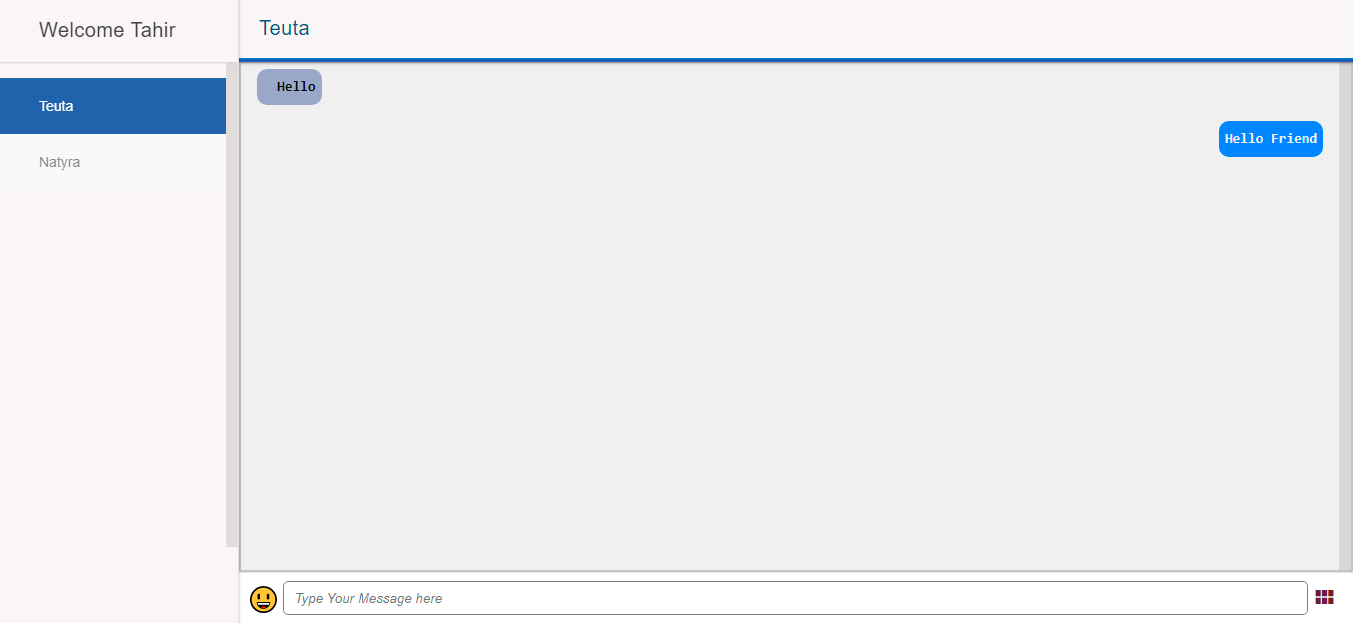
*Fig.11. Download-imi i fajllit mp3*

Pas dërgimit të mesazhit nga useri (përdoruesi) tek bashkëbiseduesi, në chatin e tij ju shaqet një notification( informim) që ke një mesazh të ri.



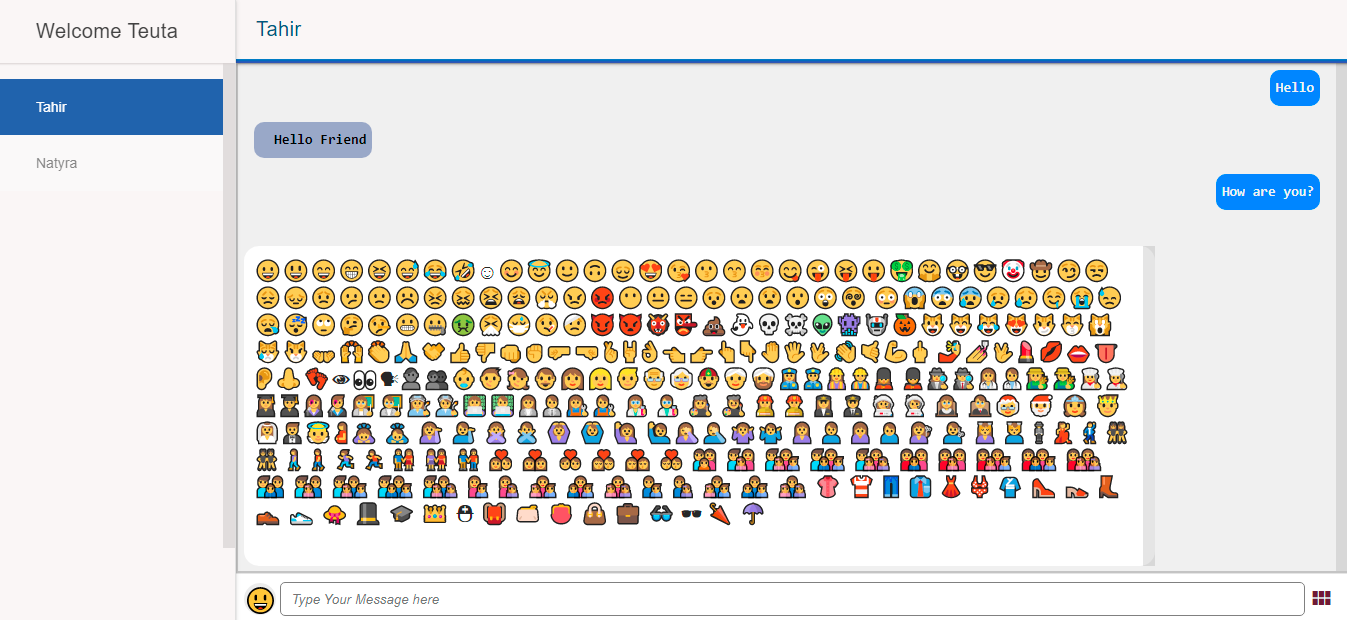
*Fig.12. Informimi për mesazh të ri*

Dhe kështu fillon bashkëbisedimi



*Fig.13. Bashkëbisedimi*

Po ashtu mundësohet edhe dërgimi i emoji (figurave) në bashkëbisedim



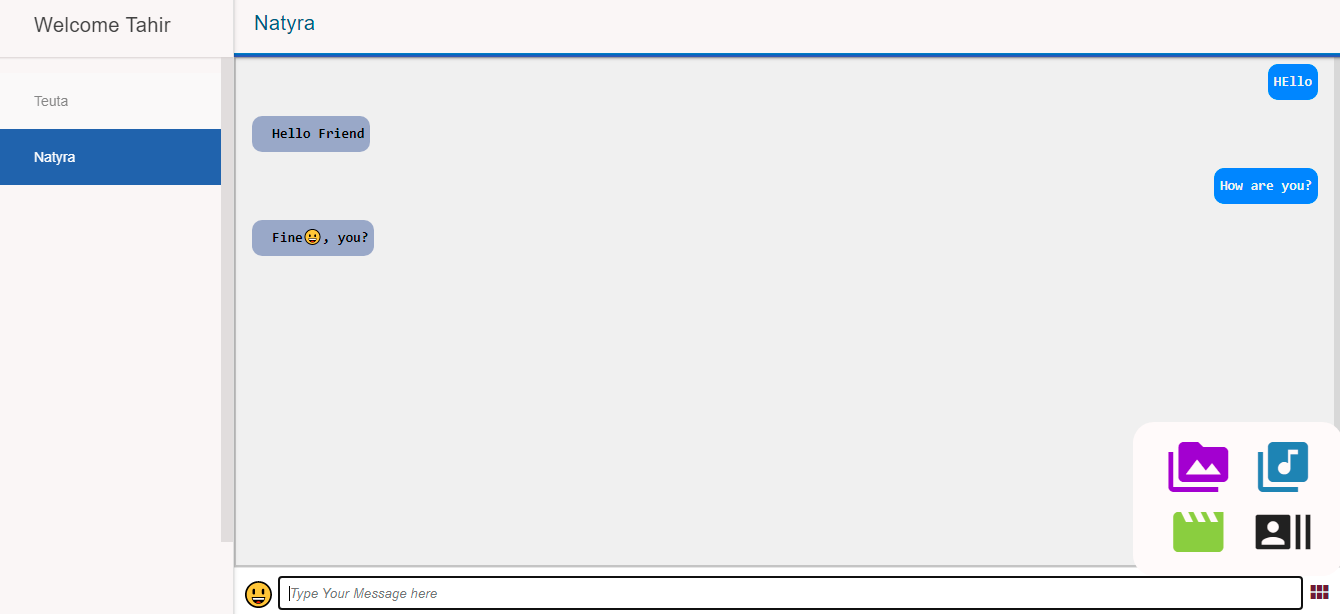
*Fig.14. Emoji-t*

Komunikimi me anë të emoji që të bëhet sa më atraktiv.

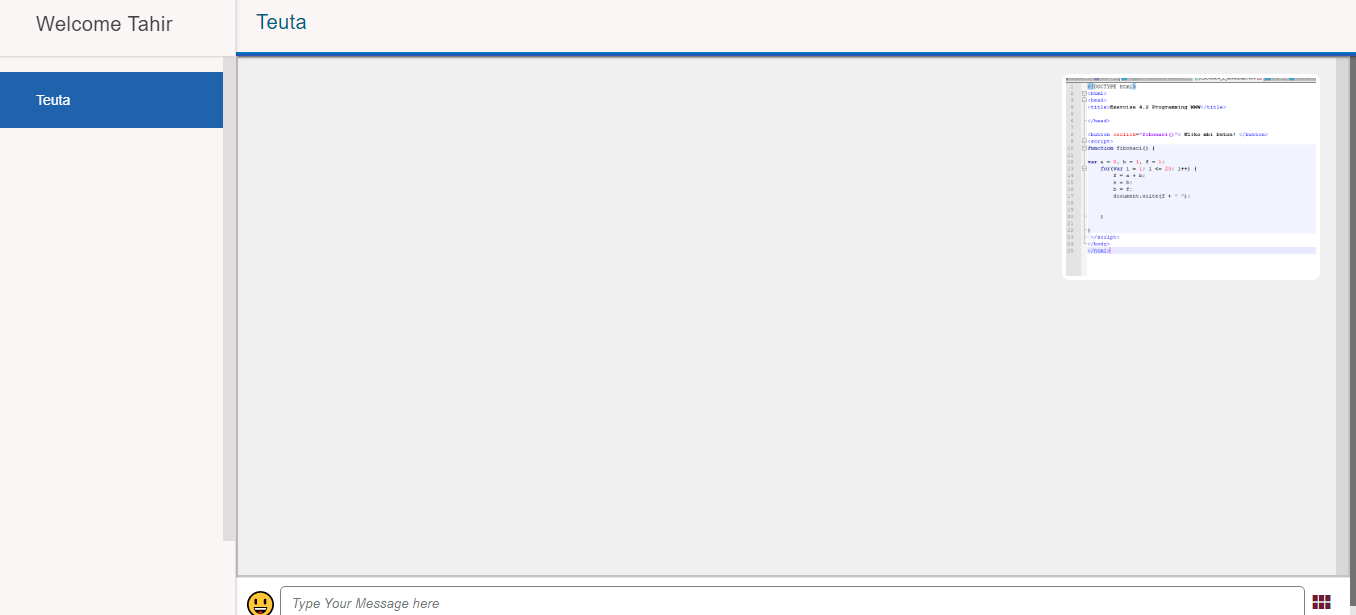


*Fig.15. Komunikimi me emoji*

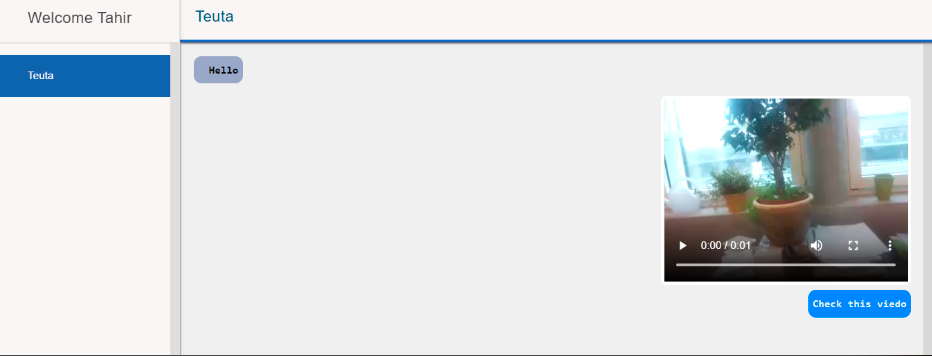
Chati mundëson edhe upload-imi e fajllave si foto,pdf, video dhe audio



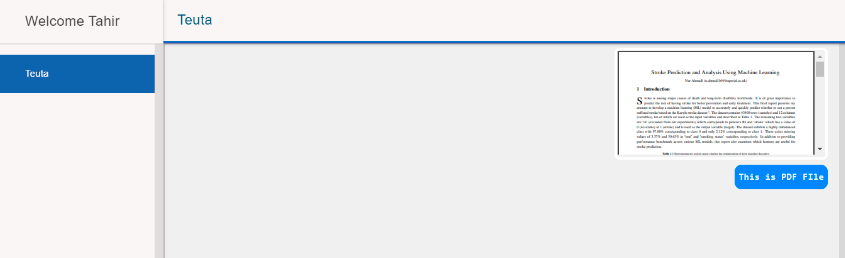
*Fig.16. Butoni për upload-im*



*Fig.17. Dërgimi i fajllave img*



*Fig.18. Dërgimi i fajllave .mp4*



*Fig.19. Dërgimi i fajllave .pdf*



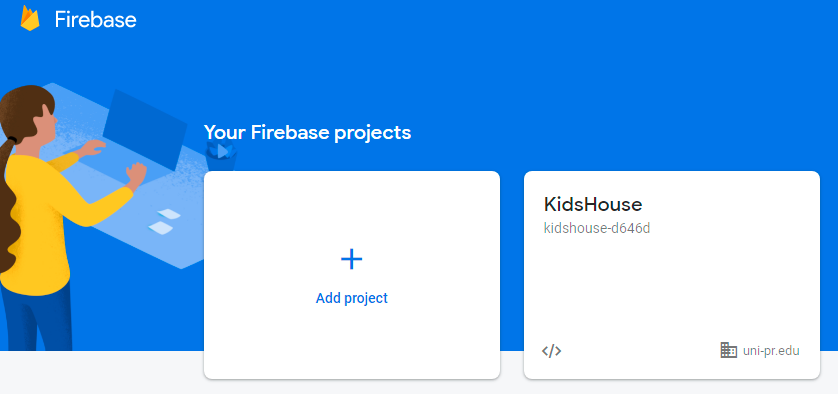
*Fig.20. Dërgimi i ikonave*

# 4. Implementimi i databasës- Firebase

Firebase është një program i zhvillimit të aplikacioneve i mbështetur nga Google që u mundëson zhvilluesve të zhvillojnë aplikacione IOS, Android dhe Ueb. Firebase ofron mjete për gjurmimin e analizave, raportimin dhe rregullimin e ndërprerjeve të aplikacioneve, krijimin e marketingut dhe eksperimentin e produkteve. Firebase Hosting punon jashtë kutisë me shërbimet Firebase, duke përfshirë funksionet e Cloud, vërtetimin, baza e të dhënave në kohë reale, Cloud Firestore dhe Cloud Messaging. Ju mund të ndërtoni mikrosherbime dhe aplikacione të fuqishme në internet duke përdorur këto shërbime plotësuese Firebase

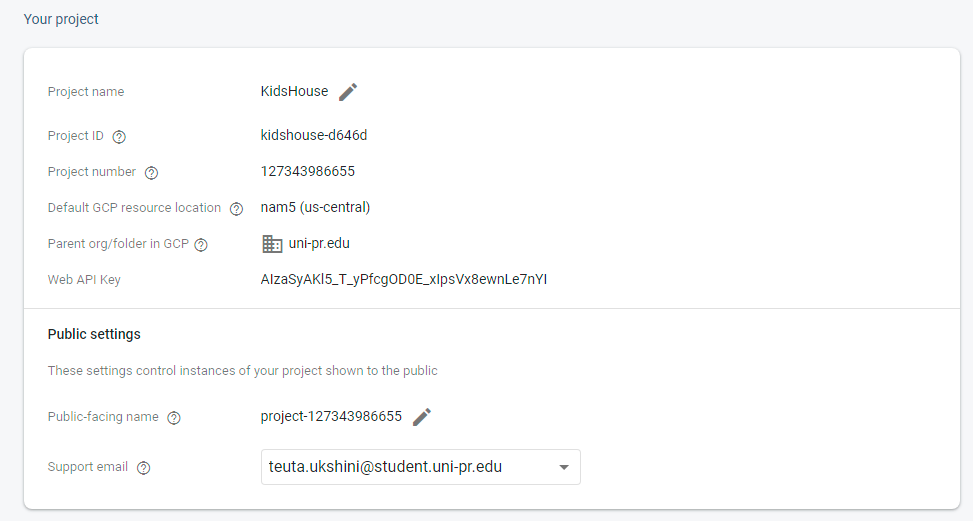
Firebase është platforma e zhvillimit të aplikacioneve mobile të Google që ju ndihmon të ndërtoni, përmirësoni dhe rritni aplikacionin tuaj. Firebase është platforma e zhvillimit të aplikacioneve mobile të Google që ju ndihmon të ndërtoni, përmirësoni dhe rritni aplikacionin tuaj.

Startimi i Firebase-it tonë është bërë me emërtimin KidsHouse



*Fig.21.Krijimi i Firebase*

Krijimi i projektit tonë me detajet qe krijuara automatikisht

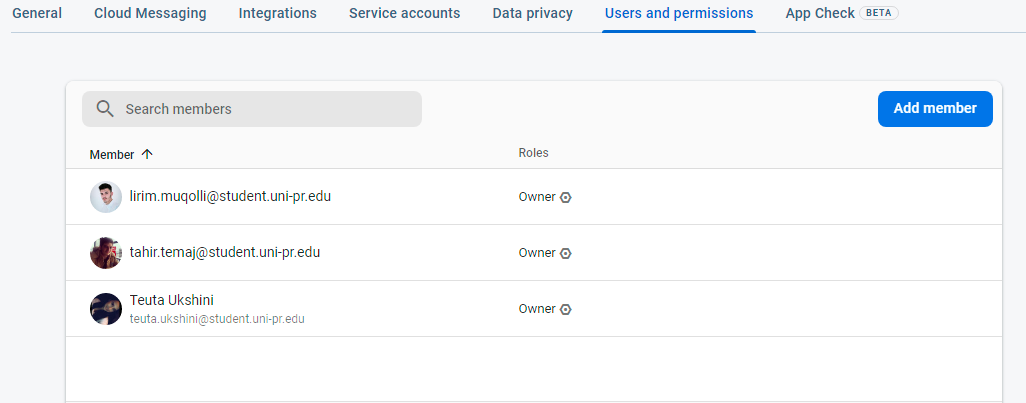


*Fig.22. Detajet e KidsHouse*

Autentifikimi i Firebase-it me anë të email-it që mundëson qasjen më të lehtë



*Fig.23. Autentifikikmi me email*

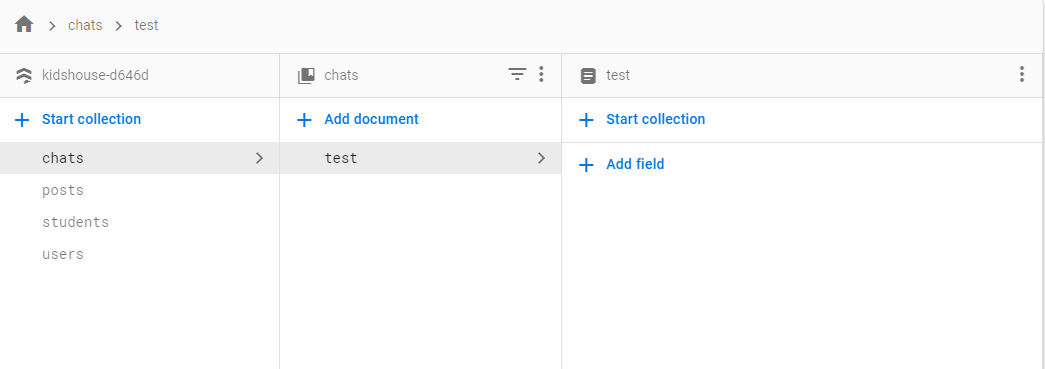


*Fig.24. Lejimi i userave në Firebase*

Lejimi i userave (përdoruesve) në Firebase bëhet përmes gjenerimit të private key si:

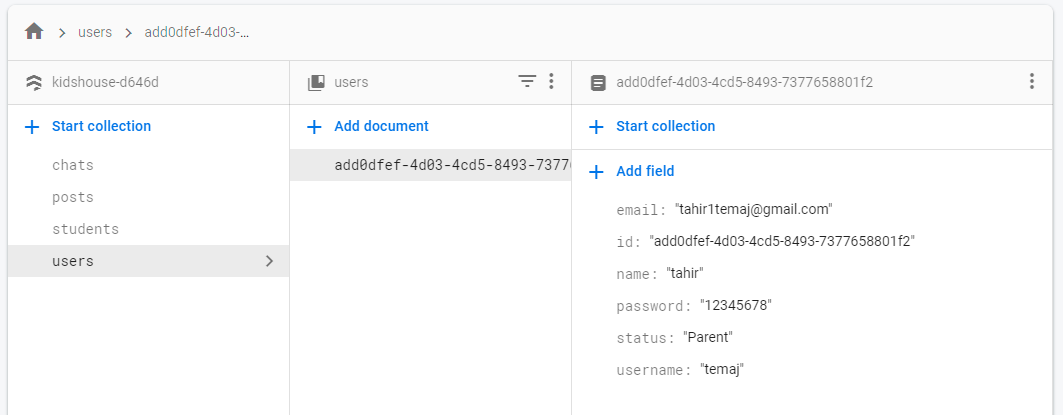


Cloud Firestore është një bazë e të dhënave e dokumenteve NoSQL e ndërtuar për shkallëzimin automatik, performancën e lartë dhe lehtësinë e zhvillimit të aplikacionit. Ndërsa ndërfaqja Firestore ka shumë nga të njëjtat karakteristika si bazat e të dhënave tradicionale, si një bazë e të dhënave NoSQL ajo ndryshon prej tyre në mënyrën se si përshkruan marrëdhëniet midis objekteve të të dhënave.



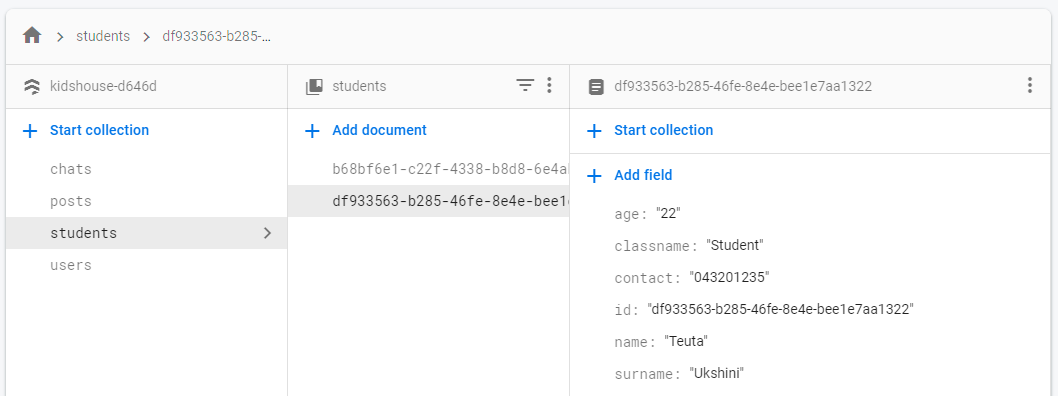
*Fig.25. Firestore database*

Në Firestore është implementuar databaza për user-a (përdorues) që mundësohet përmes komandës Register dhe ruhen tek users. Kur duam të kyqemi në sistem nga komanda Login mirren userat ekzistues.



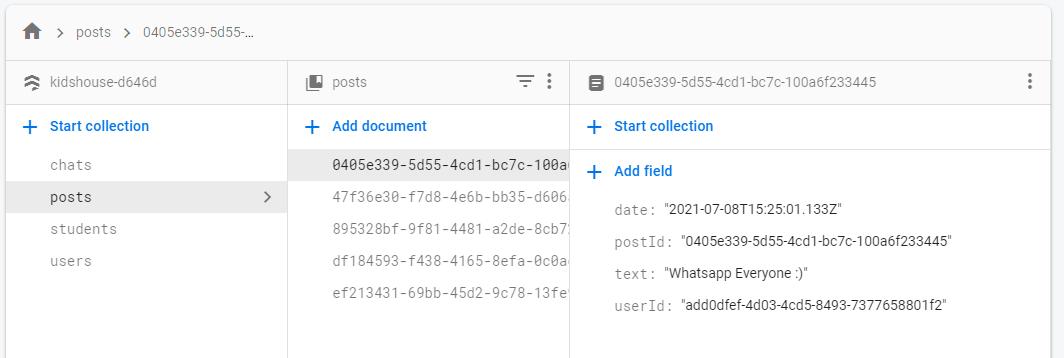
*Fig.26. Userat e regjistruar në databazë*

Cdo shtim që e bëjmë tek studentët ruhet në databazë, po ashtu edhe edit-imet dhe fshirjet.



*Fig.27.Studentët në Firestore*

Si dhe postimet që ndodhin në aplikacion ruhen në Firestore databazë



*Fig.28. Postimet e ruajtura në Firestore*

# **5. Llojet e Komunikimit**

Ekzistojnë disa ndryshime midis komunikimit sinkron dhe asinkron. Dallimi kryesor është se komunikimet sinkrone ndodhin në moment, që do të thotë se është më e shpejtë dhe më dinamike. Kjo do të thotë që është gjithashtu e mirë për pjesëmarrje dhe diskutime aktive me grupe më të vogla.

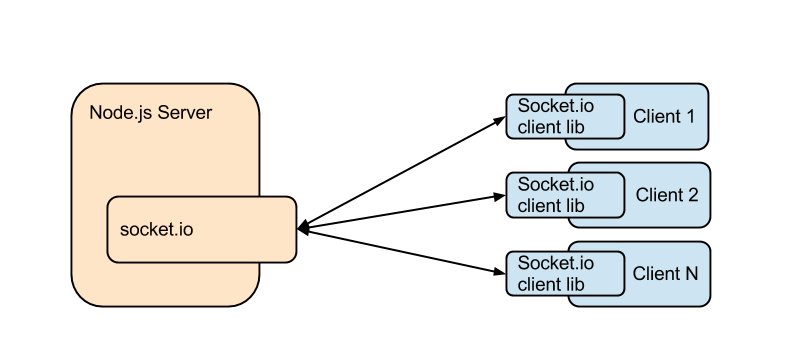
Anasjelltas, komunikimi asinkron ndodh gjatë një periudhe kohe, kështu që nuk ka nevojë të planifikoni asgjë! Ju mund të shkoni me ritmin tuaj, pa aq shumë ndërprerje.

Komunikimi asinkron është më i mirë për të punuar me zona të ndryshme kohore pasi krijon një regjistër të përhershëm të ideve, vendimeve dhe diskutimeve.

Në thelb, komunikimet sinkrone ndodhin në kohë reale, ku komunikimet asinkrone ndodhin gjatë një periudhe kohe. Secili stil komunikimi përdoret për shkëmbimin e informacionit në situata që janë më të përshtatshme për njërën ose tjetrën, përkatësisht.

Deri relativisht kohët e fundit, Uebi, si në shfletuesit, nuk ishte ekuivalent me soketat. Ju mund të bëni vetëm kërkesa asinkrone për burime nga klienti, por serveri kurrë nuk kishte një mënyrë për t'u përgjigjur pa klientin që i kërkoi diçka. Rreth kohës kur erdhi Node.js, aftësia për serverin për të dërguar mesazhe të bandës tek klienti u pa si tepër e dobishme, veçanërisht për një server async, prandaj Socket.IO u bë kaq popullor.

Kjo është diçka që zakonisht duhet të llogaritni me TCP soketat për t'u siguruar që skaji tjetër nuk është zhdukur.

****

*Fig.29. Komunikimi me socket.io*

# **6. Implementimi në Kod**

Sa i perket strukturës së kodit,te gjitha funksionet e moduleve janë bërë nëpër kontrollerë përkates. Lidhja e fushave të databazës ëhtë bërë nëpër modele të veçanta për secilin modul.

Te kontrolleri i studentëve janë bërë të gjitha funksionet për menaxhimin e tyre(shtimin e një studenti,update-imin dhe fshierjen e tij). Pjesa e frontend-it është krijuar përmes Vuejs, ndërsa pjesa e backend -t është krijuar në Nodejs.

Pjesa e frontend-it main.js e implementuar në Vue

import Vue from 'vue'

import App from './App.vue'

import router from './router'

import store from './store'

import axios from 'axios'

import VueAxios from 'vue-axios'

axios.defaults.baseURL = 'http://localhost:5001/kidshouse-d646d/us-central1/app/api';

axios.defaults.headers.common.Accept = 'application/json';

Vue.use(VueAxios, axios)

Vue.config.productionTip = false

new Vue({

router,

store,

render: h => h(App),

}).$mount('#app')

Krijimi i një user-i (përdorues) të ri që mundëson kyqjen në sistem.

class userController {

constructor() {}

createUser(req, res) {

(async () => {

try {

const id = uuidv4()

const user = new users({

id: id,

username: req.body.username,

name: req.body.name,

surname: req.body.surname,

email: req.body.email,

password: req.body.password,

status: req.body.status

})

db.collection('users')

.doc('/' + id + '/')

.create(deserializeJson(serializeJson(user)))

return res.status(HttpStatus.CREATED).send(id)

} catch (err) {

console.log(err)

return res.status(500).send(error)

}

})()

}

Fshirja e userit t regjisëtruar më parë

deleteUser(req, res) {

(async () => {

try {

const document = db.collection('users').doc(req.params.userId)

await document.delete()

return res.status(200).send()

} catch (error) {

console.log(error)

return status(500).send(error)

}

})()

}

}

Funksioni për krijimin apo regjistrimin e një studenti të ri:

createStudent(req, res) {

(async () => {

try {

const id = uuidv4()

const student = new students({

id: id,

name: req.body.name,

surname: req.body.surname,

age: req.body.age,

contact: req.body.contact,

classname: req.body.classname

})

db.collection('students')

.doc('/' + id + '/')

.create(deserializeJson(serializeJson(student)))

return res.status(HttpStatus.CREATED).send()

} catch (err) {

console.log(err)

return res.status(500).send(error)

}

})()

}

Funksioni për të bërë update një student:

updateStudent(req, res) {

(async () => {

try {

const document = db.collection('students').doc(req.params.studentId)

await document.update({

name: req.body.name,

surname: req.body.surname,

age: req.body.age,

contact: req.body.contact,

classname: req.body.classname

})

return res.status(HttpStatus.OK).send()

} catch (error) {

console.log(error)

return res.status(500).send(error)

}

})()

}

Funksioni për fshierjen e një studenti:

deleteStudent(req, res) {

(async () => {

try {

const document = db.collection('students').doc(req.params.studentId)

await document.delete()

return res.status(200).send()

} catch (error) {

console.log(error)

return status(500).send(error)

}

})()

}

}

Ngjashëm edhe në kontrollerët e userave dhe të postimeve janë kryer të gjitha CRUD operacionet. Në modelet e secilit modul janë bërë konstruktorë për të iu qasur atributeve të tabelës përkatëse në databazë.

Konstruktori në modelin e studentëve:

'use strict'

class students {

constructor ({ id, name, age, surname, contact, classname }) {

this.id = id

this.name = name

this.age = age

this.surname = surname

this.contact = contact

this.classname = classname

}

}

## **6.1. Serializimi dhe Deserializimi**

Në zhvillimin modern të aplikacioneve në internet, komunikimi i të dhënave luan një rol shumë të rëndësishëm. Së fundmi, gjithçka ka të bëjë me të dhënat që i duhen aplikacionit të klientit nga serveri. Mjedisi i pritjes për aplikacionin në internet duhet të ndërmarrë veprime proaktive për t'u siguruar që të dhënat e kërkuara duhet të serializohen sipas standardeve të internetit. Aftësia e serializimit të të dhënave në skenarët ndër-aplikues dhe ndër-rrjetor në një format të hapur ka një rëndësi të madhe në aplikacionet në internet dhe host-ueb duhet të marrë këtë përgjegjësi.

Serializimi është procesi i shndërrimit të një objekti në një lumë bajtësh për të ruajtur objektin ose për ta transmetuar atë në memorje, një bazë të dhënash ose një skedar. Qëllimi kryesor i tij është të shpëtojë gjendjen e një objekti në mënyrë që të jetë në gjendje ta rikrijojë atë kur është e nevojshme. Procesi i kundërt quhet deserialization

Meqenëse Node.js bazohet në JavaScript, ne mund të përdorim formatin e mesazhit JavaScript (JSON) dhe mesazhin XML për komunikimin e të dhënave. JSON duke qenë më afër objekteve JavaScript në mënyrën e vet, ne lehtë mund të implementojmë serializimin e të dhënave nga Node.js. Në aplikacionin e zhvilluar duke përdorur Node.js, ne duhet të vendosim se nga mund t'i lexojmë të dhënat dhe t'i serializojmë ato në formatin JSON dhe XML.

'use strict'

const serializeJson = obj => {

const json = JSON.stringify(obj)

return json

}

const deserializeJson = json => {

return JSON.parse(json)

}

module.exports = {

serializeJson,

deserializeJson

}

Krijimi i klasës postController që të bëhet krijimi i postit në bazë të kërkesës dhe përgjigjjes

class postController {

constructor() {}

createPost(req, res) {

(async () => {

try {

const postId = uuidv4()

const post = new posts({

postId: postId,

userId: req.body.userId,

text: req.body.text,

date: new Date()

})

db.collection('posts')

.doc('/' + postId + '/')

.create(deserializeJson(serializeJson(post)))

return res.status(HttpStatus.CREATED).send()

} catch (err) {

console.log(err)

return res.status(500).send(error)

}

})()

}

Update-imi i postit bëhet me anë të funksionit

updatePost(req, res) {

(async () => {

try {

const document = db.collection('posts').doc(req.params.postId)

await document.update({

userId: body.req.userId,

text: body.req.text,

date: new Date()

})

return res.status(HttpStatus.OK).send()

} catch (error) {

console.log(error)

return res.status(500).send(error)

}

})()

}

Fshirja e postimit bëhet në rast se nuk na nevojitet më posti i caktuar, i cili bëhet me anë të kodit

deletePost(req, res) {

(async () => {

try {

const document = db.collection('posts').doc(req.params.postId)

await document.delete()

return res.status(200).send()

} catch (error) {

console.log(error)

return status(500).send(error)

}

})()

}

}

Lidhja me databazen Firebase

var admin = require('firebase-admin')

var serviceAccount = require('../permission.json')

admin.initializeApp({

credential: admin.credential.cert(serviceAccount),

databaseURL: 'https://kidshouse-d646d-default-rtdb.firebaseio.com'

})

module.exports = admin.firestore()

Inkorporimi i Chatit në kod

var usernames={};

datainfo="";

app = express(),

app.use('/public', express.static(path.join(\_\_dirname + '/public')));

server = require('http').createServer(app),

io = require('socket.io').listen(server);

server.listen(PORT);

app.get('/',function(req,res){

res.sendFile(\_\_dirname+'/chat.html');

});

io.sockets.on('connection',function(socket){

// new user enter in chat

socket.on('adduser', function(username){

socket.username = username;

usernames[username] = socket.id;

console.log(username+' has connected to the server');

io.sockets.emit('friend',username);

io.to(usernames[username]).emit("friends",usernames);

});

socket.on('send\_user',function(data){

io.to(usernames[data.to]).emit("type\_user",data);

});

socket.on('nosend\_user',function(data){

io.to(usernames[data.to]).emit("notype\_user",data);

});

socket.on('send\_message',function(data){

message\_data=emoji.emojify(data.message, null, format);

console.log(message\_data);

data1={

"from" :data.from,

"to":data.to,

"message":message\_data

};

io.to(usernames[data.from]).emit("new message",data1);

io.to(usernames[data.to]).emit("new message",data1);

});

var delivery = dl.listen(socket);

delivery.on('receive.success',function(file,info){

datainfo=file.params;

buf=file.buffer;

io.to(usernames[datainfo.to]).emit("image", { image: true, buffer: buf.toString('base64') ,datainfo });

io.to(usernames[datainfo.from]).emit("image", { image: true, buffer: buf.toString('base64') ,datainfo });

console.log(datainfo);

});

});

# **7. Rreth projektit**

Projektin mund ta gjeni në platformën [www.github.com](http://www.github.com) në repozitori me anë të linkut: <https://github.com/TeutaUkshini/Sisteme_te_Shperndara_Gr.5>.

Gjithsej projekti përmban 53 commits që tregon aktivitetin tonë gjatë kohës që kemi punuar.

Projekti përmban fajllin READ.md për komandat që përdoren për ekzekutim që të bëhet më e lehte casja në projekt për këdo që ka nevojë.

# 

*Fig.30.Aktiviteti jonë në Github*

# 

# **8. Konkluzioni**

Me anë të implementimit të platformës sonë për menxhimin e çerdheve kemi arritur në përfundim që do të jetë shumë e dobishme për komunitetin ku jetojmë.

Në këtë projekt, ne kemi një sistem automatizimi të menaxhimit të çerdheve që lehtëson aktivitetet e ndryshme që ndodhin. Sistemi i përdorur në projekt kërkon që të zbatohet në internet. Janë aplikacione të ndryshme në bazë të të dhënave. Është treguar se sistemi regjistron në mënyrë efektive studentët së bashku me informacionin e prindërve, rimerr lehtësisht informacionin rreth një studenti dhe gjeneron raportet e kërkuara siç janë transkriptimi dhe karta raportuese. Për më tepër është paraqitur aplikimi në internet dhe sistemi i ndihmës së regjistrimit të fëmijëve nga mësuesit dhe prindërit mund të shikojnë statusin dhe fëmijët e tyre duke përdorur platformën tonë