Aplikacija afektivnog virtualnog agenta temeljena na višemodalnoj interakciji

Ričko, Andrija

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:959860

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-30**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Andrija Ričko

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

doc. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Andrija Ričko

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se, doc. dr. sc. Tomislavu Stipančiću, na stručnim savjetima i pruženoj pomoći pri izradi ovoga rada.

Također zahvaljujem se doktorandu Leonu Korenu na velikoj pomoći i danim savjetima kod izrade tehničkog dijela rada. Zahvaljujem se i djevojci, roditeljima, bratu, sestrama i prijateljima na podršci i razumijevanju.

Andrija Ričko

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Procesno-energetski, konstrukcijski, inženjersko modeliranje i računalne simulacije i brodostrojarski

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 22 - 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

DIPLOMSKI ZADATAK

Student: **Andrija Ričko** JMBAG: 0035200783

Naslov rada na hrvatskom jeziku: **Aplikacija afektivnog virtualnog agenta temeljena na višemodalnoj interakciji**

Naslov rada na engleskom jeziku: **An affective virtual agent application based on multimodal interaction**

Opis zadatka:

Upravljački mehanizam afektivnog virtualnog agenta PLEA koji se razvija u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava omogućuje analizu i procjenu emocionalnog stanja osobe za vrijeme interakcije. Višemodalni pristup povećava percepcijski potencijal virtualnog agenta te omogućava pouzdaniju procjenu.
Vizualizacija informacija kroz interakcijska sučelja omogućuje bolje razumijevanje od strane osobe u interakciji. U tu je svrhu moguće koristiti različite metode vizualizacije informacija, od grafičkih prikaza podataka ili teksta pa sve do korištenja virtualne stvarnosti prilikom prikazivanja animacija.

U radu je potrebno:

- izraditi cjelovito softversko rješenje afektivnog virtualnog agenta PLEA tako da on može razmjenjivati neverbalne komunikacijske znakove s osobom u interakciji koristeći geste vlastitog lica,
- povezati upravljački mehanizam za stvaranje hipoteza o emocionalnom stanju osobe u interakciji s izražajima na virtualnom licu,
- izraditi web aplikaciju kao interakcijsko sučelje za ljudskog operatera za pristup spremljenim podacima te pogledu virtualnog agenta.

Rješenje je potrebno temeljiti na Unreal Engine platformi za stvaranje virtualnih okoliša te WebRTC mrežnom protokolu koji omogućuje istorazinsku (eng. Peer to Peer) komunikaciju između udaljenih korisnika.
Dobiveno softversko rješenje je potrebno eksperimentalno evaluirati uključivši ljudske subjekte.
U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
5. svibnja 2022.	7. srpnja 2022.	18. – 22. srpnja 2022.

Zadatak zadao:

Doc.dr.sc. Tomislav Stipančić

Predsjednik Povjerenstva:

Prof. dr. sc. Tanja Jurčević Lulić

SADRŽAJ

SADRŽAJ I

POPIS SLIKA III

SAŽETAK V

SUMMARY VI

1. Uvod 1

1.1. Opis sustava 2

2. Stražnja strana (eng. backend) 4

2.1. Kreiranje projekta 4

2.1.1. Opis korištenih biblioteka 5

2.2. Posluživanje statičkih datoteka 6

2.3. Baza podataka 7

2.4. WebRTC poslužitelj 10

2.5. Websocket poslužitelj 11

3. Vizualizacija virtualnog agenta 13

3.1. *Metahuman Creator* 13

3.1.1. Kreiranje realističnog modela 14

3.1.2. Prijenos Metahuman Creator modela u Unreal Engine 15

3.2. Kreiranje projekta u *Unreal Engine*-u 15

3.2.1. Kreiranje animacija lica 18

3.2.2. Mehanizam tranzicije animacija lica 21

3.2.3. Websocket komunikacija u Unreal Engine-u 24

4. Prednja strana (eng. frontend) 25

4.1. Stranica za registraciju i prijavu korisnika 25

4.2. Profilna stranica 26

4.3. Interakcija s virtualnim agentom 27

4.4. Pregled spremljenih podataka 28
4.5. WebRTC klijent za virtualnog agenta 29
4.6. Pregled spojenih korisnika 30
4.7. Povijest povezanih korisnika 30
5. Verzioniranje projekta 31
6. Zaključak 32
Literatura 33
7. PRILOG 34

POPIS SLIKA

Slika 1.1. Cjelokupni sustav....2
Slika 1.2. Profilna stranica osnovnog korisnika....3
Slika 1.3. Profilna stranica administratorskog korisnika....3
Slika 2.1. *Express* logo....6
Slika 2.2. *MongoDB* logo....8
Slika 2.3. Struktura spremanja osobnih podataka klijenata....9
Slika 2.4. Struktura spremanja snimljenih podataka....9
Slika 2.5. WebRTC logo....10
Slika 2.6. Dijagram komunikacije Websocket protokolom....11
Slika 2.7. Socket.IO logo....11
Slika 3.1. *Unreal Engine 4* logo....13
Slika 3.2. *MetaHuman Creator* - odabir baznog modela....14
Slika 3.3. *Metahuman Creator* - sučelje za modeliranje....14
Slika 3.4. Quixel Bridge sučelje....15
Slika 3.5. *Epic Games Launcher* sučelje....16
Slika 3.6. *Unreal Engine* kreiranje projekta....16
Slika 3.7. 3D model virtualnog agenta u *Unreal Engine*-u....17
Slika 3.8. 3D model virtualnog agenta bez tijela....17
Slika 3.9. Kreiranje *Level Sequencer* objekta....18
Slika 3.10. *Level Sequencer* sučelje....19
Slika 3.11. Animiranje lica....19
Slika 3.12. Kreiranje objekta animacije....20
Slika 3.13. Sve kreirane animacije....20
Slika 3.14. Kreiranje objekta *Animation Blueprint*....21
Slika 3.15. *Animation Blueprint*....21
Slika 3.16. *Event Graph* logika....22
Slika 3.17. Enumerator mogućih emocija....22
Slika 3.18. *Anim Graph* logika....23
Slika 3.19. Mehanizam stanja animacije lica....23
Slika 3.20. Primjer uvjeta tranzicije za prelazak u srditu animaciju....23
Slika 3.21. Logika uspješne konekcije s poslužiteljem....24
Slika 3.22. Logika prilikom novog stanja emocije....24

Slika 4.1. Stranica za registraciju korisnika.....25
Slika 4.2. Stranica za prijavu korisnika.....26
Slika 4.3. Profilna stranica.....27
Slika 4.4. Stranica za promjenu lozinke.....27
Slika 4.5. Interakcija s virtualnim agentom.....28
Slika 4.6. Pregled spremljenih podataka.....28
Slika 4.7. Odabir željenog klijenta.....29
Slika 4.8. Odabir željene sesije.....29
Slika 4.9. WebRTC klijent za virtualnog agenta.....29
Slika 4.10. Pregled spojenih korisnika.....30
Slika 4.11. Povijest povezanih korisnika.....30

SAŽETAK

Tema ovog rada je izrada programske upravljačke platforme i vizualizaciju afektivnog virtualnog robota PLEA koji se razvija u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava. Takva platforma krajnjem korisniku omogućuje kreiranje vlastitog profila, konfiguracija spremanja podataka te pregledavanje i brisanje spremljenih podataka te komunikaciju s virtualnim agentom. Ako je krajnji korisnik administrator, platforma omogućuje dodatne opcije kao što su uživo pregledavanje svih povezanih korisnika, konfiguriranje virtualnog agenta, daljinsko otvaranje komunikacije s određenim klijentima i pregledavanje povijesti svih konekcija.

Vizualizacija virtualnog agenta izrađena je u programu *Unreal Engine 4* pomoću programskog dodatka *Metahuman Creator*. Izrađen je set animacija lica koje se aktiviraju kao odgovor na emociju korisnika s kojim PLEA komunicira. Prijenos podataka (slika i zvuk) između korisnika i virtualnog agenta ostvaruje se pomoću istorazinske (eng. Peer to Peer) komunikacije koja omogućuje vizualnu interakciju realnom vremenu.

Ključne riječi: afektivna robotika, Unreal Engine 4, WebRTC, internetska platforma, imitacija

SUMMARY

The topic of this paper is the creation of a program management platform and visualization of the affective virtual robot PLEA, which is being developed as part of the Laboratory for Designing Manufactured and Assembled Systems. Such a platform enables the end user to create his own profile, configure data storage, view and delete saved data, and communicate with a virtual agent. If the administrator is the end user, the platform offers additional options such as directly viewing all connected users, configuring a virtual agent, remotely opening communication with specific clients and viewing the history of all connections.

The visualization of the virtual agent was created in Unreal Engine 4 using the Metahuman Creator plugin. A set of facial animations was created that are activated in response to the emotion of the user with whom PLEA communicates. Data transfer (image and sound) between the user and the virtual agent is achieved using peer-to-peer communication, which enables visual interaction in real time.

Keywords: affective robotics, Unreal Engine 4, WebRTC, web platform, imitation

1. UVOD

U sklopu ovog rada opisano je programsko rješenje za upravljanje virtualnim softverskim agentom koji ima mogućnost djelovanja u sklopu različitih okolina, od one stvarne kroz glavu fizičkog robota, pa do one virtualne uključujući okolinu proširene stvarnosti. Kao rezultat razvijene metodologija, softverski agent u sklopu svojeg djelovanja može koristiti kibernetički prostor koji nije ograničen udaljenostima ili korištenim interakcijskim sučeljem. Interakcijsko sučelja mogu biti naočale za virtualnu ili uronjenu stvarnost, ekran mobitela ili prijenosnog računala, veliki ekran na zidu i sl. Na poziv agent se može pojaviti na sučelju bilo gdje u svijetu. Prisutnost agenta je ista bez obzira da li se korisnik nalazi uz poslužiteljsko računalo unutar laboratorija ili negdje na drugoj polovici zemaljske kugle. Temeljem tehnika za vizualizaciju informacija softverski je agent dobio svoje umjetno generirano lice čiji se izražaji mijenjaju zajedno s promjenom izražaja lica korisnika - operatera. Programski agent se može kretati kroz kibernetički prostor od sučelja do sučelja u ovisnosti na kojem sučelju je zahtijevana komunikacija. Takav je agent je sveprisutan i jedinstven bez obzira na korišteno sučelje i okolinu interakcije [1]. PLEA softverski agent koristi kontekstualni pristup kod zaključivanja i učenja, gdje se robot ovdje vidi kao dio okoliša bez obzira radi li se o realnom svijetu ili kibernetičkom prostoru.

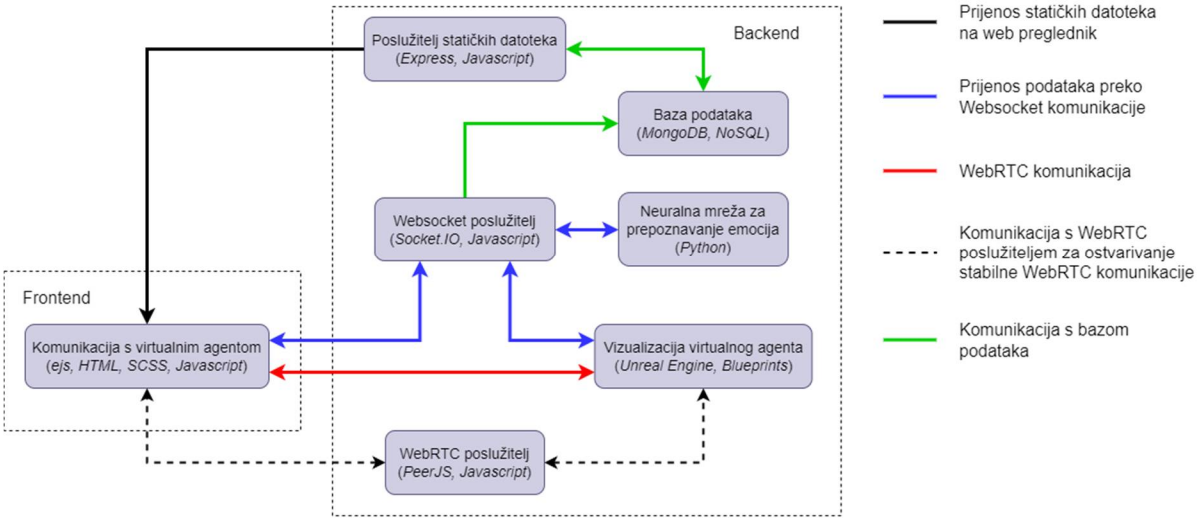
Primijenjena su recentna istraživanja o kognitivnim sustavima, umjetnoj inteligenciji, računalnim mrežama, vizualizaciji informacija te interakciji čovjeka i sustava. Te se spoznaje potom koriste kao temelj za razvoj novih interakcijskih strategija te novih primjena računalne tehnologije. S gledišta znanstvenog razvoja doprinos je u unaprijeđenoj i sigurnijoj vezi između ljudi, uređaja i objekata zbog boljih karakteristika rada, boljeg međusobnog razumijevanja te veće učinkovitosti. U sklopu rada je primijenjen kombinirani metodološki pristup koji uključuje:

1. razvoj računalnog modela za prepoznavanje i mapiranje karakterističnih točaka na licu stvarne osobe,
2. dizajn lica virtualnog agenta te njegovo povezivanje s licem stvarne osobe,
3. razvoj mrežne infrastrukture i pripadajućih upravljačkih mehanizama,
4. eksperimentalnu validaciju razvijene aplikacije u sklopu laboratorijskog postava.

5. razvoj računalnog modela za prepoznavanje emocija koristeći dvije osjetilne modalnosti (vizija i zvuk, PLEA Core) [2, 3, 4, 5].

1.1. Opis sustava

Sustav se sastoji od servera kao centralne jedinice na kojoj je pokrenut algoritam za prepoznavanje emocija. Algoritam je sastavljen od više neuronskih mreža čija je funkcija prepoznavanje akustičkih i vizualnih značajki dobivenih od strane klijenata [7]. Klijent može biti PLEA-robot ili bilo koje računalo s internetskim preglednikom, uključujući i pametne telefone te naočale za proširenu i virtualnu stvarnost

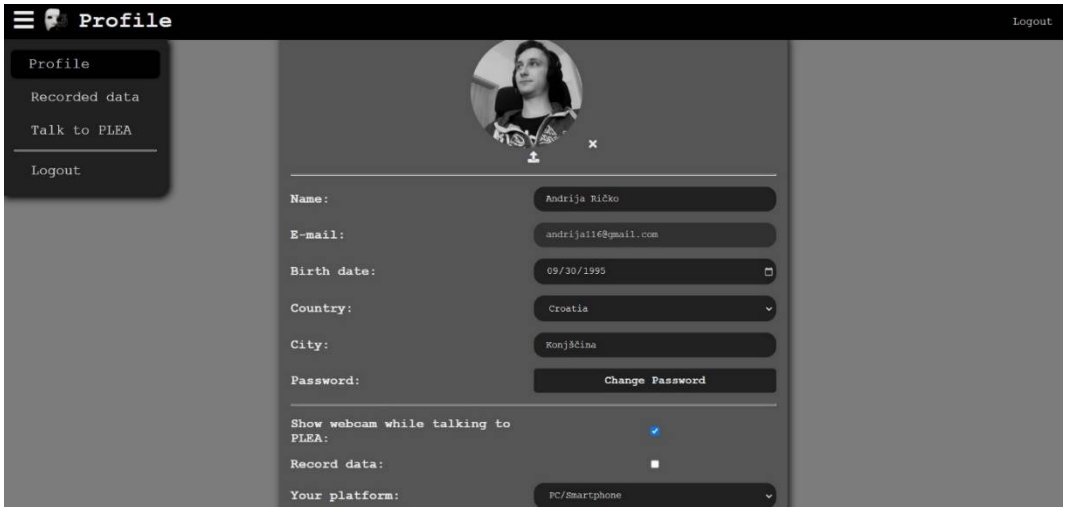


Slika 1.1. Cjelokupni sustav

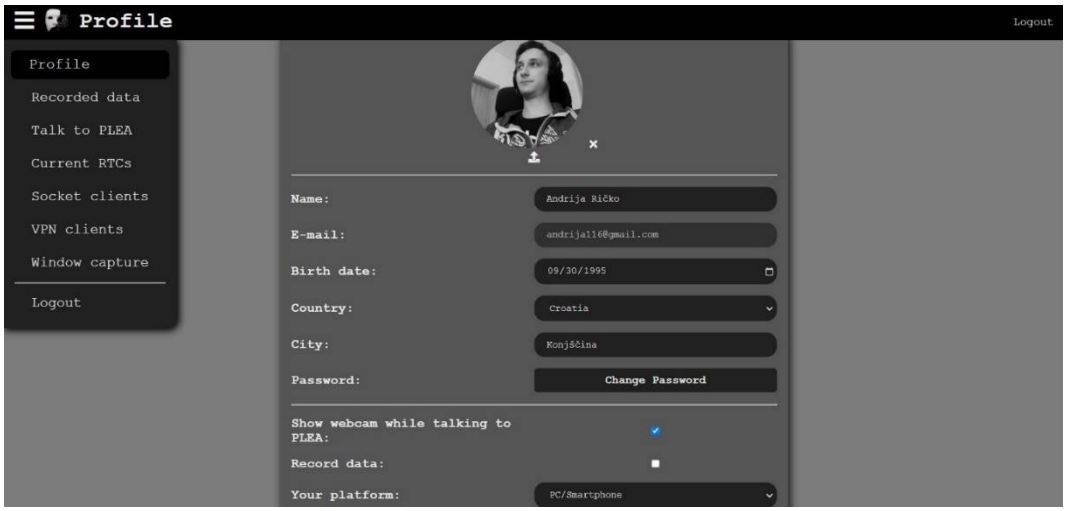
Internetska stranica na kojoj se može vidjeti cjelokupni sustav nalazi se na linku https://pro11.fsb.hr gdje je također detaljno opisan cijeli projekt. Na stranici postoje linkovi za prijavu odnosno registraciju na PLEA platformu. Nakon što se korisnik prijavi sa svojim korisničkim podacima, internetski preglednik sprema kolačiće (eng. cookies) kojima se korisniku omogućuje pristup platformi. Na početnoj stranici PLEA platforme korisnik može se vidjeti svoj profil [Slika 1.2. i Slika 1.3.] i svoje osobne podatke koje u svakom trenutku može izmijeniti i također, ako to želi, može izbrisati svoj korisnički račun. U slučaju brisanja korisničkog računa svi osobni podaci i svi spremljeni podaci koji su služili za unaprjeđenje sustava biti će izbrisani. To je vrlo važno zbog zaštite osobnih podataka i zakonske regulative koja to nalaže.

Na platformi su omogućene dvije vrste uloga korisnika, a to su administratorska uloga i osnovna uloga. Administratorska uloga dodijeljena je određenim korisnicima koji aktivno rade

na ovome projektu i osobe su od povjerenja jer administratorske privilegije uključuju rad s osjetljivim podacima i konfiguracijom virtualnog agenta.



Slika 1.2. Profilna stranica osnovnog korisnika



Slika 1.3. Profilna stranica administratorskog korisnika

Svakom novom registriranom korisniku dodjeljuje se uloga osnovnog korisnika koji ima ograničene mogućnosti na platformi. Te mogućnosti uključuju pregledavanje i uređivanje vlastitog profila, promjena zaporke, interakcija s virtualnim agentom PLEA, pregledavanje, preuzimanje i brisanje svih svojih spremljenih podataka.

2. STRAŽNJA STRANA (ENG. BACKEND)

Struktura stražnjeg (eng. backend) dijela platforme izrađena je u programskom jeziku Javascript odnosno Node.js okruženju. Podijeljena je na više modula koji međusobno komuniciraju. Sastoji se od tri glavna modula, a to su server za posluživanje statičkih datoteka, websocket server za brzu dvosmjernu komunikaciju između klijenta i poslužitelja, i WebRTC poslužitelj za otvaranje istorazinske audio vizualne komunikacije. Takav način pisanja kôda omogućava bolju čitljivost te mogućnost lakše nadogradnje sustava ako je to potrebno.

2.1. Kreiranje projekta

Kreiranje projekta u kojemu se koristi Javascript kao programski jezik potrebno je napisati konfiguracijsku datoteku (prikazano niže) u kojoj se nalazi opis projekta, popis svih biblioteka koji se koriste u tom projektu te skripta za pokretanje cijele aplikacije. Nakon kreiranja konfiguracijske datoteke potrebno je izvršiti komandu *npm i* u konzoli kako bi se u direktorij projekta instalirale sve potrebne biblioteke.

```
1.  {
2.     "name": "pleaserver",
3.      "author": "Andrija Ričko",
4.      "version": "1.0.0",
5.      "description": "An affective virtual agent application based on multimodal
    interaction",
6.      "license": "MIT",
7.      "scripts": {
8.          "runDev": "nodemon server.js"
9.      },
10.     "nodemonConfig": {
11.         "ignore": [
12.             "public/*"
13.         ]
14.     },
15.     "keywords": ["affective robotics", "Unreal Engine 4", "WebRTC", "web platform",
    "imitation"
16.     ],
17.     "dependencies": {
18.         "express": "^4.17.1",
19.         "ejs": "^3.1.6",
20.         "express-flash": "^0.0.2",
21.         "express-session": "^1.17.2",
22.         "socket.io": "^4.3.2"
23.         "peer": "^0.6.1",
24.         "mongodb": "^4.2.2",
25.         "passport": "^0.5.2",
26.         "passport-local": "^1.0.0",
27.         "bcrypt": "^5.0.1",
28.         "google-recaptcha": "^1.1.0",
29.         "dotenv": "^10.0.0",
30.     },
31.     "devDependencies" : {
32.         "nodemon": "^2.0.13"
33.     }
34. }
```

2.1.1. Opis korištenih biblioteka

Express je minimalistička i fleksibilna biblioteka za internetske aplikacije za rad unutar Node.js okruženja. Vrlo je robusna i ima veliki broj značajki koje olakšavaju posluživanje statičkih datoteka na Internet[6].

Ejs biblioteka koristi se u sklopu s Express bibliotekom kako bi se omogućilo renderiranje html datoteka sa serverske strane kako bi dobili dinamički sadržaj internetskih stranica ovisno o tome je li korisnik registriran ili ne odnosno radi li se o administratoru ili osnovnom korisniku.

Express-flash i **Express-session** koriste se u sklopu s Express bibliotekom i omogućuju jednostavno praćenje jesu li kolačići validni te je li klijent prijavljen ili nije.

Socket.io je biblioteka koja omogućuje nisku latenciju, dvosmjernu komunikaciju koja se temelji na događajima između klijenta i poslužitelja [7].

Peer objedinjuje WebRTC implementaciju preglednika kako bi pružio konfigurabilno i jednostavno sučelje za otvaranje istorazinske (eng. peer-to-peer) audio-vizualne komunikacije između dva ili više klijenata. Da bi se ostvarila komunikacija samo je potrebno da klijenti znaju identifikacijske brojeve klijenata s kojim se žele povezati [8].

Mongodb biblioteka nam omogućava otvaranje komunikacije prema bazi podataka koja se koristi za spremanje profila korisnika te spremanje podataka prilikom kominkacije korisnika s virtualnim agentom. Također ova biblioteka nudi set metoda za upisivanje, čitanje, uređivanje i brisanje podataka iz baze [9].

Passport i **Passport-local** je biblioteka koji olakšava implementaciju autentifikacije i autorizacije koja se koristi u Node.js aplikacijama. Omogućava kreiranje kriptiranih kolačića vezanih za osjetljive podatke korisnika koji se mogu sigurno poslati na internetski preglednik.

Bcrypt biblioteka koristi se za sigurno spremanje kriptirane lozinke korisnika u bazu podataka. Prilikom spremanja lozinke korisničkog računa u bazu podataka vrlo je važno da lozinka ne bude čitljiva, odnosno da bude kriptirana u slučaju ako netko, tko ne bi trebao imati, dobije pristup bazi podataka. Na taj način ostvaruje se dodatna sigurnost jer tako kriptirane lozinke ne mogu se dekriptirati nekim jednostavnim metodama.

Google-recaptcha je jednostavna biblioteka koja omogućuje provjeru autentičnosti svake registracije novog korisnika. Važno je raditi provjeru autentičnosti jer se može dogoditi zlonamjerno punjenje baze podataka s nevaljalim podacima.

Dotenv služi kako bi mogli uvesti varijable okoline (eng. environment variables) u samu aplikaciju. Prilikom razvoja aplikacije koristi se malo drugačija konfiguracija projekta (portovi na kojima se otvaraju poslužitelji) pa je korisno imati konfiguracijsku datoteku u kojoj definiramo varijable ovisno u kojoj okolini se nalazimo (razvojna okolina ili produkcijska okolina).

2.2. Posluživanje statičkih datoteka

Server za posluživanje statičkih datoteka temelji se na biblioteci *Express* [6] pomoću koje je postavljanje takvog poslužitelja vrlo jednostavna.

Slika 2.1. *Express* logo

Pozivanjem biblioteke dobivamo objekt koji u sebi sadrži metode koje nam pomažu kod stvaranja ruta na koje korisnik može pristupiti te posluživanje određenog html dokumenta za pojedinu rutu. Primjer je prikazan u kôdu dolje.

```
1.  const express = require("express");
2.
3.  const app = express();
4.
5.  app.use(express.static("public"));
6.
7.  app.get("/", (req, res) => {
8.      res.render("index");
9.  });
10.
11. app.get("/plea", (req, res) => {
12.     res.render("plea");
13. });
14.
15. app.get("/recorded-data", (req, res) => {
16.     res.render("recorded-data");
17. });
18.
19. app.listen(3000);
20.
```

Također kako bi osigurali sigurno registriranje i prijavljivanje korisnika potrebno je inicijalizirati biblioteke passport express-flash i express-session.

```
const passport = require("passport");
const flash = require("express-flash");
const session = require("express-session");

app.use(flash());
app.use(
    session({
        secret: process.env.SESSION_SECRET,
        resave: false,
        saveUninitialized: false,
        cookie: { maxAge: 31_536_000_000 }, // Login is valid for one year
    })
);
app.use(passport.initialize());
app.use(passport.session());

```

Kako bi osigurali da na željenim rutama korisnik bude registriran odnosno validan potrebno je dodati međufunkciju koja će provjeravati validnost korisnika i ako korisnik to nije, sustav će ga prebaciti na stranicu za prijavu.

```
app.get("/plea", checkAuthenticated, (req, res) => {
    res.render("plea");
});

app.get("/recorded-data", checkAuthenticated, (req, res) => {
    res.render("recorded-data");
});

// međufunkcija za provjeru validnosti korisnika
function checkAuthenticated(req, res, next) {
    if (req.isAuthenticated()) {
        return next();
    }
    res.redirect("/login");
}

```

2.3. Baza podataka

U projektu se zahtijeva spremanje korisničkih podataka te spremanje informacija tijekom komunikacije s virtualnim agentom. Kako bi ostvarili taj zahtjev potrebno je odabrati neku vrstu pohrane. Obično se za takvu primjenu koriste baze podataka koje mogu biti ili SQL ili NoSQL.

SQL baze podataka koriste strukturirani jezik upita (eng. structured query language) i imaju unaprijed definiranu shemu za definiranje i rukovanje podacima. SQL je jedan od najsvestranijih i najraširenijih tipova baze podataka, što ga čini sigurnim izborom za mnoge upotrebe. Savršen je za složene upite (eng. query). Međutim, SQL može biti previše restriktivan. Prije samog početka rada sa SQL bazom potrebno je unaprijed definirati sve sheme

tablica i svi podaci moraju slijediti istu strukturu. Ovaj proces zahtijeva značajnu prethodnu pripremu.

NoSQL baze podataka imaju dinamičke sheme za nestrukturirane podatke, a podaci se mogu pohranjuju na mnogo načina. Za pohranu podataka može se koristiti pohrana orijentirana na stupce, orijentiranu na dokumente, temeljenu na grafikonima ili ključ-vrijednost par (key-value pair). To nam daje veliku fleksibilnost prilikom kreiranja određene baze podataka. Dokumente možemo kreirati bez prethodnog definiranja njihove strukture, svaki dokument može imati svoju jedinstvenu strukturu te se u hodu mogu dodavati nova polja u dokumente. Mana NoSQL baze u odnosu na SQL je što upiti ne mogu biti previše složeni te je odaziv baze na upit malo sporiji.

Odabrana baza podataka za ovaj projekt je *MongoDB* [9] koja je bazirana na NoSQL tipu jer dobivamo veliku fleksibilnost prilikom razvoja ovakvog sustava. Prilikom izrade rada više puta je promjenjena struktura pohrane podataka, a pošto je korištena NoSQL baza, nije bilo problema kod migracije podataka.

Slika 2.2. *MongoDB* logo

Nadalje potrebno je napraviti konekciju s *MongoDB* bazom podataka u koju će se spremati podaci od klijenata te povijest konekcijama svih klijenata.

```
1.  const { MongoClient } = require("mongodb");
2.
3.  let dataDb;
4.  let userDb;
5.  let logsDb;
6.
7.  const mongoClient = new MongoClient('mongodb://localhost:27017');
8.  mongoClient
9.      .connect()
10.     .then(() => {
11.         dataDb = mongoClient.db("plea_db");
12.         userDb = mongoClient.db("user");
13.         logsDb = mongoClient.db("logs");
14.         console.log("Connected to Database");
15.     })
16.     .catch(console.log);
```

Slika 2.3. i Slika 2.4. prikazuju strukturu spremanja osobnih podataka registriranih klijenata i snimljenih podataka prilikom interakcije s virtualnim agentom.

```
_id: ObjectId("620060eace45b034801c5835")
email: "andrija116@gmail.com"
name: "Andrija Ričko"
country: "Croatia"
password: "$2b$10$.KSdnDe57y03W3v6vPfWN.4dvjqPYA5IjuVZsKTxUd4iu1MpT5aQ."
role: "admin"
platform: "pc"
city: "Konjščina"
record: true
showWebcam: true
lastActive: 1656876909855
birthDate: "1995-09-30"
image: "..."
```


Slika 2.3. Struktura spremanja osobnih podataka klijenata


```
_id: ObjectId("62c1ef70423206c4c0943bb5")
connectedTimestamp: 1656876911777
userAgent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l..."
camData: Array
  0: Object
     timestamp: 1656876910895
     boundingBox: Object
        x: 258
        y: 197
        width: 126
        height: 178
     expressions: Array
        0: Object
           expression: "Happy"
           probability: 0.8920000195503235
        1: Object
        2: Object
        3: Object
        4: Object
        5: Object
        6: Object
     camImg: "/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQEAAAIYAAAAAAQwAABtbn..."
     pleaImg: "/9j/4AAQSkZJRgABAQAAAQABAAD/4gIoSUNDX1BST0ZJTEUAAQEAAAIYAAAAAAQwAABtbn..."
  1: Object
  2: Object
  3: Object
  4: Object
  5: Object
```


Slika 2.4. Struktura spremanja snimljenih podataka

2.4. WebRTC poslužitelj

WebRTC (Web Real-Time Communication) je tehnologija koja omogućuje internetskim aplikacijama komunikaciju u realnom vremenu koja uključuje strujanje audio i/ili video medija, kao i razmjenu proizvoljnih podataka između preglednika bez potrebe za posrednikom. Skup standarda koji čine WebRTC omogućuje dijeljenje podataka i izvođenje telekonferencija, bez potrebe da korisnik instalira dodatke ili bilo koji drugi softver treće strane.

Slika 2.5. WebRTC logo

Kako bi ostvarili takvu komunikaciju između dva internetska preglednika potrebna je razmjena raznih podataka vezanih uz stanje internetske veze, vrsta internetskog preglednika i slično. To je vrlo kompleksan zadatak te ga je potrebno iznimno dobro odraditi inače veza neće biti stabilna ili u najgorem slučaju klijenti se neće moći povezati. Stoga postoje razne biblioteke koje olakšavaju taj zadatak. U ovom radu koristi se biblioteka PeerJs koja nam omogućuje da razmijenimo samo identifikacijske brojeve internetskih preglednika, a sve ostalo rješava ta biblioteka. Nakon što je istorazinski komunikacijski kanal otvoren, podaci putuju direktno s jednog internetskog preglednika na drugi.

PeerJs biblioteka nam daje na izbor korištenje njihovog poslužitelja kao posrednika za otvaranje komunikacijskog kanala te također mogućost da kreiramo vlastiti poslužitelj. Pošto se u ovom projektu radi o razmjeni osobnih podataka, odlučeno je da će se kreirati vlastiti server koji se jednostavno pokreće.

```
1.  function initPeerjsServer(port, successCallback) {
2.      const { PeerServer } = require("peer");
3.
4.      const peerServer = PeerServer({ port });
5.
6.      if (successCallback) successCallback(peerServer);
7.  }
8.
```

2.5. Websocket poslužitelj

WebSocket je napredna tehnologija koja omogućuje otvaranje dvosmjerne interaktivne komunikacijske sesije između korisničkog preglednika ili nekog drugog programa i poslužitelja. IETF (Internet Engineering Task Force) standardizirao je Websocket protokol 2011. godine pod imenom RFC 6455 [10]. Pomoću ovog protokola mogu se slati poruke poslužitelju i primati odgovore vođene događajima bez potrebe da tražite odgovor od poslužitelja.

U sklopu ovog rada potrebno je ostvariti brzu i pouzdanu dvosmjernu komunikaciju između nekoliko različitih modula sustava [Slika 2.6.].

Slika 2.6. Dijagram komunikacije Websocket protokolom

Prilikom početka komunikacije s virtualnim agentom potrebno je poslati sve potrebne podatke uz tog klijenta k poslužitelju koji zatim te podatke prosljeđuje prema *Unreal Engine* aplikaciji kako bi se otvorio istorazinski (eng. Peer to Peer) audio-vizualni komunikacijski kanal između te dvije strane. Nakon što je komunikacija otvorena, preko socket komunikacije se počinju slati slike s web kamere korisnika koje se prosljeđuju prema neuralnoj mreži za prepoznavanje emocija. Informacija o prepoznatoj emociji se zatim prosljeđuje prema Unreal Engine aplikaciji gdje se generira odgovarajući odgovora (ekspresija lica) virtualnog agenta kojeg krajnji korisnik može vidjeti.

Slika 2.7. Socket.IO logo

Da bi to ostvarili takvu komunikaciju potrebno je kreirati *Websocket* poslužitelja. U ovom radu za to će se koristiti biblioteka *Socket.IO* [7]. Ona nam omogućuje jednostavno kreiranje takvog poslužitelje u kojem je potrebno definirati događaje i odgovore na te događaje tako da sve funkcionira kako je gore opisano.

```
1.  const io = require("socket.io")(port, { cors: { origin: "*" } });
2.
3.  let unrealEngineSocket, neuralNetworkSocket;
4.  let webBrowserClients = [];
5.  io.on("connection", socket => {
6.
7.      socket.on("connection", data =>{
8.          switch(data.type){
9.              case "unreal-engine-client":
10.                 unrealEngineSocket = socket;
11.                 break;
12.             case "neural-network-client":
13.                 neuralNetworkSocket = socket;
14.                 break;
15.             case "web-browser-client":
16.                 webBrowserClients.push(socket)
17.                 if(unrealEngineSocket != null){
18.                     io.to(unrealEngineSocket.id).emit("lets-connect", data);
19.                 }
20.                 break;
21.             }
22.         });
23.
24.         socket.on("web-cam-image", data =>{
25.             if(neuralNetworkSocket != null){
26.                 io.to(neuralNetworkSocket.id).emit('process-web-cam-data', data)
27.             }
28.         });
29.
30.     socket.on("neural-network-result", data =>{
31.         if(unrealEngineSocket != null){
32.             io.to(unrealEngineSocket.id).emit("emotion-to-display", data);
33.         }
34.     })
35. });
36.
```

3. VIZUALIZACIJA VIRTUALNOG AGENTA

Prilikom izrade ovoga rada bilo je potrebno izraditi virtualnog agenta koji će moći vizualno komunicirati s korisnikom. Interakcijsko sučelja mogu biti fizički robot, naočale za virtualnu ili uronjenu stvarnost, ekran mobitela ili prijenosnog računala, veliki ekran na zidu i sl. Stoga je bilo potrebno odabrati razvojnu platformu koja može ostvariti te zahtjeve, a to je *Unreal Engine 4*.

Unreal Engine (UE) je platforma za razvoj 3D računalnih igra koju je razvio *Epic Games* 1998. godine. U početku razvijena je za računalne igre u prvom licu, a također usvojena je i u drugim industrijama, ponajviše u filmskoj i televizijskoj industriji. Napisana je u C++ programskom jeziku. *Unreal Engine* ima visok stupanj prenosivosti, podržavajući širok raspon platformi za stolna računala, mobilne uređaje, konzole, virtualnu i uronjenu stvarnost.

Slika 3.1. *Unreal Engine 4* logo

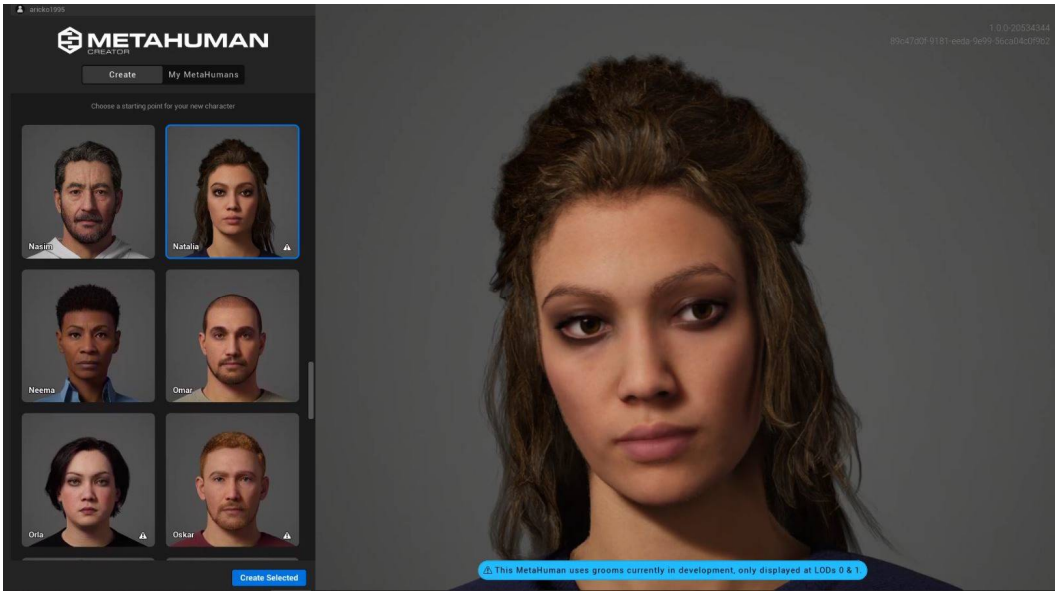
U ovom radu odabrana je *Unreal Engine* verzija 4.27.2. jer je visoko zastupljena u zajednici razvojnih programera, ima velik broj dodataka koji olakšavaju rad i besplatna je za korištenje. Primarni razlog korištenja ove verzije bio je kompatibilnost s dodatkom pod imenom *Metahuman Creator* koji je bio ključan za izradu realističnog lica virtualnog agenta kojemu se mogu animirati izražaji lica.

3.1. *Metahuman Creator*

Metahuman Creator je alat koji je razvio Epic Games kako bi dizajnerima i programerima olakšao izradu realističnih modela ljudskog tijela i lica s potpunim kosturom kojega je moguće jednostavno animirati. Razvijan je desetak godina u suradnji s tvrtkama *3Lateral* i *Cubic Motion.* Početkom 2021. godine Metahuman Creator postao je dostupan za ranu uporabu (eng. early access). Alat je namijenjen da se koristi u internetskom pregledniku stoga nije potrebno korištenje nikakvih drugih programa.

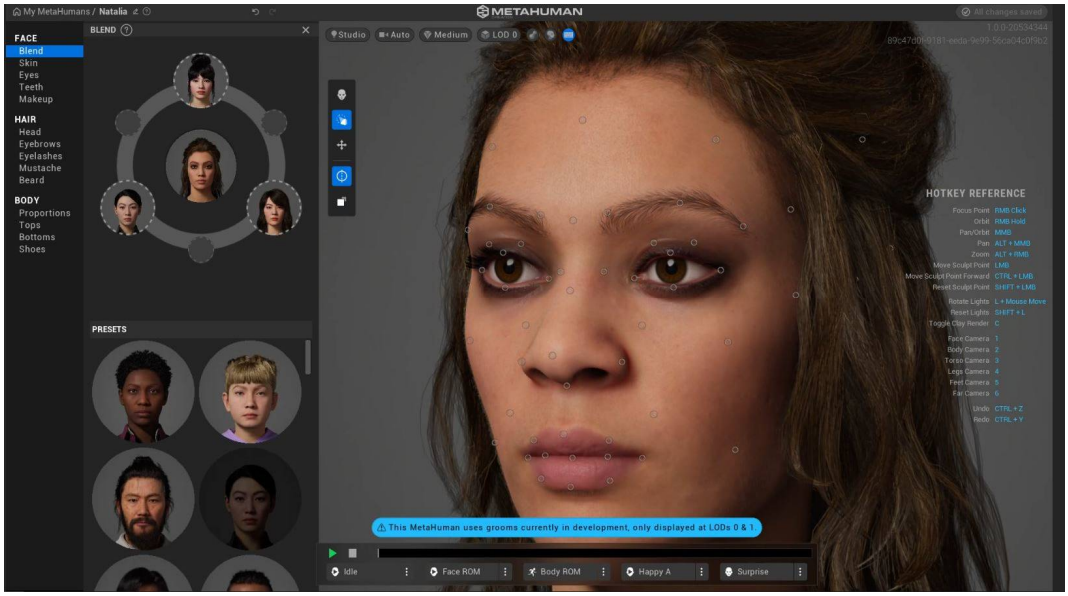
3.1.1. Kreiranje realističnog modela

Postupak kreiranja modela u *Metahuman Creator*-u vrlo je jednostavan. Potrebno je kreirati korisnički račun na Epic Games internetskoj stranici i nakon toga otvoriti internetsku stranicu Metahuman Creator-a. Otvaranjem stranice [Slika 3.2.] dobivamo na izbor dvadesetak već izrađenih modela od kojih se odabire jedan koji će služiti kao baza.



Slika 3.2. *MetaHuman Creator* - odabir baznog modela

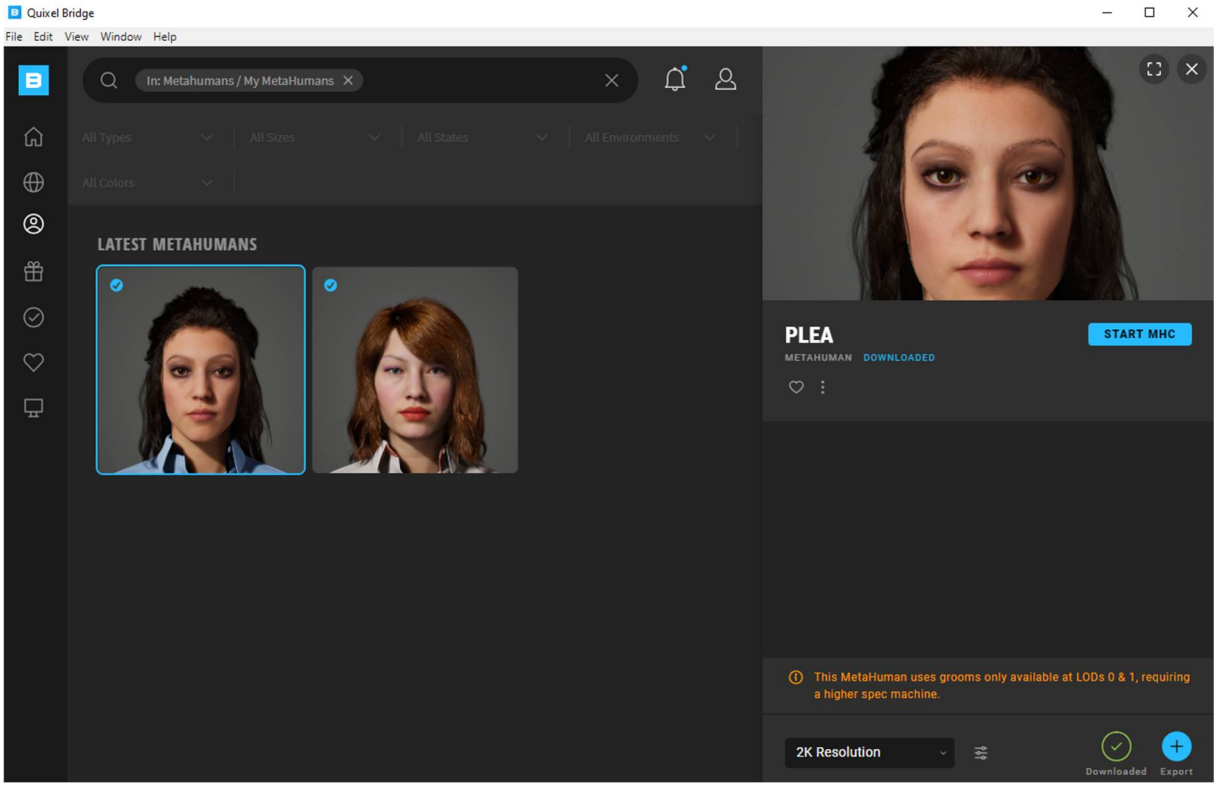
Nakon odabira modela otvara se sučelje [Slika 3.3.] koje nudi razne mogućnosti za oblikovanje lica i tijela modela. Vrlo je jednostavno i intuitivno te se u vrlo kratkom vremenu može dobiti jedinstveni model kojega potom možemo koristiti u daljnjem razvoju aplikacije.



Slika 3.3. *Metahuman Creator* - sučelje za modeliranje

3.1.2. Prijenos Metahuman Creator modela u Unreal Engine

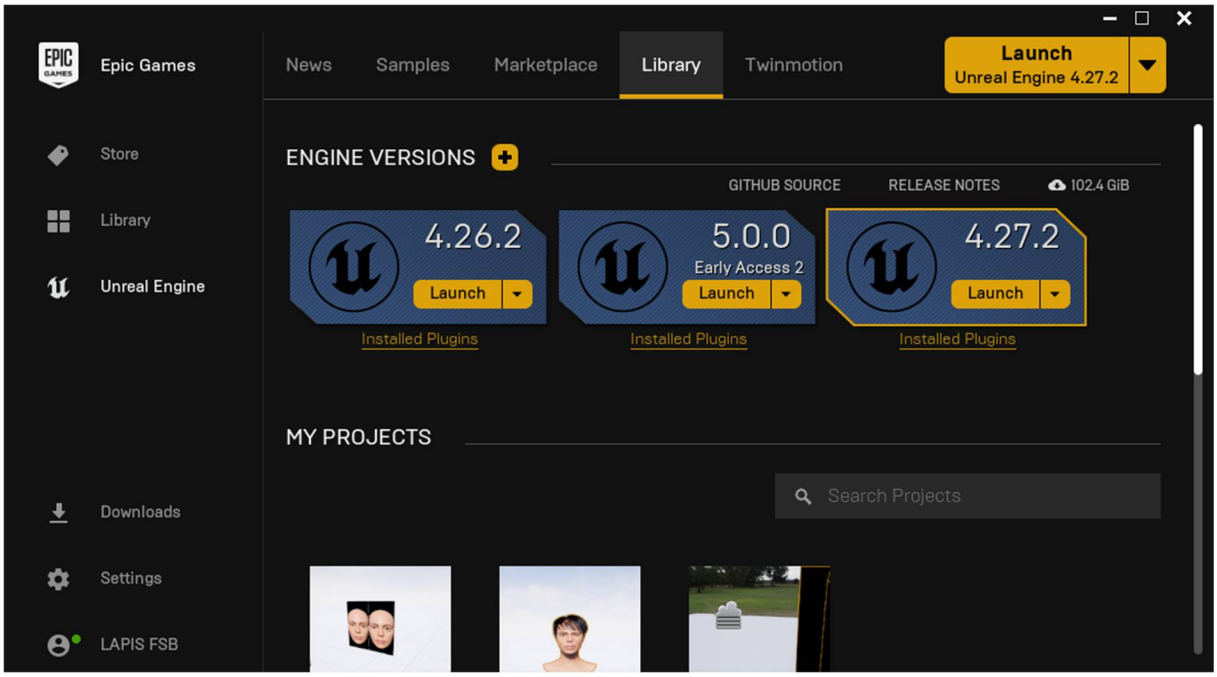
Nakon modeliranja nije potrebno spremati model već se on automatski sprema u oblaku (eng. cloud) gdje je povezan s prije kreiranim korisničkim računom na *Epic Games*-u. Kako bi se kreirani model mogao prenijeti u *Unreal Engine* potrebno je instalirati program *Quixel Bridge* [Slika 3.4.] kojemu se potrebno prijaviti s korisničkim računom *Epic Games*-a. Nakon prijave, svi kreirani modeli vezani za taj korisnički račun, postaju dostupni za preuzimanje i automatsko prebacivanje u *Unreal Engine.* Prije samog prebacivanja potrebno je otvoriti *Unreal Engine Editor* i kreirati novi projekt, nakon čega se model može prebaciti.



Slika 3.4. Quixel Bridge sučelje

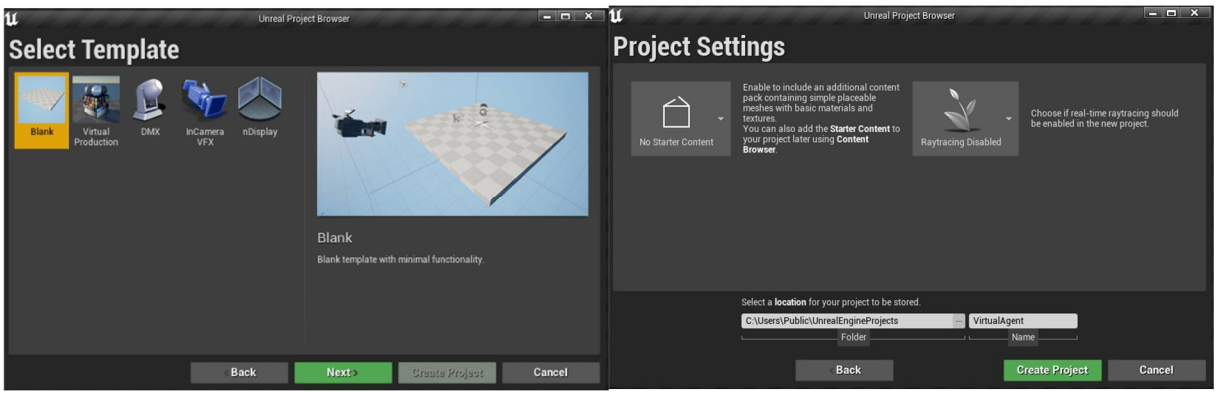
3.2. Kreiranje projekta u *Unreal Engine*-u

Prije svega potrebno je s internetske stranice *Epic Games*-a preuzeti aplikaciju *Epic Games Launcher* [Slika 3.5.] i instalirati ju na računalo. Otvaranjem te aplikacije dobiva se na izbor mogućnost preuzimanja nekoliko različitih verzija *Unreal Engine*-a, a u ovome radu koristi se verzija 4.27.2.



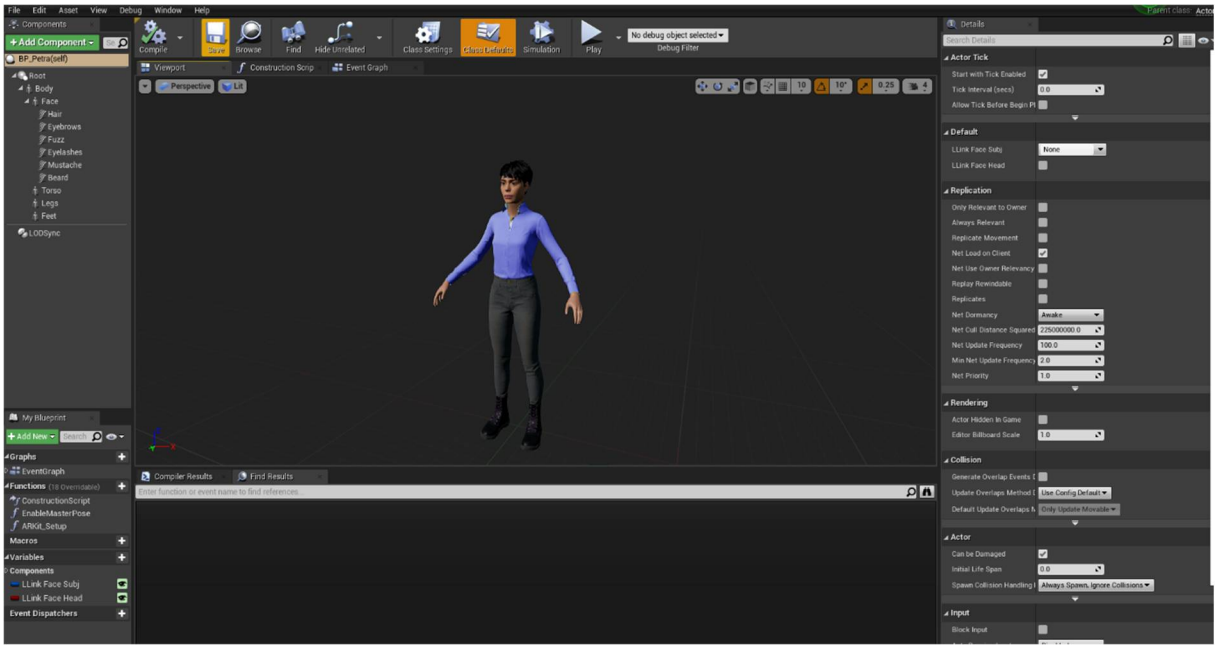
Slika 3.5. ***Epic Games Launcher*** **sučelje**

Nakon preuzimanja željene verzije, klikom na gumb *Launch* otvara se *Unreal Engine Editor* u kojemu se kreira prazni projekt [Slika 3.6.].



Slika 3.6. ***Unreal Engine*** **kreiranje projekta**

Nakon kreacije projekta, preko programa *Quixel Bridge* model koji je kreiran u *Metahuman Creator*-u prebacuje se u *Unreal Engine Editor* gdje dobivamo 3D model tijela [Slika 3.7].



Slika 3.7. 3D model virtualnog agenta u *Unreal Engine*-u

Pošto je u opsegu ovoga rada potrebno samo lice virtualnog agenta, ostatak tijela može se izbrisati [Slika 3.8.].

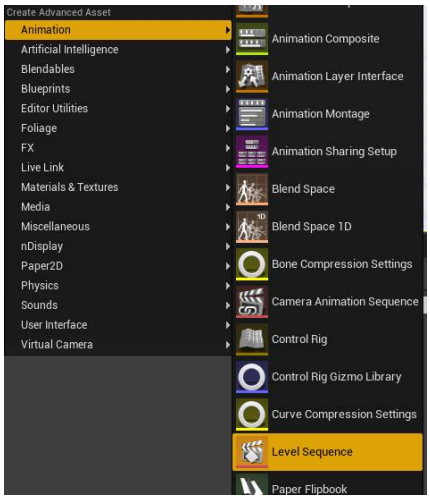
Slika 3.8. 3D model virtualnog agenta bez tijela

3.2.1. Kreiranje animacija lica

Jedan od ciljeva ovoga rada je izrada virtualnog agenta koji će moći razmjenjivati neverbalne komunikacijske znakove s osobom u interakciji koristeći geste vlastitog lica. To se je postiglo tako da je izrađen set od sedam animacija prikladnih za sedam različitih raspoloženja virtualnoga agenta. To su sreća, tuga, srditost, zgađenost, iznenađenost, uplašenost i neutralno raspoloženje.

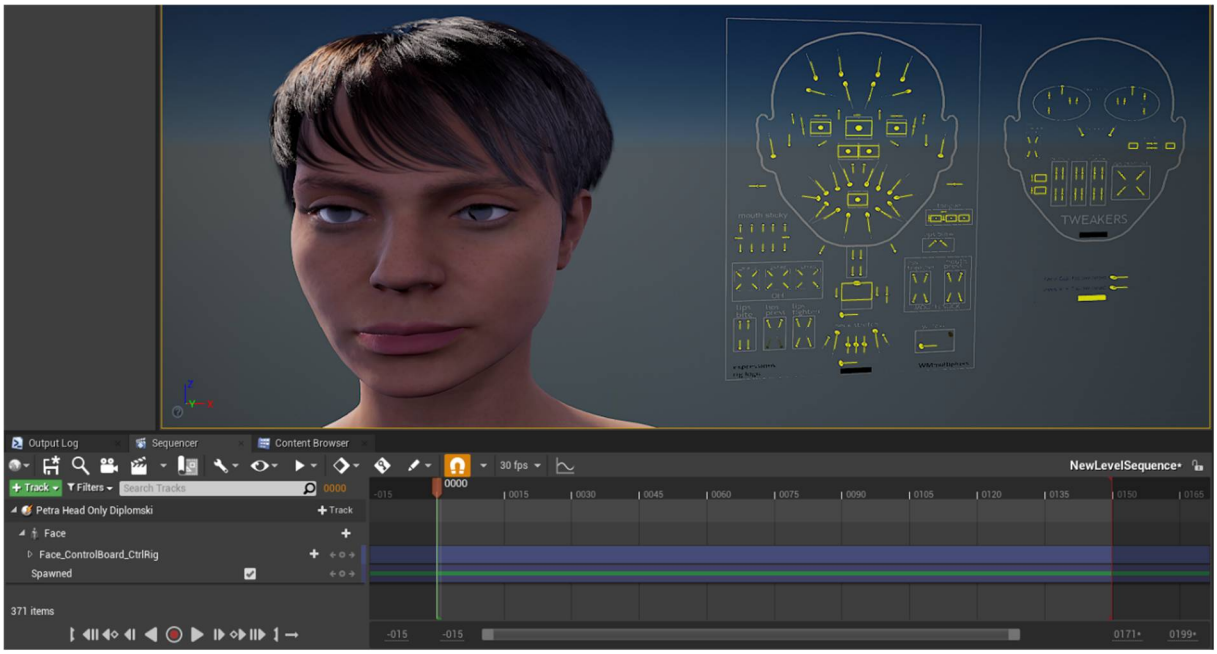
Izrada takvih animacija u *Unreal Engine*-u zahtijeva nekoliko koraka:

1. Kreiranje objekta Level Sequence koji nam omogućuje da pomak svake točke lica spremimo kao jedan moment animacije i stvaranje glatke animacije između više različitih momenata. Kreira se tako da desnim klikom miša unutar *Content Browsera* u grupi *Animations* odaberemo *Level Sequencer*.



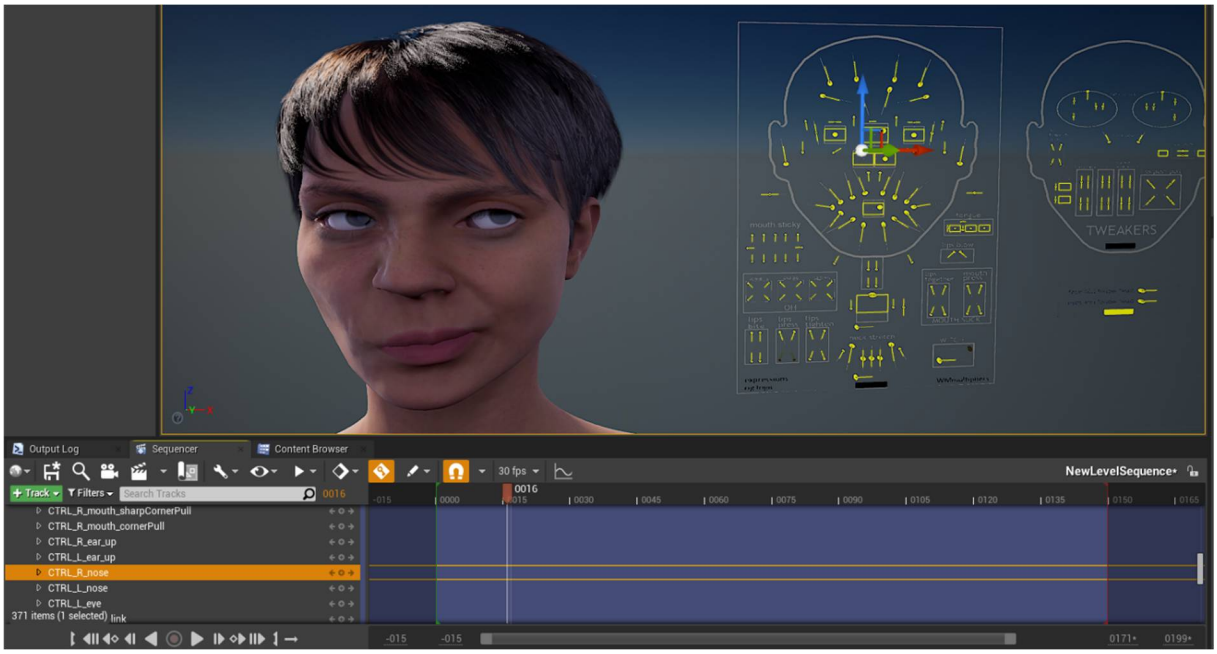
Slika 3.9. Kreiranje *Level Sequencer* objekta

2. Nakon otvaranja novo kreiranog *Level Sequencer*-a, metodom povuci i otpusti (eng. drag and drop), iz *Content Browser*-a povlači se objekt virtualnog agenta (*Blueprint*) unutar vremenske trake *Level Sequencer*-a. S time se dobiva sučelje s više od 270 stupnjeva slobode koje nam omogućuje kreiranje izražaja lica [Slika 3.10].



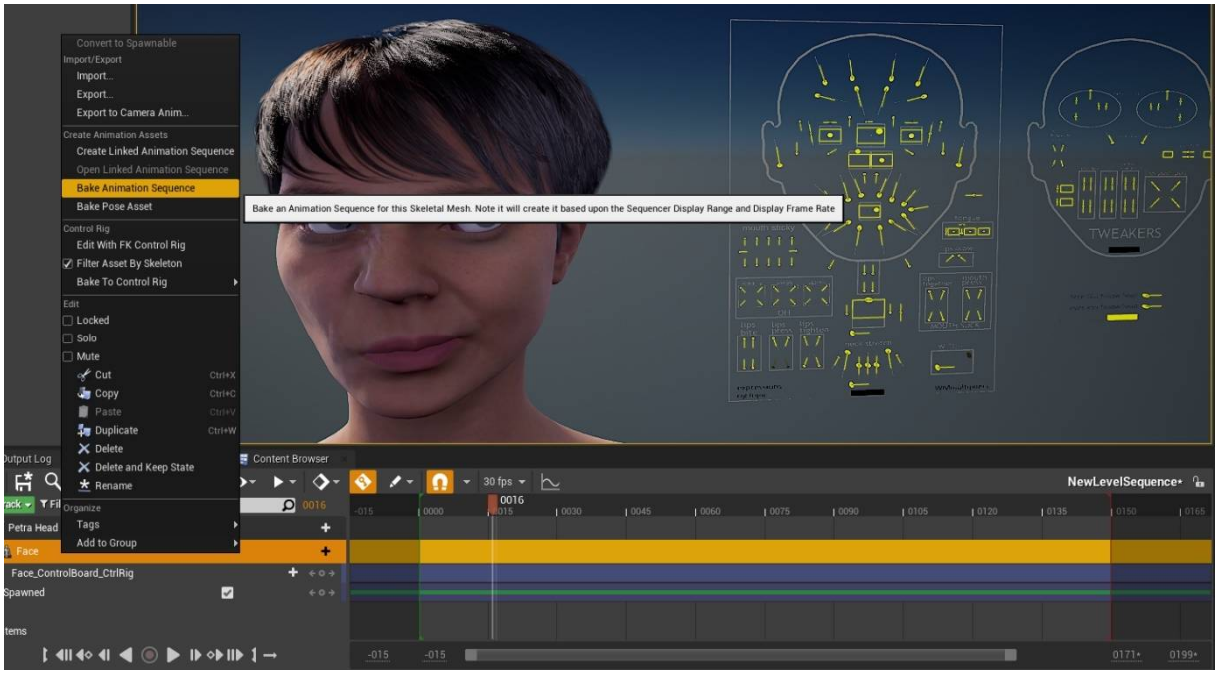
Slika 3.10. *Level Sequencer* sučelje

3. Kreiranje izražaja lica. Namještanje određenog izražaja lica za pojedine momente u animaciji (eng. frame by frame) [Slika 3.11.]. Trajanje jedne animacije je između 10 i 15 sekundi.



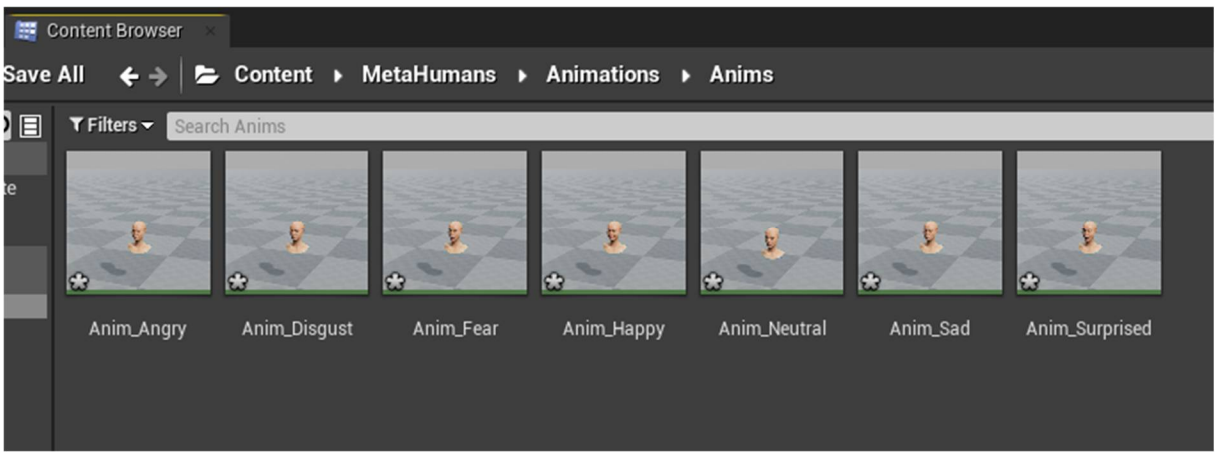
Slika 3.11. Animiranje lica

4. Nakon definiranja izražaja lica u svim željenim momentima, kreira se objekt animacije koji se onda koristi u daljnjem procesu. Unutar vremenske trake desnim klikom na objekt *Face* odabire se *Bake Animation Sequence* što nam daje novi objekt animacije.



Slika 3.12. Kreiranje objekta animacije

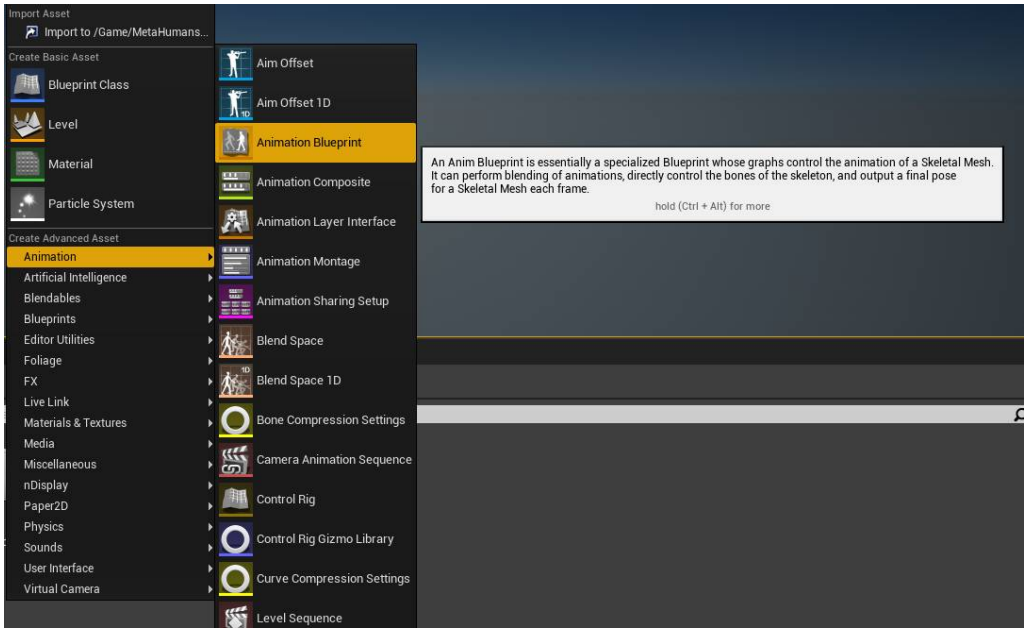
Ta četiri koraka potrebna su kako bi se izradila animacija jedne emocije. Stoga je postupak ponovljen još šest puta kako bi se dobile sve animacije potrebne za uspješno komuniciranje.



Slika 3.13. Sve kreirane animacije

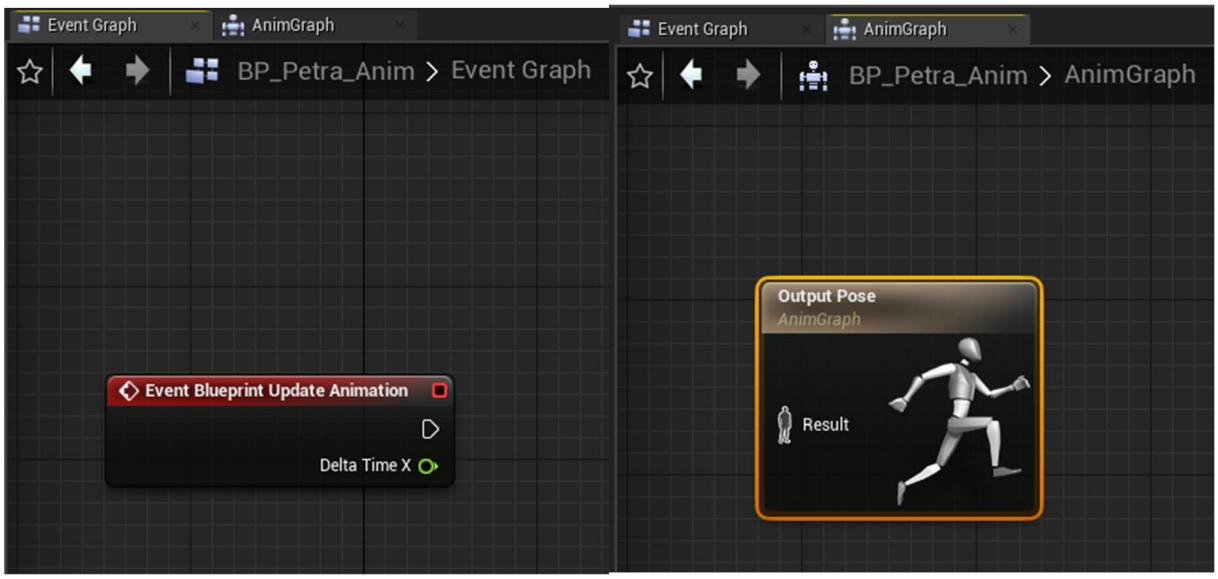
3.2.2. Mehanizam tranzicije animacija lica

Kako bi se dobio glatki prijelaz iz jedne animacije u drugu potrebno je izraditi mehanizam koji će to omogućiti. To se izvodi tako da se kreira objekt *Animation Blueprint* [Slika 3.14.] u kojem se definira u kojem trenutku će se pokretati koja animacija.



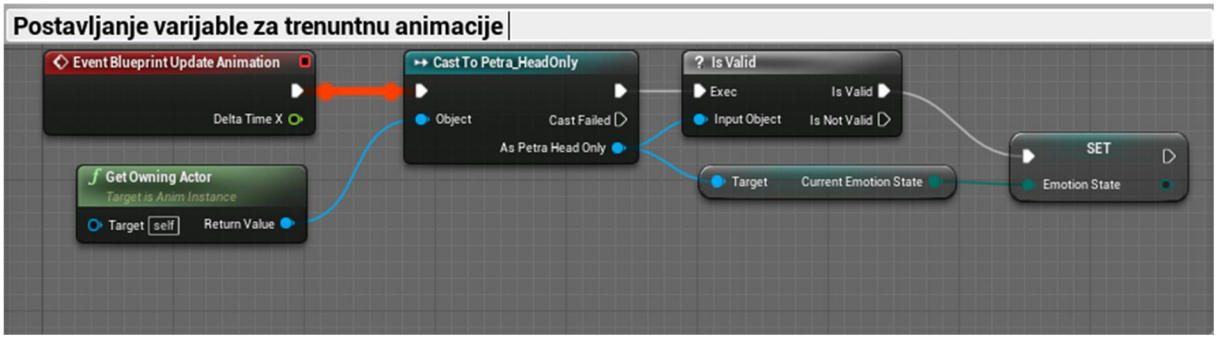
Slika 3.14. Kreiranje objekta *Animation Blueprint*

Unutar tog objekta postoje svije strukture koje su bitne, *Event Graph* i *Anim Graph* [Slika 3.15.].



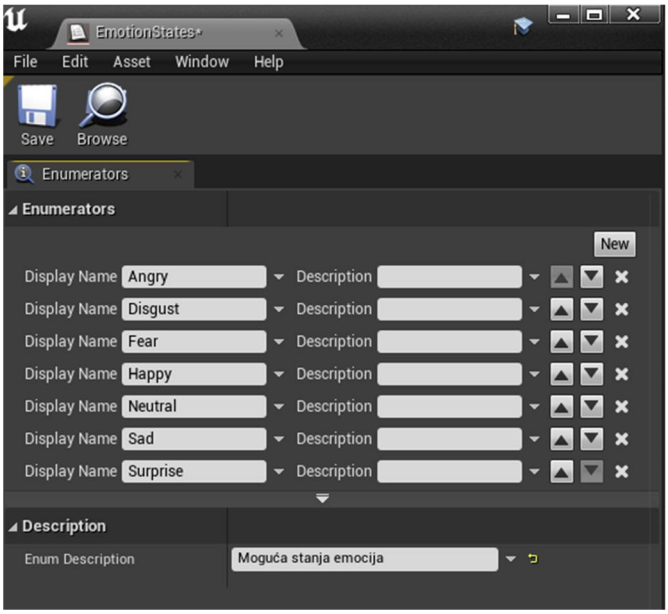
Slika 3.15. *Animation Blueprint*

Event Graph nam daje metodu koja se izvršava 60 puta u sekundi te provjerava jesu li se dogodile kakve promjene unutar objekta (*Blueprint* virtualnog agenta) na kojega se veže taj *Animation Blueprint.* U *Event Graph*-u potrebno je lokalnu varijablu *Emotion State* [Slika 3.16.] ovisno o tome kakav podatak dolazi preko *Websocket* komunikacije.



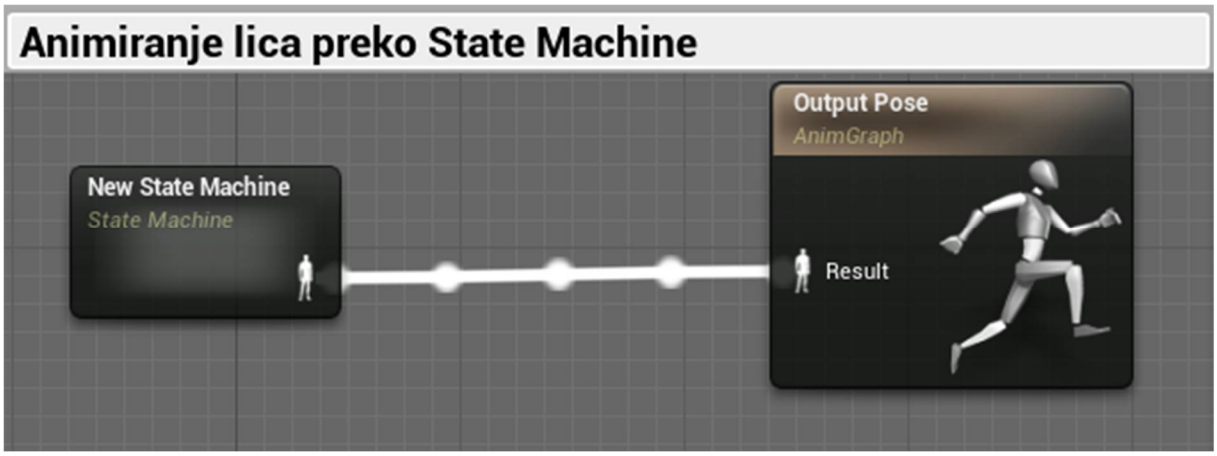
Slika 3.16. *Event Graph* logika

Emotion State varijabla nije ništa drugo nego jedno od mogući emocionalnih stanja, a definirana je kao enumerator tih 7 emocija kao što prikazuje Slika 3.17.



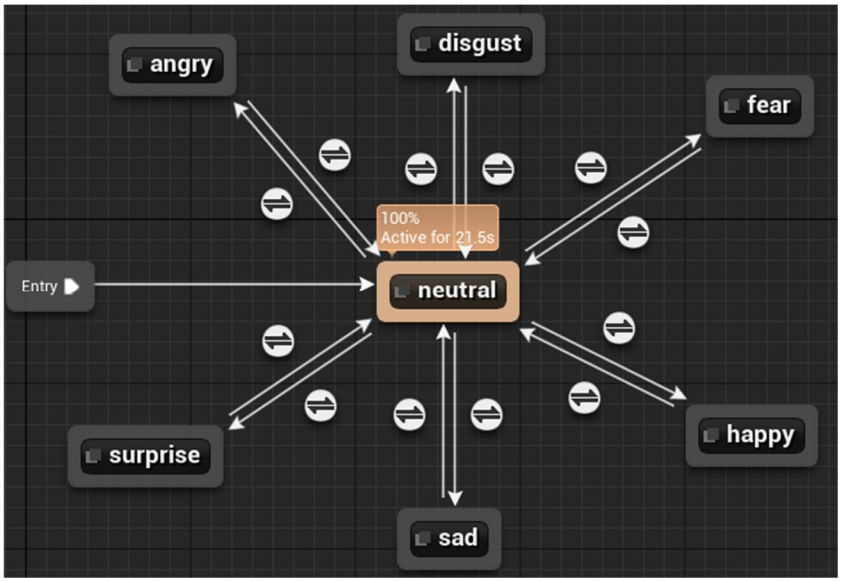
Slika 3.17. Enumerator mogućih emocija

Anim Graph postoji blok u kojega je potrebno uvesti animaciju. Pošto je ovdje moguće odabrati 7 različitih animacija, kreiran je mehanizam stanja (eng. state machine) koji će upravljati pokretanjem određene animacije u pravom trenutku [Slika 3.18.].

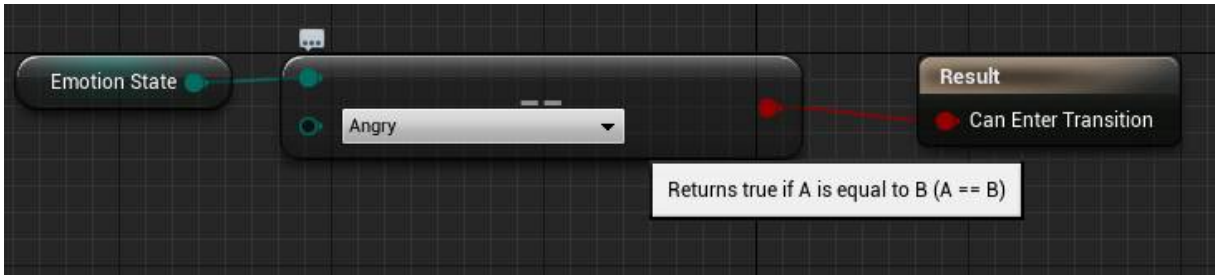


Slika 3.18. *Anim Graph* logika

Mehanizam stanja [Slika 3.19.] definiran je tako da se kontinuirano vrti neutralna animacija, osim ako postoje uvjeti za tranziciju u neku od preostalih animacija. Ti uvjeti definirani su preko lokalne varijable *Emotion State* [Slika 3.20.].



Slika 3.19. Mehanizam stanja animacije lica

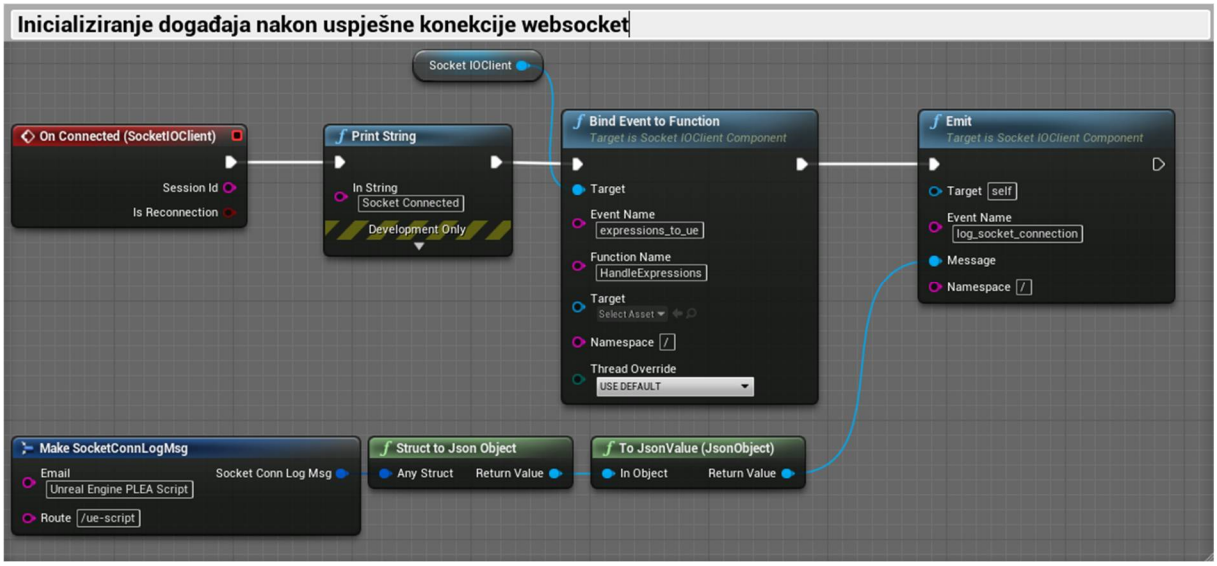


Slika 3.20. Primjer uvjeta tranzicije za prelazak u srditu animaciju

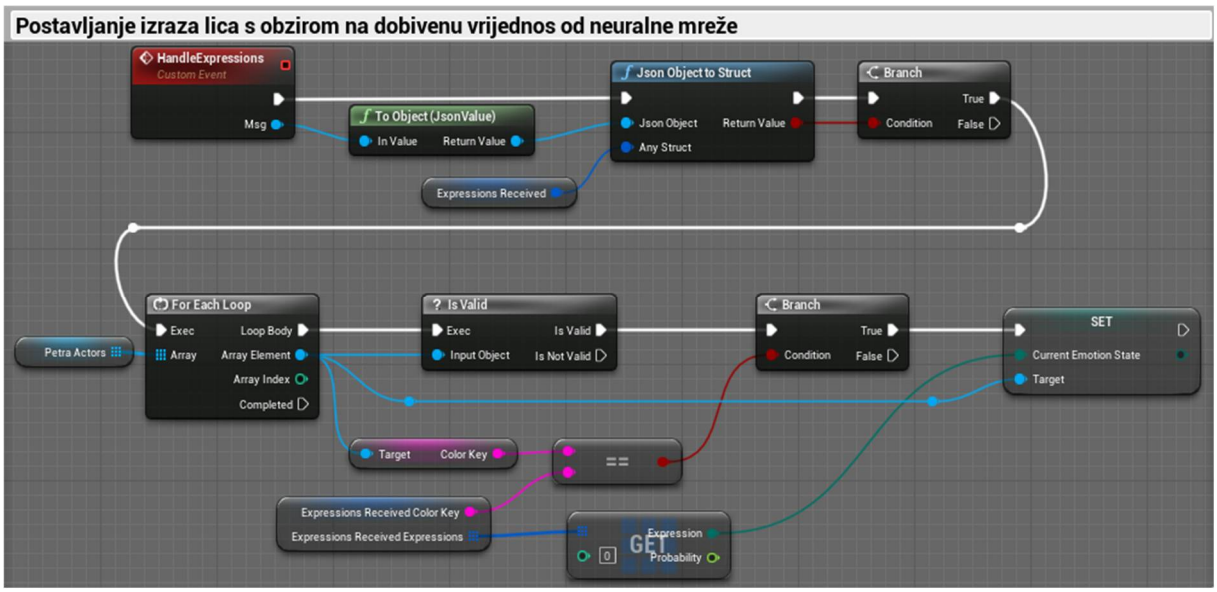
3.2.3. Websocket komunikacija u Unreal Engine-u

Kako bi se ostvarila *Websocket* komunikacija sa stražnjim dijelom (eng. backend) sustava potrebno je instalirati dodatak *SocketIOClinet-Unreal* u *Unreal Engine Editor* koji je kompatibilan s bibliotekom *Websocket* poslužitelja.

Nakon instalacije kreiran je događaj [Slika 3.21.] koji reagira prilikom ostvarivanja konekcije s poslužiteljem gdje se povezuje metoda koja se aktivira prilikom nove poruke od Websocket poslužitelja o stanju emocije [Slika 3.22.].



Slika 3.21. Logika uspješne konekcije s poslužiteljem



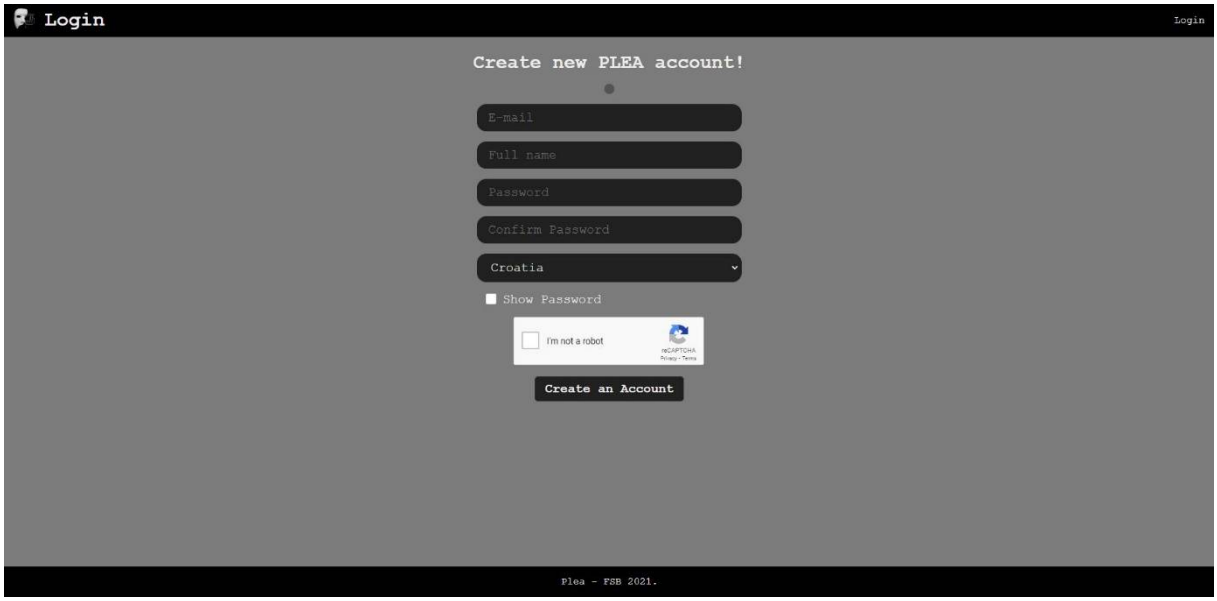
Slika 3.22. Logika prilikom novog stanja emocije

4. PREDNJA STRANA (ENG. FRONTEND)

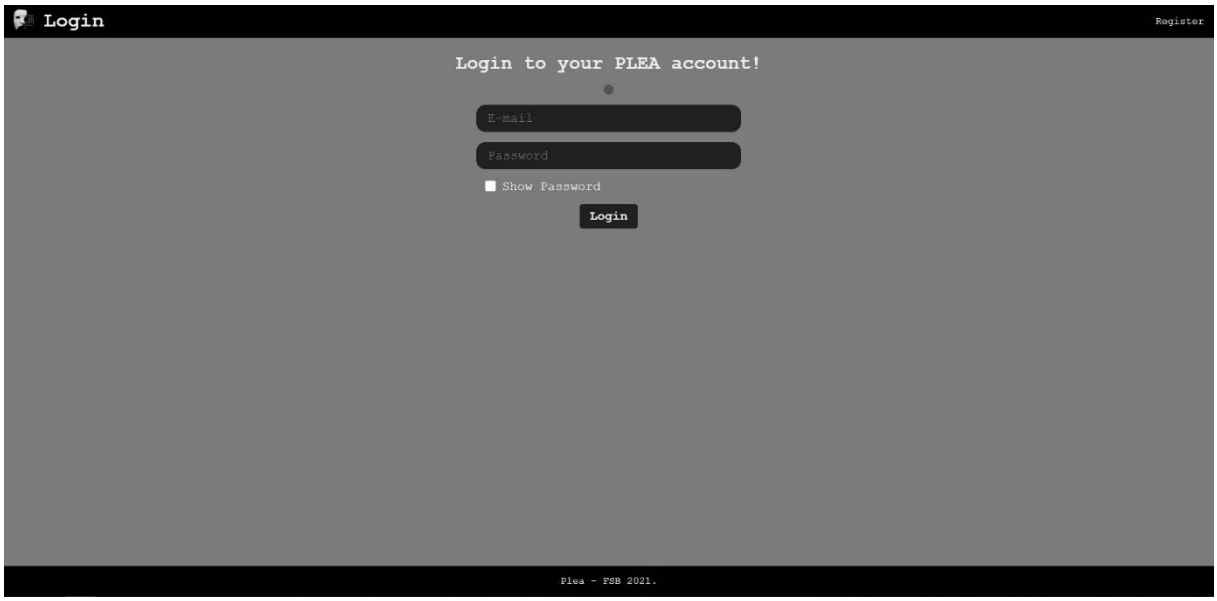
Struktura prednjeg (eng. frontend) dijela platforme izrađena je pomoću *HTML*-a, *CSS*-a i *Javascript*-a. Kreirane su datoteke za svaku rutu na platformi koje onda poslužitelj dijeli na internet.

4.1. Stranica za registraciju i prijavu korisnika

Kako bi korisnik mogao pristupiti platformi potrebno je izvršiti registraciju [Slika 4.1.]. U registracijskoj formi potrebno je navesti osnovne podatke kao što su e-mail adresu, ime i prezime, te državu. Također uz formu nalazi se i sistem koji provjerava da li je korisnik zapravo čovjek, a ne neka zlonamjerna skripta. Kada je korisnik registriran, može se prijaviti [Slika 4.2] kako bi pristupio platformi.



Slika 4.1. Stranica za registraciju korisnika

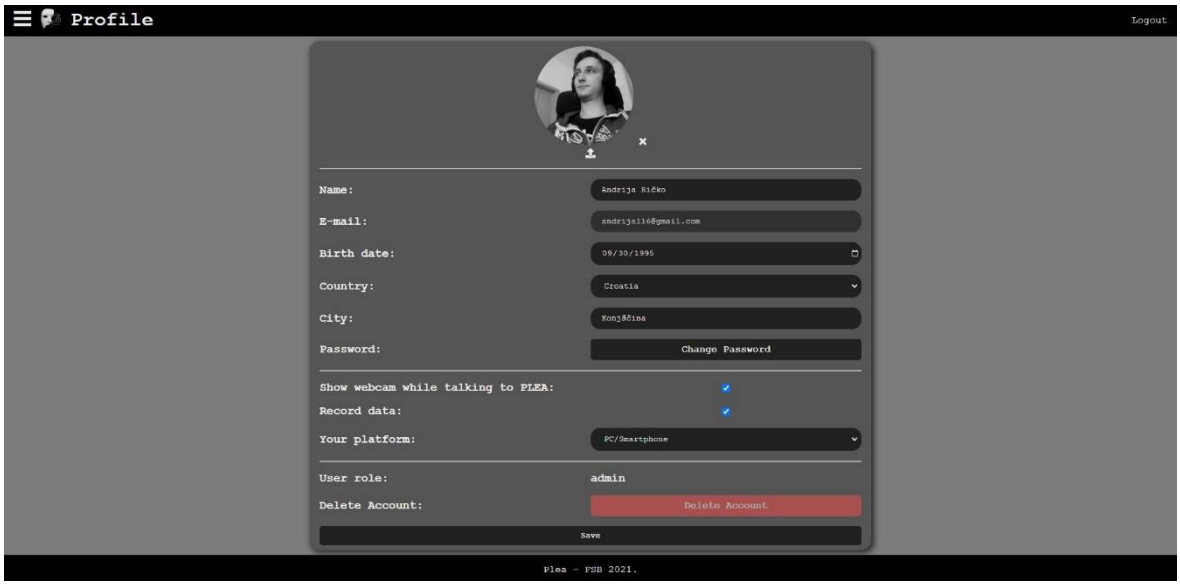


Slika 4.2. Stranica za prijavu korisnika

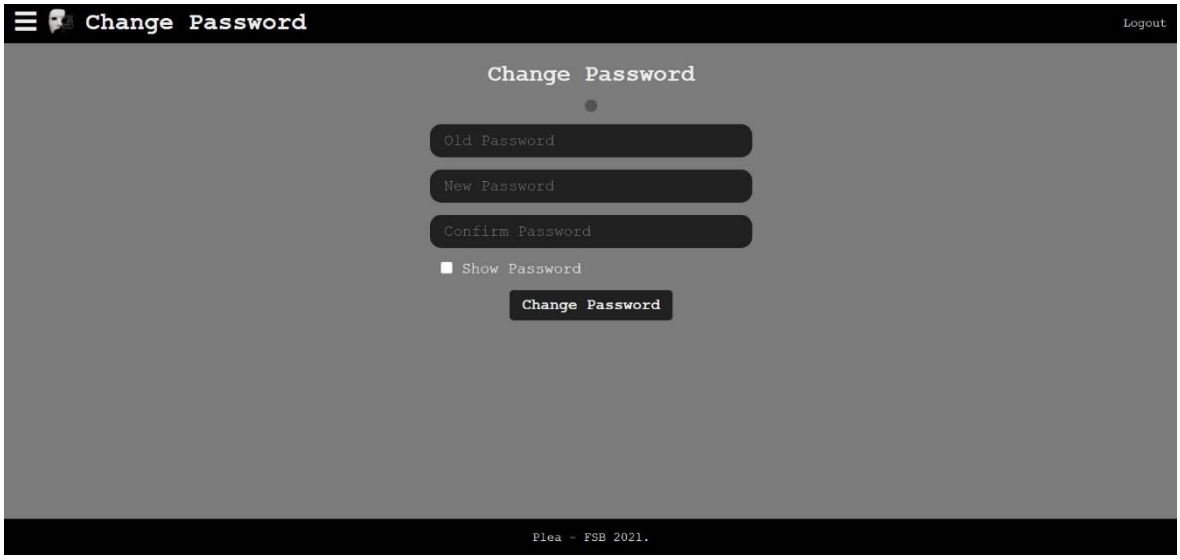
4.2. Profilna stranica

Nakon što se korisnik prijavio sa svojim podacima, prebacuje se na profilnu stanicu [Slika 4.3.]. Ona sadrži sve osobne podatke korisnika koji su spremljeni u bazu podataka. Svaki podatak korisnik može promijeniti u bilo kojem trenutku ako to želi osim e-mail adrese. Pritiskom na gumb *Save* spremaju se sve promjene. Sustav također omogućava promjenu lozinke.

Jedna od bitniji stvari koja se može upravljati je uključivanje i isključivanje snimanja interakcije s virtualnim agentom. Ako je ta opcija uključena, prilikom interakcije s virtualnim agentom, svaku sekundu će se u bazu podataka spremati slika virtualnog agenta, slika korisnika i dodatni podaci vezani za te slike. To su trenutna raspoznata emocija, pozicija glave korisnika na slici, vrijeme i datum,



Slika 4.3. Profilna stranica



Slika 4.4. Stranica za promjenu lozinke

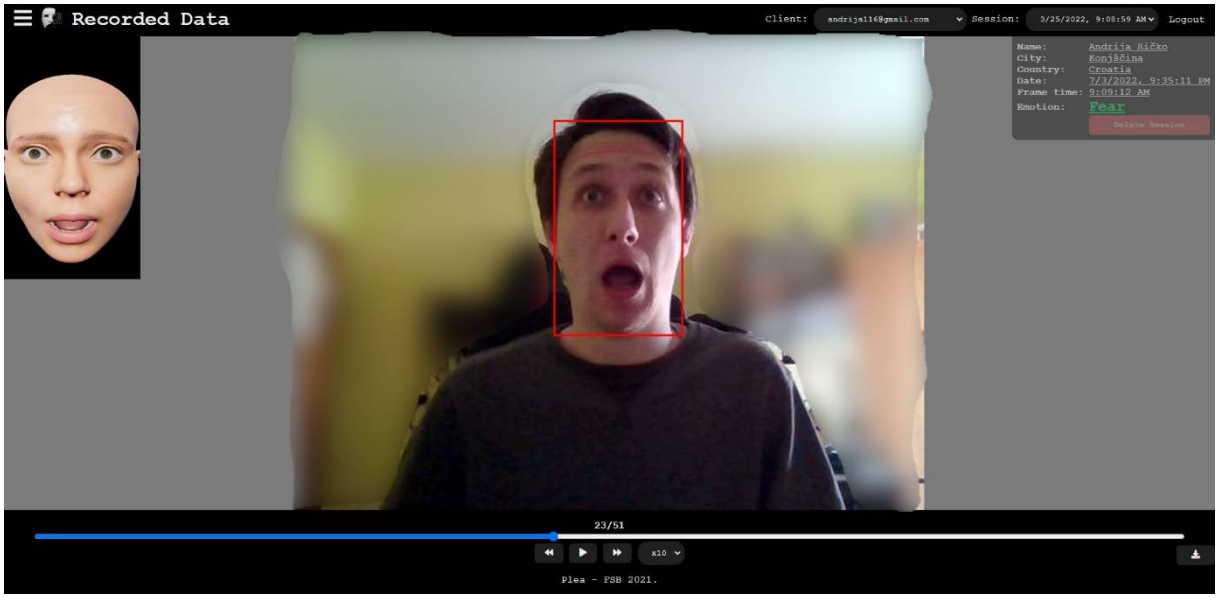
4.3. Interakcija s virtualnim agentom

Prilikom interakcije s virtualnim agentom uključuje se web kamera korisnika koja uzima sliku korisnika svaku sekundu te ju preko *Websocket* komunikacije šalje prema poslužitelju. Poslužitelj zatim šalje tu sliku na obradu u neuralnu mrežu za raspoznavanje emocija te dobiveni rezultat šalje prema *Unreal Engine* aplikaciji gdje se onda pokreće potrebna animacija.

Slika 4.5. Interakcija s virtualnim agentom

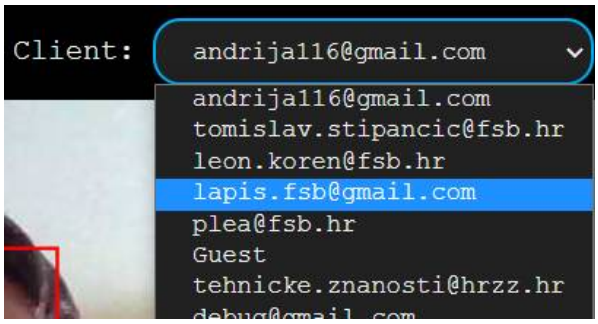
4.4. Pregled spremljenih podataka

Stranica za prikazivanje spremljenih podataka [Slika 4.6.] omogućuje prikazivanje svih spremljenih podataka za pojedinog korisnika. Podatci su spremljeni po sesijama. Jedna sesija je jedna interakcija s virtualnim agentom. U padajućem izborniku *Session* [Slika 4.8.] korisnik može odabrati koju sesiju želi pregledati ili obrisati. Također u donjem desnom kutu nalazi se gumb koji služi za preuzimanje svih slika pojedine sesije. To omogućuje jednostavniju tranziciju podataka na daljnju obradu s kojom je se unaprjeđivati sustav raspoznavanja emocije i generiranje boljeg odgovora.

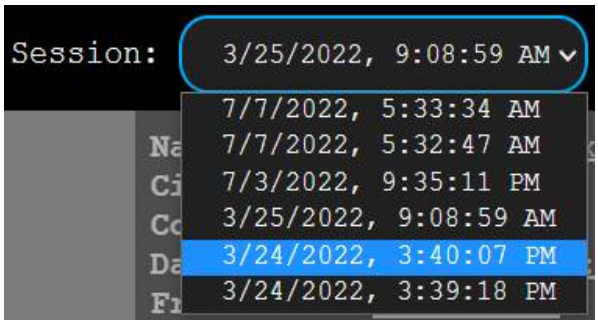


Slika 4.6. Pregled spremljenih podataka

Ako se radi o korisniku koji ima administratorske privilegije, dodatno će biti prikazan padajući izbornik za odabir klijenata. To daje pravo administratoru da pregledava sve spremljene slike od svih klijenata. U slučaju da korisnik nema administratorske privilegije, ta opcija nije prikazane, već se samo dostupna opcija za pregledavanje vlastitih sesija.



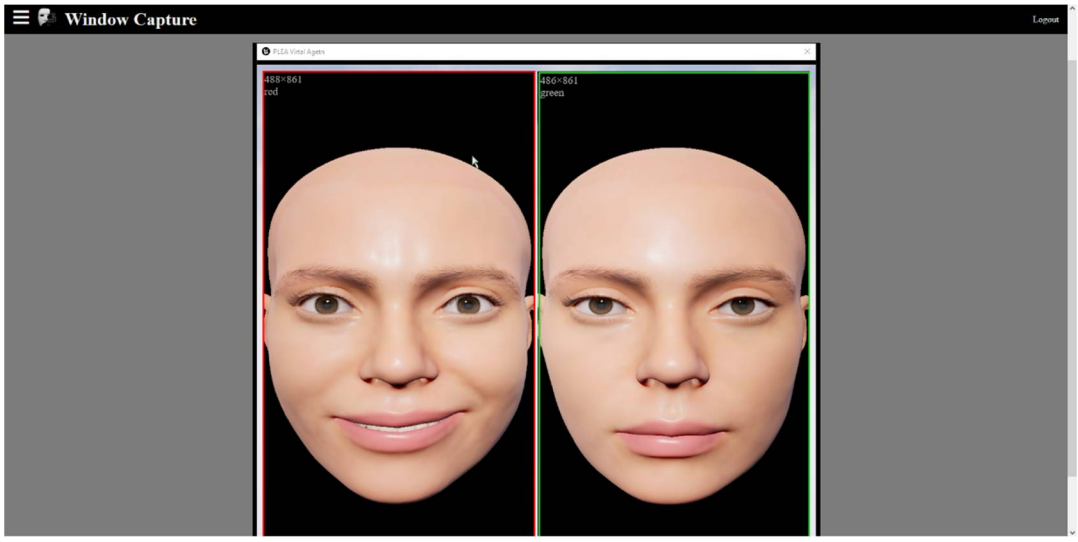
Slika 4.7. Odabir željenog klijenta



Slika 4.8. Odabir željene sesije

4.5. WebRTC klijent za virtualnog agenta

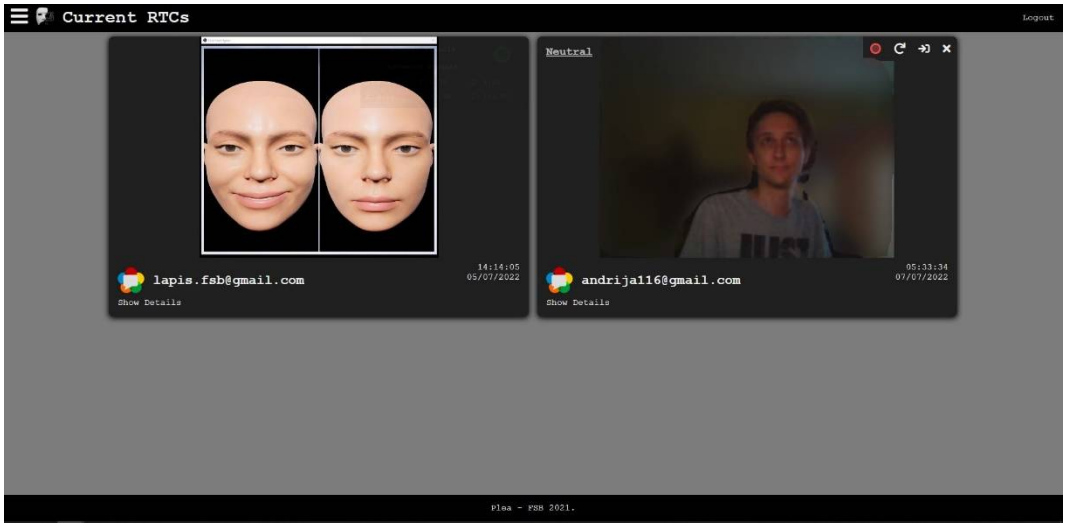
Kako bi se ostvarila WebRTC komunikacija bilo je potrebno kreirati posebnu stranicu koja se otvara na računalu gdje je pokrenuta i *Unreal Engine* aplikacija. Na toj stranici koristi se dodatak koji služi za snimanje zaslona i slanje te snimke u realnom vremenu preko WebRTC komunikacije prema korisniku koji komunicira s virtualnim agentom. Stranica je izrađena tako da se istovremeno može snimati do 5 različitih segmenata zaslona. Slika 4.9 prikazuje dvije instance virtualnog agenta kako bi se omogućilo neometano testiranje agenta na dvije lokacije (FSB sjeverna zgrada 3. kat i Art & AI festival u Leicesteru, Velika Britanija). Ovoj stranici može pristupiti samo korisnik s administratorskim privilegijama.



Slika 4.9. WebRTC klijent za virtualnog agenta

4.6. Pregled spojenih korisnika

Kako bi se vršio nadzor komunikacije bilo je važno kreirati stranicu na kojoj će se moći vidjeti svi spojeni WebRTC klijenti. Moguće je daljinsko osvježavanje, prijavljivanje, izbacivanje povezani klijenata. Ovoj stranici može pristupiti samo korisnik s administratorskim privilegijama.



Slika 4.10. Pregled spojenih korisnika

4.7. Povijest povezanih korisnika

Tijekom testiranja postojali su problemi prilikom ostvarivanja i održavanja komunikacije. Stoga je bilo korisno kreirati stranicu koja će prikazivati povijest svih konekcija kako bi se lakše mogli otkriti i ispraviti mogući kvarovi. Ovoj stranici može pristupiti samo korisnik s administratorskim privilegijama.

Slika 4.11. Povijest povezanih korisnika

5. VERZIONIRANJE PROJEKTA

Sustav za kontrolu verzija (eng. Version Control System), također poznata kao kontrola izvora, kojoj je zadatak praćenja i upravljanja promjenama softverskog koda. Sustavi kontrole verzija softverski su alati koji pomažu softverskim timovima upravljati promjenama izvornog kôda koje se događaju prilikom razvoja.

Softver za kontrolu verzija prati svaku promjenu koda u posebnoj vrsti baze podataka. Ako se napravi pogreška, razvojni programeri mogu vratiti sat unatrag i usporediti ranije verzije kôda kako bi mogli ispraviti pogrešku.

Pošto je ovaj projekt rađen s ciljem da se omogući nadogradnja na postojeću platformu važno je ta postoji sustav za kontrolu verzije kako bi se u budućnosti lakše mogle raditi takve nadogradnje. Takav sustav nam omogućava da za svaku promjenu izvornog kôda opišemo kakve smo izmjene radili, te da sve verzije kôda imamo na jednom mjestu gdje ih lako možemo pregledavati, mijenjati i vraćati na prethodne verzije.

Git je najpoznatiji softver za praćenje promjena u bilo kojem skupu datoteka. Njegovi ciljevi uključuju brzinu, integritet podataka i podršku za distribuirane, nelinearne tijekove rada (tisuće paralelnih grana koje rade na različitim sustavima). Taj softver se koristi u ovom projektu jer je vrlo jednostavan za korištenje i odličan je za rad u timu.

Kako bi inicijalizirali projekt s *Git-om* potrebno je samo izvršiti komandu *git init* u konzoli i to je to. Nakon što je *Git* inicijaliziran možemo nakon promjene programskog kôda u konzoli izvršiti naredbu *git commit -m "commit message"* kako bi spremili novu verziju kôda.

6. ZAKLJUČAK

Izrada ovog rada omogućila mi je rad s raznim modernim tehnologijama i također pružila priliku u radnu na realno sustavu koji se bio temeljito testiran. Razvijena aplikacija PLEA bila je testirana u nekoliko testnih scenarija. Prvi uključuje interakciju studenata FSB-a koristeći veliki ekran smješten na zid ispred Laboratorija u sjevernoj zgradi Fakulteta strojarstva i brodogradnje. Prikaz PLEA agenta generiran je u realnom vremenu na velikom ekranu na temelju procijenjenog emocionalnog stanja korisnika s kojim virtualni agent komunicira pomoću neverbalnih komunikacijskih znakova iskazujući emocije preko lica. Prilikom te interakcija nisu se snimali nikakvi podaci zbog zaštite privatnosti korisnika.

Testiranje PLEA agenta također je bilo provedeno na Art & AI festivala koji se održalo u Leicesteru, Velika Britanija (http://www.art-ai.io/). PLEA robot bio je fizički dostupan kao instalacija na festivalu u srpnju, kolovozu i rujnu 2021. godine. Virtualni agent koji se prikazuje kroz robota bio je inicijaliziran i upravljan iz LAPIS laboratorija s FSB-a. Analitika sustava bila je provedena na laboratorijskom poslužitelju. Time direktno dolazi do izražaja snaga i važnost računalne podrške razvijene i predstavljene u sklopu ovog rada. Kroz suradnju s timom Centra za kreativne tehnologije s De Montfort sveučilišta (DMU) koji je organizator festivala je dogovorena suradnja i zajedničko istraživanje koje se je provodilo u veljači 2022. godine tako da se je snimala interakcija posjetitelja s robotom. Prikupljeno je više od 10 000 slika koje će se analizirati i evaluirati u narednim godinama. Očekuje se da će zajedničko istraživanje dovesti do razvoja novih interakcijskih strategija te novih spoznaja u područjima kognitivne robotike i interakcije čovjeka i robota.

LITERATURA

[1] T. Stipančić, Y. Ohmoto, S. A. Badssi i T. Nishida, »Computation Mechanism for Situated Sentient Robot,« u *SAI Computing Conference*, London, 2017.

[2] Uređivači Encyclopaedia Britannica: Sonar, Encyclopaedia Britannica inc., 2019.

[3] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. J. Lang: Phoneme recognition using timedelay neural networks, IEEE Transactions on Acoustics, Speech, and Signal Processing, 1989.

[4] Fiberology, https://fiberlogy.com/en/, 5.5.2022.

[5] ESP32 Technical Reference Manual, https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf, 5.5.2022.

[6] ExpressJS Technical Reference Manual, https://expressjs.com, 15.6.2022

[7] SocketIO Technical Reference Manual, https://socket.io, 15.6.2022

[8] PeerJS Technical Reference Manual, https://peerjs.com, 15.6.2022

[9] MongoDB Technical Reference Manual, https://www.mongodb.com, 15.6.2022

[10] https://datatracker.ietf.org/doc/html/rfc6455, 15.6.2022

7. PRILOG

- Programski kôd

server.js

```
1.  require("dotenv").config();
2.  const express = require("express");
3.  const { MongoClient } = require("mongodb");
4.  const ObjectId = require("mongodb").ObjectId;
5.  const bcrypt = require("bcrypt");
6.  const passport = require("passport");
7.  const flash = require("express-flash");
8.  const session = require("express-session");
9.  const GoogleRecaptcha = require("google-recaptcha");
10.
11. const googleRecaptcha = new GoogleRecaptcha({ secret: process.env.GOOGLE_RECAPTCHA_SECRET
    });
12.
13. const app = express();
14. const { initSocketServer } = require("./socketServer");
15. const { initPeerjsServer } = require("./peerjsServer");
16. const { initPassport } = require("./passport-config");
17.
18. const EXPRESS_SERVER_PORT = process.env.EXPRESS_SERVER_PORT || 8050;
19. const SOCKET_SERVER_PORT = process.env.SOCKET_SERVER_PORT || 8060;
20. const PEERJS_SERVER_PORT = process.env.PEERJS_SERVER_PORT || 8070;
21.
22. let dataDb;
23. let userDb;
24. let logsDb;
25.
26. const mongoClient = new MongoClient(process.env.DB_URL);
27. mongoClient
28.     .connect()
29.     .then(() => {
30.         dataDb = mongoClient.db("plea_db");
31.         userDb = mongoClient.db("user");
32.         logsDb = mongoClient.db("logs");
33.         console.log("Connected to Database");
34.         initSocketServer(SOCKET_SERVER_PORT, dataDb, logsDb, () => console.log(`Socket
    server started on: http://localhost:${SOCKET_SERVER_PORT}`));
35.     })
36.     .catch(console.log);
37.
38. initPeerjsServer(PEERJS_SERVER_PORT, () => {
39.     console.log(`Peerjs server started on: http://localhost:${PEERJS_SERVER_PORT}`);
40. });
41.
42. initPassport(
43.     passport,
44.     async email => await userDb.collection("users").findOne({ email: email }),
45.     async id => await userDb.collection("users").findOne({ _id: new ObjectId(id) })
46. );
47.
48. app.set("view engine", "ejs");
49. app.use(express.static("public"));
50. app.use(express.urlencoded({ limit: "5mb", extended: false }));
51. app.use(flash());
52. app.use(
53.     session({
54.         secret: process.env.SESSION_SECRET,
55.         resave: false,
56.         saveUninitialized: false,
57.         cookie: { maxAge: 31_536_000_000 }, // Login is valid for one year
```

```
58.     })
59. );
60. app.use(passport.initialize());
61. app.use(passport.session());
62.
63. /** @description Web page routes */
64. app.get("/login", checkNotAuthenticated, (req, res) => {
65.     res.render("login", { originalUrl: req.query.originalUrl });
66. });
67.
68. app.post("/login", passport.authenticate("local", { failureRedirect: "/login",
    failureFlash: true }), (req, res) => {
69.     if (req.body.isExternalReq) return res.redirect("/plea0");
70.     res.redirect("/profile");
71. });
72.
73. app.get("/register", checkNotAuthenticated, (req, res) => {
74.     res.render("register", { googleRecaptchaSiteKey:
    process.env.GOOGLE_RECAPTCHA_SITE_KEY });
75. });
76.
77. app.post("/register", checkNotAuthenticated, (req, res) => {
78.     googleRecaptcha.verify({ response: req.body["g-recaptcha-response"] }, async error =>
    {
79.         if (error) return res.render("register", { user: req.body, messages: { error:
    "Are you a robot?" } });
80.         else {
81.             const existingUser = await userDb.collection("users").findOne({ email:
    req.body.email });
82.             if (existingUser) return res.render("register", { user: req.body, messages: {
    error: "User with that email already registered" } });
83.             if (req.body.password !== req.body.confirmPassword) return
    res.render("register", { messages: { error: "Passwords do not match" } });
84.             const hashedPassword = await bcrypt.hash(req.body.password, 10);
85.             const { insertedId } = await userDb.collection("users").insertOne({
86.                 email: req.body.email,
87.                 name: req.body.name,
88.                 country: req.body.country,
89.                 password: hashedPassword,
90.                 role: "basic",
91.                 platform: "pc",
92.                 city: "Undefined",
93.                 record: false,
94.                 showWebcam: false,
95.             });
96.             const user = await userDb.collection("users").findOne({ _id: insertedId });
97.             if (user) {
98.                 req.login(user, err => {
99.                     if (err) {
100.                         return res.render("register", {
101.                             messages: {
102.                                 error: "Sorry but something went wrong while
    registering. Try to login with this account and if not successful, try to create account
    again.",
103.                             },
104.                         });
105.                     } else return res.redirect("/profile");
106.                 });
107.             } else {
108.                 return res.render("register", {
109.                     messages: {
110.                         error: "Sorry but something went wrong while registering.
    Database is unavailable. Try again later.",
111.                     },
112.                 });
113.             }
114.         }
115.     });
116. });
117.
```

```
118.  app.get("/profile", checkAuthenticated, (req, res) => {
119.      res.render("profile", { user: req.user });
120.  });
121.
122.  app.get("/logout", checkAuthenticated, (req, res) => {
123.      req.logOut();
124.      res.redirect("/login");
125.  });
126.
127.  app.post("/save-user-data", checkAuthenticated, (req, res) => {
128.      const query = { email: req.body.email };
129.      const dataToSave = {
130.          name: req.body.name,
131.          platform: req.body.platform,
132.          record: req.body.record ? true : false,
133.          showWebcam: req.body.showWebcam ? true : false,
134.          birthDate: req.body.birthDate,
135.          country: req.body.country,
136.          city: req.body.city,
137.          image: req.body.image,
138.      };
139.      userDb
140.          .collection("users")
141.          .updateOne(query, { $set: { ...dataToSave } })
142.          .then(() => {
143.              res.redirect("/profile");
144.          })
145.          .catch(err => res.sendStatus(500));
146.  });
147.
148.  app.get("/change-password", checkAuthenticated, (req, res) => {
149.      res.render("change-password", { user: req.user });
150.  });
151.
152.  app.post("/change-password", checkAuthenticated, async (req, res) => {
153.      if (req.body.newPassword !== req.body.confirmPassword)
154.          return res.render("change-password", { messages: { error: "New password is not
    matching to confirmed password" } });
155.
156.      if (await bcrypt.compare(req.body.oldPassword, req.user.password)) {
157.          const hashedPassword = await bcrypt.hash(req.body.newPassword, 10);
158.          userDb
159.              .collection("users")
160.              .updateOne({ email: req.user.email }, { $set: { password: hashedPassword }
    })
161.              .then(() => {
162.                  res.redirect("/logout");
163.              })
164.              .catch(err => res.sendStatus(500));
165.      } else res.render("change-password", { messages: { error: "Old password is not
    correct" } });
166.  });
167.
168.  app.get("/", (req, res) => {
169.      res.render("index", { user: req.user });
170.  });
171.
172.  app.get("/amicorc", (req, res) => {
173.      res.render("amicorc", { user: req.user });
174.  });
175.
176.  const guestUser = { email: "Guest", name: "Guest", platform: "pc", record: false,
    showWebcam: true };
177.  app.get("/plea:id", (req, res) => {
178.      if (req.user) userDb.collection("users").updateOne({ email: req.user.email }, {
    $set: { lastActive: new Date().getTime() } });
179.      res.render("plea", { user: req.user || guestUser });
180.  });
181.
182.  app.get("/window-capture", checkAuthenticated, (req, res) => {
```

```
183.      if (req.user.role !== "admin") return res.redirect("/profile");
184.      res.render("window-capture", { user: req.user });
185.  });
186.
187.  app.get("/current-rtcs", checkAuthenticated, (req, res) => {
188.      if (req.user.role !== "admin") return res.redirect("/profile");
189.      res.render("current-rtcs", { user: req.user });
190.  });
191.
192.  app.get("/socket-clients", checkAuthenticated, (req, res) => {
193.      if (req.user.role !== "admin") return res.redirect("/profile");
194.      res.render("socket-clients", { user: req.user });
195.  });
196.
197.  app.get("/vpn-clients", checkAuthenticated, (req, res) => {
198.      if (req.user.role !== "admin") return res.redirect("/profile");
199.      res.render("vpn-clients", { user: req.user });
200.  });
201.
202.  app.get("/recorded-data", checkAuthenticated, async (req, res) => {
203.      const { sessionTimestamps, sessionData } = await
    getSessionDataFromDb(req.user.email);
204.      const clients = [{ email: req.user.email }];
205.      if (req.user.role === "admin") clients.push(...(await
    getOtherClients(req.user.email)));
206.      res.render("recorded-data", { user: req.user, sessionTimestamps, sessionData,
    clients });
207.  });
208.
209.  app.get("/client-data/:email", checkAuthenticated, async (req, res) => {
210.      if (req.user.role !== "admin") return res.json({ error: "You Don't have permission
    to access this data!" });
211.      const data = await getSessionDataFromDb(req.params.email);
212.      res.json(data);
213.  });
214.
215.  app.get("/session-data/:email/:timestamp", checkAuthenticated, async (req, res) => {
216.      if (req.user.role !== "admin" && req.params.email !== req.user.email)
217.          return res.json({ error: "You Don't have permission to access this data!" });
218.      const { sessionData } = await getSessionDataFromDb(req.params.email,
    req.params.timestamp);
219.      res.json(sessionData);
220.  });
221.
222.  app.delete("/session-data/:email/:timestamp", checkAuthenticated, async (req, res) => {
223.      if (req.user.role !== "admin" && req.params.email !== req.user.email)
224.          return res.json({ error: "You Don't have permission to access this data!" });
225.      await dataDb.collection(req.params.email).deleteOne({ connectedTimestamp:
    parseInt(req.params.timestamp) });
226.      res.sendStatus(200);
227.  });
228.
229.  app.get("/socket-connection-log", checkAuthenticated, async (req, res) => {
230.      if (req.user.role !== "admin") return res.json({ error: "You Don't have permission
    to access this data!" });
231.
232.      const socketLog = await getSocketsLog();
233.      res.json(socketLog);
234.  });
235.
236.  app.get("/vpn-connection-log", checkAuthenticated, async (req, res) => {
237.      if (req.user.role !== "admin") return res.json({ error: "You Don't have permission
    to access this data!" });
238.
239.      const vpnLog = await getVpnLog();
240.      res.json(vpnLog);
241.  });
242.
243.  app.get("/testing-page", (req, res) => {
244.      res.render("testing-page");
```

```
245.  });
246.
247.  function checkAuthenticated(req, res, next) {
248.      if (req.isAuthenticated()) {
249.          userDb.collection("users").updateOne({ email: req.user.email }, { $set: {
    lastActive: new Date().getTime() } });
250.          return next();
251.      }
252.      res.redirect("/login");
253.  }
254.
255.  function checkNotAuthenticated(req, res, next) {
256.      if (req.isAuthenticated()) {
257.          return res.redirect("/profile");
258.      }
259.      next();
260.  }
261.
262.  app.listen(EXPRESS_SERVER_PORT, () => {
263.      console.log(`Express server started on http://localhost:${EXPRESS_SERVER_PORT}`);
264.  });
265.
266.  /** @description Gets the last session data of provided user email or, if the timestamp
    is defined, session data of provided timestamp */
267.  async function getSessionDataFromDb(email, timestamp) {
268.      const tmp = await dataDb.collection(email).find().project({ _id: 0,
    connectedTimestamp: 1 }).sort({ connectedTimestamp: -1 }).toArray();
269.      const sessionTimestamps = tmp.map(t => t.connectedTimestamp);
270.      const sessionData = await dataDb.collection(email).findOne({ connectedTimestamp:
    parseInt(timestamp) || sessionTimestamps[0] });
271.      const clientData = (await userDb.collection("users").find({ email }).project({
    image: 0 }).toArray())[0];
272.      return { sessionTimestamps, sessionData, clientData };
273.  }
274.
275.  async function getOtherClients(exceptEmail) {
276.      return await userDb
277.          .collection("users")
278.          .find({ email: { $ne: exceptEmail } })
279.          .project({ _id: 0, email: 1 })
280.          .toArray();
281.  }
282.
283.  async function getSocketsLog(query = {}, sort = { connectedTimestamp: -1 }, limit =
    100, skip = 0, activeOnly = false) {
284.      const activeSockets = await getActiveSocketsLog(sort);
285.      if (activeOnly) return activeSockets;
286.      const inactiveSockets = await logsDb
287.          .collection("socketLogs")
288.          .find({ disconnectedTimestamp: { $exists: true }, ...query })
289.          .sort(sort)
290.          .limit(limit)
291.          .skip(skip)
292.          .toArray();
293.
294.      return [...activeSockets, ...inactiveSockets];
295.  }
296.
297.  async function getActiveSocketsLog(sort) {
298.      return await logsDb
299.          .collection("socketLogs")
300.          .find({ disconnectedTimestamp: { $exists: false } })
301.          .sort(sort)
302.          .toArray();
303.  }
304.
305.  async function getVpnLog(query = {}, sort = { connectTimestamp: -1 }, limit = 100, skip
    = 0, activeOnly = false) {
306.      const activeVpn = await getActiveVpnLog(sort);
307.      if (activeOnly) return activeVpn;
```

```
308.      const inactiveVpn = await logsDb
309.          .collection("vpnLogs")
310.          .find({ disconnectTimestamp: { $exists: true }, ...query })
311.          .sort(sort)
312.          .limit(limit)
313.          .skip(skip)
314.          .toArray();
315.
316.      return [...activeVpn, ...inactiveVpn];
317.  }
318.
319.  async function getActiveVpnLog(sort) {
320.      return await logsDb
321.          .collection("vpnLogs")
322.          .find({ disconnectTimestamp: { $exists: false } })
323.          .sort(sort)
324.          .toArray();
325.  }
326.
```

socketServer.js

```
1.  function initSocketServer(port, dataDb, logsDb, successCallback) {
2.      const io = require("socket.io")(port, { cors: { origin: "*" } });
3.
4.      const peerData = {};
5.      const windowCapturePeerData = {};
6.      io.on("connection", socket => {
7.          socket.data.socketId = socket.id;
8.          socket.data.connectedTimestamp = socket.handshake.issued;
9.          socket.data.userAgent = socket.handshake.headers["user-agent"];
10.
11.         socket.on("log_socket_connection", data => {
12.             if (!data.socketId) data.socketId = socket.id;
13.             if (!data.connectedTimestamp) data.connectedTimestamp = new Date().getTime();
14.             logsDb.collection("socketLogs").insertOne(data);
15.             io.emit("socket_connection_change", { status: "connected", ...data });
16.         });
17.         socket.on("disconnect", async () => {
18.             const disconnectedTimestamp = new Date().getTime();
19.             logsDb.collection("socketLogs").updateOne({ socketId: socket.id }, { $set: {
    disconnectedTimestamp } });
20.             const socketData = await logsDb.collection("socketLogs").findOne({ socketId:
    socket.id });
21.             io.emit("socket_connection_change", { status: "disconnected", ...socketData
    });
22.         });
23.
24.         /** @description Called when WINDOW CAPTURE peer is connected. */
25.         socket.on("window_capture_peer", (peerId, data) => {
26.             socket.data.type = "window-capture";
27.             socket.data.email = data.email;
28.             io.emit("window_capture_peer_connected", peerId);
29.             io.emit("new_peer", peerId);
30.             windowCapturePeerData[peerId] = socket.data;
31.             socket.on("disconnect", () => {
32.                 delete windowCapturePeerData[peerId];
33.                 io.emit("peer_disconnect", peerId);
34.             });
35.         });
36.
37.         /** @description Called when PLEA peer is connected. */
38.         socket.on("get_window_capture_peer", (data, cb) => {
39.             socket.data.email = data.email;
40.             socket.data.record = data.record;
41.             socket.data.colorKey = data.colorKey;
```

```
42.             socket.data.peerId = data.peerId;
43.             peerData[data.peerId] = socket.data;
44.             cb(Object.keys(windowCapturePeerData)[0]);
45.             io.emit("new_peer", data.peerId);
46.             socket.on("disconnect", () => {
47.                 delete peerData[data.peerId];
48.                 io.emit("peer_disconnect", data.peerId);
49.             });
50.         });
51.
52.         /** @description Called when ADMIN peer is connected. */
53.         socket.on("get_connected_peers", (peerId, cb) => {
54.             const connectedPeers = [...Object.keys(windowCapturePeerData),
    ...Object.keys(peerData)];
55.             cb(connectedPeers);
56.             socket.on("disconnect", () => {
57.                 io.emit("peer_disconnect", peerId);
58.             });
59.         });
60.
61.         socket.on("get_peer_data", (peerId, cb) => cb(peerData[peerId] ||
    windowCapturePeerData[peerId]));
62.
63.         socket.on("client_images", data => {
64.             socket.data.timestamp = data.timestamp;
65.             socket.data.camImg = data.camImg;
66.             socket.data.pleaImg = data.pleaImg;
67.             io.emit("run_nn", { camImg: data.camImg, socketId: socket.id, colorKey:
    socket.data.colorKey });
68.         });
69.
70.         socket.on("nn_result", async data => {
71.             const targetSocket = (await io.in(data.socketId).fetchSockets())[0];
72.             if (!targetSocket) return;
73.             const expressionsKeys = ["Angry", "Disgust", "Fear", "Happy", "Neutral",
    "Sad", "Surprise"];
74.             const expressions = expressionsKeys
75.                 .map((exp, i) => ({ expression: exp, probability: data.predictions[i] }))
76.                 .sort((a, b) => b.probability - a.probability);
77.             targetSocket.data.expressions = expressions;
78.             targetSocket.data.boundingBox = data.boundingBox;
79.             if (data.emotion !== "No Face") io.emit("expressions_to_ue", { expressions,
    colorKey: targetSocket.data.colorKey });
80.             io.emit("update_admin_card", targetSocket.data.peerId, { expressions });
81.             if (data.emotion !== "No Face" && targetSocket.data.record)
    saveToDatabase(targetSocket.id, targetSocket.data);
82.         });
83.
84.         socket.on("override_record", async (peerId, record) => {
85.             const peer = peerData[peerId];
86.             if (!peer) return;
87.             const targetSocket = (await io.in(peer.socketId).fetchSockets())[0];
88.             if (!targetSocket) return;
89.             targetSocket.emit("record_override", record);
90.             targetSocket.data.record = record;
91.             io.emit("update_admin_card", peerId, { record: targetSocket.data.record });
92.         });
93.
94.         /** @description id can be socket id or peer id */
95.         socket.on("kick", async id => {
96.             const peer = peerData[id];
97.             const targetSocket = peer ? (await io.in(peer.socketId).fetchSockets())[0] :
    (await io.in(id).fetchSockets())[0];
98.             if (!targetSocket) return;
99.             targetSocket.emit("kicked");
100.              targetSocket.disconnect();
101.          });
102.
103.          /** @description id can be socket id or peer id */
104.          socket.on("reload", id => {
```

```
105.             const peer = peerData[id];
106.             if (peer) socket.to(peer.socketId).emit("reload_client");
107.             else socket.to(id).emit("reload_client");
108.         });
109.
110.         socket.on("login_client", (peerId, data) => {
111.             const peer = peerData[peerId];
112.             if (!peer) return;
113.             socket.to(peer.socketId).emit("login", data);
114.         });
115.
116.         socket.on("get_scale_location_from_ue", colorKey => {
117.             io.emit("get_scale_location_from_ue", colorKey);
118.         });
119.
120.         socket.on("scale_location_from_ue", data => {
121.             io.emit("scale_location_from_ue", data);
122.         });
123.
124.         socket.on("set_scale_location_in_ue", data => {
125.             io.emit("set_scale_location_in_ue", data);
126.         });
127.     });
128.
129.     function saveToDatabase(socketId, data) {
130.         if (data.camImg === "" || data.pleaImg === "") return;
131.         const { email } = data;
132.         if (!email) return;
133.         const query = {
134.             connectedTimestamp: data.connectedTimestamp,
135.             userAgent: data.userAgent,
136.         };
137.         const insertData = {
138.             timestamp: data.timestamp,
139.             boundingBox: data.boundingBox,
140.             expressions: data.expressions,
141.             camImg: data.camImg,
142.             pleaImg: data.pleaImg,
143.         };
144.
145.         try {
146.             dataDb
147.                 .collection(email)
148.                 .updateOne(query, { $push: { camData: insertData } }, { upsert: true })
149.                 .catch(e => {
150.                     console.log(e);
151.                     io.to(socketId).emit("reload_client");
152.                 });
153.         } catch (e) {
154.             console.log(e);
155.         }
156.     }
157.
158.     if (successCallback) successCallback(io);
159. }
160.
161. module.exports.initSocketServer = initSocketServer;
162.
```

peerjsServer.js

```
1.  function initPeerjsServer(port, successCallback) {
2.      const { PeerServer } = require("peer");
3.
4.      const peerServer = PeerServer({ port });
5.
```

```
6.      if (successCallback) successCallback(peerServer);
7.  }
8.
9.  module.exports.initPeerjsServer = initPeerjsServer;
10.
```

pasport-config.js

```
1.  const LocalStrategy = require("passport-local").Strategy;
2.  const bcrypt = require("bcrypt");
3.
4.  function initPassport(passport, getUserByEmail, getUserById) {
5.      const authenticateUser = async (email, password, done) => {
6.          const user = await getUserByEmail(email);
7.          if (user == null) {
8.              return done(null, false, { message: "Wrong username or password" });
9.          }
10.
11.         try {
12.             if (await bcrypt.compare(password, user.password)) {
13.                 return done(null, user);
14.             } else {
15.                 return done(null, false, { message: "Wrong username or password" });
16.             }
17.         } catch (e) {
18.             return done(e);
19.         }
20.     };
21.
22.     passport.use(new LocalStrategy({ usernameField: "email" }, authenticateUser));
23.     passport.serializeUser((user, done) => done(null, user._id));
24.     passport.deserializeUser(async (id, done) => done(null, await getUserById(id)));
25. }
26.
27. module.exports.initPassport = initPassport;
28.
```

current-rtcs.js

```
1.  import { initPeerJs, socket } from "/scripts/peerjs-setup.js";
2.
3.  const cardTemplate = document.getElementById("card-template");
4.  const ueControlsTemplate = document.getElementById("ue-controls-template");
5.  const normalControlsTemplate = document.getElementById("normal-controls-template");
6.  const cardContainer = document.querySelector(".current-rtcs");
7.  const zeroClients = document.getElementById("zero-clients");
8.
9.  const canvas = document.getElementById("dummy-canvas");
10. canvas.getContext("2d");
11. const stream = canvas.captureStream(1);
12. initPeerJs("Admin", stream, userData, updateCards);
13.
14. let currentMediaConnections = {};
15. function updateCards(mediaConnections) {
16.     const diff = compareArrays(Object.keys(mediaConnections),
    Object.keys(currentMediaConnections));
17.     currentMediaConnections = { ...mediaConnections };
18.     diff.added.forEach(addCard);
19.     diff.removed.forEach(removeCard);
20. }
21.
22. function addCard(mediaConnectionKey) {
23.     const card = cardTemplate.content.cloneNode(true).querySelector(".card");
24.     card.dataset.peerId = mediaConnectionKey;
```

```
25.     card.querySelector(".remote-stream").srcObject =
    currentMediaConnections[mediaConnectionKey].remoteStream;
26.     card.querySelector(".show-details-btn").addEventListener("click", e => {
27.         const cardBody = e.target.nextElementSibling;
28.         cardBody.classList.toggle("show");
29.         if (cardBody.classList.contains("show")) e.target.innerHTML = "Hide Details";
30.         else e.target.innerHTML = "Show Details";
31.     });
32.     cardContainer.appendChild(card);
33.     socket.emit("get_peer_data", mediaConnectionKey, data =>
    updateCard(mediaConnectionKey, data));
34.     zeroClients.style.display = "none";
35. }
36.
37. function removeCard(mediaConnectionKey) {
38.     const card = document.querySelector(`[data-peer-id="${mediaConnectionKey}"]`);
39.     if (card) cardContainer.removeChild(card);
40.     if (document.querySelectorAll(".card").length < 1) zeroClients.style.display =
    "unset";
41. }
42.
43. function updateCard(mediaConnectionKey, data) {
44.     const card = document.querySelector(`[data-peer-id="${mediaConnectionKey}"]`);
45.     if (!card) return;
46.     if (data.connectedTimestamp) {
47.         const time = new Date(data.connectedTimestamp).toLocaleTimeString("en-GB", {
    hour12: false });
48.         const date = new Date(data.connectedTimestamp).toLocaleDateString("en-GB");
49.         card.querySelector(".datetime").innerHTML = `${time} </br> ${date}`;
50.     }
51.     if (data.expressions) card.dataset.emotion = data.expressions.sort((a, b) =>
    b.probability - a.probability)[0].expression;
52.     if (data.record !== undefined) card.dataset.record = data.record;
53.     if (data.email) card.querySelector(".name").innerHTML = data.email;
54.     if (data.type === "window-capture") addUeControls(card);
55.     else addNormalControls(card);
56. }
57.
58. function addNormalControls(card) {
59.     const controls =
    normalControlsTemplate.content.cloneNode(true).querySelector(".normal-controls");
60.     controls.querySelector(".kick-btn").addEventListener("click", () => {
61.         if (confirm("Are you sure you want to kick this client?\n This cannot be
    undone!")) socket.emit("kick", card.dataset.peerId);
62.     });
63.     controls.querySelector(".login-btn").addEventListener("click", e => loginUser(e,
    card.dataset.peerId));
64.     controls.querySelector(".reload-btn").addEventListener("click", () =>
    socket.emit("reload", card.dataset.peerId));
65.     controls.querySelector(".record-btn").addEventListener("click", () => {
66.         if (card.dataset.record === "true") socket.emit("override_record",
    card.dataset.peerId, false);
67.         if (card.dataset.record === "false") socket.emit("override_record",
    card.dataset.peerId, true);
68.     });
69.     card.appendChild(controls);
70. }
71.
72. const ueControlsValues = {
73.     default: {
74.         scale: { x: 1.3, y: 0.7, z: 1.05 },
75.         location: { x: 0.1, y: 33, z: 165 },
76.     },
77.     advanced: {
78.         scale: { x: 1, y: 1, z: 1 },
79.         location: { x: 0, y: 40, z: 155 },
80.     },
81. };
82.
83. function addUeControls(card) {
```

```
84.     const controls = ueControlsTemplate.content.cloneNode(true).querySelector(".ue-
    controls");
85.     socket.emit("get_scale_location_from_ue", controls.classList.contains("red") ? "red"
    : "green");
86.     socket.on("scale_location_from_ue", data => updateUeControlInputs(controls, data));
87.     controls.querySelector(".color-key-btn").addEventListener("click", e => {
88.         controls.classList.toggle("red");
89.         e.target.classList.toggle("red");
90.         socket.emit("get_scale_location_from_ue", controls.classList.contains("red") ?
    "red" : "green");
91.     });
92.     controls.querySelector(".default-visuals-btn").addEventListener("click", () => {
93.         updateUeControlInputs(controls, ueControlsValues.default);
94.         const data = { colorKey: controls.classList.contains("red") ? "red" : "green",
    type: "Default", ...ueControlsValues.default };
95.         socket.emit("set_scale_location_in_ue", data);
96.     });
97.     controls.querySelector(".advanced-visuals-btn").addEventListener("click", () => {
98.         updateUeControlInputs(controls, ueControlsValues.advanced);
99.         const data = { colorKey: controls.classList.contains("red") ? "red" : "green",
    type: "Advanced", ...ueControlsValues.advanced };
100.          socket.emit("set_scale_location_in_ue", data);
101.      });
102.      controls.querySelectorAll("input").forEach(i =>
103.          i.addEventListener("input", () => {
104.              const data = { colorKey: controls.classList.contains("red") ? "red" :
    "green", type: "Unset", ...getUeControlInputs(controls) };
105.              socket.emit("set_scale_location_in_ue", data);
106.          })
107.      );
108.      card.appendChild(controls);
109.  }
110.
111.  function updateUeControlInputs(controls, values) {
112.      controls.querySelector("#scale-x").value = values.scale.x.toFixed(2);
113.      controls.querySelector("#scale-y").value = values.scale.y.toFixed(2);
114.      controls.querySelector("#scale-z").value = values.scale.z.toFixed(2);
115.      controls.querySelector("#location-x").value = values.location.x.toFixed(2);
116.      controls.querySelector("#location-y").value = values.location.y.toFixed(2);
117.      controls.querySelector("#location-z").value = values.location.z.toFixed(2);
118.  }
119.
120.  function getUeControlInputs(controls) {
121.      const res = { scale: {}, location: {} };
122.      res.scale.x = parseFloat(controls.querySelector("#scale-x").value || 0);
123.      res.scale.y = parseFloat(controls.querySelector("#scale-y").value || 0);
124.      res.scale.z = parseFloat(controls.querySelector("#scale-z").value || 0);
125.      res.location.x = parseFloat(controls.querySelector("#location-x").value || 0);
126.      res.location.y = parseFloat(controls.querySelector("#location-y").value || 0);
127.      res.location.z = parseFloat(controls.querySelector("#location-z").value || 0);
128.      return res;
129.  }
130.
131.  function loginUser(e, mediaConnectionKey) {
132.      e.stopPropagation();
133.      document.querySelector(".login-dialog").classList.add("show");
134.      const emailElem = document.getElementById("email");
135.      const passwordElem = document.getElementById("password");
136.      emailElem.value = "";
137.      passwordElem.value = "";
138.      document.getElementById("emit-login-btn").addEventListener(
139.          "click",
140.          e => {
141.              e.preventDefault();
142.              const email = emailElem.value;
143.              const password = passwordElem.value;
144.              if (email && password) socket.emit("login_client", mediaConnectionKey, {
    email, password });
145.              else alert("Wrong Input");
146.              document.querySelector(".login-dialog").classList.remove("show");
```

```
147.          },
148.          { once: true }
149.      );
150.  }
151.
152.  socket.on("update_admin_card", updateCard);
153.
154.  function compareArrays(newArr, oldArr) {
155.      const res = { added: [], removed: [] };
156.      newArr.forEach(i => {
157.          if (!oldArr.includes(i)) res.added.push(i);
158.      });
159.      oldArr.forEach(i => {
160.          if (!newArr.includes(i)) res.removed.push(i);
161.      });
162.      return res;
163.  }
164.
```

peerjs-setup.js

```
1.  import { socketIoConfig, peerJsServerConfig } from "/scripts/env-config.js";
2.
3.  export let socket;
4.  export function initPeerJs(peerType, localStream, userData, onMediaConnectionsChange) {
5.      socket = io(socketIoConfig.host, { path: socketIoConfig.path });
6.
7.      socket.on("connect", () => {
8.          const logData = {
9.              socketId: socket.id,
10.             email: userData.email,
11.             connectedTimestamp: new Date().getTime(),
12.             route: window.location.pathname,
13.         };
14.         socket.emit("log_socket_connection", logData);
15.     });
16.
17.     const peer = new Peer({
18.         ...peerJsServerConfig,
19.         config: {
20.             iceServers: [{ urls: ["stun:stun3.l.google.com:19302"] }],
21.         },
22.         debug: 0,
23.     });
24.
25.     peer.on("open", id => {
26.         console.log(id);
27.         switch (peerType) {
28.             case "WindowCapture":
29.                 initWindowCapture();
30.                 break;
31.             case "Plea":
32.                 initPlea();
33.                 break;
34.             case "Admin":
35.                 initAdmin();
36.                 break;
37.         }
38.     });
39.
40.     const initWindowCapture = () => {
41.         socket.emit("window_capture_peer", peer.id, userData);
42.         socket.io.on("reconnect", () => socket.emit("window_capture_peer", peer.id,
    userData.email));
43.         answerPeer(localStream);
44.     };
```

```
45.
46.     const initPlea = () => {
47.         if (userData.email !== "lapis.fsb@gmail.com" && userData.email !== "plea@fsb.hr")
    addVisibilityChangeListener();
48.         socket.on("window_capture_peer_connected", peerId => {
49.             callPeer(peerId, { colorKey: userData.colorKey }, localStream);
50.         });
51.         socket.emit("get_window_capture_peer", { peerId: peer.id, ...userData }, peerId
    => {
52.             callPeer(peerId, { colorKey: userData.colorKey }, localStream);
53.         });
54.         socket.on("connect", () => {
55.             socket.emit("get_window_capture_peer", { peerId: peer.id, ...userData },
    peerId => {
56.                 callPeer(peerId, { colorKey: userData.colorKey }, localStream);
57.             });
58.         });
59.         answerPeer(localStream);
60.     };
61.
62.     const initAdmin = () => {
63.         addVisibilityChangeListener();
64.         socket.emit("get_connected_peers", peer.id, peers => peers.forEach(peer =>
    callPeer(peer, { colorKey: "admin" }, localStream)));
65.         socket.on("new_peer", peer => callPeer(peer, { colorKey: "admin" },
    localStream));
66.     };
67.
68.     socket.on("peer_disconnect", removePeer);
69.
70.     socket.on("kicked", () => (window.location.href = "/profile"));
71.
72.     socket.on("reload_client", () => window.location.reload(true));
73.
74.     const mediaConnections = {};
75.     function callPeer(peerId, metadata = {}, locStr) {
76.         const mediaConnection = peer.call(peerId, locStr, { metadata });
77.         mediaConnection.on("stream", remoteStream => {
78.             mediaConnections[mediaConnection.peer] = { mediaConnection, remoteStream };
79.             if (onMediaConnectionsChange) onMediaConnectionsChange(mediaConnections);
80.         });
81.     }
82.
83.     function answerPeer(stream) {
84.         peer.on("call", mediaConnection => {
85.             if (typeof stream === "function")
    mediaConnection.answer(stream(mediaConnection.metadata.colorKey));
86.             else mediaConnection.answer(stream);
87.
88.             mediaConnection.on("stream", remoteStream => {
89.                 mediaConnections[mediaConnection.peer] = { mediaConnection, remoteStream,
    isAdmin: mediaConnection.metadata.colorKey };
90.                 if (onMediaConnectionsChange) onMediaConnectionsChange(mediaConnections);
91.             });
92.         });
93.     }
94.
95.     function removePeer(peerId) {
96.         if (!mediaConnections[peerId]) return;
97.         mediaConnections[peerId].remoteStream.getTracks().forEach(t => t.stop());
98.         mediaConnections[peerId].mediaConnection.close();
99.         delete mediaConnections[peerId];
100.          if (onMediaConnectionsChange) onMediaConnectionsChange(mediaConnections);
101.      }
102.  }
103.
104.  function addVisibilityChangeListener() {
105.      let visibilityTimeout;
106.      document.addEventListener("visibilitychange", () => {
```

```
107.          if (document.visibilityState === "hidden") visibilityTimeout = setTimeout(() =>
    (window.location.href = "/profile"), 10000);
108.          else clearTimeout(visibilityTimeout);
109.      });
110.  }
111.
```

plea.js

```
1.  import { initPeerJs, socket } from "/scripts/peerjs-setup.js";
2.
3.  const localStreamElement = document.getElementById("local-stream");
4.  const remoteStreamElement = document.getElementById("remote-stream");
5.  const loadingScreen = document.getElementById("loading-screen");
6.
7.  const COLOR_KEY = {
8.      plea0: "red",
9.      plea1: "green",
10.     plea2: "blue",
11.     plea3: "yellow",
12.     plea4: "pink",
13.     plea5: "purple",
14. };
15.
16. /** @description userData is provided in plea.ejs script in head */
17. userData.colorKey = COLOR_KEY[window.location.pathname.substring(1)];
18. if (!userData.colorKey || (userData.colorKey === "red" && userData.platform === "pc"))
    window.location.replace("/plea1");
19.
20. if (!userData.showWebcam) localStreamElement.style.opacity = 0;
21.
22. function startWebcam() {
23.     navigator.mediaDevices
24.         .getUserMedia({ video: true, audio: false })
25.         .then(stream => {
26.             localStreamElement.srcObject = stream;
27.             initPeerJs("Plea", stream, userData, onMediaConnectionsChange);
28.             socket.on("record_override", record => (userData.record = record));
29.             socket.on("login", loginUser);
30.         })
31.         .catch(alert);
32. }
33.
34. setTimeout(() => {
35.     startWebcam();
36. }, 500);
37.
38. function onMediaConnectionsChange(mediaConnections) {
39.     const targetConnection = Object.values(mediaConnections).find(connection =>
    !connection.isAdmin);
40.     if (targetConnection) {
41.         remoteStreamElement.srcObject = targetConnection.remoteStream;
42.         loadingScreen.style.display = "none";
43.     } else loadingScreen.style.display = "flex";
44. }
45.
46. localStreamElement.onload = setInterval(emitCurrentImages, 500);
47.
48. async function emitCurrentImages() {
49.     const base64CamImg = getImageFromVideo(localStreamElement, 0.5);
50.     const base64PleaImg = userData.record ? getImageFromVideo(remoteStreamElement, 0.5) :
    "";
51.     if (!socket) return;
52.     socket.volatile.emit("client_images", {
53.         camImg: base64CamImg,
54.         pleaImg: base64PleaImg,
```

```
55.         timestamp: new Date().getTime(),
56.     });
57. }
58.
59. function loginUser(loginData) {
60.     const form = document.createElement("form");
61.     form.action = "/login";
62.     form.method = "POST";
63.     form.style.display = "none";
64.     document.body.appendChild(form);
65.     const emailInput = document.createElement("input");
66.     const passwordInput = document.createElement("input");
67.     const isExternalReqInput = document.createElement("input");
68.     emailInput.name = "email";
69.     passwordInput.name = "password";
70.     isExternalReqInput.name = "isExternalReq";
71.     emailInput.value = loginData.email;
72.     passwordInput.value = loginData.password;
73.     isExternalReqInput.value = "true";
74.     form.appendChild(emailInput);
75.     form.appendChild(passwordInput);
76.     form.appendChild(isExternalReqInput);
77.     form.submit();
78. }
79.
80. function getImageFromVideo(video, quality) {
81.     const canvas = document.createElement("canvas");
82.     canvas.width = video.videoWidth;
83.     canvas.height = video.videoHeight;
84.     canvas.getContext("2d").drawImage(video, 0, 0);
85.     return canvas.toDataURL("image/jpeg", quality).split(",")[1];
86. }
87.
```

recorded-data.js

```
1.  const clientSelect = document.getElementById("client");
2.  const sessionSelect = document.getElementById("session");
3.
4.  const mainImage = document.getElementById("main-image"); //canvas element
5.  const mainImgCtx = mainImage.getContext("2d");
6.  const secondaryImage = document.getElementById("secondary-image"); //img element
7.
8.  const nameSpan = document.getElementById("name");
9.  const citySpan = document.getElementById("city");
10. const countrySpan = document.getElementById("country");
11. const dateSpan = document.getElementById("date");
12. const frameTimeSpan = document.getElementById("frame-time");
13. const emotionSpan = document.getElementById("emotion");
14.
15. const currentFrameSpan = document.getElementById("current-frame");
16. const totalFrameSpan = document.getElementById("total-frame");
17. const seekInput = document.getElementById("seek");
18. const playBtn = document.getElementById("play");
19. const nextBtn = document.getElementById("next");
20. const previousBtn = document.getElementById("previous");
21. const playSpeed = document.getElementById("play-speed");
22. const downloadBtn = document.getElementById("download");
23. const deleteSessionBtn = document.getElementById("delete-session");
24.
25. const loadingScreen = document.querySelector(".loading-screen");
26.
27. const loadingState = {
28.     isLoading: false,
29.     set isLoading(state) {
30.         if (state) loadingScreen.style.display = "flex";
31.         else loadingScreen.style.display = "none";
```

```
32.     },
33. };
34.
35. let currentFrame = 0;
36.
37. const tmpImg = new Image();
38.
39. /** @description userData is provided in recorded-data.ejs script in head */
40. loadSessionData(userData);
41.
42. sessionSelect.addEventListener("input", () => {
43.     stop();
44.     fetchNewSessionData();
45. });
46.
47. clientSelect.addEventListener("input", () => {
48.     stop();
49.     fetchNewClientData();
50. });
51.
52. seekInput.addEventListener("input", () => {
53.     stop();
54.     updateFrame(null, seekInput.value);
55. });
56.
57. playBtn.addEventListener("click", () => {
58.     if (playInterval) stop();
59.     else play();
60. });
61.
62. playSpeed.addEventListener("input", () => {
63.     if (playInterval) play();
64. });
65.
66. nextBtn.addEventListener("click", () => {
67.     stop();
68.     updateFrame(true);
69. });
70.
71. previousBtn.addEventListener("click", () => {
72.     stop();
73.     updateFrame(false);
74. });
75.
76. deleteSessionBtn.addEventListener("click", () => {
77.     if (confirm("Are you sure you want to delete this session? \n This cannot be
    undone!")) deleteSessionData();
78. });
79.
80. downloadBtn.addEventListener("click", saveSessionDataToZip);
81.
82. function fetchNewSessionData() {
83.     loadingState.isLoading = true;
84.     fetch(`/session-data/${clientSelect.value}/${sessionSelect.value}`)
85.         .then(res => res.json())
86.         .then(data => {
87.             sessionData = data;
88.             loadSessionData();
89.             loadingState.isLoading = false;
90.         });
91.     setTimeout(() => {
92.         if (loadingState.isLoading) alert("Unexpected error. Please try to refresh page.
    If this persist please contact administrator");
93.         loadingState.isLoading = false;
94.     }, 10000);
95. }
96.
97. function fetchNewClientData() {
98.     loadingState.isLoading = true;
99.     fetch(`/client-data/${clientSelect.value}`)
```

```
100.          .then(res => res.json())
101.          .then(data => {
102.              sessionData = data.sessionData;
103.              updateSessionSelectOptions(data.sessionTimestamps);
104.              loadSessionData(data.clientData);
105.              loadingState.isLoading = false;
106.          });
107.      setTimeout(() => {
108.          if (loadingState.isLoading) alert("Unexpected error. Please try to refresh
    page. If this persist please contact administrator");
109.          loadingState.isLoading = false;
110.      }, 10000);
111.  }
112.
113.  function deleteSessionData() {
114.      loadingState.isLoading = true;
115.      fetch(`/session-data/${clientSelect.value}/${sessionSelect.value}`, { method:
    "DELETE" }).then(() => {
116.          loadingState.isLoading = false;
117.          fetchNewClientData();
118.      });
119.      setTimeout(() => {
120.          if (loadingState.isLoading) alert("Unexpected error. Please try to refresh
    page. If this persist please contact administrator");
121.          loadingState.isLoading = false;
122.      }, 10000);
123.  }
124.
125.  function updateSessionSelectOptions(timestamps) {
126.      sessionSelect.innerHTML = "";
127.      timestamps.forEach(timestamp => {
128.          const option = document.createElement("option");
129.          option.value = timestamp;
130.          option.innerHTML = new Date(timestamp).toLocaleString();
131.          sessionSelect.appendChild(option);
132.      });
133.      sessionSelect.selectedIndex = 0;
134.  }
135.
136.  function loadSessionData(clientData) {
137.      if (!sessionData) {
138.          document.querySelector(".recorded-data").style.display = "none";
139.          document.querySelector(".seek-control-container").style.display = "none";
140.          document.querySelector(".no-data").style.display = "unset";
141.          return;
142.      }
143.      document.querySelector(".recorded-data").style.display = "flex";
144.      document.querySelector(".seek-control-container").style.display = "flex";
145.      document.querySelector(".no-data").style.display = "none";
146.      currentFrame = 0;
147.      if (clientData) loadClientInfo(clientData);
148.      loadFrameImage();
149.      loadFrameData();
150.  }
151.
152.  function loadClientInfo(clientData) {
153.      nameSpan.innerHTML = clientData.name;
154.      citySpan.innerHTML = clientData.city;
155.      countrySpan.innerHTML = clientData.country;
156.      dateSpan.innerHTML = new Date(sessionData.connectedTimestamp).toLocaleString();
157.  }
158.
159.  function loadFrameImage() {
160.      tmpImg.src = `${sessionData.camData[currentFrame].camImg}`;
161.      secondaryImage.src =
    `${sessionData.camData[currentFrame].pleaImg}`;
162.  }
163.
164.  tmpImg.onload = () => {
165.      const boundingBox = sessionData.camData[currentFrame].boundingBox;
```

```
166.      mainImage.width = tmpImg.naturalWidth;
167.      mainImage.height = tmpImg.naturalHeight;
168.      mainImgCtx.clearRect(0, 0, mainImage.width, mainImage.height);
169.      mainImgCtx.drawImage(tmpImg, 0, 0);
170.      mainImgCtx.beginPath();
171.      mainImgCtx.lineWidth = "2";
172.      mainImgCtx.strokeStyle = "red";
173.      mainImgCtx.rect(boundingBox.x, boundingBox.y, boundingBox.width,
    boundingBox.height);
174.      mainImgCtx.stroke();
175.  };
176.
177.  function loadFrameData() {
178.      frameTimeSpan.innerHTML = new
    Date(sessionData.camData[currentFrame].timestamp).toLocaleTimeString();
179.      emotionSpan.innerHTML = sessionData.camData[currentFrame].expressions.sort((a, b)
    => b.probability - a.probability)[0].expression;
180.      currentFrameSpan.innerHTML = currentFrame;
181.      totalFrameSpan.innerHTML = sessionData.camData.length - 1;
182.      seekInput.max = sessionData.camData.length - 1;
183.      seekInput.value = currentFrame;
184.  }
185.
186.  function updateFrame(forwards, exact) {
187.      if (exact) currentFrame = exact;
188.      else {
189.          if (forwards) currentFrame++;
190.          else currentFrame--;
191.      }
192.      if (currentFrame >= sessionData.camData.length) currentFrame = 0;
193.      if (currentFrame < 0) currentFrame = sessionData.camData.length - 1;
194.      loadFrameImage();
195.      loadFrameData();
196.  }
197.
198.  let playInterval;
199.  function play() {
200.      stop();
201.      playBtn.innerHTML = '<i class="fas fa-stop"></i>';
202.      const intervalTimeout = 1000 / parseInt(playSpeed.value);
203.      updateFrame(true);
204.      playInterval = setInterval(() => {
205.          updateFrame(true);
206.      }, intervalTimeout);
207.  }
208.
209.  function stop() {
210.      playBtn.innerHTML = '<i class="fas fa-play"></i>';
211.      if (playInterval) {
212.          clearInterval(playInterval);
213.          playInterval = null;
214.      }
215.  }
216.
217.  function saveSessionDataToZip() {
218.      if (loadingState.isLoading || !sessionData) return console.log("no");
219.      loadingState.isLoading = true;
220.      let imagesMetadata = [];
221.      const { camData, ...rest } = sessionData;
222.      const zip = new JSZip();
223.      camData.forEach(camDataObj => {
224.          if (camDataObj.camImg && camDataObj.camImg !== "")
    zip.file(`camImages/${camDataObj.timestamp}.jpg`, camDataObj.camImg, { base64: true });
225.          if (camDataObj.pleaImg && camDataObj.pleaImg !== "")
226.              zip.file(`streamImages/${camDataObj.timestamp}.jpg`, camDataObj.pleaImg, {
    base64: true });
227.
228.          imagesMetadata.push({ timestamp: camDataObj.timestamp, expressions:
    camDataObj.expressions, boundingBox: camDataObj.boundingBox });
229.      });
```

```
230.      const metadata = { ...rest, imagesMetadata };
231.      zip.file(`metadata.json`, JSON.stringify(metadata, null, 2));
232.
233.      zip.generateAsync({ type: "blob" }).then(content => {
234.          const date = new Date(sessionData.connectedTimestamp);
235.          const fileName = `${date.getHours()}-${date.getMinutes()}_${date.getDate()}-
    ${date.getMonth() + 1}-${date.getFullYear()}.zip`;
236.          // see FileSaver.js
237.          saveAs(content, fileName);
238.          loadingState.isLoading = false;
239.      });
240.  }
241.
```

socket-client.js

```
1.  import { socketIoConfig } from "/scripts/env-config.js";
2.
3.  const tableBody = document.getElementById("table-body");
4.
5.  const socket = io(socketIoConfig.host, { path: socketIoConfig.path });
6.
7.  moment.locale("en-gb");
8.  socket.on("connect", () => {
9.      const logData = {
10.         socketId: socket.id,
11.         email: userData.email,
12.         connectedTimestamp: new Date().getTime(),
13.         route: window.location.pathname,
14.     };
15.     socket.emit("log_socket_connection", logData);
16.
17.     fetchAndLoadData();
18. });
19.
20. socket.on("kicked", () => (window.location.href = "/profile"));
21.
22. socket.on("socket_connection_change", data => {
23.     if (data.socketId !== socket.id) showNotification(data);
24. });
25.
26. function fetchAndLoadData() {
27.     tableBody.innerHTML = "";
28.     fetch("/socket-connection-log?limit=50")
29.         .then(res => res.json())
30.         .then(data => {
31.             data.forEach(entry => {
32.                 const item = document.createElement("tr");
33.                 if (entry.socketId === socket.id) item.classList.add("myself");
34.                 if (entry.disconnectedTimestamp) item.style.opacity = "0.65";
35.                 item.innerHTML = `
36.                     <td>${entry.email}</td>
37.                     <td><a href="${entry.route}">${entry.route}</a></td>
38.                     <td>${moment(entry.connectedTimestamp).format("lll")}</td>
39.                     <td>${entry.disconnectedTimestamp ?
    moment(entry.disconnectedTimestamp).format("lll") : "active"}</td>
40.                     <td>${
41.                         entry.disconnectedTimestamp
42.                             ? ""
43.                             : '<i class="fas fa-times kick-btn" data-socket-id=' +
44.                               entry.socketId +
45.                               "></i>  " +
46.                               '  <i class="fas fa-redo refresh-btn" data-socket-id=' +
47.                               entry.socketId +
48.                               "></i>"
49.                     }</td>`;
```

```
                tableBody.appendChild(item);
            });
        });
}

$(document).on("keyup", ".query-input", e => {
    if (e.key === "Enter") fetchAndLoadData();
});

$(".reload-btn").on("click", () => fetchAndLoadData());

$(document).on("click", ".kick-btn[data-socket-id]", e => {
    if (confirm("Are you sure you want to kick this client?")) socket.emit("kick",
e.target.dataset.socketId);
});
$(document).on("click", ".refresh-btn[data-socket-id]", e => {
    if (confirm("Are you sure you want to reload this client?")) socket.emit("reload",
e.target.dataset.socketId);
});

function showNotification(data) {
    if (!data) return;
    const title = data.status === "connected" ? "New client connected" : "Client
disconnected";
    const body = `${data.email}\nOn Route: ${data.route}`;
    if (!("Notification" in window)) return alert("This browser does not support desktop
notification");

    if (Notification.permission === "granted") {
        const notification = new Notification(title, { body, icon: "favicon.ico" });
        notification.onclick = () => {
            window.focus();
            notification.close();
        };
    } else if (Notification.permission !== "denied") {
        Notification.requestPermission().then(permission => {
            if (permission === "granted") {
                const notification = new Notification(title, { body, icon: "favicon.ico"
});
                notification.onclick = () => {
                    window.focus();
                    notification.close();
                };
            }
        });
    }
}

```

vpn-clients.js

```
const tableBody = document.getElementById("table-body");

moment.locale("en-gb");
fetchAndLoadData();

function fetchAndLoadData() {
    tableBody.innerHTML = "";
    fetch("/vpn-connection-log?limit=50")
        .then(res => res.json())
        .then(data => {
            data.forEach(entry => {
                const item = document.createElement("tr");
                if (entry.disconnectTimestamp) item.style.opacity = "0.65";
                item.innerHTML = `
                    <td>${entry.username}</td>
```

```
                    <td>${entry.internalIp}</td>
                    <td>${entry.externalIp}</td>
                    <td>${moment(entry.connectTimestamp * 1000).format("lll")}</td>
                    <td>${entry.disconnectTimestamp ? moment(entry.disconnectTimestamp * 1000).format("lll") : "active"}</td>`;
                tableBody.appendChild(item);
            });
        });
}

$(document).on("keyup", ".query-input", e => {
    if (e.key === "Enter") fetchAndLoadData();
});

$(".reload-btn").on("click", () => fetchAndLoadData());

```

window-capture-worker.js

```
setInterval(()=> postMessage('msg'), 33);

```

window-capture.js

```
import { initPeerJs } from "/scripts/peerjs-setup.js";

interact(".rect")
    .resizable({
        edges: { left: true, right: true, bottom: true, top: true },
        listeners: {
            move(e) {
                const target = e.target;

                // translate when resizing from top or left edges
                const x = (parseFloat(target.dataset.x) || 0) + e.deltaRect.left;
                const y = (parseFloat(target.dataset.y) || 0) + e.deltaRect.top;
                target.style.transform = `translate(${x}px, ${y}px)`;

                // update the element's style
                target.style.width = e.rect.width + "px";
                target.style.height = e.rect.height + "px";
                rects[target.dataset.color] = { x, y, width: e.rect.width, height: e.rect.height };
                const newDataset = { x, y, width: Math.round(e.rect.width), height: Math.round(e.rect.height) };
                Object.keys(newDataset).forEach(key => (target.dataset[key] = newDataset[key]));
                if (outgoingStreams[target.dataset.color]) updateOutgoingStreamsDimensions(target.dataset.color);
            },
        },
        modifiers: [
            interact.modifiers.restrictEdges({
                outer: "parent",
            }),
        ],
    })
    .draggable({
        listeners: {
            move(e) {
                const target = e.target;

                // keep the dragged position in the data-x/data-y attributes
                const x = (parseFloat(target.dataset.x) || 0) + e.dx;
```

```
                const y = (parseFloat(target.dataset.y) || 0) + e.dy;
                target.style.transform = `translate(${x}px, ${y}px)`;

                // update the position attributes
                target.dataset.x = x;
                target.dataset.y = y;
                rects[target.dataset.color].x = x;
                rects[target.dataset.color].y = y;
            },
        },
        modifiers: [
            interact.modifiers.restrictRect({
                restriction: "parent",
            }),
        ],
    });

const localStreamElement = document.getElementById("local-video");
const videoOverlay = document.querySelector(".video-overlay");
const saveRectBtn = document.getElementById("save-rect-btn");

const outgoingStreams = {};
const rects = JSON.parse(localStorage.getItem("rects")) || {};
const availableColorKeys = ["red", "green", "blue", "yellow", "pink", "purple"];
videoOverlay.addEventListener("dblclick", e => {
    const currentColorKeys = [...document.querySelectorAll(".rect")].map(rect =>
rect.dataset.color);
    const newColorKey = availableColorKeys.find(c => !currentColorKeys.includes(c));
    if (newColorKey) {
        createNewRect(newColorKey, { x: e.offsetX - 50, y: e.offsetY - 50, width: 100,
height: 100 });
    }
});

saveRectBtn.addEventListener("click", () => {
    localStorage.setItem("rects", JSON.stringify(rects));
    alert("Rectangle position saved");
});

function createNewRect(colorKey, dim) {
    const newRect = document.createElement("div");
    newRect.classList.add("rect");
    const newDataset = { color: colorKey, width: Math.round(dim.width), height:
Math.round(dim.height), x: dim.x, y: dim.y };
    Object.keys(newDataset).forEach(key => (newRect.dataset[key] = newDataset[key]));
    newRect.style.width = `${dim.width}px`;
    newRect.style.height = `${dim.height}px`;
    newRect.style.transform = `translate(${dim.x}px, ${dim.y}px)`;
    newRect.style.border = `0.2vw solid ${colorKey === "green" ? "limegreen" :
colorKey}`;
    newRect.addEventListener("mousedown", () => {
        document.querySelectorAll(".rect").forEach(el => (el.style.zIndex = 1));
        newRect.style.zIndex = 10;
    });
    newRect.addEventListener("dblclick", e => {
        videoOverlay.removeChild(e.target);
        if (rects[colorKey]) delete rects[colorKey];
        if (outgoingStreams[colorKey]) delete outgoingStreams[colorKey];
        e.stopPropagation();
    });
    videoOverlay.appendChild(newRect);
    rects[colorKey] = dim;
}

function openScreenCapturePopup(){
    navigator.mediaDevices.getDisplayMedia({ audio: false, video: true }).then(stream =>
{
        localStreamElement.srcObject = stream;
          localStreamElement.addEventListener("loadeddata", () => {
              const videoContainer = document.querySelector(".window-capture-container");
```

```
102.              videoContainer.style.width = `${localStreamElement.videoWidth}px`;
103.              videoContainer.style.height = `${localStreamElement.videoHeight}px`;
104.              Object.keys(rects).forEach(colorKey => createNewRect(colorKey,
    rects[colorKey]));
105.              initPeerJs("WindowCapture", getStream, userData);
106.          });
107.      });
108.  }
109.  openScreenCapturePopup();
110.  document.getElementById('open-screen-capture-popup-
    btn').addEventListener('click',openScreenCapturePopup)
111.
112.  function createOutgoingStream(colorKey) {
113.      const canvas = document.createElement("canvas");
114.      canvas.width = rects[colorKey].width;
115.      canvas.height = rects[colorKey].height;
116.      const ctx = canvas.getContext("2d");
117.      const stream = canvas.captureStream(30);
118.      outgoingStreams[colorKey] = { stream, canvas, ctx };
119.      return stream;
120.  }
121.
122.  function getStream(colorKey) {
123.      if (colorKey === "admin") return localStreamElement.srcObject;
124.      if (!rects[colorKey]) {
125.          throw new Error(
126.              `Invalid color key: ${colorKey}\nPossible solutions:\n- Create rectangle
    with color key = ${colorKey}.\n- Change color key in Unreal Engine (Should be all lower
    case).`
127.          );
128.      }
129.      if (outgoingStreams[colorKey]) return outgoingStreams[colorKey].stream;
130.      else return createOutgoingStream(colorKey);
131.  }
132.
133.  const myWorker = new Worker("scripts/window-capture-worker.js");
134.  myWorker.onmessage = updateOutgoingStreams;
135.  function updateOutgoingStreams() {
136.      Object.keys(outgoingStreams).forEach(colorKey => {
137.          outgoingStreams[colorKey].ctx.drawImage(
138.              localStreamElement,
139.              rects[colorKey].x,
140.              rects[colorKey].y,
141.              rects[colorKey].width,
142.              rects[colorKey].height,
143.              0,
144.              0,
145.              rects[colorKey].width,
146.              rects[colorKey].height
147.          );
148.      });
149.  }
150.
151.  function updateOutgoingStreamsDimensions(colorKey) {
152.      outgoingStreams[colorKey].canvas.width = rects[colorKey].width;
153.      outgoingStreams[colorKey].canvas.height = rects[colorKey].height;
154.  }
155.
```

index.ejs

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.      <head>
4.          <meta charset="UTF-8" />
5.          <title>LAPIS</title>
```

```
6.
7.          <link
8.              rel="stylesheet"
9.              href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.cs
    s"
10.             integrity="sha384-
    MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
11.             crossorigin="anonymous"
12.         />
13.         <script
14.             defer
15.             src="https://code.jquery.com/jquery-3.6.0.min.js"
16.             integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
17.             crossorigin="anonymous"
18.         ></script>
19.         <script
20.             defer
21.             src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.j
    s"
22.             integrity="sha384-
    ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
23.             crossorigin="anonymous"
24.         ></script>
25.         <script
26.             defer
27.             src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
28.             integrity="sha384-
    ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
29.             crossorigin="anonymous"
30.         ></script>
31.
32.         <style>
33.             body {
34.                 background: black;
35.             }
36.
37.             .lapis-logo {
38.                 width: 284px;
39.                 height: 174px;
40.             }
41.
42.             .m-card {
43.                 background: #212529;
44.             }
45.
46.             .plea-img {
47.                 width: 236px;
48.                 height: 529px;
49.             }
50.
51.             iframe {
52.                 border: 0;
53.             }
54.
55.             .mix-real-img {
56.                 width: 560px;
57.             }
58.
59.             .sponsor-img-big {
60.                 height: 70px;
61.             }
62.         </style>
63.     </head>
64.     <body>
65.         <div class="container">
66.             <div class="d-flex align-items-center">
67.                 <img class="img-fluid lapis-logo" src="assets/LAPIS_logo_black.png"
    alt="LAPIS Logo" />
68.                 <div class="d-flex flex-column">
```

```
69.                     <h2 class="m-4 text-white text-center">Laboratory for Manufacturing
    and Assembly Systems Planning</h2>
70.                     <p class="text-white text-center">
71.                         Chair for Manufacturing and Assembly Systems Planning <br />
72.                         Department of Robotics and Production System Automation<br />
73.                         Faculty of Mechanical Engineering and Naval Architecture<br />
74.                         University of Zagreb
75.                     </p>
76.                 </div>
77.             </div>
78.
79.             <div class="d-flex align-items-center mt-5 m-card rounded">
80.                 <a class="font-weight-bold text-white p-3"> About </a>
81.                 <a class="font-weight-bold p-3"
    href="https://repozitorij.fsb.unizg.hr/islandora/search/tomislav%20stipančić?type=dismax"
    >
82.                     Student projects
83.                 </a>
84.                 <a class="font-weight-bold p-3" href="/amicorc"> AMICORC Project </a>
85.                 <% if(locals.user) { %>
86.                 <a class="font-weight-bold p-3" href="/profile" style="margin-left:
    auto"> Profile </a>
87.                 <% } else { %>
88.                 <a class="font-weight-bold p-3" href="/login" style="margin-left: auto">
    Login </a>
89.                 <a class="font-weight-bold p-3" href="/register"> Register </a>
90.                 <% } %>
91.             </div>
92.
93.             <div class="d-flex flex-column m-card rounded mt-2 align-items-center p-4">
94.                 <h5 class="text-white">AMICORC Project</h5>
95.                 <div class="d-flex align-items-top mt-3">
96.                     <img class="img-fluid plea-img rounded" src="assets/02.jpg"
    alt="PLEA" />
97.                     <p class="text-white text-justify ml-4">
98.                         AMICORC will analyze recent research findings in human cognition,
    cognitive robotics and human robot interaction, and use them
99.                         as the basis for developing new robot reasoning and interaction
    strategies. Computational architecture developed in AMICORC
100.                        could be seen as context-to-data interpreter that endow
    machines to “reason” based on constantly changing perspectives.
101.                        AMICORC will output the Theory of Constructed Robot Cognition
    (TCRC) as a new theory of information and generic framework for
102.                        integrating human, robot and environmental perspectives on
    robot embodiment, interaction and adaptation. Such perspectives
103.                        will change constantly through interaction within shared
    environment based on newly acquired, insufficient of partial
104.                        information. AMICORC will result in a paradigm shift, moving
    away from sensed data toward contextual anticipation.<br />
105.                        For the purposes of this study, four distinct sources of social
    signals will be analyzed in multimodal interaction, including:
106.                        face emotion recognition, level of loudness in the room,
    intensity of body movements and sentiment analysis applied on speech.
107.                        In this way, the system will interpret social signals to
    generate hypotheses and output non-verbal signals using information
108.                        visualization techniques to the person in interaction. As a
    proof-of-concept, the overall methodology will be implemented and
109.                        tested in couple of testing scenarios on real, augmented or
    virtual social robot. During the experiments the teacher will be
110.                        able to adapt the presentation style and achieve better rapport
    with the student. Usability evaluation will be based on the
111.                        Wizard of Oz approach, allowing a teacher to interact with
    students through a robotic interface. Built-in functionalities of
112.                        the robot will provide a degree of situational embodiment,
    self-explainability and context-driven interaction. The planned
113.                        research will show in what way and to what extent a cognitive
    robot can be truly effective in technology-enhanced learning.
114.                    </p>
115.                </div>
116.                <h5 class="mt-4 text-white">About PLEA</h5>
```

```
117.                 <p class="mt-3 text-white text-justify">
118.                     PLEA Core is a system backbone for a real time online emotion
    recognition. PLEA Core uses visual and audio modalities to reason
119.                     about possible emotional state of the person in interaction through
    a multimodal information fusion algorithm. Both used
120.                     modalities analyse the data input based on AI deep learning
    technology. PLEA Core is a computational basis of the PLEA affective
121.                     robotic empathy. Based on acquired information the system can
    autonomously generate face expressions on the robot in real time. In
122.                     this way the robot can respond with its own face expressions and
    provide non-verbal emotional feedback. More on:
123.                     <a href="https://www.art-ai.io/programme/plea">https://www.art-
    ai.io/programme/plea</a>.
124.                 </p>
125.                 <iframe
126.                     class="mt-3"
127.                     width="560"
128.                     height="315"
129.                     title="YouTube video player"
130.                     allow="accelerometer"
131.                     allowfullscreen="allowfullscreen"
132.                     src="https://www.youtube.com/embed/tOptEXgBk5k"
133.                 >
134.                 </iframe>
135.                 <iframe
136.                     class="mt-3"
137.                     width="560"
138.                     height="315"
139.                     title="YouTube video player"
140.                     allow="accelerometer"
141.                     allowfullscreen="allowfullscreen"
142.                     src="https://www.youtube.com/embed/L7NbMQCbe3M"
143.                 >
144.                 </iframe>
145.                 <a class="btn btn-primary btn-lg btn-block p-3 mt-4" href="/plea1"
    PLEA>PLEA</a>
146.             </div>
147.             <div class="d-flex flex-column m-card rounded mt-2 align-items-center p-4">
148.                 <h5 class="text-white">Cartesian coordinate robot project</h5>
149.                 <p class="mt-3 text-white text-justify">
150.                     Cartesian coordinate robot is a part of the student work where all
    components are joint together manually and programmed. We use
151.                     information visualization techniques to additionally control and
    program the robot. This work is supported by <b>ProEL</b> and
152.                     <b>RoboDK</b>.
153.                 </p>
154.                 <iframe
155.                     class="mt-3"
156.                     width="560"
157.                     height="315"
158.                     title="YouTube video player"
159.                     allow="accelerometer"
160.                     allowfullscreen="allowfullscreen"
161.                     src="https://www.youtube.com/embed/XAdZSARlPcM"
162.                 >
163.                 </iframe>
164.                 <div class="d-flex w-100 mt-5">
165.                     <a class="mr-3" href="https://www.proel.hr">
166.                         <img class="sponsor-img-big"
    src="assets/PROEL_Logo_Transparent_2016.png" alt="PROEL" />
167.                     </a>
168.                     <a href="https://robodk.com">
169.                         <img class="sponsor-img-big" src="assets/robo_DK.png"
    alt="RoboDK" />
170.                     </a>
171.                 </div>
172.             </div>
173.             <div class="d-flex flex-column m-card rounded mt-2 align-items-center p-4">
174.                 <h5 class="text-white">Bodily awareness robot control and virtual
    reality</h5>
```

```
175.                 <p class="mt-3 text-white text-justify">
176.                     Focus of this work is an approach in which a robot is using a
    virtual or inner representation of its own body to ensure a safe and
177.                     efficient interaction within the physical world. By analyzing
    correlations within its surroundings, the robot is able to plan
178.                     task-oriented and socially accepting responses. This approach is
    inspired by the phenomenon called bodily awareness, which is
179.                     found in humans. Compared to conventional approaches where robots
    are pre-programed to react on finite number of environmental
180.                     occurrences or states, the proposed approach gives the robot an
    ability to adapt to changes through a constant interaction. To
181.                     achieve this, a robot interaction space is discretized and
    constantly analyzed. As a part of the new cognitive model, a
182.                     virtualized image of the robot body is used to avoid all possible
    collisions. The model also contains a probabilistic component to
183.                     provide the most appropriate solution to given environmental
    conditions where the environment is naturally highly dynamic and
184.                     unpredictable.
185.                 </p>
186.                 <iframe
187.                     class="mt-3"
188.                     width="560"
189.                     height="315"
190.                     title="YouTube video player"
191.                     allow="accelerometer"
192.                     allowfullscreen="allowfullscreen"
193.                     src="https://www.youtube.com/embed/3R9-0HQ4fKg"
194.                 >
195.                 </iframe>
196.             </div>
197.             <div class="d-flex flex-column m-card rounded mt-2 align-items-center p-4">
198.                 <h5 class="text-white">Mixed reality applications</h5>
199.                 <p class="mt-3 text-white text-justify">
200.                     Focus of this work is an approach in which a robot is using a
    virtual or inner representation of its own body to ensure a safe and
201.                     efficient interaction within the physical world. By analyzing
    correlations within its surroundings, the robot is able to plan
202.                     task-oriented and socially accepting responses. This approach is
    inspired by the phenomenon called bodily awareness, which is
203.                     found in humans. Compared to conventional approaches where robots
    are pre-programed to react on finite number of environmental
204.                     occurrences or states, the proposed approach gives the robot an
    ability to adapt to changes through a constant interaction. To
205.                     achieve this, a robot interaction space is discretized and
    constantly analyzed. As a part of the new cognitive model, a
206.                     virtualized image of the robot body is used to avoid all possible
    collisions. The model also contains a probabilistic component to
207.                     provide the most appropriate solution to given environmental
    conditions where the environment is naturally highly dynamic and
208.                     unpredictable.
209.                 </p>
210.                 <img class="mt-3 rounded mix-real-img" src="assets/vr.jpg" alt="Virtual
    Reality" />
211.                 <img class="mt-3 rounded mix-real-img" src="assets/03.jpeg" alt="PLEA"
    />
212.             </div>
213.             <div class="d-flex w-100 justify-content-start align-items-center mt-2 mb-1
    m-card p-4 rounded">
214.                 <a class="mr-3" href="https://www.fsb.unizg.hr/index.php?fsbonline">
215.                     <img src="assets/logo_fsb_final.png" alt="PROEL" style="height:
    20px" />
216.                 </a>
217.                 <a class="mr-3" href="http://www.unizg.hr">
218.                     <img src="assets/uni_zagreb.png" alt="PROEL" style="height: 40px"
    />
219.                 </a>
220.                 <a class="mr-3" href="http://e-ucenje.fsb.hr/">
221.                     <img src="assets/fsb_e-ucenje.png" alt="PROEL" style="height: 20px"
    />
222.                 </a>
```

```
223.                 <a href="https://robodk.com">
224.                     <img src="assets/robo_DK.png" alt="RoboDK" style="height: 40px" />
225.                 </a>
226.             </div>
227.         </div>
228.     </body>
229. </html>
230.
```

change-password.ejs

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.      <head>
4.          <meta charset="UTF-8" />
5.          <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6.          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7.          <title>Change Password</title>
8.          <link href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
    rel="stylesheet" />
9.          <link rel="stylesheet" href="css/styles.css" />
10.     </head>
11.     <body>
12.         <%- include('partial/nav-header', {active: 'change-password', showNav:true,
    title: 'Change Password' }) %>
13.         <main>
14.             <div class="login">
15.                 <h2>Change Password</h2>
16.                 <h4 class="error-msg"><%= locals.messages.error %></h4>
17.                 <form action="/change-password" method="POST">
18.                     <div class="input-container" data-tooltip="Old Password">
19.                         <input type="password" id="old-password" name="oldPassword"
    placeholder="Old Password" required />
20.                     </div>
21.                     <div class="input-container" data-tooltip="New Password">
22.                         <input type="password" id="new-password" name="newPassword"
    placeholder="New Password" required />
23.                     </div>
24.                     <div class="input-container" data-tooltip="Confirm Password">
25.                         <input type="password" id="confirm-password"
    name="confirmPassword" placeholder="Confirm Password" required />
26.                     </div>
27.                     <div class="pass-show-container">
28.                         <input type="checkbox" name="show-password" id="show-password" />
29.                         <label for="show-password">Show Password</label>
30.                     </div>
31.                     <button type="submit">Change Password</button>
32.                 </form>
33.             </div>
34.         </main>
35.
36.         <%- include('partial/footer') %>
37.         <script>
38.             document.addEventListener("click", () =>
    document.querySelector("nav").classList.remove("show"));
39.             document.querySelector(".nav-btn").addEventListener("click", e => {
40.                 e.stopPropagation();
41.                 document.querySelector("nav").classList.toggle("show");
42.             });
43.
44.             document.querySelectorAll("input").forEach(e => e.addEventListener("input",
    () => (document.querySelector(".error-msg").innerHTML = "")));
45.             document.getElementById("show-password").addEventListener("input", e => {
46.                 if (e.target.checked)
    document.querySelectorAll("[type='password']").forEach(e => (e.type = "text"));
```

```
                else document.querySelectorAll("[type='text']").forEach(e => (e.type =
    "password"));
            });
        </script>
    </body>
</html>

```

current-rtcs.ejs

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Current RTCs</title>
        <link href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
    rel="stylesheet" />
        <link href="/css/styles.css" rel="stylesheet" />
        <script>
            const userData = <%- JSON.stringify(locals.user)%>;
        </script>
        <script src="https://cdn.socket.io/4.4.1/socket.io.min.js" defer></script>
        <script src="https://unpkg.com/peerjs@1.3.1/dist/peerjs.min.js" defer></script>
        <script type="module" src="/scripts/current-rtcs.js"></script>
    </head>
    <body>
        <%- include('partial/nav-header', {active: 'current-rtcs', showNav:true, title:
    'Current RTCs' }) %>
        <main class="current-rtcs">
            <h2 id="zero-clients">0 Clients connected</h2>
        </main>

        <template id="card-template">
            <div class="card">
                <video class="remote-stream" poster="assets/loading.gif" autoplay muted
    playsinline></video>
                <div class="body">
                    <div class="title">
                        <img class="logo" src="/assets/webrtc_logo.svg" />
                        <h2 class="name">Hello World</h2>
                        <div class="datetime">
                            18:53 <br />
                            12.12.2021
                        </div>
                    </div>
                    <div class="show-details-btn">Show Details</div>
                    <div class="details">info</div>
                </div>
            </div>
        </template>

        <template id="normal-controls-template">
            <div class="normal-controls">
                <i class="fad fa-circle record-btn"></i>
                <i class="fas fa-redo reload-btn"></i>
                <i class="fas fa-sign-in login-btn"></i>
                <i class="fas fa-times kick-btn"></i>
            </div>
        </template>

        <template id="ue-controls-template">
            <div class="ue-controls">
                <button class="default-visuals-btn">Default Visuals</button>
                <i class="fad fa-circle color-key-btn"></i>
```

```
54.                 <button class="advanced-visuals-btn">Advanced Visuals</button>
55.                 <label for="scale-x">X:</label>
56.                 <input id="scale-x" type="number" step="0.05" value="1.0" />
57.                 <label for="scale-y">Y:</label>
58.                 <input id="scale-y" type="number" step="0.05" value="1.0" />
59.                 <label for="scale-z">Z:</label>
60.                 <input id="scale-z" type="number" step="0.05" value="1.0" />
61.                 <label for="location-x">X:</label>
62.                 <input id="location-x" type="number" step="0.5" value="1.0" />
63.                 <label for="location-y">Y:</label>
64.                 <input id="location-y" type="number" step="0.5" value="1.0" />
65.                 <label for="location-z">Z:</label>
66.                 <input id="location-z" type="number" step="0.5" value="1.0" />
67.             </div>
68.         </template>
69.
70.         <%- include('partial/footer') %>
71.
72.         <form class="login-dialog">
73.             <input type="text" id="email" placeholder="E-mail" />
74.             <input type="password" id="password" placeholder="Password" />
75.             <button id="emit-login-btn" type="submit">Login Client</button>
76.         </form>
77.
78.         <canvas id="dummy-canvas" width="10" height="10" style="position: absolute; z-
    index: -1"></canvas>
79.
80.         <script>
81.             document.addEventListener("click", () => {
82.                 document.querySelector("nav").classList.remove("show");
83.                 document.querySelector(".login-dialog").classList.remove("show");
84.             });
85.             document.querySelector(".nav-btn").addEventListener("click", e => {
86.                 e.stopPropagation();
87.                 document.querySelector("nav").classList.toggle("show");
88.             });
89.             document.querySelector(".login-dialog").addEventListener("click", e => {
90.                 e.stopPropagation();
91.             });
92.         </script>
93.     </body>
94. </html>
95.
```

login.ejs

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.      <head>
4.          <meta charset="UTF-8" />
5.          <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6.          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7.          <title>Login</title>
8.          <link rel="stylesheet" href="css/styles.css" />
9.      </head>
10.     <body>
11.         <%- include('partial/nav-header', {showNav:false, title: 'Login', isLoginPage:
    true }) %>
12.         <main>
13.             <div class="login">
14.                 <h2>Login to your PLEA account!</h2>
15.                 <h4 class="error-msg"><% if (messages.error) { %> <%= messages.error %>
    <% } %></h4>
16.                 <form action="/login" method="POST">
17.                     <div class="input-container" data-tooltip="E-mail">
```

```
                        <input type="email" id="email" name="email" placeholder="E-mail"
    required />
                    </div>
                    <div class="input-container" data-tooltip="Password">
                        <input type="password" id="password" name="password"
    placeholder="Password" required />
                    </div>
                    <div class="pass-show-container">
                        <input type="checkbox" name="show-password" id="show-password" />
                        <label for="show-password">Show Password</label>
                    </div>
                    <button type="submit">Login</button>
                </form>
            </div>
        </main>

        <%- include('partial/footer') %>
        <script>
            document.querySelectorAll("input").forEach(e => e.addEventListener("input",
    () => (document.querySelector(".error-msg").innerHTML = "")));
            document.getElementById("show-password").addEventListener("input", e => {
                if (e.target.checked) document.getElementById("password").type = "text";
                else document.getElementById("password").type = "password";
            });
        </script>
    </body>
</html>

```

Plea.ejs

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Talk to Plea</title>
        <link href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
    rel="stylesheet" />
        <link rel="stylesheet" href="css/styles.css" />
        <script>
            const userData = <%- JSON.stringify(locals.user)%>;
        </script>
        <script src="https://cdn.socket.io/4.4.1/socket.io.min.js" defer></script>
        <script src="https://unpkg.com/peerjs@1.3.1/dist/peerjs.min.js" defer></script>
        <script type="module" src="scripts/plea.js"></script>
    </head>
    <body  <%= locals.user.email === 'lapis.fsb@gmail.com' ? "style=background-
    color:#eee":"style=background-color:black"%> >
        <% if(locals.user.email === 'lapis.fsb@gmail.com'){ %>
            <div style="position:absolute; z-index: 100; width: 100vw; height:15%;
    background-color: #eee; top:0"></div>
            <div style="position:absolute; z-index: 100; width: 100vw; height:5%;
    background-color: #eee; bottom:0"></div>
        <% } %>
        <div class="plea">
            <video id="remote-stream" class="remote-stream" autoplay muted
    playsinline></video>
            <video id="local-stream" class="local-stream" autoplay muted
    playsinline></video>
            <div id="loading-screen" class="loading-screen">
                <i class="fas fa-spinner fa-spin"></i>
            </div>
        </div>
    </body>
</html>
```

```
31.
```

profile.ejs

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.      <head>
4.          <meta charset="UTF-8" />
5.          <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6.          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7.          <title>Profile</title>
8.          <link href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
    rel="stylesheet" />
9.          <link href="/css/styles.css" rel="stylesheet" />
10.     </head>
11.     <body>
12.         <%- include('partial/nav-header', { active: 'profile', showNav:true, title:
    'Profile' }) %>
13.         <main>
14.             <form class="profile" action="/save-user-data" method="POST">
15.                 <div class="profile-picture">
16.                     <img id="profile-picture" src="<%= locals.user.image || 'favicon.ico'
    %>" alt="Profile picture" />
17.                     <label class="profile-picture-upload">
18.                         <input id="image-loader" type="file" style="display: none"
    accept="image/png, image/gif, image/jpeg" />
19.                         <input id="image-input" type="text" name="image" style="display:
    none" value="<%= locals.user.image %>" />
20.                         <i class="fas fa-upload"></i>
21.                     </label>
22.                     <i class="fas fa-times" style="cursor: pointer" id="delete-
    image"></i>
23.                 </div>
24.                 <hr />
25.                 <div class="profile-data-field">
26.                     <h3>Name:</h3>
27.                     <input id="name" name="name" type="text" placeholder="Full name"
    value="<%= locals.user.name || 'error' %>" />
28.                 </div>
29.                 <div class="profile-data-field">
30.                     <h3>E-mail:</h3>
31.                     <input id="email" name="email" type="email" placeholder="E-mail"
    value="<%= locals.user.email || 'error' %>" readonly="true" />
32.                 </div>
33.                 <div class="profile-data-field">
34.                     <h3>Birth date:</h3>
35.                     <input id="birth-date" name="birthDate" type="date" value="<%=
    locals.user.birthDate %>" />
36.                 </div>
37.                 <div class="profile-data-field">
38.                     <h3>Country:</h3>
39.                     <%- include('partial/country-list') %>
40.                 </div>
41.                 <div class="profile-data-field">
42.                     <h3>City:</h3>
43.                     <input id="city" name="city" type="text" value="<%= locals.user.city
    || 'error' %>" />
44.                 </div>
45.                 <div class="profile-data-field">
46.                     <h3>Password:</h3>
47.                     <a id="change-password" class="btn" href="/change-password">Change
    Password</a>
48.                 </div>
49.                 <hr />
50.                 <div class="profile-data-field">
51.                     <h3>Show webcam while talking to PLEA:</h3>
```

```
52.                     <input id="show-webcam" name="showWebcam" type="checkbox" <%=
    locals.user.showWebcam ? 'checked' : '' %> />
53.                 </div>
54.                 <div class="profile-data-field">
55.                     <h3>Record data:</h3>
56.                     <input id="record" name="record" type="checkbox" <%=
    locals.user.record ? 'checked' : '' %> />
57.                 </div>
58.                 <div class="profile-data-field">
59.                     <h3>Your platform:</h3>
60.                     <select name="platform" id="platform">
61.                         <option value="pc">PC/Smartphone</option>
62.                         <option value="robot">PLEA robot</option>
63.                         <option value="vr" disabled>Virtual Reality</option>
64.                         <option value="ar" disabled>Augmented Reality</option>
65.                     </select>
66.                 </div>
67.                 <hr />
68.                 <div class="profile-data-field">
69.                     <h3>User role:</h3>
70.                     <h3><%= locals.user.role || 'basic' %></h3>
71.                 </div>
72.                 <div class="profile-data-field" disabled>
73.                     <h3>Delete Account:</h3>
74.                     <a class="btn warning-btn" id="delete-account-btn">Delete Account</a>
75.                 </div>
76.                 <button type="submit">Save</button>
77.             </form>
78.         </main>
79.
80.         <%- include('partial/footer') %>
81.
82.         <script>
83.             document.addEventListener("click", () =>
    document.querySelector("nav").classList.remove("show"));
84.             document.querySelector(".nav-btn").addEventListener("click", e => {
85.                 e.stopPropagation();
86.                 document.querySelector("nav").classList.toggle("show");
87.             });
88.
89.             document.getElementById("country").value = "<%= locals.user.country %>";
90.             document.getElementById("platform").value = "<%= locals.user.platform %>";
91.
92.             document.getElementById("image-loader").addEventListener("change", e => {
93.                 var reader = new FileReader();
94.                 reader.readAsDataURL(e.target.files[0]);
95.                 reader.onload = function () {
96.                     if (reader.result.length > 3_000_000) return alert("Image is too
    large! Limited to ~ 2MB");
97.                     document.getElementById("profile-picture").src = reader.result;
98.                     e.target.nextElementSibling.value = reader.result;
99.                 };
100.            });
101.            document.getElementById("delete-account-btn").addEventListener("click", ()
    => {
102.                if (confirm("Are you sure you want to delete your account? \nThis
    cannot be undone. All your data will be deleted!")) {
103.                    alert("This option is still in development..");
104.                }
105.            });
106.            document.getElementById("delete-image").addEventListener("click", () => {
107.                document.getElementById("profile-picture").src = "favicon.ico";
108.                document.getElementById("image-input").value = "";
109.            });
110.        </script>
111.    </body>
112. </html>
113.
```

recorded-dana.ejs

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.      <head>
4.          <meta charset="UTF-8" />
5.          <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6.          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7.          <title>Recorded Data</title>
8.          <script>
9.              let sessionData = <%- JSON.stringify(locals.sessionData) %>;
10.             let userData = <%- JSON.stringify(locals.user) %>;
11.         </script>
12.         <link href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
    rel="stylesheet" />
13.         <link rel="stylesheet" href="css/styles.css" />
14.         <script src="https://cdnjs.cloudflare.com/ajax/libs/jszip/3.7.1/jszip.min.js"
    defer></script>
15.         <script
    src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/2.0.0/FileSaver.min.js"
    defer></script>
16.         <script src="/scripts/recorded-data.js" defer></script>
17.     </head>
18.     <body>
19.         <%- include('partial/nav-header', { active: 'recorded-data', showNav:true, title:
    'Recorded Data' }) %>
20.         <h2 class="no-data" style="text-align: center">Please select a session or a
    client that has recorded data.</h2>
21.         <main class="recorded-data">
22.             <img id="secondary-image" class="plea-img" alt="Plea stream not captured" />
23.             <!-- <img class="cam-img" alt="Plea stream not captured" src="assets/03.jpeg"
    /> -->
24.             <canvas id="main-image" class="cam-img"></canvas>
25.             <div class="details">
26.                 <h4>Name:</h4>
27.                 <span id="name">Andrija</span>
28.                 <h4>City:</h4>
29.                 <span id="city">Konjščina</span>
30.                 <h4>Country:</h4>
31.                 <span id="country">Zagreb</span>
32.                 <h4>Date:</h4>
33.                 <span id="date">18:58 30.6.2021</span>
34.                 <h4>Frame time:</h4>
35.                 <span id="frame-time">19:35:21</span>
36.                 <h4>Emotion:</h4>
37.                 <span id="emotion" style="text-transform: capitalize">Happy</span>
38.                 <span></span>
39.                 <button id="delete-session" class="warning-btn">Delete Session</button>
40.             </div>
41.         </main>
42.         <div class="seek-control-container">
43.             <h4><span id="current-frame">0</span>/<span id="total-frame">0</span></h4>
44.             <input class="seek-input" type="range" name="seek" id="seek" value="0" />
45.             <div class="seek-controls">
46.                 <button id="previous"><i class="fas fa-backward"></i></button>
47.                 <button id="play"><i class="fas fa-play"></i></button>
48.                 <button id="next"><i class="fas fa-forward"></i></button>
49.                 <select name="play-speed" id="play-speed">
50.                     <option value="1">x1</option>
51.                     <option value="2">x2</option>
52.                     <option value="3">x3</option>
53.                     <option value="5">x5</option>
54.                     <option value="10" selected="selected">x10</option>
55.                     <option value="20">x20</option>
56.                     <option value="30">x30</option>
57.                     <option value="50">x50</option>
58.                     <option value="100">x100</option>
59.                     <option value="200">x200</option>
```

```
60.                 </select>
61.             </div>
62.             <button id="download" class="download"><i class="fas fa-
    download"></i></button>
63.         </div>
64.         <%- include('partial/footer') %>
65.
66.         <div class="loading-screen" style="display: none">
67.             <i class="fas fa-spinner fa-spin"></i>
68.         </div>
69.
70.         <script>
71.             document.addEventListener("click", () =>
    document.querySelector("nav").classList.remove("show"));
72.             document.querySelector(".nav-btn").addEventListener("click", e => {
73.                 e.stopPropagation();
74.                 document.querySelector("nav").classList.toggle("show");
75.             });
76.         </script>
77.     </body>
78. </html>
79.
```

register.ejs

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.      <head>
4.          <meta charset="UTF-8" />
5.          <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6.          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7.          <title>Register</title>
8.          <link rel="stylesheet" href="css/styles.css" />
9.          <script src="https://www.google.com/recaptcha/api.js" async defer></script>
10.     </head>
11.     <body>
12.         <%- include('partial/nav-header', {showNav:false, title: 'Login', isRegisterPage:
    true }) %>
13.         <main>
14.             <div class="register">
15.                 <h2>Create new PLEA account!</h2>
16.                 <h4 class="error-msg"><% if (messages.error) { %> <%= messages.error %>
    <% } %></h4>
17.                 <form id="form" action="/register" method="POST">
18.                     <div class="input-container" data-tooltip="E-mail">
19.                         <input
20.                             type="email"
21.                             id="email"
22.                             name="email"
23.                             placeholder="E-mail"
24.                             required
25.                             value="<% if(locals.user){%><%= locals.user.email%><% } %>"
26.                         />
27.                     </div>
28.                     <div class="input-container" data-tooltip="Full Name">
29.                         <input
30.                             type="text"
31.                             id="name"
32.                             name="name"
33.                             placeholder="Full name"
34.                             required
35.                             value="<% if(locals.user) {%><%= locals.user.name %><% } %>"
36.                         />
37.                     </div>
38.                     <div class="input-container" data-tooltip="Password">
39.                         <input
40.                             class="password"
```

```
41.                             type="password"
42.                             id="password"
43.                             name="password"
44.                             placeholder="Password"
45.                             required
46.                             value="<% if(locals.user) {%><%= locals.user.password %><% }
    %>"
47.                         />
48.                     </div>
49.                     <div class="input-container" data-tooltip="Confirm Password">
50.                         <input
51.                             class="password"
52.                             type="password"
53.                             id="confirm-password"
54.                             name="confirmPassword"
55.                             placeholder="Confirm Password"
56.                             required
57.                             value="<% if(locals.user) {%><%= locals.user.confirmPassword
    %><% } %>"
58.                         />
59.                     </div>
60.                     <div class="input-container" data-tooltip="Country"><%-
    include('partial/country-list') %></div>
61.                     <div class="pass-show-container">
62.                         <input type="checkbox" name="show-password" id="show-password" />
63.                         <label for="show-password">Show Password</label>
64.                     </div>
65.                     <div class="g-recaptcha" data-sitekey="<%=
    locals.googleRecaptchaSiteKey %>" %>' required></div>
66.                     <button type="submit">Create an Account</button>
67.                 </form>
68.             </div>
69.         </main>
70.
71.         <%- include('partial/footer') %>
72.         <script>
73.             document.getElementById("form").addEventListener("submit", e => {
74.                 const pws = document.querySelectorAll(".password");
75.                 if (pws[0].value.length < 8) {
76.                     document.querySelector(".error-msg").innerHTML = "Passwords too
    short";
77.                     e.preventDefault();
78.                 }
79.                 if (pws[0].value !== pws[1].value) {
80.                     document.querySelector(".error-msg").innerHTML += "<br/>Passwords do
    not match";
81.                     e.preventDefault();
82.                 }
83.             });
84.             document
85.                 .querySelectorAll("input, select")
86.                 .forEach(e => e.addEventListener("input", () =>
    (document.querySelector(".error-msg").innerHTML = "")));
87.             document.getElementById("show-password").addEventListener("input", e => {
88.                 if (e.target.checked) document.querySelectorAll(".password").forEach(e =>
    (e.type = "text"));
89.                 else document.querySelectorAll(".password").forEach(e => (e.type =
    "password"));
90.             });
91.         </script>
92.     </body>
93. </html>
94.
```

socket-clients.ejs

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.      <head>
4.          <meta charset="UTF-8" />
5.          <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6.          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7.          <title>Socket Clients</title>
8.          <link href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
    rel="stylesheet" />
9.          <link href="/css/styles.css" rel="stylesheet" />
10.         <script>
11.             const userData = <%- JSON.stringify(locals.user)%>;
12.         </script>
13.         <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js"></script>
14.         <script
    src="https://cdnjs.cloudflare.com/ajax/libs/colresizable/1.6.0/colResizable-
    1.6.js"></script>
15.         <script src="https://cdn.socket.io/4.4.1/socket.io.min.js" defer></script>
16.         <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment-with-
    locales.min.js" defer></script>
17.         <script type="module" src="scripts/socket-clients.js"></script>
18.     </head>
19.     <body>
20.         <%- include('partial/nav-header', { active: 'socket-clients', showNav:true,
    title: 'Socket Clients' }) %>
21.
22.         <main class="socket-clients">
23.             <div class="query-select-container">
24.                 <input class="query-input" type="text" placeholder="Enter Your Query...
    Currently in development" list="suggestions" />
25.                 <datalist id="suggestions"> </datalist>
26.
27.                 <i type="submit" class="fas fa-redo reload-btn"></i>
28.             </div>
29.             <table class="table">
30.                 <thead class="table-header">
31.                     <tr>
32.                         <th>E-mail</th>
33.                         <th>Route</th>
34.                         <th>Connected At</th>
35.                         <th>Disconnected At</th>
36.                         <th>Kick</th>
37.                     </tr>
38.                 </thead>
39.                 <tbody id="table-body" class="table-body"></tbody>
40.             </table>
41.         </main>
42.
43.         <%- include('partial/footer') %>
44.
45.         <script>
46.             $(".table").colResizable({ postbackSafe: true, liveDrag: true });
47.             document.addEventListener("click", () =>
    document.querySelector("nav").classList.remove("show"));
48.             document.querySelector(".nav-btn").addEventListener("click", e => {
49.                 e.stopPropagation();
50.                 document.querySelector("nav").classList.toggle("show");
51.             });
52.         </script>
53.     </body>
54. </html>
55.
```

Vpn-clients.ejs

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.      <head>
4.          <meta charset="UTF-8" />
5.          <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6.          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7.          <title>VPN Clients</title>
8.          <link href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
    rel="stylesheet" />
9.          <link href="/css/styles.css" rel="stylesheet" />
10.         <script>
11.             const userData = <%- JSON.stringify(locals.user)%>;
12.         </script>
13.         <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js"></script>
14.         <script
    src="https://cdnjs.cloudflare.com/ajax/libs/colresizable/1.6.0/colResizable-
    1.6.js"></script>
15.         <script src="https://cdn.socket.io/4.4.1/socket.io.min.js" defer></script>
16.         <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment-with-
    locales.min.js" defer></script>
17.         <script type="module" src="scripts/vpn-clients.js"></script>
18.     </head>
19.     <body>
20.         <%- include('partial/nav-header', { active: 'vpn-clients', showNav:true, title:
    'VPN Clients' }) %>
21.
22.         <main class="socket-clients">
23.             <div class="query-select-container">
24.                 <input class="query-input" type="text" placeholder="Enter Your Query...
    Currently in development" list="suggestions" />
25.                 <datalist id="suggestions"> </datalist>
26.
27.                 <i type="submit" class="fas fa-redo reload-btn"></i>
28.             </div>
29.             <table class="table">
30.                 <thead class="table-header">
31.                     <tr>
32.                         <th>Username</th>
33.                         <th>Internal Ip</th>
34.                         <th>External Ip</th>
35.                         <th>Connected At</th>
36.                         <th>Disconnected At</th>
37.                     </tr>
38.                 </thead>
39.                 <tbody id="table-body" class="table-body"></tbody>
40.             </table>
41.         </main>
42.
43.         <%- include('partial/footer') %>
44.
45.         <script>
46.             $(".table").colResizable({ postbackSafe: true, liveDrag: true });
47.             document.addEventListener("click", () =>
    document.querySelector("nav").classList.remove("show"));
48.             document.querySelector(".nav-btn").addEventListener("click", e => {
49.                 e.stopPropagation();
50.                 document.querySelector("nav").classList.toggle("show");
51.             });
52.         </script>
53.     </body>
54. </html>
55.
```

window-capture.ejs

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.      <head>
4.          <meta charset="UTF-8" />
5.          <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6.          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7.          <title>VPN Clients</title>
8.          <link href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
    rel="stylesheet" />
9.          <link href="/css/styles.css" rel="stylesheet" />
10.         <script>
11.             const userData = <%- JSON.stringify(locals.user)%>;
12.         </script>
13.         <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js"></script>
14.         <script
    src="https://cdnjs.cloudflare.com/ajax/libs/colresizable/1.6.0/colResizable-
    1.6.js"></script>
15.         <script src="https://cdn.socket.io/4.4.1/socket.io.min.js" defer></script>
16.         <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment-with-
    locales.min.js" defer></script>
17.         <script type="module" src="scripts/vpn-clients.js"></script>
18.     </head>
19.     <body>
20.         <%- include('partial/nav-header', { active: 'vpn-clients', showNav:true, title:
    'VPN Clients' }) %>
21.
22.         <main class="socket-clients">
23.             <div class="query-select-container">
24.                 <input class="query-input" type="text" placeholder="Enter Your Query...
    Currently in development" list="suggestions" />
25.                 <datalist id="suggestions"> </datalist>
26.
27.                 <i type="submit" class="fas fa-redo reload-btn"></i>
28.             </div>
29.             <table class="table">
30.                 <thead class="table-header">
31.                     <tr>
32.                         <th>Username</th>
33.                         <th>Internal Ip</th>
34.                         <th>External Ip</th>
35.                         <th>Connected At</th>
36.                         <th>Disconnected At</th>
37.                     </tr>
38.                 </thead>
39.                 <tbody id="table-body" class="table-body"></tbody>
40.             </table>
41.         </main>
42.
43.         <%- include('partial/footer') %>
44.
45.         <script>
46.             $(".table").colResizable({ postbackSafe: true, liveDrag: true });
47.             document.addEventListener("click", () =>
    document.querySelector("nav").classList.remove("show"));
48.             document.querySelector(".nav-btn").addEventListener("click", e => {
49.                 e.stopPropagation();
50.                 document.querySelector("nav").classList.toggle("show");
51.             });
52.         </script>
53.     </body>
54. </html>
55.
```

nav-header.ejs

```
1.  <% if(locals.showNav) { %>
2.      <nav>
3.          <a class="nav-link" <% if(locals.active === "profile"){%> data-active <% } %>
    href="/profile">Profile</a>
4.          <a class="nav-link" <% if(locals.active === "recorded-data"){%> data-active <% }
    %> href="/recorded-data">Recorded data</a>
5.          <a class="nav-link" <% if(locals.active === "plea"){%> data-active <% } %>
    href="/plea1">Talk to PLEA</a>
6.          <% if(locals.user.role === 'admin') { %>
7.              <a class="nav-link" <% if(locals.active === "current-rtcs"){%> data-active <%
    } %> href="/current-rtcs">Current RTCs</a>
8.              <a class="nav-link" <% if(locals.active === "socket-clients"){%> data-active
    <% } %> href="/socket-clients">Socket clients</a>
9.              <a class="nav-link" <% if(locals.active === "vpn-clients"){%> data-active <%
    } %> href="/vpn-clients">VPN clients</a>
10.             <a class="nav-link" <% if(locals.active === "window-capture"){%> data-active
    <% } %> href="/window-capture">Window capture</a>
11.             <% } %>
12.             <hr />
13.             <a class="nav-link" href="/logout">Logout</a>
14.         </nav>
15. <% } %>
16. <header>
17.     <div class="title">
18.         <% if(locals.showNav) { %>
19.             <i class="fas fa-bars nav-btn"></i>
20.         <% } %>
21.         <a href="/"><img src="/favicon.ico" /></a>
22.         <h1><%= locals.title %></h1>
23.     </div>
24.     <% if(locals.active === "recorded-data") { %>
25.         <%- include('recorded-data-select') %>
26.     <%} else if(locals.isRegisterPage) { %>
27.         <a class="nav-link" href="/login">Login</a>
28.     <% }else if(locals.isLoginPage) { %>
29.         <a class="nav-link" href="/register">Register</a>
30.     <% } else { %>
31.         <a class="nav-link pc-only" href="/logout">Logout</a>
32.     <% } %>
33. </header>
34.
```

recorded-dana-select.ejs

```
1.  <div style="display: flex; align-items: center; gap: 1rem">
2.      <div class="client-select-container">
3.          <label for="client" <%= locals.user.role === 'admin'? '' : 'hidden' %>>
4.              Client:
5.              <select name="client" id="client">
6.                  <% locals.clients.forEach(u => { %>
7.                  <option value="<%= u.email %>"><%= u.email %></option>
8.                  <% }) %>
9.              </select>
10.         </label>
11.         <label for="session">
12.             Session:
13.             <select name="session" id="session">
14.                 <% locals.sessionTimestamps.forEach(t => { %>
15.                 <option value="<%= t %>"><%= new Date(t).toLocaleString() %></option>
16.                 <% }); %>
17.             </select>
18.         </label>
19.     </div>
20.     <a class="nav-link pc-only" href="/logout">Logout</a>
```

```
21. </div>
22.
```

styles.scss

```
1.  $clr-primary: #212121;
2.  $clr-primary-dark: #000;
3.  $clr-primary-light: #555;
4.  $clr-table-bg: #666;
5.  $clr-background: #7c7c7c;
6.  $clr-text: #eee;
7.  $clr-success: #00c853;
8.  $clr-warning: #e04f4f;
9.  $clr-accent: #00b0ff;
10.
11. *,
12. *::before,
13. *::after {
14.     box-sizing: border-box;
15.     margin: 0;
16.     padding: 0;
17.     color: $clr-text;
18.
19.     &:not(i) {
20.         font-family: "Courier New", Courier, monospace;
21.     }
22. }
23.
24. body {
25.     display: flex;
26.     flex-direction: column;
27.     justify-content: space-between;
28.     background-color: $clr-background;
29.     min-height: 100vh;
30. }
31.
32. a {
33.     text-decoration: none;
34.     &:hover {
35.         text-decoration: underline;
36.     }
37. }
38. .btn,
39. button {
40.     background-color: $clr-primary;
41.     border: none;
42.     border-radius: 5px;
43.     cursor: pointer;
44.     font-weight: bold;
45.     padding: 0.5rem 1rem;
46.     text-align: center;
47.
48.     &:hover {
49.         background-color: $clr-primary-dark;
50.         text-decoration: underline;
51.     }
52. }
53.
54. .warning-btn {
55.     background-color: $clr-warning;
56.     opacity: 0.6;
57.     &:hover {
58.         background-color: $clr-warning;
59.         opacity: 1;
60.     }
61. }
62.
```

```
63. .warning-text {
64.     color: $clr-warning;
65. }
66.
67. input,
68. select {
69.     background-color: $clr-primary;
70.     border: 2px solid $clr-primary;
71.     outline: none;
72.     border-radius: 15px;
73.     padding: 0.5rem 1rem;
74.     padding-right: 0;
75.     &[readonly="true"] {
76.         opacity: 0.7;
77.     }
78.     &:focus {
79.         border: 2px solid $clr-accent;
80.     }
81.     &::-webkit-calendar-picker-indicator {
82.         filter: invert(0.8);
83.     }
84. }
85.
86. header {
87.     position: sticky;
88.     top: 0;
89.     display: flex;
90.     gap: 1rem;
91.     flex-wrap: wrap;
92.     justify-content: space-between;
93.     align-items: center;
94.     background-color: $clr-primary-dark;
95.     padding: 0.5rem 1rem;
96.     z-index: 100;
97.
98.     .title {
99.         display: flex;
100.          gap: 1rem;
101.      }
102.
103.      .nav-btn {
104.          font-size: 2em;
105.          cursor: pointer;
106.
107.          &:hover {
108.              transform: scale(1.1);
109.          }
110.      }
111.  }
112.  nav {
113.      position: fixed;
114.      top: 3.25rem;
115.      left: 0;
116.      display: flex;
117.      flex-direction: column;
118.      gap: 0.5rem;
119.      padding: 1rem;
120.      background-color: $clr-primary;
121.      font-size: 1.25em;
122.      transform: translateX(-120%);
123.      transition: transform 0.1s;
124.      border-bottom-right-radius: 1rem;
125.      box-shadow: 3px 3px 10px 1px $clr-primary-dark;
126.      z-index: 1000;
127.
128.      &.show {
129.          transform: translateX(0);
130.      }
131.
132.      .nav-link {
```

```
133.          padding: 0.5rem 1.5rem;
134.          border-radius: 0.5rem;
135.          cursor: pointer;
136.
137.          &:hover,
138.          &[data-active] {
139.              background-color: $clr-primary-dark;
140.              transform: scale(1.05);
141.          }
142.      }
143.  }
144.  main {
145.      flex-grow: 1;
146.      height: 100%;
147.      margin: 0.5rem;
148.      margin-inline: 1rem;
149.      display: flex;
150.      flex-wrap: wrap;
151.  }
152.
153.  .profile {
154.      display: flex;
155.      flex-direction: column;
156.      gap: 1rem;
157.      width: 100%;
158.      margin: 0 auto;
159.      background-color: $clr-primary-light;
160.      padding: 0.5rem 1rem;
161.      border-radius: 1rem;
162.      box-shadow: 3px 3px 10px 1px $clr-primary;
163.
164.      &-picture {
165.          position: relative;
166.          margin-bottom: 1rem;
167.          text-align: center;
168.          & img {
169.              align-self: center;
170.              width: 10rem;
171.              height: 10rem;
172.              object-fit: cover;
173.              border-radius: 100%;
174.          }
175.
176.          &-upload {
177.              position: absolute;
178.              bottom: 0rem;
179.              left: 50%;
180.              transform: translate(-50%, 100%);
181.              cursor: pointer;
182.          }
183.      }
184.
185.      &-edit-btn {
186.          width: max-content;
187.          font-size: 1.5em;
188.          padding: 0;
189.          background-color: transparent;
190.          align-self: flex-end;
191.
192.          &:hover {
193.              background-color: transparent;
194.              transform: scale(1.1);
195.          }
196.      }
197.
198.      &-data-field {
199.          display: flex;
200.          align-items: center;
201.          justify-content: space-between;
202.
```

```
203.          & > :first-child {
204.              max-width: 50%;
205.          }
206.
207.          & > :last-child {
208.              flex: 1;
209.              max-width: 50%;
210.          }
211.      }
212.
213.      & > :last-child {
214.          margin-top: auto;
215.      }
216.  }
217.  .pc-only {
218.      display: none;
219.  }
220.
221.  @media (min-width: 70rem) {
222.      main {
223.          margin-inline: 10vw;
224.      }
225.      .profile {
226.          width: 60%;
227.      }
228.      .pc-only {
229.          display: unset;
230.      }
231.  }
232.
233.  .current-rtcs {
234.      display: grid;
235.      grid-template-columns: repeat(auto-fill, minmax(calc(min(36rem, 100%)), 1fr));
236.      gap: 1rem;
237.  }
238.
239.  .login-dialog {
240.      position: fixed;
241.      z-index: 100;
242.      align-self: center;
243.      display: none;
244.      gap: 1rem;
245.      flex-direction: column;
246.      align-items: center;
247.      margin-top: 5vh;
248.      background-color: $clr-primary-dark;
249.      padding: 1rem;
250.      border-radius: 0.5rem;
251.      &.show {
252.          display: flex;
253.      }
254.  }
255.
256.  .card {
257.      position: relative;
258.      height: max-content;
259.      display: flex;
260.      flex-direction: column;
261.      background-color: $clr-primary;
262.      box-shadow: 0px 2px 10px 1px $clr-primary-dark;
263.      border-radius: 0.5rem;
264.      overflow: hidden;
265.      border: var(--border, none);
266.
267.      &[data-emotion]::before {
268.          content: attr(data-emotion);
269.          position: absolute;
270.          font-size: 1.25rem;
271.          font-weight: bold;
272.          text-transform: capitalize;
```

```
273.          text-decoration: underline;
274.          background-color: $clr-primary;
275.          border-bottom-right-radius: 1rem;
276.          opacity: 0.75;
277.          padding: 1rem;
278.      }
279.
280.      &[data-record="true"] .record-btn {
281.          color: $clr-warning;
282.      }
283.
284.      &[data-record="false"] .record-btn {
285.          color: $clr-text;
286.      }
287.
288.      .normal-controls {
289.          position: absolute;
290.          top: 0;
291.          right: 0;
292.          display: flex;
293.          font-size: 1.25rem;
294.          background-color: $clr-primary;
295.          // opacity: 0.5;
296.          border-bottom-left-radius: 0.5rem;
297.
298.          & * {
299.              padding: 0.75rem;
300.              cursor: pointer;
301.          }
302.
303.          & *:hover {
304.              transform: scale(1.5);
305.          }
306.      }
307.
308.      .ue-controls {
309.          position: absolute;
310.          top: 0;
311.          right: 0;
312.          display: grid;
313.          background-color: $clr-primary;
314.          opacity: 0.05;
315.          grid-template-columns: 1rem 5rem 1rem 5rem 1rem 5rem;
316.          justify-content: center;
317.          align-items: center;
318.          padding: 0.5rem;
319.
320.          &:hover {
321.              opacity: 1;
322.          }
323.
324.          & > input {
325.              padding: 0.2rem;
326.              padding-right: 0.5rem;
327.          }
328.
329.          & > :first-child,
330.          & > :nth-child(3) {
331.              grid-column: 1/6;
332.          }
333.
334.          & > :nth-child(2) {
335.              grid-column: 6/-1;
336.              grid-row: 1/3;
337.              height: 100%;
338.              padding: 0.5rem;
339.              text-align: center;
340.              font-size: 2rem;
341.              color: $clr-success;
342.              cursor: pointer;
```

```
343.
344.             &.red {
345.                 color: $clr-warning;
346.             }
347.
348.             &:hover {
349.                 font-size: 2.5rem;
350.                 padding: 0.25rem;
351.             }
352.         }
353.     }
354.
355.     .remote-stream {
356.         max-height: 25rem;
357.     }
358.
359.     .body {
360.         padding: 1rem;
361.         display: flex;
362.         flex-direction: column;
363.         gap: 0.5rem;
364.     }
365.
366.     .title {
367.         display: flex;
368.         align-items: center;
369.         gap: 1rem;
370.         font-size: 100%;
371.     }
372.
373.     .name {
374.         max-width: 70%;
375.         overflow: hidden;
376.         text-overflow: ellipsis;
377.     }
378.
379.     .logo {
380.         width: 48px;
381.         height: 48px;
382.     }
383.
384.     .datetime {
385.         align-self: start;
386.         flex-grow: 1;
387.         text-align: right;
388.         padding-left: 1rem;
389.         transform: translateY(-0.5rem);
390.     }
391.
392.     .show-details-btn:hover {
393.         text-decoration: underline;
394.         cursor: pointer;
395.     }
396.
397.     .details {
398.         display: none;
399.
400.         &.show {
401.             display: flex;
402.         }
403.     }
404. }
405.
406. .clickable:hover {
407.     background-color: $clr-primary-dark;
408.     box-shadow: 0px 2px 10px 3px $clr-primary-dark;
409.     cursor: pointer;
410. }
411.
412. .window-capture {
```

```
413.      display: flex;
414.      width: 100%;
415.      flex-direction: column;
416.      align-items: center;
417.      gap: 1rem;
418.
419.      & button {
420.          font-size: 1vw;
421.      }
422.
423.      &-container {
424.          position: relative;
425.          & > * {
426.              position: absolute;
427.              top: 0;
428.              left: 50%;
429.              transform: translateX(-50%);
430.          }
431.      }
432.
433.      & .video-overlay {
434.          position: relative;
435.          height: 100%;
436.          width: 100%;
437.          & > .rect {
438.              position: absolute;
439.
440.              &::before {
441.                  font-size: 1vw;
442.                  content: attr(data-width) "\00d7"attr(data-height) "\A"attr(data-
    color);
443.                  white-space: pre;
444.                  background-color: $clr-primary-dark;
445.                  opacity: 0.7;
446.              }
447.          }
448.      }
449.  }
450.
451.  .login,
452.  .register {
453.      display: flex;
454.      flex-direction: column;
455.      width: 100%;
456.      align-items: center;
457.      gap: 1rem;
458.      margin-top: 1rem;
459.      font-size: 1.25rem;
460.
461.      form {
462.          display: flex;
463.          flex-direction: column;
464.          gap: 1rem;
465.          align-items: center;
466.      }
467.
468.      input:not([type="checkbox"]),
469.      select {
470.          font-size: 1.25rem;
471.          width: 35ch;
472.          max-width: 90vw;
473.      }
474.      input[type="checkbox"] {
475.          transform: scale(1.5);
476.      }
477.
478.      .input-container {
479.          position: relative;
480.          &:hover::before {
481.              position: absolute;
```

```
482.                 content: attr(data-tooltip);
483.                 transform: translateY(calc(-100% - 0.5rem));
484.                 background-color: $clr-primary-light;
485.                 padding: 0.5rem 1rem;
486.                 border-radius: 100vw;
487.                 pointer-events: none;
488.             }
489.             &:hover::after {
490.                 position: absolute;
491.                 content: "";
492.                 left: 0;
493.                 transform: translate(25%, -50%);
494.                 border-style: solid;
495.                 border-width: 1em 1em 0 1em;
496.                 border-color: $clr-primary-light transparent transparent transparent;
497.                 pointer-events: none;
498.             }
499.         }
500.
501.         button {
502.             font-size: 1.25rem;
503.         }
504.         .pass-show-container {
505.             margin-left: 1rem;
506.             align-self: flex-start;
507.         }
508.         .error-msg {
509.             color: $clr-warning;
510.             padding: 0.5rem;
511.             background-color: $clr-primary-light;
512.             border-radius: 10px;
513.         }
514.     }
515.     .recorded-data {
516.         position: relative;
517.         margin: 0;
518.         .cam-img {
519.             position: absolute;
520.             height: 100%;
521.             width: 100%;
522.             object-fit: contain;
523.         }
524.
525.         .plea-img {
526.             position: absolute;
527.             max-height: 20vmax;
528.             max-width: 20vmax;
529.             z-index: 10;
530.         }
531.         .details {
532.             position: absolute;
533.             right: 0;
534.             display: grid;
535.             grid-template-columns: auto auto;
536.             column-gap: 0.5rem;
537.             align-items: center;
538.             background-color: $clr-primary;
539.             padding: 0.5rem;
540.             border-bottom-left-radius: 10px;
541.             opacity: 0.5;
542.             span {
543.                 text-decoration: underline;
544.             }
545.             #emotion {
546.                 color: $clr-success;
547.                 font-size: 1.5rem;
548.                 font-weight: bold;
549.             }
550.         }
551.     }
```

```
552.  .seek-control-container {
553.      position: sticky;
554.      width: 100%;
555.      bottom: 0;
556.      padding-top: 1rem;
557.      display: flex;
558.      flex-direction: column;
559.      align-items: center;
560.      background-color: $clr-primary-dark;
561.      z-index: 15;
562.
563.      .seek-input {
564.          width: 95vw;
565.          padding: 0;
566.          border: none;
567.      }
568.      .download {
569.          position: absolute;
570.          bottom: 0;
571.          right: 0;
572.          padding: 0.5rem 1.5rem;
573.      }
574.  }
575.
576.  .client-select-container {
577.      display: flex;
578.      align-items: center;
579.      flex-wrap: wrap;
580.      gap: 0.5rem;
581.  }
582.
583.  .socket-clients {
584.      .query-select-container {
585.          background-color: $clr-primary;
586.          display: flex;
587.          height: 100%;
588.          width: 100%;
589.          align-items: center;
590.          justify-content: flex-end;
591.          gap: 1rem;
592.          padding: 0.5rem 1rem;
593.          margin-bottom: 1rem;
594.          box-shadow: 0 0 5px 1px $clr-primary;
595.
596.          input {
597.              margin: 0;
598.              flex-grow: 1;
599.              border: 1px solid $clr-text;
600.          }
601.
602.          .reload-btn {
603.              cursor: pointer;
604.
605.              &:hover {
606.                  transform: scale(1.2);
607.              }
608.          }
609.      }
610.      .table {
611.          width: 100%;
612.          height: 100%;
613.          box-shadow: 0 0 15px 0.5px $clr-primary;
614.          background-color: $clr-table-bg;
615.      }
616.
617.      .table-header th {
618.          background-color: $clr-primary;
619.          padding: 0.5rem;
620.          overflow: hidden;
621.          text-overflow: ellipsis;
```

```
622.      }
623.
624.      .table-body tr:nth-of-type(even) {
625.          background-color: $clr-primary-light;
626.      }
627.
628.      .myself {
629.          background-color: $clr-success !important;
630.      }
631.
632.      .table-body td {
633.          text-align: center;
634.          overflow: hidden;
635.          text-overflow: ellipsis;
636.      }
637.
638.      .kick-btn,
639.      .refresh-btn {
640.          cursor: pointer;
641.      }
642.  }
643.
644.  .plea {
645.      .remote-stream {
646.          position: absolute;
647.          top: 50%;
648.          left: 50%;
649.          transform: translate(-50%, -50%);
650.          height: 100%;
651.          width: 100%;
652.          object-fit: contain;
653.      }
654.      .local-stream {
655.          position: absolute;
656.          top: 0;
657.          left: 0;
658.          margin: 1rem;
659.          width: 7rem;
660.          height: 7rem;
661.          border-radius: 100%;
662.          object-fit: cover;
663.      }
664.  }
665.
666.  .loading-screen {
667.      position: absolute;
668.      top: 0;
669.      left: 0;
670.      width: 100vw;
671.      height: 100vh;
672.      display: flex;
673.      align-items: center;
674.      justify-content: center;
675.      background-color: rgba(0, 0, 0, 0.6);
676.      z-index: 100;
677.      font-size: 7rem;
678.  }
679.
680.  footer {
681.      width: 100%;
682.      height: 3rem;
683.      bottom: 0;
684.      display: flex;
685.      justify-content: center;
686.      flex-direction: column;
687.      background-color: $clr-primary-dark;
688.      align-items: center;}
```

plea_python_client.py

```python
import base64
from PIL import Image
import io
import numpy as np
from visual_modality import VisualModality
import asyncio
import socketio
import time

vm = VisualModality(0.5, remote=True)
sio = socketio.AsyncClient()

expressions = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']


@sio.event
async def connect():
    print('SIO server connected')
    await sio.emit('log_socket_connection', {
        'email': 'Python NN Script',
        'connectedTimestamp': round(time.time() * 1000),
        'route': '/python-script'
    })

@sio.on('run_nn')
async def run_trough_nn(data):
    try:
        b64pic = data['camImg'].encode('ascii')
        pic = base64.b64decode(b64pic)
        img = Image.open(io.BytesIO(pic))
        vm_results, flow = vm.perform_detection(bounding_box=True, exp_text=True, frame=np.array(img))
        emotion_to_show = 'No Face'
        bounding_box = {'x': 0, 'y': 0, 'width': 0, 'height': 0}
        vm_res_rounded = [0.0, 0.0, 0.0, 0.2, 0.8, 0.0, 0.0]  # default -> slightly happy
        if len(vm_results['predictions']) == len(expressions):
            vm_res_rounded = list(map(lambda x: float(round(x, 3)), vm_results['predictions']))
            bounding_box = {'x': vm_results['coords'][0], 'y': vm_results['coords'][1],
                            'width': vm_results['coords'][2] - vm_results['coords'][0],
                            'height': vm_results['coords'][3] - vm_results['coords'][1]}
            emotion_to_show = expressions[vm_res_rounded.index(max(vm_res_rounded))]
        await sio.emit(
            'nn_result',
            {
                'socketId': data['socketId'],
                'colorKey': data['colorKey'],
                'predictions': vm_res_rounded,
                'emotion': emotion_to_show,
                'boundingBox': bounding_box
            }
        )

    except Exception as e:
        print(f'Neural network exception: {e}')


async def main():
    await sio.connect('http://192.168.0.1:8005')
    await sio.wait()


if __name__ == "__main__":
    asyncio.run(main())
```