

PROJECT

POLYGLOT ROOM

Acknowledgement

We would like to express our sincere appreciation and gratitude to the high institute of computer Science & information systems and all its individuals who have contributed in providing us the facilities to the completion of this project.

We would like to express our heartfelt thanks to our professors and mentors for their support, and our heartfelt thanks to Dr. Ashraf Fahmy for giving us an opportunity to undertake this project and provided us with the right guidance, and our heartfelt thanks to Eng. Walaa Zaghloul for her constant encouragement and support during the development of this project.

We would also like to mention the efforts and hard work of each member of our group, each member's unique skills and dedication have played a crucial role in the success of this project, this project could not have been completed without the commitment and of each member.

We are grateful to our friends and family for their understanding, help, and encouragement throughout this year. Their constant support has been a source of motivation during challenging times.

Project Abstract

Language barriers have been a big obstacle to communication, as individuals not sharing a common language has led to misunderstandings, miscommunication, and missed opportunities, this problem impacts Businesses, professionals, Students, and anyone seeking to connect across cultures

To bridge this gap, we represent a video conference platform with live audio translation capabilities, the platform supports multiple languages, ensuring that users can communicate across the world without any language difficulties, our solution offers an instant, accurate, and seamless communication experience, eliminating the need for pre-recorded subtitles or manual translation.

By breaking down language barriers, our platform enables effective communication, international collaboration and knowledge exchange.

Table of Content

Acknowledgement.....	3
Project Abstract.....	4
Chapter 1: Introduction.....	8
1.1 Introduction.....	9
1.2 Purpose.....	9
1.3 Scope.....	9
1.4 Product perspective.....	10
1.5 Objectives.....	10
1.6 Motivates.....	11
1.7 Related works.....	11
Chapter 2: Theoretical Background and Tools.....	12
2.1 Background.....	13
2.2 Definition, Acronyms, Abbreviation.....	14
2.3 Software Requirements.....	18
2.4 Hardware Requirements.....	18
2.5 Functional Requirements.....	19
2.6 Non-Functional Requirements.....	22
2.7 User characteristic.....	23

2.8 Constrains.....	23
2.9 Assumptions and dependencies.....	24
2.10 Problem statement.....	26

Chapter 3: Analysis and Design.....27

3.1 Use-case diagram.....	28
3.2 Activity diagram.....	29
3.3 Sequence diagram.....	30
3.4 Block diagram.....	31
3.5 Class diagram.....	32
3.6 Website Pages.....	33

- 3.6.1 Register page.....33
- 3.6.2 Login page.....33
- 3.6.3 Home page.....34
- 3.6.4 Chatbot page.....35
- 3.6.5 Room page.....35
- 3.6.6 Features page.....36

Chapter 4: Project Implementation.....37

- 4.1 Register new user.....38
- 4.2 Login users.....39
- 4.3 Node.js server.....40
- 4.4 Express router.....41
- 4.5 Video chat.....42
- 4.6 Text chat.....43

• 4.7 WebRTC server.....	44
• 4.8 Screen sharing.....	45
• 4.9 Speech recognition.....	46
• 4.10 Translation.....	47
• 4.11 Text to speech.....	48
• 4.12 Chatbot request handling.....	49
• 4.13 Flask server.....	50
• 4.14 Chatbot text pre-processing.....	51
• 4.15 Chatbot model.....	52
Conclusion.....	53
References.....	54
Appendices.....	55
Summary in Arabic.....	56

Chapter one

Introduction

1.1 Introduction

language barriers have historically presented obstacles in various domains, such as business, education, and personal relationships. These barriers often hinder effective communication, limit access to information, and create misunderstandings among individuals who speak different languages. However, our innovative video conference with live audio translation website aims to overcome these challenges by offering a seamless, user-friendly platform that caters to a diverse global audience.

1.2 Purpose

The primary purpose of our video conference with live audio translation website is to provide a seamless, real-time communication experience for users who speak different languages. By offering instantaneous language translation services, we aim to bridge the gap between people from diverse linguistic backgrounds, enabling them to connect, share ideas, and work together more effectively.

1.3 Scope

the development of the software required for real-time translation, webRTC communication, chatbot integration. The platform will support the selection of languages, video chat and text chat.

1.4 Product perspective

Our video conference website offers real-time language translation breaking barriers for seamless global communication, offers a multi-language chatbot powered by AI, offers a user-friendly design.

1.5 Objectives

- Design and implement a system that enables real-time translated communication in multiple languages.
- Design a multi-language chatbot.
- Designing a user-friendly, web-based platform that integrates video conferencing with real-time audio translation, ensuring a smooth and efficient communication experience for all participants.
- Ensuring multi language support by providing a wide range of languages and dialects, making it accessible to a diverse global audience.

1.6 Motivates

The language barriers often effect communication in business negotiations, education, and personal interactions, by using live translation, this project seeks to bridge these gaps and enhance the quality of global communication.

1.7 Related works

The field of video conferencing has many platforms offering real-time communication solutions, for example Zoom, Google Meet, Microsoft Teams however, despite these advancements, there is still a need for more efficient and effective live audio translation systems that can accurately translate spoken language in real-time, our proposed system aims to address these limitations by developing a more robust and flexible live audio translation platform that can be easily integrated with various video conferencing platforms.

Chapter Two

Theoretical Background and Tools

2.1 Background

In the creation of the project we used many technologies, for the front end we used HTML, CSS, and JavaScript to design a visually appealing interface that ensures smooth navigation for users, building on this foundation, we integrated text-to-speech, speech recognition and a translation API into the website to enable real-time translation capabilities, providing users with the ability to communicate effectively in multiple languages.

On the backend side, our choice of technologies included Node.js and Express.js for the backend server and routing, along with the templating engine EJS to render web pages, along with nodemon for automatic running and refresh of the server, in order to provide audio and video communication between users, we implemented WebRTC technology using Peer.js and Socket.io, coupled with UUID for unique session identification, ensuring secure and reliable real-time communication channels.

To store and manage user data effectively and handling authentication logic we used firebase as our database solution, enabling seamless data synchronization and real-time updates across multiple devices.

Additionally, we created and integrated a chatbot feature into the website using python and PyTorch for the creation and training of the ai model, NLTK for the data pre-processing, flask server to integrate the chatbot to our website and handling user requests, also adding the live audio translation functionality to the chatbot as using tts, speech recognition, translation technologies.

2.2 Definition, Acronyms, Abbreviation

- **HTML** (Hyper Text Markup Language): HTML is the standard markup language used to structure and create web pages. It allows developers to define the content and layout of a webpage, including text, images, videos, and other multimedia elements.
- **CSS** (Cascading Style Sheets): CSS is a style sheet language used to describe the presentation and visual formatting of a webpage. It enables developers to control various aspects of a webpage's layout, color scheme, fonts, and other design elements, ensuring a cohesive and aesthetically pleasing user experience.
- **JS** (JavaScript): JavaScript is a powerful, versatile scripting language that adds interactivity and dynamic behavior to webpages. It allows developers to create responsive interfaces, implement animations, validate user input, and much more, enhancing the overall functionality and usability of our website.
- **Node.js**: Node.js is an open-source, JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to execute JavaScript code outside of a web browser, enabling the creation of scalable, high performance server-side applications.
- **Express.js**: Express.js is a minimalist, flexible web application framework for Node.js. It helps developers build robust and efficient server-side applications, facilitating the development of APIs and handling HTTP requests.

- **Peer.js:** Peer.js is a JavaScript library for real-time peer-to-peer communication between web browsers. It enables the creation of decentralized applications and real-time features, such as chat rooms and collaborative editing, within our website.
- **WebRTC** (Web Real-Time Communication): WebRTC is an open-source technology that enables real-time communication, including voice and video calls, directly from web browsers without the need for plugins. It allows users to communicate seamlessly through our website.
- **EJS** (Embedded JavaScript): is a templating engine that allows you to generate dynamic web pages using JavaScript. It's a server-side templating engine that allows you to separate your presentation layer from your application logic. EJS provides a simple and efficient way to render dynamic data in your web applications.
- **UUID** (Universally Unique Identifier): is a 128-bit value that is used to identify information in computer systems. It's often used as a primary key in databases or as an identifier in various software systems.
- **Socket.IO:** is a JavaScript library that enables real-time communication between a client and a server. It's often used for building real-time web applications that require bi-directional communication between the client and server.
- **Nodemon:** is a popular open-source utility for Node.js that automatically restarts a Node.js application when it detects changes to the code. It's often used for development and testing purposes to simplify the process of iterating on and debugging code.

- **TTS (Text-to-Speech):** is technology works by processing written text and converting it into sound waves that can be heard as spoken words. TTS systems use various algorithms and synthetic voices to generate human-like speech. This technology has advanced significantly in recent years, with more natural-sounding voices and improved accuracy, making it a valuable tool for a wide range of applications. TTS can improve accessibility, provide a more inclusive user experience, and help people consume information in different ways.
- **Speech Recognition:** Speech recognition is a technology that enables computers to recognize and understand human speech, converting it into text. This technology can be implemented on our website to allow users to interact with the platform using voice commands, enhancing accessibility and user experience.
- **Python:** Python is a high-level, general-purpose programming language with a strong focus on readability and simplicity. It is widely used for web development, data analysis, artificial intelligence, and various other applications.
- **PyTorch:** PyTorch is an open-source machine learning library based on the Torch library. It is primarily used for developing deep learning models and applications. Our website may utilize PyTorch for implementing advanced AI features, such as natural language processing or computer vision.
- **Flask:** Flask is a lightweight, easy-to-use web application framework written in Python. It allows developers to create web services and APIs with minimal fuss, making it suitable for building microservices and small-to-medium-sized web applications.

- **NLTK** (Natural Language Toolkit): is a popular open-source library for natural language processing (NLP) in Python. It provides a wide range of tools and resources for text processing, tokenization, token classification, named entity recognition, semantic reasoning, and much more.
- **Chatbot**: a chatbot is an AI-powered conversational interface that simulates human-like conversations with users. Chatbots can be integrated into our website to provide instant customer support, answer frequently asked questions, or even assist with specific tasks, enhancing user engagement and satisfaction.
- **myMemoryAPI**: is a popular language translation service that provides an API for developers to integrate translation capabilities into their applications. Users can access the API to translate text or documents between different languages. This API allows developers to easily incorporate language translation features into their software, websites, or mobile apps.
- **Firebase**: Firebase Database is a real-time, cloud-based NoSQL database provided by Google as part of the Firebase platform. It allows developers to store and sync data between users in real-time using JSON format.
- **neural network**: is a computer system inspired by the structure and functioning of the human brain. It is a type of machine learning algorithm that attempts to recognize patterns or relationships in data through a process that mimics the way the human brain operates.

2.3 Software Requirements

- **Front End:**

- HTML
- CSS
- JS

- **Back End:**

- Node.js
- Express.js
- UUID
- Socket.io
- WebRTC
- Peer.js
- EJS
- Nodmon
- Firebase Database

- **AI Development:**

- Python
- PyTorch
- Flask
- NLTK

2.4 Hardware Requirements

- Compatible web browser
- Microphone
- Good internet Connection
- Camera

2.5 Functional Requirements

- R.1: Register

- R.1.1: Sign up

- Description: new users will have to sign up first and give data like name, email and password
 - Input: details about user
 - Output: succeeded or failed
 - Processing: checking input details, in case of error a message will appear

- R.1.2: Login

- Description: user shall enter needed data
 - Input: enter user name and password
 - Processing: checking invalid data and send an error message in case of it

- R.2: Chat bot operations

- R.2.1: select language

- Description: select the language to chat with bot
 - Input: language form list

- R.2.2: chat with chat bot

- Description: ask the culture bot a question
 - Output: a respond

- R.3: Room operations
 - R.2.1: create room
 - Description: user can create new room or join one
 - Processing: generate a new room code
 - Output: room code
 - R.2.2: join room
 - Description: user can join existing room
 - Input: room code
 - Processing: connect to room
 - R.2.3: mute
 - Description: close the microphone
 - R.2.4: close camera
 - Description: close the camera
 - R.2.5: share screen
 - Description: share user screen to other users
 - Input: selected screen
 - R.2.6: text chat
 - Description: the user can chat separately in text chat area
 - Input: message
 - Output: message send to all users

- R.2.7: select language
 - Description: user can select the translation language
 - Input: selected language
- R.2.8: stop translation
 - Description: user has the ability to stop translation and just see the raw messages
- R.2.9: stop text-to-speech
 - Description: user has the ability to stop text-to-speech and don't hear messages voice
- R.2.10: TTS
 - Description: messages turned into voice TTS
 - Output: TTS voice
- R.2.11: close call
 - Description: user end call
 - Output: return to home

2.6 Non-Functional Requirements

- **Performance Requirement:** Maintain stable performance, even during high-traffic periods, to ensure smooth communication.
- **Scalability Requirement:** Ensure the system can accommodate an increasing number of users and languages without compromising quality
- **Accessibility Requirement:** Make the platform accessible to users with different abilities, including those with hearing or visual impairments.
- **Compatibility Requirement:** Support various devices, operating systems, and browsers to cater to a diverse user base.
- **Maintainability Requirement:** Develop the system with modular architecture and clear documentation for easy updates and bug fixes.
- **Reliability Requirement:** The service has to be 100% reliable as incorrect data ruin the purpose of it, the website shall be working 24/7.

2.7 User characteristic

- **Video conference user:**
 - Create new room
 - Join room
 - Select language
 - Text chat
 - Video chat
 - TTS
 - Speech recognition

- **Chat bot user:**
 - Select language
 - Ask question
 - Get respond

2.8 Constrains

The user's data should be secured, Ensure the translation quality meets industry standards for effective communication, Guarantee the platform's consistent performance and availability to users, interface shall be easy to learn and use, minimizing the learning curve for users, Facilitate seamless integration with other.

2.9 Assumptions and dependencies

The project used third party application for development:

- **Visual studio code:** commonly referred to as VS Code, is a free source-code editor developed by Microsoft. It has gained immense popularity among developers due to its lightweight, yet powerful features. Some key features of Visual Studio Code include debugging capabilities, IntelliSense for code completion, Git integration, built-in terminal, extensions marketplace for customization, and support for various programming languages. It is highly customizable, user-friendly, and supports a wide range of extensions to enhance productivity.
- **Figma:** is a web-based, collaborative interface design tool used for creating user interfaces, prototypes, and interactive designs. It is popular among UI/UX designers for its ease of use, real-time collaboration features, and cloud-based platform that allows team members to work together on projects simultaneously. Figma offers features like vector graphics editing, prototyping tools, design systems, and version history tracking. Designers can create interactive prototypes, share designs with stakeholders, and get feedback in real-time. Figma aims to streamline the design process and enhance collaboration among design teams.

- **Photoshop:** is a powerful image editing software widely used by graphic designers, photographers, and artists for editing and manipulating digital images. It offers a wide range of features including layers, filters, adjustments, and various tools for retouching and creating complex compositions.
- **Chrome:** is a popular web browser developed by Google. Known for its speed, user-friendly interface, and extensive library of extensions, Chrome is commonly used for browsing the web and accessing web applications.
- **GitHub Desktop:** is a graphical user interface (GUI) application that simplifies the process of working with Git repositories on the popular code hosting platform, GitHub. It provides an intuitive interface for cloning repositories, managing branches, and pushing changes to GitHub.
- **Pycharm:** is a Python Integrated Development Environment (IDE) developed by JetBrains. It offers features like code completion, code analysis, debugging tools, and integration with version control systems. PyCharm is popular among Python developers for its productivity tools and support for web development frameworks.

- **Word:** Microsoft Word is a word processing software used for creating, editing, and formatting text documents. It is part of the Microsoft Office suite and offers features like spell checking, templates, styles, and collaboration tools.
- **Visio:** is a diagramming and vector graphics application available within the Microsoft Office suite. It is commonly used for creating flowcharts, organizational charts, network diagrams, and other visual representations of data or processes. Visio is designed to simplify complex information and make it easier to understand through visuals

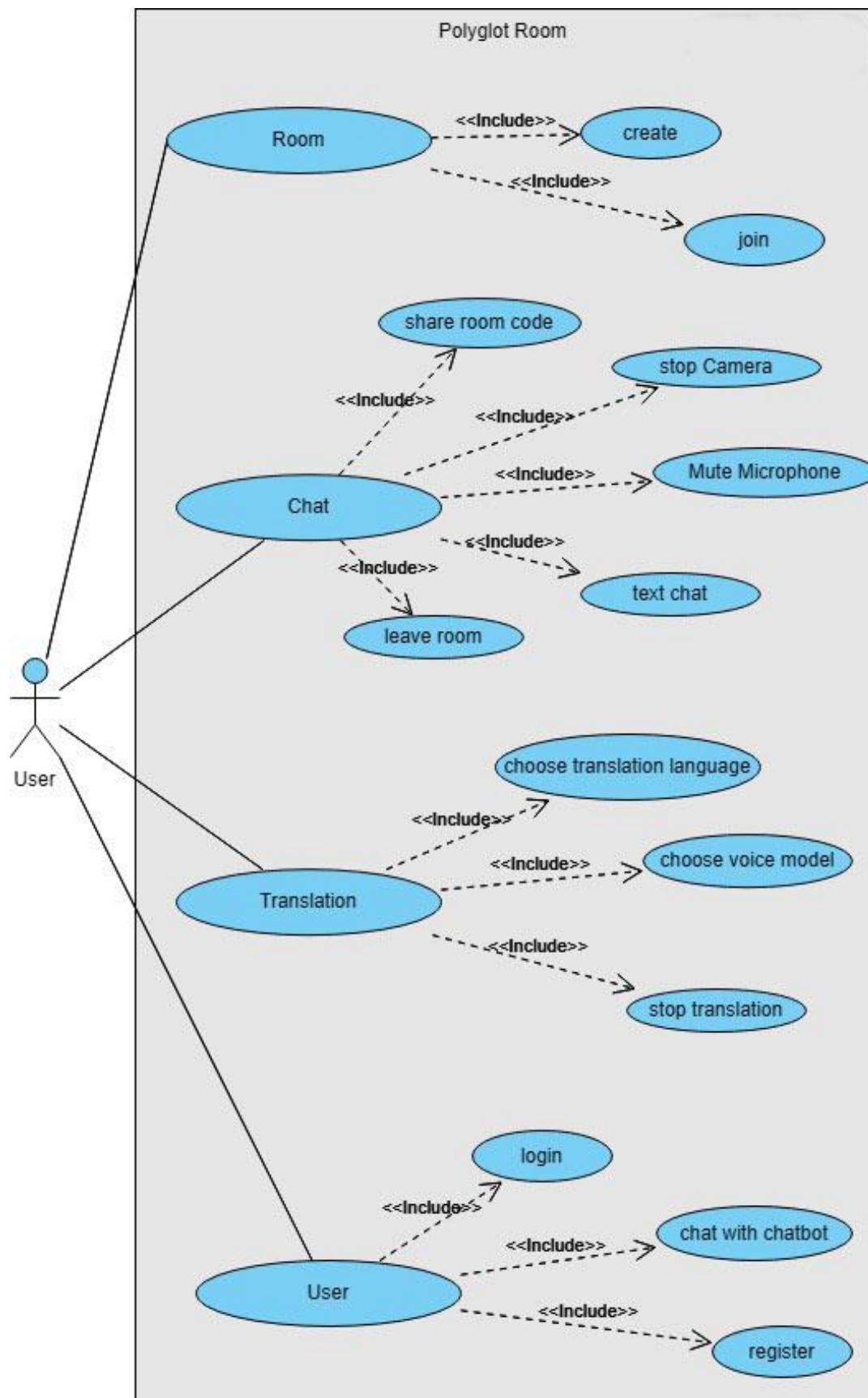
2.10 Problem statement

video translation platforms are needed to solve many problems all over the world, it is available online, these platforms must be available because not all people can talk more than their language, These platforms will solve this problem, so all people can to each other easily, when people want to learn or talk to someone foreign they can get the link of the platform and communicate with each other.

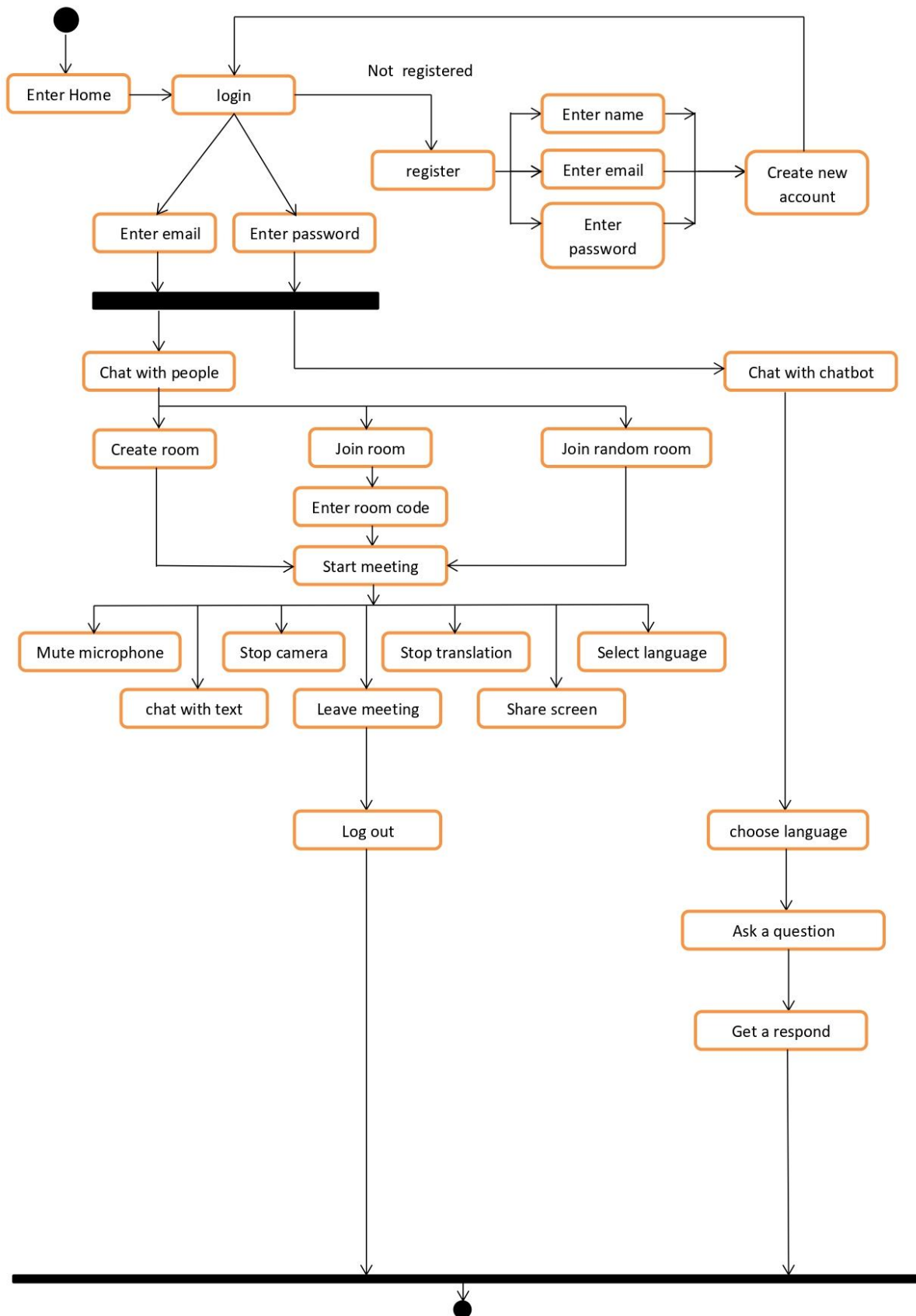
Chapter three

Analysis and Design

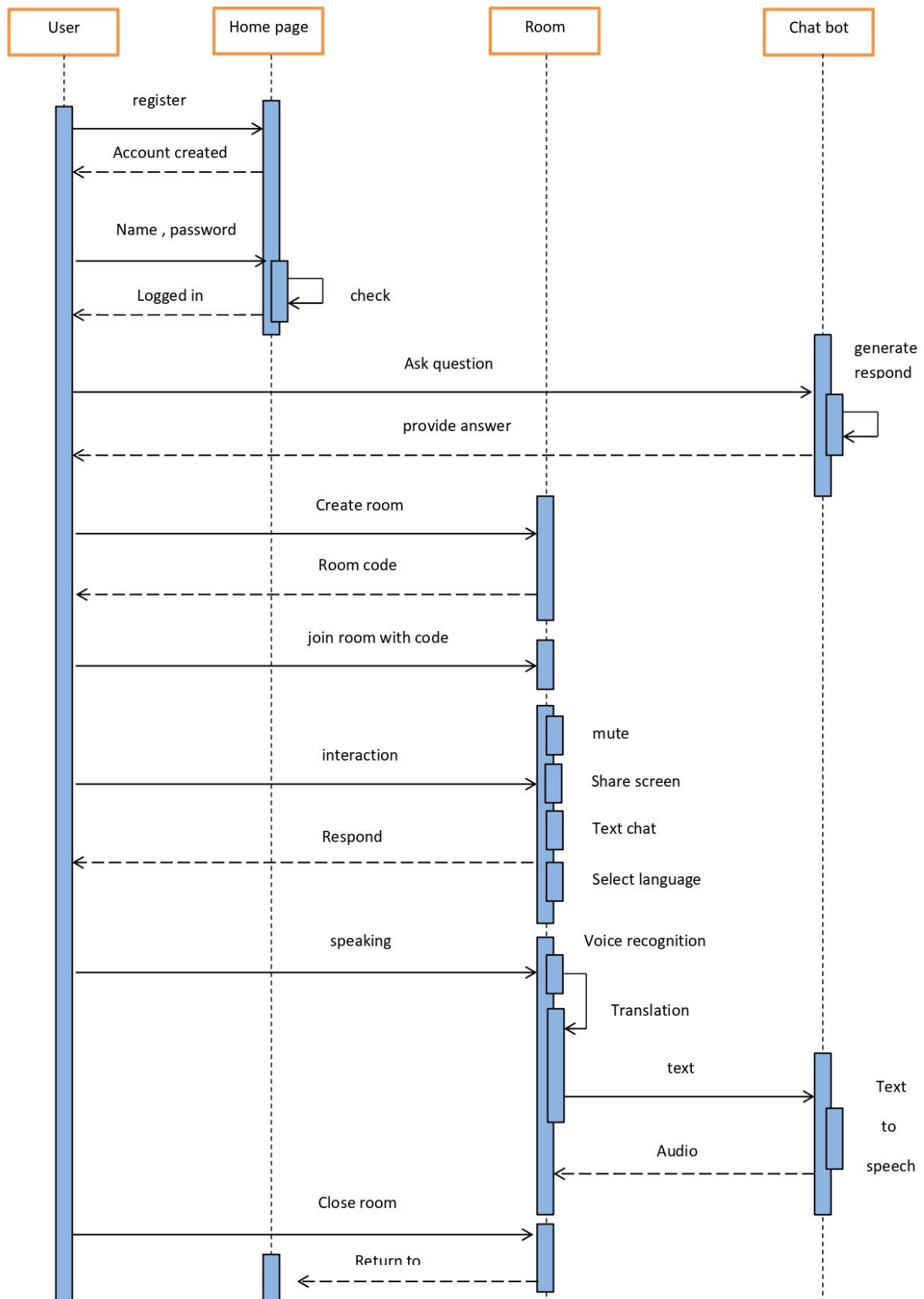
3.1 Use-case diagram



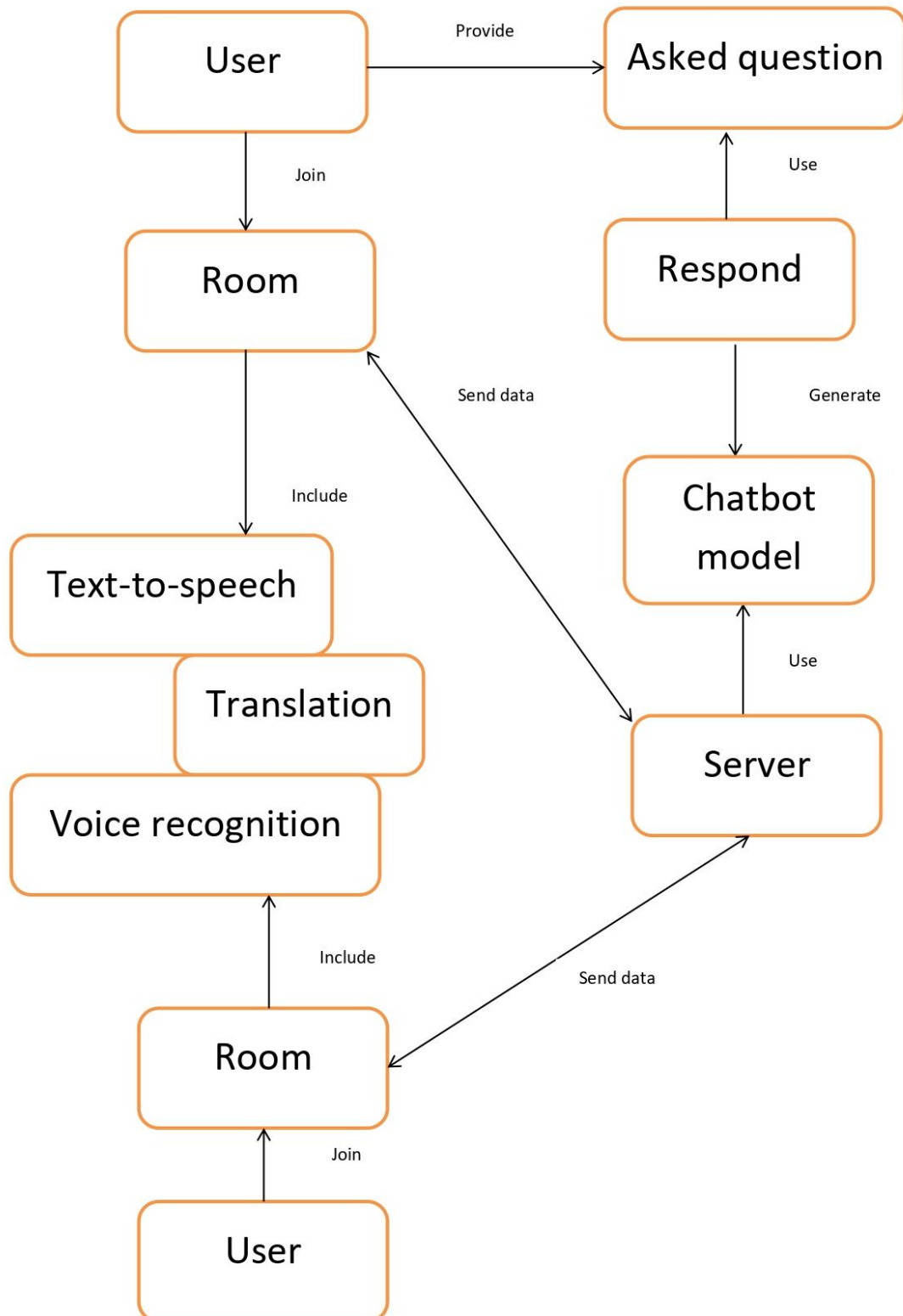
3.2 Activity diagram



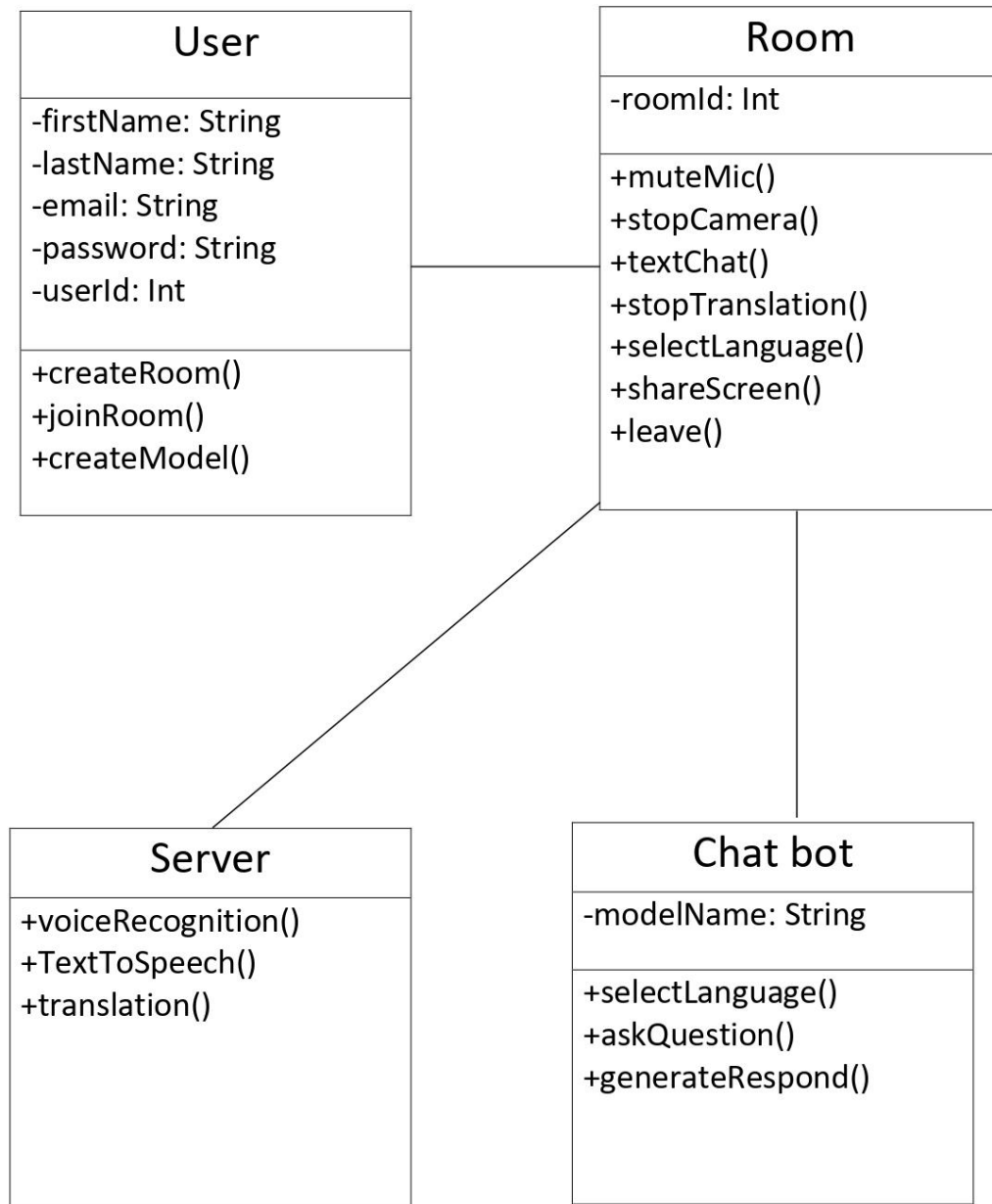
3.3 Sequence diagram



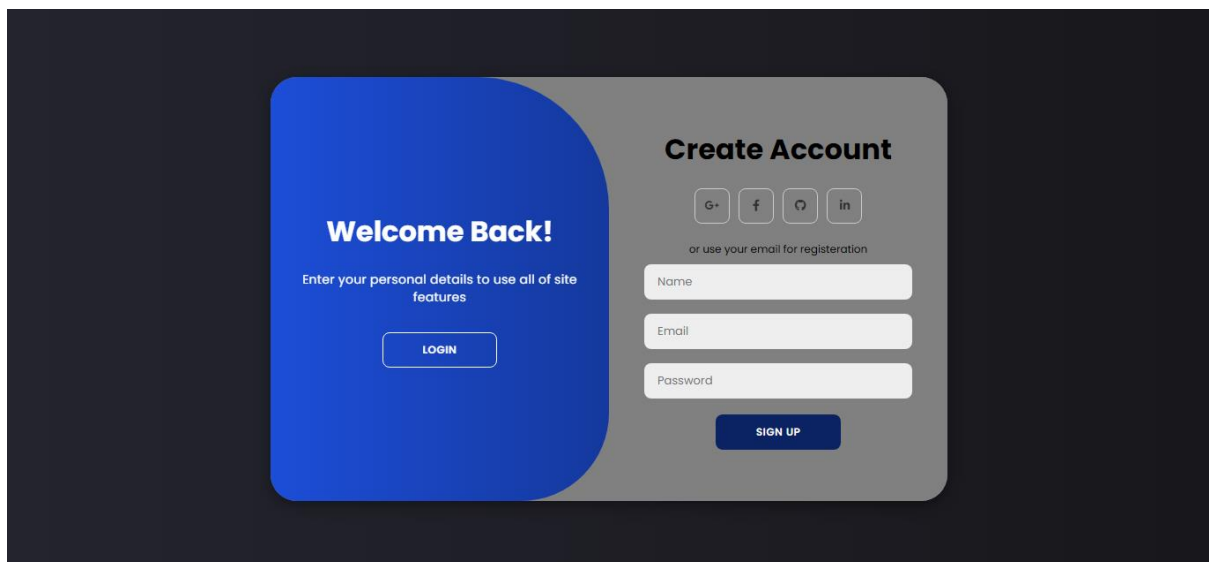
3.4 Block diagram



3.5 Class diagram



3.6.1 Register page



The register page features a dark background with a central light gray rounded rectangle. On the left, a blue rounded rectangle contains the text "Welcome Back!" and "Enter your personal details to use all of site features" with a "LOGIN" button. On the right, the "Create Account" section includes social media login options (G+, f, Q, in), a link to "or use your email for registration", and input fields for "Name", "Email", and "Password", followed by a "SIGN UP" button.

Welcome Back!

Enter your personal details to use all of site features

LOGIN

Create Account

G+ f Q in

or use your email for registration

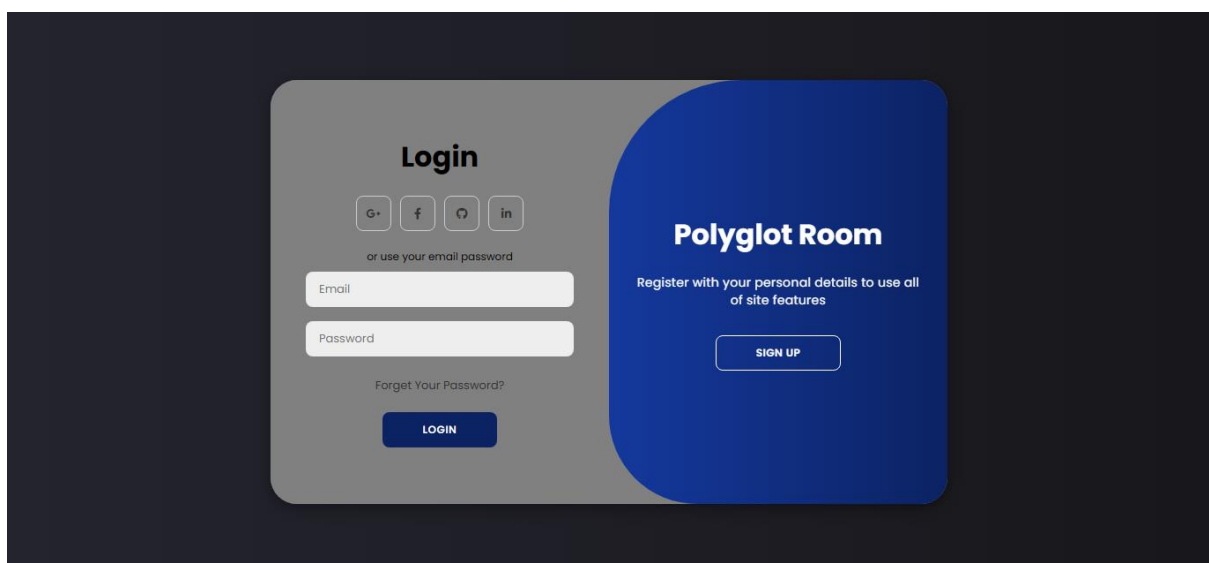
Name

Email

Password

SIGN UP

3.6.2 Login page



The login page features a dark background with a central light gray rounded rectangle. On the left, the "Login" section includes social media login options (G+, f, Q, in), a link to "or use your email password", and input fields for "Email" and "Password", followed by a "LOGIN" button and a "Forgot Your Password?" link. On the right, a blue rounded rectangle contains the text "Polyglot Room" and "Register with your personal details to use all of site features" with a "SIGN UP" button.

Login

G+ f Q in

or use your email password

Email

Password

Forgot Your Password?

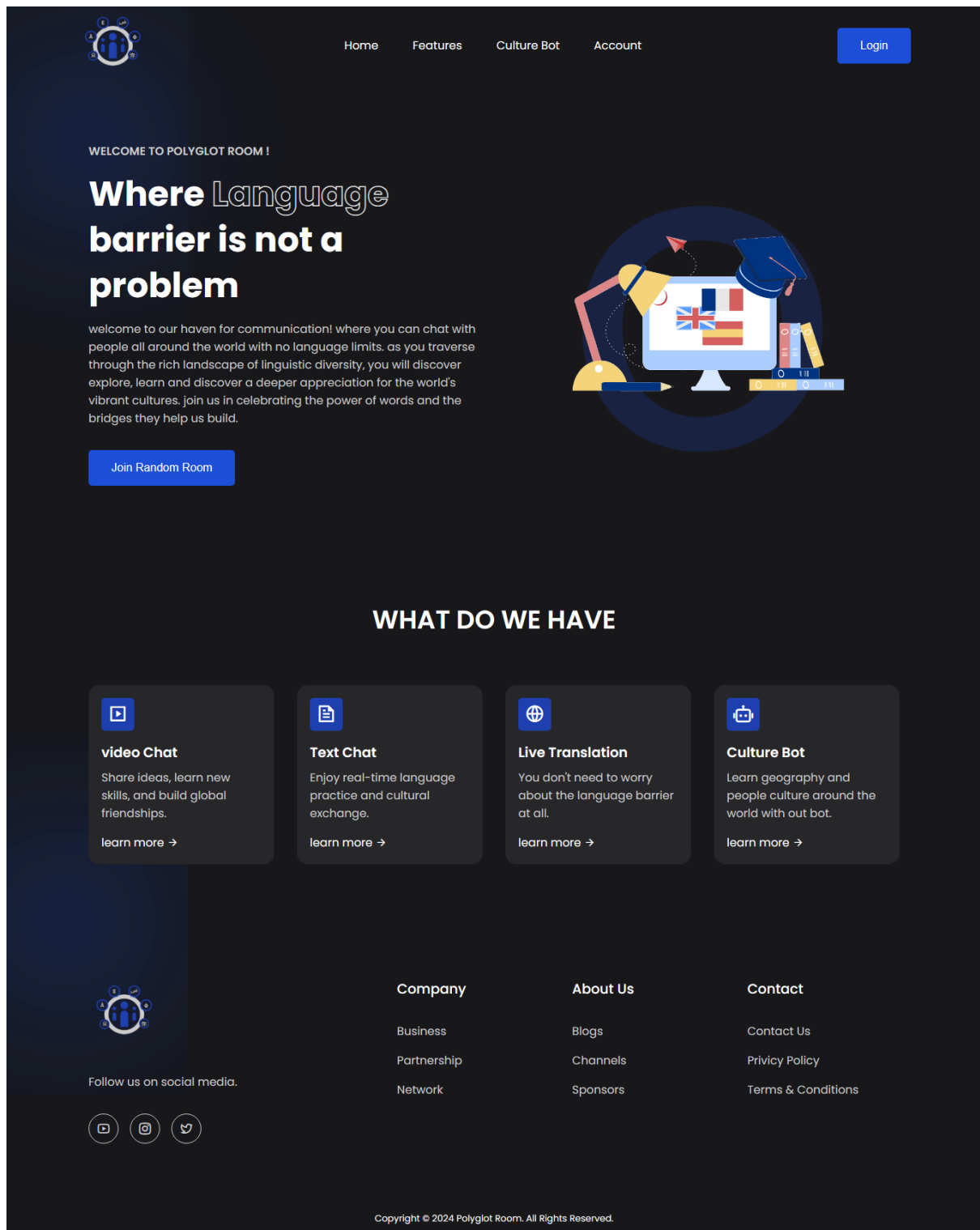
LOGIN

Polyglot Room

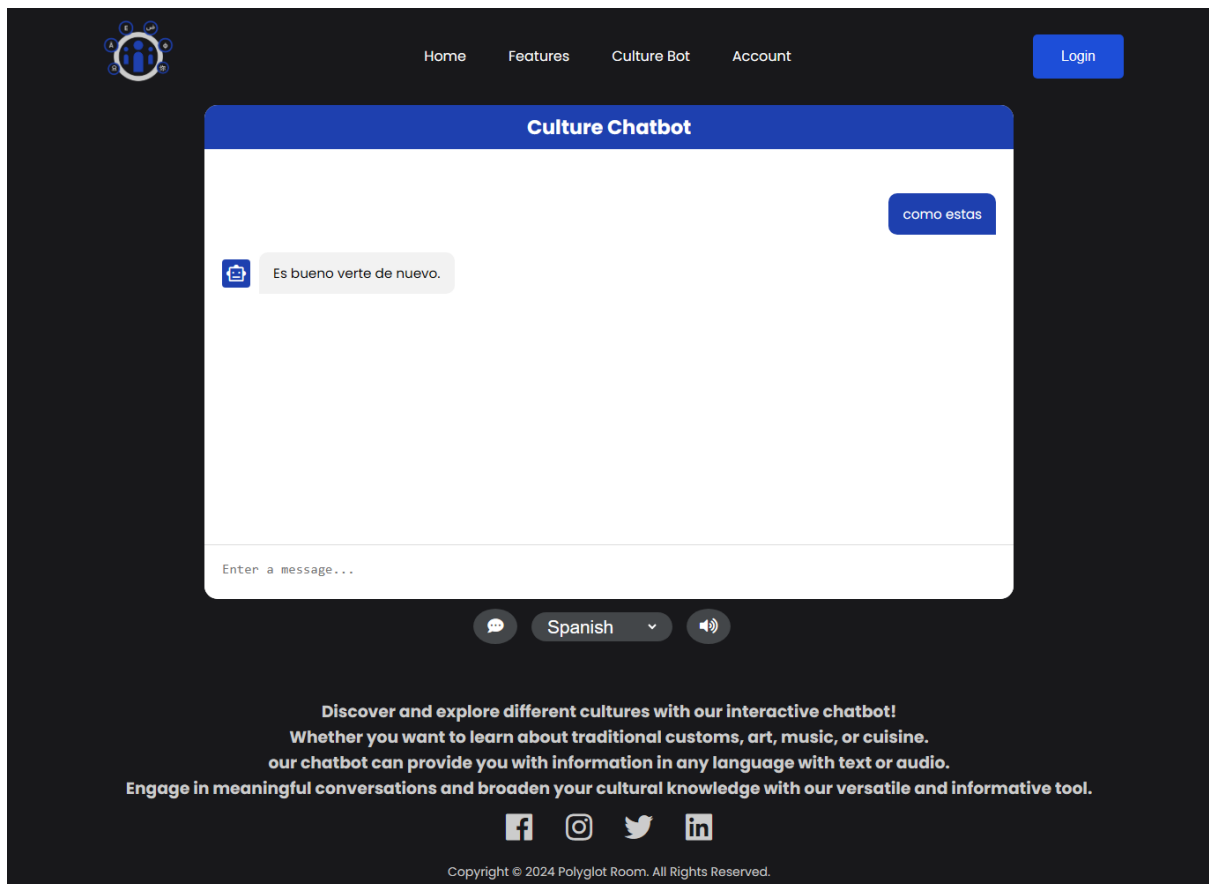
Register with your personal details to use all of site features

SIGN UP

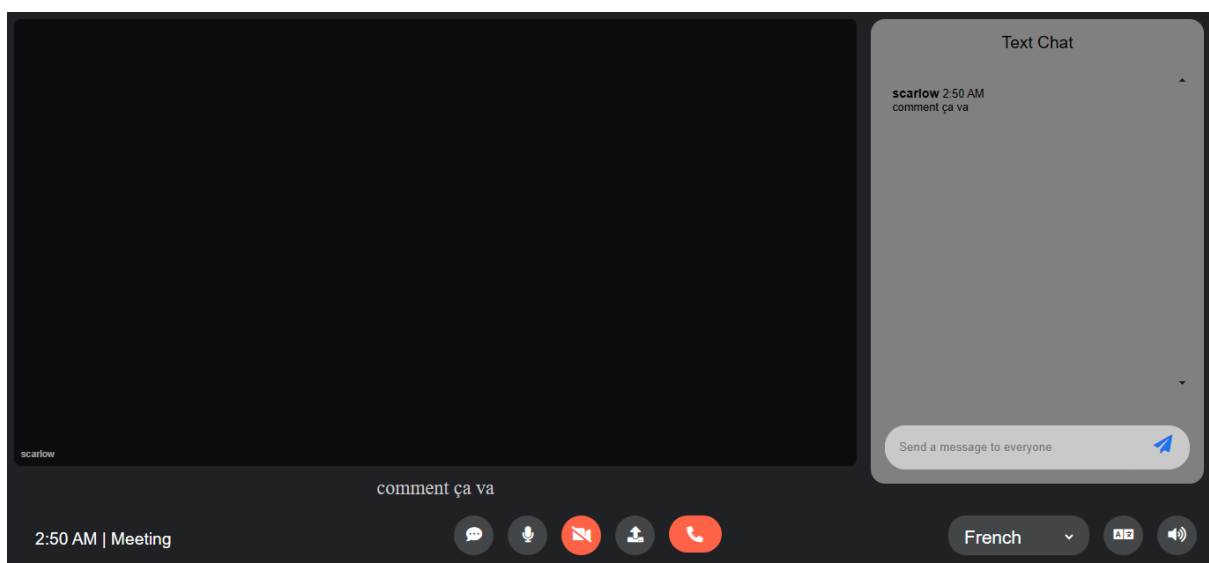
3.6.3 Home page



3.6.4 Chatbot page



3.6.5 Room page



3.6.6 Features page

HomeFeaturesCulture BotAccountLogin

Here you can read about our website used technology



Video Conference

WebRTC video conferencing simplifies online communication by allowing people to talk and share visuals directly through their web browsers. This technology ensures smooth, secure, and high-quality interactions. With WebRTC, users can engage in group discussions, share their screens, and collaborate effectively, breaking down geographical barriers and fostering productivity in various settings.



Text Chat

Text chat platforms facilitate easy and convenient online communication through written messages exchanged via web browsers. These platforms provide a user-friendly environment for people to engage in discussions, share ideas, and collaborate. Text chat systems often include features like group messaging, file sharing, and real-time updates, making it simple for users to connect and work together, regardless of their physical locations.



Culture Chat Bot

Culture chatbots are AI-driven conversational agents designed to engage users in discussions related to various cultural aspects, such as traditions, history, art, music, and cuisine. These bots are developed to provide educational and entertaining experiences, fostering a deeper understanding of diverse cultures and promoting global awareness. By using natural language processing and machine learning algorithms, culture chatbots can adapt to user preferences and tailor their responses to deliver personalized interactions.



Live Translation

Translation is a digital service that offers language conversion , enabling users to communicate across linguistic barriers. It empowers individuals and businesses to transcend cultural boundaries by providing accurate, contextual translations. This user-friendly online tool caters to diverse needs, including legal, medical, and creative content, fostering global understanding and collaboration.



Text To Speech

Text-to-speech (TTS) technology converts written text into spoken words, bridging the gap between reading and listening. It benefits individuals with visual impairments, learning disabilities, or those preferring auditory content. TTS systems use advanced algorithms and natural-sounding voices to deliver accurate pronunciation and intonation. This innovative tool enhances accessibility and promotes inclusivity in various applications, such as e-learning, audiobooks, and voice assistants.



Voice Recognition

Speech recognition, also known as automatic speech recognition (ASR) or voice recognition, is a technology that enables computers to transcribe spoken words into written text. It facilitates hands-free interaction, improving efficiency and accessibility. This advanced system employs machine learning algorithms to analyze and interpret human speech, continuously refining its accuracy. Speech recognition plays a vital role in virtual assistants, dictation software, and language translation, fostering seamless communication and enhancing user experience.



Copyright © 2024 Polyglot Room. All Rights Reserved.

Chapter four

Project Implementation

4.1 Register new user

Users can register and create a new account on the platform using Firebase Authentication. The user can input their email address, password, and other necessary details, which are then securely stored in Firebase's user authentication system. Upon successful registration, the user's information is saved in the Firebase database, allowing them to access personalized features and content on your website.

```
// Setup
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const registerSubmit = document.getElementById('register-submit');

// Storing data
registerSubmit.addEventListener("click",function(event){
    event.preventDefault()
    const registerEmail = document.getElementById('register-email').value;
    const registerPassword = document.getElementById('register-password').value;
    const registerName = document.getElementById('register-name').value;

    createUserWithEmailAndPassword(auth, registerEmail, registerPassword)
    .then((userCredential) => {
        // Signed up
        const user = userCredential.user;
        window.location.href = '/';
        alert("register")
    })
})
```

4.2 Login users

Users can log in to the website using their registered credentials stored in Firebase Authentication. Users can enter their email and password to authenticate their identity and gain access to their account. Upon successful login, users are granted access to restricted content, personalized settings, and other user-specific features on your website.

```
// Initialize Firebase
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const loginSubmit = document.getElementById('login-submit');
// Storing data
loginSubmit.addEventListener("click",function(event){
    event.preventDefault()
    const loginEmail = document.getElementById('login-email').value;
    const loginPassword = document.getElementById('login-password').value;

    signInWithEmailAndPassword(auth, loginEmail, loginPassword)
    .then((userCredential) => {
        // Login
        const user = userCredential.user;
        window.location.href = '/';
        alert("login")
    })
})
```

4.3 Node.js server

setting up a Node.js server to handle the backend logic and serve the website's content. Node.js allows for server-side JavaScript execution, enabling to build dynamic web applications. The Node.js server can handle incoming requests, communicate with databases, perform data processing, and generate responses to client-side requests, ensuring smooth functionality and data management for your website.

```
const express = require('express'); //requiring express
const app = express(); //app has all the properties of express
const server = require('http').Server(app); //creting http server
const io = require('socket.io')(server); //socket runs on this server
const { ExpressPeerServer } = require('peer'); //WebRTC api for real time
communication

app.use(express.static('./assets')); //setting up static path
app.set('view engine', 'ejs'); //setting up view engine
app.set('views', './views'); //setting up view path
app.use('/', require('./router'));

//running the server
server.listen(8000, function (err) {
  if (err) {
    console.log(`Error :: ${err} occured while starting the server in
server.js!`);
  }
  console.log(`Server is up and running on port 8000`);
});
```


4.4 Express router

utilizing Express.js to define routes and manage the flow of data within the website to redirect users to requested URL and render web pages, Express Router helps in organizing backend code by grouping related routes and functionalities together. using Express routers to create modular, maintainable code that handles different HTTP requests, such as GET, POST, PUT, and DELETE, directing them to the appropriate server-side logic and resources.

```
const express = require('express');
const router = express.Router();//instance to express app
const { v4: uuidv4 } = require('uuid'); //used to get unique room ids

//create new room
router.get('/room', (req, res) =>{
    return res.redirect(`/room/${uuidv4()}`);
});
//join room
router.get('/room/:room', (req, res) =>{
    return res.render('room', { roomId: req.params.room });
});

module.exports = router;
```

4.5 Video chat

Using Peer.js library that simplifies the implementation of WebRTC technology by providing a high-level API for establishing peer-to-peer connections. It allows to implement video sharing, starting calls, ending calls and data sharing into web applications without dealing with the complexities of WebRTC protocol implementation, video chat feature enables users to engage in real-time video communication on the website. Users can establish video calls with one another, allowing for face-to-face interactions remotely.

```
function connectToNewUser(userId, userName, stream) {  
  const call = myPeer.call(userId, stream);  
  const video = document.createElement('video');  
  
  call.on('stream', userVideoStream => {  
    addVideoStream(video, userVideoStream, userName);  
  })  
  
  call.on('close', () => {  
    video.remove();  
  })  
  
  peers[userId] = call  
  currentPeer = call;  
}
```

4.6 Text chat

Users can exchange messages in real-time through text-based conversations on the website using socket.io which enables real-time, bidirectional communication between clients and servers, it handles connection, disconnection, joining rooms, sending messages, receiving messages and enables the exchange of data in real-time, making it ideal for implementing features like chat applications

```
io.on('connection', socket => {  
  //request for joining room  
  socket.on('join-room', (roomId, userId, userName) => {  
    socket.join(roomId); //joining the mentioned room  
    socket.broadcast.to(roomId)  
      .emit('user-connected', userId, userName);  
  
    socket.on('send-message', (inputMsg, userName, org) => {  
      io.to(roomId)  
        .emit('recieve-message', inputMsg, userName, org);  
    })  
  
    socket.on('disconnect', () => {  
      socket.broadcast.to(roomId)  
        .emit('user-disconnected', userId, userName);  
    })  
  })  
})
```

4.7 WebRTC connection

Handling video calls using WebRTC which enables peer-to-peer communication between browsers, allowing users to establish audio, video, and data connections directly without the need for plugins or third-party software. It provides secure and high-quality real-time communication over the internet, making it ideal for applications like video calls, online meetings, and live streaming. WebRTC simplifies the process of setting up audio and video communication channels, enabling seamless interaction between users in web applications.

```
//handling video calling
navigator.mediaDevices.getUserMedia({
  video: true, //we want video
  audio: true //we want audio
}).then(stream => {
  //storing the video stream returned to the myVideoStream variable
  myVideoStream = stream;
  //appended my stream to 'video-grid' div
  addVideoStream(myVideo, stream, userName);

  myPeer.on('call', call => {
    call.answer(stream);
    console.log('Hello')
    const video = document.createElement('video');
    call.on('stream', userVideoStream => {
      console.log('video displayed');
      addVideoStream(video, userVideoStream, userName)
    });
  });
});
```

4.8 Screen sharing

Using WebRTC to allow users to allows users to broadcast their computer screens in real-time to others over the internet. It is a valuable tool for collaboration, remote assistance, presentations, and online education. With screen sharing functionality, users can share their entire screen or specific application windows with others, facilitating visual communication, demos, and problem-solving scenarios. Screen sharing enhances remote collaboration by enabling participants to view and interact with shared content in real-time.

```
function startScreenShare() {
  navigator.mediaDevices.getDisplayMedia({ video: true }).then((stream) => {
    screenStream = stream;
    let videoTrack = screenStream.getVideoTracks()[0];
    videoTrack.onended = () => {
      console.log('Screen sharing stopped!');
      stopScreenSharing()
    }
    if (myPeer) {
      let sender = currentPeer.peerConnection.getSenders().find(function
(s) {return s.track.kind == videoTrack.kind;
      })
      sender.replaceTrack(videoTrack)
      screenSharing = true
    })})
    let sender = currentPeer.peerConnection.getSenders().find(function (s) {
      return s.track.kind == videoTrack.kind;
    })
    sender.replaceTrack(videoTrack)
  })
}
```

4.9 Speech recognition

Using web kit Speech Recognition service enables computers to convert spoken language into text or commands. It allows users to interact with applications through voice input, enabling hands-free operation and enhancing accessibility for users with disabilities. Speech recognition can be used for dictation, voice commands, voice search, and transcribing spoken content. By accurately interpreting and processing spoken words, speech recognition technology enhances user experience and productivity in various applications.

```
window.SpeechRecognition = window.webkitSpeechRecognition;
const recognition = new window.SpeechRecognition();
recognition.addEventListener("result", (e) => {
    console.log('start');
    const text = Array.from(e.results)
        .map((result) => result[0])
        .map((result) => result.transcript)
        .join(" ");

    speechArea.innerHTML = text;
    if(e.results[0].isFinal){
        let finalSpeech = text;
    }
});

recognition.addEventListener("end", ()=>{
    console.log('end');
    turnSpeechOnOff();
});
```

4.10 Translation

Using myMemoryApi to translate text from one language to another by sending fetch request to the API with text and selected language and receiving translated message from it, allowing users to overcome language barriers and communicate effectively across different linguistic backgrounds. Automated translation tools leverage machine learning algorithms to analyze and translate content accurately and efficiently.

```
//translation function
const languages = document.getElementById('selection');
var translating = true;

//translation function
async function translate(text, from, to) {
  if (!text) return;
  let apiUrl =
`https://api.mymemory.translated.net/get?q=${text}&langpair=${from
}|${to}`;
  let response = await fetch(apiUrl);
  let data = await response.json();
  return data.responseData.translatedText;
}
```

4.11 Text to speech

Using Speech Synthesis Utterance library to implement Text-to-speech technology which converts written text into spoken language, providing audible output for digital content, as it accept text and provide audio for that text with the ability to choose of many voice models.

```
//text to speech
var speaking = true;
function textToSpeech(lan,text){
    let utterance = new SpeechSynthesisUtterance(text);
    utterance.lang = lan;
    speechSynthesis.speak(utterance);
}
```


4.12 Chatbot request handling

Using JavaScript CRUD operations and HTTP request to perform fetch requests on the chatbot API allowing users to interact with the chatbot by sending a message or a query, the request handling mechanism is responsible for processing that input, understanding the user's intent, sending request to the chatbot API and generating a response.

```
const generateResponse = (chatElement) => {  
  const messageElement = chatElement.querySelector("p");  
  let data = chatInput.value  
  translate(data, botLang, defaultLang).then((data) => {  
    const url = 'http://127.0.0.1:5000/predict'  
    fetch(url, {  
      method : 'POST',  
      mod      : 'cors',  
      body     : JSON.stringify({message : data}),  
      headers : {  
        'Content-Type' : 'application/json'  
      },  
    }).then(r => r.json())  
  })  
})
```

4.13 Flask server

Using Flask to create an API that works as the backend infrastructure that handles client requests, processes data, and generates responses dynamically, when user sends a request from the website the flask app hands that request and generates a response and then sends it the website.

```
# Import necessary modules
from flask import Flask, request, jsonify
from flask_cors import CORS
from chat import get_response

app = Flask(__name__)
CORS(app)

# Define a POST route at "/predict"
@app.post("/predict")
def predict():
    # Extract the message from the POST request
    text = request.get_json().get("message")
    # Get response using the extracted message
    response = get_response(text)
    print("request received")
    # Create a response message
    message = {"answer" : response}
    return message

# Run the Flask app in debug mode if the script is executed directly
if __name__ == "__main__":
    app.run(debug=True)
```

4.14 Chatbot text preprocessing

Using NLTK and NLP to preparing and optimizing text data before it is fed into the chatbot model for understanding and generating responses. This step is essential for cleaning and structuring the input text to improve the efficiency and accuracy of the chatbot's natural language processing (NLP) capabilities.

```
def tokenize(sentence):
    return nltk.word_tokenize(sentence)
def stem(word):
    return stemmer.stem(word.lower())
def bag_of_words(tokenized_sentence, words):
    # stem each word
    sentence_words = [stem(word) for word in tokenized_sentence]
    # initialize bag with 0 for each word
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1

    return bag
```

4.15 Chatbot model

Using PyTorch and Neural Networks to create the chatbot model which consist of 3 layers

```
import torch
import torch.nn as nn
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()

    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        # no activation and no softmax at the end
        return out
```

Conclusion

In conclusion, our video conference with live audio translation project represents a step forward in breaking down language barriers and providing communication across different cultures and languages, by providing real-time translation during meetings, we are enabling individuals and businesses to collaborate and connect in ways that were previously impossible, the future is bright for this platform, and we look forward to see the positive impact it will have on global communication, thanks to our team for their hard work and dedication in bringing this project to life, together, we believe that this project has the potential to improve the way we communicate, we have created a solution that has the power to transform the way we communicate and collaborate on a global scale, we're excited to share our achievement with the world and look forward to continuing to refine and improve our technology to meet the evolving needs of our users.

References

1. **PROJECT GITHUB** - <https://github.com/mohamedffffff/video-conference-with-live-audio-translation>
2. **WEBRTC** - <https://www.w3.org/TR/webrtc/>
3. **NODE.JS** - <https://nodejs.org/en/docs/>
4. **TTS** - <https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesis>
5. **SPEECH RECOGNITION** - https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API
6. **MYMEMORYAPI** - <https://mymemory.translated.net/doc/spec.php>
7. **FIREBASE** - <https://firebase.google.com/docs/database/web/start>
8. **FLASK** - <https://flask.palletsprojects.com/en/2.0.x/api/>
9. **NLTK** - <https://realpython.com/nltk-nlp-python/>
10. **PYTORCH** - <https://pytorch.org/docs/stable/index.html>
11. **API COMMUNICATION** - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

Appendices

- <https://www.geeksforgeeks.org/express-js/tutorial>
- <https://ably.com/topic/socketio/learn>
- <https://www.cockroachlabs.com/blog/what-is-a-uuid/>
- https://en.wikipedia.org/wiki/Natural_language_processing
- [https://en.wikipedia.org/wiki/Neural_network_\(machine_learning\)](https://en.wikipedia.org/wiki/Neural_network_(machine_learning))
- https://en.wikipedia.org/wiki/Natural_Language_Toolkit
- <https://www.tutorialspoint.com/webrtc/index.htm>
- <https://www.youtube.com/watch?v=SccSCuHhOw0>
- <https://www.youtube.com/watch?v=VShtPwEkDD0>
- <https://www.youtube.com/watch?v=y0srtrtK1Q0>
- https://www.youtube.com/watch?v=Z1RJmh_OqeA
- <https://hackernoon.com/creating-a-screen-sharing-application-with-javascript>
- https://www.w3schools.com/jsref/api_fetch.asp
- https://www.w3schools.com/js/js_promise.asp
- <https://www.youtube.com/watch?v=tHL5STNJKag>
- https://www.youtube.com/watch?v=WM178YopjFI&ab_channel=VectorM%3A
- https://www.w3schools.com/howto/howto_js_typewriter.asp
- <https://www.digitalocean.com/community/tutorials/workflow-nodemon>
- https://en.wikipedia.org/wiki/Easy_Java_Simulations
- https://www.youtube.com/watch?v=KMtLqPi2wiU&ab_channel=MuruganS
- <https://github.com/topics/voice-recognition>
- <https://openai.com/chatgpt/>

تصميم منصة الترجمة الفورية

يعالج هذا المشروع التحدي الدائم المتمثل في الحواجز اللغوية من خلال تطوير منصة توفر الترجمة الصوتية المباشرة, يعزز هذا الحل التواصل والتعاون السلس بين المشاركين الذين يتحدثون لغات مختلفة, فهذا المشروع موجه لكل من يريد ان يتواصل مع اي فرد من غير لغته فهو ببساطة يعمل كمترجم شخصي لكل من يستخدمه فهو يتعرف علي الكلام المنطوق من قبل الراسل ثم يترجم هذا الكلام الي لغة المرسل اليه ثم يحول هذا النص المكتوب الي صوت يستطيع المرسل اليه ان يسمعه علي الفور.

تستخدم المنصة تقنيات مؤتمرات الفيديو مما يسمح للمشاركين برؤية بعضهم البعض والتفاعل مع بعضهم البعض لتدفق أكثر طبيعية للمحادثة كما تستخدم تقنيات التعرف على الكلام والترجمة الآلية المتطورة لتحويل اللغات المنطوقة إلى ترجمات أو تعليقات صوتية مما يزيل الحواجز اللغوية ويضمن الفهم الواضح لجميع المشاركين .

للمنصة العديد من الاستخدامات حيث تمكن من التواصل مع العائلة والأصدقاء, وتمكن الشركات العالمية من التواصل السهل مع العملاء والشركاء الدوليين بغض النظر عن اللغة المنطوقة, وفي المؤسسات التعليمية تسهل التواصل بين الطلاب والمدرسين من خلفيات متنوعة وتثري تجربة التعلم وتعزز بيئة تعليمية أكثر شمولاً, ويمكن استخدام لحجز استشارات طبية مع الأطباء أو أخصائيي الرعاية الصحية, ويمكن للشركات استخدام لتقديم خدمات العملاء للعملاء, وتساعد على نشر الثقافة والمعلومات بربط المستخدمين في جميع أنحاء العالم, وتحت على تعلم اللغات ونشر الثقافات المختلفة بين المعارف والاصدقاء حول العالم.

تبرز أهمية المشروع في كونه أداة تواصل فعالة لربط الناس في أي مكان وزمان, حيث يمكن الانضمام إلى الاجتماعات من خلال أي متصفح ويب حديث ومشاركة رابط الاجتماع المشاركين الآخرين, كما يوفر جودة صوت وفيديو عالية الدقة, مما يضمن تجربة تواصل واضحة وسلسة, ويمكن مشاركة الشاشة لعرض العروض التقديمية أو المستندات أو أي محتوى آخر على شاشتك, ويوفر ميزة الدردشة النصية للتواصل مع المشاركين الآخرين خلال الاجتماع.