***METIOR***

**Problem Statement**

How to conduct online examination without live proctoring. Conducting online exam is challenging mainly because of two factors. Lack of invigilation and identification of proper platform for conducting the examination. Currently in market, there are good online platform available to use and conduct the examination online like mettl. But these platforms are high cost to use and also does not prevent complete malpractice.

Proctoring should be done automatically, that is, system should take care.

**Functionalities**

1. Authentication of the student
2. Identification of presence of third parties
3. Identification of usages of malpractices like internet searching during examination

**Innovative features**

1. Students will be asked to speak out the answers

The eye movement will be captured by the system while they answer and this will be analyzed to find out whether they are reading from a note or from recollecting the answers

Feasibility

To implement this feature we do not require any additional tools. The camera of laptop/ mobile which is required for authenticating the student can be used to record eye movements. Start and stop of camera can be controlled by the system. Audio message can be recorded at the server side so that there is no need of sending the answers separately. Couple of face images also can be saved at server side.

A simple headphone is enough to record their audio answers.

Challenges

Video recording. The eye movement can be effectively analyzed only when we have good pictures and for this we require reasonable camera and proper lighting in the room

Our model addresses the issues in the following categories

* Invigilation
* Components of assessment
* Preparation of question bank
* Normalization

1. Invigilation

Invariably one cannot do invigilation perfectly during the online examination platform using laptop camera or mobile camera whatever the students use for writing their examination. So to reduce the usages of sophisticated tools and devices we recommend parent/guardian invigilation. That is parents or guardian of the student will be monitoring their ward’s examination. Even though this is not ideal for couple of reason , but still very good solution to the current scenario. It is not ideal because very few parents may help their wards for getting good marks by compromising ethics. This support from parents or guardians also will not last and may be for a longer run this will be stopped automatically by themselves. Another issue that may arise is parent’s availability. This can address either my making this as an optional or by asking them to substitute with a suitable guardian.

In addition to this manual invigilation process we introduce an AI based automated proctoring.

1. Assessment components

Here we recommend the following components with respective weightage

Viva- 20 %

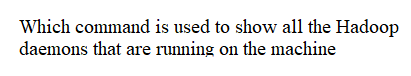
Multiple choice questions- 20 %

Descriptive- 60 %

The viva component will give a complete assessment of the student and will normalize the mark automatically

The multiple choice questions and descriptive questions should not be given as such, instead can be provided as an image.

For example



This will prevent a student to copy the content from the question paper and paste in a search engine to search for answers. If they want to do then they have to type the complete question which will be a time consuming one.

1. Question bank preparation

There should be enough questions in the question bank for which almost unique question should be given to every individual students. The questions are also of different levels like basic, moderate, complex etc.

1. Normalization

To eliminate the malpractice attempts we recommend to reduce the examination duration drastically. This methodology will give pressure to students and they may not get time to switch between answer window and to other malpractice options. If they try to copy then obviously they can very less time and will be not be able to complete not even 50 %.

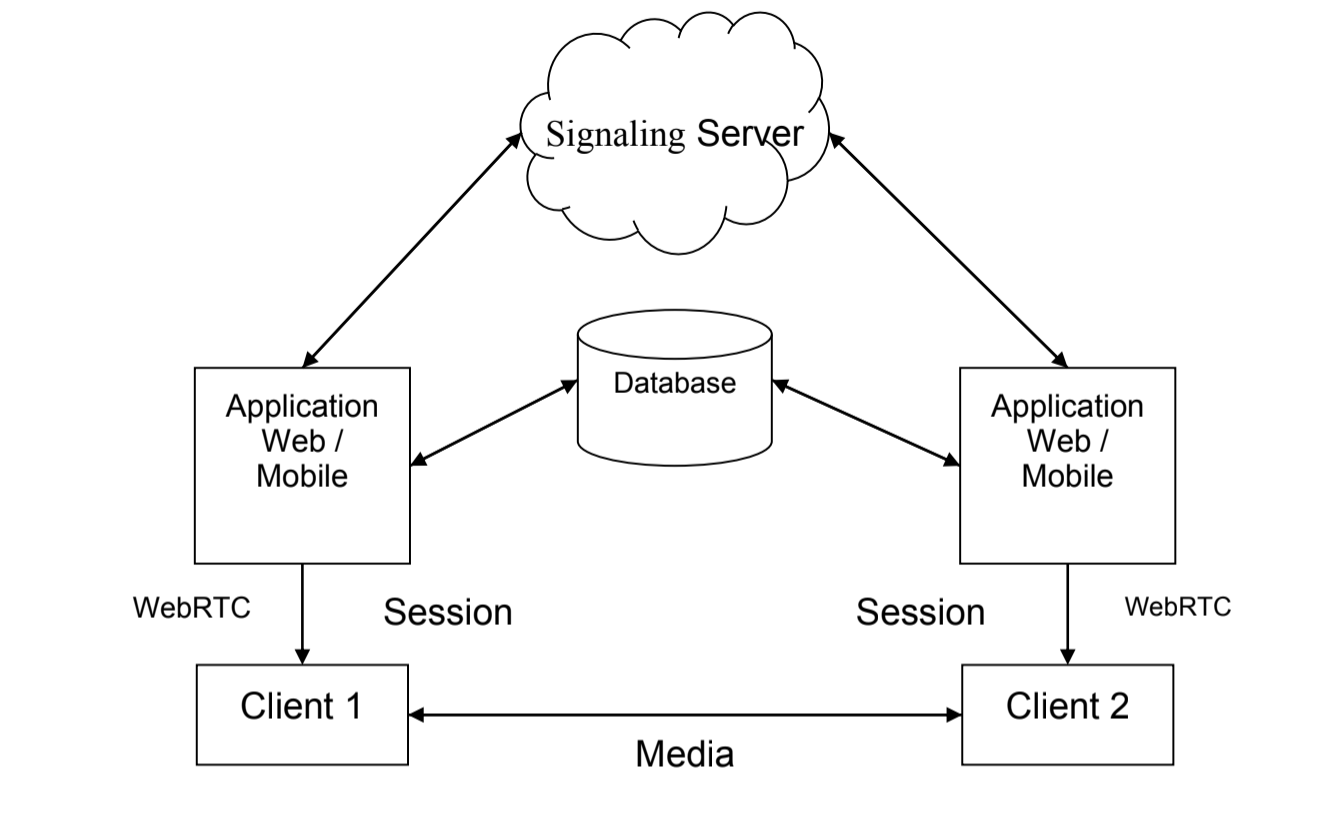
This will affect the good students’ performance also and will not be the actual representation of a student assessment. To address this we recommend normalize the marks received by all the students to higher grade if required.

Our model consist of parent invigilation, appropriate assessment components like viva and proper question bank and also method of normalization

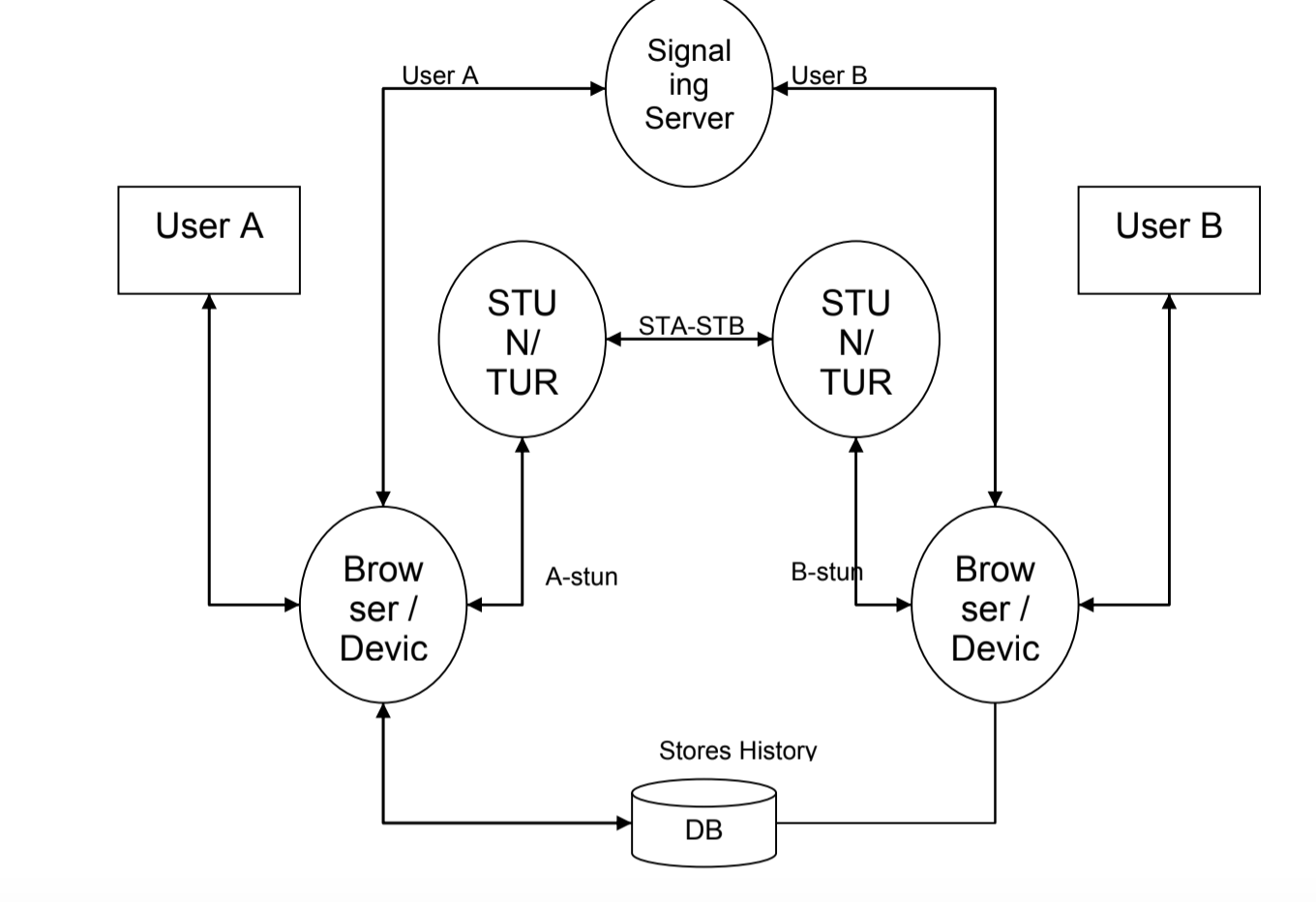
**Architecture/Design**

**2. Architect - Design Assessments**

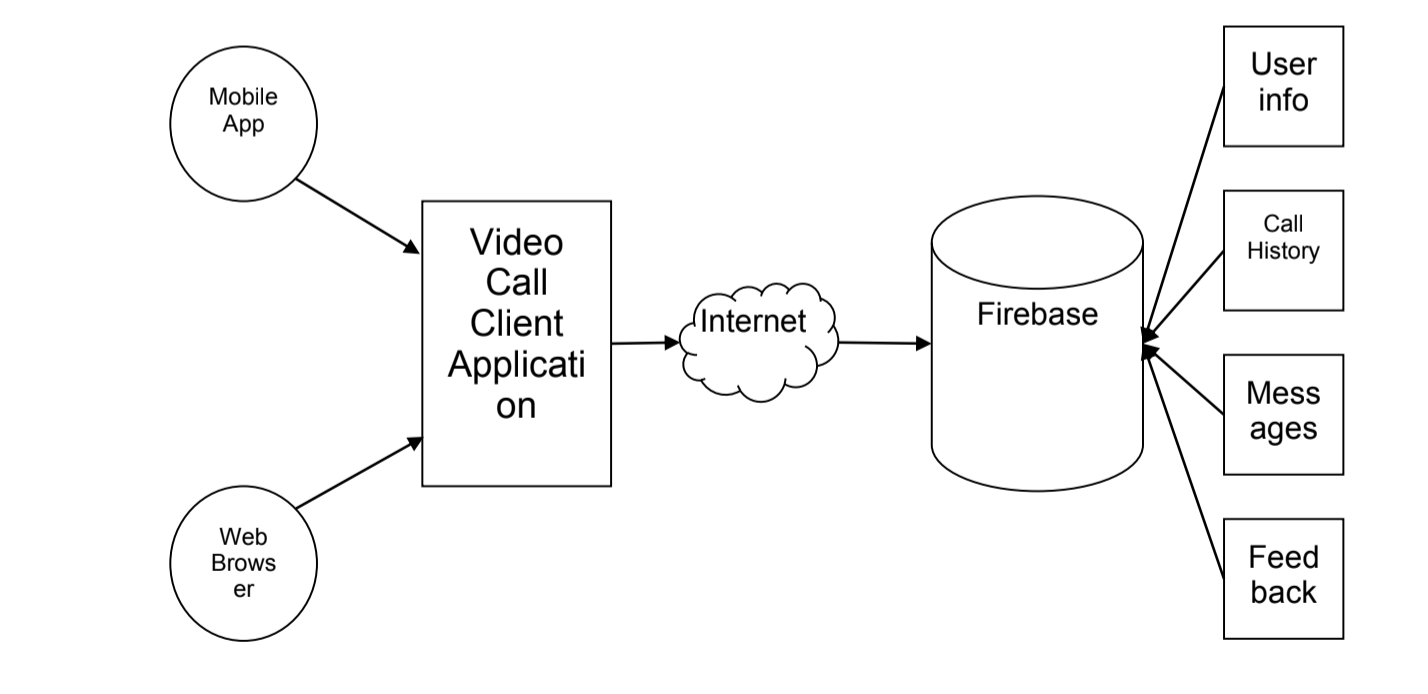
**Application Architecture**



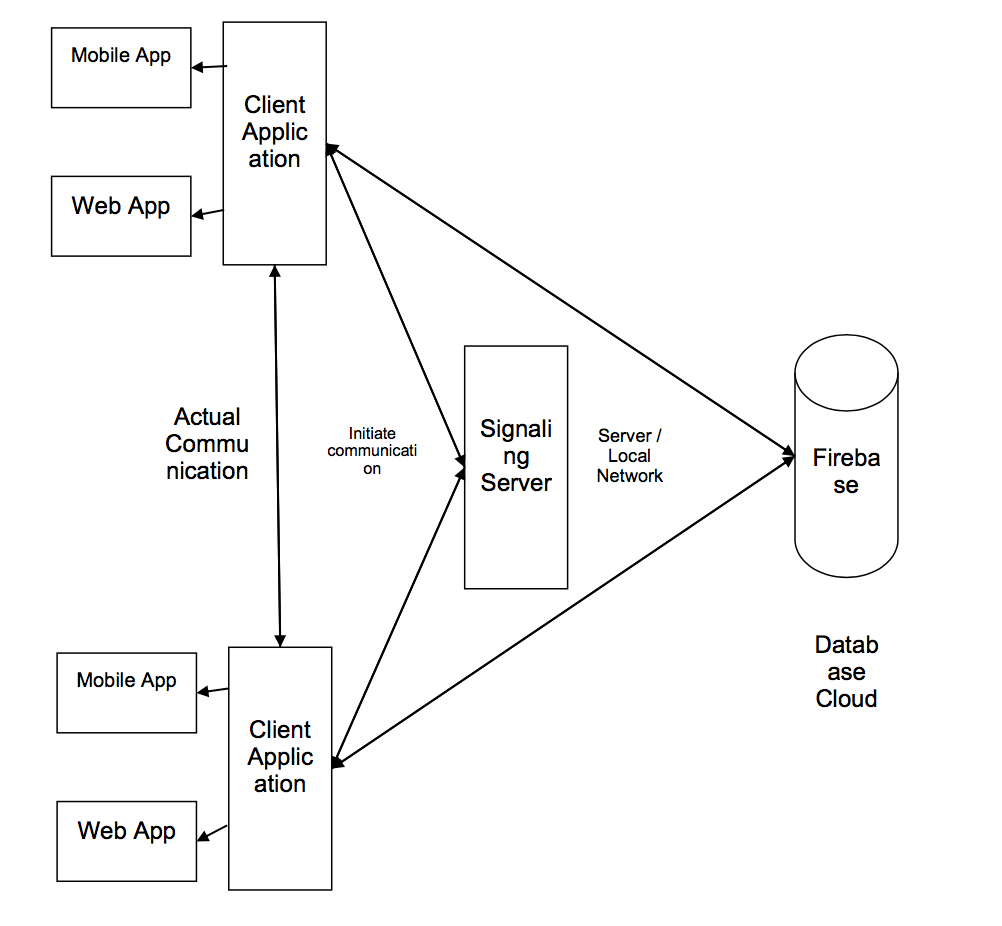
**Functional Architecture**

****

**Technical Architecture**

****

**Deployment Architecture**

****

**3. Coding, Testing Assessments**

**3.1 Package.json - Client**

{

"name": "videocall-react", "version": "0.1.0",

"private": true,

"proxy": "http://localhost:8000", "dependencies": {

"@testing-library/jest-dom": "^4.2.4", "@testing-library/react": "^9.5.0", "@testing-library/user-event": "^7.2.1", "emailjs-com": "^2.6.3",

"express": "^4.17.1",

"firebase": "^7.17.1", "nodemailer": "^6.4.11", "react": "^16.13.1", "react-dom": "^16.13.1", "react-firebaseui": "^4.1.0", "react-scripts": "^3.4.3", "styled-components": "^5.2.0", "use-screen-recording": "^2.0.0"

}, "scripts": {

"start": "react-scripts start", "build": "react-scripts build", "test": "react-scripts test", "eject": "react-scripts eject"

}, "eslintConfig": {

"extends": "react-app" },

"browserslist": { "production": [

">0.2%",

"not dead",

"not op\_mini all"

], "development": [

"last 1 chrome version", "last 1 firefox version", "last 1 safari version"

] },

"devDependencies": { "react-router-dom": "^5.2.0", "socket.io": "^2.3.0", "socket.io-client": "^2.3.0", "uuid": "^8.3.0"

} }

**3.2 package.json - Server**

{

"name": "server", "version": "1.0.0", "main": "server.js", "scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node server.js" },

"author": "", "license": "ISC", "keywords": [],

"description": "", "dependencies": {

"express": "^4.17.1",

"socket.io": "^2.3.0" }

}

**3.3 Server.js**

// importing dependencies

const express = require("express"); const http = require("http");

//creating server

const app = express();

const server = http.createServer(app);

//creating socket.io instace

const socket = require("socket.io"); const io = socket(server);

const rooms = {};

// creating room with the URL or joining the room io.on("connection", socket => {

//creating or joining room socket.on("join room", roomID => {

if (rooms[roomID]) { rooms[roomID].push(socket.id);

} else {

rooms[roomID] = [socket.id];

}

const otherUser = rooms[roomID].find(id => id !== socket.id);

if (otherUser) {

socket.emit("other user", otherUser); socket.to(otherUser).emit("user joined", socket.io);

} });

// offer a connection socket.on("offer",payload => {

io.to(payload.target).emit("offer",payload); });

socket.on("answer",payload =>{ io.to(payload.target).emit("answer",payload);

});

socket.on("ice-candidate",incoming => { io.to(incoming.target).emit("ice-candidate",incoming.candidate);

}); });

server.listen(8000, () => console.log('Server is running in port 8000'));

**3.4 Index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import \* as serviceWorker from './serviceWorker';

ReactDOM.render( <React.StrictMode>

<App />

</React.StrictMode>, document.getElementById('root')

);

// If you want your app to work offline and load faster, you can change // unregister() to register() below. Note this comes with some pitfalls. // Learn more about service workers: https://bit.ly/CRA-PWA serviceWorker.unregister();

**3.5 App.js**

import React from 'react';

import { BrowserRouter, Route, Switch } from "react-router-dom"; import CreateRoom from "./routes/CreateRoom";

import Room from "./routes/Room";

import './App.css';

import Home from "./routes/Home";

import TextRoom from "./routes/TextRoom";

import CreateTextRoom from "./routes/CreateTextRoom";

function App() { return (

<div className="App"> {/\*Different Routes in the App 1. Home / Root

2. Create Video Call Room

3. Create Text Chat Room

4. Join the Room - Video Call 5. Join the Text Chat Room\*/} <BrowserRouter>

<Switch>

<Route path="/" exact component={Home} />

<Route path="/create-room" exact component={CreateRoom} />

<Route path={"/create-text-room"} exact component={CreateTextRoom}/> <Route path="/room/:roomID" component={Room} />

<Route path="/textroom/:roomID" component={TextRoom}/>

</Switch> </BrowserRouter>

</div> );

}

export default App;

***3.6 CreateRoom.js***

import React, {Component} from "react"; import {v1 as uuid} from "uuid";

import {Link} from 'react-router-dom'; function CreateRoom() {

return (

<Link to={"/room/" + uuid()} style={{color: "white", textDecoration: false}}

className={"btn btn-success"}>Video Call Room</Link> );

}

export default CreateRoom;

**3.7 Room.js**

import React, {useRef, useEffect} from "react"; import io from "socket.io-client";

import logo from "./logo.png";

import {Link} from 'react-router-dom';

import Record from "./recording"; import emailjs from 'emailjs-com';

const Room = (props) => {

const userVideo = useRef(); const partnerVideo = useRef(); const peerRef = useRef(); const socketRef = useRef(); const otherUser = useRef(); const userStream = useRef(); const senders = useRef([]);

useEffect(() => {

navigator.mediaDevices.getUserMedia({audio: true, video: true}).then(stream => {

userVideo.current.srcObject = stream; userStream.current = stream;

socketRef.current = io.connect("/");

socketRef.current.emit("join room", props.match.params.roomID);

socketRef.current.on('other user', userID => { callUser(userID);

otherUser.current = userID;

});

socketRef.current.on("user joined", userID => { otherUser.current = userID;

});

socketRef.current.on("offer", handleRecieveCall);

socketRef.current.on("answer", handleAnswer);

socketRef.current.on("ice-candidate", handleNewICECandidateMsg); });

}, []);

function callUser(userID) {

peerRef.current = createPeer(userID);

userStream.current.getTracks().forEach(track => senders.current.push(peerRef.current.addTrack(track, userStream.current)));

}

function createPeer(userID) {

const peer = new RTCPeerConnection({ iceServers: [

{

urls: "stun:stun.stunprotocol.org"

}, {

}, ]

});

peer.onicecandidate = handleICECandidateEvent;

peer.ontrack = handleTrackEvent;

peer.onnegotiationneeded = () => handleNegotiationNeededEvent(userID);

return peer; }

function handleNegotiationNeededEvent(userID) { peerRef.current.createOffer().then(offer => {

return peerRef.current.setLocalDescription(offer); }).then(() => {

const payload = {

urls: 'turn:numb.viagenie.ca', credential: 'muazkh', username: 'webrtc@live.com'

target: userID,

caller: socketRef.current.id,

sdp: peerRef.current.localDescription

};

socketRef.current.emit("offer", payload); }).catch(e => console.log(e));

}

function handleRecieveCall(incoming) {

peerRef.current = createPeer();

const desc = new RTCSessionDescription(incoming.sdp); peerRef.current.setRemoteDescription(desc).then(() => {

userStream.current.getTracks().forEach(track => peerRef.current.addTrack(track, userStream.current));

}).then(() => {

return peerRef.current.createAnswer();

}).then(answer => {

return peerRef.current.setLocalDescription(answer);

}).then(() => { const payload = {

target: incoming.caller,

caller: socketRef.current.id,

sdp: peerRef.current.localDescription

}

socketRef.current.emit("answer", payload); })

}

function handleAnswer(message) {

const desc = new RTCSessionDescription(message.sdp);

peerRef.current.setRemoteDescription(desc).catch(e => console.log(e)); }

function handleICECandidateEvent(e) {

if (e.candidate) { const payload = {

target: otherUser.current,

candidate: e.candidate, }

socketRef.current.emit("ice-candidate", payload); }

}

function handleNewICECandidateMsg(incoming) {

const candidate = new RTCIceCandidate(incoming);

peerRef.current.addIceCandidate(candidate) .catch(e => console.log(e));

}

function handleTrackEvent(e) {

partnerVideo.current.srcObject = e.streams[0]; }

function shareScreen() {

navigator.mediaDevices.getDisplayMedia({cursor: true}).then(stream => {

const screenTrack = stream.getTracks()[0];

senders.current.find(sender => sender.track.kind === 'video').replaceTrack(screenTrack);

screenTrack.onended = function () { senders.current.find(sender => sender.track.kind ===

"video").replaceTrack(userStream.current.getTracks()[1]); }

}) }

function sendEmail(e) { e.preventDefault();

emailjs.sendForm('gmail', 'template\_ttvbgw9', e.target, 'user\_Y0k7xnXg4KCoslA8VrLfM')

.then((result) => {

// window.location.reload() //This is if you still want the page to reload (since

e.preventDefault() cancelled that behavior)

alert('Mail with the Meeting Link is Send to the Other User !');

}, (error) => {

// console.log(error.text); alert(error.toString());

}); }

return (

<div className={""} style={{backgroundColor: "black"}}>

<br/>

<img src={logo} style={{height: 150}} alt={""}/>

<br/>

<h5 style={{color: "white"}}>Meeting Url : {window.location.href}</h5> <br/>

<form onSubmit={sendEmail} method={"post"}>

<label style={{color: "white"}}>Enter Email(s) to invite to the meeting</label>&nbsp;&nbsp;

<input type={"text"} name={"to\_email"}/>&nbsp;&nbsp;

<input type={"hidden"} name={"link"} value={window.location.href}/> <input type={"submit"} className={"btn btn-success"} value={"Send"}/>

</form> <br/> <div>

<video controls style={{width: 500}} autoPlay ref={userVideo}/> &nbsp;&nbsp;

<video controls style={{width: 500}} autoPlay ref={partnerVideo}/>

</div>

<br/><br/>

{/\*<button className={"btn btn-success"}>Record Meeting</button>\*/} <span style={{alignItems:"center"}}>

<Record/>

<br/>

<button onClick={shareScreen} className={"btn btn-warning"}>Share

Screen</button> &nbsp;&nbsp;

<Link to={'/'} className={"btn btn-danger"}>End Call</Link> </span>

<br/><br/> <br/>

<div style={{color: "white"}}>

<span style={{textAlign: "center"}}>About US | Contact US | Support US </span>

<br/>

<span style={{textAlign: "center"}}>@VideoCall. All Rights Recieved</span>

<br/>

<span style={{textAlign: "center"}}>Made in <span style={{color:

"red"}}> </span> with Open Source Software</span> <br/><br/>

</div> </div>

); };

export default Room;

**3.8 Home.js**

import React, {Component} from 'react'; import firebase from "firebase";

import logo from "./logo.png";

import StyleFirebaseAuth from "react-firebaseui/StyledFirebaseAuth"; import {StyledFirebaseAuth} from "react-firebaseui/index";

import CreateRoom from "./CreateRoom";

import CreateTextRoom from "./CreateTextRoom";

firebase.initializeApp({

apiKey: "AIzaSyBF0G562vX2uA3zbBBM\_oY6L5j-l1hdmfE", authDomain: "videocall-4ec29.firebaseapp.com",

});

// IF sigined in Then the Dashboard, Else Sign in Screen class Home extends Component {

state = {isSignedIn: false} uiConfig = {

signInFlow: "popup", signInOptions: [

firebase.auth.GoogleAuthProvider.PROVIDER\_ID, firebase.auth.FacebookAuthProvider.PROVIDER\_ID, firebase.auth.GithubAuthProvider.PROVIDER\_ID, firebase.auth.EmailAuthProvider.PROVIDER\_ID,

], callback: {

signInSuccess: () => false }

} componentDidMount() {

firebase.auth().onAuthStateChanged(user => { this.setState({isSignedIn: !!user})

// console.log(user);

}) }

render() { return (

<div className="App"> {this.state.isSignedIn ? (

<div style={{backgroundColor: "black"}}> <br/><br/>

<img src={logo} style={{height: 150}} alt={""}/> <br/><br/>

<h1 style={{color: "white"}}>Welcome

{firebase.auth().currentUser.displayName}</h1>

<p style={{color: "grey"}}>Simple and easy way to connect with

friends !</p>

<br/><br/>

<img src={firebase.auth().currentUser.photoURL} alt={"Profile Image"} height={"200"}/><br/>

<div style={{color: "white"}}> <p>{firebase.auth().currentUser.displayName}</p> <p>{firebase.auth().currentUser.email}</p> <p>{firebase.auth().currentUser.phoneNumber}</p>

</div>

<br/><br/>

<button onClick={() => firebase.auth().signOut()} className={"btn

btn-danger"}> Sign out ! </button>

&nbsp; &nbsp;

<CreateRoom/> &nbsp; &nbsp; <CreateTextRoom/> <br/><br/><br/><br/><br/> <footer style={{color: "white"}}>

<div>

<span style={{textAlign: "center"}}>

About US | Contact US | Support US

</span> </div>

<br/>

<span style={{textAlign: "center"}}>@VideoCall. All Rights

Recieved</span>

<br/>

"red"}}>

<span style={{textAlign: "center"}}>Made in <span style={{color: </span> with Open Source Software</span>

<br/><br/> </footer>

</div>) :

(<div style={{backgroundColor: "black"}}>

<br/><br/>

<img src={logo} style={{height: 150}} alt={""}/>

<br/> <br/>

<h1 style={{color: "white"}}>Welcome to VideoCall App</h1> <br/><br/>

<h4 style={{color: "white"}}>SignIn to your Account</h4> <StyledFirebaseAuth

uiConfig={this.uiConfig}

firebaseAuth={firebase.auth()} />

<footer style={{color: "white"}}> <div>

<span style={{textAlign: "center"}}> About US | Contact US | Support US

</span>

</div>

<br/>

<span style={{textAlign: "center"}}>@VideoCall. All Rights

Recieved</span>

<br/>

<span style={{textAlign: "center"}}>Made in <span style={{color:

</span> with Open Source Software</span> <br/><br/>

</footer> </div>)

} </div>

); }

}

export default Home;

**3.9 Recording.js**

import React from "react";

import useScreenRecording from "use-screen-recording"; export default function Record() {

const { isRecording, recording, toggleRecording } = useScreenRecording(); let file\_url

return (

<div>

<button onClick={toggleRecording} className={"btn btn-success"}>

{isRecording ? "Stop Recording" : "Start Recording"} </button>

{!!recording && window.open(URL.createObjectURL(recording))

// (file\_url = URL.createObjectURL(recording))

// <video autoPlay src={recording && URL.createObjectURL(recording)} />

} </div>

); }

**3.10 Eye Ball Detection**

{

 "cells": [

  {

   "cell\_type": "code",

   "execution\_count": 1,

   "metadata": {},

   "outputs": [],

   "source": [

    "import os\n",

    "import cv2\n",

    "import numpy as np\n",

    "\n",

    "\n",

    "def init\_cv():\n",

    "    \"\"\"loads all of cv2 tools\"\"\"\n",

    "    face\_detector = cv2.CascadeClassifier(\"haarcascade\_frontalface\_default.xml\")\n",

    "    eye\_detector = cv2.CascadeClassifier('haarcascade\_eye.xml')\n",

    "    detector\_params = cv2.SimpleBlobDetector\_Params()\n",

    "    detector\_params.filterByArea = True\n",

    "    detector\_params.maxArea = 1500\n",

    "    detector = cv2.SimpleBlobDetector\_create(detector\_params)\n",

    "\n",

    "    return face\_detector, eye\_detector, detector\n",

    "\n",

    "\n",

    "def detect\_face(img, img\_gray, cascade):\n",

    "    \"\"\"\n",

    "    Detects all faces, if multiple found, works with the biggest. Returns the following parameters:\n",

    "    1. The face frame\n",

    "    2. A gray version of the face frame\n",

    "    2. Estimated left eye coordinates range\n",

    "    3. Estimated right eye coordinates range\n",

    "    5. X of the face frame\n",

    "    6. Y of the face frame\n",

    "    \"\"\"\n",

    "    coords = cascade.detectMultiScale(img, 1.3, 5)\n",

    "\n",

    "    if len(coords) > 1:\n",

    "        biggest = (0, 0, 0, 0)\n",

    "        for i in coords:\n",

    "            if i[3] > biggest[3]:\n",

    "                biggest = i\n",

    "        biggest = np.array([i], np.int32)\n",

    "    elif len(coords) == 1:\n",

    "        biggest = coords\n",

    "    else:\n",

    "        return None, None, None, None, None, None\n",

    "    for (x, y, w, h) in biggest:\n",

    "        frame = img[y:y + h, x:x + w]\n",

    "        frame\_gray = img\_gray[y:y + h, x:x + w]\n",

    "        lest = (int(w \* 0.1), int(w \* 0.45))\n",

    "        rest = (int(w \* 0.55), int(w \* 0.9))\n",

    "        X = x\n",

    "        Y = y\n",

    "\n",

    "    return frame, frame\_gray, lest, rest, X, Y\n",

    "\n",

    "\n",

    "def detect\_eyes(img, img\_gray, lest, rest, cascade):\n",

    "    \"\"\"\n",

    "    :param img: image frame\n",

    "    :param img\_gray: gray image frame\n",

    "    :param lest: left eye estimated position, needed to filter out nostril, know what eye is found\n",

    "    :param rest: right eye estimated position\n",

    "    :param cascade: Hhaar cascade\n",

    "    :return: colored and grayscale versions of eye frames\n",

    "    \"\"\"\n",

    "    leftEye = None\n",

    "    rightEye = None\n",

    "    leftEyeG = None\n",

    "    rightEyeG = None\n",

    "    coords = cascade.detectMultiScale(img\_gray, 1.3, 5)\n",

    "\n",

    "    if coords is None or len(coords) == 0:\n",

    "        pass\n",

    "    else:\n",

    "        for (x, y, w, h) in coords:\n",

    "            eyecenter = int(float(x) + (float(w) / float(2)))\n",

    "            if lest[0] < eyecenter and eyecenter < lest[1]:\n",

    "                leftEye = img[y:y + h, x:x + w]\n",

    "                leftEyeG = img\_gray[y:y + h, x:x + w]\n",

    "                leftEye, leftEyeG = cut\_eyebrows(leftEye, leftEyeG)\n",

    "            elif rest[0] < eyecenter and eyecenter < rest[1]:\n",

    "                rightEye = img[y:y + h, x:x + w]\n",

    "                rightEyeG = img\_gray[y:y + h, x:x + w]\n",

    "                rightEye, rightEye = cut\_eyebrows(rightEye, rightEyeG)\n",

    "            else:\n",

    "                pass  # nostril\n",

    "    return leftEye, rightEye, leftEyeG, rightEyeG\n",

    "\n",

    "\n",

    "def process\_eye(img, threshold, detector, prevArea=None):\n",

    "    \"\"\"\n",

    "    :param img: eye frame\n",

    "    :param threshold: threshold value for threshold function\n",

    "    :param detector:  blob detector\n",

    "    :param prevArea: area of the previous keypoint(used for filtering)\n",

    "    :return: keypoints\n",

    "    \"\"\"\n",

    "    \_, img = cv2.threshold(img, threshold, 255, cv2.THRESH\_BINARY)\n",

    "    img = cv2.erode(img, None, iterations=2)\n",

    "    img = cv2.dilate(img, None, iterations=4)\n",

    "    img = cv2.medianBlur(img, 5)\n",

    "    keypoints = detector.detect(img)\n",

    "    if keypoints and prevArea and len(keypoints) > 1:\n",

    "        tmp = 1000\n",

    "        for keypoint in keypoints:  # filter out odd blobs\n",

    "            if abs(keypoint.size - prevArea) < tmp:\n",

    "                ans = keypoint\n",

    "                tmp = abs(keypoint.size - prevArea)\n",

    "        keypoints = np.array(ans)\n",

    "\n",

    "    return keypoints\n",

    "\n",

    "def cut\_eyebrows(img, imgG):\n",

    "    height, width = img.shape[:2]\n",

    "    img = img[15:height, 0:width]  # cut eyebrows out (15 px)\n",

    "    imgG = imgG[15:height, 0:width]\n",

    "\n",

    "    return img, imgG\n",

    "\n",

    "\n",

    "def draw\_blobs(img, keypoints):\n",

    "    \"\"\"Draws blobs\"\"\"\n",

    "    cv2.drawKeypoints(img, keypoints, img, (0, 0, 255), cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)\n"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": null,

   "metadata": {},

   "outputs": [],

   "source": [

    "import sys\n",

    "import cv2\n",

    "import numpy as np\n",

    "from PyQt5.QtCore import QTimer\n",

    "from PyQt5.QtWidgets import QApplication, QMainWindow\n",

    "from PyQt5.uic import loadUi\n",

    "from PyQt5.QtGui import QPixmap, QImage\n",

    "\n",

    "\n",

    "class Window(QMainWindow):\n",

    "    def \_\_init\_\_(self):\n",

    "        super(Window, self).\_\_init\_\_()\n",

    "        loadUi('GUImain.ui', self)\n",

    "        with open(\"styles.css\", \"r\") as css:\n",

    "            self.setStyleSheet(css.read())\n",

    "        self.face\_decector, self.eye\_detector, self.detector = init\_cv()\n",

    "        self.startButton.clicked.connect(self.start\_webcam)\n",

    "        self.stopButton.clicked.connect(self.stop\_webcam)\n",

    "        self.camera\_is\_running = False\n",

    "        self.previous\_right\_keypoints = None\n",

    "        self.previous\_left\_keypoints = None\n",

    "        self.previous\_right\_blob\_area = None\n",

    "        self.previous\_left\_blob\_area = None\n",

    "\n",

    "    def start\_webcam(self):\n",

    "        if not self.camera\_is\_running:\n",

    "            self.capture = cv2.VideoCapture(cv2.CAP\_DSHOW)  # VideoCapture(0) sometimes drops error #-1072875772\n",

    "            if self.capture is None:\n",

    "                self.capture = cv2.VideoCapture(0)\n",

    "            self.camera\_is\_running = True\n",

    "            self.timer = QTimer(self)\n",

    "            self.timer.timeout.connect(self.update\_frame)\n",

    "            self.timer.start(2)\n",

    "\n",

    "    def stop\_webcam(self):\n",

    "        if self.camera\_is\_running:\n",

    "            self.capture.release()\n",

    "            self.timer.stop()\n",

    "            self.camera\_is\_running = not self.camera\_is\_running\n",

    "\n",

    "    def update\_frame(self):  # logic of the main loop\n",

    "\n",

    "        \_, base\_image = self.capture.read()\n",

    "        self.display\_image(base\_image)\n",

    "\n",

    "        processed\_image = cv2.cvtColor(base\_image, cv2.COLOR\_RGB2GRAY)\n",

    "\n",

    "        face\_frame, face\_frame\_gray, left\_eye\_estimated\_position, right\_eye\_estimated\_position, \_, \_ = detect\_face(\n",

    "            base\_image, processed\_image, self.face\_decector)\n",

    "\n",

    "        if face\_frame is not None:\n",

    "            left\_eye\_frame, right\_eye\_frame, left\_eye\_frame\_gray, right\_eye\_frame\_gray = detect\_eyes(face\_frame,\n",

    "face\_frame\_gray,\n",

    "left\_eye\_estimated\_position,\n",

    " right\_eye\_estimated\_position,\n",

    "self.eye\_detector)\n",

    "\n",

    "            if right\_eye\_frame is not None:\n",

    "                if self.rightEyeCheckbox.isChecked():\n",

    "                    right\_eye\_threshold = self.rightEyeThreshold.value()\n",

    "                    right\_keypoints, self.previous\_right\_keypoints, self.previous\_right\_blob\_area = self.get\_keypoints(\n",

    "                        right\_eye\_frame, right\_eye\_frame\_gray, right\_eye\_threshold,\n",

    "                        previous\_area=self.previous\_right\_blob\_area,\n",

    "                        previous\_keypoint=self.previous\_right\_keypoints)\n",

    "                    draw\_blobs(right\_eye\_frame, right\_keypoints)\n",

    "\n",

    "                right\_eye\_frame = np.require(right\_eye\_frame, np.uint8, 'C')\n",

    "                self.display\_image(right\_eye\_frame, window='right')\n",

    "\n",

    "            if left\_eye\_frame is not None:\n",

    "                if self.leftEyeCheckbox.isChecked():\n",

    "                    left\_eye\_threshold = self.leftEyeThreshold.value()\n",

    "                    left\_keypoints, self.previous\_left\_keypoints, self.previous\_left\_blob\_area = self.get\_keypoints(\n",

    "                        left\_eye\_frame, left\_eye\_frame\_gray, left\_eye\_threshold,\n",

    "                        previous\_area=self.previous\_left\_blob\_area,\n",

    "                        previous\_keypoint=self.previous\_left\_keypoints)\n",

    "                    draw\_blobs(left\_eye\_frame, left\_keypoints)\n",

    "\n",

    "                left\_eye\_frame = np.require(left\_eye\_frame, np.uint8, 'C')\n",

    "                self.display\_image(left\_eye\_frame, window='left')\n",

    "\n",

    "        if self.pupilsCheckbox.isChecked():  # draws keypoints on pupils on main window\n",

    "            self.display\_image(base\_image)\n",

    "\n",

    "    def get\_keypoints(self, frame, frame\_gray, threshold, previous\_keypoint, previous\_area):\n",

    "\n",

    "        keypoints = process\_eye(frame\_gray, threshold, self.detector,\n",

    "                                        prevArea=previous\_area)\n",

    "        if keypoints:\n",

    "            previous\_keypoint = keypoints\n",

    "            previous\_area = keypoints[0].size\n",

    "        else:\n",

    "            keypoints = previous\_keypoint\n",

    "        return keypoints, previous\_keypoint, previous\_area\n",

    "\n",

    "    def display\_image(self, img, window='main'):\n",

    "        # Makes OpenCV images displayable on PyQT, displays them\n",

    "        qformat = QImage.Format\_Indexed8\n",

    "        if len(img.shape) == 3:\n",

    "            if img.shape[2] == 4:  # RGBA\n",

    "                qformat = QImage.Format\_RGBA8888\n",

    "            else:  # RGB\n",

    "                qformat = QImage.Format\_RGB888\n",

    "\n",

    "        out\_image = QImage(img, img.shape[1], img.shape[0], img.strides[0], qformat)  # BGR to RGB\n",

    "        out\_image = out\_image.rgbSwapped()\n",

    "        if window == 'main':  # main window\n",

    "            self.baseImage.setPixmap(QPixmap.fromImage(out\_image))\n",

    "            self.baseImage.setScaledContents(True)\n",

    "        if window == 'left':  # left eye window\n",

    "            self.leftEyeBox.setPixmap(QPixmap.fromImage(out\_image))\n",

    "            self.leftEyeBox.setScaledContents(True)\n",

    "        if window == 'right':  # right eye window\n",

    "            self.rightEyeBox.setPixmap(QPixmap.fromImage(out\_image))\n",

    "            self.rightEyeBox.setScaledContents(True)\n",

    "\n",

    "\n",

    "if \_\_name\_\_ == \"\_\_main\_\_\":\n",

    "    app = QApplication(sys.argv)\n",

    "    window = Window()\n",

    "    window.setWindowTitle(\"GUI test\")\n",

    "    window.show()\n",

    "    sys.exit(app.exec\_())\n"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": null,

   "metadata": {},

   "outputs": [],

   "source": []

  }

 ],

 "metadata": {

  "kernelspec": {

   "display\_name": "Python 3",

   "language": "python",

   "name": "python3"

  },

  "language\_info": {

   "codemirror\_mode": {

    "name": "ipython",

    "version": 3

   },

   "file\_extension": ".py",

   "mimetype": "text/x-python",

   "name": "python",

   "nbconvert\_exporter": "python",

   "pygments\_lexer": "ipython3",

   "version": "3.7.6"

  }

 },

 "nbformat": 4,

 "nbformat\_minor": 4

}

3.11 Eye mark

import cv2

from google.colab.patches import cv2\_imshow

image = cv2.imread("/content/Rajesh R Photo.jpg")

eye\_cascade = cv2.CascadeClassifier('haarcascade\_eye.xml')

eyes = eye\_cascade.detectMultiScale(image, scaleFactor = 1.1, minNeighbors = 2)

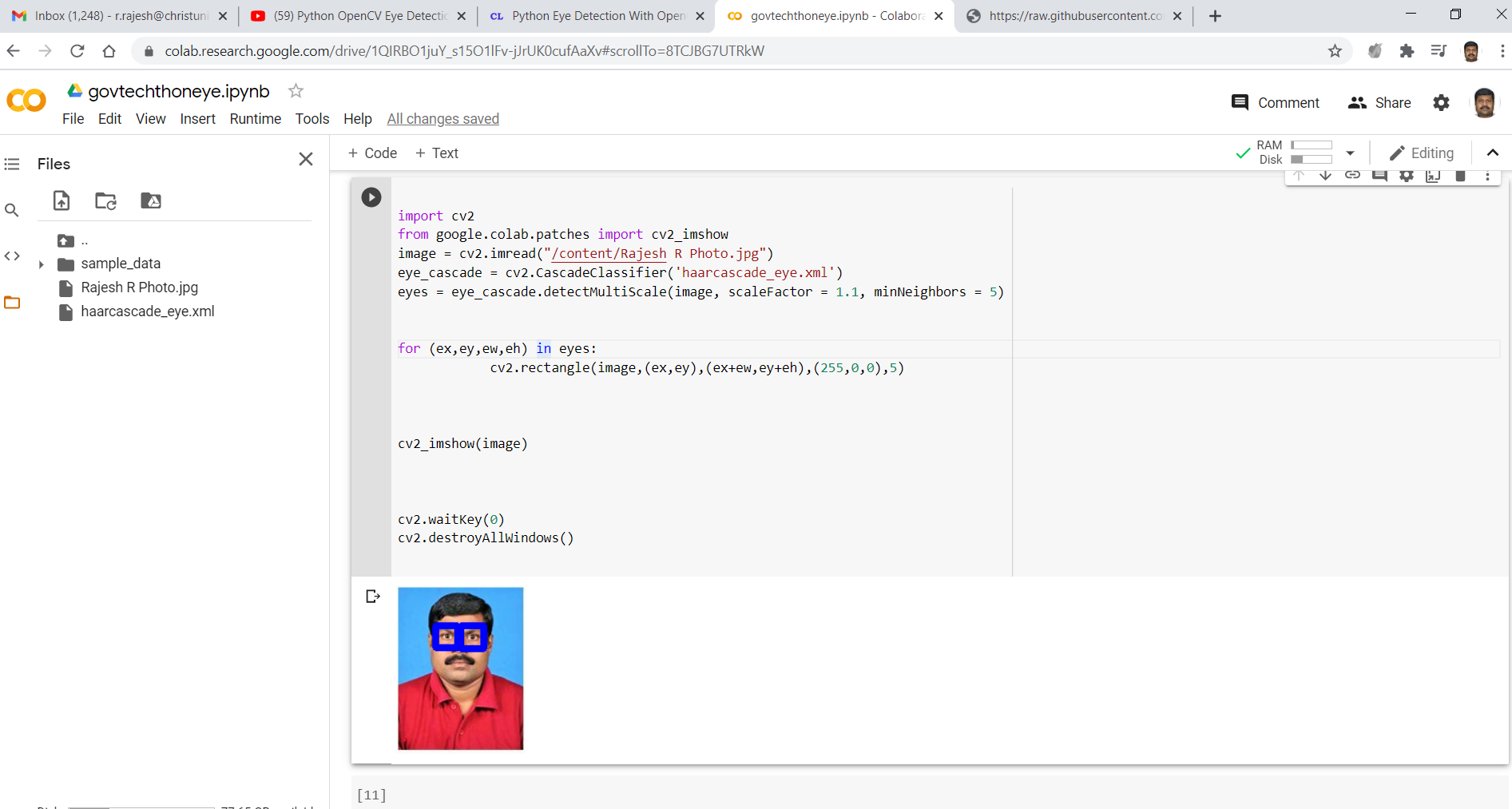
for (ex,ey,ew,eh) in eyes:

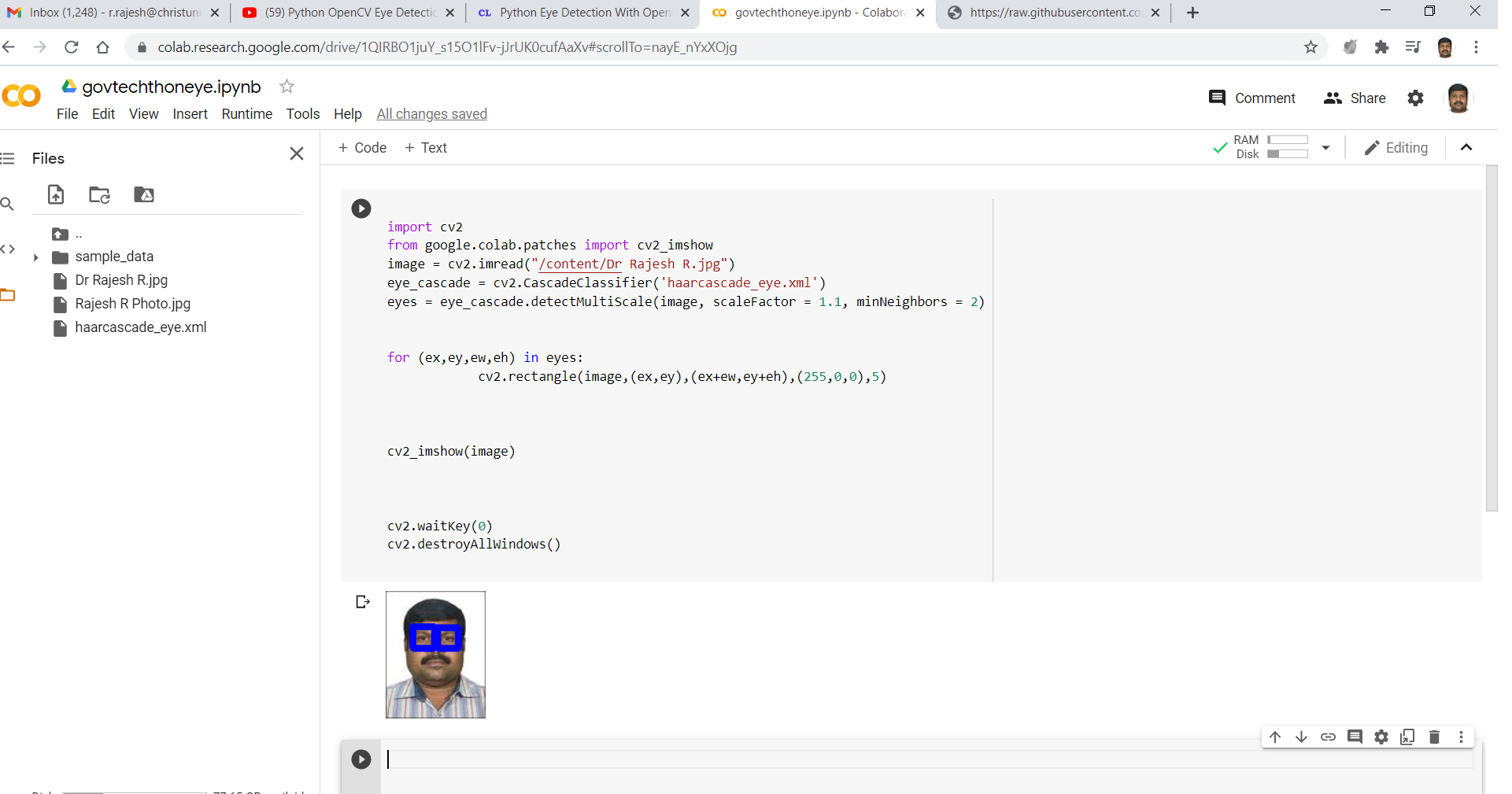
            cv2.rectangle(image,(ex,ey),(ex+ew,ey+eh),(255,0,0),5)

cv2\_imshow(image)

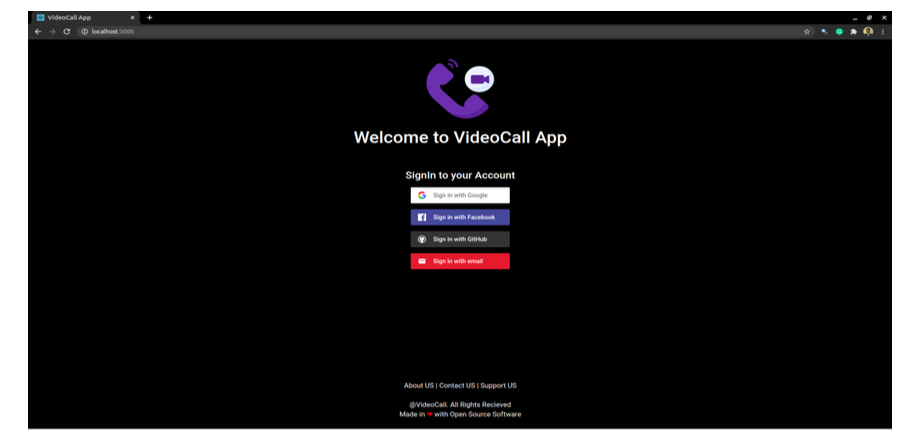
cv2.waitKey(0)

cv2.destroyAllWindows()

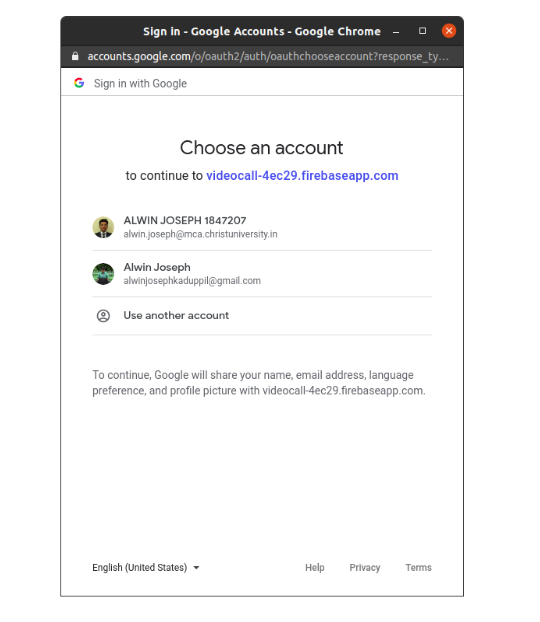




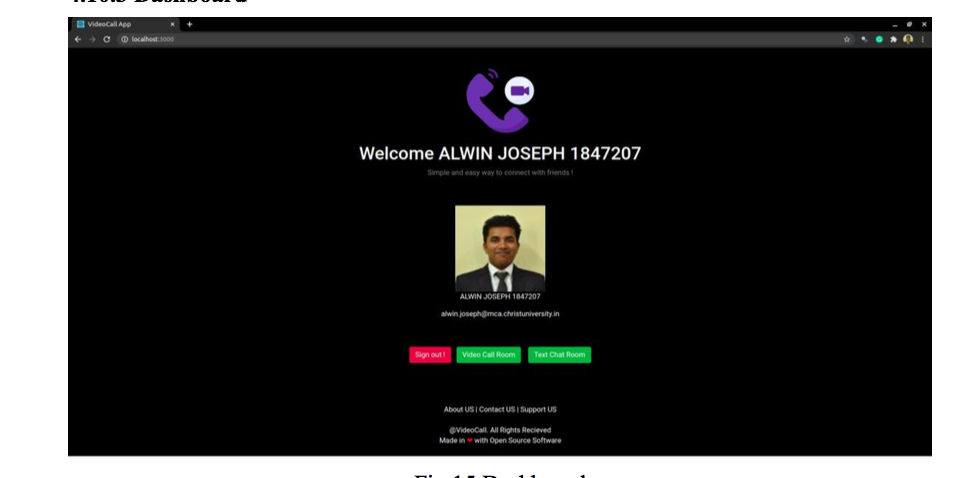
**Home Screen**

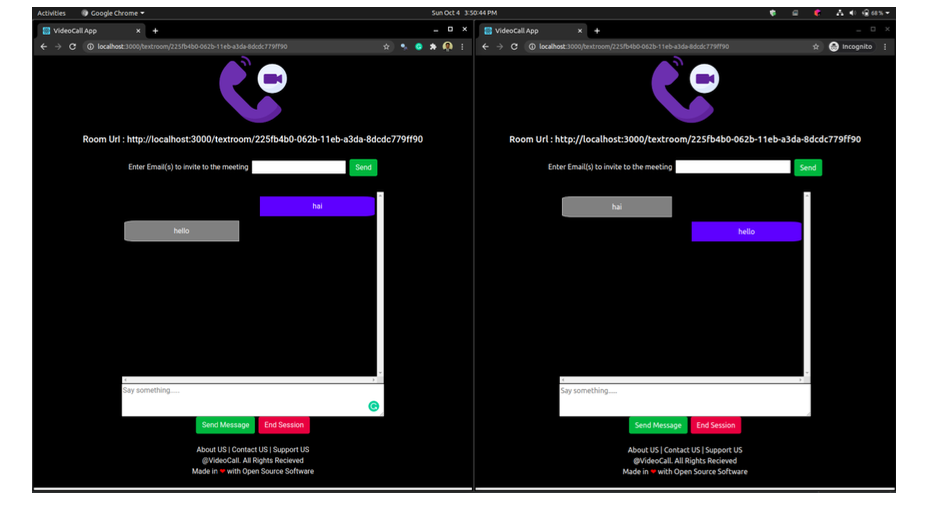


**Google Authentication**

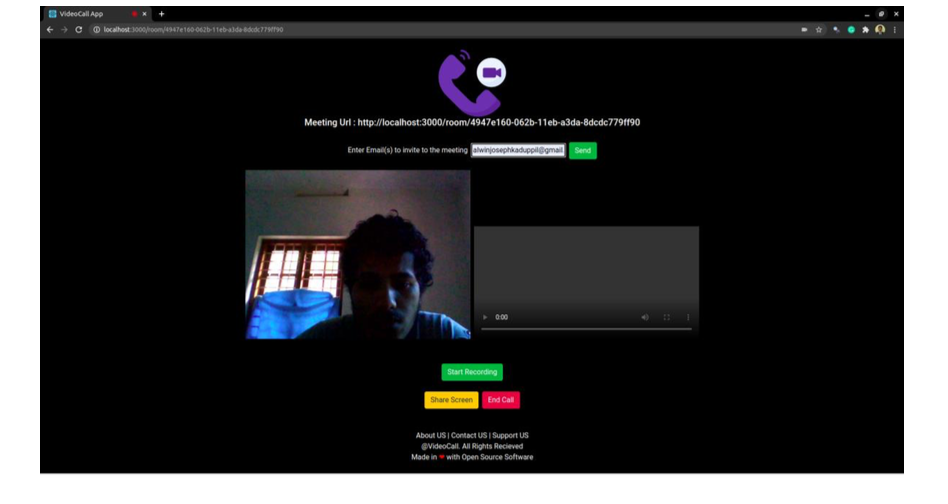


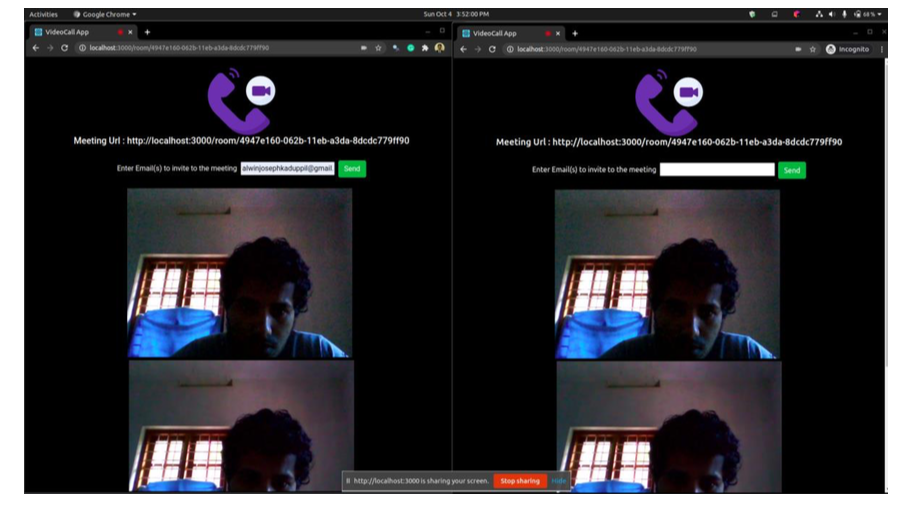
**Dashboard**



**Chat Rooms**  

**Video Call Room**

****

****

**Tools used**

Web based application using python