SHRI BHAGUBHAI MAFATLAL POLYTECHNIC

Irla, N.R.G. Marg, Vile Parle(West), Mumbai:- 400056

PROJECT REPORT ON

STAMP DRONE

Submitted By

Prinkal Doshi	T107
Mihir Panchal	T118
Arshkumar Sakaria	T127
Tanay Desai	T137
Sarid Qureshi	T154

Guided By

Mr. Abhijit Dongaonkar

Term - 2022 - 2023

Information Technology Department

SHRI BHAGUBHAI MAFATLAL POLYTECHNIC

Vile Parle(West), Mumbai:- 400056

PROJECT REPORT ON

Title: _____

Submitted in Parallel Fulfilment of the Requirements

The Diploma Course/ Post Diploma Course/ Advance Technology Course by

Name of the Student: _____

_____

Roll No: _____

SAP No: _____

Academic Year: _____

Name of Department: _____

Year / Semester: _____

Name and Address of the Company: _____

(If any sponsored Project/ Internship Training)

_____

_____

THIS IS TO CERTIFY THAT

Shri./Smt./Kum. _____

Exam Seat No: _____Has Satisfactorily Completed

his/her Project Work and Submitted

Guide Head Of Department Principal

Date: _____

SHRI BHAGUBHAI MAFATLAL POLYTECHNIC

Vile Parle(West), Mumbai:- 400056

PROJECT APPROVAL SHEET

THIS IS TO CERTIFY THAT

Shri./Smt./Kum._____

_____

SAP No. _____

has presented a Project Entitled_____

In Partial Fulfilment of

Diploma Program in Computer Engineering / Information Technology

Same is Approved By:

1. **External Examiner** _____
   Name and Signature

2. **Internal Examiner** _____
   Name and Signature

Date: _____

ABSTRACT

Abstract

This project presents a comprehensive drone-based surveillance system designed to enhance rescue operations, improve security measures, and facilitate disaster analysis. The system primarily focuses on implementing facial recognition technology (Haar Cascade) [6], enabling the identification of individuals within crowded areas. The use of drones allows for efficient data collection through video feeds, providing valuable insights for law enforcement agencies and rescue teams.

The implemented facial recognition feature serves as a powerful tool for identifying specific individuals within a crowd, aiding in security efforts and enabling quicker response times. By automating the identification process, law enforcement agencies can streamline their operations and enhance public safety. The system utilizes advanced algorithms to analyze facial features and match them against a database of known individuals, ensuring accurate identification even in challenging environments.

In addition to facial recognition, the system utilizes video analysis techniques to extract valuable information from the collected video data. This analysis includes disaster assessment, people distribution, and situational awareness. By analyzing the footage captured by the drones, the system can provide insights into the depth and extent of a disaster, allowing emergency management teams to better understand the affected areas and allocate resources accordingly. The distribution of people within a disaster-stricken location can also be assessed, aiding in the planning and execution of rescue operations.

The integration of the surveillance system with facial recognition technology and video analysis capabilities enables a wide range of applications. In terms of security, the system can identify persons of interest, such as known criminals or missing persons, within a crowd, facilitating faster response times and improved public safety. Law enforcement agencies can also use the system to monitor public gatherings, identify potential threats, and maintain law and order.

In the context of disaster management, the system plays a crucial role in assessing the extent of a disaster and aiding in rescue operations. By analyzing the video data, emergency response teams can gain insights into the severity and spread of the disaster, allowing for more informed decision-making. Additionally, it can aid in the identification and tracking of individuals who may require immediate assistance, improving the efficiency and effectiveness of rescue operations.

The drone-based surveillance system has been developed with a user-friendly interface, allowing operators to easily control and monitor the drones in real-time. The system is scalable and adaptable, capable of integrating with existing infrastructure and adapting to different environments and scenarios.

In conclusion, this project presents a drone-based surveillance system that leverages facial recognition technology and video analysis capabilities to enhance rescue operations, improve security measures, and facilitate disaster analysis. The integration of advanced algorithms and high-resolution cameras enables accurate identification of individuals within crowded areas and provides valuable insights into disaster scenarios. The system demonstrates the potential of drone technology and advanced analytics in enhancing situational awareness, optimizing resource allocation, and ultimately improving public safety and disaster response strategies.

INDEX

INTRODUCTION

INTRODUCTION

A quadcopter, commonly known as a drone, is a multi-rotor helicopter propelled and lifted by four rotors. Unlike traditional helicopters, quadcopters utilize two sets of identical fixed-pitched propellers, consisting of two clockwise (CW) and two counter-clockwise (CCW) propellers. The speed and direction of rotation of these propellers can be controlled to achieve lift, thrust, and torque. The control of a quadcopter's motion is achieved by adjusting the rotation rate of one or more rotor discs, thereby altering its thrust, lift, and torque characteristics. This control is facilitated by a microcontroller embedded in the drone's system.

The rotation speed and direction of the quadcopter's propellers are adjusted based on the user's input, allowing the device to move in various directions such as takeoff, landing, forward, backward, left, and right motions. Each rotor generates both thrust and torque around its center of rotation, as well as a drag force opposite to the direction of the quadcopter's flight.

In this project, the drone is equipped with IR sensors on either side, enabling continuous sensing of obstacles along its flight path. These sensors detect objects and trigger the object avoidance code, causing the drone to maneuver and avoid collisions. This obstacle sensing capability enhances the safety and stability of the drone's flight.

Simultaneously, the drone captures video footage using its camera module, which is attached to the front of the drone. The recorded video is then uploaded to a connected device for further processing and analysis. This video feed serves as a valuable resource for various functions and applications.

For instance, in the case of a missing person, the captured video is processed using a Haar cascade facial recognition system, enabling the successful identification of the missing individual. This application can greatly assist in search and rescue operations, enhancing the chances of locating the person in a timely manner.

Furthermore, in situations involving man-made or natural disasters or calamities, the recorded video footage can be utilized to create a comprehensive report. This report can provide valuable insights and information for conducting rescue operations more effectively and efficiently. By analyzing the recorded video, emergency response teams can gain a better understanding of the affected areas, assess the extent of the disaster, and make informed decisions regarding resource allocation and strategic planning.

For the purposes of this project, the team utilized the Drona Aviation PlutoX, a programmable drone [4]. This drone is equipped with the necessary features and capabilities required for the implementation of the proposed system. Its programmable nature allowed for customization and integration of various sensors and modules, making it an ideal platform for developing a sophisticated surveillance and rescue system.

In summary, this project leverages the capabilities of a quadcopter drone, equipped with obstacle sensing, video capturing, and facial recognition functionalities, to enhance safety, facilitate search and rescue operations, and improve disaster response strategies. The utilization of advanced technologies in a programmable drone demonstrates the potential for innovation and effectiveness in addressing real-world challenges.

TIMELINE CHART

	Week															
Work	l	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Project Analysis																
UI Development																
Dataset Collection/ Analyzing																
Dataset cleaning																
Object avoidance																
Object detection																
Stamp Support																
Hand Gesture Recognition																
Facial Recognition																
Cloud Connectivity																
Drone Programming																
Presentation																
Documentation																

SOFTWARE REQUIREMENTS

1. VS CODE

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality.

2. Cygnus:

Cygnus is an Eclipse based IDE. It comes bundled with GCC compiler, prebuilt libraries and Wireless flashing for Primus flight controllers. It is used to program the drone's flight controller.

3. Pluto Blocks:

Pluto Blocks is a software for programming your Pluto drone's firmware using block Programming. It comes bundled with GCC compiler and prebuilt libraries for Pluto flight Controllers. It is used to program the drone's flight controller.

4. Node.js:

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting-running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser [1].

5.GITHUB

GitHub, Inc. is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

6.AWS

Amazon Web Services, Inc. is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Often times, clients will use this in combination with autoscaling [7].

LANGUAGES USED

1. CPP:

The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low level memory manipulation. It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Oracle, and IBM, so it is available on many platforms. C++ was designed with an orientation toward systems programming and embedded, resource-constrained software and large systems, with performance, efficiency, and flexibility of use as its design highlight.

2. HTML/CSS/JavaScript:

The Hypertext Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

JavaScript often abbreviated JS is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

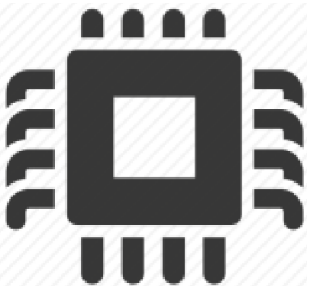
3. GO

Go is a statically typed, compiled high-level programming language designed at Google by Robert Griesemer, Rob Pike, and Ken Thompson. It is syntactically similar to C, but with memory safety, garbage collection, structural typing, and CSP-style concurrency.
Software developers use Go in an array of operating systems and frameworks to develop web applications, cloud and networking services, and other types of software.

4. PYTHON

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often describes as a "battery included" language due to its comprehensive standard library [3].



HARDWARE REQUIREMENTS

COMPONENTS	DESCRIPTION
Programmable Flight Controller (Pluto X)	Primus X is a flight controller for Pluto X. When implementing your idea, modularity in hardware, software and design is important.
Propellers	There are 4 propellers in use which is approximately about 135 mm
Brushed Motor	A brushed DC electric motor is an internally commutated electric motor designed to be run from a direct current power source and utilizing an electric brush for contact brushed DC motors can be varied in speed by changing the operating voltage or the strength of the magnetic field. Size is 8.5 mm
Battery	The battery used here is of 3.7v (1200m Ah)
Mobile to control the drone	We will us Pluto controller app to control the drone using WIFI instead of RC controller.
X-breakout board X	X-Breakout is designed to expose basic communication ports and prototyping.
Sensors	Accelerometer, Barometer, Magnetometer, Gyroscope. To used IR sensor to achieve obstacles avoidance

Drone Specifications[4]

CASE STUDY

CASE STUDY

In this case study, we present the development of a comprehensive web console for our security and surveillance drone project. This web console serves as a central hub for various functionalities related to the drone, including video upload and analysis, facial recognition algorithms, drone calibration, motor testing, and customer support. The website aims to provide a user-friendly interface and enhance the overall experience of operating and managing the drone.

Web Console Features:

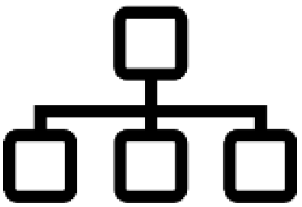
1. Video Upload and Analysis:
The web console allows users to conveniently upload video footage captured by the drone. Once uploaded, the website facilitates the analysis of the video using advanced facial recognition algorithms. This analysis can help identify individuals and enhance the efficiency of security and rescue operations. The analyzed results are then presented to the user, providing valuable insights and information.

2. Drone Calibration and Motor Testing:
To ensure optimal performance and stability, the web console includes features for drone calibration and motor testing. Users can access these functionalities, which enable them to fine-tune and adjust the drone's settings. Through the web console, users can configure the drone's flight characteristics, such as sensitivity, responsiveness, and stability. Additionally, motor testing allows users to verify the functionality and performance of each of the four motors, ensuring smooth and reliable drone operation.

3. Contact Us and Support Bot:
The web console incorporates a "Contact Us" section, providing users with a means to reach out to the development team for any queries, feedback, or technical support. This feature enables effective communication between users and the project team, fostering a collaborative environment. Additionally, a support bot is integrated into the website, offering automated assistance for common drone-related problems. Users can interact with the bot to seek guidance, troubleshooting tips, or general information, enhancing the overall user experience.

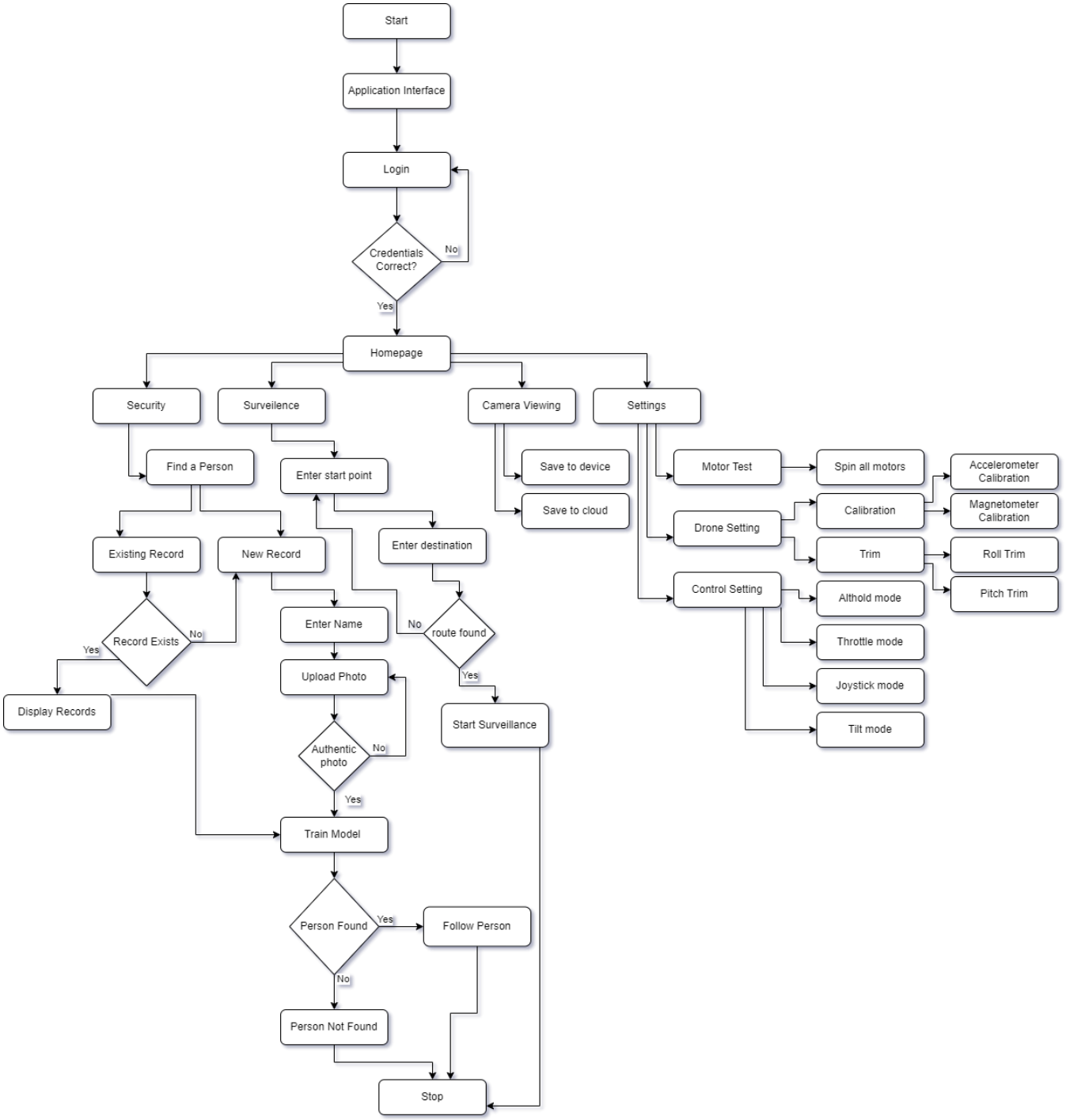
4. Gallery:
The web console includes a gallery section where users can access and view a collection of images and videos captured by the drone during its surveillance missions. This feature allows users to explore the drone's capabilities and the visual documentation it provides. The gallery serves as a showcase of the drone's accomplishments and contributes to a better understanding of its functionality.

In conclusion, the web console developed for our security and surveillance drone project serves as a versatile and user-friendly platform. Its features include video upload and analysis, facial recognition algorithms, drone calibration, motor testing, a contact us section, a support bot, and a gallery. This comprehensive web console enhances the overall experience of operating and managing the drone, providing users with valuable tools and resources. By incorporating these features, we have created a centralized hub for all drone-related functionalities and support, making the security and surveillance operations more efficient and effective.



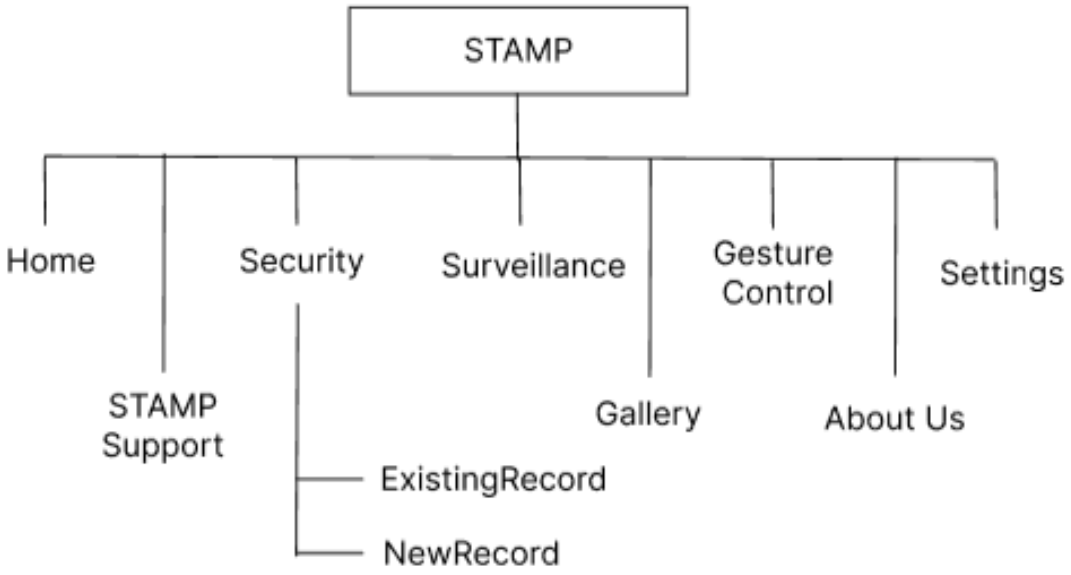
UML DIAGRAMS

FLOW CHART



The above flowchart represents the working flow of STAMP React Web Application with
Its Integration with other Modules like Face Recognition, Object Detection, Cloud Integration ,Drone Settings and Calibration.

Components



The above flowchart represents the logical schema of STAMP React Web Application which is broadly classified into 8 Components along with their sub components.

Home – Basic Overview of Project along with description of features and components used

STAMP Support – End-to-end messaging components enabled with AI and features to communicate with drone remotely via MSP Protocol

Security – Includes components to train model with faces for Haar Cascade Face recognition algorithm [6].

Surveillance – Includes Yolo Object Detection algorithm with COCO Dataset loaded with 80 object categories [5].

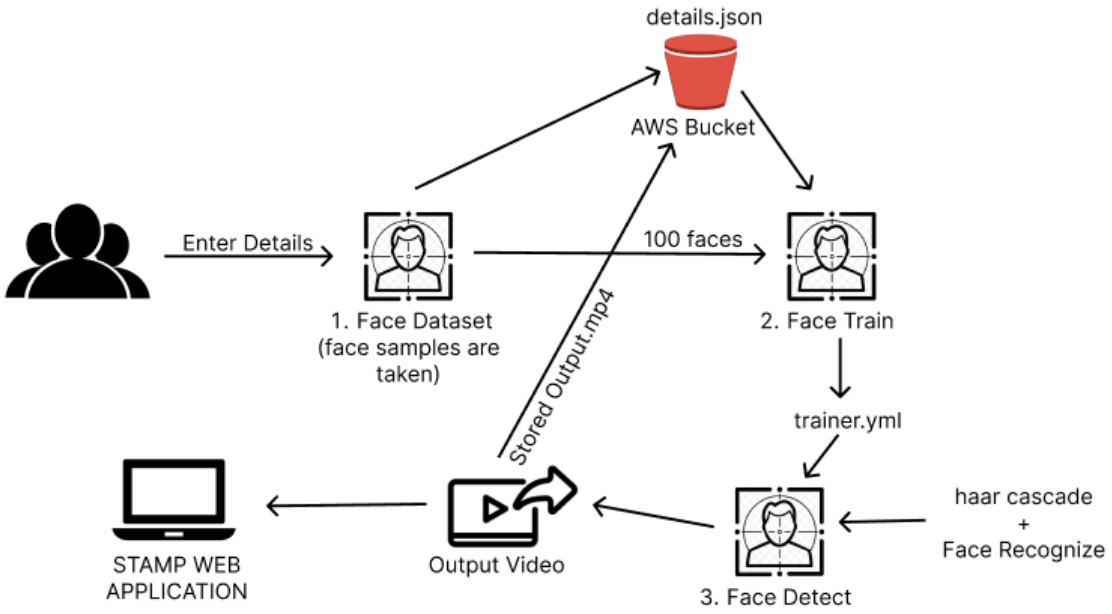
Gallery – Includes all the achievements and participation in Project Competitions.

Gesture Control – Remotely Control drone via hand gesture recognition model build with tensorflow, keras and mediapipe.

About Us – Includes summary of project along with project details and contact us module connected to AWS Cloud

Settings – Remotely Configure Drone via Web Application for optimizing drone health and monitoring drone state

Face Recognition



The above diagram represents the working of Face Recognition module of STAMP [6]

1. User will enter his/her details on the web application

2. OpenCV Instance will be instatianted to capture 100 faces of user

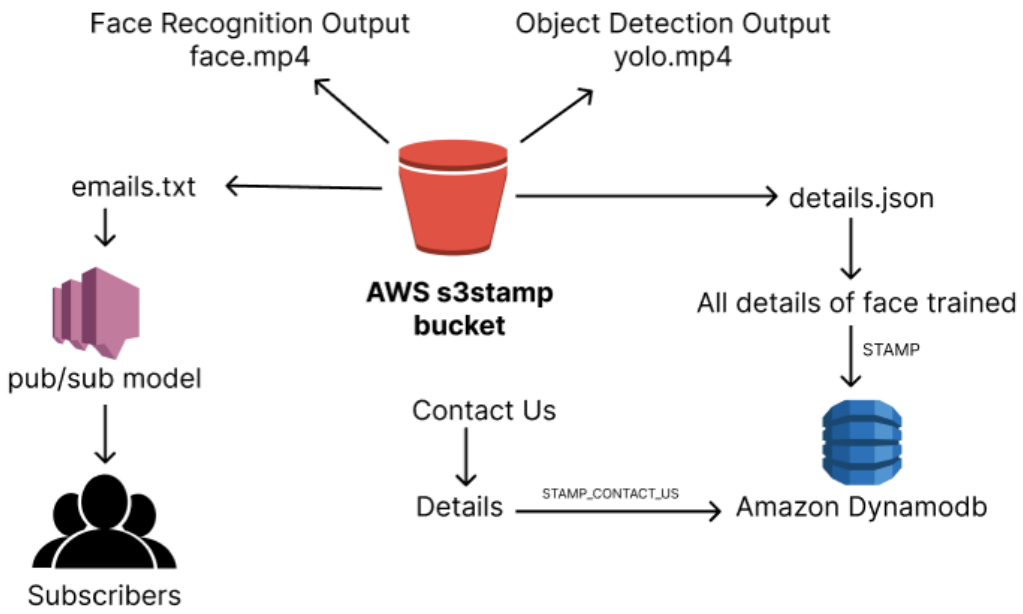
3. Faces captured are saved creating trainer.yaml containing numpy arrays for face recognition along with details.json file

4. Haar Cascade Face Recognition algorithm is applied over the input video file with reference to trainer.yaml file created.

5. Output Video is processed to generate video with boxes detect face recognition along with their confidence value.

6. Output Video is displayed over STAMP React Web Application along with features to generate and mail a report, save to disk and save to cloud functionality

Cloud Architecture



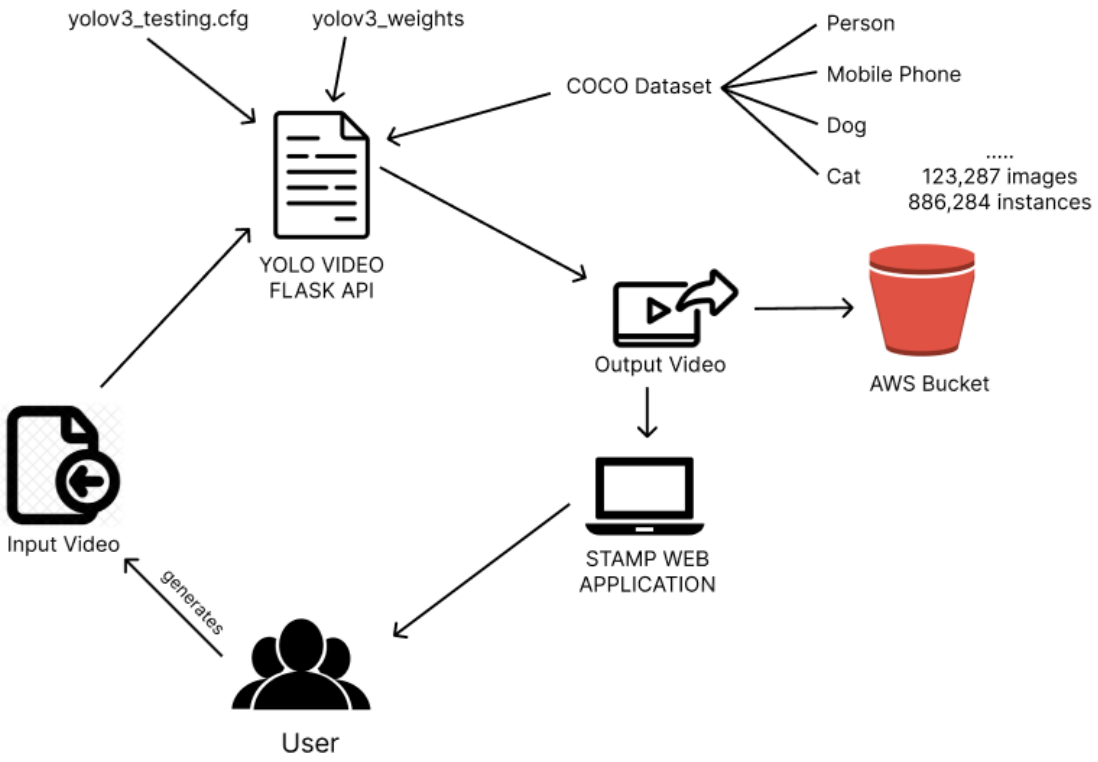
The above diagram represents the cloud architecture applied to STAMP with its implementation and working with the end users and servers [7].

AWS Simple Storage Service (S3) : It contains all the output files of face recognition and object detection algorithm along with emails.txt and details.json which are generated by AWS SNS and AWS DynamoDB.

AWS Simple Notification Service (SNS) : It contains the pub/sub model for STAMP Web application for end users to subscribe to our newsletter for further updates and notification about our project

AWS DynamoDB : It contains two NOSQL database tables namely stamp contact us and stamp details.json holding values of users who want to contact us and review our project and record of face recognition algorithm.

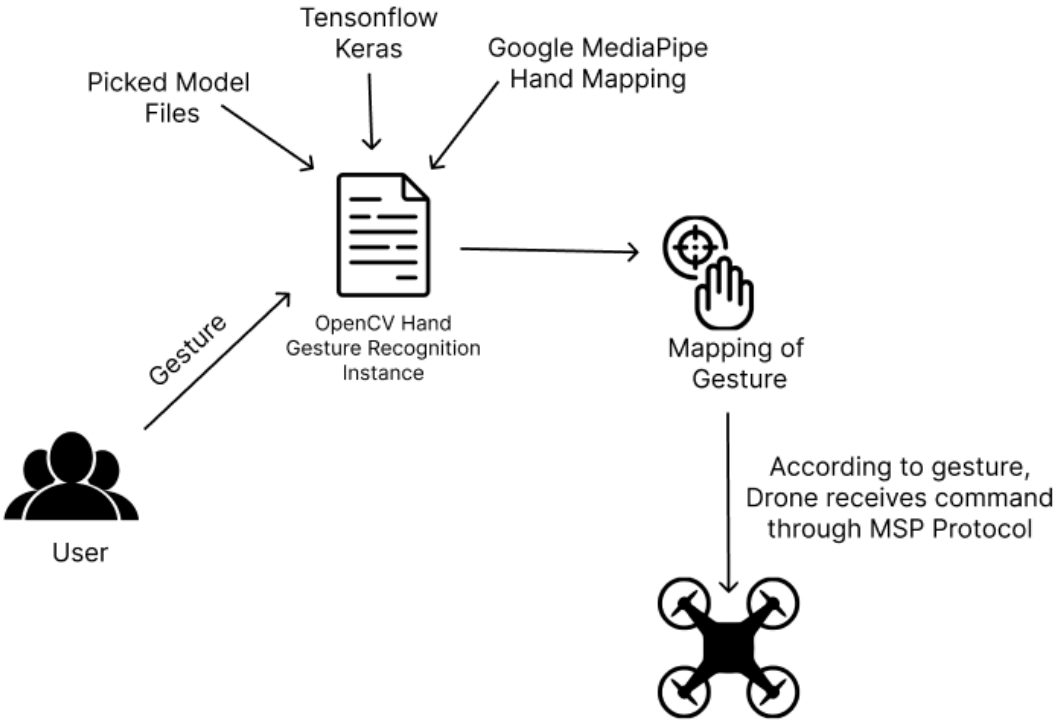
Object Detection



The above diagram represents the working of Object Detection module of STAMP [5].

1. User will upload the video file he/she wants to analyze to the STAMP Web Application.
2. The video file is feed to yolo object detection model via flask api.
3. The Model is trained with the coco dataset mapping and creating boxes of each object detected in each frame of video file.
4. The video file is uploaded to cloud and report is generated and mailed to the end user.

Hand Detection



The above diagram represents the working of Remote Drone Controlling via Hand Gesture Detection module of STAMP

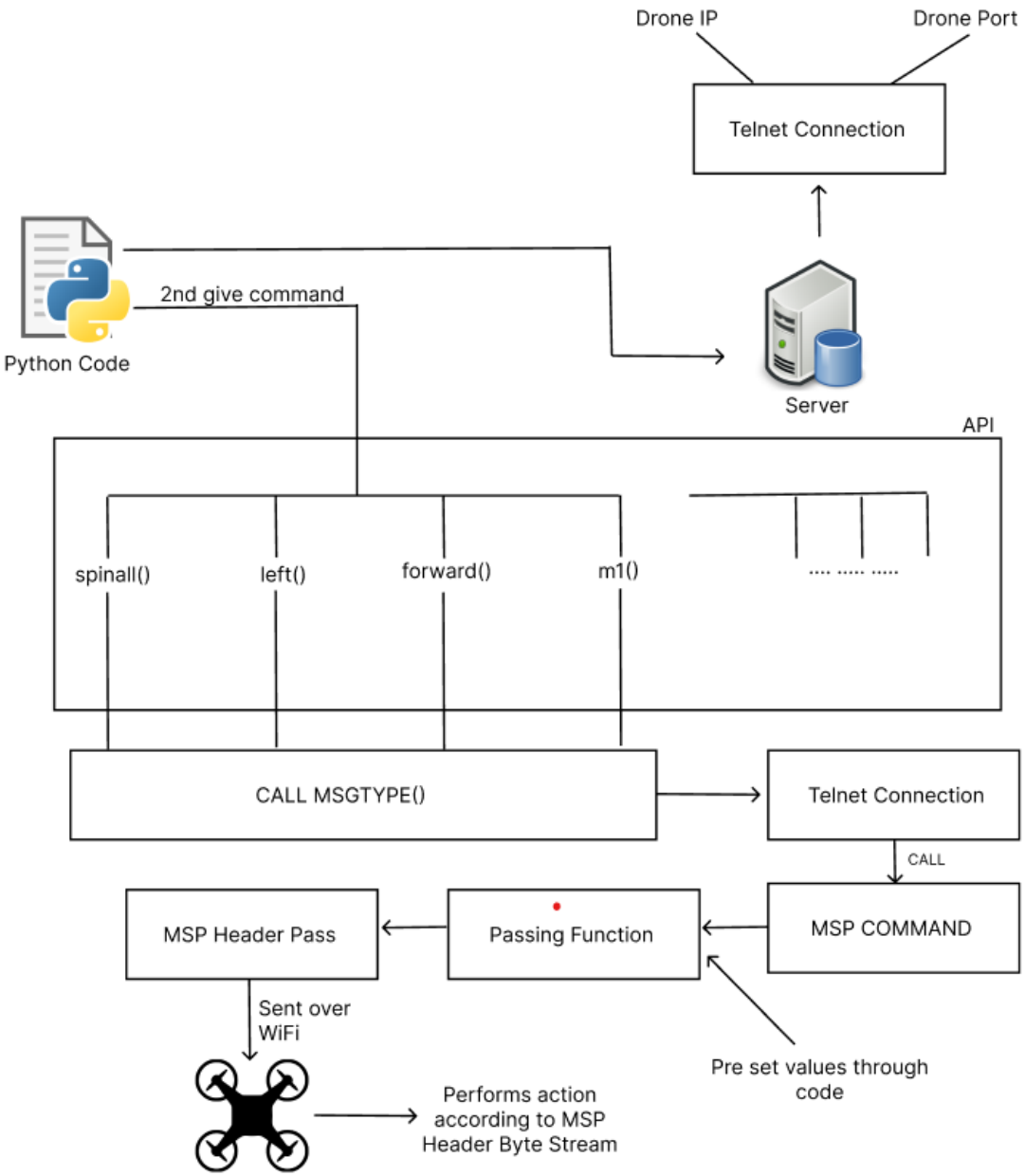
1. User will enable hand gesture detection module by clicking a button on STAMP React Web Application calling a flask api.

2. OpenCV Instance is instatiated capturing hand gestures made by user

3. Hand Gestures are recognized by the algorithm by creating pipes for each joint of the hand gesture through mediapipe and analyzed via tensorflow model.

4. Hand Gesture is mapped with associated MSP Protocol function component to send message to drone about its commanded action.

MSP Protocol



The above diagram represents the working of Plutox Drone Python API [8].

1. Drone Connection is established using Drone IP Address and Drone Port using Telnet Protocol

2. High level python functions are generated which translate MSP Protocol to set channel and pass parameters.

3. Passed Parameters are converted and compressed to a byte stream MSP Header file.

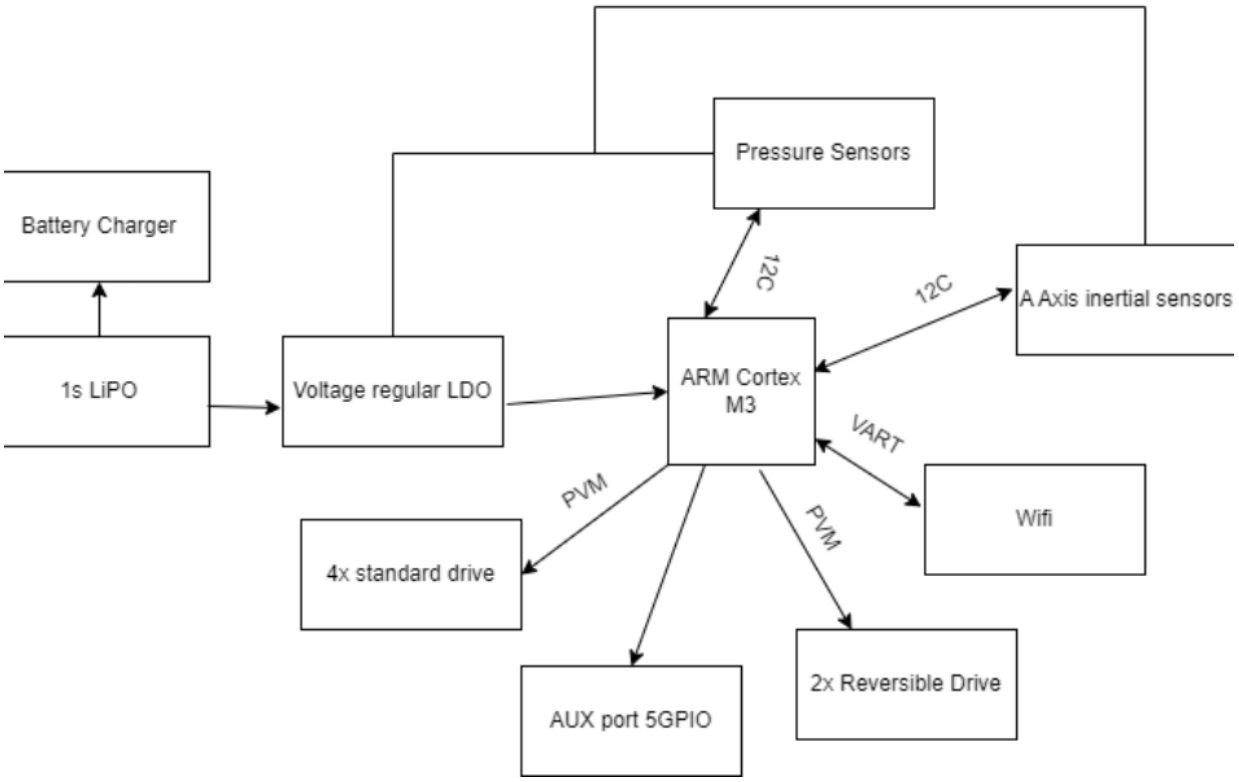
4. Header file containing command data along with command channel are send to flight controller of drone via wifi connection

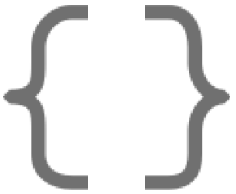
5. Drone performs according to MSP Header file and channel received via the Python API

BASIC ARCHITECTURE

BASIC ARCHITECTURE

Drone Architecture[4]





SOURCE CODE

DRONE PROGRAM

```
#include "PlutoPilot.h"
#include "Xranging.h"
#include "Sensor.h"
#include "User.h"
#include "Utils.h"

//The setup function is called once at PlutoX's hardware startup
void plutoInit ()
{
	/*Add your hardware initialization code here*/
	XRanging.init(LEFT); /*Initialize Left Ranging sensor*/
	XRanging.init(RIGHT); /*Initialize Right Ranging sensor*/
	setUserLoopFrequency( 4 ); /*Change loop frequency for faster loop
	execution*/
}

void onLoopStart ()
{
	/*do your one time tasks here*/
	LED.flightStatus(DEACTIVATE); /*Disable default Led behavior*/
}

//The function is called once before plutoLoop() when you activate developer mode
void plutoLoop ()
{
	/*Add your repeated code here*/
	/*if the sensor detects an obstacle on the left side i.e. range less than 500,
	roll right*/
	if (Acceleration.getNetAcc()< 2 &&(!FlightStatus.check(FS_CRASHED)))
	/*Condition for free fall*/
	{
		Command.arm(); /*Arm the drone*/
		LED. set (RED, ON);

		LED. set (GREEN, ON);
	}
	if (XRanging.getRange(LEFT)< 500 && XRanging.getRange(LEFT)> 0 )
	{
		RcCommand. set (RC_ROLL, 1600);
		LED. set (RED,ON);
		LED. set (BLUE,OFF);
	}
	/*if the sensor detects an obstacle on the right side i.e. range less than
	500, roll left*/
	else if (XRanging.getRange(RIGHT)< 500 && XRanging.getRange(RIGHT)> 0 )
	{
		RcCommand. set (RC_ROLL, 1400);
		LED. set (RED,OFF);
		LED. set (BLUE,ON);
	}
```

```
		/*when no obstacle is detected, let control be with the user*/
	else
	{
		RcCommand. set (RC_ROLL,RcData.get(RC_ROLL));
		LED. set (RED,OFF);
		LED. set (BLUE,OFF);
	}
}

//The function is called once after plutoLoop() when you deactivate developer mode
void onLoopFinish()
{
//Do your cleanup stuff here
	LED.flightStatus(ACTIVATE); /*Enable the default led behavior*/
}
```

WEBSITE UI

App.js

```
import React from 'react';
import {
  ChakraProvider,
  Link,
  VStack,
  Code,
  Grid,
  theme,
} from '@chakra-ui/react';
import { ColorModeSwitcher } from './ColorModeSwitcher';
import { Logo } from './Logo';
import Navbar from './components/Navbar';
import Footer from './components/Footer';
import Train from './components/Security/Train';
import Loading from './components/Security/Loading';
import Homepage from './pages/Homepage';
import About from './pages/About';
import ExistingRecord from './pages/Security/ExistingRecord'
import NewRecord from './pages/Security/NewRecord'
import TestRecord from './components/Security/TestRecord'
import Propellers from './pages/Settings/Propellers';
import Directions from './pages/Settings/Directions';
import Surveillance from './components/Surveillance/Surveillance'
import CameraViewing from './components/GestureControl/GestureViewing';
import Detect from './components/Security/Detect'
import { BrowserRouter, Route, Routes} from "react-router-dom";

function App() {
  return (
    <ChakraProvider theme={theme}>
      <BrowserRouter>
      <Navbar/>
      <Routes>
        <Route path='/' element={<Homepage/>} />
        <Route path='/About' element={<About />} />
        <Route path='/Security/ExistingRecord' element={<ExistingRecord />} />
        <Route path='/Security/NewRecord' element={<NewRecord />} />
        <Route path='/Security/Train' element={<Train />} />
        <Route path='/Security/Loading' element={<Loading />} />
```

```
        <Route path='/Security/Detect' element={<Detect />} />
        <Route path='/Security/TestRecord' element={<TestRecord />} />
        <Route path='/Settings/Propellers' element={<Propellers />} />
        <Route path='/Settings/Directions' element={<Directions />} />
        <Route path='/Surveillance' element={<Surveillance />} />
        <Route path='/GestureControl' element={< CameraViewing/>} />
      </Routes>
      </BrowserRouter>
      <Footer/>
    </ChakraProvider>
  );
}

export default App;
```

Homepage.js

```
import React from 'react';
import {
  Image,
  ChakraProvider,
  Box,
  Text,
  Link,
  VStack,
  Code,
  Grid,
  theme,
  Stack,
} from '@chakra-ui/react';
import { ColorModeSwitcher } from '../ColorModeSwitcher';
import { Logo } from '../Logo';
import Section1 from '../components/Homepage/Section1';
import Section2 from '../components/Homepage/Section2';
// import Section3 from '../components/Homepage/Section3';
import Section4 from '../components/Homepage/Section4';
import bg1 from '../assets/bg1.png';

function App() {
  return (
    <>
      <Section1></Section1>
      <Section2></Section2>
      <Section4></Section4>
    </>
  );
}

export default App;
```

Section1.js

```
import React from "react";
import { chakra, Box, useColorModeValue, Icon, Image } from "@chakra-ui/react";
import Dronesbg from '../../assets/dronesbg.jpg';

export default function App(){
  const bg = useColorModeValue("white", "gray.800");
  return (
    <Box pos="relative" overflow="hidden" bg={bg} mt={0}>
      <Box maxW="7xl" mx="auto">
        <Box
          pos="relative"
          pb={{ base: 8, sm: 16, md: 20, lg: 28, xl: 32 }}
          maxW={{ lg: "2xl" }}
          w={{ lg: "full" }}
          zIndex={1}
          bg={bg}
          border="solid 1px transparent"
        >
          <Icon
            display={{ base: "none", lg: "block" }}
            position="absolute"
            right={0}
            top={0}
            bottom={0}
            h="full"
            w={48}
            color={bg}
            transform="translateX(50%)"
            fill="currentColor"
            viewBox="0 0 100 100"
            preserveAspectRatio="none"
            aria-hidden="true"
          >
            <polygon points="50,0 100,0 50,100 0,100" />
```

```
</Icon>
<Box
  mx="auto"
  maxW={{ base: "7xl" }}
  px={{ base: 4, sm: 6, lg: 8 }}
  mt={{ base: 8, sm: 10, md: 14, lg: 18, xl: 26 }}
>
  <Box
    w="full"
    textAlign={{ sm: "center", lg: "left" }}
    justifyContent="center"
    alignItems="center"
  >
    <chakra.h1
      fontSize={{ base: "4xl", sm: "5xl", md: "6xl" }}
      letterSpacing="tight"
      lineHeight="short"
      fontWeight="extrabold"
      color="gray.900"
      _dark={{ color: "white" }}
    >
      <chakra.span display={{ base: "block", xl: "inline" }}>
      Empowering Your Security, {" "}
      </chakra.span>
      <chakra.span
        display={{ base: "block", xl: "inline" }}
        color="brand.600"
        _dark={{ color: "brand.400" }}
      >
        One Flight at a Time.
      </chakra.span>
    </chakra.h1>
    <chakra.p
      mt={{ base: 3, sm: 5, md: 5 }}
      fontSize={{ sm: "lg", md: "xl" }}
```

```
    maxW={{ sm: "xl" }}
    mx={{ sm: "auto", lg: 0 }}
    color="gray.500"
  >
```
STAMP is a cutting-edge drone services company that specializes in surveillance, security, and locating individuals. Our user-friendly UI offers a range of features, including arming and setting up your drone, making it simple and efficient to manage your security needs.
```
  </chakra.p>
  <Box
    mt={{ base: 5, sm: 8 }}
    display={{ sm: "flex" }}
    justifyContent={{ sm: "center", lg: "start" }}
    fontWeight="extrabold"
    fontFamily="fantasy"
  >
    <Box rounded="full" shadow="md">
      <chakra.a
        w="full"
        display="flex"
        alignItems="center"
        justifyContent="center"
        border="solid 1px transparent"
        fontSize={{ base: "md", md: "lg" }}
        rounded="md"
        color="black"
        letterSpacing={ '3px' }
        bg="brand.600"
        _hover={{ color: '#0e76fd', }}
        px={{ base: 8, md: 10 }}
        py={{ base: 3, md: 4 }}
        cursor="pointer"
        href="http://127.0.0.1:8550/"
        target="_blank"
      >
        Get started
```

```
            </chakra.a>
          </Box>
          <Box mt={[3, 0]} ml={[null, 3]}>
            <chakra.a
              w="full"
              display="flex"
              alignItems="center"
              justifyContent="center"
              px={{ base: 8, md: 10 }}
              letterSpacing={ '3px' }
              py={{ base: 3, md: 4 }}
              border="solid 1px transparent"
              fontSize={{ base: "md", md: "lg" }}
              rounded="md"
              color="brand.700"
              bg="brand.100"
              _hover={{ bg: "brand.200" }}
              cursor="pointer"
              href="/Settings/Propellers"
              target="_blank"
            >
              Live demo
            </chakra.a>
          </Box>
        </Box>
      </Box>
    </Box>
  </Box>
</Box>
<Box
  position={{ lg: "absolute" }}
  top={{ lg: 0 }}
  bottom={{ lg: 0 }}
  right={{ lg: 0 }}
  w={{ lg: "50%" }}
```

```
          border="solid 1px transparent"
        >
          <Image
            h="84%"
            w="full"
            fit="cover"
            src={Dronesbg}
            alt=""
            loading="lazy"
          />
        </Box>
      </Box>
    );
};
```

Section2.js

```
import React from "react";
import { chakra, Box, Flex, Icon, Stack } from "@chakra-ui/react";

export default function App(){
  const Feature = (props) => {
    return (
      <Flex>
        <Flex shrink={0}>
          <Flex
            alignItems="center"
            justifyContent="center"
            h={12}
            w={12}
            rounded="md"
            _light={{ bg: "brand.500" }}
            color="white"
          >
            <Icon
              boxSize={6}
              fill="none"
              viewBox="0 0 24 24"
              stroke="currentColor"
              aria-hidden="true"
            >
              {props.icon}
            </Icon>
          </Flex>
        </Flex>
        <Box ml={4}>
          <chakra.dt
            fontSize="lg"
            fontWeight="medium"
            lineHeight="6"
```

```
          _light={{ color: "gray.900" }}
        >
          {props.title}
        </chakra.dt>
        <chakra.dd mt={2} color="gray.500" _dark={{ color: "gray.400" }}>
          {props.children}
        </chakra.dd>
      </Box>
    </Flex>
  );
};
return (
  <Flex
    bg="#edf3f8"
    _dark={{ bg: "#3e3e3e" }}
    p={20}
    w="auto"
    justifyContent="center"
    alignItems="center"
  >
    <Box py={12} bg="white" _dark={{ bg: "gray.800" }} rounded="xl">
      <Box maxW="7xl" mx="auto" px={{ base: 4, lg: 8 }}>
        <Box textAlign={{ lg: "center" }}>
          <chakra.p
            mt={2}
            fontSize={{ base: "3xl", sm: "4xl" }}
            lineHeight="8"
            fontWeight="extrabold"
            letterSpacing="tight"
            _light={{ color: "gray.900" }}
          >
           Advanced Drone Services by STAMP
          </chakra.p>
          <chakra.p
```

```
            mt={4}
            maxW="2xl"
            fontSize="xl"
            mx={{ lg: "auto" }}
            color="gray.500"
            _dark={{ color: "gray.400" }}
          >
            Revolutionizing Surveillance, Security, and Tracking
          </chakra.p>
        </Box>

        <Box mt={10}>
          <Stack
            spacing={{ base: 10, md: 0 }}
            display={{ md: "grid" }}
            gridTemplateColumns={{ md: "repeat(2,1fr)" }}
            gridColumnGap={{ md: 8 }}
            gridRowGap={{ md: 10 }}
          >
            <Feature
              title="Highly skilled personnel "
              icon={
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth="2"
                  d="M21 12a9 9 0 01-9 9m9-9a9 9 0 00-9-9m9 9H3m9 9a9 9
0 01-9-9m9 9c1.657 0 3-4.03 3-9s-1.343-9-3-9m0 18c-1.657 0-3-4.03-3-
9s1.343-9 3-9m-9 9a9 9 0 019-9"
                />
              }
            >
```

STAMP employs highly trained and experienced drone pilots and support staff to ensure the safe and effective operation of its drones. The company also provides regular training and development programs to keep its personnel up to date with the latest technologies and techniques.

```
</Feature>
<Feature
  title="Intuitive user interface "
  icon={
    <path
      strokeLinecap="round"
      strokeLinejoin="round"
      strokeWidth="2"
      d="M13 10V3L4 14h7v7l9-11h-7z"
    />
  }
>
```

STAMP's user interface (UI) is designed to be easy to use and navigate, even for those with little or no drone experience. The UI allows users to access a range of features, including arming and disarming their drone, setting up custom flight paths, and adjusting camera settings.

```
</Feature>
<Feature
  title=" Real-time tracking:"
  icon={
    <path
      strokeLinecap="round"
      strokeLinejoin="round"
      strokeWidth="2"
      d="M3 6l3 1m0 0l-3 9a5.002 5.002 0 006.001 0M6 7l3 9M6 7l6-2m6 2l3-1m-3 1l-3 9a5.002 5.002 0 006.001 0M18 7l3 9m-3-9l-6-2m0-2v2m0 16V5m0 16H9m3 0h3"
    />
  }
>
```

STAMP's advanced AI algorithms allow for real-time tracking of people and objects, making it an ideal solution for law enforcement agencies and private investigators.

```
            </Feature>



            <Feature
              title="Advanced drone technology"
              icon={
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth="2"
                  d="M7 8h10M7 12h4m1 8l-4-4H5a2 2 0 01-2-2V6a2 2 0
012-2h14a2 2 0 012 2v8a2 2 0 01-2 2h-3l-4 4z"
                />
              }
            >
```

STAMP uses cutting-edge drone technology to provide surveillance, security, and tracking solutions to its clients. The company's drones are equipped with high-quality cameras, advanced AI algorithms, and a range of tools and weapons.

```
            </Feature>
          </Stack>
        </Box>
      </Box>
    </Box>
  </Flex>
 );
};
```

Section4.js

```
import React from "react";

import {
  chakra,
  Box,
  Flex,
  Icon,
  SimpleGrid,
  Stack,
  GridItem,
} from "@chakra-ui/react";
export default function App(){
  const Feature = (props) => {
    return (
      <Flex>
        <Flex shrink={0}>
          <Icon
            boxSize={5}
            mt={1}
            mr={2}
            color="brand.500"
            _dark={{ color: "brand.300" }}
            viewBox="0 0 20 20"
            fill="currentColor"
          >
            <path
              fillRule="evenodd"
              d="M16.707 5.293a1 1 0 010 1.414l-8 8a1 1 0 01-1.414 0l-4-4a1 1 0 011.414-1.414L8
12.586l7.293-7.293a1 1 0 011.414 0z"
              clipRule="evenodd"
            ></path>
          </Icon>
        </Flex>
        <Box ml={4}>
          <chakra.dt
            fontSize="lg"
            fontWeight="bold"
            lineHeight="6"
            _light={{ color: "gray.900" }}
          >
            {props.title}
          </chakra.dt>
```

```
        <chakra.dd mt={2} color="gray.500" _dark={{ color: "gray.400" }}>
          {props.children}

        </chakra.dd>
      </Box>
    </Flex>
  );
};
return (
  <Flex
    bg="#edf3f8"
    _dark={{ bg: "#3e3e3e" }}
    p={20}
    w="auto"
    justifyContent="center"
    alignItems="center"
  >
    <Box
      shadow="xl"
      bg="white"
      _dark={{ bg: "gray.800" }}
      px={8}
      py={20}
      mx="auto"
      rounded = "xl"
    >
      <SimpleGrid
        alignItems="center"
        columns={{ base: 1, lg: 3 }}
        spacingY={{ base: 10, lg: 32 }}
        spacingX={{ base: 10, lg: 24 }}
      >
        <Box alignSelf="start">
          <chakra.h2
            _light={{ color: "brand.500" }}
            fontWeight="semibold"
            textTransform="uppercase"
            letterSpacing="wide"
          >
            Everything you need
          </chakra.h2>
          <chakra.h2
            mb={3}
            fontSize={{ base: "3xl", md: "4xl" }}
            fontWeight="extrabold"
            textAlign={{ base: "center", sm: "left" }}
            _light={{ color: "black" }}
```

```
            lineHeight="shorter"
            letterSpacing="tight"
          >
            All-in-one platform
          </chakra.h2>
          <chakra.p
            mb={6}
            fontSize={{ base: "lg", md: "xl" }}
            textAlign={{ base: "center", sm: "left" }}
            color="gray.600"
            _dark={{ color: "gray.500" }}
          >
            STAMP is a forward-thinking drone services company that is revolutionizing the way we think about surveillance, security, and tracking. With its cutting-edge technology and highly skilled personnel, the company is well-positioned to become a leader in the industry.
          </chakra.p>
        </Box>
        <GridItem colSpan={2}>
          <Stack
            spacing={{ base: 10, md: 0 }}
            display={{ md: "grid" }}
            gridTemplateColumns={{ md: "repeat(2,1fr)" }}
            gridColumnGap={{ md: 8 }}
            gridRowGap={{ md: 10 }}
          >
            <Feature title="Surveillance">
            24/7 monitoring with high-quality cameras for large premises.            </Feature>
            <Feature title="Security">
            Armed drones with tools and weapons for protecting high-value assets.            </Feature>
            <Feature title="Tracking">
              {" "}
              Advanced AI algorithms for real-time tracking of people and objects.            </Feature>
            <Feature title="User interface (UI)">
              {" "}
              Intuitive and easy-to-use UI for drone control and customization.            </Feature>
            <Feature title="Customization">
              {" "}
              Customization options for drones to meet specific client needs.            </Feature>
            <Feature title="Expertise">
              {" "}
              Highly skilled personnel for safety and security, including experienced drone pilots and security professionals.            </Feature>
          </Stack>
        </GridItem>
```

```
        </SimpleGrid>
      </Box>
    </Flex>
  );
};
```

About.js

```
import {
  Heading,
  Box,
  Text,
  CardBody,
  Image,
  Stack,
  Card,
  CardFooter,
  Button,
} from '@chakra-ui/react';
import images from '../assets/images.jpg';
import mihir from '../assets/mihir.jpg';
import prinkal from '../assets/prinkal.jpg';
import arsh from '../assets/arsh.jpg';
import tanay from '../assets/tanay.jpg';
import sarid from '../assets/sarid.jpg';
import React from 'react';
import '../components/AboutUs/about.css';
import ContactUs from '../components/AboutUs/ContactUs';
import TeamMembers from '../components/AboutUs/TeamMembers';
import Section1 from '../components/AboutUs/Section1';
// import ContactUs from '../components/AboutUs/ContactUs';

const About = () => {
  return (
    <div>

      <Section1></Section1>
       <div id="team">
          <TeamMembers></TeamMembers>
       </div>
       <div id="contact">
          <ContactUs></ContactUs>
       </div>
```

```
    </div>

  );
};

export default About;
```

TeamMembers.js

```
import React from 'react';
import { Flex, Spacer, Text, useMediaQuery, Icon, Image, useColorMode } from '@chakra-
ui/react';
import { FaTools, FaHandshake, FaStar } from 'react-icons/fa';
import mihir from '../../assets/mihir.jpg';
import prinkal from '../../assets/prinkal.jpg';
import arsh from '../../assets/arsh.jpg';
import tanay from '../../assets/tanay.jpg';
import sarid from '../../assets/sarid.jpg';

const AboutUs = () => {
  const [isLargerThan48] = useMediaQuery('(min-width: 48em)');
  const { colorMode, toggleColorMode } = useColorMode();


  const array = [
    {
      id: 1,
      text: 'Mihir Panchal',
      image: mihir,
      subheading: 'Full Stack Developer',
    },
    {
      id: 2,
      text: 'Prinkal Doshi',
      image: prinkal,
      subheading: 'Web Developer',

    },
    {
      id: 3,
      text: 'Tanay Desai',
      image: tanay,
      subheading: 'Web Developer',
    },
```

```
  {
    id: 4,
    text: 'Sarid Qureshi',
    image: sarid,
    subheading: 'Backend Developer',
  },
  {
    id: 5,
    text: 'Arsh Sakaria',
    image: arsh,
    subheading: 'Drone Pilot',
  },
];

return (
  <>
    <center><Text
      mb={2}
      fontSize="5xl"
      fontWeight="bold"
      lineHeight="tight"
      bgGradient="linear(to-r, brand.300, brand.600)"
      bgClip="text"
      color={colorMode === 'light' ? 'black' : 'white'}
    >
      Team Members
    </Text></center>
    <Flex
      minH="0vh"
      alignItems="center"
      justifyContent="space-between"
      w="full"
      py="16"
      px={isLargerThan48 ? '20' : '0'}
      flexWrap="wrap"
      flexDirection={isLargerThan48 ? 'row' : 'column'}
    >
      {array.map((arr) => (
        <>
          <Flex
            height="200px"
            bg="blackAlpha.200"
            width={isLargerThan48 ? '18%' : 'full'}
            shadow="md"
            p="6"
```

```
            alignItems="center"
            justifyContent="center"
            borderRadius="md"
            flexDirection="column"
            textAlign="center"
            mb={isLargerThan48 ? '0' : '4'}
            border="1px solid #C4DDFF"
          >

            <Image src={arr.image} borderRadius="full" boxSize="60%" objectFit="cover" />
            <Text fontSize="xl" fontWeight="semibold" mt="4">{arr.text}</Text>
            <Text fontSize="sm" color="gray.600">{arr.subheading}</Text>
          </Flex>

          <Spacer />
        </>
      ))}
    </Flex>
  </>
 );
};

export default AboutUs;
```

ContactUs.js

```
import {
  Box,
  Button,
  Flex,
  FormControl,
  FormLabel,
  Heading,
  IconButton,
  Input,
  InputGroup,
  InputLeftElement,
  Link,
  Stack,
  Textarea,
  Tooltip,
  useClipboard,
  useColorModeValue,
  useToast,
  VStack,
} from '@chakra-ui/react';
import React from 'react';
import { useState } from 'react'
// import { BsGithub, BsLinkedin, BsPerson, BsTwitter } from 'react-icons/bs';
// import { MdEmail, MdOutlineEmail } from 'react-icons/md';

const confetti = {
  light: {
    primary: '4299E1', // blue.400
    secondary: 'BEE3F8', // blue.100
  },

  dark: {
    primary: '1A365D', // blue.900
    secondary: '2A4365', // blue.800
  },
};
```

const CONFETTI_LIGHT = `url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='1490' height='745' viewBox='0 0 1600 800'%3E%3Cpath fill='%23${confetti.light.primary}' d='M1102.5 734.8c2.5-1.2 24.8-8.6 25.6-7.5.5.7-3.9 23.8-4.6 24.5C1123.3 752.1 1107.5 739.5 1102.5 734.8zM1226.3 229.1c0-.1-4.9-9.4-7-14.2-.1-.3-.3-1.1-.4-1.6-.1-.4-.3-.7-.6-.9-.3-.2-.6-.1-.8.1l-13.1 12.3c0 0 0 0 0 0-.2.2-.3.5-.4.8 0 .3 0 .7.2 1 .1.1 1.4 2.5 2.1 3.6 2.4 3.7 6.5 12.1 6.5 12.2.2.3.4.5.7.6.3 0 .5-.1.7-.3 0 0 1.8-2.5 2.7-3.6 1.5-1.6 3-3.2 4.6-4.7 1.2-1.2 1.6-1.4 2.1-1.6.5-.3 1.1-.5 2.5-1.9C1226.5 230.4 1226.6 229.6 1226.3 229.1zM33 770.3C33 770.3 33 770.3 33 770.3c0-.7-.5-1.2-1.2-1.2-.1 0-.3 0-.4.1-1.6.2-14.3.1-22.2 0-.3 0-.6.1-.9.4-.2.2-.4.5-.4.9 0 .2 0 4.9.1 5.9l.4 13.6c0 .3.2.6.4.9.2.2.5.3.8.3 0 0 .1 0 .1 0 7.3-.7 14.7-.9 22-.6.3 0 .7-.1.9-.3.2-.2.4-.6.4-.9C32.9 783.3 32.9 776.2 33 770.3z'/%3E%3Cpath fill='%23${confetti.light.secondary}' d='M171.1 383.4c1.3-2.5 14.3-22 15.6-21.6.8.3 11.5 21.2 11.5 22.1C198.1 384.2 177.9 384 171.1 383.4zM596.4 711.8c-.1-.1-6.7-8.2-9.7-12.5-.2-.3-.5-1-.7-1.5-.2-.4-.4-.7-.7-.8-.3-.1-.6 0-.8.3L574 712c0 0 0 0 0 0-.2.2-.2.5-.2.9 0 .3.2.7.4.9.1.1 1.8 2.2 2.8 3.1 3.1 3.1 8.8 10.5 8.9 10.6.2.3.5.4.8.4.3 0 .5-.2.6-.5 0 0 1.2-2.8 2-4.1 1.1-1.9 2.3-3.7 3.5-5.5.9-1.4 1.3-1.7 1.7-2 .5-.4 1-.7 2.1-2.4C596.9 713.1 596.8 712.3 596.4 711.8zM727.5 179.9C727.5 179.9 727.5 179.9 727.5 179.9c.6.2 1.3-.2 1.4-.8 0-.1 0-.2 0-.4.2-1.4 2.8-12.6 4.5-19.5.1-.3 0-.6-.2-.8-.2-.3-.5-.4-.8-.5-.2 0-4.7-1.1-5.7-1.3l-13.4-2.7c-.3-.1-.7 0-.9.2-.2.2-.4.4-.5.6 0 0 0 .1 0 .1-.8 6.5-2.2 13.1-3.9 19.4-.1.3 0 .6.2.9.2.3.5.4.8.5C714.8 176.9 721.7 178.5 727.5 179.9zM728.5 178.1c-.1-.1-.2-.2-.4-.2C728.3 177.9 728.4 178 728.5 178.1z'/%3E%3Cg fill-opacity='0.48' fill='%23FFF'%3E%3Cpath d='M699.6 472.7c-1.5 0-2.8-.8-3.5-2.3-.8-1.9 0-4.2 1.9-5 3.7-1.6 6.8-4.7 8.4-8.5 1.6-3.8 1.7-8.1.2-11.9-.3-.9-.8-1.8-1.2-2.8-.8-1.7-1.8-3.7-2.3-5.9-.9-4.1-.2-8.6 2-12.8 1.7-3.1 4.1-6.1 7.6-9.1 1.6-1.4 4-1.2 5.3.4 1.4 1.6 1.2 4-.4 5.3-2.8 2.5-4.7 4.7-5.9 7-1.4 2.6-1.9 5.3-1.3 7.6.3 1.4 1 2.8 1.7 4.3.5 1.1 1 2.2 1.5 3.3 2.1 5.6 2 12-.3 17.6-2.3 5.5-6.8 10.1-12.3 12.5C700.6 472.6 700.1 472.7 699.6 472.7zM740.4 421.4c1.5-.2 3 .5 3.8 1.9 1.1 1.8.4 4.2-1.4 5.3-3.7 2.1-6.4 5.6-7.6 9.5-1.2 4-.8 8.4 1.1 12.1.4.9 1 1.7 1.6 2.7 1 1.7 2.2 3.5 3 5.7 1.4 4 1.2 8.7-.6 13.2-1.4 3.4-3.5 6.6-6.8 10.1-1.5 1.6-3.9 1.7-5.5.2-1.6-1.4-1.7-3.9-.2-5.4 2.6-2.8 4.3-5.3 5.3-7.7 1.1-2.8 1.3-5.6.5-7.9-.5-1.3-1.3-2.7-2.2-4.1-.6-1-1.3-2.1-1.9-3.2-2.8-5.4-3.4-11.9-1.7-17.8 1.8-5.9 5.8-11 11.2-14C739.4 421.6 739.9 421.4 740.4 421.4zM261.3 590.9c5.7 6.8 9 15.7 9.4 22.4.5 7.3-2.4 16.4-10.2 20.4-3 1.5-6.7 2.2-11.2 2.2-7.9-.1-12.9-2.9-15.4-8.4-2.1-4.7-2.3-11.4 1.8-15.9 3.2-3.5 7.8-4.1 11.2-1.6 1.2.9 1.5 2.7.6 3.9-.9 1.2-2.7 1.5-3.9.6-1.8-1.3-3.6.6-3.8.8-2.4 2.6-2.1 7-.8 9.9 1.5 3.4 4.7 5 10.4 5.1 3.6 0 6.4-.5 8.6-1.6 4.7-2.4 7.7-8.6 7.2-15-.5-7.3-5.3-18.2-13-23.9-4.2-3.1-8.5-4.1-12.9-3.1-3.1.7-6.2 2.4-9.7 5-6.6 5.1-11.7 11.8-14.2 19-2.7 7.7-2.1 15.8 1.9 23.9.7 1.4.1 3.1-1.3 3.7-1.4.7-3.1.1-3.7-1.3-4.6-9.4-5.4-19.2-2.2-28.2 2.9-8.2 8.6-15.9 16.1-21.6 4.1-3.1 8-5.1 11.8-6 6-1.4 12 0 17.5 4C257.6 586.9 259.6 588.8 261.3 590.9z'/%3E%3Ccircle cx='1013.7' cy='153.9' r='7.1'/%3E%3Ccircle cx='1024.3' cy='132.1' r='7.1'/%3E%3Ccircle cx='1037.3' cy='148.9' r='7.1'/%3E%3Cpath d='M1508.7 297.2c-4.8-5.4-9.7-10.8-14.8-16.2 5.6-5.6 11.1-11.5 15.6-18.2 1.2-1.7.7-4.1-1-5.2-1.7-1.2-4.1-.7-5.2 1-4.2 6.2-9.1 11.6-14.5 16.9-4.8-5-9.7-10-14.7-14.9-1.5-1.5-3.9-1.5-5.3 0-1.5 1.5-1.5 3.9 0 5.3 4.9 4.8 9.7 9.8 14.5 14.8-1.1 1.1-2.3 2.2-3.5 3.2-4.1 3.8-8.4 7.8-12.4 12-1.4 1.5-1.4 3.8 0 5.3 0 0 0 0 0 0 1.5 1.4 3.9 1.4 5.3-.1 3.9-4 8.1-7.9 12.1-11.7 1.2-1.1 2.3-2.2 3.5-3.3 4.9 5.3 9.8 10.6 14.6 15.9.1.1.1.1.2.2 1.4 1.4 3.7 1.5 5.2.2C1510 301.2 1510.1 298.8 1508.7 297.2zM327.6 248.6l-.4-2.6c-1.5-11.1-2.2-23.2-2.3-37 0-5.5 0-11.5.2-18.5 0-.7 0-1.5 0-2.3 0-5 0-11.2 3.9-13.5 2.2-1.3 5.1-1 8.5.9 5.7 3.1 13.2 8.7 17.5 14.9 5.5 7.8 7.3 16.9 5 25.7-3.2 12.3-15 31-30 32.1L327.6 248.6zM332.1 179.2c-.2 0-.3 0-.4.1-.1.1-.7.5-1.1 2.7-.3 1.9-.3 4.2-.3 6.3 0 .8 0 1.7 0 2.4-.2 6.9-.2 12.8-.2 18.3.1 12.5.7 23.5 2 33.7 11-2.7 20.4-18.1 23-27.8 1.9-7.2.4-14.8-4.2-21.3l0

0C347 188.1 340 183 335 180.3 333.6 179.5 332.6 179.2 332.1 179.2zM516.3 60.8c-.1 0-.2 0-.4-.1-2.4-.7-4-.9-6.7-.7-.7 0-1.3-.5-1.4-1.2 0-.7.5-1.3 1.2-1.4 3.1-.2 4.9 0 7.6.8.7.2 1.1.9.9 1.6C517.3 60.4 516.8 60.8 516.3 60.8zM506.1 70.5c-.5 0-1-.3-1.2-.8-.8-2.1-1.2-4.3-1.3-6.6 0-.7.5-1.3 1.2-1.3.7 0 1.3.5 1.3 1.2.1 2 .5 3.9 1.1 5.8.2.7-.1 1.4-.8 1.6C506.4 70.5 506.2 70.5 506.1 70.5zM494.1 64.4c-.4 0-.8-.2-1-.5-.4-.6-.3-1.4.2-1.8 1.8-1.4 3.7-2.6 5.8-3.6.6-.3 1.4 0 1.7.6.3.6 0 1.4-.6 1.7-1.9.9-3.7 2-5.3 3.3C494.7 64.3 494.4 64.4 494.1 64.4zM500.5 55.3c-.5 0-.9-.3-1.2-.7-.5-1-1.2-1.9-2.4-3.4-.3-.4-.7-.9-1.1-1.4-.4-.6-.3-1.4.2-1.8.6-.4 1.4-.3 1.8.2.4.5.8 1 1.1 1.4 1.3 1.6 2.1 2.6 2.7 3.9.3.6 0 1.4-.6 1.7C500.9 55.3 500.7 55.3 500.5 55.3zM506.7 55c-.3 0-.5-.1-.8-.2-.6-.4-.7-1.2-.3-1.8 1.2-1.7 2.3-3.4 3.3-5.2.3-.6 1.1-.9 1.7-.5.6.3.9 1.1.5 1.7-1 1.9-2.2 3.8-3.5 5.6C507.4 54.8 507.1 55 506.7 55zM1029.3 382.8c-.1 0-.2 0-.4-.1-2.4-.7-4-.9-6.7-.7-.7 0-1.3-.5-1.4-1.2 0-.7.5-1.3 1.2-1.4 3.1-.2 4.9 0 7.6.8.7.2 1.1.9.9 1.6C1030.3 382.4 1029.8 382.8 1029.3 382.8zM1019.1 392.5c-.5 0-1-.3-1.2-.8-.8-2.1-1.2-4.3-1.3-6.6 0-.7.5-1.3 1.2-1.3.7 0 1.3.5 1.3 1.2.1 2 .5 3.9 1.1 5.8.2.7-.1 1.4-.8 1.6C1019.4 392.5 1019.2 392.5 1019.1 392.5zM1007.1 386.4c-.4 0-.8-.2-1-.5-.4-.6-.3-1.4.2-1.8 1.8-1.4 3.7-2.6 5.8-3.6.6-.3 1.4 0 1.7.6.3.6 0 1.4-.6 1.7-1.9.9-3.7 2-5.3 3.3C1007.7 386.3 1007.4 386.4 1007.1 386.4zM1013.5 377.3c-.5 0-.9-.3-1.2-.7-.5-1-1.2-1.9-2.4-3.4-.3-.4-.7-.9-1.1-1.4-.4-.6-.3-1.4.2-1.8.6-.4 1.4-.3 1.8.2.4.5.8 1 1.1 1.4 1.3 1.6 2.1 2.6 2.7 3.9.3.6 0 1.4-.6 1.7C1013.9 377.3 1013.7 377.3 1013.5 377.3zM1019.7 377c-.3 0-.5-.1-.8-.2-.6-.4-.7-1.2-.3-1.8 1.2-1.7 2.3-3.4 3.3-5.2.3-.6 1.1-.9 1.7-.5.6.3.9 1.1.5 1.7-1 1.9-2.2 3.8-3.5 5.6C1020.4 376.8 1020.1 377 1019.7 377zM1329.7 573.4c-1.4 0-2.9-.2-4.5-.7-8.4-2.7-16.6-12.7-18.7-20-.4-1.4-.7-2.9-.9-4.4-8.1 3.3-15.5 10.6-15.4 21 0 1.5-1.2 2.7-2.7 2.8 0 0 0 0 0 0-1.5 0-2.7-1.2-2.7-2.7-.1-6.7 2.4-12.9 7-18 3.6-4 8.4-7.1 13.7-8.8.5-6.5 3.1-12.9 7.4-17.4 7-7.4 18.2-8.9 27.3-10.1l.7-.1c1.5-.2 2.9.9 3.1 2.3.2 1.5-.9 2.9-2.3 3.1l-.7.1c-8.6 1.2-18.4 2.5-24 8.4-3 3.2-5 7.7-5.7 12.4 7.9-1 17.7 1.3 24.3 5.7 4.3 2.9 7.1 7.8 7.2 12.7.2 4.3-1.7 8.3-5.2 11.1C1335.2 572.4 1332.6 573.4 1329.7 573.4zM1311 546.7c.1 1.5.4 3 .8 4.4 1.7 5.8 8.7 14.2 15.1 16.3 2.8.9 5.1.5 7.2-1.1 2.7-2.1 3.2-4.8 3.1-6.6-.1-3.2-2-6.4-4.8-8.3C1326.7 547.5 1317.7 545.6 1311 546.7z'/%3E%3C/g%3E%3C/svg%3E")`;
const CONFETTI_DARK = `url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='1490' height='745' viewBox='0 0 1600 800'%3E%3Cpath fill='%23${confetti.dark.primary}' d='M1102.5 734.8c2.5-1.2 24.8-8.6 25.6-7.5.5.7-3.9 23.8-4.6 24.5C1123.3 752.1 1107.5 739.5 1102.5 734.8zM1226.3 229.1c0-.1-4.9-9.4-7-14.2-.1-.3-.3-1.1-.4-1.6-.1-.4-.3-.7-.6-.9-.3-.2-.6-.1-.8.1l-13.1 12.3c0 0 0 0 0 0-.2.2-.3.5-.4.8 0 .3 0 .7.2 1 .1.1 1.4 2.5 2.1 3.6 2.4 3.7 6.5 12.1 6.5 12.2.2.3.4.5.7.6.3 0 .5-.1.7-.3 0 0 1.8-2.5 2.7-3.6 1.5-1.6 3-3.2 4.6-4.7 1.2-1.2 1.6-1.4 2.1-1.6.5-.3 1.1-.5 2.5-1.9C1226.5 230.4 1226.6 229.6 1226.3 229.1zM33 770.3C33 770.3 33 770.3 33 770.3c0-.7-.5-1.2-1.2-1.2-.1 0-.3 0-.4.1-1.6.2-14.3.1-22.2 0-.3 0-.6.1-.9.4-.2.2-.4.5-.4.9 0 .2 0 4.9.1 5.9l.4 13.6c0 .3.2.6.4.9.2.2.5.3.8.3 0 0 .1 0 .1 0 7.3-.7 14.7-.9 22-.6.3 0 .7-.1.9-.3.2-.2.4-.6.4-.9C32.9 783.3 32.9 776.2 33 770.3z'/%3E%3Cpath fill='%23${confetti.dark.secondary}' d='M171.1 383.4c1.3-2.5 14.3-22 15.6-21.6.8.3 11.5 21.2 11.5 22.1C198.1 384.2 177.9 384 171.1 383.4zM596.4 711.8c-.1-.1-6.7-8.2-9.7-12.5-.2-.3-.5-1-.7-1.5-.2-.4-.4-.7-.7-.8-.3-.1-.6 0-.8.3L574 712c0 0 0 0 0 0-.2.2-.2.5-.2.9 0 .3.2.7.4.9.1.1 1.8 2.2 2.8 3.1 3.1 3.1 3.1 8.8 10.5 8.9 10.6.2.3.5.4.8.4.3 0 .5-.2.6-.5 0 0 1.2-2.8 2-4.1 1.1-1.9 2.3-3.7 3.5-5.5.9-1.4 1.3-1.7 1.7-2 .5-.4 1-.7 2.1-2.4C596.9 713.1 596.8 712.3 596.4 711.8zM727.5 179.9C727.5 179.9 727.5 179.9 727.5 179.9c.6.2 1.3-.2 1.4-.8 0-.1 0-.2 0-.4.2-1.4 2.8-12.6 4.5-19.5.1-.3 0-.6-.2-.8-.2-.3-.5-.4-.8-.5-.2 0-4.7-1.1-5.7-1.3l-13.4-2.7c-.3-.1-.7 0-.9.2-.2.2-.4.4-.5.6 0 0 0 .1 0 .1-.8 6.5-2.2 13.1-3.9 19.4-.1.3 0 .6.2.9.2.3.5.4.8.5C714.8 176.9 721.7 178.5 727.5 179.9zM728.5 178.1c-.1-.1-.2-.2-.4-.2C728.3 177.9 728.4 178 728.5 178.1z'/%3E%3Cg fill-

opacity='0.05' fill='%23FFF'%3E%3Cpath d='M699.6 472.7c-1.5 0-2.8-.8-3.5-2.3-.8-1.9 0-4.2 1.9-5 3.7-1.6 6.8-4.7 8.4-8.5 1.6-3.8 1.7-8.1.2-11.9-.3-.9-.8-1.8-1.2-2.8-.8-1.7-1.8-3.7-2.3-5.9-.9-4.1-.2-8.6 2-12.8 1.7-3.1 4.1-6.1 7.6-9.1 1.6-1.4 4-1.2 5.3.4 1.4 1.6 1.2 4-.4 5.3-2.8 2.5-4.7 4.7-5.9 7-1.4 2.6-1.9 5.3-1.3 7.6.3 1.4 1 2.8 1.7 4.3.5 1.1 1 2.2 1.5 3.3 2.1 5.6 2 12-.3 17.6-2.3 5.5-6.8 10.1-12.3 12.5C700.6 472.6 700.1 472.7 699.6 472.7zM740.4 421.4c1.5-.2 3 .5 3.8 1.9 1.1 1.8.4 4.2-1.4 5.3-3.7 2.1-6.4 5.6-7.6 9.5-1.2 4-.8 8.4 1.1 12.1.4.9 1 1.7 1.6 2.7 1 1.7 2.2 3.5 3 5.7 1.4 4 1.2 8.7-.6 13.2-1.4 3.4-3.5 6.6-6.8 10.1-1.5 1.6-3.9 1.7-5.5.2-1.6-1.4-1.7-3.9-.2-5.4 2.6-2.8 4.3-5.3 5.3-7.7 1.1-2.8 1.3-5.6.5-7.9-.5-1.3-1.3-2.7-2.2-4.1-.6-1-1.3-2.1-1.9-3.2-2.8-5.4-3.4-11.9-1.7-17.8 1.8-5.9 5.8-11 11.2-14C739.4 421.6 739.9 421.4 740.4 421.4zM261.3 590.9c5.7 6.8 9 15.7 9.4 22.4.5 7.3-2.4 16.4-10.2 20.4-3 1.5-6.7 2.2-11.2 2.2-7.9-.1-12.9-2.9-15.4-8.4-2.1-4.7-2.3-11.4 1.8-15.9 3.2-3.5 7.8-4.1 11.2-1.6 1.2.9 1.5 2.7.6 3.9-.9 1.2-2.7 1.5-3.9.6-1.8-1.3-3.6.6-3.8.8-2.4 2.6-2.1 7-.8 9.9 1.5 3.4 4.7 5 10.4 5.1 3.6 0 6.4-.5 8.6-1.6 4.7-2.4 7.7-8.6 7.2-15-.5-7.3-5.3-18.2-13-23.9-4.2-3.1-8.5-4.1-12.9-3.1-3.1.7-6.2 2.4-9.7 5-6.6 5.1-11.7 11.8-14.2 19-2.7 7.7-2.1 15.8 1.9 23.9.7 1.4.1 3.1-1.3 3.7-1.4.7-3.1.1-3.7-1.3-4.6-9.4-5.4-19.2-2.2-28.2 2.9-8.2 8.6-15.9 16.1-21.6 4.1-3.1 8-5.1 11.8-6 6-1.4 12 0 17.5 4C257.6 586.9 259.6 588.8 261.3 590.9z'/%3E%3Ccircle cx='1013.7' cy='153.9' r='7.1'/%3E%3Ccircle cx='1024.3' cy='132.1' r='7.1'/%3E%3Ccircle cx='1037.3' cy='148.9' r='7.1'/%3E%3Cpath d='M1508.7 297.2c-4.8-5.4-9.7-10.8-14.8-16.2 5.6-5.6 11.1-11.5 15.6-18.2 1.2-1.7.7-4.1-1-5.2-1.7-1.2-4.1-.7-5.2 1-4.2 6.2-9.1 11.6-14.5 16.9-4.8-5-9.7-10-14.7-14.9-1.5-1.5-3.9-1.5-5.3 0-1.5 1.5-1.5 3.9 0 5.3 4.9 4.8 9.7 9.8 14.5 14.8-1.1 1.1-2.3 2.2-3.5 3.2-4.1 3.8-8.4 7.8-12.4 12-1.4 1.5-1.4 3.8 0 5.3 0 0 0 0 0 0 1.5 1.4 3.9 1.4 5.3-.1 3.9-4 8.1-7.9 12.1-11.7 1.2-1.1 2.3-2.2 3.5-3.3 4.9 5.3 9.8 10.6 14.6 15.9.1.1.1.1.2.2 1.4 1.4 3.7 1.5 5.2.2C1510 301.2 1510.1 298.8 1508.7 297.2zM327.6 248.6l-.4-2.6c-1.5-11.1-2.2-23.2-2.3-37 0-5.5 0-11.5.2-18.5 0-.7 0-1.5 0-2.3 0-5 0-11.2 3.9-13.5 2.2-1.3 5.1-1 8.5.9 5.7 3.1 13.2 8.7 17.5 14.9 5.5 7.8 7.3 16.9 5 25.7-3.2 12.3-15 31-30 32.1L327.6 248.6zM332.1 179.2c-.2 0-.3 0-.4.1-.1.1-.7.5-1.1 2.7-.3 1.9-.3 4.2-.3 6.3 0 .8 0 1.7 0 2.4-.2 6.9-.2 12.8-.2 18.3.1 12.5.7 23.5 2 33.7 11-2.7 20.4-18.1 23-27.8 1.9-7.2.4-14.8-4.2-21.3l0 0C347 188.1 340 183 335 180.3 333.6 179.5 332.6 179.2 332.1 179.2zM516.3 60.8c-.1 0-.2 0-.4-.1-2.4-.7-4-.9-6.7-.7-.7 0-1.3-.5-1.4-1.2 0-.7.5-1.3 1.2-1.4 3.1-.2 4.9 0 7.6.8.7.2 1.1.9.9 1.6C517.3 60.4 516.8 60.8 516.3 60.8zM506.1 70.5c-.5 0-1-.3-1.2-.8-.8-2.1-1.2-4.3-1.3-6.6 0-.7.5-1.3 1.2-1.3.7 0 1.3.5 1.3 1.2.1 2 .5 3.9 1.1 5.8.2.7-.1 1.4-.8 1.6C506.4 70.5 506.2 70.5 506.1 70.5zM494.1 64.4c-.4 0-.8-.2-1-.5-.4-.6-.3-1.4.2-1.8 1.8-1.4 3.7-2.6 5.8-3.6.6-.3 1.4 0 1.7.6.3.6 0 1.4-.6 1.7-1.9.9-3.7 2-5.3 3.3C494.7 64.3 494.4 64.4 494.1 64.4zM500.5 55.3c-.5 0-.9-.3-1.2-.7-.5-1-1.2-1.9-2.4-3.4-.3-.4-.7-.9-1.1-1.4-.4-.6-.3-1.4.2-1.8.6-.4 1.4-.3 1.8.2.4.5.8 1 1.1 1.4 1.3 1.6 2.1 2.6 2.7 3.9.3.6 0 1.4-.6 1.7C500.9 55.3 500.7 55.3 500.5 55.3zM506.7 55c-.3 0-.5-.1-.8-.2-.6-.4-.7-1.2-.3-1.8 1.2-1.7 2.3-3.4 3.3-5.2.3-.6 1.1-.9 1.7-.5.6.3.9 1.1.5 1.7-1 1.9-2.2 3.8-3.5 5.6C507.4 54.8 507.1 55 506.7 55zM1029.3 382.8c-.1 0-.2 0-.4-.1-2.4-.7-4-.9-6.7-.7-.7 0-1.3-.5-1.4-1.2 0-.7.5-1.3 1.2-1.4 3.1-.2 4.9 0 7.6.8.7.2 1.1.9.9 1.6C1030.3 382.4 1029.8 382.8 1029.3 382.8zM1019.1 392.5c-.5 0-1-.3-1.2-.8-.8-2.1-1.2-4.3-1.3-6.6 0-.7.5-1.3 1.2-1.3.7 0 1.3.5 1.3 1.2.1 2 .5 3.9 1.1 5.8.2.7-.1 1.4-.8 1.6C1019.4 392.5 1019.2 392.5 1019.1 392.5zM1007.1 386.4c-.4 0-.8-.2-1-.5-.4-.6-.3-1.4.2-1.8 1.8-1.4 3.7-2.6 5.8-3.6.6-.3 1.4 0 1.7.6.3.6 0 1.4-.6 1.7-1.9.9-3.7 2-5.3 3.3C1007.7 386.3 1007.4 386.4 1007.1 386.4zM1013.5 377.3c-.5 0-.9-.3-1.2-.7-.5-1-1.2-1.9-2.4-3.4-.3-.4-.7-.9-1.1-1.4-.4-.6-.3-1.4.2-1.8.6-.4 1.4-.3 1.8.2.4.5.8 1 1.1 1.4 1.3 1.6 2.1 2.6 2.7 3.9.3.6 0 1.4-.6 1.7C1013.9 377.3 1013.7 377.3 1013.5 377.3zM1019.7 377c-.3 0-.5-.1-.8-.2-.6-.4-.7-1.2-.3-1.8 1.2-1.7 2.3-3.4 3.3-5.2.3-.6 1.1-.9 1.7-.5.6.3.9 1.1.5 1.7-1 1.9-2.2 3.8-3.5 5.6C1020.4 376.8 1020.1 377 1019.7 377zM1329.7 573.4c-1.4 0-2.9-.2-4.5-.7-8.4-2.7-16.6-12.7-18.7-20-.4-1.4-.7-2.9-.9-4.4-8.1 3.3-15.5 10.6-15.4 21 0 1.5-1.2

```
2.7-2.7 2.8 0 0 0 0 0 0-1.5 0-2.7-1.2-2.7-2.7-.1-6.7 2.4-12.9 7-18 3.6-4 8.4-7.1 13.7-8.8.5-6.5 3.1-12.9 7.4-17.4 7-7.4 18.2-8.9 27.3-10.1l.7-.1c1.5-.2 2.9.9 3.1 2.3.2 1.5-.9 2.9-2.3 3.1l-.7.1c-8.6 1.2-18.4 2.5-24 8.4-3 3.2-5 7.7-5.7 12.4 7.9-1 17.7 1.3 24.3 5.7 4.3 2.9 7.1 7.8 7.2 12.7.2 4.3-1.7 8.3-5.2 11.1C1335.2 572.4 1332.6 573.4 1329.7 573.4zM1311 546.7c.1 1.5.4 3 .8 4.4 1.7 5.8 8.7 14.2 15.1 16.3 2.8.9 5.1.5 7.2-1.1 2.7-2.1 3.2-4.8 3.1-6.6-.1-3.2-2-6.4-4.8-8.3C1326.7 547.5 1317.7 545.6 1311 546.7z'/%3E%3C/g%3E%3C/svg%3E")`;


export default function ContactUs() {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [message, setMessage] = useState('');
  const toast = useToast();

  const handleContactUsSubmit = () =>{
    if(name && message && email) {
      fetch("http://127.0.0.1:5000/contact_us", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          name,
          email,
          message
        })
    })
    .then(response => {
        if (response.ok) {
          console.log("details sent");
          toast({
              title: `details sent successfully`,
              status: "success",
              duration: 3000,
              isClosable: true,
            });
        } else {
        console.log("Error while sending details");
        }
    })
    .catch(error => {
        console.log(error);
    });
  }
```

```
}
 return (
   <Flex
     bg={useColorModeValue('gray.100', 'gray.900')}
     align="center"
     justify="center"
     css={{
       backgroundImage: useColorModeValue(CONFETTI_LIGHT, CONFETTI_DARK),
       backgroundAttachment: 'fixed',
     }}
     id="contact">
     <Box
       borderRadius="lg"
       m={{ base: 0, md: 0, lg: 0 }}
       p={{ base: 0, lg: 0 }}
       maxW="100%"
     >
       <Box>
         <VStack spacing={{ base: 0, md: 0, lg: 0 }}>
           <Heading
             fontSize={{
               base: '4xl',
               md: '5xl',
             }}>
             Get in Touch
           </Heading>

           <Stack
             spacing={{ base:0, md: 0, lg: 0 }}
             direction={{ base: 'column', md: 'row' }}>
             <Stack
               align="center"
               justify="space-around"
               direction={{ base: 'row', md: 'column' }}>


               </Stack>

             <Box
               bg={useColorModeValue('white', 'gray.700')}
               borderRadius="lg"
               w={600}
               p={8}
               color={useColorModeValue('gray.700', 'whiteAlpha.900')}
```

```
shadow="base">
    <VStack spacing={5}>
      <FormControl isRequired>
        <FormLabel>Name</FormLabel>

          <Input
            type="text"
            name="name"
            placeholder="Your Name"
            value={name}
            onChange={(e) => setName(e.target.value)}
          />
      </FormControl>

      <FormControl isRequired>
        <FormLabel>Email</FormLabel>

          <Input
            type="email"
            name="email"
            placeholder="Your Email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
          />
      </FormControl>

      <FormControl isRequired>
        <FormLabel>Message</FormLabel>

        <Textarea
          name="message"
          placeholder="Your Message"
          rows={6}
          resize="none"
          value={message}
          onChange={(e) => setMessage(e.target.value)}
        />
      </FormControl>

      <Button
        colorScheme="blue"
        bg="blue.400"
        color="white"
```

```
_hover={{
                    bg: 'blue.500',
                  }}
                  isFullWidth
                  onClick={
                  handleContactUsSubmit
                  // navigate('/Security/Loading')
                  }>
                  >
                  Send Message
                </Button>
              </VStack>
            </Box>
          </Stack>
        </VStack>
      </Box>
    </Box>
  </Flex>
 );
}
```

Navbar.js

```
import logo from '../assets/logo.png';
import {
  Image,
  Box,
  Flex,
  Text,
  IconButton,
  Button,
  Stack,
  Collapse,
  Icon,
  Link,
  Popover,
  PopoverTrigger,
  PopoverContent,
  useColorModeValue,
  useBreakpointValue,
  useDisclosure,
  useColorMode,
} from '@chakra-ui/react';
import {
  HamburgerIcon,
```

```
  CloseIcon,
  ChevronDownIcon,
  ChevronRightIcon,
} from '@chakra-ui/icons';
import { MoonIcon, SunIcon } from '@chakra-ui/icons';
import Wallet from '../blockchain/Wallet';
import LogoSTAMP from '../assets/logo.png';

export default function WithSubnavigation() {
  const { isOpen, onToggle } = useDisclosure();
  const { colorMode, toggleColorMode } = useColorMode();
  return (
    <Box>
      <Flex
        bg={useColorModeValue('white', 'gray.800')}
        color={useColorModeValue('gray.600', 'white')}
        minH={'60px'}
        py={{ base: 2 }}
        px={{ base: 4 }}
        borderBottom={1}
        borderStyle={'solid'}
        borderColor={useColorModeValue('gray.200', 'gray.900')}
        align={'center'}>
        <Flex
          flex={{ base: 1, md: 'auto' }}
          ml={{ base: -2 }}
          display={{ base: 'flex', md: 'none' }}>
          <IconButton
            onClick={onToggle}
            icon={
              isOpen ? <CloseIcon w={3} h={3} /> : <HamburgerIcon w={5} h={5} />
            }
            variant={'ghost'}
            aria-label={'Toggle Navigation'}
          />
        </Flex>
        <Flex flex={{ base: 1 }} justify={{ base: 'center', md: 'start' }}>
        {/* <Image src={Logo} alt='Logo'  boxSize='50px' objectFit='cover'/> */}
        <Box w='6%' h='10%'>
              <a href="/">
            <Image src={LogoSTAMP}></Image>
          </a>
```

```
</Box>
          <Text
            textAlign={useBreakpointValue({ base: 'center', md: 'left' })}
            fontFamily={'heading'}
            color={useColorModeValue('gray.800', 'white')}>

            <a href = "/">

            </a>
          </Text>

          <Flex display={{ base: 'none', md: 'flex' }} ml={10}>
            <DesktopNav />
          </Flex>
        </Flex>

        {/* <Stack
          flex={{ base: 1, md: 0 }}
          justify={'flex-end'}
          direction={'row'}
          spacing={6}>
          <Button onClick={toggleColorMode}>
             {colorMode === 'light' ? <MoonIcon /> : <SunIcon />}
          </Button>
        </Stack> */}
       <Button onClick={toggleColorMode} style={{ marginRight : '10px' }}>
  {colorMode === 'light' ? <MoonIcon /> : <SunIcon />}
</Button>
<Wallet></Wallet>

        {/* </Stack> */}

       </Flex>

       <Collapse in={isOpen} animateOpacity>
         <MobileNav />
       </Collapse>
     </Box>
   );
  }
```

```
const DesktopNav = () => {
    const linkColor = useColorModeValue('gray.600', 'gray.200');
    const linkHoverColor = useColorModeValue('gray.800', 'white');
    const popoverContentBgColor = useColorModeValue('white', 'gray.800');

    return (
      <Stack direction={'row'} spacing={4}>
        {NAV_ITEMS.map((navItem) => (
          <Box key={navItem.label}>
            <Popover trigger={'hover'} placement={'bottom-start'}>
              <PopoverTrigger>
                <Link
                  p={2}
                  href={navItem.href ?? '#'}
                  fontSize={'sm'}
                  fontWeight={500}
                  color={linkColor}
                  _hover={{
                    textDecoration: 'none',
                    color: linkHoverColor,
                  }}>
                  {navItem.label}
                </Link>
              </PopoverTrigger>

              {navItem.children && (
                <PopoverContent
                  border={0}
                  boxShadow={'xl'}
                  bg={popoverContentBgColor}
                  p={4}
                  rounded={'xl'}
                  minW={'sm'}>
                  <Stack>
                    {navItem.children.map((child) => (
                      <DesktopSubNav key={child.label} {...child} />
                    ))}
                  </Stack>
                </PopoverContent>
              )}
            </Popover>
          </Box>
```

```
))}
      </Stack>
    );
  };

  const DesktopSubNav = ({ label, href, subLabel }: NavItem) => {
    return (
      <Link
        href={href}
        role={'group'}
        display={'block'}
        p={2}
        rounded={'md'}
        _hover={{ bg: useColorModeValue('blue.50', 'gray.900') }}>
        <Stack direction={'row'} align={'center'}>
          <Box>
            <Text
              transition={'all .3s ease'}
              _groupHover={{ color: 'blue.400' }}
              fontWeight={500}>
              {label}
            </Text>
            <Text fontSize={'sm'}>{subLabel}</Text>
          </Box>
          <Flex
            transition={'all .3s ease'}
            transform={'translateX(-10px)'}
            opacity={0}
            _groupHover={{ opacity: '100%', transform: 'translateX(0)' }}
            justify={'flex-end'}
            align={'center'}
            flex={1}>
            <Icon color={'blue.400'} w={5} h={5} as={ChevronRightIcon} />
          </Flex>
        </Stack>
      </Link>
    );
  };

  const MobileNav = () => {
    return (
      <Stack
        bg={useColorModeValue('white', 'gray.800')}
```

```
p={4}
        display={{ md: 'none' }}>
        {NAV_ITEMS.map((navItem) => (
          <MobileNavItem key={navItem.label} {...navItem} />
        ))}
      </Stack>
    );
  };

  const MobileNavItem = ({ label, children, href }: NavItem) => {
    const { isOpen, onToggle } = useDisclosure();

    return (
      <Stack spacing={4} onClick={children && onToggle}>
        <Flex
          py={2}
          as={Link}
          href={href ?? '#'}
          justify={'space-between'}
          align={'center'}
          _hover={{
            textDecoration: 'none',
          }}>
          <Text
            fontWeight={600}
            color={useColorModeValue('gray.600', 'gray.200')}>
            {label}
          </Text>
          {children && (
            <Icon
              as={ChevronDownIcon}
              transition={'all .25s ease-in-out'}
              transform={isOpen ? 'rotate(180deg)' : ''}
              w={6}
              h={6}
            />
          )}
        </Flex>

        <Collapse in={isOpen} animateOpacity style={{ marginTop: '0!important' }}>
          <Stack
            mt={2}
            pl={4}
```

```
borderLeft={1}
          borderStyle={'solid'}
          borderColor={useColorModeValue('gray.200', 'gray.700')}
          align={'start'}>
          {children &&
            children.map((child) => (
              <Link key={child.label} py={2} href={child.href}>
                {child.label}
              </Link>
            ))}
        </Stack>
      </Collapse>
    </Stack>
  );
};

interface NavItem {
  label: string;
  subLabel?: string;
  children?: Array<NavItem>;
  href?: string;
}

const NAV_ITEMS: Array<NavItem> = [
  {
    label: 'Home',
    href: '/',
  },
  {
    label: 'Security', children: [
      {
        label: 'New Record',
        subLabel: 'User not yet recorded',
        href: '/Security/NewRecord',
      },
      {
        label: 'Existing Record',
        subLabel: 'Find your Record',
        href: '/Security/ExistingRecord',
      },
    ],
  },
```

```
{
    label: 'Surveillance',
    href: '/Surveillance',
  },
  {
    label: 'Gesture Control',
    href: '/GestureControl',
  },
  {
    label: 'Settings',
    children: [
      {
        label: 'Motor Callibration',
        subLabel: 'Test your Propellers',
        href: '/Settings/Propellers',
      },
      {
        label: 'Drone Callibration',
        subLabel: 'Callibrate your drone',
        href: '/Settings/Directions',
      },
    ],
  },
  {
    label: 'About Us',
    href: '/About'
  },
  {
    label: 'STAMP Support',
    href: 'http://127.0.0.1:8550/',
  },
  ];
```

Footer.js

```
import {
  Box,
  Image,
  chakra,
  Container,
  Link,
  SimpleGrid,
  Stack,
  Text,
```

```
  VisuallyHidden,
  Input,
  IconButton,
  useColorModeValue,
} from '@chakra-ui/react';
import { ReactNode, useState } from 'react';
import { FaInstagram, FaTwitter, FaYoutube } from 'react-icons/fa';
import { BiMailSend } from 'react-icons/bi';
import LogoSTAMP from '../assets/logo.png';

const Logo = (props: any) => {
  return (
    <>
    <Box w='70%' h='50%' p={0} color='white'>
    <Image src={LogoSTAMP}></Image>
</Box>

    </>
  );
};

const SocialButton = ({
  children,
  label,
  href,
}: {
  children: ReactNode;
  label: string;
  href: string;
}) => {
  return (
    <chakra.button
      bg={useColorModeValue('blackAlpha.100', 'whiteAlpha.100')}
      rounded={'full'}
      w={8}
      h={8}
      cursor={'pointer'}
      as={'a'}
      href={href}
      display={'inline-flex'}
      alignItems={'center'}
      justifyContent={'center'}
      transition={'background 0.3s ease'}
      _hover={{
```

```
        bg: useColorModeValue('blackAlpha.200', 'whiteAlpha.200'),
      }}>
      <VisuallyHidden>{label}</VisuallyHidden>
      {children}
    </chakra.button>
  );
};

const ListHeader = ({ children }: { children: ReactNode }) => {
  return (
    <Text fontWeight={'500'} fontSize={'lg'} mb={2}>
      {children}
    </Text>
  );
};

const sendEmail = async (subject, recipient, body) => {
  const response = await fetch('http://localhost:5000/subscribe-email', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({ subject, recipient, body })
  });
  const data = await response.json();
  console.log(data.message);
}


export default function LargeWithNewsletter() {

  const [subject, setSubject] = useState('');
  const [recipient, setRecipient] = useState('');
  const [body, setBody] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    setSubject('Stay up to date');
    setBody('Thanks for subscribing to our newsletter!');

    // write emails in a file
    fetch('http://localhost:5000/write-file-email', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
```

```
    },
    body: JSON.stringify({ data: recipient })
  })
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.error(error));

  sendEmail(subject, recipient, body);

};

return (
  <>
  <Box
    bg={useColorModeValue('white.50', 'white.900')}
    color={useColorModeValue('gray.700', 'gray.200')}>
    <Container as={Stack} maxW={'6xl'} py={10}>
      <SimpleGrid
        templateColumns={{ sm: '1fr 1fr', md: '2fr 1fr 1fr 2fr' }}
        spacing={8}>
        <Stack spacing={6}>
          <Box>
            <Logo color={useColorModeValue('gray.700', 'white')} />
          </Box>
          <Text fontSize={'sm'}>
          Empowering Your Security, One Flight at a Time.
          </Text>
          <Stack direction={'row'} spacing={6}>
            <SocialButton label={'Twitter'} href={'#'}>
              <FaTwitter />
            </SocialButton>
            <SocialButton label={'YouTube'} href={'#'}>
              <FaYoutube />
            </SocialButton>
            <SocialButton label={'Instagram'} href={'#'}>
              <FaInstagram />
            </SocialButton>
          </Stack>
        </Stack>
        <Stack align={'flex-start'}>
          <ListHeader>Company</ListHeader>
          <Link href={'/About'}>About us</Link>
          <Link href={'#'}>Gallery</Link>
          <Link href={'/About#contact'}>Contact us</Link>
```

```
            <Link href={'/About#team'}>Our Team</Link>
          </Stack>
          <Stack align={'flex-start'}>
            <ListHeader>Support</ListHeader>
            <Link href={'/Surveillance'}>Surveillance</Link>
            <Link href={'Security/NewRecord'}>Security</Link>
            <Link href={'/Settings/Propellers'}>Settings</Link>
          </Stack>
          <Stack align={'flex-start'}>
            <ListHeader>Stay up to date</ListHeader>
            <Stack direction={'row'}>
              <Input
                placeholder={'Your email address'}
                bg={useColorModeValue('blackAlpha.100', 'whiteAlpha.100')}
                border={0}
                _focus={{
                  bg: 'whiteAlpha.300',
                }}
                value={recipient}
                onChange={(e) => setRecipient(e.target.value)}
              />

              <IconButton
                onClick={handleSubmit}
                bg={useColorModeValue('blue.400', 'blue.800')}
                color={useColorModeValue('white', 'gray.800')}
                _hover={{
                  bg: 'blue.600',
                }}
                aria-label="Subscribe"
                icon={<BiMailSend />}
              />
            </Stack>
          </Stack>
        </SimpleGrid>
      </Container>
    </Box>
  </>
 );
};
```

NewRecord.js

```
import React from "react";
import "./NewRecord.css";
import Loading from "./Loading.js";
import {
  FormControl,
  Input,
  FormLabel,
  FormErrorMessage,
  HStack,
  useToast,
  Box,
  chakra,
  Checkbox,
  Radio,
  RadioGroup,
  Divider,
  Select,
  Heading,
  FormHelperText,
  Flex,
  GridItem,
  Stack,
  Text,
  SimpleGrid,
  Button,
} from '@chakra-ui/react'
import { BrowserRouter, Route, Routes, useNavigate,  } from "react-router-dom";
import { useState } from 'react'
function NewRecord() {
  const [firstName, setFirstName] = useState('');
  const [lastName, setLastName] = useState('');
  const [email, setEmail] = useState('');
  const [image, setImage] = useState('');
  const navigate = useNavigate();
  const toast = useToast();

    const handleSubmit = () => {

        // if(firstName && lastName && email) {
            fetch("http://127.0.0.1:5000/newrecord", {
              method: "POST",
              headers: {
                "Content-Type": "application/json",
              },
```

```
            body: JSON.stringify({
              firstName,
              lastName,
              email
            })
          })
          .then(response => {
            if (response.ok) {
              console.log("details sent");
              toast({
                title: `details sent successfully`,
                status: "success",
                duration: 3000,
                isClosable: true,
              });
              window.location.assign('http://localhost:3000/Security/Loading');
              // navigate('/Security/Loading');
              // navigate(-1);
              // setShowLoading(true);
              // window.location.assign('http://localhost:3000/Security/Detect');
            } else {
            console.log("Error while sending details");
            }
          })
          .catch(error => {
            console.log(error);
          });
        // }
        // else {
        //   toast({
        //     title: `Please enter all fields`,
        //     status: "error",
        //     duration: 3000,
        //     isClosable: true,
        //   });
        // }
      }

    return (
      <>
        <Flex justifyContent="center" alignItems="center">
          <SimpleGrid
            display={{
              base: "initial",
```

```
    md: "grid",
  }}
  columns={{
    md: 2,
  }}
  spacing={{
    md: 6,
  }}
>
  <GridItem
    mt={[5, null, 0]}
    colSpan={{
      md: 2,
    }}
  >
    <chakra.form
      method="POST"
      shadow="base"
      width="100%"
      rounded={[null, "md"]}
      overflow={{
        sm: "hidden",
      }}
    >
      <Stack
        px={4}
        py={5}
        p={[null, 6]}
        bg="white"
        _dark={{
          bg: "#141517",
        }}
        spacing={6}
      >
        <SimpleGrid columns={6} spacing={6}>
          <FormControl isRequired as={GridItem} colSpan={[6, 3]}>
            <FormLabel
              htmlFor="first_name"
              fontSize="sm"
              fontWeight="md"
              color="gray.700"
              _dark={{
                color: "gray.50",
              }}
```

```
  >
    First name
  </FormLabel>
  <Input
    type="text"
    name="first_name"
    id="first_name"
    autoComplete="given-name"
    mt={1}
    focusBorderColor="brand.400"
    shadow="sm"
    size="sm"
    w="full"
    rounded="md"
value={firstName}
onChange={(e) => setFirstName(e.target.value)}
  />
</FormControl>

<FormControl isRequired as={GridItem} colSpan={[6, 3]}>
  <FormLabel
    htmlFor="last_name"
    fontSize="sm"
    fontWeight="md"
    color="gray.700"
    _dark={{
      color: "gray.50",
    }}
  >
    Last name
  </FormLabel>
  <Input
    type="text"
    name="last_name"
    id="last_name"
    autoComplete="family-name"
    mt={1}
    focusBorderColor="brand.400"
    shadow="sm"
    size="sm"
    w="full"
    rounded="md"
value={lastName}
```

```
    onChange={(e) => setLastName(e.target.value)}
      />
    </FormControl>
    <FormControl isRequired as={GridItem} colSpan={[6, 4]}>
      <FormLabel
        htmlFor="email_address"
        fontSize="sm"
        fontWeight="md"
        color="gray.700"
        _dark={{
          color: "gray.50",
        }}
      >
        Email address
      </FormLabel>
      <Input
        type="text"
        name="email_address"
        id="email_address"
        autoComplete="email"
        mt={1}
        focusBorderColor="brand.400"
        shadow="sm"
        size="sm"
        w="full"
        rounded="md"
    value={email}
    onChange={(e) => setEmail(e.target.value)}
      />
    </FormControl>
  </SimpleGrid>
</Stack>
<Box
  px={{
    base: 4,
    sm: 6,
  }}
  py={3}
  bg="gray.50"
  _dark={{
    bg: "#121212",
  }}
  textAlign="right"
>
```

```
                <Button
                  type="submit"
                  onClick={
                    handleSubmit
                    // navigate('/Security/Loading')
                  }>
                  Save
                </Button>
                {/* <Routes>
                  <Route path="/Security/Loading" element={<Loading />} /></Routes> */}
              </Box>
            </chakra.form>
          </GridItem>
        </SimpleGrid>
      </Flex>
    </>
  );
}

export default NewRecord;
```

ExistingRecord.js

```
import React from 'react';
import { Flex, Spacer, Button, Text, useMediaQuery, Image, Stack, useToast } from '@chakra-
ui/react';
import { FaTools, FaHandshake, FaStar } from 'react-icons/fa';
import mihir from '../../assets/mihir.jpg';
import prinkal from '../../assets/prinkal.jpg';
import arsh from '../../assets/arsh.jpg';
import tanay from '../../assets/tanay.jpg';
import sarid from '../../assets/sarid.jpg';
import data from './details.json';
import defaultProfile from './default.png';
import { useNavigate } from 'react-router-dom';

const AboutUs = () => {
  const navigate = useNavigate();
  const [isLargerThan48] = useMediaQuery('(min-width: 48em)');
  const toast = useToast();

  const array = data.map(item => ({
    id: item.id,
    text: `${item.first_name} ${item.last_name}`,
    image: item.image,
    subheading: item.email,
  }));

  const find = () => {
    navigate("/Security/Detect");
  }

  const retrain = (fullName, email) => {
    console.log(fullName, email);
    fetch("http://127.0.0.1:5000/single_face_train", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          fullName,
          email
        })
    })
    .then(response => {
```

```
      if (response.ok) {
        console.log("Face Trained Successfully");
        toast({
          title: `Face Trained successfully`,
          status: "success",
          duration: 3000,
          isClosable: true,
        });
      } else {
      console.log("Error while sending details");
      }
  })
  .catch(error => {
      console.log(error);
  });
}

return (
  <Flex
    minH="70vh"
    alignItems="center"
    justifyContent="space-between"
    w="full"
    py="16"
    px={isLargerThan48 ? '20' : '6'}
    flexWrap="wrap"
    flexDirection={isLargerThan48 ? 'row' : 'column'}
  >
    {array.map((arr) => (
      <>
        <Flex
          height="230px"
          bg="blackAlpha.200"
          width={isLargerThan48 ? '18%' : 'full'}
          shadow="md"
          p="6"
          marginTop='20px'
          alignItems="center"
          justifyContent="center"
          borderRadius="md"
          flexDirection="column"
          textAlign="center"
          mb={isLargerThan48 ? '0' : '4'}
          border="1px solid #C4DDFF"
```

```
          >
            <Image src={defaultProfile} borderRadius="full" boxSize="60%" objectFit="cover" />
            <Text fontSize="xl" fontWeight="semibold" >{arr.text}</Text>
            <Text fontSize="sm" color="gray.600">{arr.subheading}</Text>
            <Stack spacing={4} direction={'row'} align={'center'} paddingTop='10px'>
              <Button colorScheme='blue' size={'sm'} onClick={find}>Find</Button>
              <Button colorScheme='blue' size={'sm'} onClick={() => retrain(arr.text,
arr.subheading)}>Retrain</Button>
            </Stack>
          </Flex>
          <Spacer />
        </>
      ))}
    </Flex>
  );
};

export default AboutUs;
```

Detect.js

```
import {
  CardBody,
  useToast,
  Image,
  Card,
  CardFooter,
  Button,
  FormControl,
  Input,
  FormLabel,
  FormErrorMessage,
  HStack,
  Box,
  chakra,
  Checkbox,
  Radio,
  RadioGroup,
  Divider,
  Select,
  Heading,
  FormHelperText,
  Flex,
  GridItem,
  Stack,
  Text,
  SimpleGrid,
  Progress,
} from '@chakra-ui/react';
import React, { useState } from 'react';
import Loading from './Loading.js';
import { BrowserRouter, Route, Routes, useNavigate } from 'react-router-dom';
// import video from '../../../../../1.mp4';
import video from './face.mp4';

const Train = () => {
  const navigate = useNavigate();
  const toast = useToast();

  const [selectedFile, setSelectedFile] = useState(null);
  const [filename, setFilename] = useState('');
  const [outputFilename, setOutputFilename] = useState('');
  const [reportMail, setReportMail] = useState('');
```

```
const [isLoading, setIsLoading] = useState(false);
const startLoading = () => setIsLoading(true);
const stopLoading = () => setIsLoading(false);

const handleFileInput = e => {
  setSelectedFile(e.target.files[0]);
  setFilename(e.target.files[0].name);
};

const uploadFile = () => {
  if (outputFilename === '') {
    toast({
      title: `Please Enter an output file name`,
      status: 'warning',
      duration: 2000,
      isClosable: true,
    });
    return;
  }
  if (selectedFile) {
    const formData = new FormData();
    formData.append('file', selectedFile);
    formData.append('outputFilename', outputFilename);
    startLoading();
    fetch('http://localhost:5000/upload', {
      method: 'POST',
      body: formData,
    })
      .then(response => {
        if (response.ok) {
          console.log('File uploaded successfully');
          toast({
            title: `File "${filename}" uploaded successfully`,
            status: 'success',
            duration: 3000,
            isClosable: true,
          });
          setFilename('');

          //   run code for detection
          fetch('http://127.0.0.1:5000/detect', {
            method: 'POST',
            headers: {
              'Content-Type': 'application/json',
```

```
  },
})
  .then(response => {
    if (response.ok) {
      console.log('Detected');
      toast({
        title: `Detected successfully`,
        status: 'success',
        duration: 3000,
        isClosable: true,
      });
      // send mail report here

      if (reportMail === '') {
        toast({
          title: 'Please enter an email to recieve report',
          status: 'warning',
          duration: 2000,
          isClosable: true,
        });
        return;
      }
      fetch('http://localhost:5000/send_face_report', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ reportMail, outputFilename }),
      })
        .then(response => {
          if (response.ok) {
            stopLoading();
            console.log('Report Sent successfully');
            toast({
              title: `Report Sent successfully`,
              status: 'success',
              duration: 2000,
              isClosable: true,
            });
          } else {
            console.log('Error while Sending report');
          }
        })
        .catch(error => {
```

```
                stopLoading();
                console.log(error);
              });

            window.location.assign(
              'http://localhost:3000/Security/Detect'
            );
          } else {
            stopLoading();
            console.log('Error while Detecting');
          }
        })
        .catch(error => {
          stopLoading();
          console.log(error);
        });
    } else {
      stopLoading();
      console.log('Error uploading file');
    }
  })
  .catch(error => {
    stopLoading();
    console.log(error);
  });
 } else {
  toast({
    title: 'Please select a video file',
    status: 'error',
    duration: 3000,
    isClosable: true,
  });
 }
};

const saveToDisc = () => {
 if (outputFilename === '') {
  toast({
    title: 'Please enter an output file Name',
    status: 'warning',
    duration: 2000,
    isClosable: true,
  });
 } else {
```

```
    startLoading();
    fetch('http://localhost:5000/save_to_disc')
      .then(response => {
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
        return response.blob();
      })
      .then(blob => {
        const url = window.URL.createObjectURL(blob);
        const link = document.createElement('a');
        link.href = url;
        link.setAttribute('download', `${outputFilename}.mp4`);
        document.body.appendChild(link);
        link.click();
        link.parentNode.removeChild(link);
        stopLoading();
      })
      .catch(error => {
        stopLoading();
        console.error('Error downloading file:', error);
      });
  }
};
const saveToCloud = () => {
  if (outputFilename === '') {
    toast({
      title: 'Please enter an output file Name',
      status: 'warning',
      duration: 2000,
      isClosable: true,
    });
  } else {
    startLoading();
    fetch('http://localhost:5000/save_to_cloud', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ outputFilename: outputFilename }),
    })
      .then(response => {
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
```

```
        return response.blob();
      })
      .then(response => {
        stopLoading();
        toast({
          title: `File "${filename}" uploaded successfully to cloud`,
          status: 'success',
          duration: 3000,
          isClosable: true,
        });
      })
      .catch(error => {
        stopLoading();
        console.error('Error downloading file:', error);
      });
  }
};

return (
  <div>
    {isLoading && <Progress size="xs" isIndeterminate />}
    <Box
      bg="#edf3f8"
      _dark={{
        bg: '#111',
      }}
      p={10}
    >
      <Flex justifyContent="center" alignItems="center">
        <Box w="80%">
          <SimpleGrid
            display={{
              base: 'initial',
              md: 'grid',
            }}
            columns={{
              md: 1,
            }}
            spacing={{
              md: 6,
            }}
          >
            <GridItem
              mt={[5, null, 0]}
```

```
  colSpan={{
    md: 2,
  }}
>
  <chakra.form
    method="POST"
    shadow="base"
    rounded={[null, 'md']}
    overflow={{
      sm: 'hidden',
    }}
  >
    <div>
      <video controls style={{ width: '100%' }}>
        <source src={video} type="video/mp4" />
      </video>
    </div>
    <Box
      px={{
        base: 4,
        sm: 6,
      }}
      py={3}
      bg="gray.50"
      _dark={{
        bg: '#121212',
      }}
      textAlign="right"
    >
      <div style={{ display: 'flex', justifyContent: 'left' }}>
        <Input
          value={outputFilename}
          style={{ margin: '0 10px' }}
          onChange={e => setOutputFilename(e.target.value)}
          placeholder="Output Filename"
          width="430px"
        />

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

        <input
          type="file"
          onChange={handleFileInput}
```

```
    style={{ display: 'none' }}
  />

  <Input
    value={reportMail}
    type="email"
    style={{ margin: '0 10px' }}
    onChange={e => setReportMail(e.target.value)}
    placeholder="Mail to send Report"
    width="430px"
  />
</div>

<div
  style={{
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
  }}
>
  {filename && (
    <Text mt={2}>Selected file: {filename}</Text>
  )}
</div>

<div
  style={{
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
  }}
>
  <Stack direction="row" align="center" spacing={4}>
    <Button
      onClick={() =>
        document.querySelector("input[type='file']").click()
      }
    >
      Upload File
    </Button>

    <Button
      onClick={uploadFile}
      disabled={!selectedFile}
```

```
                mt={2}
              >
                Submit
              </Button>

              <Button onClick={saveToDisc}>Save to Disc</Button>

              <Button onClick={saveToCloud}>Save to Cloud</Button>
            </Stack>
          </div>

        </Box>
      </chakra.form>
    </GridItem>
  </SimpleGrid>
 </Box>
</Flex>
</Box>
</div>
);
};

export default Train;
```

Train.js

```
import {
  CardBody,
  Image,
  Card,
  CardFooter,
  Button,
  FormControl,
  Input,
  FormLabel,
  FormErrorMessage,
  HStack,
  Box,
  chakra,
  Checkbox,
  Radio,
  RadioGroup,
  Divider,
  Select,
  Heading,
  FormHelperText,
  Flex,
  GridItem,
  Stack,
  Text,
  SimpleGrid,
} from '@chakra-ui/react'
import React from "react";
import Loading from "./Loading.js";
import { BrowserRouter, Route, Routes, useNavigate } from "react-router-dom";

const Train = () => {
  const navigate = useNavigate();
  return (
    <div>
      <Box
        bg="#edf3f8"
        _dark={{
          bg: "#111",
        }}
        p={10}
      >
        <Flex justifyContent="center" alignItems="center">
          <Box w="80%">
```

```
<SimpleGrid
  display={{
    base: "initial",
    md: "grid",
  }}
  columns={{
    md: 1,
  }}
  spacing={{
    md: 6,
  }}
>
  <GridItem
    mt={[5, null, 0]}
    colSpan={{
      md: 2,
    }}
  >
    <chakra.form
      method="POST"
      shadow="base"
      rounded={[null, "md"]}
      overflow={{
        sm: "hidden",
      }}
    >
      <div>
        <h1>Camera Model Here</h1>
        {/* <Webcam /> */}
      </div>
      <Box
        px={{
          base: 4,
          sm: 6,
        }}
        py={3}
        bg="gray.50"
        _dark={{
          bg: "#121212",
        }}
        textAlign="right"
      >
        <Button
          type="submit"
          onClick={() =>
```

```
                navigate('/Security/Loading')
              }>
              Done
            </Button>
            <Routes>
              <Route path="/Security/Loading" element={<Loading />} /></Routes>

          </Box>
        </chakra.form>
      </GridItem>
    </SimpleGrid>
  </Box></Flex>
</Box>

</div>

);
};

export default Train;
```

Surveillance.js

```
import {
  CardBody,
  Image,
  Card,
  CardFooter,
  Button,
  FormControl,
  Input,
  FormLabel,
  FormErrorMessage,
  HStack,
  Box,
  chakra,
  Checkbox,
  Radio,
  RadioGroup,
  Divider,
  Select,
  Heading,
  FormHelperText,
  Flex,
  GridItem,
  Stack,
  Text,
  SimpleGrid,
  useToast,
  Progress,
} from '@chakra-ui/react';
import React from 'react';

import { BrowserRouter, Route, Routes, useNavigate } from 'react-router-dom';
import { useState } from 'react';
import video from './yolo.mp4';

const Train = () => {
  const navigate = useNavigate();
  const toast = useToast();

  const [selectedFile, setSelectedFile] = useState(null);
  const [filename, setFilename] = useState('');
  const [outputFilename, setOutputFilename] = useState('');
  const [reportMail, setReportMail] = useState('');
```

```
const [isLoading, setIsLoading] = useState(false);
 const startLoading = () => setIsLoading(true);
 const stopLoading = () => setIsLoading(false);

 const handleFileInput = e => {
   setSelectedFile(e.target.files[0]);
   setFilename(e.target.files[0].name);
 };

 const uploadFile = () => {
   if (outputFilename === '') {
     toast({
       title: `Please Enter an output file name`,
       status: 'warning',
       duration: 2000,
       isClosable: true,
     });
     return;
   }
   if (selectedFile) {
     const formData = new FormData();
     formData.append('file', selectedFile);
     formData.append('outputFilename', outputFilename);
     startLoading();
     fetch('http://localhost:5000/yoloupload', {
       method: 'POST',
       body: formData,
     })
       .then(response => {
         if (response.ok) {
           console.log('File uploaded successfully');
           toast({
             title: `File "${filename}" uploaded successfully`,
             status: 'success',
             duration: 3000,
             isClosable: true,
           });
           setFilename('');

           //   run code for YOLO detection
           fetch('http://127.0.0.1:5000/yolo', {
             method: 'POST',
             headers: {
               'Content-Type': 'application/json',
```

```
},
    })
      .then(response => {
       if (response.ok) {
        console.log('Detected');
        toast({
         title: `Detected successfully`,
         status: 'success',
         duration: 3000,
         isClosable: true,
        });
        // send mail report here

        if (reportMail === '') {
         toast({
          title: 'Please enter an email to recieve report',
          status: 'warning',
          duration: 2000,
          isClosable: true,
         });
         return;
        }
        fetch('http://localhost:5000/send_yolo_face_report', {
         method: 'POST',
         headers: {
          'Content-Type': 'application/json',
         },
         body: JSON.stringify({ reportMail, outputFilename }),
        })
         .then(response => {
          if (response.ok) {
           stopLoading();
           console.log('Report Sent successfully');
           toast({
            title: `Report Sent successfully`,
            status: 'success',
            duration: 2000,
            isClosable: true,
           });
          } else {
           console.log('Error while Sending report');
          }
         })
```

```
                .catch(error => {
                  stopLoading();
                  console.log(error);
                });
              window.location.assign('http://localhost:3000/Surveillance');
            } else {
              stopLoading();
              console.log('Error while Detecting');
            }
          })
          .catch(error => {
            stopLoading();
            console.log(error);
          });
      } else {
        stopLoading();
        console.log('Error uploading file');
      }
    })
    .catch(error => {
      stopLoading();
      console.log(error);
    });
  } else {
    toast({
      title: 'Please select a video file',
      status: 'error',
      duration: 3000,
      isClosable: true,
    });
  }
};

const saveToDisc = () => {
  if (outputFilename === '') {
    toast({
      title: 'Please enter an output file Name',
      status: 'warning',
      duration: 2000,
      isClosable: true,
    });
  } else {
    startLoading();
    fetch('http://localhost:5000/save_yolo_to_disc')
      .then(response => {
```

```
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
        return response.blob();
      })
      .then(blob => {
        const url = window.URL.createObjectURL(blob);
        const link = document.createElement('a');
        link.href = url;
        link.setAttribute('download', `${outputFilename}.mp4`);
        document.body.appendChild(link);
        link.click();
        link.parentNode.removeChild(link);
        stopLoading();
      })
      .catch(error => {
        stopLoading();
        console.error('Error downloading file:', error);
      });
  }
};

const saveToCloud = () => {
  if (outputFilename === '') {
    toast({
      title: 'Please enter an output file Name',
      status: 'warning',
      duration: 2000,
      isClosable: true,
    });
  } else {
    startLoading();
    fetch('http://localhost:5000/save_yolo_to_cloud', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ outputFilename: outputFilename }),
    })
      .then(response => {
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
        return response.blob();
      })
```

```
      .then(response => {
        stopLoading();
        toast({
          title: `File "${filename}" uploaded successfully to cloud`,
          status: 'success',
          duration: 3000,
          isClosable: true,
        });
      })
      .catch(error => {
        stopLoading();
        console.error('Error downloading file:', error);
      });
  }
};
return (
  <div>
    {isLoading && <Progress size="xs" isIndeterminate />}
    <Box
      bg="#edf3f8"
      _dark={{
        bg: '#111',
      }}
      p={10}
    >
      <Flex justifyContent="center" alignItems="center">
        <Box w="80%">
          <SimpleGrid
            display={{
              base: 'initial',
              md: 'grid',
            }}
            columns={{
              md: 1,
            }}
            spacing={{
              md: 6,
            }}
          >
            <GridItem
              mt={[5, null, 0]}
              colSpan={{
                md: 2,
              }}
            >
```

```
<chakra.form
  method="POST"
  shadow="base"
  rounded={[null, 'md']}
  overflow={{
    sm: 'hidden',
  }}
>
  <div>
    <video controls style={{ width: '100%' }}>
      <source src={video} type="video/mp4" />
    </video>
    {/* <Webcam /> */}
  </div>
  <Box
    px={{
      base: 4,
      sm: 6,
    }}
    py={3}
    bg="gray.50"
    _dark={{
      bg: '#121212',
    }}
    textAlign="right"
  >
    <div style={{ display: 'flex', justifyContent: 'left' }}>
      <Input
        value={outputFilename}
        onChange={e => setOutputFilename(e.target.value)}
        placeholder="Output Filename"
        width="500px"
        style={{ justifyContent: 'left' }}
      />

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

      <input
        type="file"
        onChange={handleFileInput}
        style={{ display: 'none' }}
      />
```

```
  <Input
    value={reportMail}
    type="email"
    style={{ margin: '0 10px' }}
    onChange={e => setReportMail(e.target.value)}
    placeholder="Mail to send Report"
    width="430px"
  />
</div>
<div
  style={{
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
  }}
>
  {filename && (
    <Text mt={2}>Selected file: {filename}</Text>
  )}
</div>
<div
  style={{
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
  }}
>
  <Stack direction="row" align="center" spacing={4}>
    <Button
      onClick={() =>
        document.querySelector("input[type='file']").click()
      }
    >
      Upload File
    </Button>

    <Button
      onClick={uploadFile}
      disabled={!selectedFile}
      mt={2}
    >
      Submit
    </Button>
```

```
                    <Button onClick={saveToDisc}>Save to Disc</Button>

                    <Button onClick={saveToCloud}>Save to Cloud</Button>
                  </Stack>
                </div>
              </Box>
            </chakra.form>
          </GridItem>
        </SimpleGrid>
      </Box>
    </Flex>
  </Box>
 </div>
 );
};

export default Train;
```

GestureButton.js

```
import React, { useRef, useEffect } from 'react';

const GestureButton = ({ enabled, onClick }) => {
  return (
    <button onClick={onClick}>
      {enabled ? '' : ''}
    </button>
  );
};

export default GestureButton;
```

GestureFeed.js

```
import React, { useRef, useEffect } from 'react';

const getCameraStream = async () => {
   try {
     const stream = await navigator.mediaDevices.getUserMedia({ video: true });
     return stream;
   } catch (err) {
     console.error(err);
   }
};

const GestureFeed = ({ enabled }) => {
   const videoRef = useRef();

   useEffect(() => {
     const initialize = async () => {
       const stream = await navigator.mediaDevices.getUserMedia({ video: true });
       if (stream) {
         videoRef.current.srcObject = stream;
       }
     };
     if (enabled) {
       initialize();
     }
   }, [enabled]);

   return (
     <div>
       <video ref={videoRef} autoPlay muted playsInline />
     </div>
   );
};

  export default GestureFeed;
```

GestureViewing.js

```
import GestureFeed from "./GestureFeed";
import GestureButton from "./GestureButton";
import { useState } from "react";

import {
    Flex,
    Button,
    Text,
    useToast
  } from '@chakra-ui/react';

const GestureViewing = () => {

    const [enabled, setEnabled] = useState(false);
    const toast = useToast();
    // const handleToggleCamera = () => {
    //     setEnabled(!enabled);
    // };

    const detect = () =>{
        setEnabled(true);
        fetch("http://localhost:5000/hand_gesture", {
                method: "GET",
        })
        .then(() => {
            setTimeout(() => {
            }, 1000);
        })
        .catch((error) => {
            console.error(error);
            toast({ description: "Error spinning propellers", status: "error" });
        });
    }

    return (
      <div>

        <Flex
        bg="#edf3f8"
        _dark={{
            bg: "#3e3e3e",
        }}
```

```
      p={50}
      w="full"
      alignItems="center"
      justifyContent="center"
      >

        <Text fontSize='3xl'>Hand Gesture Control</Text>

      </Flex>
      <Flex
      bg="#edf3f8"
      _dark={{
        bg: "#3e3e3e",
      }}
      p={50}
      w="full"
      alignItems="center"
      justifyContent="center"
      >
      </Flex>
      <Flex
      bg="#edf3f8"
      _dark={{
        bg: "#3e3e3e",
      }}
      p={50}
      w="full"
      alignItems="center"
      justifyContent="center"
      >

        <Button colorScheme='teal' variant='solid' onClick={detect}>
          Enable Gesture Control
        </Button>




 </Flex>    </div>
  );
};

export default GestureViewing;
```

Wallet.js

```
import '@rainbow-me/rainbowkit/styles.css';
import {
  getDefaultWallets,
  RainbowKitProvider,
} from '@rainbow-me/rainbowkit';
import { configureChains, createClient, WagmiConfig } from 'wagmi';
import { mainnet, polygon, optimism, arbitrum } from 'wagmi/chains';
import { alchemyProvider } from 'wagmi/providers/alchemy';
import { publicProvider } from 'wagmi/providers/public';
import { ConnectButton } from '@rainbow-me/rainbowkit';
import React from 'react'

const { chains, provider } = configureChains(
  [mainnet, polygon, optimism, arbitrum],
  [
    alchemyProvider({ apiKey: process.env.ALCHEMY_ID }),
    publicProvider()
  ]
);
const { connectors } = getDefaultWallets({
  appName: 'My RainbowKit App',
  chains
});
const wagmiClient = createClient({
  autoConnect: true,
  connectors,
  provider
})



export default function Wallet() {



return (
        <WagmiConfig client={wagmiClient}>
          <RainbowKitProvider chains={chains}>
            <ConnectButton chainStatus="icon" label="STAMP" />
          </RainbowKitProvider>
        </WagmiConfig>
      );
}
```

Direcions.js

```
import React, { useState } from 'react';
import {
  CardBody,
  Image,
  Card,
  CardFooter,
  Button,
  FormControl,
  Input,
  FormLabel,
  FormErrorMessage,
  HStack,
  Box,
  chakra,
  Checkbox,
  Radio,
  RadioGroup,
  Divider,
  Select,
  Heading,
  FormHelperText,
  Flex,
  GridItem,
  Stack,
  Text,
  useToast,
  SimpleGrid,
} from '@chakra-ui/react';
import propellers from "../../assets/propellers.png"
import forwardimg from "../../assets/forward.png"
import backwardimg from "../../assets/backward.png"
import leftimg from "../../assets/left.png"
import rightimg from "../../assets/right.png"


import { BrowserRouter, Route, Routes, useNavigate } from 'react-router-dom';

const Directions = () => {
  const navigate = useNavigate();
  const toast = useToast();
  const toastIdRef = React.useRef();

  const [imageSrc, setImageSrc] = useState(propellers);
```

```
  const left = () => {
    setImageSrc(leftimg);
    toastIdRef.current = toast({description: 'Drone Moving left'})
    fetch("http://localhost:5000/left", {
        method: "POST",
    })
    .then(() => {
        setTimeout(() => {
            setImageSrc(propellers);
        }, 1000); // wait for 5 seconds
    })
    .catch((error) => {
        console.error(error);
        toast({ description: "Error spinning propellers", status: "error" });
    });
  };
  const backward = () => {
    setImageSrc(backwardimg);
    toastIdRef.current = toast({description: 'Drone Moving Backward'})
    fetch("http://localhost:5000/backward", {
        method: "POST",
    })
    .then(() => {
        setTimeout(() => {
            setImageSrc(propellers);
        }, 1000); // wait for 5 seconds
    })
    .catch((error) => {
  console.error(error);



toast({ description: "Error spinning propellers", status: "error" });
    });
  };



  const right = () => {
    setImageSrc(rightimg);
    toastIdRef.current = toast({description: 'Drone Moving Right'})
    fetch("http://localhost:5000/right", {

     method: "POST",
    })
```

```
  .then(() => {
    setTimeout(() => {
      setImageSrc(propellers);
    }, 1000); // wait for 5 seconds
  })
  .catch((error) => {
    console.error(error);
    toast({ description: "Error spinning propellers", status: "error" });
  });
};
const forward = () => {
  setImageSrc(forwardimg);
  toastIdRef.current = toast({description: 'Drone Moving Forward'})
  fetch("http://localhost:5000/forward", {
    method: "POST",
  })
  .then(() => {
    setTimeout(() => {
      setImageSrc(propellers);
    }, 1000); // wait for 5 seconds
  })
  .catch((error) => {
    console.error(error);
    toast({ description: "Error spinning propellers", status: "error" });
  });
};

return (
  <div>



  <Box
    bg="white"
    _dark={{
      bg: '#111',
    }}
    p={10}
   >
    <Flex justifyContent="center" alignItems="center">
```

```
<Box
        w="80%"
        shadow="base"
        rounded={[null, 'md']}
        overflow={{
          sm: 'hidden',
        }}
       >



   <SimpleGrid
         display={{
           base: 'initial',
           md: 'grid',
         }}
         columns={{
           md: 1,
         }}
         spacing={{
           md: 6,
         }}
       >
         <GridItem
           mt={[5, null, 0]}
           colSpan={{
             md: 2,
           }}
         >
           <div style={{ position: 'relative', height: '80vh' }}>
             <button
               type="submit"
               onClick={left}


  style={{
                  position: 'absolute',
                  top: '50%',
                  left: '25%',
                  transform: 'translateY(-50%)',
                  borderRadius: '50%',
                  border: '2px solid black',
                  width: '50px',
                  height: '50px',
                  backgroundColor: '#4299e1',
```

```
}}
            >
              ◄
            </button>
            <button


  type="submit"
              onClick={right}
              style={{
                position: 'absolute',
                top: '50%',
                right: '25%',
                transform: 'translateY(-50%)',
                borderRadius: '50%',
                border: '2px solid black',
                width: '50px',


 height: '50px',
                backgroundColor: '#4299e1',
              }}
            >
              ►{' '}
            </button>
            <button
              type="submit"
              onClick={backward}
              style={{
                position: 'absolute',
                bottom: '0%',
                left: '50%',


   transform: 'translateX(-50%)',
                borderRadius: '50%',
                border: '2px solid black',
                width: '50px',
                height: '50px',
                backgroundColor: '#4299e1',
              }}
```

```
>
                ▼
              </button>
              <button
                type="submit"
                onClick={forward}
                style={{
                  position: 'absolute',
                  top: '0%',
                  left: '50%',
                  transform: 'translateX(-50%)',
                  borderRadius: '50%',
                  border: '2px solid black',
                  width: '50px',


  height: '50px',
                  backgroundColor: '#4299e1',
                }}
              >
                ▲
              </button>
              <div
                style={{
                  display: 'flex',
                  alignItems: 'center',
                  justifyContent: 'center',
                  height: '100%',
                }}


    >
                <img
                  style={{ maxWidth: '50%', maxHeight: '80%' }}
                  src={imageSrc}


    alt="Propellers"
                />
              </div>
            </div>
```

```
            {/* <div style={{ position: 'relative', height: '50vh' }}>
                        <button type="submit"
                           //onClick={fourth}
                           style={{ position: 'absolute', top: '0%', left: '28%', borderRadius: '50%',
border: '2px solid black', width: '50px', height: '50px', backgroundColor: '#4299e1' }}>M4</button>
                        <button type="submit"
                           //onClick={second}
                           style={{ position: 'absolute', top: '0%', right: '28%', borderRadius: '50%',
border: '2px solid black', width: '50px', height: '50px', backgroundColor: '#4299e1' }}>M2</button>
                        <button type="submit"
                           //onClick={third}
                           style={{ position: 'absolute', bottom: '20%', left: '28%', borderRadius: '50%',
border: '2px solid black', width: '50px', height: '50px', backgroundColor: '#4299e1' }}>M3</button>
                        <button type="submit"
                           //onClick={first}
                           style={{ position: 'absolute', bottom: '20%', right: '28%', borderRadius:
'50%', border: '2px solid black', width: '50px', height: '50px', backgroundColor: '#4299e1'
}}>M1</button>
                        <div style={{ display: 'flex', alignItems: 'center', justifyContent: 'center', height:
'100%' }}>
                           <img style={{ maxWidth: '30%', maxHeight: '100%' }} src={directions}
alt="Propellers" />



   </div>
                     </div> */}

            <Box
              px={{
                base: 4,
                sm: 6,
              }}
              py={3}
              bg="gray.50"
              _dark={{
                bg: '#121212',
              }}
              textAlign="right"

  >
            </Box>
          </GridItem>
        </SimpleGrid>
      </Box>
```

```
        </Flex>
      </Box>
    </div>
  );
};

export default Directions;
```

Propellers.js

```
import React, { useState } from 'react';
import {
    CardBody,
    Image,
    Card,
    CardFooter,
    Button,
    FormControl,
    Input,
  FormLabel,
    FormErrorMessage,
    HStack,
    Box,
    chakra,
    Checkbox,
    Radio,
    RadioGroup,
    Divider,
    Select,
    Heading,
    FormHelperText,

  Flex,
    GridItem,
    Stack,
    Text,
    SimpleGrid,
    useToast,
} from '@chakra-ui/react'
```

```
import propellers from '../../assets/propellers.png';
import propellers_onclick from '../../assets/propellers_onclick.gif';
import M4 from '../../assets/M4.gif';
import M3 from '../../assets/M3.gif';
import M2 from '../../assets/M2.gif';
import M1 from '../../assets/M1.gif';
import { BrowserRouter, Route, Routes, useNavigate } from "react-router-dom";

const Propellers = () => {
    const navigate = useNavigate();
    const toast = useToast();
    const toastIdRef = React.useRef();

    const [imageSrc, setImageSrc] = useState(propellers);

    const spinall = () => {
        setImageSrc(propellers_onclick);
        toastIdRef.current = toast({description: 'All Propellers running'});
        fetch("http://localhost:5000/spinall", {
                method: "POST",
        })
        .then(() => {
            setTimeout(() => {
                setImageSrc(propellers);



        }, 1000); // wait for 5 seconds
        })
        .catch((error) => {
            console.error(error);
            toast({ description: "Error spinning propellers", status: "error" });
        });



   };

    const fourth = () => {
        setImageSrc(M4);
        toastIdRef.current = toast({description: 'M1 propeller running'})
        fetch("http://localhost:5000/m1", {
            method: "POST",
        })
```

```
        .then(() => {
            setTimeout(() => {
                setImageSrc(propellers);
            }, 1000); // wait for 5 seconds
        })
        .catch((error) => {
            console.error(error);
            toast({ description: "Error spinning propellers", status: "error" });
        });
    };
    const third = () => {
        setImageSrc(M3);
        toastIdRef.current = toast({description: 'M4 propeller running'})
        fetch("http://localhost:5000/m4", {
            method: "POST",
        })
        .then(() => {
            setTimeout(() => {
                setImageSrc(propellers);
            }, 1000); // wait for 5 seconds
        })
        .catch((error) => {
            console.error(error);
            toast({ description: "Error spinning propellers", status: "error" });
        });
    };
    const second = () => {
        setImageSrc(M2);
        toastIdRef.current = toast({description: 'M2 propeller running'})
        fetch("http://localhost:5000/m2", {
            method: "POST",
        })
        .then(() => {

setTimeout(() => {
                setImageSrc(propellers);
            }, 1000); // wait for 5 seconds
        })
        .catch((error) => {
            console.error(error);
            toast({ description: "Error spinning propellers", status: "error" });
        });
    };
```

```
const first = () => {
    setImageSrc(M1);
    toastIdRef.current = toast({description: 'M3 propeller running'})
    fetch("http://localhost:5000/m3", {
        method: "POST",
    })
    .then(() => {
        setTimeout(() => {
            setImageSrc(propellers);
        }, 1000); // wait for 5 seconds
    })
    .catch((error) => {
        console.error(error);
        toast({ description: "Error spinning propellers", status: "error" });
    });
};

return (
    <div>
        <Box
            bg="white"
            _dark={{
                bg: "#111",
            }}
            p={10}
        >
            <Flex justifyContent="center" alignItems="center">
                <Box w="80%" shadow="base"
                    rounded={[null, "md"]}
                    overflow={{
                        sm: "hidden",
                    }}>
                    <SimpleGrid
                        display={{

base: "initial",
                            md: "grid",
                        }}

  columns={{
                            md: 1,
                        }}
                        spacing={{
                            md: 6,
                        }}
```

```
                >
                    <GridItem
                        mt={[5, null, 0]}
                        colSpan={{
                            md: 2,
                        }}
                    >
                        <div style={{ position: 'relative', height: '50vh' }}>
                            <button type="submit"
                                onClick={fourth} style={{ position: 'absolute', top: '20%', left: '28%',
borderRadius: '50%', border: '2px solid black', width: '50px', height: '50px', backgroundColor:
'#4299e1' }}>M1</button>
                            <button type="submit"
                                onClick={second} style={{ position: 'absolute', top: '20%', right: '28%',
borderRadius: '50%', border: '2px solid black', width: '50px', height: '50px', backgroundColor:
'#4299e1' }}>M2</button>
                            <button type="submit"
                                onClick={third} style={{ position: 'absolute', bottom: '20%', left: '28%',
borderRadius: '50%', border: '2px solid black', width: '50px', height: '50px', backgroundColor:
'#4299e1' }}>M4</button>
                            <button type="submit"
                                onClick={first} style={{ position: 'absolute', bottom: '20%', right: '28%',
borderRadius: '50%', border: '2px solid black', width: '50px', height: '50px', backgroundColor:
'#4299e1' }}>M3</button>
                            <div style={{ display: 'flex', alignItems: 'center', justifyContent: 'center', height:
'100%' }}>
                                <img style={{ maxWidth: '30%', maxHeight: '100%' }} src={imageSrc}
alt="Propellers" />
                            </div>

                        </div>

                        <Box
                            px={{



  base: 4,
                                sm: 6,
                            }}
                            py={3}
                            bg="gray.50"
                            _dark={{
                                bg: "#121212",
```

```
                        }}
                        textAlign="right"
                    >
                    <div style={{display: 'flex', justifyContent: 'center', alignItems: 'center',
marginTop: '10px'}}>
                    <Button
                        type="submit"
                        onClick={spinall}
                    >
                        SPIN ALL
                    </Button>
                    </div>


                    </Box>
                  </GridItem>
                </SimpleGrid>
              </Box></Flex>
          </Box >

       </div >

   );
};

export default Propellers;
```

Flask Backend

server.py

```
from flask import Flask,jsonify,request,Response,make_response, send_file
from flask_cors import CORS
from flask_mail import Mail, Message
from Face_Recognition.one_face_dataset import face_train, add_to_json
from Face_Recognition.two_face_training import yml_train
from Face_Recognition.three_face_recognition import face_detect
from Face_Recognition.savevideo import save_video
from plutox import *
from Cloud_Backend.sns_subscribe import sns_subscribe
from Cloud_Backend.s3 import upload_s3
from Cloud_Backend.dynamodb_contact_us import add_to_dynamodb_contact_us
from Hand_Gesture_Recognition.HandDetection import handDetect
import cv2, os, json, time
import numpy as np
import shutil

app = Flask(__name__)
CORS(app)

app.config['MAIL_SERVER'] = 'smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USE_SSL'] = True
app.config['MAIL_USERNAME'] = 'mihirtestprogrammer@gmail.com'
app.config['MAIL_PASSWORD'] = 'yflrrgfqxxfpyvve'
app.config['MAIL_DEFAULT_SENDER'] = ('STAMP', 'mihirtestprogrammer@gmail.com')

mail = Mail(app)

dic_apis={
   "/surveillance":"takes video input and runs - Object Detection\yoloVideo.py",
   "/dataset":"display opencv stream in a react component and run - Face
Recognition\01_face_dataset.py",
   "/train":"run - Face Recognition\02_face_training.py",
   "/detect":"run - Face Recognition\03_face_recognition.py",
   "/arm":{
     '''
       from plutox import *
       import time
       client = Drone()
       client.arm()
       time.sleep(n)                <---- take n in api
       client.disArm()
     '''
   },
   "/upload":"upload the video to aws bucket",
   "/cameraviewing":"abhi ke liye laptop camera but will give you live camera feed python code if
possible",
   "/sendmail":"get email from footer and send mail to saridqureshi299@gmail.com",
   "/blockchainconnect":"create a python function which adds blockchaindetails to firebase",
```

```
dic_apis_react_call={
    "/surveillance":"http://localhost:3000/surveillance",
    "/dataset":"http://localhost:3000/security/newrecord/dataset",
    "/train":"http://localhost:3000/security/newrecord/train",
    "/detect":"http://localhost:3000/security/detect",
    "/arm":"http://localhost:3000/settings/arm",
    "/upload":"in multiple components wherever video is rendered eg /security , /surveillance ,
/cameraviewing",
    "/cameraviewing":"abhi ke liye laptop camera but will give you live camera feed python code if
possible",
    "/sendmail":"footer me se ",
    "/blockchainconnect":"navbar me :- STAMP\stamp\src\blockchain\Wallet.js
http://localhost:3000/blockchainconnect",
}


@app.route('/surveillance', methods=['POST', 'GET'])
def surveillance():
    return jsonify({'message': 'Hello, World!'})

@app.route('/subscribe-email', methods=['POST'])
def send_email():
    data = request.json
    recipient = data['recipient']
    sns_subscribe(recipient)
    return {'message': 'Email Subscribed'}

@app.route('/write-file-email', methods=['POST'])
def write_file():
    data = request.json['data']
    with open('STAMP/stamp/src/files/emails.txt', 'a') as f:
        f.write(data + '\n')
    upload_s3("STAMP/stamp/src/files/emails.txt")
    return {'success': True}

def get_box_dimensions(outputs, height, width):
    boxes = []
    confs = []
    class_ids = []
    prev_frame_time=0
    new_prev_frame_time=0


    for output in outputs:
        for detect in output:
            scores = detect[5:]
            class_id = np.argmax(scores)
            conf = scores[class_id]
            if conf > 0.4:   #Try .6
                center_x = int(detect[0] * width)
                center_y = int(detect[1] * height)
                w = int(detect[2] * width)
                h = int(detect[3] * height)
```

```
                x = int(center_x - w/2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confs.append(float(conf))
                class_ids.append(class_id)

    return boxes, confs, class_ids

def load_yolo():
    net = cv2.dnn.readNetFromDarknet("Object_Detection/python/yolov3_testing.cfg",
"Object_Detection/python/yolov3.weights")
    with open("Object_Detection/python/coco.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]
    output_layers = [layer_name for layer_name in net.getUnconnectedOutLayersNames()]
    colors = np.random.uniform(0, 255, size=(len(classes), 3))
    return net, classes, colors, output_layers

def detect_objects(img, net, outputLayers):
    blob = cv2.dnn.blobFromImage(img, scalefactor=0.00392, size=(320, 320), mean=(0, 0, 0),
swapRB=True, crop=False)
    net.setInput(blob)
    outputs = net.forward(outputLayers)
    return blob, outputs

def draw_labels(boxes, confs, colors, class_ids, classes, img):
    for i in range(len(boxes)):
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        color = colors[class_ids[i]]
        cv2.rectangle(img, (x,y), (x+w, y+h), color, 2)
        cv2.putText(img, label + str(round(confs[i]*100,2)), (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX,
0.5, color, 2)

def process_frame(frame, net, output_layers, classes, colors):
    height, width, channels = frame.shape
    blob, outputs = detect_objects(frame, net, output_layers)
    boxes, confs, class_ids = get_box_dimensions(outputs, height, width)
    draw_labels(boxes, confs, colors, class_ids, classes, frame)
    return frame

def generate_frames():
    global yolo_file
    # yolo_file = yolo_file.filename
    model, classes, colors, output_layers = load_yolo()
    cap = cv2.VideoCapture("input/" + yolo_file)
    frame_count = 0
    output_folder = "yolo_processing"
    os.makedirs(output_folder, exist_ok=True)
    while True:
        success, frame = cap.read()
        if not success:
            break
        frame = process_frame(frame, model, output_layers, classes, colors)
        ret, buffer = cv2.imencode('.jpg', frame)
```

```
            frame_path = os.path.join(output_folder, f"output_file{frame_count:04d}.jpg")
            cv2.imwrite(frame_path, frame)

            frame = buffer.tobytes()
            frame_count+=1
        generateyolo()
        print("Video Generated")

        source = yolo_file
        dest = "STAMP/stamp/src/components/Surveillance/"

        print("before copy")
        if not os.path.exists(source):
            print("path doesnt exist")
            return f"Source file '{source}' does not exist", 404

        try:
            shutil.copy2(source, os.path.join(dest, "yolo.mp4"))
            print("copied")
        except Exception as e:
            return f"Error copying file: {e}", 500

            # yield (b'--frame\r\n'
            #        b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

    yolo_file = ""

    @app.route('/yoloupload', methods=['POST'])
    def yoloupload():
        global yolo_file
        output_filename = request.form['outputFilename'] + ".mp4"
        if 'file' not in request.files:
            return jsonify({'error': 'No file in request'}), 400
        yolo_file = request.files['file']
        if yolo_file.filename == '':
            return jsonify({'error': 'No file selected'}), 400
        if yolo_file:
            if not os.path.exists('input'):
                os.makedirs('input')
            yolo_file.save('input/' + output_filename)
            yolo_file = output_filename
            return jsonify({'message': f'{yolo_file} uploaded successfully'}), 200
        else:
            return jsonify({'error': 'Failed to upload file'}), 500

    @app.route('/yolo', methods=['POST', 'GET'])
    def index():
        return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

    @app.route('/generateyolo', methods=['POST', 'GET'])
    def generateyolo(): # take videofile from user via user
        global yolo_file
        import cv2
        import os
```

```
    import shutil
    # Path to the directory containing the JPEG images
    image_dir = "yolo_processing/"

    # Output video file name
    # video_file = filename
    video_file = yolo_file

    # Get a list of all the JPEG images in the directory
    image_files = [os.path.join(image_dir, f) for f in os.listdir(image_dir) if f.endswith(".jpg")]

    # Sort the image files in ascending order
    image_files.sort()

    # Read the first image to get the image size
    frame = cv2.imread(image_files[0])
    height, width, channels = frame.shape

    # Define the codec and create a VideoWriter object
    fourcc = cv2.VideoWriter.fourcc('m','p','4','2') # MPEG-4 codec
    out = cv2.VideoWriter(video_file, fourcc, 30.0, (width, height))

    # Loop through all the image files and add them to the video
    for image_file in image_files:
        frame = cv2.imread(image_file)
        out.write(frame)

    # Release the VideoWriter and close all windows
    out.release()
    cv2.destroyAllWindows()
    shutil.rmtree("yolo_processing")

file = ""

@app.route('/upload', methods=['POST'])
def upload():
    global file
    output_filename = request.form['outputFilename'] + ".mp4"
    if 'file' not in request.files:
        return jsonify({'error': 'No file in request'}), 400
    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': 'No file selected'}), 400
    if file:
        if not os.path.exists('input'):
            os.makedirs('input')
        file.save('input/' + output_filename)
        file = output_filename
        return jsonify({'message': f'{file} uploaded successfully'}), 200
    else:
        return jsonify({'error': 'Failed to upload file'}), 500

firstName = ""
lastName = ""
email = ""
```

```
@app.route('/newrecord', methods=['POST'])
def train_face():
    global firstName, lastName, email
    data = request.get_data()
    # print(data)
    parsed_data = json.loads(data)
    firstName = parsed_data['firstName']
    lastName = parsed_data['lastName']
    email = parsed_data['email']
    # print(firstName, lastName, email)
    return jsonify({'message': "details recieved successfully"}), 200

@app.route('/train', methods=['POST'])
def train():
    face_train(firstName, lastName, email)
    yml_train()
    return jsonify({'message': "trained successfully"}), 200

@app.route('/single_face_train', methods=['POST'])
def single_face_train():
    data = request.get_data()
    parsed_data = json.loads(data)
    fullname = parsed_data['fullName'].split()
    fName = fullname[0]
    lName = fullname[1]
    email = parsed_data['email']
    print(fName, lName, email)
    face_train(fName, lName, email)
    yml_train()
    return jsonify({'message': "Single Face trained successfully"}), 200

@app.route('/detect', methods=['POST'])
def detect():
    global file
    face_detect(file)
    save_video(file)
    source = file
    dest = "STAMP/stamp/src/components/Security/"

    print("before copy")
    if not os.path.exists(source):
        print("path doesnt exist")
        return f"Source file '{source}' does not exist", 404

    try:
        shutil.copy2(source, os.path.join(dest, "face.mp4"))
        print("copied")
    except Exception as e:
        return f"Error copying file: {e}", 500

    return jsonify({'message': "detected successfully"}), 200

@app.route('/save_to_disc', methods=['GET'])
def save_to_disc():
    file_path = 'face.mp4'
```

```
    return send_file(file_path, as_attachment=True)

@app.route('/save_to_cloud', methods=['GET', 'POST'])
def save_to_cloud():
    data = request.get_json()
    file_path = data['outputFilename'] + ".mp4"
    upload_s3(file_path)
    return jsonify({'message': "Saved to Cloud"}), 200

@app.route('/send_face_report', methods=['GET', 'POST'])
def send_face_report():
    data = request.json
    subject = "Face Detection report"
    recipient = data['reportMail']
    outputFilename = data['outputFilename'] + ".mp4"
    body = "Please find the attached report for your Face Detection"

    message = Message(subject=subject, recipients=[recipient], body=body)
    print(outputFilename)
    with app.open_resource(outputFilename) as fp:
        message.attach(outputFilename, "video/mp4",fp.read())
        mail.send(message)

    return jsonify({'message': "Report sent to mail successfully"}), 200

@app.route('/send_yolo_face_report', methods=['GET', 'POST'])
def send_yolo_face_report():
    data = request.json
    subject = "Yolo Detection report"
    recipient = data['reportMail']
    outputFilename = data['outputFilename'] + ".mp4"
    body = "Please find the attached report for your Yolo Detection"

    message = Message(subject=subject, recipients=[recipient], body=body)
    print(outputFilename)
    with app.open_resource(outputFilename) as fp:
        message.attach(outputFilename, "video/mp4",fp.read())
        mail.send(message)

    return jsonify({'message': "Report sent to mail successfully"}), 200

@app.route('/save_yolo_to_disc', methods=['GET'])
def save_yolo_to_disc():
    file_path = 'yolo.mp4'
    return send_file(file_path, as_attachment=True)

@app.route('/save_yolo_to_cloud', methods=['GET', 'POST'])
def save_yolo_to_cloud():
    data = request.get_json()
    file_path = data['outputFilename'] + ".mp4"
    upload_s3(file_path)
    return jsonify({'message': "Saved to Cloud"}), 200

@app.route('/hand_gesture', methods=['GET'])
def hand_gesture():
```

```
    handDetect()
    return jsonify({'message': "Hand Gesture"}), 200

@app.route('/contact_us', methods=['POST'])
def contact_us():
    data = request.get_data()
    parsed_data = json.loads(data)
    name = parsed_data['name']
    email = parsed_data['email']
    message = parsed_data['message']
    add_to_dynamodb_contact_us(name,email,message)
    return jsonify({'message': "details recieved successfully"}), 200

@app.route('/test', methods=['GET'])
def test():
    # for testing only
    yml_train()
    return jsonify({'message': "Success"}), 200

@app.route('/spinall', methods=['POST'])
def spinall():
    client = Drone()
    client.arm()
    time.sleep(5)
    client.disArm()
    return jsonify({'message': "Success"}), 200

@app.route('/m1', methods=['POST'])
def m1():
    client = Drone()
    client.m1()
    time.sleep(5)
    client.m1stop()
    return jsonify({'message': "Success"}), 200

@app.route('/m2', methods=['POST'])
def m2():
    client = Drone()
    client.m2()
    time.sleep(5)
    client.m2stop()
    return jsonify({'message': "Success"}), 200

@app.route('/m3', methods=['POST'])
def m3():
    client = Drone()
    client.m3()
    time.sleep(5)
    client.m3stop()
    return jsonify({'message': "Success"}), 200

@app.route('/m4', methods=['POST'])
def m4():
    client = Drone()
    client.m4()
```

```
    time.sleep (5)
    client.m4stop()
    return jsonify({'message': "Success"}), 200

@app.route('/left', methods=['POST'])
def left():
    client = Drone()
    client.left()
    time.sleep(5)
    client.leftstop()
    return jsonify({'message': "Success"}), 200

@app.route('/right', methods=['POST'])
def right():
    client = Drone()
    client.right()
    time.sleep(5)
    client.rightstop()
    return jsonify({'message': "Success"}), 200

@app.route('/forward', methods=['POST'])
def forward():
    client = Drone()
    client.forward()
    time.sleep(5)
    client.forwardstop()
    return jsonify({'message': "Success"}), 200

@app.route('/backward', methods=['POST'])
def backward():
    client = Drone()
    client.backward()
    time.sleep(5)
    client.backwardstop()
    return jsonify({'message': "Success"}), 200

if __name__ == '__main__':
        app.run(debug=True)
```

FACE RECOGNITION

one_face_dataset.py

```
import json
import uuid
import os

def add_to_json(first_name, last_name, email, image):

    id = str(uuid.uuid4())

    # Create a dictionary for the new entry
    new_entry = {
        "id": id,
        "first_name": first_name,
        "last_name": last_name,
        "email": email,
        "image": image
    }

    # Check if the JSON file exists
    if os.path.isfile('STAMP/stamp/src/components/Security/details.json'):
        # If the file exists, load the existing data
        with open('STAMP/stamp/src/components/Security/details.json', 'r') as f:
            data = json.load(f)

        # Check if the name already exists in the JSON file
        for entry in data:
            if entry["first_name"] == first_name and entry["last_name"] == last_name:
                # If the name already exists, update the entry and exit the loop
                entry.update(new_entry)
                break
        else:
            # If the name does not exist, append the new entry to the data
            data.append(new_entry)

        # Write the updated data to the JSON file
        with open('STAMP/stamp/src/components/Security/details.json', 'w') as f:
            json.dump(data, f, indent=4)
    else:
        # If the file does not exist, create a new list with the new entry
        data = [new_entry]

        # Write the data to the JSON file
        with open('STAMP/stamp/src/components/Security/details.json', 'w') as f:
            json.dump(data, f, indent=4)


def face_train(first_name, last_name, email):
    import cv2
    import os

    cam = cv2.VideoCapture(0)
```

```
    cam.set(3, 640) # set video width
    cam.set(4, 480) # set video height

    face_detector =
cv2.CascadeClassifier('Face_Recognition/haarcascade_frontalface_default.xml')

    font = cv2.FONT_HERSHEY_SIMPLEX

    # For each person, enter one numeric face id
    # face_id = input('\n enter user id end press <return> ==>  ')

    print("\n [INFO] Initializing face capture. Look the camera and wait ...")
    # Initialize individual sampling face count
    count = 1

    while(True):

        ret, img = cam.read()
        img = cv2.flip(img, 1) # flip video image vertically
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_detector.detectMultiScale(gray, 1.3, 5)

        for (x,y,w,h) in faces:

            cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
            count += 1

            # Save the captured image into the datasets folder
            cv2.putText(img, str(count), (x+5,y-5), font, 1, (255,255,255), 2)
            cv2.imwrite("Face_Recognition/dataset/User." + first_name + last_name + '.' + str(count)
+ ".jpg", gray[y:y+h,x:x+w])
        cv2.imshow('image', img)

        k = cv2.waitKey(100) & 0xff # Press 'ESC' for exiting video
        if k == 27:
            break
        elif count >= 100: # Take 100 face sample and stop video
            break

    # Do a bit of cleanup
    print("\n [INFO] Exiting Program and cleanup stuff")
    cam.release()
    cv2.destroyAllWindows()


    # adding data to json
    add_to_json(first_name, last_name, email, f"User.{first_name}{last_name}.1.jpg")
```

save_video.py

```
def save_video(filename):
    import cv2
    import os
    import shutil
    # Path to the directory containing the JPEG images
    image_dir = "face_processing/"

    # Output video file name
    video_file = filename

    # Get a list of all the JPEG images in the directory
    image_files = [os.path.join(image_dir, f) for f in os.listdir(image_dir) if f.endswith(".jpg")]

    # Sort the image files in ascending order
    image_files.sort()

    # Read the first image to get the image size
    frame = cv2.imread(image_files[0])
    height, width, channels = frame.shape

    # Define the codec and create a VideoWriter object
    fourcc = cv2.VideoWriter.fourcc('m','p','4','2') # MPEG-4 codec
    out = cv2.VideoWriter(video_file, fourcc, 30.0, (width, height))

    # Loop through all the image files and add them to the video
    for image_file in image_files:
        frame = cv2.imread(image_file)
        out.write(frame)

    # Release the VideoWriter and close all windows
    out.release()
    cv2.destroyAllWindows()
    shutil.rmtree("face_processing")
```

two_face_training.py

```
def add_to_json(image_label_set):
    import json

    # Define the original set of tuples

    # Read the JSON object from a file
    with open("STAMP/stamp/src/components/Security/details.json", "r") as f:
        users_list = json.load(f)

    # Loop through the list of users to find those whose names are in the original set
    for user in users_list:
        for image_id, name in image_label_set:
            if name == user["first_name"] + user["last_name"]:
                # If the user is found, add an "image_id" field with the corresponding value from the
original set
                user["image_id"] = image_id

    # Write the updated JSON object back to the file
    with open("STAMP/stamp/src/components/Security/details.json", "w") as f:
        json.dump(users_list, f)

def rearrange_json():
    import json

    # load the JSON file
    with open('STAMP/stamp/src/components/Security/details.json') as f:
        data = json.load(f)

    # sort the data by image_id
    data_sorted = sorted(data, key=lambda x: x['image_id'])

    # save the sorted data to a new file
    with open('STAMP/stamp/src/components/Security/details.json', 'w') as f:
        json.dump(data_sorted, f, indent=4)

def yml_train():
    import cv2
    import numpy as np
    from Cloud_Backend.dynamodb import upload_dynamodb_details
    from Cloud_Backend.s3 import upload_s3
    from PIL import Image
    import os

    # Path for face image database
    path = 'Face_Recognition/dataset'

    image_label_set=set()

    recognizer = cv2.face.LBPHFaceRecognizer_create()
    detector = cv2.CascadeClassifier("Face_Recognition/haarcascade_frontalface_default.xml")
    labels = []
    # function to get the images and label data
```

```
    def getImagesAndLabels(path):

        imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
        faceSamples=[]
        labels_dict = {}
        label = 0

        for imagePath in imagePaths:

            PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
            img_numpy = np.array(PIL_img,'uint8')

            name = os.path.split(imagePath)[-1].split(".")[1]
            if name not in labels_dict:
                labels_dict[name] = label
                label += 1
            label_id = labels_dict[name]
            faces = detector.detectMultiScale(img_numpy)

            for (x,y,w,h) in faces:
                faceSamples.append(img_numpy[y:y+h,x:x+w])
                labels.append(label_id)
            image_label_set.add((label_id,name))
        return faceSamples, labels

    print ("\n [INFO] Training faces. It will take a few seconds. Wait ...")
    faces, labels = getImagesAndLabels(path)
    recognizer.train(faces, np.array(labels))

    # Save the model into trainer/trainer.yml
    recognizer.write('Face_Recognition/trainer/trainer.yml') # recognizer.save() worked on Mac,
but not on Pi

    add_to_json(image_label_set)
    rearrange_json()
    upload_dynamodb_details()
    upload_s3("STAMP\stamp\src\components\Security\details.json")
    # Print the number of faces trained and end program
        # adding data to json
    print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(labels))))
```

three_face_recognition.py

```
def face_detect(input_file, output_file = "faceoutput.mp4"):
    import cv2
    import numpy as np
    import json,os

    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read('Face_Recognition/trainer/trainer.yml')
    cascadePath = "Face_Recognition/haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(cascadePath)

    font = cv2.FONT_HERSHEY_SIMPLEX

    #iniciate id counter
    id = 0

    with open('STAMP/stamp/src/components/Security/details.json') as f:
        data = json.load(f)

    # print(data)
    names = []
    for i in data:
        names.append(i['first_name'] + i['last_name'])
    # names related to ids: example ==> Marcelo: id=1,  etc

    # print(names)
    input_file = input_file

    # Initialize and start realtime video capture
    cam = cv2.VideoCapture("input/" + input_file)
    if not cam.isOpened():
        print("Error opening Video File.")
    cam.set(3, 640) # set video widht
    cam.set(4, 480) # set video height

    # Define min window size to be recognized as a face
    minW = 0.1*cam.get(3)
    minH = 0.1*cam.get(4)

    output_folder="face_processing"
    os.makedirs(output_folder, exist_ok=True)

    # Initialize a frame counter
    frame_count = 0
    while True:

        ret, img =cam.read()
        if img is None:
            break
```

```
  img = cv2.flip(img, 1) # Flip vertically
        # img=cv2.imread("./test.jpg",1)
        # if img is not None:
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

        faces = faceCascade.detectMultiScale(
            gray,
            scaleFactor = 1.2,
            minNeighbors = 5,
            minSize = (int(minW), int(minH)),
        )

        for(x,y,w,h) in faces:

            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)

            id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
            print(f"{id} => {confidence}")
            # Check if confidence is less them 100 ==> "0" is perfect match
            if (confidence < 70):
                id = names[id]
                print("Id is", id)
                confidence = "  {0}%".format(round(100 - confidence))
            else:
                id = "unknown"
                confidence = "  {0}%".format(round(100 - confidence))

            cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)
            cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)

        # Write the frame into a JPEG file
        frame_path = os.path.join(output_folder, f"output_file{frame_count:04d}.jpg")
        cv2.imwrite(frame_path, img)

        # Increment the frame counter
        frame_count += 1

        # cv2.imshow('camera',img)

        # Write the frame into the output video file

        k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video
        if k == 27:
            break


    # Release everything if job is finished
    cam.release()
    cv2.destroyAllWindows()


# face_detect()
```

HAND GESTURE DETECTION

HandDetection.py

```
# import necessary packages
import cv2
import numpy as np
import time
from plutox import *
import mediapipe as mp
import tensorflow as tf
from keras.models import load_model
from gtts import gTTS
import pyttsx3

def tts(text):
    # initialize Text-to-speech engine
    engine = pyttsx3.init()
    voices = engine.getProperty('voices')
    engine.setProperty("rate", 178)
    engine.setProperty('voice', voices[1].id) #changing index changes voices but ony 0 and 1 are
working here
    # convert this text to speech
    engine.say(text)
    # play the speech
    engine.runAndWait()

def handDetect():

    mpHands = mp.solutions.hands
    hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)
    mpDraw = mp.solutions.drawing_utils

    model = load_model('Hand_Gesture_Recognition\mp_hand_gesture')
    print(f"MODEL => {model}")
    f = open('Hand_Gesture_Recognition\gesture.names', 'r')
    classNames = f.read().split('\n')
    f.close()

    cap = cv2.VideoCapture(0)
    while True:
        _, frame = cap.read()

        x, y, c = frame.shape

        frame = cv2.flip(frame, 1)
        framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        result = hands.process(framergb)

        className = ''

        if result.multi_hand_landmarks:
            landmarks = []
            for handslms in result.multi_hand_landmarks:
                for lm in handslms.landmark:
                    lmx = int(lm.x * x)
```

```
                lmy = int(lm.y * y)

                landmarks.append([lmx, lmy])

            mpDraw.draw_landmarks(frame, handslms, mpHands.HAND_CONNECTIONS)
            prediction = model.predict([landmarks])
            classID = np.argmax(prediction)
            className = classNames[classID]


        cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2,
cv2.LINE_AA)

        cv2.imshow("Output is :", frame)

        k = cv2.waitKey(100) & 0xff # Press 'ESC' for exiting video
        if k == 27:
            break
        checkGesture(className)

    cap.release()

    cv2.destroyAllWindows()

def checkGesture(hd):

    if hd=='rock':
        #spinall
        tts("Rock Gesture Detected")
        tts("PlutoX Takeoff Instantiated")
        spinall()
    elif hd=='thumbs up':
        #forward
        tts("Thumbs Up Gesture Detected")
        tts("PlutoX Forward Motion Instantiated")
        forward()
    elif hd=='thumbs down':
        #backward
        tts("Thumbs Down Gesture Detected")
        tts("PlutoX Backward Motion Instantiated")
        backward()
    elif hd=='fist':
        #left
        tts("Fist Gesture Detected")
        tts("PlutoX Left Motion Instantiated")
        left()
    elif hd=='call me':
        #right
        tts("Call Me Gesture Detected")
        tts("PlutoX Right Motion Instantiated")
        right()
    elif hd=='peace':
        #m1
        tts("Peace Gesture Detected")
```

```
            tts("PlutoX M1 Propeller Instantiated")
            m1()
        elif hd=='okay':
            #m2
            tts("Okay Gesture Detected")
            tts("PlutoX M2 Propeller Instantiated")
            m2()
        elif hd=='stop':
            #m3
            tts("Stop Gesture Detected")
            tts("PlutoX M3 Propeller Instantiated")
            m3()
        elif hd=='smile':
            #m4
            tts("Smile Gesture Detected")
            tts("PlutoX M4 Propeller Instantiated")
            m4()
        return

    def m1():
        client = Drone()
        client.m1()
        time.sleep(5)
        client.m1stop()

    def m2():
        client = Drone()
        client.m2()
        time.sleep(5)
        client.m2stop()

    def m3():
        client = Drone()
        client.m3()
        time.sleep(5)
        client.m3stop()

    def m4():
        client = Drone()
        client.m4()
        time.sleep(5)
        client.m4stop()

    def left():
        client = Drone()
        client.left()
        time.sleep(5)
        client.leftstop()

    def right():
        client = Drone()
        client.right()
        time.sleep(5)
        client.rightstop()
```

```
def forward():
    client = Drone()
    client.forward()
    time.sleep(5)
    client.forwardstop()

def backward():
    client = Drone()
    client.backward()
    time.sleep(5)
    client.backwardstop()

def spinall():
    client = Drone()
    client.arm()
    time.sleep(5)
    client.disArm()
```

OBJECT DETECTION

videoCreate.py

```
import cv2
import glob


img_array = []
file_list=[]
for file_n in glob.glob('frames/*jpg'):
    file_n=file_n.replace('frames/','')#TODO save the first and last part in variables
    file_n = file_n.replace('.jpg', '')
    file_list.append(file_n)
    print(file_n)


file_list.sort(key=int)
print(file_list)
for filename in file_list:
    img = cv2.imread('frames/'+filename+'.jpg')
    height, width, layers = img.shape
    size = (width, height)
    img_array.append(img)
    print('appending',filename)

out = cv2.VideoWriter('videos/project.mp4', cv2.VideoWriter_fourcc(*'H264'), 15, size)

for i in range(len(img_array)):
    out.write(img_array[i])
    print('Frame:',i)
out.release()

print('Done')
```

yoloVideo.py

```
import cv2
import numpy as np
import time
import glob
import os
global cnt
cnt=0

def load_yolo():


net=cv2.dnn.readNetFromDarknet("./public/python/yolov3_testing.cfg","./public/python/yolov3.w
eights")

    with open("./public/python/coco.names", "r") as f:
      classes = [line.strip() for line in f.readlines()]

    output_layers = [layer_name for layer_name in net.getUnconnectedOutLayersNames()]
    colors = np.random.uniform(0, 255, size=(len(classes), 3))
    return net, classes, colors, output_layers


def start_video(video_path):
   model, classes, colors, output_layers = load_yolo()
   cap = cv2.VideoCapture(0)
   cnte=0
   ret=True
   # out = cv2.VideoWriter('videos/proc.mp4', cv2.VideoWriter_fourcc(*'mp4v'), 20.0, (416,416))
   while ret:
      ret, frame = cap.read()
      #print(ret,cnte)
      if not ret:
         break
      # cv2.imshow('Win',frame)
      # cv2.waitKey(1500)
      try:
         height, width, channels = frame.shape
         blob, outputs = detect_objects(frame, model, output_layers)
      except:
         pass

      boxes, confs, class_ids = get_box_dimensions(outputs, height, width)
      draw_labels(boxes, confs, colors, class_ids, classes, frame)
      cnte+=1
      # out.write(img) #--- out----FUNC
      key = cv2.waitKey(1)
      if key == 27:
         break
   cap.release()
```

```
def get_box_dimensions(outputs, height, width):
  boxes = []
  confs = []
  class_ids = []
  prev_frame_time=0
  new_prev_frame_time=0


  for output in outputs:
    for detect in output:
      scores = detect[5:]
      class_id = np.argmax(scores)
      conf = scores[class_id]
      if conf > 0.4:   #Try .6
        center_x = int(detect[0] * width)
        center_y = int(detect[1] * height)
        w = int(detect[2] * width)
        h = int(detect[3] * height)
        x = int(center_x - w/2)
        y = int(center_y - h / 2)
        boxes.append([x, y, w, h])
        confs.append(float(conf))
        class_ids.append(class_id)
  # codec = cv2.VideoWriter_fourcc(*"MJPG")
  # new_frame_time = time.time()
  # fps = 1 / (new_frame_time - prev_frame_time)
  # prev_frame_time = new_frame_time
  # fps_n = int(fps)

  return boxes, confs, class_ids

def detect_objects(img, net, outputLayers):
   blob = cv2.dnn.blobFromImage(img, scalefactor=0.00392, size=(320, 320), mean=(0, 0, 0),
swapRB=True, crop=False)
   net.setInput(blob)
   outputs = net.forward(outputLayers)
   return blob, outputs

def save_vdo():
  img_array = []
  file_list=[]
  for file_n in glob.glob('./public/python/frames/*jpg'):
     file_n=file_n.replace('./public/python/frames/','')#TODO save the first and last part in
variables
     file_n = file_n.replace('.jpg', '')
     file_list.append(file_n)
     # print(file_n)


  file_list.sort(key=int)
  #print(file_list)
  for filename in file_list:
```

```
        img = cv2.imread('./public/python/frames/'+filename+'.jpg')
        height, width, layers = img.shape
        size = (width, height)
        img_array.append(img)
        #print('appending',filename)

    out = cv2.VideoWriter('./public/python/videos/project.mp4', cv2.VideoWriter_fourcc(*'H264'),
15, size)

    for i in range(len(img_array)):
        out.write(img_array[i])
        #print('Frame:',i)
    out.release()

    #print('Appended and framed')


def draw_labels(boxes, confs, colors, class_ids, classes, img):
    global cnt
    indexes = cv2.dnn.NMSBoxes(boxes, confs, 0.5, 0.4) #(0.2,0.2) // (0.7,0.6)
    font = cv2.FONT_HERSHEY_DUPLEX
    # image_folder = 'data-set-race-01'
    video_file = './public/python/videos/proc.mp4'
    image_size = (416, 416)
    fps = 24
    # out = cv2.VideoWriter(video_file, cv2.VideoWriter_fourcc('M','P','E','G'), fps, image_size)

    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            # color = colors[i]

            if confs[i] > 0.85:
                color=(0,255,0)#GREEN
            elif confs[i] > 0.6:
                color = (255,0,0)#ORange
            elif confs[i] >0.4:
                color=(255,0,255)#RED


            cv2.rectangle(img, (x,y), (x+w, y+h), color, 2)
            cv2.putText(img, label + str(round(confs[i]*100,2)), (x, y - 5), font, 1, color, 1)
            # out.write(img)

    cv2.imshow("Image", img)
    cv2.imwrite('./public/python/frames/'+str(cnt)+'.jpg',img)
    #print('\t',cnt)
    cnt += 1
```

```
if os.path.exists("./public/python/videos/project.mp4"):
  os.remove("./public/python/videos/project.mp4")

dir = './public/python/frames/'
for f in os.listdir(dir):
    os.remove(os.path.join(dir, f))

if os.path.exists("./public/python/videos/raw1.mp4"):
   video_path = './public/python/videos/raw1.mp4'
   #print('Opening ' + video_path + " .... ")
   start_video(video_path)
   # save_vdo()
   print('Video computed')
else:
   print("Video doesn't exists")
```

STAMP SUPPORT

Stamp_support.py

```
import time
import flet as ft
from tts import tts
from plutox import *

class Message():
    def __init__(self, user_name: str, text: str, message_type: str):
        self.user_name = user_name
        self.text = text
        self.message_type = message_type

class ChatMessage(ft.Row):
    def __init__(self, message: Message):
        super().__init__()
        self.vertical_alignment="start"
        self.controls=[
                ft.CircleAvatar(
                    content=ft.Text(self.get_initials(message.user_name)),
                    color=ft.colors.WHITE,
                    bgcolor=self.get_avatar_color(message.user_name),
                ),
                ft.Column(
                    [
                        ft.Text(message.user_name, weight="bold"),
                        ft.Text(message.text, selectable=True),
                    ],
                    tight=True,
                    spacing=5,
                ),
            ]

    def get_initials(self, user_name: str):
        return user_name[:1].capitalize()

    def get_avatar_color(self, user_name: str):
        colors_lookup = [
            ft.colors.AMBER,
            ft.colors.BLUE,
            ft.colors.BROWN,
            ft.colors.CYAN,
            ft.colors.GREEN,
            ft.colors.INDIGO,
            ft.colors.LIME,
            ft.colors.ORANGE,
            ft.colors.PINK,
            ft.colors.PURPLE,
            ft.colors.RED,
            ft.colors.TEAL,
            ft.colors.YELLOW,
```

```
    ]
    return colors_lookup[hash(user_name) % len(colors_lookup)]

def main(page: ft.Page):
    page.horizontal_alignment = "stretch"
    page.title = "STAMP Support"

    def join_chat_click(e):
        if not join_user_name.value:
            join_user_name.error_text = "Name cannot be blank!"
            join_user_name.update()
        else:
            if join_user_name.value=="3511plutox":
                page.session.set("user_name", "Admin")
                page.dialog.open = False
                new_message.prefix = ft.Text(f"{join_user_name.value}: ")
                page.pubsub.send_all(Message(user_name=join_user_name.value, text=f"Admin
 has joined the chat.", message_type="login_message"))
                page.update()
            elif join_user_name.value=="Admin":
                join_user_name.error_text = "Name cannot be Admin"
                join_user_name.update()
            else:
                page.session.set("user_name", join_user_name.value)
                page.dialog.open = False
                new_message.prefix = ft.Text(f"{join_user_name.value}: ")
                page.pubsub.send_all(Message(user_name=join_user_name.value,
text=f"{join_user_name.value} has joined the chat.", message_type="login_message"))
                page.update()

    def send_message_click(e):
        if new_message.value != "":
            page.pubsub.send_all(Message(page.session.get("user_name"), new_message.value,
message_type="chat_message"))
            temp=new_message.value
            new_message.value = ""
            new_message.focus()
            if temp.startswith("/?"):
                res=chatgpt(temp)
                if len(res) > 220: # adjust the maximum length as needed
                    res = '\n'.join([res[i:i+220] for i in range(0, len(res), 220)])
                page.pubsub.send_all(Message("STAMP Support", res,
message_type="chat_message"))
                tts(res)
            elif page.session.get("user_name")=="Admin":
                if temp=="?spinall" or temp=="take off":
                    res="PlutoX Takeoff Instantiated"
                    page.pubsub.send_all(Message("Pluto X", res, message_type="chat_message"))
                    tts(res)
                    spinall()
                elif temp=="?backward" or temp=="backward":
                    res="PlutoX Backward Motion Instantiated"
```

```
page.pubsub.send_all(Message("Pluto X", res, message_type="chat_message"))
                tts(res)
                backward()
            elif temp=="?forward" or temp=="forward":
                res="PlutoX Forward Motion Instantiated"
                page.pubsub.send_all(Message("Pluto X", res, message_type="chat_message"))
                tts(res)
                forward()
            elif temp=="?left" or temp=="left":
                res="PlutoX Left Motion Instantiated"
                page.pubsub.send_all(Message("Pluto X", res, message_type="chat_message"))
                tts(res)
                left()
            elif temp=="?right" or temp=="right":
                res="PlutoX Right Motion Instantiated"
                page.pubsub.send_all(Message("Pluto X", res, message_type="chat_message"))
                tts(res)
                right()
            elif temp=="?m1" or temp=="M1":
                res="PlutoX M1 Propeller Instantiated"
                page.pubsub.send_all(Message("Pluto X", res, message_type="chat_message"))
                tts(res)
                m1()
            elif temp=="?m2" or temp=="M2":
                res="PlutoX M2 Propeller Instantiated"
                page.pubsub.send_all(Message("Pluto X", res, message_type="chat_message"))
                tts(res)
                m2()
            elif temp=="?m3" or temp=="M3":
                res="PlutoX M3 Propeller Instantiated"
                page.pubsub.send_all(Message("Pluto X", res, message_type="chat_message"))
                tts(res)
                m3()
            elif temp=="?m4" or temp=="M4":
                res="PlutoX M4 Propeller Instantiated"
                page.pubsub.send_all(Message("Pluto X", res, message_type="chat_message"))
                tts(res)
                m4()
            temp=""
        else:
            pass
        page.update()

 def spinall():
    client = Drone()
    client.arm()
    time.sleep(5)
    client.disArm()

 def backward():
    client = Drone()
    client.backward()
```

```
 time.sleep(5)
    client.backwardstop()

def forward():
    client = Drone()
    client.forward()
    time.sleep(5)
    client.forwardstop()



def right():
    client = Drone()
    client.right()
    time.sleep(5)
    client.rightstop()

def left():
    client = Drone()
    client.left()
    time.sleep(5)
    client.leftstop()

def m1():
    client = Drone()
    client.m1()
    time.sleep(5)
    client.m1stop()

def m2():
    client = Drone()
    client.m2()
    time.sleep(5)
    client.m2stop()

def m3():
    client = Drone()
    client.m3()
    time.sleep(5)
    client.m3stop()

def m4():
    client = Drone()
    client.m4()
    time.sleep(5)
    client.m4stop()


def chatgpt(message):
    import openai

    # Set up the OpenAI API client
```

```
        openai.api_key = "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"

        # Set up the model and prompt
        model_engine = "text-davinci-003"
        # prompt = "Can you provide information about drones and their capabilities? : " +
message
        prompt=message
        print(prompt)
        # Generate a response
        completion = openai.Completion.create(
            engine=model_engine,




            prompt=prompt,
            max_tokens=1024,
            n=1,
            stop=None,
            temperature=0.5,
        )

        response = completion.choices[0].text.strip()
        if response.startswith('\n'):
            response = response[1:]
        return response


    def on_message(message: Message):
        if message.message_type == "chat_message":
            m = ChatMessage(message)
        elif message.message_type == "login_message":
            m = ft.Text(message.text, italic=True, color=ft.colors.BLACK45, size=12)
        chat.controls.append(m)
        page.update()

    page.pubsub.subscribe(on_message)

    # A dialog asking for a user display name
    join_user_name = ft.TextField(
        label="Enter your name to join the chat",
        autofocus=True,
        on_submit=join_chat_click,
    )
    page.dialog = ft.AlertDialog(
        open=True,
        modal=True,
        title=ft.Text("Welcome!"),
        content=ft.Column([join_user_name], width=300, height=70, tight=True),
        actions=[ft.ElevatedButton(text="Join chat", on_click=join_chat_click)],
        actions_alignment="end",
    )
```

```
    # Chat messages
    chat = ft.ListView(
        expand=True,
        spacing=10,
        auto_scroll=True,
    )

    # A new message entry form
    new_message = ft.TextField(
        hint_text="Write a message...",
        autofocus=True,
        shift_enter=True,
        min_lines=1,
        max_lines=5,




        filled=True,
        expand=True,
        on_submit=send_message_click,
    )

    # Add everything to the page
    page.add(
        ft.Container(
            content=chat,
            border=ft.border.all(1, ft.colors.OUTLINE),
            border_radius=5,
            padding=10,
            expand=True,
        ),
        ft.Row(
            [
                new_message,
                ft.IconButton(
                    icon=ft.icons.SEND_ROUNDED,
                    tooltip="Send message",
                    on_click=send_message_click,
                ),
            ]
        ),
    )

ft.app(port=8550, target=main, view=ft.WEB_BROWSER)
# ft.app(target=main)
```

stt.py

```
import speech_recognition as sr
from tts import tts

def stt():
    r = sr.Recognizer()
    while True:
        with sr.Microphone(device_index=2) as source:
            audio = r.listen(source)
            try:
                text = r.recognize_google(audio)
                print(text)
            except:
                tts('Couldnt Recognize your voice')
                return None
            return text
```

tts.py

```
from gtts import gTTS
import pyttsx3

def tts(text):
    # initialize Text-to-speech engine
    engine = pyttsx3.init()
    voices = engine.getProperty('voices')
    engine.setProperty("rate", 178)
    engine.setProperty('voice', voices[1].id) #changing index changes voices but ony 0 and 1 are
working here
    # convert this text to speech
    engine.say(text)
    # play the speech
    engine.runAndWait()
```

CLOUD BACKEND

Dynamodb_contact_us.py

```
def add_to_dynamodb_contact_us(name, email, message):
    import boto3
    import uuid
    id = str(uuid.uuid4())
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('STAMP_Contact_Us')
    item = {
        'ID': id,
        'name': name,
        'email': email,
        'message': message
    }
    response = table.put_item(Item=item)
    print(response)
```

dynamodb.py

```
def upload_dynamodb_details():
    import boto3
    import json

    # Create a client for DynamoDB
    dynamodb = boto3.client('dynamodb')

    # Open the JSON file and load its contents as a Python object
    with open('STAMP\stamp\src\components\Security\details.json') as f:
        data = json.load(f)

    # Iterate over the objects in the JSON file and add them to DynamoDB
    for item in data:
        response = dynamodb.put_item(
            TableName='STAMP',
            Item={
                'ID': {'S': item['id']},
                'first_name': {'S': item['first_name']},
                'last_name': {'S': item['last_name']},
                'email': {'S': item['email']},
                'image': {'S': item['image']}
            }
        )
        print(response)
```

s3.py

```
import boto3

def upload_s3(file_path):
    # Set up a client for S3
    s3 = boto3.client('s3')

    # Set up the name of the bucket and the name of the file you want to upload
    bucket_name = 'bucket-name
    file_name = file_path.split("/")[-1]

    # Use the put_object method to upload the file to S3
    with open(file_path, "rb") as f:
        s3.upload_fileobj(f, bucket_name, file_name)
```

sns_publish.py

```
import boto3

# create an SNS client
sns = boto3.client('sns')

# set the topic ARN for your SNS topic
topic_arn = 'arn:aws:sns:ap-northeast-1:109417029150:STAMP'

# get the list of email addresses subscribed to the SNS topic
response = sns.list_subscriptions_by_topic(TopicArn=topic_arn)
subscription_list = response['Subscriptions']
email_list = [subscription['Endpoint'] for subscription in subscription_list]

# read the email list from S3
bucket_name = 's3stamp'
file_key = 'emails.txt'
s3 = boto3.resource('s3')
bucket = s3.Bucket(bucket_name)
obj = bucket.Object(file_key)
all_email_list = obj.get()['Body'].read().decode('utf-8').splitlines()

# send a message to each subscribed email address
for email in email_list:
    if email in all_email_list:
        message = 'STAMP Welcome'
        sns.publish(TopicArn=topic_arn, Message=message, Subject='STAMP',
MessageAttributes={
            'email': {
                'DataType': 'String',
                'StringValue': email
            }

        })
        print(f'Sent message to {email}')
```

```
    else:
        print(f'{email} is not in the email list')
```

sns_subscribe.py

```
def sns_subscribe(email):
    import boto3

    # create an SNS client
    sns = boto3.client('sns')

    # set the topic ARN for your SNS topic
    topic_arn = 'arn:aws:sns:ap-northeast-1:109417029150:STAMP'

    # subscribe the email address to the SNS topic
    sns.subscribe(TopicArn=topic_arn, Protocol='email', Endpoint=email)
```

data_encryption.go

```
package main

import (
    "crypto/aes"
    "crypto/cipher"
    "crypto/rand"
    "encoding/json"
    "fmt"
    "io/ioutil"
    "os"
)
func main() {
    // Read the JSON data from the file
    jsonFile, err := os.Open("details.json")
    if err != nil {
        fmt.Println(err)
    }
    defer jsonFile.Close()
    byteValue, _ := ioutil.ReadAll(jsonFile)
    jsonData := []byte(byteValue)
    key := make([]byte, 32)
    if _, err := rand.Read(key); err != nil {
        fmt.Println(err)
    }
    iv := make([]byte, aes.BlockSize)
    if _, err := rand.Read(iv); err != nil {
        fmt.Println(err)
    }

    // Use the key and IV to create a CBC cipher
    block, err := aes.NewCipher(key)
    if err != nil {
        fmt.Println(err)
    }

    stream := cipher.NewCTR(block, iv)

    // Encrypt the JSON data using the cipher
    encryptedData := make([]byte, len(jsonData))
    stream.XORKeyStream(encryptedData, jsonData)
    encryptedFile, err := os.Create("encrypted_data.json")
    if err != nil {
        fmt.Println(err)
    }
    defer encryptedFile.Close()
    encoder := json.NewEncoder(encryptedFile)
    if err := encoder.Encode(encryptedData); err != nil {
        fmt.Println(err)
    }
}
```

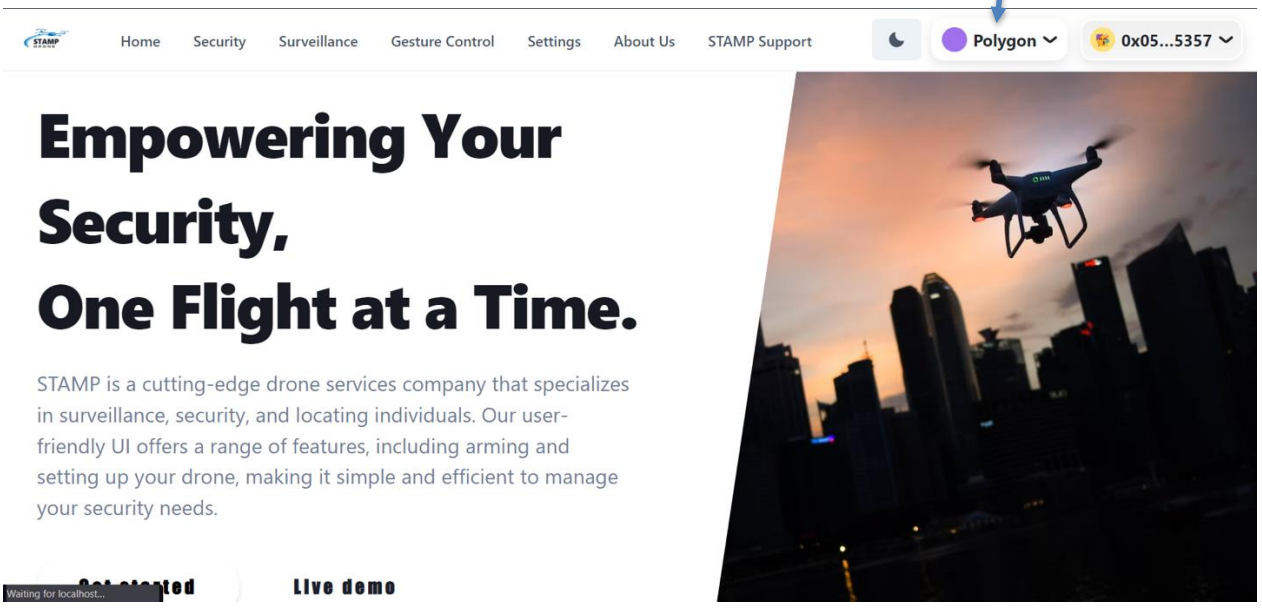
data_decryption.go

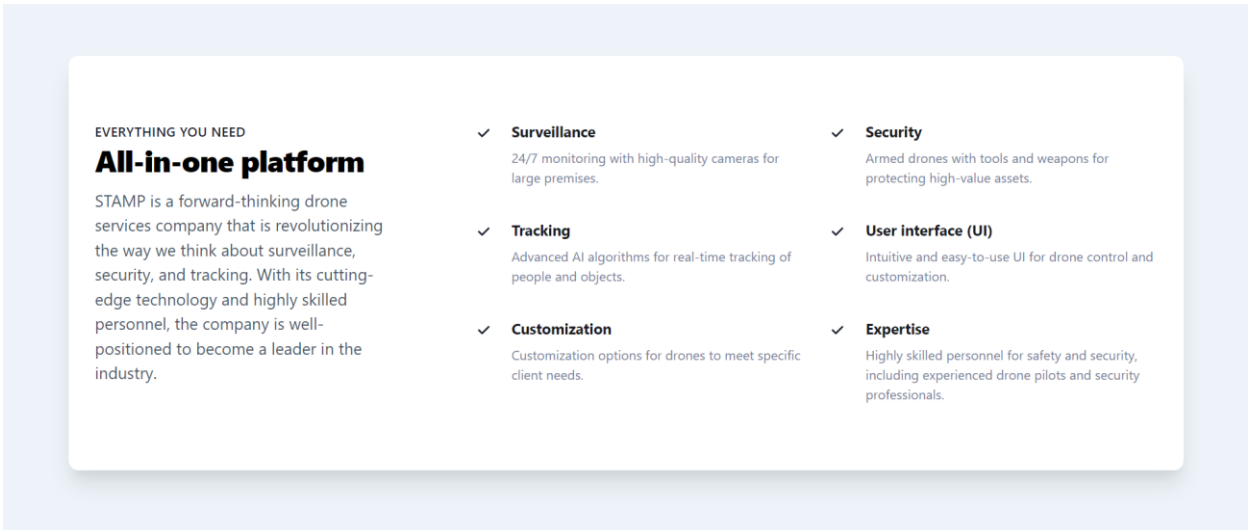
```
package main

import (
	"crypto/aes"
	"crypto/cipher"
	"crypto/rand"
	"encoding/json"
	"fmt"
	"io"
	"os"
)

func main() {
	// Read the encrypted JSON data from the file
	encryptedFile, err := os.Open("encrypted_data.json")
	if err != nil {
		fmt.Println(err)
	}
	defer encryptedFile.Close()
	var encryptedData []byte
	if err := json.NewDecoder(encryptedFile).Decode(&encryptedData); err != nil {
		fmt.Println(err)
	}
	key := make([]byte, 32)
	if _, err := io.ReadFull(rand.Reader, key); err != nil {
		fmt.Println(err)
	}
	iv := make([]byte, aes.BlockSize)
	if _, err := io.ReadFull(rand.Reader, iv); err != nil {
		fmt.Println(err)
	}
	block, err := aes.NewCipher(key)
	if err != nil {
		fmt.Println(err)
	}
	stream := cipher.NewCTR(block, iv)
	decryptedData := make([]byte, len(encryptedData))
	stream.XORKeyStream(decryptedData, encryptedData)
	decryptedFile, err := os.Create("decrypted_data.json")
	if err != nil {
		fmt.Println(err)
	}
	defer decryptedFile.Close()

	if _, err := decryptedFile.Write(decryptedData); err != nil {
		fmt.Println(err)
	}
}
```

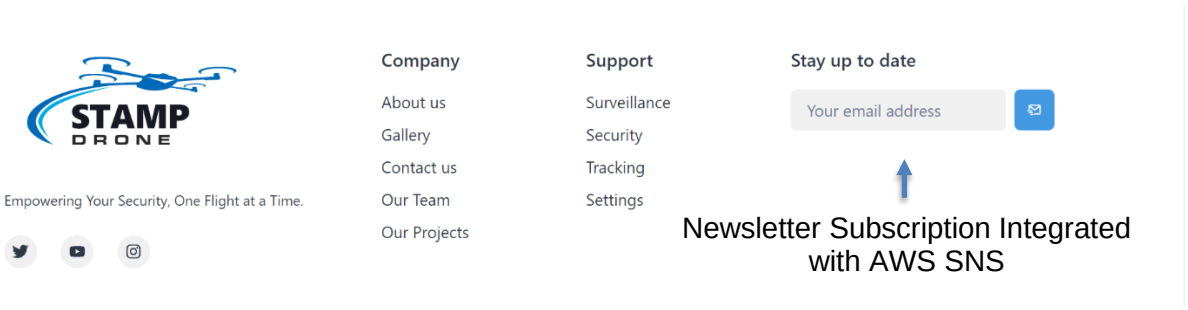
SCREEN SHOTS

HOME

Metamask Wallet Connection

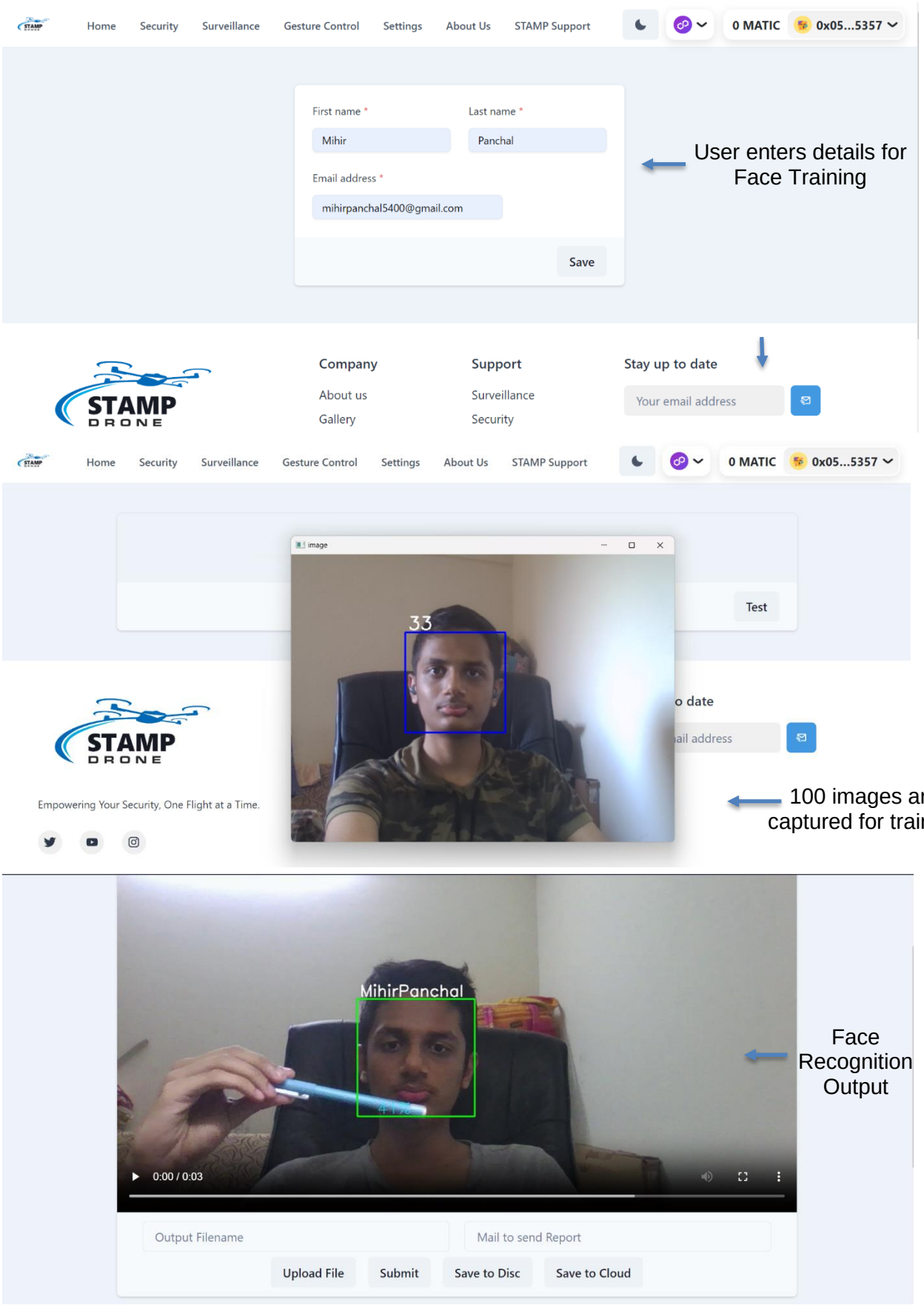




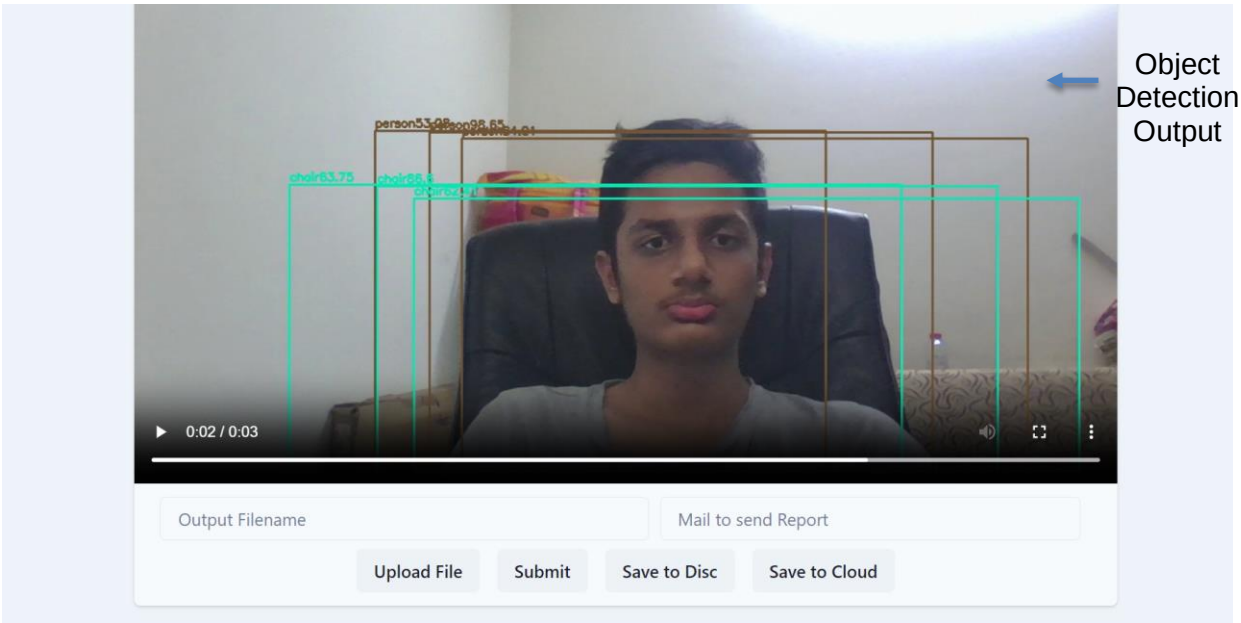


Newsletter Subscription Integrated with AWS SNS

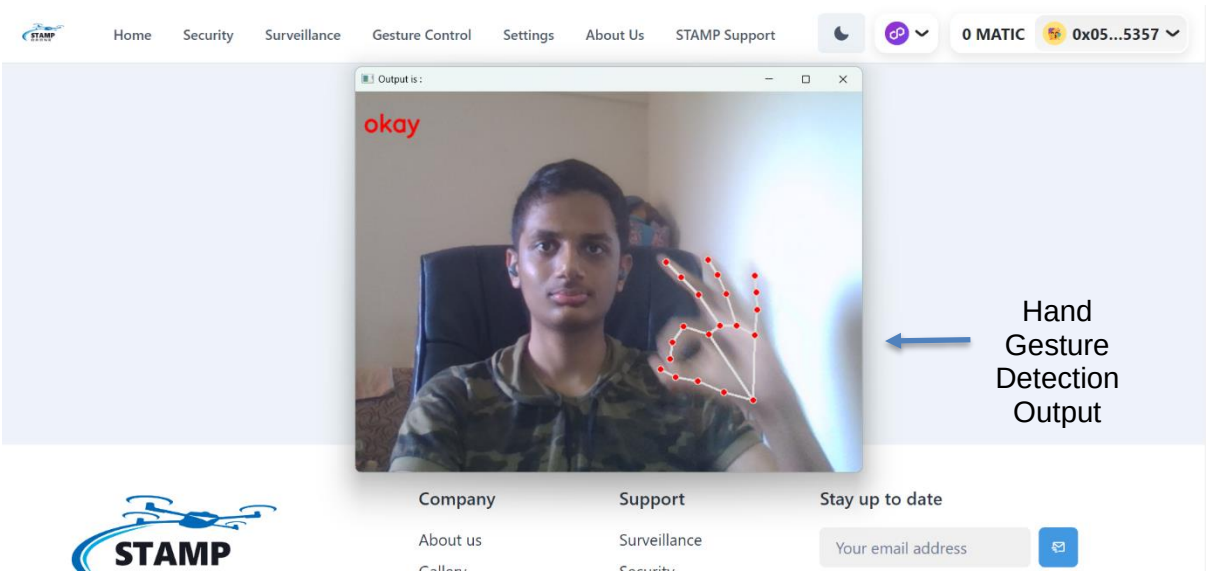
SECURITY


Home
Security
Surveillance
Gesture Control
Settings
About Us
STAMP Support
0 MATIC
0x05...5357
First name *
Mihir
Last name *
Panchal
Email address *
mihirpanchal5400@gmail.com
Save
User enters details for Face Training
STAMP DRONE
Company
About us
Gallery
Support
Surveillance
Security
Stay up to date
Your email address
Test
image
33
Empowering Your Security, One Flight at a Time.
100 images are captured for training
MihirPanchal
0:00 / 0:03
Face Recognition Output
Output Filename
Mail to send Report
Upload File
Submit
Save to Disc
Save to Cloud

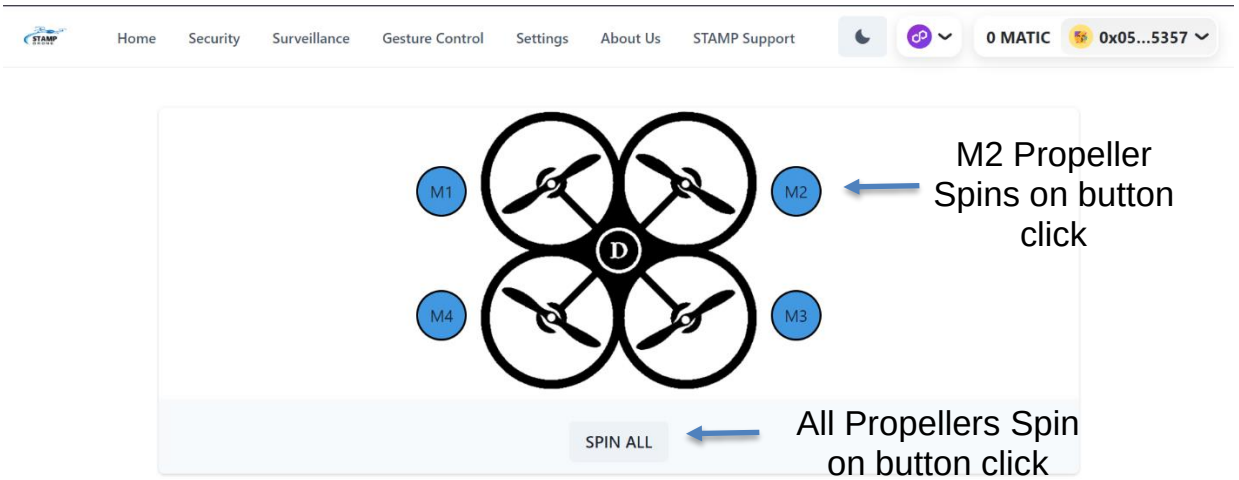
SURVEILLANCE

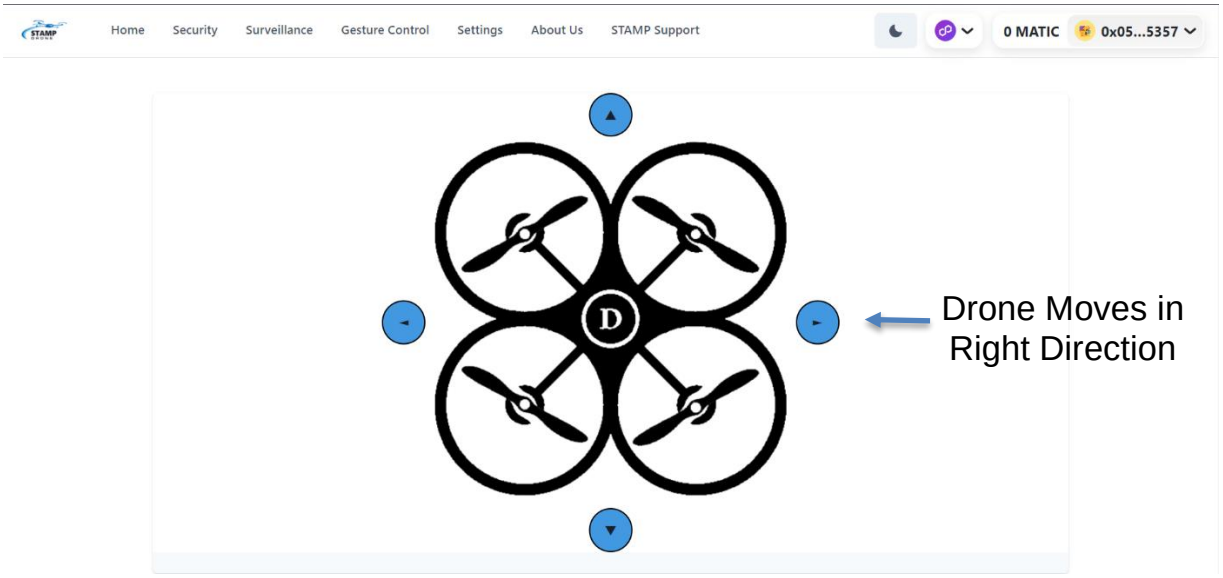


GESTURE CONTROL

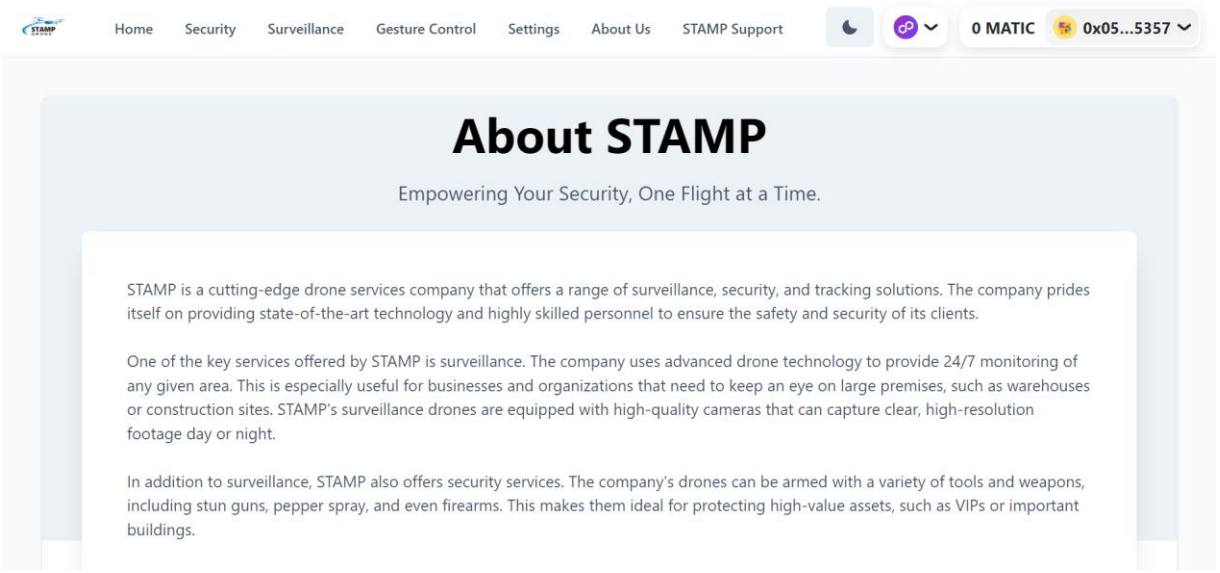


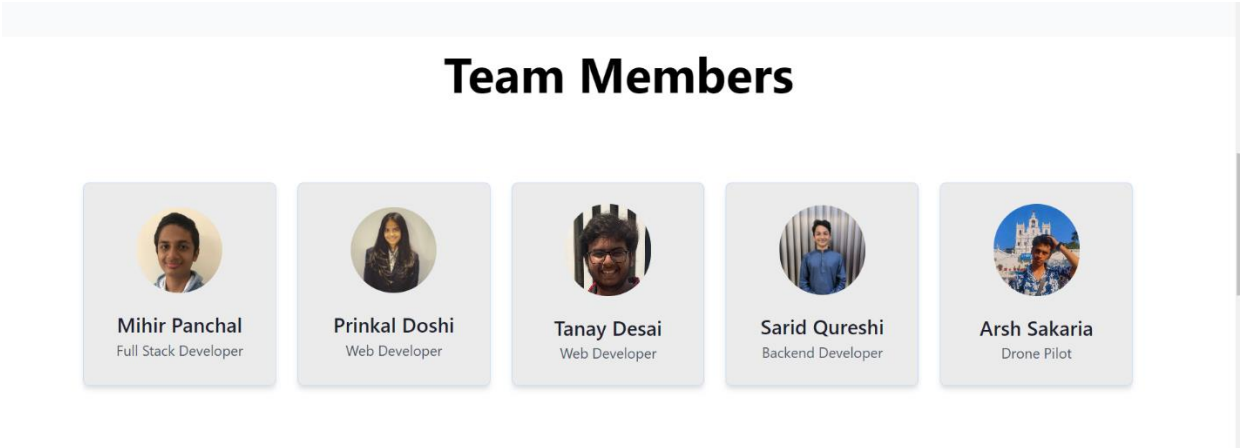
SETTINGS

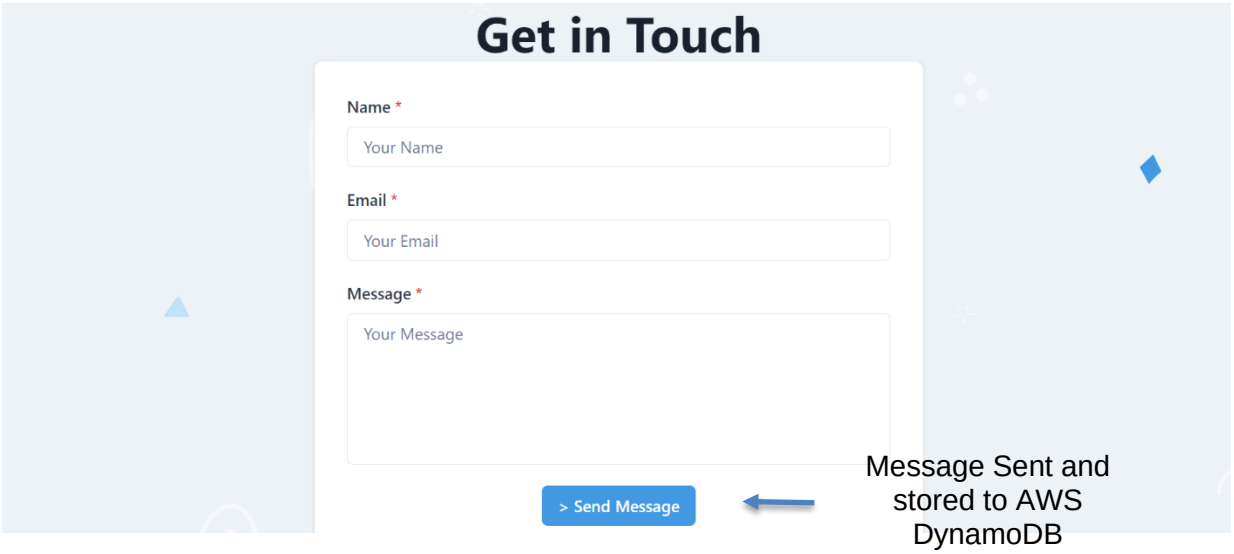

Home
Security
Surveillance
Gesture Control
Settings
About Us
STAMP Support
0 MATIC
0x05...5357
M1
M2
M3
M4
D
M2 Propeller Spins on button click
SPIN ALL
All Propellers Spin on button click


Home
Security
Surveillance
Gesture Control
Settings
About Us
STAMP Support
0 MATIC
0x05...5357
D
Drone Moves in Right Direction

ABOUT US

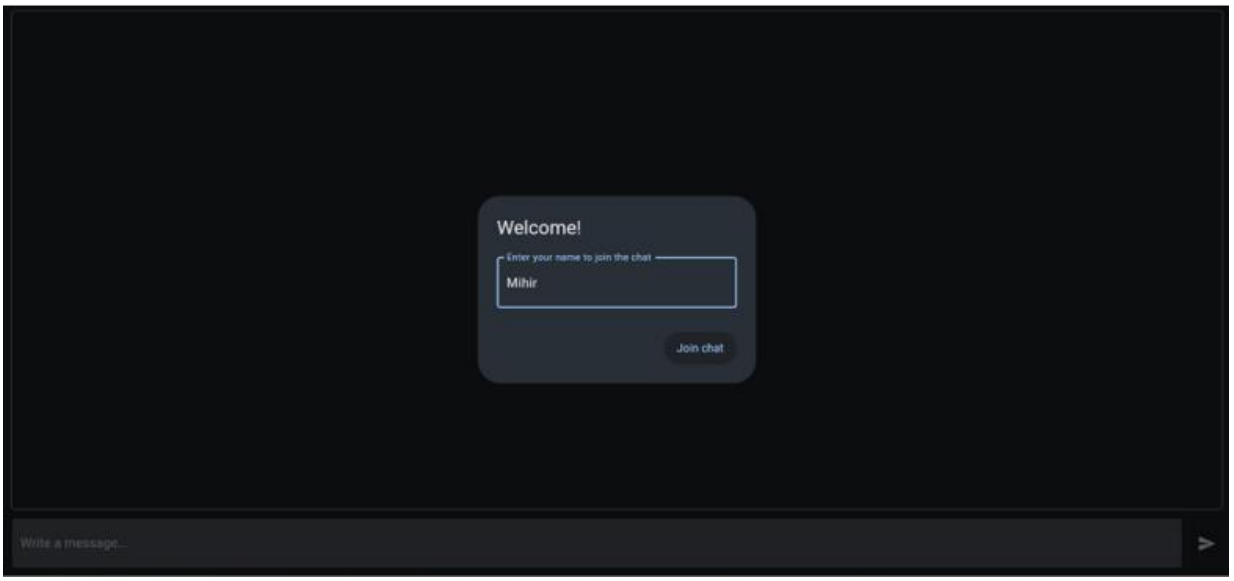


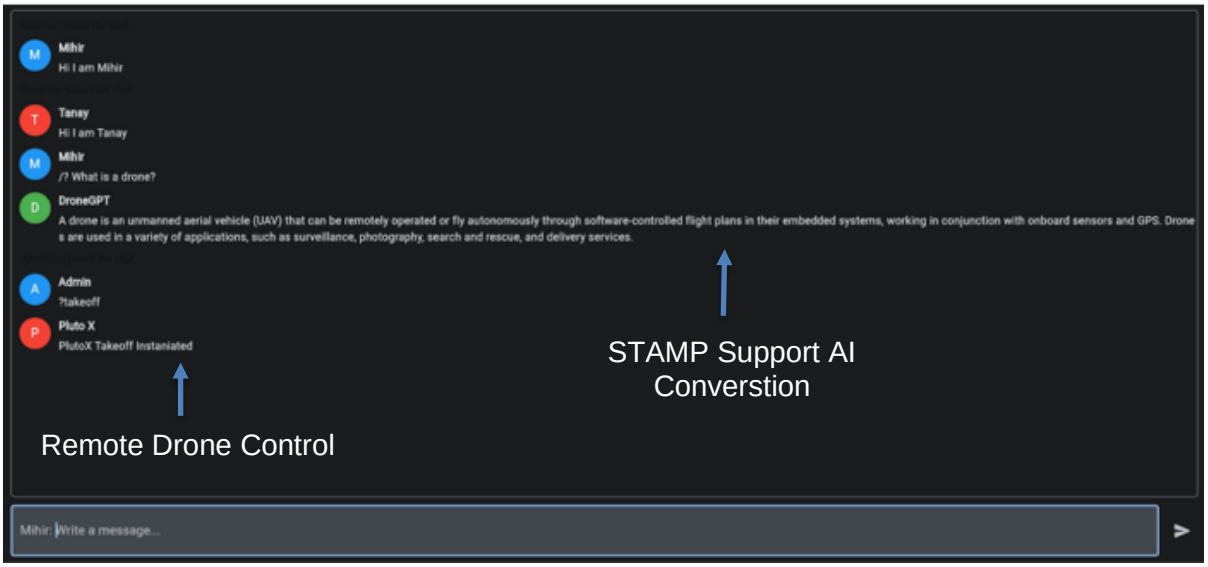




Message Sent and stored to AWS DynamoDB

STAMP SUPPORT





USER INSTALLATION GUIDE

USER INSTALLATION GUIDE

CLONE THE REPOSITORY

To clone this repository to your local machine, follow these steps:

1.Navigate to the directory where you want to store the repository in your terminal or command prompt.
2.Run the following command: git clone https://github.com/MihirRajeshPanchal/Final-Year-Project- Drone.git
3.Once the repository has been cloned, navigate into the project directory using the cd command:
cd Final-Year-Project-Drone

REVIEW AND EDIT

You can now view and customize the project files on your local machine according to your requirements.

INSTALLATION STEPS

Frontend :
From File Path Final-Year-Project- Drone\STAMP\stamp
npm install

Backend :
From File Path Final-Year-Project-Drone
pip install -r requirements.txt

RUN THE PROJECT

Frontend :

From File Path Final-Year-Project-Drone\STAMP\stamp
npm start

Backend :

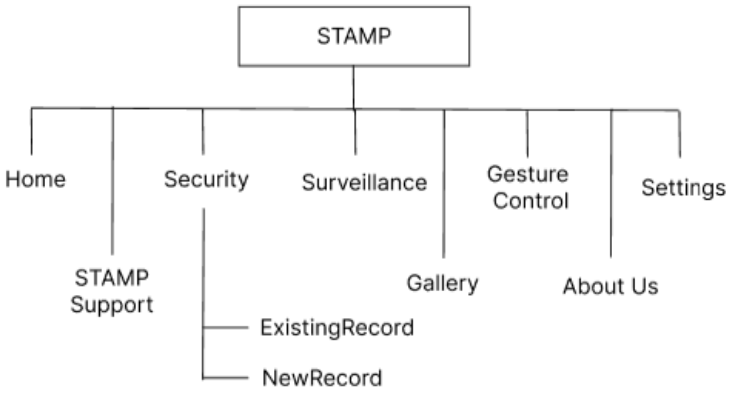
From File Path Final-Year-Project-Drone
py server.py

Stamp Support :

From File Path Final-Year-Project-Drone\STAMP_Support
py stamp_support.py

USER MANUAL

USER MANUAL



STAMP Support

Use following prompts to use STAMP Support to communicate to Drone through your web Console :

?<your-command>

?m1 m2 m3 m4
This can be used to check the functioning of the motors of drone. Each motor is represented by ID m1,m2,m3,4.
?spinall
This prompt spins all the motor at balanced levels
?left ?right ?backward ?right
This prompts help in determining if the thrust of the drone with all the direction of drone is balanced and can help in calibrating.

Security

This section is where you can upload the video feed recorded from the drone and run the Haar Cascade facial recognition algorithm to find a specific match from your stored dataset.

1.To use this function user has to upload the video file.
2.Specify outfit file name (report name) in “Output Filename”.
3.Enter Email ID to which user wants to send the report in “Mail to send Report”.
4.Then click on submit.

This will send a report to Email ID mentioned and also provides an option to save it to the cloud using “Save to Cloud’’ button or local directory using “Save to Disc” button.

This also navigates user to 2 option:

ExistingRecord
This opens up the list of all the trained dataset of individuals with the Machine Learning Model is trained with. User can navigate and view details of the profile cards.
NewRecord
This opens up an option to train Machine Learning Model with new Dataset and store it.
It revolves around taking multiPle images of the subject which help in better recognition of overall recognition algorithms.
1.Enter user details as prompted.
2.Look at the camera and stay in the visible screen(100 images captured for training).

Surveillance

This section is where you can upload the video feed recorded from the drone and run the YOLO object detection algorithm.

1.To use this function user has to upload the video file.
2.Specify outfit file name (report name) in “Output Filename”.
3.Enter Email ID to which user wants to send the report in “Mail to send Report”.
4.Then click on submit.

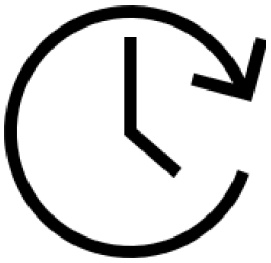
This will send a report to Email ID mentioned and also provides an option to save it to the cloud using “Save to Cloud’’ button or local directory using “Save to Disc” button.

Settings

This page allows user to do operations on drone through User Interface:
User can click on individual button for all 4 motor to spin M1,M2,M3,M4 OR Can directly click on “Spin All” button to spin all the motors at the same time.
There is also an option to move drone in particular direction using button controls provide; just click on desired movement of direction and Web Console takes care of the rest using MSP Protocol.

About Us

About us provides information on team members of Security and Surveillance Drone Project.
To Get in Touch a user can directly enter their Name, Email and Message and Message will directly be forwarded to all Team Members using AWS DynamoDB



FUTURE SCOPE

FUTURE SCOPE

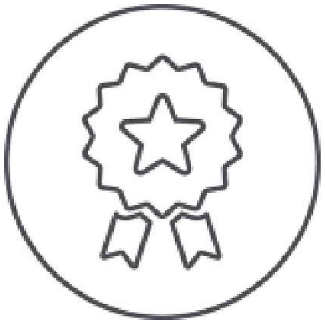
A project never ends completely; it either gives birth to other projects carrying pieces of it ahead onto a next level or is put on a pause in hopes of someone will return to upgrade the project. In a similar way, the Security and Surveillance Drone project has a wide scope that it can be upgraded to. Some of the scopes include:

1. Enhanced Connectivity: As drones become more advanced, they can be equipped with enhanced connectivity capabilities, such as 5G, enabling them to transmit data and video in real-time. This can help emergency responders quickly assess the situation and take action.

2. Increased Range: Drones with increased range can cover larger areas, making them more effective in search and rescue operations or disaster response efforts. This can be achieved through the development of more powerful batteries, advanced propulsion systems, or other innovative technologies.

3. Improved Durability: Emergency situations can be unpredictable and dangerous, and drones must be able to withstand harsh environments and potential collisions. Future developments in drone materials and construction could improve their durability and resilience, enabling them to perform more effectively in challenging conditions.

Overall, the future scope for Security and Surveillance Drones is vast and exciting. With continued advancements in technology, these drones will become even more effective in emergency situations, providing critical support to emergency responders and helping to save lives.



ADVANTAGES AND LIMITATIONS

ADVANTAGES

1. Quick response time: Security and Surveillance Drones can be deployed quickly, enabling emergency responders to assess the situation and take action in real-time.

2. Cost-effectiveness: Drones can cover large areas quickly and efficiently, reducing the need for extensive ground-based search efforts or traditional surveying methods, ultimately reducing the overall cost of emergency response efforts.

3. Accuracy: Drones are equipped with advanced technologies such as facial recognition, and high-resolution cameras, enabling them to provide accurate data about the location of missing individuals and the extent of damage during calamities.

4. Flexibility: Security and Surveillance Drones can be deployed in a wide range of emergency situations, from natural disasters to terrorist attacks, providing critical support to emergency responders.

LIMITATIONS

1. Limited Battery Life: Most drones have a limited battery life, which can impact their range and effectiveness.

2. Weather Dependence: Weather conditions, such as high winds or heavy rain, can impact the ability of drones to operate effectively, limiting their use during emergency situations.

3. Limited Payload Capacity: Drones have limited payload capacity, limiting their ability to carry heavy equipment or rescue supplies during emergency response efforts.

CONCLUSION

CONCLUSION

The use of Security and Surveillance Drones for finding lost individuals in public gatherings and analysing the depth of effects during calamities has proven to be a game-changer in emergency response efforts. These drones, equipped with advanced technologies, enable quick response times, cost-effective operations, accurate data analysis, and unmatched flexibility, making them invaluable tools in search and rescue operations and disaster response efforts. By leveraging their capabilities, lives can be saved and the impact of emergency situations can be minimised.

The future scope for Security and Surveillance Drones is vast and promising, driven by ongoing advancements in technology. Integration with artificial intelligence will enhance their capabilities, allowing for sophisticated data analysis, automated decision-making, and improved situational awareness. Enhanced connectivity will enable real-time data transmission, facilitating seamless communication between drones and ground control stations. Additionally, extended range capabilities will expand the operational reach of these drones, enabling them to cover larger areas and reach remote locations more efficiently. Improved durability will ensure their resilience in challenging environmental conditions, making them reliable assets in various emergency scenarios.

As technology continues to evolve, the potential for Security and Surveillance Drones to become even more effective in emergency situations is immense. Their ability to provide aerial views, access hard-to-reach areas, and gather critical information in real-time offers a significant advantage in emergency response efforts. The data collected by these drones, such as video footage and sensor readings, can be analysed to assess the severity of a situation, identify potential hazards, and aid in decision-making for resource allocation and strategic planning.

Moreover, Security and Surveillance Drones have broader applications beyond emergency response. They can be utilised in monitoring public events, enhancing security measures, and supporting law enforcement agencies in ensuring public safety. Their versatility and adaptability make them valuable assets in various domains, including search and rescue, disaster management, surveillance, and infrastructure inspections.

In conclusion, Security and Surveillance Drones have shown immense potential in improving emergency response efforts. By utilising the latest technologies and providing critical data to emergency responders, these drones can help save lives and minimise the impact of emergency situations.

CERTIFICATES

AICTE-SPICES & IDEA LAB. SPONSORED: IDEATION AND PROJECT EXHIBITION EVENT
INNOVATION CUP -23

CERTIFICATE OF PARTICIPATION

Awarded to

ARSHKUMAR SAKARIA

Shri Bhagubai Mafatlal Polytechnic, Mumbai

for actively participating and successfully completing Evaluation Round 2 of **S.P.I.T Innovation Cup-23: An AICTE-SPICES & AICTE IDEA LAB sponsored event** conducted on **8th April 2023** by Electronics Student Association (ESA), Electronics and Electronics & Telecommunication Engineering Department & Institute of Electrical and Electronics Engineers (IEEE) of Bhartiya Vidya Bhavan's, Sardar Patel Institute of Technology, Munshi Nagar, Andheri (W), Mumbai-400058, Maharashtra.

DR. B.N CHAUDHARI PRINCIPAL	DR Y.S.RAO DEAN ACADEMICS,R&D	DR. D.R.KALBANDE CO-ORDINATOR (AICTE IDEA LAB)	DR.P.V.KASAMBE CO-ORDINATOR (AICTE SPICES)

8/4/2023

A Project by Rotary Club of Bombay Worli District 3141 — Baldev Sharma Marshalls Foundation

marshalls India's No.1 Wallcoverings presents CiiA Innovation Competition, Exhibition & Awards

Certificate Of Merit

Awarded to MIHIR RAJESH PANCHAL

of Shri Bhagubhai Mafatlal Polytechnic for Presenting an IDEA

for an Innovation Project Aerial Surveillance & Agricultural Drone

at the **CiiA-2 Innovation Exhibition** held on 1st - 3rd **February 2023**

at Nehru Centre, Worli, Mumbai.

OFFICIAL PARTNERS

MU Ideas · SOMAIYA VIDYAVIHAR UNIVERSITY · D Y PATIL UNIVERSITY

Rtn. Sachin Singhvi
President, Rotary Club of Bombay Worli 2022-23

Baldevkrishan Sharma
Chairman -CiiA & BS Marshalls Foundation

Workshop on Drone Development

wecan EDUCATION · SVKM · DRONA AVIATION · Dalvik Apps

Certificate of Participation

Ashkumar Sakaria

has successfully completed the Workshop conducted on **"Drone Development"** by **Dalvik Apps** in association with **WeCan Educational Organisation** & **Drona Aviation Private Limited** at **Shri Bhagubhai Mafatlal Polytechnic, Mumbai** on **10th & 11th November 2022**

Mr. Gaurav Singh	Mr. Amit Thool	Mrs. Anita A Kulkarni	Dr.Mohd. Zafar Shaikh
WeCan Educational Organisation	Dalvik Apps	HOD, Electronics SBMP	Principal SBMP

SOMAIYA VIDYAVIHAR UNIVERSITY

K.J. SOMAIYA COLLEGE OF ENGINEERING

Presents

TECHNOVALLEY : PRAKALPA'23

National Level Paper Presentation and Working Model Project Competition

CERTIFICATE

This is to certify that

Mavia Qureshi

~~has secured~~ participated ~~prize~~ in Paper Presentation/Working Model Project in the category UG Project in Prakalpa '23 held on 13th of April.

Rajvardhan Patil	Prof. Sushma Kadge	Prof. Abhishek Bhaduria	Dr. Shubha Pandit
Chairperson	Faculty Advisor	Faculty Advisor	Principal

HUST

SOMAIYA
VIDYAVIHAR UNIVERSITY

K.J. SOMAIYA COLLEGE OF ENGINEERING

Presents

TECHNOVALLEY : PRAKALPA'23

National Level Paper Presentation and Working Model Project Competition

CERTIFICATE

This is to certify that

Tanay Desai

~~has secured~~ participated ~~prize~~ in ~~Paper Presentation~~/Working Model Project in the

category UG Project in Prakalpa '23 held on 13th of April.

Rajvardhan Patil	Prof. Sushma Kadge	Prof. Abhishek Bhaduria	Dr. Shubha Pandit
Chairperson	Faculty Advisor	Faculty Advisor	Principal

SPICES ESA IIC S.P.I.T. IDEALab

AICTE-SPICES & IDEA LAB SPONSORED: IDEATION AND PROJECT EXHIBITION EVENT

INNOVATION CUP -23

CERTIFICATE OF PARTICIPATION

Awarded to

TANAY DESAI

Shri Bhagubai Mafatlal Polytechnic, Mumbai

for actively participating and successfully completing Evaluation Round 2 of **S.P.I.T Innovation Cup-23: An AICTE-SPICES & AICTE IDEA LAB sponsored event** conducted on **8th April 2023** by Electronics Student Association (ESA), Electronics and Electronics & Telecommunication Engineering Department & Institute of Electrical and Electronics Engineers (IEEE) of Bhartiya Vidya Bhavan's, Sardar Patel Institute of Technology, Munshi Nagar, Andheri (W), Mumbai-400058, Maharashtra.

DR. B.N CHAUDHARI	DR Y.S.RAO	DR. D.R.KALBANDE	DR.P.V.KASAMBE
PRINCIPAL	DEAN ACADEMICS,R&D	CO-ORDINATOR (AICTE IDEA LAB)	CO-ORDINATOR (AICTE SPICES)

8/4/2023

SOMAIYA
VIDYAVIHAR UNIVERSITY

K.J. SOMAIYA COLLEGE OF ENGINEERING

Presents

TECHNOVALLEY : PRAKALPA'23

National Level Paper Presentation and Working Model Project Competition

CERTIFICATE

This is to certify that

Mihir Panchal

~~has secured~~ participated ~~prize in Paper Presentation~~/Working Model Project in the

category UG Project in Prakalpa '23 held on 13th of April.

Rajvardhan Patil	Prof. Sushma Kadge	Prof. Abhishek Bhaduria	Dr. Shubha Pandit
Chairperson	Faculty Advisor	Faculty Advisor	Principal

Somaiya TRUST

SPICES ESA IIC S.P.I.T. IDEALab

AICTE-SPICES & IDEA LAB SPONSORED: IDEATION AND PROJECT EXHIBITION EVENT

INNOVATION CUP -23

CERTIFICATE OF PARTICIPATION

Awarded to

MIHIR PANCHAL

Shri Bhagubai Mafatlal Polytechnic, Mumbai

for actively participating and successfully completing Evaluation Round 2 of **S.P.I.T Innovation Cup-23: An AICTE-SPICES & AICTE IDEA LAB sponsored event** conducted on **8th April 2023** by Electronics Student Association (ESA), Electronics and Electronics & Telecommunication Engineering Department & Institute of Electrical and Electronics Engineers (IEEE) of Bhartiya Vidya Bhavan's, Sardar Patel Institute of Technology, Munshi Nagar, Andheri (W), Mumbai-400058, Maharashtra.

DR. B.N CHAUDHARI	DR Y.S.RAO	DR. B.R.KALBANDE	DR.P.V.KASAMBE
PRINCIPAL	DEAN ACADEMICS,R&D	CO-ORDINATOR (AICTE IDEA LAB)	CO-ORDINATOR (AICTE SPICES)

8/4/2023

AICTE-SPICES & IDEA LAB. SPONSORED: IDEATION AND PROJECT EXHIBITION EVENT
INNOVATION CUP -23

CERTIFICATE OF PARTICIPATION

Awarded to

PRINKAL DOSHI

Shri Bhagubai Mafatlal Polytechnic, Mumbai

for actively participating and successfully completing Evaluation Round 2 of **S.P.I.T Innovation Cup-23: An AICTE-SPICES & AICTE IDEA LAB sponsored event** conducted on **8th April 2023** by Electronics Student Association (ESA), Electronics and Electronics & Telecommunication Engineering Department & Institute of Electrical and Electronics Engineers (IEEE) of Bhartiya Vidya Bhavan's, Sardar Patel Institute of Technology, Munshi Nagar, Andheri (W), Mumbai-400058, Maharashtra.

DR. B.N CHAUDHARI	DR Y.S.RAO	DR. B.R.KALBANDE	DR.P.V.KASAMBE
PRINCIPAL	DEAN ACADEMICS,R&D	CO-ORDINATOR (AICTE IDEA LAB)	CO-ORDINATOR (AICTE SPICES)

8/4/2023

A Project by Rotary — BALDEV SHARMA MARSHALLS FOUNDATION

marshalls India's No.1 Wallcoverings presents CiiA Innovation Competition, Exhibition & Awards

Certificate Of Merit

Awarded to PRINKAL DOSHI

of Shri Bhagubhai Mafatlal Polytechnic for Presenting an IDEA

for an Innovation Project Aerial Surveillance & Agricultural Drone

at the **CiiA-2 Innovation Exhibition** held on **1st - 3rd February 2023**

at Nehru Centre, Worli, Mumbai.

OFFICIAL PARTNERS

Rtn. Sachin Singhvi
President, Rotary Club of Bombay Worli 2022-23

Baldevkrishan Sharma
Chairman -CiiA & BS Marshalls Foundation

SOMAIYA VIDYAVIHAR UNIVERSITY

K.J. SOMAIYA COLLEGE OF ENGINEERING

Presents

TECHNOVALLEY : PRAKALPA'23

National Level Paper Presentation and Working Model Project Competition

CERTIFICATE

This is to certify that

Prinkal Doshi

~~has secured~~ participated ~~prize~~ in ~~Paper Presentation~~/Working Model Project in the category UG Project in Prakalpa '23 held on 13th of April.

Rajvardhan Patil	Prof. Sushma Kadge	Prof. Abhishek Bhaduria	Dr. Shubha Pandit
Chairperson	Faculty Advisor	Faculty Advisor	Principal

Somaiya TRUST

REFERENCES

REFERENCES

1. Node.js:
https://nodejs.org/

2. Chakra UI:
https://chakra-ui.com/

3. Flask:
https://flask.palletsprojects.com/

4. Pluto block drone programming:
https://create.dronaaviation.com/software/drone-programming/api-reference/introduction

5. YOLO algorithm for object detection:
Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement.
https://arxiv.org/abs/1804.02767

6. Haar cascade for facial recognition:
Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1, I-511.
https://ieeexplore.ieee.org/document/990517

7. Amazon Web Services
https://docs.aws.amazon.com/

8.Multi Wii Serial Protocol
http://www.multiwii.com/wiki/index.php?title=Multiwii_Serial_Protocol

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

As a team, we would like to express our heartfelt gratitude to the following individuals who have played a significant role in the successful completion of our final year project. Their unwavering support, guidance, and encouragement have been instrumental in our journey:

We extend our sincere appreciation to the Head of the IT Department, Mrs. N.G.Kadukar for her constant support and encouragement. Her vision and leadership have provided us with a conducive environment to pursue our project and have been a source of inspiration throughout.

We are grateful to the faculty members and staff of our department for their dedication and commitment. Their extensive knowledge, insightful feedback, and continuous assistance have been invaluable in shaping our project and enhancing our understanding of the subject matter.

We would like to acknowledge the assistance provided by the lab assistants who tirelessly supported us during the experimental phase of our project. Their technical expertise, willingness to help, and troubleshooting skills have greatly contributed to the smooth execution of our experiments.

We would like to express our sincere gratitude to the Head of the Electronics and Telecommunication Department, Mrs. A.A.Kulkarni for their support and approval of our drone project. Their belief in our capabilities and the trust they placed in us allowed us to explore innovative ideas and develop a project that bridges the gap between IT and Electronics and Telecommunication Department.

Our deepest appreciation goes to our project mentor, Mr. A.B.Dongaonkar whose guidance and mentorship have been invaluable throughout this project. Their expertise, patience, and dedication to our growth have shaped our project and helped us overcome challenges along the way. We are grateful for their constant encouragement, insightful suggestions, and unwavering belief in our abilities.

We would also like to thank our friends, family, and loved ones for their understanding, encouragement, and support throughout this journey. Their belief in our capabilities has been a constant source of motivation.

Finally, we acknowledge the support and cooperation of all the individuals who have contributed directly or indirectly to the success of this project. We are grateful for the opportunity to work on this project and the invaluable lessons we have learned along the way.

Thank you all for being an integral part of our journey and for making this project a reality.