AN EFFICIENT MOBILE APPLICATION FOR KEEPING THE SCHOOL BUS SYSTEM INFORMED USING FLUTTER AND FACIAL RECOGNITION TECHNOLOGIES

Jiaqi Ji[1], Garret Washburn[2]

[1]Dr. TJ Owen's Gilroy Early College Academy, 5055 Santa Teresa Blvd, Gilroy, CA 95020
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

The technology currently used to monitor school buses and allow for effective communication within them is outdated and does not reflect the technological advances we've made as a society. Specifically, it was very common nowadays for parents to be uninformed about the status of their child on a school bus, and the student to be unaccounted for by the driver [2]. To solve this problem, this paper proposes the AI Smart Route mobile application as a solution. The app utilizes facial mesh recognition in order to take attendance and keep track of students getting on and off buses. This allows driver, school, and parents to be informed of whether a student is on the bus. The major technologies used to develop the AI Smart Route mobile include but are not limited too the Flutter framework for the mobile app development, Mediapipe for the facial mesh recognition, and AWS for the back end server hosting [3].To ensure the mobile application worked as intended, experiments were performed to ensure proper function. The AI Smart Route mobile app is an effective application that surpasses the current methods of taking bus attendance and keeping everyone involved informed of who is on the school bus.

KEYWORDS

AI , Facial Recognition, App, School Bus

1. INTRODUCTION

The problem identified in this research paper, the issue of an undocumented system of students coming on and off the bus, is one of importance as it affects the mindfulness of parents as well as the efficiency and reliability of the public school bus system. Bus drivers often feel overwhelmed with the amount of kids they are responsible for. Additionally, it has always been a concern for parents that send their kids to school on the bus of where the children currently are, as they have no sure way of knowing if they got on the bus and made it to school. Most families nowadays prefer not to send their kids on the bus, and prefer to drive their kids to school themselves, as they worry it is unsafe for their children[1]. Parents not wanting to send their kids to school on the bus is likely due to the uninformed nature of the school bus system; currently, the school bus system is designed to keep parents in the know. On top of all of this, given the nature of young students, keeping track of where they are at and if they are ok is one of society's top priorities [4]. As such, the need for a system to keep track of students attending school is very apparent. Overall, providing an assistance app for bus drivers to keep track of all of the students on the bus gives them peace of mind; and keeps parents informed and children on the bus documented and safe.

Additionally, this paper also includes evaluations of other methodologies of solving the same problem as identified within this paper. The methodologies evaluated include the industry standard Walkie-Talkies, the Bus Attendance mobile application, and the SBAS suite. When taking a look at Walkie-Talkies, it is apparent that there are quite a few issues with their use in keeping track of students and keeping everyone informed, as their range is limited and all communication is done through word of mouth. The Bus attendance app, while solving some of the issues of Walkie-Talkies, like better ability to keep track of individuals on the bus, doesn't provide a medium of communication with anyone outside of the bus. Lastly, the School Bus Attendance System seeked to solve all of these issues, however, didn't provide an effective and quick way to check students in for their attendance [5]. The AI Smart Route mobile application seeks to solve all of these issues prevalent in other methodologies by providing a quick and easy attendance taking system that uses student facial mesh to check in their attendance, and a mobile app that informs parents their child is on the bus.

The solution proposed in this paper to solve the issues identified in the public school bus system above is the AI Smart Route mobile application [6]. The AI Smart Route application remedies the common overwhelmedness that Bus Driver's usually face, as it includes an interface to take pictures of the student's entering the bus and logs their attendance using facial recognition AI. Once the student has had their picture taken and their attendance is logged, the parent is able to go into the app and see that their student has been logged in, giving the parents the peace of mind that their student is safely on the bus. Going further, the location and attendance of the student are kept track of the entire bus ride, keeping the student safe and giving the parents information they should have as right. The AI Smart Route App is an effective solution because it provides detailed information to all parties involved and keeps track of students effectively [7]. Compared to the alternative actively in use, walkie talkie radio comms to communicate with a headquarters of what student is getting on the bus, the AI Smart Route app is superior as it provides a list of students that should be on the bus and an easy way to log students attendance that doesn't involve interaction with a radio. Additionally, the AI Smart Route app keeps parents informed of their student's attendance which was not a practice employed previously. Overall, the AI Smart Route app is a more effective solution than previous methods of bus attendance, as it is more detailed and informative.

Within this paper, two differing experiments were conducted to evaluate the efficacy of the AI Smart Route mobile application. The experiments were designed to test the time efficiency of the back end server as well as the effectiveness of the facial mesh detection of the employed AI model. The first experiment, designed to test the average time for a request to the back end server to be processed and returned to the user, found that the average response time was around 3.14 seconds. This data was not necessarily surprising, as that is around the average time it takes for a normal back end server. However, the second experiment designed to test the capabilities of the facial mesh detection of the mediapipe library found that the facial meshes could be detected up to a 7 foot distance. This exceeded our expectations, as we originally predicted it would only work up to a distance of around 4 feet.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. How to Perform a Valid HTTPs Request

One of the major components of the AI Smart Route application is the uploadFileToServer background function that serves the purpose of receiving the user inputted image and uploading the file to the back-end server [8]. The first challenge we encountered during development was how to perform a valid https request considering the back-end server does not have a registered ssl certificate. To remedy this issue, we created a custom http client within the application that skips the ssl validation step during the http transaction. This solution works, and is not a security concern as the http requests are only performed to our registered back-end server.

2.2. The Facial Recognition Backend Server

Another major component that we encountered some challenges with was the facial recognition backend server which was written in Python and hosted on an AWS EC2 server instance [9]. The original Python library we used, face-recognition, proved to be too resource intensive for the AWS EC2 server, and as a result we needed to switch to a new library. Our solution, the mediapipe library, does functionally the same tasks, however, mediapipe performs all of its image processing tasks through the Google api. Meaning, all of the image processing is done off-site, greatly decreasing the backend resource consumption of the server.

2.3. The User Interface Pages

The last major component that we encountered some challenges with was the user interface pages within the mobile application. What we found difficult was less any one page, and more so adapting and learning how Flutter as a framework is used. However, there was one page, the busPage.dart, that we found particularly difficult due to the fact that there are a lot of moving parts in the page, including a lot of user visible widget design as well as backend functions for both the camera functionality as well as the back-end server connection functionality. We especially found Flutter’s diverse and abundant collection of widgets to be overwhelming, and had difficulty navigating the implementation of each widget.

3. SOLUTION

The encompassing structure for the AI Smart Route mobile application consists of three major components: the file upload to server functionality, the facial recognition back-end server, and the Flutter frontend mobile app design. Upon opening the app, the user is greeted by a splash screen that quickly moves into a menu screen. On the menu screen the user is given three possible directions they can follow in the shape of three image buttons. The first option, the Driver Page intended for the bus driver themselves, contains a list of all students that are on the drivers route as well as their attendance status and address [10]. The second option, the Parent Page meant for the parents, contains a list of all students present on the bus as well as their attendance status. Lastly, the Bus Page is a page in which the bus driver or students can upload and submit images of their faces to count for their attendance for the day. The image the students or bus driver upload are sent to the backend server, where the image is processed and matched to each of the saved student facial meshes saved in the database. If a matching facial mesh is found, that student’s attendance is marked as present and can be seen in either of the other two pages for the day. Once the image has been uploaded, the student is good to go and marked present on the bus.

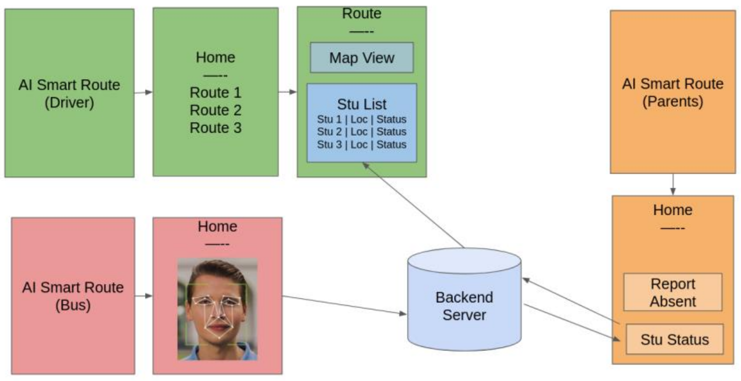


Figure 1. Overview of the solution

The first component previously mentioned, the ability to upload the file to the server, which takes the shape as the uploadFileToServer function, serves the purpose of taking the user inputted image and uploading it to the server so the facial mesh may be identified and analyzed. The file is sent over the internet using the Http protocol, with which it uses a custom client.

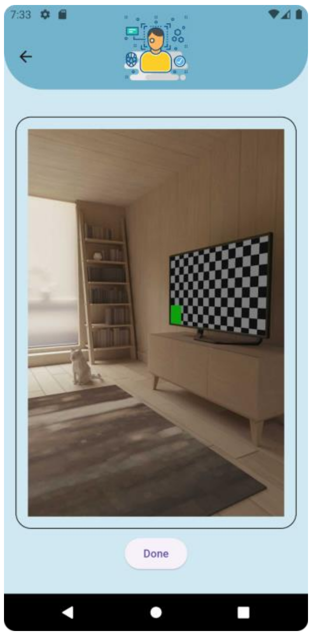
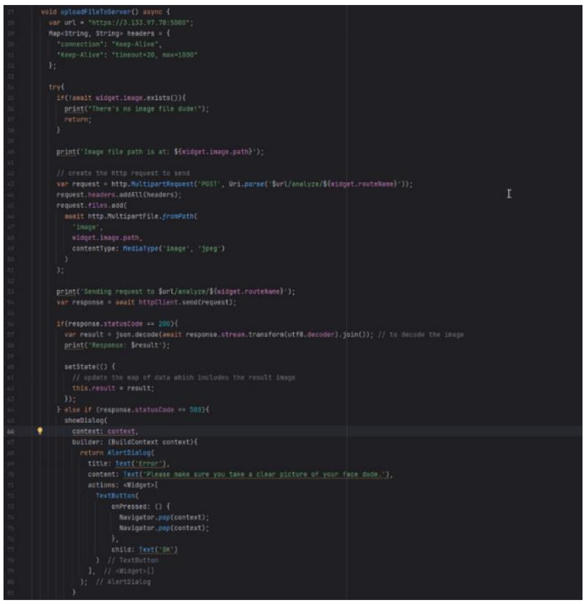


Figure 2. Screenshot of the function

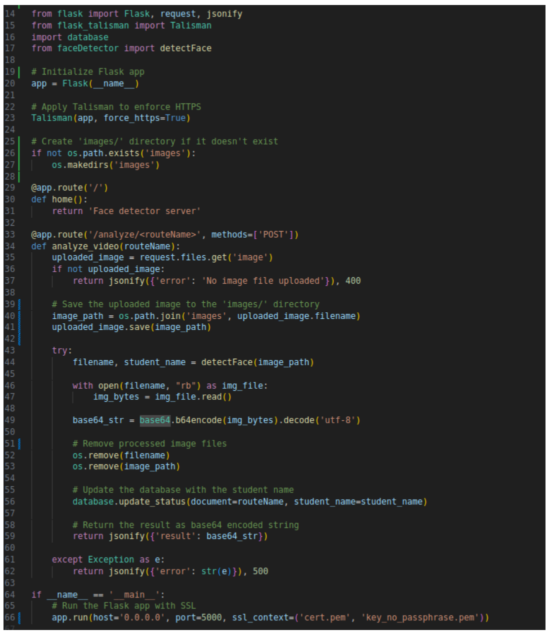


```
void uploadFileToServer() async {
  var url = "https://3.133.97.78:5000";
  Map<String, String> headers = {
    "connection": "Keep-Alive",
    "Keep-Alive": "timeout=20, max=1000"
  };

  try{
    if(!await widget.image.exists()){
      print("There's no image file dude!");
      return;
    }

    print('Image file path is at: ${widget.image.path}');

    // create the http request to send
    var request = http.MultipartRequest('POST', Uri.parse('$url/analyze/${widget.routeName}'));
    request.headers.addAll(headers);
    request.files.add(
      await http.MultipartFile.fromPath(
        'image',
        widget.image.path,
        contentType: MediaType('image', 'jpeg')
      )
    );

    print('Sending request to $url/analyze/${widget.routeName}');
    var response = await httpClient.send(request);

    if(response.statusCode == 200){
      var result = json.decode(await response.stream.transform(utf8.decoder).join()); // to decode the image
      print('Response: $result');

      setState(() {
        // update the map of data which includes the result image
        this.result = result;
      });
    } else if (response.statusCode == 500){
      showDialog(
        context: context,
        builder: (BuildContext context){
          return AlertDialog(
            title: Text('Error'),
            content: Text('Please make sure you take a clear picture of your face dude.'),
            actions: <Widget>[
              TextButton(
                  onPressed: () {
                    Navigator.pop(context);
                    Navigator.pop(context);
                  },
                  child: Text('OK')
              )  // TextButton
            ],  // <Widget>[]
          );  // AlertDialog
        }
```

Figure 3. Screenshot of code 1

The first few steps within the uploadFileToServer function involve declaring the endpoint http address of the back end server, as well as initializing the headers. After this configuration and a check to see if the image has been selected by the user, the request is initialized by creating a MultipartRequest Post request and the headers and file are added to it. After the request has been created, it is then sent to the backend server and the response variable is created to store the response from the server. After the response is received and the statuscode is received, an if statement is used to check and handle the responses. If the response code is 200, the result from the back end server is taken out of the JSON it was sent in and the state of the page is updated. However, if the status code is returned as 500, an alert notification is created and the user is encouraged to try again and make sure they include a picture of their face.

The back end server for the AI Smart Route mobile application performs the majority of the apps intended functionality in that it manages the facial mesh detection of the images, the comparison of these meshes with its local database of meshes, and accesses the Firebase database to mark student present based on these comparisons. The most prominent technology utilized within the backend server is the mediapipe library, as it is the facial mesh detection engine.



```python
from flask import Flask, request, jsonify
from flask_talisman import Talisman
import database
from faceDetector import detectFace

# Initialize Flask app
app = Flask(__name__)

# Apply Talisman to enforce HTTPS
Talisman(app, force_https=True)

# Create 'images/' directory if it doesn't exist
if not os.path.exists('images'):
    os.makedirs('images')

@app.route('/')
def home():
    return 'Face detector server'

@app.route('/analyze/<routeName>', methods=['POST'])
def analyze_video(routeName):
    uploaded_image = request.files.get('image')
    if not uploaded_image:
        return jsonify({'error': 'No image file uploaded'}), 400

    # Save the uploaded image to the 'images/' directory
    image_path = os.path.join('images', uploaded_image.filename)
    uploaded_image.save(image_path)

    try:
        filename, student_name = detectFace(image_path)

        with open(filename, "rb") as img_file:
            img_bytes = img_file.read()

        base64_str = base64.b64encode(img_bytes).decode('utf-8')

        # Remove processed image files
        os.remove(filename)
        os.remove(image_path)

        # Update the database with the student name
        database.update_status(document=routeName, student_name=student_name)

        # Return the result as base64 encoded string
        return jsonify({'result': base64_str})

    except Exception as e:
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    # Run the Flask app with SSL
    app.run(host='0.0.0.0', port=5000, ssl_context=('cert.pem', 'key_no_passphrase.pem'))
```

Figure 4. Screenshot of code 2

For the hosting of the back end server, we decided to utilize one of Amazon's AWS EC2 instances to host a flask server. The main reason for utilizing an EC2 instance over something like a free Render.com instance, is because most other services don't allow file operations on their servers, however, AWS does. In the image above, the first steps are initializing the Flask app using Talisman to force HTTPS on the server [14]. After the server is initialized, the root and /analyze routes are created. In the analyze route, the image is first grabbed from the request and saved locally on the server. After the image is saved locally, it is sent to the detectFace function where the facial mesh of the image is found and then compared to the facial meshes of those in the dataset of students that should be on the bus. If no facial mesh is found, an error is returned. However, if a mesh is found matching, the detectFace function uses a separate function to update the Firebase database with the student's attendance. After that update is made, the response is then sent back to the client that their attendance was taken.

The third and final major component within the AI Smart Route mobile app is the user interface design using the Flutter framework. The approach to designing the UI had a heavy focus on keeping things simple, as the user base is going to have a wide range of ages.

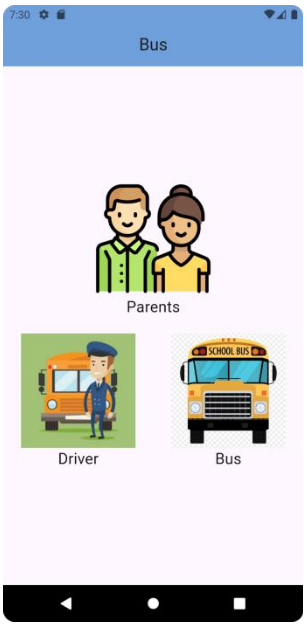
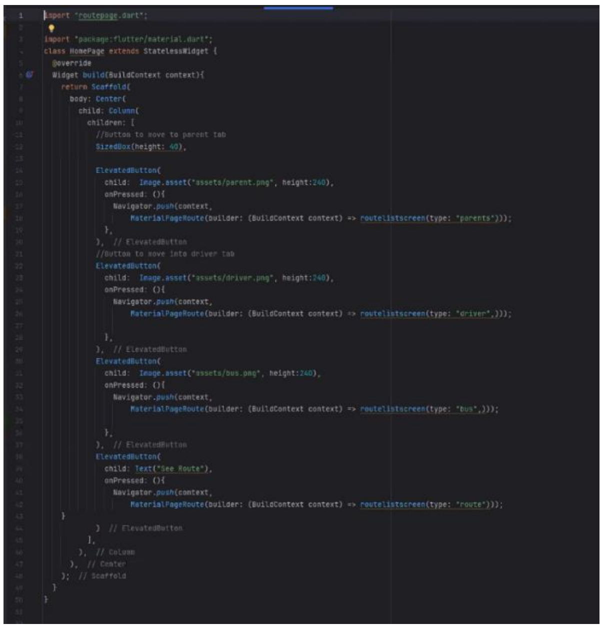


Figure 5. Screenshot of the bus function



```
import "routepage.dart";

import "package:flutter/material.dart";
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context){
    return Scaffold(
      body: Center(
        child: Column(
          children: [
            //Button to move to parent tab
            SizedBox(height: 40),

            ElevatedButton(
              child:  Image.asset("assets/parent.png", height:240),
              onPressed: (){
                Navigator.push(context,
                    MaterialPageRoute(builder: (BuildContext context) => routelistscreen(type: "parents")));
              },
            ),  // ElevatedButton
            //Button to move into driver tab
            ElevatedButton(
              child:  Image.asset("assets/driver.png", height:240),
              onPressed: (){
                Navigator.push(context,
                    MaterialPageRoute(builder: (BuildContext context) => routelistscreen(type: "driver",)));

              },
            ),  // ElevatedButton
            ElevatedButton(
              child:  Image.asset("assets/bus.png", height:240),
              onPressed: (){
                Navigator.push(context,
                    MaterialPageRoute(builder: (BuildContext context) => routelistscreen(type: "bus",)));

              },
            ),  // ElevatedButton
            ElevatedButton(
              child: Text("See Route"),
              onPressed: (){
                Navigator.push(context,
                    MaterialPageRoute(builder: (BuildContext context) => routelistscreen(type: "route")));
      }
            )  // ElevatedButton
          ],
        ),  // Column
      ),  // Center
    );  // Scaffold
  }
}
```

Figure 6. Screenshot of code 3

The image depicted above is the Dart code utilizing the Flutter framework for creating the home page the user first sees upon opening the AI Smart Route mobile app. The HomePage class inherits from the StatelessWidget class, which is essentially a Flutter defined template for creating any widget, however, most of the time it is pages. What the widget actually looks like on the screen is defined in the overridden build method. In the build method, a main Scaffold is created, which creates the basic structure for the page. Inside the Scaffold, the body is centered and a Column is created to start the stacking of items. Inside the Column, the three visible ElevatedButtons are created to act as the different pathways the user can use to navigate the app.

4. EXPERIMENT

4.1. Experiment 1

One major component that we suspect of potentially being a weak spot within the application is the AI facial recognition back end server. We are specifically worried about how efficient it is and how much time it takes for a user to receive a response back.

To get an idea of how long it typically will take the user to receive a response from the back end server, we will conduct an experiment to find the average response time. For this experiment, one team member will make multiple submissions to the back end server with a valid facial mesh in the image, and another team member will record the time it takes after the previous team member submits for the response to get back. Eight of these submissions will be performed with a different image each time to ensure that a fair average is found.

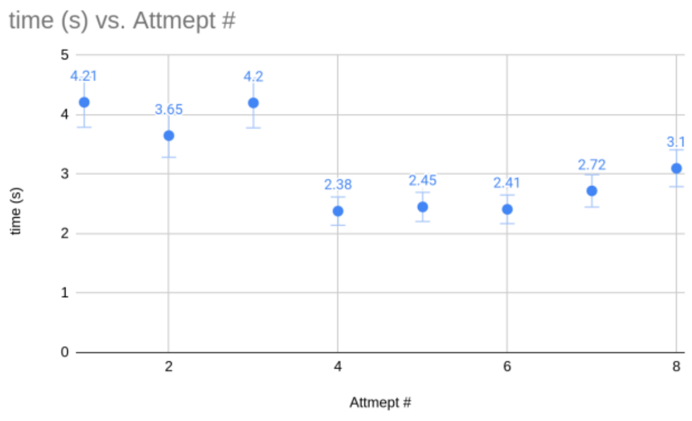


Figure 7. Figure of experiment 1

After conducting the experiment, it is quite noticeable that our concerns for the amount of time spent by the back end server we’re nothing to worry about. The back end server proved to handle the requests in a time efficient manner and were able to get the responses back to the client at an average of 3.14 seconds. The highest value during the experiment took around 4.21 seconds, whereas the quickest was a response back from the server in 2.38 seconds. We believe that the data we recorded demonstrates the effectiveness of the redesigns we did on the back end server, as the mediapipe library has proven to be quicker that utilizing a local model on the server. As far as the research team’s expectations went, this server did neither exceed or not exceed expectations. Rather, it worked as expected.

4.2. Experiment 2

As the intended environment that this application is meant to be used in is one of a generally fast paced system of students getting on a bus and the bus driver getting them all to school, the ease of use and effective identification of facial meshes is critical for the application to be used effectively. To ensure the app properly identifies facial meshes in a timely fashion, we will test how prominent a facial mesh must be in an image for it to be found.

To evaluate and test to see how practical taking an image and finding a facial mesh is for a bus driver to do while taking the attendance of the students, we will design an experiment to see how distance and image conditions affect the identification of a facial mesh. We will start by taking a picture of a team member at a distance of 2 feet away from their face, take a new picture every added foot away from their face until we get to 10 feet from the user. This will allow us to see how far away the bus driver can take a picture of a student and still identify and find their facial mesh.

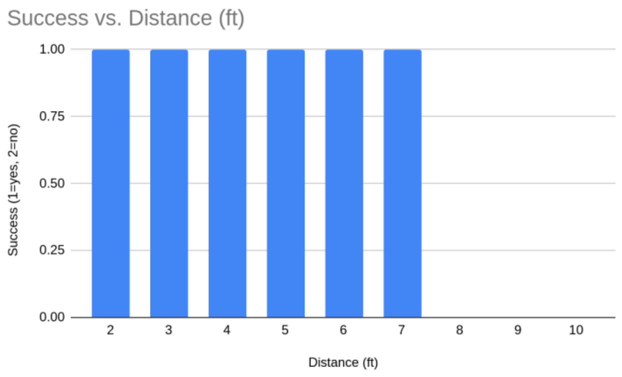


Figure 8. Figure of experiment 2

Throughout the experiment process, we found that the success rate as the distance from the camera increased was a lot higher than we expected. As the distance increased, the facial mesh recognition proved to be effective at distances 7 feet or closer. This quite surprised us, as we originally expected the functional distance for the facial mesh to be recognized would be around 4 feet or closer, which is quite a noticeable distance from 7 feet. We believe that the reason facial meshes were able to be detected at this range is because of two main reasons: the camera quality and lighting as well as the effectiveness of the mediapipe library. The device we performed the experiment with, an Iphone X, has a camera with a decent quality, which we believe to have impacted our results plus the quality lighting in the room we performed the experiment in. Additionally, it is important to note that the mediapipe library is quite developed and very effective.

5. RELATED WORK

The most common current form of communication used by bus drivers today to communicate with the school district about student's attendance is the use of Walkie-Talkies [11]. The obvious potential issues of the use of Walkie-Talkies are countless and range from communication issues to distance gaps causing cutouts. The AI Smart Route mobile application remedies these issues by providing an easy way for bus drivers to take attendance and attendance to be taken and stored in a database. With AI Smart Route, the common issues of Walkie-Talkies are no longer present.
Bus Attendance is a mobile application designed for bus drivers to be able to mark attendance for passengers as they come onto the bus [12]. However, Bus Attendance does not have any external resources such as a back end server or database that enables real time updates with the school district or parents on students' attendance. Additionally, the Bus Attendance mobile application requires the bus driver to manually log each student's attendance, whereas the AI Smart Route app only requires an image of the student's face.

SBAS is a complete bus attendance system available for schools to purchase as a package which includes an administrative panel, a mobile app for parents to download, custom hardware for buses to install, and fail safe systems to ensure all students are accounted for and parents stay informed [13]. The most prominent issue we found when analyzing the SBAS system was how students were checked in. Students' attendance is logged by the bus driver using RFID school ID cards or manually inputted by the bus driver. If a student forgets their id, the process takes much longer compared to our system. As the AI Smart Route mobile app relies on facial meshes to take attendance, it doesn't require any extra hardware and is more efficient.

6. CONCLUSIONS

After the development and experimentation phases of the project we completed, we found that the biggest area that the application could see some work done would be the AI Smart Route mobile app front end. The mobile app's current condition enables any user to be able to open the app and submit an image through the bus driver portal [15]. On top of that, it also enables the user to check the status of any bus and what students are on it. This is obviously an issue that needs to be resolved in the future, and to do so we would employ the use of a user database that controls access permissions. Depending on who signs in, a student, parent, or bus driver, they would only have access to their respective pages and only see information relevant to their titles. Alongside these renovations, we would also like to improve the front end design of the app, and make them a bit more user friendly.

To conclude, the AI Smart Route mobile application, despite its flaws previously mentioned, remains an effective solution to common bus communication issues that plague the current public school bus system. Throughout the development process, the development team has learned all of what can potentially go into designing a mobile application, and the experience has proven to be quite valuable.

REFERENCES

[1] Parusel, Sylvia, and Arlene Tigar McLaren. "Cars before kids: automobility and the illusion of school traffic safety." Canadian Review of Sociology/Revue canadienne de sociologie 47.2 (2010): 129-147.

[2] Ellegood, William A., et al. "School bus routing problem: Contemporary trends and research directions." Omega 95 (2020): 102056.

[3] Park, Junhyuk, Hyunchul Tae, and Byung-In Kim. "A post-improvement procedure for the mixed load school bus routing problem." European Journal of Operational Research 217.1 (2012): 204-213.

[4] Pedde, Meredith, et al. "Randomized design evidence of the attendance benefits of the EPA School Bus Rebate Program." Nature Sustainability 6.7 (2023): 838-844.

[5] Ayoub, Jackie, et al. "Otto: An autonomous school bus system for parents and children." Extended abstracts of the 2020 chi conference on human factors in computing systems. 2020.

[6] Li, Shaowei, Wenbin Xu, and Zhiwei Li. "Review of the SBAS InSAR Time-series algorithms, applications, and challenges." Geodesy and Geodynamics 13.2 (2022): 114-126.

[7] Yi, Sangho, Derrick Kondo, and Artur Andrzejak. "Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud." 2010 IEEE 3rd International Conference on Cloud Computing. IEEE, 2010.

[8] Relan, Kunal, and Kunal Relan. "Beginning with flask." Building REST APIs with Flask: Create Python Web Services with MySQL (2019): 1-26.

[9] Sansaloni, Carolina P., et al. "A high-density Diversity Arrays Technology (DArT) microarray for genome-wide genotyping in Eucalyptus." Plant Methods 6 (2010): 1-11.

[10] Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." arXiv preprint arXiv:1906.08172 (2019).

[11] Xu, Yaqi, et al. "The research of safety monitoring system applied in school bus based on the internet of things." Procedia Engineering 15 (2011): 2464-2468.

[12] Kadam, Anilkumar J., et al. "Developing a Smart Bus for Smart City using IOT Technology." 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2018.

[13] Raj, Judy Thyparampil, and Jairam Sankar. "IoT based smart school bus monitoring and notification system." 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC). IEEE, 2017.

[14] Korkmaz, Ilker, et al. "A smart school bus tracking system." 2019 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2019.

[15] Dang, Thai-Viet. "Smart attendance system based on improved facial recognition." Journal of Robotics and Control (JRC) 4.1 (2023): 46-53.