EECS 581/582 Final Project

Team 18

Project Artifacts Sprint 9

Completed

Requirement ID	Description of Requirement	Story Point	Priority	Sprint No.
54	Record footage with our phones	1	1	9

-We went to the burge parking lot and recorded a series of videos of cars entering and leaving the parking garage from different angles to train and test our model.

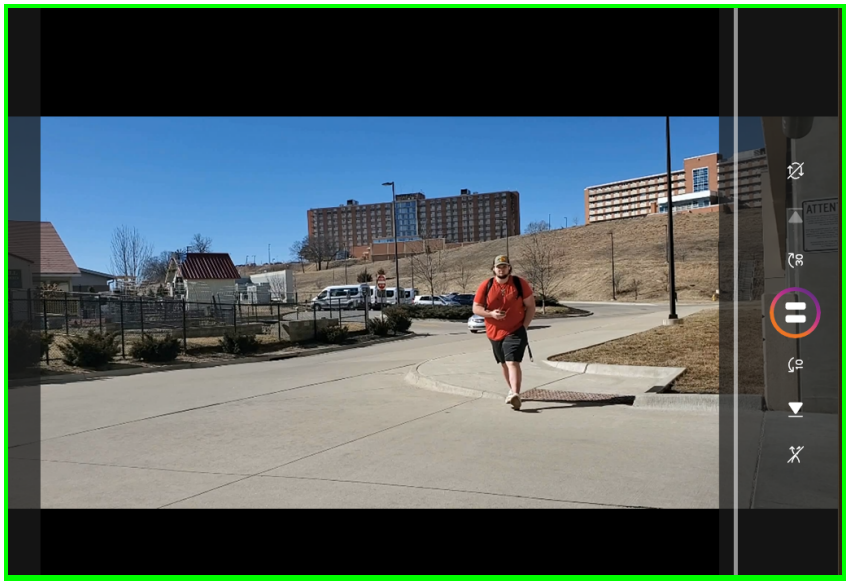


Fig 54.1: a snapshot of the video recorded

58	Model conversion to TFlite	5	2	9

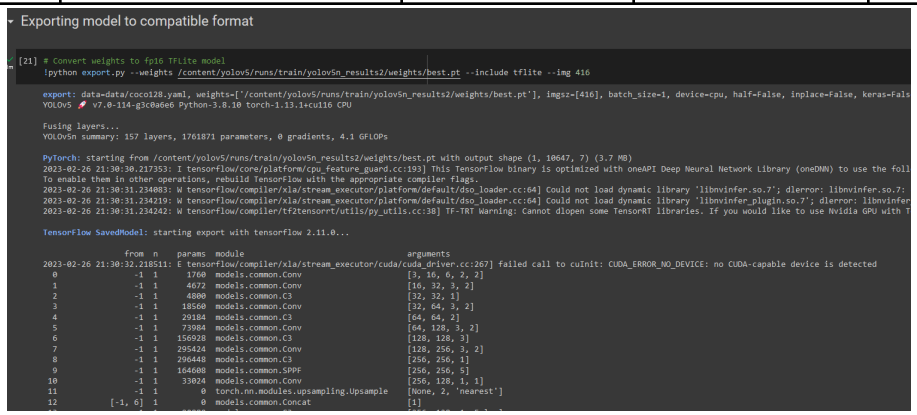
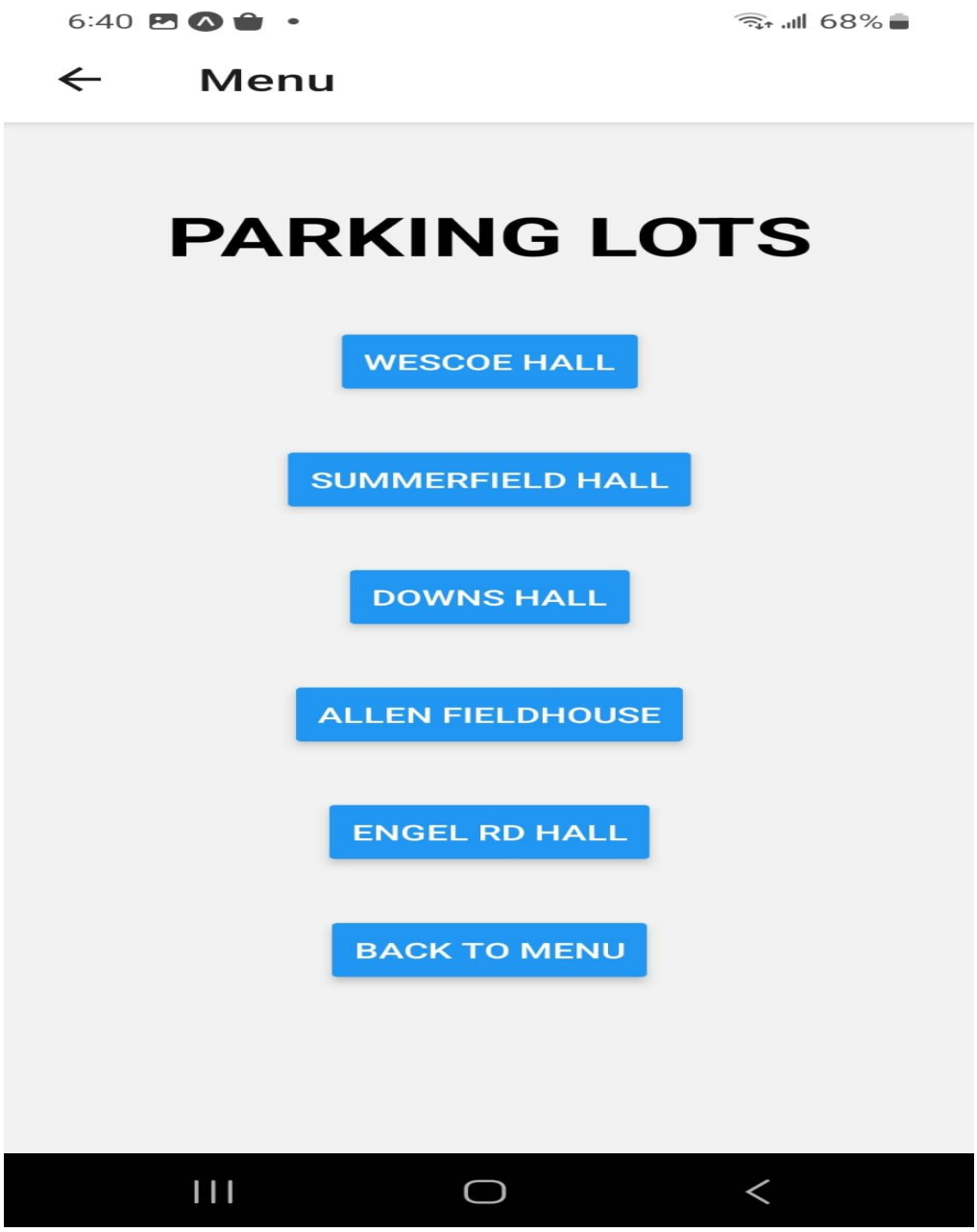


Fig: Conversion of the original model to tflite format for the mobile app

-Due to the size of the model, we aren't able to run the model on the app so we have to compress it to tflite. We did the compression and conversion already, we are going to test it and debug as necessary.
-The next biggest challenge is incorporating the ML model within the app itself. We have an android and ios version so it's a big jump for all of us with this part.
-Another challenge with the tflite model is the inaccurate detection which is kind of unexpected. Will keep checking to see if higher epochs change the tflite model performance. The bigger model seems to be working fair.

Requirement 55 and 57:
We have added a short lots list for red permit. We will be doing the same for the other screens as well.
below is the figure:

Requirement 56:
We have the buttons and styling added to this screen.



Requirement 36:
Link to buy a permit has been added to the explore lots screen. It takes the user to the KU parking website where they can buy any type of permit.

Not Completed

-We had an exam and extra work we didn't anticipate having to deal with. Our plan is to push following to Sprint 10 and get caught as quickly as possible.

-

18	Which permit is required to park	1	3	10

- Decided to display the permit required by adding a corresponding background color to the lot name in the X lot screen. Anticipated completion sprint 10.

11	Parking fares display	2	5	-
12	Total spots per level	1	3	10

- 11: Omitted due to redundancy [36 redirects to the shop permit website]
- 12: Decided to hardcode the total spots per level and not the available spots. Anticipated completion sprint 10.

6	Accurate real-time car counter	8	2	9

- The model can produce the count of cars in real-time, however we haven't been able to integrate it into the app. This is the core of the project and it will take a considerable amount of effort. We anticipate accomplishing this within the next two sprints.

The following diagram shows what the flow of our project would look like:

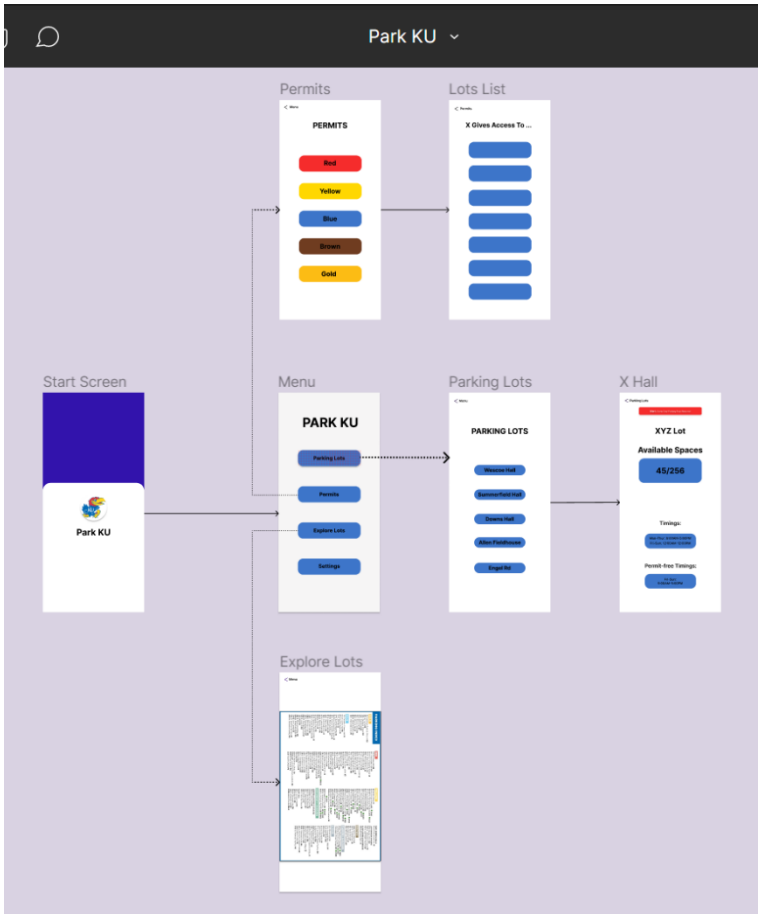


Fig5: The flow of our project; ParkKU

The tentative flow of our app can be viewed on the following link:

https://github.com/hzahid99/EECS-Final-Project/blob/main/Demo%20of%20the%20app.mp4

The following GitHub repository holds our progress so far:

https://github.com/hzahid99/EECS-Final-Project

Following are the screenshots of the screens we have made so far for the app:

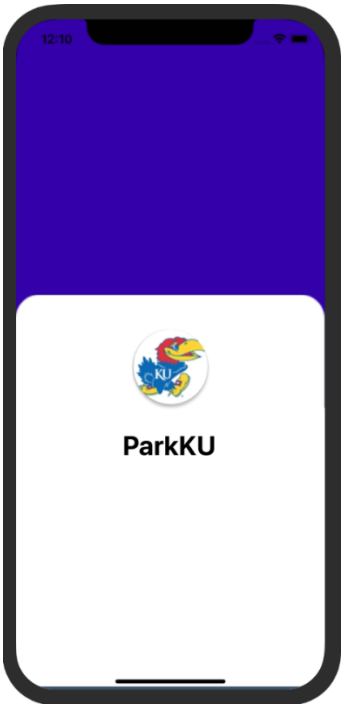


Fig5: Start Screen of the app

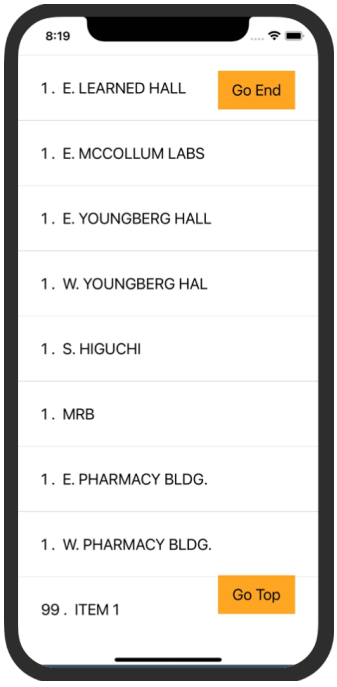


Fig6: Screen that will list down all the lots and permits they require

This screen has the names of the lots but needs to be updated to include the color-coded permits and have the lots divided permit-wise.

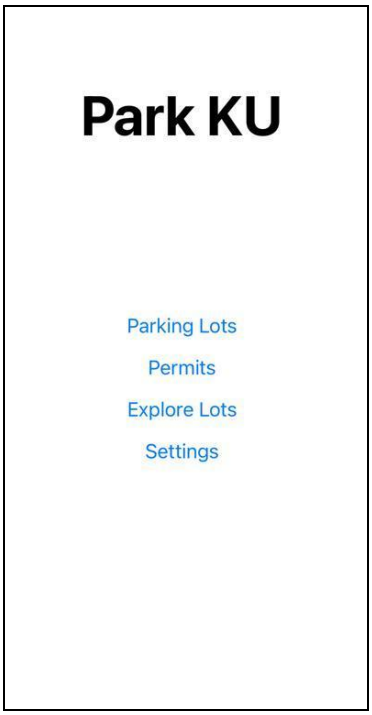


Fig7: Menu Screen

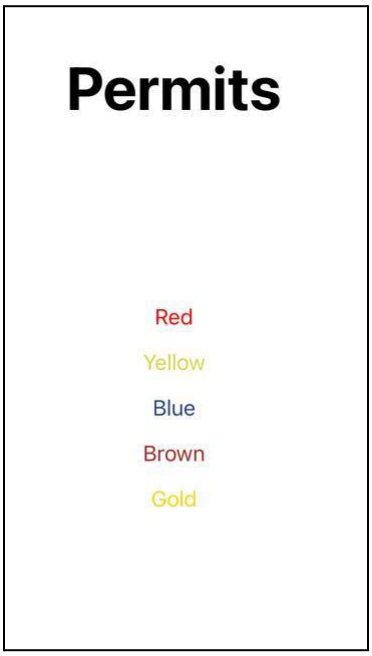


Fig8: Permits Screen

Worked on the recognition model and increased the epochs to 170. Ran the model over the video collected from our camera. The video is inside an engineering building. The model detects some cars for some reason inside, especially when the camera is being moved. But, we will have to debug the situation more. The camera still doesn't connect to the WIFI and we may need to take it to the IT so that they can help us connect it.

The validation error is going down which is a good indicator for the model accuracy. We will still keep working on it as we collect video from our camera.The test video has been uploaded on github.

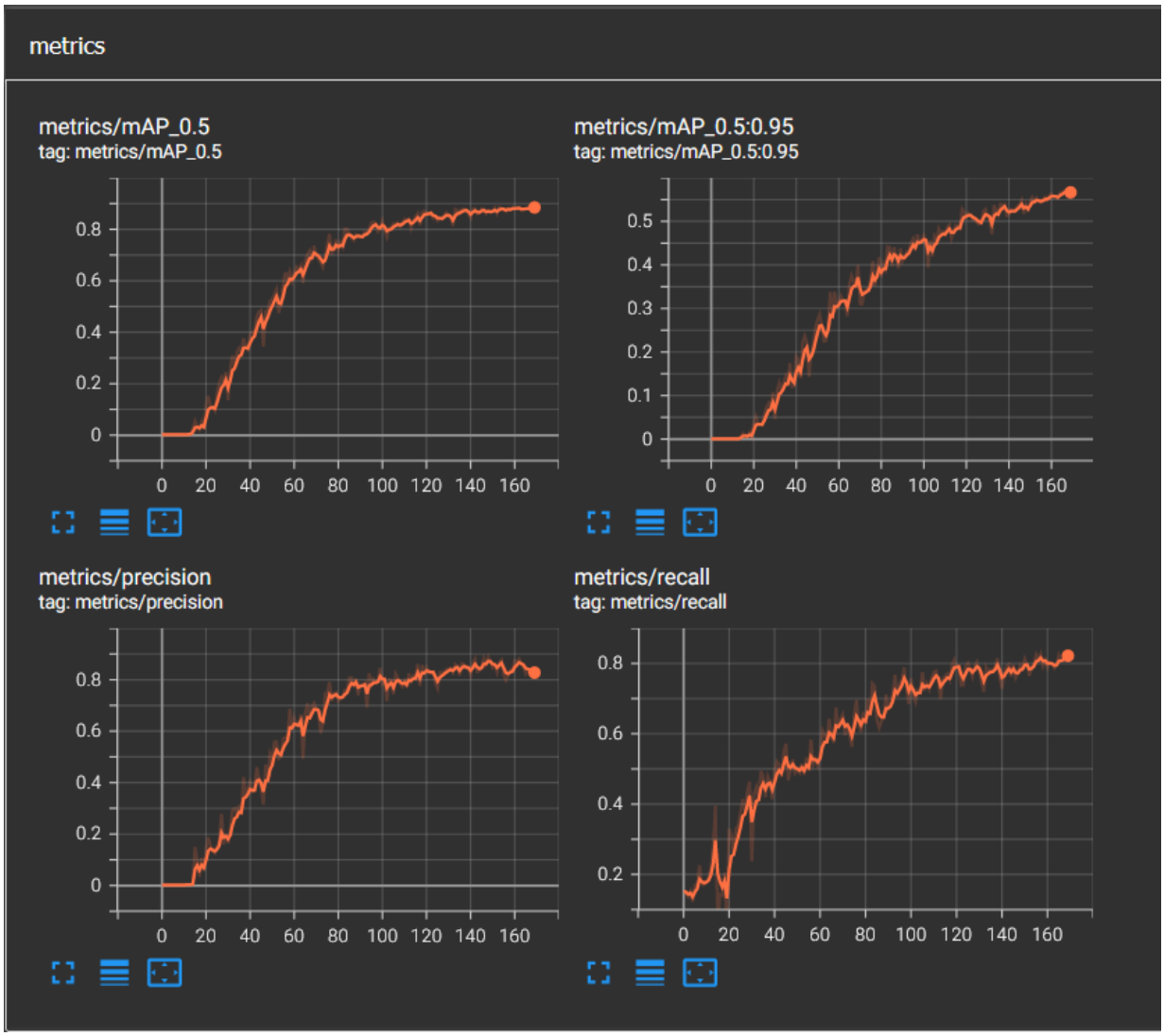


Fig9: Metrics on the model

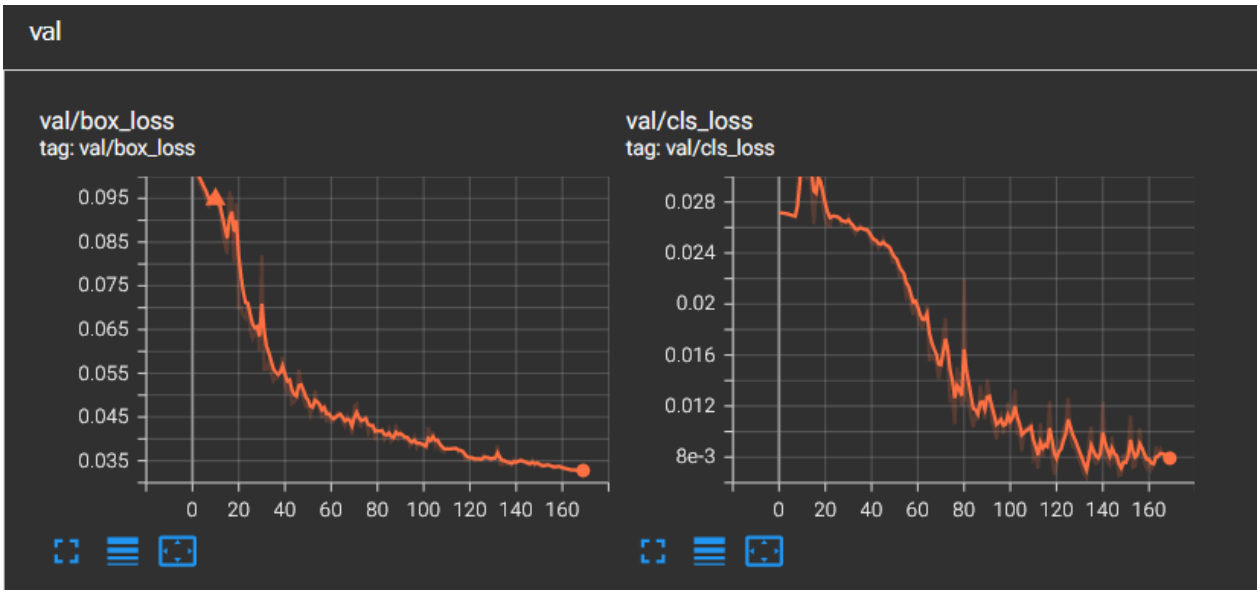
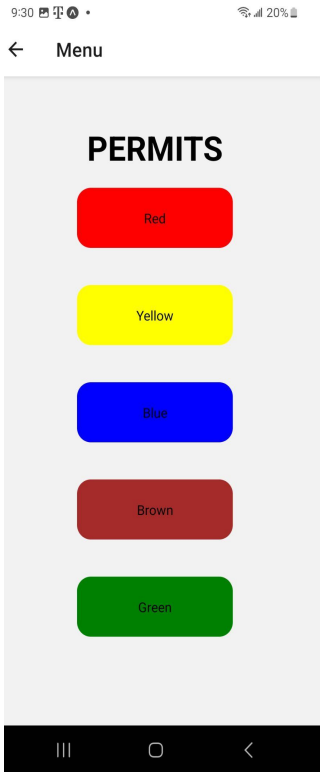
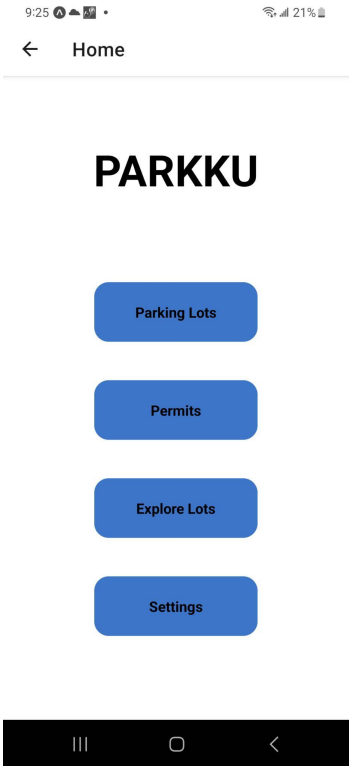


Fig9: Validation metric of the model

Requirements 34 and 43:





We have added the buttons to both the menu and permits screen. Navigation has been added to them too and from Menu, we can easi;y go to the permits screen by just hitting the permits button. For the Permits screen, we needed a little more work to do with the buttons since they required different colors. Each color represents the text inside it.

snippet of Permits screen code:

```
<View style = {styles.button}>
      <Pressable
      style = {[styles.pressable, {backgroundColor: "red"}]}
        onPress = {() => navigation.navigate('LotsList')}
        //onPressIn = {fontSize: 25 }
        android_ripple = {{color: 'LotsList'}}
        activeOpacity = {0.5}
      >
        <Text style = {styles.text}>
          Red
        </Text>

      </Pressable>

      <Pressable
    style = {[styles.pressable, {backgroundColor: "yellow"}]}
        onPress = {() => navigation.navigate('LotsList')}
        //onPressIn = {fontSize: 25 }
        android_ripple = {{color: 'green'}}
        activeOpacity = {0.5}
      >
```

```
        <Text style = {styles.text}>
          Yellow
        </Text>


      </Pressable>
```

snippet of menu screen code:

```
<View style = {{flex: 1, alignItems: 'center', justifyContent: 'space-evenly', backgroundColor: 'white'}}>
  <Text style = {{fontWeight: 'bold', fontSize: 45, textAlign: 'center' }}> PARKKU </Text>


  <View>
    <Pressable
    style = {styles.pressable}
      onPress = {() => navigation.navigate('ParkingLots')}
      //onPressIn = {fontSize: 25 }
      android_ripple = {{color: 'green'}}
      activeOpacity = {0.5}
    >
      <Text style = {styles.text}>
        Parking Lots
      </Text>


    </Pressable>
```

```
<Pressable
  style = {styles.pressable}
  onPress = {() => navigation.navigate('Permits')}
  //onPressIn = {fontSize: 25 }
  android_ripple = {{color: 'green'}}
  activeOpacity = {0.5}
>
  <Text style = {styles.text}>
    Permits
  </Text>

</Pressable>
```

Fig 10.0: Menu and Permit on expo snack emulator

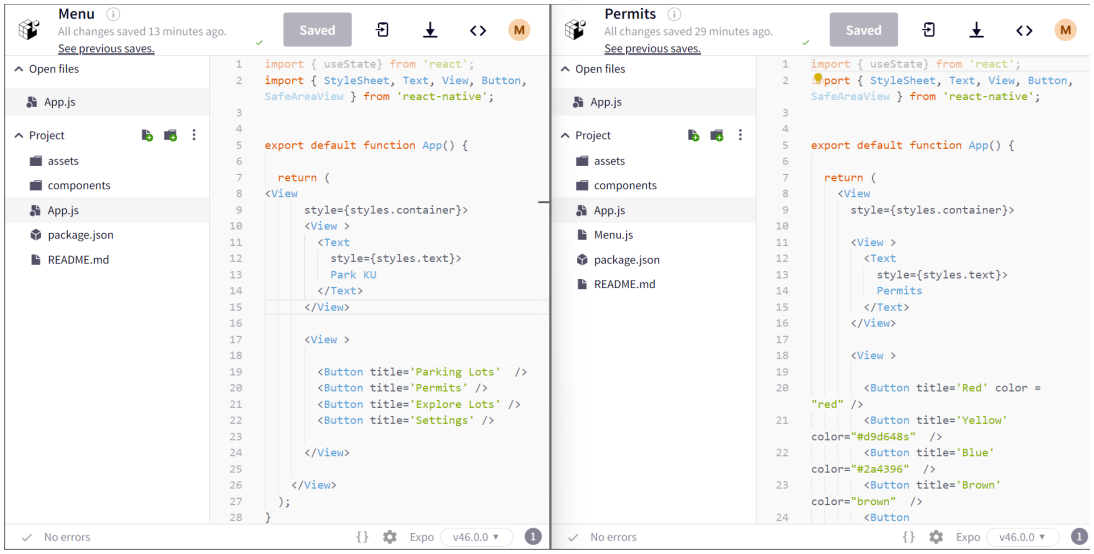
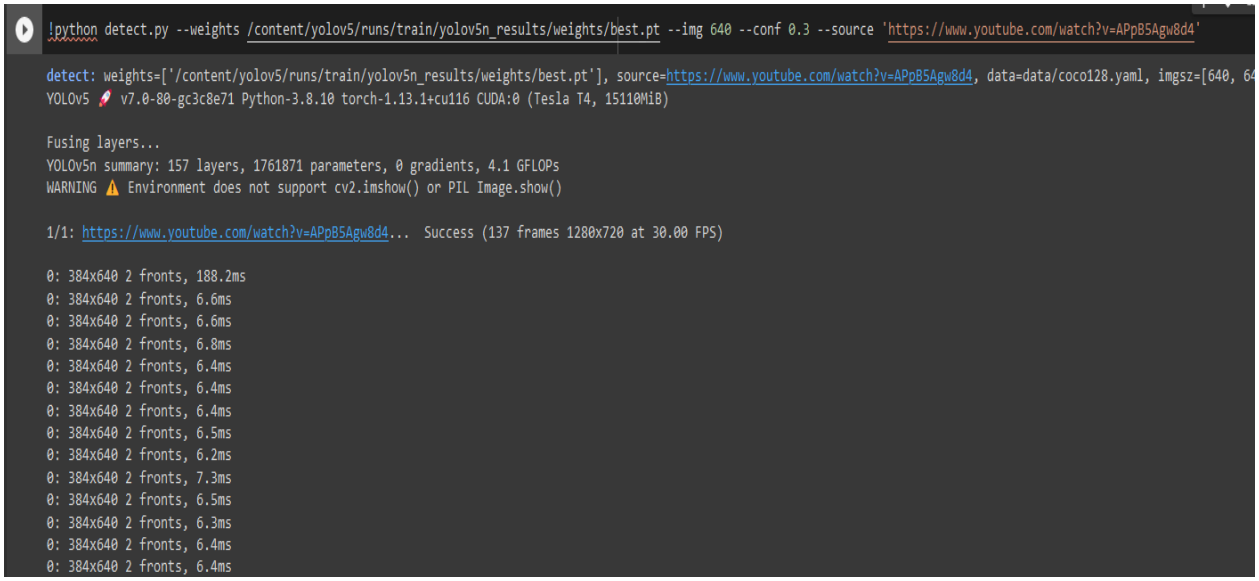


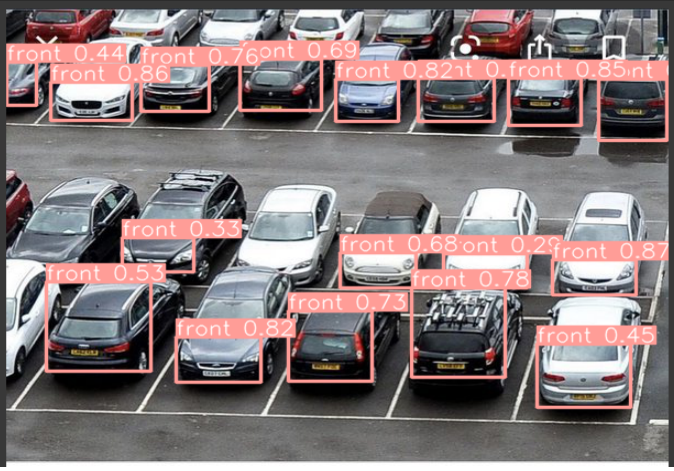
FIg 10.1: Menu and permit expo code snippet

Requirements 38 & 39

Our camera isn't connected to the campus WIFI, and KU IT couldn't help us further. We are moving forward with youtube videos for testing, we might use our camera to take videos outside engineering further.



```
# Image
img = '/content/Test-images/t1.jpg'
# Inference
results = model_yolo(img)
# Results, change the flowing to: results.show()
results.show()  # or .show(), .save(), .crop(), .pandas(), etc
```



The team's app developers needed the number of sides being detected:

```
[ ]  # we have 16 objects which are front of the car

[ ]  detected_object = results.pandas().xyxy[0].value_counts('class')  # class counts (pandas)
     print(detected_object)

     class
     1    17
     dtype: int64

[ ]  type(detected_object)

     pandas.core.series.Series

[ ]  from collections import OrderedDict, defaultdict
     # empty dictionary of car sides

     detected_object.to_dict(OrderedDict)

     # we can see that the class 1: front has been counted 17 times
     # just saving this in dictionary format might be useful for accessing it later on and
     # displaying it on our app

     OrderedDict([(1, 17)])
```

Some of the demo videos are on
Github:https://github.com/hzahid99/EECS-Final-Project/tree/main/Demo

Note: I'm still re-running the model just so that I can come with an optimum number of epochs for training. However, our model is in good shape. I will start looking into the model deployment alongside collecting the parking video with the rest of the team in future.

-