50.001 1D Report

Group 2 Members:
(1) Tay Qin Long (1003113) (2) Teo Yong Quan (1002828) (3) Chloe Zheng (1002934)
(4) Joseph Chan (1002948) (5) Gary Ong (1002758) (6) Andri (1002849)

BACKGROUND AND PROBLEM

The canteen was chosen as our target location, because we felt that we did not want to the large scale project that involved placing sensors and objects all around the school, yet we wanted to have the opportunity to impact the majority of the school's population. We felt that the canteen was the perfect location for us to implement our solutions, because the size of it is small enough for us to work on, and most members of the school tend to visit the canteen at least once everyday.

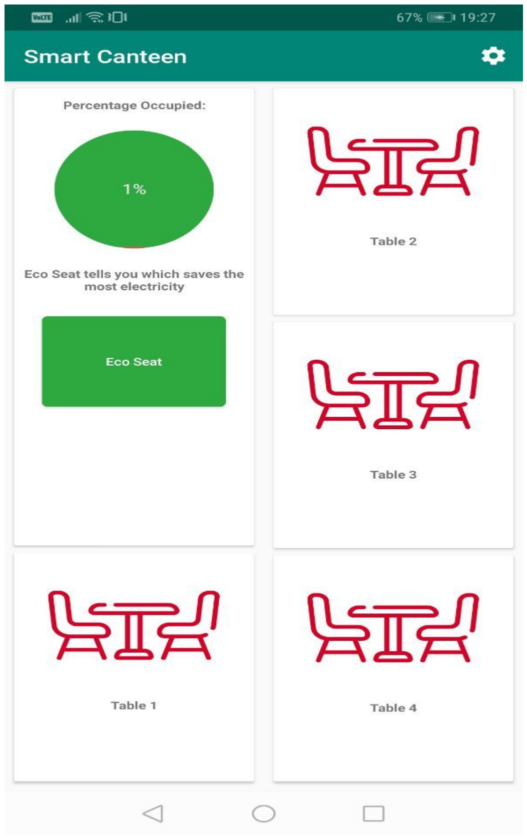
After narrowing down to the canteen, we figured that there were 4 main problems.
1) The canteen is very crowded during lunch time, making it hard to find seats.
2) The canteen has long queues during peak hours, and order processing and payment becomes very inefficient.
3) During off peak hours, electricity wastage can be observed.
4) The canteen is often dirty because users are lazy to clear their table.

We decided to tackle problem 1 and 3, and if successful problem 2 can be alleviated.

System design & implementation

Our solution revolves around the use of computer vision and recognition of humans to count the number of people in a given area. This information will be updated to firebase which will be pulled into our user app. To implement the above we have designed 2 apps for this project. The first is a backend user app called human counter, it is forked from Tensorflow examples. The second app is called Smart Canteen this is the front end app for users to get information about the canteen.

On our app, we have the following functions,
1) Display occupancy of the canteen.
2) Allow user to set a notification when canteen occupancy falls below certain rates
3) Allow user to see live updates and plot of how long has an area been occupied
4) Recommends users to group in one area, such that the minimum amount of lights and fans will be turned on.

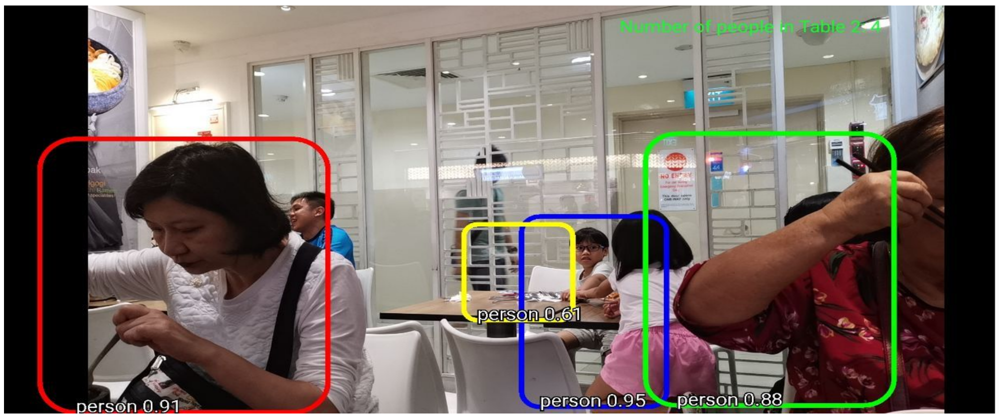


This screenshot is the home page of our app

In the user app, we mainly have,

1) Canteen module - which is the small circle on the top left on the app, which displays the number of people in the canteen. This is implemented by pulling data from firebase and putting the data onto a Circle Progress View.
2) Fireapp - which initiates the firebase
3) Mainactivity - this is the home page, where we pass the controls to other activities when called.
4) Pointvalue - creates a data class for graph plotting.
5) Settings - This starts the settings activity which allows one to set his definition of crowdedness, which creates a notification when the capacity falls below his input, which will be handled by the class below
6) Smart notification - as mentioned above
7) Spotlight helper - This module implements the eco seat as seen above, which tells users to sit at an occupied table, if there are sections of the canteen unoccupied. This is a helper class that we create to use the Spotlight library.
8) Table activity - Clicking this will bring you to the section of the canteen, which tells you how many people are there in that particular section. In this activity, you will be able to see the live graph update.

Backend



In our backend app there are many classes provided by the Google TensorFlow examples. we added an extra activity called **Main Activity** so that users can choose what they want to be uploaded into Firebase. There are also 2 classes that we made modifications and the two classes are Camera Activity and Detector Activity.

Detector Activity was changed so that the camera receives a widescreen video feed instead of the default 1:1 ratio video feed. In addition a thread and a handler is created in order to handle some UI elements when the video feed is live and also to upload relevant data into Firebase.

Camera Activity was changed as there were some bugs when older phones were used with our app with a widescreen view. This is because these older phones used the LegacyCamera api which mishandled the wide video feed that we made modifications to.

Controlling the lights and fans

Using the data stored in firebase, we used raspberry pi to programme when to turn on and off the lights and fans. This was pretty straightforward, by using what we have learnt in The Digital World last year.

Design Choices for better maintenance,upgradability and debugging

Using 1 Activity to handle multiple intents

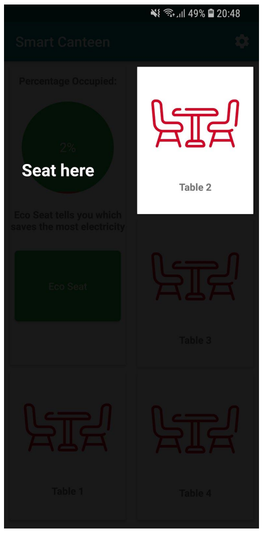
In our app we have 4 different tables. Tapping of one of the tables brings us to another page where the number of people at that table and a graph of number of people vs time is shown.

Since all the tables do the essentially the same function it would be much more convenient to use one activity to handle them all.

Firstly, we created an ArrayList that takes in a CardView. Then we use *findViewById* on all the tables and added the them on to the array list. Next we created a method *setTableListeners* that takes in that ArrayList loops through all of them so as to set the listeners and at the same time sending the intent with the string extra that specifies which table was clicked. So 4 different intents call the same activity!

Shared PointValue class between both apps

This class is used in both our backend and frontend apps to allow us to easily generate data points, upload it into firebase and then from our frontend app parse and plot these data points onto a graph.



Creating a helper class to set spotlight

The spotlight library is a class that literally creates a spotlight to allow the user to see what view we want to show them.

This is a spotlight. To do this we created a helper class to help us implement the many functions from this library.
In this helper class we have 2 static methods *createSpotlight,createTarget* and an inner class Rectangle shape. *createSpotlight* takes in an activity and a target. It then creates the spotlight for us.The createTarget creates the target of where the spotlight should be and the shape you want it to be. We put the shape as an inner class where the shape is drawn using a canvas.

Dynamically setting onClickListeners

The spotlight above is shown whenever the user clicks the Eco button which point to them where the seat should be. However, the table in the spotlight changes according to the data from firebase. Hence the onClickListener needs to be dynamically added. The first step in doing this is to create a HashMap that maps the text ‘Table 1’ to the cardView associated with the table. Then the firebase *valueEventListener* would then set the onClickListener on the eco Button dynamically by calling *setEcoButton* which takes in the view from the hashmap and sets the spotlight. Hence the same button click can produce different outputs based on Firebase.

Lessons Learnt, future work and conclusion

The 1D project definitely helped us put what we learnt in class into an actual working app, which was surely good practice to allow us to be familiar with android studio and application coding. Another good practice we found was to make our codes modular, so as to allow ease of combining, as every group member can work on their own function individually, and combine them in main.

In addition, we picked up stuffs there were out of syllabus, which were very useful skills to have. Implementation of graphview, for example, is certainly going to be useful in future projects. Computer vision, data training and tensorflow are nice to know, in preparation for the upcoming terms and even for internship.

Perhaps one thing we could work on would be to improve our Human counter app, which fails to detect humans when they are too far or in the background. This is because 1080x1920 image is fed into a neural net trained by Google on a 300x300 image (this app's apk is already 100mb!). This leads us to have somewhat inaccurate data, and to mitigate this, we could have used more cameras such that each camera only have a small area of coverage. An improvement to the human counter app would surely allow just one camera for a large sector, which unfortunately, is beyond our level at the moment.

All in all, a lot was learnt through developing the app, and it definitely made us better android programmers.

References

Human Counter app forked from Tensor Detect:
https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android

GraphView open source library http://www.android-graphview.org/

Circle Progress:https://github.com/lzyzsd/CircleProgress