We used Android Studio in order to develop our android application as it is the official IDE for Android development. The language used to develop the entire logic of the application was java. Also, the UI screens are visually designed and coded as XML layout files. First, we created an empty android project. After that, whenever we need to add new service, we have to add New Activity to create XML file for designing UI and java file coding.

Check Availability:

* Algorithm:

Step1: wait till the VIP user selects zone

Step2: load all the spots that are in the selected zone

Step3: check each spot status, if the status is “available” set the visibility of the car image to be “invisible” and if it is “not available” set the visibility of the car image to be “visible”

* Design:

|  |  |
| --- | --- |
| Application Design: | |
| User clicks  Zone list was implemented by creating custom list view | Map was designed by using image view |

Reserve a parking:

* Algorithm:

Step1: wait till the VIP user selects date

Step2: load all the reservation that are in the selected date and the status not equal to cancel

Step3: for each reservation, loop through the time array to find how many reservations in each hour

Step4: for each hour find availability percentage (number of reservation in specified hour/ total allowable reservation per hour) \* 100. Based on the calculated percentage the color of each hour will be changed in the UI. Green if >75% is left*,* orange if 50% is left, red if 25% is left and gray if 0% is left.

Step5: wait till the VIP user selects start time and duration and clicks on reserve button

Step6: whenever user click on the reserve button, 5 conditions will be checked

* Check if the user does not have a reservation at the selected hours.
* Check if there is an available parking spot at all selected hours
* Check if selected date is equal to today or tomorrow (selected date == current date || selected date == current date +1) as the user can only reserve at the same day or one day before the reservation date.
* Check if the reservation start time has passed, if selected date is today by checking if the selected start hour is greater than current hour
* Check if the number of selected hours with selected date’s reserved hours is less than or equal to the number of allowable reservation hours per day (6 hours). Reserved hours in the selected date are calculated by going through the user reservations

Total reservation hours += number of hours in a reservation – number of hours extended – number of hours cancelled

If one of these conditions is not met an error message will appear to the user

Step7: if all the above conditions are met then a reservation record with status = “created” and price = 5 QR per each hour will be added to the database.

Step8: notifies the VIP user 30 minutes before expiry time

Show reservations:

* Algorithm:

Step1: load all current and upcoming reservations related to the VIP user

Loop through all reservations

* If reservation date > current date (which mean no need to check the time)

check if (reservation plate number == user logged plate number && status != “cancelled” && status != “subcancelled”)

* If reservation date == current date ( time need to be checked)

For each reservation check if (reservation plate number == user logged plate number && status != “cancelled” && status != “subcancelled” && current time <= reservation start time)

Extend a reservation:

* Algorithm:

Step1: wait till the VIP user selects extend option

Step2: whenever user click on extend option, 2 conditions will be checked

* Check if the selected reservation is at its last hour (current hour == last hour of reservation) as the VIP user can only extend his reservation in the last hour of the reservation
* Check if there is an available parking spot after the reservation time

If one of these conditions is not met an error message will appear to the user

Step3: if all the above conditions are met then extend reservation by updating the following field in the database: status changed to “extended”, price incremented by 5 QR and the number of hours extended is incremented by 1

Cancel a reservation:

* Algorithm:

Step1: wait till the VIP user selects cancel option

Step2: whenever user click on extend option, 2 conditions will be checked

* If the reservation has not started (current time < start time), cancel the whole reservation by updating the following field in the database: status changed to “cancelled”, price decremented by 50 % and number of hours cancelled incremented based on reservation hours
* If the reservation has started (current time >= start time), cancel remaining reservation hours by updating the following field in the database: status changed to “subcancelled”, price decremented by 50 % for each cancelled hour and number of hours cancelled incremented based on number of hours that are cancelled

Request car care:

* Algorithm:

Step1: wait till the VIP user selects one of the car care services options

Step2: whenever user click on extend option, 2 conditions will be checked

* If the application is installed in the phone, VIP user is successfully redirected to the Servesni application (talk about it in the functional requirement section)
* If the application is not installed in the phone VIP user redirected to the Play Store

Availability percentage:

* Algorithm:

Step1: load all the zones from database

Step1: For each zone the availability percentage of spots is calculated by looping through all the spots and count how many spots are available

Step2: availability percentage is computed by following equation

availability percentage = (number of available spots (counted in step1) / total number of spots in zone) \*100

Step3: based on the computed percentage the progress bar color will be changed. Green if more than 75% of the spots are available, orange if 50% of the spots are available and red if 25% of the spots are available.

Currently looking:

* Algorithm:

We expand our database to have new relation called currently looking which has id, login time and zone name.

Step1: When user click to view a zone, currently looking record will be added to the database

Step:2 When user leave zone page, currently looking record will be deleted from the database

To handle the lost connection case, we applied the following logic:

Step1: For each zone, load all the currently looking records from database

Step2: remove all records older than 10 minutes

Step3: show how many people are currently viewing each zone

* Design:

Current Occupancy Trend:

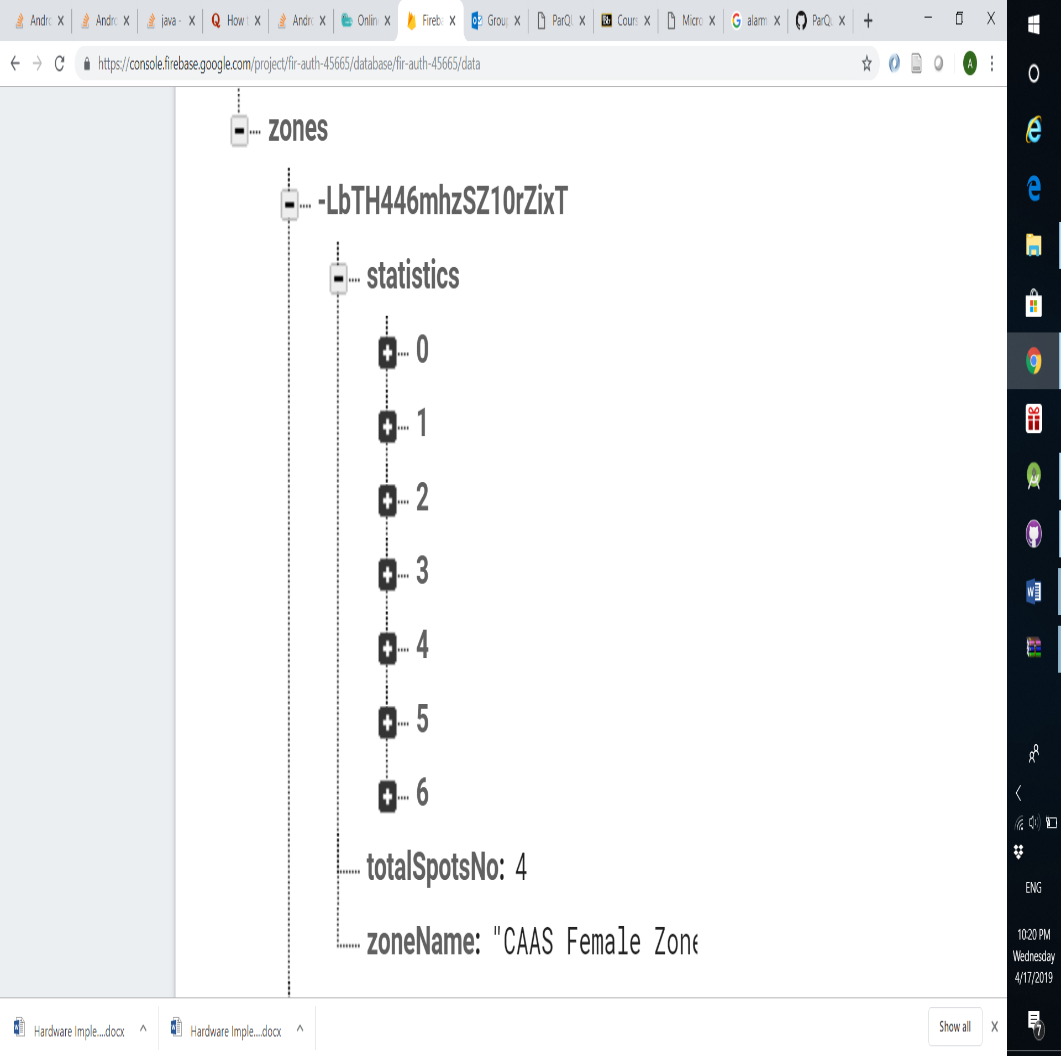
Structure of statistics field for each zone in database

Statistics is an array of object where each object storing the data for week days (Sat, Sun, Mon, etc...) (see figure)

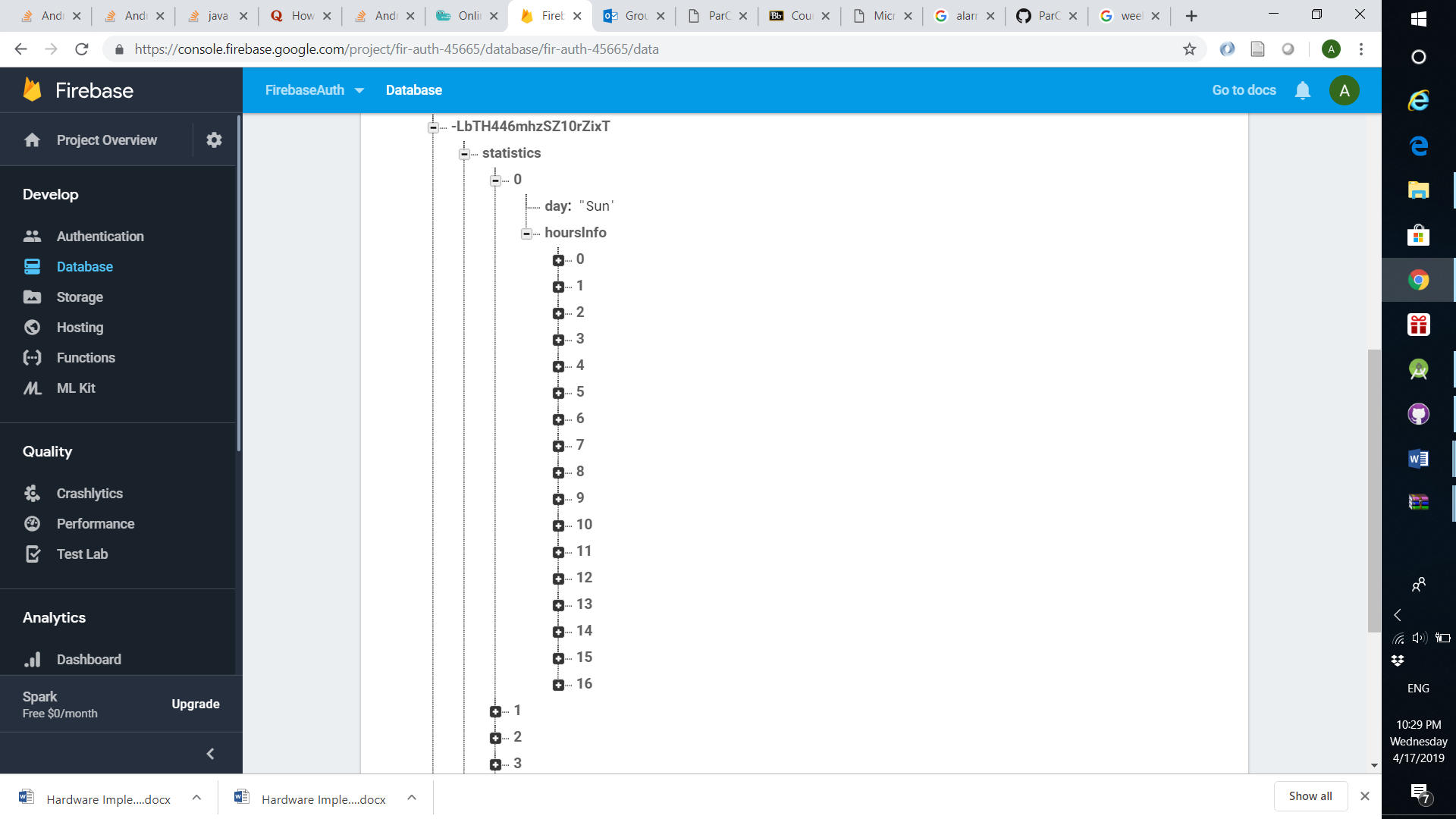
Each day object will have field called hoursInfo which is an array of object where each object storing the data for each hour (6 AM – 9 PM). For example, the first index (0) store data for 6 AM (see figure). Each hour object stores 3 data which are hour, count array and date array where count is the number of reservations in that specific hour at the corresponding date in date array. For example, in 2019-03-24 there is no reservations at 6 AM. We only storing the data for recent 4 (Sunday) in weeks







Zooming in first day object



Zooming in first hour object



Updating statistics data for reservation zones

* Algorithm:

Step1: check which week of a day the date of reservation belongs to

Step2: update all hours that are equal to reservation hours in that date by the following logic

* if date exist in date array, increment the correspond count of date by 1
* if date is not existing, remove the oldest date corresponding with its count value (always in index zero), shift other dates and counts upwards then finally add the new date and count

Updating statistics data for check availability zones

* Refer to section

Displaying data in term of histogram:

To display data as bar chart we used an Android chart view library called MPAndroidChart

* Algorithm:

Step1: wait till user selects day of a week

Step2: for each hour, occupancy percentage is calculated as following

% Occupancy = (total sum of count values / (number of dates \* total number of spots)) \* 100

Step3: use setData function from MPAndroidChart to set the calculated percentage for each hour into bar chart

* Design:

Get Directions:

* Algorithm:
* Design:

1. Currently looking implementation

Challenge:

It was quite challenging to implement this feature as the implementation logic in the application was different from the website. In the website, it was a bit easy to know how many one visit specific page by using IP address. However, this logic is impossible to be implemented in the application.

To implement this feature in the application, we benefit from a concept called "activity life cycle" where each activity goes through several stages. we benefited from onStart () and onStop (). whenever the activity gets started, onStart () method will be invoked so we should place the logic needed for increasing the number of people looking inside the onStart () method. In contrast, the logic needed for decrementing the number of people looking was placed in onStop () as this method will be called whenever the user leaves the page (activity).

After implementing this logic, we figure out that it will not work properly in the case that the user lost internet connection. Logically if the user lost the connection the number of people looking should be decremented by 1, however in reality the number stay as it is. The reason behind that was the database has not been updated as there is no internet connection.

Solution:

To solve this issue, we thought about adding new relation called “currently looking” in the database to store zone name, login time and temporary id for each access. Whenever the user closes the page, the stored record for this user will be removed. Whenever other users visiting page, 2 things will happen:

1. All the records that are older than 10 minutes will be removed.
2. Show users how many people are currently looking.

The logic used to solve this issue discussed above, refer to section ()

By applying this logic, we can give an approximate number of how many currently looking, as in some rare cases the user might stay looking in the page more than 10 minutes.

1. Current Occupancy Trend implementation:

Challenge: We calculated the occupancy percentage based on all the history data stored in the database so with time this will affect the accuracy and not reflect what really happen in real life as any change in data is not sensitive after a long duration.

Solution: We decide to apply moving average concept by calculating the occupancy percentage based on the data of recent 4 weeks

XML used as Markup Languages to design the user interface.

