|  |  |
| --- | --- |
| **Date:** | 29 October 2015 |
| **Time:** | 1840 – 2340 |
| **Venue:** | Library Study Booth 4.2 |
|  |  |
| **Attendees:** | All |
|  |  |
| **Agenda:** | 1. Review of Iteration 4  2. Code Review |

**Review of Iteration 4**

The team had overlook the importance of the test cases, should allocate more time for test case. It is difficult for PM to cover the whole test cases. PM will be in-charge but the pair programmers could help to verify the test cases too. The team has decided to remove Mastercopy in the git folder and intead only have 1 copy which is inside individual testing>iteration folder.

Longer hours for test cases will be scheduled in Iteration 5 to ensure that everyone work together to create the test cases. The team will be covering more test cases to verify the correct output in terms of validation and logic codes most importantly.

On a side note, Graphical UI test cases will be combined with the normal functionalities. Next, there will only be one test data file for each function.

Code clean be will still be implemented in Iteration 5 to mainly correct the naming convention. But the priority will be a working application.

The team is on track for Iteration 4. The schedule metric is at 1. The bug metric is 5. The bug on Top-K School functionality (rating medium) will be brought forward to Iteration 5.

Iteration 5 planning

Currently, Bootstrap functionality is accounted for the changing column but bootstrap has not been optimised. The team will be continuing to work on the optimization function. In addition to that, there will be code clean-up for the remaining functions for Top K and Advanced Smartphone Overuse. Iteration 5 will commence on 30th October, it will be 9 days and end on 7th November.

**Code Review:**

**Bootstrap**

Corrected algorithm to account for changes in columns:

* The change in column is done by checking the header field. Then based on the field value, an index is assigned to that field.
* Following that, for subsequent lines, when retrieving out lines from the CSV which comes in the form of an array, the column value is fetched based on the earlier index assigned to that field.

**Diurnal Basic App**

Instead of having multiple database calls to calculate diurnal App usage for the hour, only one database call is made instead.

Naming conventions for Basic App functions corrected.

**Smartphone Overuse Report**

Smartphone Overuse Report code Clean-up was to standardize naming conventions.

Removed test code as well as standardised date validation method for JSON and implemented a common validation method

**Smartphone Usage Heatmap**

Code was cleaned up to standardize naming conventions. Removed test code as well as standardised date validation method for JSON and implemented a common validation method.

**Bootstrap Code**

Removed test code as well as standardised date validation method for JSON and implemented a common validation method.

**Advanced Smartphone Overuse**

The advanced smartphone overuse report is classified into 4 portions: Calculating average daily duration in class, calculating average daily duration in small groups, calculating average daily duration non-productive app usage and the overuse index.

For calculating average daily duration in class, the program will first fetch out all location updates pertaining from the database. After which, all data belonging to the logged-in user is added into a separate List<Location>. Then, the data will be calculated from this List<Location> to calculate the average class time:

1. This will be done by looping through the List to find all locations that contain “SR” within their semantic places and then adding the total time together.
2. A duration spent at the location is only considered class time when the string of the semantic place contains “SR” and the total continuous duration spent within that same SR is **more than or equals to an hour**.

Next would be calculating of average duration in small groups, all the location data which was fetched earlier will be used. The algorithms works by sorting the user’s location data based on semantic place and assigning a start date and end date to it:

1. At the end of it, all data will be stored within a HashMap<String, HashMap<String, List<TimeObject>>.
2. The first key is macAddress, while the inner HashMap’s key is semantic place and the list of TimeObjects associated with it.
3. **TimeObject** is an entity used to store the continuous time spend at one semantic place by the current user. After which, the logged in user’s entry within the HashMap will be used to find out the interactions between the logged in user and the other users.

As long as the other user’s HashMap contains an entry of the semantic place that the logged-in user was at and there was an overlap of timings for the TimeObjects, TimeObjects will be created to reflect that interaction and stored within an ArrayList.

At the end of this, an ArrayList of TimeObjects between the current logged in user and the other users will be created.

This data is then stored within a HashMap<String, List<TimeObject>> where the key is the **mac addresses of the other users**. The List of TimeObjects stored are the start time and end times of all the interactions between the logged in user and the current entry of the HashMap.

Following which, TimeObjects that have overlapping timelines will be merged together and those that have a combined time of less than 5 minutes after merging are removed.

After the merging, overlapping timelines between the other users are stored.

For timelines that have 3 or more than 3 users will have their start times and end times stored in a TimeObject and then added to a List. At the end, this List of TimeObjects which are the big group timings will be compared with the times of the other users and cut out.

At the end, all the TimeObjects will be combined together to form one complete timeline which represents the small group time that the user had.

After which, all the TimeObjects start times and endtimes are calculated and added together to get the total small group time.

Combine the List of TimeObjects for the class time and small group time together.

* This will result in removal of overlapping times.
* Calculate the total time for this List of TimeObjects. This represents the total time spent in small groups and classrooms.
* After which, calculate total app usage time and minus that by the total time spend in class and small groups to get the duration of non-productive app usage.

After which, based on the numbers for each calculation, the overuse index will be generated.

**Graphical UI (Top-K Reports)**

JavaScript files used:

- jquery-2.1.4.min.js, Chart.min.js

Top K

1. A ‘data’ object is created so that labels, dataset and colour can be assigned to it
2. The context is retrieved using jQuery’s .get() method
3. Using **jQuery .getJSON()** method, the entire JSON object is retrieved using the JSON URL with parameters passed in
4. The data is retrieved via a forEach loop to retrieve each inner JSON object (TopKApp/ TopKSchool/ TopKStudent)
5. The ‘name’ (for TopKApp) or ‘school’ (for TopKSchool) or ‘name’ (for TopKStudent) in each object is pushed into labels array (x-axis) and the ‘duration‘ is pushed into the data array (y-axis)
6. After all objects have their required fields pushed into the ‘data’ object, a bar chart is created based on the context and data provided
7. A canvas is then created to display the chart on the web page

**Graphical UI (Basic App Usage)**

Chart.js library is used to display the charts.

BasicAppUsageTime.jsp

Pie chart is used for display of results. The data were extracted from calling the JSON method.

var str = 'json/basic-usetimereport?startdate=<%=request.getParameter("startDate")%>&enddate=<%=request.getParameter("endDate")%>&token=smuaAPP';

$.getJSON(str, function (results) will read the objects return from JSON call. A for loop is run through the objects that stores the data intensity count and percentages, percentage value will be retrieved from each object.

**Pie chart** data will be populated with the percentage and the labels (Intense, Normal, Mild)

var myNewChart = new Chart(ctx).Pie(data);

BasicAppUsageTimeBreakdown.jsp

Arrays will be created to store the values of the possible choices that the user can choose to breakdown by.

Next, values will be populated into drop-down list based on the user’s category input.

**Pie chart** is used for display of results. The data were extracted from calling the JSON method.

var str = 'json/basic-usetime-demographics-report?startdate=<%=request.getParameter("startDate")%>&enddate=<%=request.getParameter("endDate")%>&order=' + order + '&token=smuaAPP';

$.getJSON(str, function (results)

For every loop, the attribute will be retrieved from each object. The attribute retrieved will be the label such as Gender, Year, School, CCA. createData(concat, data, label, area) was created to facilitate the **recursive method**, as it will keep breaking down further, depending on the user’s choice.

If it is the first level of breakdown, it will run through a for-loop to retrieve out all the values from each sort key from the JSON. All these results will be concatenated, it will continue to call **createData(concat, data, label, area)** till it finishes breaking down.

If it is not the first level of breakdown, it will means that it is the last level, where the intensity usage will be stored.

Arrays will be created to store the values of the possible choices. Upon clicking the breakdown button, **breakdownfunction()** will be called. Data will be displayed by using the if-else loop in a for loop to display the output that the users have inputted.

BasicAppUsageDiurnal.jsp

Line chart is used for display of results. The data were extracted from calling the JSON method.

var str = 'json/basic-diurnalpattern-report?date=<%=request.getParameter("date")%>&genderfilter='+genderFilter+'&yearfilter='+yearFilter+'&schoolfilter='+schoolFilter+'&token=smuaAPP';

$.getJSON(str, function (results)will read the objects return from JSON call. A for loop is run through the objects that stores the period and duration, period and duration will be retrieved from each object to be displayed as x and y axis respectively.

**Line chat** data will be populated with the percentage and the labels (Intense, Normal, Mild)

var myNewChart = new Chart(ctx).Line(data);

BasicAppUsageCategory.jsp

**Pie chart** and **Bar chart** are used for display of results.

* The bar chart x-axis will be the label of app category, and the y-axis will be the average usage time.
* The pie chart will be the percentages of the average usage time.

All data were extracted from calling the JSON method.

var str = 'json/basic-appcategory-report?startdate=<%=request.getParameter("startDate")%>&enddate=<%=request.getParameter("endDate")%>&token=smuaAPP';

$.getJSON(str, function (results)

$.getJSON(str, function (results)will read the objects return from JSON call. The first for loop will run through the objects that stores the category name and category duration, category name and category duration will be retrieved from each object to be displayed as x and y axis respectively for the Bar chart.

The next for loop will run through the objects that stores the category-percent, for the display in the Pie chart.

**Graphical UI (Heatmap)**

1. Pdf file is converted to **svg** before including in the code
2. Colour block class is used to display the legend based on main.css
3. A ‘data’ object and a colour array is created
4. Using **jQuery .getJSON()**, the JSON object is retrieved
5. Using forEach, the inner JSON object is obtained and its ‘semantic-place’ is stored as id and ‘crowd-density’ is stored as density
6. The id is stored in ‘data’ with value as the JSON object so that the object can be retrieved in the mouse-over method to display the semantic place, number of people using phone and crowd density
7. The respective level of heatmap displayed will be the one corresponding to the level the user selects, and the page will be included using **jsp:include**
8. if element id corresponds to id, the respective values will be displayed for the particular mouse-over area
9. Also, by using .css(), the colour with index == crowd-density in the colour array will replace the fill-colour for every id

The meeting was adjourned at 11.40pm. These minutes will be circulated and adopted if there are no amendments reported in the next three days.

Prepared by,

Koh Chu Qian

Vetted and edited by,

Tang Shing Hei