|  |  |
| --- | --- |
| **Date:** | 18th September 2015 (Friday) |
| **Time:** | 1PM |
| **Venue:** | SMUX MEETING 2-2 |
|  |  |
| **Attendees:** | ALL |
|  |  |
| **Agenda:** | 1. Use Case Diagram 2. Domain Diagram 3. Use Case Scenario 4. SD |

**Use Case**

After discussion, the actors for use cases are User, Admin and JSON.

JSON will serve as a third party system to retrieve data from our application.

The use cases are mainly the main functionalities and for the main functionality such as Basic App Report, we will be splitting it into the sub tasks as the individual use cases.

**Domain Diagram**

The issue that we discussed was that every demographic users will be inside the location database table. However we are unsure if we are adding the link between demographic and location table because not all the macaddress from location will be found in the demographics.

In java context, we may not need the demographic attritube in the location attribute.

**Use Case Scenario**

After discussion, we will have in total of 15 use cases.

Login, Bootstrap Data, Add Data, View Basic Usage App Report (Breakdown by usage time category), View Basic Usage App Report (Breakdown by usage time category and demographics), View Basic Usage App Report (Breakdown by app category), View Basic Usage App Report (Diurnal pattern of app usage time), Top-k App Usage Report (Top-k most used apps (given a school)), Top-k students with most app usage (given an app category), Top-k schools with most app usage (given an app category), Smartphone Overuse Report (Advanced), Smartphone Usage Heatmap, Smartphone Usage Heatmap, Social activeness report, Delete user by web UI.

**Sequence Diagram**

**Bootstrap SD**

|  |  |  |
| --- | --- | --- |
| **Topic** | **Discussion** | **Conclusion** |
| Wipe table sequentially or wipe schema during bootstrap | 1. Started with the choice of wiping individual tables separately 2. Suggested to wipe schema 3. Decided that it was not appropriate because if the schema is wiped, if any of the tables had issues during uploading of data, all data in subsequent tables would also have been wiped. 4. Came to the conclusion to wipe a table just before doing validation for data which is going to be inserted into that particular table | Including a loop to wipe a table before validating and inserting the data which that table is supposed to contain |
| Using HashMap, SQL query or ArrayList during validation of data | 1. Suggestion to use SQL query to validate every row of data 2. Suggestion to use ‘select’ statement to look for record to check for duplicates 3. Discussed that it was not feasible because doing so also means committing every row of data separately (started discussion on doing batch commit) 4. Following up, we started discussing on whether to use HashMap or ArrayList for the validation 5. Suggestion to use HashMap instead of ArrayList because there will not be a need to loop (required for ArrayList) and HashMap compares the keys by memory. 6. Discussed about what the key would comprise. The key will contain the timestamp concatenated with mac address | Use HashMap during validation of data |
| Doing batch commit or doing commits for every row of data | 1. While discussing about the above topic, we also discussed about committing every row of data separately after doing validation for each row 2. But doing so would be inefficient as compared to committing a large amount of data at once 3. Decided to store into HashMap first, subsequently doing batch commit once all data has been validated and stored into the HashMap | Do batch commit |
| Creating interfaces for DAO and Validator class | 1. Suggestion to create interfaces for the respective classes 2. Discussed about how the all DAO classes (e.g. DemographicDAO and AppDAO) have the same method such as loadData(), likewise for the Validator classes 3. Using interfaces would then organise the code better, but after much consideration, we decided to not implement interfaces | Maintain all the DAOs and Validator classes as separate entities without implementing an interface |

**Basic App Usage: Breakdown by Usage Time Category SD**

|  |  |  |
| --- | --- | --- |
| **Topic** | **Discussion** | **Conclusion** |
| The number of DAOs we should have for the given data, and whether to have DAOs for each sub function | 1. Discussed about whether it is necessary to have one DAO per function as well as where to place additional functions such as sorting 2. Decided on having 3 DAOs: AppDAO, LocationDAO and DemographicDAO. 3. Also decided that sorting will be included in either of the above DAOs as multiple functions will be accessing the same DAO | Finalised on 3 DAOs and including sorting methods in the respective DAOs |
| Doing constructor chaining in order to incorporate similar functions into the existing classes in this function or to have separate controllers and entities | 1. Suggested to do constructor chaining because some functions require some same data (e.g. Location data). 2. Discussed that although the functions may use the same data, user input varies and cannot be chained (e.g. diurnal only needs a date while time category requires start date and end date) | Have separate controllers and entities for every function |

**Basic App Usage: Breakdown by Usage Time Category and Demographics SD**

* Completed with similar considerations as the ‘breakdown by usage time category’ SD

**Diurnal Pattern**

DAO

Fetch out appData

List out the macadress who fulfils the condition

Use sql “select \* from app inner join demo where

If (school) {

school=?,

}

If (gender) {

gender=?

} order by timestamp, MacAddress

Controller

Counter = 1 (once It hits 12) looping stop

For each timeframe

We need have counter for user and the total app usage time

Avgsum/user

Put the results into HashMap <String(time), avg(integer)

**SmartPhone Usage Heatmap**

locationLookIp

fetch all the semantic place and store it into HashMap<id, place>

For each MacAddress

If (app.containskey(macaddress)) {

String pace = getValue(id)

Int I = getNumOfPeople

put(place,int)

}

LocationLookUp<id,place> (the different places has to be placed inside Hashmap<Place,Int>

HashMap<Place,Int>

**Task assigned:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | **Due Date** |
| 1  2 | Edit on the SD  Clarification on the Domain Model with supervisor | All  All | 19 September  22 September |
|  |  |  |  |

The meeting was adjourned at 8 pm. These minutes will be circulated and adopted if there are no amendments reported in the next three days.

Prepared by,

Jennifer

Vetted and edited by,

Koh Chu Qian