CSCI 3901
Final Project

COVID-19 pandemic is contact tracing

Name	Banner Id	Mail id
Bommera Geetanjali	B00881511	gt584369@dal.ca

Table of Contents

1. Test cases..... 3

1. Input Validations..... 3
2. Boundary cases..... 3
3. Control flow..... 4
4. Data flow..... 4

2. Database Schema..... 5

3. Queries..... 5

4. Test Plan..... 6

5. Algorithm..... 6

6. Assumptions..... 6

1. Test cases

1. Input Validations

a) Null value as path to config file for both MobileDevice class and Government class.
b) Individual id as null for recordContact method in MobileDevice.
c) Testhash as null for postiveTest method in MobileDevice.
d) Testhash as null for recordTestResult in Government.
e) Null value as initiator for mobileContact method in Government
f) Null value as contactInfo for mobileContact method in Government.
g) Date as negative value for recordContact method.
h) Date as negative value for recordtestresults method.
i) Density is 0 for findGatherings method.

2. Boundary cases

a) **MobileDevice(String configFile, Government contactTracer)**
   i. configFile as empty
   ii. configFile consists of one line
   iii. More than two lines of data in the configFile
b) **recordContact(String individual, int date, int duration)**
   i. Date as 0.
   ii. Date as 1.
   iii. Date is future to the current date.
   iv. Duration greater than 1440 minutes(more than a day).
   v. Individual as already hashed value.
c) **positiveTest(String testHash)**
   i. Testhash as empty string
d) **Government(String configFile)**
   i. configFile as empty
   ii. configFile consists of one line
   iii. More than two lines of data in the configFile
e) **mobileContact(String initiator, String contactInfo)**
   i. Contactinfo has only one contact
   ii. Initiator as one of the contact in contactinfo.
   iii. Only one contact that is initiator.
   iv. no contacts in contactinfo
f) **recordTestResult(String testHash, int date, boolean result)**
   i. Special characters in the testHash.
   ii. Date is future to the current date.
   iii. Date is 0.
   iv. Date is 1.
g) **findGatherings(int date, int minSize, int minTime, float density)**
   i. Date is future to the current date.
   ii. Date is 0.
   iii. Date is 1
   iv. minTime is 0.
   v. minTime is 1.
   vi. minTime is 1440(a day).
   vii. minTime is greater than 1440.
   viii. minSize is 2.

ix. minSize greater than 2.
x. Density is 1.
xi. Density is greater than 1.
xii. Density is total number of people.

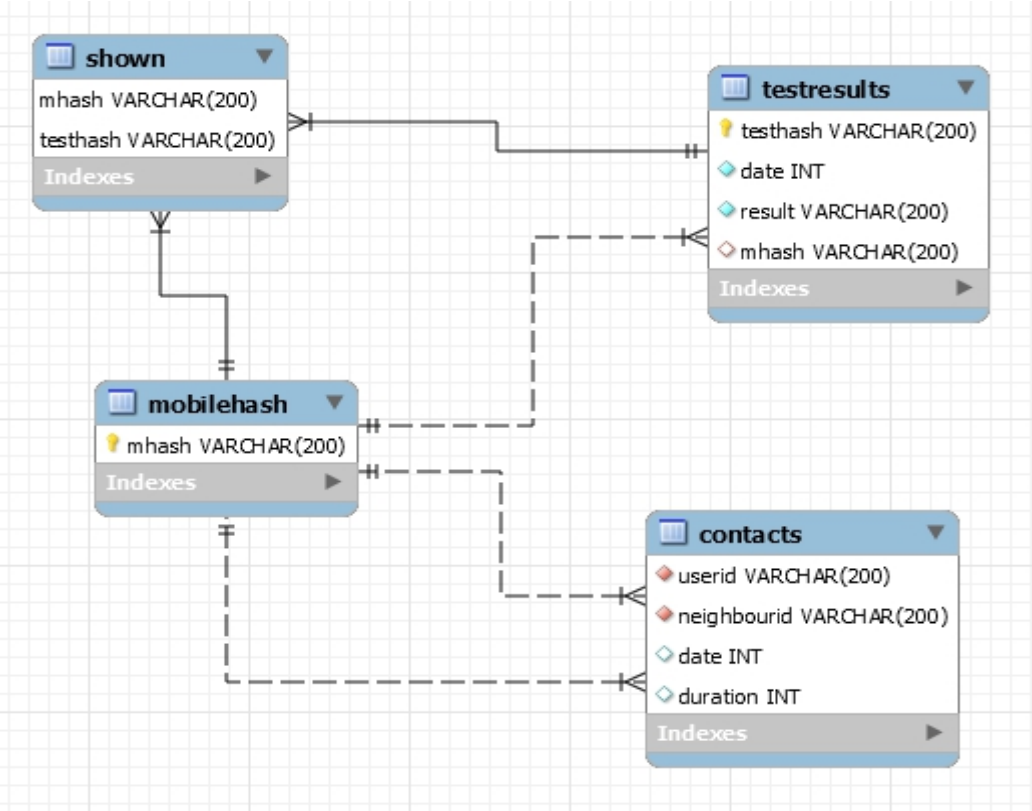
3. Control flow

a) **MobileDevice(String configFile, Government contactTracer)**
   i. configFile consists of device address and name in the given format.
b) **recordContact(String individual, int date, int duration)**
   i. 1<date<currentdate
   ii. 0<duration<1440
   iii. Correct hash of the device
c) **positiveTest(String testHash)**
   i. testHash that is store in DB for government.
   ii. Unique testHash every time.
d) **Government(String configFile)**
   i. configFile consists of correct details to connect to DB.
e) **mobileContact(String initiator, String contactInfo)**
   i. ContactInfo with at least one contact and positivetesthash.
   ii. contactInfo with more than 1 contact.
   iii. contactInfo with no postivetesthash
   iv. Initiator as hash of own device.
f) **recordTestResult(String testHash, int date, boolean result)**.
   i. 1<date<currentdate
   ii. Result as true.
   iii. Result as false.
   iv. testHash already exists in Database.
g) **findGatherings(int date, int minSize, int minTime, float density)**
   i. Provide data such that results no gatherings.
   ii. Provide data such that it finds gatherings.
   iii. 1<date<currentdate

4. Data flow

a) Postivetest and synchronizedata is called before recordtestresult method.
b) Synchronizedata is called twice.
c) Synchronizedata is called before postivetest and after postivetest.
d) Findgathering is called at the end and start of all the methods.
e) No method in mobiledevice class is called after recordtestresult.
f) No method in government class is called.
g) Synchronizedata with no postivetest
h) Synchronizedata with no contacts

2. Database Schema



a) Keys
   i. Mhash - Mobile hash of the user.
   ii. Testhash - testhash of the user
   iii. Date in testresults - date of test is taken
   iv. Date in contacts - date neighbour contacted user
   v. Result - result of the test
   vi. Userid - user id
   vii. Neighbourid - id of neighbors who got in touch with user
   viii. Duration - duration of contact with neighbour

b) Tables
   i. Shown - for each mhash stores testhash whose result is already informed to the user.
   ii. Mobilehash - consists of all the mobile hashes
   iii. Contacts - stores the contacts of each user along with duration
   iv. Testresults - stores the result of tests of every test taken.

3. Queries

a) CREATE TABLE mobilehash (mhash VARCHAR(200) PRIMARY KEY NOT NULL);
b) CREATE TABLE contacts (userid VARCHAR(200) NOT NULL, neighbourid VARCHAR(200) NOT NULL, date INT, duration INT, FOREIGN KEY (userid) REFERENCES mobilehash (mhash), FOREIGN KEY (neighbourid) REFERENCES mobilehash (mhash));
c) CREATE TABLE testresults (testhash VARCHAR(200) PRIMARY KEY NOT NULL,date INT NOT NULL, result VARCHAR(200) NOT NULL, mhash VARCHAR(200), FOREIGN KEY (mhash) REFERENCES mobilehash (mhash));

d) CREATE TABLE shown (mhash VARCHAR(200) NOT NULL, testhash VARCHAR(200) NOT NULL, FOREIGN KEY (testhash) REFERENCES testresults (testhash),FOREIGN KEY (mhash) REFERENCES mobilehash (mhash), PRIMARY KEY (mhash , testhash));

4. Test Plan

i. Internally documented the requirements and constraints of both the classes pictorially (plan.pdf) and implemented each method one by one accordingly.
ii. All the test cases mentioned in the input validations,boundary cases, control flow and data flow are tested.
iii. At each insert in the java program, checked the data in the tables with the help of debugging.
iv. Test cases are created to check the functionalities of each method.
v. Program is tested such that it triggers exception that may raise and handled in the code.
vi. Test.java consists of main method that helps in testing the program functionality.

5. Algorithm

a) Recordcontact method from mobiledevice class store all the contact information in the class.
b) Postivetest method stores the testhash if tested positive.
c) Synchronize data converts information to XML format string. This XML format string data is retrieved by mobilecontact method in government class
d) Mobilecontact method inserts all the information to Database. Find if any contact is tested positive, stores this information in shown table and returns true.
e) Shown method in government class helps to show only new contacts if tested positive.
    i. Findgathering method
        1. Retrieve all the contacts from contacts table on given date and duration greater than mintime and store these details in 2d arraylist.
        2. Form a hashmap with keys as all the user ids and values as list of contacts user interacted.
        3. Iterate through the result set and this hashmap to find the intersections of each pair in result set. Proceed if size of intersections is greater than minsize
        4. Count individual interactions for each pair and find whether count is greater than or equal to density. If yes, increment total gatherings.

6. Assumptions

a) Recordtestresult method is called first once test is taken to store the result before any method in mobiledevice class.
b) Data in configfile of both the classes is always true.
c) Considered Testhash as already hashed string hence not hashed again anywhere in the program.