FINAL PROJECT - SOLUTION

Submitted By,

Dhruv Doshi
B00883311
Dh722257@dal.ca

PREREQUISITES:

- Create the config file for Government to fetch details.
- Create the config files for different Mobile Devices to fetch data.
- Setup MYSQL – JDBC connection.
- JVM setup

DATABASE:

There are three tables used named test, info_contacts and info_person.

Test Table:

- Stores the data which is inserted by recordTestResult (String testHash, int date, boolean result) method of government class.
- This table includes fields like test_hash, test_date and test_result in which test_hash is the primary key and test_date is in integer format whereas test_hash and test_result is in varchar type.

Field	Type	Null	Key	Default	Extra
test_hash	varchar(500)	NO	PRI	NULL	
test_date	int(11)	NO		NULL	
test_result	varchar(10)	NO		NULL	

Info_person Table:

- Major database to store the information of the people who are tested positive.
- There are two fields in this database named id and test_hash where test_hash works as the foreign key and id + test_hash is the composite primary key.

Field	Type	Null	Key	Default	Extra
id	varchar(500)	NO	PRI	NULL	
test_hash	varchar(500)	NO	PRI	NULL	

Info_contacts Table:

- This table is used to save the contacts for each person. All the information regarding the tree which shows that which person had been in contact with the whom.
- There are four fields in the database named id, id_validator, duration, and date.
- Here we are using 3 factor composite primary made with id, id_validator and duration as this will be unique each time and would reduce the concurrency.

	Field	Type	Null	Key	Default	Extra
▶	id	varchar(250)	NO	PRI	NULL	
	id_initiator	varchar(500)	NO	PRI	NULL	
	duration	int(11)	NO		NULL	
	date	int(11)	NO	PRI	NULL	

JAVA PROGRAM:

As per the document here I had developed two java class named MobileDevice and Government to implement the solution. A Junit test named finalTest is also created for testing.

Execution of the Program:

- Create a government object with the config file.
- Create MobileDevice object for multiple devices.
- Record contact and positive test for mobile devices
- Invoke recordTestResult (String testHash, int date, boolean result) t from the Government Class.
- Synchronize all devices.
- Invoke the findGatherings (int date, int minSize, int minTime, float density) method.

Class 1: MobileDevice Class:

I. public class MobileDevice (String configFile, Government contactTracer)
   a. Constructor for the class.
   b. Accepts data such as address, devicename and configuration file.
   c. Store the data into specific id.
   d. Using SHA-256 encryption the hash is created.
   e. If the passed argument is null, then it will take configFile as the file name for new government object.
   f. It will try to get the address and deviceName from the config file.
   g. Final if is the concatenation of devicename and address string.

II. public void recordContact (String individual, int date, int duration)
   a. When our Bluetooth driver (not part of this project) detects another device, that driver calls recordContact to locally record that individual (an alphanumeric string) was near us on date for duration minutes.

III. public void positiveTest (String testHash
   a. IF the user is updated with the covid positive then the entry needed to be updated
   b. This single function will update the user specific information to the positive

IV. public boolean synchronizeData ()
   a. This method is used to synchronize the data at the end by creating the XML file which would be further used.
   b. This XML file holds all the required data for the upcomming functions.

Class 2: Government class:

I. public boolean mobileContact (String initiator, String contactInfo)
   a. This function deals with most of the database work in the project
   b. We will be using the password and username from the config file for the database connection.
   c. It inserts the data into the info_person database with the query.
   d. It inserts the data into the info_contacts database with the query.
   e. With the help of test_date we could check for the 14 days whether they are over or not.

II. String bytesToHex (byte [] hash)

a. This function converts the bytes hash to the hex codes.
b. And return hex string.

III. public void recordTestResult (String test_Hash, int date, boolean result)
a. Runs query to check the user is positive for covid-19 or not.
b. The data will be fed using the query.
c. We will be using the password and username from the config file for the database connection.

IV. public int findGatherings (int date, int min_Size, int min_Time, float density)
a. This function finds the gathering number.
b. date is counted from 1st January 2021.
c. min_Size is the size of the cluster.
d. min_time is the time spent with the potential positive person.
e. density is derived from special mathematical equations n(n-1)/2 where calculating c/m.
f. Here we are using two sets to save the information and compute on that specific information.
g. If the duration is more than the min_time, then the cluster is added into the adjacency list.
h. At the end we will be comparing the output with the density given according to the function

Testing File:

This file contains the Junit test cases. File name finalTest.java

TEST CASES:

Input Validation Cases:

- public class MobileDevice (String configFile, Government contactTracer)
  - If configfile is empty or null.
  - If id field is empty or null.
  - If contact tracer is null.
- public void recordContact (String individual, int date, int duration)
  - If individual is empty or in form of integer.
  - If date is negative or 0 or in float or double.
  - If duration is negative.
- public void positiveTest (String testHash)
  - If testHash is empty or null.
  - More than one argument passed.
- Public Boolean synchronizeData ()
  - If id is null or empty.
  - If no contacts are recorded
  - Invalid information passed in argument.
- Government (String configFile)
  - If config file is empty or null.
  - If config file's information is false or not valid.
  - Incorrect Password.
- mobileContact (String initiator, string contactinfo)
  - No contact details in XML file.
  - Initiator and Id are different.
- public void recordTestResult (String test_Hash, int date, boolean result)
  - Invalid test hash or not present.
  - Date lower than 0 as negative.
  - Result in form other than Boolean.
- public int findGatherings (int date, int min_Size, int min_Time, float density)
  - If date is less than 0
  - If min_size is less than 0
  - If min_time is less than 0
  - If float density is less than 0

Boundary Case:

- public class MobileDevice (String configFile, Government contactTracer)
  - If contact tracer is null.
  - If configfile is empty or null.
  - If id field is empty or null.
- public void recordContact (String individual, int date, int duration)
  - duration is more than 1440 / a day.
  - date is 1.
  - date is higher integer.
  - duration is 1.
  - individual string is just a character.
  - individual string is out of bound.
- public void positiveTest (String testHash)
  - Testhash is out of context of SHA-256.
  - If testhash is extremely long string
  - Testhash is of single character.
- Public boolean synchronizeData ()
  - No test result recorded at the end of synchronization.
  - If no contact recorded to synchronize data.
  - Contacts are recorded before synchronization.
  - Concurrency in test result with same id.
- Public mobileContact (String initiator, string contactinfo)
  - End of file is not defined.
  - No details in contactinfo
  - Multiple test result and string in the contact info
- public void recordTestResult (String test_Hash, int date, boolean result)
  - test hash is not a legal string.
  - Date is out of bound.
  - Testhash is out of bound in terms of length.
  - If date is 1 for all instances.
- public int findGatherings (int date, int min_Size, int min_Time, float density)
  - data parameter violates boundary rules.
  - For density id the input is 1 then the subsets would be there for each gathering.

Control Flow Cases:

- Government (String configFile)
  - concurrency in the configfile name – called with the existing file name.
  - Called after the other classes.

- public class MobileDevice (String configFile, Government contactTracer)
  - When MobileDevice is called more than once for same object with different configuration.
  - MobileDevice is called for single time for the same object but with invalid configuration.
  - MobileDevice is called with the existing filename over and over.

- public void recordContact (String individual, int date, int duration)
  - Called after the synchronization will make concurrent data.
  - Badly write when called after synchronization.
  - For 2 contacts one is before sync and one after sync would change the database dependency.

- public void positiveTest (String testHash)
  - For this the testHash should be called before in the government recordtestresult and the testhash should be available in the set there.

- public void recordTestResult (String test_Hash, int date, boolean result)
  - should be called before the synchronization to successfully save the data.
  - If called after the synchronization could create a dummy process and will not write anything.
  - Dummy write will violate SQL constraints as there are primary and foreign key associated with some parameters.

- Public boolean synchronizeData ()
  - Should be called once till that time all the data needs to be saved in the sets and arrays. At specific time, every day the whole information could be transferred to the database to reduce the risk of concurrency.

- Public mobileContact (String initiator, string contactinfo)
  - XML string for test result verifications and contact details.
  - Stores testhash which should be done after the positive information is validated.
  - Check whether the user had been convicted with covid in last 14 days then true else false.

- public int findGatherings (int date, int min_Size, int min_Time, float density)
  - Method must be called after recording all the information else will give the wrong answers.

Data Flow Cases:

- The mobile device constructor is called for more than one time to create multiple objects.
- Record contact method is called for the devices.
- The positive call method is called whenever needed.
- All the created devices call for synchronize data.
- All synchronize data call mobilecontact with the generated xml string.
- The government constructor is called with new object.
- The mobile device constructor is called for the first time to create an object.