CSCI 3901 Project

Name: Janvi Ghanshyambhai Patel
Dal ID: B00863421
Dal Email: jn410076@dal.ca
Date: 13/12/2020

Overview

Our main aim is to design and develop a tool which helps us to manage the COVID-19 pandemic. This system is being used by both the end user and government. End user enters the details of contact, covid positive test result which is being stored in government data. Here we are using a hashcode system to hide the identity of the user. Agency also provides test result inputs to verify the hashcodes.

System has major 2 features. One is to identify the contact with the person who is COVID-positive allows us to notify the mobile user which is helpful in terms of alert and gives ideas to get tested and start quarantine. This limits the spread of disease faster. Second feature is to use contact information to detect the frequency of large gatherings on a particular day. This helps us to understand the community's compliance with physical distancing advisories.

Files details

Program contains the following files:

1) mainUI.java :- It is basically designed for asking the user for the input which includes add mobile device, contact device, test result, synchronizeData, record test result, find gatherings, quit options to connect with methods.

2) Government.java:-
   a) Government(): Performs configuration file reading operations and calling connectionManager class to establish connections.
   b) readXML(): reading content from XML file and storing it to document
   c) mobileContact(): After getting a document from readXML() file it Inserts contact details and positive test hash details into database and calls notify_mobiledevice function() to notify users about covid-19 contacts.
   d) notify_mobiledevice(): using sql queries it finds out whether or not to notify the mobile device user about being contacted with other COVID-19 contacts.
   e) recordTestResult(): It records the test result provided by the agency to the SQL database.
   f) findGatherings(): Find the number of gatherings based on day, number of people, density and duration of meeting.

3) MobileDevice.java:-

    a) MobileDevice(): checks valid name of configuration file and calls mobilehashGenerator() and stores the details of Government objects.
    b) MobilehashGenerator(): reads the configuration file and after validating the input it generates the hashcode.
    c) recordContact(): inserting the details of contact hash code, date and duration into the data structure.
    d) positiveTest(): inserting the details of positive test hash to data structure.
    e) synchronizeData(): calls mobileContact() into government class with the details of initiator and xml file name.
    f) xml_contentFormation(): designing xml file content as String including all tags and values.
    g) xml_fileOperations(): xml file creation and writing to xml file.

4) JUnitTests.java:- It includes test cases of Mobile device class and Government class using assertTrue, assertFalse and assertEqual functions:

    a) filename_null_empty(): Testing configuration filename into mobile device class.
    b) configurationFile_notexist(): Testing configuration file does not exist in mobile devices.
    c) filename_valid(): Testing by providing a valid file name for mobile device class.
    d) recordContact_inputvalidation(): input validation for record contact for mobile device class.
    e) positiveTest_inputvalidation(): positive test testing for mobile device class.
    f) synchronize(): Test set for synchronize in mobile device class.
    g) Government_recordTestResult(): Test cases for recordTestResult for government class.
    h) Government_findGatherings(): Test cases for finGathering for government class.

5) contactDeviceDetails.java:- It contains the details of contact, date and duration.
6) ConnectionManager.java:- It performs SQL connection tasks surrounded with try catch blocks.
7) ConfigurationFile_Government:- It contains username,password and database details separated by equal sign without space.
8) ConfigurationFile_MobileDevice:- It contains address, deviceName details separated by equal sign without space.
9) Project_SQL: It contains the details of creation, deletion and selection of tables queries.
10) contactInfo.xml: It contains the details of mobile device tag, Initiator block, individual tag, date tag, duration tag, positive hash tag if present, closing tags.

DALHOUSIE UNIVERSITY
Inspiring Minds

Data Structures, Abstract Data Type and how they are efficient

----

Hash Map: Hash is used as a data structure for storing the details of contact details and positive test hash. Where the key is mobile device hash and values are contact device details or positive hash. I have used adj_list hashmap to store contact using key and value into findGathering.I can easily trace the connected devices,positive test and have considered every mobile device as key.

Example,

recordContactMap: Key: Device C hashcode
Values: [[Device A hashcode, date, duration], [Device B hashcode, date, duration}}

positiveTestdetails: Key: Device C hashcode
Values: {A1,B1,C1}

adj_list: Key: Device A hashcode
Values: [Device B hashcode, Device C hashcode, ...]

As HashMap contains a key and a value, I represent device hash as keys and their contact details as list in values. A Map is designed for the faster lookups. Data is stored in key-value pairs and every key is unique which in this case is very suitable.

List: Program reads the contact details and stores them in a list. It is used here because list does not require pre-defined size, we can dynamically increase the while reading. For the provided example below is how I store it into a list. Using lists, I can easily check and store the details of contactDevice and positive hash.

List< contactDeviceDetails > collection_contactDeviceDetails:
[[Device A hashcode, date, duration], [Device B hashcode, date, duration]]
List< String> collection_positiveTestDetails : [A1, B1, C1]

List<String> Individual_list: This list is to keep track of contact devices of this device. I am adding the record contact hash into just to keep track while notifying the user about whether the device is being contacted.

List<List<String>> duplicateHashSet is used to remove duplicate Set 'S' from counting.
List<String> hash_Set is used as Set 'S' which indicates that Device A and B both contacted the device into hash_Set and including A,B.
List<String> final_Set is used to handle the final counted sets after considering the density, size and which is further used to avoid the counting of subSet in gatherings.

Set: Set is used to store a keySet of hashMap which will be always a unique set of keys.

Steps taken for Effective Design and system:

- For the Data privacy implementation, I have defined all the classes locally used by a sub method as private and method which is being used by other methods as protected method.

- Also measured performance to find large gathering, I am not locally storing all the connections and not storing the details of density. I am filtering the and excluding the unnecessary storage and cutting down the unnecessary records.

- Using JUnit it was easy for me to check every step I wanted to perform and that matched without output. Furthermore, I have created below test cases and checked them which provided the correct output.

- Also provided exceptions where needed such as connection fail, bad input, file path not exist, different file format, not a valid input of configuration file etc.

- I have performed the problem 2 query first into mysql to make it easy to understand the output which is being tested there.

- Also tested the queries based on different scenarios where tested by a single customer, office and employee and using different periods.

- Design program has high cohesion as every method contains only it's responsibility, low coupling as following interaction:

    - Interaction: ConnectionManager with Government using one method
    - Interaction: MobileDevice with Government class using one method used getter setter.
    - Interaction: contactDeviceDetails with MobileDevice which is being used as Object

- From the SOLID principle, it follows Single responsibility for each method, Liskov substitution principle for List, Set, Map and Dependency inversion for less direct interaction.

- I have designed in such a way that no one can miss any interaction or information.

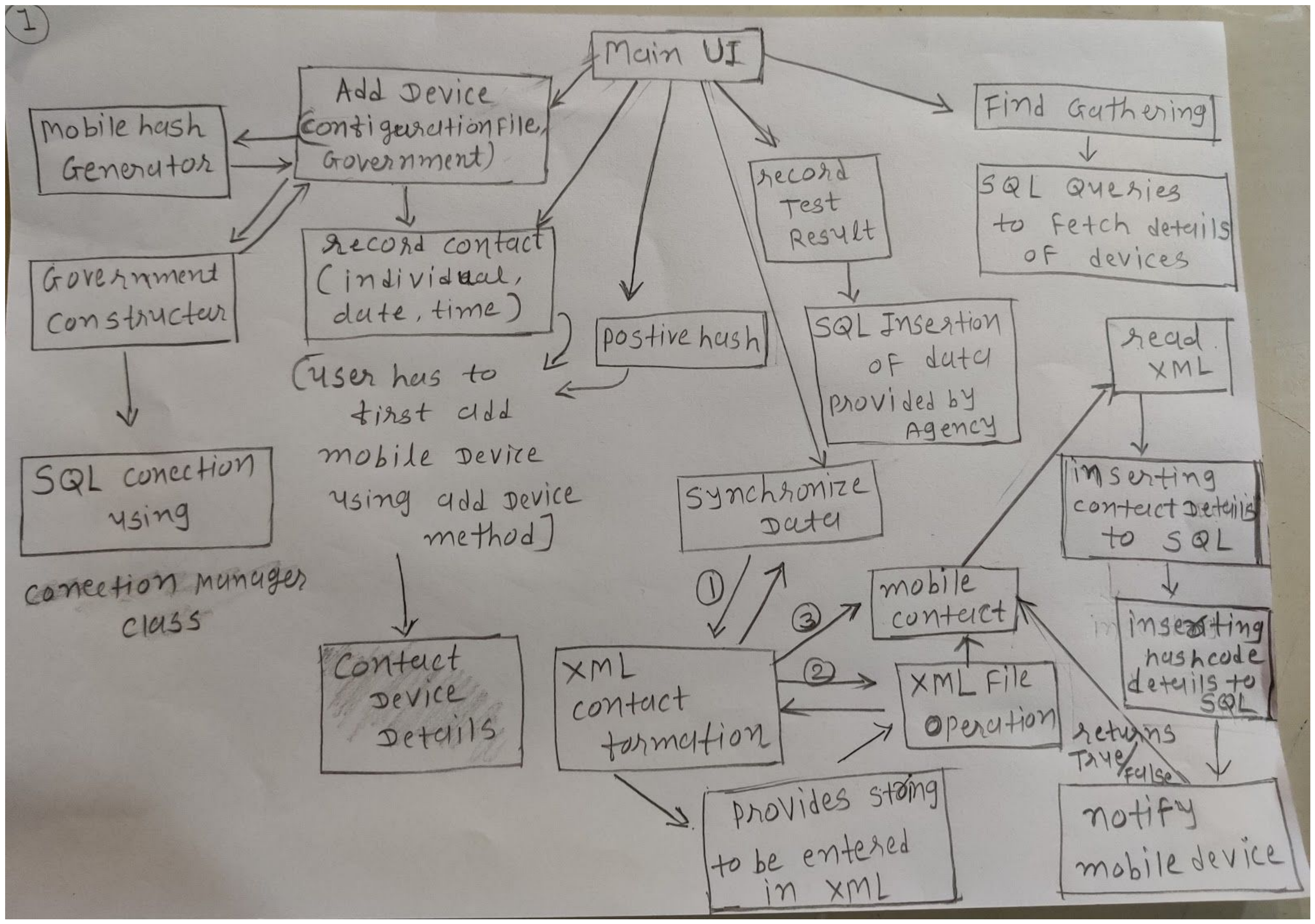
- Easy to Maintain

- Easy to debug

- One can Include further changes easily.

Design elements, Key algorithms and methods

Block diagram:



1
Main UI
Add Device (configurationFile, Government)
Mobile hash Generator
Government Constructer
SQL conection using
conection manager class
record contact (individual, date, time)
(user has to first add mobile Device using add Device method)
Contact Device Details
postive hash
record Test Result
SQL Insertion of data provided by Agency
Find Gathering
SQL Queries to fetch details of devices
read XML
inserting contact Details to SQL
inserting hashcode details to SQL
notify mobile device
Synchronize Data
1
2
3
mobile contact
XML contact formation
XML File Operation
returns True/False
Provides string to be entered in XML

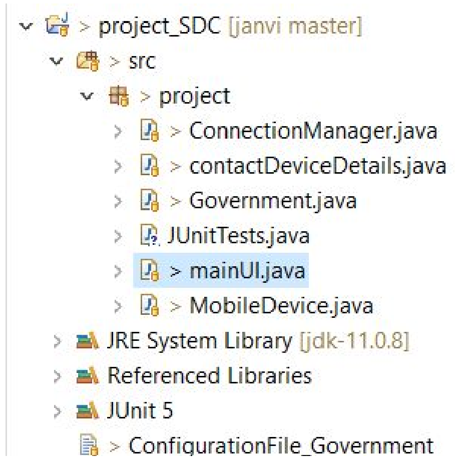
Note: This class diagram may not contain all the details of the program; it is just an overview.

MainUI:

I have provided a menu to user as below:

addDevice <space configuration file names>
contactDevice <individual> <date> <duration>
testResult<Positive test hashcode>
synchronizeData
recordTestResult<testHash> <date> <result>
findGatherings<date> <minsize> <minTime> <density>
quit

By entering every command method call is being performed. I have provided configuration file name equals ConfigurationFile_Government and in the main menu I am creating the object of Government class.



```
project_SDC [janvi master]
  src
    project
      ConnectionManager.java
      contactDeviceDetails.java
      Government.java
      JUnitTests.java
      mainUI.java
      MobileDevice.java
  JRE System Library [jdk-11.0.8]
  Referenced Libraries
  JUnit 5
  ConfigurationFile_Government
```

- This image represents where I have stored it in my project directly.

Note: Please store it in the project folder by directly pasting it to the project folder. Providing it in a package will give errors.

- Using getAbsolutePath() I am finding the path and then using file reader to read the file.

contactDeviceDetails class:

It stores the value of contact details, date and duration.

ConnectionManager class:

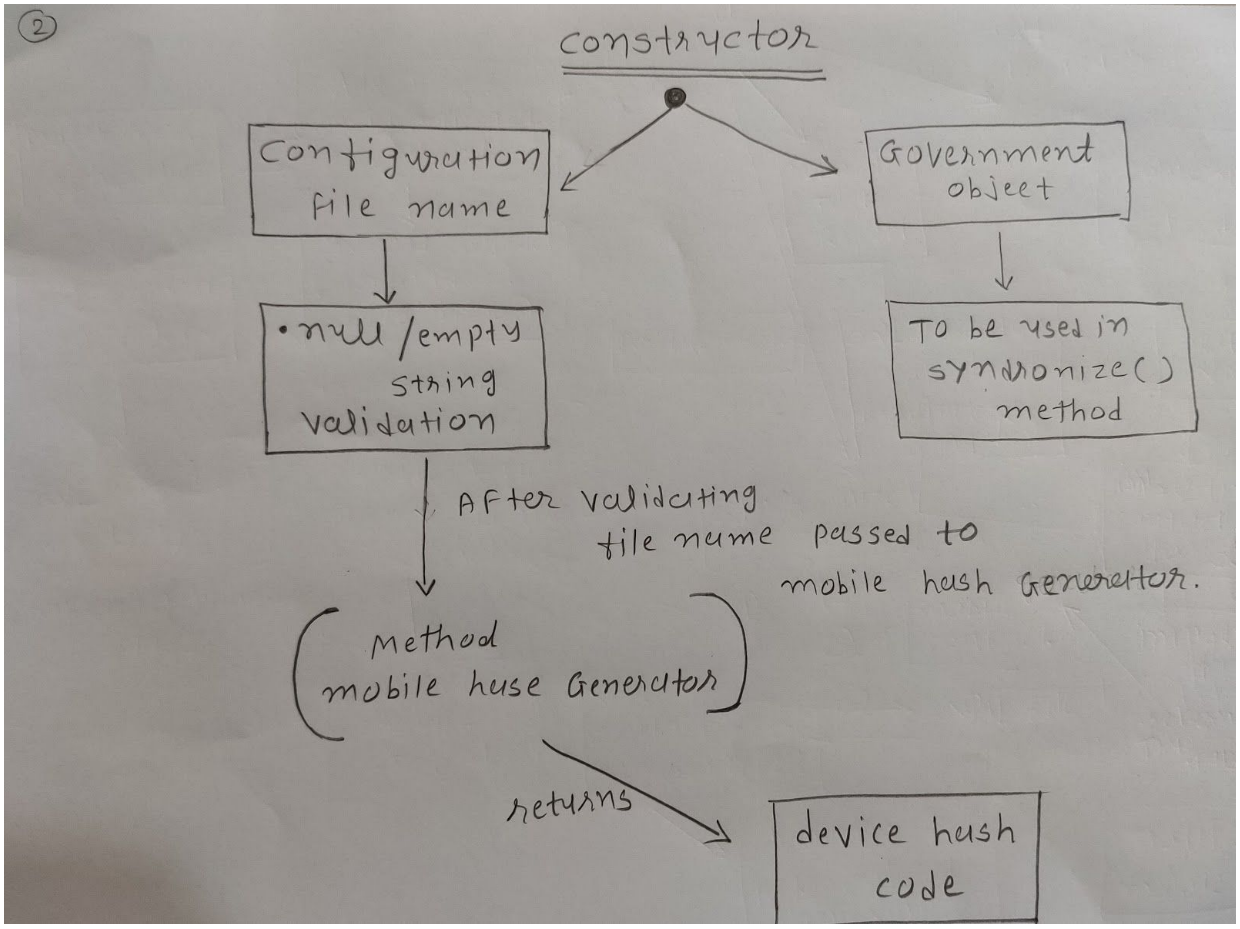
It performs SQL connection tasks surrounded with try catch blocks.

MobileDevice Class:

Constructor:

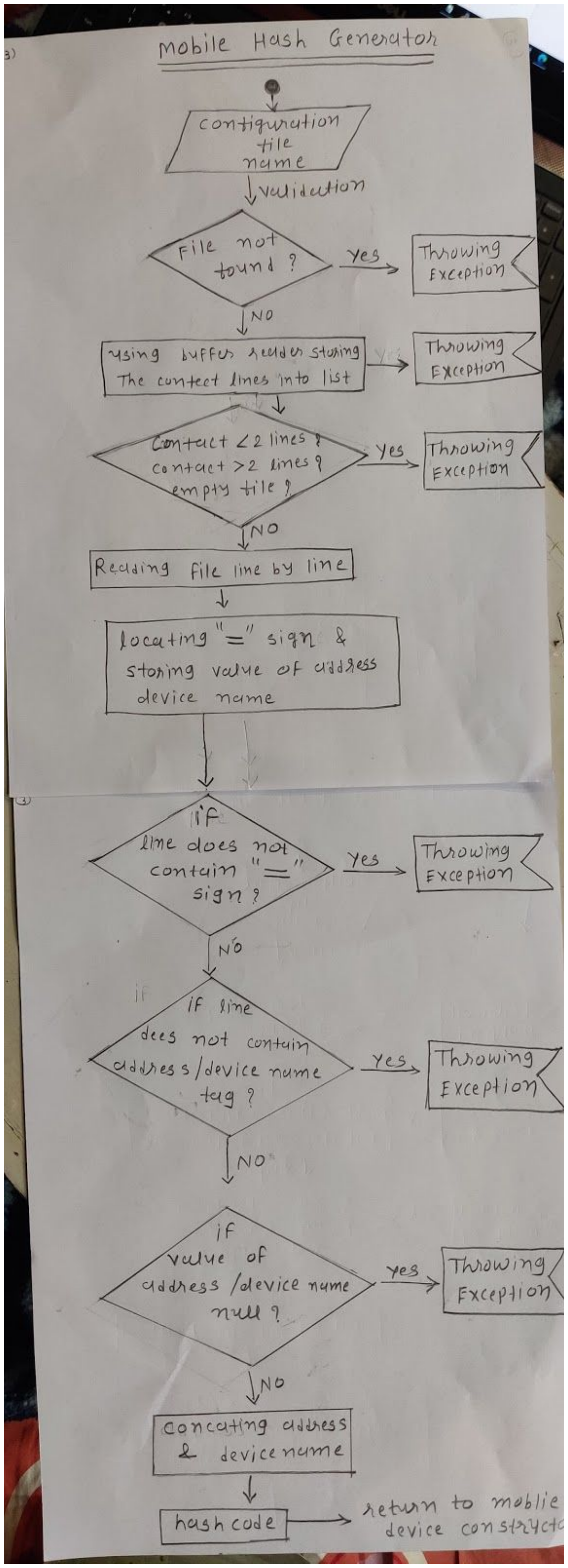
protected MobileDevice(String configurationFile, Government contactTracer) throws Exception:

- Initially the constructor checks if the configurationFile name is null or empty. If yes then printing a message stating invalid configurationFile and returning from that point.
- otherwise call mobileHashGenerator function which reads the file and generates the hashcode.
- Setting the value of the government object which is going to be used into synchronizeData() method.


2
constructor
Configuration file name
Government object
• null /empty string validation
To be used in syndronize() method
After validating file name passed to mobile hash Generator.
Method mobile huse Generator
returns
device hash code

[1] private String mobileHashGenerator(String configurationFile) throws Exception

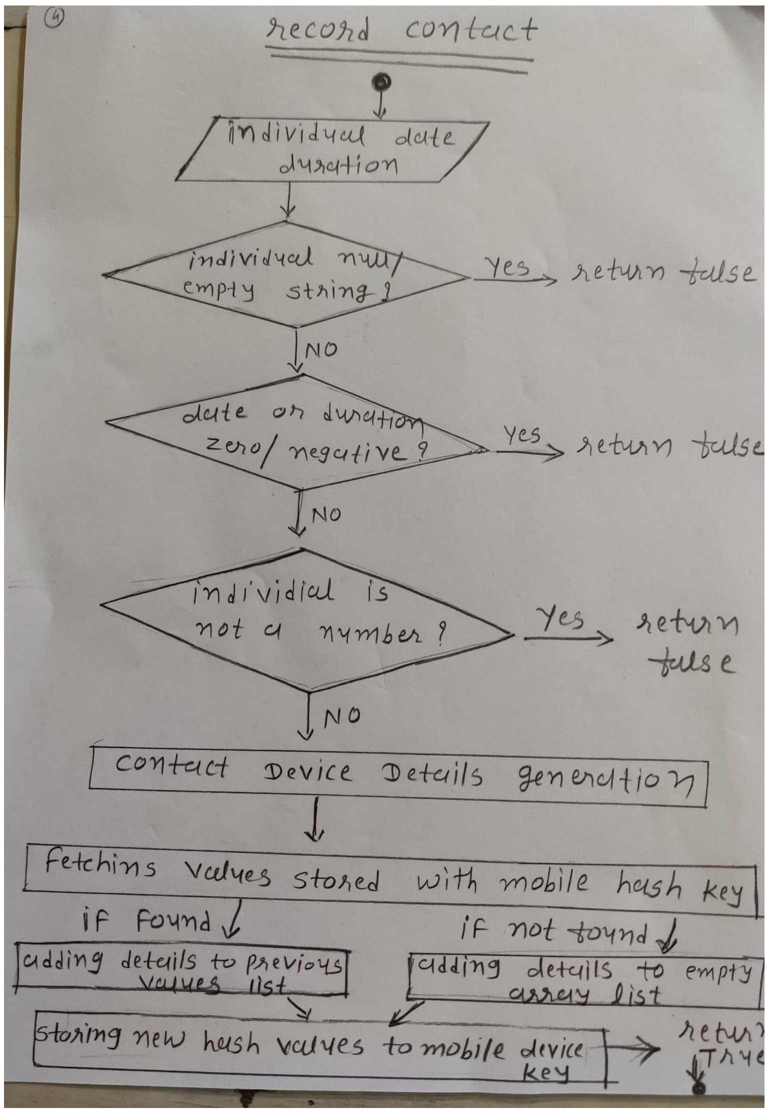
- Using filereader creating objects and reading the configuration file details.
- With the help of a buffer reader, using while loop reading every line of file and storing it to a list and then closing the buffer reader.
- After storing the lines to list I am checking whether the content is of 2 lines or not. If it is less than or greater than 2 lines or file is empty then return from that stage.
- If content length is 2 then for every string in list of content I am iterating and looking for 4 basic check points:
    - index of equals sign.
    - Checking whether a line contains address, deviceName keyword to differentiate it and storing the detail into a particular variable.
    - Getting substring from the next index of equals to the end of line.
    - If the line does not contain = or any keywords from listed above returning from that point by stating the 'please check the content'.
- After that, checking where any variable address, deviceName equals null/empty string.
- The next step is concating the address and deviceName string and generating the hashCode using a hashcode function and returning hashCode to the constructor.


Mobile Hash Generator
configuration file name
validation
File not found ?
Yes
Throwing Exception
NO
using buffer reader storing The contect lines into list
Throwing Exception
Contact <2 lines ? contact >2 lines ? empty file ?
Yes
Throwing Exception
NO
Reading file line by line
locating "=" sign & storing value of address device name
if line does not contain "=" sign ?
Yes
Throwing Exception
NO
if line dees not contain address/device name tag ?
Yes
Throwing Exception
NO
if value of address /device name null ?
Yes
Throwing Exception
NO
Concating address & device name
hash code
return to moblie device constructo

[2] protected boolean recordContact(String individual, int date, int duration)

record contact is used to enter details of contacted device, date and duration

- If an individual is null or empty string and not contains only numeric value then returning false with the message indicating the invalid input of the individual.
- If date or duration is zero or negative and duration is more then 1140 (which is maximum minutes in a day) then return false with a message indicating the invalid input of the individual.
- Generating contactDeviceDetails object using individual, date and duration.
- If there is no entry of this mobile device into the hash table then clearing the list of collection contact device details and then adding the details to the hash table with key as mobile device and value as contactDeviceDetails.
- And if there is already some value present as value into the hash table then fetching the values of key mobile devices and then adding into that list and then adding it to the hash table again.
- Returning true at the end after successfully inserting at value.

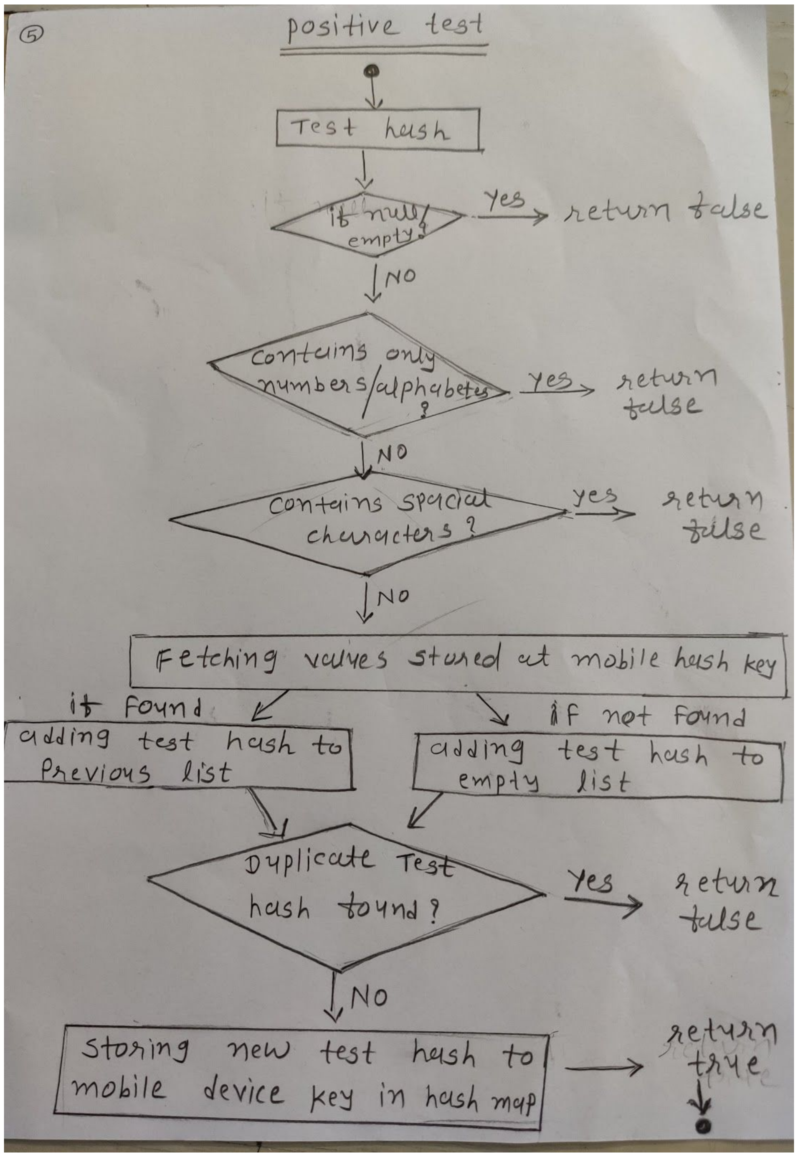


4
record contact
individual date duration
individual null/ empty string?
Yes
return false
NO
date or duration zero/ negative?
Yes
return false
NO
individial is not a number?
Yes
return false
NO
contact Device Details generation
Fetching values stored with mobile hash key
if Found
if not found
adding details to previous values list
adding details to empty array list
storing new hash values to mobile device key
return True

[3] protected boolean positiveTest(String testHash)

adding details of positive test hash value of mobile device

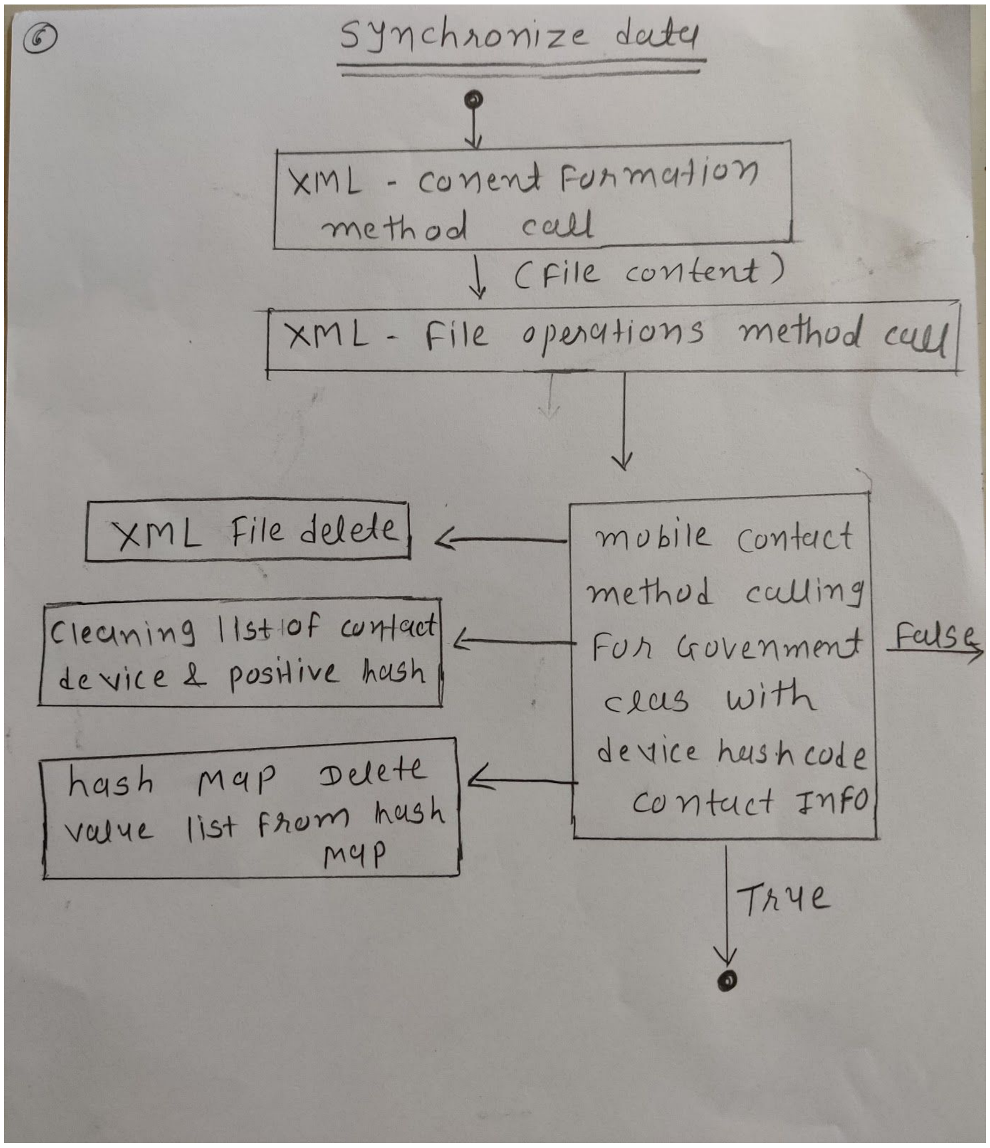
- If a testhash is null or empty string and contains only numeric/ alphabets or special characters value then returning false with the message indicating the invalid input of the individual.
- If there is no entry of this mobile device into the hash table then clearing the list of collection contact device details and then adding the details to the hash table with key as mobile device and value as testHash.
- And if there is already some value present as value into the hash table then fetching the values of key mobile devices and then adding into that list and then adding it to the hash table again.
- Returning true at the end after successfully inserting at value.


5
positive test
Test hash
if null/ empty?
yes
return false
NO
contains only numbers/alphabets ?
yes
return false
NO
contains special characters ?
yes
return false
NO
Fetching values stored at mobile hash key
if found
if not found
adding test hash to previous list
adding test hash to empty list
Duplicate Test hash found ?
Yes
return false
NO
Storing new test hash to mobile device key in hash map
return true

[4] protected boolean synchronizeData()

Calls MobileDevice class in Government and returns back the COVID-19 contact with device in last 14 days

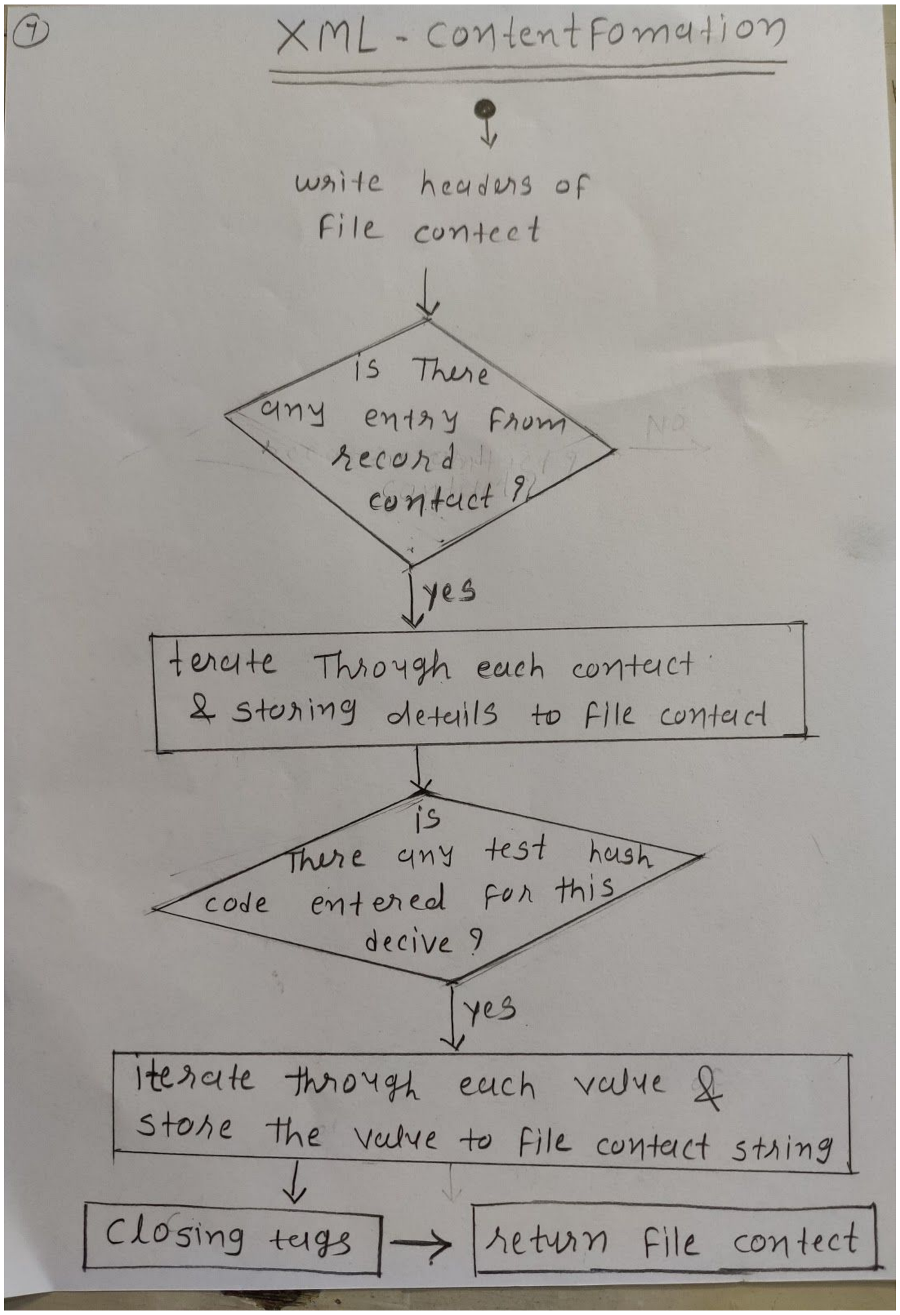
- Calls xml_conent formation method and it returns formatted string to synchronizeData.
- Passing this string to xml_filoperations method to write the string into file.
- Once successfully written, the mobile device method calls mobileContact into the government class with parameter initiator as String and xml file name.
- If mobileContact returns true meaning this mobile device was in contact with another device during the last 14 days so returning true. And if mobileContact returns false then returning false to the user.
- Operations like XML file deletion, cleaning the list of contact device and positive hash and removing entries from hashmap of contact device and positive test is being taken place.


6
synchronize data
XML - conent Formation method call
(File content)
XML - File operations method call
XML File delete
Cleaning list of contact device & positive hash
hash map Delete value list from hash map
mobile contact method calling For Govenment clas with device hash code contact Info
False
True

[5] private String xml_contentFormation()

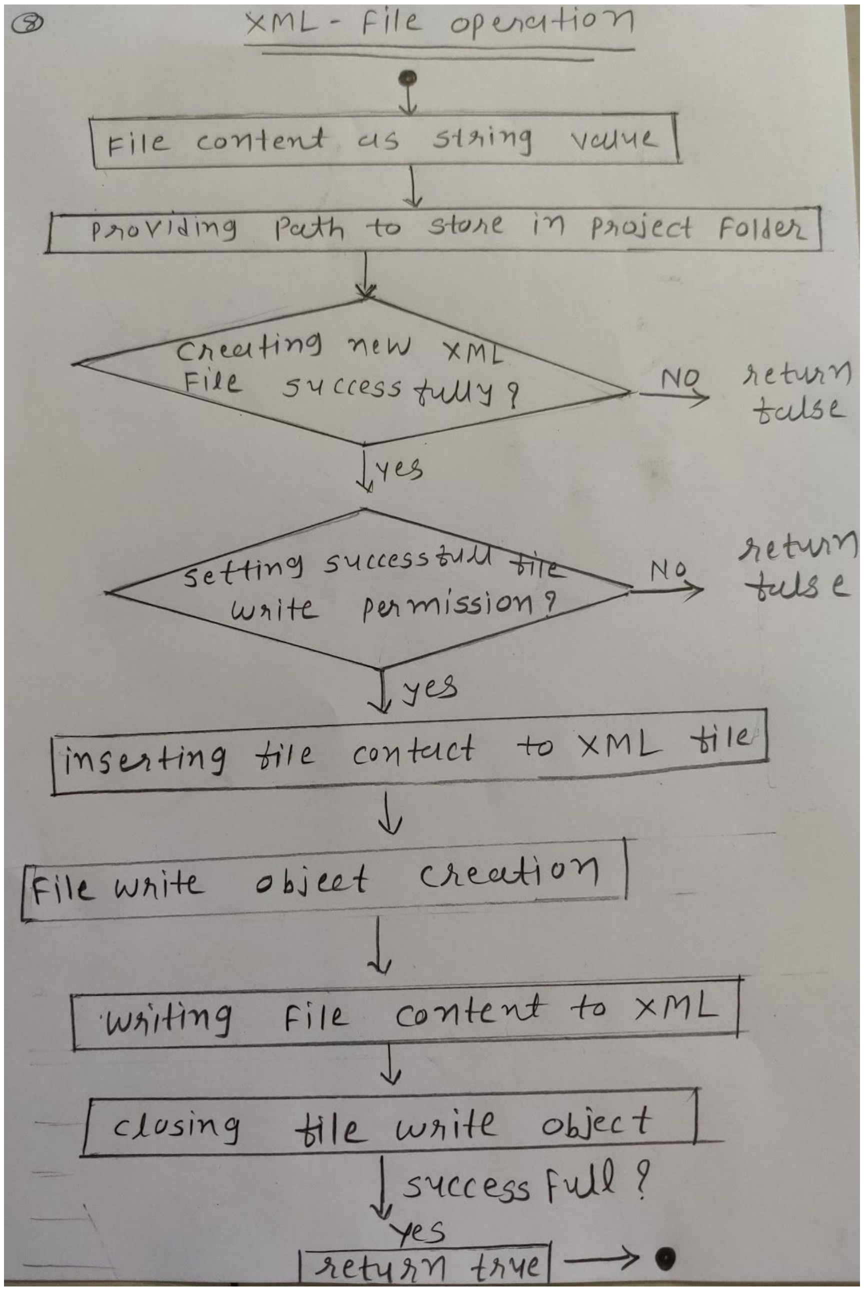
Generating String in xml file format which contains contact details and positive test details.

- Inserted header xml version, encoding value and inserted mobiledevicedetails tag.
- If there is any entry from record contact then, iterate through all the list of contactdeviceDetails and store them with XML tags.
- If there is any entry from positive test hash then, iterate through all the test hash and store them into XML tags.
- Close the mobiledevicedetails tag and return the file content to sychronizeData().


7
XML - contentFomation
write headers of File contect
is There any entry from record contact ?
NO
yes
terate Through each contact & storing details to File contact
is There any test hash code entered for this decive ?
yes
iterate through each value & store the value to File contact string
Closing tags
return File contect

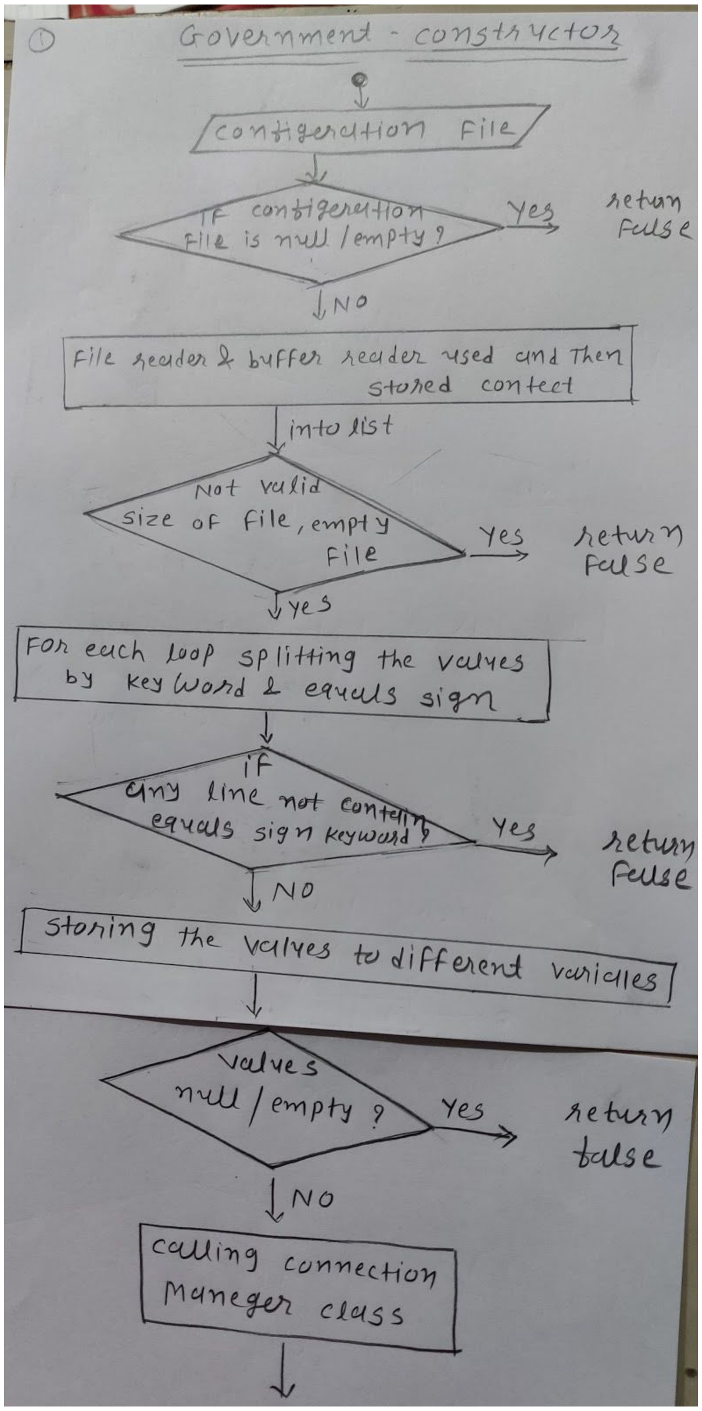
[6] private boolean xml_FileOperations(String file_content)

- Creating XML file and writing the string provided by xml_content method().
- File creation using the string file name and getting file path.
- If file creation is successful then setting the writable permission.
- Then, file writer object creation is being processed and using the write command I am inserting the string file_content into file and after this closing the file.
- Returning true to synchronizeData once successfully inserted into the file.


8
XML - File operation
File content as string value
Providing Path to store in project Folder
Creating new XML File successfully?
NO
return false
yes
setting successfull ~~file~~ write permission?
No
return false
yes
inserting file contact to XML file
File write object creation
writing File content to XML
closing file write object
successfull?
yes
return true

Government Class:

Constructor: protected Government(String configurationFile): constructor to read configuration file and calling Connection Manager class to connect with SQL.



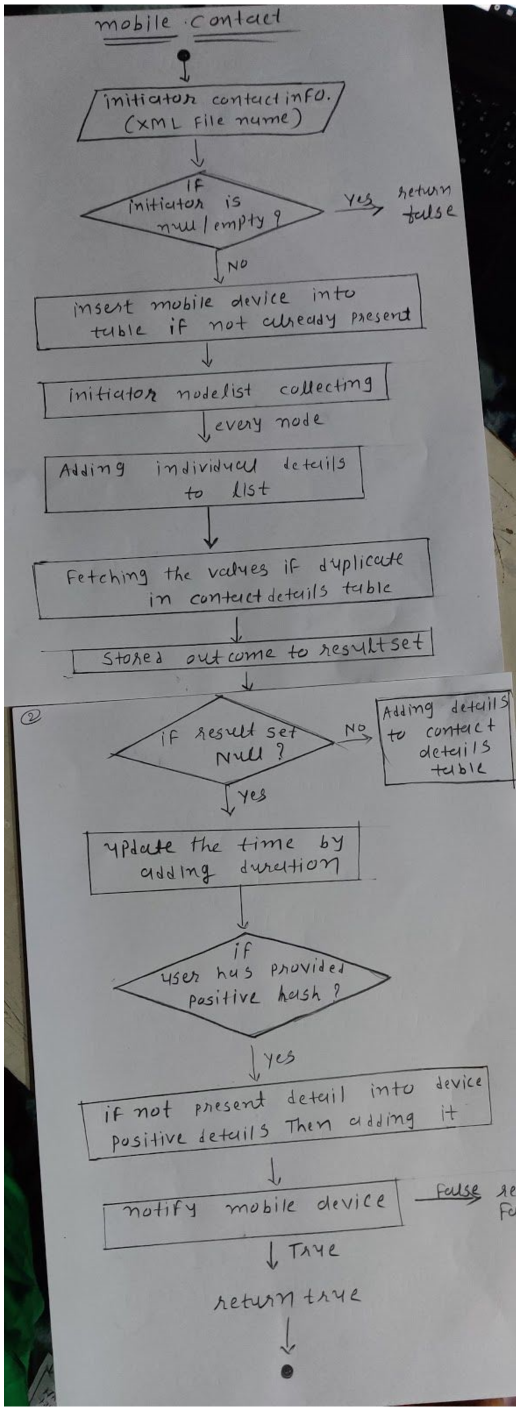
①
Government - constructor
configeration File
If configeration File is null / empty ?
Yes
return False
NO
File reader & buffer reader used and Then stored contect
into list
Not valid size of File, empty File
Yes
return False
yes
For each loop splitting the values by key Word & equals sign
if any line not contain equals sign keyword ?
Yes
return False
NO
Storing the values to different varicles
Values null / empty ?
Yes
return false
NO
calling connection Maneger class

Configuration File name: "ConfigurationFile_Government" This string(the file name) is stored in mainUI class and then passed to Government class.

To handle details in configuration File:

- Initially the method checks if the configurationFile name is null or empty. If yes then printing a message stating invalid configurationFile and returning from that point.
- Using filereader creating objects and reading the configuration file details.
- With the help of a buffer reader, using while loop reading every line of file and storing it to a list and then closing the buffer reader.
- After storing the lines to list I am checking whether the content is of 3 lines or not. If it is less than or greater than 3 lines or file is empty then returning from that stage.
- If content length is 3 then for every string in list of content I am iterating and looking for 4 basic check points:
    - index of equals sign.
    - Checking whether a line contains a database, user or password keyword to differentiate it and storing the detail into a particular variable.
    - Getting substring from the next index of equals to the end of line.
    - If the line does not contain = or any keywords from listed above returning from that point by stating the 'please check the content'.
- After that, checking where any variable database, user or password equals null/empty string.
- The next step is calling connectionManager class for SQL connection and once the connection is established generate the statement.

Methods:


mobile contact
initiator contact info. (XML File name)
if initiator is null / empty ?
Yes
return false
No
insert mobile device into table if not already present
initiator nodelist collecting
every node
Adding individual details to list
Fetching the values if duplicate in contact details table
Stored out come to result set
②
if result set Null ?
No
Adding details to contact details table
yes
update the time by adding duration
if user has provided positive hash ?
yes
if not present detail into device positive details Then adding it
notify mobile device
False
True
return true

[1] protected boolean mobileContact(String initiator, String contactInfo)

Method is implemented to insert the details into SQL database and handle notification which is being provided for the device as the indication that device is being contacted with COVID-19 positive person.

- Initially the method checks whether the initiator is a null/ empty string.
- Executing the SQL query to insert mobile device hash into mobileDeviceDetails table using insert ignore statement. Insert ignore is used to ignore duplicate entry.
- Collecting the node list by tag “initiator” and iteration for each node. Storing the details from every xml element and storing it to local variables.
- For the use in notify record contact adding this contact to list.
- As the next step, I check whether the data already present into contactDetails table and storing the values in resultSet. resultSet contains recordTime from duplicate entry in table.
- If resultset is null then insert the data directly to contactDetails table but if not then adding the current duration and record time value from database and then updating the existing entry into database.
- Meanwhile, I am keeping track of the number of queries that are successfully performed. After this checking if total success queries is equals to no of contact inserted or not. If not then printing a message as an indication that there was some detail(s) that is not valid.
- If the user has also provided a positive Test result then, storing those details of the node list and again as previously performed, iterating for each node.
- Then locally storing the element value from the xml tag and then inserting the details of positive hash into devicePositiveResult if it is not already there.
- Meanwhile, here also I am keeping track of the number of queries that are successfully performed. After this checking if total success queries is equals to no of positive hash code

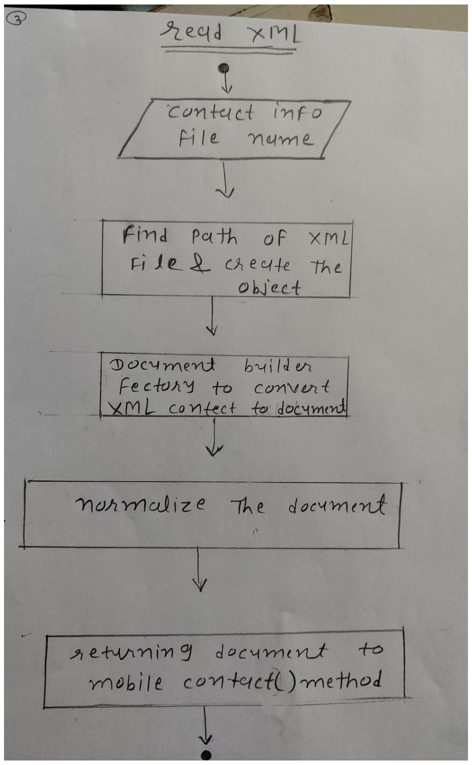
inserted or not. If not then printing a message as an indication that there was some detail(s) that is not valid.

- Note: there is a higher possibility of getting an error message here, but it is compulsory to have a positiveHash detail from the agency first and then the user enters the positiveHash.
- Now, to notify mobiledevice the method is called here as the function's return value I provide the return statement to the user indicating the contact with COVID-positive user.
- Returning true if being contacted with a positive user within the last 14 days and false otherwise.
- Note: here I will also consider providing the notification if notification is not being provided previously by the same user.

[2] private Document readXML (String contactInfo)

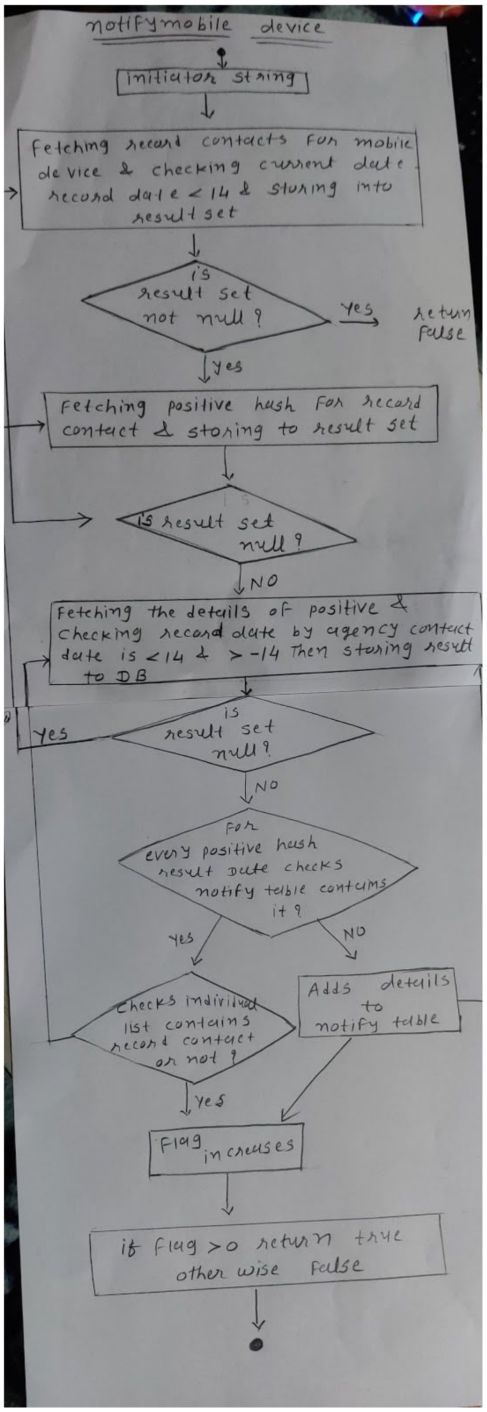
Method is implemented to read details from xml file to document.

- Initially find the path of the xml file and then create the object.
- Document builder factory is used to store details into documents.
- Object creation of document builder and storing the file content into document format using parse function.
- As a next step normalize the object and then return it to the mobileContact() method.


3
read XML
contact info file name
Find path of XML File & create the object
Document builder Factory to convert XML content to document
normalize the document
returning document to mobile contact() method

[3] private boolean notify_mobiledevice(String initiator) throws SQLException

- As a first step, fetching the contacts, record date and for a particular device and with the consideration of difference between today's date and recordDate is not more than 14 days. I have this filter of 14 days here because if today's date is the 15th day of being contacted with than there is no need to notify the device.


notifymobile device
initiator string
Fetching record contacts for mobile device & checking current date - record date < 14 & storing into result set
is result set not null ?
Yes
return false
yes
Fetching positive hash for record contact & storing to result set
is result set null ?
NO
Fetching the details of positive & checking record date by agency contact date is < 14 & > -14 Then storing result to DB
is result set null ?
Yes
NO
For every positive hash result Date checks notify table contains it ?
Yes
NO
Checks individual list contains record contact or not ?
Adds details to notify table
Yes
Flag increases
if flag > 0 return true other wise false

- Previous query is stored in a resultset and if the resultset is not empty then for each record contact firstly store the record contact and record date into a local variable.

- for this recordContact fetching the positive hash values from Table devicePositiveResult and after storing it to result set and checking it is not empty I am considering the next step for every device positive result.

- matching test hashcode with the agency's result table and finding the result date and positiveHash value , I am checking the result date provided by the agency and the record Date's difference is between 14 to -14 (14 and -14 excluding). Note: I am checking for 28 days duration because it might be the case that the person is contacted before having corona or within the period of corna test or first day of corona and results positive.

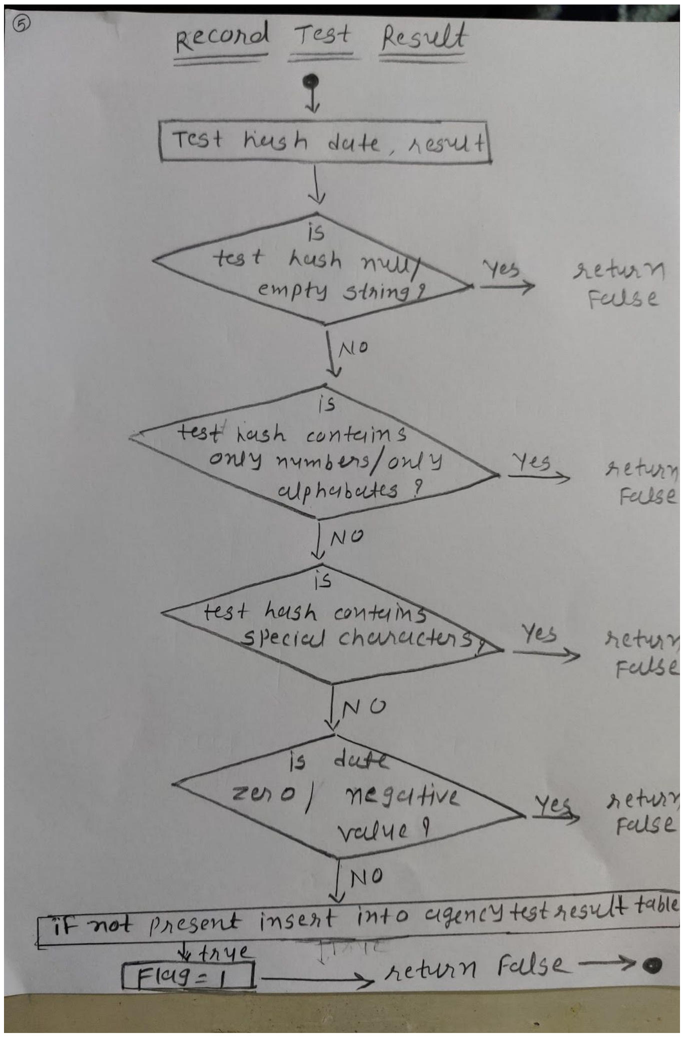
- As a next step, for every positive hash I am checking if the initiator, positive hash, record contract has already been present in notifydeviceDetails. I am using the notify device details table to keep track of already notified devices for this mobile device and not being contacted again.

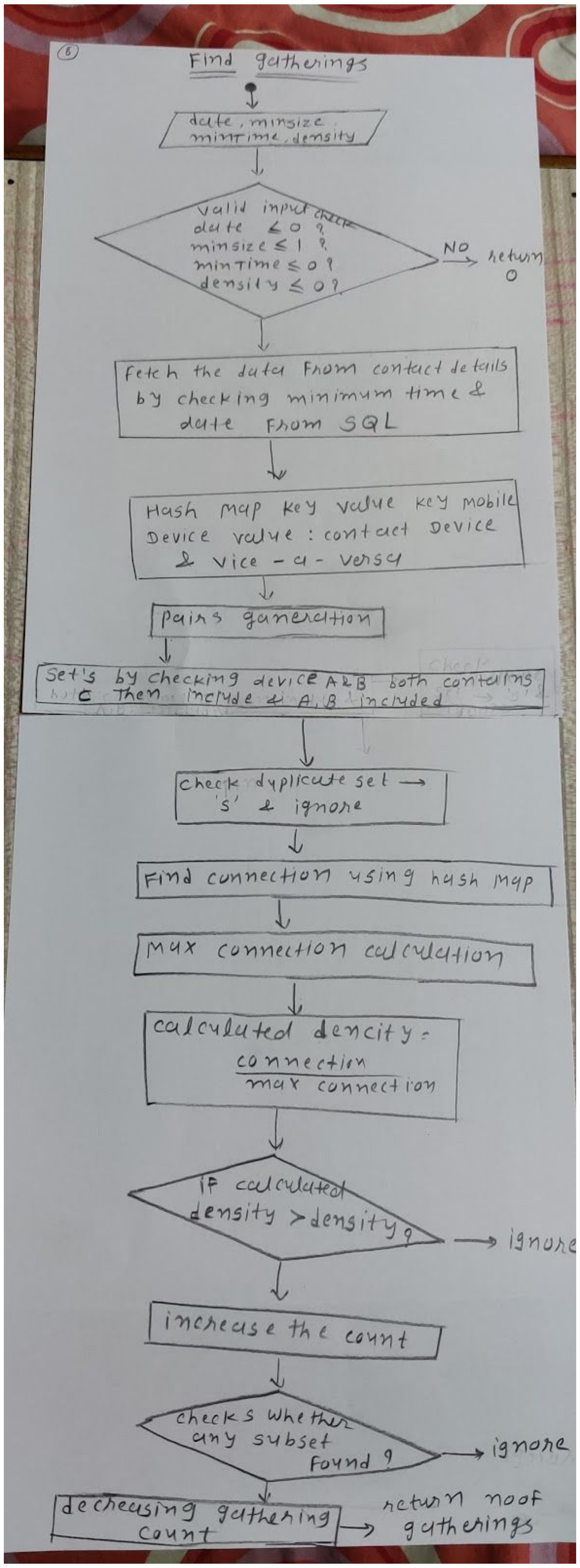
- If not present into table then adding it to notify device details and increasing the flag.
- If present then checking it with the current contact details to make sure that I am not going to provide a notification to the user till 14 days if I have already provided one. But If the current device is contacted with other devices then I am going to provide notification.
- If flag is greater than zero then return true or return false.

[4] protected boolean recordTestResult(String testHash, int date, boolean result)

Inserting the testResult provided by the agency to SQL database.

- If testhash is null or empty string then returning false.
- If testhash contains only numbers or alphabets then returning false.
- If testHash contains special characters then returning false.
- If the date is zero or negative, then return false.
- If testHash is not present in the data table agency test result then insert it to the agencyTestResults table and update the success flag as one.
- If the success flag is zero, then provide a message that unsuccessful insertion and return false otherwise return true.


5
Record Test Result
Test hash date, result
is test hash null/ empty string?
Yes
return False
NO
is test hash contains only numbers/only alphabates ?
Yes
return False
NO
is test hash contains special characters?
Yes
return False
NO
is date zero/ negative value?
Yes
return False
NO
if not present insert into agency test result table
true
Flag = 1
return False


6
Find gatherings
date, minsize, mintime, density
valid input check
date ≤ 0 ?
minsize ≤ 1 ?
mintime ≤ 0 ?
density ≤ 0 ?
NO
return 0
Fetch the data from contact details by checking minimum time & date from SQL
Hash map key value key mobile Device value : contact Device & vice - a - versa
pairs generation
Set's by checking device A&B both contains C then include & A, B included
check duplicate set 's' & ignore
Find connection using hash map
Max connection calculation
calculated dencity = connection / max connection
if calculated density > density ?
ignore
increase the count
checks whether any subset Found ?
ignore
decreasing gathering count
return noof gatherings

[5] protected int findGatherings(int date, int minSize, int minTime, float density)

To Get information on the number of gatherings on a particular date by considering minimum gathering of no of people, minimum time and minimum density.

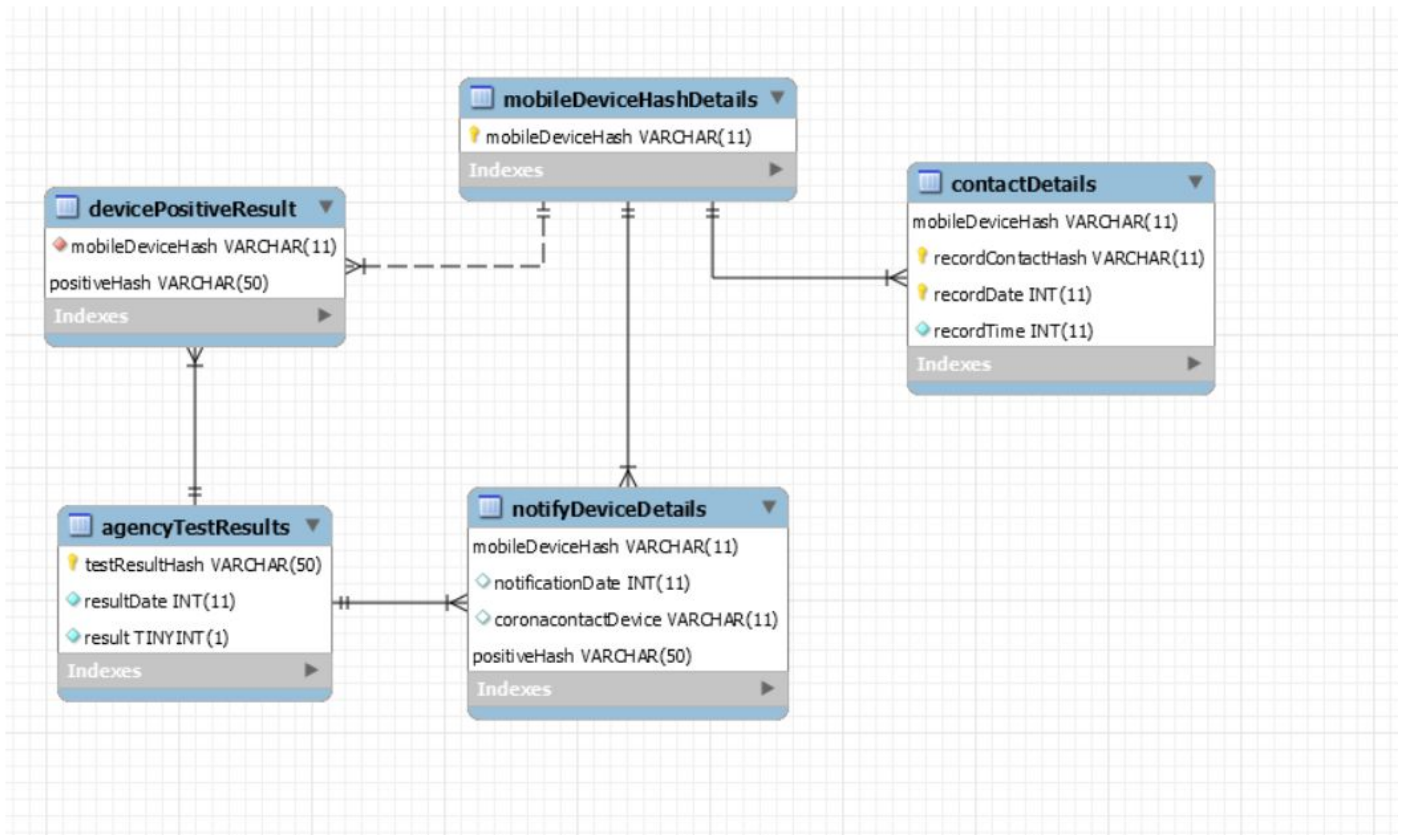
- Fetch the data from contactDetails by checking minimum time and date
- Hash table key-value insertion for contacted devices.
- Generate pairs of keyset from hashMap for example, having 5 keys we will end up with 10 pairs using arrays and Set.
- Find Set S for A,B and consider all the devices present in A,B both including A,B.
- Filter with minSize of group need and find connections in Set S.
- Maximum Possible Connection and find density, increase the gathering count if density is greater then the provided density.
- Removing the subset S from superSet to avoid considering the same gathering with less number of people again and decreasing the gathering count.
- Returning the gathering count.

Database

Below schema has 5 tables:

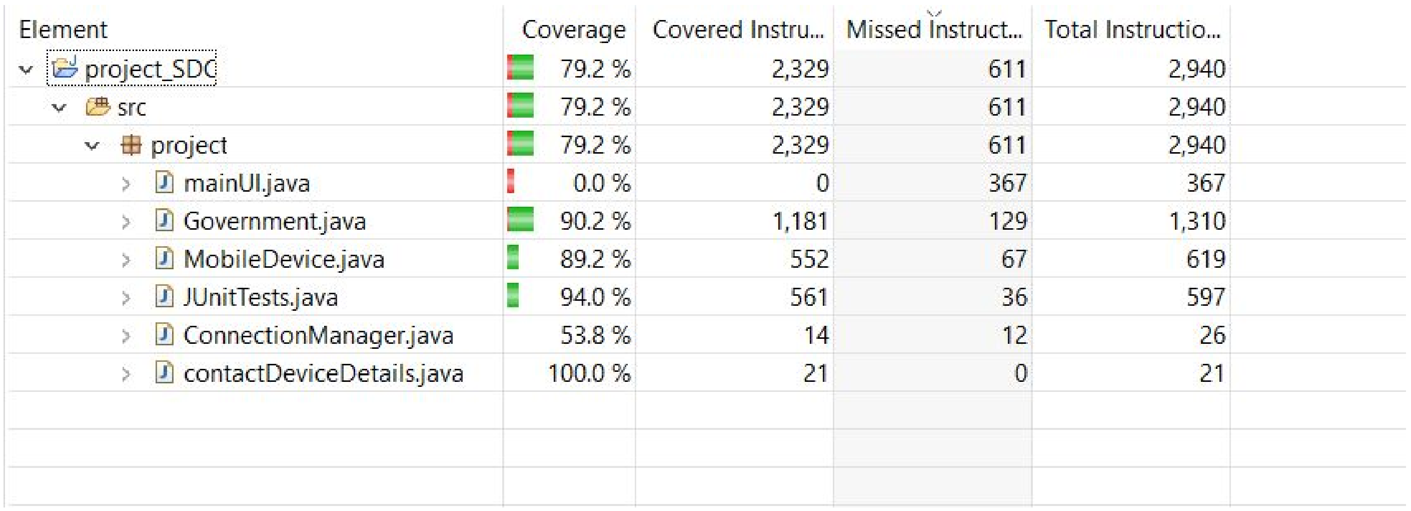
MobileDeviceHashDetails
DevicePositiveResult
ContactDetails
notifyDeviceDetails
agencyTestResults

The design is in 3NF format.



Test Cases:

I have generated the JUnit test case provided in JUnitTests.java. This method has provided me more ideas on what went wrong and how I can check it differently. Using JUnit test cases I was able to re-run the test cases if I changed anything in the program. Below is the coverage of codes when I performed JUnit test cases:



Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
project_SDC	79.2 %	2,329	611	2,940
src	79.2 %	2,329	611	2,940
project	79.2 %	2,329	611	2,940
mainUI.java	0.0 %	0	367	367
Government.java	90.2 %	1,181	129	1,310
MobileDevice.java	89.2 %	552	67	619
JUnitTests.java	94.0 %	561	36	597
ConnectionManager.java	53.8 %	14	12	26
contactDeviceDetails.java	100.0 %	21	0	21

Input Validation Tests

Class MobileDevice:

(1) MobileDevice(String configurationFile, Government contactTracer)

(a) Null value passed as filename. - false
(b) An empty value passed as filename. - false
(c) Valid value passed as filename. - true

(2) recordContact(String individual, int date, int duration)

(a) Null value passed as individual. - false
(b) Empty value passed as individual. - false
(c) Valid value passed as individual. - true
(d) Individual contains only numeric values - true
(e) date is equal to zero. - false (considering 1st january,2020 as date 1)
(f) date as a negative value. - false
(g) Valid positive integer as date. - true
(h) duration equal to zero. - false
(i) duration is negative value. - false
(j) Valid positive integer as duration. - true

(3) positiveTest(String testHash)

(a) Null value as testHash. - false
(b) Empty string as testHash. - false
(c) testHash contains special characters. - false
(d) testHash contains only alphabets. - false
(e) testHash contains only numeric values. - false
(f) testHash is an alphanumeric string. - true

Class Government:

(1) mobileContact(String initiator, String contactInfo)

(a) Valid value passed as initiator. - true

(2) recordTestResult(String testHash, int date, boolean result)

(a) testHash contains special characters. - false
(b) testHash contains only alphabets. - false
(c) testHash contains only numeric values. - false
(d) testHash valid is alphanumeric values. - true
(e) Null value passed as testHash. - false
(f) Empty value passed as testHash. - false
(g) date is equal to zero. - false
(h) date as a negative value. - false
(i) Valid date passed as positive integer value. - true
(j) False or True passed as result. - true

(3) int findGatherings(int date, int minSize, int minTime, float density)

(a) date is equal to zero. - false
(b) date as a negative value. - false
(c) Valid date passed as positive integer value. - true
(d) minSize is equal to two. - false
(e) minSize as a negative value. - false
(f) minSize as positive integer. - true
(g) minTime is equal to zero. - false
(h) minTime as a negative value. - false
(i) minTime as a positive integer. - true
(j) density is less than or equal to zero. - false
(k) density is greater than zero. - true

BoundaryTests

Class MobileDevice:

1) MobileDevice(String configurationFile, Government contactTracer)

    a) Configuration file does not exist.
    b) Empty file.

2) recordContact(String individual, int date, int duration)

    a) date passed as 1.
    b) date passed as greater than 1.
    c) Passed the duration as value 1 minute.
    d) Passed the duration as value more than 1 minute.
    e) First time calling recordContact for a particular MobileDevice.

3) positiveTest(String testHash)

    a) String testHash consists of a combination of 1 character and 1 number.
    b) Long value of teshHash consists of a combination of multiple characters and numbers .

4) synchronizeData( )

    a) Data contains one mobile device with one contact details.
    b) Data contains multiple mobile devices with multiple contact details.

Class Government:

1) Government(String configurationFile)

    a) ConfigurationFile has only the details of one database.
    b) ConfigurationFile has details of more than one database.
    c) File does not exist.
    d) Empty file.
    e) Valid file with details of domain name of database, username and password.

2) boolean mobileContact(String initiator, String contactInfo)

    a) Data contains one mobile device with one contact details.

b) Data contains multiple mobile devices with multiple contact details.
c) Call mobileContact when the initiator has only one contact.
d) Call mobileContact when the initiator has more than one contact.

3) recordTestResult(String testHash, int date, boolean result)

a) String teshHash consists of a combination of 1 character and 1 number.
b) Long value of teshHash consists of a combination of multiple characters and numbers.
c) date passed as 1.
d) date passed as greater than 1.

4) int findGatherings(int date, int minSize, int minTime, float density)

a) date passed as 1.
b) date passed as greater than 1.
c) Passed minTime as 1 minutes.
d) Passed minTime as more than 1 minutes.
e) Passed minSize as 2 members of the gathering.
f) Passed minSize as more than 2 members of the gathering.
g) Inputted density input is zero.
h) Inputted density is greater than zero.

Control Flow Tests

Class MobileDevice:

1) For constructor MobileDevice(String configurationFile, Government contactTracer)

a) Details of the same mobile device inserted multiple times.
b) Inserting new mobiledevice details from configurationFile.
c) Configuration file consists of details of more than 1 mobile device.
d) Configuration file contains either address or devicename and not both the details.
e) Value of address or devicename passed as null or empty string.
f) Valid configuration file consists of 2 lines, one for address and one for deviceName.

2) recordContact(String individual, int date, int duration)

a) Duplicate entry which is already present with the same details from recordContact.
b) Call when there is an entry with different duration and date but same individual.
c) Call when there is an entry with the same duration and date but different individual.
d) Call when there are multiple contacts to one mobile device with different duration and time.

3) positiveTest(String testHash)

a) Call positiveTest when there is already a testHash value present for a mobile device. (multiple test of COVID-19 for single user)
b) Duplicate positiveTest for the same user.

4) synchronizeData( )

a) Call when contact device or positive hash does not exist but there is only detail of mobile device.
b) Call when we have details of mobiledevice of more than one mobile device to pass it to the government.

Class Government:

1) Government(String configurationFile)

a) Configuration file incorrect username / password /database.
b) Configuration file contains multiple entries of databases, usernames and passwords.
c) Values of either of databases, usernames or passwords null or empty string.
d) One of the values of either database, username or password is missing.
e) Configuration file contains the database, user, password in different lines.

2) mobileContact(String initiator, String contactInfo)

a) Call when a contact device does not exist but only a mobile device exists. (mobile device without contacts)
b) Call when inserting the first entry of a mobile device in the database.
c) MobileDevice has been in contact with COVID positive test case or negative test case.

3) recordTestResult(String testHash, int date, boolean result)

a) TestHash duplicate but date and result different.
b) Duplicate testHash, date and result.
c) Handle entry of when the result is positive or negative.
d) Inserting the first test result in the database.

4) findGatherings(int date, int minSize, int minTime, float density)

a) Details of a particular date do not exist. (no gatherings)
b) One of the conditions or all the conditions followed to find gathering.
c) When only one no of gathering .
d) When more than one gathering for the same day.

Data Flow Tests

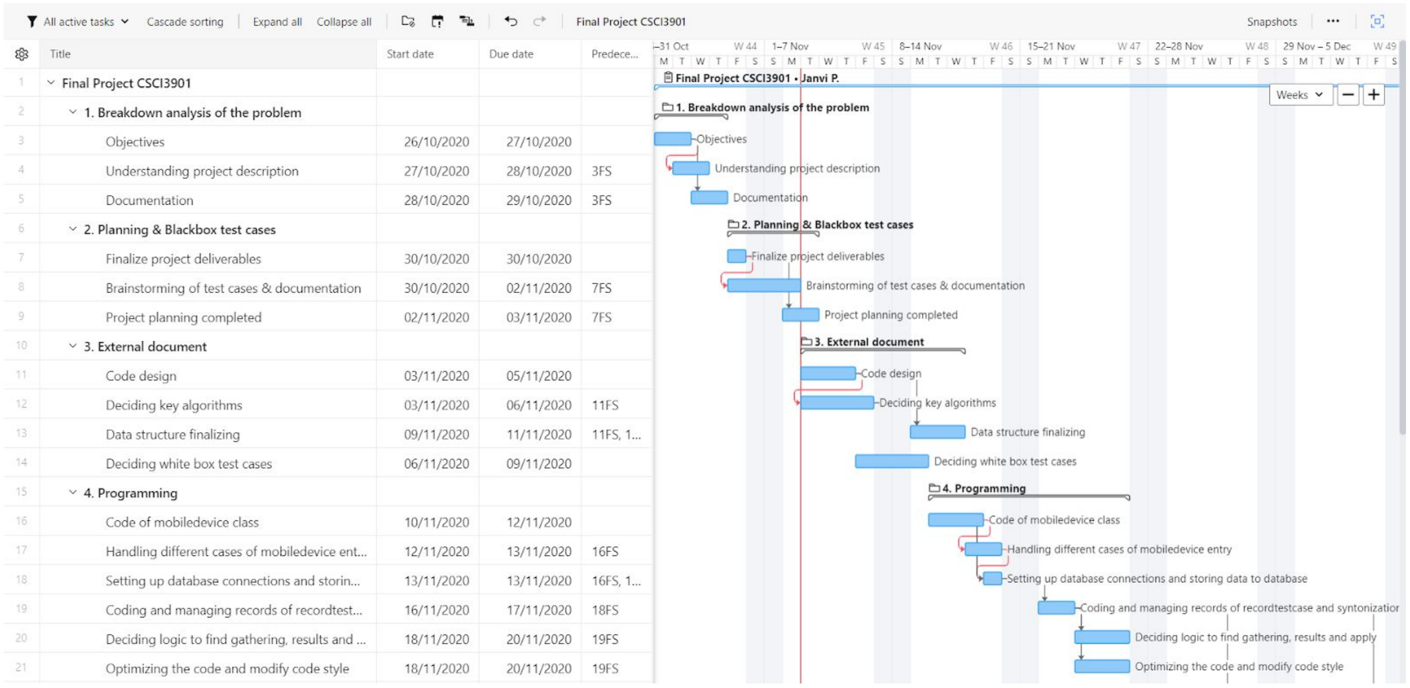
Class MobileDevice:

1. Call positiveTest without calling recordContact.
2. Call recordContact before calling positiveTest.
3. Call Synchronize without entering the value of recordContacts.
4. Call Synchronize without setting hashValue of mobile device or creating new government object in constructor mobileDevice.

Class Government :

1. Call find gathering without ever calling mobileContact.
2. Call recordTestResults without ever calling mobileContact.

<u>Plan of feature development:</u>



	Title	Start date	Due date	Predece...
1	Final Project CSCI3901			
2	1. Breakdown analysis of the problem			
3	Objectives	26/10/2020	27/10/2020	
4	Understanding project description	27/10/2020	28/10/2020	3FS
5	Documentation	28/10/2020	29/10/2020	3FS
6	2. Planning & Blackbox test cases			
7	Finalize project deliverables	30/10/2020	30/10/2020	
8	Brainstorming of test cases & documentation	30/10/2020	02/11/2020	7FS
9	Project planning completed	02/11/2020	03/11/2020	7FS
10	3. External document			
11	Code design	03/11/2020	05/11/2020	
12	Deciding key algorithms	03/11/2020	06/11/2020	11FS
13	Data structure finalizing	09/11/2020	11/11/2020	11FS, 1...
14	Deciding white box test cases	06/11/2020	09/11/2020	
15	4. Programming			
16	Code of mobiledevice class	10/11/2020	12/11/2020	
17	Handling different cases of mobiledevice ent...	12/11/2020	13/11/2020	16FS
18	Setting up database connections and storin...	13/11/2020	13/11/2020	16FS, 1...
19	Coding and managing records of recordtest...	16/11/2020	17/11/2020	18FS
20	Deciding logic to find gathering, results and ...	18/11/2020	20/11/2020	19FS
21	Optimizing the code and modify code style	18/11/2020	20/11/2020	19FS

Final Project CSCI3901 • Janvi P.
1. Breakdown analysis of the problem
Objectives
Understanding project description
Documentation
2. Planning & Blackbox test cases
Finalize project deliverables
Brainstorming of test cases & documentation
Project planning completed
3. External document
Code design
Deciding key algorithms
Data structure finalizing
Deciding white box test cases
4. Programming
Code of mobiledevice class
Handling different cases of mobiledevice entry
Setting up database connections and storing data to database
Coding and managing records of recordtestcase and syntonizatior
Deciding logic to find gathering, results and apply
Optimizing the code and modify code style

Assumptions

- If a user will only provide a positive hash test result as an entry for the database.
- Users will only insert a date which is less than or equal to the current date.
- System date is up to date while running the program.

Limitations

- I have used varchar(50) to store positive testhash results which might be exceeded in some point of time.
- I am not deleting any entry from the database which might cause an increase in time complexing and unnecessary overload to the system.
- Storing every notification details to notifyDeviceDetails table which might cause problems in the long run regarding time complexity and system maintenance.
- I might need to change the data structure if I am considering hashing.

References

---

[1] https://mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/

[2] I have referred professor's mainUI function to create a likewise menu.