Final Project Report

➢ **Covid-19 Contact Tracer:**

To design this project, I made two different classes “MobileDevice.java” and “Government.java” as mentioned in the requirement. For testing all the methods of both classes and its calls I made one main driver class file named “Contact_Tracer.java”.

- **Database:**

To transfer data of all the mobile devices from government class to database I made four different tables.

1. MobileDevice Table: This table consist of one column named HashValue which includes hash value of mobile device’s address and device name.
2. ContactInfo Table: This table includes four columns. mdHash includes HashValue, cdHash includes contact device’s alphanumeric string(i.e of individual), date and duration of meet.
3. TestResult Table: This table includes testHash of mobile devices which is unique to an individual. It also stores result whether positive or negative as well as date on which collection has been taken.
4. Device_Result Table: This table is a connection of two tables(TestResult and MobileDevice). It provides result of an individual whether it is positive or negative and transfers information to database along with date on which it was reported.

- **Flow of the project design:**
  1. The project flow moves from MobileDevice class to Government class to Database of Government.
  2. In MobileDevice class:
     - MobileDevice(String configFile, Government contactTracer):

       This constructor of class first reads the configFile. Fetches values of the key (Address and DeviceName) from the file, combines them as one and converts it into hash value. As a file, I used “.properties” extension instead of “.txt” for a change. To fetch the values from file I made object of properties class. Government contactTracer object if there as a parameter so to call methods from each device.

     - recordContact(String individual, int date, int duration):

       This method will record all the details of an individual who met you on date and for how much duration. It then stores all the information into one list as to combine them.

- positiveTest(String testHash):

  This method will store an alphanumeric testHash which is provided by testing agency for an individual as a unique id. It also stores this value in the same list where recordContact method details are stored. This list is then again converted into string as one so that it can be passed as a single parameter in MobileContact method of government class.

- synchronizeData():

  This method will synchronize all MobileDevice data to Government class in MobileContact method. It will work periodically by invoking MobileContact method to transfer data to it using contactTracer object.

3. In Government class:

- Government(String configFile):

  This class constructor will read configFile which contains information to connect to the database. It will read username, password and database name. Here also I used properties class object and ".properties" file. Using this details it connects to database through JDBC.

- MobileContact(String initiator, String contactInfo):

  This method gets all the data of different mobile devices from MobileDevice class when synchronizeData invokes this method.
  It will then pass on to the database to store on server. It runs different insert queries to insert values into different tables. I tried using prepareStatement instead of Statement for executing queries. This method also fetches data of such individuals who tested positive and notifies to synchronizeData method.

- recordTestResult(String testHash, int date, boolean result):

  This method will record results of individuals based on testHash and notifies date on which they collected details. It checks whether test is positive or negative and returns true or false respectively. It will then transfer all the information to database by inserting into tables.

- findGatherings(int date, int minSize, int minTime, float density):

  This method firstly fetches data(i.e mobile device hash(mdHash) and contact hash(cdHash)) from the database based on reported date. Then it make pairs of this cdHash and mdHash into set. After that it performs intersection on this different pair of sets to find which all individual's details that who met whom. This way it returns a set of individual's based on intersection that can be considered as number of gatherings which in turn is useful to calculate whether gathering is large or not by comparing it with minimum size provided by government. After that based on density provided it is checked that if the gathering was large then it should be reported and if not then it should not be considered. This calculations are done with the provided equations.

3. Call from main class(Contact_Tracer)/Test Code:

   - This class is driver class. I made this class for the purpose of testing the methods and constructor of MobileDevice class and Government class. I tested them by calling each method and constructor in main class. They all are working fine and moving in the flow. As the main class calls synchronizeData method it calls MobileContact method from Government class and synchronizes all the data which in turn transfers all data to database by inserting in tables. And when recordTestResult method is called it directly transfers data to database by inserting in tables.
   - Constructors in main class provides file path which includes an individual's device information and the other file from Government class includes information for making connection to database. Both are called by using object of the class.
   - All insertion values are also passed from this main class to insert into tables in database sequentially. And it also checks large gatherings by calling findgatherings method that whether the gathering is worth reporting or not.

- **Conclusion:**
  1. Basically, what I am doing in this project is collecting information from different mobile devices and transferring them into Government class time to time.
  2. Firstly, the information is stored locally on the user's mobile device. All this information is then transferred to Government class periodically. Government class sends this information and the other data of class itself to database for purpose of storing and notifying to any individual if they had

been in contact with someone who tested positive in last 14 days along with days and duration of meet.

3. It also facilitates to find number of large gatherings that occurs frequently and the number of people who meets frequently. It covered all the concepts that were covered during the course.
4. For designing the solution of this project I haven't used so advanced approach, I took it in a simple way and it works smoothly.

- **Test Cases:**
  - **Class MobileDevice():**

    - **Input Validations:**
    - MobileDevice(String configFile, Government contactTracer):
      1. configFile is null or empty.
      2. Writes any random string other than file path.
      3. configFile containing Address should be in its format like (192.10.0.0) and DeviceName can be any alphanumeric string.

    - recordContact(String individual, int date, int duration):
      1. Individual is empty or null or containing spaces between string.
      2. Date and Duration are empty.
      3. Date and Duration is less than 0.

    - positiveTest(String testHash):
      1. testHash is empty or null.
      2. Value of testHash should be less than or equal to as per specified size in schema.

  - **Class Government():**

    - **Input Validations:**
    - Government(String configFile):
      1. configFile is null or empty.
      2. Writes any random string other than file path.
      3. Database, username and password contained by configFile should not be incorrect or mismatched. It should be correct.

    - MobileContact(String initiator, String contactInfo):
      1. Both initiator and contactInfo are empty or null.

- recordTestResult(String testHash, int Days, boolean result):
  1. testHash is empty of null.
  2. Value of testHash should be less than or equal to as per specified size in schema.
  3. Days is empty or less than 0.
  4. Values other than boolean types.

- findGatherings(int date, int minSize, int minTime, float density):
  1. date is empty or less than 0.
  2. minSize is less than 0.
  3. minTime is less than 0.
  4. density other than float type.
  5. density is less than 0.

❖ **Class MobileDevice() and class Government():**

Boundary Cases:

According to my vision I believe that there are no boundary cases in both the class.

❖ **Control Flow:**

The control of the program flow moves from MobileDevice class to the Government class by transferring data to database on server from Government class. In detail flow of the program is already discussed in the design portion of the document.

❖ **Data Flow:**

1. Any method is called before the object of the class constructor is initialized.
2. synchronizeData() method is called before calling positiveTest() method.
3. synchronizeData() method is called before calling recordContact() method.
4. recordTestResult() method is called before calling synchronizeData() method.
5. recordTestResult() method is called before calling positiveTest() method.
6. findGatherings() method is called before calling recordContact() method.
7. findGatherings() method is called before calling positiveTest() method.

8. findGatherings() method is called before calling synchronizeData() method.
9. findGatherings() method is called before calling recordTestResult() method.
10. synchronizeData() method is called at last of all the methods.

- **Testing Methodology Used:**
  1. In my project I used manual testing methodology for testing my program's flow.
  2. All the methods have been tested with data flow and control flow in both the classes because it was needed to check its flow.
  3. My focus for testing was on the flow of data because this project completely revolves around the flow of data and its control.
  4. I came across several situations where my code was failing and throwing exceptions. And there I decided to handle such errors and exceptions by using test cases so that my code does not stop in between and can run smoothly.