Test Scenario Group	Test Case ID	Test case Description	Prerequisities	Test steps	Expected Result	Actual Result
Creation of Government using Government(String configFile)	G1	Creation of government object using correct configFile path and correct inputs	Correct file path(with just file name at the end) and valid dbname, user and password in key=value format dbname=jdbc:mysql://db.cs.dal.ca:3306/dhruvp user=dhruvp password=B00868931	1. Give correct path for govt configFile and valid input in configFile 2. Initialise government object with passing path with just the file path and name "new Government(String path) "	Government object should be created without any errors.	Same as expected
	G2	Passing invalid path for the configFile	Correct file path(with just file name at the end) and valid dbname, user and password in key=value format dbname=jdbc:mysql://db.cs.dal.ca:3306/dhruvp user=dhruvp password=B00868932	1. Provide invalid path for govt configFile. 2. Initialise government object using this path as "new Government(invalidPath)"	It should throw FileNotFoundException.	Same as expected
	G3	Passing invalid data inside configFile. For example: wrong dbname,user or password	Correct file path(with just file name at the end) and valid dbname, user and password in key=value format dbname=jdbc:mysql://db.cs.dal.ca:3306/dhruvp user=dhruvp password=B00868933	1. Give wrong input paramters in configFile. 2. Initialise government object using this values	It should throw SQLSyntaxErrorException: Access denied for user	Same as expected
	G4	Passing file path with .properties extension at the end	Correct file path(with just file name at the end) and valid dbname, user and password in key=value format dbname=jdbc:mysql://db.cs.dal.ca:3306/dhruvp user=dhruvp password=B00868934	1. Provide path with .properties extension for govt configFile. 2. Initialise government object using this path as "new Government(invalidPath)"	It should throw FileNotFoundException.	Same as expected
	G5	Passing improper fomat in key=value pair in configFile	Correct file path(with just file name at the end) and valid dbname, user and password in key=value format dbname=jdbc:mysql://db.cs.dal.ca:3306/dhruvp user=dhruvp password=B00868935	1. Provide incorrect key=value format or skip one line. Example: userdhruv (No equals) 2. Initialise govt object using the above file	It should throw SQLSyntaxErrorException: Access denied for user	Same as expected
Creation of MobileDevice(String configFile,Government gov)	M1	Creation of mobileDevice object passing valid key=value pair and government object	Correct file path(with just file name at the end) and valid address and deviceName address=127.0.0.1 deviceName=Dhruv1	1. Initialise govt object. 2. Provide correct key=value pair with correct path 3. Initialise mobileDevice using "new mobileDevice(String path,govt object)"	mobileDevice object should be created without any errors.	Same as expected
	M2	Passing incorrect path to mobileDevice contructor	Correct file path(with just file name at the end) and valid address and deviceName address=127.0.0.1 deviceName=Dhruv2	1. Initialise govt object. 2. Provide incorrect path in path to mobileDevice 3. Try to initialise mobileDevice using the above path	It should throw FileNotFoundException.	Same as expected
	M3	Passing null instead of govt object in mobileDevice constructor	Correct file path(with just file name at the end) and valid address and deviceName address=127.0.0.1 deviceName=Dhruv3	1. Try to initialise mobileDevice using correct path but pass null instead of govt object as "new mobileDevice(String path,null)"	It should throw NullPointerException	Same as expected
	M4	Passing file path with .properties extension in end of filePath	Correct file path(with just file name at the end) and valid address and deviceName address=127.0.0.1 deviceName=Dhruv4	1. Try to initialise mobileDevice using correct path and .properties extension as "new mobileDevice(String path,gov object)"	It should throw FileNotFoundException.	Same as expected

Record contact between two mobileDevice using recordContact(String individual, int date, int duration)	RC1	Record contact between two different mobiledevices with valid input	1. Two different mobile Contact are needed. 2. Zero or positive date and duration variables 3. mobileDevice.getConfig() returns the config data containing SHA value of address and deviceName.	mb=>mobileDevice object 1. Call mb1.recordContact(mb2.getConfig(),5,5)	A valid contact must be registered which will then be stored in database when mb.synchronisedData() is called	Same as expected
	RC2	Record contact between two same mobileDevice with valid input	1. Two different mobile Contact are needed. 2. Zero or positive date and duration variables 3. mobileDevice.getConfig() returns the config data containing SHA value of address and deviceName.	mb=>mobileDevice object 1. Call mb1.recordContact(mb1.getConfig(),5,5)	A WrongInputException with message "Invalid Contact Input. Please check again" must be shown on console.	Same as expected
	RC3	Pass negative date in recordContact	1. Two different mobile Contact are needed. 2. Zero or positive date and duration variables 3. mobileDevice.getConfig() returns the config data containing SHA value of address and deviceName.	mb=>mobileDevice object 1. Call mb1.recordContact(mb2.getConfig(),-5,5)	A WrongInputException with message "Invalid Contact Input. Please check again" must be shown on console.	Same as expected
	RC4	Pass negative duration in recordContact	1. Two different mobile Contact are needed. 2. Zero or positive date and duration variables 3. mobileDevice.getConfig() returns the config data containing SHA value of address and deviceName.	mb=>mobileDevice object 1. Call mb1.recordContact(mb2.getConfig(),5,-5)	A WrongInputException with message "Invalid Contact Input. Please check again" must be shown on console.	Same as expected
	RC5	Pass null in individual in recordContact	1. Two different mobile Contact are needed. 2. Zero or positive date and duration variables 3. mobileDevice.getConfig() returns the config data containing SHA value of address and deviceName.	mb=>mobileDevice object 1. Call mb1.recordContact(null,5,-5)	A WrongInputException with message "Invalid Contact Input. Please check again" must be shown on console.	Same as expected
	RC6	Pass empty string in individual in recordContact	1. Two different mobile Contact are needed. 2. Zero or positive date and duration variables 3. mobileDevice.getConfig() returns the config data containing SHA value of address and deviceName.	mb=>mobileDevice object 1. Call mb1.recordContact("",5,-5)	A WrongInputException with message "Invalid Contact Input. Please check again" must be shown on console.	Same as expected
	RC7	Multiple Record contact between two different mobile device on same date	1. Two different mobile Contact are needed. 2. Zero or positive date and duration variables 3. mobileDevice.getConfig() returns the config data containing SHA value of address and deviceName.	mb=>mobileDevice object 1. Call mb1.recordContact(mb2.getConfig(),5-5) 2. Call the same statement again similar to 1	When mb.syncData() will be called there should be only single entry where duration will be sum of both duration mb1-mb2-5-10 (mb1 came into contact with mb2 on date 5 for total 10 min)	Same as expected
	P1	Pass valid not null not empty unique string to positiveTest	1. Unique string which is not already present in database is needed. 2. Not null not empty string as testHash 3. Similar testHash should be present in TEST_RESULTS which we get from government object	gov=>Government object mb=>mobileDevice object 1 Call gov.recordTestResult("Abc",5,true) 2.Call mb.positiveTest("Abc")	When mb.syncData() will be called the testHash will be stored in database table POSITIVE_COVID_LIST.	Same as expected

Report positive covid cases to mobile Device using positiveTest(String testHash)	P2	Pass null string in testHash	1. Unique string which is not already present in database is needed. 2. Not null not empty string as testHash 3. Similar testHash should be present in TEST_RESULTS which we get from government object	1. Call mb.positiveTest(null)	A WrongInputException with message Invalid testHash value passed. Please check again will be shown on console	Same as expected
	P3	Pass empty string in testHash	1. Unique string which is not already present in database is needed. 2. Not null not empty string as testHash 3. Similar testHash should be present in TEST_RESULTS which we get from government object	1. Call mb.positiveTest("")	A WrongInputException with message Invalid testHash value passed. Please check again will be shown on console	Same as expected
	P4	Pass only spaces string in testHash	1. Unique string which is not already present in database is needed. 2. Not null not empty string as testHash 3. Similar testHash should be present in TEST_RESULTS which we get from government object	1. Call mb.positiveTest(" ")	A WrongInputException with message Invalid testHash value passed. Please check again will be shown on console	Same as expected

Store all the accumulate data in database by calling md.synchronizeData() which will return if any contacts previosuly tested positive	S1	Call multiple mb.synchronizeData() and with no new contact or test registered between two syncs.	1 We should have mobileDevice and government object.	mb1.recordContact(mb2.getConfig(), 5, 50);
				
				gov.recordTestResult("CovidTest", 15, true);
				
				mb2.positiveTest("CovidTest");
				
				mb2.synchronizeData();
				
				mb1.synchronizeData();
				mb1.synchronizeData();
					Only new data and not the previous data should be stored when calling multiple synchronize on same mobileDevice object. Here we have no new data when calling on mb1 multiple time. Therefore, the data should not duplicate in database	Same as expected

	S2	Call multiple mb.synchronizeData() and with new contact or test registered between two syncs.	1 We should have mobileDevice and government object.	mb1.recordContact(mb2.getConfig(), 5, 50);
				
				gov.recordTestResult("CovidTest", 15, true);
				
				mb2.positiveTest("CovidTest");
				
				mb2.synchronizeData();
				
				mb1.synchronizeData();
				mb1.recordContact("test",1.3);
				mb1.synchronizeData();
					Only new data and not the previous data should be stored when calling multiple synchronize on same mobileDevice object. Here we have new data when calling on mb1 multiple time. Therefore, only new data i.e recordContact should be stored in database.	Same as expected

	S3	Call synchronizeData() on the individual tested positive.	1 We should have mobileDevice and government object.	mb1.recordContact(mb2.getConfig(), 5, 50);
				
				gov.recordTestResult("CovidTest", 15, true);
				
				mb1.positiveTest("CovidTest");
				
				mb1.synchronizedData()
					The method should return false regardless of the individual being positive	Same as expected

	S4	Call synchronizeData() before any contact or positive Test reported	1 We should have mobileDevice and government object.	mb1.synchronizedData();
				
				mb1.recordContact(mb2.getConfig(), 5, 50);
				
				gov.recordTestResult("CovidTest", 15, true);
				
				mb1.positiveTest("CovidTest");
					The method should return false.	Same as expected

	S5	Call synchronizeData() multiple time such that we change the result from false to true by recording positive contact	1 We should have mobileDevice and government object.	mb1.recordContact(mb8.getConfig(), 5, 50);
				
				mb1.synchronizeData();
				
				gov.recordTestResult("CovidTest6", 15, true);
				mb8.positiveTest("CovidTest6");
				mb8.synchronizeData();
				
				mb1.synchronizeData();
					The first mb1.synchronize should return false and then we change it to true in last line by recording positive case	Same as expected

Store COVID test results in database using Government object and method recordTestResult(String testHash,int data,boolean result)	RTS1	Pass negative date in recordTestResult	1. Positive date and not null not empty unique testHash	1. Call gov.recordTestResult("Test1",-5,true)	The method should throw InvalidInputException	Same as expected
	RTS2	Pass null string in recordTestResult	1. Positive date and not null not empty unique testHash	1. Call gov.recordTestResult(null,5,true)	The method should throw InvalidInputException	Same as expected
	RTS3	Pass empty string in recordTestResult	1. Positive date and not null not empty unique testHash	1. Call gov.recordTestResult("",5,true)	The method should throw InvalidInputException	Same as expected
	RTS4	Pass only spaces string in recordTestResult	1. Positive date and not null not empty unique testHash	1. Call gov.recordTestResult(" ",5,true)	The method should throw InvalidInputException	Same as expected
	RTS5	Pass duplicate testHash in recordTestResult	1. Positive date and not null not empty unique testHash	1. Call gov.recordTestResult("Test1",5,true) 2. Call gov.recordTestResult("Test1",5.true)	The method should throw SQLIntegrityConstraintViolationException	Same as expected
	RTS6	Pass valid input and unique testHash in recordTestResult	1. Positive date and not null not empty unique testHash	1. Call gov.recordTestResult("Covidtest",5,true)	CovidTest1 will be saved in TEST_RESULTS table in database	Same as expected
Ability to report large gatherings at particular date using findGatherings(int date,int minSize,int minTime,float density)	FG1	Pass negative date in findGatherings	1 The paramteters must be valid being 0 or positive. 2. The density must be between 0 and 1	1. Call gov.findGatherings(-5, 5, 5, 0.01f)	The method should throw InvalidInputException	Same as expected
	FG2	Pass negative minSize in findGatherings	1 The paramteters must be valid being 0 or positive. 2. The density must be between 0 and 1	1. Call gov.findGatherings(5,- 5, 5, 0.01f)	The method should throw InvalidInputException	Same as expected
	FG3	Pass negative minTime in findGatherings	1 The paramteters must be valid being 0 or positive. 2. The density must be between 0 and 1	1. Call gov.findGatherings(5,5,- 5, 0.01f)	The method should throw InvalidInputException	Same as expected
	FG4	Pass density such that it is not in between 0 and 1	1 The paramteters must be valid being 0 or positive. 2. The density must be between 0 and 1	1. Call gov.findGatherings(5,5, 5, 1.01f) OR 2. Call gov.findGatherings(5,5,5,-1f);	The method should throw InvalidInputException	Same as expected
	FG5	Pass valid input in findGatherings	1 The paramteters must be valid being 0 or positive. 2. The density must be between 0 and 1	1. Call gov.findGatherings(5,5,-5, 0.01f)	The method should return 0 as there are no contacts yet reported before findGatherings is called	Same as expected

recordContact

Input Validation

- Negative date
- Negative duration
- Null individual string
- Empty individual string

Boundary Cases

- Date as 0
- Duration as 0
- Individual string only contains spaces

Control Flow

- Normal operation
- Duplicate contact having same individual string and date; => We add duration in this case
- Contact with itself

positiveTest

Input validation

- Null testHash
- Empty testHash

- Only space testHash

Boundary Cases
- Long individual string
- Only 1 letter
- Only 1 alphabet
- Short individual string

Control Flow
- Normal operation
- Passed testHash does not match with testHash passed by recordTestResult of government
- Pass duplicate testHash

synchronizeData

Boundary cases
- Absolute difference between test date and contact date is 0.
- Absolute difference between test date and contact date is 14.
- Absolute difference between test date and contact date is more than 14.
- Absolute difference between test date and contact date is less than 14.

Control Flow
- Normal operation
- Multiple synchronizeData with no new data between subsequent sync
- Multiple synchronizeData with new data between subsequent sync
- Change synchronizeData from false to true by inserting new positive contact
- Call synchronizeData on individual tested positive
- Call synchronizeData before recording contact or before any positive test

recordTestResult

Input Validation
- Negative date
- Null testHash
- Empty testHash
- Only space in testHash

Boundary cases
- Long testHash string
- testHash having length as 1 with one letter or one alphabet
- Date 0

Control Flow
- Normal operation
- Pass duplicate testHash string

findGatherings

Input Validation
- Negative date
- Negative minTime
- Negative minSize
- Negative density
- Density not between 0 and 1

Boundary Cases
- Date 0
- minSize 0
- minTime 0
- density 0
- density 1
- high density not more than 1
- low density not less than 0

Control Flow
- Normal operation
- Call findGatherings before any contact
- Change findGathering value by recording new contacts.

Data flow

- Normal Flow
  - Record contacts
  - Sync contacts
  - Record positive test by government
  - Record positive result by mobileDevice
  - Sync contacts to find if anyone was near any covid patients
  - findGatherings at particular date
- Call positive results before recording contacts
- Call sync contact before recording contacts
- findGathering at any point in between recording contacts, syncing contacts, etc
- Call record contacts at any point
- Call positive result by mobileDevice before calling positive result by government
- Call positive result after findGatherings