Concordia University
Engineering and Computer Science

COMP 6231
Distributed Systems Design

Distributed Class Management System (DCMS) using Java RMI

Submitted To: Prof. M. Taleb

Contributors

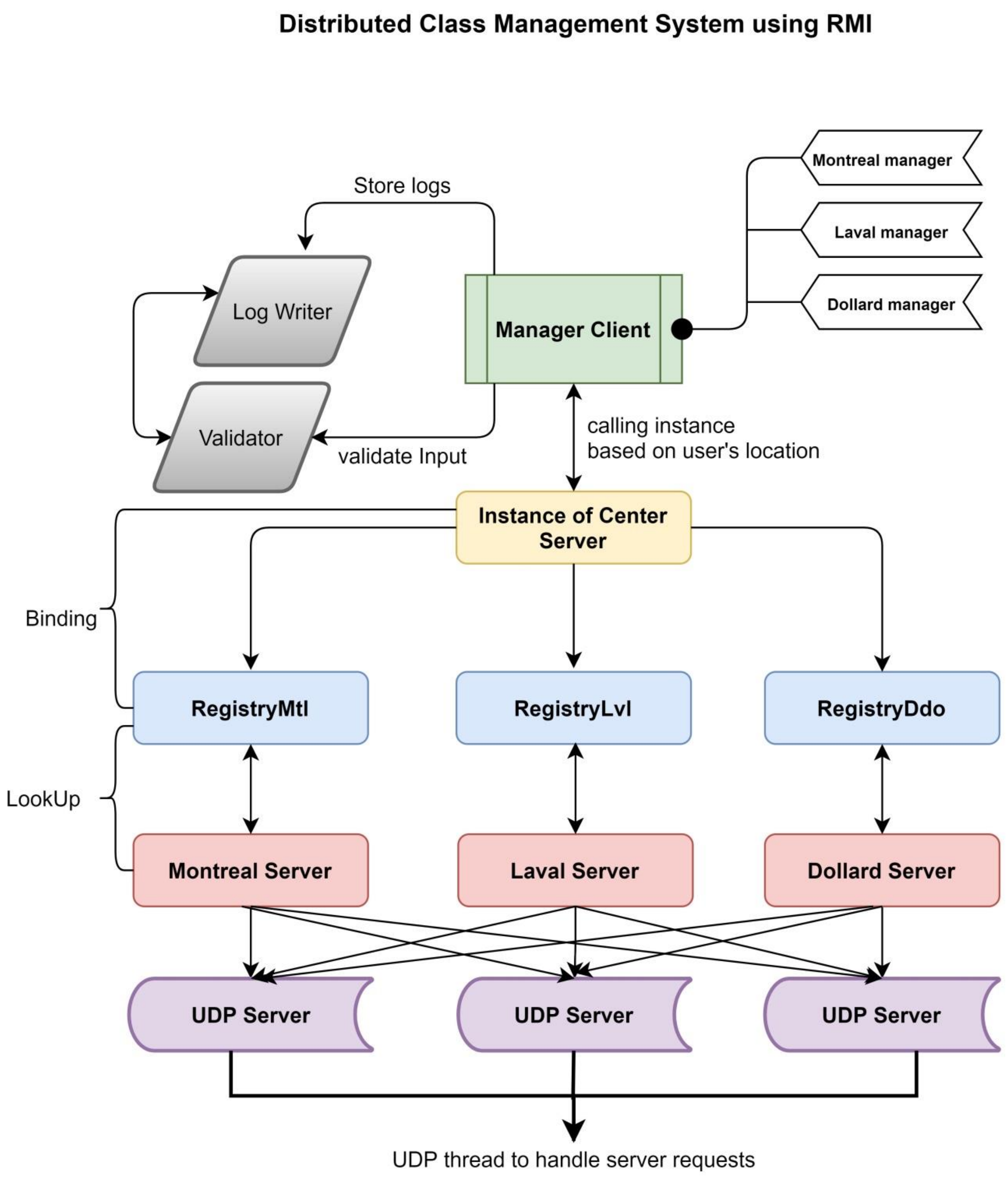
Zalakben Rajendrakumar Patel (40164315)

Manimaran Palani (40167543)

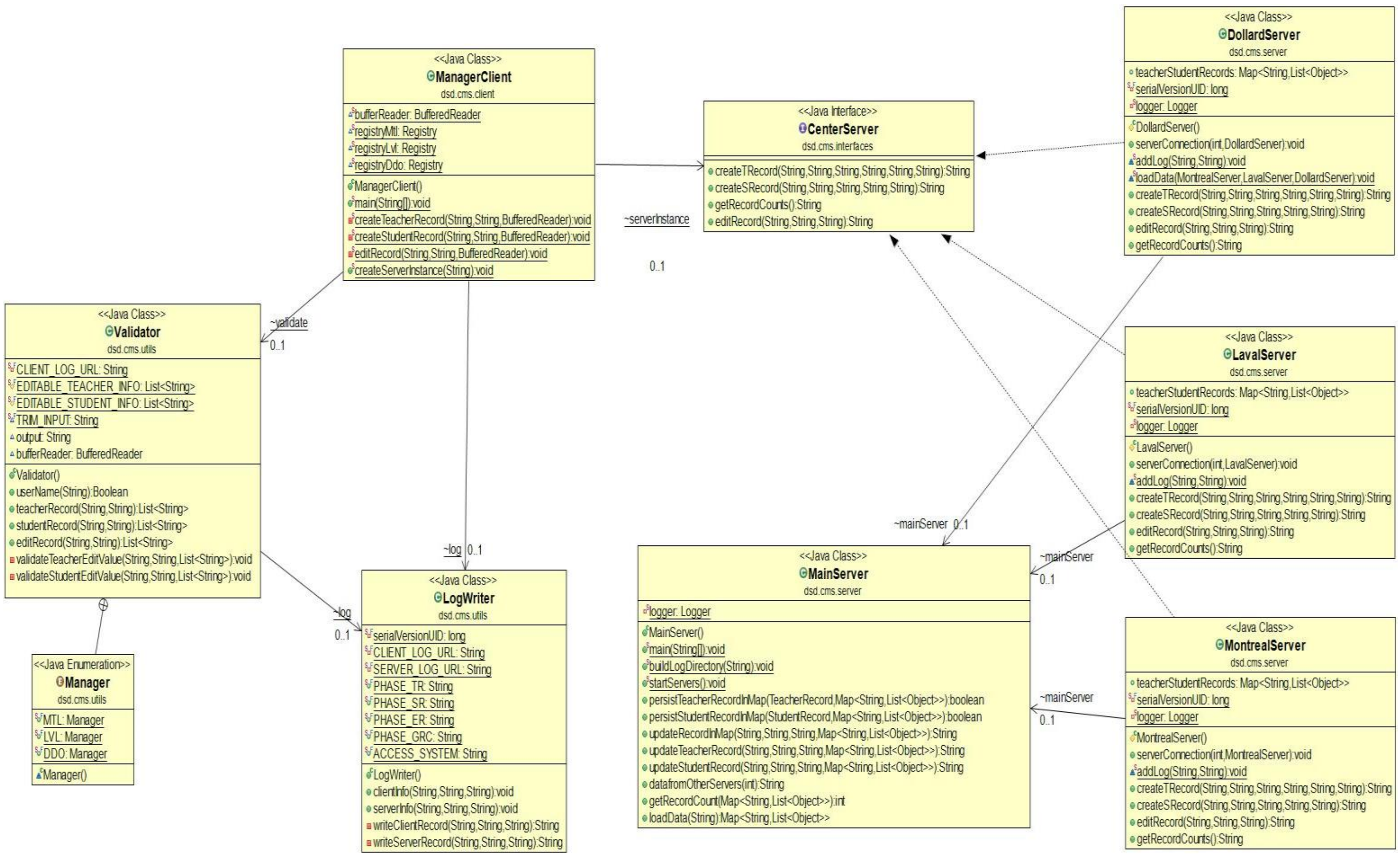
Description:

Distributed Class Management system (DCMS) is implemented as a distributed system to be used by centre managers, who are responsible for managing teachers' and students' information across 3 centres: Montreal, Laval and Dollard-des-Ormeaux. Each location has its own server and maintains teachers' and students' records. Centre managers are responsible for adding, editing teachers' and students' records at his/her own centre. Additionally, manager can also get count of records present at other servers through UDP connection. The system is built using Java RMI where manager can see a single system handling simultaneous requests with proper synchronization.

Design Architecture:



Class Diagram:



<<Java Class>>
ManagerClient
dsd.cms.client
bufferReader: BufferedReader
registryMtl: Registry
registryLvl: Registry
registryDdo: Registry
ManagerClient()
main(String[]):void
createTeacherRecord(String,String,BufferedReader):void
createStudentRecord(String,String,BufferedReader):void
editRecord(String,String,BufferedReader):void
createServerInstance(String):void
<<Java Interface>>
CenterServer
dsd.cms.interfaces
createTRecord(String,String,String,String,String,String):String
createSRecord(String,String,String,String,String):String
getRecordCounts():String
editRecord(String,String,String):String
~serverInstance
0..1
<<Java Class>>
DollardServer
dsd.cms.server
teacherStudentRecords: Map<String,List<Object>>
serialVersionUID: long
logger: Logger
DollardServer()
serverConnection(int,DollardServer):void
addLog(String,String):void
loadData(MontrealServer,LavalServer,DollardServer):void
createTRecord(String,String,String,String,String,String):String
createSRecord(String,String,String,String,String):String
editRecord(String,String,String):String
getRecordCounts():String
<<Java Class>>
Validator
dsd.cms.utils
CLIENT_LOG_URL: String
EDITABLE_TEACHER_INFO: List<String>
EDITABLE_STUDENT_INFO: List<String>
TRIM_INPUT: String
output: String
bufferReader: BufferedReader
Validator()
userName(String):Boolean
teacherRecord(String,String):List<String>
studentRecord(String,String):List<String>
editRecord(String,String):List<String>
validateTeacherEditValue(String,String,List<String>):void
validateStudentEditValue(String,String,List<String>):void
~validate
0..1
<<Java Class>>
LavalServer
dsd.cms.server
teacherStudentRecords: Map<String,List<Object>>
serialVersionUID: long
logger: Logger
LavalServer()
serverConnection(int,LavalServer):void
addLog(String,String):void
createTRecord(String,String,String,String,String,String):String
createSRecord(String,String,String,String,String):String
editRecord(String,String,String):String
getRecordCounts():String
~mainServer 0..1
~log 0..1
~log
0..1
<<Java Class>>
LogWriter
dsd.cms.utils
serialVersionUID: long
CLIENT_LOG_URL: String
SERVER_LOG_URL: String
PHASE_TR: String
PHASE_SR: String
PHASE_ER: String
PHASE_GRC: String
ACCESS_SYSTEM: String
LogWriter()
clientInfo(String,String,String):void
serverInfo(String,String,String):void
writeClientRecord(String,String,String):String
writeServerRecord(String,String,String):String
<<Java Class>>
MainServer
dsd.cms.server
logger: Logger
MainServer()
main(String[]):void
buildLogDirectory(String):void
startServers():void
persistTeacherRecordInMap(TeacherRecord,Map<String,List<Object>>):boolean
persistStudentRecordInMap(StudentRecord,Map<String,List<Object>>):boolean
updateRecordInMap(String,String,String,Map<String,List<Object>>):String
updateTeacherRecord(String,String,String,Map<String,List<Object>>):String
updateStudentRecord(String,String,String,Map<String,List<Object>>):String
datafromOtherServers(int):String
getRecordCount(Map<String,List<Object>>):int
loadData(String):Map<String,List<Object>>
~mainServer
0..1
~mainServer
0..1
<<Java Enumeration>>
Manager
dsd.cms.utils
MTL: Manager
LVL: Manager
DDO: Manager
Manager()
<<Java Class>>
MontrealServer
dsd.cms.server
teacherStudentRecords: Map<String,List<Object>>
serialVersionUID: long
logger: Logger
MontrealServer()
serverConnection(int,MontrealServer):void
addLog(String,String):void
createTRecord(String,String,String,String,String,String):String
createSRecord(String,String,String,String,String):String
editRecord(String,String,String):String
getRecordCounts():String

Architecture Description:

1) CenterServer (RMI Interface):

Responsible for giving abstract declaration of functionalities like, adding Teacher/Student record, editing existing record, getting count of records from all servers as below:

- createTRecord(String firstName, String lastName, String address, String phone, String specialization, String location)
- createSRecord(String firstName, String lastName, String courseRegistered, String status, String statusDate)
- getRecordCounts()
- editRecord(String recordID, String fieldName, String newValue)

2) MontrealServer, LavalServer, DollardServer (RMI Servers):

These servers provide actual implementation for functionalities defined in RMI Interface (CenterServer). These servers are being invoked from manager client based on the manager’s location. Servers are also responsible for registering their instances in RMI registries and receiving UDP data packets from other servers.

3) MainServer:

MainServer is a main class which is responsible for starting all three locations' servers by invoking their 'serverConnection' method. It also has few methods which are common across all the servers in terms of implementation and which avoids the duplication of code.

4) ManagerClient:

This class is responsible for taking inputs from user about identifying manager ID, operations which are to be performed at that location and all required inputs for performing that operation. ManagerClient looks up the registries in order to find correct location sever and invokes required method.

7) StudentRecord (Model):

StudentRecord class is a data model class which has fields like first name, last name, courses registered, status (Active/InActive), status date and record id which are responsible for persisting new student's data.

8) TeacherRecord (Model):

TeacherRecord class is a data model class which has fields like first name, last name, address, phone, specialization, location and record id which are responsible for persisting new teacher's data at any location.

9) Regex:

This class is responsible for performing the pattern matching according to defined regex patterns for inputs of type string, number, date etc.

10) Validator:

Validator class performs all required validations on the given inputs by manager before redirecting to actual implementation of servers. It requires first name, last name to be string, phone number to be a number, location from specific list: MTL, LVL, DDO, status date to be in specific date format etc.

RMI Registry:

Instances of MontrealServer, LavalServer and DollardServer are bound with the 3 different RMI registries which can later be looked up by ManagerClient.

- LocateRegistry.createRegistry(9991).bind("Montreal",montrealServer);
- LocateRegistry.createRegistry(9992).bind("Laval",lavalServer);
- LocateRegistry.createRegistry(9993).bind("Dollard",dollardServer);

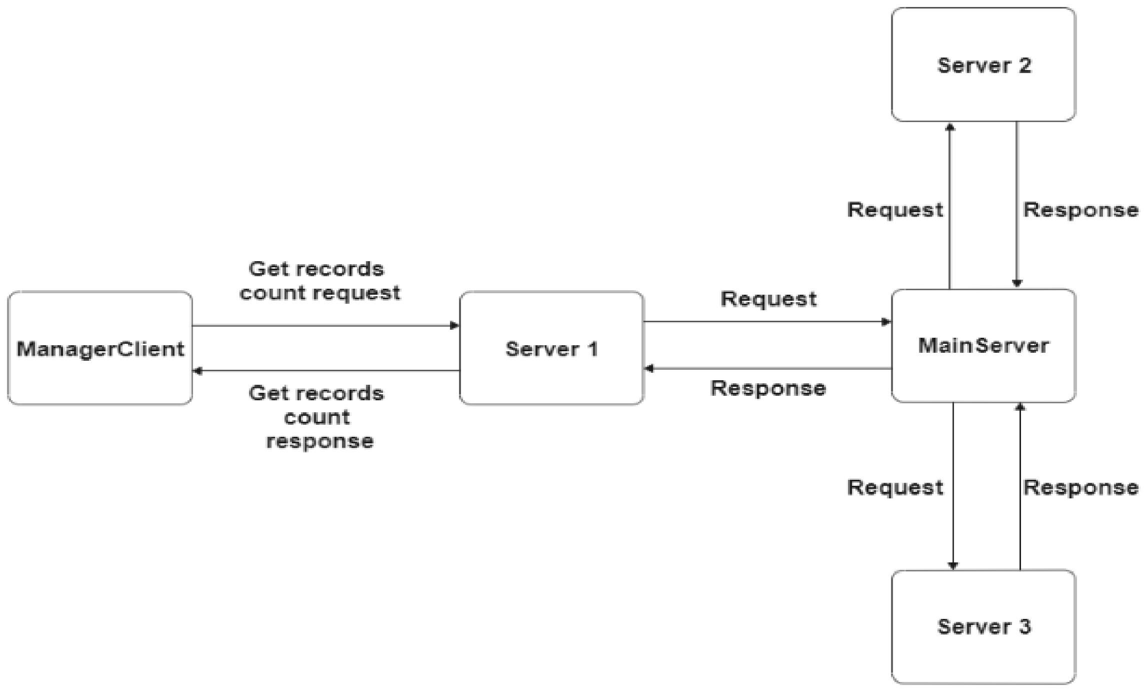
Data Models:

Hashmap is used as a data model, which is maintained at each server locations. Key of a hashmap is an alphabet whereas value against that is a list of Teacher/Student records whose last name starts with that alphabet.

The Most Important/Difficult Part:

Handling multiple requests coming at the same or different locations concurrently, blocking resources for proper updates of the data, implementing concurrency and synchronization have been important and challenging part of the assignment.

UDP Server Connection:



Logging:

To keep track of all the operations performed by manager and all the operations performed at particular location, separate logs files are being maintained for both managers and locations.

Concurrency:

New thread is created to communicate to each of the servers in order to handle multiple add/edit/get record count requests coming at the same time at different locations.

Synchronization:

In order to handle multiple requests in single location, coming at the same time, synchronized methods are used, which can block its implementation to have only one thread executing inside them. With this synchronization technique, we can avoid multiple requests trying to update the same hashmap at the same time which can lead to faulty results.

Initial Data Loading:

All three location servers will be loaded initially with sample data stored in text files: MontrealData, DollardData, LavalData located at src/main/resources.

Overall Execution Flow:

Below are the steps performed in sequence when any manager tries to log in and performs any operation.

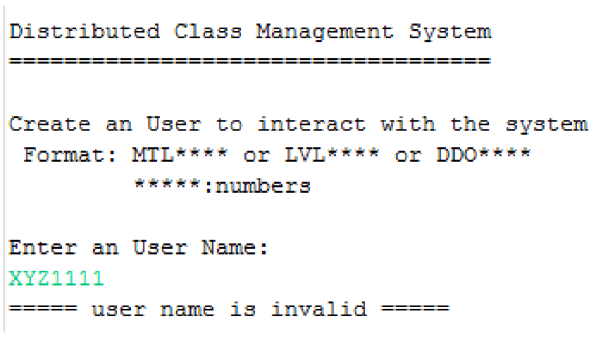
- All 3 location based servers have to be started by running MainServer.
- ManagerClient performs required validations of input information passed by manager.
- Post validation success, ManagerClient communicates with either of three servers based on manager ID passed and RMI registry lookup.
- Location server receives the data and performs the requested operation. If required, it also invokes other locations' server to fetch the data through UDP connection.
- After performing operation, location server will return back the results ManagerClient.

Steps mentioned below are used to run the application.

- ➢ Run MainServer.java to start all three location servers.
- ➢ Run ManagerClient.java to start manager client.

Test Scenarios:

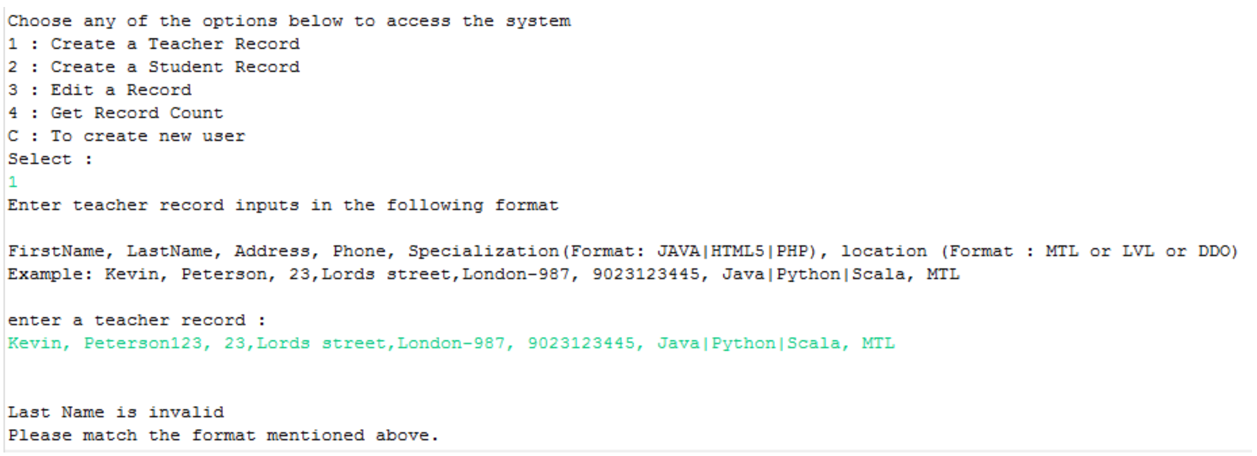
1) Manager ID given in incorrect format :



```
Distributed Class Management System
===================================

Create an User to interact with the system
 Format: MTL**** or LVL**** or DDO****
         *****:numbers

Enter an User Name:
XYZ1111
===== user name is invalid =====
```

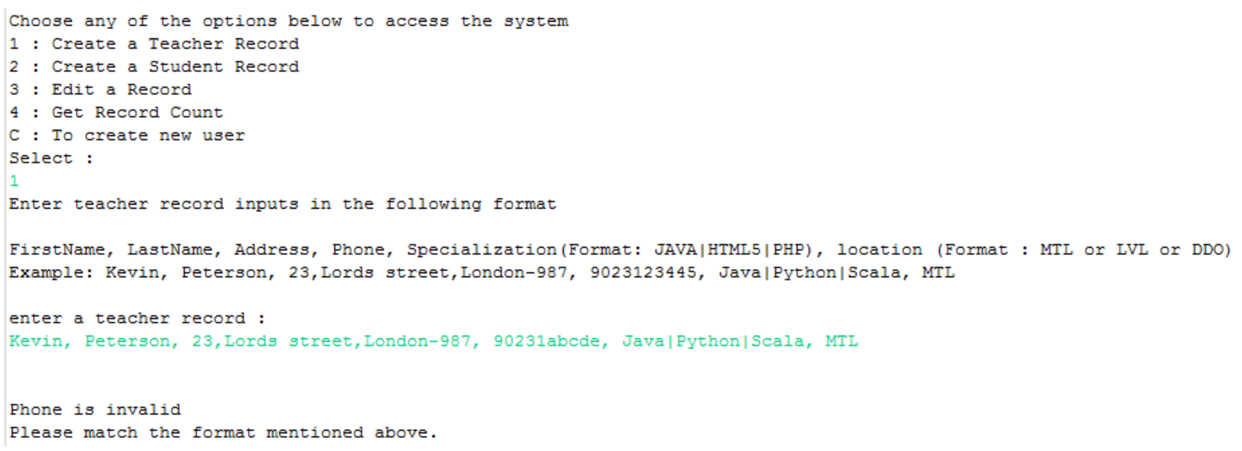
2) New teacher creation test scenarios :

a. Validation failure in creating teacher with first name/last name having a number in it :



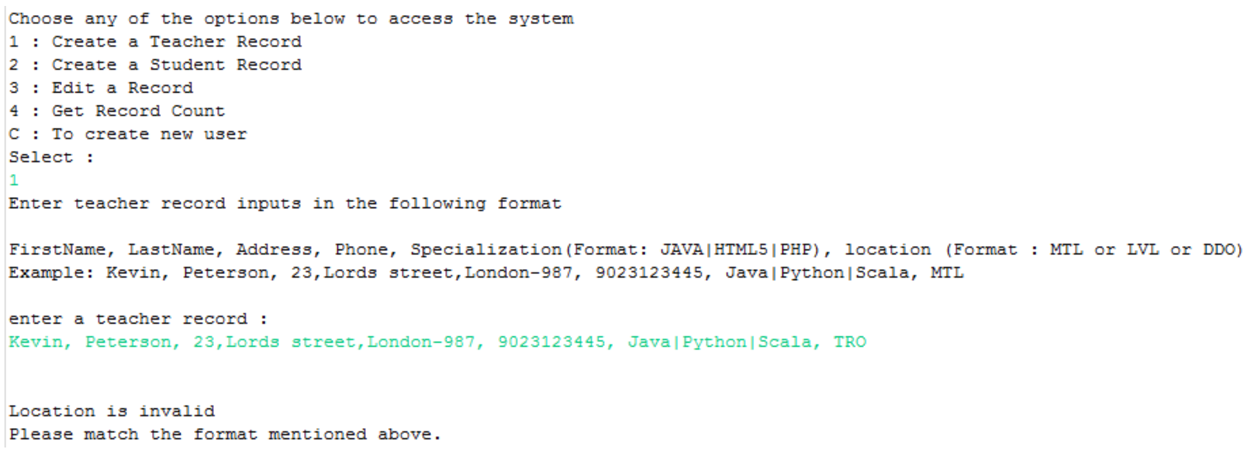
```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
1
Enter teacher record inputs in the following format

FirstName, LastName, Address, Phone, Specialization(Format: JAVA|HTML5|PHP), location (Format : MTL or LVL or DDO)
Example: Kevin, Peterson, 23,Lords street,London-987, 9023123445, Java|Python|Scala, MTL

enter a teacher record :
Kevin, Peterson123, 23,Lords street,London-987, 9023123445, Java|Python|Scala, MTL


Last Name is invalid
Please match the format mentioned above.
```

b. Validation failure in creating teacher with incorrect phone number format :



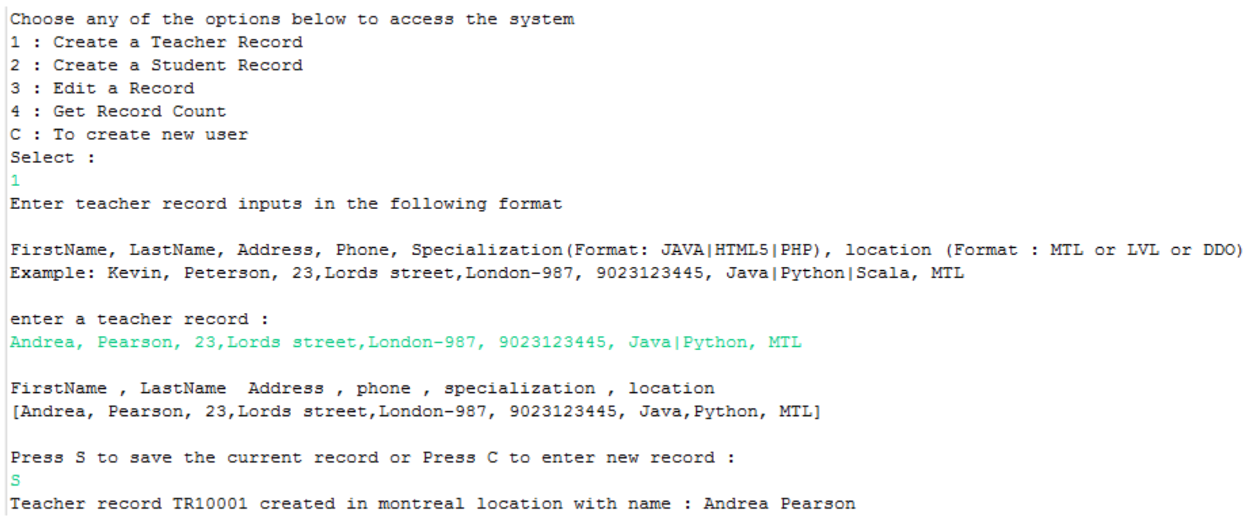
```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
1
Enter teacher record inputs in the following format

FirstName, LastName, Address, Phone, Specialization(Format: JAVA|HTML5|PHP), location (Format : MTL or LVL or DDO)
Example: Kevin, Peterson, 23,Lords street,London-987, 9023123445, Java|Python|Scala, MTL

enter a teacher record :
Kevin, Peterson, 23,Lords street,London-987, 90231abcde, Java|Python|Scala, MTL


Phone is invalid
Please match the format mentioned above.
```

c. Validation failure in creating teacher with location other than MTL / LVL / DDO :



```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
1
Enter teacher record inputs in the following format

FirstName, LastName, Address, Phone, Specialization(Format: JAVA|HTML5|PHP), location (Format : MTL or LVL or DDO)
Example: Kevin, Peterson, 23,Lords street,London-987, 9023123445, Java|Python|Scala, MTL

enter a teacher record :
Kevin, Peterson, 23,Lords street,London-987, 9023123445, Java|Python|Scala, TRO


Location is invalid
Please match the format mentioned above.
```

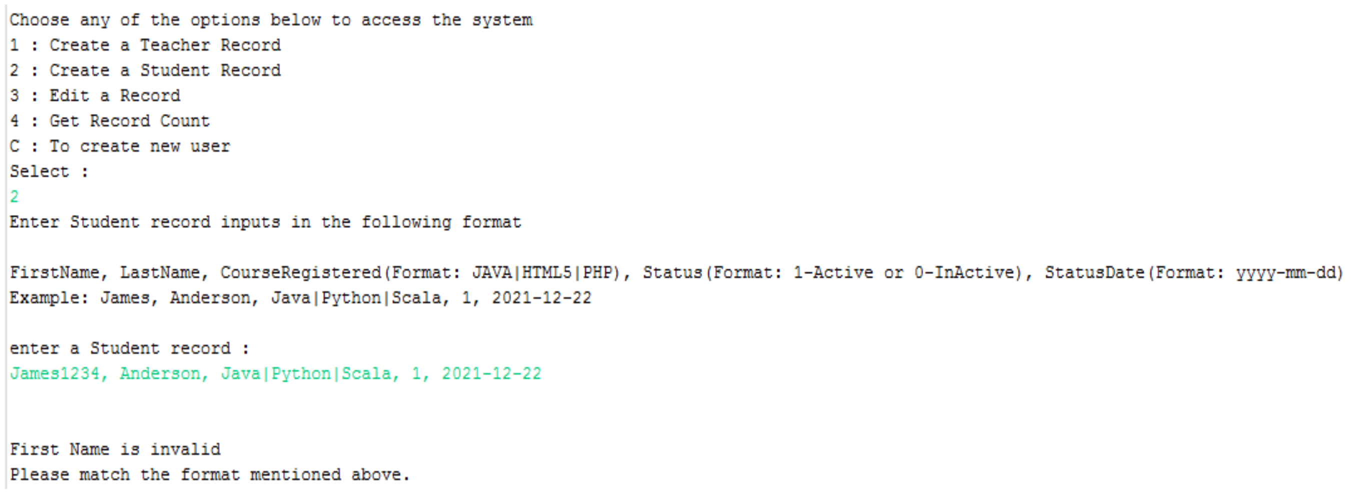
d. Successfully creating new teacher record :



```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
1
Enter teacher record inputs in the following format

FirstName, LastName, Address, Phone, Specialization(Format: JAVA|HTML5|PHP), location (Format : MTL or LVL or DDO)
Example: Kevin, Peterson, 23,Lords street,London-987, 9023123445, Java|Python|Scala, MTL

enter a teacher record :
Andrea, Pearson, 23,Lords street,London-987, 9023123445, Java|Python, MTL

FirstName , LastName  Address , phone , specialization , location
[Andrea, Pearson, 23,Lords street,London-987, 9023123445, Java,Python, MTL]

Press S to save the current record or Press C to enter new record :
S
Teacher record TR10001 created in montreal location with name : Andrea Pearson
```

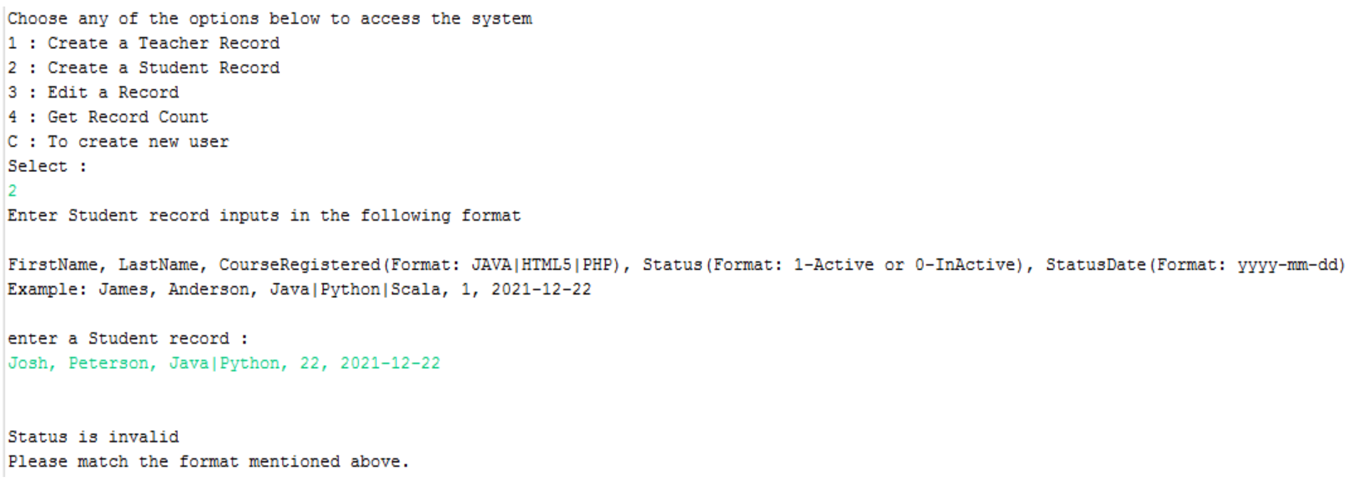
3) New student creation test scenarios :

a. Validation failure in creating student with first name/last name having a number in it :



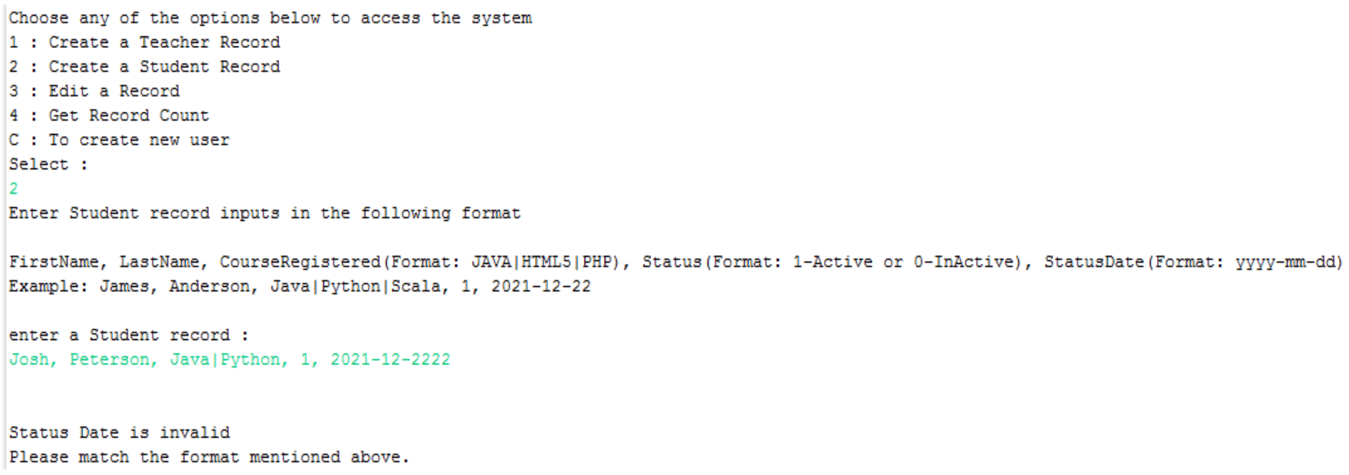
```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
2
Enter Student record inputs in the following format

FirstName, LastName, CourseRegistered(Format: JAVA|HTML5|PHP), Status(Format: 1-Active or 0-InActive), StatusDate(Format: yyyy-mm-dd)
Example: James, Anderson, Java|Python|Scala, 1, 2021-12-22

enter a Student record :
James1234, Anderson, Java|Python|Scala, 1, 2021-12-22


First Name is invalid
Please match the format mentioned above.
```

b. Validation failure in creating student with incorrect status (Other than : 0 – Active / 1 – InActive) :



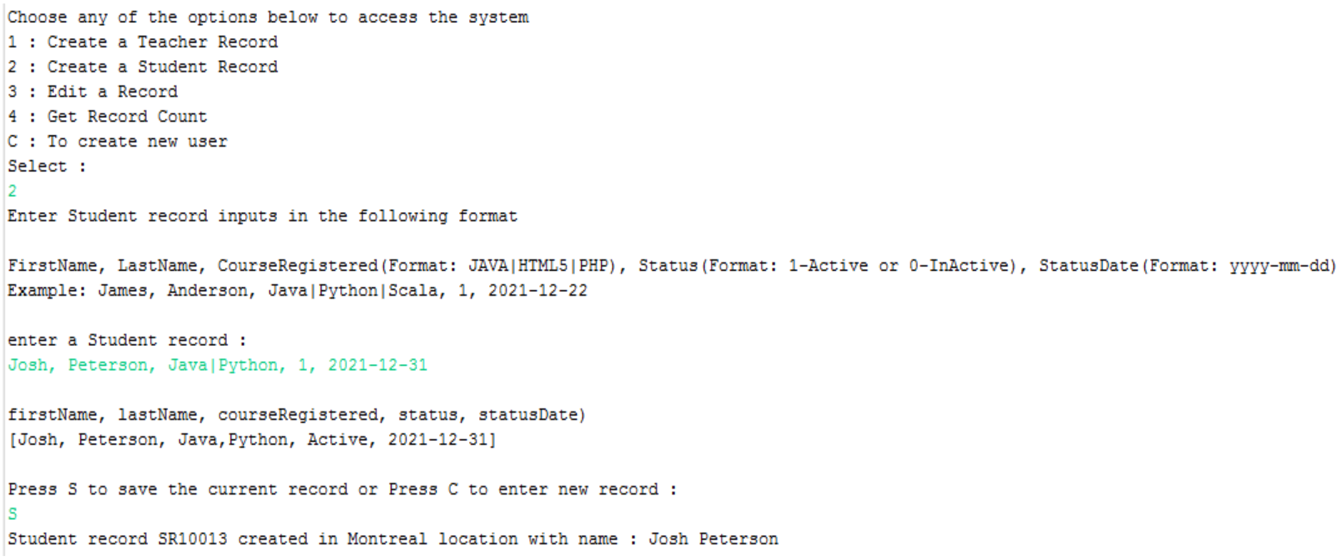
```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
2
Enter Student record inputs in the following format

FirstName, LastName, CourseRegistered(Format: JAVA|HTML5|PHP), Status(Format: 1-Active or 0-InActive), StatusDate(Format: yyyy-mm-dd)
Example: James, Anderson, Java|Python|Scala, 1, 2021-12-22

enter a Student record :
Josh, Peterson, Java|Python, 22, 2021-12-22


Status is invalid
Please match the format mentioned above.
```

c. Validation failure in creating student with incorrect status date format :



```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
2
Enter Student record inputs in the following format

FirstName, LastName, CourseRegistered(Format: JAVA|HTML5|PHP), Status(Format: 1-Active or 0-InActive), StatusDate(Format: yyyy-mm-dd)
Example: James, Anderson, Java|Python|Scala, 1, 2021-12-22

enter a Student record :
Josh, Peterson, Java|Python, 1, 2021-12-2222


Status Date is invalid
Please match the format mentioned above.
```

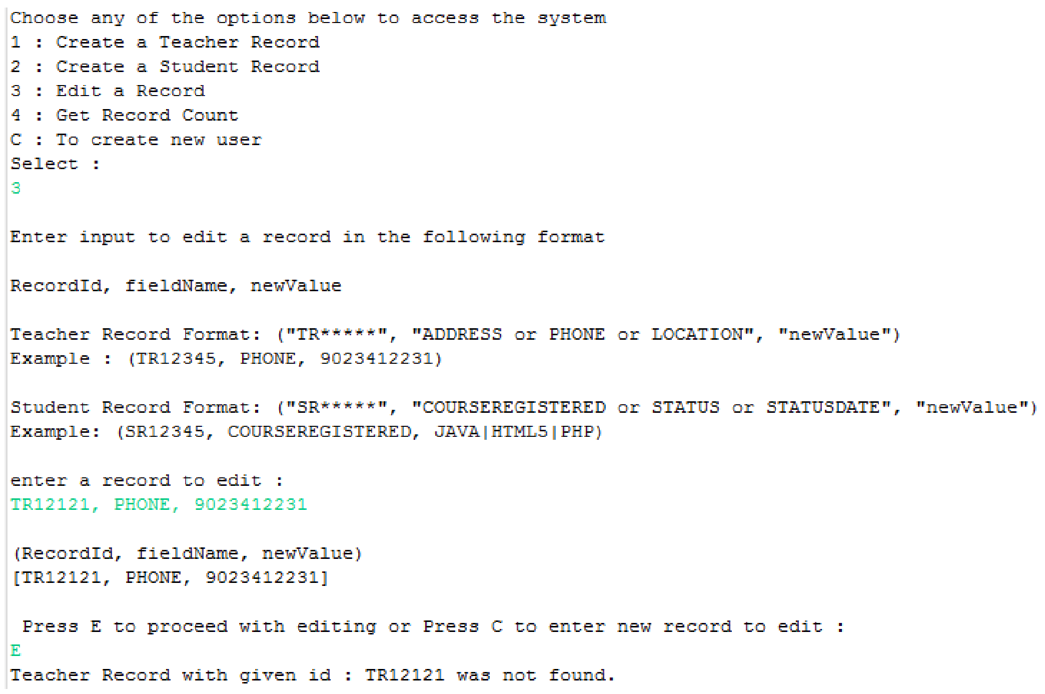
d. Successfully creating new student record :



```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
2
Enter Student record inputs in the following format

FirstName, LastName, CourseRegistered(Format: JAVA|HTML5|PHP), Status(Format: 1-Active or 0-InActive), StatusDate(Format: yyyy-mm-dd)
Example: James, Anderson, Java|Python|Scala, 1, 2021-12-22

enter a Student record :
Josh, Peterson, Java|Python, 1, 2021-12-31

firstName, lastName, courseRegistered, status, statusDate)
[Josh, Peterson, Java,Python, Active, 2021-12-31]

Press S to save the current record or Press C to enter new record :
S
Student record SR10013 created in Montreal location with name : Josh Peterson
```

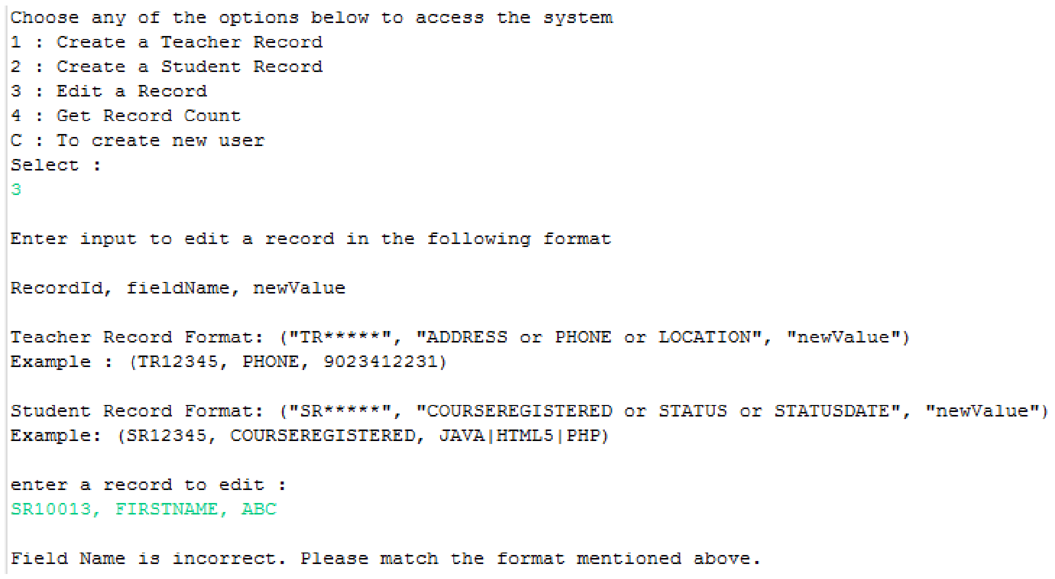
4) Edit existing teacher/student record :

a. Editing teacher/student record with invalid Record ID :



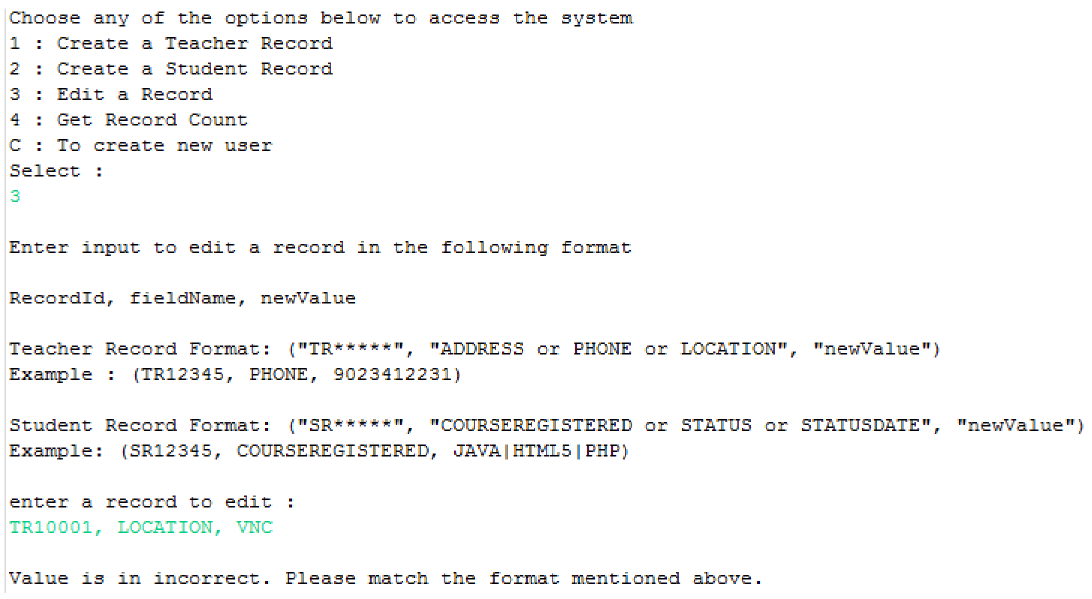
```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
3

Enter input to edit a record in the following format

RecordId, fieldName, newValue

Teacher Record Format: ("TR*****", "ADDRESS or PHONE or LOCATION", "newValue")
Example : (TR12345, PHONE, 9023412231)

Student Record Format: ("SR*****", "COURSEREGISTERED or STATUS or STATUSDATE", "newValue")
Example: (SR12345, COURSEREGISTERED, JAVA|HTML5|PHP)

enter a record to edit :
TR12121, PHONE, 9023412231

(RecordId, fieldName, newValue)
[TR12121, PHONE, 9023412231]

 Press E to proceed with editing or Press C to enter new record to edit :
E
Teacher Record with given id : TR12121 was not found.
```

b. Editing teacher/student record with invalid field name :



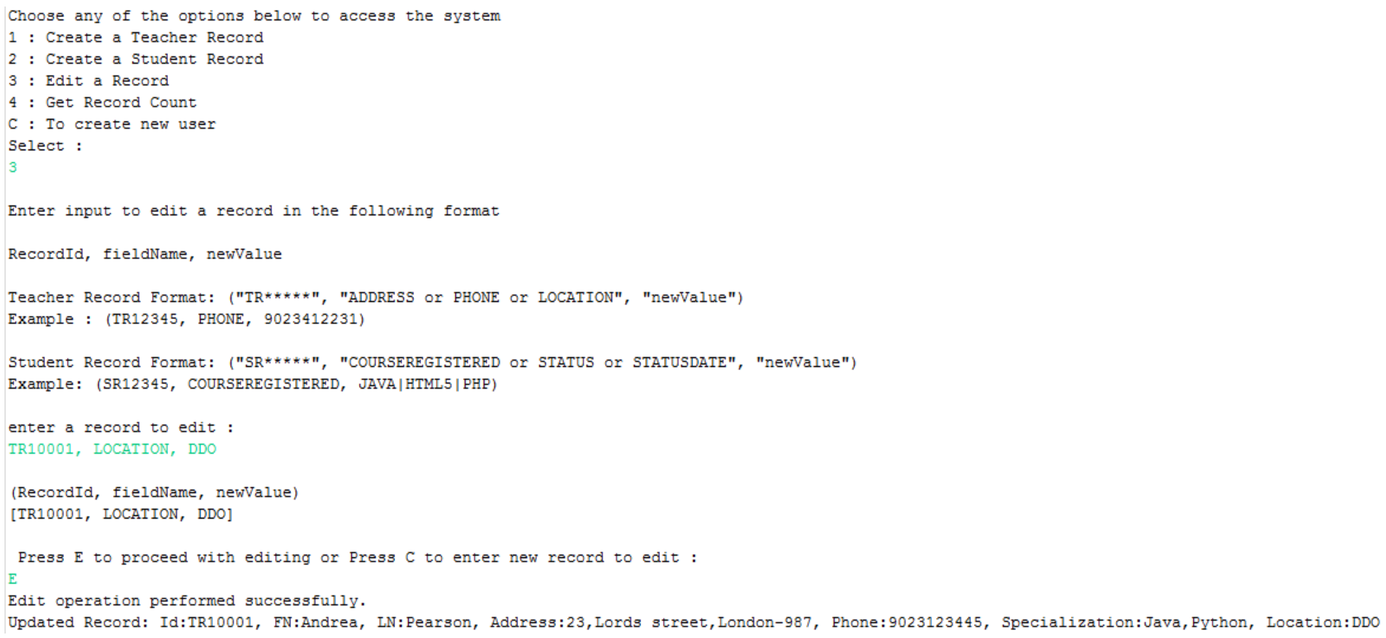
```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
3

Enter input to edit a record in the following format

RecordId, fieldName, newValue

Teacher Record Format: ("TR*****", "ADDRESS or PHONE or LOCATION", "newValue")
Example : (TR12345, PHONE, 9023412231)

Student Record Format: ("SR*****", "COURSEREGISTERED or STATUS or STATUSDATE", "newValue")
Example: (SR12345, COURSEREGISTERED, JAVA|HTML5|PHP)

enter a record to edit :
SR10013, FIRSTNAME, ABC

Field Name is incorrect. Please match the format mentioned above.
```

c. Editing teacher/student record with incorrect new value (Example : Invalid value for location) :



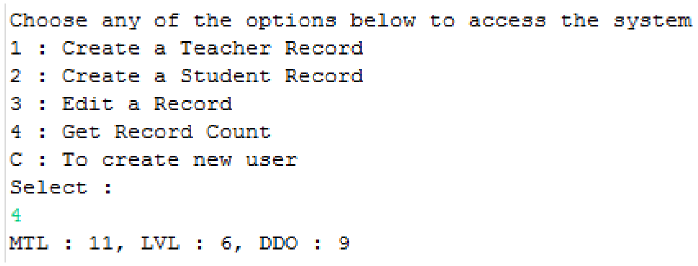
```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
3

Enter input to edit a record in the following format

RecordId, fieldName, newValue

Teacher Record Format: ("TR*****", "ADDRESS or PHONE or LOCATION", "newValue")
Example : (TR12345, PHONE, 9023412231)

Student Record Format: ("SR*****", "COURSEREGISTERED or STATUS or STATUSDATE", "newValue")
Example: (SR12345, COURSEREGISTERED, JAVA|HTML5|PHP)

enter a record to edit :
TR10001, LOCATION, VNC

Value is in incorrect. Please match the format mentioned above.
```

d. Successfully editing teacher/student record :



```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
3

Enter input to edit a record in the following format

RecordId, fieldName, newValue

Teacher Record Format: ("TR*****", "ADDRESS or PHONE or LOCATION", "newValue")
Example : (TR12345, PHONE, 9023412231)

Student Record Format: ("SR*****", "COURSEREGISTERED or STATUS or STATUSDATE", "newValue")
Example: (SR12345, COURSEREGISTERED, JAVA|HTML5|PHP)

enter a record to edit :
TR10001, LOCATION, DDO

(RecordId, fieldName, newValue)
[TR10001, LOCATION, DDO]

 Press E to proceed with editing or Press C to enter new record to edit :
E
Edit operation performed successfully.
Updated Record: Id:TR10001, FN:Andrea, LN:Pearson, Address:23,Lords street,London-987, Phone:9023123445, Specialization:Java,Python, Location:DDO
```

5) Finding number of records present at each server locations :



```
Choose any of the options below to access the system
1 : Create a Teacher Record
2 : Create a Student Record
3 : Edit a Record
4 : Get Record Count
C : To create new user
Select :
4
MTL : 11, LVL : 6, DDO : 9
```