Compact Distributed Objects

Course Name: Topics in Distributed Systems

Course: CS632A

GROUP - 02

Name: BIDYA SARKAR, Roll -18111011

Name: NIRJHAR ROY, Roll -18111409

Date: 10-NOV-2018

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

Contents:

ABSTRACT:

Student Management System is an application of distributed compact object. Students and administrator are allowed to perform certain operations on student database based on their role. The idea is having encapsulating data through a compact object service mechanism. The main database is maintained at a central server. The client requests for database services are supplied in form of compact objects which contains the mutable and nonmutable fields, permitted mutating operations on data with range checks, last modified and any other field which can be used to store log for the purpose of recoveries. The client gets the object and operates on the field as per specification and returns the same to the servers. The server after gathering all compact objects from same database examines non-conflicting updates and applies them to database. In case of conflicting updates, the server assembles the log and leaves it for manual updates. The database is not updated directly when conflicting updates are detected.

FUNCTIONALITY:

For Client-side we have implemented two roles:

a. Student
b. Principal

Based on “role” client is allowed to update certain fields:

“Student” and “principal” will be allowed to update certain fields in student record based on their role. Accordingly the constraints on the student record field will be changed.

“Student” is allowed to update below mentioned fields of student record:

1. Name
2. Stream
3. Contact
4. Address
5. Doj

“Principal” is allowed to update below mentioned fields of student record:

1. Name
2. Stream
3. Degree
4. Contact
5. Cpi
6. Address

7. Doj

“Student” role will have constraints on fields like:

1. Stream: Name of streams allowed like CSE,ECE,ME,MSE
2. DOJ : year range for student’s date of joining like 1900 to 2018

“Principal” role will have constraints on fields like:

1. Stream: Name of streams allowed like CSE,ECE,ME,MSE
2. CPI : number range is provided between 0 to 10
3. Degree: Degree allowed like B.Tech,M.Tech,MS,DUAL
4. DOJ : year range for student’s date of joining like 1900 to 2018

For server-side “Database admin” will be performing manual updates for conflicting updates recording in Logs.

Loss In Internet connectivity :

From client side if internet connection goes down, update request will be saved and client will be able to see the pending update requests. Once the internet connection becomes available, client triggers the pending update requests.

Conflicting updates:

In case of conflicting updates, direct database will not be updated. Conflicting attributes update will be saved in LOG in server side and database admin can manually resolve conflicting updates.

Students Role:

1. Get student list respective of a roll number
2. Update mutable fields of students record accessible by student in database
3. Check pending update requests
4. Triggers Pending Update requests

Principal Role:

1. Get student list respective of a roll number
2. Update fields of students record accessible by principal in database
3. Check pending update request
4. Triggers Pending Update requests

Database Admin Role:

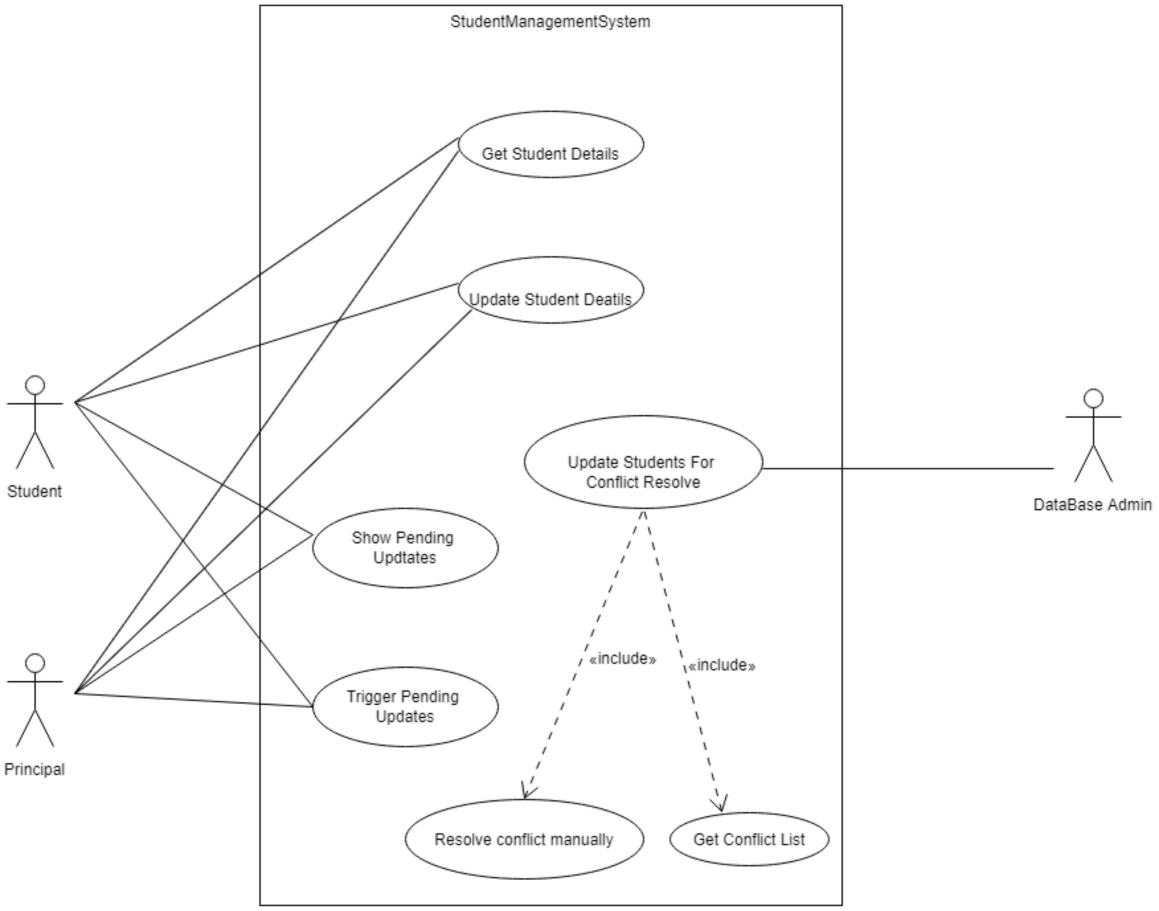
1. Check conflicting updates and corresponding conflict number and version number
2. Manually update conflicting updates

Compact Object Concept:

When ever client requests for update, based on it's "role", compact object will be transferred between client and server through network. Based on clients's role, only accessible fields will be transferred to client hence compact object concept is used. Hence less amount of data will be sent over network.

USE CASE DIAGRAM:

There are two roles: Student and Admin. They are allowed to perform operations on database based on their role.

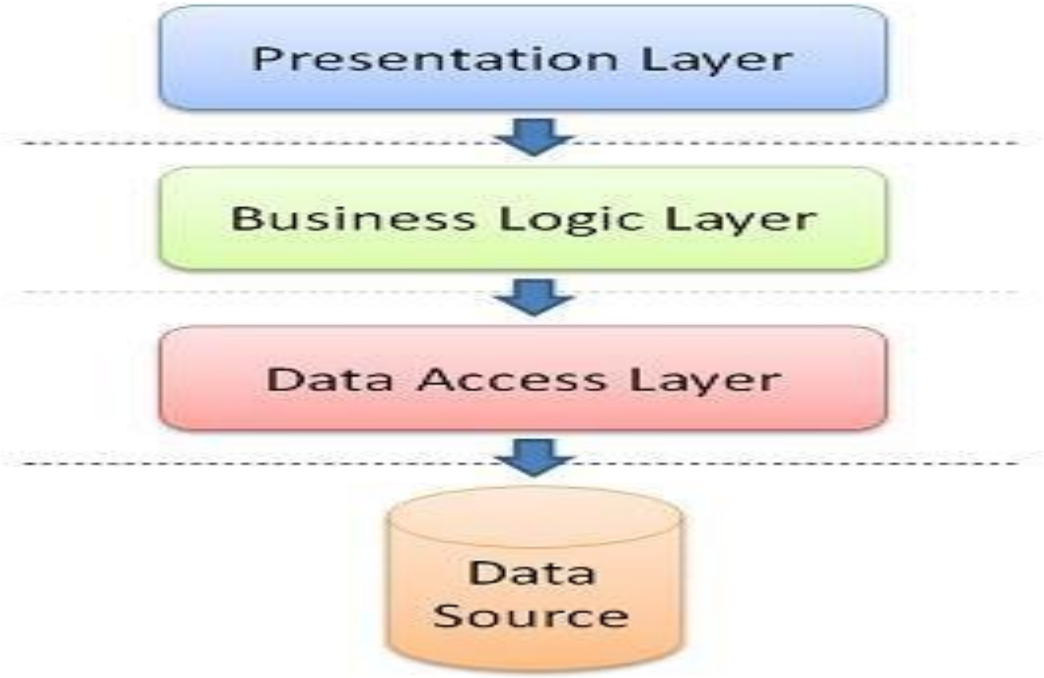


Architecture of model

Client and server side is implemented based on 3 tier architecture.

3 tier architecture: Implementation is divided into three sections.

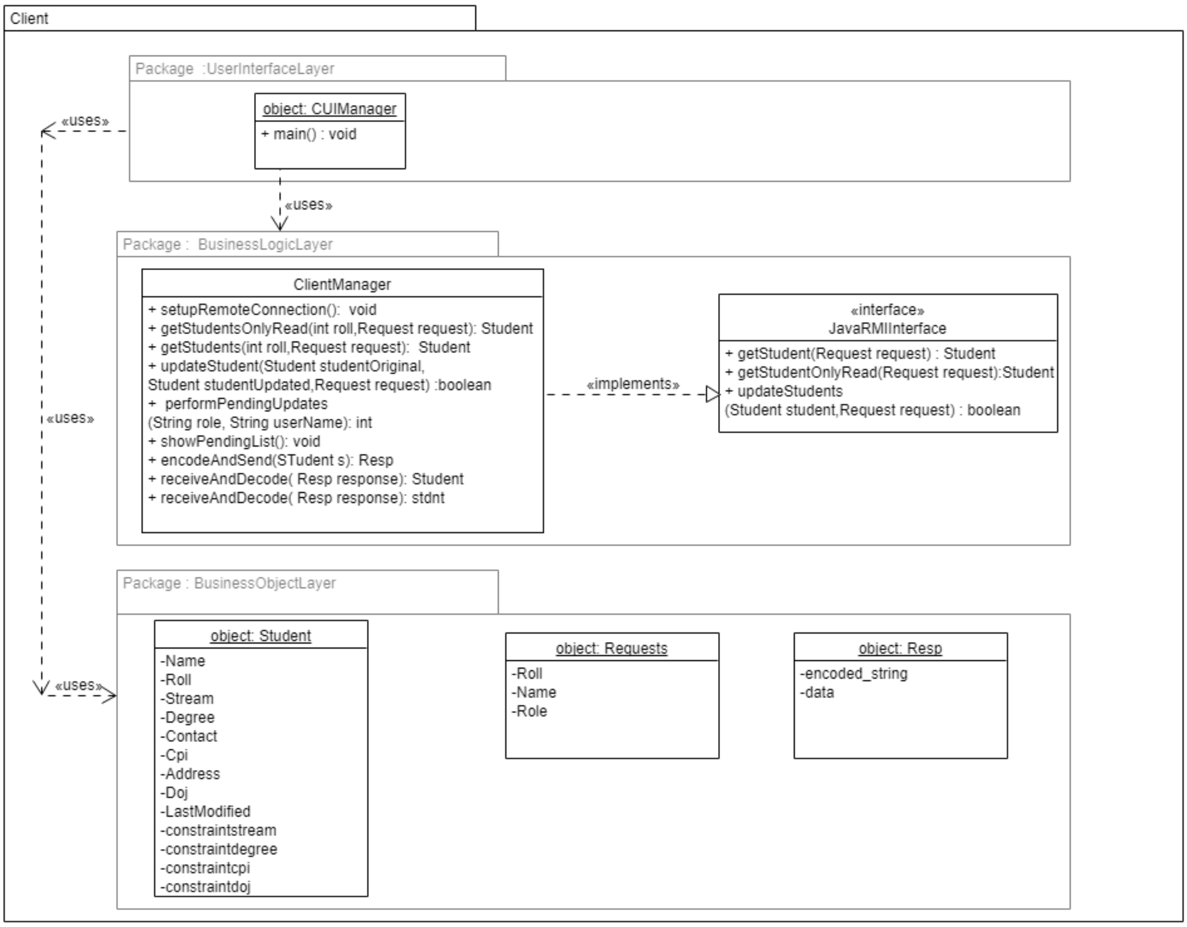
1. Presentation layer
2. Logic layer
3. Data layer



Client side architecture:

Client side is implemented according to three tier architecture which consists:

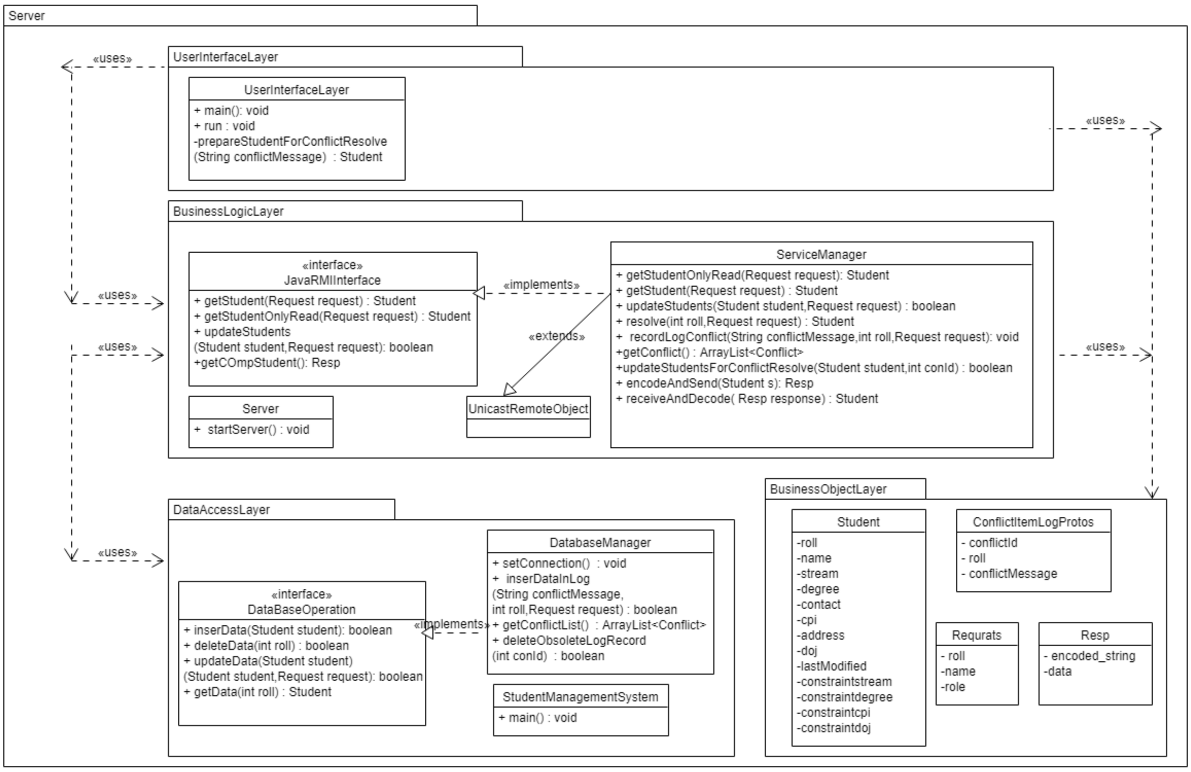
1. UserInterfaceLayer
2. BusinessLogicLayer
3. BusinessObjectLayer


Client
Package :UserInterfaceLayer
object: CUIManager
+ main() : void
«uses»
«uses»
Package : BusinessLogicLayer
ClientManager
+ setupRemoteConnection(): void
+ getStudentsOnlyRead(int roll,Request request): Student
+ getStudents(int roll,Request request): Student
+ updateStudent(Student studentOriginal,
Student studentUpdated,Request request) :boolean
+ performPendingUpdates
(String role, String userName): int
+ showPendingList(): void
+ encodeAndSend(STudent s): Resp
+ receiveAndDecode(Resp response): Student
+ receiveAndDecode(Resp response): stdnt
«implements»
«interface»
JavaRMIInterface
+ getStudent(Request request) : Student
+ getStudentOnlyRead(Request request):Student
+ updateStudents
(Student student,Request request) : boolean
«uses»
Package : BusinessObjectLayer
«uses»
object: Student
-Name
-Roll
-Stream
-Degree
-Contact
-Cpi
-Address
-Doj
-LastModified
-constraintstream
-constraintdegree
-constraintcpi
-constraintdoj
object: Requests
-Roll
-Name
-Role
object: Resp
-encoded_string
-data

Server side architecture:

Server side is consists of :

1. UserInterfaceLayer
2. BusinessLogicLayer
3. BusinessObjectLayer
4. DataAccessLayer


Server
«uses»
UserInterfaceLayer
UserInterfaceLayer
+ main(): void
+ run : void
-prepareStudentForConflictResolve
(String conflictMessage) : Student
«uses»
BusinessLogicLayer
«interface»
JavaRMIInterface
+ getStudent(Request request) : Student
+ getStudentOnlyRead(Request request) : Student
+ updateStudents
(Student student,Request request): boolean
+getCOmpStudent(): Resp
«implements»
«extends»
ServiceManager
+ getStudentOnlyRead(Request request): Student
+ getStudent(Request request) : Student
+ updateStudents(Student student,Request request) : boolean
+ resolve(int roll,Request request) : Student
+ recordLogConflict(String conflictMessage,int roll,Request request): void
+getConflict() : ArrayList<Conflict>
+updateStudentsForConflictResolve(Student student,int conId) : boolean
+ encodeAndSend(Student s): Resp
+ receiveAndDecode(Resp response) : Student
Server
+ startServer() : void
UnicastRemoteObject
«uses»
«uses»
«uses»
«uses»
DataAccessLayer
DatabaseManager
+ setConnection() : void
+ inserDataInLog
(String conflictMessage,
int roll,Request request) : boolean
+ getConflictList() : ArrayList<Conflict>
+ deleteObsoleteLogRecord
(int conId) : boolean
«interface»
DataBaseOperation
+ inserData(Student student): boolean
+ deleteData(int roll) : boolean
+ updateData(Student student)
(Student student,Request request): boolean
+ getData(int roll) : Student
«implements»
StudentManagementSystem
+ main() : void
BusinessObjectLayer
Student
-roll
-name
-stream
-degree
-contact
-cpi
-address
-doj
-lastModified
-constraintstream
-constraintdegree
-constraintcpi
-constraintdoj
ConflictItemLogProtos
- conflictId
- roll
- conflictMessage
Requrats
- roll
-name
-role
Resp
- encoded_string
-data

DB structure:

1. “StudentDatabase” schema having fields:
    a. ROLL_NUMBER
    b. NAME
    c. STREAM
    d. DEGREE
    e. CONTACT
    f. CPI
    g. ADDRESS
    h. JOIN YEAR
    i. LAST MODIFIED
2. “LOGS” schema having fields:
    a. CONFLICT_ID
    b. ROLL_NUMBER
    c. CONFLICT_MESSAGE
    d. TIMESTAMP
    e. UsersName
    f. Role

TOOLS USED:

Server Side:

1. NetBeans – java Development IDE
2. JDK 1.8 for programming
3. Linux- operating system
4. MySql Server – Database
5. Java RMI for distributed system communication

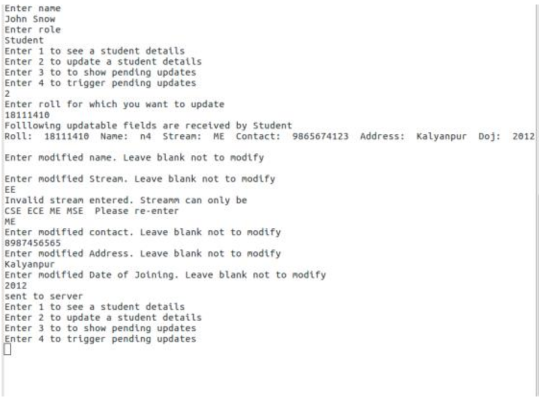
Client side:

1. NetBeans – java Development IDE
2. JDK 1.8 for programming
3. Linux- operating system
4. MySql Server – Database
5. Java RMI for distributed system communication

Output and Evaluation:

Test cases for “student”, “principal” as client and server admin as administrator is shown.

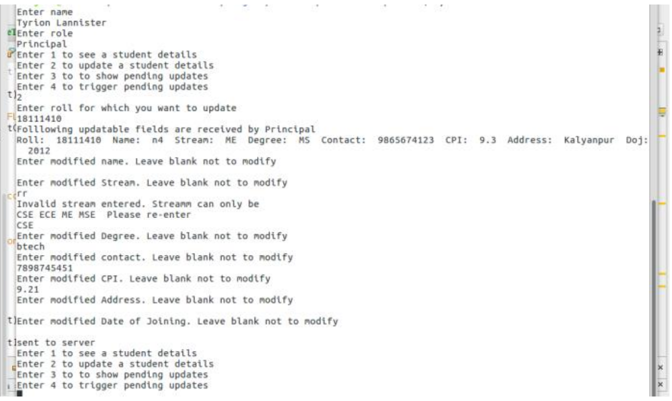
“Student” and “ principal” simultaneously sends requests to update record.



```
Enter name
John Snow
Enter role
Student
Enter 1 to see a student details
Enter 2 to update a student details
Enter 3 to to show pending updates
Enter 4 to trigger pending updates
2
Enter roll for which you want to update
18111410
Folllowing updatable fields are received by Student
Roll:  18111410  Name:  n4  Stream:  ME  Contact:  9865674123  Address:  Kalyanpur  Doj:  2012

Enter modified name. Leave blank not to modify

Enter modified Stream. Leave blank not to modify
EE
Invalid stream entered. Streamm can only be
CSE ECE ME MSE  Please re-enter
ME
Enter modified contact. Leave blank not to modify
8987456565
Enter modified Address. Leave blank not to modify
Kalyanpur
Enter modified Date of Joining. Leave blank not to modify
2012
sent to server
Enter 1 to see a student details
Enter 2 to update a student details
Enter 3 to to show pending updates
Enter 4 to trigger pending updates
```



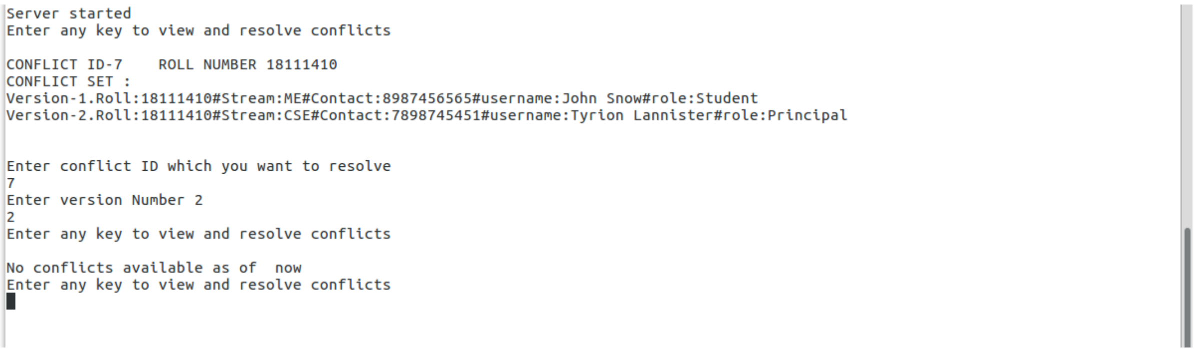
```
Enter name
Tyrion Lannister
Enter role
Principal
Enter 1 to see a student details
Enter 2 to update a student details
Enter 3 to to show pending updates
Enter 4 to trigger pending updates
2
Enter roll for which you want to update
18111410
Folllowing updatable fields are received by Principal
Roll:  18111410  Name:  n4  Stream:  ME  Degree:  MS  Contact:  9865674123  CPI:  9.3  Address:  Kalyanpur  Doj:
  2012
Enter modified name. Leave blank not to modify

Enter modified Stream. Leave blank not to modify
rr
Invalid stream entered. Streamm can only be
CSE ECE ME MSE  Please re-enter
CSE
Enter modified Degree. Leave blank not to modify
btech
Enter modified contact. Leave blank not to modify
7898745451
Enter modified CPI. Leave blank not to modify
9.21
Enter modified Address. Leave blank not to modify

Enter modified Date of Joining. Leave blank not to modify

sent to server
Enter 1 to see a student details
Enter 2 to update a student details
Enter 3 to to show pending updates
Enter 4 to trigger pending updates
```

Server admin checks for conflict availability and update conflict manually if conflict is present.



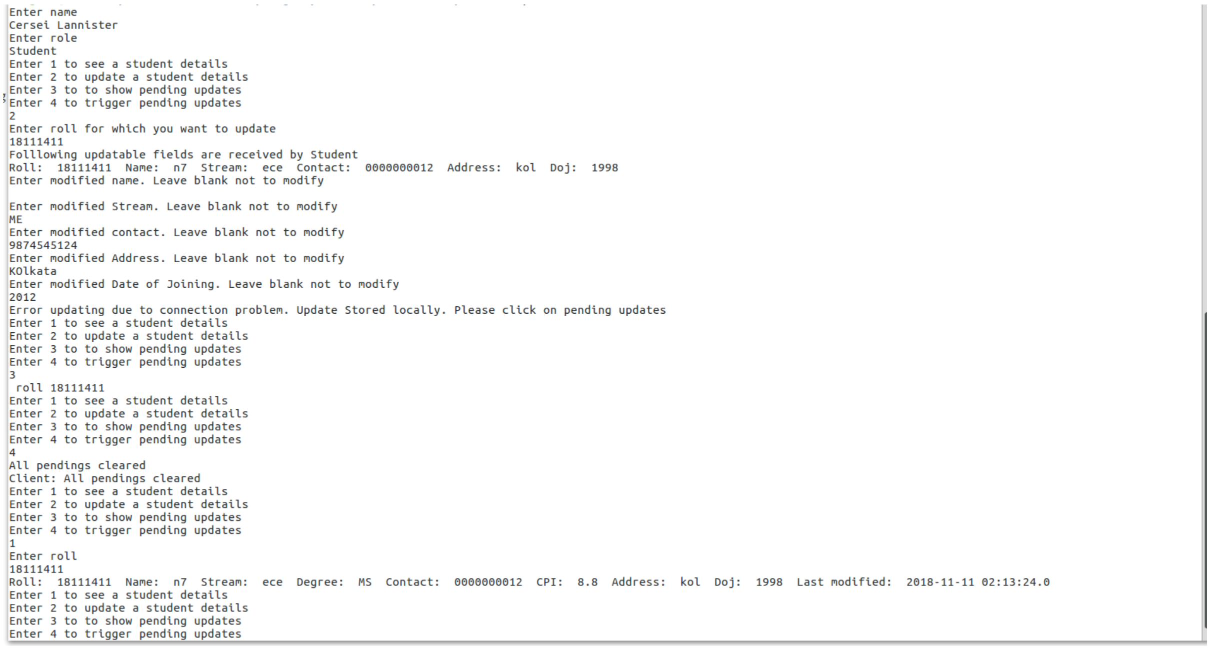
```
Server started
Enter any key to view and resolve conflicts

CONFLICT ID-7    ROLL NUMBER 18111410
CONFLICT SET :
Version-1.Roll:18111410#Stream:ME#Contact:8987456565#username:John Snow#role:Student
Version-2.Roll:18111410#Stream:CSE#Contact:7898745451#username:Tyrion Lannister#role:Principal


Enter conflict ID which you want to resolve
7
Enter version Number 2
2
Enter any key to view and resolve conflicts

No conflicts available as of  now
Enter any key to view and resolve conflicts
█
```

For client, when internet connection goes down, update requests will be saved and client will trigger the update requests once internet connection comes available



```
Enter name
Cersei Lannister
Enter role
Student
Enter 1 to see a student details
Enter 2 to update a student details
Enter 3 to to show pending updates
Enter 4 to trigger pending updates
2
Enter roll for which you want to update
18111411
Folllowing updatable fields are received by Student
Roll:  18111411  Name:  n7  Stream:  ece  Contact:  0000000012  Address:  kol  Doj:  1998
Enter modified name. Leave blank not to modify

Enter modified Stream. Leave blank not to modify
ME
Enter modified contact. Leave blank not to modify
9874545124
Enter modified Address. Leave blank not to modify
KOlkata
Enter modified Date of Joining. Leave blank not to modify
2012
Error updating due to connection problem. Update Stored locally. Please click on pending updates
Enter 1 to see a student details
Enter 2 to update a student details
Enter 3 to to show pending updates
Enter 4 to trigger pending updates
3
 roll 18111411
Enter 1 to see a student details
Enter 2 to update a student details
Enter 3 to to show pending updates
Enter 4 to trigger pending updates
4
All pendings cleared
Client: All pendings cleared
Enter 1 to see a student details
Enter 2 to update a student details
Enter 3 to to show pending updates
Enter 4 to trigger pending updates
1
Enter roll
18111411
Roll:  18111411  Name:  n7  Stream:  ece  Degree:  MS  Contact:  0000000012  CPI:  8.8  Address:  kol  Doj:  1998  Last modified:  2018-11-11 02:13:24.0
Enter 1 to see a student details
Enter 2 to update a student details
Enter 3 to to show pending updates
Enter 4 to trigger pending updates
```

Future Scope:

1. This application can be extended to a mobile application
2. There is a scope to make this application more robust to network connection failure
3. Database updates can be performed batch wise
4. Data base can be more sophisticated
5. Many other Client role can be implemented