# MOBILE LEARNING APPLICATION ANDROID VERSION

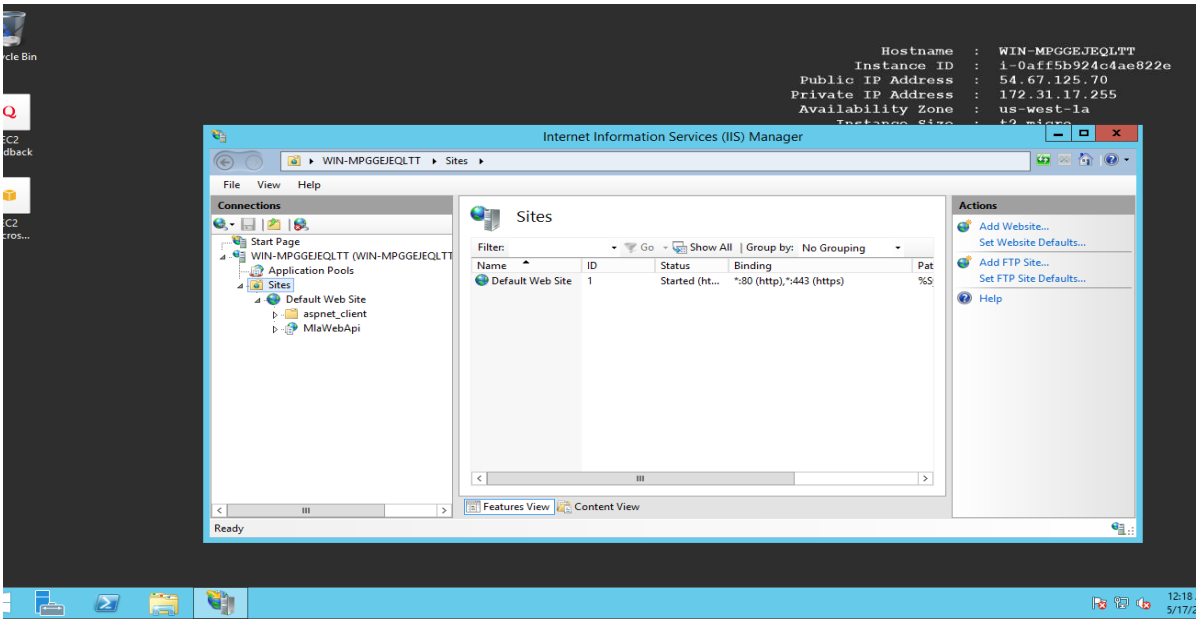
## REPORT

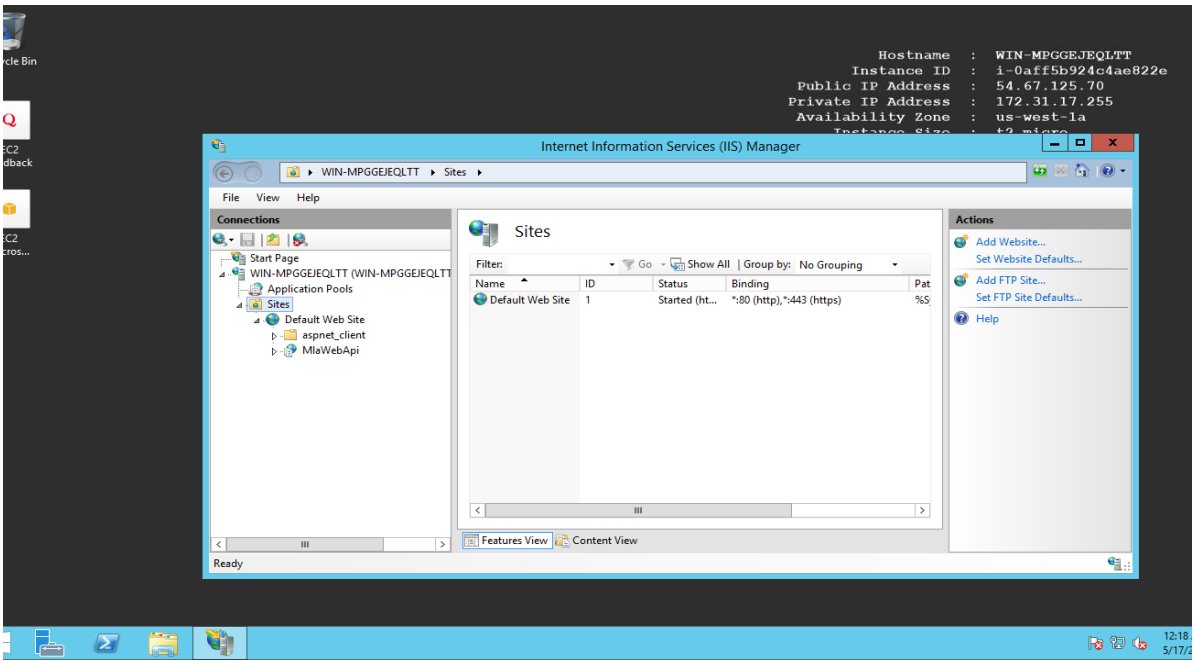
**By**

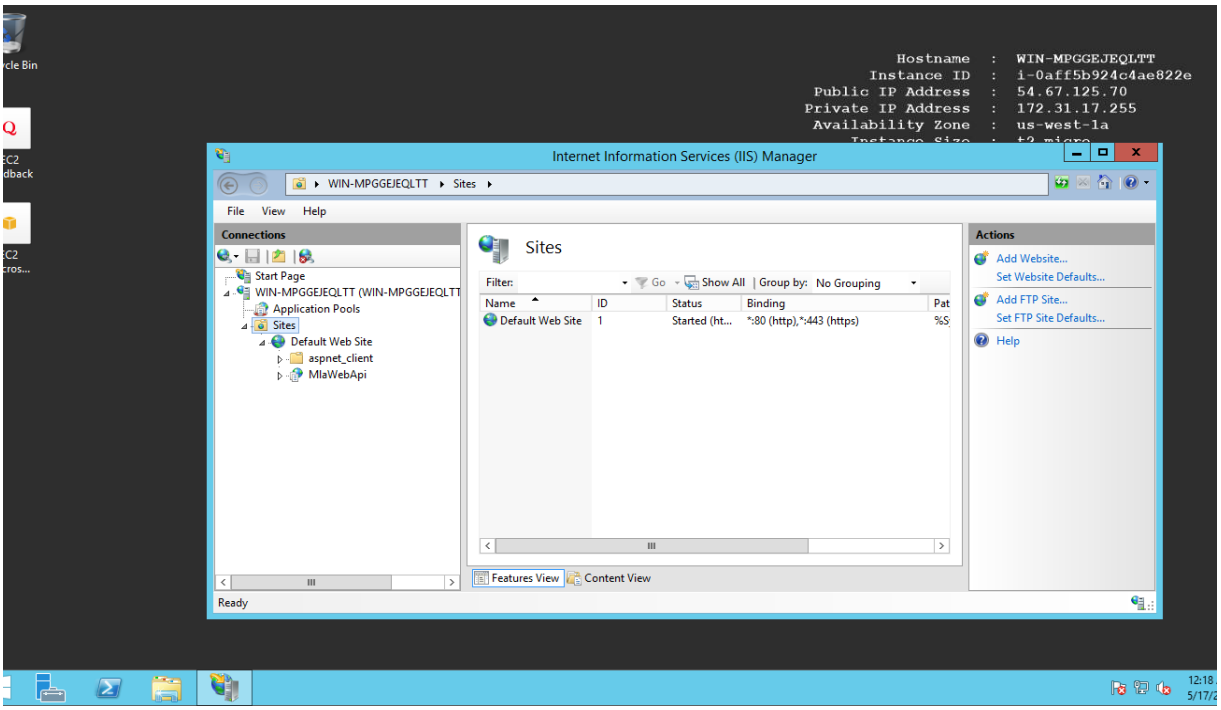
**Anand Masurkar**

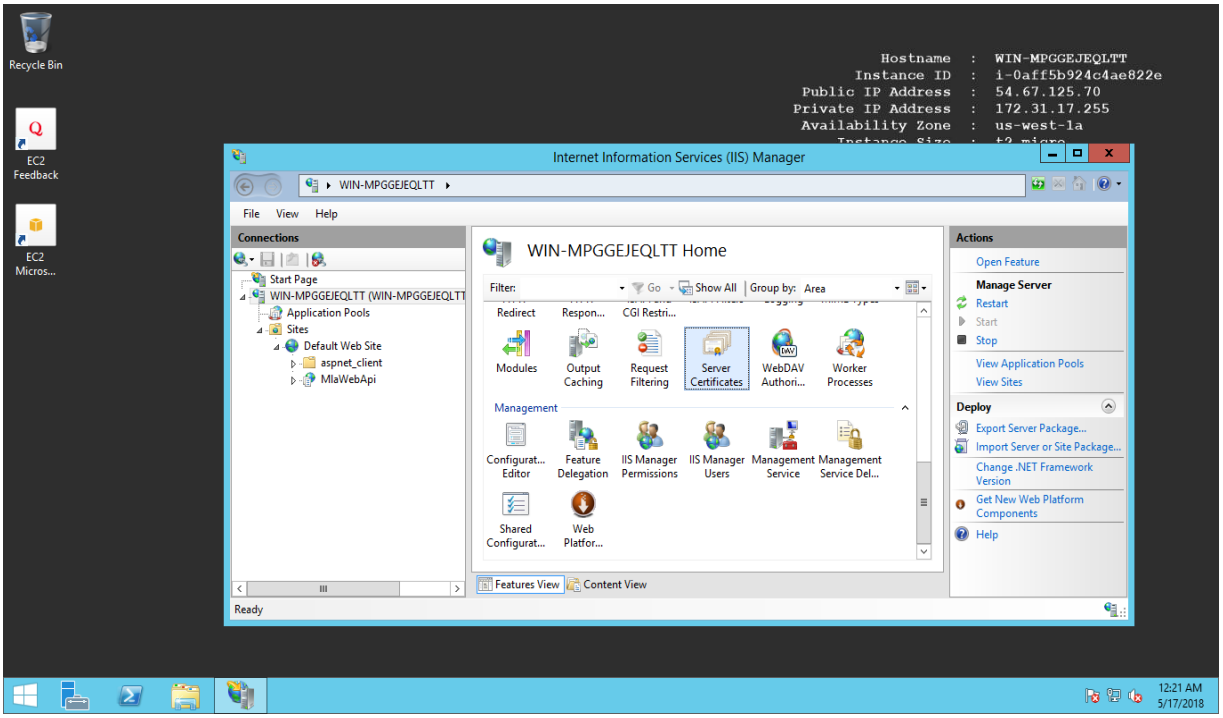
# INDEX

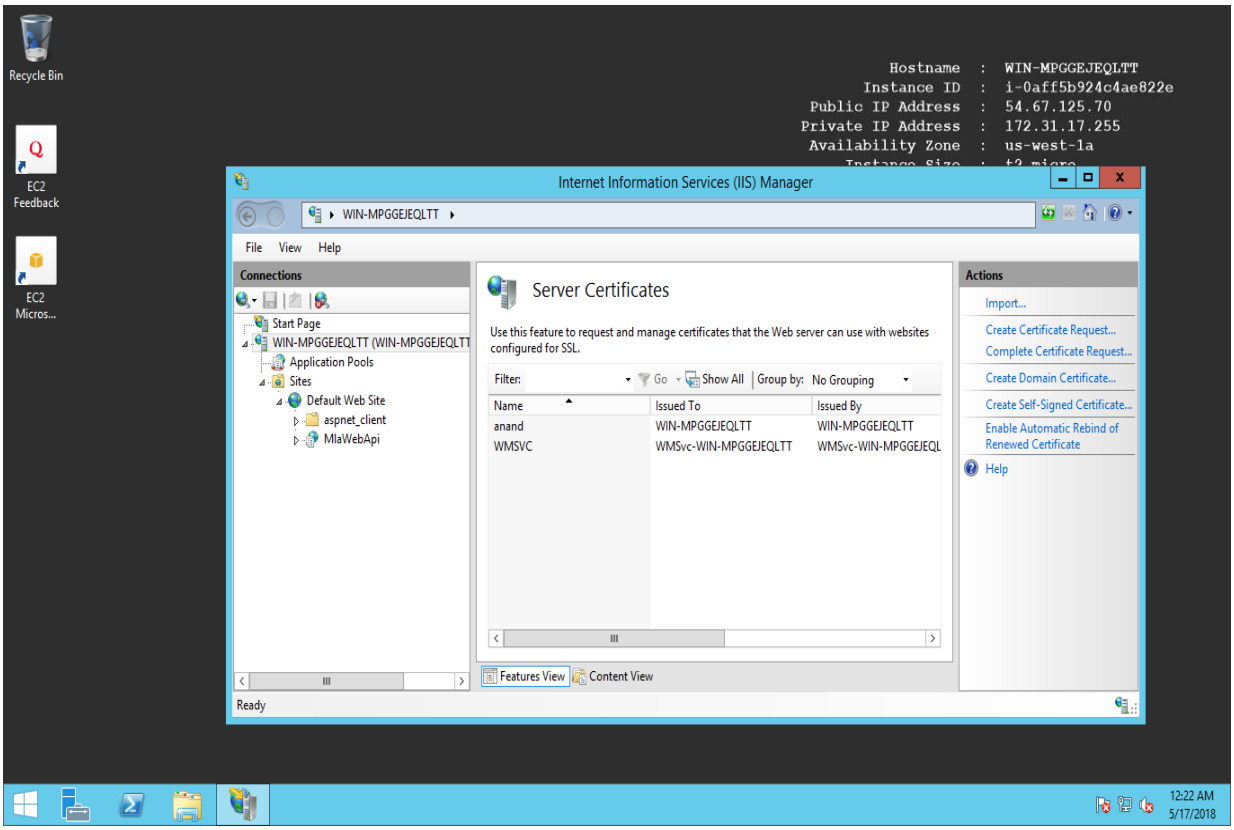
# I. Configuration of HTTPS to access site with a self-signed certificate

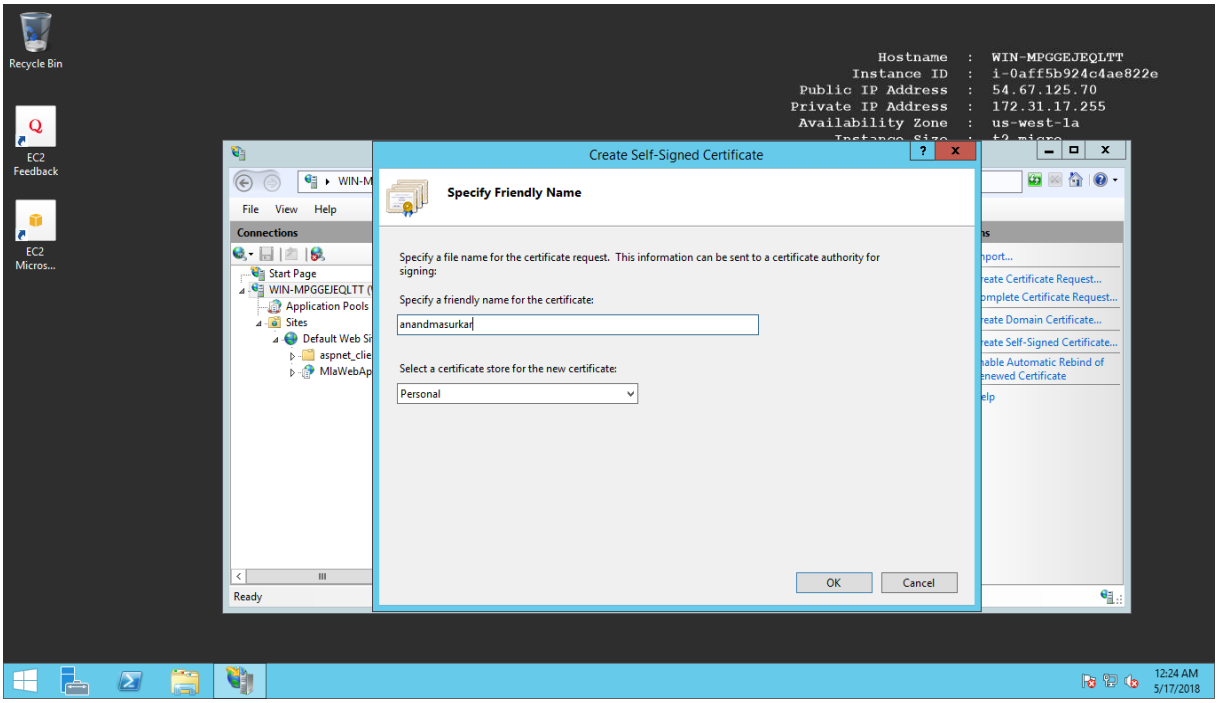


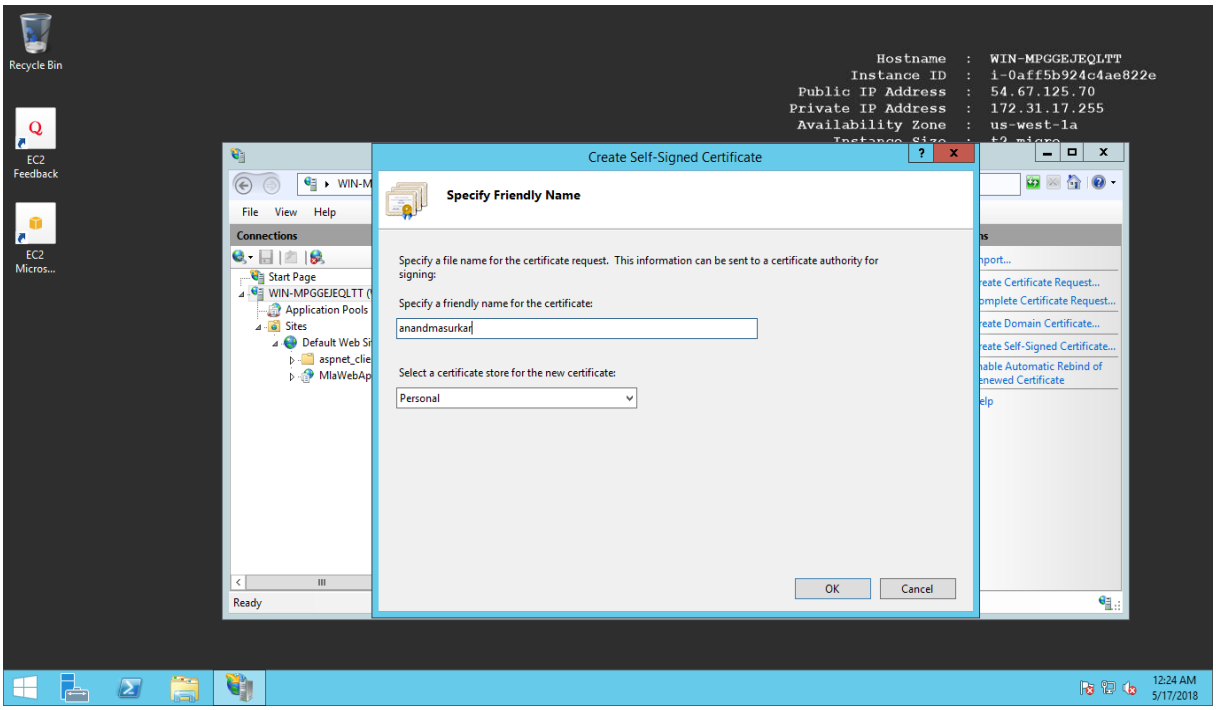

Hostname : WIN-MPGGEJEQLTT
Instance ID : i-0aff5b924c4ae822e
Public IP Address : 54.67.125.70
Private IP Address : 172.31.17.255
Availability Zone : us-west-1a
Internet Information Services (IIS) Manager
WIN-MPGGEJEQLTT ▸ Sites ▸
File View Help
Connections
Start Page
WIN-MPGGEJEQLTT (WIN-MPGGEJEQLTT
Application Pools
Sites
Default Web Site
aspnet_client
MlaWebApi
Sites
Filter:
Go
Show All
Group by: No Grouping
Name
ID
Status
Binding
Default Web Site
1
Started (ht...
*:80 (http),*:443 (https)
Features View
Content View
Actions
Add Website...
Set Website Defaults...
Add FTP Site...
Set FTP Site Defaults...
Help
Ready

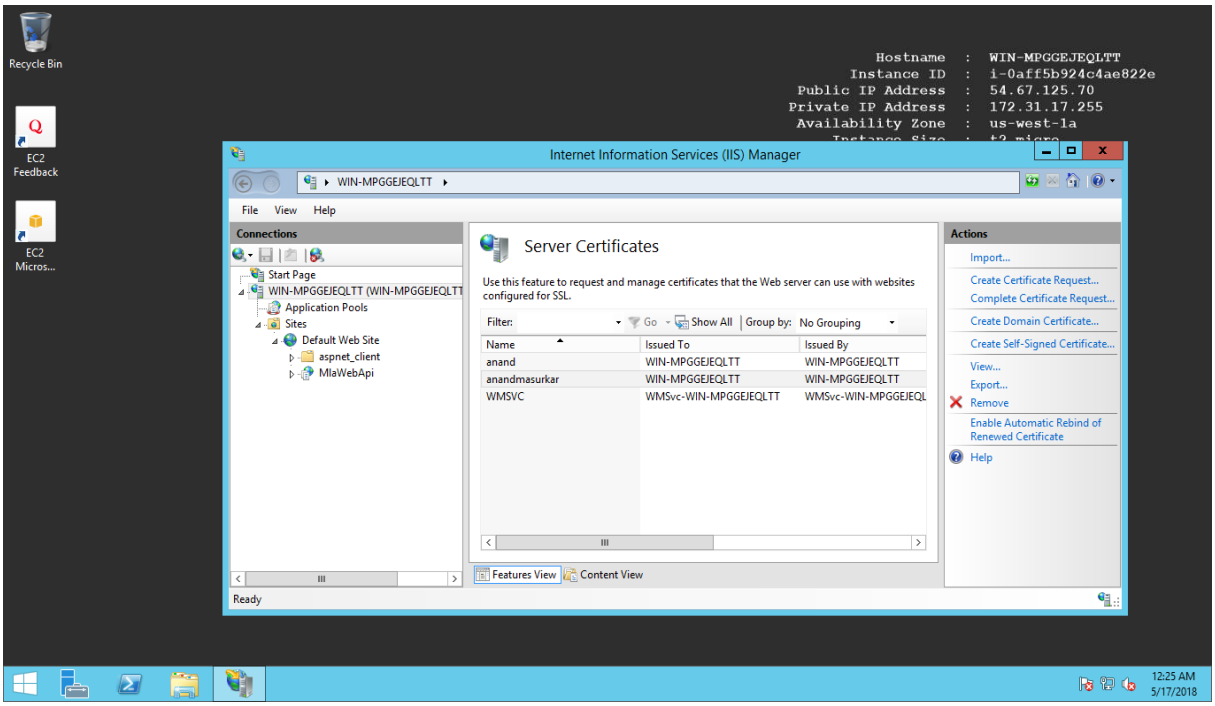

Hostname : WIN-MPGGEJEQLTT
Instance ID : i-0aff5b924c4ae822e
Public IP Address : 54.67.125.70
Private IP Address : 172.31.17.255
Availability Zone : us-west-1a
Internet Information Services (IIS) Manager
WIN-MPGGEJEQLTT ▸ Sites ▸
File View Help
Connections
Start Page
WIN-MPGGEJEQLTT (WIN-MPGGEJEQLTT
Application Pools
Sites
Default Web Site
aspnet_client
MlaWebApi
Sites
Filter: Go Show All Group by: No Grouping
Name ID Status Binding Pat
Default Web Site 1 Started (ht... *:80 (http),*:443 (https) %S
Features View Content View
Actions
Add Website...
Set Website Defaults...
Add FTP Site...
Set FTP Site Defaults...
Help
Ready

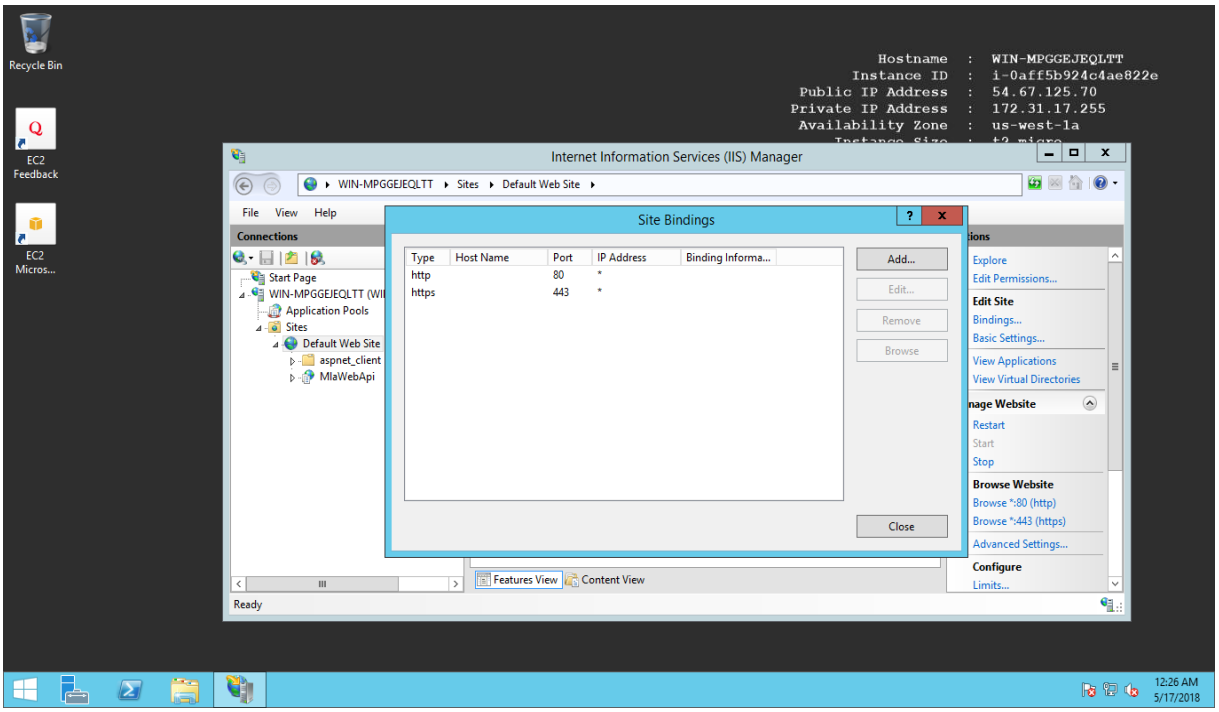

Recycle Bin
EC2 Feedback
EC2 Micros...
Hostname : WIN-MPGGEJEQLTT
Instance ID : i-0aff5b924c4ae822e
Public IP Address : 54.67.125.70
Private IP Address : 172.31.17.255
Availability Zone : us-west-1a
Internet Information Services (IIS) Manager
WIN-MPGGEJEQLTT ▸
File View Help
Connections
Start Page
WIN-MPGGEJEQLTT (WIN-MPGGEJEQLTT
Application Pools
Sites
Default Web Site
aspnet_client
MlaWebApi
WIN-MPGGEJEQLTT Home
Filter: Go Show All Group by: Area
Redirect
Respon...
CGI Restri...
Modules
Output Caching
Request Filtering
Server Certificates
WebDAV Authori...
Worker Processes
Management
Configurat... Editor
Feature Delegation
IIS Manager Permissions
IIS Manager Users
Management Service
Management Service Del...
Shared Configurat...
Web Platfor...
Features View Content View
Actions
Open Feature
Manage Server
Restart
Start
Stop
View Application Pools
View Sites
Deploy
Export Server Package...
Import Server or Site Package...
Change .NET Framework Version
Get New Web Platform Components
Help
Ready
12:21 AM
5/17/2018

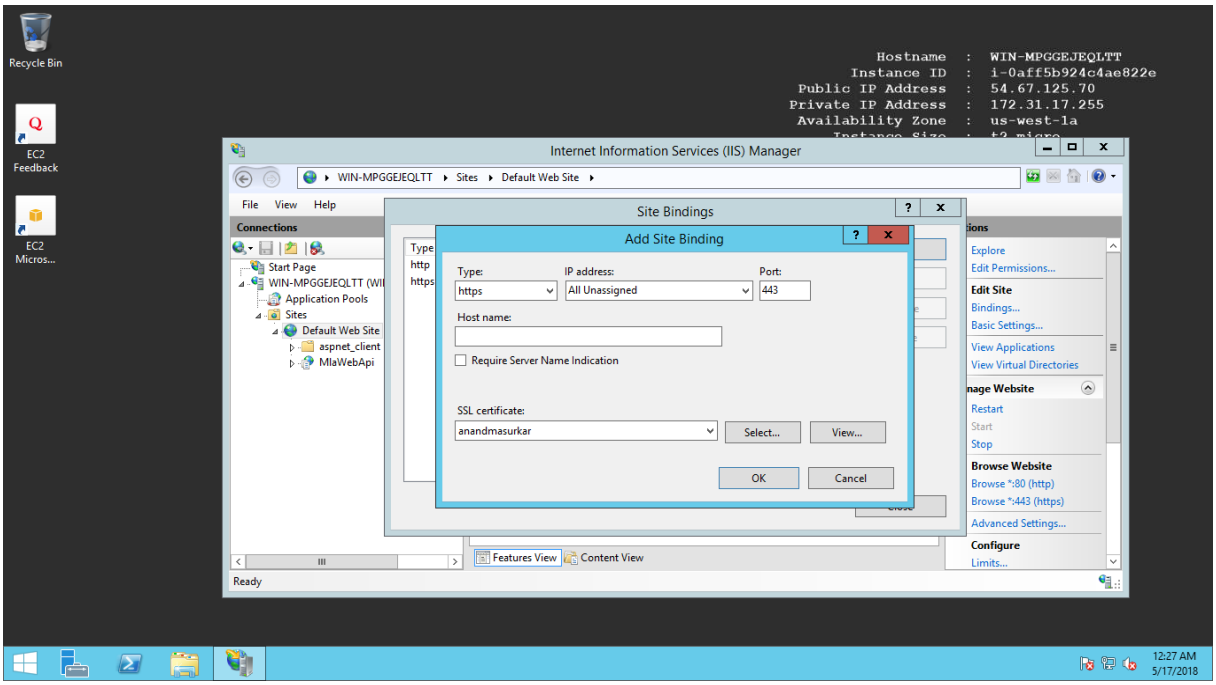

Hostname : WIN-MPGGEJEQLTT
Instance ID : i-0aff5b924c4ae822e
Public IP Address : 54.67.125.70
Private IP Address : 172.31.17.255
Availability Zone : us-west-1a
Internet Information Services (IIS) Manager
WIN-MPGGEJEQLTT
File View Help
Connections
Start Page
WIN-MPGGEJEQLTT (WIN-MPGGEJEQLTT
Application Pools
Sites
Default Web Site
aspnet_client
MlaWebApi
Server Certificates
Use this feature to request and manage certificates that the Web server can use with websites configured for SSL.
Filter:
Go
Show All
Group by: No Grouping
Name
Issued To
Issued By
anand
WIN-MPGGEJEQLTT
WIN-MPGGEJEQLTT
WMSVC
WMSvc-WIN-MPGGEJEQLTT
WMSvc-WIN-MPGGEJEQL
Actions
Import...
Create Certificate Request...
Complete Certificate Request...
Create Domain Certificate...
Create Self-Signed Certificate...
Enable Automatic Rebind of Renewed Certificate
Help
Features View
Content View
Ready
12:22 AM
5/17/2018


Hostname : WIN-MPGGEJEQLTT
Instance ID : i-0aff5b924c4ae822e
Public IP Address : 54.67.125.70
Private IP Address : 172.31.17.255
Availability Zone : us-west-1a
Create Self-Signed Certificate
Specify Friendly Name
Specify a file name for the certificate request. This information can be sent to a certificate authority for signing:
Specify a friendly name for the certificate:
anandmasurkar
Select a certificate store for the new certificate:
Personal
OK
Cancel
Ready
12:24 AM
5/17/2018


Hostname : WIN-MPGGEJEQLTT
Instance ID : i-0aff5b924c4ae822e
Public IP Address : 54.67.125.70
Private IP Address : 172.31.17.255
Availability Zone : us-west-1a
Create Self-Signed Certificate
Specify Friendly Name
Specify a file name for the certificate request. This information can be sent to a certificate authority for signing:
Specify a friendly name for the certificate:
anandmasurkar
Select a certificate store for the new certificate:
Personal
OK
Cancel
Ready
12:24 AM
5/17/2018


Hostname : WIN-MPGGEJEQLTT
Instance ID : i-0aff5b924c4ae822e
Public IP Address : 54.67.125.70
Private IP Address : 172.31.17.255
Availability Zone : us-west-1a
Internet Information Services (IIS) Manager
WIN-MPGGEJEQLTT
File View Help
Connections
Start Page
WIN-MPGGEJEQLTT (WIN-MPGGEJEQLTT
Application Pools
Sites
Default Web Site
aspnet_client
MlaWebApi
Server Certificates
Use this feature to request and manage certificates that the Web server can use with websites configured for SSL.
Filter:
Go
Show All
Group by: No Grouping
Name
Issued To
Issued By
anand
WIN-MPGGEJEQLTT
WIN-MPGGEJEQLTT
anandmasurkar
WIN-MPGGEJEQLTT
WIN-MPGGEJEQLTT
WMSVC
WMSvc-WIN-MPGGEJEQLTT
WMSvc-WIN-MPGGEJEQL
Actions
Import...
Create Certificate Request...
Complete Certificate Request...
Create Domain Certificate...
Create Self-Signed Certificate...
View...
Export...
Remove
Enable Automatic Rebind of Renewed Certificate
Help
Features View
Content View
Ready
12:25 AM
5/17/2018


Hostname : WIN-MPGGEJEQLTT
Instance ID : i-0aff5b924c4ae822e
Public IP Address : 54.67.125.70
Private IP Address : 172.31.17.255
Availability Zone : us-west-1a
Internet Information Services (IIS) Manager
Site Bindings
Type Host Name Port IP Address Binding Informa...
http 80 *
https 443 *
Add...
Edit...
Remove
Browse
Close
Ready
12:26 AM
5/17/2018


Hostname : WIN-MPGGEJEQLTT
Instance ID : i-0aff5b924c4ae822e
Public IP Address : 54.67.125.70
Private IP Address : 172.31.17.255
Availability Zone : us-west-1a
Internet Information Services (IIS) Manager
Site Bindings
Add Site Binding
Type: https
IP address: All Unassigned
Port: 443
Host name:
Require Server Name Indication
SSL certificate: anandmasurkar
Select...
View...
OK
Cancel
Ready
12:27 AM
5/17/2018

## In Android Studio Go to API.java

```
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;


public class Api {

  private static Retrofit retrofit = null;

  public static APIInterface getClient() {

     HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();
     interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
     //OkHttpClient client = new OkHttpClient.Builder().addInterceptor(interceptor).build();
     OkHttpClient client = UnsafeOkHttpClient.getUnsafeOkHttpClient();
     if (retrofit == null) {
        retrofit = new Retrofit.Builder()
              .baseUrl(CommonUtils.MlaBaseURL)
              .addConverterFactory(GsonConverterFactory.create())
              .client(client)
              .build();
     }

     return retrofit.create(APIInterface.class);
  }


}
```

# II. Transactions in MLA mobile learning application

### a) Need of transaction in Mla application.

If you supposed to insert data into master and child tables in the SQL Server

**Case 1:** During the above task you inserted one row into master table and then some error occurred then what would happen to that task? Because of the error child data is not inserted you must rollback the master table data also, otherwise data inconsistency will cause

**Case 2:** For suppose you wish to delete record from a parent/super/master table then you must delete data from child first and then from master tables. So, both statements must be considered as a single unit. At that time transactions will be useful. Such situations arise in MLA application in cases like registration of new users i.e. Instructor, Student or admin.

### b) Understanding current application scenarios to keep database consistent and need of transactions in current application.

While performing insertion of the new user we register the user with its credentials like username and save its user type (to identify who is that person) and its userID. This user id is then inserted as a foreign key in the table matching its usertype. Username is unique for tables admin given by idAdmin, Instructor by idInstructor and student by idstudent.
Now the interesting situation arises,
**1st -** each user type i.e. admin, instructor and student can have same username but no two admins or instructor or student can share same username as it's a primary key in respective tables.
**2nd-** Now if this is the case register table can hold same usernames and we cannot keep them as primary key because username of admin can be same as the username of the Instructor or student. Therefore if a new user arises and registers using same username its valid but corresponding to its specific

usertype table we cannot hold it as it violates constraint of primary key. To avoid this problem, we use transactions that is either insert in both tables or nothing at all and reply to the user as username already taken. Assign transaction object to sql queries that need to execute as a whole or nothing at all. Commit transaction after executing multiple queries assigned with single transaction. This needs to be done in try block and in catch block we need to rollback if any exception is raised.

**c) Understanding underlying concepts of transaction read committed isolation level to understand the behavior of transactions in Ml application.**

In read committed Isolation level, we read only committed data. And this is important. As by looking at an example like if instructor added in the register table but not in Instructor table then it will roll back but in meantime admin can see a new added user and try to add other details for the instructor course. So, to avoid this we don't allow uncommitted data to be seen or read once the transaction starts.

Read committed is the default isolation level of sql

| Isolation Level | Dirty Read | Non-Repeatable Read | Phantom |
|---|---|---|---|
| **Read committed** | **No** | **Yes** | **Yes** |

**d) How these transactions are implemented in the actual C# code in visual studio.**

```
DataSet dsData = new DataSet("tasks");
cnn = new SqlConnection(cfmgr);
cnn.Open();
```

```
transaction = cnn.BeginTransaction();

SqlCommand comm5 = new SqlCommand("DELETE FROM student_tasks
where subject_id ='" + subject_id + "'", cnn, transaction);
SqlDataAdapter Sqlda5 = new SqlDataAdapter(comm5);
dsData = new DataSet();
Sqlda5.Fill(dsData);

SqlCommand comm6 = new SqlCommand("DELETE FROM tasks where
subject_id ='" + subject_id + "'", cnn, transaction);
SqlDataAdapter Sqlda6 = new SqlDataAdapter(comm6);
dsData = new DataSet();
Sqlda6.Fill(dsData);

SqlCommand comm8 = new SqlCommand("DELETE FROM subject_roster
where subject_id='" + subject_id + "'", cnn, transaction);
SqlDataAdapter Sqlda8 = new SqlDataAdapter(comm8);
dsData = new DataSet();
Sqlda8.Fill(dsData);

SqlCommand comm7 = new SqlCommand("DELETE FROM subject where
idSubject ='" + subject_id + "'", cnn, transaction);
SqlDataAdapter Sqlda7 = new SqlDataAdapter(comm7);
dsData = new DataSet();
Sqlda7.Fill(dsData);

var response =
Request.CreateResponse<Tasks>(System.Net.HttpStatusCode.Found, null);
transaction.Commit();
cnn.Close();
return response;
}
catch (SqlException ex)
```

```
{
transaction.Rollback();
var response =
Request.CreateResponse<Tasks>(System.Net.HttpStatusCode.BadRequest,
null);
cnn.Close();
return response;
}
```

# III. Deadlock handling in concurrent transactions.

**a) What are deadlocks, why deadlocks occur and how to visualize them in the application.**

A deadlock occurs when 2 processes are competing for exclusive access to a resource but is unable to obtain exclusive access to it, because the other process is preventing it. This results in a standoff where neither process can proceed. The only way out of a deadlock is for one of the processes to be terminated. SQL Server automatically detects when deadlocks have occurred and takes action by killing one of the processes known as the victims.

**b) Handling deadlock by detecting them and then recovering from it.**

There are two ways to handle deadlock

1. Deadlock prevention
2. Deadlock detection and recovery

In Mla application we will use Deadlock detection and recovery mechanism. Now to understand how we can detect deadlock we need to understand how sql server manages such situations. Typically, when deadlock occurs sql server throws an exception saying deadlock has occurred and need to retry that transaction. Taking advantage of this feature we can detect deadlock occurrence in our application. The exception thrown has deadlock error number associated with it, so this way we can detect that deadlock has occurred. Now to recover from it we need to retry transaction. In current Mla application developed I have used mechanism to again start the transaction. It tries to run the transaction for 2 more times in case it again gets into deadlock, but such a situation is rare but possible. After 2 attempts the transaction fails if it's still in deadlock and error msg is displayed.

**c) Demonstrating deadlock of current running application in sql server management studio.**

**1. In sql server management studio.**

Run the following sql commands in transactions A and B in the specified numbering order. It demonstrates possible scenario which can occur in mla application.

**Transaction A:**

```
Use mydb;

BEGIN TRANSACTION

insert into register values('q1wweft',1234567,'instructor'); -- 1
select * from register;                                      -- 3

COMMIT TRANSACTION
```

**Transaction B:**

```
Use mydb;

BEGIN TRANSACTION

insert into register values('q1wweft',1234567,'instructor'); -- 2
select * from register;                                      -- 4

COMMIT TRANSACTION
```

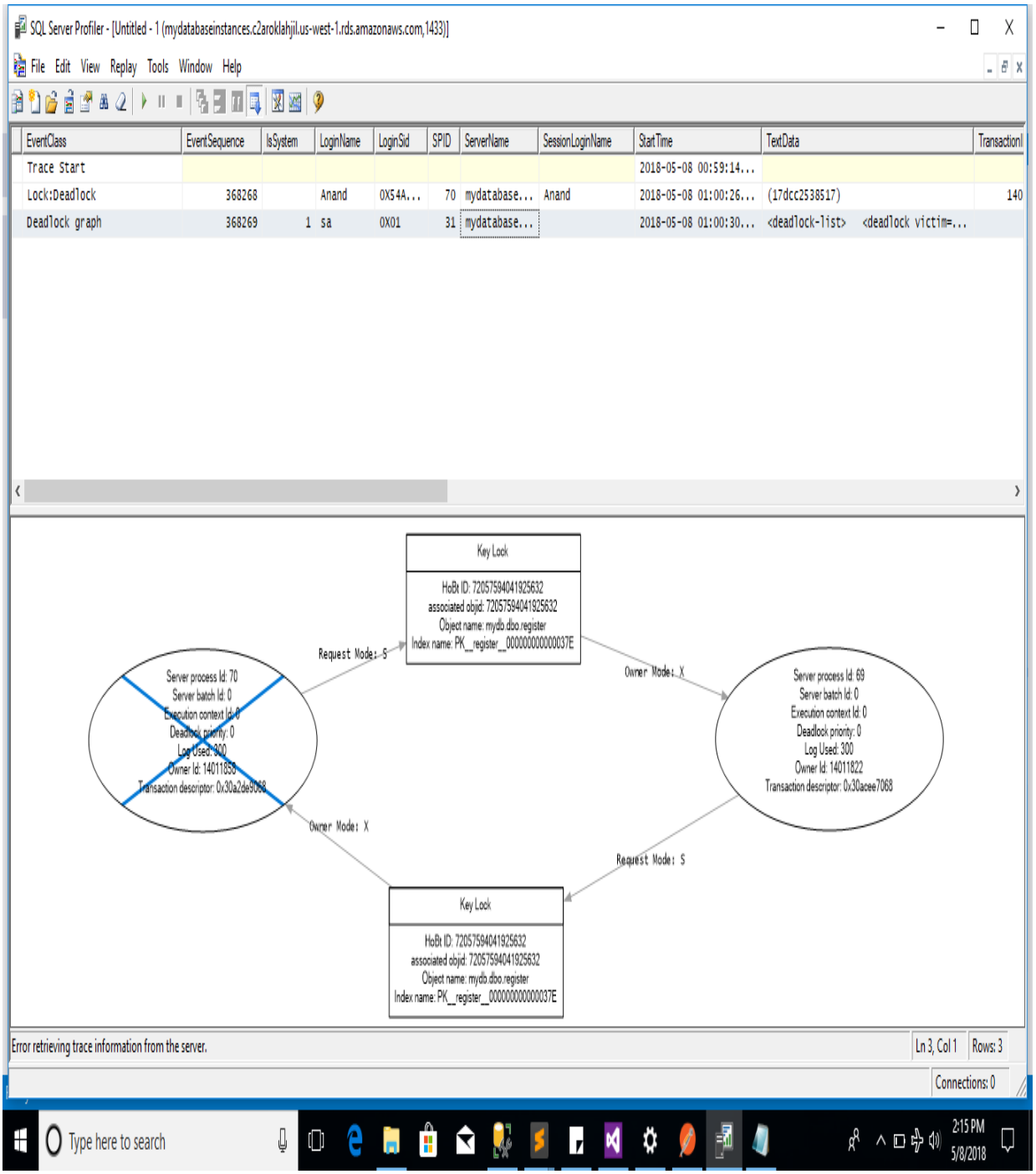
If the order of the instructions is followed by two concurrent applications, then the system will experience deadlock situation.
Output as per sql server management studio

**Transaction A**

Msg 1205, Level 13, State 51, Line 6
Transaction (Process ID 73) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.

**Transaction B**

| | | | |
|---|---|---|---|
| Admin | password | admin | 73 |
| anand1 | 123456 | admin | 80 |
| anand2 | 123456 | admin | 81 |
| qw1eseft | 1234567 | instructor | 316 |


SQL Server Profiler - [Untitled - 1 (mydatabaseinstances.c2aroklahjil.us-west-1.rds.amazonaws.com,1433)]
File Edit View Replay Tools Window Help
EventClass EventSequence IsSystem LoginName LoginSid SPID ServerName SessionLoginName StartTime TextData TransactionI
Trace Start 2018-05-08 00:59:14...
Lock:Deadlock 368268 Anand 0X54A... 70 mydatabase... Anand 2018-05-08 01:00:26... (17dcc2538517) 140
Deadlock graph 368269 1 sa 0X01 31 mydatabase... 2018-05-08 01:00:30... <deadlock-list> <deadlock victim=...
Key Lock
HoBt ID: 72057594041925632
associated objid: 72057594041925632
Object name: mydb.dbo.register
Index name: PK__register__000000000000037E
Request Mode: S
Owner Mode: X
Server process Id: 70
Server batch Id: 0
Execution context Id: 0
Deadlock priority: 0
Log Used: 300
Owner Id: 14011858
Transaction descriptor: 0x30a2de9068
Server process Id: 69
Server batch Id: 0
Execution context Id: 0
Deadlock priority: 0
Log Used: 300
Owner Id: 14011822
Transaction descriptor: 0x30acee7068
Owner Mode: X
Request Mode: S
Key Lock
HoBt ID: 72057594041925632
associated objid: 72057594041925632
Object name: mydb.dbo.register
Index name: PK__register__000000000000037E
Error retrieving trace information from the server.
Ln 3, Col 1
Rows: 3
Connections: 0
Type here to search
2:15 PM
5/8/2018

**d) Debugging to find more details about blocking query**
**Script to identify Blocking query**

```
SELECT
db.name mydb,
tl.request_session_id,
wt.blocking_session_id,
OBJECT_NAME(p.OBJECT_ID) BlockedObjectName,
tl.resource_type,
h1.TEXT AS RequestingText,
h2.TEXT AS BlockingTest,
tl.request_mode
FROM sys.dm_tran_locks AS tl
INNER JOIN sys.databases db ON db.database_id = tl.resource_database_id
INNER JOIN sys.dm_os_waiting_tasks AS wt ON tl.lock_owner_address =
wt.resource_address
INNER JOIN sys.partitions AS p ON p.hobt_id = tl.resource_associated_entity_id
INNER JOIN sys.dm_exec_connections ec1 ON ec1.session_id = tl.request_session_id
INNER JOIN sys.dm_exec_connections ec2 ON ec2.session_id =
wt.blocking_session_id
CROSS APPLY sys.dm_exec_sql_text(ec1.most_recent_sql_handle) AS h1
CROSS APPLY sys.dm_exec_sql_text(ec2.most_recent_sql_handle) AS h2
```

**e) How deadlocks are detected and recovered from them is implemented in the actual C# code in visual studio.**

```
public HttpResponseMessage PostAddInstructor(string instUserName, string
instPassword, string instFirsName, string instLastName, string
instTelephone, string instAddress, string instAliasMailId, string instEmailId,
string instSkypeId)
{
int retrycounter = 1;

SqlTransaction transaction = null;
string userType = "instructor"; // userType = instructor or student or admin
```

```
int userId = 0;
RETRY:
try{
DataSet dsData = new DataSet("register");
cnn = new SqlConnection(cfmgr);
cnn.Open();

transaction = cnn.BeginTransaction();
//first add to register table then to the instructor table.
SqlCommand comm = new SqlCommand("Insert int
register(userName,password,userType) values('"
+ instUserName
+ "','" + instPassword
+ "','" + userType
+ "')", cnn,transaction);
SqlDataAdapter sqlada = new SqlDataAdapter(comm);
sqlada.Fill(dsData);

]// retrive the userId since it is auto incremented in the database and need to
be added to the instructor table

System.Threading.Thread.Sleep(10000);
comm = new SqlCommand("select userId from register where userName = '"
+ instUserName + "'",cnn,transaction);
comm.Transaction = transaction;
sqlada = new SqlDataAdapter(comm);
sqlada.Fill(dsData);
System.Threading.Thread.Sleep(10000);

foreach (DataRow row in dsData.Tables[0].Rows)
{
   userId = Int16.Parse(Convert.ToString(row["userId"]));
}
```

```
 Instructor inst = new Instructor();
 inst.idInstructor = instUserName;
 inst.firstName = instFirsName;
 inst.lastName = instLastName;
 inst.userId = userId;
 inst.telephone = instTelephone;
 inst.address = instAddress;
 inst.aliasMailId = instAliasMailId;
 inst.emailId = instEmailId;
 inst.skypeId = instSkypeId;
 // now add to instructor table.

comm = new SqlCommand("Insert into
instructor(idInstructor,firstName,lastName,userId,telephone,address,aliasMa
ilId,emailId, skypeId) values('"
        + inst.idInstructor
        + "','" + inst.firstName
        + "','" + inst.lastName
        + "','" + inst.userId
        + "','" + inst.telephone
        + "','" + inst.address
        + "','" + inst.aliasMailId
        + "','" + inst.emailId
        + "','" + inst.skypeId
        + "')", cnn,transaction);

sqlada = new SqlDataAdapter(comm);
sqlada.Fill(dsData);
transaction.Commit();
cnn.Close();
```

```
var response =
Request.CreateResponse<Instructor>(System.Net.HttpStatusCode.Created,
inst);
return response;}
catch (SqlException ex)
{
transaction.Rollback();
Instructor inst = new Instructor();
var ErrorMessage = "Deadlock error";
int doRetry = 0;

if(ex.Number == 1205) //-- Deadlock Error Number
  {
    doRetry = 1; //-- Set @doRetry to 1 only for Deadlock
  }
if( doRetry ==1)
  {
    retrycounter = retrycounter + 1 ;//-- Increment Retry Counter By one
    if(retrycounter > 2) //-- Check whether Retry Counter reached to 3
    {
      //error even after 3 try
       var response =
Request.CreateResponse<Instructor>(System.Net.HttpStatusCode.BadRequ
est, inst);
        cnn.Close();
        return response;
    }
    else{
          //goto RETRY;
           var response = "Deadlock handling";
           cnn.Close();//-- Go to Label RETRY
```

```
 var response1 =
Request.CreateResponse(System.Net.HttpStatusCode.BadRequest,
"Deadlock handling");
                return response1;

     }
 }
  else {
            //errormessage
            var response =
Request.CreateResponse<Instructor>(System.Net.HttpStatusCode.BadRequ
est, inst);
            cnn.Close();
            return response;
            }
            }
          }
```