**CSC 258 - Distributed Systems**

**Spring 2019**



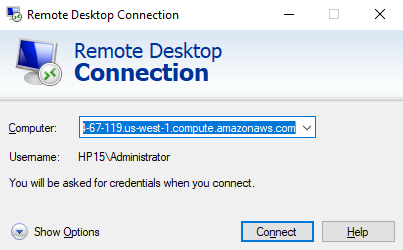
**MOBILE LEARNING APPLICATION**

**Masum Shah 220269466**

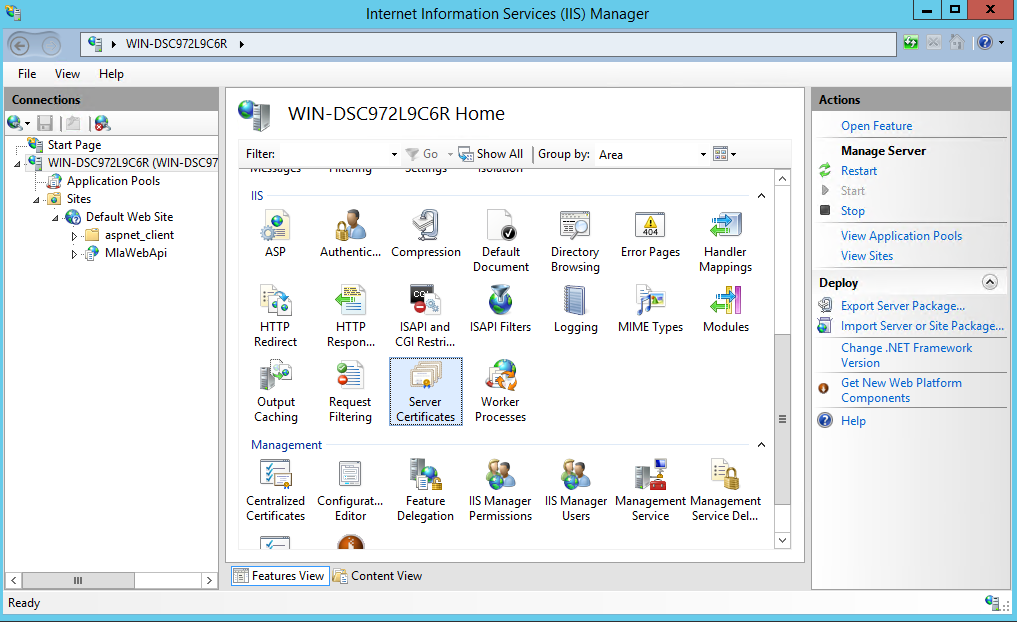
**HTTP to HTTPS: -**

**Generate a Self-Signed Certificate on IIS (Internet Information Services)**

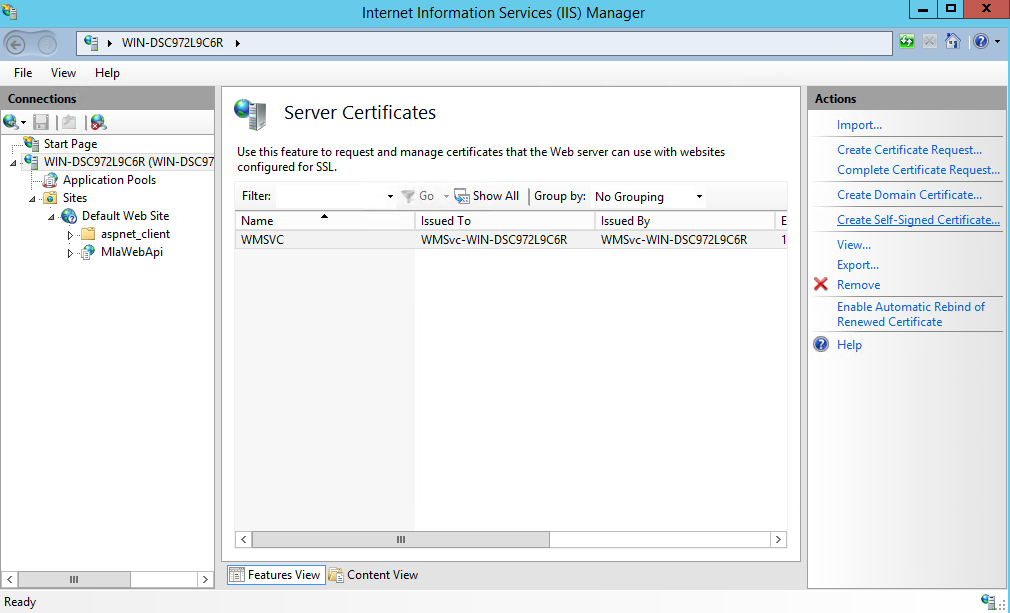
1. Open remote desktop connection enter the EC2 Public DNS with username and password.



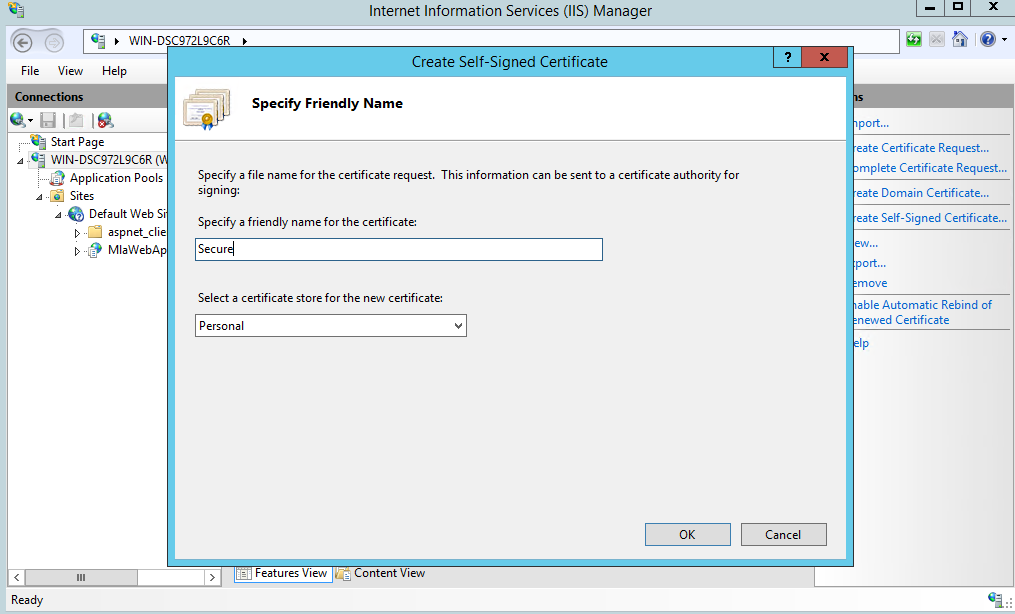
1. Open the Server Manger → Tools → IIS →Under “connections” →Select a root option
2. Then under IIS select server certificates.



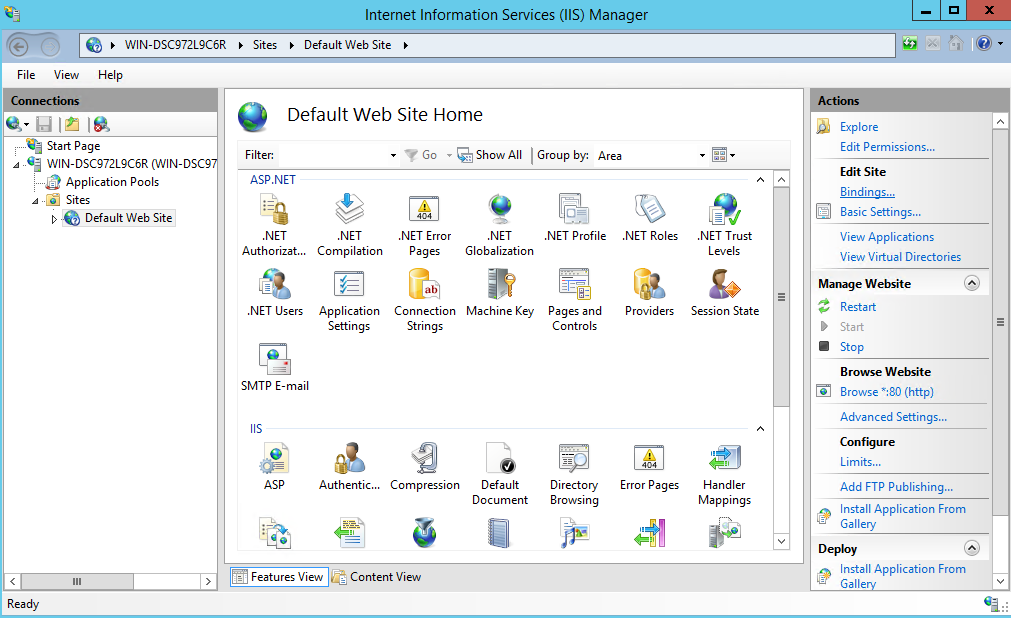
1. After selecting “Server Certificates”. Select “Create Self-Signed Certificate” under Actions in right side of the panel.



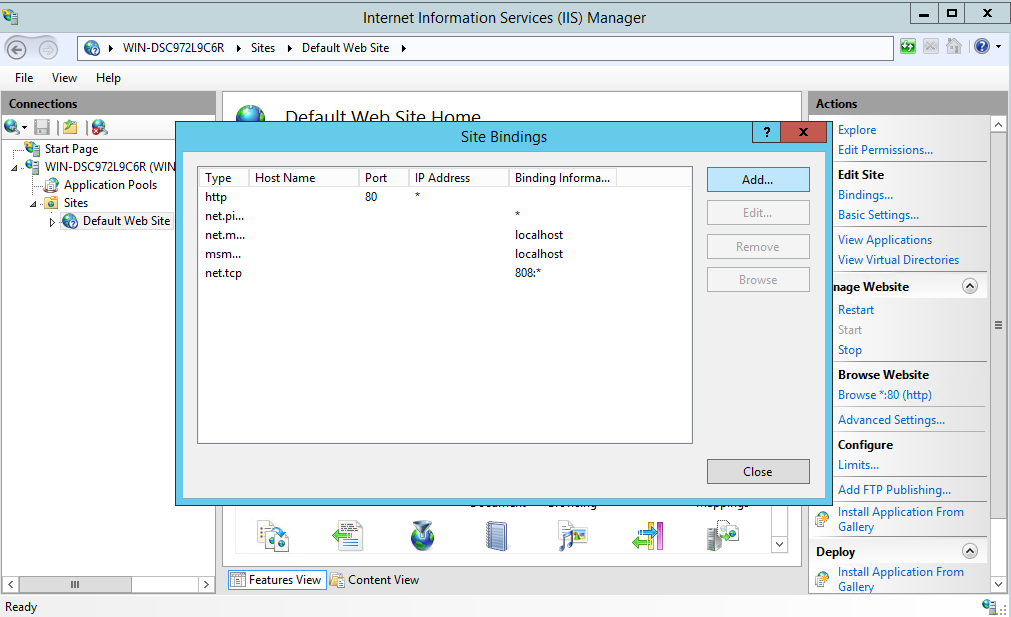
1. Specify the name for the certificate and then click OK.



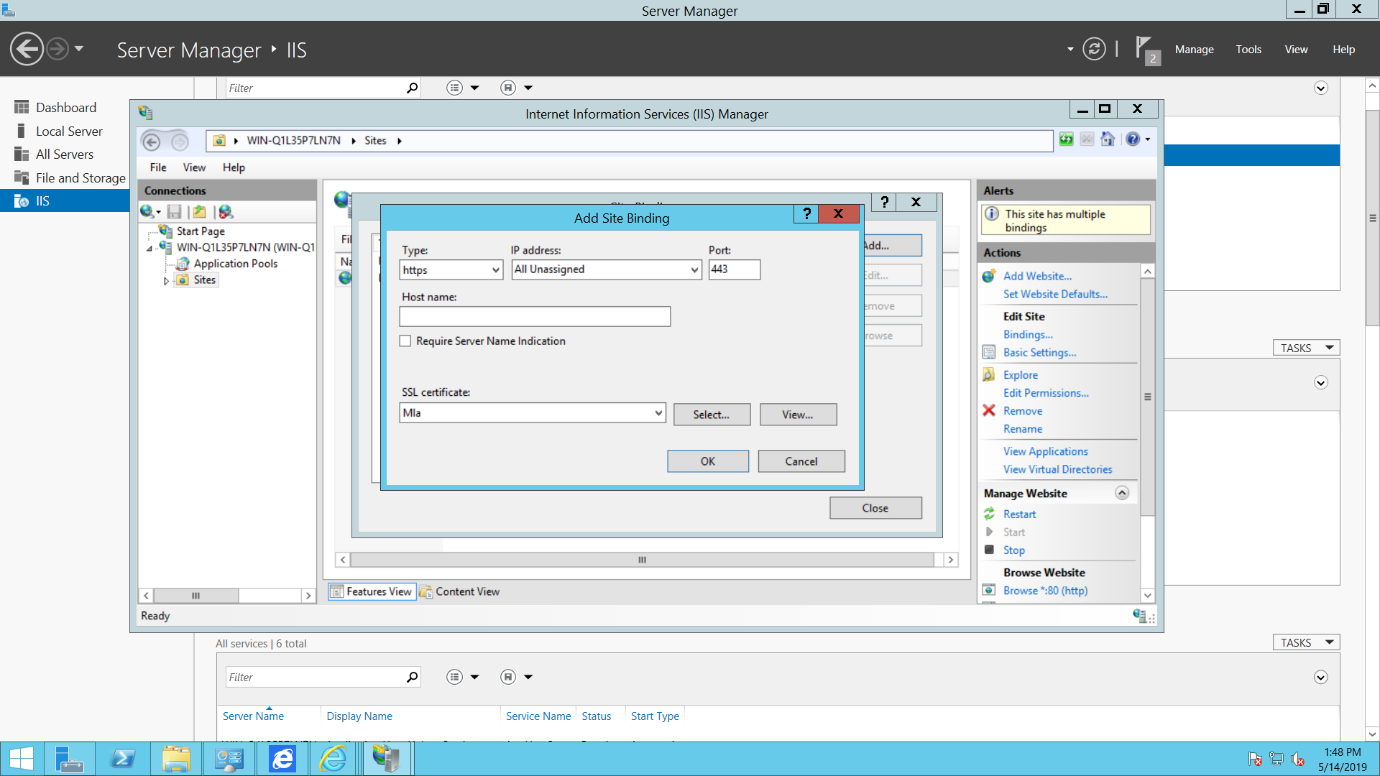
1. Now select “Default Web Site” in Sites folder under connection panel.
2. Select Bindings under “Actions” Tab in right hand side panel.



1. After selecting Bindings, the new window will be as shown.

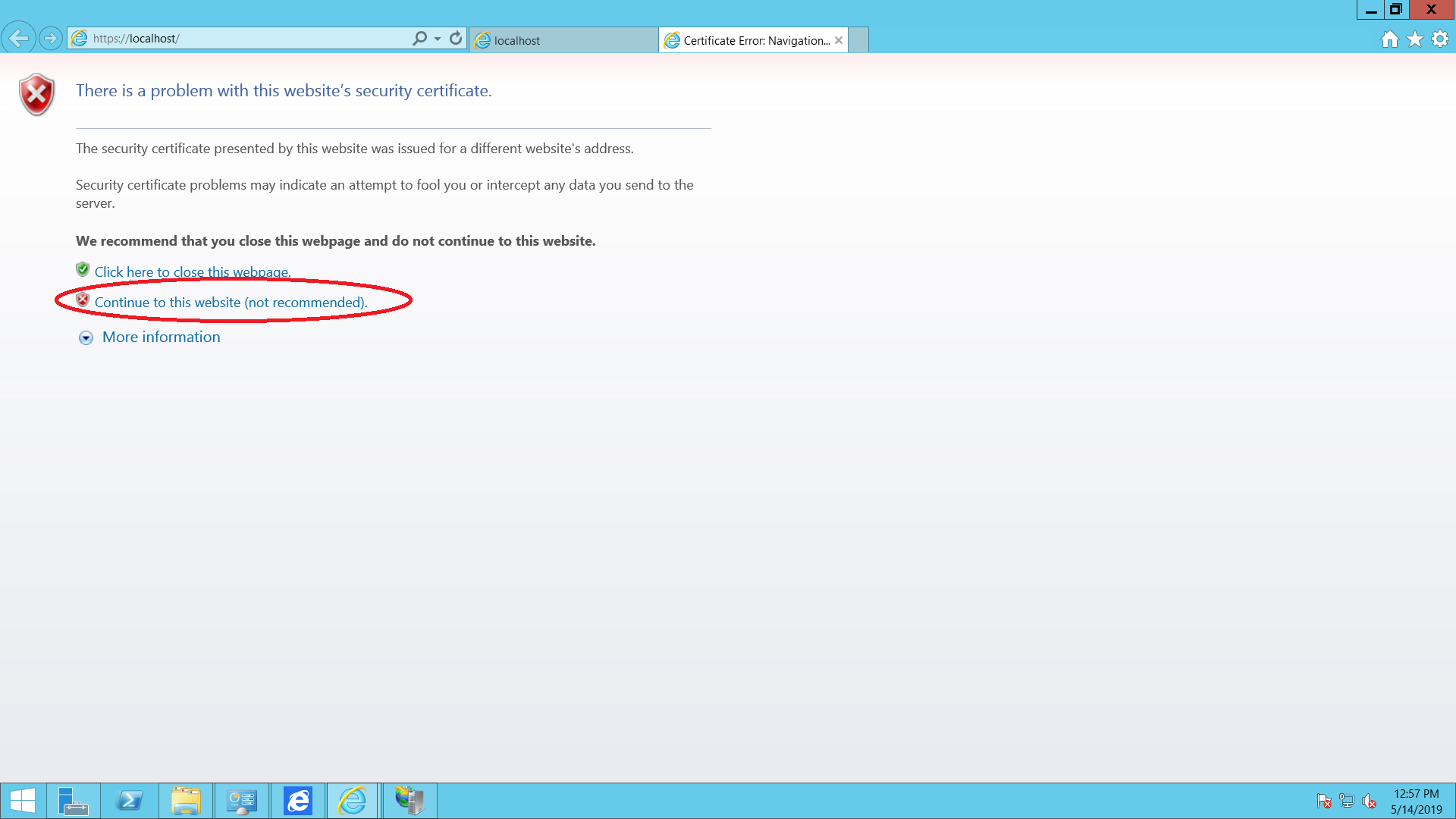


1. Click on Add and add the certificate. Select “https” from type drop down list and the default port assigned is 443. Then select the name of the self-signed certificate that you created previously.

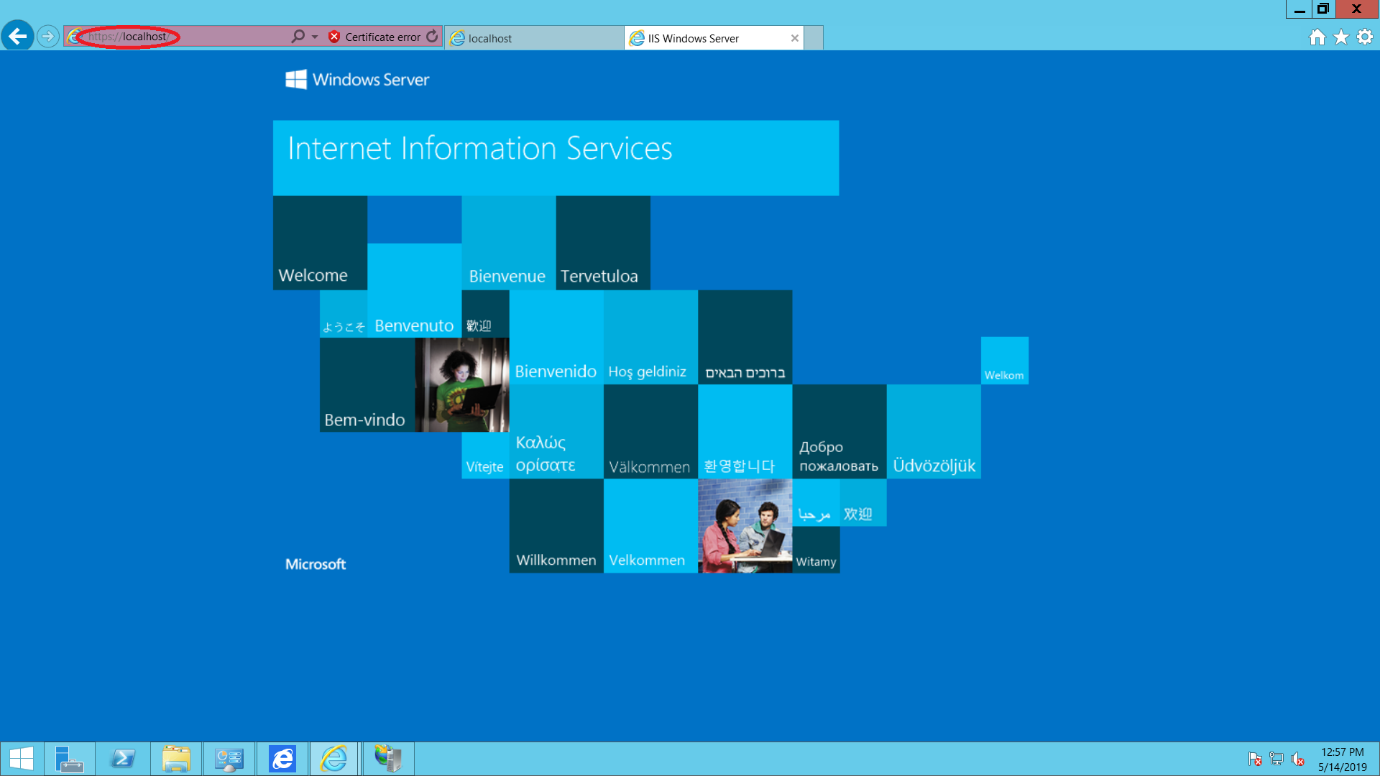


1. For testing HTTPS, select “Default Wes Site”.

11. From right panel select “Browse \*:443(https)”. IIS homepage will be opened in a new web browser. If you have used self-signed certificate, then you will get a certificate error in the web browser.

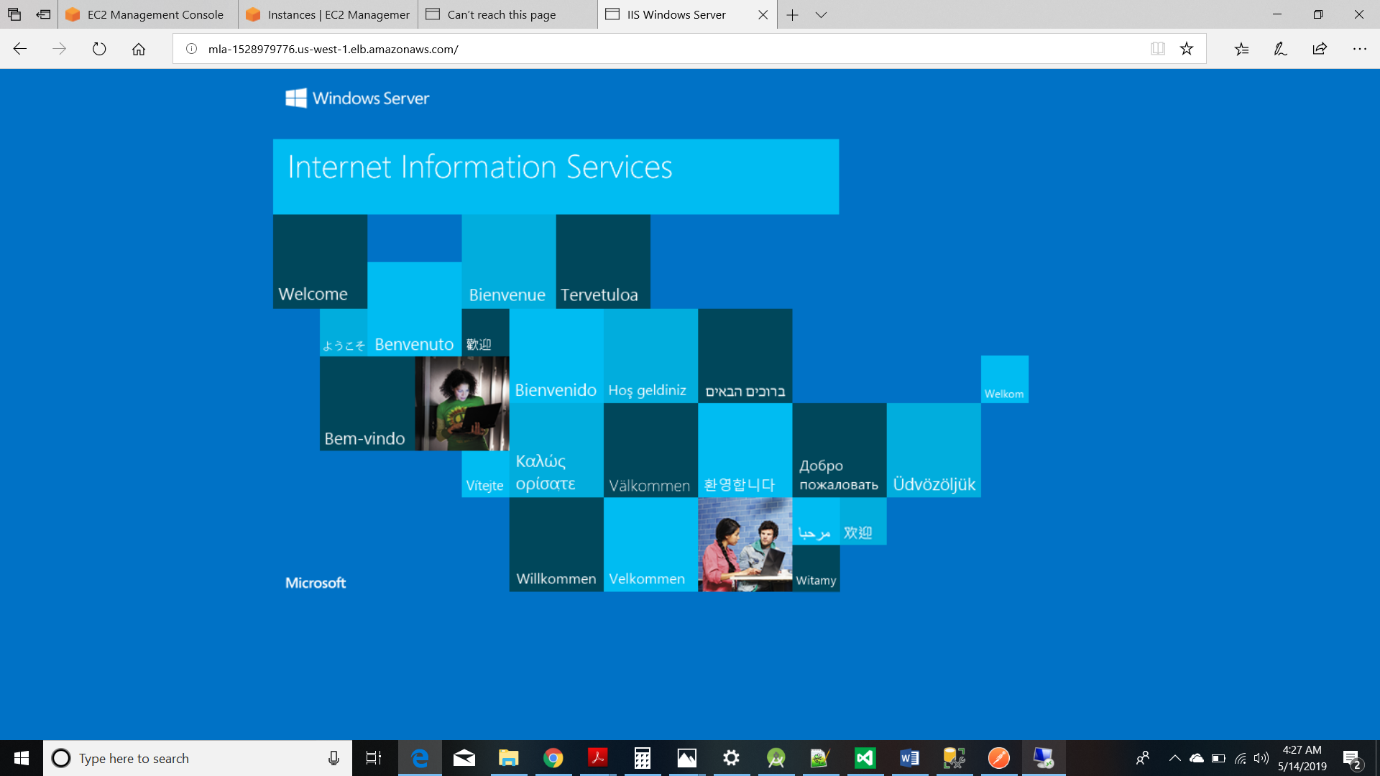
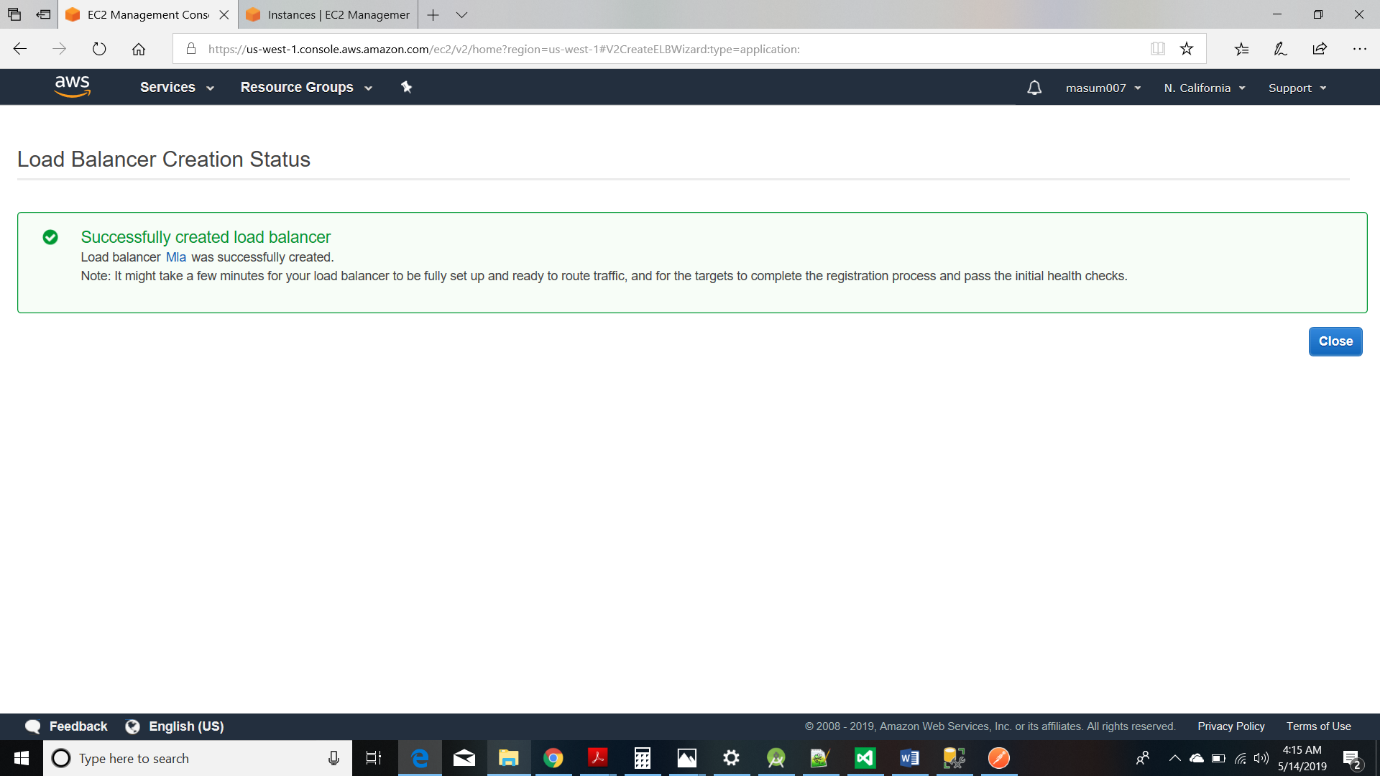
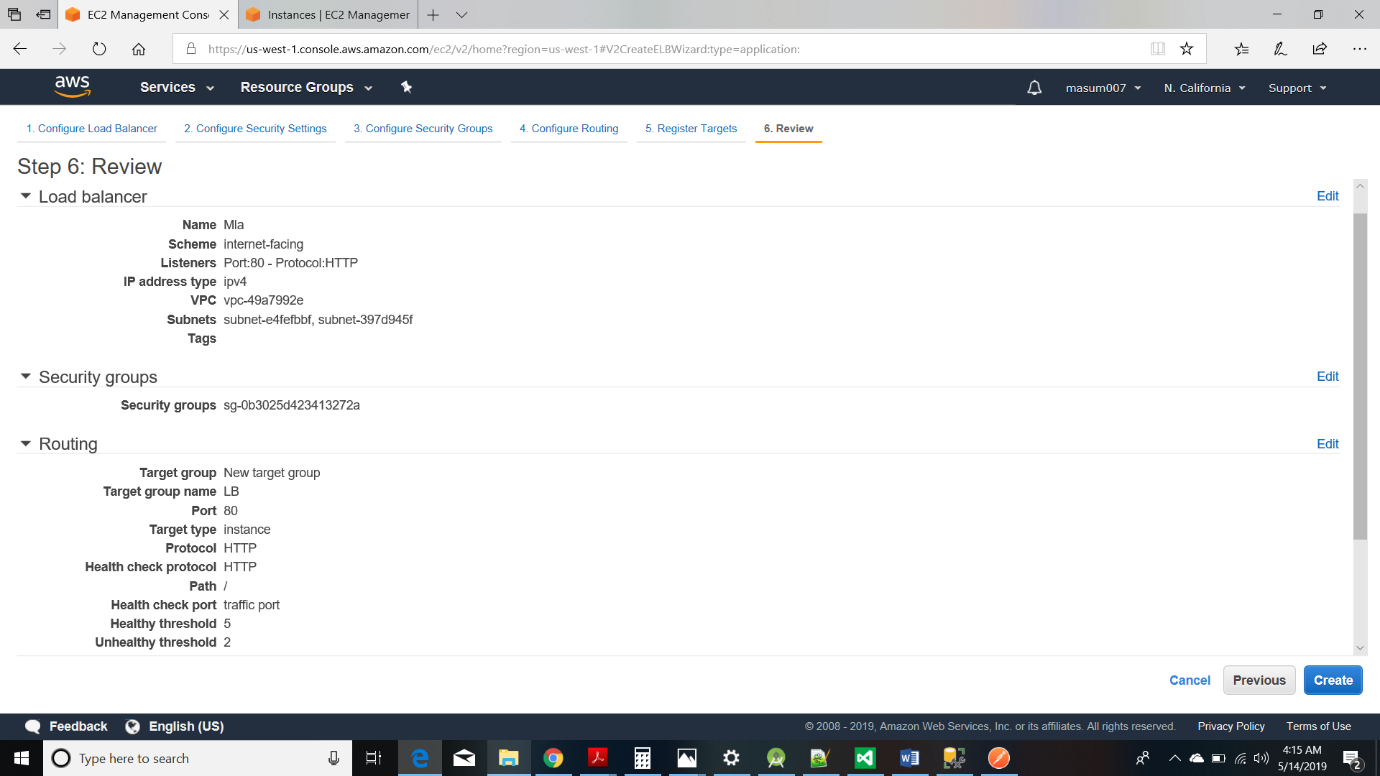
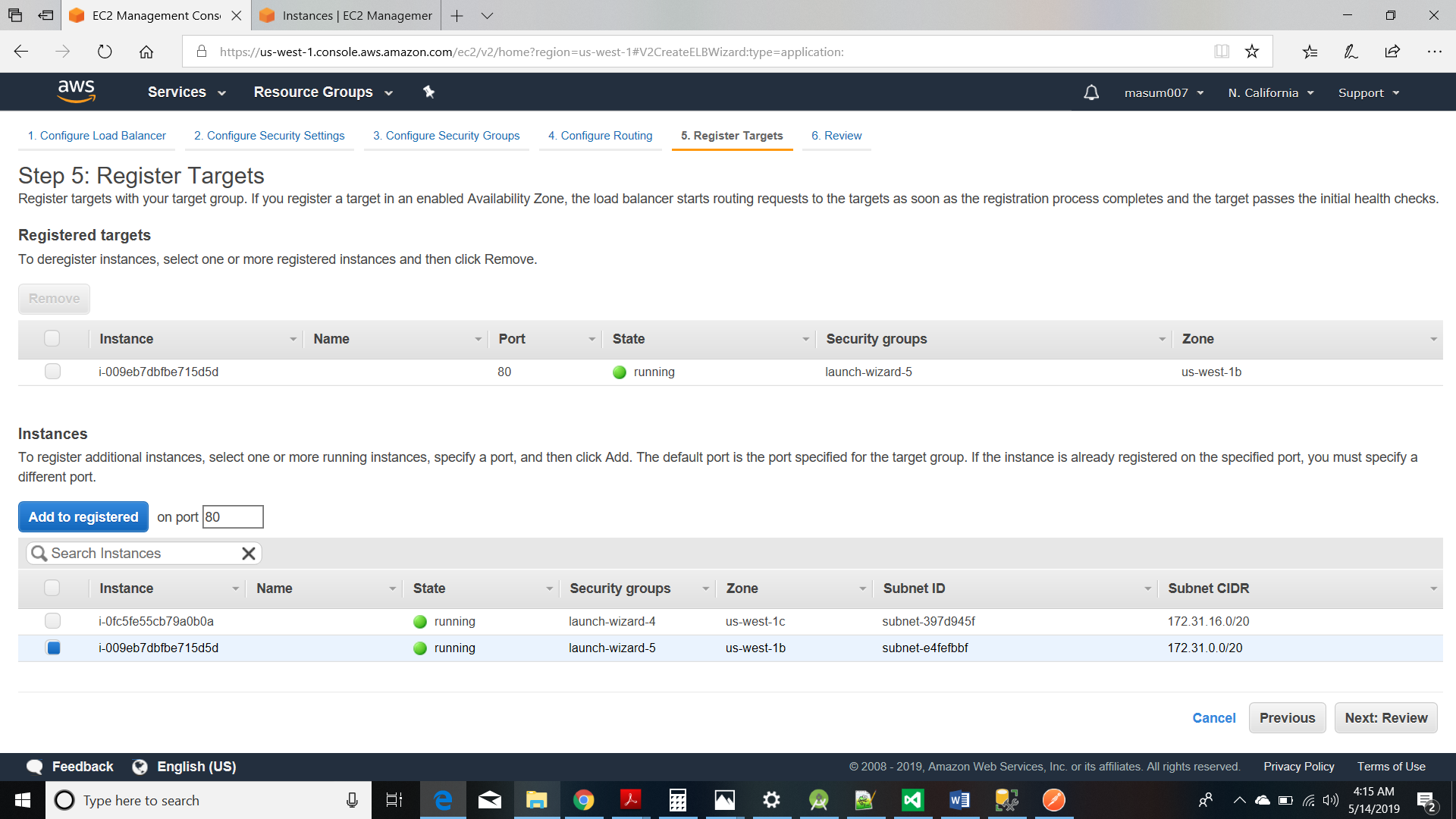
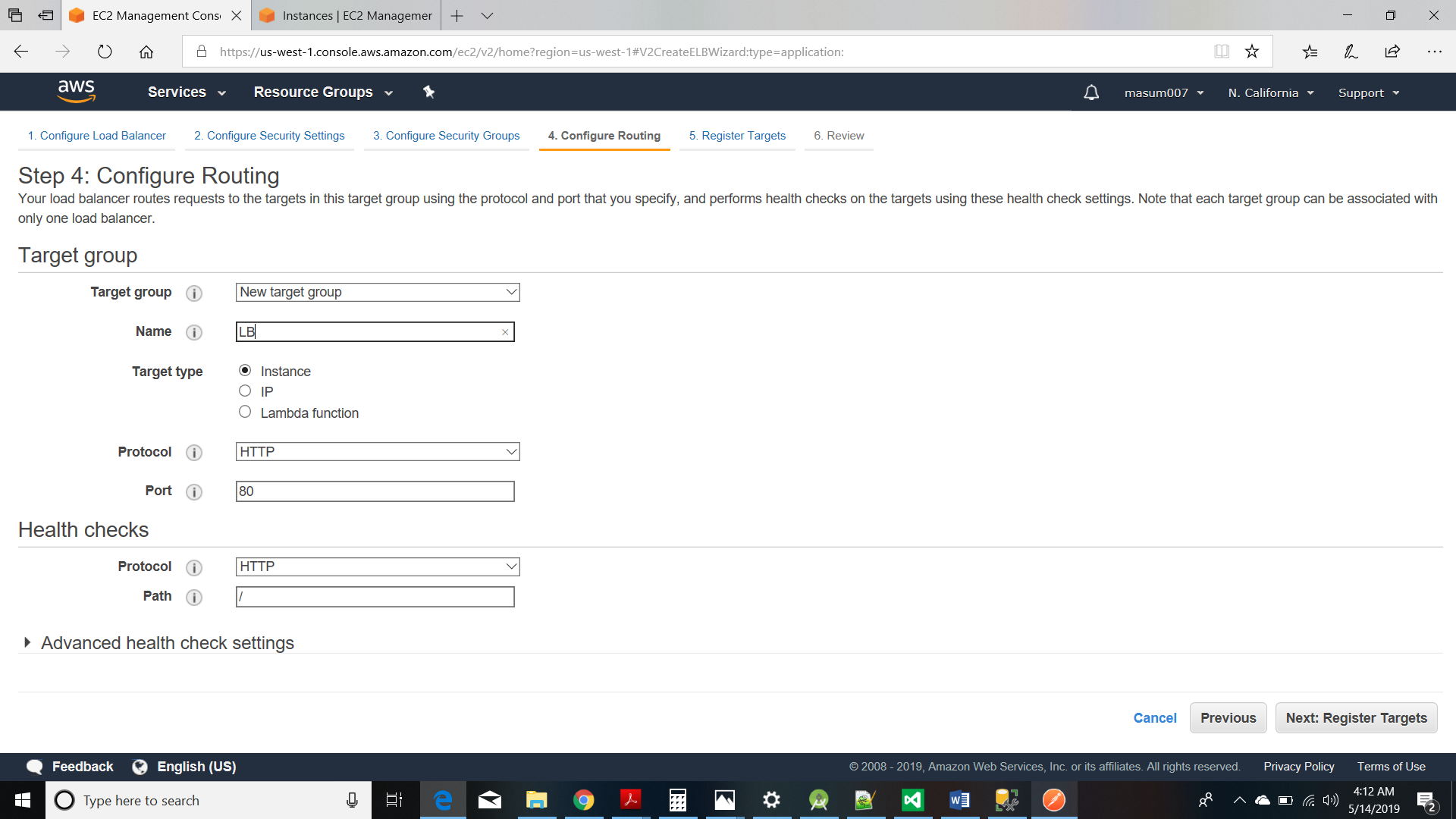
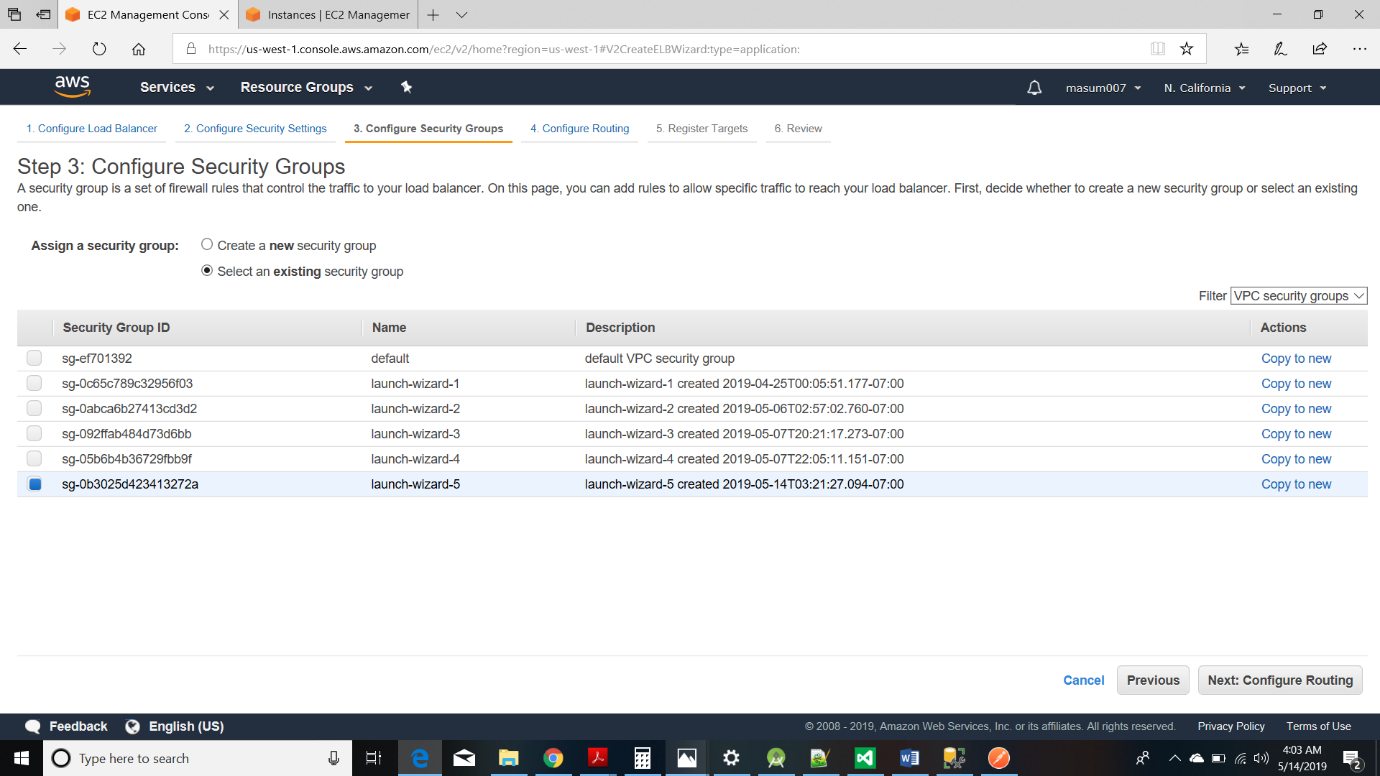
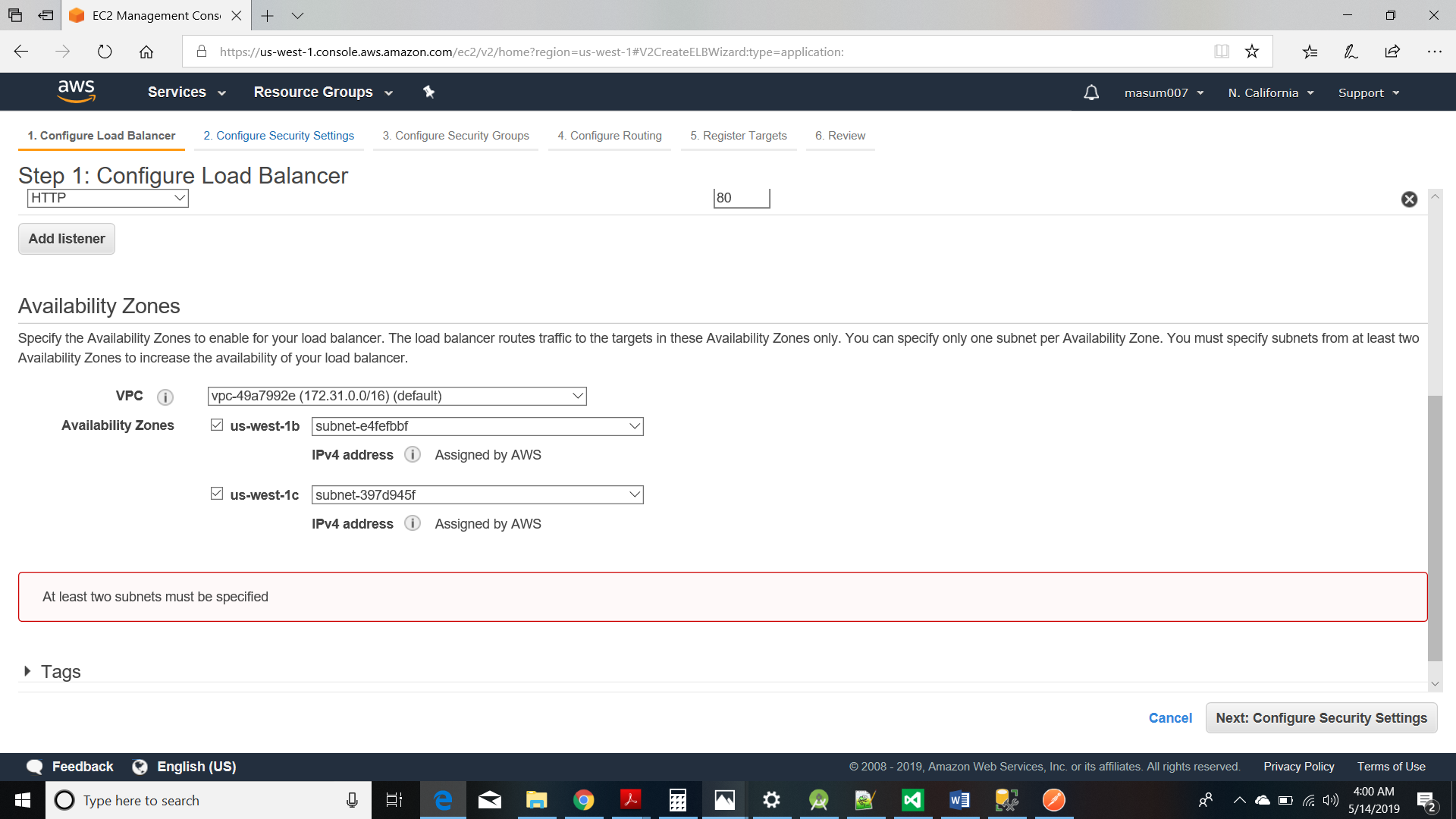
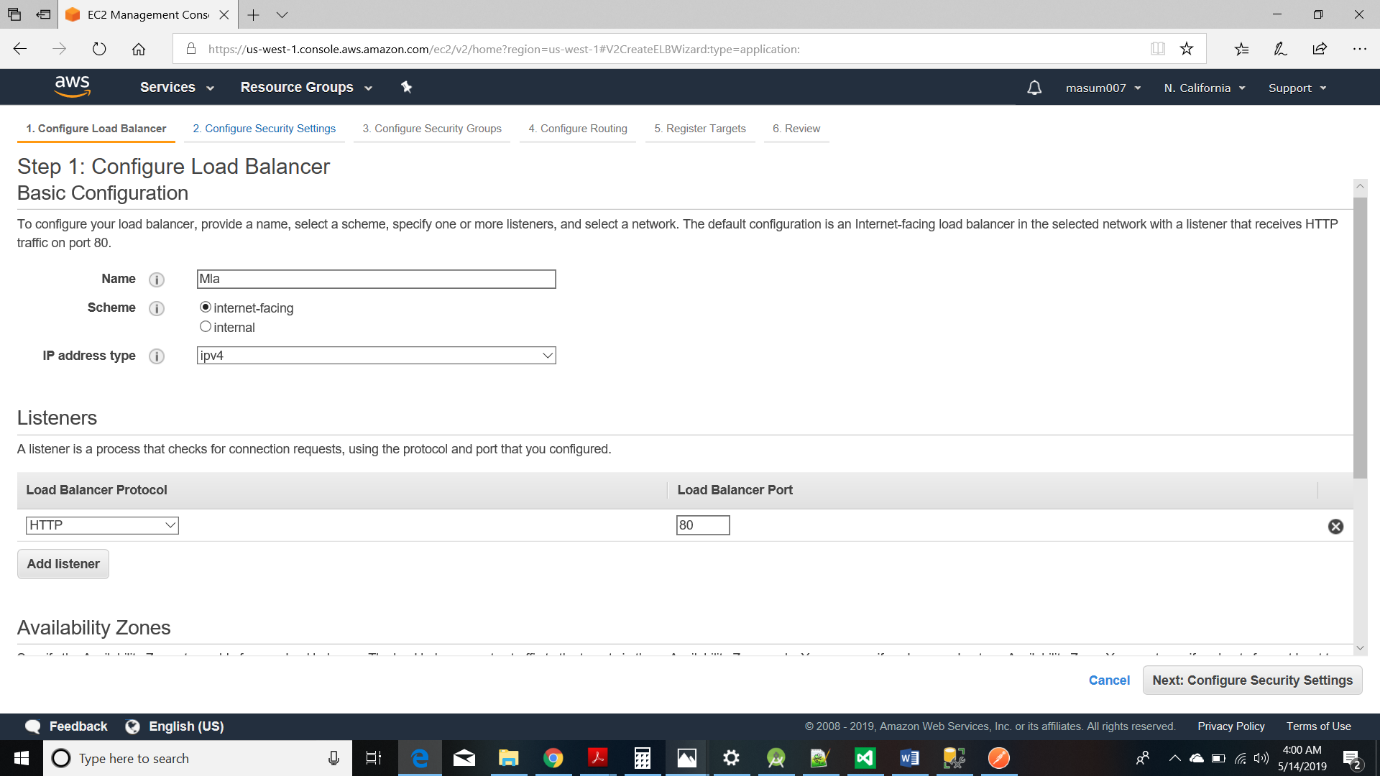


12. Now, select “Continue to the website (not recommended).

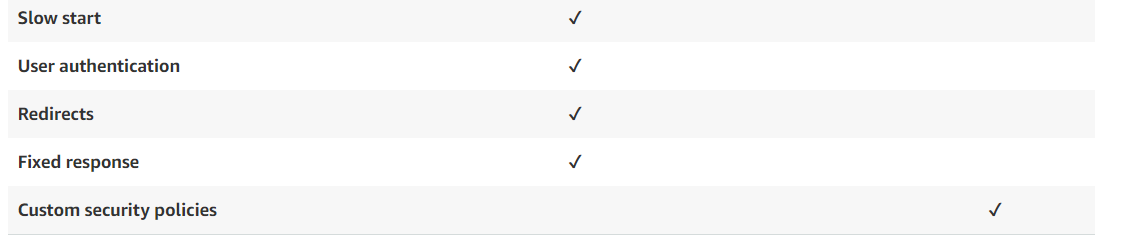
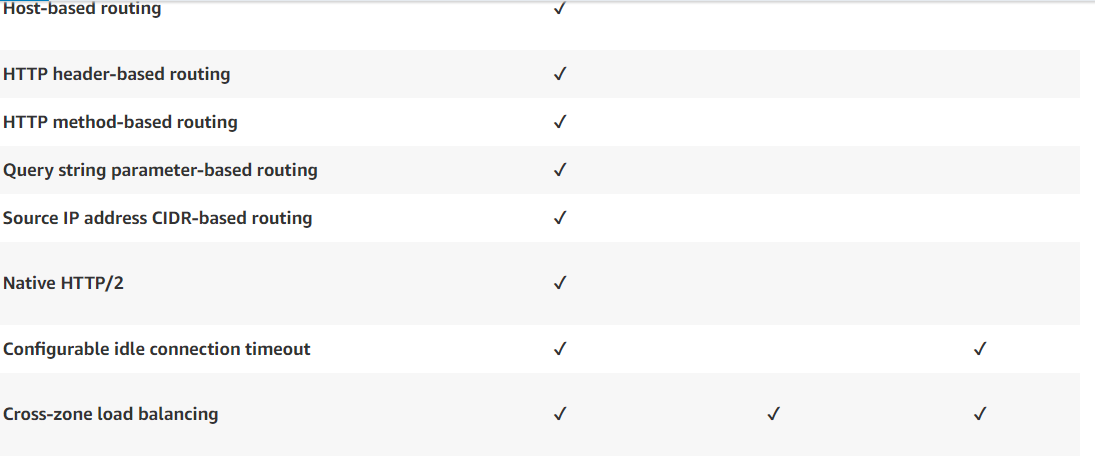
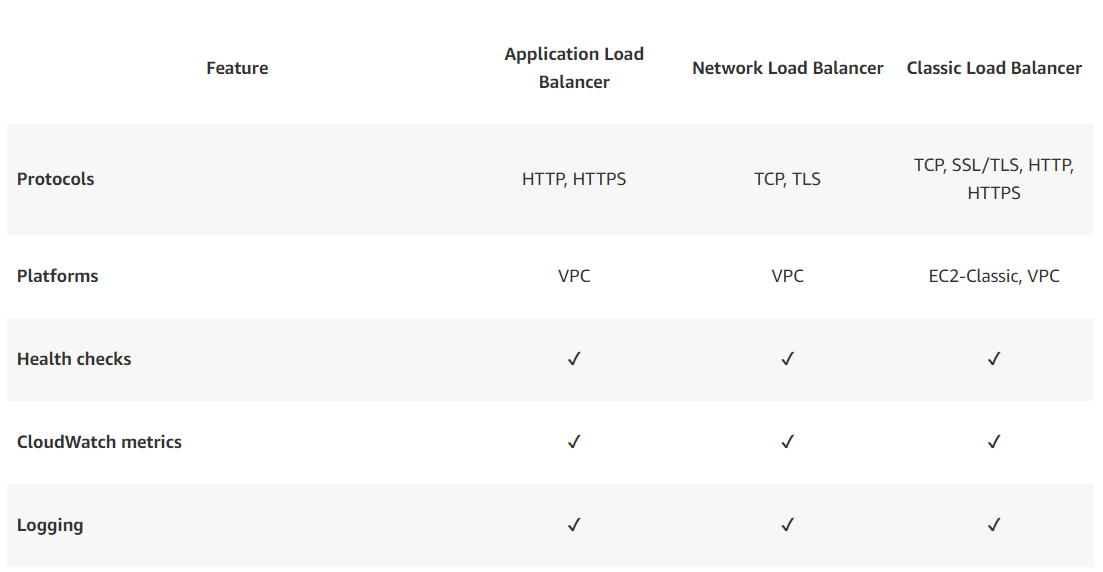


**AWS Load Balancing: -**

After much research on the three types of Load Balancing part, I concluded that Application Load Balancer is the most suitable one in following cases: -

* For path-based routing. You can configure rules for your listener that forward requests based on the URL in the request. This enables you to structure your application as smaller services, and route requests to the correct service based on the content of the URL.
* For host-based routing. You can configure rules for your listener that forward requests based on the host field in the HTTP header. This enables you to route requests to multiple domains using a single load balancer.
* For routing based on fields in the request, such as standard and custom HTTP headers and methods, query parameters, and source IP addresses.
* For routing requests to multiple applications on a single EC2 instance. You can register each instance or IP address with the same target group using multiple ports.
* For redirecting requests from one URL to another.
* For returning a custom HTTP response.
* For registering targets by IP address, including targets outside the VPC for the load balancer.
* For registering Lambda functions as targets.
* For the load balancer to authenticate users of your applications through their corporate or social identities before routing requests.
* For containerized applications. Amazon Elastic Container Service (Amazon ECS) can select an unused port when scheduling a task and register the task with a target group using this port. This enables you to make efficient use of your clusters.
* For monitoring the health of each service independently, as health checks are defined at the target group level and many CloudWatch metrics are reported at the target group level. Attaching a target group to an Auto Scaling group enables you to scale each service dynamically based on demand.

As seen from the above screenshot if the DNS of load balancer is browsed, then it will browse to the IIS home website.

****

**Transactional Processing**

Transaction Processing

• Transaction processing is a logical unit of database processing which either includes one or more access operations that are either succeeded or failed as a complete unit.

• A transaction is made up of many steps, every step in the transaction must succeed for the transaction to be successful.

• If any one part of the transaction fails, then the entire transaction fails and when this happens the system needs to return to the state that it was before the transaction started. This is known as rollback.

• If the transaction is successfully executed, then the transaction is committed that means it transits to another state and the changes are saved. Such changes are reflected in database.

• It follows the four key ACID properties of a transaction: atomicity, consistency, isolation and durability.

• In this project, I have implemented transaction processing for the post method in DeleteStudent controller.

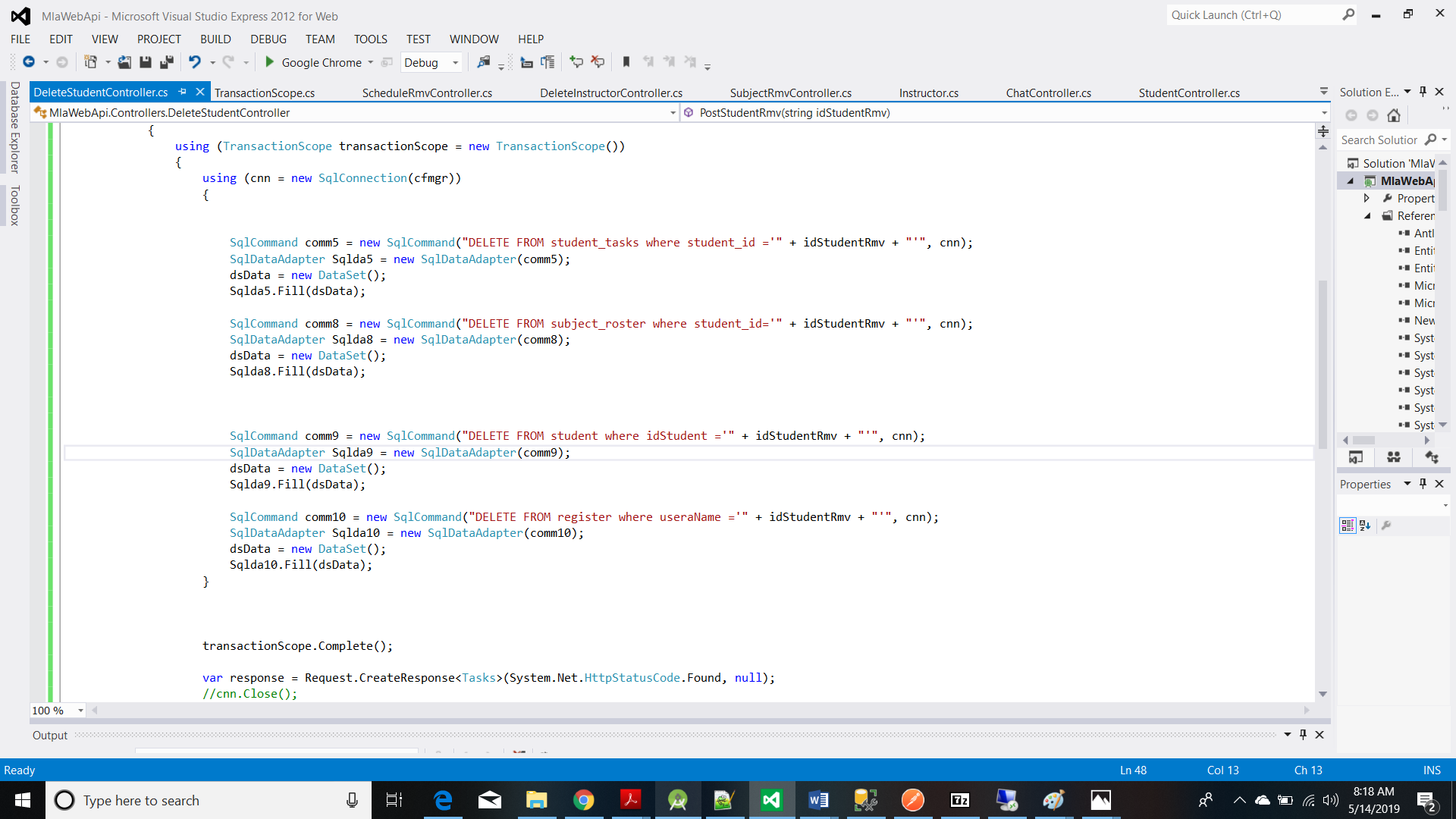
• I have implemented transaction processing for DELETE query. It can also be implemented for Insert and Update query.

**Need of transaction processing?**

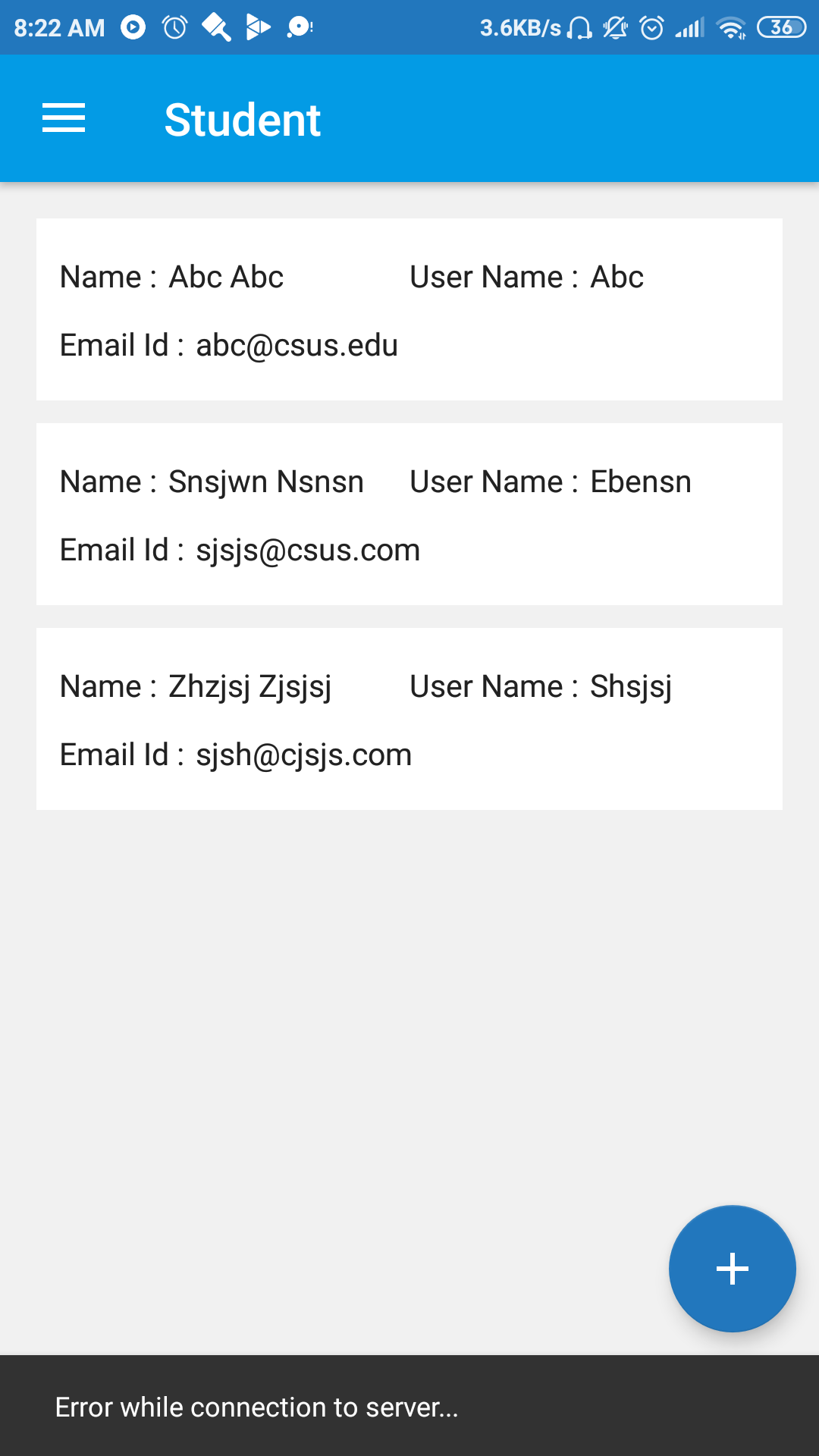
• Transaction processing is used to provide data consistency and concurrency control in our system. Due to concurrency control it ensures that data is updated correctly and appropriately, when the multiple transactions are executed concurrently.

• Data consistency ensures that the transaction must change specific data only in the allowed ways.

• The following code will give a glimpse about the transaction processing.

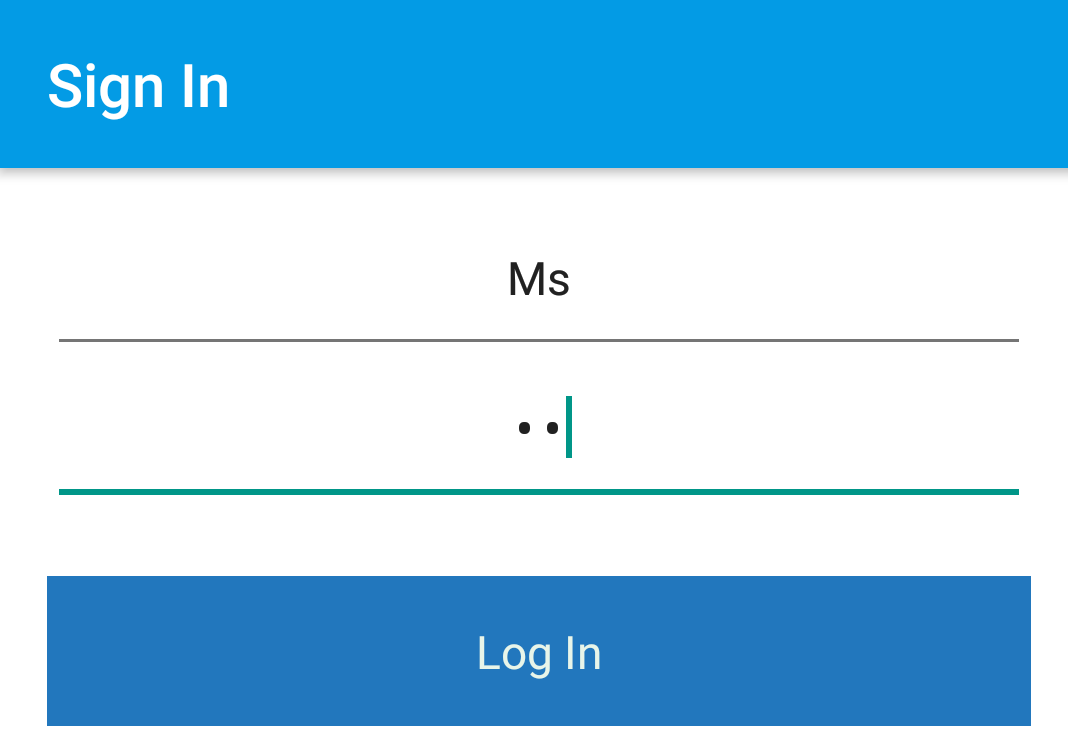


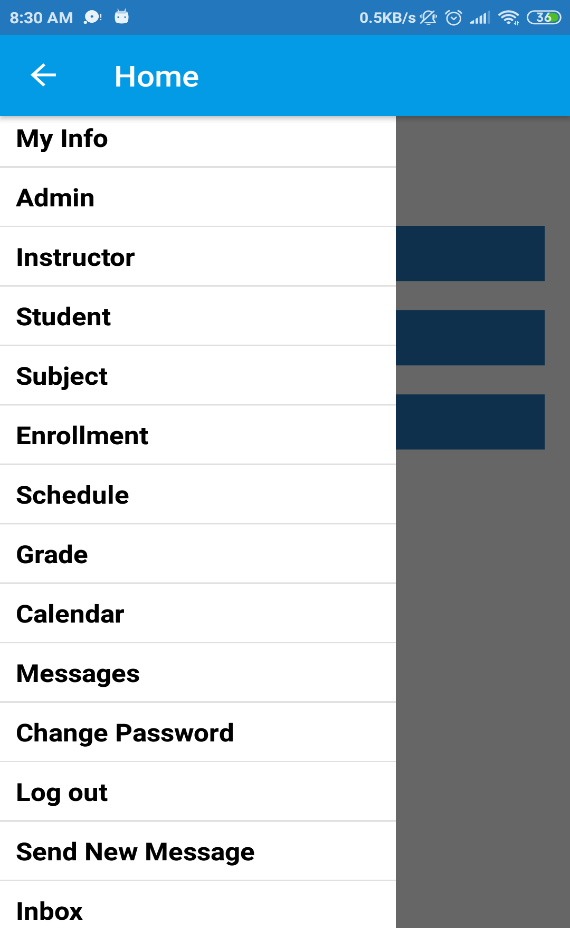
So, we can see that the query to delete student from the register table is written wrong, so using Transaction.scope() we ensure that it is not performed and the whole transaction is rolled back. When I tried to delete student, it showed me error. If I would have not used Transaction.scope(), then it would have deleted student from other tables except the register table which would have cause inconsistency in the data base.

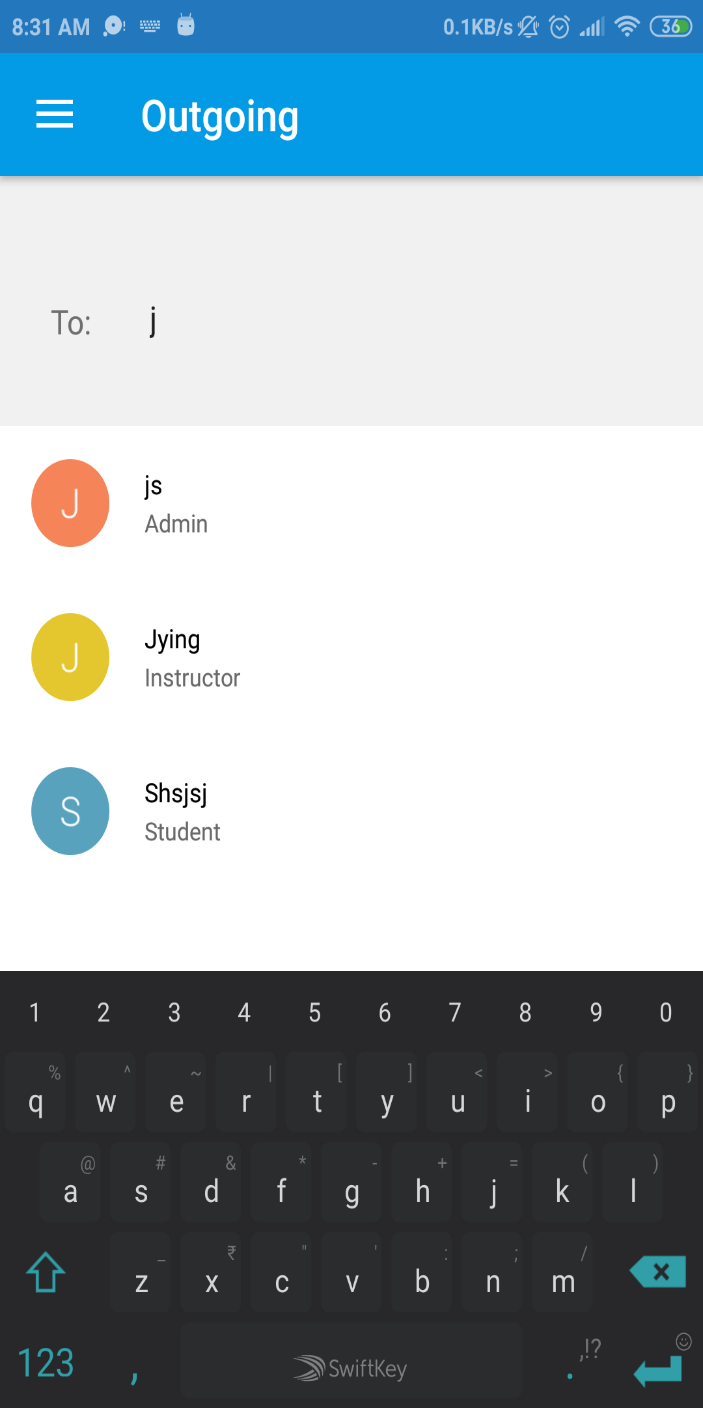


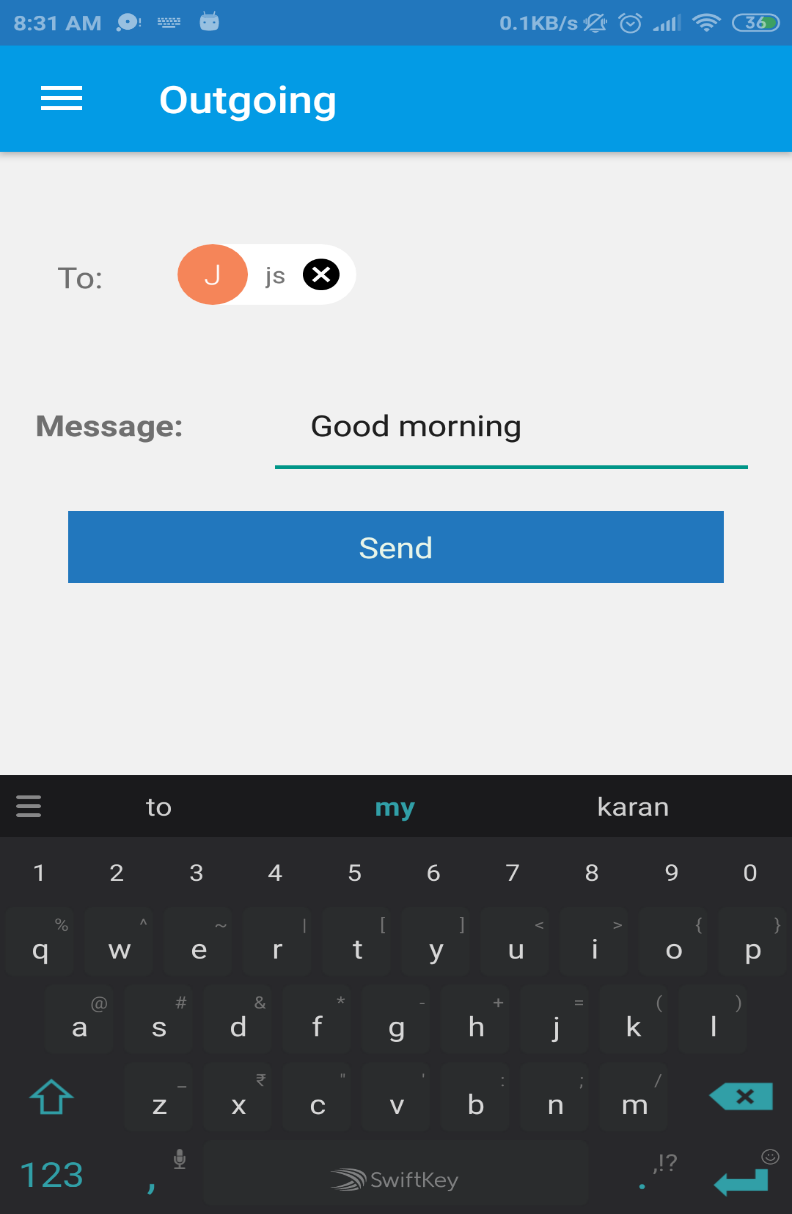
**Chat Application:**

I have implemented chat application in which a sender sends the encrypted message with the help of public key and the receiver decrypts it using private key. For sending a message, I have created a new ChatFragment.java file. The sending process is illustrated in the below snapshots:









For receiving end, I created ReceiveMessageFragment.java and ReceiveMessageAdapter.java files. The receiving process can be summarized from below snapshots:





Finally, we have a look at the database side. I have created a chat table for reflecting all the entries of sending and receiving messages.

