Module name and Code	Distributed Systems and Cloud Computing 6BUIS014C
CW Weighting	60%
Lecturer Setting the task with contact details and office hours	Monday 9:00 -11:00
Submission deadline	07 Nov 2023
Results data and type of feedback	28 Nov 2023
The CW check the following learning outcomes	
2- Understand Principle of the building distributed and cloud software 3- Apply on practice the distributed algorithms 4- Gain Experience in development of distributed and cloud applications	

Task

This is an independent coursework. The Coursework will require you to use your programming skills to develop a simple distributed application. You will develop Create-Retrieve-Update-Delete microservice based API using ASP.NET Core to serve as data store and a web interface to interact this API. Please note that you will be developing a simple front-end application to access and show the availability of the data on your database instance.

Web application with Microservices [*Total 70%*]

MVC front-end [*10%*]

This application will serve as the front-end interface for accessing and managing data provided by the microservice API. Here's a breakdown of the requirements and grading criteria for this project:

- **Structured Models in MVC** (5%)**:**
  Presence of screenshots or code snippets showcasing the structured models within your ASP.NET MVC application. These models should align with the data structures of your company's data stored in the microservice.

- **Screenshots of MVC with CRUD Requests to API** (5%)**:**
  Demonstrate the functionality of your MVC application by providing screenshots or code snippets that show CRUD (Create, Read, Update, Delete) operations being performed on the data retrieved from the microservice API. This should include:
  - **Viewing**: Screenshots of data being displayed in the MVC interface.
  - **Creating**: Screenshots of data creation forms or actions in MVC.
  - **Modifying**: Screenshots of data modification forms or actions in MVC.
  - **Deleting**: Screenshots of data deletion actions in MVC.

Microservice API and DB Instances *[30%]*

An imperative task is to undertake the development of an Application Programming Interface (API) using ASP.NET Core. This API is a critical component of a larger application and is tasked with providing comprehensive CRUD (Create, Read, Update, Delete) functionality. It's essential to note that the resulting Microservice API will be deployed on a Linux-based infrastructure. Therefore, your development process should center on ASP.NET Core, ensuring seamless integration and operation within a Linux environment. Here's a breakdown of your task list:

- **Well-Structured Models** (4%):
  Create well-structured models that accurately represent the data you'll be handling within your API. These models serve as the foundation for how your API interacts with and manages data.
- **CRUD Operations** (8%):
  Ensure that your API provides complete CRUD functionality. This includes:
  - Creating new records in the database.
  - Reading data from the database.
  - Updating existing records.
  - Deleting records from the database.

  The whole CRUD operations per controller should be well commented on.
- **Functioning API that has all connections and working** (3%):
  Develop a fully functional API using ASP.NET Core. This API should be capable of handling CRUD operations for the specific data segment it's designed to serve. Screenshot of its` working state should be present in the report.
- **Screenshots with Version Information** (1%):
  Capture screenshots of your API development environment, emphasizing the ASP.NET Core version you're using. This helps document the technology stack used in your project.
- **Database Migrations** (4%):
  Implement database migrations as part of your project. Migrations enable you to manage database schema changes efficiently. Include these migrations in your project for database updates.

- **Stand-alone SQL File with Insert Statements** (4%)**:**
  Prepare a stand-alone SQL file that contains the necessary Insert statements to populate your company's data within the microservice database. These statements should be well-commented and organized.
- **Database Structure Diagram** (4%)**:**
  Create a visual representation of your database structure in the form of a diagram. This diagram should illustrate the tables, relationships, and key attributes in your database. Include this diagram in your project report.
- **Screenshots of the Database with Inserted Items** (2%)**:**
  Capture screenshots of your database tables after inserting the data using the SQL statements. These screenshots should demonstrate that your data has been successfully added to the microservice database.

Important Note: There must be 30-40 instances of information in the database.

Version controlling (GitHub). [5 %]

Both your project and ASP.NET MVC web part interface and a microservice API – should be uploaded to GitHub and be available in public repositories of your account and should have visible in history. You will deploy the code from the repositories to your virtual machines. Availability of GitHub repository link that is public, that does not contain your name or anything that refers to your nickname online.

Only Student ID should be visible within that Repository (2%)
Screenshot of Repository create, first and last submission (Git Push) with time (3%)

If your name of anything referring to your online nickname is found from any document submitted, you will get absolute 0(zero) for Report part of this Coursework.

Hosting and Code Deploy. [25%]

The application should be hosted on AWS EC2. Three machines are booted for hosting: Windows server machine for ASP.NET application, one Linux machine for hosting Microservice API, and one for serving database server. Note: Machines are not exposed to high load so you can create a VM within the free tier.

For deploying your program to virtual machine, you should use AWS CodeDeploy service, which is automated the deployment of your code on the GitHub repository to AWS. Screenshots of each step should be present in your report. Optionally, AWS Elastic Beanstalk can be used. Note: The names of Deployment Group and application in the AWS CodeDeploy service must contain Student ID.

- **Screenshots of CodeDeploy/Beanstalk Steps** (10%):
  Document your deployment process with screenshots for each step, starting from configuring CodeDeploy or Elastic Beanstalk, connecting to your GitHub repository, and the deployment itself. This step-by-step documentation is crucial for your report.
- **Microsoft Server Setup** (7%):
  Capture screenshots of each stage of setting up the Microsoft server, for the ASP.NET application. This includes configuring the server environment and ensuring it's ready for hosting your ASP.NET application.
- **Database Server Setup** (5%):
  Provide screenshots demonstrating the setup process of the database server. Show how you're configuring and launching the database server, including any necessary security settings.
- **Connecting MVC to API** (3%):
  Showcase how you are connecting your MVC application to the Microservice API. Include screenshots of how and which portion of the code it is done in front-end MVC application and the back-end API.

Report (Screenshots mentioned above are must) [*Total 30%*]

API Description [10%]

The developed API should be properly and thoroughly described to allow other developers to use your API. Suggestion: OpenAPI/ Swagger specification can be used for describing.

Availability of screenshots of descriptions of API (5%)
Availability of OpenAPI/Swagger (5%)

Diagrams [10%]

Flow of work should be done as GANTT chart. It should be a clear and brief description of the developed application provided in this section. You should include various diagrams explaining and illustrating how your application works.

Code Deploy description [10%]

Describe your AWS CodeDeploy/Beanstalk process. Illustrate it using screenshots from the process. GitHub commit keys and deployment group/application names should be present in the screenshot. Include URLs for your Microservice API and ASP.NET application into the report. Also include the reference into your public repositories containing the code.

Absence of URLs in the report will automatically mean a failure for the implementation of hosting/deployment/version controlling parts.

Presentation procedure will be organized during seminars after the submission. The schedule will be sent by the module leader. Failure to attendance of any member of group presentation will result in **a 0 mark** for the **web application with microservices** component (70%).

WESTMINSTER INTERNATINAL UNIVERSITY IN TASHKENT Format

1.Word-processed Times New Roman/Arial 12, on A4 paper.

2.The cover sheet should state your ID number, module title and marker's name. Use the university's standard cover sheet.

3.Include a contents page giving the headings and page numbers of each section.

4.Pages should be numbered.

5.Use Harvard method of referencing if needed

Submission requirements

Coursework must be submitted to University Intranet to the related assessment section. Submission must include references list (bibliography) and appendices. Report must include your student ID number. Report and archive with source code must be submitted in one zip file.

File naming conventions:

Always save the file as follows - Modulename.CW_Number.IDnumber

Example - DSCC.CW1.5467.zip