## Written Test

### Question 1

You have taken over the management of a web application written in ASP.NET C# MVC. The application requires a username and password to login, and the url to access the system is:

<http://made-up-application.com/>

When a user logs into the application they can view their existing details that are stored in the database by going to the following url:

<http://made-up-application.com/customer/details?customerid=1001>

Here is all of the code used in the details action:

public ActionResult details()

{

string database = "Data Source=localhost;Initial Catalog=CustomerDatabase;User Id=sa;Password=password";

string id = Request.QueryString["customerid"];

SqlConnection connection = new SqlConnection(database);

connection.Open();

string query = "SELECT \* FROM Customers WHERE ID = " + id;

SqlCommand cmd = new SqlCommand();

cmd.Connection = connection;

cmd.CommandText = query;

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

ViewBag.Firstname = (string)reader[0];

ViewBag.Surname = (string)reader[1];

ViewBag.Email = (string)reader[2];

ViewBag.Telephone = (string)reader[3];

}

return View();

}

Please list any improvements you think can be made to the above code.

1. DB Connection string should not be hard coded. Best alternative would depend on the hosting environment but as a bare minimum it should be stored in a config file and then retrieved from that but using something like Azure Key Vault (or similar) would be preferable. If hosting environment allows using Windows authentication rather than a password would be better. Not code related, but assuming this is a connection string for a default SQL Server install it should absolutely NOT be using the sa account but a separate db account with access strictly limited to what the application needs to run and the password needs to be changed to something other than “password”.

2. Using numeric id’s to identify customers isn’t ideal. Should be using a GUID as a unique reference to stop customer 1001 changing the number in the url to 1002 and potentially accessing another customer’s details (which should be stopped by other security features but it never hurts to be extra cautious)

3. Sql Connection is opened but never closed. Should use a using statement like this:

using (SqlConnection connection = new SqlConnection(database))

{

SqlCommand cmd = new SqlCommand(queryString, connection);

command.Connection.Open();

}

so the connection is automatically closed after use. (or failing that manually close the connection)

The same applies to the SqlDataReader.

4. The query string should be using a parameter instead of the id string to prevent SQL injection attacks.

5. Depends on the database design but selecting all fields with SELECT \* could be retrieving a lot of unnecessary data if the table has a lot of fields, could be possible performance improvements made by just selecting the fields that are required. (Though if the table itself only has 5 fields this wouldn’t matter). Using

6. Error handling needs to be added. Everything related to the db connection should be wrapped in a try catch block, with appropriate for the application error handling and logging added if the connection or query failed for any reason.

7. You could refactor the code so that there is a separate GetCustomerDetails(string id) function in a class library that handles the database call and returns a CustomerDetails object containing the information. The CustomerDetails object could then be used as a model and returned to the view rather than using ViewBag and would also be available for any other places in the application that require the same data to be retrieved.

### Question 2

Given a base URL of <https://made-up-api.com/> for a set of RESTful web services, what URL and HTTP verb might you expect to use to carry out each of the following actions:

* Get a list of all customers.
* Get the details of order number 25.
* Delete the customer with an ID of 133.
* Update a single field for all orders.
* Update all fields for the customer with an ID of 1701.

## Practical Test

### Task 1

Please find included an attached file Students.csv. This file includes details for 20 students. Take this csv file and import the values into a database so it can be accessed using SQL Server Management Studio.

### Task 2

Using Visual Studio create a new ASP .NET MVC Web application that can access the database that has been created. Please generate some Create, Read, Update and Delete controller actions and views to allow the data to be managed from the web application.

### Task 3

Update the database to include a Year of Study field (1st Year, 2nd Year, 3rd Year) for each student, and populate this field for all existing students in the database.

Once the database has been updated then amend the controllers and views created in Task 2 to include the Year of Study field.

### Task 4

Please find included an attached file Subjects.csv. This file includes details of 5 subjects studied by the students. Please import the subject information into the database, and setup the required relationships in the database so that the following is correct:

Students 1 - 5 study Biology and History

Students 6 - 10 study Chemistry

Students 11 - 15 study History

Students 15 - 20 study Physics and Geography

Update the Read student Controller and View to display the subjects the student is studying

### Task 5

Update your Visual Studio solution to use a Web API 2 controller to allow the Student information to be output so it can then be easily consumed by other applications.