|  |  |
| --- | --- |
| MIT%20Award%20smal(bw) | **Faculty of Business and Information Technology: BDT – Bachelor of Digital Technologies**  **502.632 - Full-Stack Web Development**  **PRACTICE Practical Test**  **SAMPLE Assessment Cover Sheet** |
| Course | 502.632 - Full-Stack Web Development |
| Assessment name | PRACTICE QUESTIONS - "Music Store" |
| Assessment type | Practical Test – "Open Book" – reference allowed to The Internet, notes, examples, Canvas Resources. Live information exchange between students and anyone else (except the supervisor(s)) by any means is not allowed during the test. |
| Date | REAL TEST WILL BE Tue, 12 Mar 2019 |
| Course contribution | REAL TEST WILL BE 20%  (Test has 20 marks - each test mark is a course mark). |
| Time Allowed | 135 minutes |
| **Student ID:** |  |
| **Student NAME:** |  |
| **Statement of Valid Authorship**  I hereby confirm that these test answers are my own work done in the test time. This means that to the best of my knowledge and belief, this test contains only my own created and assembled responses to these particular questions without communication with any other person except the course lecturer. I make this statement fully understanding that, should it be found false, I will, in most circumstances, receive zero marks for this test.  Signed by student: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| **Relates to all 4 Learning Outcomes, but especially to 2 and 4:**   1. Demonstrate an understanding of the architecture of web applications including the roles of clients and servers and the use of protocols for interaction. 2. **Learn and apply a server-side programming platform and/or language.** 3. Implement visually rich interactive web applications using HTML, CSS, JavaScript, and AJAX. 4. **Implement and publish web-based software solutions that interact with a variety of data sources and public APIs.** | |

**PRACTICE TEST 01 - MvcMusicStore**

Use ASP.NET Core 2.2 if available, otherwise ASP.NET Core 2.1 is acceptable.

**Question ONE - Setup [1 mark]**From Canvas, Download file "PracticeMusic.zip".  
Move this into "C:\Users\<Your Username>"  
Or Mac or personal laptop defined working area.

Unzip -- 7-Zip recommended -- using "Extract Here".  
You should now have folder "PracticeMusic".  
as in "C:\Users\<Your Username>\PracticeMusic" for lab machines.  
Rename this adding your first name. e.g. John would rename his folder "PracticeMusic\_John"

Use Visual Studio to create a new ASP.NET Core 2.2(or 2.1) web app under "PracticeMusic\_<FirstName>".  
Do NOT check (tick) the checkbox "Create directory for solution"   
Name the app "MVCManukauTech"  
Select option "Web Application (Model-View-Controller)"  
Click button "Change Authentication" and change authentication to "Individual User Accounts".  
Create this app.  
 **Question TWO - Database Setup** **[1 mark]**  
Use your database account to create a new empty database to work with.  
Name it in a similar way to your starter database using "PMusic" as the 3rd part of the database name.  
eg John has a starter database "F191\_John\_Starter"  
He creates this new one as "F191\_John\_PMusic"

Then load the supplied script "MvcMusicStore-Create\_JPC.sql" into SQL Management Studio (or Visual Studio). Edit the name of your new database into the script .

Run the script to populate the database. Check that tables have appeared.  
Show the result to your marking lecturer who will assess this "on the spot".  
It is essential for you to complete this question to be able to move on to most of the other questions.  
For that reason the lecturer may help you complete Questions ONE and TWO but in such a case you will probably get reduced marks for these questions depending on the reasons for you getting stuck.

**Question THREE [1 mark]**Edit app file "appsettings.json" to include the connection string for the database you have created.

Test this by registering a test user. This should generate the "AspNetUsers" and related Identity tables in the Database. Check for this result.

**Question FOUR** **[2 marks]**  
Add 4 new membership fields (columns) for extra member information  
in table "AspNetUsers". These new fields are:  
“AverageAnnualSales”, "Address01", “Address02”, "IsMarried"  
You choose appropriate data types for them.

**Question FIVE** **[4 marks]**  
Make the necessary code and UI changes so that the Registration Form now includes   
these 4 extra fields and the code saves their user data entry to table "AspNetUsers".

**Question SIX [2 marks]**  
Work through the "Scaffold-DbContext" command process which causes your app systems   
to scan the database tables and automatically generate the classes for data access.

**Question SEVEN [2 marks]**  
Get database table “Genre” to display as an on screen web user interface.  
with Create, Update and Delete capabilities.  
  
**Question EIGHT [2 marks]**  
In the "Details" view generated in Question SEVEN, replace the generated TagHelpers and HtmlHelpers  
with code that is as close to standard HTML as possible.

**Question NINE [5 marks]**Here is a URL with QueryString giving input of an Invoice in JSON format

http://localhost:7151/Calc/Inv?j=[ {"Product":"Firework","Qty":4,"UnitPrice":2.20}, {"Product":"Formation","Qty":2,"UnitPrice":1.90}, {"Product":"Mercy","Qty":5,"UnitPrice":2.10}]

Create the Remote Method Web Service to calculate and display the Invoice Total.  
The expected output is 23.10

Extra help because this is a practice test: Here is the code for a "ViewModel" to match this JSON.  
Create folder "ViewModels", use "Add" 🡪 "Class" to create a file "Invoice.cs" and paste this code into it.

namespace MVCManukauTech.ViewModels

{

public class Invoice

{

public string Product { get; set; }

public double Qty { get; set; }

public double UnitPrice { get; set; }

}

}

You will need these "using" statements at the top of your new Controller.  
using Newtonsoft.Json;  
using MVCManukauTech.ViewModels;

You can then feed the JSON data into an object variable of type List<Invoice>

**Test will have 20 marks**