|  |  |  |
| --- | --- | --- |
| **Qualification Details** | | |
| **Training Package Code & Title** | ICT - Information and Communications Technology (Release 7.2) | |
| **Qualification National Code & Title** | | **State code:** |
| ICT50220 - Diploma of Information Technology | | **AC21** |
| **Units of Competency (UoC) detailed in this DAP | Cluster: MVC and Non-relational databases** | | |
| **Unit National code and title** | | **State Code** |
| ICTPRG554 Manage data persistence using NoSQL data stores | | OBS89 |
| ICTPRG556 Implement and use a model view controller framework | | OBS87 |

*Students to sign this document when submitting an assessment*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assessment description** | | | AT2 Project | | | | | | |
| **Assessment date** | | | Weeks 8-18 | | | | | | |
| **Student Name** | | | Andrew Millett | | | | | | |
| **Student ID** | | | M164821 | | | | | | |
| **Student Declaration** | | | I have read and understood the details of the assessment.  I have been informed of the conditions of the assessment and the appeals process.  I agree to participate in this assessment.  I certify that the attached is my work.  *Andrew Millett* | | | | | | |
| **Assessors Name** | | |  | | | | | | |
| **Date Due:** | | |  | | **Date Submitted:** | | | |  |
| **STUDENT FEEDBACK** | | | | | | | | | |
| **Assessment Decision** | Attempt 1 | | | ☐ Satisfactory | | | ☐ Not Yet Satisfactory | | |
| Attempt 2 | | | ☐ Satisfactory | | | ☐ Not Yet Satisfactory | | |
| Attempt 3 | | | ☐ Satisfactory | | | ☐ Not Yet Satisfactory | | |
| **Assessor Name** |  | | | | | | | | |
| **Assessor Signature** |  | | | | | **Date:** | |  | |
| **Feedback to student** | | | | | | | | | |
| Feedback will be provided to you in class or via Blackboard | | | | | | | | | |
| **Feedback from student** | | | | | | | | | |
|  | | | | | | | | | |
| **Student signature** | |  | | | | **Date:** | |  | |

|  |  |
| --- | --- |
| **INFORMATION FOR INSTRUCTORS/ASSESSORS** | |
| Type of Assessment | AT2 |
| Duration of Assessment | 10 Weeks |
| Location of Assessment | * Classroom and home |
| Conditions | * Assessment may be completed in class or at home. All documentation must be submitted via Blackboard, including proof of individual work in places where directed. * To verify the authenticity of the student’s assessment, you may ask the student to again produce an answer to an existing question. |
| **Elements and Criteria:** | |
| Unit of Competence Elements:   * Review and select NoSQL options * Determine and create storage of data types * Build and configure indexes * Use queries and retrieve objects * Confirm interaction of objects * Create an MVC project * Use the MVC framework * Finalise MVC project   **Performance Evidence**  The candidate must demonstrate the ability to complete the tasks outlined in the elements, performance criteria, and foundation skills of this unit, including evidence of the ability to:   * create at least one model view controller (MVC) project and confirm the functionality of the framework, including:   + HTTP handers and routes for getting, POST, PUT, and DELETE requests   + HTML templates, view models, and dynamic rendering   + HTTP requests, responses, and redirects. * specify partition and sort keys * optimize the data. * create at least three different queries, including updating, deleting, and creating data types * create at least two indexes.   **Knowledge Evidence**  The candidate must be able to demonstrate knowledge to complete the tasks outlined in the elements, performance criteria, and foundation skills of this unit, including knowledge of:   * benefits and functions of NoSQL database and schema-free data persistence, as well as traditional relational data models * methods and different features and functions between scaling out and scaling up (horizontal and vertical) * language used in required programming language for NoSQL applications * partitioning in a NoSQL environment and its related terms * functions and features for time-to-live (TTL) requirements * authorization and authentications procedures and levels of responsibility according to client access requirements * distribution of data storage across partitions * debugging and testing methodologies and techniques * functions and features of sort keys in NoSQL storage * features of transport encryptions, authentication, and authorization * different NoSQL data store formats, including:   + key value   + document based   + column-based   + graph based * different NoSQL data types, including:   + numeric   + string   + boolean   + complex   + date time | |

Structure and format reports in a clear manner that conforms to organizational requirements

|  |  |
| --- | --- |
| **INFORMATION FOR STUDENTS** | |
| Purpose of Assessment | This assessment is a research assessment that encompasses all knowledge learned throughout the semester and promotes research on upcoming trends and technologies within the Information MVC and NoSQL and it must adhere to APA6 referencing.  This assessment also covers knowledge questions related to:   * Record keeping requirements * Privacy, policies, and legislation around securing workplace information * Research concepts   To verify the authenticity of your assessment, your lecturer may ask questions to substantiate it is your work. |
| Allowable materials | Weekly Readings, Class notes, Weekly Activities |
| Required resources | Computer with:   * Computer operating system; * Internet Access; * Word processing software; * Access to Blackboard |
| Assessment Presentation and Submission | The material and links to related resources are available within the Blackboard course shell created for this unit.  All questions and activities must be attempted.  Use of research tools and peers in formulating answers is acceptable – but the work submitted must be your work.  Final documentation is to be uploaded to the appropriate area in the Blackboard course shell created for this unit.  If you are marked as NYS (Not Yet Satisfactory) on your first attempt, you will be provided with another opportunity to re-attempt the assessment at the discretion of the lecturer. |
| Reasonable adjustment | In some circumstances, adjustments to assessments may be made for you.  See the DAP for more information |
| Assessment contents | This assessment consists of:  Design requirements->Project Report  TASK 1: Plan your app  Task 2: Create a Web App that will use data from our API  Task 3: Test and fix the data persistence process according to business requirements |

*Continue to the next page*

|  |  |  |  |
| --- | --- | --- | --- |
| **Student’s Name** |  | **Date marked** |  |

**ASSESSMENT SUBMISSION CHECKLIST**

Use the checklist below to ensure you have submitted all the necessary files

|  |  |
| --- | --- |
| Design requirements Project Report | **Check** |
| Answer questions A to q through a small report |  |

|  |  |
| --- | --- |
| TASK 1: Plan your app | **Check** |
| Organize your workplace and read client functional requirements. **MONGO DB USING** |  |

|  |  |
| --- | --- |
| Task 2: Create a Web App that will use data from our API | **Check** |
| Design phase, program your API CRUD functionality |  |

|  |  |
| --- | --- |
| Task 3: Test and fix the data persistence process according to business requirements | **Check** |
| Test phase, test, and document all functionalities |  |

**Introduction:**

This assessment is a report which encompasses all knowledge learned throughout the weeks of teaching and promotes analytic skills on upcoming trends and technologies within the MVC and NoSQL.

**Scenario**

You have been hired as the new programmer by ITEMS, you are required to work on a project to analyze a business situation, build an API, MVC build a NoSQL database, and then provide a Web interface for that database. Also, you should be able to securely login into this website. Make sure you can process so many data and if necessary, implement vertical or horizontal scaling.

You will adhere to coding standards (for this the **organizational documentation** is found at [LINK](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions) ensure proper commenting includes screenshots of debugging in a document and have a test table with screenshots.

You will need to identify which information repositories and organizational documents you have used.

Provide for your team: Seed\_data\_1.txt and Seed\_data\_2.txt given to be implemented through the code**.[PE1.1]**

 

Each student will use different data seed.

Development tools to be used: **[AC1-3]**

1. Internet connection to download all NuGet connectivity packages
2. Visual Studio For development
3. Visual Code for testing
4. Postman for code testing

Design requirements

**Your final design at the end should look like this:**

1. API: Name it xxxxSales.API. The example is given for the CarSalesAPI. You will replace XXXX with the product you are selling

|  |  |
| --- | --- |
|  | **[PE 2 ] [PE 2.1 ]**  **[AC 2.1]** |

1. Web App should have a login and register page (seed data are random and you can’t use the same ones we used in the class:

|  |  |
| --- | --- |
|  |  |

1. Explain the implementation of vertical or horizontal scaling.

(Please include this question in the report)

|  |  |
| --- | --- |
| To address the potential need for processing large volumes of data in the ITEMS project, the implementation of vertical or horizontal scaling is essential. Vertical scaling, or scaling up, involves enhancing the capacity of the existing server by adding additional resources, such as CPU, RAM, or storage, to improve performance. This is ideal for applications where a single, more powerful server can handle the increased load. Horizontal scaling, or scaling out, involves incorporating more servers to distribute the load, ensuring the system can handle more requests simultaneously. This is achieved by utilising a load balancer to distribute traffic across multiple servers, which is particularly beneficial for the NoSQL database and API endpoints. The choice between vertical and horizontal scaling will be based on the specific performance requirements and architecture of the application, with horizontal scaling generally providing more flexibility and resilience. Ensuring secure login and adhering to coding standards will be essential throughout this process to ensure data integrity and security. | [PE1.2] |

1. Explain why using NoSQL in this project is better than the traditional data model.

|  |  |
| --- | --- |
| Using NoSQL in the ITEMS project is advantageous over traditional relational databases due to its flexibility, scalability, and performance in handling large volumes of unstructured or semi-structured data. NoSQL databases, such as MongoDB, are designed to efficiently manage large amounts of data across distributed systems, making horizontal scaling effortless and cost-effective. This is particularly advantageous for the API and web interface components that may experience fluctuating loads and require quick read/write operations. Additionally, NoSQL's schema-less nature allows for easier modifications and adaptations to evolving data requirements without the need for extensive schema revisions, enabling faster development cycles and responsiveness to business requirements. This flexibility, combined with high availability and fault tolerance features inherent in many NoSQL solutions, makes it a superior choice for the dynamic and scalable requirements of the ITEMS project. | [PE1.3] |

1. Propose one vendor for our client to use and why this one should benefit their business

|  |  |
| --- | --- |
| I highly recommend MongoDB as the vendor for the ITEMS project due to its robust feature set, flexibility, and scalability. MongoDB is a widely used NoSQL database that provides document-oriented storage, which allows for the storage of complex data structures in a flexible, JSON-like format. This flexibility makes it effortless to handle evolving data models without requiring extensive schema changes. Furthermore, MongoDB's horizontal scaling capabilities, such as sharding, enable the database to handle large volumes of data and high traffic loads efficiently. Its powerful query language and indexing capabilities ensure a high performance for both read and write operations. Additionally, MongoDB Atlas, the fully managed cloud service, offers automated backups, security features, and performance monitoring, reducing the operational burden for the client. These attributes make MongoDB a reliable and efficient choice, aligning well with the business needs of handling large, dynamic data sets with ease and ensuring high availability and scalability. | **[pe1.4]** |

1. Create one data table with all data types you will use in the project.

|  |  |
| --- | --- |
| [  {  " **\_id**": ObjectId,  "Category": String,  "Name": String,  "StoreLocation": String,  "PostCode": Int32,  "Price": Int32,  "IsAvailable": Boolean  }  ] |  |

TASK 1: Plan your app

1. Build an app where users can purchase products. This app should have one running API which will enable users to search for products using different criteria (most important your Web/app and Web/API are written in MVC framework). **MONGO NOSQL MUST BE USED to**:
   1. Get all products.

|  |  |
| --- | --- |
|  | **[PE1.1] [PE2.1]** **[AC 2.1]** |

* 1. Return a list of items

|  |  |
| --- | --- |
|  | [PE1.1] [PE 2.1 ] |

* 1. Returning multiple items

|  |  |
| --- | --- |
|  | [PE1.1]  [pe2.3] |

* 1. Error handling

|  |  |
| --- | --- |
|  |  |

* 1. Making all calls to your API Async

|  |  |
| --- | --- |
|  | **[PE1.2]** |

* 1. Write data using HTTP methods.

|  |  |
| --- | --- |
|  | **[PE1.1]** [PE2.1] |

* 1. Add data using the POST method.

|  |  |
| --- | --- |
|  | **[PE1.1] [PE2.1]** |

* 1. Update one record using the PUT method.

|  |  |
| --- | --- |
|  | **[PE1.1] [PE1.2] [PE2.1]** |

* 1. Delete one or many records using DELETE.

|  |  |
| --- | --- |
|  | **[PE1.1] [PE1.2] [PE2.1]** |

1. For this application you will receive DUMMY data from your lecturer (many versions/one version per student)
2. This application should be designed to Search and retrieve complex data.

|  |  |
| --- | --- |
|  | **[PE1.1] [PE1.2] [PE2.1]** |

1. Filtering data statements should be used like %oyo% where you will find all the results which contain those letters.

|  |  |
| --- | --- |
|  | **[PE1.1] [PE1.2]** |

1. Searching the web with null values or by some criteria

|  |  |
| --- | --- |
|  |  |

1. Sorting data is important, and you can choose how to sort your data

|  |  |
| --- | --- |
|  | **[PE1.1] [PE1.2]** |

1. Versioning your API to run only available products.

|  |  |
| --- | --- |
| Hint: Screenshot | [**PE1.3] [PE1.4] [PE2.2]** |

1. Decide how your HTTP Header will be called and implement that (one of those below):
   1. APIVersionReader or

|  |  |
| --- | --- |
| Hint: Screenshot | **[PE1.3] [PE1.4] [PE2.2]** |

* 1. Using Query String

|  |  |
| --- | --- |
| Hint: Screenshot | **[PE1.3] [PE1.4] [PE 2.2]** |

1. Securing your API is crucial, implement your solution.

|  |  |
| --- | --- |
| Hint: Screenshot |  |

1. If the data storage you have selected a volatile one, what is the next step, this business need data consistency too?

|  |  |
| --- | --- |
| Hint: Screenshot | **[PE2.1]** |

Task 2: Create a Web App that will use data from our API

1. Our Web App should be able to read all the data from our API and present all products on the landing page: (make sure all your controllers are in the controllers section, models in the model section and views in the view section:

|  |  |
| --- | --- |
|  | [PE2.3] **[PE1.1] [PE1.2]** |

2.After enabling CORS and migrating your database adds identity to your application. Registration and login must be fully functional.

|  |  |
| --- | --- |
| Hint: Screenshot | **[AC1.1] [PE2.3] [PE1.1] [PE1.2] [PE1.3] [PE1.4]** |

Task 3: Test and fix the data persistence process according to business requirements

NOTE: YOUR TEST DATA WILL BE DIFFERENT

1. After completing all the above we need to document and finalize testing:
   1. Use POSTMAN and Visual Code to test your API

|  |  |
| --- | --- |
| Hint: Screenshot | **[PE1.1] [PE1.2] [PE1.3] [PE1.4]**  **[AC 1.1]** |

* 1. Open the web console and find any error which happened during your design:

|  |  |
| --- | --- |
|  | **[PE1.1] [PE1.2] [PE1.3] [PE1.4]** [PE2.3 ] |
|  |  |

1. Provide testing evidence that you have received a payload from the API in a JSON format:

|  |  |
| --- | --- |
|  | **[PE1.1]** |

1. In existing data try to search for one record which does not exist. Your code shouldn’t be 204 ==No Content

|  |  |
| --- | --- |
|  | **[PE1.1] [PE1.2]** |

1. Show a screenshot of a get request to receive all products:

|  |  |
| --- | --- |
|  | **[PE2.1] [PE1.1] [PE1.2] [PE1.3] [PE1.4]** |

1. Show a screenshot of a post request to create a new product:

|  |  |
| --- | --- |
|  | **[PE2.1] [PE1.1] [PE1.2] [PE1.3] [PE1.4]** |

1. Show a screenshot of a put request to update a new product:

|  |  |
| --- | --- |
|  | **[PE2.1] [PE1.1] [PE1.2] [PE1.3] [PE1.4]** |

1. Show 2 screenshot of filtering data through web parameters:

|  |  |
| --- | --- |
|  | **[PE 2.1 ] [PE1.1] [PE1.2] [PE1.3] [PE1.4]** |
|  |  |

1. Show a get request from version 1 and version 2 of your API.

|  |  |
| --- | --- |
| Hint: Screenshot 1 | **[PE1.3] [PE1.4]** |
| Hint: Screenshot 2 |  |

1. Show a screenshot of a get request from the version 2 API using a header.

|  |  |
| --- | --- |
|  | **[PE2.1][PE2.2]PE2.3] [PE1.1] [PE1.2] [PE1.3] [PE1.4]** |

1. Show a screenshot of get request using query strings or a action result.

|  |  |
| --- | --- |
|  | **[PE1.1] [PE1.2] [PE1.3] [PE1.4]** |
|  |  |