|  |  |  |
| --- | --- | --- |
| **Qualification Details** | | |
| **Training Package Code & Title** | ICT Information and Communications Technology Training Package (Release 8.1) | |
| **Qualification National Code & Title** | | **State code:** |
| ICT50220 - Diploma of Information Technology (Advanced Programming) (Release 2) | | **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** | | | AT1- Knowledge Based Assessment | | | | | | |
| **Assessment date** | | | Weeks 1-8 | | | | | | |
| **Student Name** | | | Samuel Bailey | | | | | | |
| **Student ID** | | | 30106121 | | | | | | |
| **Student Declaration** | | | I have read and understand 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 own work.  SBailey……………….. | | | | | | |
| **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 ASSESSORS AND STUDENTS** | | |
| **TO THE ASSESSOR**  Type of Assessment | AT1 Written – Knowledge | |
| Duration of Assessment | 8 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 | | As detailed in the assessment plan | |
| **TO THE STUDENT** | |  | |
| Purpose of the Assessment | | 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: | |
|  | | *ICTPRG554* | |
|  | | * 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 * authorisation 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 authorisation * 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.   *ICTPRG556*   * *language used in object-oriented (OO) programming* * *HTTP protocol* * *functions and features of debugging and testing tools* * *principles of model view controller design pattern, conventions and architecture, including:*  1. *scalability* 2. *maintainability* 3. *reusability*  * *features, structures, logic and modes of interactions between models, controllers and views, including:* * *HTTP Request/Response and redirects* * *HTTP request handlers, routes and parameters* * *Query strings and key/vale pairs* * *model binding* * *convention over configuration* * *HTML language, templates and dynamic rendering* * *view models and data models.* | |
| Assessment Conditions | | Complete the following assessment in class or at home.  All documentation must be submitted via Blackboard, including proof of individual work in places where directed.  In order to verify the authenticity of the your assessment, your lecturer will ask you to provide an answer to a question that you have already submitted. | |
|  | | |

|  |  |
| --- | --- |
| **INFORMATION FOR STUDENTS** | |
| Purpose of Assessment | This research assessment is designed to encompass all the knowledge acquired throughout the semester. It encourages investigation into emerging trends and technologies within the NoSQL and MVC domains, focusing on their analysis in relation to NoSQL and MVC. Additionally, the assessment must adhere to APA 6th edition referencing guidelines.  This assessment also covers knowledge questions related to:   * 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 * authorisation 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 authorisation * different noSQL data store formats, including: |
| 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 are acceptable – but work submitted must be your own work.  All external sources used (images, websites, articles, or otherwise) must be referenced using APA7th formatting.  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:  Task 1 - Answer Knowledge Questions |

*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

|  |  |
| --- | --- |
| **Task 1 – Answer Knowledge Questions** | **Check** |
| Completely filled all research, and knowledge questions |  |

**Introduction:**

This assessment is a case study report which encompasses all knowledge learnt until this point when assessed and promotes research on upcoming trends and technologies within the NoSql/MVC field in relation to data capture, store, analyse and validate.

**Scenario**

As part of your pre-employment training with CITE MS Pty Ltd, you'll be focusing on NoSQL and MVC technologies. This training is designed to enhance your learning experience and deepen your understanding of these technologies, which are crucial for improving your future job prospects.

In this assessment, you will explore how NoSQL and MVC can be effectively utilized to capture and store data, benefiting the organization. Given that NoSQL offers flexibility and is suitable for scenarios where consistency is not the primary concern, you will investigate how these features can be leveraged for organizational advantage. The training will help you grasp how to apply these technologies in real-world scenarios, aligning with CITE MS Pty Ltd’s focus on innovative data management solutions.

*Continue to the next page*

**Information search preparation and activities**

This activity will demonstrate your initial research and process of obtaining information. This will show where you have sourced materials and enable you to solidify answers to below topics and the subtopics associated.

**Note for Future Me: Bolded and Blue text is information to use in your answer. It is copied and pasted.**

## Answer Knowledge Questions - Task 1

1. Data persistence refers to the method of storing data in a manner that allows it to be retained across sessions, reboots, or application restarts. This concept ensures that information remains available and consistent over time, regardless of the state of the system.

List 6 benefits of data persistence: **[ICTPRG554 KE1.1]**

|  |
| --- |
| * + 1. Durability.     2. Availability.     3. Data Recovery     4. Data Consistency     5. Increased Performance     6. Historical Tracking and Versioning |

1. Explain all 6 benefits using from the previous question.

**[ICTPRG554 KE1.1]**

|  |
| --- |
| * + 1. Durability: Data remains even after the system is restarted, crashes or |

1. Comparing NoSQL Schema-Free Models and Relational Data Models with Focus on the CAP Theorem.
   1. Explain :NoSQL Schema-Free-Model
   2. CAP Theorem **[ICTPRG554 KE1.1 ]**

|  |
| --- |
| NoSQL Schema Free   * Data doesn’t need a fixed schema or definition * **Key Features:** * **Flexible structure: Each record (or document) can have a different set of fields.** * **Easy to evolve: You can add or remove fields without altering the entire database structure.** * **Best suited for unstructured or semi-structured data (e.g., JSON, XML).** * **Supports various types of NoSQL databases: Document-based, key-value, column-family, and graph databases.**   CAP Theorem   * **Key Features:** * **Flexible structure: Each record (or document) can have a different set of fields.** * **Easy to evolve: You can add or remove fields without altering the entire database structure.** * **Best suited for unstructured or semi-structured data (e.g., JSON, XML).** * **Supports various types of NoSQL databases: Document-based, key-value, column-family, and graph databases.** |

1. Explaining the Difference Between Scaling Out and Scaling Up (Horizontal vs. Vertical Scaling)
2. Horizontal Scaling
3. Vertical Scaling **[ICTPRG554 KE1.2]**

|  |
| --- |
| Horizontal Scaling (Scaling Out)  **Definition:**  **Horizontal scaling means adding more machines or nodes to your system to handle increased load. Instead of making one server more powerful, you add more servers to work together.**  **Key Characteristics:**  **Adds more servers/nodes to distribute the load.**  **Often used in cloud and distributed systems.**  **Improves availability and fault tolerance (if one node fails, others can take over).**  **Typically more complex to manage (requires load balancers, data replication, etc.).**  **Example systems: NoSQL databases like Cassandra, web servers behind a load balancer.**  **Example: If your application is slow due to high traffic, you add two more web servers, and place a load balancer in front of them to distribute requests.**  Vertical Scaling (Scaling Up)  **Definition:**  **Vertical scaling means adding more power (CPU, RAM, storage) to an existing server to improve performance.**  **Key Characteristics:**  **Increases the capacity of a single machine.**  **Easier to implement (no need for distributed setup).**  **Limited by hardware – there’s only so much CPU/RAM you can add.**  **If the server fails, the whole system may go down.**  **Common in relational databases or legacy systems.**  **Example: Upgrading your server from 16 GB RAM to 64 GB RAM and from 4 CPUs to 16 CPUs to handle more operations.** |

1. What Programming Languages Can Be Used for NoSQL Applications? **[ICTPR554-KE1.3][ICTPRG556KE2.1]**

|  |
| --- |
| **You can use almost any major programming language to build NoSQL applications, depending on the database and the application’s needs. Most NoSQL databases provide official drivers or client libraries for multiple languages.** |

1. Understanding Partitioning in a NoSQL Environment and Its Reflection on the CAP Theorem. Explain in your own words **[ICTPRG554 KE1.4]**

|  |
| --- |
| **What is Partitioning in NoSQL?**  **Partitioning (also called sharding) in a NoSQL environment means dividing data across multiple machines or nodes so that no single machine has to store all the data. This is done to improve scalability, performance, and fault tolerance.**  **Think of it like splitting a big library into different branches — each branch holds part of the books based on category or title.**  **In NoSQL databases, data is often partitioned by a key, and each partition is stored on a different server or cluster node.**  **Why Partitioning Affects CAP:**  **Partitioning is natural and necessary in NoSQL systems because they deal with huge amounts of data and must run across multiple servers.**  **But if one partition (server) goes down or becomes unreachable (a network partition), the database must make a choice:**  **Choice 1: Prioritize Availability**  **Let the system continue responding to requests even if some data is outdated or missing.**  **This sacrifices Consistency.**  **Example: Cassandra, DynamoDB → AP systems.**  **Choice 2: Prioritize Consistency**  **Block access until all partitions are in sync again.**  **This sacrifices Availability.**  **Example: HBase, MongoDB (in some settings) → CP systems.** |

1. Functions and Features of Time-to-Live (TTL) Requirements. Explain:
   1. Automatic Data Expiration
   2. Cashe Management and Performance Optimisation  **[ICTPRG554 KE1.5]**

|  |
| --- |
| **Time-to-Live (TTL) is a timer that decides how long data should exist before it disappears or refreshes.**  **Automatic Data Expiration means that data is automatically deleted or marked invalid once its TTL value expires. Each piece of data is assigned a lifespan (e.g., 60 seconds, 1 hour, or 7 days). After that period, the data is removed or refreshed without manual intervention.**  **TTL plays a major role in cache systems (like web servers, CDNs, or databases) by determining how long data stays cached before being refreshed or replaced. This balances speed and data accuracy.**  **Functions and Benefits:**   * **Improves performance: Frequently accessed data stays in cache, reducing the need to repeatedly fetch it from the main server or database.** * **Reduces server load: By serving cached data within the TTL period, network and processing demands are greatly reduced.** |

1. Authorization and Authentication Procedures and Levels of Responsibility According to Client Access Requirements. Explain:
   1. Authentication Procedures
   2. Authorization Procedures

**[ICTPRG554KE1.6]**

|  |
| --- |
| **The question is about how systems control access — who gets in, and what they can do once they’re in. It mentions two key concepts:**   1. **Authentication → proving *who you are*** 2. **Authorization → controlling *what you’re allowed to do***   **It also mentions “levels of responsibility according to client access requirements,” which just means:**  **Different people (or clients) get different levels of access depending on their role or need.**  **Passwords – the most basic method.**  **PINs or passphrases – short codes for identification.**  **Biometric data – fingerprints, face scans, or voice recognition.**  **Two-Factor Authentication (2FA) – using something extra, like a one-time code or mobile app confirmation.** |

1. Framework for NoSQL and MVC Storage with Multi-Partition Usage. Explain that framework /collection:

**[ICTPRG554 KE1.7]**

|  |
| --- |
| **Authorization is what happens after authentication — it determines what you can do inside the system.**  **How it’s done: The system checks your role, permissions, or security level to decide what data or functions you can access.**  **Example: In a company system:**   * **A regular employee can view their own timesheet.** * **A manager can view and edit their team’s timesheets.** * **An administrator can change system settings.** |

1. Testing Methodology for MVC with Azure NoSQL and Tools Used. Explain:
   1. Integration Testing
   2. Azure Cosmos DB Emulator **[ICTPRG554 -AC1.8 ][ICTPRG556 -KE2.3]**

|  |
| --- |
| **Integration testing is a type of software testing where you check how different parts of your system work together. In an MVC app, that means testing how the Model, View, and Controller layers interact — and how they connect to external services like databases (Cosmos DB) or APIs. In simpler terms:**  **Instead of testing one piece of code by itself, you test how multiple pieces fit and function together.**  **The Azure Cosmos DB Emulator is a local tool provided by Microsoft that lets you develop and test apps that use Cosmos DB — without connecting to the real Azure cloud. Think of it as a practice version of the real Cosmos DB that runs on your computer.**  **What It Does**   * **It mimics the behavior of Azure Cosmos DB.** * **Lets you create databases, containers, and queries locally.** * **Supports the same APIs (like SQL, MongoDB, Cassandra, Gremlin, etc.) as the cloud version.** * **Allows testing and debugging your app offline or before deployment.** |

1. Handling the Collection, Storage, and Sorting of Large Amounts of Data Using Keys in NoSQL Storage. Explain:
   1. Data Collection
   2. Data Storage
   3. Data Modelling **[ICTPRG445 -KE1.9]**

|  |
| --- |
| **Data collection is the process of gathering data from different sources before it’s stored in a NoSQL database.**  **In large systems, this data can come from:**   * **User interactions (apps, websites, sensors)** * **APIs and web services** * **IoT devices** * **Logs and analytics**   **How It Works in NoSQL Systems**   * **Data is often unstructured or semi-structured (JSON, documents, logs).** * **Systems use pipelines or data ingestion tools (like Azure Data Factory, Kafka, or APIs) to collect and send data into the database.** * **Each piece of data usually includes a key — like an ID — to make it easy to find and group later.**   **Example**  **In a shopping app:**   * **Every customer action (view, click, purchase) is collected.** * **Each record is tagged with a unique key (e.g., UserID or OrderID).** * **Data is sent in JSON format to the NoSQL database for fast storage and access.**   **What It Means**  **Data storage in NoSQL systems refers to how the collected data is saved across servers or databases for quick access, scalability, and reliability.**  **Unlike traditional SQL databases, NoSQL:**   * **Doesn’t rely on fixed tables or schemas.** * **Stores data as key-value pairs, documents, columns, or graphs.** * **Can handle massive data volumes by distributing it across multiple machines (horizontal scaling).**   **How Keys Are Used**   * **Each record is stored with a unique key that identifies it.** * **This key acts like a label or address to find that data instantly.** * **For example, a document with UserID: 1234 can be retrieved in milliseconds using that key.**   **Data modelling is the process of designing how your data is structured and related in the NoSQL database — deciding what keys, collections, and relationships exist.**  **Since NoSQL databases don’t use tables and rows like SQL, you design your model around how your application will access the data, not just how it’s stored.**  **Key Concepts**   * **Primary key: A unique ID for each document (e.g., UserID, OrderID).** * **Partition key: Used to group related data and distribute it evenly across servers.** * **Denormalization: Instead of splitting data into multiple tables, NoSQL often stores related data together for faster reads.** |

12) Concise a list of techniques and tools related to NoSQL databases and MVC (Model-View-Controller) architecture, focusing on data transport encryption, authentication, and authorization:   **[ICTPRG554- KE1.10]**

|  |
| --- |
| **Data Transport Encryption**   * **Techniques : TLS Encryption, HTTPS Protocol** * **Tools: Tls Certificates**   **Authentication**   * **Techniques: Username and Password, OAuth 2.0** * **Tools: ASP.NET Identity**   **Authorization** |

13) NoSQL databases handle various data formats, each designed to meet different requirements for scalability, flexibility, and performance. List the primary NoSQL data formats:

**[ICTPRG554 KE1.11 ]**

|  |  |  |
| --- | --- | --- |
|  | Explanation 1  (example) | Explanation 2  (example) |
| 1. Key value |  |  |
| 1. Document based |  |  |
| 1. column based |  |  |
| 1. graph based |  |  |

14)Explain different NoSQL data formats: **[KE1.11 ]**

|  |  |  |
| --- | --- | --- |
|  | Explanation 1  (example) | Explanation 2  (example) |
| 1. numeric |  |  |
| 1. string |  |  |
| 1. Boolean |  |  |
| 1. complex |  |  |
| 1. date time. |  |  |

15) Purpose and Function of the HTTP Protocol. Explain:

a. Request-response model

b. Stateless Communication

c. Resource Access

d. Header and Body Handling

e. Status Codes

f. Protocol Layers

**[ICTPRG556 -KE2.1]**

|  |
| --- |
|  |

16) Explain principles of MVC, conventions and architecture including: [ICTPRG556-KE2.4]

|  |  |
| --- | --- |
| 1. scalability |  |
| 1. maintainability |  |
| 1. reusability |  |

17) Explain interactions between models, controllers and views including: [ICTPRG556-KE2.5]

|  |  |  |
| --- | --- | --- |
|  | Explanation  (example) | ICTPRG556 |
| 1. HTTP Request/Response and redirects |  | **[KE2.5.1]** |
| 1. HTTP request handlers, routes, and parameters |  | **[KE2.5.2]** |
| 1. Query strings and key/vale pairs |  | **[KE2.5.3]** |
| 1. Model binding |  | **[KE2.5.4]** |
| 1. convention over configuration |  | [KE2.5.5] |
| 1. HTML language, templates and dynamic rendering |  | [KE2.5.6] |
| 1. view models and data models. |  | [KE2.5.7] |