**900102-000-00-KM-05, REST API and Modularization, NQF Level 4, Credits 2**

**Summative Assessment Memo**

**Module Five (5`)**

|  |  |
| --- | --- |
| **Module Code** | 900102-000-00-KM-05 |
| **NQF Level** | 4 |
| **Credits** | 2 |
| **Skills Programme ID Number** | SP- 220329 |
| **Curriculum Title** | Java Programmer |
| **Curriculum Code** | 900102-000-00-00 |

|  |  |
| --- | --- |
| **Name** |  |
| **Contact Address** |  |
| **Telephone (H)** |  |
| **Telephone (W)** |  |
| **Facsimile** |  |
| **Cellular** |  |
| **E-mail** |  |

**Note to the learner**

This Learner Guide provides a comprehensive overview of the module. It is designed to improve the skills and knowledge of learners, and thus enabling them to effectively and efficiently complete specific tasks.

**Purpose of the Module**

The main focus of the learning in this knowledge module is to build an understanding of the functionalities of REST API and modularization and when to use them.

The learning will enable learners to demonstrate an understanding of:

* KM-05-KT01: REST API (Application Programming Interface) for Java 50%
* KM-05-KT02: Java Modularization 50%

**Provider Accreditation Requirements for the Knowledge Module**

**Physical Requirements:**

* The provider must have lesson plans and structured learning material or provide learners with access to structured learning material that addresses all the topics in all the knowledge modules as well as the applied knowledge in the application.
* QCTO/ MICT SETA requirements

**Human Resource Requirements:**

* Qualification of lecturer (SME):
* NQF 5 qualified in industry recognised qualifications with 1 year experience in the IT industry o Cybersecurity vendor certification
* Assessors and moderators: accredited by the MICT SETA

**Legal Requirements:**

* Legal (product) licences to use the software for learning and training
* OHS compliance certificate

**Exemptions**

* RPL based

**Venue, Date and Time:**

Consult your facilitator should there be any changes to the venue, date and/or time.Refer to your timetable.

**Assessments**

**Integrated Formative Assessment:** The skills development provider will use the curriculum to guide them on the stipulated internal assessment criteria and weighting. They will also apply the scope of practical skills and applied knowledge as stipulated by the internal assessment criteria. This formative assessment leads to entrance into the integrated external summative assessment.

**Integrated Summative Assessment**: An external integrated summative assessment conducted through the relevant QCTO Assessment Quality Partner is required to issue this qualification. The external integrated summative assessment will focus on the exit level outcomes and associated assessment criteria.

**Skills Programme Purpose**

A Java Programmer will be able to implement solutions to solve real-life problems in an efficient manner, applying a knowledge and understanding of the principles of programming with Java and applicable tools. Tasks that the learner will be able to know, do and understand after achievement of the skills programme include:

* Create well-written and readable Java programs, using a disciplined coding style, including documentation and indentation standards.
* Use Git functionalities for working collaboratively in a team and execute version control.

**Skills Programme Rationale**

Realising the importance and future impact of the Fourth Industrial Revolution (4IR) on the economy of South Africa and its competitiveness, the Minister of Communications gazetted the Presidential Commission on the Fourth Industrial Revolution (PC4IR) on 9 April 2019. By March 2020 this Commission delivered a report with wide ranging recommendations for Human Capital Development that will drive the 4IR forward. It clearly indicated the speed at which companies will have to invest in big data analysis, web-enabled market investment and the use of cloud computing and machine learning.

Software development is central to these initiatives. Software developers are the creative minds behind computer programs. Some develop the applications that allow people to do specific tasks on a computer or another device. Others develop the underlying systems that run the devices or that control networks. The software developer is the important cog in designing advanced computerised technologies. South Africa has a scarcity of software developers and there is a clear need for a qualification focusing specifically on the training and education of software developers.

**Entry Requirements**

Grade 11 with Maths Lit and English.

Access to equipment, internet connectivity and how to work remotely

**EXIT LEVEL OUTCOMES**

**Exit Level Outcomes (ELO) 1**

Describe the basics of Java Programming

Associated Assessment Criteria (AACs)

* The fundamentals of the Java programming language are explained.
* The basic concepts and methods of object-oriented programming and object-oriented design are described.
* The development life-cycle as a means of creating applications is described.

**Exit Level Outcomes (ELO) 2**

Programme effectively using Java frameworks and functionalities

Associated Assessment Criteria (AACs)

* Java syntax is demonstrated, using the Java API.
* Well-written and readable Java programs are created, using a disciplined coding style, including documentation and indentation standards.
* Problems with application development are addressed by troubleshooting.

**Exit Level Outcomes (ELO) 3**

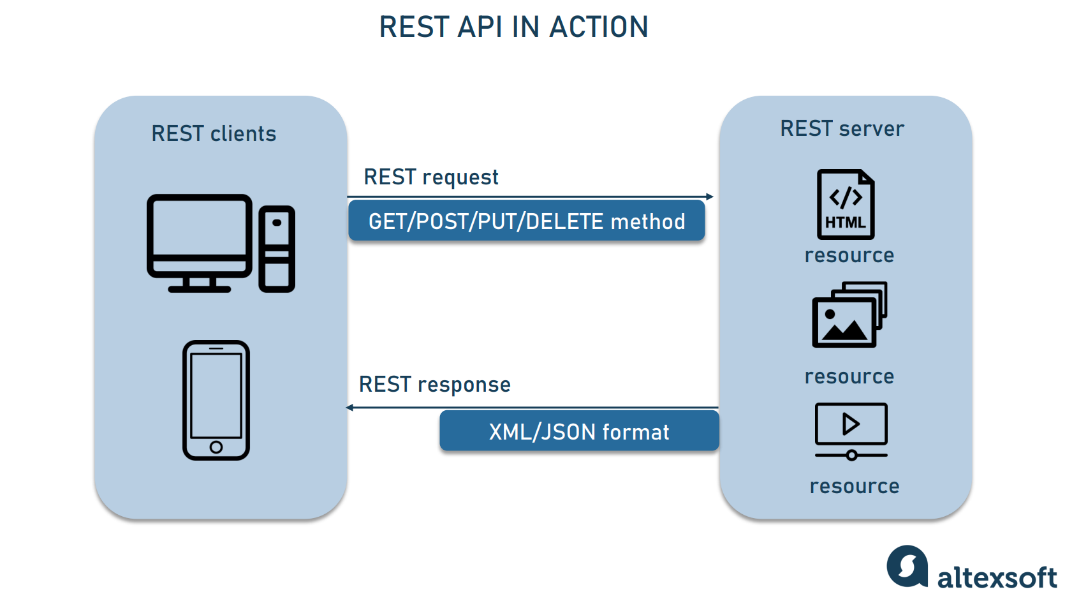
Work collaboratively in a team using GitHub platform

Associated Assessment Criteria (AACs)

* An ability to work with GitHub is demonstrated.
* Working in a team collaboratively is achieved by using GitHub.
* Version control is exercised using GitHub. functionalities such as repositories, branches, commits and pull requests

**Session 1:** **KM-05-KT01: REST API (Application Programming Interface) for Java**

1. **IAC0101 Explain the understanding of REST API - 10000 feet overview**



Understanding REST API at a high level, often referred to as a "10,000 feet overview," involves grasping the fundamental concepts and principles behind RESTful architecture without delving into detailed technical specifics. Here's a high-level overview of REST API:

**1. REST Defined:**

* REST stands for Representational State Transfer.
* It is an architectural style for designing networked applications, emphasizing simplicity, scalability, and statelessness.

**2. Resources:**

* In REST, everything is considered a resource, which can be any piece of data or functionality.
* Resources are identified by unique URIs (Uniform Resource Identifiers), which are analogous to URLs (Uniform Resource Locators).

**3. HTTP Methods (Verbs):**

* REST uses standard HTTP methods (also known as HTTP verbs) to perform actions on resources.
* Common HTTP methods include GET (retrieve), POST (create), PUT (update), and DELETE (delete).

**4. Stateless:**

* RESTful systems are stateless, meaning each request from a client to the server must contain all the information needed to understand and process the request.
* Servers do not store client state between requests.

**5. Representations:**

* Resources can have multiple representations (e.g., JSON, XML) to accommodate various client preferences.
* Clients specify their preferred representation using the **Accept** header in the request.

**6. Uniform Interface:**

* REST APIs provide a uniform and consistent interface for interacting with resources.
* A standard set of operations, such as GET for retrieval and POST for creation, simplifies client interactions.

**7. Status Codes:**

* HTTP status codes indicate the outcome of a client's request. For example, 200 means OK, 201 means Created, and 404 means Not Found.
* Status codes help clients understand the result of their actions.

**8. Stateless Servers:**

* RESTful servers do not maintain client state between requests. Each request is independent.
* Stateless design simplifies server architecture and supports scalability.

**9. Hypermedia (Optional):**

* Hypermedia as the Engine of Application State (HATEOAS) is an optional feature in REST APIs.
* It allows resources to include links to related resources, making API discovery and navigation easier.

**10. Practical Use Cases:** - REST APIs are widely used in web services, mobile app backends, and IoT (Internet of Things) applications. - They provide a flexible and efficient means of communication between clients and servers.

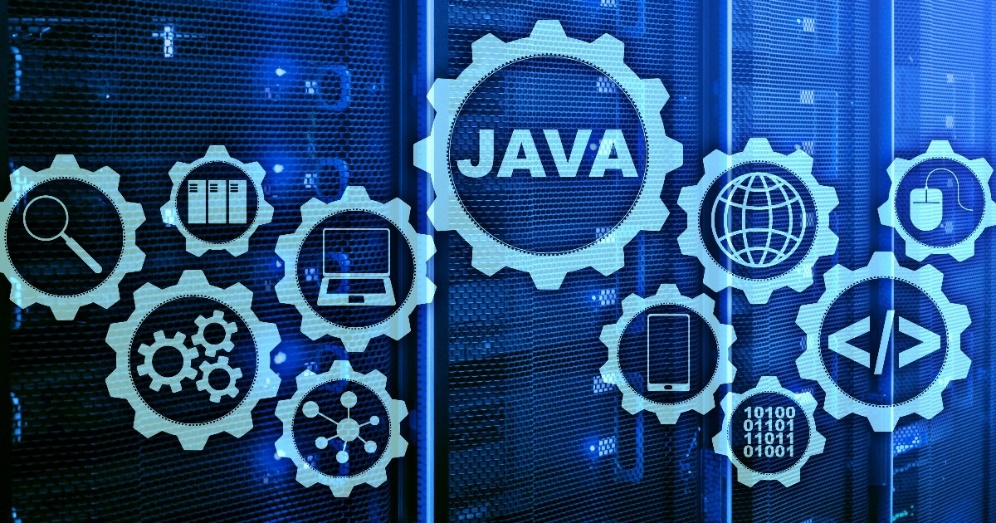
**11. Security and Authentication:** - REST APIs often implement security measures, including authentication and authorization, to protect resources. - Authentication mechanisms like OAuth and JWT are commonly used.

**12. Versioning (Optional):** - APIs may include versioning to manage changes and updates to resource representations over time. - Version information can be included in the URI or request headers.

This high-level overview captures the core principles and characteristics of REST API architecture. In practice, building and consuming RESTful services involves more detailed considerations, such as choosing appropriate HTTP status codes, designing resource URIs, handling request and response headers, and implementing secure communication. However, this overview provides a solid foundation for understanding the essence of RESTful design.

**Session 2:** **KM-05-KT02 :Java Modularization**

1. **IAC0201 Describe the Java New Features**
2. **Local Variable Type Inference**
3. **Switch Expression**
4. **Text Blocks**
5. **Records**

****

Java has introduced several new features and enhancements in recent versions to improve code readability, maintainability, and developer productivity. Here are some of the notable features introduced in Java:

1. **Local Variable Type Inference (var)**:
   * Starting with Java 10, you can use the **var** keyword to declare local variables with inferred types. The type is determined by the compiler based on the assigned value.
   * This feature reduces verbosity and enhances code readability for local variables, especially when working with complex types.

var name = "John"; // Inferred as String var numbers = List.of(1, 2, 3); // Inferred as List<Integer>

1. **Switch Expression (Java 12+)**:
   * In Java 12 and later versions, the traditional **switch** statement is enhanced to support switch expressions.
   * Switch expressions allow you to assign a value to a variable directly based on the result of a switch case.

int dayOfWeek = switch (day) { case "Monday", "Tuesday", "Wednesday", "Thursday", "Friday" -> 1; case "Saturday" -> 2; case "Sunday" -> 3; default -> throw new IllegalArgumentException("Invalid day"); };

1. **Text Blocks (Java 13+)**:
   * Java 13 introduced text blocks, a new way to write multi-line strings with improved readability and reduced escape sequences.
   * Text blocks are enclosed in triple double-quotes **"""** and maintain the formatting within the block.

String html = """ <html> <body> <p>Hello, world!</p> </body> </html> """;

1. **Records (Java 16+)**:
   * Records are a concise way to declare simple data classes, reducing boilerplate code.
   * Records automatically generate constructors, **equals()**, **hashCode()**, and **toString()** methods based on the declared fields.

record Point(int x, int y) {} Point p1 = new Point(1, 2);

These features contribute to making Java code more concise, expressive, and readable. They enhance developer productivity by reducing the need for boilerplate code and allowing developers to focus on the logic of their applications. It's important to note that the availability of these features depends on the Java version you are using, so you should check the Java version compatibility for each feature.