EXTERNAL INTEGRATED SUMMATIVE ASSESSMENT (EISA)													
QUALIFICATION	**OCCUPATIONAL CERTIFICATE: SOFTWARE DEVELOPER**												
SAQA ID	**118707**												
NQF LEVEL	**5**												
CREDITS	**220**												
DURATION:	**120 MINS**												
TOTAL MARKS:	**100**												
PASS MARK:	**80**												
SURNAME													
NAMES													
ID NUMBER													
EISA REGISTRATION NUMBER													

INSTRUCTIONS TO CANDIDATES

1. Read all instructions carefully before attempting the assessment
2. This is an individual assessment. No communication or collaboration with other candidates is permitted
3. All responses must be typed in a Microsoft Word document
4. Answer all questions as instructed. Ensure that your responses are:
    - Clear and well-structured
    - Relevant to the question
    - Written in a professional manner
5. Where applicable, include:
    - Code snippets
    - Explanations of your approach
    - Comments within your code
6. Save your work regularly. The invigilator will provide instructions regarding:
    - File naming conventions
    - Saving location
    - Submission process
7. Only the final submitted version of your document will be marked.
8. Unauthorised resources or assistance are not permitted, unless explicitly stated
9. If you experience any technical difficulties, notify the invigilator immediately.
10. Manage your time effectively. Ensure that you review your work before submission

GOODLUCK!

ELO 2: Build a logical flow using industry standard frameworks and methodologies to propose possible solutions to business challenges to meet both functional and technical requirements.

Answer all questions that follow

Task 1

Scenario - SmartBridge Solutions — Staff Leave Management System

SmartBridge Solutions is a mid-sized software development company that builds custom business systems for corporate clients. The company has been contracted by one of their long-standing clients, Nexus Financial Services, to develop a new Staff Leave Management System (SLMS).

Nexus Financial Services currently tracks staff leave using spreadsheets and paper forms. They need a system that allows HR administrators to capture employee records, record and approve leave applications, and generate basic leave reports. The development team at SmartBridge Solutions has completed the requirements gathering phase and has provided you with the system documentation.

You have been assigned as the Software Developer responsible for the database design, programming logic, debugging, testing, and deployment of this system.

System Requirements Summary

Area	Requirement
Employees	Store employee details: EmployeeID, FirstName, LastName, Email, Department, JobTitle, DateJoined, and AnnualLeaveBalance (days).
Leave Types	Support three leave types: Annual Leave, Sick Leave, and Family Responsibility Leave. Each type has a MaxDaysAllowed per year.
Leave Applications	Record each application: ApplicationID, EmployeeID (FK), LeaveTypeID (FK), StartDate, EndDate, NumberOfDays, Reason, Status, and ApprovedBy.

Status Values	Application Status must be one of: Pending, Approved, Rejected, or Cancelled.
Business Rules	NumberOfDays is calculated automatically from StartDate and EndDate. AnnualLeaveBalance must be reduced when leave is approved. A leave application cannot be submitted if the employee has insufficient leave balance.
Reporting	HR must be able to retrieve all pending applications, all approved leave per department, and the leave balance per employee.

The following is a data dictionary that shows the three tables in the SLMS database, their fields, data types, and rules.

Data Dictionary — SLMS Database

Table: Employees			
Field	**Data Type**	**Key / FK**	**Constraint**
EmployeeID	INT	PRIMARY KEY	AUTO INCREMENT
FirstName	VARCHAR(50)		NOT NULL
LastName	VARCHAR(50)		NOT NULL
Email	VARCHAR(100)	UNIQUE	NOT NULL
Department	VARCHAR(80)		NOT NULL
JobTitle	VARCHAR(80)		
DateJoined	DATE		NOT NULL
AnnualLeaveBalance	INT		DEFAULT 20

Table: LeaveTypes			
Field	**Data Type**	**Key / FK**	**Constraint**
LeaveTypeID	INT	PRIMARY KEY	AUTO INCREMENT

TypeName	VARCHAR(50)		NOT NULL
MaxDaysAllowed	INT		NOT NULL

Table: LeaveApplications			
Field	**Data Type**	**Key / FK**	**Constraint**
ApplicationID	INT	PRIMARY KEY	AUTO INCREMENT
EmployeeID	INT	FK → Employees	NOT NULL
LeaveTypeID	INT	FK → LeaveTypes	NOT NULL
StartDate	DATE		NOT NULL
EndDate	DATE		NOT NULL
NumberOfDays	INT		NOT NULL
Reason	TEXT		
Status	VARCHAR(20)		DEFAULT 'Pending'
ApprovedBy	VARCHAR(100)		

Answer the following questions.

QUESTION 1 **[30]**

QUESTION 1.1.1 **[14]**

Write SQL queries for each part below. Each query must work correctly with the tables shown in the data dictionary.

1. Write SQL query to retrieve the EmployeeID, FirstName, LastName, Email, and AnnualLeaveBalance for all employees. Display the results from the lowest leave balance to the highest. C (5)

   Take a screenshot that shows the SQL query and records retrieved and paste in word document

2. Write SQL query to retrieve all leave applications that have a Status of 'Pending' which must display the following fields: ApplicationID, EmployeeID, StartDate, EndDate, NumberOfDays, and Reason. C (4)

   Take a screenshot that shows the SQL query and records retrieved and paste in word document

3. Write SQL query to retrieve all records showing the total number of approved leave days per Department. HINT: You need to join the LeaveApplications and Employees tables. The SQL query must display the following fields: Department name and total approved days, from highest to lowest. C (5)

   Take a screenshot that shows the SQL query and records retrieved and paste in word document

The following information shows how the three tables relate to each other.

Table Relationships:

- One Employee can have many LeaveApplications (EmployeeID links the two tables)
- One LeaveType can be used in many LeaveApplications (LeaveTypeID links the two tables)
- A leave application must always be linked to a real employee and a real leave type.

QUESTION 1.2.1 **[16]**

Using your chosen database tool (e.g. MySQL, SQL Server, SQLite, or PostgreSQL), write a SQL script that does the following

1. Creates a database called SLMS_Nexus.

   Take a screenshot that shows the SQL script for creating the database and output and paste in word document (C) (2)

2. Creates all three tables — Employees, LeaveTypes, and LeaveApplications — using the exact field names, data types, and rules from the data dictionary. (C) (3)

   Take a screenshot that shows the 3 tables created and paste in word document

3. Set up all primary keys and foreign keys correctly.

   Take a screenshot in design view that shows primary and foreign keys set and paste in word document (C) (2)

4. Add a validation on leave application so that it cannot be saved without a valid employee and leave type.

   Take a screenshot that shows the validation condition on leave application and paste in word document (C) (2)

5. Add 4 sample records using the INSERT statement of employees from at least 2 different departments. (C) (5)

   Take a screenshot that shows sample 5 records added and paste in word document

6. Add sample records for all 3 leave types: Annual Leave, Sick Leave, and Family Responsibility Leave with the correct maximum days for each.

   Take a screenshot that shows the sample records showing leave types captured / selected and paste in word document. C (1)

7. Add at least 5 sample leave applications for different employees and statuses which must include at least one Pending, one Approved, and one Rejected. C (1)

   Take a screenshot that shows records showing leave applications and paste in word document

ELO: 3 Programme effectively using a suitable programming language to develop new solutions and update existing solutions.

Answer all questions that follow

Task 2

QUESTION 2 **[40]**

QUESTION 2.1.1 **[7]**

Scenario

The leave application module must be developed and must check that the employee has enough leave days available, work out how many days are being requested, save the application, and reduce the employee's leave balance if the application goes through.

1. **Plan how the leave application module will work before writing any code**

   Write pseudocode OR draw a flowchart in word document that shows the steps for the leaveApplication() function. It must cover: C (4)

- Input EmployeeID, LeaveTypeID, StartDate, EndDate, and Reason
- Check the employee's current leave balance
- Work out how many days are being requested (from StartDate to EndDate)
- If the employee has enough days — save the application with Status = 'Pending' and display a success message
- If the employee does not have enough days — reject the application and display their current balance

2. **Screen Design - sketch or description**

Draw or describe what the Leave Application screen will look like in word document. Display all the input fields, the submit button, and where the success or error message will appear.

C (3)

QUESTION 2.1.2 **[6]**

Scenario

The system requirement specifies that the system needs to count the number of working days between two dates. Public holidays and weekends (Saturday and Sunday) must not be counted.

1. **Writing algorithms**

Write a step-by-step algorithm for a function called calculateWorkingDays(startDate, endDate).
Your algorithm must:

- Take in a start date and an end date
- Go through each day from the start date to the end date
- Only count days, Monday to Friday
- Return the total number of working days counted
- Handle the situation where the end date is before the start date: return 0 or display an error

C (6)

You can write your algorithm as **pseudocode or plain steps**. Keep it clear — each step should be easy to follow.

QUESTION 2.1.3 **[9]**

1. Write the calculateWorkingDays function in your chosen programming language. Your code must:
   - Be written as a function
   - Take in a start date and an end date
   - Return the correct number of working days — Monday to Friday only
   - Handle the case where the end date is before the start date
   - Include THREE test cases that shows that your function works:

   C (9)

   - Take a screenshot that shows the function running and paste in word document

QUESTION 2.2 **[15]**

The following is a block of code in pseudocode in which you are required to identify and record any 5 errors in the table after the pseudocode.

Pseudocode

```
FUNCTION getLowBalanceEmployees
    conn = connectDatabase("SLMS_Nexus.db")
    query = "SELECT EmployeeID, FirstName, LastName, AnnualLeaveBalance"
    query = query + "FROM Employees"
    query = query + "WHERE AnnualLeaveBalance =< 5"
    results = conn.executeQuery(query)
    IF results.count = 0
```

```
        PRINT "No employees with low leave balance."
    ELSE
        FOR EACH employee IN results
            PRINTER "WARNING: Employee " + employee.EmployeeID + " " + employee.LastName
+ " has only " + employee.AnnualLeaveBalance + " days left"
        END FOR
    END IF
    conn.close
END FUNCTION
CALL getLowBalanceEmployees
```

Draw table as below in word document and write all 5 errors in the code

1

#	Line / Section	Error Type	What the Error Is	How to Fix It
1				
2				
3				
4				
5				

C (15)

QUESTION 2.3 **[3]**

2.

You will work with the following GitHub repository:

C (3)

https://github.com/engineer264/slms-nexus-starter

Instructions

Follow the steps below to update the project and save your changes back to the repository.

Step 1 — Clone the project
Step 2 — Open the project
Step 3 — Create a new branch
Step 4 — Make some changes
Step 5 — Save your work
Step 6 — Push your branch

Take screenshots of the following commands and execution output: clone, creating new branch committing and push commands and paste in your word document.

ELO 4 Test and maintain software and recommend improvements to ensure strong functionality and optimisation to meet both functional and technical requirements

Task 3

QUESTION 3	**[30]**
QUESTION 3.1.1	**[16]**

1.

Write unit tests for the calculateWorkingDays function (created in Task 2).

Use the testing tool that matches your programming language (e.g. unittest or pytest for Python, JUnit for Java, NUnit for C#, Jest for JavaScript). Write a test for each of the four cases below:

- Test 1: 5 days in a row — all weekdays (Monday to Friday). The answer should be 5.
- Test 2: 9 calendar days that cross two weekends. The answer should be 5 working days.
- Test 3: The start and end dates are the same weekday. The answer should be 1.
- Test 4: The end date is before the start date. The function should return 0 or handle the error.

C (8)

Take a screenshot showing the unit tests running and paste in your word document.

NOTE:

If you did not create the function earlier under Task 2, then create similar function calculateWorkingDays function in your chosen programming language and then attempt this task of unit testing as explained below.

Your code must:

- Be written as a function

- Take in a start date and an end date
- Return the correct number of working days — Monday to Friday only
- Handle the case where the end date is before the start date
- Include THREE test cases that display your function works:

Take a screenshot that shows the unit tests running.

2 Run all four tests and write down the results in the table below. NOTE: If you did not create the unit tests from previous question 1, then write **sample results** on the following table on unit testing in your word document.

#	Test Case	Expected Result	Actual Result	Pass / Fail
1	5 consecutive working days (Mon–Fri)			
2	9-day range spanning 2 weekends			
3	Same start and end date (weekday)			
4	End date before start date			

C (4)

3 How many of your four tests passed? C (1)

4 Did any test fail? If yes, explain what went wrong and what you would change to fix it. If all passed, write: "All tests passed." C (2)

5 Why is it important to test a function before using it in the full leave system? C (1)

QUESTION 3.1.2 **[8]**

1. Write an integration test that checks the full leave application process.

   Your test must check that:
   - A leave application is saved correctly when valid details are given.
   - The employee's leave balance goes down by the correct number of days.
   - The application is saved with a Status of 'Pending'.
   - The correct error message appears when the employee does not have enough leave days.

   C (4)

   Take a screenshot that shows the above integration aspects and paste in word document.

2. Run your integration test and record the result and explain whether the test passed or failed?
   If the test fails, explain what the error was and what you changed to fix it. If it passed, write: “Integration test passed.” C (4)

   Take a screenshot that shows the above integration aspects and paste them in word document. Add the words “Integration test passed.” If the integration passed OR your explanation below your screenshot of whether the test failed

QUESTION 3.2.1 **[6]**

1. Show that your full SLMS application runs and works.

   Do the following:
   - Start your application (run it on localhost or on a platform like Heroku, Render, or PythonAnywhere).
   - Open the Leave Application screen in a browser or terminal.
   - Submit a test leave application and confirm it is saved correctly in the database.
   - Take a screenshot that shows the application running.

   C (6)

GRAND TOTAL 100