**GENERAL INSTRUCTIONS:** Please carefully read the below instructions

The objective of this assessment is to check your ability to complete a project as per the provided “Project Design”.

**You are expected to –**

1. Write the source code for the classes, methods and packages **EXACTLY** as mentioned in the “**Project Design**” section.

2. Ensure that the names of the packages, classes, methods and variables **EXACTLY MATCH** with the names specified in the “Project Design” section.

3. Understand the project requirements and ACCORDINGLY WRITE the code and logic in the classes and methods so as to meet all given requirements.

**Creating the project and testing it –**

1. You are expected to create your project locally using eclipse (or any other IDE) on your desktop.

2. Once you are ready with the code, you should upload the src folder of your project in .zip format, using the “Upload Zip File” button.

IMPORTANT NOTE 1 : The extension of the zip file should be ONLY .zip (any other zip formats such as .7z will produce unexpected results)

IMPORTANT NOTE 2 : The .zip file should contain zip of ONLY the src folder structure from your project. (If the zip file has anything other than the src folder structure, the result will be unexpected. Do not zip the entire project folder structure. Just do the zip of the src folder structure and upload it)

IMPORTANT NOTE 3 : The name of the .zip file should be <your employee number>.zip For e.g., if your emp no. is 12345, the zip file should be named 12345.zip.

3. After uploading the zip file, you can click on “Compile & Test” button and the assessment engine will compile your source code and test it using its pre-defined test-cases.

4. If some of the test-cases fail, you can make the fixes in your source code locally on your desktop, and again repeat the above two steps.

5. Once you are finished with all the fixes, you can click on “Final Submission” button, which will show you the final result/score.

**NOTE that –**

6. The assessment engine will create objects and invoke methods as per the project design, and while doing so, it will use your packages, classes and methods. If your packages, classes and methods have a name mismatch or method prototype mismatch with respect to the expected “Project Design”, the tool will show it as an ERROR. If your packages, classes and methods match as per the names but do not perform the expected functionality, the tool will show it as a FAILURE.

7. Unless specified in the Project Design, DO NOT use **System.exit(0)** anywhere in your code. Using **System.exit(0)** in your project code will cause the CPC test engine to exit and it will not be able to run all test-cases.

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

**Flight Management System**

**Project Objective:**

Create a console based application, that will help the admin of a FMS to add new Flight details to the database .

**Note: This application will cover only limited functionalities for the assessment.**

**Project Design:**

**A. Database Design: You can skip this step if you have created this user earlier**

**1. Create a new user in database [ To be done in the backend by using sql commands ]**

a) **Note:** Do **NOT** use the default scott/tiger account of oracle for this project. You will have to create a new user in the below mentioned format.

b) Username/password : B<batchnumber><employeeid>

For example, if your batch number is **39806** and Employee number is **12345**, then the oracle user should be **B3980612345** and the password should be **B3980612345**

c) For JDBC connection, only use **orcl** as service name and **1521** as port number

**2. Steps for creating a new user**

a) Open command prompt

b) **Type** Sqlplus / as sysdba

c) **Create user <username> identified by <password>;** [ For example to create a user named “test” with password “test” : create user test identified by test; ]

d) **Grant connect,resource to <username>;** [ E.g: grant connect,resource to test;]

e) Commit;

f) Exit;

**3. Create Table [ To be done using sql commands, after logging-in as the new user that has been created in above step ]**

**Table Name:Flight\_Tbl[ Master Table ]**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Datatype** | **Constraints** | **Description** |
| **FlightId** | Varchar2(10) | Primary Key | This field will be automatically be created whenever a new flight gets added |
| **FlightName** | Varchar2(20) | Not Null | Name of the Flight |
| **Source** | Varchar2(25) | Not Null | Starting City |
| **Destination** | Varchar2(25) | Not Null | Destination City |
| **EconomySeats** | Number(2) | Not Null | Number of seats in Economy class |
| **BusinessSeats** | Number(2) | Not Null | Number of seats in Business class |
| **FirstClassSeats** | Number(2) | Not Null | Number of seats in First class |
| **FlightType** | Varchar2(20) | Not Null | International/Domestic |

**4. Create Sequence**

**Sequence Name : FlightId\_Seq**

|  |  |  |
| --- | --- | --- |
| **Sequence Name** | **Incremental value** | **Start Value** |
| **FlightId\_Seq** | 1 | 10000 |

**B. System Design: Server Side**

|  |  |
| --- | --- |
| **Name of the package** | **Usage** |
| com.wipro.flight.bean | Contains all the Bean Classes |
| com.wipro.flight.dao | Contains the DAO Class |
| com.wipro.flight.service | Contains the Service Class |
| com.wipro.flight.util | Contains the Database connection clas |

**Package: com.wipro.flight.bean**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables\*** | **Description** |
| **Flight** |  | **Bean Class** |
| String | flightID | **Primary Key, Auto Generated** (first 2 letters of name with 5 digits of auto-generated value from given sequence from dual table) |
| String | flightName | **Name of the Flight** |
| String | source | **Starting City** |
| String | destination | **Destination city** |
| int | economySeats | **Number of seats in Economy class** |
| int | businessSeats | **Number of seats in Business class** |
| int | firstClassSeats | **Number of seats in First Class** |
| String | flightType | **Could be International/Domestic** |

**\* All variables & method names are case sensitive**

**Package: com.wipro.flight.dao**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **FlightDAO** |  |  |
|  | **String getComputedId(String name, String seqName)** | Purpose: This method is used to compute the flightid, which will be a combination of **first 2 alphabetical letters of name** in **uppercase** followed by **5 digit number that will be generated by the oracle sequence seqName**    · If received arguments (either name or seqName) are null then the method should return “**FAIL**”  · If received value from argument 1 (**name**) has less than 2 characters or non-alphabetic letters then this method should return **INVALID\_INPUT**  · If received value from argument 2 is different from predefined **oracle sequence** name (**FlightId\_Seq)** then the method should return **INVALID\_INPUT**  · **If received inputs are valid then return the flight Id generated.** |
|  | **public String addFlight(Flight flight)** | Purpose: This method is expected to insert the flight object’s attribute values into predefine table (**Flight\_Tbl)**  · On successful insertion, return “**SUCCESS**”.  · For any exceptions return “**FAIL**” |

**Package: com.wipro.flight.service**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **FlightService** |  | **Class** |
|  | **public String createFlight(Flight flight)** | This method first, generates the flight id, initializes the flight id property and sends the Flight object to addFlight method of DAO to perform insertion operation into table.  1. If received argument (Flight object) is null then return “**FAIL**”.  2. If unable to generate flight id then return “**INVALID\_INPUT**”  3. If the flightID is generated successfully, then initialize the id attribute and call the addFlight(Flight flight) method of FlightDAO  4. If insertion of record is successful then return “**SUCCESS**”  else return **“FAIL”** |

**Package: com.wipro.flight.util**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **DBUtil** |  | DB connection class |
|  | public static Connection **getDBConnection**() | Establish a connection to the database and return the java.sql.Connection reference |

**Testcases:**

1. **Test FlightId computation for valid inputs**
2. **Test FlightId computation for invalid sequence name**
3. **Test FlightId computation with invalid name**
4. **Test FlightId computation with valid values**
5. **Test addFlight method of DAO class with invalid values**
6. **Test createFlight method of Service class with invalid values**
7. **Test createFlight method of Service class for null values**
8. **Test createFlight method of Service class with valid values**