# System Requirements Specification Index

For

# Random Number Generation

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

## Table of Contents

## 1 Project Abstract

This assessment focuses on evaluating the understanding and ability to generate random numbers in Java. You need to use both the Math.random() method and the Random class to generate random numbers for different applications.

## 2 Assessment Tasks

**Task 1: Generate a Random Decimal Number (0 to 1):**

- Use `Math.random()` to generate a random decimal number between `0` and `1`.
- Store the result in a variable named `randomDecimal` of `double` datatype.
- **Print** the result: `"Random Decimal (0-1):"` followed by the generated number.

**Task 2: Generate a Random Integer within a Range:**

- Create a new **`Random` object** named `random`.
- Use `random.nextInt(100) + 1` to generate a random integer in the range `1` to `100`.
- Store the result in a variable named `randomInt` of `integer` datatype.
- **Print** the result: `"Random Integer (1-100):"` followed by the generated number.

**Task 3: Generate a Random Floating-Point Number within a Specified Range:**

- Generate a random floating-point number between `5.5` and `20.5` using the formula:

  **min + (max - min) * random.nextDouble()**

- Store the result in a variable named `randomFloat` of `double` datatype.
- **Print** the result: `"Random Float (5.5-20.5):"` followed by the generated number.

**Task 4: Simulate a Coin Toss:**

- Declare a variable named `coinToss` of `integer` datatype, Use `random.nextInt(2)` to randomly generate `0` or `1`.
- Store the result in a variable named `coinToss` of `integer` datatype.
- Assign `"Heads"` for `0` and `"Tails"` for `1`. (or) If the result is `0`, print `"Heads"`, otherwise print `"Tails"`.
- **Print** the result: `"Coin Toss:"` followed by `"Heads"` or `"Tails"`.

**Task 5: Random Selection from an Array:**

- Declare and initialize an array `colors` containing `{"Red", "Green", "Blue", "Yellow", "Orange"}`.
- Use `random.nextInt(colors.length)` to generate a random index.
- Retrieve the randomly selected color from the array using this index.
- Store the result in a variable named `randomIndex` of `integer` data type.
- **Print** the result: `"Random Color:"` followed by the randomly selected color.

**Expected Output:**

```
Random Decimal (0-1): 0.9454783500504568
Random Integer (1-100): 98
Random Float (5.5-20.5): 18.70766020297663
Coin Toss: Heads
Random Color: Yellow
```

(**Note**: The actual output values will vary since they are generated randomly.)

## 3 Template Code Structure

### 3.1 Package: com.yaksha.assignment.RandomNumberGenerationAssignment

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **RandomNumberGenerationAssignment (class)** | • Main class demonstrating random number generation using `Math.random()` and `Random` class. Includes examples of generating random decimals, integers, floating-point numbers, simulating coin toss, and selecting random elements from an array. | Need to be implemented. |

## 4 Execution Steps to Follow

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top)  Terminal New Terminal.**
3. **This editor Auto Saves the code.**
4. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
5. **To run your project use command:**

   ```
   sudo JAVA_HOME=$JAVA_HOME /usr/share/maven/bin/mvn compile exec:java
   -Dexec.mainClass="com.yaksha.assignment.RandomNumberGenerationAssignment"
   ```

   ***If it asks for the password, provide password : pass@word1**

6. **To test your project test cases, use the command**

   ```
   sudo JAVA_HOME=$JAVA_HOME /usr/share/maven/bin/mvn test
   ```

   ***If it asks for the password, provide password : pass@word1**