

### CS23333 OBJECT ORIENTED PROGRAMMING USING JAVA LAB

# NUMBER GUESSING GAME

# A MINI PROJECT REPORT

**Submitted by**

**SANTHOSH KUMAR S 231501147**

**SANJAY KISHORE 231501179**

In partial fulfillment for the award of the degree of BACHELOR OF ENGINEERING IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM CHENNAI-602105

### 2024 - 2025

# BONAFIDE CERTIFICATE

Certified that this project report “**NUMBER GUESSING GAME**” is the Bonafide work of **“SANTHOSH KUMAR (231501147), SANJAY KISHORE (231501145)”** who carried out the project work under my supervision.

Submitted for the Practical Examination held on

## SIGNATURE

Mr.Devandra Rao, Assistant Professor, AIML,

Rajalakshmi Engineering College, (Autonomous)

Thandalam, Chennai – 602 105

### INTERNAL EXAMINER EXTERNAL

### EXAMINER

# ABSTRACT

**This paper presents the design and implementation of a Number Guessing Game, a console-based application that provides an interactive and entertaining experience for users. The system is built entirely using Java and offers a simple yet engaging interface for players to participate in guessing a randomly generated number within a defined range. The application is designed to track user attempts and provide feedback on each guess, indicating whether the guessed number is too high, too low, or correct.**

**The Number Guessing Game is a standalone Java application, showcasing basic programming concepts such as loops, conditionals, and random number generation. Users can perform operations like starting a new game, guessing numbers, and viewing their attempt count for each session. By offering clear instructions and real-time feedback, the system aims to enhance user engagement and improve logical thinking skills.**

**Each session in the game generates a unique random number, ensuring a new and challenging experience every time. Players must guess the number within a specific range provided by the system. Upon successfully guessing the number, the application displays the number of attempts taken and offers the option to play another round.**

**This application demonstrates the simplicity and versatility of Java programming while providing an enjoyable and interactive experience for users.**

## TABLE OF CONTENTS

### INTRODUCTION

* 1. [INTRODUCTION 1](#_TOC_250000)
  2. OBJECTIVES 2
  3. MODULES

1. SURVEY OF TECHNOLOGIES
   1. SOFTWARE DESCRIPTION 3
   2. LANGUAGES
      1. JAVA
      2. MYSQL

### 3. REQUIREMENTS AND ANALYSIS

* + 1. REQUIREMENT SPECIFICATION 4
    2. HARDWARE AND SOFTWARE REQUIREMENTS

### PROGRAM CODE 5

* 1. **RESULT** 45

### TESTING 50

### CONCLUSION 51

### RESEARCH AND REFERENCE 52

**Chapter 1**

**Introduction**

The Number Guessing Game is a simple yet engaging console-based application designed to provide users with an entertaining and interactive experience. This project focuses on demonstrating fundamental programming concepts in Java while offering a fun and educational way for users to test their logical thinking and problem-solving skills.

The system is easy to use, with clear instructions guiding the user through the game. Players attempt to guess a randomly generated number within a specified range, and the application provides real-time feedback to help narrow down the guesses. Key features include:

* Random Number Generation:  
  The game generates a random number each time the user starts a new session, ensuring unique gameplay every time.
* Real-Time Feedback:  
  The program provides immediate feedback after each guess, indicating whether the guessed number is too high, too low, or correct.
* Attempt Tracking:  
  The system tracks the number of attempts taken by the user to guess the correct number, adding a competitive element to the game.
* Replay Option:  
  After successfully guessing the number, the user is given the option to play again, ensuring continuous engagement.

The Number Guessing Game demonstrates the simplicity and versatility of Java, combining entertainment with basic programming principles. It serves as an excellent introduction to programming for beginners while delivering a delightful experience for

**1.1 OBJECTIVE**

The main objective of the **Number Guessing Game** is to develop an interactive and entertaining console-based application. The game is built to enhance logical thinking and problem-solving skills by challenging players to guess a randomly generated number. The project is entirely built using **Java** and demonstrates core programming concepts such as loops, conditionals, and random number generation.

The application provides users with clear feedback on their guesses, tracks the number of attempts, and allows them to replay the game for an engaging experience.

**1.2 MODULES**

The **Number Guessing Game** includes the following key modules:

* **Game Initialization:** Generates a random number within a predefined range for the user to guess.
* **Input Validation:** Ensures valid user inputs within the given range.
* **Feedback System:** Provides feedback on whether the guessed number is too high, too low, or correct.
* **Attempt Tracker:** Tracks the number of guesses the user makes during the game.
* **Replay Option:** Allows the user to replay the game or exit after completing a session.

**CHAPTER 2**

**2.1 SOFTWARE DESCRIPTION**

**Visual Studio Code**  
Visual Studio Code is a versatile and lightweight code editor ideal for Java development. It supports debugging, IntelliSense code completion, and version control integration, enabling a streamlined development process.

**2.2 LANGUAGES**

**2.2.1 JAVA**

Java is a versatile programming language widely used in software development. For this project, Java is utilized to implement a console-based game, showcasing basic concepts like random number generation, input handling, loops, and conditionals.

**CHAPTER 3**

**3.1 REQUIREMENTS SPECIFICATION**

**User Requirements**

The system should provide a smooth and interactive gaming experience, with clear instructions, instant feedback on guesses, and accurate attempt tracking.

**System Requirements**

The system should run seamlessly on any device with the following requirements:

**3.2 HARDWARE AND SOFTWARE REQUIREMENTS**

**Software Requirements**

* Operating System: Windows 10 (or any OS with Java Runtime Environment)
* Development Tool: Visual Studio Code
* Language: Java (Console-Based)

**Hardware Requirements**

* Desktop PC or Laptop
* Intel® Core™ i3 Processor or higher
* 4 GB RAM
* 64-bit Operating System
* Keyboard and Mouse

**CHAPTER 4**

**PROGRAM CODE**

Below is the complete **Java Number Guessing Game** code for your project:

**Front-End (Console UI) and Back-End Code**

java

Copy code

import java.util.Scanner;

import java.util.Random;

public class NumberGuessingGame {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Random random = new Random();

boolean playAgain = true;

System.out.println("Welcome to the Number Guessing Game!");

while (playAgain) {

int randomNumber = random.nextInt(100) + 1; // Random number between 1 and 100

int attempts = 0;

boolean hasGuessedCorrectly = false;

System.out.println("I have picked a number between 1 and 100. Can you guess it?");

while (!hasGuessedCorrectly) {

System.out.print("Enter your guess: ");

int userGuess = scanner.nextInt();

attempts++;

if (userGuess < randomNumber) {

System.out.println("Too low! Try again.");

} else if (userGuess > randomNumber) {

System.out.println("Too high! Try again.");

} else {

System.out.println("Congratulations! You guessed the number in " + attempts + " attempts.");

hasGuessedCorrectly = true;

}

}

System.out.print("Would you like to play again? (yes/no): ");

String userResponse = scanner.next();

playAgain = userResponse.equalsIgnoreCase("yes");

if (!playAgain) {

System.out.println("Thank you for playing! Goodbye.");

}

}

scanner.close();

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Number Guessing Game</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="container">

<h1>Number Guessing Game</h1>

<p id="message">Guess a number between 1 and 100!</p>

<input type="number" id="guess" placeholder="Enter your guess" />

<button id="guessButton">Guess</button>

<button id="resetButton">Reset</button>

<p id="result"></p>

</div>

<script src="script.js"></script>

</body>

</html>

**CSS (Styling)**

css

Copy code

/\* styles.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f4f4f9;

color: #333;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.container {

text-align: center;

background: #fff;

border-radius: 8px;

padding: 20px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

width: 300px;

}

h1 {

font-size: 24px;

margin-bottom: 10px;

}

#guess {

width: 100%;

padding: 10px;

margin: 10px 0;

font-size: 16px;

border: 1px solid #ccc;

border-radius: 4px;

}

button {

padding: 10px 15px;

margin: 5px;

font-size: 16px;

color: #fff;

background-color: #007bff;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

#message, #result {

margin: 10px 0;

}

#result {

font-weight: bold;

color: #28a745;

}

**JavaScript (Logic)**

javascript

Copy code

// script.js

// Backend Logic: Generate a random number

let randomNumber = Math.floor(Math.random() \* 100) + 1;

let attempts = 0;

// Get DOM elements

const message = document.getElementById("message");

const guessInput = document.getElementById("guess");

const result = document.getElementById("result");

const guessButton = document.getElementById("guessButton");

const resetButton = document.getElementById("resetButton");

// Event Listener for the Guess Button

guessButton.addEventListener("click", () => {

const userGuess = parseInt(guessInput.value);

attempts++;

if (isNaN(userGuess) || userGuess < 1 || userGuess > 100) {

result.textContent = "Please enter a valid number between 1 and 100.";

result.style.color = "red";

} else if (userGuess < randomNumber) {

result.textContent = "Too low! Try again.";

result.style.color = "orange";

} else if (userGuess > randomNumber) {

result.textContent = "Too high! Try again.";

result.style.color = "orange";

} else {

result.textContent = `Congratulations! You guessed it in ${attempts} attempts.`;

result.style.color = "green";

guessButton.disabled = true; // Disable guessing after correct answer

}

guessInput.value = ""; // Clear the input field

guessInput.focus(); // Focus on the input field for convenience

});

// Event Listener for the Reset Button

resetButton.addEventListener("click", () => {

randomNumber = Math.floor(Math.random() \* 100) + 1; // Reset random number

attempts = 0; // Reset attempts

result.textContent = ""; // Clear result message

message.textContent = "Guess a number between 1 and 100!";

guessButton.disabled = false; // Enable guessing

guessInput.value = ""; // Clear the input field

});

**HTML (Login and Registration Pages)**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login & Registration</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="container">

<div id="form-wrapper">

<!-- Registration Form -->

<form id="registerForm" class="form" style="display: none;">

<h2>Register</h2>

<input type="text" id="registerUsername" placeholder="Username" required>

<input type="email" id="registerEmail" placeholder="Email" required>

<input type="password" id="registerPassword" placeholder="Password" required>

<button type="button" onclick="registerUser()">Register</button>

<p>Already have an account? <a href="#" onclick="toggleForm('login')">Login</a></p>

</form>

<!-- Login Form -->

<form id="loginForm" class="form">

<h2>Login</h2>

<input type="text" id="loginUsername" placeholder="Username" required>

<input type="password" id="loginPassword" placeholder="Password" required>

<button type="button" onclick="loginUser()">Login</button>

<p>Don't have an account? <a href="#" onclick="toggleForm('register')">Register</a></p>

</form>

</div>

</div>

<script src="script.js"></script>

</body>

</html>

**CSS (Styling)**

css

Copy code

/\* styles.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f4f4f9;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.container {

width: 300px;

background: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

text-align: center;

}

h2 {

margin-bottom: 20px;

font-size: 24px;

}

input {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 4px;

font-size: 16px;

}

button {

width: 100%;

padding: 10px;

background-color: #007bff;

color: white;

border: none;

border-radius: 4px;

font-size: 16px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

p {

font-size: 14px;

color: #555;

}

a {

color: #007bff;

text-decoration: none;

cursor: pointer;

}

**JavaScript (Frontend Logic)**

javascript

Copy code

// script.js

// Toggle between login and registration forms

function toggleForm(formType) {

const loginForm = document.getElementById("loginForm");

const registerForm = document.getElementById("registerForm");

if (formType === "login") {

loginForm.style.display = "block";

registerForm.style.display = "none";

} else {

loginForm.style.display = "none";

registerForm.style.display = "block";

}

}

// Handle user registration

function registerUser() {

const username = document.getElementById("registerUsername").value;

const email = document.getElementById("registerEmail").value;

const password = document.getElementById("registerPassword").value;

fetch("http://localhost:3000/register", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify({ username, email, password })

})

.then(response => response.json())

.then(data => {

alert(data.message);

if (data.success) toggleForm("login");

})

.catch(err => console.error(err));

}

// Handle user login

function loginUser() {

const username = document.getElementById("loginUsername").value;

const password = document.getElementById("loginPassword").value;

fetch("http://localhost:3000/login", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify({ username, password })

})

.then(response => response.json())

.then(data => {

alert(data.message);

if (data.success) window.location.href = "/dashboard";

})

.catch(err => console.error(err));

}

**Backend Code (Node.js with Express)**

**Server Setup**

javascript

Copy code

const express = require("express");

const bodyParser = require("body-parser");

const cors = require("cors");

const app = express();

const port = 3000;

// Middleware

app.use(cors());

app.use(bodyParser.json());

// Temporary user storage

const users = [];

// Registration endpoint

app.post("/register", (req, res) => {

const { username, email, password } = req.body;

// Check if user already exists

const existingUser = users.find(user => user.username === username);

if (existingUser) {

return res.json({ success: false, message: "Username already taken." });

}

// Register user

users.push({ username, email, password });

res.json({ success: true, message: "User registered successfully!" });

});

// Login endpoint

app.post("/login", (req, res) => {

const { username, password } = req.body;

// Authenticate user

const user = users.find(user => user.username === username && user.password === password);

if (user) {

res.json({ success: true, message: "Login successful!" });

} else {

res.json({ success: false, message: "Invalid username or password." });

}

});

// Start the server

app.listen(port, () => {

console.log(`Server running on http://localhost:${port}`);

});

**Server Code: server.js**

javascript

Copy code

const express = require("express");

const bodyParser = require("body-parser");

const cors = require("cors");

const bcrypt = require("bcrypt");

const jwt = require("jsonwebtoken");

const app = express();

const port = 3000;

// Middleware

app.use(cors());

app.use(bodyParser.json());

// Temporary in-memory storage (replace with a database in production)

const users = [];

// Secret key for JWT

const JWT\_SECRET = "your\_secret\_key"; // Replace with a secure secret in production

/\*\*

\* @route POST /register

\* @desc Register a new user

\* @body { username, email, password }

\*/

app.post("/register", async (req, res) => {

const { username, email, password } = req.body;

// Check if user already exists

const existingUser = users.find(user => user.username === username || user.email === email);

if (existingUser) {

return res.status(400).json({ success: false, message: "Username or email already taken." });

}

// Hash the password

const hashedPassword = await bcrypt.hash(password, 10);

// Save the user

users.push({ username, email, password: hashedPassword });

res.status(201).json({ success: true, message: "User registered successfully!" });

});

/\*\*

\* @route POST /login

\* @desc Authenticate a user

\* @body { username, password }

\*/

app.post("/login", async (req, res) => {

const { username, password } = req.body;

// Find the user

const user = users.find(user => user.username === username);

if (!user) {

return res.status(401).json({ success: false, message: "Invalid username or password." });

}

// Compare the hashed password

const isPasswordValid = await bcrypt.compare(password, user.password);

if (!isPasswordValid) {

return res.status(401).json({ success: false, message: "Invalid username or password." });

}

// Generate a JWT token

const token = jwt.sign({ username }, JWT\_SECRET, { expiresIn: "1h" });

res.json({ success: true, message: "Login successful!", token });

});

/\*\*

\* @route GET /users

\* @desc Get all registered users (for debugging, optional)

\*/

app.get("/users", (req, res) => {

res.json(users);

});

// Start the server

app.listen(port, () => {

console.log(`Server is running on http://localhost:${port}`);

});

* + - CHAPTER 6
    - 6.1 Unit Testing
    - Unit testing is a testing technique where individual modules of a program are tested separately to ensure their functionality is correct. For the Number Guessing Game, each module (such as random number generation, user input validation, feedback mechanism, and attempt tracking) was tested individually during the development process. This ensured that each component performed as expected before being integrated into the full program.
    - 6.2 Integration Testing
    - Integration testing involves combining individual modules and testing them as a group to ensure they work together seamlessly. In the Number Guessing Game, the modules such as number generation, user input handling, feedback provision, and replay functionality were tested together to ensure the game flowed correctly and provided the desired user experience.
    - 6.3 System Testing
    - System testing was conducted on the entire game as a whole to validate that all requirements were met. The application was tested in different environments, such as various versions of Java and operating systems, to ensure compatibility. Any identified bugs were resolved to improve the game's stability and usability.
    - 6.4 Acceptance Testing
    - Acceptance testing was performed to confirm that the Number Guessing Game met the specified requirements and provided a satisfactory user experience. The game was tested by users to verify:
    - The clarity of instructions.
    - The accuracy of the feedback system.
    - The proper functioning of the replay feature.
    - After successful user testing, the application was deemed ready for deployment.
    - CHAPTER 7
    - 7.1 CONCLUSION
    - The Number Guessing Game project successfully fulfills its objectives by providing an interactive and entertaining experience for users. The game demonstrates core programming concepts such as random number generation, input validation, and real-time feedback.
    - The development process emphasized simplicity and usability, ensuring that players of all skill levels could engage with the game. Through effective testing, the application was optimized for performance, with all modules functioning seamlessly.
    - The main focus of the project was to create a lightweight yet engaging game that could serve as both an educational tool for programming beginners and a source of entertainment. With a simple replay feature and accurate attempt tracking, the game encourages users to challenge themselves repeatedly.
    - This project highlights the potential of Java as a language for developing dynamic and user-friendly applications. It serves as a foundation for creating more complex projects in the future.
    - CHAPTER 8
    - RESEARCH AND REFERENCES
    - The following resources were referenced during the development of the Number Guessing Game:
    - https://www.geeksforgeeks.org/random-in-java/
    - <https://www.w3schools.com/java/>
    - <https://docs.oracle.com/javase/tutorial/>
    - https://www.javatpoint.com/java-tutorial
    - <https://github.com/example/Number-Guessing-Game>