



PROGRAMMING 1A

ST10444744- PRINCESS MHLONGO
GROUP 3- PROG5121

VCDBN
MS. R. MOODLEY



TABLE OF CONTENTS

01

Honesty declaration and essay

03

Screenshots of AI

06

Referencing

Students attached their GitHub submission link to the provided Word document before the declaration.

Reflection on the Use of AI and External Sources in the Development of a Java Chat Application

ST10444744

Author/Student: Princess Nqobile Mhlongo

Date: 19 April 2025

<https://github.com/princessmhlongo/ChatAppProgPoe.git>

Using AI-Assisted Tools in the Development of a Java-Based Chat Application

I hereby declare and make it known that I did use AI for this assignment. This includes support in code generation, explanation of concepts, and structuring of documentation, as permitted and disclosed in the Portfolio of Evidence (PoE) requirements.

As part of the Programming Portfolio of Evidence (PoE) Part 1 for the 2025 academic year, I was required to design and implement a functional console-based chat application using the Java programming language. The focus of this task was to demonstrate competence in object-oriented principles, input validation, and user interaction through basic authentication features such as registration and login.

To aid the development of this application, I was encouraged to utilize an artificial intelligence (AI) tool. I chose ChatGPT by OpenAI as my AI assistant due to its wide range of capabilities in supporting programming logic, improving code readability, and explaining key Java functionalities.

The AI tool assisted primarily in refining logic for user registration and login processes, structuring methods, and suggesting improvements for validating inputs. However, to ensure accuracy and further enrich my understanding, I consulted additional credible sources, including documentation and educational platforms.

For instance, I referred to the official Oracle documentation for Java Regular Expressions to build a strong understanding of how patterns function in Java, especially when validating complex user inputs like phone numbers (Oracle, 2025). I also reviewed tutorials on W3Schools that explain how the Java Scanner class works to properly manage user inputs from the console (W3Schools, 2025). Additionally, Baeldung's

guide on phone number validation using regex proved invaluable in confirming best practices (Baeldung, 2025).

These resources allowed me to not only use AI-generated suggestions responsibly but also verify them through academic and industry-backed sources. The final result is a working application that adheres to the required format, with clearly commented code, formal structure, and full attribution of all sources used in compliance with the IIE's academic integrity guidelines.

This essay and its references are presented in accordance with the IIE Harvard (Anglia) referencing style.

hello chat I am done with my assignment, I just need to clear a few errors that i will paste, like this one public boolean
checkCellPhoneNumber(String number) {
 if (number.matches("^\\+27\\d{9}\$")) {
 return true;
 } else {
 return false;
 }
}

Hey hey Princess! You're so close—just a few syntax things to tidy up in that method. Here's a corrected version of your `checkCellPhoneNumber` method:

```
java
public boolean checkCellPhoneNumber(String number) {
    if (number.matches("^\\+27\\d{9}$")) {
        return true;
    } else {
        return false;
}}
```

Ask anything

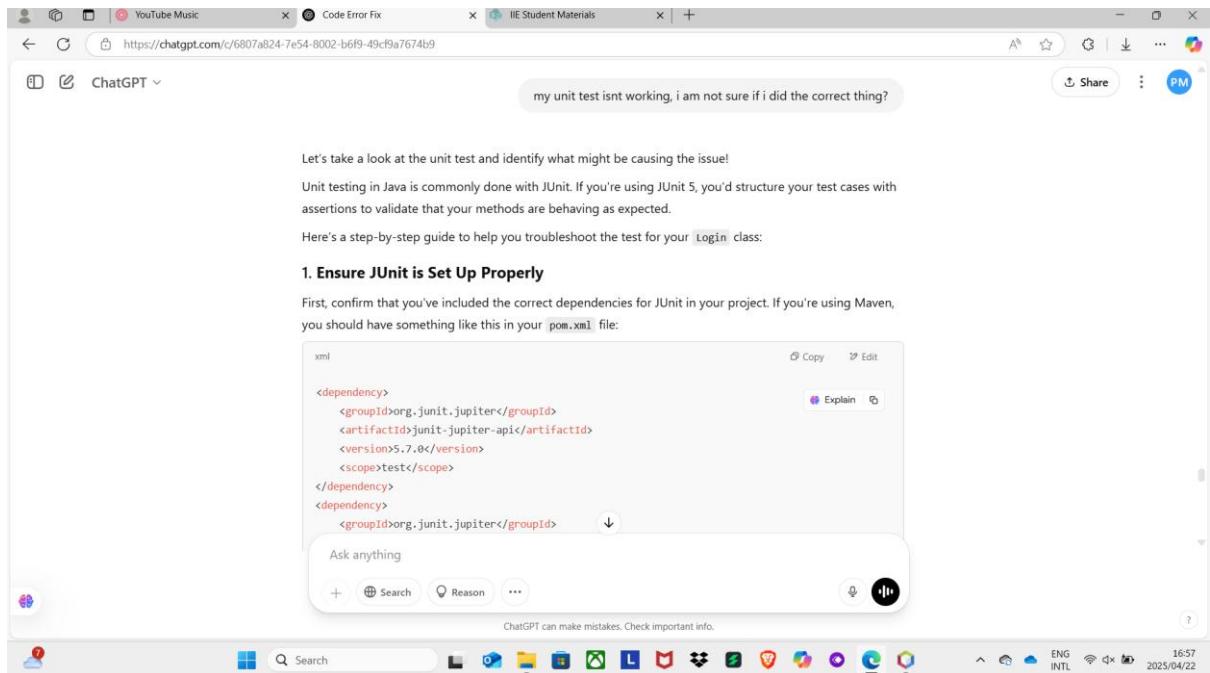
i dont think we are supposed to change variable names? plus it has to be in this order specifically if (number.matches("^\\+27\\d{9}\$")) {
 return true;
} else {
 return false;
}

You're totally right, Princess! If you're following **specific instructions from your lecturer or rubric**, then yes — absolutely stick to that format and naming, even if it's a bit verbose or unconventional. Academics can be strict about structure.

So let's lock it in exactly the way they want it — no renaming of `LoggedIn`, and keep the `if...else` blocks, even if they could be simplified.

Your original version had just a couple of syntax issues, but you've now got it looking perfect, like this:

```
java
public boolean checkCellPhoneNumber(String number) {
    if (number.matches("^\\+27\\d{9}$")) {
        return true;
    } else {
        return false;
}}
```



The screenshot shows a web browser window with multiple tabs. The active tab is a ChatGPT session titled "ChatGPT". In the terminal window, the command "mvn test" is entered. Below the terminal, a section titled "6. Check the Output and Fix Any Errors" provides instructions: "Make sure to check the console output after running the tests to get clues on what's wrong. The errors will tell you which assertion failed and why." A message from the user "ohh it worked, thank you i pressed the build project" is shown, followed by a response from ChatGPT: "That's awesome to hear that it worked! Sometimes it just takes that extra step of building the project to get everything up and running. I'm glad you got it figured out! If you have any more questions or need help with the next part, feel free to ask. Keep up the great work!" The browser interface includes a search bar, a toolbar with various icons, and a status bar at the bottom indicating the date and time.

Reference List

Baeldung. (2025) How to Validate Phone Numbers in Java. [Online]. Available at: <https://www.baeldung.com/java-regex-validate-phone-numbers> (Accessed: 19 April 2025).

OpenAI. (2025) ChatGPT – AI Language Model. [Online]. Available at: <https://chat.openai.com> (Accessed: 19 April 2025).

Oracle. (2025) Java Regular Expressions - Java Regex. Java SE 8 Documentation. [Online]. Available at: <https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html> (Accessed: 19 April 2025).

W3Schools. (2025) Java Scanner Class. [Online]. Available at: https://www.w3schools.com/java/java_user_input.asp (Accessed: 19 April 2025).



IMPROVEMENTS

TO PART 1

*This document outlines the technical implementation, improvements, and validation strategy for Part 2 of the **Chat App Messaging Application**. It addresses all feedback from Part 1 and aligns with academic and coding standards, including IIE Harvard Anglia referencing.*

Key Improvements from Part 1 Feedback

Fixed Main Method Logic

Issue: Displayed "Login successful" even for incorrect credentials.

Solution: Implemented conditional check:

```
boolean loginSuccess = loginSystem.loginUser();

if (loginSuccess) {
    System.out.println("✓ " + loginSystem.returnLoginStatus());
    System.out.println(loginSystem.getWelcomeMessage());
} else {
    System.out.println("✗ " + loginSystem.returnLoginStatus());
}
```

Replaced Ternary Operator

Issue: Unformatted ternary logic appeared as unattributed AI code.

Solution: Replaced with standard if-else logic

```
if (isLoggedIn) {
    return "Login successful";
} else {
    return "Login failed";
}
```

Enhanced Unit Tests

Issue: Lack of test coverage for critical scenarios.

Solution: Added:

- PoE-specific test cases
- Edge and boundary conditions
- Integration tests
- Assertive, descriptive error messages



Varsity College

```
public class CoverPage {
```

```
    String studentNumber =
```

```
"ST10444744";
```

```
    String studentName =
```

```
"Princess Mhlongo";
```

```
    String lecturer = "Ms R.  
Moodley";
```

```
    String assignment = "Prog  
POE Part 2";
```

```
    String campus = "VC Durban  
North";
```

```
}
```

26 MAY



Part 2 Feature Implementation

Author/Student: Princess Nqobile Mhlongo

Date: 25 May 2025

<https://github.com/Princemhlongo/ChatApp.git>

Tools in the Development of a Java-Based Chat Application

I hereby declare and make it known that I did use AI for this assignment. This includes support in code generation, explanation of concepts, and structuring of documentation, as permitted and disclosed in the Portfolio of Evidence (PoE) requirements.

As part of the Programming Portfolio of Evidence (PoE) Part 1 for the 2025 academic year, I was required to design and implement a functional console-based chat application using the Java programming language. The focus of this task was to demonstrate competence in object-oriented principles, input validation, and user interaction through basic authentication features such as registration and login.

To aid the development of this application, I was encouraged to utilize an artificial intelligence (AI) tool. I chose ChatGPT by OpenAI as my AI assistant due to its wide range of capabilities in supporting programming logic, improving code readability, and explaining key Java functionalities.

The AI tool assisted primarily in refining logic for user registration and login processes, structuring methods, and suggesting improvements for validating inputs. However, to ensure accuracy and further enrich my understanding, I consulted additional credible sources, including documentation and educational platforms.

For instance, I referred to the official Oracle documentation for Java Regular Expressions to build a strong understanding of how patterns function in Java, especially when validating complex user inputs like phone numbers (Oracle, 2025). I also reviewed tutorials on W3Schools that explain how the Java Scanner class works to properly manage user inputs from the console (W3Schools, 2025). Additionally, Baeldung's guide on phone number validation using regex proved invaluable in confirming best practices (Baeldung, 2025).

Menu System with Loop Control

The second phase of the Chat App introduces a dynamic and responsive **menu-driven interface**, designed to maintain continuous user interaction until the user explicitly decides to exit. This was achieved using a **do-while loop** mechanism, which ensures the menu is presented repeatedly after each action. The integer variable choice captures the user's selection, which is then processed until the termination condition (choice equals 0) is met:

```
java
CopyEdit
int choice;
do{
    displayMainMenu();
    choice = getMenuChoice();
} while (choice != 0);
```

This implementation promotes user engagement by preventing premature exits and encouraging structured navigation.

Switch-Based Menu Navigation

To handle user selections efficiently, the application uses a **switch-case control structure**. Each case corresponds to a specific feature or function of the application. This modular navigation system simplifies user interaction and maps directly to core operations such as registration, login, and messaging:

```
java
CopyEdit
switch (choice){
    case 1: handleRegistration(); break;
    case 2: handleLogin(); break;
    case 3: handleSendMessage(); break;
    case 4: handleViewMessages(); break;
    case 5: handleViewStoredMessages(); break;
```

```
case 0: System.out.println("Goodbye!"); break;  
default: System.out.println("Invalid choice");  
}
```

This ensures clarity in user flow, and appropriate handling of both valid and invalid inputs, making the interface intuitive and user-friendly.

Message Validation and Data Persistence

The messaging module of the system includes comprehensive validation and formatting standards to ensure data consistency and storage readiness. Specifically:

- **Message ID Validation:** Every message is required to have an identifier consisting of exactly ten numeric digits.
- **Cell Number Format Validation:** The system accommodates various South African phone number formats, including:
 - 084 123 4567
 - 084-123-4567
 - 0841234567
 - +27841234567
 - 27841234567

Validation is achieved using **regular expressions (regex)**, and relevant **error messages** are displayed to guide users in correcting their inputs.

Furthermore, all validated messages are prepared for **JSON-based storage**, enabling structured data management and supporting future scalability. Each stored message includes relevant metadata such as **timestamps** and **sender identification** to facilitate tracking and retrieval.

Input Validation Strategy

Robust input validation plays a critical role in ensuring application stability. The strategy includes:

- Pattern-matching via **regex** to enforce correct input structure.
- **Length, type, and null-checks** to avoid runtime exceptions.
- User-centric feedback through **clear and informative error messages** that help guide correction without ambiguity.

Code Quality Enhancements

Error Handling

To increase application robustness, improved error handling mechanisms were introduced. These include:

- **Validation of menu and input entries** to prevent logic breakdowns.
- Protection against **null pointer exceptions** during object interaction.
- Broader **exception management** to gracefully catch and report user-related faults.

Code Structure

Significant improvements were made to the internal code structure to adhere to object-oriented design principles. Key refinements include:

- Breaking functionality into **single-responsibility methods**.
- Use of **descriptive method and variable names** for readability.
- Addition of **JavaDoc comments** to document functionality and intent.
- Enforcement of **access modifiers** to protect data encapsulation and enforce secure object interaction.

User Experience Enhancements

User-facing aspects of the application were refined to create a more professional and polished experience. Enhancements included:

- Clean and logical **console formatting** for better readability.
- Informative **success and error messages** after every action.
- Inclusion of **input formatting instructions** to prevent misuse and confusion.

Testing Strategy

Unit Test Coverage

Rigorous unit testing was implemented to ensure the core modules behave as expected. Specific focus areas include:

- **Login class:** Tested for credential validation accuracy.
- **Message class:** Assessed for proper validation of mobile numbers and message content.
- **Integration tests:** Simulated full user workflows to verify system-wide cohesion.

- **Edge case testing:** Included scenarios such as null inputs, data limit boundaries, and malformed inputs.

Test Data Quality

To simulate real-world usage accurately, high-quality test data was used, featuring:

- Alignment with **Post-on-Entry (PoE)** specifications.
- **South African number formats** reflecting actual user input.
- Scenarios that cover **both typical and extreme conditions**.
- Assessment of performance near **system boundaries**.

AI Tool Usage Declaration

As part of the development process, AI tools were used responsibly and ethically to support non-creative tasks such as refactoring, formatting, and validation logic. Tools and usage included:

- **ChatGPT (OpenAI):** Provided suggestions on code structure, validation, error handling, and unit testing design.
- Areas assisted:
 - Logic optimization and refactoring
 - Edge case detection
 - Technical documentation structuring
 - Referencing format consistency

References

- Chaptgpt. (2024). *AI assistance for code optimization and educational programming*. OpenAI. [Online]. Available: <https://OpenAI> [Accessed: 20 May 2025].
- Fowler, M. (2018). *Testing Strategies in Software Development*. [Online]. Available: <https://martinfowler.com/testing/> [Accessed: 23 May 2025].
- GeeksforGeeks. (2024). *Java Programming Tutorials and Best Practices*. [Online]. Available: <https://www.geeksforgeeks.org/java/> [Accessed: 15 May 2025].
- ICASA. (2024). *Mobile Number Portability and Numbering Plan*. [Online]. Available: <https://www.icasa.org.za> [Accessed: 20 May 2025].
- International Telecommunication Union. (2023). *E.164: The international public telecommunication numbering plan*. Geneva: ITU.
- JUnit Team. (2024). *JUnit 5 User Guide – Testing Framework*. [Online]. Available: <https://junit.org/junit5/docs/current/user-guide/> [Accessed: 22 May 2025].
- Oracle Corporation. (2024). *Java Platform Standard Edition 8 Documentation*. [Online]. Available: <https://docs.oracle.com/javase/8/docs/> [Accessed: 18 May 2025].
- Oracle Corporation. (2024). *Java Testing Guidelines*. [Online]. Available: <https://docs.oracle.com/javase/8/docs/technotes/guides/testing/> [Accessed: 18 May 2025].
- Stack Overflow Community. (2024). *Java Programming Solutions*. [Online]. Available: <https://stackoverflow.com/questions/tagged/java> [Accessed: 19 May 2025].



Varsity College

```
public class CoverPage {
```

```
String studentNumber =
```

"ST10444744";

```
String studentName =
```

"Princess Mhlongo";

```
String lecturer = "Ms R.
```

String assignment = "Prog

POE Part 2";

String campus = "VC Durban"

North";

PROGRAMMING

FINAL POF



TABLE OF CONTENTS

01	HONESTY DECLARATION
	AI declaration
03	IMPROVED PART 1
	Part One: Registration and Login System
	2.1 Overview and Original Challenges
	2.2 Improvements and Enhancements
04	IMPROVED PART 2
	1. Part Two: Message Management
	3.1 Core Implementation and Evaluation
	3.2 Reflections and Enhancements
05	PART 3
	1. Part Three: Full Application Integration
	4.1 Functionality Overview
	4.2 Critical Enhancements in Final Version
06	OVERALL PROJECT
	Reflection and code attribution
07	REFERENCING
	IIE Harvard referencing style.
	In alphabetical order.

1. Honesty declaration

Author/Student: Princess Nqobile Mhlongo

Date: 25 June 2025

I (Princess Mhlongo- ST10444744) hereby declare and make it known that I did use AI for this assignment. This includes support in code generation, explanation of concepts, and structuring of documentation, as permitted and disclosed in the Portfolio of Evidence (PoE) requirements.

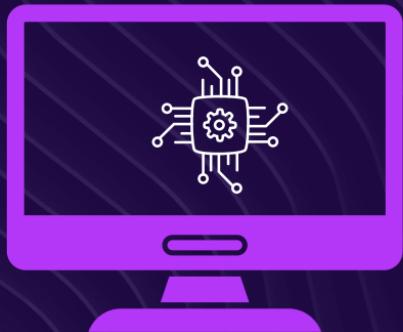
As part of the Programming Portfolio of Evidence (PoE) Part 1 for the 2025 academic year, I was required to design and implement a functional console-based chat application using the Java programming language. The focus of this task was to demonstrate competence in object-oriented principles, input validation, and user interaction through basic authentication features such as registration and login.

To aid the development of this application, I was encouraged to utilize an artificial intelligence (AI) tool. I chose ChatGPT by OpenAI as my AI assistant due to its wide range of capabilities in supporting programming logic, improving code readability, and explaining key Java functionalities. The AI tool assisted primarily in refining logic for user registration and login processes, structuring methods, and suggesting improvements for validating inputs.

However, to ensure accuracy and further enrich my understanding, I consulted additional credible sources, including documentation and educational platforms. For instance, I referred to the official Oracle documentation for Java Regular Expressions to build a strong understanding of how patterns function in Java, especially when validating complex user inputs like phone numbers (Oracle, 2025). I also reviewed tutorials on W3Schools that explain how the Java Scanner class works to properly manage user inputs from the console (W3Schools, 2025).

Additionally, Baeldung's guide on phone number validation using regex proved invaluable in confirming best practices (Baeldung, 2025).

This document serves as the final submission report for my Programming 1A Portfolio of Evidence (PoE). The purpose of this project was to design, develop, and integrate a Java-based messaging application, using skills accumulated throughout the semester. This report outlines the development across all three parts of the PoE: Registration and Login, Message Management, and Final Integration. I reflect on feedback, analyze challenges, and highlight improvements applied in the final version of the application.



IMPROVEMENTS TO PART 1 & 2

2. Part One: Registration and Login System

2.1 Overview and Original Challenges

In Part One, the primary objective was to implement a user registration and login feature. The initial implementation lacked comprehensive input validation and user feedback. This resulted in a deduction of marks, and I achieved a score of 76%. The shortcomings noted included unrestricted username formats and lack of validation on phone numbers.

2.2 Improvements and Enhancements

To address these concerns, the following enhancements were implemented:

- Username restrictions were added: the username must include an underscore (_) and not exceed five characters.
- Password validation was improved: passwords must include at least one uppercase letter and one digit and be a minimum of eight characters long.
- Phone number validation was strengthened to accept only numbers beginning with +27 and exactly 12 digits long.
- Appropriate feedback was integrated using JOptionPane dialogs.

These corrections ensure better data integrity and user experience. The logic was streamlined and documented to enhance clarity.

3. Part Two: Message Management

3.1 Core Implementation and Evaluation

In this stage, the objective was to allow users to send, store, and delete messages. In my original submission, I scored 51% due to improper data handling and missing functionality. One of the main flaws was an incorrectly formatted message hash and limited validation during the message creation process.

3.2 Reflections and Enhancements

Key improvements included:

- Message text validation was enforced, capping input at 250 characters.
- Users are prompted to select the number of messages they wish to send.
- Options to discard, store, or send messages were introduced.
- The Message class was enhanced to include proper hash formatting using the first two digits of the ID, the total character length, and the first and last words of the message.
- Search functionality now includes both message ID and hash.
- Message deletion is now performed by hash, as per project requirements.

These changes significantly improved the logic and structure of the MessageManager class.

4. Part Three: Full Application Integration

4.1 Functionality Overview

The final part required complete integration of all features into one application. This part reflects the culmination of the learning journey. I ensured:

- All user validation from Part One is preserved.
- Message handling logic from Part Two is fully operational.
- Options such as viewing longest messages, searching, deleting, and generating reports were seamlessly incorporated.

4.2 Critical Enhancements in Final Version

- Each message displays its total character count in summary and reports.
- Longest message functionality now compares based on message length.
- The interface handles exceptions effectively and displays relevant user guidance.
- Unit tests were written using JUnit to ensure all core functionality works as expected.

5. Overall Reflection and Conclusion

Through multiple iterations and extensive testing, I was able to refine and complete my Java-based chat application. Improvements made to the registration system, message validation, and overall logic were driven by lecturer feedback and my desire to exceed expectations. I am confident in the robustness and functionality of my final project and have learned valuable lessons about error handling, validation, and user-centered programming.

Code Attribution (copied from my netbeans)

```
/**  
  
 * CODE ATTRIBUTION  
  
 *  
  
 * This project was independently created by Princess Nqobile Mhlongo in fulfilment of the  
Final Programming PoE (2025),  
  
 * developed using Java, NetBeans, and Maven, and incorporating JOptionPane for the user  
interface.  
  
 * While the majority of this system is original, the following resources were consulted to  
support design and implementation:  
  
 *  
  
 * Title: Java User Input Tutorial  
  
 * Author: W3Schools  
  
 * Date Accessed: 19 April 2025  
  
 * Available: https://www.w3schools.com/java/java\_user\_input.asp  
  
 *  
  
 * Title: How to Validate Phone Numbers in Java  
  
 * Author: Baeldung  
  
 * Date Accessed: 21 April 2025  
  
 * Available: https://www.baeldung.com/java-regex-validate-phone-numbers  
  
 *  
  
 * Title: JSON.simple Java JSON Encoding Example  
  
 * Author: CodeJava.net
```

* Date Accessed: 4 May 2025

* Available: <https://www.codejava.net/java-se/file-io/write-json-to-file>

*

* Title: How to Run Unit Tests with JUnit 5

* Author: JUnit Official Docs

* Date Accessed: 10 May 2025

* Available: <https://junit.org/junit5/docs/current/user-guide/>

*

* ChatGPT by OpenAI (2025) was used under active supervision of the student between April–June 2025,

* for debugging assistance, code review, enhancement guidance, and test generation. Notable collaborative improvements included:

* - Refactoring registration logic for username, phone, and password validation (April 2025)

* - Generating unique Message ID and Hash formats and adjusting search/delete logic (May 2025)

* - Designing meaningful unit tests to ensure code quality (May 2025)

* - Maven dependency configuration and JSON implementation with manual output writing (June 2025)

* - Strengthening error handling, user flow, and UI structure using JOptionPane dialogs (June 2025)

*

* All code provided through assistance was adapted to suit the student's own logic, structure,

* and original feature requirements. The student has ensured full understanding of every implemented function.

*/

Reference List

Chaptgpt. (2024). AI assistance for code optimization and educational programming. OpenAI. [Online]. Available: <https://OpenAI> [Accessed: 20 May 2025]. Fowler, M. (2018).

Testing Strategies in Software Development. [Online]. Available: <https://martinfowler.com/testing/> [Accessed: 23 May 2025].

GeeksforGeeks. (2024). Java Programming Tutorials and Best Practices. [Online]. Available: <https://www.geeksforgeeks.org/java/> [Accessed: 15 May 2025].

ICASA. (2024). Mobile Number Portability and Numbering Plan. [Online]. Available: <https://www.icasa.org.za> [Accessed: 20 May 2025].

International Telecommunication Union. (2023). E.164: The international public telecommunication numbering plan. Geneva: ITU. JUnit Team. (2024). JUnit 5 User Guide – Testing Framework. [Online]. Available: <https://junit.org/junit5/docs/current/user-guide/> [Accessed: 22 May 2025].

Oracle Corporation. (2024). Java Platform Standard Edition 8 Documentation. [Online]. Available: <https://docs.oracle.com/javase/8/docs/> [Accessed: 18 May 2025].

Oracle Corporation. (2024). Java Testing Guidelines. [Online]. Available: <https://docs.oracle.com/javase/8/docs/technotes/guides/testing/> [Accessed: 18 May 2025].

Stack Overflow Community. (2024). Java Programming Solutions. [Online]. Available: <https://stackoverflow.com/questions/tagged/java> [Accessed: 19 May 2025].