System Test Report: NONSENSE Generator

Riccardo De Nat, Jacopo Di Lauro, Lorenzo Gatto, Asia Chiarion Sileni

May 2025

1. Introduction

This document represents the **System Test Report** of the "NONSENSE Generator" project.

The purpose of this **System Test Report** is to present in detail the testing procedures performed on the entire "NONSENSE Generator" application and the related results. The features of the user interface, the saving of terms within the dictionaries, the correct grammatical analysis, the behavior with irregular verbs, the creation of sentences from templates, and the toxicity control feature have been tested. The various Tests will be carried out on a Windows 11 OS and Linux Fedora42 The document created by the **Unit Tests**, generated by IntelliJ, can be found inside the file in /Documents/Test/ called <Test Result.html>.

The link to the Jira project that contains the User Stories is: https://jacopodilauro04.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog

2. User Stories e Acceptance Criteria

Epic: Content Moderation

SCRUM-1 - Text Toxicity Validation

Description: As a user, I want the generated sentence to be automatically checked for toxicity, so that I can ensure that the generated content is appropriate for all users.

Acceptance Criteria:

- **CA1.1:** Given a system-generated sentence, when the system detects toxicity above a predefined threshold, then the user is notified that the sentence is potentially toxic.
  Outcome: OK. Date: 23/05/2025. Comments: no bug found.
- **CA1.2:** Given a system-generated sentence, when the sentence is below the toxicity threshold, then the system allows its use without any toxicity warning.
  Outcome: OK. Date: 23/05/2025. Comments: no bug found.
- **CA1.3:** Given a generated sentence, if it is considered toxic it lists its characteristics (e.g., Toxic, Insult, Profanity).
  Outcome: OK. Date: 23/05/2025. Comments: no bug found.
- **CA1.4:** The toxicity check of the sentence is done automatically without additional input from the user after sentence generation.
  Outcome: OK. Date: 23/05/2025. Comments: no bug found.

Epic: Sentence Processing

SCRUM-2 - Sentence Syntactic Analysis

Description: As a user, I want to input a sentence and obtain a detailed syntactic analysis, So that I can clearly understand the grammatical structure of the sentence.

Acceptance Criteria:

- **CA2.1:** Given a user who enters a sentence in the designated text box, after submitting (e.g., clicking the "Analyze and Generate Nonsense" button), the system divides the sentence into [nouns], [verbs], [adjectives].
  Outcome: OK. Date: 23/05/2025. Comments: Correct.
- **CA2.2:** For example, once the sentence "The dog is on the white table.", is entered, it is divided and displayed as: Nouns: dog, table | Verbs: is | Adjectives: white.
  Outcome: OK. Date: 23/05/2025. Comments: examples handled correctly.
- **CA2.3:** The system accurately identifies POS tags for common English grammati-cal structures.
  Outcome: OK. Date: 23/05/2025. Comments: POS tagging is correct and sufficient.

SCRUM-11 - Verb Tense Selection

Description: As a user, I want to choose whether the generated sentence should be in past, present, or future tense, so that the sentence fits the desired time frame.

Acceptance Criteria:

- **CA3.1:** Given the sentence generation interface, when the user selects 'Past Tense' from the "Desired verb tense" drop-down menu, then the subsequently generated sentence's main verb(s) are conjugated in a past tense.
  Outcome: OK. Date: 23/05/2025. Comments: past tense conjugation working, the verb to be is handled in a superfluous but functional way.
- **CA3.2:** Given the sentence generation interface, when the user selects 'Present Tense', then the subsequently generated sentence's main verb(s) are conjugated in a present tense.
  Outcome: OK. Date: 23/05/2025. Comments: present conjugation working.
- **CA3.3:** Given the sentence generation interface, when the user selects 'Future Tense', then the subsequently generated sentence's main verb(s) are conjugated in a future tense.
  Outcome: OK. Date: 23/05/2025. Comments: future conjugation working, there are many types of future based on the context, but we chose one, otherwise we would have needed another context-aware API.
- **CA3.4:** By default, the verb tense selected in the drop-down menu is 'Present'.
  Outcome: OK. Date: 23/05/2025. Comments: default verb tense correctly.
- **CA3.5:** The interface clearly shows the available tense options (Present, Past, Future) through a drop-down menu.
  Outcome: OK. Date: 23/05/2025. Comments: clear verb-tense options menu.

Epic: Sentence Generation & Dictionary

SCRUM-3 - Customizable Internal Dictionary

Description: As a user, I want to be able to add terms to the internal dictionaries (nouns, verbs, adjectives), so that they can be reused in future sentence generations.

Acceptance Criteria:

- **CA4.1:** The system provides an interface (e.g., accessed through the "Manage Dictionaries" button) to add new nouns, adjectives, and verbs to their respective dictionaries.
  Outcome: OK. Date: 23/05/2025. Comments: interface to add terms present.
- **CA4.2:** The interface shows input fields for "New noun...", "New adjective...", "New verb..." and corresponding "Add" buttons.
  Outcome: OK. Date: 23/05/2025. Comments: input fields and "Add" buttons working.
- **CA4.3:** After entering a term and clicking the corresponding "Add" button, the term is saved in the appropriate internal dictionary for use in template creation.
  Outcome: OK. Date: 23/05/2025. Comments: term saved correctly into the dictionary.
- **CA4.4:** The system prevents the addition of duplicate terms to a dictionary. If a term already exists, a warning banner is displayed (e.g., "Verb 'try' is already in the dictionary.").
  Outcome: OK. Date: 23/05/2025. Comments: effective duplicate term prevention.
- **CA4.5:** Once a new term has been successfully inserted, a success banner is displayed (e.g., "Adjective 'black' added successfully!").
  Outcome: OK. Date: 23/05/2025. Comments: success banner displayed.

SCRUM-4 - Random Generation of Nonsense Sentences

Description: As a user, I want the system to generate grammatically correct random sentences using predefined templates, so that I can explore creative linguistic combinations in a fun and useful way.

Acceptance Criteria:

- **CA5.1:** The system generates a grammatically correct nonsense sentence when the "Analyze and Generate Nonsense" button is clicked, using either a user-chosen template or a randomly selected one if "Random (based on sentence)" is chosen.
  Outcome: OK. Date: 23/05/2025. Comments: correct sentence generation.
- **CA5.2:** Given custom words have been added to the dictionary (as per SCRUM-3), when sentences are generated, these custom words can appear in the generated sentences if the chosen/random template accommodates those word types.
  Outcome: OK. Date: 23/05/2025. Comments: custom words appear in the sentence.
- **CA5.3:** A drop-down menu "Choose a Template (optional)" is displayed, allowing the user to select from a list of at least 100 predefined templates or choose random generation.
  Outcome: OK. Date: 23/05/2025. Comments: Drop-down template containing at least 100 options.
- **CA5.4:** The system includes templates capable of generating sentences of varying complexity, including some templates that result in sentences with multiple clauses or more than [10] words.
  Outcome: OK. Date: 23/05/2025. Comments: variety of templates by complexity.

Epic: Design

SCRUM-10 - Design Interface

Description: As a user, I want the interface to have a modern and intuitive design, so that I can navigate and use the web application efficiently and enjoyably.

Acceptance Criteria:

- **CA6.1:** The design is consistent (e.g., same color palette, font styles, tone) across all pages/views of the application. (Visually inspect all screens).
  Outcome: OK. Date: 23/05/2025. Comments: consistent design on all pages.

- **CA6.2:** Buttons for similar actions (e.g., "Add Noun", "Add Adjective", "Add Verb") are visually identical or highly similar in style and placement. The main action button "Analyze and Generate Nonsense" is prominent.
  Outcome: OK. Date: 23/05/2025. Comments: consistent buttons design.

- **CA6.3:** Error messages (e.g., for duplicate dictionary entries, API errors) are clear, easy to understand, and prominently placed near the cause of the error.
  Outcome: OK. Date: 23/05/2025. Comments: clear and understandable error messages.

- **CA6.4:** There is a subtle visual feedback (e.g., color change, slight shadow) when a button is hovered over with the mouse cursor.
  Outcome: OK. Date: 23/05/2025. Comments: hover animation on buttons present.

3. Problems found

Critical issues encountered

- **Incomplete handling of future tense:** The logic for future tense verb conjugation in `TemplateFiller.java` only handles the verbs "is" and "are" specifically, adding "will be". For all other verbs, it simply prepends "will" to the base verb. This approach does not consider the complexities of future-tense conjugation for all types of verb, especially those that are irregular or require different auxiliary forms, leading to grammatically incorrect or unnatural sentences.

- **Improvable verb conjugation handling:** The past tense conjugation logic in `TemplateFiller.java` is based on a `pastTenseVerbs` map and a "verb + ed" fallback for verbs not on the map. This is a very basic approach that does not cover all the irregularities of the English language. For more robust conjugation, a verb conjugation library would be needed.

- **Moderation threshold:** The 0.1 value for text moderation confidence is hardcoded in `NonsenseApplication.java`. This value could be configurable to allow administrators to adjust the sensitivity of moderation without recompiling the code.

- **Verb conjugation issue:** During testing of TemplateFiller.java, it was observed that input phrases must be written in the present tense. Additionally, verbs in the third person singular form (typically ending in `-s`) are not correctly recognized by the application. This results in incorrect syntactic analysis and improper phrase generation, since the system fails to detect third person conjugation.