Software Requirements Specification

for

<Multilingual Toxicity Classification System>

Version 1.0 approved

Prepared by <Illia Keretsman>

<Kharkiv National University of Radioelectronics>

<June 2025>

Table of Contents

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 1

2.1 Product Perspective 1

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 4

3.3 Software Interfaces 4

3.4 Communications Interfaces 4

4. System Features 4

4.1 Text Classification 4

4.2 Model Comparison 4

4.3 TOPSIS Analisys 4

4.4 Translation Layer 5

5. Other Nonfunctional Requirements 5

5.1 Performance Requirements 5

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Initial release | June 2025 | Initial version completed | 1.0 |
|  |  |  |  |

# Introduction

## Purpose

This document specifies the requirements for a web-based machine learning application designed for detecting and classifying toxic text content across multiple languages.

## Document Conventions

Textual requirements are marked as "FR" for functional and "NFR" for non-functional. Priorities: High (H), Medium (M), Low (L).

## Intended Audience and Reading Suggestions

Developers, researchers, educators, and NLP system designers.

## Product Scope

The product enables automatic detection of text toxicity using a trained ensemble of methods. It supports multi-language input via translation, visualizes model performance, and uses TOPSIS analysis for optimal model selection.

## References

* scikit-learn documentation
* TOPSIS theory in MCDM
* FastAPI and React documentation
* TF-IDF vectorization

# Overall Description

## Product Perspective

A client-server system using FastAPI (backend) and React (frontend), leveraging saved ML models.

## Product Functions

* Detect language and translate text if needed;
* Vectorize input text using TF-IDF;
* Apply multiple ML models for toxicity classification;
* Display model predictions and probabilities;
* Rank models using TOPSIS;
* Visual feedback through frontend.

## User Classes and Characteristics

* General users with internet access
* Moderators validating text toxicity
* Researchers analyzing model performance

## Operating Environment

* Browser (Chrome, Firefox, Edge)
* OS: Any
* Python 3.11, Node.js 18+

## Design and Implementation Constraints

* all models are trained exclusively on the Kaggle Toxic Comment Classification Dataset
* The system performs binary classification
* Only TF-IDF vectorization is supported
* Input text is expected to be in English or translated to English before classification
* Only scikit-learn–compatible models are supported

## User Documentation

* Built-in UI with helper messages
* Optional markdown/README file

## Assumptions and Dependencies

* Text is valid UTF-8
* Dataset is binary-labeled
* Internet connection required for translation

# External Interface Requirements

## User Interfaces

The system provides a dark-themed, responsive single-page application with two main interface tabs: Model Performance and Text Analysis.

In the Model Performance view, users can:

* view a list of all trained models and their metrics (accuracy, precision, recall, F1-score, training time)
* see which model is currently ranked highest according to TOPSIS analysis
* check a visual TOPSIS-based ranking of models with efficiency index bars
* refresh model data manually with a button

In the Text Analysis view, users can:

* enter any text for analysis in a large textarea
* submit the text to the backend and get results from all models
* see detected language and translated version (if applicable)
* check individual predictions including probability and toxicity status
* view predefined multilingual examples (safe and toxic) to test the system quickly
* clear input and results with a button
* see summary statistics: total toxic predictions, total safe predictions, average toxicity score

The interface is built using React 19, styled with TailwindCSS, and communicates with the backend using REST over HTTP.

## Hardware Interfaces

No specific hardware requirements

## Software Interfaces

Libraries used: FastAPI, React, Tailwind, scikit-learn, numpy, pandas, googletrans, langdetect

## Communications Interfaces

* RESTful HTTP
* JSON payloads
* CORS enabled for local frontend

# System Features

## Text Classification

* FR-1: The system shall accept user input text. (H)
* FR-2: The system shall detect the language of input. (H)
* FR-3: The system shall translate non-English text to English. (H)
* FR-4: The system shall vectorize text using TF-IDF. (H)
* FR-5: The system shall classify the text using all models. (H)
* FR-6: The system shall return prediction, probability, and is\_toxic flag per model. (H)

## Model Comparison

* **FR-7:** The system shall return all models' accuracy, precision, recall, and F1-score. (H)
* **FR-8:** The system shall show training time. (M)

## TOPSIS Analysis

* **FR-9:** The system shall apply TOPSIS to rank models based on weighted metrics. (H)
* **FR-10:** The system shall return the model with the highest efficiency index. (H)

## Translation Layer

* **FR-11**: The system shall use langdetect to identify the text language. (H)
* **FR-12**: The system shall use googletrans to translate to English before classification. (H)
* **FR-13**: The system shall indicate if translation was applied. (M).

# Other Nonfunctional Requirements

## Performance Requirements

* **NFR-1:** Classification request shall return in ≤2 seconds (excluding translation).
* **NFR-2**: Translation adds ≤3 seconds on average.

## Safety Requirements

* **NFR-3**: The system shall validate that the input text is non-empty.
* **NFR-4**: The system shall handle missing models or vectorizer safely.

## Security Requirements

* **NFR-5**: No user data is stored.
* **NFR-6**: All models are local and immutable after training

## Software Quality Attributes

* **Usability**: modern UI, instant feedback
* **Reliability**: error handling for all API endpoints
* **Maintainability**: modular code structure
* **Portability**: runs on any OS with Python/Node

## Business Rules

* Translation only occurs if language ≠ English
* Only binary classification is supported

# Other Requirements

* Future support: multilingual fine-tuning
* Extendibility to Transformer-based models
* Optional export of analysis as report
* Possible batching API for bulk input