Software Requirements Specification

For

INSTANDER :A HATE SPEECH DETECTION SYSTEM FOR ONLINE SOCIAL NETWORKING

6th of March 2025

Prepared by

Specialization	SAP ID	Roll No.	Name
AIML(NH)	500101976	R2142220195	Vaishnavi Dubey
AIML(NH)	500102209	R2142220125	Parth Soni
AIML(NH)	500105417	R2142220150	Rishi Raj Jain
AIML(NH)	500102177	R2142220061	Chitransh Soni

UPES
UNIVERSITY WITH A PURPOSE

Department of Artificial Intelligence

School Of Computer Science
UPES
DEHRADUN- 248007. Uttarakhand

Software Requirements Specification

For

INSTANDER :A HATE SPEECH DETECTION SYSTEM FOR ONLINE SOCIAL NETWORKING

6th of March 2025

Prepared by

Specialization	SAP ID	Roll No.	Name
AIML(NH)	500101976	R2142220195	Vaishnavi Dubey
AIML(NH)	500102209	R2142220125	Parth Soni
AIML(NH)	500105417	R2142220150	Rishi Raj Jain
AIML(NH)	500102177	R2142220061	Chitransh Soni

UPES
UNIVERSITY WITH A PURPOSE

Department of Artificial Intelligence

School Of Computer Science
UPES
DEHRADUN- 248007. Uttarakhand

Table of Contents

Figure Table

1. INTRODUCTION

1.1. Purpose of the Project

Hate speech on social media and digital platforms has increased significantly, leading to the need for automated detection and moderation tools. Instander is designed to address this challenge by providing a real-time hate speech detection system using machine learning and natural language processing (NLP). The system will analyze text inputs, classify them as hate speech or non-hate speech, and suggest appropriate actions (flagging, removal, or warnings).

1.2. Target Beneficiary

- Social Media Platforms (Facebook, Twitter, Instagram, etc.)
- Online Forums and Communities
- Government and Legal Authorities (monitoring hate speech trends)
- Educational Institutions (ensuring safe online discussions)

1.3. Project Scope

Instander will provide:

- Real-time text classification for detecting hate speech.
- Customisable filters based on the level of offensive content.
- Multi-lingual support for hate speech detection.
- User-friendly dashboard for reports and analytics.

2. PROJECT DESCRIPTION

2.1. Reference Algorithm

1. Hate speech detection using deep learning relies on Convolutional Neural Networks (CNNs) and BERT (Bidirectional Encoder Representations from Transformers) to classify text based on linguistic patterns. The system analyzes textual features such as word embeddings, sentiment, and context to identify and categorize different forms of hate speech.

Working of the Algorithm:

- Text as Input: The system takes user-generated text as input from social media, forums, or messaging platforms.
- Preprocessing: The text undergoes tokenization, stop-word removal, and lemmatization to improve clarity and consistency.
- Feature Extraction using CNNs: CNNs identify n-grams, word embeddings, and semantic patterns that indicate hate speech.

- Contextual Understanding with BERT: BERT improves deep learning by using bidirectional attention, allowing it to understand nuanced and context-dependent hate speech.
- Prediction & Output: The model classifies the text as neutral, offensive, or hate speech and suggests appropriate moderation actions (flagging, removal, or warnings).

Key Steps of CNN and BERT in Hate Speech Detection:

1. Text Preprocessing
    - Tokenize sentences into words or subwords.
    - Remove stopwords and apply lemmatization.
    - Convert text into numerical embeddings.
2. Feature Learning using CNNs
    - Convolutional layers extract text features, word associations, and contextual meanings.
    - Pooling layers reduce feature dimensions while preserving critical language structures.
    - Activation functions (ReLU, Softmax) classify text effectively.
3. Improved Performance with BERT
    - BERT uses bidirectional transformers to understand both preceding and succeeding words in a sentence.
    - Prevents loss of contextual meaning, improving classification accuracy.
4. Classification and Prediction
    - The fully connected layers of the model process extracted features.
    - The final layer predicts whether the text is neutral, offensive, or hate speech.

Why Use CNN and BERT for Hate Speech Detection?

- High Accuracy: CNNs and BERT outperform traditional methods by learning context-dependent patterns in text.
- Automatic Feature Extraction: Unlike rule-based methods, deep learning models detect implicit hate speech.
- Robustness to Variations: The system works efficiently across different languages, slang, and dialects.
- Efficient Training and Classification: BERT enhances text understanding, making real-

time predictions feasible.

2.2. Data/Data Structure

The system integrates structured text data and deep learning models to efficiently detect hate speech.

1. Hate Speech Dataset:
   The dataset consists of labeled textual data classified as hate speech, offensive language, or neutral text. It is sourced from online social media comments, tweets, and discussions.

- Sources: Hate Speech and Offensive Language Dataset
- Preprocessing Steps:
  - Tokenize text into words or subwords.
  - Remove stopwords and lemmatize text.
  - Convert text into numerical embeddings for processing.

2. Convolutional Neural Networks (CNN) for Feature Extraction:
   CNNs serve as the primary deep learning architecture to extract textual features.

- Convolutional Layers: Identify patterns in text, such as offensive phrases and context.
- Pooling Layers: Reduce dimensionality while retaining important semantic structures.
- Dropout Layers: Prevent overfitting by randomly ignoring some neurons during training.
- Fully Connected Layers: Transform extracted features into meaningful classifications.

3. BERT for Enhanced Performance:
   BERT improves contextual understanding by utilizing deep bidirectional learning.

- Prevents Context Loss: Ensures words are understood in relation to surrounding text.
- Improves Accuracy: Allows the model to detect subtle forms of hate speech.
- Handles Large Datasets Efficiently: Processes large-scale textual data for real-time analysis.

By combining CNN and BERT, this project provides a fast, accurate, and scalable solution for identifying hate speech. The system helps online platforms moderate content effectively, reducing harmful interactions and fostering a safer online environment.

2.3.SWOT Analysis

Strengths:

- Real-time detection with high accuracy.
- Scalable for large datasets.
- Supports multiple languages.

Weaknesses:

- May struggle with context-based hate speech.
- Requires continuous model updates.

Opportunities:

- Integration with major social media platforms.
- Can be extended for cyberbullying detection.

Threats:

- Evolving hate speech tactics.
- Ethical concerns regarding censorship.

2.4.Project Features

The Instander Hate Speech Detection System leverages deep learning techniques to help social media platforms, government agencies, and online communities detect and mitigate hate speech, ensuring a safer digital environment. Its key features include:

1. Text-Based Hate Speech Identification: Uses Convolutional Neural Networks (CNNs) and BERT to analyze text and accurately detect hate speech.
2. Automated Content Moderation: The system processes user-generated text to classify hate speech and provides confidence scores, reducing the need for manual moderation.
3. Pre-Trained Model Integration: Utilizes pre-trained deep learning models to enhance accuracy and efficiency in hate speech classification.
4. User-Friendly Interface: Designed for ease of use, allowing moderators to input text manually or integrate with social media APIs for real-time monitoring.

5. Hate Speech Categorization & Response Suggestions: Once hate speech is detected, the system categorizes it (e.g., racism, sexism, threats) and provides recommended moderation actions.

6. Scalability & Adaptability: Can be extended to support multiple languages and emerging hate speech patterns by training on diverse datasets, making it adaptable for various platforms.

2.5. User Classes and Characteristics

This deep learning-based Hate Speech Detection Tool is designed to serve multiple user groups, each with distinct needs and functionalities:

- Social Media Moderators & Content Review Teams: The primary users responsible for detecting and moderating hate speech on platforms. They need an intuitive interface that provides real-time classifications, confidence scores, and response suggestions.

- Researchers & Data Scientists: Specialists analyzing hate speech patterns for AI and linguistic advancements. They may require detailed classification reports, model accuracy insights, and dataset analytics to improve detection performance.

- Government & Legal Authorities: Agencies monitoring online hate speech for compliance, law enforcement, and policy-making. They may need access to trend analysis, flagged content reports, and predictive analyticsto enforce regulations effectively.

- System Administrators: Professionals responsible for maintaining and updating the detection models, managing datasets, and ensuring the system remains accurate and unbiased. They require backend access for model retraining, data curation, and software enhancements.

By catering to these diverse user groups, Instander enhances online safety, supports data-driven decision-making, and contributes to ethical content moderation across digital platforms.

2.6. Design and Implementation Constraints

- Requires a large labeled dataset for training.

- High computational resources for deep learning models.

- Continuous updates to adapt to new hate speech forms.

2.7.Design Diagram – Use case

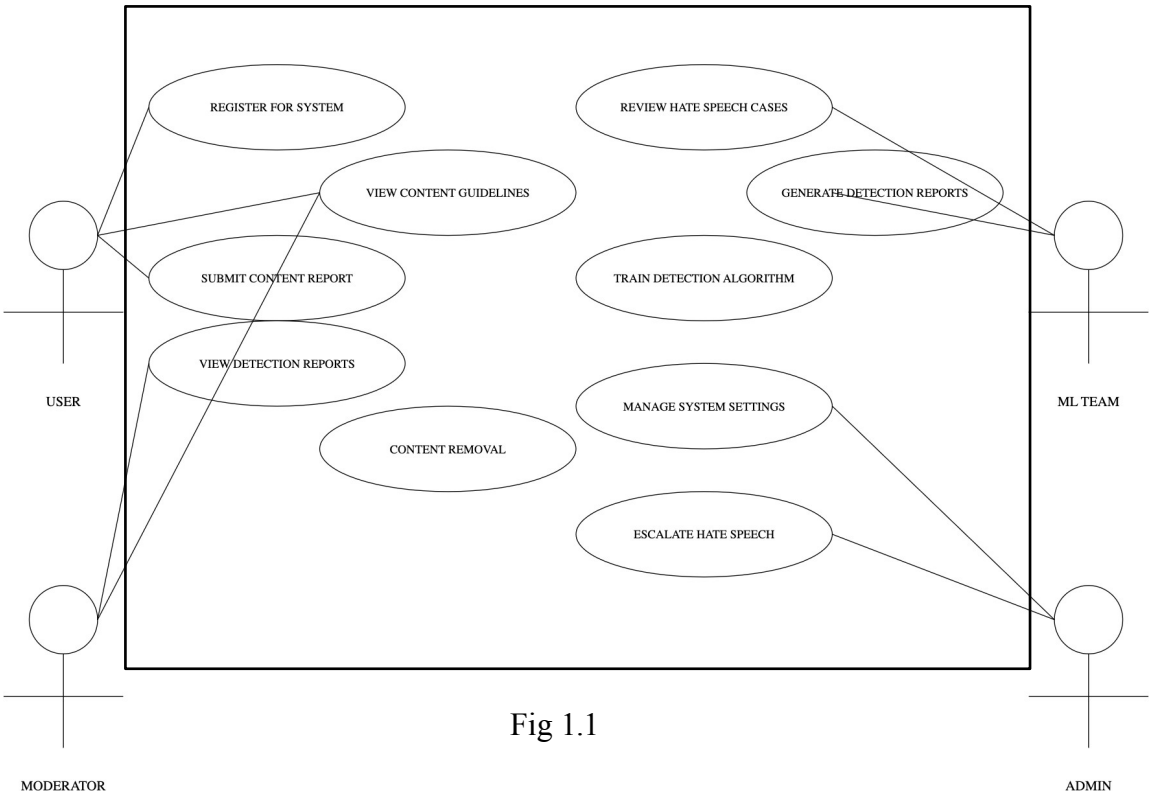

Fig 1.1

2.8.Sequence Diagram

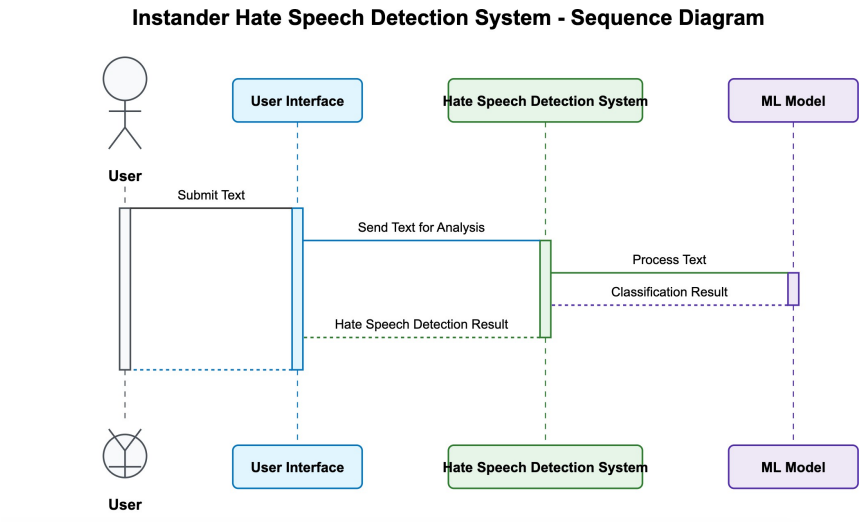

Fig 1.2

2.9. Assumptions and Dependencies

The successful development and deployment of Instander rely on several key assumptions and dependencies. These factors ensure the efficient functioning of the hate speech detection and conversion system.

Assumptions

1. Internet Connectivity for Cloud-Based ML Models:
    - The system assumes stable internet connectivity for accessing and utilizing cloud-based machine learning models. Since Instander may rely on cloud-hosted AI models, disruptions in internet service could impact real-time hate speech detection and conversion.
2. Use of Pre-Trained Models Like BERT:
    - The system assumes the availability of pre-trained NLP models such as BERT (Bidirectional Encoder Representations from Transformers) to enhance detection accuracy. These models will be fine-tuned on a custom hate speech detection corpus to improve performance.
3. Ethical Considerations and Bias Management:
    - The system assumes that datasets used for training and fine-tuning models are representative and free from significant biases. Continuous monitoring and updates are required to mitigate biases in hate speech classification.
4. User Input in Natural Language:
    - The system expects that user input consists of text-based content in a structured form, such as chat messages, social media posts, or comments. It assumes that speech-to-text conversion (if implemented) provides a reliable textual representation for processing.
5. Scalability for Large-Scale Data Processing:
    - The system assumes it can handle a growing volume of text data from various platforms, requiring scalable cloud infrastructure for real-time processing.
6. Regulatory Compliance:
    - The system assumes compliance with data privacy laws (e.g., GDPR, CCPA) and ethical AI guidelines. It must ensure user data confidentiality and avoid misuse.

Dependencies

1. Cloud-Based Infrastructure (AWS, GCP, or Azure):
    - The system depends on cloud platforms like Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure for deploying ML models, handling requests, and ensuring seamless scalability.
2. Machine Learning Libraries and Frameworks:
    - Instander relies on frameworks like TensorFlow, PyTorch, Hugging Face Transformers, and Scikit-learnfor model training, inference, and fine-tuning of

pre-trained models like BERT.

3. Hate Speech Detection Corpus:

- The project depends on a well-curated hate speech detection dataset, which includes labeled instances of offensive and non-offensive speech for effective model training. This dataset will be sourced from existing repositories (e.g., HateBase, Kaggle) and enhanced with custom annotations.

4. Natural Language Processing (NLP) APIs:
    - The system may integrate NLP APIs such as Google Cloud Natural Language API or OpenAI's GPT models for advanced language understanding and processing.
5. Real-Time Data Processing Pipelines:
    - The project depends on technologies like Apache Kafka, RabbitMQ, or Firebase to handle real-time message streaming and content moderation workflows.
6. User Interface and Experience (UI/UX):
    - Instander relies on a web or mobile interface built using frameworks like React, Flutter, or Android/iOS SDKs to deliver an intuitive and user-friendly experience.
7. Security and Authentication Services:
    - The system depends on OAuth, JWT authentication, or biometric authentication to ensure secure user access and prevent unauthorized usage.
8. Regulatory and Compliance Tools:
    - Instander may depend on legal compliance frameworks and tools to ensure that flagged content aligns with platform policies and legal requirements.

By addressing these assumptions and dependencies, Instander can function effectively as a hate speech detection and conversion platform, leveraging cloud-based AI and pre-trained NLP models for enhanced accuracy.

3. SYSTEM REQUIREMENTS

3.1. User Interface

Main Dashboard

- A user-friendly home screen providing access to key features, including hate speech detection, conversion, flagged content review, and historical analysis of detected content.

Text Input & Media Upload Interface

- Allows users to input text or upload media (comments, posts, messages) for analysis.
- Supports real-time typing analysis and bulk content evaluation.

Hate Speech Detection Result Display

- Shows detection results with labels such as Hate Speech, Offensive Speech, or Neutral Speech along with a confidence score and highlighted flagged words.
- Includes severity levels (Low, Moderate, High) and context-based explanations.

Hate Speech Conversion & Recommendation Module

- Suggests alternative phrasing or neutral language for flagged content.
- Provides recommendations based on AI-generated sentiment analysis.

Database & Learning Hub

- Offers educational content on hate speech patterns, societal impacts, and ethical language use.
- Includes real-world case studies and guidelines for responsible communication.

User Notifications & Alerts

- Pop-ups or alerts for users when their content is flagged.
- Notifications about trending hate speech patterns, policy updates, and AI improvements.

Settings & Language Preferences

- Allows users to set preferred languages, sensitivity levels for detection, and notification preferences.
- Provides multilingual support for diverse user demographics.

Admin Panel for Data Updates

- A backend dashboard for moderators and experts to update hate speech datasets, refine model parameters, and improve AI accuracy.
- Provides insights into flagged content trends and emerging hate speech patterns.

3.2. Software Interface

Text Processing & Feature Extraction Module

- Handles text tokenization, stopword removal, stemming/lemmatization, and feature extraction.
- Prepares input data for hate speech classification models.

Deep Learning Model Module

- Uses BERT, LSTM, or Transformer-based models for hate speech classification.
- Outputs class labels, severity scores, and explanation-based insights.

Hate Speech Detection & Classification Module

- Analyzes text input to determine whether the content contains hate speech, offensive language, or neutral speech.
- Classifies severity levels and provides contextual analysis.

Text Conversion & Recommendation Module

- Works alongside the detection module to suggest alternative, neutral, and non-offensive wording.
- Uses NLP-based paraphrasing and sentiment adjustment techniques.

User Interface Module

- Retrieves detection results and alternative suggestions for front-end display.
- Supports interactive UI elements for user engagement and education.

Admin Panel & Database Management Module

- Enables hate speech dataset updates, model retraining, and refinement based on user feedback.
- Stores historical flagged content for trend analysis and accuracy improvements.

3.3. Database Interface

- Unlike cloud-based applications, Instander does not rely on external databases or APIs for processing.
- Instead, all hate speech detection data is pre-stored within the application's structured files in formats like JSON, CSV, or SQLite.
- The system reads and writes flagged content data using local storage and Python-based file handling mechanisms.

3.4. Protocols

- No external network protocols are involved.
- All operations, including text analysis, classification, and conversion, are performed locally on the device.
- The system reads text data from predefined input sources, processes it in memory, and generates output for the user.
- This ensures offline functionality and privacy-focused hate speech detection without reliance on cloud-based APIs.

4. NON-FUNCTIONAL REQUIREMENTS

4.1. Performance Requirements

Response Time:

- The app should detect and classify hate speech within 1-2 seconds for a seamless user experience.
- Hate speech conversion and alternative suggestion generation should take no more than 3 seconds.

Memory Usage:

- The application should optimize memory usage, operating efficiently within 100-200 MB to accommodate NLP models.

Scalability:

- The system must handle an increasing number of text inputs and real-time content processing without significant performance degradation.
- The backend should support batch processing for large-scale text datasets.

Accuracy:

- The hate speech detection model should achieve at least 90-95% accuracy in distinguishing between hate speech, offensive language, and neutral speech.
- The conversion model should generate contextually appropriate, non-offensive alternatives with high semantic retention.

4.2. Security Requirements

Operating System:

- The application should run on modern operating systems, including Windows, macOS, and Linux for desktop applications.
- If deployed as a mobile or web app, it should be compatible with Android, iOS, and

major web browsers.

Python Runtime Environment:

- The application requires Python 3.8 or higher to ensure compatibility with NLP frameworks like Hugging Face Transformers, TensorFlow, and PyTorch.

GPU Acceleration:

- For faster processing, the application can leverage NVIDIA GPUs with CUDA and cuDNN support.
- It should also be capable of running on CPUs, albeit with slightly longer inference times.

Hardware Specifications:

- Minimum 2 GB of RAM and 200 MB of free disk space are required for smooth operation.
- Cloud-based deployments should support auto-scaling to handle increased traffic.

Data Privacy & Security:

- User-uploaded text or content should be processed locally when possible to ensure privacy.
- If cloud-based, secure API endpoints and encryption mechanisms (TLS, AES-256) must be implemented.

4.3. Software Quality Attributes

Reliability:

- The model should consistently detect hate speech with minimal false positives and false negatives.
- Regular dataset updates and model retraining should improve accuracy over time.

Usability:

- The user interface should be intuitive, allowing users to easily analyze text, view flagged content, and receive recommendations.
- It should support multilingual analysis for broader accessibility.

Performance:

- The system should process text input in real-time, ideally within 1-2 seconds.
- Background processes (e.g., dataset updates, model retraining) should be optimized to avoid disrupting user experience.

Scalability:

- The system should efficiently handle high volumes of text input, including batch processing of large social media datasets.
- It should support expanding vocabulary and hate speech variations across different languages.

Security:

- The system must ensure secure handling of user data and prevent unauthorized access.
- If deployed online, strict role-based access control (RBAC), data encryption, and API security measuresshould be in place.

5. OTHER REQUIREMENTS

Maintainability:

- The codebase should be modular, well-documented, and easy to update, allowing developers to:
  - Add new hate speech categories or languages.
  - Integrate more advanced NLP models as they become available.
  - Fine-tune detection accuracy using continuous dataset updates.
- The system should use API-based architecture for easy integration into other platforms (e.g., social media moderation tools).

Usability:

- The user interface should be intuitive and accessible, ensuring that users can easily:
  - Input text for analysis.
  - View detection results with explanations.
  - Receive alternative phrasing suggestions for flagged content.
- Explanations should be clear, helping users understand why a text was flagged and providing constructive alternatives.
- The platform should support multilingual analysis to cater to diverse audiences.

Regulatory Compliance:

- The system must adhere to relevant data privacy and security regulations, such as:
  - GDPR (General Data Protection Regulation) for EU-based users.
  - CCPA (California Consumer Privacy Act) for US-based users.
  - Other region-specific data protection laws.
- User-uploaded text and metadata should be securely handled, anonymized (if necessary), and not stored without consent.
- If deployed on social media platforms or used by organizations, compliance with platform-specific policies is required.

Deployment Considerations:

- The model should be deployable in multiple environments, including:
  - Cloud-based platforms for real-time, large-scale analysis and moderation.
  - On-device (offline) processing for users in regions with limited internet connectivity.
  - Edge computing support to enable real-time detection on social media and messaging platforms.
- The system should support APIs for easy third-party integration, allowing social media platforms, online forums, or educational institutions to embed the hate speech detection functionality.

6. REFERENCES

1. D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., 2019.

2. A. Schmidt and M. Wiegand, "A survey on hate speech detection using NLP," 2017.

3. Y. Kim, "Convolutional neural networks for sentence classification," 2014.

****