Graphic Era Hill University
DEHRADUN • BHIMTAL • HALDWANI

PROJECT AND TEAM INFORMATION

Project Title

AI-Generated Content Detection Algorithm Using Data Structures and Algorithms (DSA)

Student/Team Information

Team Name:	Sapphire
Team member 1 (Team Lead)	Atharv Gangwar – 230112386 atharvgangwar8@gmail.com
Team member 2	Dhruv Negi – 23011451 negi67291@gmail.com

Team member 3	Nivedan Belwal – 230111118 nivedanbelwal627@gmail.com
Team member 4	Aditya Awasthi – 23011858 adityaawasthi069@gmail.com

PROJECT PROGRESS DESCRIPTION

Project Abstract

The AI Content Detector web application is designed to analyze text and determine whether it is human-written or AI-generated. This system integrates multiple data structures and algorithms (DSA) and design and analysis of algorithms (DAA) approaches, including Hashing, Trie-based phrase matching, Heap analysis, and Dynamic Programming (Edit Distance) for similarity analysis. The system allows users to input text and provides detailed results based on multiple algorithmic models to increase accuracy and robustness. The frontend is built using React (Vite) and the backend leverages advanced AI/ML preprocessing with extensive DSA techniques.

Updated Project Approach and Architecture

The architecture follows a client-server model. The frontend is built in React (Vite.js) and provides an interactive UI to upload or enter text. The backend uses Python(API)/Typescript incorporating multiple DSA/DAA algorithms.

- Hashing: Calculates word frequency and detects abnormal distributions.
- Trie: Matches known AI-generated phrases.
- Heap: Ranks frequent word clusters.
- Edit Distance (Dynamic Programming): Calculates similarity scores between user input and AI-generated patterns.
- Frontend libraries include React, TailwindCSS, Lucide-react icons.

Tasks Completed

Task Completed	Team Member
1. Codebase Design and Workflow	Atharv Gangwar and Dhruv Negi
2. Data Collection	Dhruv Negi
3. Complete Frontend	Atharv Gangwar
4. Coding and Implementing Data Structure	Nivedan Belwal
5. Scoring System (classify.py)	Aditya Awasthi

Challenges/Roadblocks

- **Finding Perfect Data**: One of the primary challenges we face is sourcing high-quality, diverse datasets that accurately represent both AI-generated and human-written text. The lack of comprehensive datasets can hinder the training and validation of our algorithm, making it difficult to achieve reliable results.

- **Achieving Perfect Accuracy**: Despite our multi-faceted approach, we are still struggling to attain optimal accuracy in distinguishing between AI-generated and human-written content. Variability in writing styles, context, and the evolving nature of AI text generation complicate the detection process, leading to potential misclassifications.

- **Selecting the Right Data Structures**: Choosing the most effective data structures for our algorithm is crucial for performance and efficiency. While we have implemented HashMaps, Tries, and Heaps, optimizing their usage and ensuring they work harmoniously within the algorithm remains a challenge. Inefficient data structures can lead to increased processing time and reduced accuracy.

- **Handling Ambiguity in Language**: Natural language is inherently ambiguous and context-dependent. This complexity poses a significant roadblock in accurately interpreting the nuances of text, making it challenging to develop a robust detection mechanism that can adapt to various writing styles and contexts.

- **Scalability and Performance**: As the volume of text input increases, ensuring that our system remains scalable and performs efficiently is a concern. We need to optimize our algorithms and data structures to handle larger datasets without compromising response time or accuracy.

- **User Experience**: Designing an intuitive and responsive frontend that effectively communicates the results of the analysis poses its own set of challenges. Ensuring that users can easily understand the confidence scores and labels is essential for the system's usability and acceptance.

Tasks Pending

Task Pending	Team Member (to complete the task)
Extended testing with larger and diverse datasets without compromising with time complexity	Team

Project Outcome/Deliverables

- Fully functional AI Content Detector web application.
- Hybrid multi-algorithm analysis system using DSA/DAA.
- Clean frontend UI with detailed algorithm score breakdown.
- Code repository for future academic or commercial use.
- Testing framework for ongoing validations.

Progress Overview

We have successfully created a prototype for this algorithmic system. Since the full-scale version requires high computing power for analyzing very large datasets, we developed a scaled-down version to effectively demonstrate the algorithm's working and showcase its potential capabilities. This prototype validates the feasibility of using DSA and DAA techniques for AI content detection.

Codebase Information

(Repository link, branch, and information about important commits.)

Repository Link: https://github.com/AtharvGangwar48/AI_Content_Detector
Branch: Main
Commits: “Project Completed Succesfully” (18th June 2025)
(Total 10 commits till 18th June 2025)

Project Link: “https://ai-generated-content-eta.vercel.app”

Testing and Validation Status

Test Type	Status (Pass/Fail)	Notes
Unit Testing	Pass	Each component of the algorithm has undergone rigorous unit testing to verify its functionality in isolation. This includes testing the HashMap for word frequency analysis, the Trie for phrase detection, and the Edit Distance algorithm for text similarity.
Integration Testing	Pass	This phase has focused on ensuring that the backend API correctly processes input from the frontend and returns accurate confidence scores and labels. Any discrepancies identified during this phase have been addressed to enhance the overall system performance.
Route Testing	Pass	Each route in the application was thoroughly tested to ensure proper functionality. Route testing focuses on verifying the backend API endpoints that process the input text and return the classification results. The key objective is to ensure that the routes function correctly, handle different input cases gracefully, and return appropriate responses.
Responsive Testing	Pass	All pages of the application were tested for responsiveness across different devices and screen sizes. The layout and functionality were verified to ensure a consistent and user-friendly experience on desktops, tablets, and mobile devices.
Validation with Benchmark Datasets	Pass	The algorithm has been validated using benchmark datasets that include both AI-generated and human-written texts. Initial results indicate a promising accuracy rate; however, further refinement is needed to improve performance, particularly in edge cases.

Deliverables Progress

Algorithm Implementation: The text analysis algorithm is implemented using various data structures, including HashMaps, Tries, and the Edit Distance algorithm. Initial testing indicates promising results, though further refinement is needed to enhance accuracy.

Frontend Development: The React-based frontend is under development, featuring a user-friendly interface for text input and result display. Integration with the backend API is in progress, enabling seamless user interaction.

We have successfully implemented a prototype of our AI Content Detector system that demonstrates the functionality of our hybrid algorithm using Hashing, Trie, Heap, and Dynamic Programming (Edit Distance). This prototype effectively analyzes smaller datasets and accurately identifies AI-generated content based on multiple algorithmic signals. Due to the computational intensity required for processing larger datasets, the current version operates as a demonstration model. With further optimization and access to higher computing resources, this system can be scaled into a full-fledged application capable of handling extensive real-world datasets with high accuracy and efficiency.