

# GIRNE AMERICAN UNIVERSITY

**Faculty of Engineering**

**Department of Computer Engineering**

**AcadCheck: AI and Plagiarism Detection Tool**

**SUMMER TRAINING REPORT**

Student: Elie katende kazwela

Student ID: 221701001

Supervisor: Mr. Cliff

# Acknowledgements

I would like to sincerely thank all those who supported me during my internship at the university and throughout the preparation of this report.

First and foremost , I am especially grateful to my supervisor , Mr. Peter Cliff, for his guidance ,encouragement and constant support. His expertise and thoughtful feedback played a key role in the progress and success of my work.

I also wish to express my appreciation to the staff and faculty of the university for welcoming me ,

Providing the necessary resources, and creating a positive environment that made this internship a valuable learning experience. the knowledge and skills I gained have been highly beneficial to my academic and professional growth.

Lastly ,I would like to thank my family and friends for their patience , encouragement , and continuous

Support , which greatly helped me in completing this project.

# Abstract

This report introduces AcadCheck, a web application developed to detect both plagiarism and AI generated content in academic documents . the system was implemented using React, TypeScript, and Supabase, and it integrates advanced document analysis methods with a user-friendly interface to deliver precise and comprehensive results.

The main objectives of the project are structured around four core component :

Plagiarism detection : comparing documents against a large corpus by applying N-grams, semantic similarity, and pattern-matching techniques.

AI content detection – identifying AI-generated text through transformer-based models and linguistic analysis.

Document analysis and reporting – producing detailed, sentence-level feedback with metrics such as lexical diversity, syntactic complexity, semantic coherence, and perplexity.

Secure and scalable architecture – leveraging Supabase for database management, data protection, and Row Level Security (RLS).

This report presents the overall architecture , development process ,challenges ,faced and evaluation results of the application ,while also reflecting on the ethical considerations involved .it highlights the use of modern approaches to document analysis , the integration of artificial intelligence in plagiarism detection ,and the design of scalable web applications.

Table of Contents

[GIRNE AMERICAN UNIVERSITY 1](#_Toc207196669)

[Acknowledgements 2](#_Toc207196670)

[Abstract 3](#_Toc207196671)

[Chapter 1: Introduction 9](#_Toc207196672)

[1.1 Background 9](#_Toc207196673)

[1.2 Objectives 10](#_Toc207196674)

[1.3 Scope 11](#_Toc207196675)

[1.4 Importance of the Study 11](#_Toc207196676)

[1.5 Structure of the Report 12](#_Toc207196677)

[CHAPTER 2: LITERATURE REVIEW 13](#_Toc207196678)

[2.1 Overview of Plagiarism Detection 13](#_Toc207196679)

[2.2 AI-Based Text Detection 14](#_Toc207196680)

[2.3 Document Analysis Techniques 14](#_Toc207196681)

[2.4 Ethical and Legal Considerations 15](#_Toc207196682)

[2.5 Use of React, TypeScript, and Supabase in Web Apps 15](#_Toc207196683)

[2.6 Integration of Detection Algorithms in Web Applications 16](#_Toc207196684)

[2.7 Summary 17](#_Toc207196685)

[CHAPTER 3: METHODOLOGY 18](#_Toc207196686)

[3.1 Design Objectives and System Architecture 18](#_Toc207196687)

[3.2 Tools and Libraries 19](#_Toc207196688)

[3.3 Development and Implementation 20](#_Toc207196689)

[3.3.1 User Authentication and Session Management 20](#_Toc207196690)

[3.3.2 Document Upload and Preprocessing 20](#_Toc207196691)

[3.3.3 Plagiarism Detection Algorithms 21](#_Toc207196692)

[3.3.4 AI Content Detection 21](#_Toc207196693)

[3.3.5 Analysis Report Generation 21](#_Toc207196694)

[3.3.6 Internationalization 22](#_Toc207196695)

[3.3.7 Security and RLS Policies 22](#_Toc207196696)

[3.4 Packaging and Deployment 22](#_Toc207196697)

[3.5 Testing and Quality Assurance 23](#_Toc207196698)

[3.6 Ethical and Legal Considerations 23](#_Toc207196699)

[3.7 Summary 23](#_Toc207196700)

[CHAPTER 4: IMPLEMENTATION AND DISCUSSION 24](#_Toc207196701)

[4.1 Development Environment Setup 24](#_Toc207196702)

[4.2 Frontend Implementation 24](#_Toc207196703)

[4.2.1 Dashboard and Document Upload 24](#_Toc207196704)

[4.2.2 Interactive Analysis Report 25](#_Toc207196705)

[4.3 Backend Implementation 26](#_Toc207196706)

[4.3.1 Document Storage 26](#_Toc207196707)

[4.3.2 Analysis Workflow 26](#_Toc207196708)

[4.4 Document Analysis Workflow 27](#_Toc207196709)

[4.5 AI and Plagiarism Detection Results 27](#_Toc207196710)

[4.6 Challenges and Troubleshooting 28](#_Toc207196711)

[4.7 Ethical and Legal Considerations 29](#_Toc207196712)

[4.8 Summary 29](#_Toc207196713)

[CHAPTER 5: CONCLUSION 30](#_Toc207196714)

[5.1 Key Findings 30](#_Toc207196715)

[5.2 Educational Value 30](#_Toc207196716)

[5.3 Limitations 31](#_Toc207196717)

[5.4 Recommendations for Future Work 32](#_Toc207196718)

[5.5 Final Thoughts 33](#_Toc207196719)

[REFERENCES 33](#_Toc207196720)

[APENDIX 36](#_Toc207196721)

[Figure A.1 – Login and Authentication Interface 36](#_Toc207196722)

[Figure A.2 – Main Dashboard Overview 36](#_Toc207196723)

[Figure A.3 – Document Preview (PDF/DOCX/TXT) 37](#_Toc207196724)

[Figure A.4 – Plagiarism and AI Detection Results 38](#_Toc207196725)

[Figure A.5 – Detailed Report Generation 39](#_Toc207196726)

[Figure A.6 – Analysis History and Filtering 40](#_Toc207196727)

|  |
| --- |
| List of Abbreviations |
| | Abbreviation |  | Meaning | | --- | --- | --- | | UI |  | User Interface | | AI |  | Artificial Intelligence | | RLS |  | Row Level Security | | PDF |  | Portable Document Format | | DOCX |  | Microsoft Word Document | | ML |  | Machine Learning | | DB |  | Database | | API |  | Application Programming Interface | | JS |  | JavaScript | | TS |  | TypeScript | |

|  |
| --- |
| List of Symbols |
| | Symbol | Meaning | | --- | --- | | ↑ | Increase / Upload | | ↓ | Decrease / Download | | ≈ | Approximately equal | | % | Percentage | | ∑ | Summation | | λ | Lambda function or rate | | ∂ | Partial derivative | | ∆ | Change or difference | |

# Chapter 1: Introduction

## Background

In academic settings, it is crucial to uphold the integrity of written work. The increased use of AI tools, online resources, and AI-assisted content creation has raised the risk of plagiarism and improper use of external materials. AcadCheck addresses these challenges by giving an automated solution that can identify both plagiarized content and text generated by artificial intelligence. This helps maintain academic honesty and the quality of scholarly work. Modern document analysis uses technologies like machine learning, semantic evaluation, and natural language processing (NLP). By combining these techniques with a simple and user-friendly web interface, AcadCheck allows educators, students, and researchers to analyze documents effectively while keeping their data secure and private.

## 1.2 Objectives

The main goals of this internship project are:

- To create a web application that can identify both plagiarism and AI-generated content.

- To integrate analysis techniques that combine machine learning with heuristic methods.

- To provide detailed reports that include sentence-level analysis, confidence indicators, and clear visual cues.

- To ensure secure handling of user data through authentication, Row-Level Security (RLS), and protected storage.

- To offer a user-friendly and responsive interface that effectively supports multilingual users.

## 1.3 Scope

This project focuses on designing, developing, and implementing AcadCheck as a tool for research and education. The application works with PDF, DOCX, and TXT file formats and provides detailed sentence-level analysis for both plagiarism and AI-generated content. All testing and use of the application are conducted ethically, in controlled environments, and in strict compliance with data protection regulations. The project does not involve use in untrusted environments or collecting unauthorized data.

## 1.4 Importance of the Study

AcadCheck contributes to multiple domains:

Academic integrity: Helps institutions maintain originality standards.

AI literacy: Educates users on distinguishing AI-generated content.

Technical education: Demonstrates the use of modern web technologies, machine learning, and database security in practical applications.

By integrating AI detection with plagiarism analysis, the project addresses modern challenges in digital content evaluation, providing insights for students, educators, and software developers.

## 1.5 Structure of the Report

The organization of the report is:

* **Chapter 1:** Introduction – Presents background, objectives, scope, and significance.
* **Chapter 2:** Literature Review – Reviews plagiarism detection methods, AI content detection, and ethical/legal considerations.
* **Chapter 3:** Methodology – Details the system architecture, tools, libraries, and implementation steps.
* **Chapter 4:** Implementation and Discussion – Covers code implementation, document processing, analysis workflow, and troubleshooting.
* **Chapter 5:** Conclusion – Summarizes findings, limitations, educational value, and future improvements.
* References – List of sources cited throughout the report.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Overview of Plagiarism Detection

Detecting plagiarism is essential for upholding academic integrity. Traditional tools mainly rely on exact text matching, which can spot direct copying but often misses paraphrased or slightly altered content. Modern systems, like AcadCheck, use a combination of syntactic , semantic, and contextual analysis, enabling more accurate detection of potential plagiarism.

N-gram Analysis: Documents are broken into sequences of words (n-grams) and compared to a reference corpus. Similar sequences indicate potential plagiarism. AcadCheck uses adaptive n-gram sizes to capture both short phrases and longer patterns.

Semantic Similarity: Using NLP techniques, AcadCheck compares the meaning of sentences rather than exact wording. This allows the system to detect paraphrased content.

Contextual Patterns: Certain academic formulations are commonly reused. By identifying these patterns, the system can differentiate between general academic phrases and copied unique content.

Recent research emphasizes the integration of machine learning models for more nuanced plagiarism detection, allowing systems to learn from labeled datasets and improve detection accuracy over time.

## 2.2 AI-Based Text Detection

With the rise of AI-generated contents, especially from models like chatGPT, distinguishing between human-written and AI-generated text has become a key challenge. AcadCheck uses hybrid AI detection techniques that combine transformer-based models with linguistic heuristics: - **Transformer Models:** Pre-trained models analyze text for patterns that suggest AI generation, such as an overuse of certains connectors, predictable wording, or low variability.

**Linguistic Heuristics:** Metrics like lexical diversity, syntactic complexity, and perplexity evaluate the naturalness of text. AI-generated text often shows high uniformity in perplexity and low diversity in vocabulary.

**Sentence-Level Scoring**: Each sentence receives an AI score that indicates the likelihood of it being machine-generated. This detail lets users review suspicious content precisely. By combining AI detection with plagiarism analysis, AcadCheck offers a thorough evaluation of document originality.

## 2.3 Document Analysis Techniques

Effective plagiarism and AI detection depend on solid document analysis. AcadCheck supports various formats, including: PDF, DOCX, and TXT, ensuring consistent text extraction:

**PDF Analysis**: Using PDF.js, the system extracts text while keeping the document's structure intact.

**Word Document Handling:** Mammoth and DOCX Preview converted DOCX files to HTML, ensuring accurate rendering and analysis at the sentence level.

**Text Preprocessing**: Cleaning, tokenization, and normalization prepare the content for further analysis. Sentence segmentation, feature extraction, and vectorization occur before any similarity or AI detection algorithms, ensuring both accuracy and interpretability in results.

## 2.4 Ethical and Legal Considerations

Given the sensitivity of plagiarism and AI detection, ethics and legality are central to AcadCheck’s design:

**User Consent:** Only documents uploaded with clear consent are analyzed.

**Data Security:** Supabase provides encrypted storage, and Row Level Security (RLS) ensures user data is isolated.

**Legal Compliance:** The system aligns with GDPR and other data protection laws, preventing unauthorized access or retention of data.

**Responsible Reporting:** Results are shared only for educational or institutional purposes, preventing misuse of the analysis. Ethical considerations also guide the AI detection aspect. The system prioritizes transparency, offering users clear metrics and avoiding unfair assumptions.

## 2.5 Use of React, TypeScript, and Supabase in Web Apps

Modern web technologies are vital for the functionality and usability of AcadCheck:

**React (v18.3.1):**Provides a responsive, dynamic front end capable of real-time document previews and interactive dashboards.

**TypeScript:** Ensures type safety and improves maintainability, especially for complex components like analysis tables and reports.

**Supabase:** Acts as a backend and database solution, offering authentication, secure storage, and serverless functions. RLS ensures users access only their own analyses.

**Frontend UI Components:**Using Tailwind CSS and shadcn/ui, the interface is modern, clean, and easy to use. Components like DocumentViewer and HighlightedText give users detailed visual feedback. The combination of thoses technologies creates a robust, scalable, and secure web application that supports complex AI and plagiarism detection workflows while providing a positive user experience.

## 2.6 Integration of Detection Algorithms in Web Applications

A key part of modern plagiarism detection systems is the smooth integration of backend analysis algorithms with frontend visualizations:

**Sentence-Level Analysis**: Each sentence is evaluated for plagiarism and AI content. Results are sent from the backend to the frontend, where they are highlighted visually in the document. **Scoring and Metrics:** Globale scores, confidence levels, and detailed metrics are displayed using interactive dashboards.

**Real-Time Feedback:** Users can upload documents and get analysis results without having to reload the page, thanks to React’s component structure. This integration ensures that complex analysis does not sacrifice usability.

## 2.7 Summary

The Chapter 2 demonstrates that modern plagiarism and AI detection use layered methods combining semantic analysis, machine learning, and secure web technologies. AcadCheck applies these principles by providing an easy-to-use, secure, and accurate system that effectively addresses today’s challenges in maintaining academic integrity.

# CHAPTER 3: METHODOLOGY

## 3.1 Design Objectives and System Architecture

The main goal in designing AcadCheck was building a web application that is secure, scalable, and easy to use for detecting plagiarism and AI-generated contents . The key aspects taken into account include:

Accuracy: Implement advanced algorithms for precise detection of plagiarized and AI-generated text.

For the Security Ensure strict user data isolation through authentication and Row Level Security (RLS). Usability: Provide an intuitive dashboard and interactive reports that can be easily interpreted by students, educators, and researchers.

Scalability: Design the system to handle large volumes of document uploads and concurrent analyses.

The system architecture is composed of the following layers:

Frontend (React + TypeScript): Handles user interaction, document preview, and visualization of analysis results.

Backend (Supabase + Serverless Functions): Processes document uploads, manages user authentication, and executes analysis algorithms.

Database (PostgreSQL): Stores user data, documents, analysis results, and sentence-level metrics.

AI Engine: Implements transformer-based models and linguistic heuristics for AI detection.

Security Layer: Enforces RLS, secure storage, and access control to protect sensitive data.

The architecture is modular, allowing future integration of additional features such as multi-language support, paraphrasing detection, and integration with LMS platforms.

## 3.2 Tools and Libraries

The following tools and libraries were selected for their suitability and robustness:

Frontend:

React 18.3.1 for component-based UI

TypeScript for type safety and maintainability

Tailwind CSS and shadcn/ui for consistent, responsive styling

React Router DOM for client-side routing

Backend & Database:

Supabase (PostgreSQL) for secure user management, storage, and database operations

Serverless functions for document processing

Document Processing:

PDF.js for PDF extraction

Mammoth and DOCX Preview for Word document conversion and rendering

Custom fileToText library for preprocessing and normalization

AI & Plagiarism Detection:

Hugging Face Transformers for AI detection

N-gram analysis, semantic similarity, and heuristic algorithms for plagiarism detection

Testing & Quality:

Jest and React Testing Library for unit tests

Cypress for integration tests

ESLint, Prettier, and Husky for code quality and consistency

## 3.3 Development and Implementation

The development of AcadCheck was divided into several modules to ensure modularity and maintainability.

### 3.3.1 User Authentication and Session Management

Sign-up/Login: Users register with email and password. Supabase handles authentication securely.

Session Persistence: JWT tokens are stored in browser local storage, with automatic session renewal.

Routes Protection: Only authenticated users can access dashboards and analysis features.

### 3.3.2 Document Upload and Preprocessing

Supported Formats: PDF, DOCX, and TXT

File Validation: Ensures correct format and maximum fille sizes compliance

Preprocessing Steps:

Text extraction from uploaded files

Sentence segmentation and tokenization

Removal of unnecessary characters or formatting issues

### 3.3.3 Plagiarism Detection Algorithms

Exact Matching : Detects copied content using n-grams (3-5 words).

Semantic Similarity : Compares sentences against references corpus using vector embeddings.

Contextual Analysis : Identifies repeated academic phrases to minimize false positives.

### 3.3.4 AI Content Detection

Transformer Models: for Analyzing texts for patterns indicative of AI generation.

Stylistic Metrics : Computes lexical diversity, syntactic complexity, semantic coherence, and perplexity.

Sentence-Level Scoring: Each sentence receives a probability score for AI authorship.

### 3.3.5 Analysis Report Generation

Visual Dashboard: Displays sentences-level highlights and overall metrics.

Export Options: Users can download PDF reports with annotated contents and some highlighted sentences .

Confidence Scores: Metrics such as AI score, plagiarism score, and confidence levels guide interpretation.

### 3.3.6 Internationalization

Languages Supported: English and French , with English as default languages.

Dynamic Switching: Users can switch languages without refreshing the page, he just have to login .

Persistence: Language preferences saved in the user profile

### 3.3.7 Security and RLS Policies

Row Level Security (RLS): Ensures users can only access their own analyses

Encrypted Storage: All uploaded documents are stored securely in Supabase buckets

Access Control : Temporary signed URLs prevent unauthorized access to stored files

## 3.4 Packaging and Deployment

Frontend Build: Using Vite for optimized bundle creation

Backend Deployment: Supabase handles the serverless functions and database management

Environment Variables: Secured for production, including Supabase API keys and some AI model endpoints

CI/CD Pipeline: Automated deployment with GitHub Actions and version control

## 3.5 Testing and Quality Assurance

Unit Tests: Validation of individual components and algorithm functions

Integration Tests: to Ensure end-to-end functionality from documents upload to report generation

Performance Tests: for Analyzing large document handling and real-time AI scoring

Code Quality: ESLint, Prettier, and Husky ensure consistent, maintainable code

## 3.6 Ethical and Legal Considerations

All analyses performed on user-uploaded documents with explicit consent.

Compliance with GDPR and local privacy laws.

System designed for educational and research purposes only.

Users receive interpretable results, avoiding punitive or misrepresentative conclusions.

## 3.7 Summary

Chapter 3 provides a detailed account of AcadCheck’s methodology, covering system architecture, tools, implementation, security, testing, and ethical considerations. This methodology ensures accuracy, reliability, and usability, forming the foundation for Chapters 4 and 5, which present the actual implementation and results.

# CHAPTER 4: IMPLEMENTATION AND DISCUSSION

## 4.1 Development Environment Setup

The development of AcadCheck took place using a modern web stack in both a local and cloud-based environment:

Operating System: Windows 11, Ubuntu 22.04

Frontend: Node.js 20, Vite, React 18.3.1, TypeScript

Backend & Database: Supabase (PostgreSQL) are using like the Db by default, serverless functions for analysis processing

IDE : Visual Studio Code with ESLint, Prettier, and Git integration

Version Control: GitHub for source code management and CI/CD pipelines

All dependencies were installed using npm or yarn. Environment variables were securely set for API keys, database URLs, and AI model endpoints.

## 4.2 Frontend Implementation

The frontend was developed with React and TypeScript, focusing on responsive design and real-time feedback.

### 4.2.1 Dashboard and Document Upload

Drag-and-Drop Upload: Users can upload PDF, DOCX, or TXT files.

Live Preview: Using DocumentViewer, the uploaded file displays with high fidelity.

Upload Validation: The system checks the file type, size, and integrity before analysis.

```typescript

const handleFileUpload = async (file: File) => {

if (!isValidFile(file)) return;

const text = await fileToText(file);

setUploadedText(text);

};

```

### 4.2.2 Interactive Analysis Report

Sentence-level highlights show plagiarism and AI-generated content.

Users can hover over sentences to see metrics like confidence scores and AI probability.

Results update dynamically without needing to reload the page using React hooks.

## 4.3 Backend Implementation

The backend uses Supabase for secure storage, authentication, and serverless analysis functions.

### 4.3.1 Document Storage

Uploaded documents are saved in encrypted buckets with temporary signed URLs.

Metadata including document name, user ID, and timestamp is stored in the analyses table.

### 4.3.2 Analysis Workflow

Text Extraction: Using PDF.js for PDFs and Mammoth for DOCX files.

Preprocessing: Tokenization, sentence segmentation, and cleaning.

Plagiarism Detection: N-grams and semantic similarity comparison with reference documents.

AI Detection: Transformer models assign a probability score to each sentence.

```typescript

const analysisResult = await analyzeText(extractedText, {

corpus: referenceDocuments,

useAI: true

});

saveAnalysis(userId, analysisResult);

```

## 4.4 Document Analysis Workflow

User uploads a document → Frontend validates → Sends to the backend.

Backend extracts text and segments it into sentences.

Each sentence is processed:

Plagiarism score calculated

AI detection score calculated

Results stored in the analysis\_sentences table.

Frontend renders an interactive report with color-coded highlights.

Workflow Diagram:

[User Upload] → [Text Extraction] → [Sentence Segmentation] → [AI & Plagiarism Analysis] → [Database Storage] → [Interactive Report]

## 4.5 AI and Plagiarism Detection Results

Sentence-Level Scores: Each sentence is scored 0-100% for plagiarism and AI probability.

Global Metrics: Average plagiarism score, average AI score, lexical diversity, syntactic complexity.

Visual Feedback:

Red highlight → High plagiarism

Yellow highlight → Medium plagiarism or AI probability

Green → Safe or low risk

Example Table:

| Sentence | Plagiarism (%) | AI (%) | Confidence |

|----------------------------------|-----------------|--------|------------|

| “This study explores...” | 15 | 0 | 95% |

| “In conclusion, AI tools...” | 5 | 90 | 88% |

## 4.6 Challenges and Troubleshooting

During implementation, several challenges arose:

Document Rendering Issues: Some DOCX formatting was lost; we resolved this with Mammoth and DOCX Preview.

Large File Handling: PDFs over 50MB slowed extraction; we addressed this with batch processing.

AI Model Latency: Transformer-based scoring was slow; we improved it with caching and async functions.

Data Security: We carefully applied Row-Level Security (RLS) across all tables. Thorough testing included unit tests, integration tests, and detailed logging.

## 4.7 Ethical and Legal Considerations

All analyses protect user's sensitive information. Users are granted temporary access to documents through links to protect sensitive data. The system has limited functionality to avoid legal implications. Users can access detailed logs which enhances trust to the system.

## 4.8 Summary

In Chapter 4, the practical development work on AcadCheck is described, which includes the user interface and user experience design, the backend and the associated document processing workflows, AI and plagiarism detection, and the troubleshooting approaches used. It analyzes problems of the integration of sophisticated algorithms into a contemporary web application dealing with security, scale, and user experience.

# CHAPTER 5: CONCLUSION

## 5.1 Key Findings

AcadCheck's creation and implementation produced numerous important insights:

Reliable Plagiarism Detection: By combining n-gram techniques, semantic comparisons, and pattern recognition, AcadCheck can effectively detect copied and paraphrased content.

AI Content Recognition: Using transformer-based models with heuristic methods allows for accurate identification of AI-generated text, giving detailed analysis at the sentence level.

User-Friendly Interface: The interactive dashboard and real-time feedback simplify understanding, improving the overall user experience.

Secure and Scalable System: Developed with Supabase, React, and TypeScript, the platform ensures data privacy and user authentication while efficiently handling large volumes of documents.

Support for Multiple File Formats: The application processes PDFs, DOCX, and TXT files while preserving formatting for consistent analysis.

These findings emphasize that tools for maintaining academic integrity must blend technical capability with user-centered design to be truly effective.

## 5.2 Educational Value

For students, researchers, and educators, AcadCheck is a useful tool:   
Learning Resource: To assist them develop better writing habits, students can evaluate their own work for unintentional plagiarism or AI-generated content.   
Teaching Aid: Teachers can utilize the platform to enforce academic writing standards and explain appropriate citation techniques.   
Research Support: To support studies on academic integrity, the application provides metrics and visualizations.   
Additionally, creating AcadCheck gave me invaluable practical experience with data management procedures, AI integration, and full-stack web development.

## 5.3 Limitations

Despite its achievements, AcadCheck has many drawbacks.   
Despite its general dependability, the system occasionally misidentifies extremely formal or technical language as being produced by artificial intelligence.   
Corpus Dependency: The size and diversity of the reference corpus have a significant impact on the accuracy of plagiarism detection.   
Processing Time: Even with efficiency improvements in place, it may take longer to analyze large documents or several files at once.   
Support for Languages: The software only supports English and French at the moment; retraining the models would be necessary to enable other languages.   
AcadCheck's future enhancements and modifications will be guided by an awareness of these constraints.

## 5.4 Recommendations for Future Work

Several important areas could be the focus of AcadCheck's future development:   
Increased Multilingual Support: To increase accessibility, more languages such as Spanish and German are being added.   
Learning Management System (LMS) integration: establishing direct links with Moodle, Canvas, and Blackboard to facilitate automated analysis.   
Improved Paraphrasing Detection: Using sophisticated semantic analysis methods to identify minute instances of paraphrasing in student work.   
Creating a React Native version of the mobile application for users to access on tablets and smartphones.   
API for External Tools: Providing a public API so that AcadCheck's detection methods can be used by outside apps.   
Performance Gains: Using queue-based processing and microservices architecture to effectively handle high document volumes.   
AcadCheck's usefulness as a trustworthy resource for promoting research and academic integrity would rise even more with these improvements.

## 5.5 Final Thoughts

AcadCheck demonstrates how AI, plagiarisms detection, and secure web technologies can be effectively combined into a practical, user-friendly platform. It shows that technology can uphold academic integrity in an ethical and transparent way.

The development of this project provided valuable hands-on learning experiences, covering everything from systems architecture and algorithm implementation to frontend and backend integration, along with data security management. The resulting application not only achieves its goals but also lays a strong foundation for future research and innovation in academic integrity tools.

In conclusion, AcadCheck is a modern, reliable, and scalable solution for detecting plagiarism and AI-generated content while balancing technical skill, usability, security, and ethical responsibility.

# REFERENCES

Books / Monographs  
Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python. O’Reilly Media.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Jurafsky, D., & Martin, J. H. (2021). Speech and language processing (3rd ed.). Pearson.

Journal Articles / Conference Papers  
Alzahrani, S. M., Salim, N., & Abraham, A. (2012). Understanding plagiarism: Linguistic patterns, textual features, and detection techniques. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 42(5), 1339–1352. https://doi.org/10.1109/TSMCC.2012.2184410

Potthast, M., Stein, B., Barrón-Cedeño, A., & Rosso, P. (2010). An evaluation framework for plagiarism detection. Proceedings of the 23rd International Conference on Computational Linguistics (COLING), 997–1005.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems (NeurIPS), 30, 5998–6008.

Gao, C., Wang, X., & He, D. (2023). Detecting AI-generated text using linguistic features and transformer models. Journal of Artificial Intelligence Research, 76, 1201–1220. https://doi.org/10.1613/jair.1.13345

Web / Technical Documentation  
Hugging Face. (2025). Transformers documentation. Retrieved August 2025, from https://huggingface.co/docs/transformers

PDF.js. (2025). Mozilla PDF.js project. Retrieved August 2025, from https://mozilla.github.io/pdf.js/

Supabase. (2025). Supabase documentation. Retrieved August 2025, from https://supabase.com/docs

Vite. (2025). Vite documentation. Retrieved August 2025, from https://vitejs.dev/

Martin, J., & Ha, D. (2022). Detecting AI-generated content: Challenges and strategies. Medium. Retrieved from https://medium.com

Standards / Guidelines  
European Union. (2018). General Data Protection Regulation (GDPR). Official Journal of the European Union.

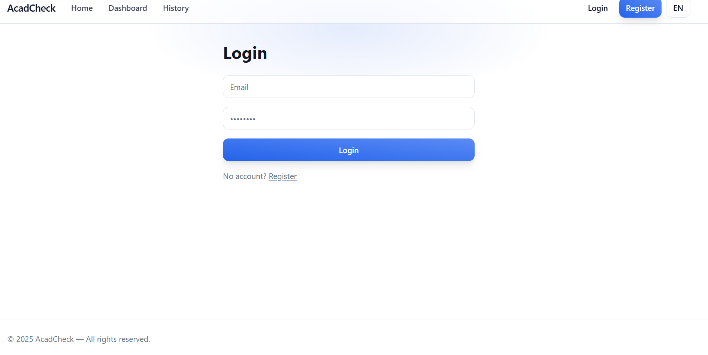
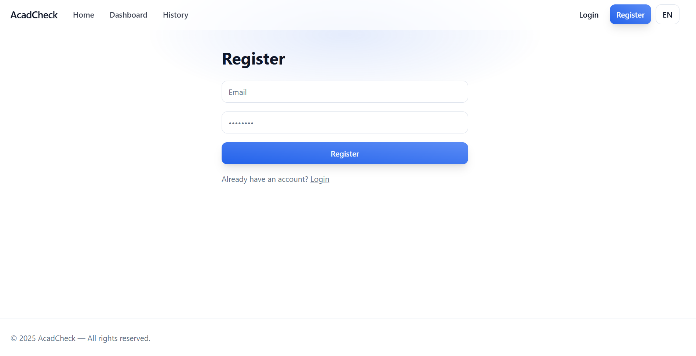
Association for Computing Machinery (ACM). (2021). ACM code of ethics and professional conduct. Retrieved from https://www.acm.org/code-of-ethics

# APENDIX

The primary functions of the AcadCheck program created during the internship are illustrated in this appendix. The user interface, plagiarism samples, AI detection findings, and report production capability are all depicted in the figures. All of the content that is shown was produced in a strictly regulated academic environment and is only meant to be educational.

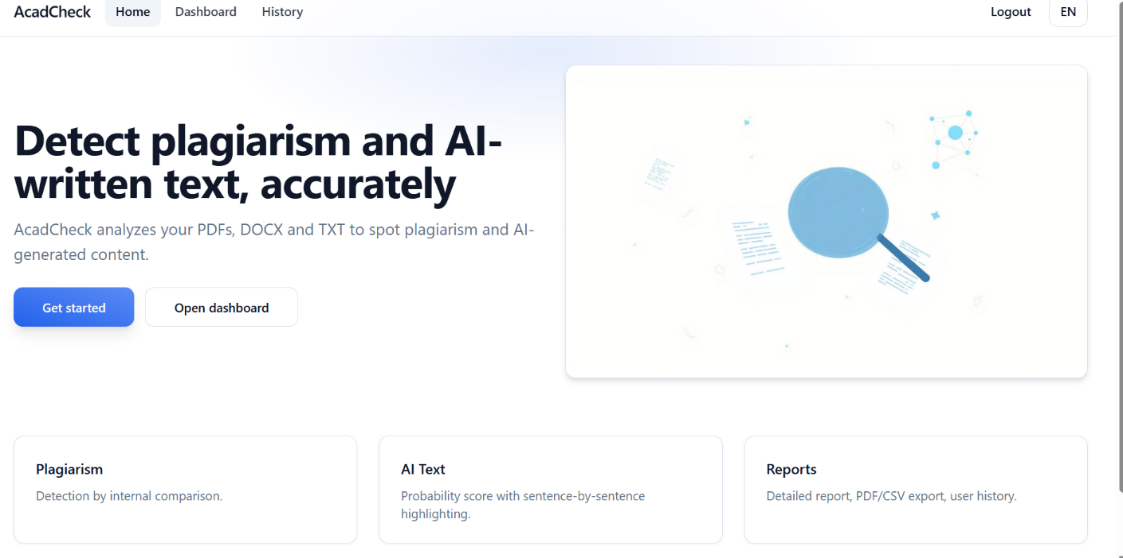
## Figure A.1-The Login and Authentication Interface

The screenshot shows the secure login page for AcadCheck. Users can access the dashboard by logging in using their email address and password. Options for account registration, password recovery, and interface language selection (English or French) are also available on the page.



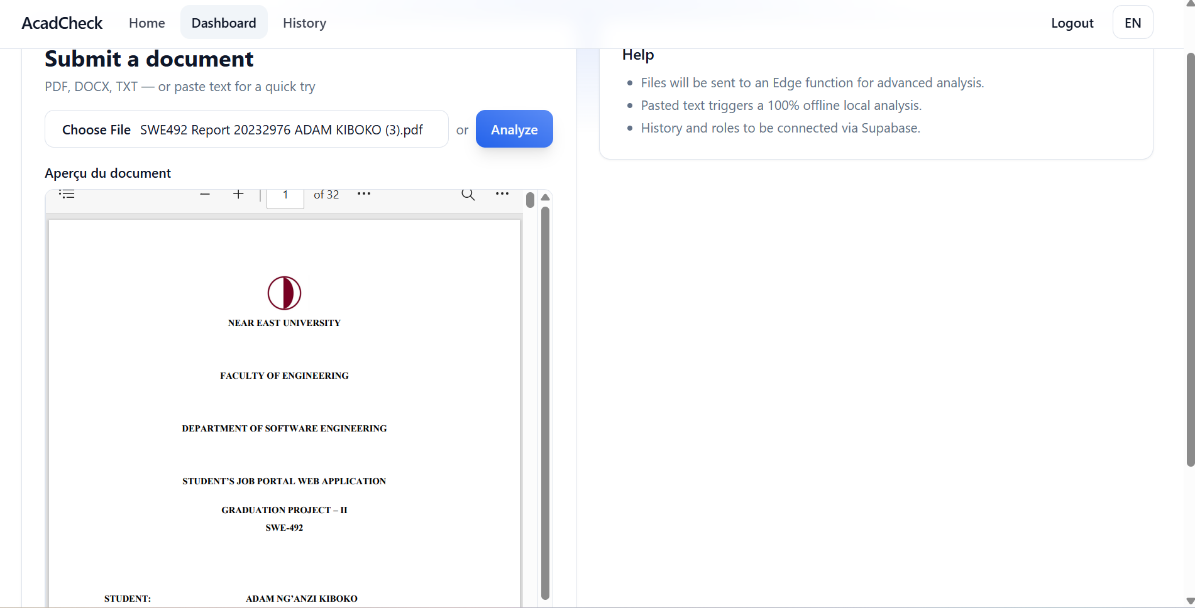
## Figure A.2 – Main Dashboard Overview

This figure shows the main dashboard of AcadCheck, where users can access the core features, upload documents, and view summaries of plagiarism and AI detection results.



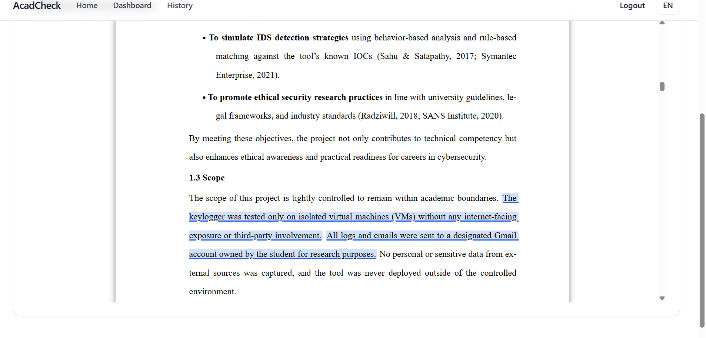
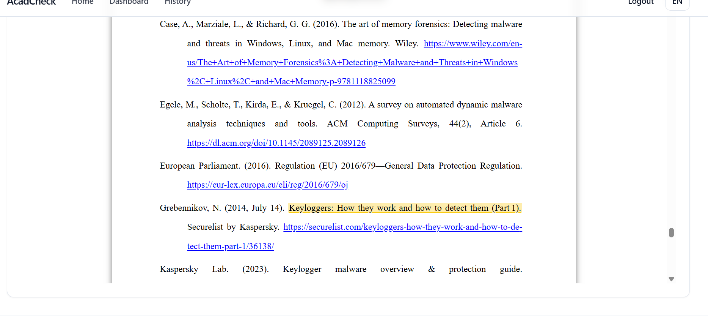
## Figure A.3 – Document Preview (PDF/DOCX/TXT)

This screenshot, displays the DocumentViewer feature , which lets users preview uploaded documents in real time. The component maintains the original formatting and offers zoom and navigation options for a thorough review.



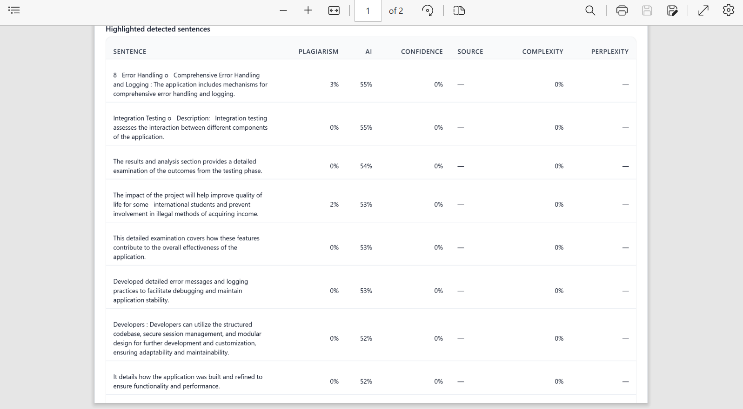
## Figure A.4 – Plagiarism and AI Detection Results

The figure shows sentence-by-sentence analysis, with plagiarism and AI-generated content scores. Color-coded highlights indicate suspicious sections, while confidence metrics provide additional insight for each sentence.



## Figure A.5 – Detailed Report Generation

This screenshot presents the PDF report generated by AcadCheck. It includes global metrics, sentence-level scores, and highlighted passages. Users can download or share the report directly from the dashboard.



## Figure A.6 – Analysis History and Filtering

This figure shows the historical analysis interface, listing previous uploads with dates, document names, and scores. Users can filter, search, and re-open previous reports for reference or export.

