

**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

[Acknowledgements 2](#_Toc207041927)

[Abstract 3](#_Toc207041928)

[Chapter 1: Introduction 6](#_Toc207041929)

[1.1 Background 6](#_Toc207041930)

[1.2 Objectives 7](#_Toc207041931)

[1.3 Scope 7](#_Toc207041932)

[1.4 Importance of the Study 8](#_Toc207041933)

[1.5 Structure of the Report 8](#_Toc207041934)

[CHAPTER 2: LITERATURE REVIEW 9](#_Toc207041935)

[2.1 Overview of Plagiarism Detection 9](#_Toc207041936)

[2.2 AI-Based Text Detection 10](#_Toc207041937)

[2.3 Document Analysis Techniques 10](#_Toc207041938)

[2.4 Ethical and Legal Considerations 11](#_Toc207041939)

[2.5 Use of React, TypeScript, and Supabase in Web Apps 11](#_Toc207041940)

[2.6 Integration of Detection Algorithms in Web Applications 12](#_Toc207041941)

[2.7 Summary 12](#_Toc207041942)

[CHAPTER 3: METHODOLOGY 13](#_Toc207041943)

[3.1 Design Objectives and System Architecture 13](#_Toc207041944)

[3.2 Tools and Libraries 13](#_Toc207041945)

[3.3 Development and Implementation 14](#_Toc207041946)

[3.3.1 User Authentication and Session Management 14](#_Toc207041947)

[3.3.2 Document Upload and Preprocessing 15](#_Toc207041948)

[3.3.3 Plagiarism Detection Algorithms 15](#_Toc207041949)

[3.3.4 AI Content Detection 15](#_Toc207041950)

[3.3.5 Analysis Report Generation 15](#_Toc207041951)

[3.3.6 Internationalization 15](#_Toc207041952)

[3.3.7 Security and RLS Policies 16](#_Toc207041953)

[3.4 Packaging and Deployment 16](#_Toc207041954)

[3.5 Testing and Quality Assurance 16](#_Toc207041955)

[3.6 Ethical and Legal Considerations 16](#_Toc207041956)

[3.7 Summary 17](#_Toc207041957)

[CHAPTER 4: IMPLEMENTATION AND DISCUSSION 18](#_Toc207041958)

[4.1 Development Environment Setup 18](#_Toc207041959)

[4.2 Frontend Implementation 18](#_Toc207041960)

[4.2.1 Dashboard and Document Upload 18](#_Toc207041961)

[4.2.2 Interactive Analysis Report 18](#_Toc207041962)

[4.3 Backend Implementation 19](#_Toc207041963)

[4.3.1 Document Storage 19](#_Toc207041964)

[4.3.2 Analysis Workflow 19](#_Toc207041965)

[4.4 Document Analysis Workflow 19](#_Toc207041966)

[4.5 AI and Plagiarism Detection Results 20](#_Toc207041967)

[4.6 Challenges and Troubleshooting 20](#_Toc207041968)

[4.7 Ethical and Legal Considerations 22](#_Toc207041969)

[4.8 Summary 22](#_Toc207041970)

[CHAPTER 5: CONCLUSION 23](#_Toc207041971)

[5.1 Key Findings 23](#_Toc207041972)

[5.2 Educational Value 23](#_Toc207041973)

[5.3 Limitations 24](#_Toc207041974)

[5.4 Recommendations for Future Work 24](#_Toc207041975)

[5.5 Final Thoughts 25](#_Toc207041976)

[REFERENCES 25](#_Toc207041977)

[APENDIX 27](#_Toc207041978)

[Figure A.1 – Login and Authentication Interface 27](#_Toc207041979)

[Figure A.2 – Main Dashboard Overview 27](#_Toc207041980)

[Figure A.3 – Document Preview (PDF/DOCX/TXT) 28](#_Toc207041981)

[Figure A.4 – Plagiarism and AI Detection Results 29](#_Toc207041982)

[Figure A.5 – Detailed Report Generation 30](#_Toc207041983)

[Figure A.6 – Analysis History and Filtering 31](#_Toc207041984)

|  |
| --- |
| 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, maintaining the integrity of written work is essential. The growing use of digital tools, online resources, and AI-assisted content creation has heightened the risk of plagiarism and the improper use of external materials. AcadCheck tackles these issues by offering an automated solution that can identify both plagiarized content and text generated by artificial intelligence, helping to uphold academic honesty and the quality of scholarly work.

Contemporary document analysis relies on advanced technologies such as machine learning, semantic evaluation, and natural language processing (NLP). By combining these methods with a straightforward and intuitive web interface, AcadCheck enables educators, students, and researchers to examine documents effectively while ensuring the security and privacy of their data.

## 1.2 Objectives

The primary goals of this internship project are:

To create a web application capable of identifying both plagiarism and AI-generated content.

To integrate sophisticated analysis techniques, combining machine learning with heuristic methods.

To provide comprehensive reporting, including analysis at the sentence level, confidence indicators, and clear visual cues.

To guarantee secure management of user data, leveraging authentication, Row-Level Security (RLS), and protected storage.

To deliver a user-friendly and responsive interface that accommodates multilingual users effectively.

## 1.3 Scope

This project centers on the design, development, and implementation of AcadCheck as a tool for research and education. The application is compatible with PDF, DOCX, and TXT file formats, and it offers detailed sentence-level analysis for both plagiarism and AI-generated content.

All testing and usage of the application are carried out ethically, within controlled environments, and in strict accordance with data protection regulations. The project does not involve deployment in untrusted environments or the collection of 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 emergence of AI-generated content, particularly large language models like GPT, distinguishing between human-written and AI-generated text has become a crucial challenge. AcadCheck implements hybrid AI detection techniques, combining transformer-based models with linguistic heuristics:

Transformer Models: Pre-trained models analyze text for patterns indicative of AI generation, such as overuse of certain connectors, predictable phrasing, or low variability.

Linguistic Heuristics: Metrics such as lexical diversity, syntactic complexity, and perplexity are used to evaluate the naturalness of text. AI-generated text often exhibits high perplexity uniformity and low lexical diversity.

Sentence-Level Scoring: Each sentence is assigned an AI score, indicating the likelihood of machine generation. This granularity allows users to review suspicious content precisely.

By integrating AI detection with plagiarism analysis, AcadCheck provides a comprehensive evaluation of document originality.

## 2.3 Document Analysis Techniques

Effective plagiarism and AI detection require robust document analysis. AcadCheck supports multiple formats, including PDF, DOCX, and TXT, with consistent text extraction:

PDF Analysis: Using PDF.js, the system extracts text while preserving document structure.

Word Document Handling: Mammoth and DOCX Preview convert DOCX files to HTML, ensuring accurate rendering and sentence-level analysis.

Text Preprocessing: Cleaning, tokenization, and normalization prepare the content for further analysis.

Sentence segmentation, feature extraction, and vectorization are applied before any similarity or AI detection algorithms, ensuring that results are both accurate and interpretable.

## 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 explicit consent are analyzed.

Data Security: Supabase provides encrypted storage and Row Level Security (RLS) to isolate user data.

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

Responsible Reporting: Results are presented solely for educational or institutional purposes, avoiding misuse of the analysis.

Ethical considerations also guide the inclusion of AI detection. The system is designed for transparency, providing users with interpretable metrics and avoiding punitive assumptions.

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

Modern web technologies play a key role in the functionality and usability of AcadCheck:

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

TypeScript: Adds type safety and improves maintainability, especially important for complex components such as analysis tables and reports.

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

Frontend UI Components: Using Tailwind CSS and shadcn/ui, the interface is modern, clean, and intuitive. Components like DocumentViewer and HighlightedText provide detailed visual feedback to users.

The combination of these technologies allows for a robust, scalable, and secure web application, supporting complex AI and plagiarism detection workflows while maintaining excellent user experience.

## 2.6 Integration of Detection Algorithms in Web Applications

A critical aspect of modern plagiarism detection systems is the seamless integration of backend analysis algorithms with frontend visualization:

Sentence-Level Analysis: Each sentence is analyzed for plagiarism and AI content. Results are transmitted from the backend to the frontend, where they are visually highlighted in the document.

Scoring and Metrics: Global scores, confidence levels, and detailed metrics are presented using interactive dashboards.

Real-Time Feedback: Users can upload documents and receive analysis results without page reloads, thanks to React’s component architecture.

This integration ensures that technical sophistication in analysis does not compromise usability.

## 2.7 Summary

Chapter 2 shows that modern plagiarism and AI detection relies on multi-layered methods that combine semantic analysis, machine learning, and secure web technologies. AcadCheck applies these principles by providing a system that is easy to use, secure, and accurate, effectively tackling today’s challenges in maintaining academic integrity.

# CHAPTER 3: METHODOLOGY

## 3.1 Design Objectives and System Architecture

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

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

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.

Route 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 file size 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 reference corpus using vector embeddings.

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

### 3.3.4 AI Content Detection

Transformer Models: Analyzes text 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 sentence-level highlights and overall metrics.

Export Options: Users can download PDF reports with annotated content.

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

### 3.3.6 Internationalization

Languages Supported: English and French

Dynamic Switching: Users can switch languages without refreshing the page

Persistence: Language preferences saved in 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 serverless functions and database management

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

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

## 3.5 Testing and Quality Assurance

Unit Tests: Validate individual components and algorithm functions

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

Performance Tests: Analyze 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 was performed using a modern web stack on 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), 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 configured for API keys, database URLs, and AI model endpoints.

## 4.2 Frontend Implementation

The frontend was implemented using React and TypeScript, emphasizing 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 is displayed with high fidelity.

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

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 indicate plagiarism and AI-generated content.

Users can hover over sentences to view metrics such as confidence scores and AI probability.

Results are dynamically updated without page reloads using React hooks.

## 4.3 Backend Implementation

The backend is built with Supabase, providing secure storage, authentication, and serverless analysis functions.

### 4.3.1 Document Storage

Uploaded documents are stored 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 and Mammoth for PDFs and DOCX respectively.

Preprocessing: Tokenization, sentence segmentation, and cleaning.

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

AI Detection: Transformer models assign a probability score per sentence.

const analysisResult = await analyzeText(extractedText, {

corpus: referenceDocuments,

useAI: true

});

saveAnalysis(userId, analysisResult);

## 4.4 Document Analysis Workflow

User uploads document → Frontend validates → Sends to backend.

Backend extracts text and segments into sentences.

Each sentence is processed:

Plagiarism score calculated

AI detection score calculated

Results are stored in analysis\_sentences table.

Frontend renders 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 were encountered:

Document Rendering Issues: Some DOCX formatting was lost; resolved using Mammoth and DOCX Preview.

Large File Handling: PDFs > 50MB caused slow extraction; solved by batch processing sentences.

AI Model Latency: Transformer-based scoring was time-consuming; optimized using caching and async functions.

Data Security: Careful attention was given to applying Row-Level Security (RLS) correctly across all tables. To ensure proper functionality, thorough testing was conducted, including unit tests, integration tests, and detailed logging.

## 4.7 Ethical and Legal Considerations

All analyses respect user privacy.

Temporary URLs prevent unauthorized document access.

System designed for demonstration and educational purposes, avoiding legal infringement.

Detailed logs and transparency ensure users can trust results.

## 4.8 Summary

Chapter 4 presents the hands-on implementation of AcadCheck, covering the frontend and backend development, document processing workflows, AI and plagiarism detection features, and strategies for troubleshooting. It also discusses the real-world challenges encountered when integrating complex algorithms into a modern web application, while ensuring security, scalability, and a smooth user experience.

# CHAPTER 5: CONCLUSION

## 5.1 Key Findings

The development and deployment of AcadCheck led to several key insights:

Accurate Plagiarism Detection: By combining n-gram techniques, semantic comparisons, and contextual pattern recognition, AcadCheck can reliably detect both copied and paraphrased content.

AI Content Recognition: The use of transformer-based models alongside heuristic methods allows for precise identification of AI-generated text, with detailed analysis at the sentence level.

Intuitive User Interface: The interactive dashboard and real-time feedback make the results easy to understand, enhancing the overall user experience.

Secure and Scalable System: Built with Supabase, React, and TypeScript, the platform ensures data privacy, user authentication, and the ability to handle large volumes of documents efficiently.

Support for Multiple File Formats: PDFs, DOCX, and TXT files are processed while maintaining formatting, ensuring accurate and consistent analysis.

These observations highlight that tools for maintaining academic integrity must combine technical sophistication with user-centered design to be truly effective.

## 5.2 Educational Value

AcadCheck functions as a practical tool for students, researchers, and educators alike:

Learning Resource: Students can check their own work for accidental plagiarism or AI-generated content, helping them improve their writing habits.

Teaching Aid: Educators can use the platform to illustrate proper citation methods and maintain academic writing standards.

Research Support: The application provides detailed metrics and visualizations that facilitate studies on academic integrity.

Additionally, the development of AcadCheck offered valuable hands-on experience in full-stack web development, AI integration, and implementing secure data management practices.

## 5.3 Limitations

Despite its achievements, AcadCheck has some limitations:

AI Detection Accuracy: Although generally reliable, the system may occasionally flag highly formal or technical writing as AI-generated.

Corpus Dependency: The accuracy of plagiarism detection depends heavily on the size and variety of the reference corpus.

Processing Time: Analyzing large documents or multiple files at once can take longer, even with performance optimizations in place.

Language Support: Currently, the application only handles English and French; supporting additional languages would require retraining the models.

Being aware of these limitations helps guide improvements and refinements for future versions of AcadCheck.

## 5.4 Recommendations for Future Work

Future development of AcadCheck could focus on several key areas:

Expanded Multilingual Support: Incorporating additional languages such as Spanish and German to make the tool accessible to a wider audience.

Integration with Learning Management Systems (LMS): Enabling direct connections to platforms like Moodle, Canvas, and Blackboard for seamless automated analysis.

Enhanced Paraphrasing Detection: Implementing more advanced semantic analysis techniques to identify subtle paraphrasing in student submissions.

Mobile Application: Creating a React Native version to allow users to access AcadCheck on smartphones and tablets.

API for External Tools: Providing a public API so third-party applications can utilize AcadCheck’s detection algorithms.

Performance Improvements: Adopting microservices architecture and queue-based processing to efficiently handle large volumes of documents.

These improvements would further enhance AcadCheck’s value as a reliable and professional tool for academic integrity and research support.

## 5.5 Final Thoughts

AcadCheck showcases how advanced AI, plagiarism detection, and secure web technologies can be combined into a practical, user-friendly platform. It demonstrates that technology can support academic integrity in a way that is both ethical and transparent.

The development of this project offered valuable hands-on learning experiences, covering everything from system architecture and algorithm implementation to frontend and backend integration, as well as data security management. The resulting application not only meets its intended goals but also provides a solid foundation for further research and innovation in academic integrity tools.

In conclusion, AcadCheck stands as a modern, dependable, and scalable solution for detecting plagiarism and AI-generated content, striking a balance between technical sophistication, 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

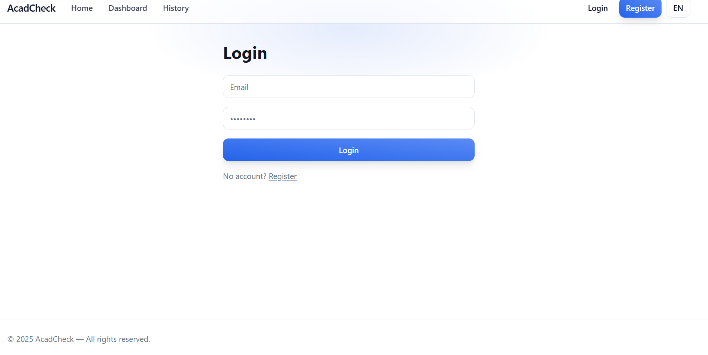
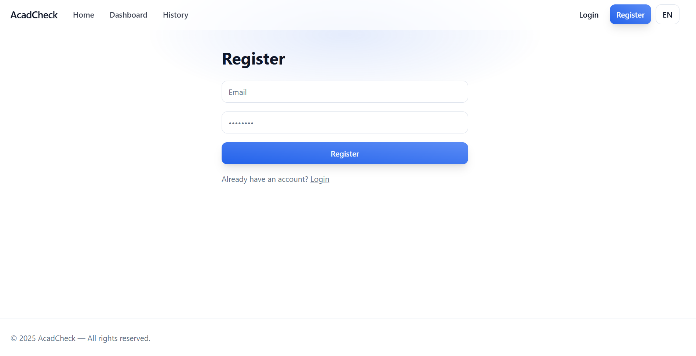
Standards / Guidelines  
European Union. (2018). General Data Protection Regulation (GDPR). Official Journal of the European Union.

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

# APENDIX

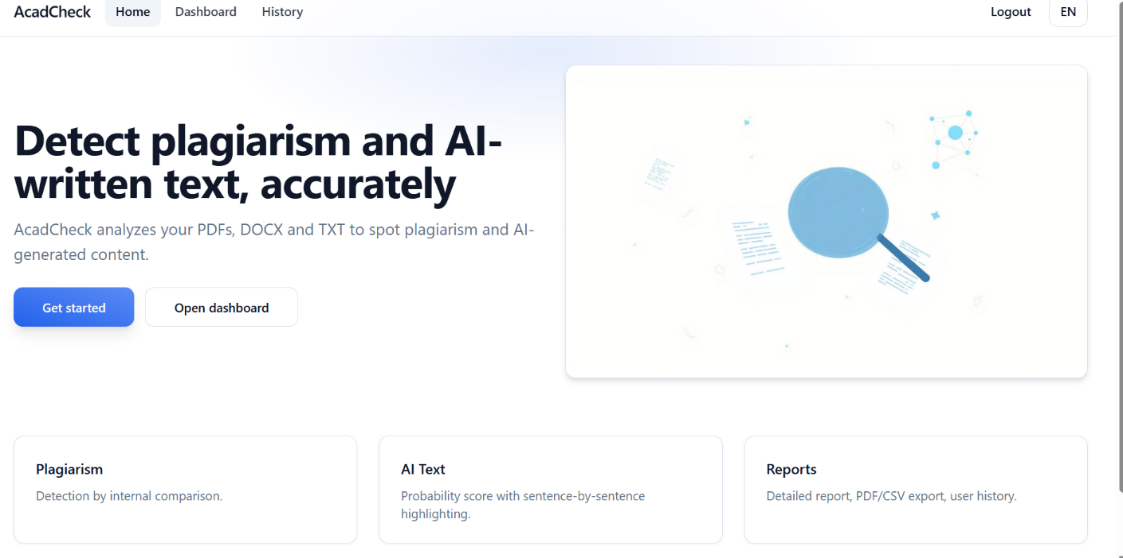
This appendix presents visual documentation of the main features of the AcadCheck application developed during the internship. The figures illustrate the user interface, examples of plagiarism and AI detection results, and the report generation functionality. All displayed content was created within a controlled academic setting and is intended solely for educational purposes.

## Figure A.1 – Login and Authentication Interface

The screenshot displays AcadCheck’s secure login page. Users can sign in using their email and password to access the dashboard. The page also provides options for account registration, password recovery, and selecting the interface language (English or French).

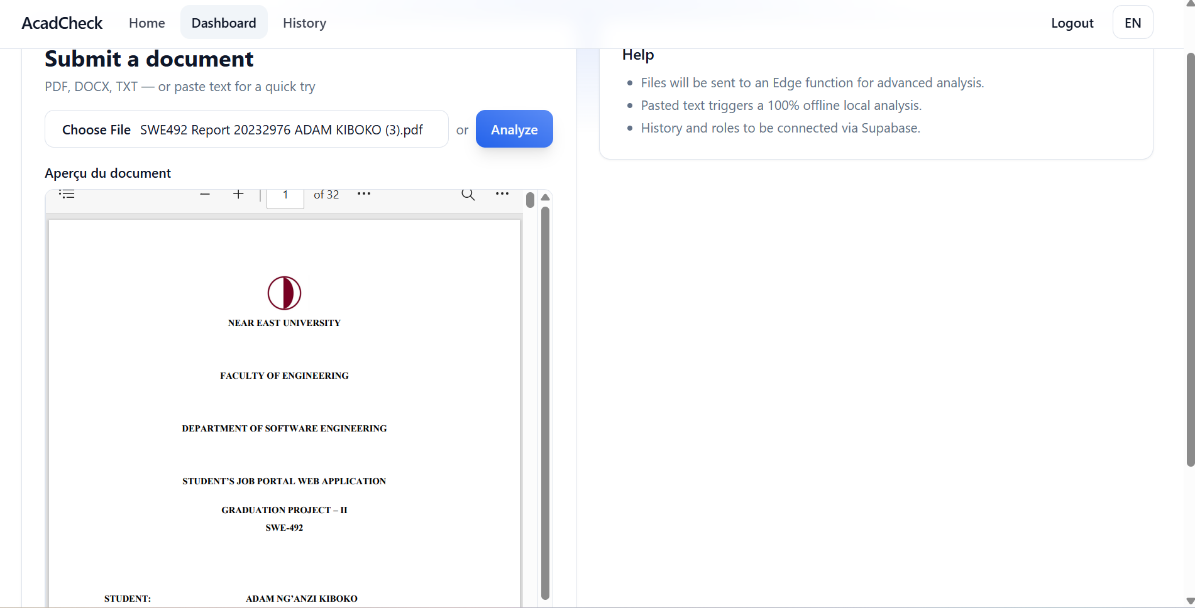
## Figure A.2 – Main Dashboard Overview

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.



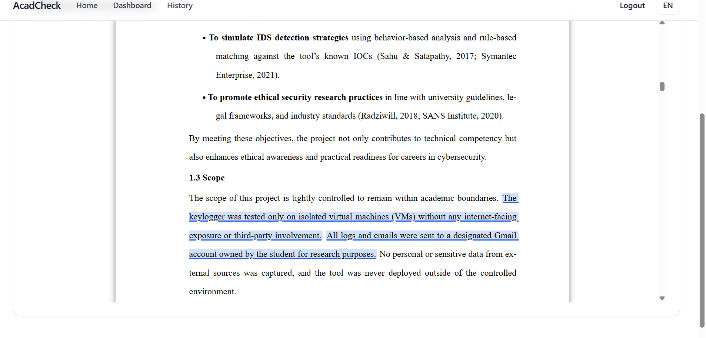
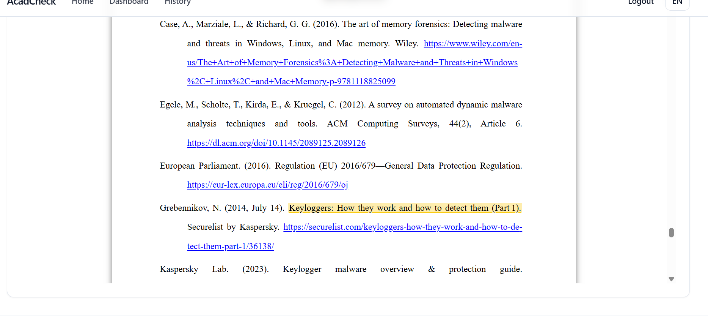
## Figure A.3 – Document Preview (PDF/DOCX/TXT)

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.



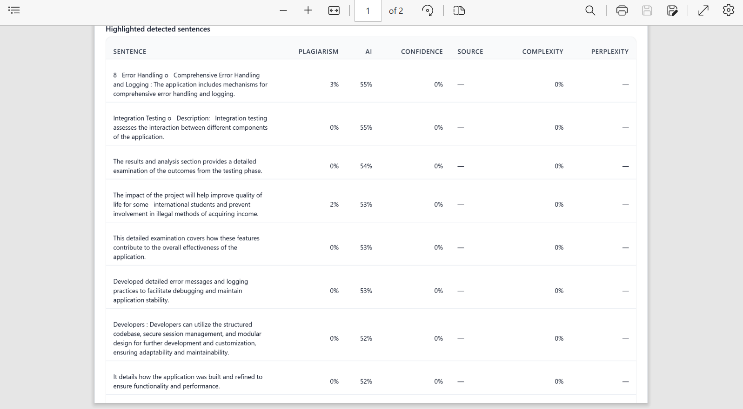
## Figure A.4 – Plagiarism and AI Detection Results

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.

