Project Report - SpamGuard

Overview

SpamGuard is a web-based application designed to classify email content as spam or not spam using advanced AI algorithms. The application leverages Google's Generative AI to analyze email content and provide a structured JSON response indicating the likelihood of the content being spam.

Key Components

1. Backend (server.js)

- **Environment Configuration**: Utilizes `dotenv` to manage environment variables, including the API key for Google Generative AI.
- **Express Server**: Sets up an Express server with CORS and JSON parsing middleware.
- **AI Integration**: Integrates with Google Generative AI using the `@google/generative-ai` package to classify email content.
- **API Endpoint**: Provides a POST endpoint `/api/classify` that accepts email content and returns a JSON response with spam classification details.

2. Frontend (HTML Files)

- **index.html**: The main page where users can input email content to be analyzed. It includes a form that submits the content to the backend for classification.
- **about.html**: Provides information about SpamGuard, including its features and statistics.
- **contact.html**: Contains a contact form for users to reach out with questions or feedback.

3. Email Content (test_email.txt)

- Contains sample email content used for testing the spam classification functionality.

Features

- **Real-time Spam Detection**: Users can input email content and receive immediate feedback on whether it is classified as spam.
- **Detailed Analysis**: The application provides a confidence score, reasons for classification, and a risk level.
- **User Interface**: A clean and responsive UI built with HTML and CSS, including Tailwind CSS for styling.

Technical Details

- **Backend**: Node.js with Express.js for server-side logic.
- **Frontend**: HTML, CSS, and JavaScript for client-side interaction.
- **AI Model**: Google Generative AI for content analysis.
- **Environment Variables**: Managed using a `.env` file for sensitive information like API keys.

Potential Improvements

- **Error Handling**: Enhance error handling in the backend to provide more detailed feedback to users.
- **UI Enhancements**: Improve the user interface for better accessibility and user experience.
- **Scalability**: Optimize the application for handling a larger volume of requests.

Conclusion

SpamGuard is a robust application that effectively utilizes AI to classify email content. With further enhancements, it can become a comprehensive solution for spam detection in various contexts.