\documentclass[12pt]{article} \usepackage{graphicx} \usepackage{hyperref} \usepackage{geometry} \geometry{margin=1in}

\title{AI Web Lab: TensorFlow.js and OpenAI API Integration} \author{Ralph Colmenar} \date{\today}

\begin{document}

\maketitle

\section{Introduction} The purpose of this lab is to design and implement a basic AI-powered web application that demonstrates both client-side and server-side artificial intelligence technologies. The project integrates TensorFlow.js for image classification in the browser and the OpenAI API through a Node.js backend to demonstrate AI-based text generation. This lab focuses on understanding how modern AI tools can be incorporated into web applications while following best practices for security and deployment.

\section{Objective} The objective of this lab is to: \begin{itemize} \item Build a simple web interface using HTML, CSS, and JavaScript \item Perform image classification using TensorFlow.js in the browser \item Connect a Node.js backend to the OpenAI API \item Securely manage API keys using environment variables \item Deploy and document the project using GitHub \end{itemize}

\section{Tools and Technologies} The following tools and technologies were used to complete this lab: \begin{itemize} \item HTML and CSS for structuring and styling the web interface \item JavaScript for client-side and server-side logic \item TensorFlow.js with the MobileNet model for image classification \item Node.js for server-side execution \item OpenAI API for AI-generated text responses \item dotenv for managing environment variables \item Git and GitHub for version control and project hosting \end{itemize}

\section{System Implementation} The project is divided into two main components: a client-side web application and a server-side Node.js script.

The client-side application consists of an HTML page that allows users to upload an image. TensorFlow.js is loaded through a CDN, and a pre-trained MobileNet model is used to classify the uploaded image directly in the browser. The prediction result is displayed dynamically without requiring a server request.

The server-side component is implemented using Node.js. A script initializes the OpenAI client and sends a text prompt to the OpenAI API. The API key is stored locally in a \texttt{.env} file and loaded using the dotenv package. For security purposes, the \texttt{.env} file is excluded from version control using a \texttt{.gitignore} file.

\section{Results} The TensorFlow.js image classification component successfully loads the MobileNet model and classifies images selected by the user. The Node.js script successfully connects to the OpenAI API and sends a request. Although the API response returned a quota limitation error, this confirms that the API integration is correctly implemented and that requests are being executed as expected.

\section{Security Considerations} To ensure proper security practices, the OpenAI API key was stored locally using environment variables. The \texttt{.env} file was excluded from the GitHub repository using a

\texttt{.gitignore} file to prevent accidental exposure of sensitive credentials. This approach follows industry-standard security practices for managing API keys.

\section{GitHub Repository} The complete project source code and documentation are available at the following GitHub repository:

\begin{center} \url{https://github.com/ralph-ie/yourinitial-ai-web-lab} \end{center}

\section{Screenshots} Figures below demonstrate the successful execution of the application, including the TensorFlow.js image classification interface and the Node.js terminal output showing a successful OpenAI API request.

\begin{figure}[h] \centering \includegraphics[width=0.9\linewidth]{screenshot1.png} \caption{TensorFlow.js image classification interface} \end{figure}

\begin{figure}[h] \centering \includegraphics[width=0.9\linewidth]{screenshot2.png} \caption{Node.js execution showing OpenAI API response} \end{figure}

\section{Conclusion} This lab successfully demonstrates how artificial intelligence can be integrated into modern web applications using both client-side and server-side approaches. TensorFlow.js enables real-time image classification directly in the browser, while the OpenAI API provides powerful AI-driven text generation capabilities. The project highlights the importance of secure API key management and version control when working with external services. Overall, this lab provides a practical foundation for building AI-enabled web applications.

\end{document}