University of Florida

DocIQ Documentation Report

github.com/domesimbana/CEN3031-Project

Group 113: DocIQ Team

Alexander Aziz, Domenica Simbana Mosquera, Kevin Wagner, Tue Tran

CEN3031

Dr. Neha Rani

4/24/2024

Table of Contents:

Section 1:

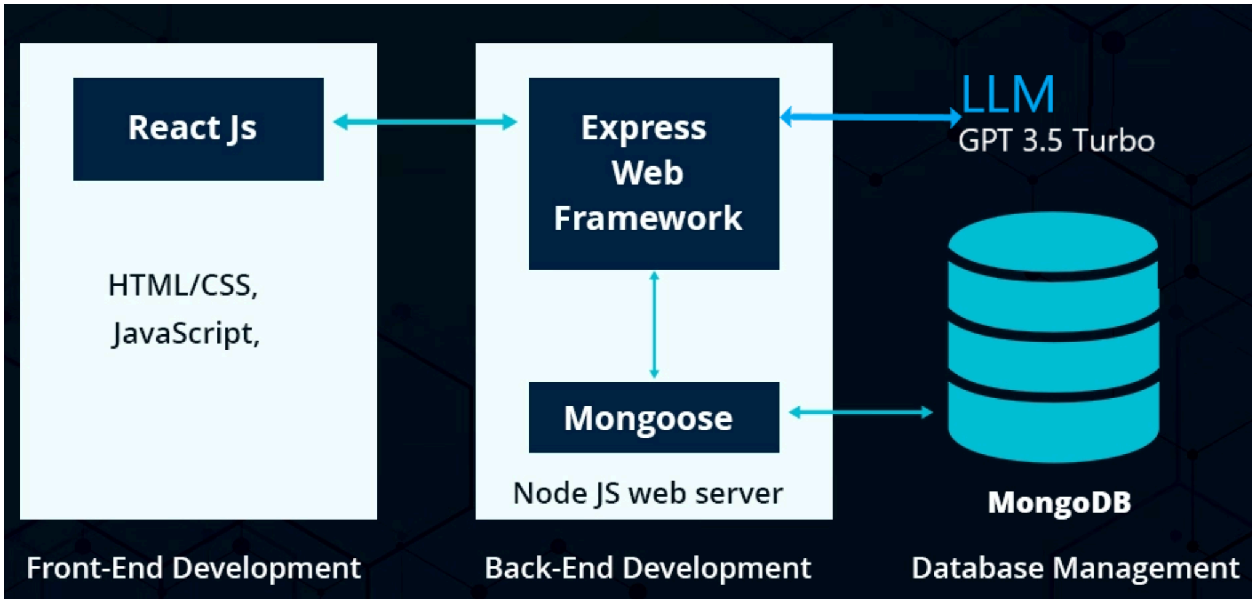
- **Project Description:** DocIQ aimed to address the challenge of leveraging generative AI for social good by providing users with a platform to gain a deeper understanding of any text. Our solution utilizes generative AI to help users grasp a deeper understanding of any text by providing answers to questions or explaining topics found in the text. The documents provided are scanned to store the information found within the text, allowing the AI bot to gain prior knowledge before answering the user's questions. By allowing users to securely upload PDF documents our web app solution is easily accessible and maintains user privacy. The documents submitted to the website are privately stored within the user's account where they can revisit and delete submitted documents.
- **How the solution addressed the challenged statement**: The solution addresses the challenged statement by combining technologies and frameworks to create a comprehensive platform for users to find accurate responses to provided questions. Here is how the solution responds to the disputed statement:
    - <u>NoSQL Database (MongoDB):</u>
        - MongoDB is used to securely store user information, ensuring scalability and adaptability for growing user populations.
        - Storing PDF files in MongoDB enables the quick retrieval and management of pdf documents provided by users.
    - <u>MERN Stack:</u>
        - The MERN stack, consisting of MongoDB, Express.js, React, and Node.js.
        - Express.js enables the development of robust backend APIs for handling user authentication, document upload, and data retrieval.
        - Node.js acts as the server-side runtime environment, allowing for seamless communication between frontend and backend components.

        - React is utilized for building dynamic and interactive user interfaces, providing a smooth user experience.
    - External LLM Component: OpenAI API
        - Our group used GPT 3.5 Turbo LLM to manage data retrieval and submission logic for interacting with the LLM.
    - Open-source PDF Parse:
        - Using Python for PDF parsing allows for efficient extraction of text and data from PDF documents, which can then be stored and handled in the MongoDB database.
    - Authorization and Authentication Component: Google Cloud PlatForm
        - Manages user authentication and authorization securely.
- **Features and Functionality:**
    - Google Authenticator:
        - Users can securely authenticate their accounts using Google Authenticator, adding an extra layer of security to the platform by connecting to the Google Cloud Platform.
    - Upload Document
        - Users can easily upload documents, including and PDF files, to the platform.
        - The upload feature allows users to easily contribute helpful material content to the platform's knowledge base.
    - Delete Uploaded Document:
        - Users can delete previously uploaded documents from their account.
        - Deleting uploaded documents provides users control over their content while also ensuring privacy and confidentiality.
    - Create a new Upload:

        - Users can start a fresh document upload procedure, allowing them to continue contributing to the platform's collection of educational content.
    - Ask and answer the AI by the chat box:
        - The AI-powered chat box provides a dynamic interface for users to interact with the platform's generative AI chatbot.
        - Users can ask relative questions to the given file and obtain accurate and useful answers based on the platform's knowledge base of AI, which includes data retrieved from uploaded documents.

- **System Model:**
    - Architectural pattern: DocIQ uses the Model-View-Controller (MVC) architectural pattern to structure its components efficiently. MongoDB serves as the database model, storing user information and data, assuring data integrity, and allowing for efficient data manipulation. React.js powers the frontend user interface, offering users a dynamic and interactive experience, while Express.js and Node.js handle backend logic coordinating data flow between model and view components. This architectural style encourages modularity, scalability, and maintainability, facilitating the development of a robust and flexible educational platform like DocIQ.
    - System Context: The system context model defines the relationship between DocIQ and its external entities, demonstrating how the platform interacts with both users and external systems. DocIQ is accessed via web browsers by users looking for educational knowledge and assistance on a variety of topics. External systems such as Google Authenticator improve account security. MongoDB Database is the principal store for user data, which includes uploaded documents and AI chatbot responses.
    - Use Case Model: The use case model describes the functionalities and interactions that users can do on the DocIQ platform. These use cases comprise the DocIQ platform's

fundamental functionalities, allowing users to effortlessly engage with the system and easily access information.



Section 2:

- **Code management:**
    - Our team utilized Git for code management, utilizing the GitHub platform for version control. We followed a branching strategy where each feature or bug fix was developed on a separate branch before being merged into the main branch through pull requests. This approach ensured that changes were well-documented, reviewed, and tested before integration into the main codebase.
- **Project management:**
    - For project management we used agile methods like Scrum. We used a 3 sprint plan in which we identified product backlog items for each week and assigned them to person and also estimated the amount of time it would take to complete each product backlog

item. We used google sheets to track who was working on what and identify which items need more resources allocated to it. Additionally we conducted Scrum twice a week for the entire semester. Scrum helped us ensure everyone was aligned in the end goal and identify any obstacles that we needed to overcome.. Discord is a communication channel that helps team members collaborate and communicate effectively.

- **Test plan:**
    - Our test plan does not include anything as formal as Unit Tests, Integration Tests, or a CI/CD pipeline. However we did conduct manual testing to ensure that the end user receives a functional product. We documented our manual tests in a previous assignment in which we identified preconditions, test steps, test data, the expected result, and post conditions. These manual tests ensure that we have a minimum functionality level for our product.

Section 3:

- **Technical Details:** DocIQ Requires the following to be installed:

    1. Node.js
    2. MongoDB

- **Installation Instructions:**
  To run DocIQ locally please follow these steps:

    1. Clone the repository
        a. git clone https://github.com/domesimbana/CEN3031-Project.git
    2. Navigate to Project Directory
        a. cd CEN3031-Project
    3. Install dependencies
        a. cd frontend
        b. cd client
        c. npm install
        d. Cd ..
        e. Cd server
        f. Npm install

4. Run Backend and Frontend
    a. Cd client
    b. Npm start
    c. Cd ..
    d. Cd server
    e. Npm start

- **Login and Access Credentials and API Keys:**
    1. OpenAI API Secret Key
        a. Place in OpenAI object in frontend/client/src/pages/Document
    2. MongoDB project cluster invite is required

Section 4:

- **Risk Management Plan:**

    - <u>Accuracy of LLM AI:</u>

        - Use intensive testing and validation techniques to improve the correctness of the AI model. Regularly update and fine-tune the model in response to user comments and performance indicators.

        - Provide clear cautions about the AI's correctness and suggest that users analyze the information critically.

    - <u>Scalability:</u>

        - Build the system with scalability, leveraging credibility infrastructure and distributed computing capabilities to handle a high rate of document uploads

    - <u>User Credibility:</u>

        - Implement user authentication by Google Authenticator to increase trust and credibility. Implement methods to ensure the authenticity and integrity of uploaded documents.

    - <u>Data Privacy and Security:</u>

        - To ensure user data privacy and security, use access controls, and secure authentication systems.

        - Conduct frequent security audits and penetration tests to identify and resolve concerns.

- **Software Quality Attributes:**
    - Accuracy of LLM AI:
        - Ensure that AI-generated solutions are correct and relevant to customer inquiries by constantly refining the AI model and checking findings against truth data.
    - Scalability:
        - Create a system that can scale effortlessly to support a growing user population and more document uploads while maintaining performance and user experience.
    - User Credibility:
        - Establish trust and credibility with users' information through using a credibility framework, delivering accurate and reliable information, and allowing for user input and validation.
    - Data Privacy and Security:
        - Prioritize user data protection by implementing strong security measures to ensure user privacy and confidentiality are respected throughout the platform.

References:

1. Node.js. "Download Node.js." Available online: https://nodejs.org/en/download/

2. OpenAI. "API Reference." Available online: https://platform.openai.com/docs/api-reference

3. MongoDB. "Getting Started Tutorial." Available online: https://www.mongodb.com/docs/manual/tutorial/getting-started/

4. YouTube. "Introduction to MongoDB." Available online: https://www.youtube.com/watch?v=-42K44A1oMA

5. PyPDF2. "Documentation." Available online: [https://pypdf2.readthedocs.io/en/3.x/