Project Documentation Report

Introduction

This project aims to create a web application that logs and displays information about incoming requests to a server, with a particular focus on capturing and displaying the IP details of the requesting clients. The application is built using Node.js and Express, and leverages external APIs such as ipinfo.io to fetch IP-related information. The project consists of several key components, including a shell script, a main JavaScript file, a dashboard HTML file, and a honeypot HTML file.

Implementation Details

Shell Script (`shart.sh`)

The `shart.sh` shell script appears to be a helper script for running the Node.js application. It checks for command-line arguments and runs the corresponding npm scripts, such as starting the development server or running tests.

Main JavaScript File (`main.js`)

The `main.js` file sets up an Express server to listen for incoming requests. It includes routes for serving the dashboard and honeypot HTML files, as well as an endpoint for fetching logs. Logs are stored in-memory and can be accessed through the `/logs` endpoint. When a request hits the `/honeypot` endpoint, the script fetches the client's IP information from the ipinfo.io API and logs it.

Dashboard HTML (`dashboard.html`)

The `dashboard.html` file contains the dashboard UI, which fetches logs from the server and displays them in a formatted manner. It dynamically creates HTML elements to display each log entry's IP information. The dashboard also includes a function to change the background image dynamically using the Unsplash API.

Honeypot HTML (`honeypot.html`)

The `honeypot.html` file displays a simple card-based UI indicating a secure connection, IP logging, and a privacy notice. It also fetches a random background image from the Unsplash API.

Key Features

The key features of this project include:

1. **IP Logging**: The application logs the IP details of incoming requests, particularly through the `/honeypot` endpoint, using the ipinfo.io API.

2. **Dashboard**: The dashboard UI provides a formatted display of the logged IP information, allowing users to view and analyze the collected data.

3. **Honeypot**: The honeypot interface presents a card-based UI that may be used to lure or attract potential attackers, while still logging their IP details.

4. **Dynamic Background**: Both the dashboard and honeypot HTML files feature the ability to dynamically change the background image using the Unsplash API, enhancing the visual appeal of the application.

The logging of IP addresses raises important security and privacy considerations, as this data can be used to identify and potentially track individuals. It is crucial to handle user data responsibly and ensure compliance with relevant data protection regulations.

Challenges Faced

During the development of this project, the team may have faced several challenges, such as:

1. **Integrating External APIs**: Incorporating the ipinfo.io API to fetch IP-related information and the Unsplash API to fetch background images may have required careful handling of asynchronous requests and error handling.

2. **UI Design**: Designing the dashboard and honeypot interfaces to be visually appealing, user-friendly, and responsive may have presented design-related challenges.

3. **Data Visualization**: Effectively presenting the logged IP information in a clear and meaningful way on the dashboard may have required research into appropriate data visualization techniques.

Future Improvements

To enhance the project further, the following improvements could be considered:

1. **User Authentication**: Implementing user authentication and access control mechanisms to secure the dashboard and limit access to the logged data.

2. **Enhanced Data Visualization**: Exploring more advanced data visualization techniques, such as charts, graphs, or geographic visualizations, to provide deeper insights into the logged IP data.

3. **Performance Optimization**: Analyzing the application's performance and implementing optimizations, such as caching or asynchronous processing, to ensure efficient handling of incoming requests and data storage.

4. **Expanded Logging Capabilities**: Considering the addition of more detailed logging features, such as recording timestamps, request headers, or other relevant metadata, to provide a more comprehensive understanding of the incoming traffic.

5. **Compliance and Privacy**: Reviewing the application's handling of user data to ensure compliance with relevant data protection regulations and best practices for privacy preservation.

Conclusion

This project demonstrates the development of a web application that logs and displays information about incoming requests, with a focus on capturing and presenting IP details. The application leverages Node.js, Express, and external APIs to provide a dashboard and honeypot interface for users to interact with the collected data.

Through the implementation of this project, the team has gained valuable experience in integrating external APIs, designing responsive and visually appealing user interfaces, and handling sensitive user data in a responsible manner. The proposed future improvements suggest ways to enhance the project's functionality, security, and compliance, further strengthening its capabilities and impact.