# **Asynchronous Programming Assignment Documentation**

## **Overview**

This project demonstrates the use of asynchronous programming techniques in JavaScript, including Callbacks, Promises, and Async/Await. The implementation involves fetching data from an API and displaying it on a webpage with enhanced styling and functionality.

## **Project Structure**

The project consists of three separate HTML files with corresponding CSS and JavaScript logic embedded in the <script> tag:

* callbacks.html
* promises.html
* async-await.html
* styles.css (common styles applied to all pages)

## **Features Implemented**

1. **Callbacks**
   * Created callbacks.html with a button and an output <div>.
   * Implemented a function that uses a callback to delay execution by 5 seconds.
   * Fetched data from https://dummyjson.com/posts and displayed post titles.
   * Used innerHTML to dynamically generate multiple <div> elements for each post, making it easier to apply alternating shading for readability.
   * Added a scrollbar to the output container to manage overflow content.
2. **Promises**
   * Created promises.html with a similar structure.
   * Implemented a Promise-based approach to fetch data asynchronously.
   * Displayed a "Loading..." message while the Promise was pending.
   * Implemented error handling: if the API request took more than 5 seconds, the Promise was rejected with an "Operation timed out" message.
3. **Async/Await**
   * Created async-await.html to demonstrate the async/await approach.
   * Used await to fetch data and handle responses more cleanly.
   * Implemented robust error handling for network failures or API issues.

## **Enhancements and Modifications**

* **Sticky Heading:** The heading inside the output container remains fixed at the top when scrolling.
* **Alternate Row Shading:** Every other post is shaded differently to improve readability.
* **Additional Data Display:** Along with post titles, the number of likes and shares for each post is retrieved and displayed.
* **Common Styling:** A styles.css file is used for better UI consistency.
* **Code Optimization:** Avoided redundant document.getElementById calls by assigning elements to variables.

## **GitHub Repository**

The complete project is available on GitHub: [GitHub Repository Link](https://github.com/Rawat107/Asynchronous-Programming-Exercises.git)

## **How to Run**

1. Clone the repository from GitHub.
2. Open callbacks.html, promises.html, or async-await.html in a browser.
3. Click the button to fetch and display API data.

## **Conclusion**

This project effectively demonstrates various asynchronous programming techniques with enhancements for better user experience and code maintainability. The implementation ensures optimized API calls, structured UI elements, and error handling to provide a robust learning experience.