**Internship Project Report**

*At*

|  |
| --- |
|  |
| **"** **IBRA Digital Branding Services Pvt. Ltd "** |

*Submitted By*

**Satvik Jayalwal**

# Table of Contents

1. Executive Summary

2. Introduction

3. Project Objectives

4. Project Scope

5. Methodology

6. Conclusion

# Acknowledgement

I am deeply appreciative of all the support and guidance I received during my internship at IBRA Digital Branding Services Pvt. Ltd. The successful completion of my project was greatly facilitated by the collective efforts of many individuals, and I am grateful for each contribution.

I would like to extend my sincere gratitude to Mr. Asad Rabbani, CEO of IBRA Digital Branding Services Pvt. Ltd. Your decision to offer me this internship was a significant opportunity for my professional growth. Your encouragement and belief in my potential provided me with the motivation to excel and contributed greatly to my success.

A special thanks goes to Mr. Rahul Sharma, our project mentor, whose dedication and insightful guidance were instrumental throughout my internship. Your regular meetings and readiness to resolve any questions I had ensured that I remained on the right track. The depth of your feedback and the clarity of your advice were invaluable in enhancing my learning experience and improving the quality of my project.

I also wish to express my gratitude to Mr. Rishabh Singh and all the other team members who provided their expertise and support. Your willingness to assist me with various challenges and your constructive input played a crucial role in my ability to overcome obstacles and achieve the project’s objectives. Your professional insights were essential to the successful completion of my tasks.

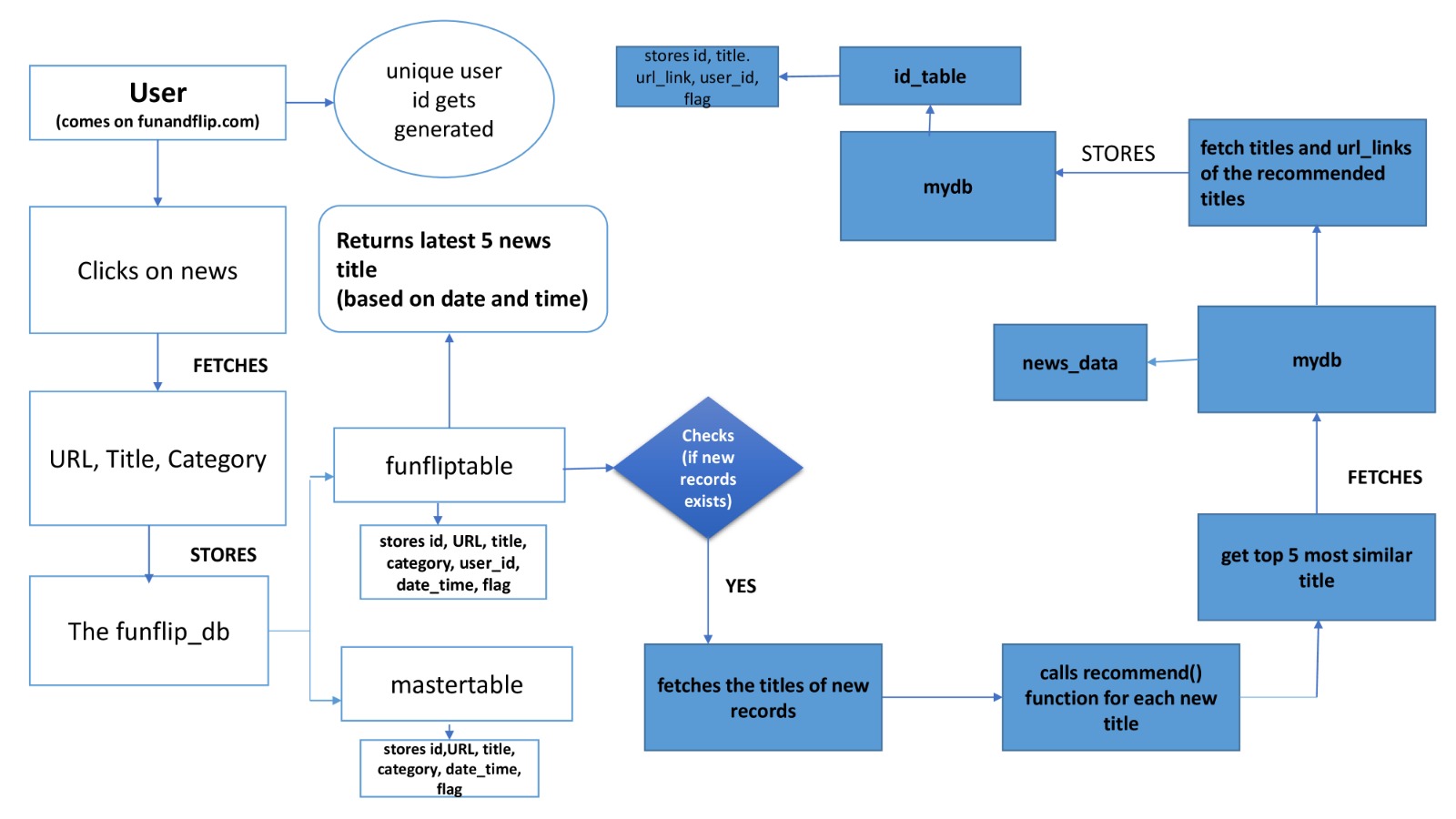
Furthermore, I am thankful to my friends who stood by me throughout this journey. Your encouragement, teamwork, and positive energy made the experience not only productive but also enjoyable. The collaborative environment and the camaraderie we shared were significant in making this internship a rewarding and memorable experience.

In summary, I am profoundly grateful to everyone who contributed to this internship experience. Each person’s support and guidance were key to my growth and the successful outcome of my project. Your collective efforts have made a lasting impact on my professional development, and I sincerely appreciate all that you have done.

Thank you for your unwavering support and for being an essential part of this journey. Your contributions have been invaluable, and I am grateful for the opportunity to have worked with such talented and supportive individuals.

SIGNATURE :

# PROJECT ARCHITECTURE

****

# 1. Executive Summary

This is a Web scraping task which efficiently fetches, and stores metadata from specified URLs. It is developed using HTML, JavaScript, PHP, and MySQL, offers several functionalities:

**Metadata Fetching:** The application retrieves metadata such as URL, title, and category using JavaScript and a proxy server to bypass CORS restrictions. It parses Open Graph (OG) metadata crucial for content aggregation, SEO, and web analytics.

**Unique User ID Generation:** To track user interactions, the application generates unique user IDs by combining the user's IP address, current datetime, and a predefined salt value, hashed using MD5. The ID is stored in a cookie for continuity across sessions.

**Cookie Management:** JavaScript functions handle cookies to store the unique user ID, ensuring consistent tracking and personalized user experiences.

**Backend Data Storage and Retrieval:** PHP scripts manage data storage and retrieval in thefunandflip\_db, a MySQL database through XAMPP phpmyadmin. Metadata and user information are stored securely with a database and the data is further accessed by an API which gets only top five latest data elements . Then they are shared to my teammate further .

# 2. Introduction

### Background Information

This project involves developing web scrapping designed to fetch metadata from a specified URL, and store it securely in a MySQL database through XAMPP phpmyadmin. It performs several tasks like generating unique user IDs, managing cookies, and storing/updating data. The application is built using a combination of HTML, JavaScript, and PHP, with the backend focusing on data storage and retrieval within a MySQL database through XAMPP phpmyadmin.

### Fetching Metadata from a URL

The core functionality of this is to fetch metadata from a given URL. Metadata typically includes information such as the webpage's URL, title, and category, which are crucial for understanding and organizing web content and to apply Ai reccommendations further . The process begins when a URL is specified, the application then uses JavaScript to send a request to a proxy server. This proxy server bypasses cross-origin resource sharing (CORS) restrictions, enabling the application to retrieve the necessary metadata. The fetched data is then parsed, extracting key details like the Open Graph (OG) metadata, which often includes the webpage's title and URL. This extracted information is then stored in the database using MySQL.

### Generating Unique User IDs

It generates unique user IDs. This feature is essential for tracking user interactions and personalizing user experiences. The user ID generation process combines the user's IP address, the current date time, and a predefined salt value. By hashing these elements together using the MD5 algorithm, the application creates a unique identifier for each user. This identifier is then stored in a cookie on the user's device, ensuring that the same user ID can be retrieved in future sessions, providing continuity and allowing the application to maintain user-specific data.

### Managing Cookies

Cookie management is another functionality of this application they are small data files stored on the user's device allowing web applications to remember information between sessions. In this project, cookies are used to store the unique user ID. The JavaScript functions within the application handle setting, retrieving, and managing these cookies. When a user visits the web application for the first time, a cookie containing their unique user ID is set.

### Asynchronous Data Updates and Display

The application leverages the power of asynchronous operations to enhance the user experience. Asynchronous JavaScript and XML (AJAX) techniques allow the application to send and receive data from the server without requiring a page refresh. This means that metadata can be fetched, processed, and displayed dynamically as the user interacts with the application. The use of local storage ensures that fetched metadata is saved on the user's device, reducing the need for repeated data retrievals and improving performance. The user interface is updated in real-time, providing a seamless and responsive experience.

### Backend Data Storage and Retrieval

On the backend, PHP scripts handle data storage and retrieval operations. When metadata is fetched, it is sent to the server, where PHP scripts insert it into the MySQL database through XAMPP phpmyadmin. The database schema includes tables for storing metadata and user information, ensuring that all data is organized and easily accessible. The backend also includes functionalities for retrieving data, such as fetching the most recent entries from the database using an API. This architecture ensures that the application is scalable and can handle large volumes of data efficiently.

In summary, this project exemplifies the integration of front-end and back-end technologies to create a web application. By combining HTML, JavaScript, and PHP, the application efficiently fetches, processes, and stores metadata, manages user identification through unique user IDs, and utilizes asynchronous operations to enhance user experience. The use of a MySQL database ensures reliable data storage and retrieval, making this application a comprehensive solution for metadata management and user interaction tracking.

### Importance of the Project

Metadata is crucial for understanding and organizing web content. This project addresses the need for efficient metadata retrieval and management, which is essential for various applications, including content aggregation, SEO optimization, and web analytics. By automating the process of fetching and storing metadata, the application enhances the ability to manage large volumes of data effectively.

### Purpose and Scope of the Report

This report details the development and functionality of the Meta Info Fetcher Web Application. It covers the core features, including metadata fetching, unique user ID generation, cookie management, asynchronous data updates, and backend data storage and retrieval. The purpose of this report is to provide a comprehensive understanding of the project's workflow and its integration of front-end and back-end technologies.

# 3. Project Objectives

**1. Fetch Metadata from a URL:**

The process begins by taking a given URL and making an HTTP request to retrieve the HTML content of the page. This is accomplished through a proxy server, to circumvent cross-origin resource sharing (CORS) restrictions, which typically prevent direct browser requests to external sites. Upon receiving the HTML content, it parses it to extract Open Graph (OG) metadata. Open Graph tags are standard metadata elements used by many websites to describe their content in a structured manner. The application looks for <meta> tags with properties like og:url and og:title to retrieve the webpage's URL and title. If these tags are absent, the application uses fallback methods to determine these values. Once extracted, this metadata is displayed on the web application interface, providing users with clear and immediate access to the key details of the specified webpage.

**2. User Identification:**

To personalize the data and ensure accurate tracking, the project generates a unique user ID for each user. This ID is created using a combination of the user’s IP address, the current date and time, and a predefined salt value. The IP address represents the user's network location, while the current date and time ensures the uniqueness of the ID at any given moment. The salt value, a static string known only to the server, adds an extra layer of uniqueness and security to the ID generation process. This combination is hashed using the MD5 cryptographic function to produce a unique identifier that is difficult to guess or replicate. This ID is then stored in a browser cookie.

**3. Data Storage:**

The project emphasizes data management by storing fetched metadata and user information in a MySQL database through XAMPP phpmyadmin. This involves creating two tables: mastertable and funfliptable. The mastertable stores the basic metadata about each webpage, including the URL, title, and category. The funfliptable using the foreign key concept extends this by including additional fields such as the user ID and timestamps. When new metadata is fetched, it is inserted into these tables via PHP scripts that handle data sanitization and validation. This not only secures the data against SQL injection attacks but also ensures that all entries are consistent and accurate.

**4. Asynchronous Operations:**

Users can continue interacting with the application while these operations are being processed. This is achieved using JavaScript's fetch API, which allows for asynchronous HTTP requests. When a request is made to fetch metadata or send data to the server, the application does not wait for the response to proceed, instead, it continues to execute other scripts. Once the response is received, callback functions handle the data appropriately, updating the UI or storing information as needed. This non-blocking behavior is crucial for maintaining a smooth and responsive user experience.

**5. User Experience:**

When metadata is fetched for the first time, it is stored in the browser's local storage. If the user revisits the page, the application first checks local storage for existing data before making a new HTTP request. This reduces the need for redundant network calls, speeding up the display of information and reducing server load. Local storage allows data to persist even after the browser is closed, ensuring that the fetched metadata is available immediately upon the user’s return. This caching mechanism not only improves performance but also provides a seamless experience where users can access previously viewed metadata without delay. By prioritizing speed and efficiency, the application ensures that users can quickly access the information they need, enhancing overall satisfaction and engagement.

# 4. Project Scope Assumptions

1. **Stable Internet Connection**: The application assumes that users have a stable internet connection for accessing and interacting with the web application.
2. **Consistent Proxy Server Performance**: The proxy server used to bypass CORS restrictions is assumed to be consistently available and reliable.
3. **Standard Browser Support**: The application assumes that users will access it using standard, up-to-date web browsers that support modern web technologies.
4. **Sufficient Server Resources**: The server hosting the application and the MySQL database is assumed to have sufficient resources to handle the expected load.

### Constraints

 **Technical Limitations:**

* The application is limited by the capabilities of the technologies used, including HTML, JavaScript, PHP, and MySQL. Certain advanced functionalities may not be feasible within these constraints.
* Browser-specific limitations, such as handling cookies, local storage capacity, and CORS policies, may affect the application's functionality and user experience.

 **Resource Availability:**

* The project’s scope is constrained by the availability of development resources, including time, budget, and personnel.
* Limited access to specialized tools or software that may enhance development, testing, or deployment processes.

 **Data Privacy Regulations:**

* The application must comply with data privacy regulations, such as GDPR, when handling user data. This may limit certain functionalities or require additional measures to ensure compliance.
* Ensuring user consent for data collection and cookie usage, which may necessitate implementing consent management tools.

 **Scalability and Performance:**

* The application must be designed to handle a growing number of users and large volumes of data without significant performance degradation.
* Limitations in server capacity and database performance may affect the application’s scalability and response times.

 **Maintenance and Updates:**

* Regular maintenance and updates are required to keep the application secure and functional. This may involve additional resource allocation for ongoing support.
* Dependency on third-party libraries or APIs, which may introduce compatibility issues or require updates.

 **User Experience:**

* Ensuring a consistent and responsive user experience across various devices and screen sizes may present challenges.
* Balancing the need for real-time updates with performance optimization to prevent lag or delays in the user interface.

 **Integration with Existing Systems:**

* Compatibility with existing systems and databases, if integration is required, may pose challenges.
* Ensuring smooth data flow and communication between different components of the application.

 **Testing and Quality Assurance:**

* Comprehensive testing is required to identify and resolve bugs, performance issues, and security vulnerabilities.
* Constraints on testing environments and scenarios may impact the thoroughness and effectiveness of the testing process.

 **Security Measures:**

* Implementing robust security measures to protect against threats such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
* Balancing security requirements with usability and performance considerations.

 **User Education and Support:**

* Providing adequate documentation and support to ensure users can effectively utilize the application.
* Addressing potential user misconceptions or difficulties in understanding how to use certain features.

# 5. Methodology

The development of the Meta Info Fetcher Web Application follows a structured methodology encompassing research design, data collection methods, and analysis techniques to ensure a robust and efficient implementation.

#### Research Design

The research design for this project is based on a systematic approach combining both qualitative and quantitative methods. The primary goal is to create a web application capable of fetching, processing, and storing metadata efficiently. The research design includes:

1. **Requirement Analysis:**
   * Identifying the functional and non-functional requirements of the application.
   * Understanding user needs and expectations through surveys and feedback sessions.
2. **System Design:**
   * Architectural planning to define the application’s structure, including client-server interactions, database schema, and data flow.
   * Selecting appropriate technologies (HTML, JavaScript, PHP, MySQL) to meet the project requirements.

#### Data Collection Methods

Data collection involves both primary and secondary methods to gather relevant information for the application development:

1. **Primary Data Collection:**
   * Direct input from potential users through surveys and interviews to understand their requirements and preferences.
   * Usability testing sessions to gather feedback on the application’s interface and functionality.
2. **Secondary Data Collection:**
   * Reviewing existing literature and case studies on similar applications to identify best practices and potential pitfalls.
   * Analyzing technical documentation and online resources to understand the capabilities and limitations of the chosen technologies.

#### Analysis Techniques

The analysis phase involves processing the collected data to inform the development process:

1. **Qualitative Analysis:**
   * Analyzing user feedback to identify common issues and improvement areas.
   * Categorizing and prioritizing user requirements to ensure critical functionalities are addressed.
2. **Quantitative Analysis:**
   * Statistical analysis of survey data to identify trends and patterns in user preferences.
   * Performance benchmarking to evaluate the application’s efficiency in fetching, processing, and storing metadata.
3. **Technical Analysis:**
   * Conducting feasibility studies to assess the practicality of implementing specific features within the given technical constraints.
   * Prototyping and iterative testing to refine the application’s design and functionality based on continuous feedback.

By integrating these research design elements, data collection methods, and analysis techniques, the development process of the Meta Info Fetcher Web Application is guided by comprehensive and data-driven insights, ensuring a well-rounded and user-centric final product.

# Conclusion

This project exemplifies a well-rounded approach to developing a modern web application that integrates both client-side and server-side technologies to provide a robust solution for metadata management and user tracking. By leveraging a combination of HTML, JavaScript, PHP, and MySQL, the application achieves several key objectives, including efficient metadata fetching, user identification, and data management.

**Client-Side Technologies**

On the client side, the HTML and JavaScript components work together to deliver a dynamic and interactive user experience. The HTML structure is minimalistic yet effective, featuring a <div> element containing three <p> elements dedicated to displaying the website URL, title, and category. This design ensures that users can quickly view the essential metadata of the target web page.

JavaScript plays a crucial role in managing the application's functionality. The getCookie and setCookie functions handle the creation and retrieval of cookies, which are essential for storing the unique user ID. This user ID is generated by the generateUniqueUserID function, which combines the user’s IP address, current date and time, and a salt value, then hashes the result using CryptoJS. This unique ID allows the application to track and identify users consistently across sessions.

The fetchMetaInfo function retrieves metadata from a specified URL, using the AllOrigins API to circumvent Cross-Origin Resource Sharing (CORS) restrictions. This function extracts Open Graph (OG) metadata, including the URL, title, and category, which is then displayed on the web page through the updateUI function. By utilizing local storage, the application stores this metadata, ensuring that it persists across page reloads. Additionally, the sendDataToServer function asynchronously sends the metadata to the server for further processing, without requiring a page refresh.

**Server-Side Technologies**

On the server side, PHP scripts establish a connection to the MySQL database, ensuring that the application can reliably store and manage data. The scripts for database and table creation dynamically set up the necessary database structure, creating tables such as mastertable and funfliptable if they do not already exist. This dynamic approach ensures that the backend infrastructure remains adaptable to changes in the data structure.

User management is handled effectively through PHP functions that generate and store user IDs, using cookies to maintain user sessions. The insertUserDetails function ensures that each user’s activity is logged in the database, providing valuable insights into user interactions.

Data insertion and retrieval are managed through PHP scripts that handle incoming POST requests to insert data into the mastertable and funfliptable. These scripts sanitize input data to prevent security vulnerabilities and ensure data integrity. For data retrieval, a PHP script processes API GET requests to fetch and return the top 5 records from the funfliptable, providing users with the most recent metadata entries.

**Overall Assessment**

The combination of client-side and server-side technologies in this project demonstrates a comprehensive approach for a functional and scalable web scrapping. The use of asynchronous operations and local storage ensures a smooth and responsive user experience, while the backend system efficiently manages data storage and retrieval. The detailed structure and functionality outlined in this report not only showcase the application's current capabilities but also provide a solid foundation for future enhancements and scalability. This project serves as a robust starting point for developing more advanced features, improving performance, and expanding the application’s capabilities to meet evolving user needs.