MYMI BUSINESS PLAN:

MARKETING CONTENT GENERATOR

AUTOMATED NEWS SCRAPING AND CONTENT GENERATION SYSTEM: DETAILED OVERVIEW

1. DATA COLLECTION (SCRAPING NEWS)

To consistently gather relevant news data, the system uses a combination of **APIs** and **manual web scraping**. APIs provide structured, reliable data, while manual scraping allows for more flexible data collection, especially in niche areas that APIs may not cover.

APIS VS. WEB SCRAPING:

- **APIs**: Structured data that reduces the risk of scraping bans. The following APIs are integrated into the system:
    - **NewsAPI**: 475 requests/day (95% of 500/day free tier).
    - **GDELT API**: No strict request limit, ideal for global event tracking.
    - **MediaStack**: 15-16 requests/day (95% of 500/month free tier).
    - **Currents API**: 31-32 requests/day (95% of 1000/month free tier).
    - **ContextualWeb API**: 31-32 requests/day (95% of 1000/month free tier), integrated for marketing, technology, and finance news.

  Daily API Rotation Strategy: Each API is used daily in a balanced manner to maximize free-tier usage without surpassing limits. Here's a breakdown:

    - **NewsAPI**: 475 requests/day, totaling 14,250 requests/month.
    - **GDELT API**: No strict limit.
    - **MediaStack**: 15-16 requests/day, totaling 475 requests/month.
    - **Currents API**: 31-32 requests/day, totaling 950 requests/month.
    - **ContextualWeb**: 31-32 requests/day, totaling 950 requests/month.
- **Manual Web Scraping**: The system allows users to manually input URLs to scrape niche content that isn't available via APIs.
    - **Functionality**: The system uses the **GuzzleHttp Client** for scraping and **HTML DOM parsing** via `HtmlDomParser`.
    - **Fallback Logic**: If the primary scraping attempt fails, it falls back to **Google Search scraping** using `scrapeGoogleSearch()`. This ensures that relevant content is retrieved even if the initial source is unavailable.

ERROR HANDLING & LOGGING:

- **Retry Logic**: The system automatically retries failed API requests with exponential backoff to handle temporary errors.
- **Logging**: Any API failures, errors, and usage data are logged into a separate table (e.g., **`bf_api_failures`**) for tracking and debugging.

DATA STORAGE:

All scraped data is stored in the `bf_marketing_temp_scraper` table, which includes the following fields:

- **Title**: The title of the article or content.
- **Source URL**: The original URL of the content.
- **Raw Content**: The full body of the scraped content.
- **Keywords/Tags**: Extracted keywords or tags related to the content.
- **Date Scraped**: The date when the data was scraped.
- **Error Logs**: Any errors or failures encountered during the scraping process.

This storage strategy ensures that all relevant content, whether gathered via API or manual scraping, is centralized for further processing and analysis. The system's capabilities for retrying failed requests, logging errors, and storing detailed content allow for a robust and reliable data collection process.

2. DATA PROCESSING (SUMMARIZATION AND KEYWORD EXTRACTION)

The **Marketing Management Module** includes robust text processing capabilities, utilizing various NLP (Natural Language Processing) models for both **summarization** and **keyword extraction**. Here's a breakdown of the key functionality already implemented, along with relevant methods from the code you provided.

SUMMARIZATION:

The module implements two primary techniques for summarizing scraped content, enabling the generation of concise summaries while retaining essential details from larger bodies of text.

1. **TextRank Summarization**:
   - **Method**: The method `summarizeContent()` in the **MyMIMarketing** library processes the scraped content into summaries using the **PageRank** algorithm combined with a **similarity matrix**.
   - **Functionality**:
     - **Sentence Splitting**: The method first splits the content into individual sentences using `splitIntoSentences()`.
     - **Similarity Matrix**: It then builds a similarity matrix using `buildSimilarityMatrix()`, where each sentence is compared to others based on word frequencies, calculated by `calculateWordFrequencies()`.
     - **PageRank Algorithm**: Once the similarity matrix is built, the **PageRank** algorithm (`pageRank()`) is applied to assign importance scores to each sentence.
     - **Summary Extraction**: The highest-ranked sentences are selected and concatenated to form a summary, ensuring the most critical content is preserved.
   - **Example Code Flow**:
     - `summarizeContent()` -> `splitIntoSentences()` -> `buildSimilarityMatrix()` -> `pageRank()` -> `buildSummaryToCharLimit()`.
2. **TfIdfTransformer**:

- **Method**: The **TfIdfTransformer** (Term Frequency-Inverse Document Frequency) model is used to summarize content by focusing on the importance of terms in a given context. This model gives more weight to terms that are frequent in a specific document but rare across multiple documents.
- **Functionality**:
    - **Term Weighting**: The module uses `WhitespaceTokenizer` to tokenize the content into individual words. These words are then analyzed using **TfIdfTransformer** to generate term importance scores.
    - **Sentence Scoring**: Each sentence in the content is scored based on the cumulative importance of the terms it contains.
    - **Summary Generation**: Sentences with the highest importance scores are selected to form the summary.

KEYWORD EXTRACTION:

The **MyMIMarketing** library implements **NLP-based keyword extraction**, using the **TfIdfTransformer** to extract key phrases and terms from the content.

1. **NLP-Based Keyword Extraction**:
    - **Method**: The method `extractKeywords()` in **MyMIMarketing** uses **TfIdfTransformer** and **WhitespaceTokenizer** from the **PhpML** library to extract important keywords.
    - **Functionality**:
        - **Tokenization**: The content is first tokenized using the `WhitespaceTokenizer` to split the text into individual words or tokens.
        - **Term Frequency Analysis**: The tokens are passed to **TfIdfTransformer**, which analyzes the frequency of each term within the content, giving more importance to terms that are unique or significant.
        - **Keyword Selection**: The highest-ranked keywords are selected and returned, representing the most relevant terms related to the content.
    - **Storage**: Extracted keywords are stored in the **`bf_marketing_temp_scraper`** table as indexed fields. This ensures that keywords can be easily searched and retrieved for fast content analysis and retrieval.

INTEGRATION WITH EXISTING STORAGE:

- The **bf_marketing_temp_scraper** table not only stores the raw scraped content but also stores the **summarized content** and **extracted keywords** as indexed fields. This makes it easy to search, query, and analyze data across multiple records based on keywords or summaries.
    - **Key Fields**:
        - `title`
        - `content` (summarized content)
        - `keywords` (extracted keywords)
        - `date_scraped`

CODE SNIPPET OVERVIEW:

- **Summarization Example**:

  php

```
Copy code
public function summarizeContent($content) {
    $sentences = $this->splitIntoSentences($content);
    $similarityMatrix = $this->buildSimilarityMatrix($sentences);
    $scores = $this->pageRank($similarityMatrix);
    return $this->extractTopSentences($sentences, $scores);
}
```

- **Keyword Extraction Example**:

```
php
Copy code
public function extractKeywords($text) {
    if (empty($text)) return [];
    $tokenizer = new WhitespaceTokenizer();
    $tokens = array_filter($tokenizer->tokenize($text));
    $documents = [$tokens];
    $tfidf = new TfIdfTransformer();
    $tfidf->transform($documents);
    $keywordScores = array_combine($documents[0], $documents[0]);
    arsort($keywordScores);
    return array_slice(array_keys($keywordScores), 0, 10);
}
```

CONCLUSION:

The **Marketing Management Module** uses sophisticated NLP-based models for both **summarization** and **keyword extraction**. With the use of **TextRank**, **TfIdfTransformer**, and **PageRank**, the system efficiently summarizes large volumes of content and extracts relevant keywords for fast retrieval and analysis. All of this data is indexed and stored in the **`bf_marketing_temp_scraper`** table, allowing for seamless integration with other components of the marketing management workflow.

3. Database Search (Keyword Matching and Relatability)

The **Marketing Management Module** includes advanced logic for keyword matching and content relatability, ensuring that scraped articles are efficiently searched and matched based on similarity. This process involves both traditional similarity algorithms as well as NLP techniques for more sophisticated matching.

Keyword Matching Logic:

The **Marketing Management Module** employs two primary algorithms for calculating keyword and article similarity:

1. **Cosine Similarity**:
   - **Method**: The `calculateCosineSimilarity()` method in **MyMIMarketing** uses **Cosine Similarity** to match articles based on their keyword vectors.
   - **Functionality**:
     - **Tokenization**: The method first tokenizes both texts into individual words or tokens using the **WhitespaceTokenizer**.
     - **Vectorization**: Once tokenized, the method calculates the cosine similarity between two sets of words by representing them as vectors in a multi-dimensional space.
     - **Score Calculation**: The cosine of the angle between these vectors is used to determine how similar the two texts are (a value between 0 and 1, where 1 means identical and 0 means completely dissimilar).
   - **Example Code Flow**:
     - `calculateCosineSimilarity($text1, $text2)` -> tokenizes both texts -> calculates vector similarity.
2. **Jaccard Similarity**:
   - **Method**: The module can also integrate **Jaccard Similarity**, a simple set-based similarity metric that compares the intersection of two sets (in this case, the keywords of two articles) against their union.
   - **Functionality**:
     - **Set Creation**: Both articles are tokenized into sets of unique keywords.
     - **Score Calculation**: The similarity score is calculated as the ratio of the intersection of these sets (common words) to the union (total unique words across both sets).
   - **Usage**: This method can be useful when the focus is on measuring similarity based on shared keywords between articles.
3. **BERT Embeddings (Optional)**:
   - **Enhancement**: To further enhance the keyword matching process, **BERT embeddings** can be integrated to account for **semantic similarity** between articles. BERT (Bidirectional Encoder Representations from Transformers) is a model that can understand the context of words in a sentence.
   - **Functionality**:
     - **Contextual Understanding**: BERT embeddings can represent words as context-aware vectors, allowing the system to match articles not just based on exact keywords, but also on **semantic similarity**. For example, an article about "cryptocurrency" could be semantically similar to one discussing "digital assets."
     - **Usage**: This feature can be integrated to improve matching where keyword overlap is minimal, but semantic similarity is high.

Synonym Expansion:

The **Marketing Management Module** can also expand keyword matching by leveraging **synonym detection** through NLP libraries such as **WordNet**.

1. **Synonym Matching Using WordNet**:
   - **Method**: By integrating **WordNet** or other NLP tools, the system can fetch synonyms for keywords to broaden the scope of matching.
   - **Functionality**:
     - **Synonym Expansion**: When searching for articles based on specific keywords, the system can fetch related terms or synonyms and include them in the search.
     - **Broader Matches**: This allows for more flexible matching, especially when articles use different terminology to describe the same concepts. For instance, "inflation" and "price rise" could be treated as related terms.

   Example Flow:

   - **Keyword Search**: The user inputs a keyword.
   - **Synonym Expansion**: The system fetches synonyms using WordNet and broadens the search query to include these related terms.
   - **Matching Process**: Articles containing the original keyword or any of its synonyms are included in the search results.

Relatability and Matching Strategy:

- The system calculates a **similarity score** for articles based on keywords and synonyms, utilizing both **Cosine Similarity** and **Jaccard Similarity** for direct keyword overlap, and optionally **BERT embeddings** for semantic matches.
- **Threshold Tuning**: A similarity threshold is applied to filter out unrelated articles. For example, only articles with a similarity score above 0.7 (70%) are considered a match, but this threshold can be adjusted over time based on the relevance of the results.

Storage of Similarity Data:

- The **bf_marketing_temp_scraper** table is central to storing both the original scraped content and the keywords extracted. Additionally, it can store similarity scores for future retrieval and analysis.
  - **Key Fields**:
    - `keywords` (extracted keywords from the article)
    - `similarity_score` (calculated similarity score between articles)
    - `related_articles` (list of related articles based on keyword matching)

Code Snippet Overview:

- **Cosine Similarity Example**:

  ```
  php
  ```

```
Copy code
public function calculateCosineSimilarity($text1, $text2) {
    $tokenizer = new WhitespaceTokenizer();
    $cosine = new CosineSimilarity();
    $set1 = $tokenizer->tokenize($text1);
    $set2 = $tokenizer->tokenize($text2);
    return $cosine->similarity($set1, $set2);
}
```

- **Synonym Expansion Example** (with WordNet integration):

```
php
Copy code
public function fetchSynonyms($keyword) {
    $wordnet = new WordNet();
    return $wordnet->getSynonyms($keyword);
}
```

Conclusion:

The **Marketing Management Module** uses advanced keyword matching logic to relate articles based on **Cosine Similarity**, **Jaccard Similarity**, and optionally **BERT embeddings** for semantic similarity. The system also supports **synonym expansion** through libraries like **WordNet**, ensuring a broader, more flexible search process. All results are stored in the **`bf_marketing_temp_scraper`** table, allowing fast access to related articles based on similarity scores.

4. Generating Fresh Content (Content Consortium)

To implement a **content generation system** in your **MyMI Wallet CI4 Application - Marketing Management Module**, we need to combine the previous data collection, keyword matching, and summarization logic to build a robust **Content Consortium**. This section focuses on how to **generate fresh marketing content** by leveraging existing data and organizing it into coherent narratives.

Content Generation Strategy:

The **Content Consortium** revolves around combining articles, using similarity scores and topic modeling techniques to generate various types of marketing content, from social media posts to long-form articles.

Combining Related Articles:

- **Similarity Scores**: Articles that have been analyzed for **keyword matching** and **similarity** in the previous steps are combined based on their similarity scores.
  - **Cosine Similarity** and **Jaccard Similarity** (discussed earlier) will help identify which articles are closely related based on keyword overlap or conceptual similarity.
  - **Storage**: Similar articles are tagged and grouped together in the database using the **`related_articles`** field in the **`bf_marketing_temp_scraper`** table.
  - **Content Stacking**: Once articles are grouped, the system can combine them into a single piece of content by summarizing them or aligning their key points into a coherent flow.

Natural Language Generation (GPT-3/GPT-4):

- **Model Integration**: To ensure that the generated content is human-readable and coherent, models like **GPT-3** or **GPT-4** can be integrated.
  - **Text Generation**: After combining related articles, the model can be used to generate fresh content by creating summaries, paraphrasing key points, or expanding on existing data.
  - **Content Types**: The system should be capable of producing various types of content based on the user's needs:
    - **Short-Form Content**: Suitable for social media posts.
    - **Long-Form Content**: Suitable for blogs or articles.
    - **Newsletters**: Comprehensive summaries, updates, or reports sent via email.

  Example Flow:

  - Identify related articles based on similarity scores.
  - Pass the grouped articles and relevant keywords to a GPT-3/GPT-4 API for content generation.
  - Generate different content templates based on the format (social media, blog post, newsletter, etc.).

Topic-Based Consortium:

- **Latent Dirichlet Allocation (LDA)**: To organize related articles into meaningful conversations or topic groups, the system can use **LDA** for **topic modeling**.
  - **Purpose**: LDA helps discover underlying topics within a set of documents, enabling the system to group related content based on broader subject areas like "Crypto Regulation" or "Inflation Trends."
  - **Method**: Each article is assigned a topic probability score, indicating how closely it relates to a particular subject.

  Example Process:

  - Articles are run through the **LDA** model, and each is assigned to one or more topic categories.

- These categories become part of the **Content Consortium**, where multiple articles on a similar subject are grouped together.
- The system then builds a timeline or narrative using these groupings, creating content that evolves over time based on related articles (e.g., a series of updates on "Crypto Regulation").

Content Generation Workflow:

1. **Fetch Related Articles**:
   - Use the **Cosine Similarity** or **Jaccard Similarity** scores stored in the database to fetch a group of related articles.
   - Optionally, fetch **synonym-related articles** if WordNet or similar systems are integrated for broader keyword matching.
2. **Topic Assignment**:
   - Run the fetched articles through **LDA** to assign them to topic categories, such as "Crypto Regulation" or "Tech Trends."
   - Group articles under these topics, forming a **topic-based consortium**.
3. **Content Generation**:
   - Based on the chosen format (social media post, blog, or newsletter), pass the grouped articles and relevant keywords to **GPT-3/GPT-4** for generating fresh content.
   - Customize the content for each format:
     - **Short-form posts** (e.g., Twitter, LinkedIn): Generate concise summaries or key takeaways.
     - **Long-form content** (e.g., blog): Generate detailed articles that incorporate insights from multiple related articles.
     - **Newsletters**: Compile a summary of recent articles, grouped by topic.
4. **Content Customization**:
   - Allow marketing teams to manually edit or approve the generated content, fine-tuning it for specific audiences or platforms.
   - **Templates**: Integrate templates for each content type (e.g., preformatted blog structures, social media post templates).

Example Code Flow:

1. **Fetching Related Articles**:

```
php
Copy code
public function getRelatedArticles($articleId) {
    $article = $this->marketingModel->find($articleId);
    $relatedArticles = $this->marketingModel-
>getArticlesBySimilarity($articleId);
    return $relatedArticles;
}
```

2. **Generating Content**:

```
php
Copy code
public function generateContentFromArticles($articleGroup,
$contentType) {
    // Fetch summaries or key points from the grouped articles
    $summaries = array_map([$this, 'summarizeContent'], $articleGroup);

    // Use GPT-3/GPT-4 to generate content based on the content type
    if ($contentType == 'social') {
        $generatedContent = $this->MyMIMarketing-
>generateSocialMediaPost($summaries);
    } elseif ($contentType == 'blog') {
        $generatedContent = $this->MyMIMarketing-
>generateBlogPost($summaries);
    } else {
        $generatedContent = $this->MyMIMarketing-
>generateNewsletter($summaries);
    }

    return $generatedContent;
}
```

Latent Dirichlet Allocation (LDA) Example:

```
php
Copy code
public function assignTopicsToArticles($articleGroup) {
    $ldaModel = new LDA(); // Assuming integration of an LDA model
    $topics = $ldaModel->assignTopics($articleGroup);
    return $topics;
}
```

Storing Generated Content:

- The generated content should be stored in the **`bf_marketing_generated_content`** table for tracking and retrieval.
    - **Key Fields**:
        - `generated_content` (the final content generated from GPT-3/GPT-4)
        - `content_type` (social post, blog, or newsletter)
        - `source_articles` (the IDs of the related articles used to generate the content)

            - `topic` (the topic assigned by LDA)
            - `date_generated`

Conclusion:

By combining related articles based on **similarity scores**, leveraging **GPT-3/GPT-4** for coherent content generation, and organizing these articles into topic-based conversations using **LDA**, the **Content Consortium** can efficiently generate fresh marketing content. The system will support the generation of various content types, such as **short social posts**, **long-form blog posts**, and **newsletters**, while allowing manual input and customization from the marketing team.

5. CONSORTIUM BUILDING (TIMELINE OF RELATED CONVERSATIONS)

In your **MyMI Wallet CI4 Application - Marketing Management Module**, the **Consortium Building** phase focuses on organizing related articles and conversations into timelines, ensuring that marketing teams can interact with and explore clusters of related content. This process involves creating an intuitive user interface and maintaining a coherent narrative across articles.

TIMELINE CREATION:

- **Interactive UI**:
    - Create a user-friendly dashboard for marketing teams to visualize the progression of related articles over time. This can be achieved through a **dynamic timeline UI** where articles related to a specific topic (e.g., "Crypto Regulation") are plotted on a timeline.
    - **Filters**: Allow marketing teams to apply filters to the timeline, such as:
        - **Date**: Filter articles based on specific time frames (e.g., "past month").
        - **Keywords**: Enable keyword-based filters to explore content clusters based on specific terms or phrases.
    - **Content Drill-Down**: The timeline UI should allow users to drill down into each article, viewing its summary, keywords, and related content. Additionally, integrating a feature for highlighting articles with high relevance scores based on **Cosine Similarity** or **BERT embeddings** would improve navigability.
- **Code Integration Example**:

```
php
Copy code
public function createTimelineView($topic) {
    $relatedArticles = $this->marketingModel-
>getArticlesByTopic($topic); // Fetch articles for the topic
    $timelineData = $this->generateTimelineData($relatedArticles); //
Organize them into a timeline
    return view('Marketing/Timeline', ['timeline' => $timelineData]);
}


private function generateTimelineData($articles) {
    $timeline = [];
    foreach ($articles as $article) {
        $timeline[$article->date_scraped][] = $article; // Group
articles by date
    }
    return $timeline;
}
```

COHERENT NARRATIVE:

- **Automated Updates**: The consortium (group of related articles) should automatically update as new articles are scraped, keeping the narrative coherent and ongoing. For example, when new updates about "Crypto Regulation" are scraped, they should be added to the existing **Crypto Regulation Consortium**.
  - The system can **append** new articles to existing conversations or merge them into related topics using **similarity scores**.

Coherent Narrative Strategy:

- **Similarity-Based Merging**: Articles with high **Cosine Similarity** or **Jaccard Similarity** scores should be merged into the same conversation thread.
- **New Conversations**: Articles that introduce entirely new topics (low similarity scores) should trigger the creation of new consortiums.
- **Code Logic**:

```
php
Copy code
public function updateConsortium($newArticle) {
    $similarArticles = $this->getSimilarArticles($newArticle); // Use
similarity scores to find related articles
    if (count($similarArticles) > 0) {
        $this->mergeArticleIntoConsortium($newArticle,
$similarArticles);
    } else {
        $this->createNewConsortium($newArticle);
    }
}
```

---

6. CRON JOBS AND AUTOMATION

In the **MyMI Wallet Marketing Management Module**, automation via **CRON jobs** plays a crucial role in scheduling and executing the tasks for scraping, summarization, and content generation.

AUTOMATION OF SCRAPING AND CONTENT GENERATION:

- **CRON Jobs** will be scheduled to perform daily scraping, summarization, and content generation tasks. The **cron_schedule()** function should handle the following activities:
  - **Scraping New Data**: Daily scraping across all APIs (e.g., NewsAPI, GDELT, Currents API, MediaStack, ContextualWeb) is executed based on the rotational strategy discussed earlier.
  - **Summarization**: Once new articles are scraped, they should be passed through **summarization algorithms** (TextRank, TfIdf).
  - **Keyword Extraction**: Keywords are extracted and stored in the database for matching and content generation purposes.
  - **Content Generation**: Based on grouped articles, the cron job should trigger **content generation** (via GPT-3/4) for creating social media posts, blog posts, and newsletters.
- **Code Example for CRON Job**:

```
php
Copy code
public function cron_schedule() {
    // Scrape new data from APIs
    $this->scrapeDailyData();

    // Summarize and extract keywords from scraped content
    $this->summarizeAndExtractKeywords();

    // Generate new content based on related articles
    $this->generateFreshContent();
}
```

CONTENT DISTRIBUTION:

- Once content is generated, it should be distributed automatically to relevant platforms:
  - **Social Media Platforms**: Automatically post generated content (e.g., short-form content like social posts) to platforms like **Twitter** and **LinkedIn** via APIs.
  - **CMS Integration**: Publish long-form content to content management systems like **WordPress** for blogs.
  - **Email Marketing**: Use platforms like **Mailchimp** or **SendGrid** to distribute newsletters.
- **Code Example for Automated Posting**:

```
php
Copy code
public function distributeContent($content, $platform) {
    if ($platform === 'social') {
        $this->postToSocialMedia($content);
    } elseif ($platform === 'blog') {
        $this->publishToBlog($content);
    } elseif ($platform === 'newsletter') {
        $this->sendNewsletter($content);
    }
}
```

---

7. DEBUGGING AND MONITORING

For ensuring the system's reliability and identifying potential issues during scraping or content generation, it's essential to integrate proper monitoring and alerting tools.

MONITORING TOOLS:

- **Prometheus** and **Grafana** can be used to monitor the system's performance, particularly the health of CRON jobs and the status of content generation. These tools provide real-time metrics on:
    - API scraping success/failure rates.
    - Summarization and content generation performance.
    - System resource usage (e.g., CPU, memory).

ALERTS SETUP:

- Alerts should be triggered for various failure cases:
    - **Failed Scraping Attempts**: If scraping fails repeatedly for a specific API or source, the system should trigger an alert (via **Slack** or **email**).
    - **Content Generation Errors**: If GPT-3/4 encounters issues in content generation (e.g., timeout or failure), the system should send an alert to the appropriate team.
    - **System Downtime**: Alerts for downtime or performance issues (e.g., overloaded CPU or memory).
- **Code Example for Monitoring**:

```
php
Copy code
public function monitorSystemHealth() {
    // Fetch system metrics
    $metrics = $this->fetchSystemMetrics();

    if ($metrics['scraping_failures'] > 10) {
        $this->sendAlert('scraping_failures', $metrics);
    }

    if ($metrics['cpu_usage'] > 90) {
        $this->sendAlert('high_cpu_usage', $metrics);
    }
}
```

---

8. TESTING AND ITERATION

For maintaining high-quality content and ensuring the system works effectively, regular testing and iterative improvements are necessary.

REGULAR TESTING:

- **A/B Testing**: Regularly A/B test the quality of generated content (e.g., social posts, blog posts) to see what resonates better with your target audience.
- **Data Quality Testing**: Ensure that scraped data is relevant and accurate through automated data quality tests.

ALGORITHM TUNING:

- **Keyword Matching Accuracy**: Continuously monitor and fine-tune the algorithms (Cosine Similarity, Jaccard Similarity, BERT embeddings) to improve the accuracy of related article matching.
- **Threshold Adjustments**: Based on feedback from users and marketing teams, adjust the thresholds for matching articles or filtering out low-quality content.
- **Example for A/B Testing**:

```
php
Copy code
public function abTestGeneratedContent($contentA, $contentB) {
    // Send Content A to half the audience, and Content B to the other
half
    $this->sendToAudience($contentA, 'group_A');
    $this->sendToAudience($contentB, 'group_B');

    // Collect engagement metrics (clicks, likes, shares) for both
versions
    $engagementA = $this->fetchEngagementMetrics('group_A');
    $engagementB = $this->fetchEngagementMetrics('group_B');

    return $this->compareEngagement($engagementA, $engagementB);
}
```

CONCLUSION:

- With a structured **Consortium Building**, **CRON Jobs**, and **Automated Monitoring** system in place, the **MyMI Wallet Marketing Management Module** can consistently generate high-quality content and automatically distribute it to relevant platforms. Continuous testing, iteration, and algorithm tuning ensure the system's effectiveness and scalability over time.

KEY CONSIDERATIONS FOR SUCCESS

To ensure the **MyMI Wallet Marketing Management Module** functions optimally, several key factors must be prioritized, focusing on **efficiency**, **scalability**, and a user-friendly **interface**.

EFFICIENCY:

- **Redis Caching**:
  - **Purpose**: To reduce database load, frequently accessed data (e.g., previously scraped articles, keyword matches) should be cached using **Redis**. This will prevent repetitive queries to the database, speeding up content retrieval and processing.
  - **Implementation**: Integrate Redis as a caching layer in the system to store frequently requested content and search results for a limited time (e.g., 24 hours).

```
php
Copy code
// Example of using Redis for caching search results
public function cacheSearchResults($keyword, $results) {
    $redis = new \Redis();
    $redis->connect('127.0.0.1', 6379);
    $redis->setex('search_results_' . $keyword, 86400,
json_encode($results)); // Cache for 24 hours
}
```

- **Message Queues (RabbitMQ)**:
  - **Purpose**: When handling large volumes of scraping tasks, such as simultaneously requesting multiple APIs or performing manual scrapes, using **message queues** ensures that tasks are processed efficiently without overwhelming the system. This enables parallel processing and load distribution.
  - **Implementation**: **RabbitMQ** can be used to manage and queue scraping tasks. Each task (e.g., API call or manual URL scrape) is added to a queue and processed asynchronously, ensuring the system remains responsive even under high demand.

```
php
Copy code
// Example of adding scraping tasks to RabbitMQ
public function addScrapingTaskToQueue($apiEndpoint, $params) {
    $connection = new AMQPStreamConnection('localhost', 5672,
'guest', 'guest');
    $channel = $connection->channel();

    $channel->queue_declare('scraping_queue', false, true, false,
false);
    $message = json_encode(['apiEndpoint' => $apiEndpoint,
'params' => $params]);
```

```
    $msg = new AMQPMessage($message, ['delivery_mode' =>
AMQPMessage::DELIVERY_MODE_PERSISTENT]);

    $channel->basic_publish($msg, '', 'scraping_queue');
    $channel->close();
    $connection->close();
}
```

SCALABILITY:

- **Database Partitioning and Horizontal Scaling**:
    - **Purpose**: As the system scales and handles more data, web scraping tasks, and content generation, it's crucial to implement a scalable database architecture. Partitioning the database ensures that queries are distributed evenly across multiple storage locations, improving performance.
    - **Implementation**: Partition tables like `bf_marketing_temp_scraper` based on factors such as **date** or **keywords**. This ensures efficient querying and retrieval even as the volume of data grows.

      ```
      sql
      Copy code
      -- Example of partitioning bf_marketing_temp_scraper by date
      ALTER TABLE bf_marketing_temp_scraper
      PARTITION BY RANGE (YEAR(date_scraped)) (
          PARTITION p2023 VALUES LESS THAN (2024),
          PARTITION p2024 VALUES LESS THAN (2025)
      );
      ```
- **Horizontal Scaling for Scraping**:
    - **Strategy**: When the scraping task load increases, deploy additional scraping instances across multiple servers (horizontal scaling) to distribute the workload evenly. Each instance can process a portion of the APIs or manual scraping tasks, ensuring that no single server is overwhelmed.

USER INTERFACE:

The **MyMI Marketing Management Module** must feature a **user-friendly dashboard** to allow marketing teams to interact with scraped data, generated content, and consortiums. This involves creating intuitive workflows for content approval, editing, scheduling, and visualization.

- **Search for Relevant Content**:
    - **Search Functionality**: Implement a powerful search feature that allows marketing teams to search for content by **keywords**, **dates**, and **similarity scores**.
    - **Redis Integration**: Cache frequent search queries to speed up retrieval times.

```
php
Copy code
public function searchContent($keyword) {
    $cachedResults = $this->getCachedSearchResults($keyword);
    if ($cachedResults) {
        return $cachedResults; // Return cached results if
available
    }

    // Otherwise, search the database
    $results = $this->marketingModel-
>searchArticlesByKeyword($keyword);
    $this->cacheSearchResults($keyword, $results);
    return $results;
}
```

- **Content Approval and Editing**:
  - **Approval Workflow**: Create a workflow that allows the marketing team to review, edit, and approve auto-generated content before it is published. This should be easily manageable through the dashboard.
  - **Content Versioning**: Implement content versioning to track changes made by the marketing team during editing and allow rollback to previous versions if necessary.
- **Content Scheduling**:
  - **Content Calendar**: Incorporate a **content scheduling calendar** where approved content can be scheduled for publication across different platforms (e.g., social media, blog posts).
  - **Automation**: Scheduled content should automatically be published to platforms like **Twitter**, **WordPress**, or sent via **Mailchimp**.

```
php
Copy code
public function scheduleContent($contentId, $platform,
$publishDate) {
    $this->marketingModel->addScheduledPost($contentId,
$platform, $publishDate);
    // Set a cron job to publish content on the scheduled date
    $this->setupCronJob($contentId, $publishDate, $platform);
}
```

- **Timeline and Consortium View**:
  - **Visualization**: Build a **timeline view** for marketing teams to track the evolution of content and topics over time. Articles related to a specific topic or consortium (e.g., "Crypto Regulation") should be grouped visually, with the ability to explore the ongoing conversation.
  - **Interactive Filtering**: Allow filtering by date, keywords, and similarity scores to explore related content easily.

CONCLUSION:

By following this structured approach, leveraging **Redis caching**, **message queues**, and **scalable architecture**, the **MyMI Wallet Marketing Management Module** will operate efficiently even under heavy data loads. The combination of automated API scraping, manual data entry, and content generation will ensure the marketing team has access to **high-quality, relevant content**. The interactive **dashboard** allows marketing teams to **search, approve, schedule**, and **track** content effortlessly, providing flexibility and control over the entire content pipeline.

Through the **API rotation strategy** and automated content generation, the system will maximize free API usage while maintaining a steady flow of fresh, engaging content across multiple channels.