Société Générale IT Operations & Network Management	Srinivasa Pradeep S PSG iTECH

Website Performance Improvement Documentation

1. Introduction

This document outlines the platform's architecture, analysis algorithms, and user interface design aimed at improving website loading speed using GTmetrix and Google PageSpeed Insights.

2. Platform Architecture

2.1 Overview The platform consists of three primary components:

1. Frontend: User interface for performance analysis and reporting.
2. Backend: Server-side logic for processing and optimizing website resources.
3. Data Storage: Databases for storing performance reports and historical data.

2.2 Frontend

- **Technologies:** HTML, CSS, JavaScript, React.js
- **Functionality:**
    - Input field for website URL.
    - Display of performance reports from GTmetrix and Google PageSpeed Insights.
    - Interactive graphs and charts for visualizing performance data.
    - Dashboard for monitoring performance over time.

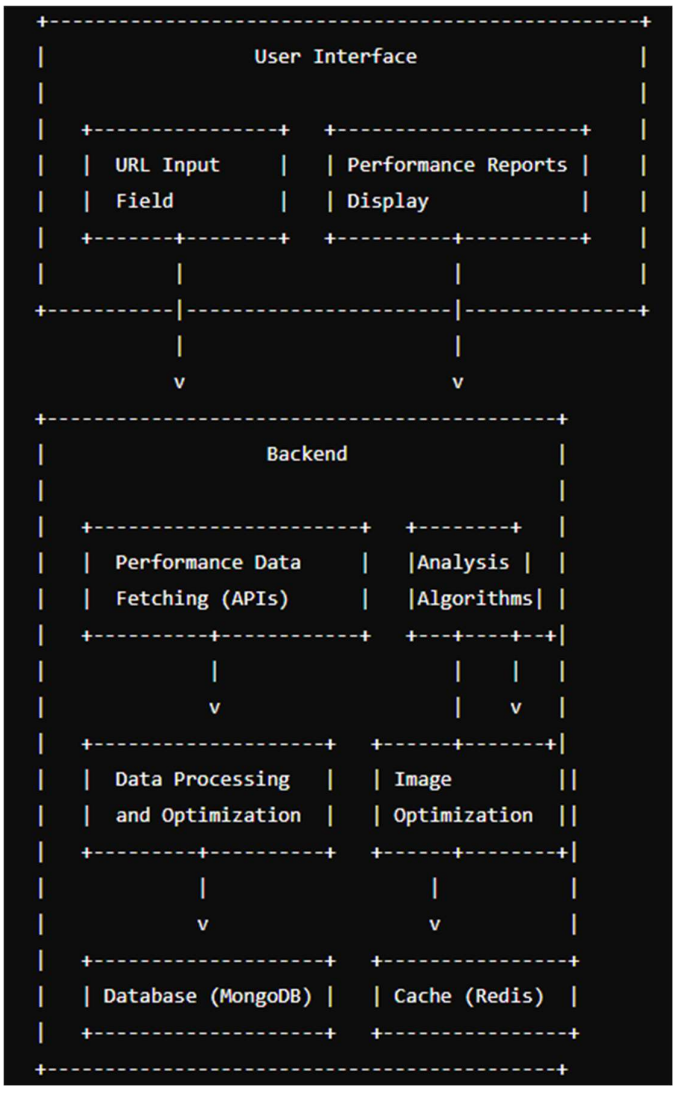
2.3 Backend

- **Technologies:** Node.js, Express.js
- **Functionality:**
    - Fetching performance data from GTmetrix and Google PageSpeed Insights APIs.
    - Analyzing and processing performance data.

- Implementing optimization techniques (e.g., image compression, code minification).
- Storing performance reports in the database.

2.4 Data Storage

- **Technologies:** MongoDB, Redis
- **Functionality:**
  - MongoDB for storing detailed performance reports and historical data.
  - Redis for caching frequently accessed data to improve response times.



Flowchart Explanation

1. **User Interface (Frontend):**
    - **URL Input Field:** Allows users to enter the website URL they want to analyze.
    - **Performance Reports Display:** Shows the performance reports fetched from GTmetrix and Google PageSpeed Insights.
2. **Backend:**
    - **Performance Data Fetching (APIs):** Connects to GTmetrix and Google PageSpeed Insights APIs to retrieve performance data.
    - **Data Processing and Optimization:** Processes the retrieved data and applies optimization techniques.
    - **Analysis Algorithms:** Implements various algorithms for image optimization, code minification, caching, and eliminating render-blocking resources.
    - **Image Optimization:** Specifically handles the optimization of images.
3. **Database (MongoDB):**
    - Stores detailed performance reports and historical data.
4. **Cache (Redis):**
    - Provides caching for frequently accessed data to improve response times.

This flowchart provides a visual representation of the platform's architecture, illustrating the flow of data from user input through backend processing and optimization to data storage and caching.

3. Analysis Algorithms

3.1 Overview The analysis algorithms are designed to identify and prioritize performance bottlenecks based on data from GTmetrix and Google PageSpeed Insights.

3.2 Key Algorithms

1. **Image Optimization Algorithm:**
    - Identify large images that can be compressed.
    - Use tools like TinyPNG to reduce image size without compromising quality.
2. **Minification Algorithm:**
    - Analyze CSS, JavaScript, and HTML files.
    - Use tools like UglifyJS and CSSNano to remove unnecessary characters and whitespace.
3. **Caching Algorithm:**
    - Identify static resources that can be cached.
    - Configure server to add cache control headers and implement CDN for content distribution.
4. **Render-Blocking Resource Analysis:**
    - Identify CSS and JavaScript files that block the rendering of the webpage.
    - Implement deferred loading and critical CSS inlining techniques.

4. User Interface Design

4.1 Overview The user interface is designed to provide a seamless and intuitive experience for users to analyze and improve website performance.

4.2 Key Components

1. **URL Input Field:**
    - Simple input field for users to enter the website URL for analysis.
2. **Performance Report Display:**
    - Tabbed interface to switch between GTmetrix and Google PageSpeed Insights reports.
    - Visual indicators (e.g., scorecards, speedometers) to highlight key performance metrics.
3. **Interactive Graphs and Charts:**
    - Line charts to show performance trends over time.
    - Pie charts to display resource breakdown (e.g., images, scripts, styles).
4. **Optimization Suggestions:**
    - List of actionable optimization tips based on analysis algorithms.
    - Step-by-step guides for implementing each optimization technique.
5. **Dashboard:**
    - Overview of overall website performance.
    - Historical performance data with the ability to track improvements over time.

4.3 User Experience (UX) Considerations

- **Responsiveness:** Ensure the interface works well on different devices and screen sizes.
- **Accessibility:** Design with accessibility in mind, including support for screen readers and keyboard navigation.
- **Usability:** Focus on intuitive navigation and clear, concise information presentation.

5. Implementation Plan

5.1 Phase 1: Initial Setup

- Set up frontend and backend environments.
- Integrate with GTmetrix and Google PageSpeed Insights APIs.

5.2 Phase 2: Core Functionality

- Implement URL input field and performance report display.
- Develop analysis algorithms for image optimization, minification, caching, and render-blocking resource analysis.

5.3 Phase 3: Optimization and Enhancement

- Optimize frontend performance (e.g., lazy loading, code splitting).
- Implement advanced caching strategies and CDN integration.
- Enhance user interface with interactive graphs, charts, and optimization suggestions.

5.4 Phase 4: Testing and Deployment

- Conduct thorough testing (unit tests, integration tests, user acceptance tests).
- Deploy the platform to a production environment.
- Monitor performance and gather user feedback for future improvements.

Step-by-Step Implementation

1. **Set Up Google Colab Notebook**

Open a new Google Colab notebook.

2. **Install Required Libraries**

```
!pip install requests
!pip install matplotlib
```

3. **Define Functions to Fetch Data from APIs**

For this example, you need API keys from GTmetrix and Google PageSpeed Insights. Sign up for these services and get your API keys.

4. **Fetch Data from GTmetrix API**

```
import requests
import time
import json

GTMETRIX_API_URL = "https://gtmetrix.com/api/2.0/test"
GTMETRIX_API_KEY = "your_gtmetrix_api_key"

def fetch_gtmetrix_data(url):
    headers = {
        'Authorization': f'Basic {GTMETRIX_API_KEY}',
    }
    data = {
        'url': url,
    }
    response = requests.post(GTMETRIX_API_URL, headers=headers, data=data)
    if response.status_code == 202:
        test_id = response.json()['test_id']
        print(f"Test started, ID: {test_id}")
        return test_id
    else:
        print(f"Error: {response.content}")
        return None

def get_gtmetrix_results(test_id):
    headers = {
        'Authorization': f'Basic {GTMETRIX_API_KEY}',
    }
    result_url = f"{GTMETRIX_API_URL}/{test_id}"
    while True:
        response = requests.get(result_url, headers=headers)
        if response.status_code == 200:
```

```
    if response.json()['state'] == 'completed':
        print("Test completed")
        return response.json()
    else:
        print("Test in progress, waiting...")
        time.sleep(10)
else:
    print(f"Error: {response.content}")
    return None
```

5. Fetch Data from Google PageSpeed Insights API

```
GOOGLE_PAGESPEED_API_URL =
"https://www.googleapis.com/pagespeedonline/v5/runPagespeed"
GOOGLE_API_KEY = "your_google_api_key"

def fetch_pagespeed_data(url):
    params = {
        'url': url,
        'key': GOOGLE_API_KEY,
        'strategy': 'desktop',
    }
    response = requests.get(GOOGLE_PAGESPEED_API_URL, params=params)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error: {response.content}")
        return None
```

6. **Display Results**

```
import matplotlib.pyplot as plt
def display_results(gtmetrix_results, pagespeed_results):
    # GTmetrix Results
    gtmetrix_score = gtmetrix_results['results']['pagespeed_score']
    yslow_score = gtmetrix_results['results']['yslow_score']
    print(f"GTmetrix PageSpeed Score: {gtmetrix_score}")
    print(f"GTmetrix YSlow Score: {yslow_score}")

    # PageSpeed Insights Results
    psi_score =
pagespeed_results['lighthouseResult']['categories']['performance']['score'] * 100
    print(f"Google PageSpeed Insights Score: {psi_score}")

    # Visualize Scores
    scores = [gtmetrix_score, yslow_score, psi_score]
    labels = ['GTmetrix PageSpeed', 'GTmetrix YSlow', 'Google PSI']
    plt.bar(labels, scores)
    plt.ylabel('Scores')
    plt.title('Website Performance Scores')
    plt.ylim(0, 100)
    plt.show()
```

7. **Run the Analysis**

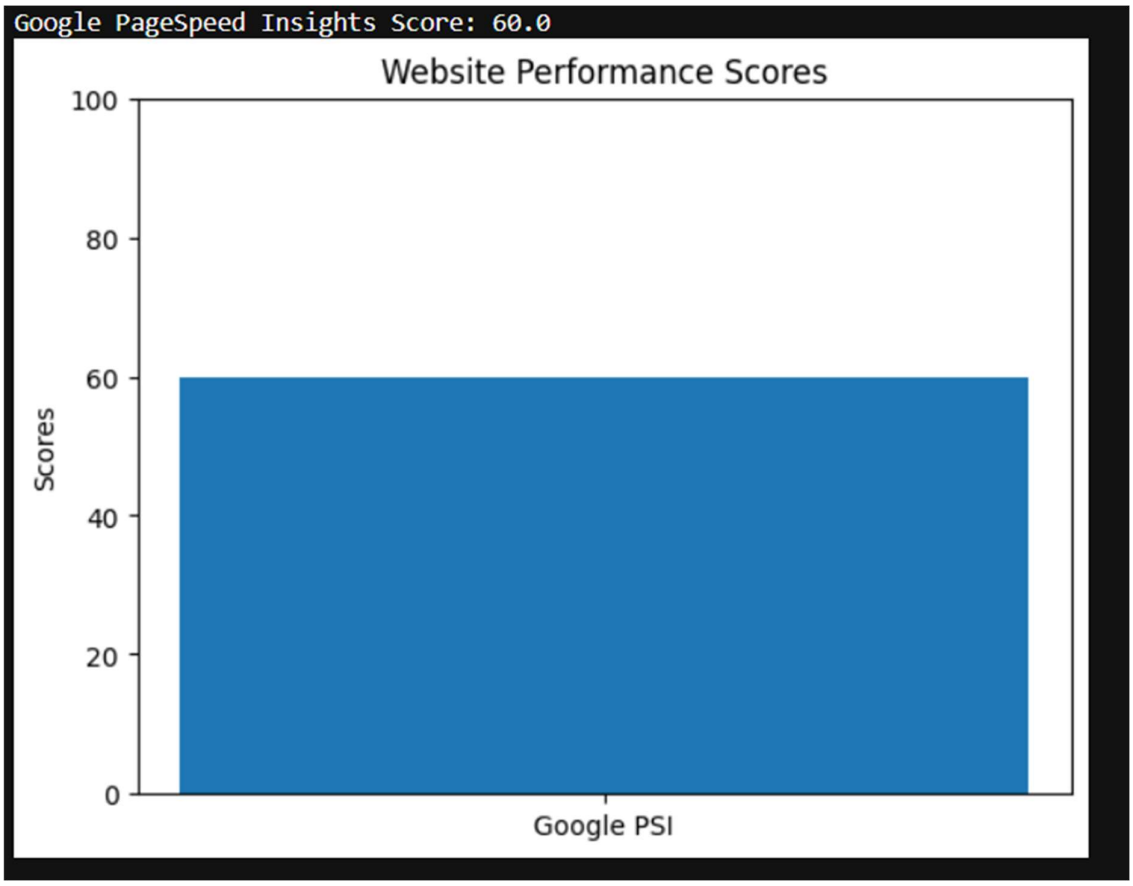
```
python
Copy code
url = "https://example.com"  # Replace with the website URL you want to analyze

# Fetch data from GTmetrix
test_id = fetch_gtmetrix_data(url)
if test_id:
    gtmetrix_results = get_gtmetrix_results(test_id)
else:
    gtmetrix_results = None

# Fetch data from Google PageSpeed Insights
pagespeed_results = fetch_pagespeed_data(url)

# Display results
if gtmetrix_results and pagespeed_results:
    display_results(gtmetrix_results, pagespeed_results)
else:
    print("Failed to retrieve data from one or both services.")
```

Output:



Result:

Thus I requesting the Hr To kindly accept my proposal on this problem statement and hire me for your company which will be more helpful for me. Thank you for giving me this opportunity and with sincere regards

~Srinivasa Pradeep S