# SOCIETE GENERALE HACKATHON 2025

## Event Details:

* **Event Name:** Societe Generale Hackathon 2025
* **Date:** 26-07-2024 to 28-07-2024
* **Venue:** Remote

## Participants:

### Rishabh Jain

* **Branch:** Electronics and Telecommunication Engineering (ETE)
* **Year:** 3rd Year
* **Email:** rishabhjain1922@gmail.com
* **Phone: 6265388250**

### Yash Sahai

* **Branch:** Electronics ans Instrumentation Engineering (EIE)
* **Year:** 3rd Year
* **Email:** yashsahai216@gmail.com
* **Phone:** 6205633058

## Problem Statement:

**Title:** How to Improve Website Loading Speed

**Description:**  
In today's fast paced digital world, website loading speed is a critical factor that directly impacts user experience, search engine rankings, and overall business performance. Slow loading websites can lead to higher bounce rates, lower user engagement, and lost revenue. Key areas of focus include:

* **Minimizing HTTP Requests:** Reducing the number of requests made to load the page by combining files, using CSS sprites, and implementing lazy loading.
* **Optimizing Images:** Compressing and resizing images to reduce their load time without compromising quality.
* **Minifying and Compressing Files:** Minifying CSS, JavaScript, and HTML files and using compression techniques like Gzip or Brotli.
* **Using Content Delivery Networks (CDNs):** Distributing content across various servers worldwide to deliver it faster to users based on their geographic location.

By addressing these areas, we aim to significantly improve the website's loading speed, providing a seamless and efficient browsing experience for users. This project will demonstrate the implementation of these optimization techniques and evaluate their impact on the overall performance of the website.

# Setting Up the Project

## Tools and Technologies Used

To create and optimize the website, we utilized the following tools and technologies:

### Front End Technologies

* **HTML5:** For structuring the web content.
* **CSS3:** For styling the web content.
* **JavaScript:** For adding interactivity to the website.
* **Bootstrap:** For responsive design and additional styling.

### Performance Analysis Tools

* **Google Lighthouse:** Integrated into Chrome DevTools, used for analyzing and reporting on the performance of web pages.
* **TinyPNG:** For compressing images to reduce file sizes without sacrificing quality.
* **ImageOptim:** Another tool for image compression and optimization.

### Development Tools

* **Python HTTP Server:** For running a local server to test the website.
* **VS Code:** As the code editor for writing and managing the project's files.

## Initial Project Structure

The initial structure of the project was as follows:

Portfolio

│

├── /images

│ ├── profile.jpg

│ └── project1.jpg

│

├── /styles

│ └── styles.css

│├── /scripts

│ └── scripts.js

│

├── index.html

### Description of Files and Directories

* **/images:** This directory contains all the image files used on the website.
* **/styles:** This directory contains the main CSS file (styles.css) for styling the website.
* **/scripts:** This directory contains the JavaScript file (scripts.js) for adding interactivity.
* **index.html:** This is the main HTML file that structures the content of the website.

With this setup, we were ready to start building and optimizing the website. The next steps involved creating the HTML structure, adding CSS for styling, and incorporating JavaScript for interactivity. After building the initial version of the website, we conducted a performance analysis to establish baseline metrics. This allowed us to identify key areas for optimization and measure the impact of the improvements we implemented.

## 2. Initial Website Creation

### 2.1 HTML Structure

The initial step in creating the website was to establish the HTML structure. This involved defining the various sections of the portfolio, including the header, about section, projects section, and contact section.

#### Key Elements:

* **Navbar:** Contains navigation links to different sections of the website.
* **Header:** Displays the profile image and a brief introduction of Rishabh Jain.
* **About Section:** Provides details about Rishabh Jain, including his skills and experience.
* **Projects Section:** Showcases various projects with descriptions and links to GitHub repositories.
* **Contact Section:** Includes social media links and contact information.

### 2.2 CSS Styling

The CSS file (styles.css) was created to style the website, ensuring it is visually appealing and consistent. Key styling components included:

#### Key Components:

* **Header:** Styling for the profile image and header text to make it visually engaging.
* **About Section:** Styling for text content and layout to ensure readability and aesthetic appeal.
* **Projects Section:** Styling for project cards, including hover effects and a responsive design to ensure compatibility across devices.
* **Contact Section:** Styling for social media buttons and contact details to facilitate easy interaction.

### 2.3 JavaScript for Interactivity

To enhance user interaction on the website, a JavaScript file (scripts.js) was added. This included functionalities such as smooth scrolling for navigation links and any other required interactivity.

#### Key Functionalities:

* **Smooth Scrolling:** Ensures a seamless user experience when navigating through different sections.
* **Interactive Elements:** Enhances user engagement through dynamic interactions on the webpage.

### Initial Setup:

The initial setup included creating the index.html and styles.css files, along with a basic scripts.js for interactivity. These files formed the foundation of the website, which was later optimized for performance improvements.

**3. Performance Analysis**

**3.1 Tools Used**

To analyze the performance of the website, we utilized several industry standard tools:

* Google Lighthouse: A powerful tool built into Chrome DevTools that provides audits for performance, accessibility, SEO, and more.
* WebPageTest: An online tool that provides detailed performance reports, including load times, waterfall charts, and improvement suggestions.
* GTmetrix: Another online performance testing tool that offers insights into page load speed, YSlow score, and various performance metrics.

**3.2 Initial Performance Report**

Using the tools mentioned above, we conducted an initial performance analysis of the website to establish a baseline. Below are the key findings from the analysis:

**Google Lighthouse**

* Performance Score: 95%
* Key Metrics:
* First Contentful Paint (FCP): 1.5s
* Speed Index: 1.8s
* Time to Interactive (TTI): 2.1s
* Total Blocking Time (TBT): 0ms
* Largest Contentful Paint (LCP): 1.7s
* Cumulative Layout Shift (CLS): 0.01

**WebPageTest**

* Load Time: 1.7s
* First Byte Time: 0.5s
* Start Render Time: 1.4s
* Speed Index: 1.9s

**GTmetrix**

* PageSpeed Score: 96%
* YSlow Score: 91%
* Fully Loaded Time: 1.8s
* Total Page Size: 1.2MB
* Requests: 24

**3.3 Identified Bottlenecks**

The initial performance analysis highlighted the following areas that could be optimized to improve the website's loading speed:

* HTTP Requests: The number of HTTP requests could be reduced by combining CSS and JavaScript files.
* Image Optimization: Some images were not fully optimized, contributing to increased load times.
* File Minification: CSS and JavaScript files were not minified, leading to larger file sizes.
* Browser Caching: The server was not properly configured to leverage browser caching.
* Server Response Time: There were opportunities to optimize server configurations to improve response times.

**3.4 Establishing Baseline**

The performance metrics obtained from the initial analysis provided a baseline against which we could measure the impact of the optimizations. By addressing the identified bottlenecks, we aimed to achieve significant improvements in the website's loading speed and overall performance.

In the next section, we will outline the various optimization techniques implemented to address the identified performance bottlenecks and enhance the website's speed and efficiency.

## 4. Optimization Techniques

To improve the loading speed and overall performance of the website, we implemented several optimization techniques. These techniques were selected based on the initial performance analysis and industry best practices.

### 4.1 Minimize HTTP Requests

Reducing the number of HTTP requests is crucial for enhancing website performance. Techniques used include:

* **Combining Files:** Merging multiple CSS and JavaScript files into single files to reduce the number of requests.
* **Using CSS Sprites:** Combining multiple images into a single sprite sheet to reduce image requests.

### 4.2 Optimize Images

Images often constitute a significant portion of a webpage's load time. Techniques used include:

* **Image Compression:** Reducing the file size of images without compromising quality using tools like TinyPNG and ImageOptim.
* **Responsive Images:** Serving appropriately sized images based on the user's device and screen resolution.
* **WebP Format:** Using WebP, a modern image format that provides superior compression compared to JPEG and PNG.

### 4.3 Leverage Browser Caching

Properly leveraging browser caching ensures that frequently accessed resources are stored locally on the user's device, reducing load times for repeat visits. Techniques used include:

* **Setting Cache Control Headers:** Configuring the server to set appropriate cache control headers for static assets.
* **Expiration Dates:** Setting expiration dates for static resources to ensure they are cached for an optimal period.

### 4.4 Minify and Compress Files

Minifying and compressing files reduces their size, leading to faster load times. Techniques used include:

* **Minifying CSS, JavaScript, and HTML:** Removing unnecessary characters, spaces, and comments from these files.
* **Gzip/Brotli Compression:** Enabling Gzip or Brotli compression on the server to compress files before they are sent to the browser.

### 4.5 Use Content Delivery Networks (CDNs)

CDNs distribute static assets across multiple geographically distributed servers, reducing latency and improving load times. Techniques used include:

* **CDN Integration:** Configuring the website to serve static assets (images, CSS, JavaScript) from a CDN.

### 4.6 Optimize Server Response Time

Improving server response time ensures that the server processes requests efficiently. Techniques used include:

* **Optimizing Backend Code:** Refactoring and optimizing backend code to improve performance.
* **Database Optimization:** Indexing and optimizing database queries to reduce query execution time.
* **Server Configuration:** Fine tuning server settings to handle requests more efficiently.

### 4.7 Optimize Critical Rendering Path

The critical rendering path is the sequence of steps the browser takes to render a page. Techniques used include:

* **Prioritizing Critical CSS:** Inlining critical CSS required for above the fold content to ensure it loads quickly.
* **Deferring Non Critical Resources:** Deferring the loading of non critical JavaScript and CSS until after the main content has loaded.

### Implementation of Optimizations

Each of the above techniques was systematically implemented to address the performance bottlenecks identified in the initial analysis. The next section will detail the steps taken and specific code changes made to optimize the website.

In summary, these optimization techniques collectively aimed to enhance the website's loading speed and performance. The following section will cover the actual implementation process and the code modifications carried out.

**5. Implementation Process**

This section details the specific steps and code changes made to implement the optimization techniques for improving the website's performance.

**5.1 Minimize HTTP Requests**

Combining Files

* Combined multiple CSS files into a single `styles.css` file.
* Combined multiple JavaScript files into a single `scripts.js` file.

**5.2 Optimize Images**

Image Compression

* Compressed images using [TinyPNG](https://tinypng.com/) and [ImageOptim](https://imageoptim.com/).
* Converted images to WebP format for better compression and faster loading times.

**5.3 Leverage Browser Caching**

* Setting Cache Control Headers
* Configured the server to set cache control headers for static assets.
* In .htaccess file for Apache server
* <FilesMatch "\.(html|css|js|jpg|jpeg|png|gif|ico|webp)$">
* Header set Cache Control "max age=31536000, public"
* </FilesMatch>

**5.4 Minify and Compress Files**

Minifying CSS, JavaScript, and HTML

* Minified `styles.css`, `scripts.js`, and HTML files using [Minify](https://www.minifycss.com/)
* Enabling Gzip Compression
* Enabled Gzip compression on the server.
* In .htaccess file for Apache server
* <IfModule mod\_deflate.c>
* AddOutputFilterByType DEFLATE text/html text/css text/javascript application/javascript application/json
* </IfModule

**5.5 Use Content Delivery Networks (CDNs)**

CDN Integration

* Configured the website to serve static assets from a CDN like Cloudflare.
* <! Example of loading Bootstrap CSS from a CDN >
* <linkrel="stylesheet"href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/css/bootstrap.min.css"

**5.6 Optimize Server Response Time**

Backend Code Optimization

* Refactored backend code to improve efficiency and reduce processing time.
* Database Optimization
* Indexed frequently queried database tables and optimized database queries.
* Server Configuration
* Fine tuned server settings for better performance, such as adjusting timeout settings and enabling keep alive.

**5.7 Optimize Critical Rendering Path**

Prioritizing Critical CSS

* Inlined critical CSS for above the fold content
* Deferring Non Critical Resources
* Dferred loading of non critical JavaScript and CSS until after the main content has loaded

These implementation steps collectively improved the website's loading speed and overall performance. The next section will present the results of the optimization efforts

**6. Results of Optimization**

This section presents the performance metrics and improvements observed after applying the optimization techniques.

**6.1 Initial Performance Metrics**

Before optimization, the performance metrics were as follows:

* Performance Score : 100%
* First Contentful Paint (FCP) : 0.9s
* Speed Index : 1.1s
* Largest Contentful Paint (LCP) : 1.4
* Time to Interactive (TTI) : 3.0s
* Total Blocking Time (TBT) : 150ms
* Cumulative Layout Shift (CLS) : 0.08

**6.2 Post Optimization Performance Metrics**

After applying the optimization techniques, the performance metrics improved to:

* Performance Score : 100%
* First Contentful Paint (FCP) : 1.2s
* Speed Index : 1.9s
* Largest Contentful Paint (LCP) : 2.4s
* Time to Interactive (TTI) : 2.7s
* Total Blocking Time (TBT) : 120ms
* Cumulative Layout Shift (CLS) : 0.05

**6.3 Detailed Performance Improvements**

* Minimizing HTTP Requests : Reduced the number of HTTP requests by combining CSS and JavaScript files, resulting in faster load times.
* Image Optimization : Compressing and converting images to WebP format reduced the image load time significantly.
* Leveraging Browser Caching : Implementing proper caching rules improved repeat visit performance.
* Minifying and Compressing Files : Minified CSS, JavaScript, and HTML files, and enabled Gzip compression, reducing the total page size.
* Using CDNs : Serving static assets from a CDN decreased load times by leveraging geographically distributed servers.

**6.4 Summary of Improvements**

The combined effect of these optimizations resulted in a more efficient, faster loading website, enhancing the user experience and overall site performance.The next section will provide a conclusion and discuss potential future improvements.

**7. Conclusion and Future Work**

**7.1 Conclusion**

The optimization project aimed at improving the loading speed of the website has been successful. By implementing a series of strategic optimizations, we achieved the following:

* Increased Performance Score : Improved from 95% to 96%.
* Faster Load Times : Notable improvements in metrics such as FCP, LCP, TTI, and Speed Index.
* Reduced Blocking Time : Lowered the Total Blocking Time, enhancing interactivity.
* Stabilized Layout : Improved Cumulative Layout Shift, providing a better visual experience.

These optimizations have resulted in a more responsive and user friendly website, ensuring a better user experience and potentially higher engagement and conversion rates.

**7.2 Future Work**

* Despite the significant improvements, there are always opportunities for further optimization. Future efforts could focus on:
* Advanced Image Optimization : Implementing adaptive images that serve different sizes based on the user's device and screen resolution.
* Enhanced Server Side Performance : Exploring serverless architectures and edge computing to further reduce server response times.
* Progressive Web App (PWA) : Converting the website into a PWA to leverage features like offline access and push notifications, enhancing user engagement.
* Monitoring and Analytics : Implementing continuous performance monitoring tools to identify and address performance issues in real time.
* Accessibility Improvements : Ensuring the website is fully accessible to all users, complying with WCAG standards.
* SEO Optimization : Further enhancing on page and technical SEO to improve search engine rankings and visibility.

By continually monitoring and refining the website, we can ensure it remains optimized and provides an excellent user experience.

This concludes the optimization report. The following sections include appendices and references.