

Nakatulong ang paggamit ng mga sariling komposisyon ng mga panitikang pambata upang mapataas ang antas ng pang unawa sa pagbasa. Notice that constraint (4) in the LP for a non-credible agent splits into constraints (10) and (11) for a credible agent. Constraint (4) considers all possible deviations over reports and actions of the agent jointly, while (10) and (11) consider such deviations separately. **1.4 Project Scope**

The scope of the A Coffee Virtual QR project is designed to support the internal administrative needs of the coffee shop's staff, including inventory management, sales tracking, and ordering processes. This project is solely intended for internal use by company personnel, such as managers and staff, and not for customer interaction.

### **Inventory System**

- Monitor stock levels in real-time to track available products.
- Set up automatic alerts for low-stock items to avoid shortages.
- Store and manage detailed product information, including descriptions and prices.
- Generate reports on stock levels, sales, and other relevant metrics to support decision-making.

### **Sales System**

- Facilitate in-store sales processes, including payment handling.
- Manage order fulfillment efficiently for accurate and timely processing.
- Monitor sales in real-time and analyze trends.
- Break down sales data by product for insights into each item's performance.

### **Ordering System**

- Create, manage, and track purchase orders for inventory restocking.
- Implement an approval process for purchase orders to ensure authorization.
- Automatically update stock levels upon successful inventory verification.
- Record and verify received inventory against purchase orders.
- Generate reports on purchase history, stock levels, and approval timelines.

The A Coffee Virtual QR project will improve the operational efficiency of the coffee shop by streamlining these key processes and enabling better data-driven decisions within the organization.

4o

**You said:**

how to optimize the network on my webpage

**ChatGPT said:**

ChatGPT

To optimize your webpage's network performance, you can focus on minimizing the number of requests, optimizing asset loading, and improving data transfer speeds. Here are some practical steps:

## 1. Minimize HTTP Requests

- **Combine CSS, JS, and Image Files:** Reduce the number of separate files that need to be loaded by combining CSS and JavaScript files where possible.
- **Use CSS Sprites:** For small icons or images, combine them into a single image sprite and use CSS to display specific parts.
- **Eliminate Unnecessary Plugins:** Remove or consolidate any plugins that require extra server calls.

## 2. Optimize Image Loading

- **Compress Images:** Use tools like TinyPNG or ImageOptim to reduce file sizes without noticeable quality loss.
- **Use Next-Gen Formats:** Switch to formats like WebP and AVIF, which have better compression rates.
- **Lazy Loading:** Only load images as they're needed, such as when they appear in the user's viewport.

## 3. Leverage Caching

- **Browser Caching:** Set expiration headers to enable browsers to cache assets. This reduces load time for repeat visitors.
- **Server-Side Caching:** Implement caching on the server for pages that don't need real-time updates.
- **Use Content Delivery Network (CDN):** CDNs distribute static assets (images, stylesheets, etc.) across global servers, reducing load times based on the user's geographic location.

## 4. Minify and Compress Assets

- **Minify CSS, JS, and HTML:** Remove whitespace, comments, and unnecessary code. Tools like UglifyJS and CSSNano can help automate this.
- **GZIP Compression:** Enable GZIP or Brotli compression on your server. This reduces the size of files sent over the network.

## 5. Optimize JavaScript Loading

- **Defer Non-Essential Scripts:** Use the `async` or `defer` attributes to load non-critical JavaScript after the main content.
- **Reduce Third-Party Scripts:** Minimize third-party libraries, analytics tools, and ads, as they can significantly impact load times.
- **Use Code Splitting:** Break JavaScript bundles into smaller parts so only essential code loads initially.

## 6. Optimize Fonts

- **Choose Web-Optimized Fonts:** Use system fonts where possible or limit the number of custom fonts.
- **Preload Fonts:** Use the `preload` attribute for important fonts to load them earlier.
- **Use Font Subsetting:** Include only the required characters (especially for languages with large character sets) to reduce font file size.

## 7. Improve Server Response Times

- **Optimize Server Hardware and Hosting Plan:** Ensure your server resources match your traffic demands.
- **Reduce Server Load:** Use caching layers, load balancers, and database optimization to minimize server processing time.
- **Optimize Database Queries:** Use indexes, reduce joins, and optimize queries to speed up database response times.

## 8. Implement Prefetching and Preloading

- **DNS Prefetching:** Use DNS prefetching for third-party assets or external domains that users may access.
- **Preload Key Requests:** Use the `preload` and `prefetch` attributes to load critical resources (like fonts or stylesheets) sooner.

## 9. Monitor and Measure Performance

- **Use Tools like Lighthouse, GTmetrix, and WebPageTest:** Regularly analyze performance and identify bottlenecks.
- **Set up Real User Monitoring (RUM):** Tools like Google Analytics or New Relic can help monitor real-time performance from users' perspectives.

Following these practices will lead to a faster, more responsive webpage, improving the user experience and helping with SEO as well.