Morent Car Rental

Day 5

On this day i have checked my marketplace through lighthouse and all things are working well.

Performance Optimization Steps Taken

1. **Lazy Loading Assets**:
    - Implemented lazy loading for images and other assets to improve page load times.
    - Verified that offscreen images load only when they are about to enter the viewport.
2. **Code Splitting**:
    - Used code splitting to load JavaScript bundles dynamically, reducing the initial load time.
    - Verified that non-critical scripts load asynchronously.
3. **Caching**:
    - Enabled caching for static assets to improve repeat load performance.
    - Tested cache behavior after updates to ensure users see the latest content.
4. **Database Query Optimization**:
    - Reduced the number of database calls by combining queries and indexing frequently accessed fields.
    - Verified query execution times and optimized where delays were observed.
5. **Minification**:
    - Minified CSS, JavaScript, and HTML files to reduce file sizes.
    - Confirmed that the minified files work without breaking the UI or functionality.
6. **Compression**:
    - Enabled Gzip compression on the server to reduce file transfer sizes.
    - Verified that compressed files load properly on various devices.
7. **Removed Unused Dependencies**:
    - Identified and removed unnecessary third-party libraries.
    - Verified that functionality was not affected after dependency cleanup.

Challenges Faced and Resolutions

1. **Challenge: Slow API Responses**
   - **Description**: API calls for retrieving product details were slow due to inefficient queries.
   - **Resolution**: Optimized database queries and added indexing for frequently accessed fields.
2. **Challenge: Large Bundle Sizes**
   - **Description**: The initial page load was slow due to large JavaScript bundles.
   - **Resolution**: Implemented code splitting and removed unused dependencies.
3. **Challenge: Poor Mobile Responsiveness**
   - **Description**: The layout was breaking on smaller screens.
   - **Resolution**: Updated CSS to use flexbox and grid systems. Tested responsiveness on various devices.

Challenge: Missing Environment Variables During Deployment

- **Description: The application failed after deployment because some required environment variables were missing in the hosting service (Netlify).**
- **Resolution: Identified missing variables, added them to Netlify's environment settings, and tested the deployment to ensure proper functionality.**

Challenge: CORS Errors When Fetching Data

- **Description**: Cross-Origin Resource Sharing (CORS) errors occurred during API calls, preventing data from being fetched.
- **Resolution**: Configured the API server to include appropriate CORS headers. Also, added the web application's domain to the server's allowed origins list.

Challenges Faced and Resolutions

->To build a car data fetching and display system, we would fetch car data from an API or a static source and render it on the `/cars` page. The page would include a search feature, allowing users to filter cars by make or model. Additionally, we can implement a price range slider to filter cars by price, while a category filter can display cars based on their type (SUV, sedan, etc.). On the `/category` route, users can view cars categorized by their type. To enhance the user experience, a recommendation system can suggest cars based on user preferences or viewing history. This system ensures users can easily search, filter, and explore car listings in a seamless manner. And further testing details are in Testing Report.csv.