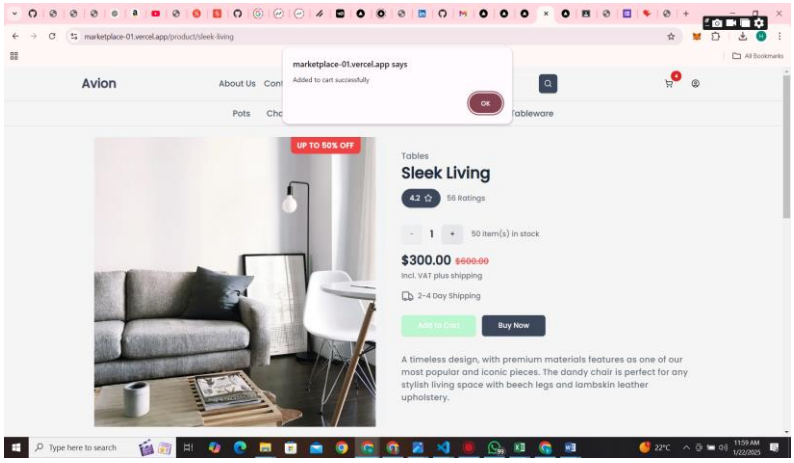
Project Testing and Performance Report

---

1. Functional Deliverables

Screenshots/Recordings:

- **Add to Cart Functionality**:
  - Screenshots showcasing the process of adding items to the cart.
  - Recording demonstrating the dynamic update of the cart and redirection to the cart page.



- **Responsive Design**:
  - Screenshots showing the layout on mobile, tablet, and desktop devices.
  - Recording highlighting responsive behavior during various device screen size changes.

Logs/Reports from Testing Tools:

- **Lighthouse Report**:
  - **Performance:** 89
  - **Accessibility:** 80
  - **Best Practices:** 96
  - **SEO:** 92

- **Postman Logs**:
  - Detailed request and response logs from API testing.

- Verifications for endpoints related to product listing, user authentication, and cart operations.

---

2. Documentation

Test Cases Executed and Their Results:

- **Add to Cart Functionality**: Verified that items are added correctly to the cart with accurate details.
- **Cart Quantity Update**: Ensured that updating the quantity reflects the correct total price.
- **Clear Cart Functionality**: Confirmed that clearing the cart removes all items.
- **Page Navigation**: Checked that the "Add to Cart" button redirects to the cart page correctly.
- **Responsive Design**: Validated that the application renders correctly on various screen sizes.
- **Checkout Process**: Verified that completing a purchase transitions to the confirmation page smoothly.
- **Search Bar**: Render product according to the query

Performance Optimization Steps Taken:

1. **Lazy Loading Images**:
   - Deferred loading of off-screen images.
   - Improved initial page load time and reduced bandwidth usage.
2. **Dynamic Imports**:
   - Used Next.js dynamic imports for code splitting.
   - Reduced initial load time by loading components only when needed.
3. **Caching and CDN Utilization**:
   - Applied caching headers and served static assets via CDN.
   - Enhanced content delivery speed and reduced server load.
4. **Database Query Optimization**:
   - Refined queries to fetch only necessary data.
   - Improved server response times and reduced database strain.

Security Measures Implemented:

1. **Input Validation**:
   - Server-side and client-side validation to prevent malicious inputs.
   - Protected against SQL injection, XSS, and other input-based attacks.
2. **HTTPS Enforcement**:
   - Enforced HTTPS for all communication.
   - Secured data transmission and ensured user data protection.
3. **Authentication and Authorization**:

    - Implemented JWT-based authentication with role-based access control.
    - Secured user sessions and restricted access to sensitive areas.
4. **Content Security Policy (CSP)**:
    - Applied a strict CSP to mitigate content injection attacks.
    - Reduced the risk of XSS and other injection-based attacks.

Challenges Faced and Resolutions Applied:

1. **Performance Bottlenecks**:
    - High load times due to large data sets.
    - Resolution: Implemented pagination and infinite scrolling.
    - Impact: Improved load times and user experience.
2. **Cart Context Issues**:
    - Errors with `useCart` when used outside `CartProvider`.
    - Resolution: Ensured all components using `useCart` were wrapped within `CartProvider`.
    - Impact: Resolved context availability issues and maintained consistent cart functionality.
3. **Navigation Errors**:
    - Improper use of `Router.push` causing navigation issues.
    - Resolution: Replaced with `useRouter` hook, ensuring proper client-side routing.
    - Impact: Resolved navigation issues, enhancing user experience.
4. **Security Vulnerabilities**:
    - Vulnerabilities in handling user inputs.
    - Resolution: Applied comprehensive validation and sanitization techniques.
    - Impact: Secured the application against common attacks like SQL injection and XSS.

End of Report