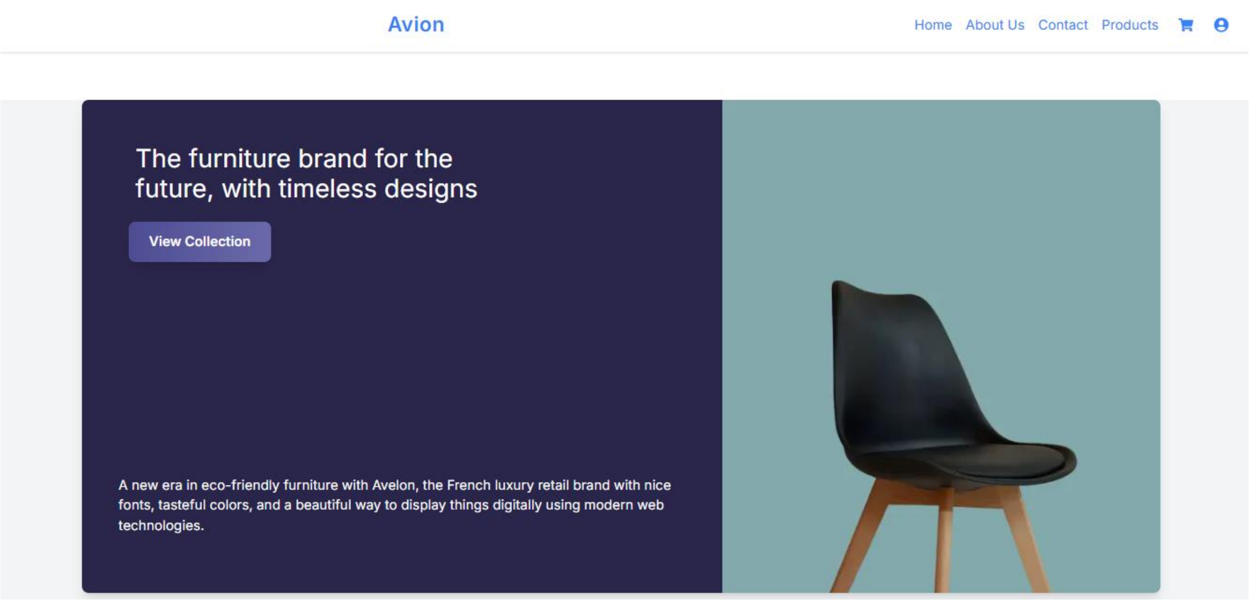
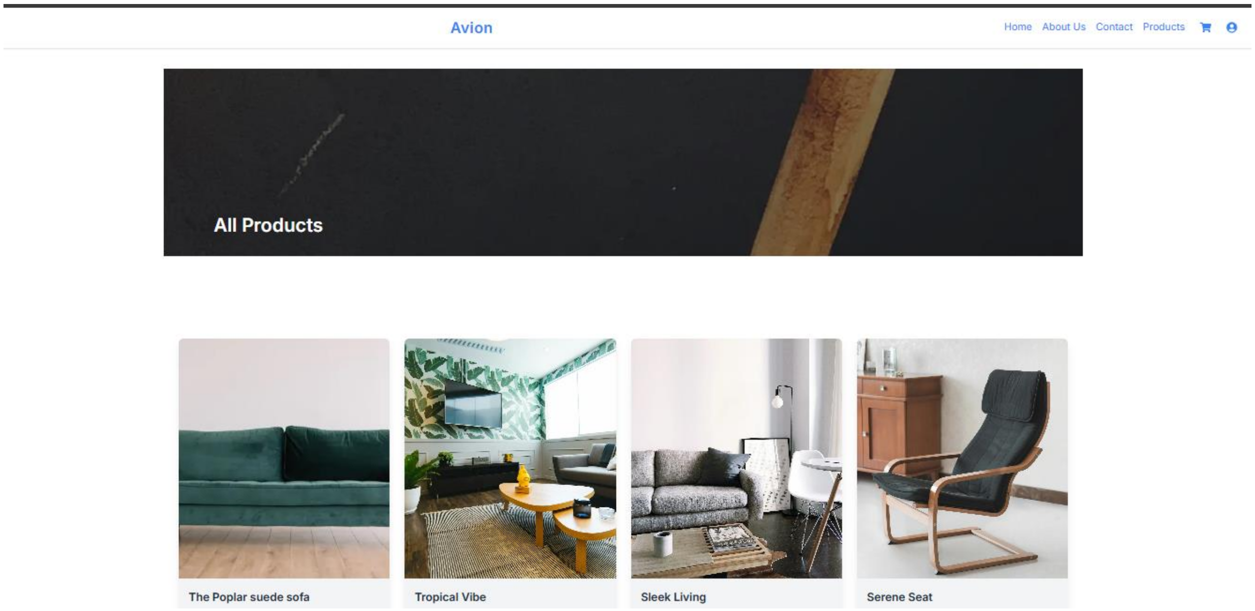
## 1. Functional Deliverables:

### Functional Image



The furniture brand for the future, with timeless designs

A new era in eco-friendly furniture with Avelon, the French luxury retail brand with nice fonts, tasteful colors, and a beautiful way to display things digitally using modern web technologies.



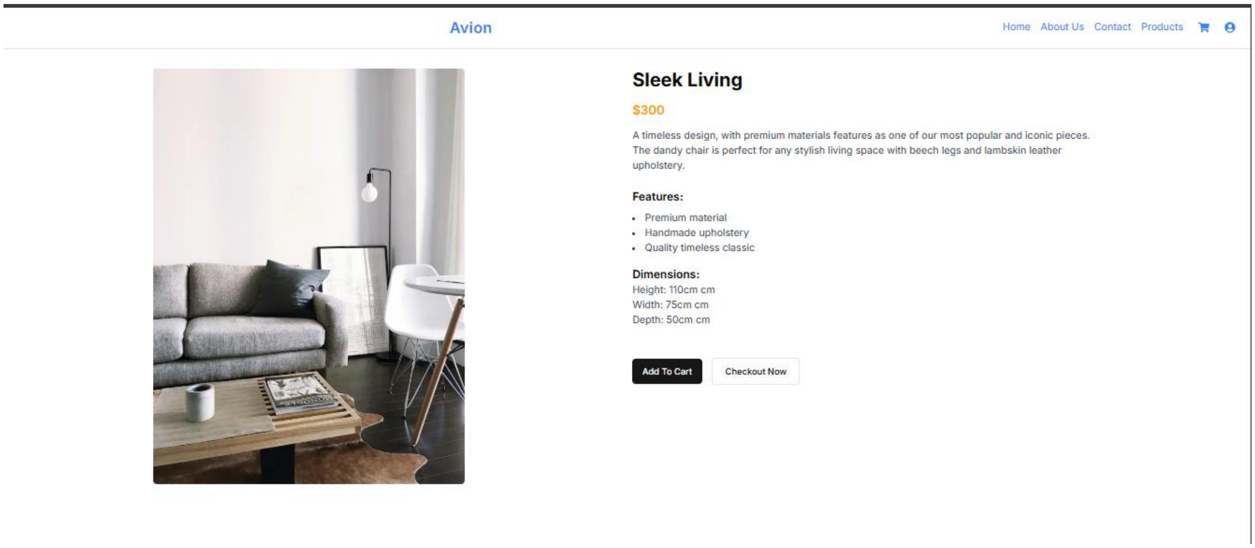
All Products

The Poplar suede sofa

Tropical Vibe

Sleek Living

Serene Seat



## Sleek Living

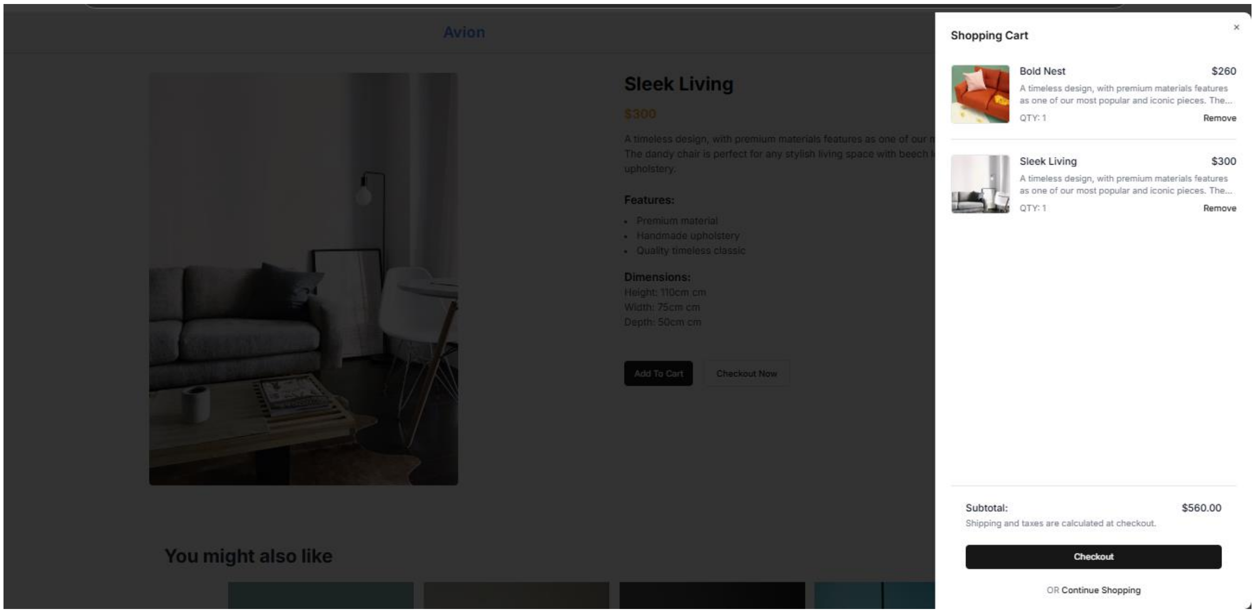
$300

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

**Features:**

- Premium material
- Handmade upholstery
- Quality timeless classic

**Dimensions:**
Height: 110cm cm
Width: 75cm cm
Depth: 50cm cm



Sleek Living

$300

A timeless design, with premium materials features as one of our m
The dandy chair is perfect for any stylish living space with beech l
upholstery.

Features:

- Premium material
- Handmade upholstery
- Quality timeless classic

Dimensions:
Height: 110cm cm
Width: 75cm cm
Depth: 50cm cm

You might also like

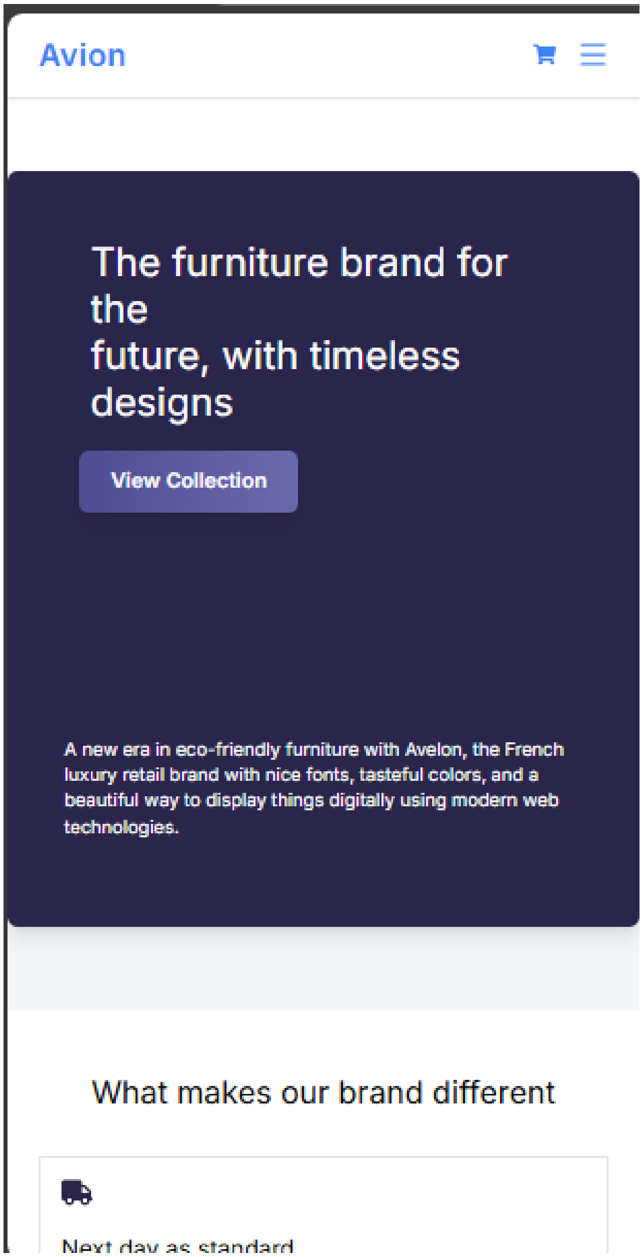
**Shopping Cart**

Bold Nest $260
A timeless design, with premium materials features as one of our most popular and iconic pieces. The...
QTY: 1 Remove

Sleek Living $300
A timeless design, with premium materials features as one of our most popular and iconic pieces. The...
QTY: 1 Remove

Subtotal: $560.00
Shipping and taxes are calculated at checkout.

## Responsiveness Image



The furniture brand for the
future, with timeless designs

A new era in eco-friendly furniture with Avelon, the French luxury retail brand with nice fonts, tasteful colors, and a beautiful way to display things digitally using modern web technologies.

What makes our brand different

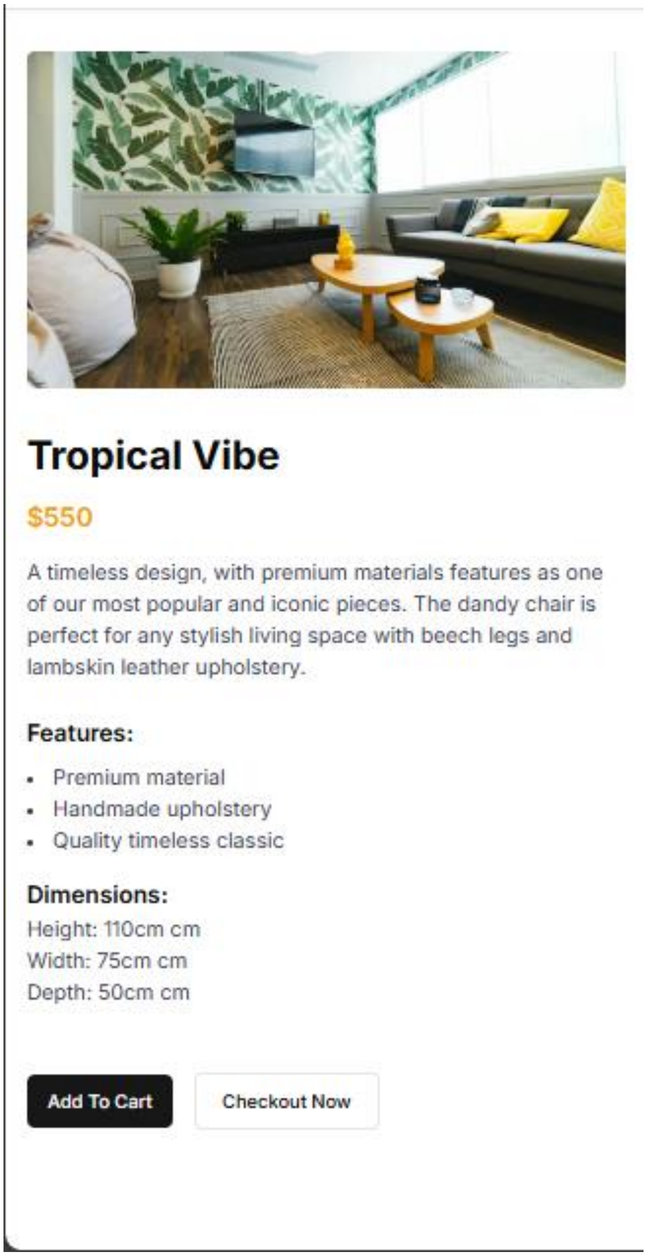
Next day as standard

### The Poplar suede sofa

$980

### Tropical Vibe

$550



# Tropical Vibe

$550

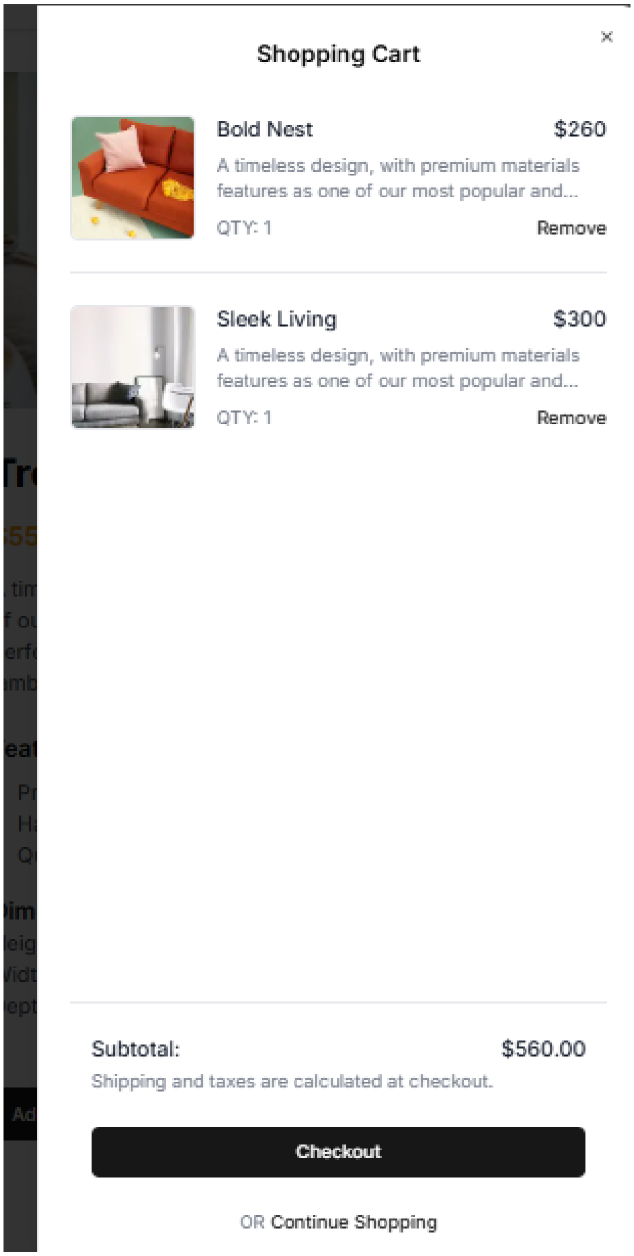
A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

**Features:**

- Premium material
- Handmade upholstery
- Quality timeless classic

**Dimensions:**

Height: 110cm cm
Width: 75cm cm
Depth: 50cm cm



Shopping Cart

Bold Nest $260

A timeless design, with premium materials features as one of our most popular and...

QTY: 1 Remove

Sleek Living $300

A timeless design, with premium materials features as one of our most popular and...

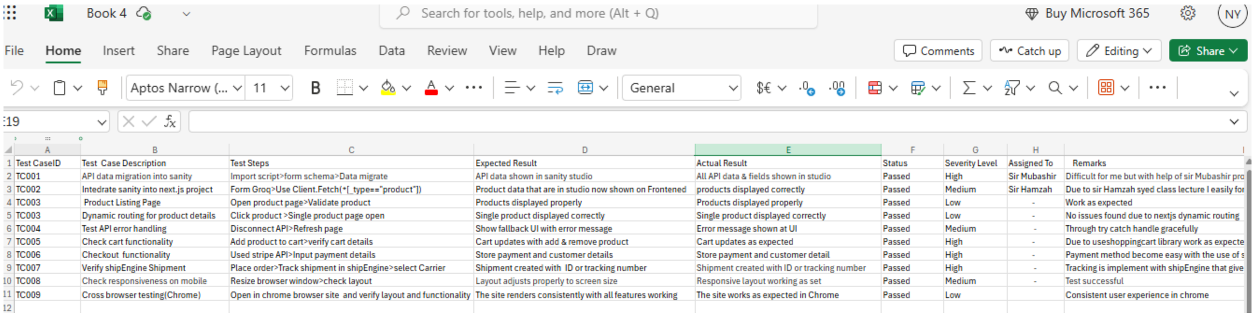
QTY: 1 Remove

Subtotal: $560.00

Shipping and taxes are calculated at checkout.

OR Continue Shopping

## 2.Testing Report :



| Test CaseID | Test  Case Description | Test Steps | Expected Result | Actual Result | Status | Severity Level | Assigned To | Remarks |
|---|---|---|---|---|---|---|---|---|
| TC001 | API data migration into sanity | Import script>form schema>Data migrate | API data shown in sanity studio | All API data & fields shown in studio | Passed | High | Sir Mubashir | Difficult for me but with help of sir Mubashir pro |
| TC002 | Intedrate sanity into next.js project | Form Groq>Use Client.Fetch(*[_type=="product"]) | Product data that are in studio now shown on Frontened | products displayed correctly | Passed | Medium | Sir Hamzah | Due to sir Hamzah syed class lecture I easily for |
| TC003 | Product Listing Page | Open product page>Validate product | Products displayed properly | Products displayed properly | Passed | Low | - | Work as expected |
| TC003 | Dynamic routing for product details | Click product >Single product page open | Single product displayed correctly | Single product displayed correctly | Passed | Low | - | No issues found due to nextjs dynamic routing |
| TC004 | Test API error handling | Disconnect API>Refresh page | Show fallback UI with error message | Error message shown at UI | Passed | Medium | - | Through try catch handle gracefully |
| TC005 | Check cart functionality | Add product to cart>verify cart details | Cart updates with add & remove product | Cart updates as expected | Passed | High | - | Due to useshoppingcart library work as expecte |
| TC006 | Checkout  functionality | Used stripe API>Input payment details | Store payment and customer details | Store payment and customer detail | Passed | High | - | Payment method become easy with the use of s |
| TC007 | Verify shipEngine Shipment | Place order>Track shipment in shipEngine>select Carrier | Shipment created with  ID or tracking number | Shipment created with ID or tracking number | Passed | High | - | Tracking is implement with shipEngine that give |
| TC008 | Check responsiveness on mobile | Resize browser window>check layout | Layout adjusts properly to screen size | Responsive layout working as set | Passed | Medium | - | Test successful |
| TC009 | Cross browser testing(Chrome) | Open in chrome browser site  and verify layout and functionality | The site renders consistently with all features working | The site works as expected in Chrome | Passed | Low |  | Consistent user experience in chrome |

## 3. Documentation:

## Test cases executed and their results:

### Test Summary: -

This report details the executed test cases and their results for various functionalities, primarily focusing on API integration, product pages, checkout, and responsiveness across devices and browsers. Each test case includes the description, steps followed, expected vs. actual results, status, severity level, assignee, and remarks.

Test Cases Executed

Test Execution Overview

- **Total Test Cases Executed**: 9
- **Total Passed**: 9
- **Total Failed**: 0
- **Overall Status**: All test cases passed successfully.

Test Summary and Key Remarks

1. **API Integration and Data Migration**:
   a. The integration of the API into the sanity studio was smooth and performed successfully after the script code was provided.

2. **Next.js Project Integration**:
   a. The integration of Sanity data into the Next.js frontend using Groq queries was achieved without issues, thanks to prior guidance from team members.

3. **Product Pages and Dynamic Routing**:
   a. Product listing and individual product detail pages rendered correctly, with dynamic routing functioning as expected.

4. **API Error Handling**:
   a. Error handling mechanisms were successfully tested, and fallback UI with error messages was properly displayed when the API connection was disrupted.

5. **Cart and Checkout Functionality**:
   a. Cart updates and checkout processes were fully functional, with successful data handling and payment processing through the Stripe API.

6. **ShipEngine Shipment Verification**:
   a. The tracking functionality for shipments via Ship Engine was successfully tested, including the creation of tracking IDs.

7. **Responsive Design**:
   a. The site's responsiveness across mobile and desktop platforms was validated, and the layout was consistent across varying screen sizes.

8. **Cross-Browser Testing**:
   a. The site’s functionality and layout were consistent in Chrome, confirming compatibility.

## Performance optimization steps taken:

- Use next js Image component (next/image) for lazy loading, resizing and optimizing images.
- The Link component unable client-side transition between pages, avoiding full page reload and speeding up navigation.
- We used in image Alt attribute for good SEO (search engine optimization) in website.
- Use Tailwind CSS for optimized styling.
- Use the next/Font module to pre load Font and reduced layout shifts.
- Use the next –script components to lazy load scripts.
- For better SEO only use one H1 tag in one page.
- **Use CDN**: Distribute content globally and cache at edge locations.

## Security measures implemented:

- **Use `.env` files** to store sensitive information like API keys, database credentials, and other environment-specific settings. Ensure these files are **never committed to version control** (e.g., add them to `.gitignore`).

- **Use environment variables** to securely store secrets and configurations instead of hardcoding them in code.

## Challenges faced and resolutions applied:

**Challenges:**

- **Asynchronous Operations**: Fetching data asynchronously from APIs can lead to issues with managing data flow, timing, and UI updates.
- **Data Fetching Delays**: Fetching data from external sources, especially large datasets, can cause delays and slow down the user experience.

- **Error Handling**: Handling failed API requests, timeouts, and unexpected responses from external APIs can be complex.
- **Data Caching**: Repeated data fetching from the same source can increase load times and cause unnecessary network requests.

### Resolutions Applied:

- **Use `async/await`**: Simplified handling of asynchronous operations by using the `async/await` syntax for fetching data, which provides cleaner, more readable code.
- **Loading States**: Implemented loading indicators or skeleton screens to show users that data is being fetched, improving user experience during delays.
- **Error Handling with Try-Catch**: Used `try-catch` blocks to handle API errors gracefully, displaying fallback content or error messages in case of failures.
- **Data Caching**: Implemented caching mechanisms such as **localStorage**, **Session Storage**, or **state management tools** like Redux or React Context to store fetched data temporarily and avoid redundant API calls.
- **Debouncing API Requests**: Implemented **debouncing** for search-related API requests to avoid triggering multiple unnecessary calls in rapid succession.
- **Optimized Data Fetching**: Used **pagination** or **lazy loading** techniques to load data in chunks, reducing initial load time and improving performance.