
Day 6: DEPLOYMENT PREPARATION AND STAGING ENVIRONMENT SETUP

Sit & Style Studio
Date: 3-2-25
Name: Syeda Hafiza Bibi Amna

Objective:

I'll be concentrating on optimising my marketplace for deployment today. This entails establishing hosting platforms, creating a staging environment, and making sure my program is completely optimised for users. Building on Day 5's testing and improvements, I will replicate a production-like setting to ensure seamless functioning. I will also look at industry best practices for managing different environments, such as production (UAT, PROD, DR) and non-production (TRN, DEV, SIT), to guarantee a smooth transition from development to live deployment.

HOSTING PLATFORM SETUP

- **Speed & User-Friendliness**: Vercel is preferred for its fast deployment and easy-to-use interface.
- **Quick & Easy Deployment**: Allows one-click deployments with integrations to Bitbucket, GitLab, and GitHub.
- **Frontend Framework Optimisation**: Optimized for modern frameworks like Next.js, React, and Vue.js.
- **Global Edge Network**: Ensures fast loading times by distributing content globally.
- **Automatic Scaling**: Automatically adjusts app scaling based on traffic demands, reducing manual server management.
- **Free SSL & Custom Domains**: Provides free HTTPS for enhanced security and easy domain configuration.

GITHUB REPOSITORY TO VERCEL

- **Sign in to Vercel**: Log in to Vercel using your GitHub account.
- **Import Repository**: Click on "New Project" and select the GitHub repository you want to deploy.
- **Configure Build Settings**: Vercel auto-detects popular frameworks like Next.js, React, etc.
- Optionally, specify a custom build command (e.g., npm run build) and output directory if needed.
- **Set Environment Variables**: Add any required environment variables (like API keys, database URLs) in Project Settings > Environment Variables.

- **Deploy the Project**: Click "Deploy" to start the build process and make your project live.

CONFIGURE ENVIRONMENT VARIABLES

Create a .env File:

```
.env
.env

NEXT_PUBLIC_SANITY_PROJECT_ID="rhnptdy2" # Add your project Id
NEXT_PUBLIC_SANITY_DATASET="production"
NEXT_PUBLIC_SANITY_AUTH_TOKEN="sktENKws4o31d4icpCEc1am2BZulH6VZ4tUm8rD7TOiCg9PA0B9rsQYBZvPlNgKa4T8UeHYAYNixtaWB2Q3IbyTRaz6
NEXT_PUBLIC_STRIPE_KEY="pk_test_51QhfvRBL1HZNe3TCbsfxXCEhpqty6zICV8LoNiQGfyvZ8ZmWVgK81wXDu9NGJV8UUUqAc8tk5ayYRvi5DTX5yBjf0



NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_ZnVua3ktd2FsbGFieS04OS5jbGVyay5hY2NvdW50cy5kZXYk
CLERK_SECRET_KEY=sk_test_txZktalhTUzqBOmpZULfZcNB1Jo1qZqL2OESnSjDkq
```

UPLOAD ENVIRONMENT VARIABLES TO VERCEL:

Go to Vercel Dashboard: Access your project's dashboard on Vercel.

Navigate to Settings: Go to the Settings section of your project.

Add Environment Variables: Under the Environment Variables section, add each variable from your .env file.

Set the Environment: Choose the appropriate environment—either "Production" or "Preview"—based on your needs.

Click Save: Save the variables to ensure they are securely added to your deployment.

DEPLOY APPLICATION

Log in to Vercel: Sign in to your Vercel account**.**

Navigate to the Project Dashboard: Go to the dashboard of the project you want to deploy.

Connect GitHub Repository (if not already connected): Import your GitHub repository to Vercel if you haven't done so yet.

Select "Deploy": Click the "Deploy" button. Vercel will automatically detect the framework and start the deployment process.

Staging Environment Deployment: Vercel will deploy the application to a staging environment, providing a unique URL (e.g., yourproject-name.vercel.app).

Environment Variables and Configuration: Vercel will use the .env variables and configuration settings to ensure everything works in the staging environment.

VALIDATE DEPLOYMENT

Ensure the Build Completes Successfully:

- Check the Vercel dashboard for any errors during the build process.
- If errors appear, review the logs to identify the issue and fix it.

Verify Basic Functionality:

- Open the staging URL provided by Vercel.
- Perform tests to ensure the application functions as expected. This includes:
    - Checking page load times.
    - Verifying interactive features (forms, buttons, links).
    - Ensuring no broken elements or missing assets.

Testing Types :

1-User Authentication and Account Management validation:
Login/Logout:

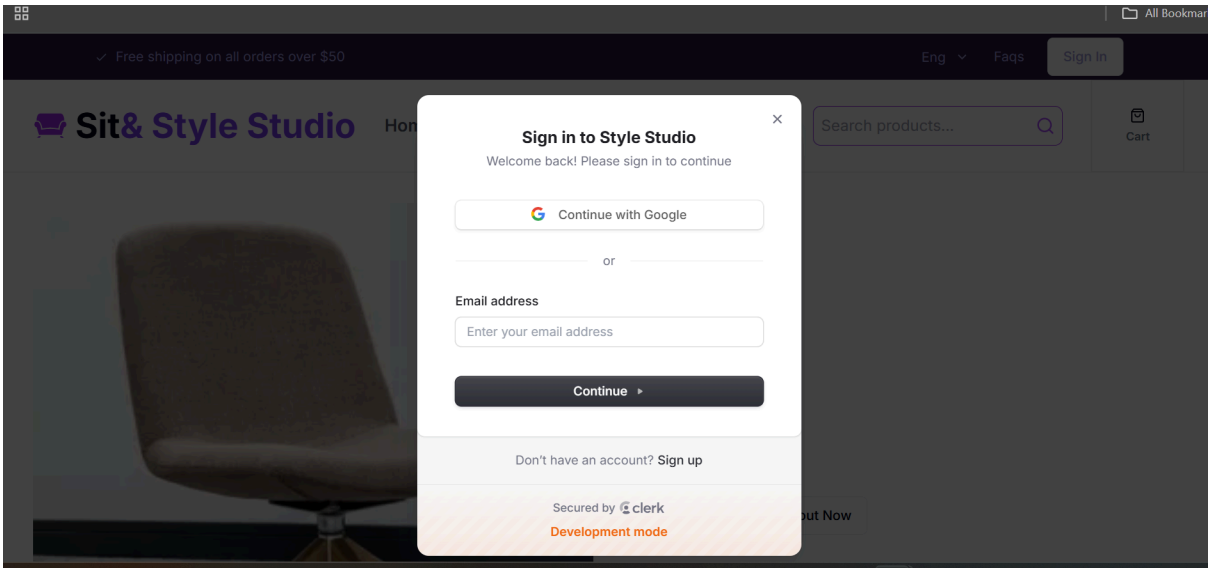
- Test login and logout functionality for registered users.
- Ensure correct user data (e.g., username, order history) is displayed after login.

Sign-Up and Password Reset:

- Verify the sign-up process, ensuring all fields are properly validated (email format, password strength).
- Test password reset functionality to confirm users can successfully reset their passwords.

Profile Update:

- Ensure users can update profile details, including email, shipping address, and payment methods.



2-Secure Payment Method Using Stripe:

Integration with Stripe:

- Ensure Stripe is properly integrated with the application.
- Verify API keys (Test & Live) are correctly configured in environment variables.

Payment Processing:

- Test successful payments using test card details provided by Stripe.
- Verify transaction details are recorded correctly in the database.

Error Handling:

- Ensure proper handling of failed transactions (e.g., insufficient funds, expired cards).
- Display user-friendly error messages for payment failures.

Secure Payment Flow:

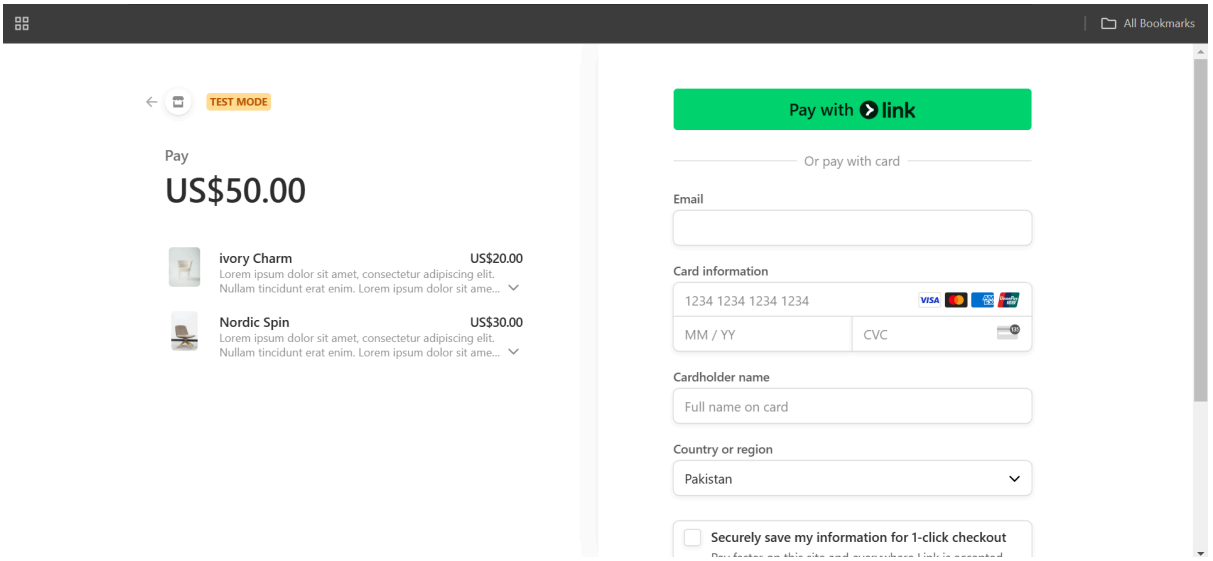
- Ensure card details are handled securely and never stored on the server.
- Use Stripe's Checkout or Payment Intent API for PCI-compliant transactions.

Order Confirmation:

- Verify that a confirmation email is sent after a successful payment.
- Ensure order details update correctly in the user's account.

Refund & Dispute Management:

- Test refund functionality and verify refunded transactions in Stripe's dashboard.
- Ensure proper handling of chargebacks and disputes.



3-Cart Operations validation

Add to Cart:

- Test adding various products to the cart.
- Ensure the correct product, quantity, and price are displayed.

Update Cart:

- Test updating the quantity of products in the cart.
- Verify that the total price updates correctly when quantities are changed.

Remove Items:

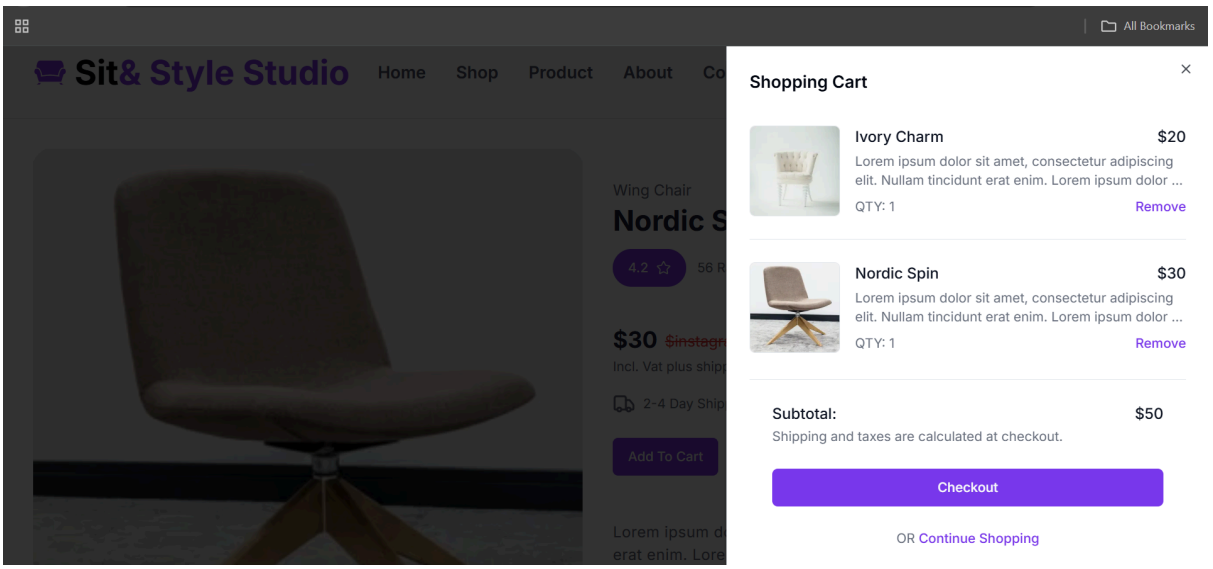
- Ensure users can remove items from the cart.
- Verify the cart updates correctly, showing the remaining items and updated totals.

Persist Cart:

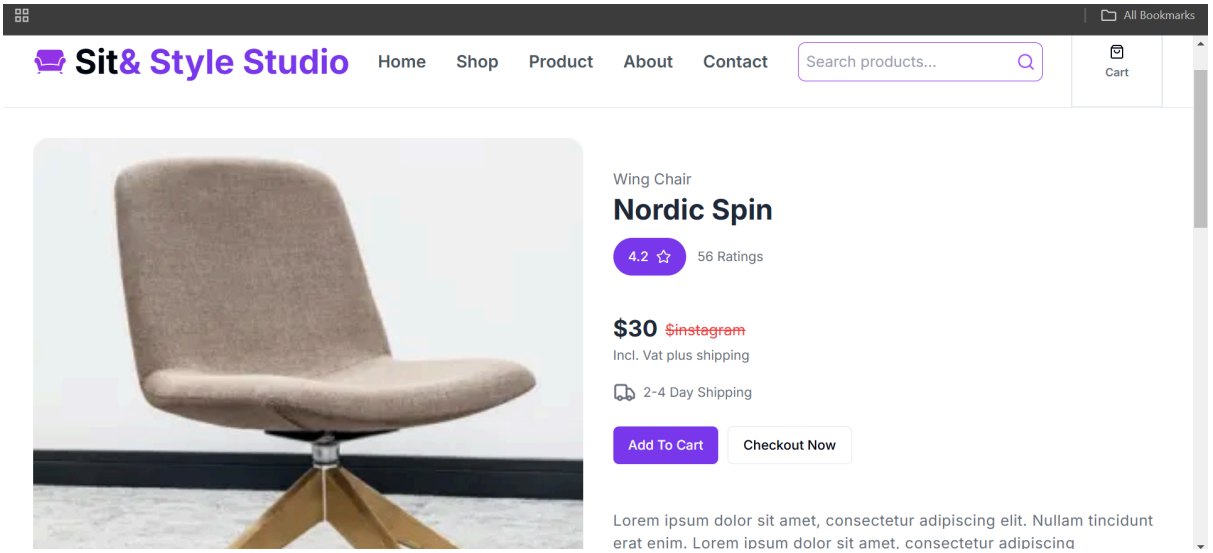
- Check that the cart persists across sessions for logged-in users.
- Ensure the cart remains intact even if the user logs out and logs back in.

Cart Visibility:

- Ensure the cart is easily accessible.
- Verify that the correct number of items is shown in the cart at all times.



Product Details Page



Search And Filter Functionality

Basic Search:

- Test the search bar with various queries (e.g., product name).
- Ensure it returns relevant and accurate results.

Advanced Search Filters:

- Test advanced search filters (e.g., , categories, Tags).
- Verify that the filters correctly refine search results as expected.

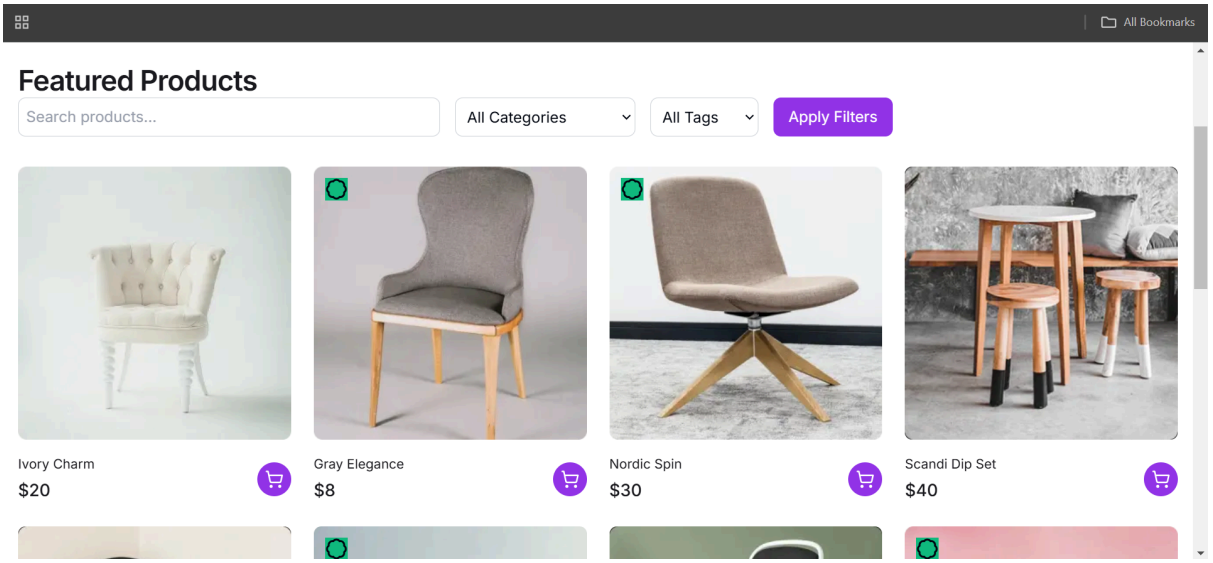
Empty Results Handling:

- Check that the system gracefully handles no results.

- Ensure it displays a relevant message (e.g., “No products found”).

Edge Cases:

- Test edge cases like very long search terms, typos, or empty searches.
- Ensure the system provides meaningful feedback, such as suggesting corrections or showing relevant results for partial matches.



Test Case Reporting

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
1	TC001	product listing page	Open product page > Verify products	Products displayed correctly	Products displayed correctly	Passed	High	No issues found
2	TC002	Test API error handling	Disconnect API > Refresh page	Show fallback UI with error message	Error message shown	Passed	Medium	Handled gracefully
3	TC003	Check cart functionality	Add product to cart > Verify cart contents	Cart updates with added product	Cart updates as expected	Passed	High	No issues found
4	TC004	Ensure responsiveness on mobile	Resize browser window > Check layout	Layout adjusts properly to screen size	Responsive layout working as intended	Passed	Medium	Test successful
5	TC005	Dynamic Routing	Very Dynamic Routing	Correct Product Details	Dynamic Routing Works	Passed	High	No issues found
6	TC006	performance optimization	optimize performance	performance is improved	improved load time	passed	medium	performance optimize
7	TC007	cross browser testing	testing on different device	website functions on all devices	compatible on all devices	passed	medium	responsiveness

Performance Testing

Page Load Speed:

- Use tools like Lighthouse or GTmetrix to measure the speed of the website.
- Analyze performance scores to ensure they meet acceptable thresholds (e.g., 90+ for performance)**.**

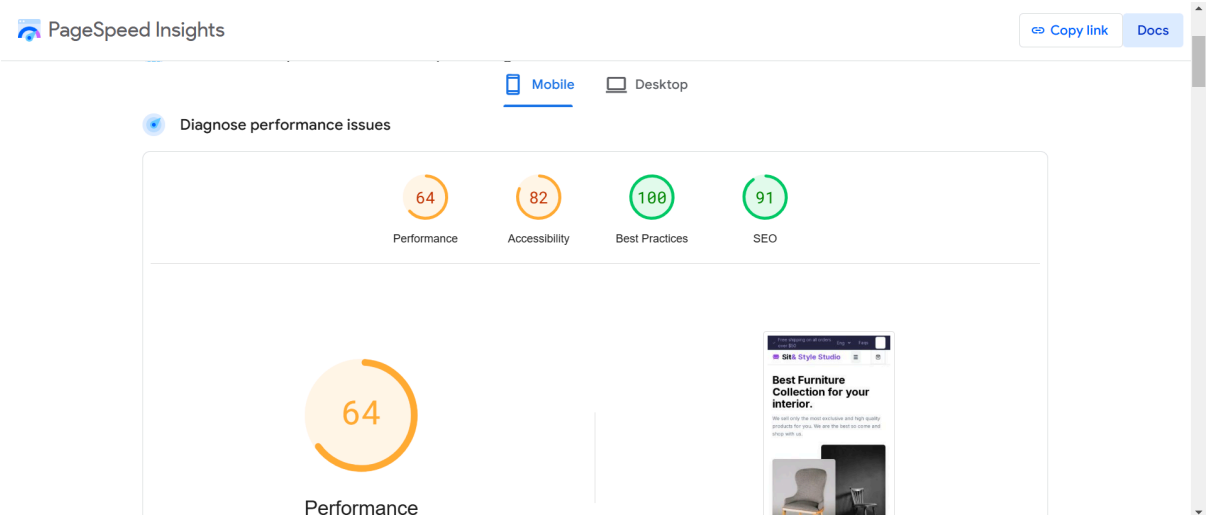
Responsiveness:

- Test the application on various screen sizes (desktop, tablet, mobile).
- Ensure the app is responsive and provides optimal performance across all devices.

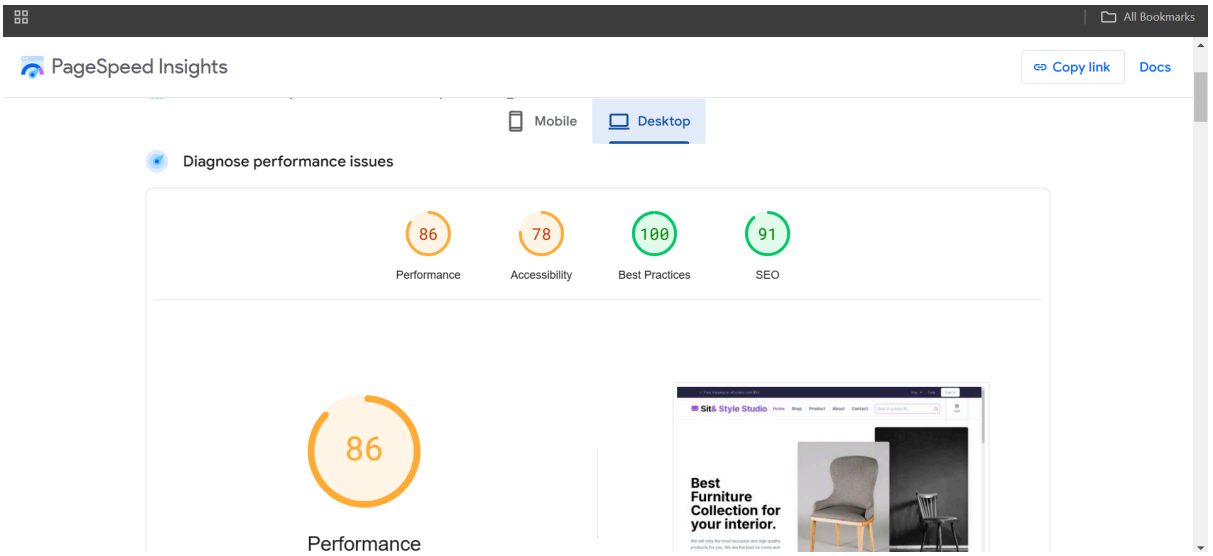
Time to Interactive (TTI):

- Ensure the application loads quickly and becomes interactive without significant delay.
- Test the TTI to confirm that users can start interacting with the site promptly after loading.

Performance Testing on Mobile :



Performance Testing on Mobile :



Checklist for Day 6 :

Staging Environment Testing:	✔
Deployment Preparation	✔
Documentation	✔

Form Submission:	✔
Final Review:	✔