

DEPLOYMENT PREPARATION AND STAGING ENVIRONMENT SETUP - Day 6

Prepared by: Fatima Faisal

Objective: Preparing the marketplace for deployment by setting up a staging environment, configuring hosting platforms, and ensuring readiness for a customer-facing application. This stage emphasizes ensuring the marketplace operates seamlessly in a production-like environment

Steps for Implementation:

Step 1: Hosting Platform Setup

1. Hosting Platform:

- Using **Vercel** for quick and seamless deployment.

2. Repository Connection:

- Link GitHub account to Vercel.
- Access vercel dashboard and navigate to **Add New... > Project** section and import the specific repository to link to vercel

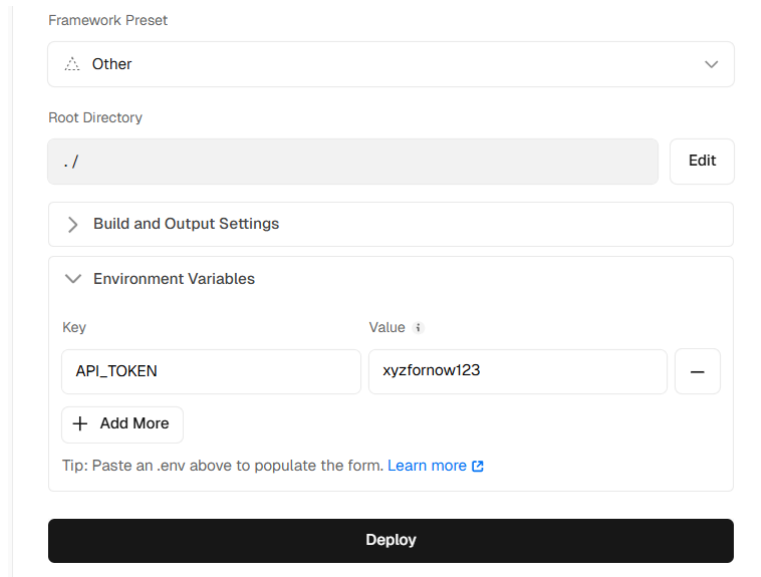
Step 2: Configure Environment Variables

1. .env File:

Sensitive data like API KEYS and TOKENS stored in **.env.local** file which is not pushed on github for security purpose.

2. Uploading Variables to Hosting Platform:

Add API KEYS and TOKENS in Environment Variables accurately.



The screenshot shows a configuration interface for a hosting platform. At the top, there is a 'Framework Preset' dropdown menu set to 'Other'. Below it is the 'Root Directory' field, which contains './' and has an 'Edit' button. A section titled 'Build and Output Settings' is collapsed. The 'Environment Variables' section is expanded, showing a table with two columns: 'Key' and 'Value'. One variable is listed: 'API_TOKEN' with the value 'xyzformow123'. There is a minus button to the right of the value. Below the table is a '+ Add More' button. At the bottom of the environment variables section, there is a tip: 'Tip: Paste an .env above to populate the form. [Learn more](#)'. At the very bottom of the interface is a large black 'Deploy' button.

Key	Value
API_TOKEN	xyzformow123

Step 3: Deployment

Deploy the project and view the build process.

- **Validation:**

Ensuring the build process completes without errors.

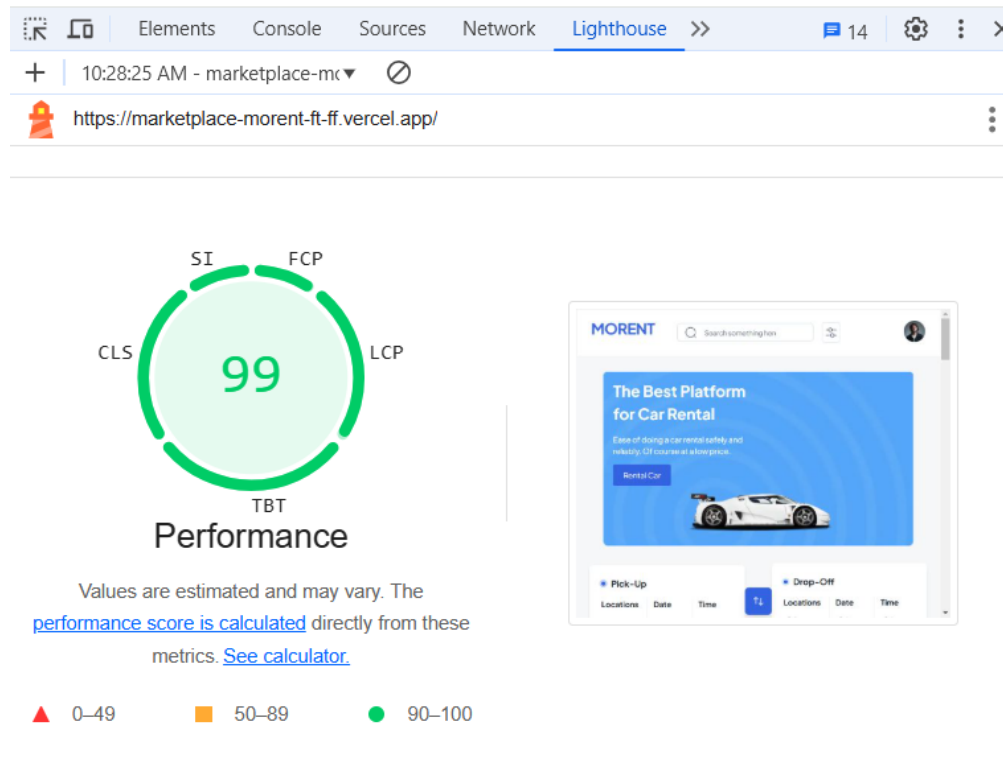
Step 4: Staging Environment Testing

1. Functionality Testing:

All features, including product listing, search functionality, and wish list operations, are working as they did on localhost.

2. Performance Testing:

Performance test with Lighthouse:



3. Security Testing:

API communications are secure. Sensitive data like API KEYS stored in secure environment variables.

User Acceptance Test (UAT)

Simulated Real-World Usage:

- Performed tasks like browsing cars, adding them to the wish list.
- No usability issues found.

Test Cases:

Test Case ID	Feature/Module	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Remarks
TC001	Product Listing	Verify all products display correctly.	Navigated to the pages having product listings.	All products are displayed correctly.	Got the expected results	Passed	No issues found.
TC002	Search Functionality	Verify the search feature returns relevant results.	1. Entered "Audi" in the search bar.	Only Audi displayed in the results.	Got the expected results	Passed	Works as expected
TC003	Filter Functionality	Verify filters work as intended.	Applied category filters	Products are sorted by category.	Cars appeared with selected categories	Passed	Works as expected
TC004	Wishlist Operations	Verify adding cars to the wish list.	1. Added a car to the wish list. 2. Verified it appears in the wishlist.	The product appears in the wishlist.	Got the expected results	Passed	No issues found
TC005	Dynamic Routing	Verify product detail pages load correctly.	1. Clicked on a car. 2. Verified the detail page loads with relative information	Product detail page loads with correct details.	Got the expected results	Passed	Rendered successfully
TC006	API error handling	Verify error message on API failure.	1. Disconnect API 2. Refresh page	Show fallback message	Fallback message shown	Passed	Handled gracefully

TC007	Performance Optimization	Verify page load time is under 2 seconds.	1. Run a Lighthouse audit. 2. Measure load time and performance score.	Load time is was around 1s – 1.5s	Got the expected results	Passed	Page loads under required time.
TC008	Responsive Design	Verify the site is responsive on different devices.	1. Resize window layout (For different screen sizes)	The layout adjusts correctly on all.	Got the expected results	Passed	Responsiveness verified.
TC009	Cross-Browser Testing	Verify functionality across major browsers.	1. Test on Chrome and Edge.	Site works consistently on all browsers.	Got the expected results	Passed	Worked well.