**Testing Performed in this project:**

---

## Smoke Testing

In this project, when the initial build is delivered and its stability is uncertain, we perform Smoke Testing. Smoke Testing helps us verify whether the basic and core features of the application are working or not.
If the smoke test fails, we immediately reject the build and send it back to the development team, as no further testing can be continued on an unstable build.
This ensures that the build is testable and safe for executing detailed test cases.

---

## Sanity Testing

After receiving a stable and functional build, we perform Sanity Testing.
Sanity Testing involves verifying specific features or bug fixes that were recently implemented.
The main purpose is to ensure that the fixed areas are working as expected without affecting other core functionalities.
This testing helps us quickly validate whether we can proceed with detailed functional or regression testing.

---

## Retesting

Retesting is performed to confirm whether previously reported defects have been fixed successfully.
Whenever a new build is released with resolutions, we execute failed test cases again to validate the fix.
If the defect still exists, it is reopened and assigned back to the development team.
Retesting ensures that no unresolved defects move forward to the next stage of testing.

---

## Regression Testing

Regression Testing is performed whenever a new build is released or when any functionality is modified, enhanced, or fixed.
The objective is to verify that the new changes have not adversely affected existing stable functionalities.
In this project, we execute regression test suites covering critical modules such as login, product selection, add to cart, checkout, and payment flow.
Regression Testing ensures overall application stability before Test Sign-Off.

---

Database Testing:

In this project, I performed Database Testing using MySQL Workbench 8.0 CE to verify that all user-related and transactional operations are stored correctly in the database. Whenever a user performs actions such as registration, login, logout, add to cart, or add to wishlist, corresponding records are inserted, updated, or deleted in the backend database tables.

The goal of database testing is to ensure that the UI and backend data remain synchronized and that data integrity, data flow, and constraints are properly maintained.

Below is a clear professional explanation that you can confidently speak in an interview. This explanation covers every functional area and aligns with the list you provided:

---

## Detailed Explanation of Functional Testing Performed in OpenCart E-Commerce Application

In this project, I performed end-to-end functional testing on the OpenCart E-Commerce application covering major user flows and transactional behavior. The objective was to validate that all modules behave correctly as per the business requirements and user expectations.

### ◆ User Account-Related Functionalities

I verified functionalities such as:

- User Registration
- Login
- Logout
- Forgot Password

Here, the main focus was to check successful account creation, valid/invalid login scenarios, password reset through email validation, and proper session handling after logout.

Additionally, features under **My Account** section such as:

- Account Information update
- Address Book modifications
- Changing Password

were validated to ensure user data is updated correctly and changes reflect at UI and database level.

---

## ◆ Product & Catalog-Related Features

I validated product search using different search criteria, product comparison feature, and verifying whether product listing and filtering displayed correct products for respective categories.

The **Product Display Page (PDP)** was tested for:

- Price update based on product options
- Stock availability
- Product description and specifications
- Image display and zoom functionality

This ensured product information was correct and consistent.

---

## ◆ Cart & Wishlist Functionalities

I tested:

- Add to Cart
- Shopping Cart
- Wishlist

Here, I focused on quantities, product variants, removal of items, price calculations, tax values, total billing amount, and UI message validations.

---

## ◆ Checkout & Order Management

The checkout process was tested with:

- Guest checkout
- Registered user checkout
- Different payment modes (if available)

The complete order flow was validated from placing an order to order confirmation page.

Under **My Orders**, I tested:

- Order History
- Order Information
- Product Return
- Downloads
- Reward Points
- Returned Requests
- Recurring Payments

- Your Transactions

This helped verify that once order is placed, backend order tracking functionality works correctly via order IDs, date, amount, and payment status.

---

## ◆ Promotional Modules

I validated:

- Gift Certificates page
- Special Offers page
- Affiliate module
- Newsletter subscription

Here, testing involved:

- Applying availability conditions
- Valid coupon code scenarios
- Subscription/unsubscribe behaviour

---

## ◆ UI-Level Functional Testing

I validated:

- Header options (Login, Cart, Categories etc.)
- Menu navigation
- Footer links like About Us, Privacy Policy, Terms & Conditions

The goal was to ensure navigation is smooth and no page breaks occur.

---

## ◆ Currency-Based Application Behavior

As OpenCart supports multiple currencies, I tested:

- Price conversion based on currency change
- Correct tax calculations
- Currency format validations

Example: USD to INR conversion changes occurred correctly across product pages, cart, and checkout page.

---

**I:**

“Overall, I performed end-to-end functional testing covering account creation, product search, cart management, checkout, promotions, user account maintenance, and currency-based behavior. Testing ensured that data flow remains consistent from UI to database and all modules behave as expected under valid and invalid scenarios. After validating all modules and ensuring stability of features, I was involved in Test Phase Sign-Off activities.”

---