**Testing a Food Ordering System**

**Test Scenarios and Test Cases**

**1. Adding/Removing Food Items to/from the Cart**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC\_001 | Add a single food item to the cart. | 1. Open the app.2. Browse the food menu.3. Select a single item. | Food Item: "Pizza" | The item should appear in the cart. |  |
| TC\_002 | Add multiple food items to the cart. | 1. Add multiple food items to the cart. | Food Items: "Pizza, Burger" | Both items appear in the cart. |  |
| TC\_003 | Remove an item from the cart. | 1. Add an item to the cart.2. Remove the item. | Item: "Pizza" | The cart should be empty. |  |
| TC\_004 | Remove a single item from multiple items in the cart. | 1. Add 2 items.2. Remove 1 item. | Items: "Pizza, Burger" | Only 1 item should remain. |  |
| TC\_005 | Add duplicate items to the cart. | 1. Add the same item twice. | Item: "Pizza" | Item count increases appropriately. |  |

**2. Calculating the Total Price**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC\_006 | Total price for single item. | 1. Add a single food item to the cart. | Item Price: ₹100 | Total Price: ₹100 |  |
| TC\_007 | Total price for multiple items. | 1. Add 2 items with different prices. | Pizza (₹100), Burger (₹50) | Total Price: ₹150 |  |
| TC\_008 | Total price after removing an item. | 1. Add 2 items.2. Remove 1 item. | Pizza (₹100), Burger (₹50) | Total Price updates correctly. |  |
| TC\_009 | Total price for duplicate items. | 1. Add the same item twice. | Pizza (₹100 x 2) | Total Price: ₹200 |  |
| TC\_010 | Total price with empty cart. | 1. Remove all items from the cart. | No items | Total Price: ₹0 |  |

**3. Empty Cart Behavior**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC\_011 | Cart displays empty state. | 1. Open the app.2. Remove all items from the cart. | No items | “Your Cart is Empty” message. |  |
| TC\_012 | Cart navigation when empty. | 1. Open cart with no items.2. Check for navigation options. | N/A | Prompt user to browse menu. |  |

**4. Checkout Process (Payment Validation)**

**Positive Test Cases**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC\_013 | Successful checkout with valid payment details. | 1. Add items to cart.2. Proceed to checkout.3. Enter valid details. | Card: 1234 5678 1234 5678, CVV: 123 | Payment successful message. |  |
| TC\_014 | Successful checkout using cash on delivery (COD). | 1. Select COD option. | Payment Option: COD | Order confirmation message. |  |

**Negative Test Cases**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC\_015 | Checkout with invalid card details. | 1. Enter invalid card number or CVV. | Card: 1111 1111 1111 1111, CVV: 000 | Payment failure message. |  |
| TC\_016 | Payment time-out behavior. | 1. Simulate a payment failure (e.g., slow response). | N/A | “Payment timed out” message. |  |

**5. Usability Testing**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_017 | Navigation to food menu is smooth. | 1. Open the app.2. Navigate between categories. | Navigation works without lag. |  |
| TC\_018 | Responsive UI on mobile/tablet devices. | 1. Test the app on mobile/tablet screen sizes. | Layout and buttons render well. |  |
| TC\_019 | All buttons and links are clickable. | 1. Interact with “Add,” “Remove,” and “Checkout” buttons. | Buttons respond correctly. |  |
| TC\_020 | UI consistency in fonts, colors, and alignment. | 1. Check fonts, colors, and alignment across screens. | UI appears professional. |  |

**Bug Report Template**

| **Bug ID** | **Description** | **Steps to Reproduce** | **Expected Result** | **Actual Result** | **Severity** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| BUG\_001 | Total price not updating. | 1. Add items.2. Remove items. | Total price should update. | Total price remains same. | High | Open |
| BUG\_002 | Payment failure on valid card. | 1. Enter valid card details. | Payment successful. | Payment failure message. | Critical | Open |

**Deliverables**

1. **Test Case Document** – Detailed test cases for all functionalities.
2. **Bug Report** – List of identified defects, severity, and status.

This structure ensures thorough **functional testing** and **usability testing** of the food ordering system, while the Bug Report helps track and resolve any identified issues.