|  |
| --- |
| ECONOMICAL MOBIWISE INSIGHT SYSTEM (EMWIS) |
| Software Test Case Generation - Report |

## White Box Testing

White Box Testing is a method where the tester has complete knowledge of the internal logic, structure, and source code of the application.

### Flow Path Testing

**Flowchart:**

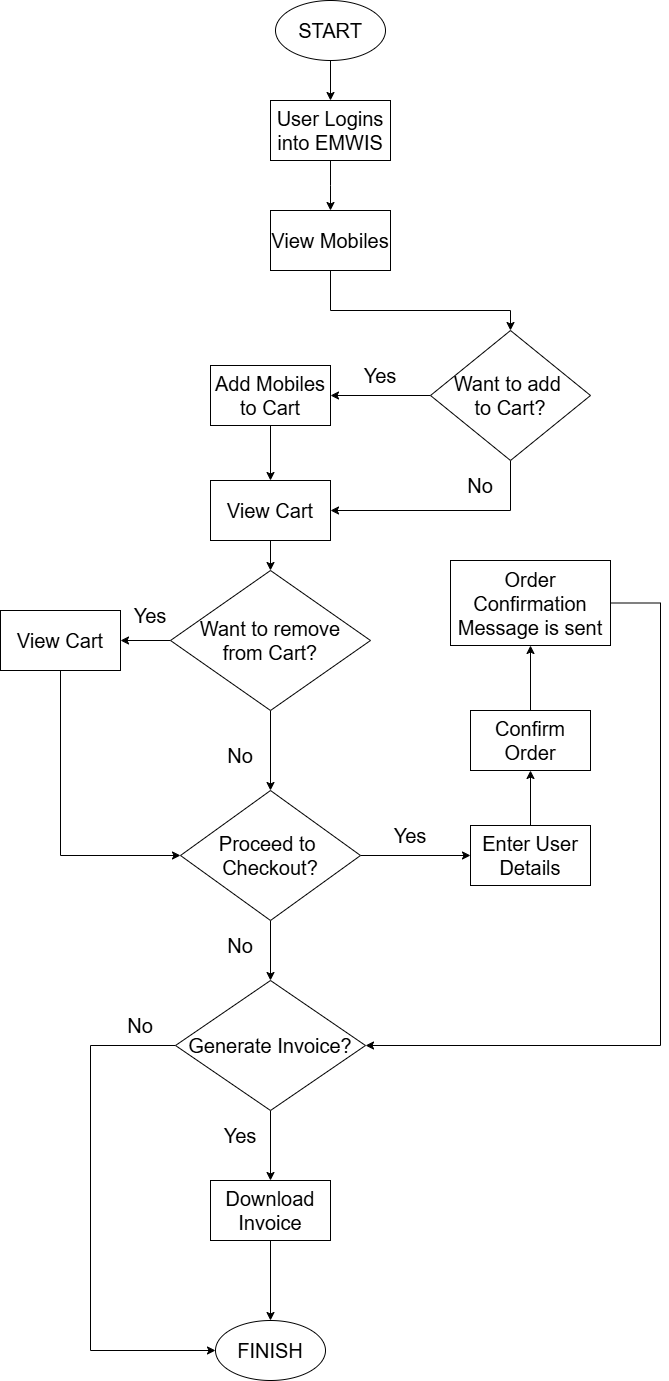
****

Fig 3.1: Flowchart – Cart and Order Management Activity

**Flowgraph:**

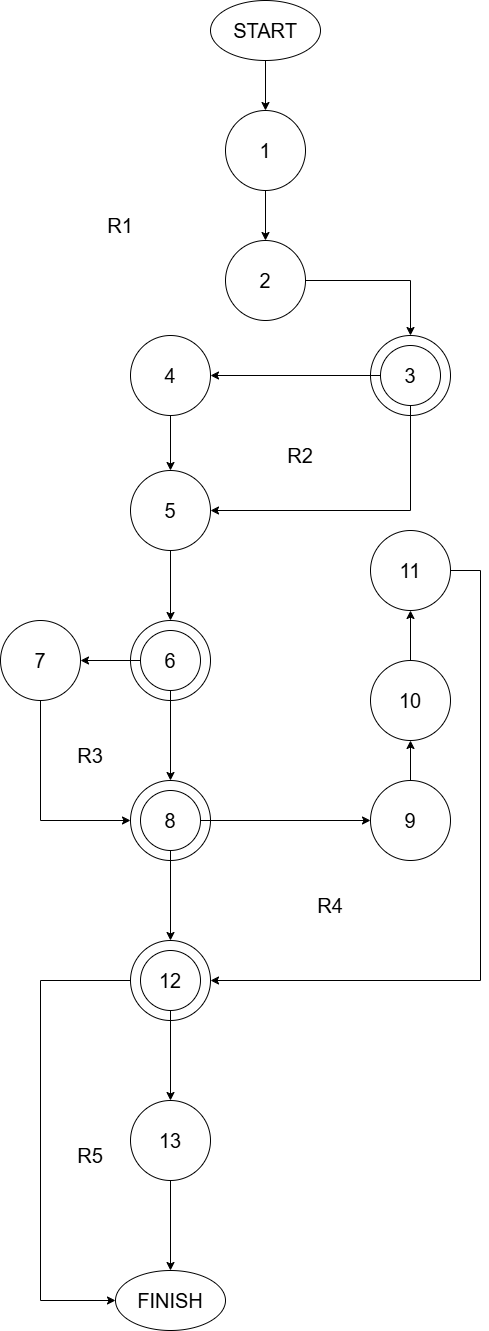
****

Fig 3.2: Flowgraph – Cart and Order Management Activity

**Cyclomatic Complexity for the Flowgraph:**

* Method 1:

Number of regions in flowgraph = **5**

* Method 2:

V(G) = E - N + 2

where,

N = Number of nodes of the control flow graph

E = Number of edges in the control flow graph

E = 16 and N = 13

Therefore, the value of the Cyclomatic complexity = 16 - 13 + 2

= **5**

* Method 3:

V(G) = Predicate nodes + 1 = 4 + 1

= **5**

### Activity Chart

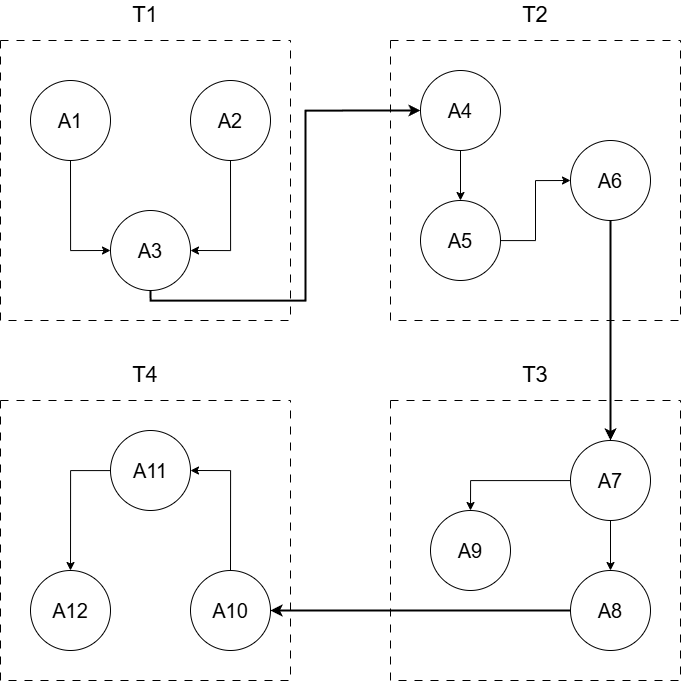


Fig 3.3: Activity Chart – Cart and Order Management Module

|  |  |
| --- | --- |
| A1 - Add, update, and remove mobiles in the cart | A7 - Update and display live order status notifications |
| A2 - Manage cart item quantities and preferences | A8 - Send order confirmation, shipping, and delivery updates |
| A3 - Store cart session persistently for logged-in users | A9 - Handle order delays or issues with user notifications |
| A4 - Capture address details for billing and shipping | A10 - Maintain a record of all past mobile orders |
| A5 - Process order confirmation and payment gateway | A11 - Enable invoice generation for completed orders |
| A6 - Validate user information before placing the order | A12 - Allow users to reorder from past purchases easily |
|  |  |
| T1 - Cart Management | T3 - Track Order |
| T2 - Checkout | T4 - Order History |

### Task Chart

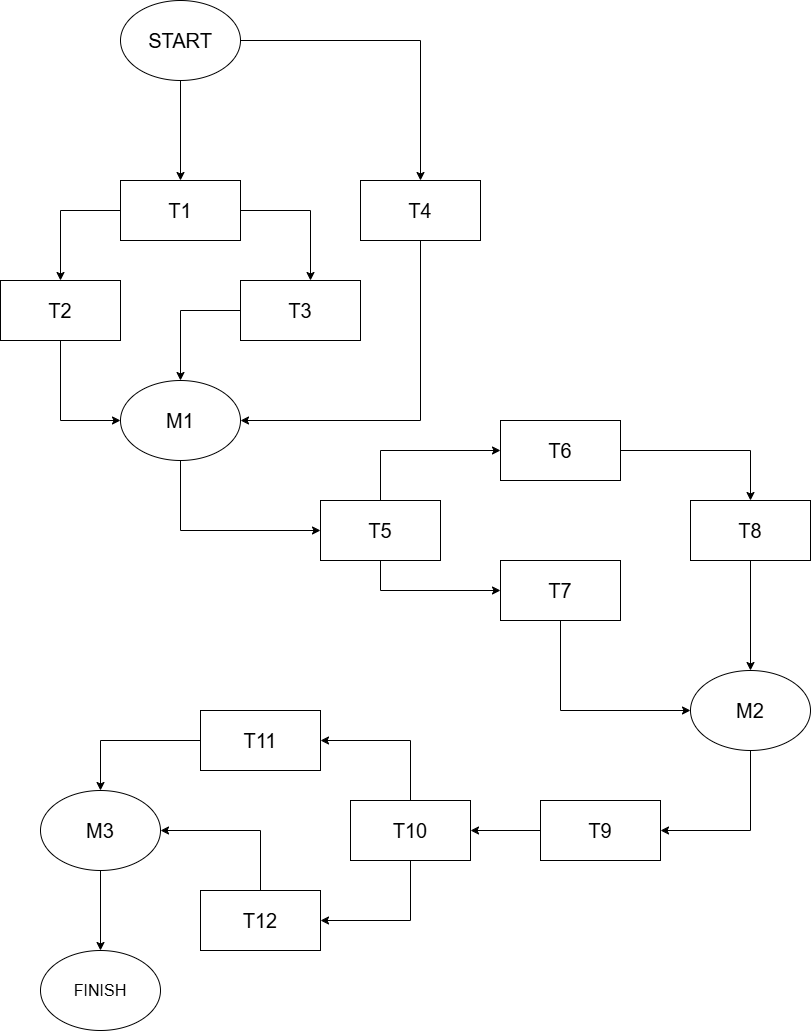


Fig 3.4: Task Chart – EMWIS

|  |  |
| --- | --- |
| T1 - Dynamic Search | T7 - Interactive Game |
| T2 - Recommended Mobiles | T8 - Cloud Integration |
| T3 - Multi-Level Filtering | T9 - Cart Management |
| T4 - AJAX Request Handling | T10 - Checkout |
| T5 - Mobile Comparison | T11 - Track Order |
| T6 - Chat Bot | T12 - Order History |
|  |  |

**Task Dependency Table:**

|  |  |  |
| --- | --- | --- |
| Task | Duration (Days) | Predecessors |
| T1 | **10** | **—** |
| T2 | **10** | **T1** |
| T3 | **8** | **T1** |
| T4 | **6** | **—** |
| T5 | **8** | **T2, T3, T4** |
| T6 | **10** | **T5** |
| T7 | **6** | **T5** |
| T8 | **6** | **T6** |
| T9 | **8** | **T7, T8** |
| T10 | **6** | **T9** |
| T11 | **10** | **T10** |
| T12 | **6** | **T10** |

### Cost Estimation for Project – Organic

Total Lines of Code, KLOC = **10.5 KLOC**

Effort, E = 2.4\*(KLOC)1.05 PM

= 2.4\*(10.5)1.05 PM

≈ **5.4 PM**

Development Time, Tdev = 2.5\*(E)0.38 Months

= 2.5\*(5.4)0.38 Months

≈ **3.2 Months**

## Unit Testing

Unit Testing is a type of software testing where **individual components (smallest parts) of a program** — like functions, methods, or classes — are tested **in isolation** to ensure they work correctly.

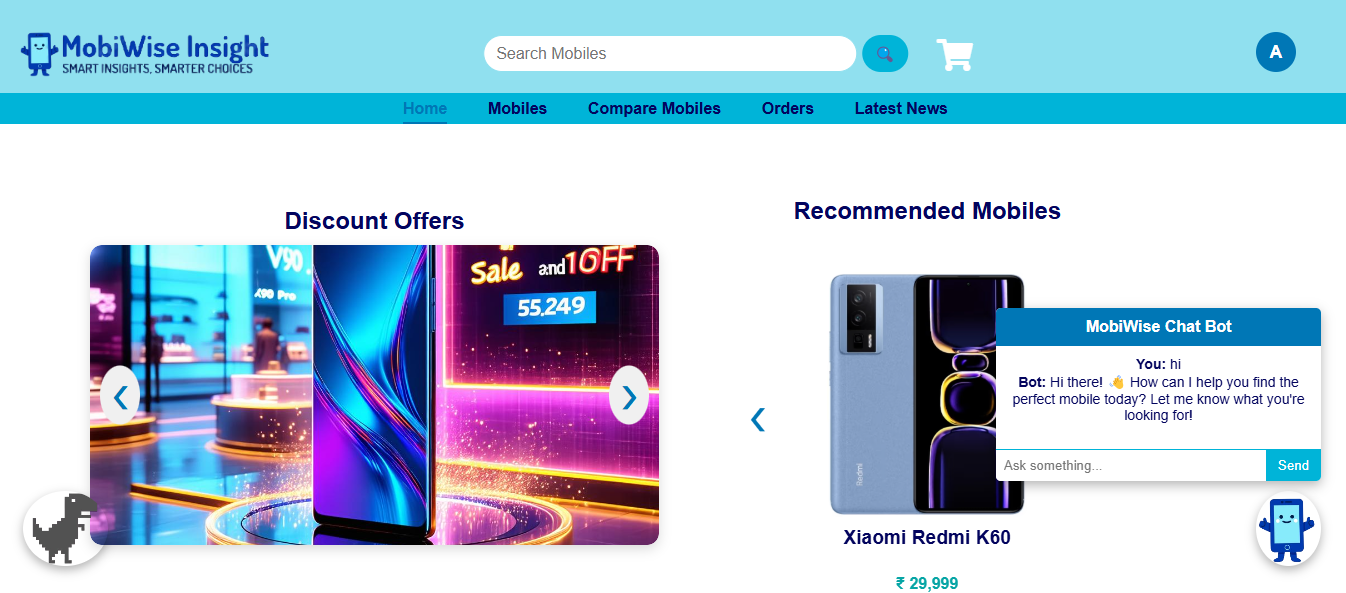


Fig 3.5: Chatbot Unit Test

## Integration Testing

Integration Testing is a type of testing where **two or more units/modules are combined** and tested **as a group** to verify that they work **together** as expected.

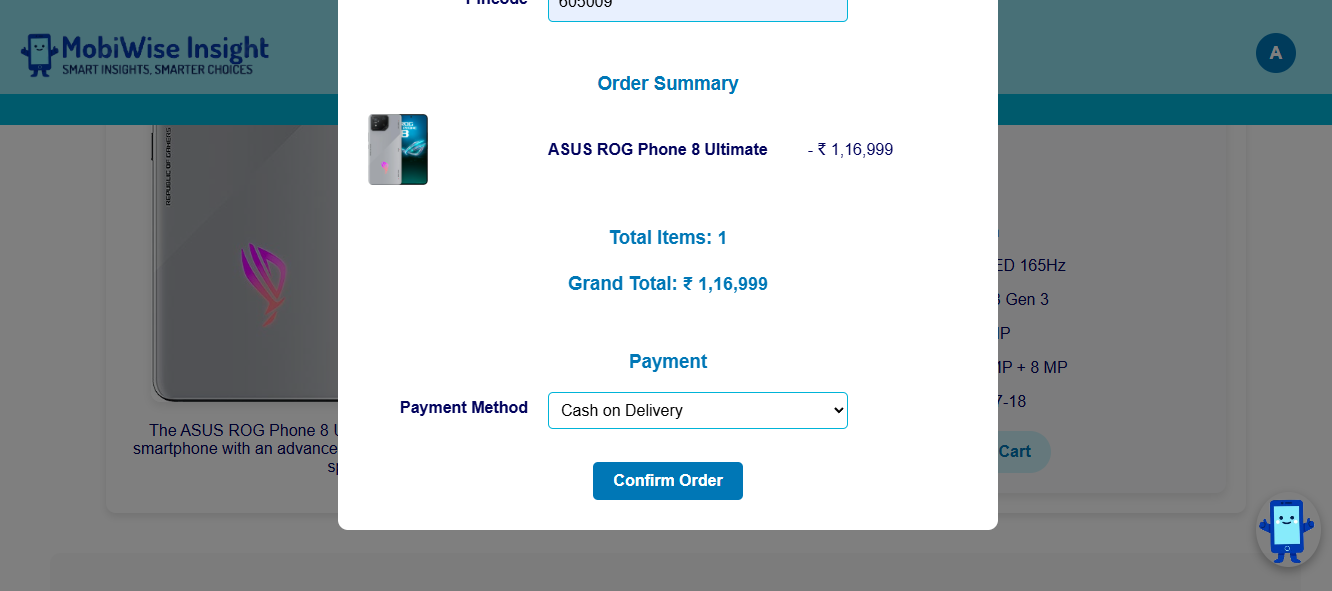


Fig 3.6: Checkout Form

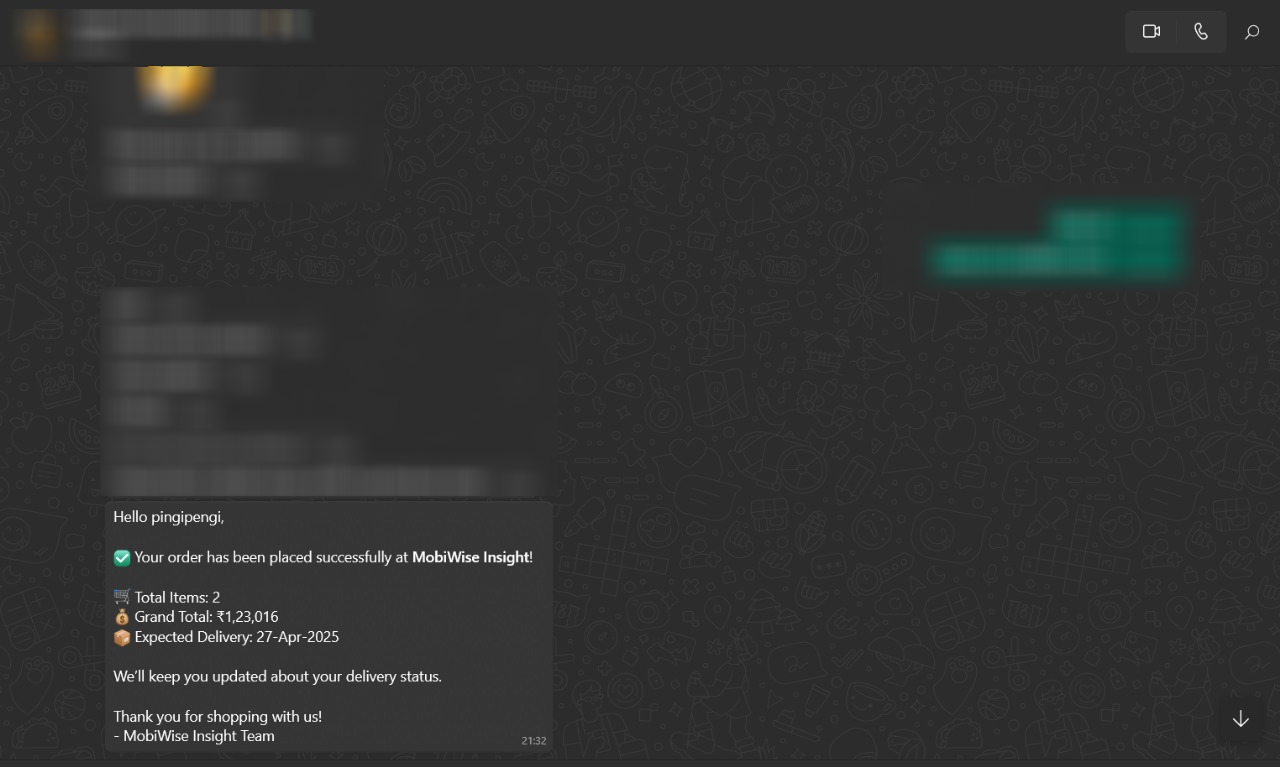


Fig 3.7: Order Confirmation WhatsApp Message

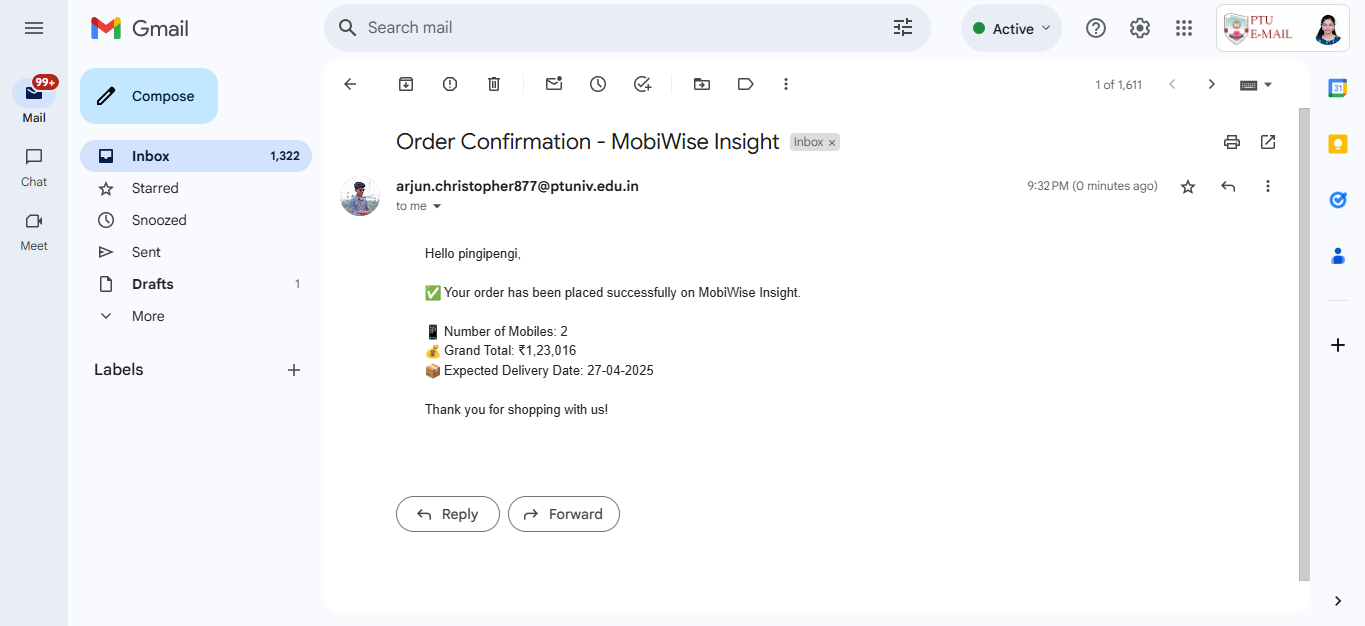


Fig 3.8: Order Confirmation Email

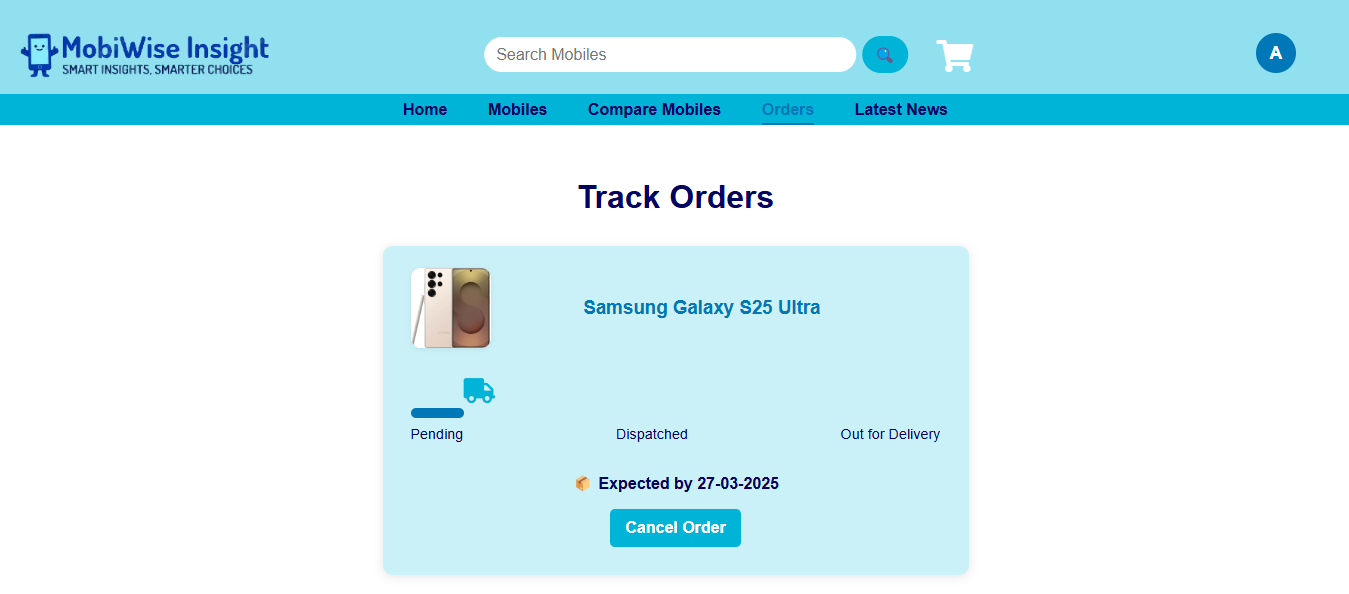


Fig 3.9: Order Tracking



Fig 3.10: Order History

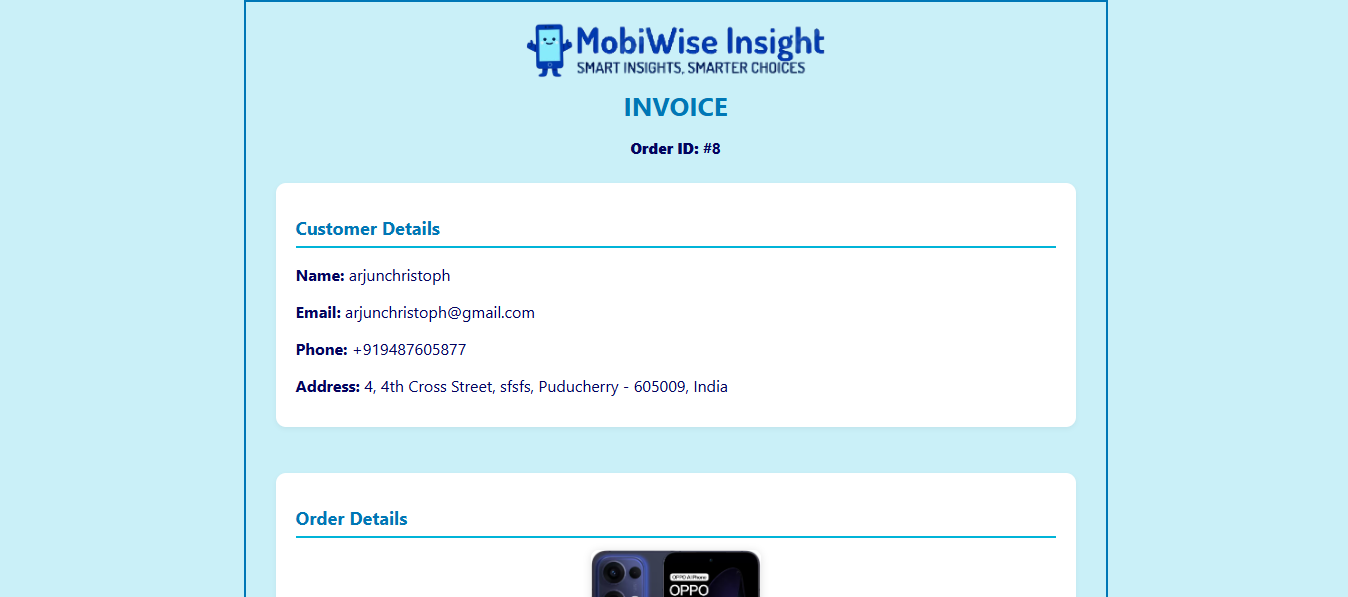


Fig 3.11: Order Invoice

## Performance Testing

Performance testing is the process of evaluating a system's speed, responsiveness, and stability under a specific workload to ensure it meets required performance standards.

For the EMWIS, performance testing was conducted using Locust, a Python-based open-source load testing tool. The objective was to simulate multiple concurrent users performing actions like mobile search, cart updates, and order checkout to assess system behavior under load. Locust allowed us to define realistic user journeys and measure critical metrics such as response time, throughput, and error rates, ensuring the platform's reliability during peak usage.

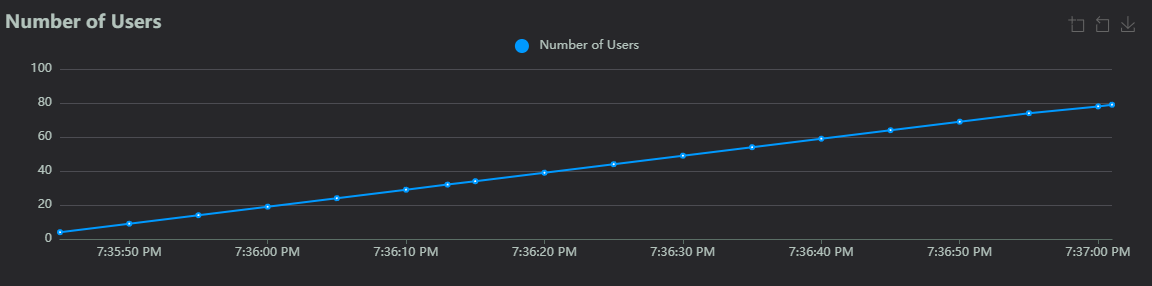


Fig 3.12: Number of Users Graph

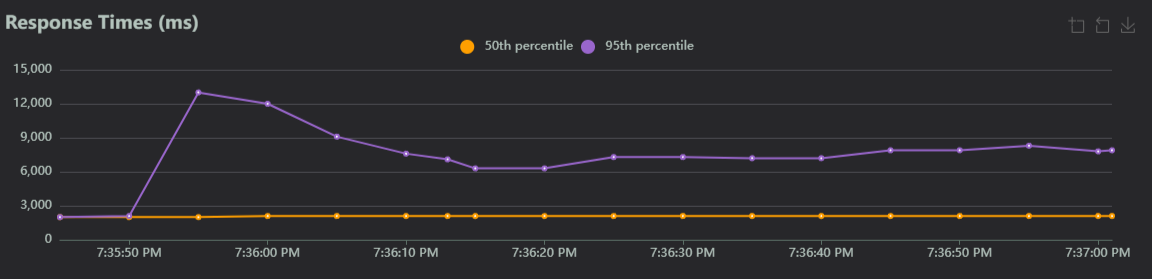


Fig 3.13: Response Times Graph

## Smoke Testing

As a critical component of white box testing, smoke testing validates that essential software functionalities perform as expected, providing an initial gauge of software stability before deeper testing commences. Daily execution of smoke tests ensures timely detection and resolution of any issues, facilitating consistent progress and upholding stringent quality assurance benchmarks in the project's lifecycle.

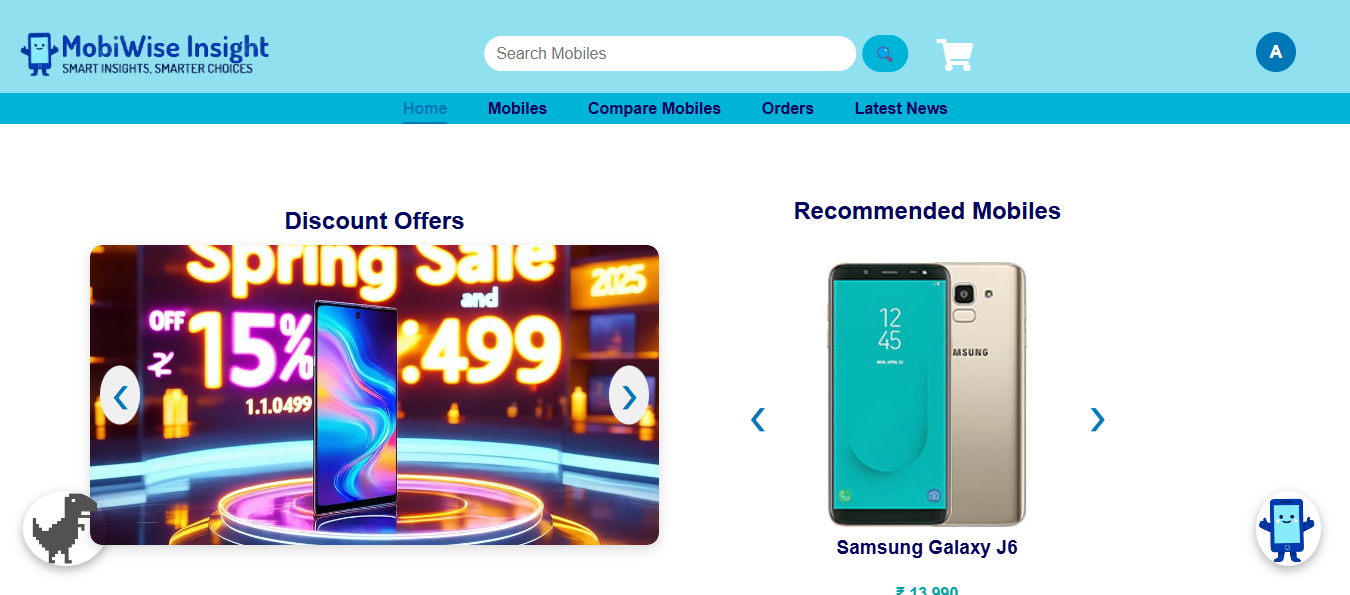


Fig 3.14: Successful loading of Home Page

## Black Box Testing

Black Box Testing is a software testing method where the internal code, structure, and implementation details are hidden from the tester. The entire Project is tested under Validation.



Fig 3.15: User Credentials based Authentication