|  |
| --- |
| 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:**

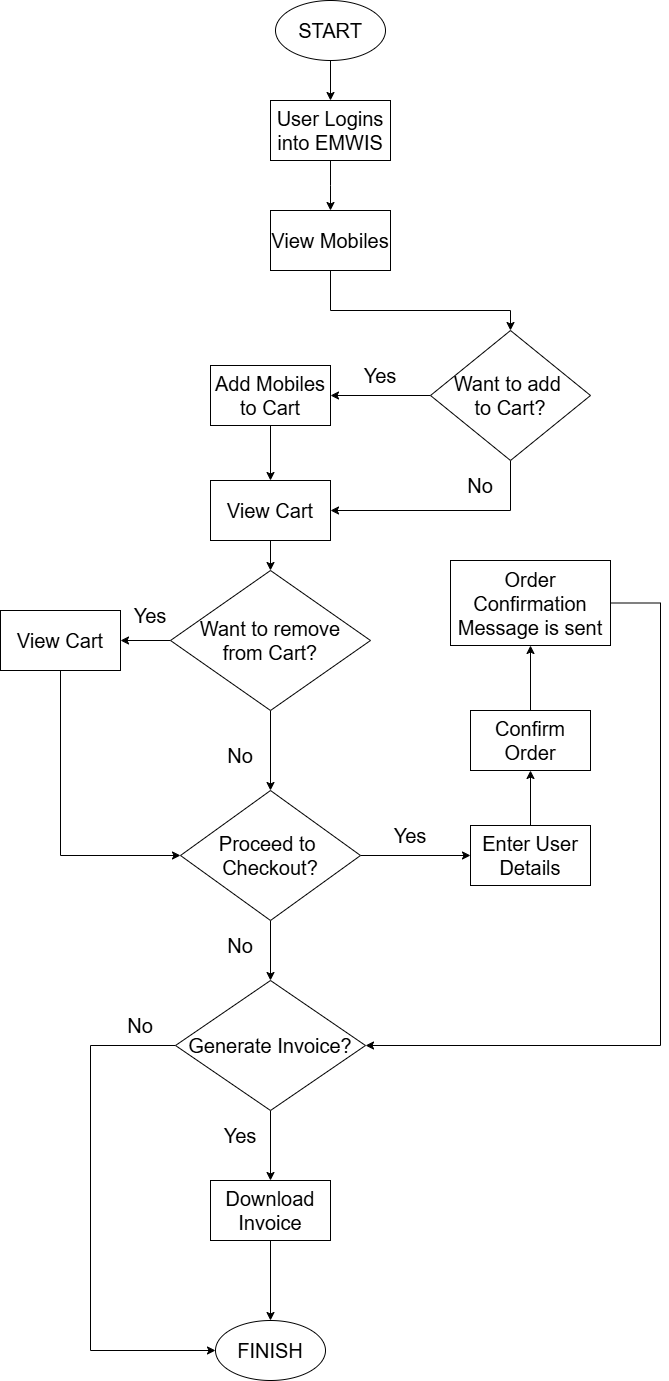
****

Fig 4.1: Flowchart – Cart and Order Management Activity

**Flowgraph:**

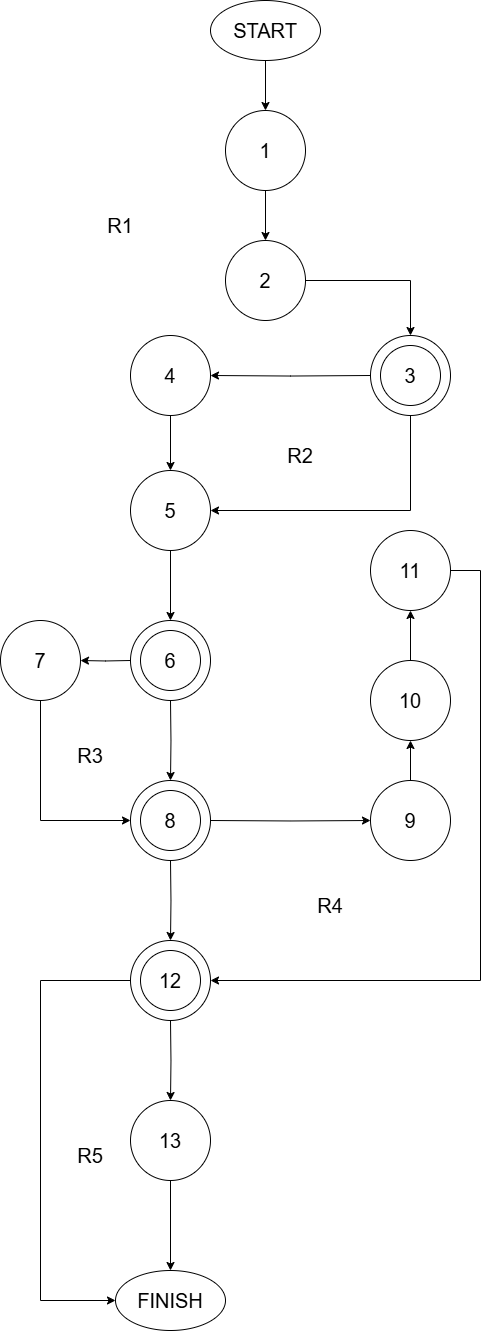
****

Fig 4.2: 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**

### Loop Testing

Predicate Nodes– **3, 6, 8, 12**

**Path 1 – R1 (Simple Loop at Node 2)**

* Path: 1 → 2 → 3 → 2 → 3 → 4 → 5...
* Covers two iterations of the loop R1.

**Path 2 – R2 (Simple Loop from 3 → 5)**

* Path: 2 → 3 → 5 → 3 → 4 → 5...
* Tests multiple entries into 3 before exiting.

**Path 3 – R3 (Back from 7 to 6)**

* Path: 5 → 6 → 7 → 6 → 8...
* Tests re-entry into node 6 from 7.

**Path 4 – R4 (Longer loop through 9-10-11)**

* Path: 8 → 9 → 10 → 11 → 8 → 12...
* Tests the sub-path loop involving multiple nodes.

**Path 5 – R5 (Back loop at node 13)**

* Path: 12 → 13 → 12 → 13 → FINISH
* Tests multiple cycles of 12–13 loop.

### Activity Chart

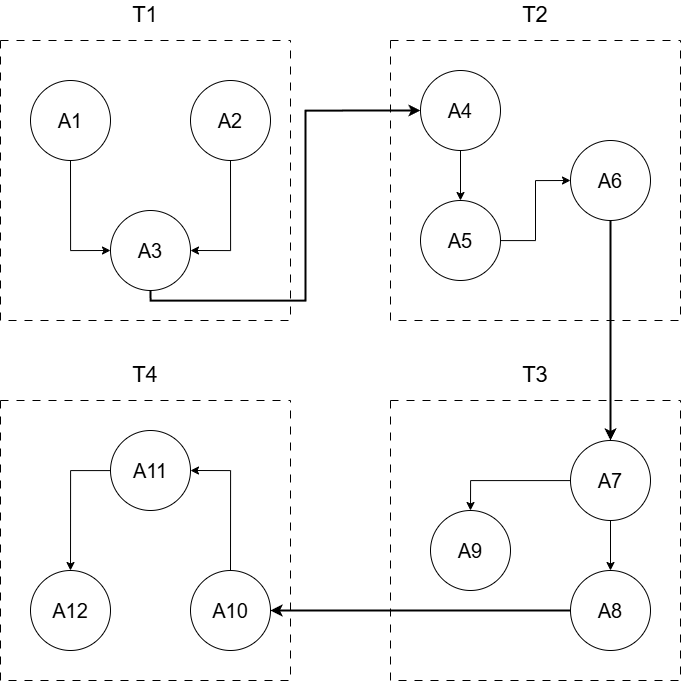


Fig 4.3: Activity Chart – Cart and Order Management Module

|  |  |
| --- | --- |
| 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 |

**Activity Dependency Table:**

|  |  |  |
| --- | --- | --- |
| Activity | Duration (Days) | Predecessors |
| A1 | **2** | **—** |
| A2 | **3** | **—** |
| A3 | **3** | **A1, A2** |
| A4 | **2** | **A3** |
| A5 | **2** | **A4** |
| A6 | **2** | **A5** |
| A7 | **4** | **A6** |
| A8 | **3** | **A7** |
| A9 | **3** | **A7** |
| A10 | **3** | **A8** |
| A11 | **2** | **A10** |
| A12 | **1** | **A11** |

### Task Chart

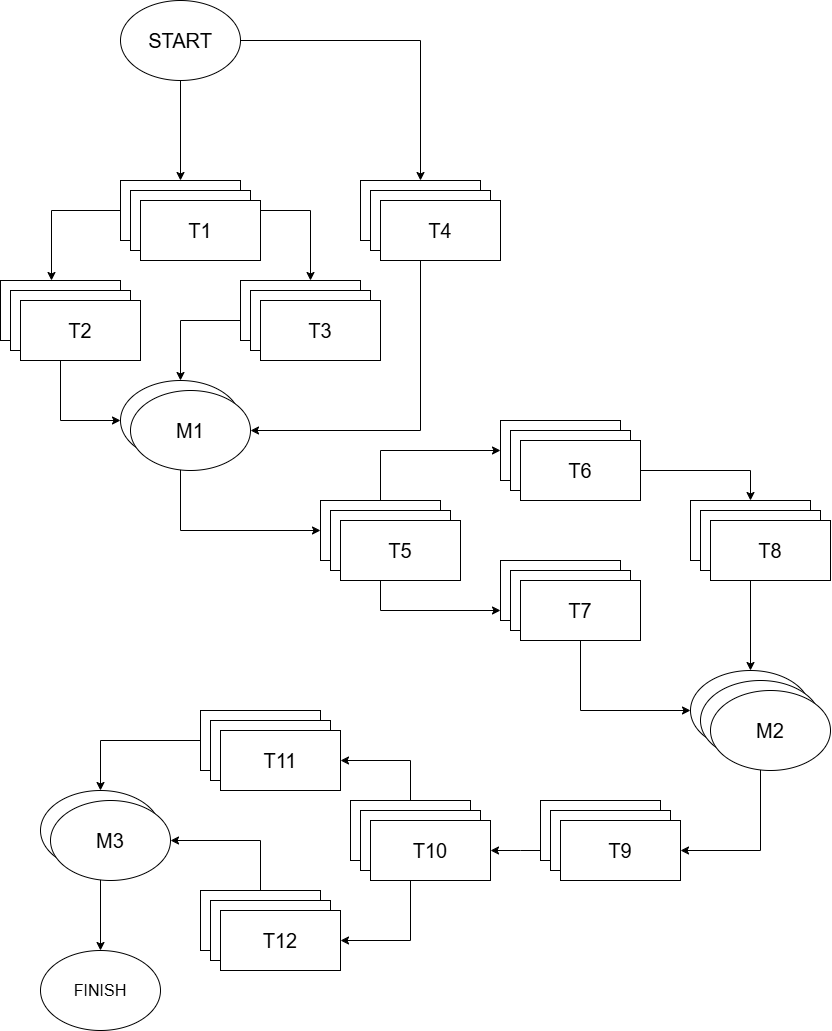


Fig 4.4: Task Chart – EMWIS

|  |  |
| --- | --- |
| 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.

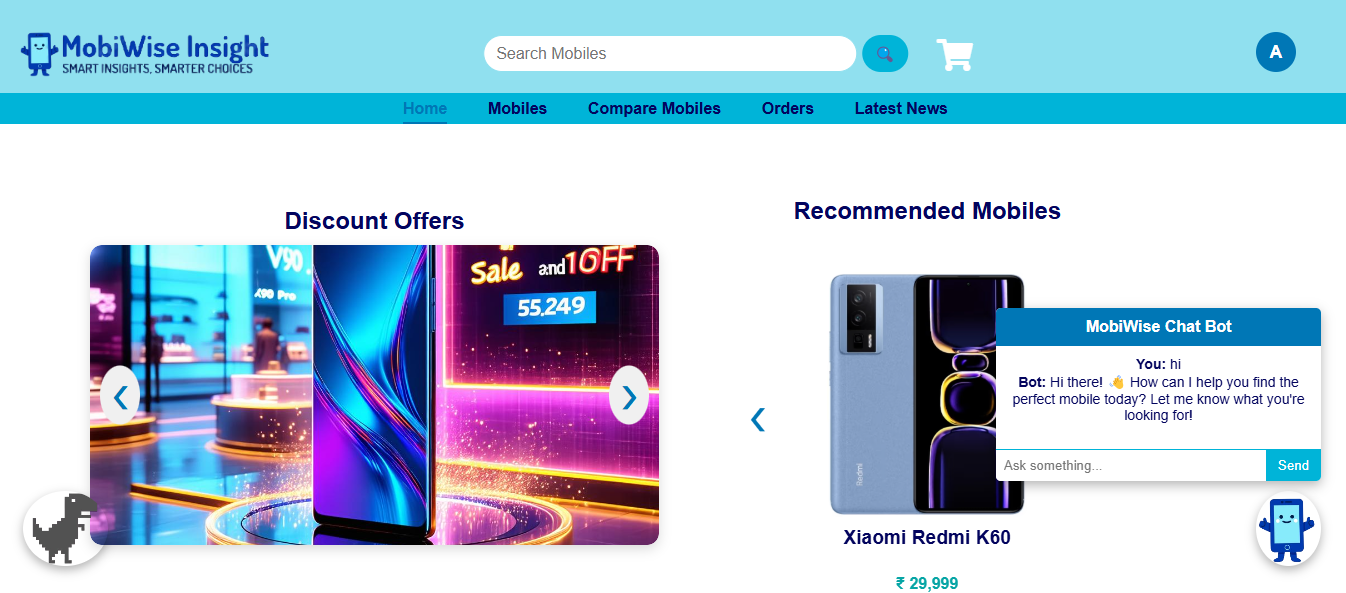


Fig 4.5: Chatbot Unit Test

## 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.

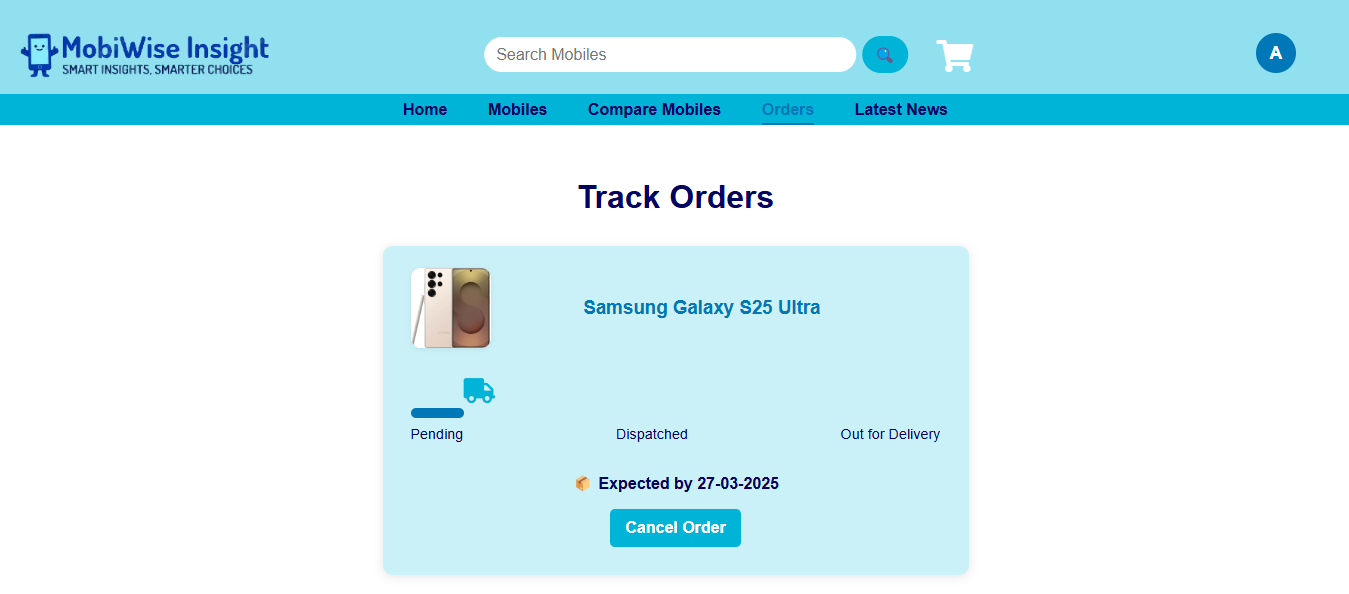


Fig 4.6: Order Tracking



Fig 4.7: Order History

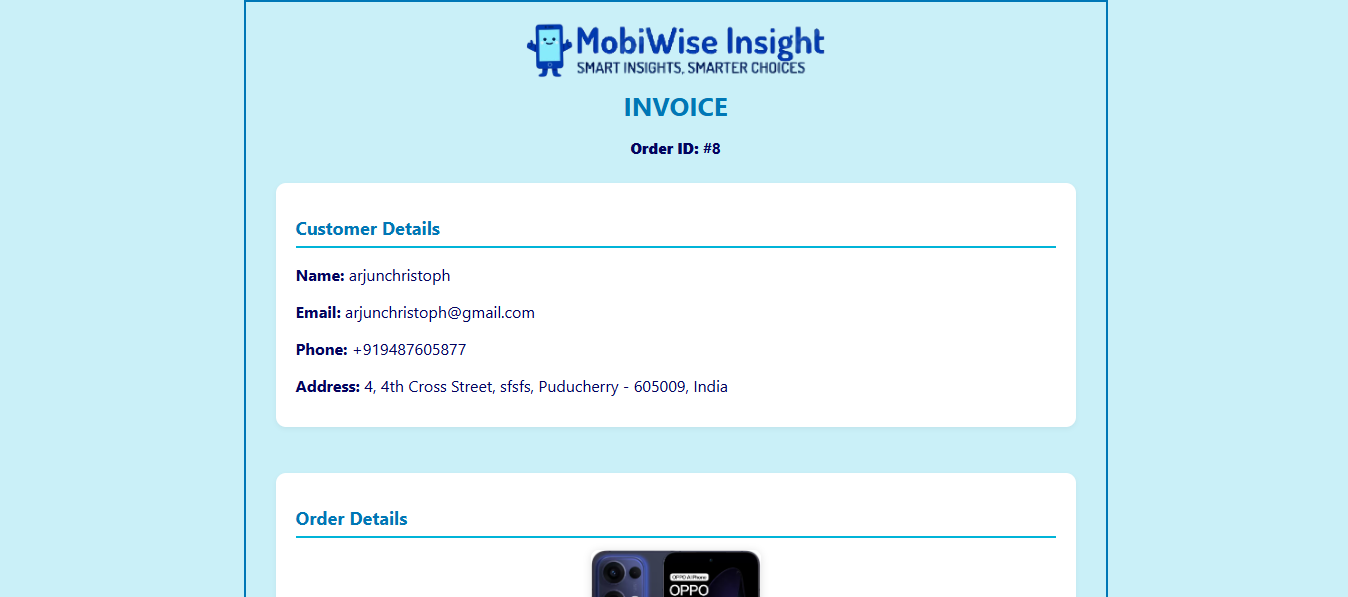


Fig 4.8: Order Invoice

## 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.

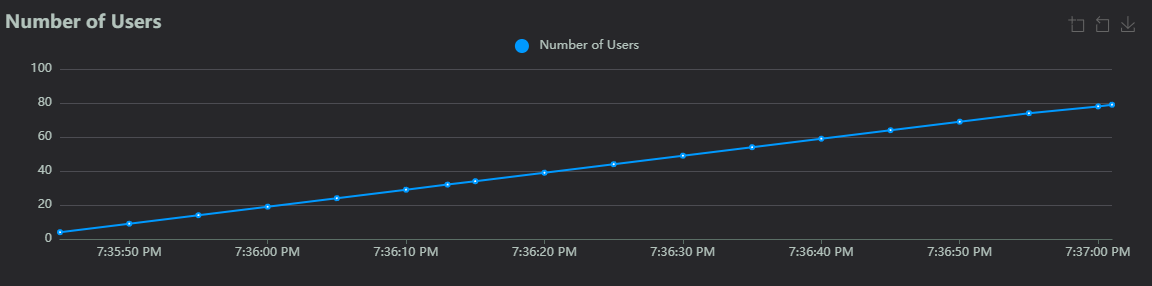


Fig 4.9: Number of Users Graph

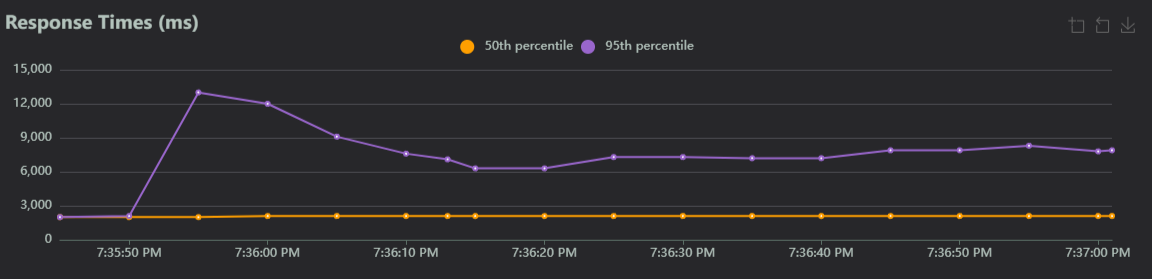


Fig 4.10: Response Times Graph

## 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.

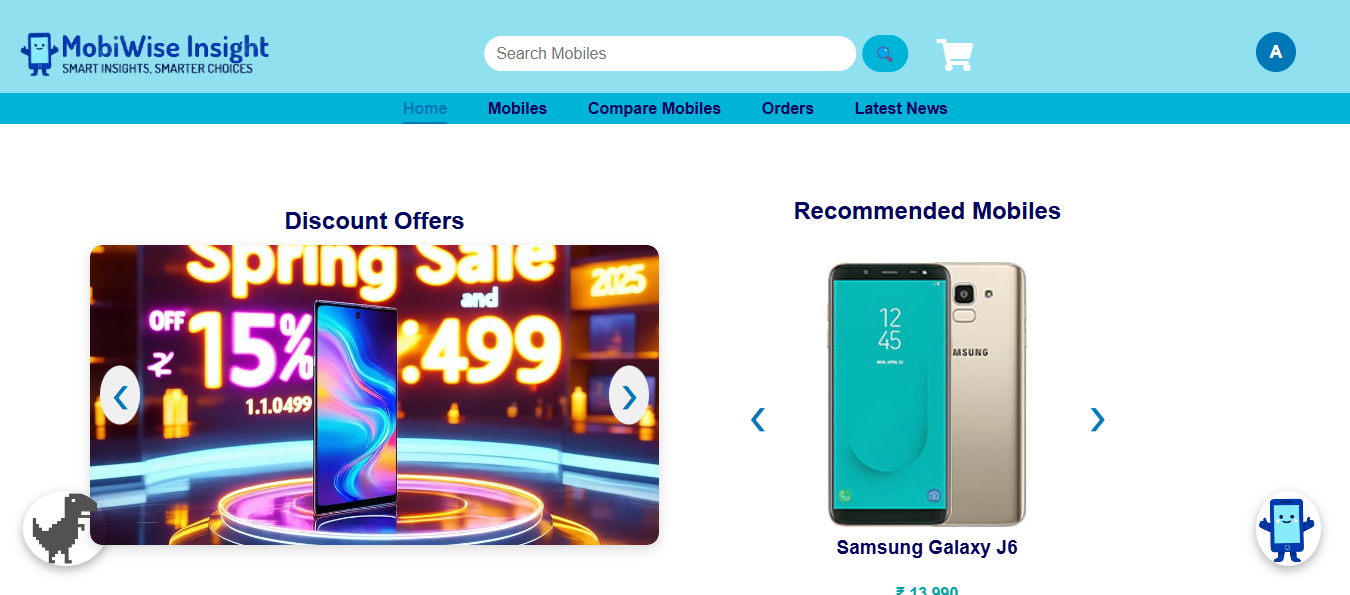


Fig 4.11: Successful loading of Home Page

## 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 4.12: User Credentials based Authentication

## Results and Discussion

* Successfully developed and deployed a fully functional real-time mobile price tracking and analysis engine, enabling users to receive accurate and timely recommendations based on personal preferences, pricing trends, and product specifications.
* Implemented dynamic user interface components, including mobile comparison dashboards, cart and checkout systems, order tracking views, and chatbot interaction—all contributing to increased user engagement and intuitive navigation.
* Integrated advanced notification systems, including WhatsApp messaging, email alerts, and SMS notifications, for communicating real-time price drops, order confirmations, and discount campaign updates.
* Leveraged cloud-based scheduling (via APScheduler) to automate time-sensitive tasks such as discount expiry handling and periodic WhatsApp notifications, improving operational efficiency.
* Enabled secure and personalized user sessions using JWT authentication and cookie-based login persistence, thereby streamlining the user experience across sessions.
* Deployed modular, role-based admin panels for efficient backend operations—allowing Super Admins, Product Managers, and Data Entry Operators to collaboratively manage data in real time.
* Enhanced decision-making confidence among users by integrating AI-powered comparison recommendations and contextual insights, ultimately reducing confusion and improving purchase satisfaction.
* Achieved performance and reliability targets through optimized database queries (Oracle), efficient session handling, and caching techniques, ensuring responsiveness even with increased data load.

## Conclusion & Future Enhancement

The **Economical MobiWise Insight System (EMWIS)** delivers a comprehensive, intelligent, and user-centric platform that empowers mobile consumers to make well-informed and timely purchasing decisions. By integrating real-time mobile feature analytics, personalized recommendation mechanisms, automated discount notifications, and a secure order management workflow, EMWIS successfully addresses the limitations present in existing mobile shopping and comparison tools.

Future enhancements will focus on:

* Expanding the range of supported e-commerce APIs to cover a broader catalog of mobile devices and pricing sources.
* Enhancing the accuracy and depth of AI-driven recommendation models.
* Improving chatbot intelligence to offer smarter query resolutions and guided buying support.
* Introducing a cross-platform **mobile application** to extend accessibility and convenience for users on the go.
* Adding user analytics and behavioral insights for more adaptive personalization and engagement.

With its foundation in scalability, usability, and innovation, EMWIS is positioned to evolve into a next-generation mobile commerce assistant for digitally empowered consumers.

## References

[1] **AI-Driven Predictive Analytics in Retail: A Review of Emerging Trends and Customer Engagement Strategies** – John Smith, Michael Brown. *International Journal of Management & Entrepreneurship Research, 2024.* Available at: <https://ijmer.com/ai-retail-2024>

[2] **AI-Driven Pricing Algorithms for Efficient Inventory and Cost Management in Retail** – Mark Wilson, Lisa Green. *The European Journal of Research and Development, 2024.* Available at: <https://ejrd.com/ai-pricing-2024>

[3] **Amazon**. Available at: [https://www.amazon.com](https://www.amazon.com/)

[4] **ChatGPT**. OpenAI conversational AI model. Available at: [https://chat.openai.com](https://chat.openai.com/)

[5] **Coolors Color Palette Generator**. UI color scheme reference. Available at: [https://coolors.co](https://coolors.co/)

[6] **Currents News API** – For real-time mobile-related news. Available at: [https://currentsapi.services](https://currentsapi.services/)

[7] **Disqus Documentation** – Embedded comment system for websites. Available at: [https://disqus.com](https://disqus.com/)

[8] **GNews API** – Global news API used for fetching mobile-related news. Available at: [https://gnews.io](https://gnews.io/)

[9] **Google Gemini API** – AI-powered suggestion engine used for mobile comparison. Available at: [https://ai.google.dev](https://ai.google.dev/)

[10] **GSMArena –** Mobile Comparison Platform. Available at: [https://www.gsmarena.com](https://www.gsmarena.com/)

[11] **Hugging Face API** – AI model hosting and inference API used for generating mobile advertisement images. Available at: [https://huggingface.co](https://huggingface.co/)

[12] **Javatpoint –** Flask Tutorial. Web framework reference. Available at: <https://www.javatpoint.com/flask>

[13] **NewsAPI** – News API to retrieve mobile-related articles. Available at: [https://newsapi.org](https://newsapi.org/)

[14] **Oracle Official Documentation** – Oracle database SQL and PL/SQL references. Available at: [https://docs.oracle.com](https://docs.oracle.com/)

[15] **Pricing Cloud IaaS Computing Services** – David Johnson, Emily Carter. *Journal of Cloud Computing: Advances, Systems and Applications, 2020.* Available at: <https://cloudcomputing-journal.com/pricing-iaas>

[16] **pywhatkit GitHub Repository** – Python module for WhatsApp messaging automation. Available at: <https://github.com/Ankit404butfound/pywhatkit>

[17] **W3Schools –** HTML/CSS Tutorials. Web development reference. Available at: [https://www.w3schools.com](https://www.w3schools.com/)

[18] **Yahoo Finance API –** Used for tracking mobile market trends and competitor pricing. Available at: <https://www.yahoofinanceapi.com>

[19] **YCharts –** Financial data and analytics platform for evaluating tech market performance.Available at: <https://ycharts.com>

[20] **YouGov –** Research and analytics platform providing consumer insights for mobile trends. Available at: <https://yougov.com>

[21] **ZDNet –** Tech news site referenced for mobile industry trends. Available at: <https://www.zdnet.com>