Application Name: SnatchIT- Smart eCommerce Web Extension Date: April 04, 2025

Prepared By: Team SnatchIT

# Overview

SnatchIT is a browser-based eCommerce extension designed to streamline online shopping by comparing product prices across various platforms in real-time. Built on a modular architecture comprising four key layers-Presentation, Business Logic, Data Access, and Infrastructure-snatchIT emphasizes speed, accuracy, and user convenience.

# Performance

Goal: Provide near-instantaneous price comparisons during user browsing.

Scenario: A user lands on a product page, and SnatchIT displays price comparisons from other sites.

Architectural Support: Caching Logic, Rate Limiter, Local Storage. Risks: Network latency, slow third-party API responses.

Mitigation: Caching, parallelized API calls, fallback to cached data.

# Usability

Goal: Deliver a smooth, intuitive user experience with minimal user effort.

Scenario: A user activates the extension and clearly sees a sorted list of prices with filtering options. Architectural Support: Browser Extension UI, Rendering Engine.

Risks: Interface overload, inconsistent UX across browsers. Mitigation: Progressive UI design, compatibility testing.

# Reliability

Goal: Ensure the extension functions consistently, even under partial failures.

Scenario: A third-party eCommerce API is down, but the extension still shows previously cached prices. Architectural Support: Error Handling Module, Local Storage Cache.

Risks: Data source changes or outages.

Mitigation: Use of robust fallback strategies and monitoring.

# Scalability

Goal: Seamlessly support growth in both user base and supported eCommerce platforms.

Scenario: During a large sale event, 1000s of users query multiple product comparisons simultaneously. Architectural Support: Backend Proxy / Cloud Infrastructure, Data Access Layer Modularization.

Risks: Performance degradation under load.

Mitigation: Use of cloud functions, load balancing, modular APIs.

# Security & Privacy

Goal: Safeguard user data and ensure secure communications.

Scenario: Users browse freely without risk of data being stored or misused. Architectural Support: Network Manager, Configuration Service.

Risks: Leakage of browsing behavior or session data.

Mitigation: Minimal data storage, local processing by default, GDPR compliance.

# Maintainability

Goal: Simplify extension updates, bug fixes, and feature additions.

Scenario: A new shopping platform (e.g., Etsy) needs to be supported with minimal code changes. Architectural Support: Layered Architecture, Data Mapper & Web Scraper Modules.

Risks: Tight coupling between layers.

Mitigation: Use of interfaces and dependency injection patterns.

# Interoperability

Goal: Function smoothly across major browsers and various eCommerce websites.

Scenario: The extension runs consistently on Chrome, Firefox, and Edge. Architectural Support: Browser APIs, DOM Parsing Strategies.

Risks: DOM structure variations and browser-specific quirks. Mitigation: Robust and adaptable DOM selectors, consistent testing.

# Conclusion

SnatchIT is designed with user-centricity, speed, and extensibility at its core. The quality attributes discussed are deeply supported by its thoughtful layered architecture, ensuring the extension remains fast, secure, reliable, and scalable as it evolves.