Software Requirements Specification

for

Shopping Aggregator

Version <1.0>

Prepared by BudBots

Group Name: BudBots

YUVAN REDDY	SE22UARI091	se22ucse098@mahindrauniversity.edu.in
AASRITH REDDY	SE22UARI092	se22ucse128@mahindrauniversity.edu.in
KSHITIJ KARMUSE	SE22UARI134	se22ucse122@mahindrauniversity.edu.in
CHANDANA GOUD	SE22UARI217	se22ucse276@mahindrauniversity.edu.in
JOSYULA MITHIL	S323LARI001	se22ucse266@mahindrauniversity.edu.in

Instructor:	Avinash Arun Chauhan
Course:	Software Engineering(CS3201)
Lab Section:	*AI-2*
Teaching Assistant:	*Sri Venkata Surya Phani Teja*
Date:	12-03-25

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	BudBots	Initial draft of the Software Requirements Specification (SRS) for Shopping Aggregator, outlining system functionalities, use cases, and constraints based on the project scope.	11/03/25

1. Introduction

1.1 Document Purpose

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements for the Shopping Aggregator, a platform designed to enhance the online shopping experience. The system enables users to effectively explore, compare, and analyse products based on qualities, pricing trends, and purchase alternatives by combining product data from several e-commerce platforms. This paper outlines the main features, limitations, and goals of the system, acting as a thorough guide for the development team, stakeholders, and end users.

This document's scope is centred on outlining the software needs for the shopping aggregator's essential features, such as price comparison, data collection, real-time updates, and an easy-to-use user interface. To help users make well-informed decisions, the platform seeks to give them comprehensive information into product trends and buying patterns. Although the system's core features are covered by this SRS, future additions like interfaces with other e-commerce platforms or mobile applications are not.

1.2 Product Scope

The buying Aggregator is a feature-rich platform that aggregates product data from several e-commerce websites to optimize the online buying experience. It helps customers to make well-informed decisions by facilitating effective pricing comparison, product trend analysis, and exploration of several purchase possibilities. The platform uses sophisticated filtering techniques and real-time data collection to deliver precise and current product information.

The Shopping Aggregator improves e-commerce transparency by providing a user-friendly interface and sophisticated search features, which saves consumers time and effort when looking for the best offers. Enhancing accessibility to competitive pricing, maximizing product discovery, and enabling smooth navigation among various online shops are some of its main goals. While preserving a scalable architecture that permits future connections with more platforms and enhanced functionalities, the system guarantees dependability and efficiency.

1.3 Intended Audience and Document Overview

This Software Requirements Specification (SRS) document is intended for multiple stakeholders involved in the development and evaluation of the Shopping Aggregator platform. The key audiences for this document include:

1. **Professor Avinash Arun Chauhan (Instructor for Software Engineering Course)**

The document serves as a reference for evaluating the project's technical feasibility, adherence to software engineering principles, and implementation strategy.

Recommended reading order:

- **Introduction (Section 1)** – Provides an overview of the project, including its purpose and scope.
- **Overall Description (Section 2)** – Covers the system's high-level functionality, constraints, and dependencies.
- **Specific Requirements (Section 3)** – Details the functional and non-functional requirements of the system.
- **Use Case Model (Section 3.3)** – Helps assess how well the system design meets real-world scenarios.
- **Software Quality Attributes (Section 4.3)** – Provides insights into software reliability, usability, and maintainability.

2. Online Shoppers (Clients and End Users)

These are the primary users of the Shopping Aggregator platform, who will utilize the system to compare prices, analyze product trends, and find the best purchasing options.

Recommended reading order:

- **Product Overview (Section 2.1)** – Explains how the platform works and what features are included.
- **Product Functionality (Section 2.2)** – Lists key functions such as price comparison, trend analysis, and real-time updates.
- **External Interface Requirements (Section 3.1)** – Describes how users will interact with the system through the UI.
- **Safety and Security Requirements (Section 4.2)** – Details measures ensuring secure transactions and data privacy.

3. Development Team

This group is responsible for designing, coding, and testing the Shopping Aggregator platform. The document provides detailed technical specifications to guide the development process.

Recommended reading order:

- **Functional Requirements (Section 3.2)** – Defines core system operations and expected behaviors.
- **Design and Implementation Constraints (Section 2.3)** – Lists technology stacks, limitations, and dependencies.
- **External Interface Requirements (Section 3.1)** – Describes how the front-end, back-end, and databases interact.
- **Non-Functional Requirements (Section 4)** – Includes performance, security, and scalability requirements.

1.4 Definitions, Acronyms and Abbreviations

AI: *Artificial Intelligence*
API: *Application Programming Interface*
AWS: *Amazon Web Services*
COMET: *Concurrent Object Modeling and Architectural Design Method*
DBMS:*Database Management System*
DDoS: *Distributed Denial of Service*
GDPR: *General Data Protection Regulation*
GUI: *Graphical User Interface*
HTML: *HyperText Markup Language*
IEEE: *Institute of Electrical and Electronics Engineers*
ISO: *International Organization for Standardization*
JSON: *JavaScript Object Notation*
MFA: *Multi-Factor Authentication*
MySQL: *Structured Query Language Database Management System*
NLP: *Natural Language Processing*
NoSQL: *Non-Relational Database Management System*
OAuth: *Open Authorization*
PCI DSS: *Payment Card Industry Data Security Standard*
RAM: *Random Access Memory*
React.js: *JavaScript Library for Building User Interfaces*
REST API: *Representational State Transfer Application Programming Interface*
SRS: *Software Requirements Specification*
SQL: *Structured Query Language*
SSL/TLS: *Secure Sockets Layer / Transport Layer Security*
UML: *Unified Modeling Language*
URL: *Uniform Resource Locator*
WCAG: *Web Content Accessibility Guidelines*

1.5 Document Conventions

1. Formatting Conventions

- The document follows the **IEEE formatting requirements** for software documentation.
- **Font**: Arial, size **11** used consistently throughout the document.
- **Line Spacing**: Single-spaced for readability and compactness.
- **Margins**: 1-inch margins on all sides of the document.

2. Naming Conventions

- **Acronyms**: Defined in the **Glossary** section when first introduced.
- **Requirement Labels**: Functional and non-functional requirements are labeled as **F1, F2, P1, P2, etc.** for easy reference.
- **Use Case Identifiers**: Each use case is labeled (e.g., **U1: Login**, **U2: Browse Items**) for tracking.

3. Typographical Conventions

- **Italics**: Used for comments, placeholder text, and special notes.
- **Bold**: Used for section headings, key terms, and important information.
- **Tables and Lists**:
  - Used for structured data presentation, following a clean and readable layout.
  - Numbered or bulleted lists are used to enhance readability and organization.

1.6 References and Acknowledgments

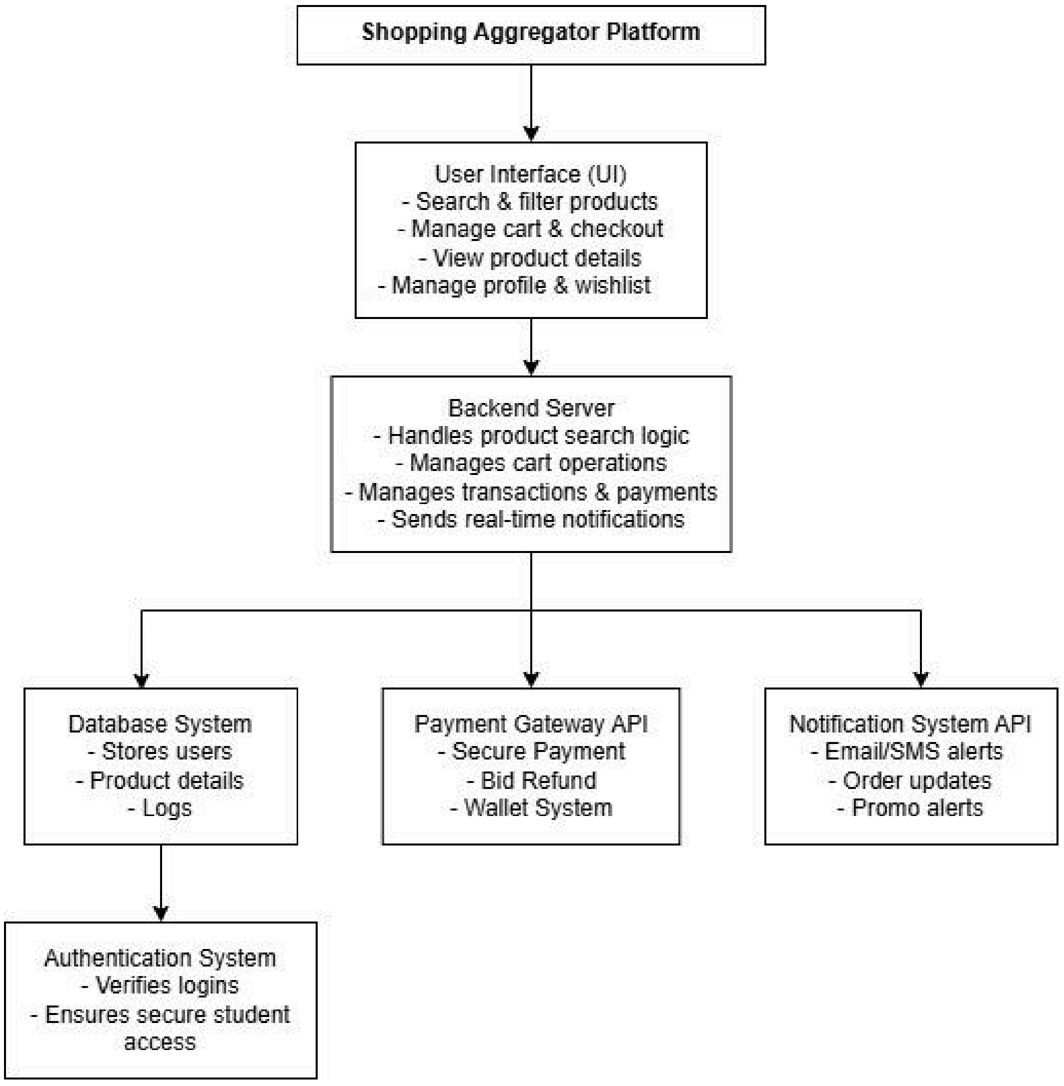
The development of this **Shopping Aggregator System** is based on best practices and reference materials, including but not limited to:

- **IEEE Std 830-1998**: Recommended Practice for Software Requirements Specifications.
- **ISO/IEC 27001**: International Standard for Information Security Management.
- **GDPR Compliance Guidelines** – https://gdpr-info.eu/
- **Payment Card Industry Data Security Standard** – https://www.pcisecuritystandards.org/

2. Overall Description

2.1 Product Overview

The Shopping Aggregator is a standalone web application that aggregates product data from many e-commerce platforms to improve online shopping. It offers consumers a smooth interface for tracking prices, searching products, and comparing them. It was created from the ground up as a stand-alone system that retrieves real-time product data by integrating with partner e-commerce APIs. When a user decides to make a purchase, the program also links to payment gateways and takes them to the checkout page.



2.2 Product Functionality

- **Real-time Data Aggregation**: Fetches and updates product details (prices, availability, reviews) from multiple e-commerce platforms.
- **Product Search and Comparison**: Allows users to search for products, view structured details, and compare options across platforms.
- **AI-Driven Product Matching**: Identifies identical or similar products using AI-based analysis of names, descriptions, images, and specifications.
- **Filtering and Sorting**: Provides filters (price range, brand, ratings) and sorting options (lowest price, best-rated) to refine search results.
- **Personalized Recommendations**: Suggests relevant products based on user behavior, past searches, and purchase history.
- **Price History and Trend Analysis**: Tracks historical pricing data and provides insights into price trends over time.
- **Notification and Alert System**: Sends price drop alerts and deal notifications via email or in-app messages.
- **User Account Management**: Supports account creation, login, profile updates, and password recovery.
- **Responsive User Interface**: Ensures an intuitive and adaptive experience across desktop and mobile devices.
- **Integration with External APIs**: Establishes secure connections with e-commerce APIs for real-time product data retrieval.
- **Data Security and Privacy**: Implements security measures to protect user data and comply with privacy regulations.
- **Administrative Reporting and Analytics**: Provides insights on user behavior, system performance, and product trends via an admin dashboard.

2.3 Design and Implementation Constraints

1. Hardware Constraints

- The system must operate efficiently on **standard consumer devices**, including desktops, laptops, and mobile devices with moderate hardware capabilities.
- **Server-side performance** must be optimized to handle high-volume API requests, ensuring low-latency product aggregation and real-time data updates.
- Storage must be scalable to accommodate a growing database of product listings, price histories, and user preferences.

2. Software Constraints

- **Technology Stack Restrictions**: The system must use:
    - **Frontend**: React.js
    - **Backend**: Node.js and Express.js
    - **Databases**: MongoDB (NoSQL) & MySQL (Relational)
    - **API Framework**: Express.js for routing and middleware support
- Use of unapproved technologies could lead to integration issues, security vulnerabilities, and performance inefficiencies.
- **Concurrency Handling**: The system must support parallel data aggregation and simultaneous user operations without race conditions, inconsistent data retrieval, or query conflicts.

3. Security Constraints

- **Data Encryption & Privacy Compliance**:
  - User data, including personal details and payment information, must be encrypted (AES-256 for storage, TLS 1.2+ for transmission).
  - Compliance with GDPR and other relevant data privacy regulations is mandatory.
- **Secure Payment Integration**:
  - Payment transactions must be processed through trusted third-party Payment Gateway APIs (e.g., Stripe, PayPal, Razorpay) with secure encryption and fraud detection mechanisms.
  - Insecure payment processing could result in financial losses and legal liabilities.

4. Regulatory & Compliance Constraints

- The platform must comply with *Mahindra University's* regulations and guidelines, ensuring adherence to institutional policies for data handling, software deployment, and security.
- Non-compliance may result in restricted access, forced shutdowns, or rejection by university administration.

5. Development Standards

- The software must follow the *COMET* method for software design.
  - Reference: *Gomaa, H. (2011). Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures.*
- The system must use UML diagrams for system modeling, including:
  - Use-case diagrams for functional interactions
  - Class diagrams for object relationships
  - Sequence diagrams for workflow representation

2.4 Assumptions and Dependencies

The development of the **Shopping Aggregator System** is based on the following key assumptions and dependencies. Any changes to these assumptions may impact the design, implementation, or performance of the system.

1. Third-Party API and Data Availability

- It is assumed that partner e-commerce platforms (e.g., Amazon, Flipkart, eBay) will provide reliable and stable APIs for real-time product data retrieval.
- If API access is restricted, unreliable, or subject to rate limits, alternative data-fetching methods (e.g., web scraping) may need to be implemented, impacting system performance and compliance with site policies.

2. Development and Operating Environment

- The system is expected to be hosted on a cloud-based infrastructure (e.g., AWS, Google Cloud, Azure) with scalability support for increasing user demand.
- Assumes consistent network availability for real-time data aggregation and API calls; disruptions may impact functionality.

- The application will be cross-platform compatible and optimized for both desktop and mobile browsers.

3. Technology Stack Stability

- It is assumed that the chosen technology stack *(React.js, Node.js, Express.js, MongoDB, MySQL)* will remain viable and supported throughout the development and deployment phases.
- Any major updates, deprecations, or security vulnerabilities in these frameworks may require modifications to the system architecture.

4. User Behavior and System Load

- It is assumed that concurrent users will remain within a manageable limit during initial deployment, and the system will be able to scale as traffic grows.
- If the user base expands rapidly beyond expectations, additional performance optimization and infrastructure upgrades may be required.

5. Payment Gateway and Compliance

- It is assumed that trusted third-party payment gateways (Stripe, PayPal, Razorpay) will be used for transactions, ensuring security and compliance with financial regulations.
- Changes in compliance requirements (e.g., PCI DSS, GDPR) could necessitate system updates.

6. Reuse of External Software Components

- Assumes integration with existing open-source libraries for AI-driven product matching, data visualization, and price trend analysis.
- If these components are no longer maintained, alternative solutions may need to be developed, affecting project timelines.

These assumptions significantly impact the system's **design, scalability, and reliability**. If any assumption is incorrect or changes, the project scope may need to be adjusted accordingly.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The shopping assistant platform will have a responsive web-based interface accessible via desktop and mobile devices. Users will interact with the platform through a graphical user interface (GUI) designed for ease of navigation and usability.

Main Features of the UI:

- **Homepage:** Displays trending deals, categories, and a search bar.
- **Search & Filters:** Users can search for products and apply filters (price range, brand, rating, etc.).
- **Product Comparison Page:** Displays multiple products side-by-side with price tracking, availability, and reviews.
- **User Dashboard:** Allows users to manage accounts, view saved items, and receive personalized recommendations.
- **Notifications Panel:** Alerts users about price drops, new deals, and personalized recommendations.

User Interaction:

- **Navigation Menus & Buttons:** Users click or tap on menus, filters, and products.
- **Search Bar & Auto-Suggestions:** Users type product names and get instant suggestions.
- **Interactive Graphs & Charts:** Visual representation of price history for selected products.
- **Login & Signup Forms:** Users enter credentials to access personalized features.

3.1.2 Hardware Interfaces

Since this is a web-based platform, it mainly interacts with user devices rather than physical sensors or dedicated hardware components. However, the platform must support:

- **Desktops/Laptops** (Windows, macOS, Linux) – For full functionality.
- **Mobile Devices** (Android, iOS) – Responsive UI optimized for smaller screens.
- **Web Servers** – Hosting the backend and serving requests from user devices.
- **Database Servers** – Storing product data, user preferences, and historical price tracking.
- **APIs from E-commerce Websites** – Fetching real-time product details, prices, and availability.

3.1.3 Software Interfaces

The shopping assistant platform will interact with multiple external software components, including:

- **E-commerce APIs & Web Scrapers** – Collecting data from platforms like Amazon, Flipkart, etc.
- **Database Management System (DBMS)** – Storing product information, user data, and purchase history.
- **AI Recommendation Engine** – Generating personalized suggestions based on user behavior.
- **Web Frameworks (e.g., React, Django, Flask, Node.js, etc.)** – Powering the frontend and backend.
- **Authentication Services (OAuth, Firebase, etc.)** – Managing user sign-ins securely.
- **Payment Integration (if applicable in the future)** – Supporting transactions if users buy directly from the platform.

3.2 Functional Requirements

F1: Real-time Data Aggregation

- The system shall continuously fetch product details (prices, availability, and user reviews) from multiple e-commerce platforms using APIs, web scraping, or data feeds.
- The system shall update product information dynamically, ensuring minimal delay between data collection and display to maintain accuracy.

F2: Product Search and Comparison

- The system shall allow users to search for products across multiple e-commerce platforms through a single search bar, providing relevant results efficiently.
- The system shall display product details, including price, features, and ratings, in a structured format to help users easily compare options.
- The system shall enable users to refine search results using filters (e.g., price range, brand, rating) and sort them based on relevance, lowest price, or highest rating.

F3: AI-Driven Product Matching

- The system shall use AI algorithms (e.g., NLP and computer vision) to analyze product names, descriptions, images, and specifications to identify identical or similar products across different platforms.

F4: Personalized Recommendations

- The system shall track user interactions, including past searches, viewed products, and purchase history, to understand preferences and shopping patterns.
- The system shall update and display recommended products in real-time on the user dashboard, ensuring relevance based on recent activities and preferences.

F5: User Account Management

- The system shall allow users to create and manage accounts, including features such as registration, login, profile updates, and password recovery.

F6: Filtering and Sorting

- The system shall allow users to refine search results using multiple filters, such as price range, brand, ratings, category, and availability.
- The system shall provide sorting functionalities, including price (low to high, high to low), best-rated products, most reviewed items, and newest arrivals.

F7: Price History and Trend Analysis

- The system shall record historical pricing data for products and present trend analysis, enabling users to track price changes over time.

F8: Notification and Alert System

- The system shall continuously monitor tracked products for significant price changes or new deals and trigger alerts when predefined conditions are met.
- The system shall send alerts via multiple channels, including email, push notifications, and , allowing users to stay informed in real-time.

F9: Responsive User Interface

- The system shall provide an intuitive and responsive graphical interface that supports both desktop and mobile device interactions, including touch-screen navigation and interactive menus.

F10: Integration with External APIs

- The system shall establish secure interfaces with external e-commerce APIs or web scraping modules to retrieve accurate product information and ensure timely updates.

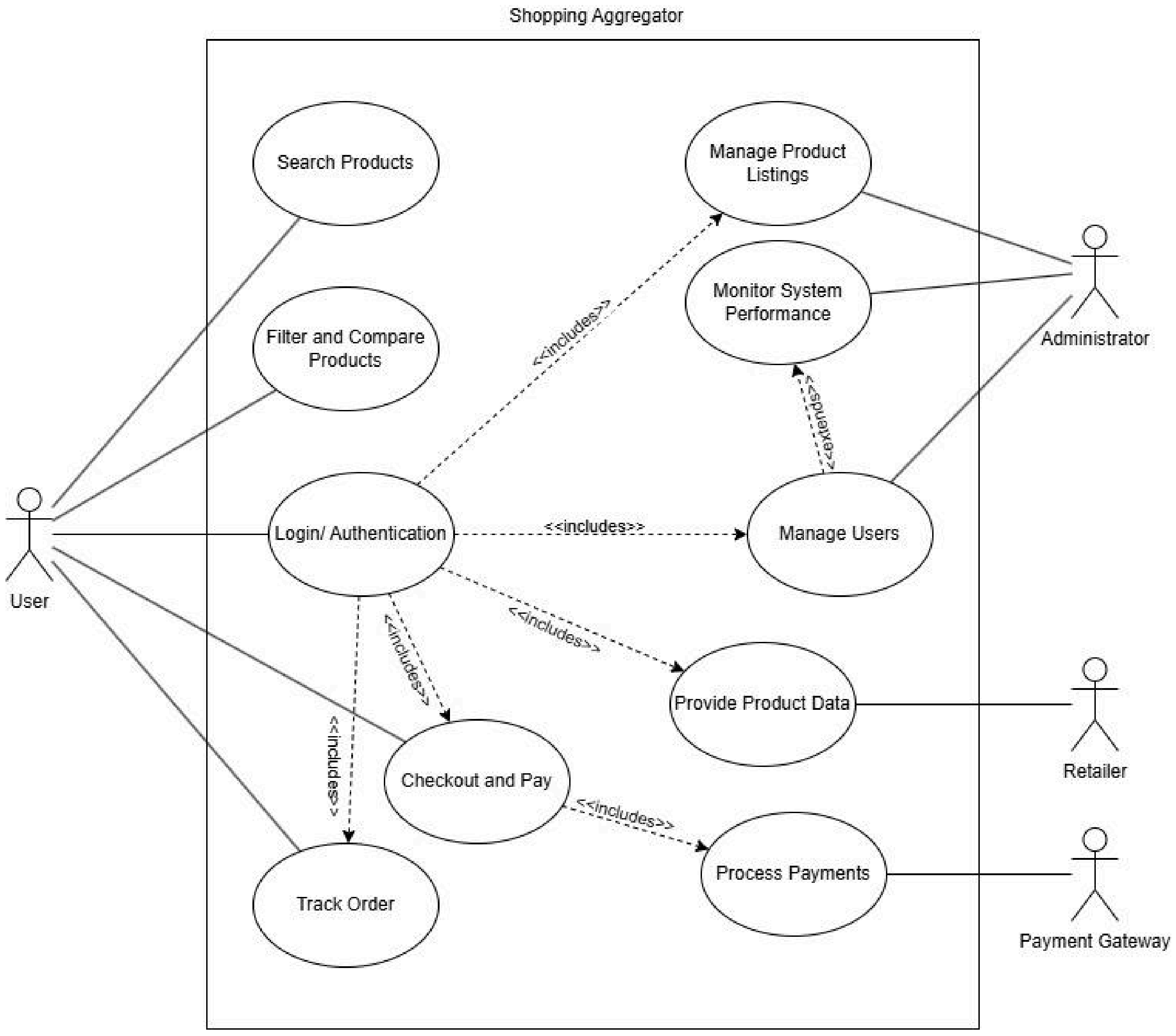
F11: Data Security and Privacy

- The system shall implement security protocols to protect user data and ensure compliance with data privacy regulations. This includes secure data transmission and proper authentication mechanisms.

F12: Administrative Reporting and Analytics

- The system shall include an administrative dashboard that provides analytics on user behavior, system performance, and product trends, enabling continuous improvement of the platform.

3.3 Use Case Model



3.3.1 Use Case #1: Login/Authentication (U1)

Author: [Author Name]
Purpose: Allows users (buyers, retailers, and administrators) to securely log into the system.
Requirements Traceability: F1
Priority: High
Preconditions: The user must be registered in the system.
Postconditions: The user is successfully authenticated and redirected to their respective dashboard.

Actors: User (Buyer, Retailer, Administrator)
Extends: None

Flow of Events:

1. **Basic Flow:**
   - The user enters their email/username and password.
   - The system verifies the credentials.
   - Upon successful authentication, the user is redirected to their dashboard.
2. **Alternative Flow:**
   - If the credentials are incorrect, the system displays an error message and allows the user to retry.
   - If the user forgets their password, they can use the "Forgot Password" feature.
3. **Exceptions:**
   - If the authentication server is down, login is temporarily unavailable.
   - If multiple failed attempts occur, the system may lock the account for security reasons.
     Includes: None
     Notes/Issues: *Multi-factor authentication (MFA) should be implemented for additional security.*

3.3.2 Use Case #2: Search Products (U2)

Author: [Author Name]
Purpose: Allows users to search for products available in the shopping aggregator system.
Requirements Traceability: F1
Priority: High
Preconditions: The user must have access to the system.
Postconditions: The system displays the list of products matching the search criteria.
Actors: User
Extends: None

Flow of Events:

1. **Basic Flow:** The user enters a keyword in the search bar and submits the request.
2. **Alternative Flow:** If no products match the search, the system displays a “No products found” message.
3. **Exceptions:** If the database is down, the system notifies the user of the temporary issue.
   Includes: None
   Notes/Issues: *The search should support filters and sorting for better usability.*

3.3.3 Use Case #3: Filter and Compare Products (U3)

Author: [Author Name]
Purpose: Allows users to apply filters and compare different products.

Requirements Traceability: F2
Priority: High
Preconditions: The user must have access to the system.
Postconditions: The user can view products based on selected filters and compare them.
Actors: User
Extends: None

Flow of Events:

1. **Basic Flow:** The user selects filters (e.g., price, brand, rating) and views the refined results.
2. **Alternative Flow:** If no products match the filters, the system displays an appropriate message.
3. **Exceptions:** If the filtering module fails, default product listings are displayed.
   Includes: None
   Notes/Issues: *Ensure optimized filtering for large datasets.*

3.3.4 Use Case #4: Checkout and Pay (U4)

Author: [Author Name]
Purpose: Allows users to complete their purchases securely.
Requirements Traceability: F3
Priority: High
Preconditions: The user must have products in their cart.
Postconditions: The order is placed, and payment is processed.
Actors: User, Payment Gateway
Extends: None

Flow of Events:

1. **Basic Flow:** The user proceeds to checkout, selects a payment method, and confirms the order.
2. **Alternative Flow:** If payment fails, the user is prompted to try again or use a different method.
3. **Exceptions:** If the payment gateway is down, the user is informed, and the transaction is halted.
   Includes: Process Payments
   Notes/Issues: *Secure payment methods must be ensured.*

3.3.5 Use Case #5: Track Order (U5)

Author: [Author Name]
Purpose: Allows users to track the status of their orders.
Requirements Traceability: F4
Priority: Medium
Preconditions: The user must have placed an order.

Postconditions: The system displays the latest tracking status.
Actors: User
Extends: None

Flow of Events:

1. **Basic Flow:** The user navigates to the order tracking section and views the order status.
2. **Alternative Flow:** If tracking details are unavailable, the system informs the user.
3. **Exceptions:** If the order tracking system fails, an error message is displayed.
   Includes: None
   Notes/Issues: *Integration with shipping partners may be required.*

3.3.6 Use Case #6: Manage Product Listings (U6)

Author: [Author Name]
Purpose: Allows administrators to manage product listings in the system.
Requirements Traceability: F5
Priority: High
Preconditions: The administrator must be logged in.
Postconditions: The product listings are updated accordingly.
Actors: Administrator
Extends: None

Flow of Events:

1. **Basic Flow:** The administrator adds, updates, or removes product listings.
2. **Alternative Flow:** If invalid data is entered, the system prompts for corrections.
3. **Exceptions:** If the system crashes, changes may not be saved.
   Includes: None
   Notes/Issues: *Data validation should be enforced.*

3.3.7 Use Case #7: Provide Product Data (U7)

Author: [Author Name]
Purpose: Allows retailers to supply product information to the system.
Requirements Traceability: F6
Priority: High
Preconditions: The retailer must be authenticated.
Postconditions: The product data is stored successfully.
Actors: Retailer
Extends: None

Flow of Events:

1. **Basic Flow:** The retailer submits product details, including price, description, and images.

2. **Alternative Flow:** If some details are missing, the system prompts for additional input.
3. **Exceptions:** If the database is down, the submission is not processed.
   Includes: None
   Notes/Issues: *Ensure proper data formatting and validation.*

3.3.8 Use Case #8: Process Payments (U8)

Author: [Author Name]
Purpose: Allows payment processing through the payment gateway.
Requirements Traceability: F7
Priority: High
Preconditions: The user must have selected a payment method.
Postconditions: The payment is processed successfully.
Actors: Payment Gateway
Extends: None

Flow of Events:

1. **Basic Flow:** The system communicates with the payment gateway and processes the transaction.
2. **Alternative Flow:** If payment details are incorrect, the transaction is declined.
3. **Exceptions:** If the payment gateway is down, the user is informed of the issue.
   Includes: None
   Notes/Issues: *Secure payment transactions must be ensured.*

3.3.9 Use Case #9: Monitor System Performance (U9)

Author: [Author Name]
Purpose: Allows administrators to monitor system health and performance.
Requirements Traceability: F8
Priority: Medium
Preconditions: The administrator must be logged in.
Postconditions: The system performance data is displayed.
Actors: Administrator
Extends: None

Flow of Events:

1. **Basic Flow:** The administrator accesses the system monitoring dashboard.
2. **Alternative Flow:** If the monitoring tool fails, the system displays an error message.
3. **Exceptions:** If the system is under heavy load, performance data may be delayed.
   Includes: None
   Notes/Issues: *Performance logs should be stored securely.*

3.3.10 Use Case #10: Manage Users (U10)

Author: [Author Name]
Purpose: Allows administrators to manage user accounts.
Requirements Traceability: F9
Priority: High
Preconditions: The administrator must be logged in.
Postconditions: User accounts are updated successfully.
Actors: Administrator
Extends: None

Flow of Events:

1. **Basic Flow:** The administrator adds, updates, or removes user accounts.
2. **Alternative Flow:** If invalid data is entered, the system prompts for correction.
3. **Exceptions:** If the system crashes, changes may not be applied.
   Includes: None
   Notes/Issues: *Role-based access control should be implemented.*

4. Other Non-functional Requirements

4.1 Performance Requirements

- **P1**: The system must authenticate users within **2 seconds** to ensure smooth login and quick access to features.
- **P2**: Product searches and filtered results must load within **2 seconds** to enhance usability and satisfaction.
- **P3**: Price and inventory data must refresh every **15 minutes** to provide accurate and up-to-date information.
- **P4**: Notifications for product availability, price drops, or offers must be delivered within **5 seconds** of the event.
- **P5**: The system must handle up to **10,000 concurrent users** during peak usage without performance degradation.
- **P6**: Payment processing must complete within **3 seconds** for secure and efficient checkout experiences.
- **P7**: The platform must maintain **99.9% uptime**, ensuring minimal disruption to services except for scheduled maintenance.

4.2 Safety and Security Requirements

1. **User Authentication**

- Implement robust authentication with **multi-factor authentication (MFA) and** enforce strong password policies.
- Require users to register with verified email addresses to access personalized features.

2. **Data Protection Compliance**

- Adhere to *GDPR* and *CCPA*, encrypting stored data with *AES-256* and transmitted data with *TLS 1.3*.
- Allow users to access, update, or delete their data upon request and implement data minimization principles.

3. **Secure Data Transmission**

- Encrypt all communications between the platform and users using SSL/TLS protocols to ensure confidentiality and integrity.
- Regularly update security protocols to address emerging vulnerabilities.

4. **Third-Party Data Restrictions**

- Only share user data with third-party services (e.g., e-commerce APIs or payment gateways) when explicitly authorized by users.
- Communicate clear data-sharing policies during account registration and provide opt-in/out options.

5. **System Resilience**

- Protect the system from SQL injection, cross-site scripting (*XSS*), and distributed denial-of-service (DDoS) attacks.
- Implement automated daily backups and a disaster recovery plan to restore operations within 30 minutes.

6. **Safety Certifications**

- Aim for *ISO/IEC 27001* certification to ensure compliance with global security standards and enhance stakeholder trust.

4.3 Software Quality Attributes

4.3.1 Reliability

- Ensure **99.9% uptime** by using load balancers, server redundancy, and database replication to handle failures.
- Real-time monitoring should immediately notify administrators of service disruptions or critical system errors.

4.3.2 Adaptability

- Design the system to support integration with additional e-commerce platforms, payment methods, or new features without major rework.
- Use API-driven architecture for seamless integration and modular design for future scalability.

4.3.3 Usability

- Provide an intuitive interface that allows users to perform key actions (e.g., searches, purchases) in **three clicks or less**.
- Adhere to **WCAG 2.1 Level AA** accessibility standards for inclusivity.
- Conduct usability testing to refine navigation and improve user satisfaction.

4.3.4 Maintainability

- Structure the codebase using clean coding practices and modular architecture to simplify debugging and updates.
- Use version control (e.g., Git) to track changes and enable rollback if issues arise.
- Ensure critical issues can be fixed and deployed within **4 hours**.

4.3.5 Portability

- Optimize the platform for all major devices, including desktops, tablets, and mobile phones.
- Ensure compatibility with major browsers like Chrome, Firefox, Safari, and Edge through responsive web design principles.

4.3.6 Robustness

- Implement comprehensive error handling to prevent crashes and maintain functionality during unexpected scenarios.
- Use database transaction rollback mechanisms to avoid data corruption.
- Test the system under stress and fault conditions to ensure it can handle high loads and failures gracefully.

5. Other Requirements

5.1 Database Requirements

We will use a relational database (e.g., MySQL or PostgreSQL) to store user profiles, product data, search history, and transactions. Queries will be optimized to ensure response times under 2 seconds. Daily automated backups with a 30-day retention period will be implemented, and ACID compliance will ensure data consistency and integrity.

5.2 Internationalization Requirements

Our platform will support at least five languages and provide daily currency conversion updates through APIs. Regional formatting for dates, times, and numbers will be implemented to cater to a global user base.

5.3 Legal Requirements

We will comply with **GDPR** and **CCPA** regulations, ensuring user data is collected, stored, and processed securely. Users will have the ability to access, modify, or delete their personal data, and regional tax regulations will be incorporated.

5.4 Reuse Objectives

We are designing modular components, such as the recommendation engine and payment integration modules, to enable their reuse in future projects. Proper documentation will ensure these components can be easily integrated into other systems.

5.5 Accessibility Requirements

Our platform will adhere to **WCAG 2.1 Level AA** standards to ensure inclusivity for all users. Features like screen reader support, keyboard navigation, and high-contrast mode will be included.

5.6 Deployment and Scalability Requirements

We plan to deploy the platform on cloud services (e.g., AWS or Azure) with auto-scaling capabilities to handle traffic variations and ensure a seamless user experience.

Appendix A – Data Dictionary

Name	***Type***	***Description***	***Possible Values/Format***

User_ID	*Integer*	*Unique identifier for each user*	*Auto-incremented integer*
Username	*String*	*Customer’s chosen username*	*Alphanumeric (max 20 characters)*
Email	*String*	*Customer’s email address*	*Valid email format*
Password	*String*	*Hashed password for account security*	*Encrypted string*
Product_ID	*Integer*	*Unique identifier for each product*	*Auto-incremented integer*
Product_Name	*String*	*Name of the product*	*Text (max 100 characters)*
Price	*Float*	*Current product price*	*Currency format (e.g., 29.99)*
Retailer	*String*	*Name of the retailer*	*Text*
Rating	*Float*	*Average customer rating for the product*	*Range from 0.0 to 5.0*
Deal_Alert_Threshold	*Float*	*Price threshold set by a user to trigger a deal notification*	*Currency format*
Price_History	*Array/Record*	*Historical pricing data for a product*	*Array of floats with timestamps*
Search_Query	*String*	*User-entered search term*	*Text*
Filter_Criteria	*Object*	*Criteria used to filter product search results*	*JSON object (e.g., price range, brand, rating)*
Recommendation_List	*Array*	*List of product IDs recommended to the user*	*Array of integers (Product_IDs)*
Notification	*Object*	*Details of alerts or messages sent to the user*	*JSON object (e.g., message, timestamp)*
User_Preferences	*Object*	*Settings for personalized recommendations*	*JSON object (e.g., preferred categories, price range)*

Session_State	*String*	*Current state of the user session*	*"Active", "Inactive", "Expired"*
Timestamp	*Date/Time*	*Date and time for transactions or system events*	*Standard datetime (YYYY-MM-DD HH:MM:SS)*
API_Response	*JSON*	*Data received from external e-commerce API requests*	*JSON format*
API_Status	*Integer/String*	*Status code returned from API requests*	*HTTP status codes (e.g., 200, 404, 500)*
MAX_RESULTS_PER_PAGE	*Constant*	*Maximum number of products displayed per search result page*	*Integer (default: 20)*
MIN_RATING	*Constant*	*Minimum rating filter for product searches*	*Float (range: 0.0 to 5.0)*

Appendix B - Group Log

Date	**Attendees**	**Discussion Topics**	**Time Spent**	**Assigned Tasks**

03.03.25	*YUVAN REDDY AASRITH REDDY KSHITIJ KARMUSE CHANDANA GOUD JOSYULA MITHIL*	*SRS sections assigned*	*120-150 min*	*1.1-1.6: JOSYULA MITHIL 2.1-2.4:YUVAN REDDY,KSHITIJ KARMUSE 3.1-4.3 : AASRITH REDDY,CHANDANA GOUD*
10.03.25	*YUVAN REDDY AASRITH REDDY KSHITIJ KARMUSE CHANDANA GOUD JOSYULA MITHIL*	*Functional Non Functional Requirements UseCases (diagram) Other Requirements*	*140-160 min*	
12.03.25	*YUVAN REDDY AASRITH REDDY KSHITIJ KARMUSE CHANDANA GOUD JOSYULA MITHIL*	*Final proofreading Submission*	*90 min*	