Software Requirements Specification

for

Deal Buddy

Version 1.0 approved

Prepared by :- **Dheerain-22103013**

Shreya-22103006

Gauri-22103012

Darren-22103019

Punjab Engineering College(PEC)

Date of Creation-18/09/2024

TABLE OF CONTENTS

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The product specified in this document is **Deal Buddy**, a Price Monitoring and Notification System for e-commerce platforms. The current version covers the core features such as web scraping for price tracking, price drop notifications, and price comparison across different platforms. This SRS focuses on the initial release of the system, covering all essential functionalities. Future updates may extend to additional platforms and mobile apps, as outlined in the Future Scope section.

1.2 Document Conventions

- **Bold** text is used for major components and modules.
- **Italic** text indicates features in the proposed future scope of the project.
- High-level requirements are assumed to apply to detailed sub-requirements unless explicitly stated otherwise.
- Requirements labeled as "REQ-XX" are functional requirements.

1.3 Intended Audience and Reading Recommendations

This document is designed for the following audiences:

- **Developers**: To gain a comprehensive understanding of the system's technical architecture, components, and functionalities, enabling efficient implementation.
- **Project Managers**: To monitor and guide the development process, ensuring alignment with user requirements, timelines, and project goals.
- **End Users**: To acquire an overview of the system's operations and the value it delivers, ensuring ease of use and clarity on key features.
- **Testers**: To derive test scenarios and cases from the requirements, ensuring that all features are thoroughly validated and work as expected.
- **Technical Writers**: To use the information provided as a foundation for creating detailed user guides, technical documentation, and support materials.

It is recommended to begin with an overview of the system in the Introduction section, followed by the Functional Requirements for a detailed breakdown of system components. Readers can refer to the Future Scope section for upcoming features and enhancements.

1.4 Product Scope

Deal Buddy is a web-based Price Monitoring and Notification System designed to help consumers track prices, receive price drop alerts, and compare product prices across e-commerce platforms. The system provides users with real-time information about product prices, historical trends, and personalized notifications, enabling them to make informed

purchasing decisions. The long-term goal is to expand its capabilities to mobile platforms and additional e-commerce sites like Flipkart and Myntra.

1.5 References

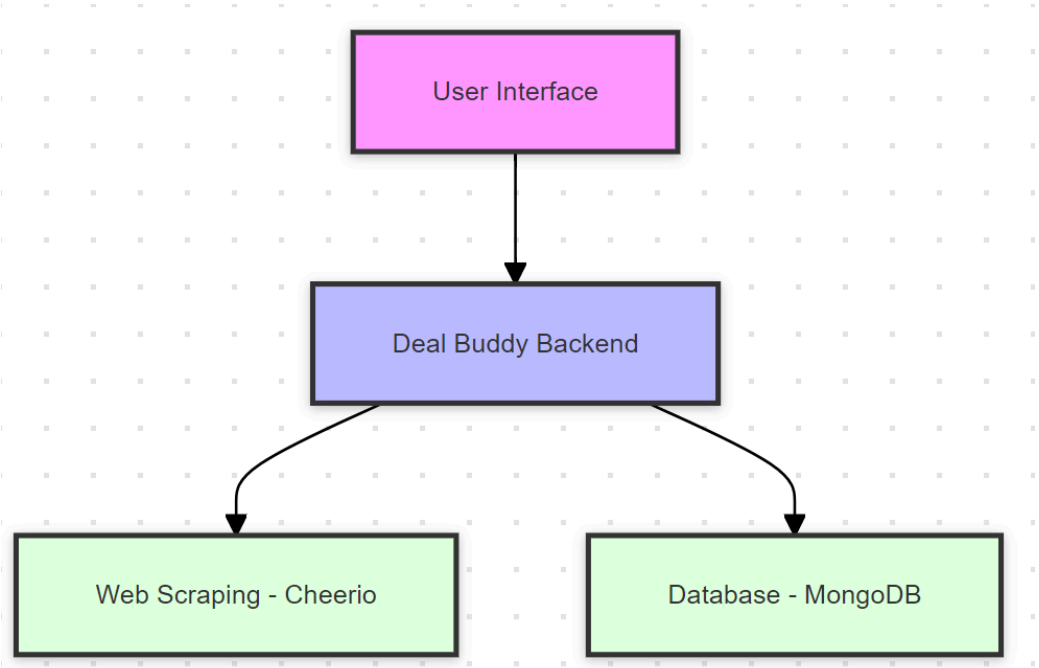
- **Next.js Documentation**: https://nextjs.org/docs
- **MongoDB Documentation**: https://www.mongodb.com/docs
- **Cheerio API Reference**: https://cheerio.js.org
- **Nodemailer Documentation**: https://nodemailer.com/about
- **Cron Jobs Guide**: https://crontab.guru
- **IEEE** Software Requirements Specification Template by Karl E. Wiegers (1999).

2. Overall Description

2.1 Product Perspective

Deal Buddy is a new, self-contained system designed to monitor e-commerce prices, notify users of price drops, and compare prices across multiple platforms. It operates independently, though it interacts with various e-commerce websites for data collection via web scraping. This system is part of the growing trend of price monitoring tools that aim to provide users with valuable insights into market trends and help them make informed purchasing decisions. The current system focuses on Amazon, with future plans to expand to other platforms like Flipkart, Myntra etc.

Here's a simple diagram showing the system components and interfaces:



2.2 Product Functions

The major functions of **Deal Buddy** are:

- **Price Tracking**: Track real-time prices of products across e-commerce platforms.
- **Price Drop Notifications**: Send personalized alerts when a product price falls below a set threshold.
- **Price Comparison**: Compare prices across multiple online stores.
- **Historical Price Analysis**: Provide users with insights into historical price trends.

2.3 User Classes and Characteristics

The anticipated user classes include:

- **General Users**: Consumers looking to monitor prices and receive notifications about price drops. These users will have basic technical knowledge and access to the web or mobile platforms.
- **Advanced Users**: Users like businesses or price analysts, who might use the system for market research and detailed price tracking.
- **Administrators**: System administrators who will manage the backend operations, user accounts, and ensure that the scraping functions are working correctly.

2.4 Operating Environment

Deal Buddy will operate in the following environments:

- **Frontend**: Accessible via web browsers on desktop and mobile platforms.
- **Backend**: Deployed on cloud services like Vercel, using Next.js framework.
- **Database**: MongoDB will store product data, user preferences, and historical price information.
- **Web Scraping**: The system will rely on the Cheerio library for xml parsing and Axios for fetching data from e-commerce websites, while Bright Data will serve as the proxy service.
- **Notifications**: Email notifications will be sent via Nodemailer, while other channels like SMS or push notifications may be integrated later.

2.5 Design and Implementation Constraints

- **Web Scraping Limitations**: Compliance with website terms of service and geographic restrictions may limit the ability to scrape certain websites. The tool must handle CAPTCHAs and other anti-scraping mechanisms efficiently.
- **Technology Constraints**: Deal Buddy must use Next.js and MongoDB as per the project requirements, meaning future scalability considerations will need to accommodate these technologies.

- **Security Considerations**: Personal data such as email addresses must be securely handled, with encryption and proper storage protocols in place.
- **Compliance with Platforms**: The system must adhere to the rate-limiting policies of platforms like Amazon to avoid being blocked.

2.6 User Documentation

The following user documentation will be provided:

- **User Manual**: A detailed manual explaining how to use the system, covering setup, price monitoring, and notification settings.
- **Online Help**: Integrated help system within the application for quick troubleshooting and guidance.
- **Tutorials**: Video tutorials and written guides to help new users navigate and make the best use of the system.

2.7 Assumptions and Dependencies

- **Assumptions**:
    - E-commerce websites will continue to allow scraping under current rate-limiting policies.
    - Users will primarily use the system on the web, with mobile expansion as a future scope.
- **Dependencies**:
    - The system relies on third-party services such as Bright Data for proxies and MongoDB for data storage.
    - Notifications depend on email providers and future integration of SMS or push notification services.

---

3. External Interface Requirements

3.1 User Interfaces

The user interface for **Deal Buddy** will be designed with simplicity and ease of use in mind. The following are the key elements of the user interface:

- **Landing Page**: It will feature a brief overview of the website, explaining its purpose and functionality through an engaging video tutorial.
- **Product Tracking Page**: Users can add products they want to track, view current and historical prices, and set custom alerts for price drops.
- **Notification Center**: A centralized location for users to manage their price drop notifications, including email settings.

GUI standards will follow modern web design practices with responsiveness to accommodate both desktop and mobile views. The layout will follow a grid-based system (using Tailwind CSS). Error messages will be displayed prominently and will offer suggestions for corrective action. Common keyboard shortcuts will be supported for quicker navigation.

3.2 Hardware Interfaces

Since **Deal Buddy** is a web-based application, it does not have any direct hardware interface requirements. However, the system will require:

- **User Devices**: Devices capable of accessing web browsers, such as desktops, laptops, tablets, or smartphones.
- **Server Hosting**: The backend system will be hosted on cloud infrastructure, which will require virtual servers capable of running Next.js and MongoDB databases.

3.3 Software Interfaces

Deal Buddy interfaces with several software components to perform its operations:

- **Web Browsers**: Compatible with major web browsers like Chrome, Firefox, Safari, and Edge, with HTML5 and CSS3 compliance.
- **Operating System**: The backend is OS-agnostic but will be deployed on cloud platforms like Vercel.
- **Database**: MongoDB will be used for storing user data, price information, and historical records.
- **Web Scraping Tools**: The system will use Cheerio and Axios for scraping e-commerce sites and Bright Data for proxy management.
- **Email Services**: Nodemailer will be used to send price drop notifications via email, interfacing with SMTP or third-party email services (like Gmail or SendGrid).

The system will send and receive JSON objects for communication between the frontend and backend. APIs will be used to transfer data between the web interface and the backend services. The system will also support integration with other e-commerce APIs, if available in the future, for direct data access rather than scraping.

3.4 Communications Interfaces

Deal Buddy requires several communication protocols for data transfer and notifications:

- **HTTP/HTTPS**: All communications between the frontend and backend, as well as between the backend and external e-commerce platforms, will use HTTPS for secure data transfer.
- **Email**: SMTP will be the primary protocol for email notifications. All emails will be secured via TLS to ensure data protection.
- **Webhooks**: Webhooks may be implemented in the future to notify users in real-time about critical price changes.

4. System Features

4.1 Price Monitoring and Scraping

4.1.1 Description and Priority

This feature allows users to monitor prices of products from various e-commerce platforms in real-time. The system will scrape the prices at regular intervals and store them for historical comparison.
Priority: High

4.1.2 Stimulus/Response Sequences

1. **User Action**: The user adds a product URL to their watchlist. **System Response**: The system confirms the product has been added and begins tracking its price.
2. **User Action**: The system detects a price change during its periodic check. **System Response**: The system stores the new price and updates the price history.

4.1.3 Functional Requirements

- **REQ-1**: The system shall scrape prices from specified e-commerce platforms every 24 hrs.
- **REQ-2**: The system shall notify the user when a price drop is detected for tracked items.
- **REQ-3**: The system must be able to scrape prices from multiple e-commerce platforms.

4.2 Price Drop Notifications

4.2.1 Description and Priority

This feature sends notifications to users when a price drop is detected for products in their watchlist.
Priority: High

4.2.2 Stimulus/Response Sequences

1. **User Action**: The system detects a price drop for a product on the user's watchlist. **System Response**: The system sends an email to the user, notifying them of the new price.

4.2.3 Functional Requirements

- **REQ-1**: The system shall send an email notification to the user within 2 minutes of detecting a price drop.

- **REQ-2**: The system shall allow users to configure notification preferences (email, SMS, or in-app notification)
- **REQ-3:** The system shall support multiple email providers such as Gmail, Outlook, and others for notifications, as well as in-app notifications.

4.3 Price Comparison

4.3.1 Description and Priority

This feature allows users to compare prices of the same product across different e-commerce platforms.
Priority: High

4.3.2 Stimulus/Response Sequences

1. **User Action**: The user selects a product for comparison. **System Response**: The system retrieves the current prices of the product from all supported platforms and displays them side by side.

4.3.3 Functional Requirements

- **REQ-1**: The system shall retrieve current prices of a product from all supported e-commerce platforms.
- **REQ-2**: The system shall display a price comparison with the prices from different platforms.
- **REQ-3**: The system shall show the lowest price of the product up to date.

4.4 Historical Price Analysis

4.4.1 Description and Priority

This feature provides users with historical price data for a product to help them make informed purchasing decisions.
Priority: Medium

4.4.2 Stimulus/Response Sequences

1. **User Action**: The user views the price history of a product. **System Response**: The system displays a graph showing the price trend of the product over time.

4.4.3 Functional Requirements

- **REQ-1**: The system shall store and display historical price data for up to one year.
- **REQ-2**: The system shall generate price history graphs that update dynamically based on the latest data.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system shall be able to scrape and update product prices every 24 hours.
- The notification system must send alerts within 2 minutes after detecting a price drop.
- The system shall handle up to 10,000 simultaneous users without significant performance degradation.
- The system's response time for user queries should not be more than 5 seconds.

5.2 Safety Requirements

- The system shall ensure data integrity during price updates, avoiding any corruption during scraping processes.
- The system must comply with all relevant e-commerce regulations to prevent fraudulent behavior or data misuse.
- Regular backups of user data should be performed to minimize data loss in case of a system crash.

5.3 Security Requirements

- All user data, including personal details and email addresses, must be encrypted in the database.
- The system must follow GDPR compliance for handling user data and notifications.
- Access to administrative functions shall be restricted to authorized personnel only.
- The system should implement role-based access control to limit user access based on privilege levels.

5.4 Software Quality Attributes

- **Reliability**: The system shall have 99.9% uptime to ensure constant price monitoring and notifications.
- **Maintainability**: The codebase should be modular and well-documented to facilitate updates and troubleshooting.
- **Usability**: The user interface should be intuitive and require minimal learning for new users.
- **Interoperability**: The system must support integration with various e-commerce platforms and operate across multiple operating systems.
- **Scalability**: The system architecture must support easy scaling to accommodate increased users and platforms.

5.5 Business Rules

- Only registered users shall be able to track prices and receive notifications.

- Prices must be sourced only from authorized e-commerce platforms to ensure accuracy and legality.

6. Other Requirements

6.1 Database Requirements

- The system must utilize MongoDB as the primary database to store user information, product details, and historical price data.
- The database must be designed for scalability to handle a large volume of data from multiple e-commerce platforms.
- It should support efficient queries for price history retrieval, including sorting by date and price trends for each product.

6.2 Internationalization Requirements

- The system must support multiple currencies and convert prices based on the user's geographical location.
- The notification system should be capable of handling various languages, depending on the user's locale.

6.3 Legal and Privacy Requirements

- The system must comply with applicable data protection laws, such as GDPR, ensuring user consent for storing and processing personal data.
- User data, especially price tracking preferences and history, should be encrypted to prevent unauthorized access.

6.4 Reuse Objectives

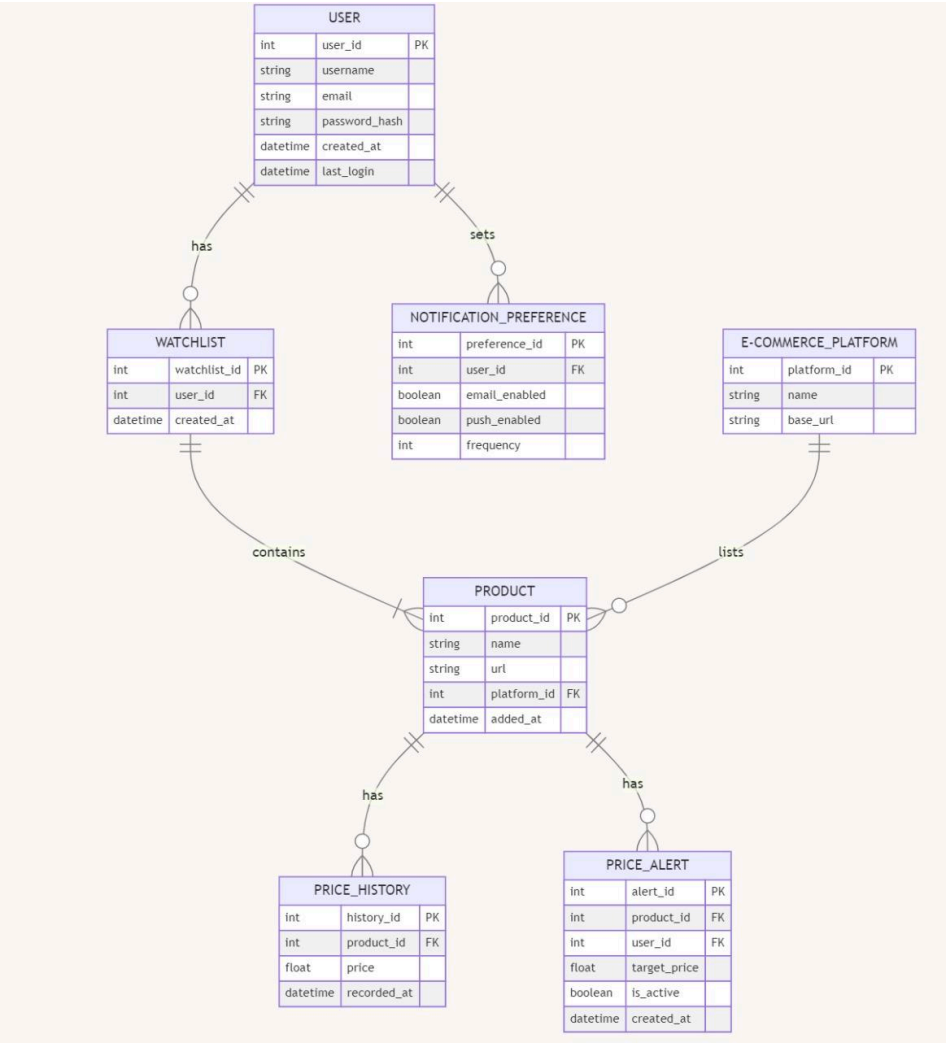
- The system's components (e.g., web scraping modules, notification services) should be modular and reusable across similar projects or for potential future expansion.

Appendix A: Glossary

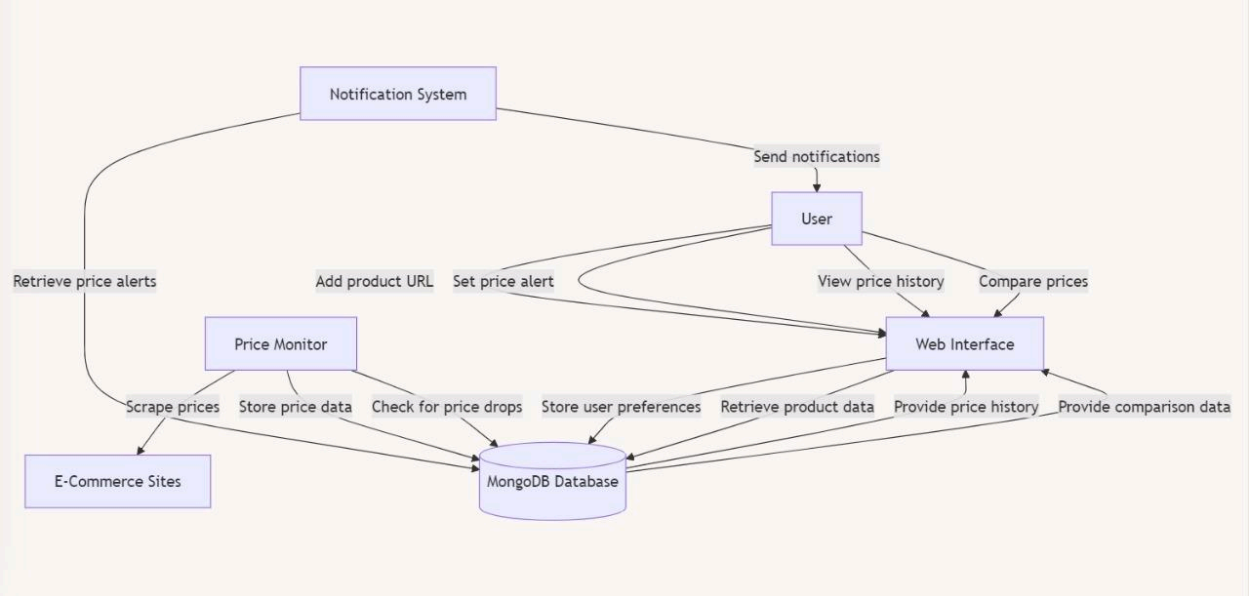
- **Deal Buddy**: The project name for the price monitoring and notification system.
- **MongoDB**: A NoSQL database used for storing large amounts of structured and unstructured data.
- **Cron Jobs**: Scheduled tasks that execute at specified intervals, often used to automate tasks like price checks.
- **Next.js**: A React framework used for building the frontend of the application.
- **Nodemailer**: A Node.js library used to send notification emails to users.
- **GDPR**: General Data Protection Regulation, a law governing data privacy and security for individuals within the European Union.

Appendix B: Analysis Models

- **Data Flow Diagram (DFD)**: A diagram detailing the flow of data between different system components, including user input, web scraping modules, and the database.
- **Entity-Relationship Diagram (ERD)**: A diagram illustrating the relationships between different data entities such as users, products, and price histories.



- **Class Diagram**: A high-level representation of the system's objects and their relationships, covering classes such as User, Product, and Notification.



Appendix C: To Be Determined List

- **TBD-1**: Finalize the communication protocol between the price monitoring backend and the notification service.
- **TBD-2**: Determine the best approach for handling multi-currency support in price comparison across platforms.
- **TBD-3**: Establish the user roles and privileges for accessing the admin panel for monitoring price history and user notifications.