University of Worms, Applied Science in Informatics

SOFTWARE ARCHITECTURE DOCUMENT
WS 2024/2025
(Course 506)

with Prof. Dr. Volker Schwarzer

Von: Büsranur Hinislioglu (678055, inf3930@hs-worms.de)

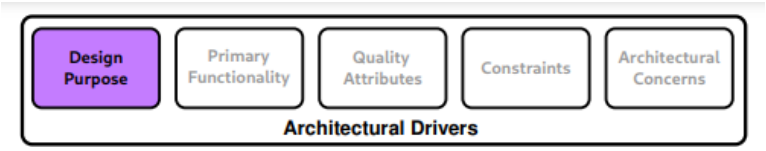
Table of Contents

Project Description: Christmas Gift Planning Website

The Christmas Gift Planning Website is designed to help families and friends plan gifts more easily. Users can create wishlists for the people they want to give gifts to. They can assign specific gifts to recipients and compare prices to find the best deals. The website shows discounts and suggests alternative products to save money and improve the shopping experience. It also includes built-in chat and email features so users can share ideas and coordinate their plans. This platform makes holiday gift planning more organized, affordable, and enjoyable for everyone.

Chapter 1: Collect ASRs (Architecturally Significant Requirements)

1.1 Design Purpose



ID	Category	Description
DP1	Purpose	The purpose of the project is to create a web-based platform for desktop users. This platform will make the gift-giving process easier and more enjoyable for families and friends. It will include features like Wishlist management to organize gift ideas. Users can compare prices to find the best shopping options. Additionally, the platform will offer communication tools such as chat and email to help users coordinate and share ideas.
DP2	Goals	The project will have an intuitive and user-friendly web interface for gift planning. Users will be able to create and manage wishlists easily. The platform will include price comparison and alternative product suggestions to optimize costs. Integrated chat and email features will enable seamless collaboration and communication.
DP3	Schedule	The university requests the release as fast as possible.

1.2 Primary Functionality

ID	CATEGORY(Requierement)	DESCRIPTION	Motivation (Reason)	PRIORITY
FR1	User Account Management	The system helps users create accounts. They can enter their email, username, and password to register. Users can log in and update their profiles anytime.	The user can create an account and log in safely. The user can manage their profile easily when needed.	High
FR2	Price Comparison	The system allows users to check prices from different sellers. It shows discounts and sales for each product so users can compare them.	The user can see prices and discounts to buy things at the best price.	High
FR3	Wishlist Management	The system lets users write down the wishes and preferences of the person who will receive the gift. Users can also update these wishes anytime.	The user can organize gift ideas better by adding wishes for each person in one place.	High
FR4	Security Features	The system protects user accounts with strong security measures, including password hashing and multi-factor authentication (MFA). MFA combines a password and a one-time code sent to a mobile device or email, ensuring account security even if the password is compromised.	The user can feel safe knowing their information and account are private and cannot be accessed by others.	High
FR5	Gift Assignment and Duplicate Prevention	The system makes sure the same gift is not given to one person twice in the same group. It shows a warning if a gift is already assigned. For example, In this way, grandparents are prevented from buying the same gift for their grandchildren.	The user can avoid giving the same gift twice to the same person and prevent confusion.	Medium
FR6	Alternative Product Suggestions	The system suggests other items if the chosen product is out of stock or if there are better deals. It shows similar options to users.	The user can pick another product easily if the first choice is not available or if better options are found.	Medium

FR7	Communication Tools	The system has chat and email tools so users can talk to others in their group. They can share ideas and updates quickly.	The user can work with group members easily by sharing ideas and plans through the system.	Medium
FR8	Personalization	The system allows users to customize their interface with holiday-themed backgrounds, color schemes, and fonts. Users can select pre-designed themes or create their own for a more festive experience.	The user can enjoy a personalized and visually engaging platform that aligns with the holiday spirit, improving user satisfaction.	Low

1.3 Use Case View

Use case view for end users. All Primary Functionalities are stated.

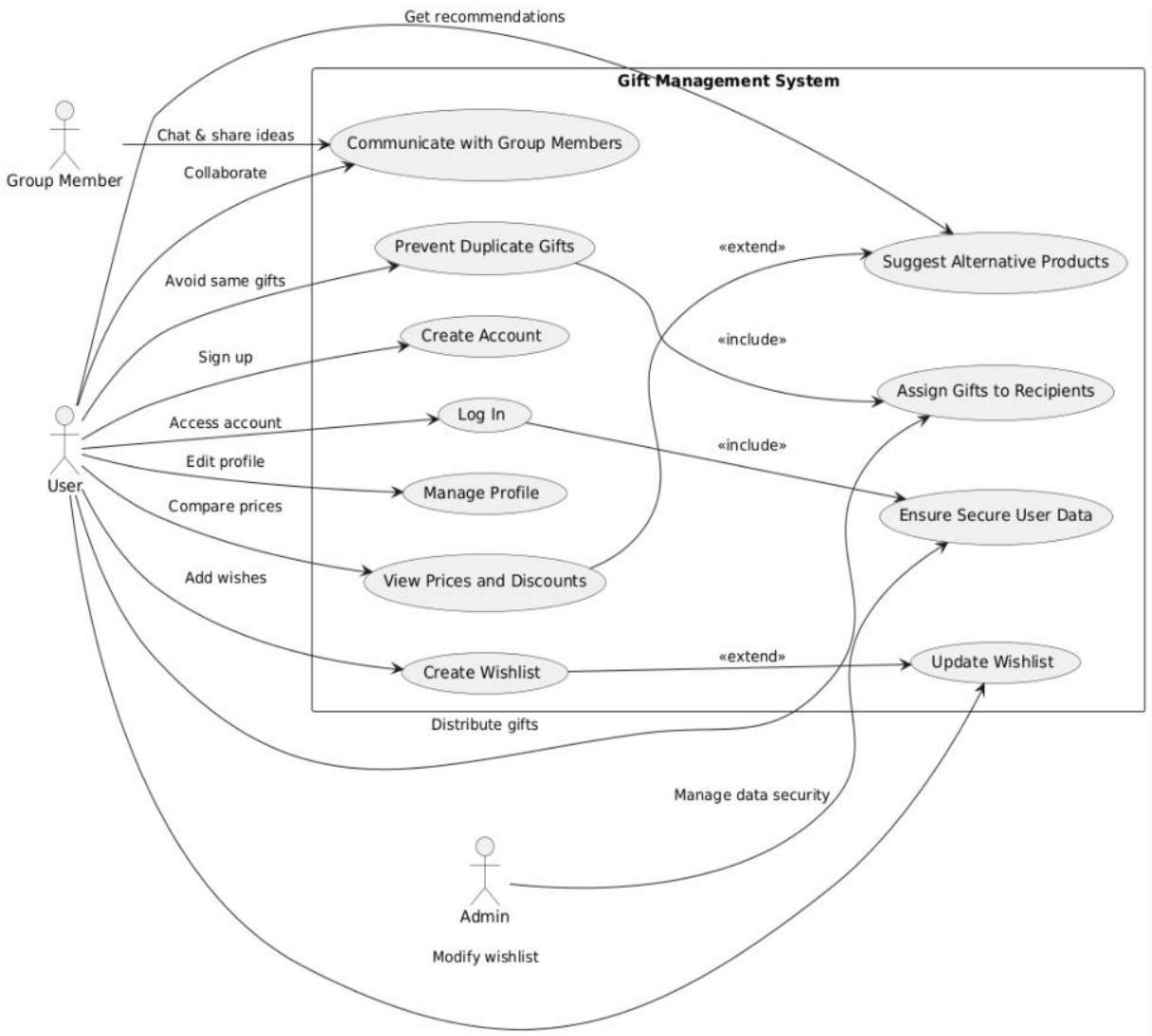


Figure 1

Das Diagramm zeigt drei Akteure: User, Group Member und Admin.

User: Kann sich registrieren, einloggen, Wunschlisten erstellen, Preise vergleichen und Geschenke zuweisen.

Erweiterte Funktionen: Das System schlägt alternative Produkte vor und verhindert doppelte Geschenke.

Admin: Bearbeitet Wunschlisten und sorgt für Datensicherheit.

Group Member: Kommuniziert mit anderen Mitgliedern und teilt Ideen.

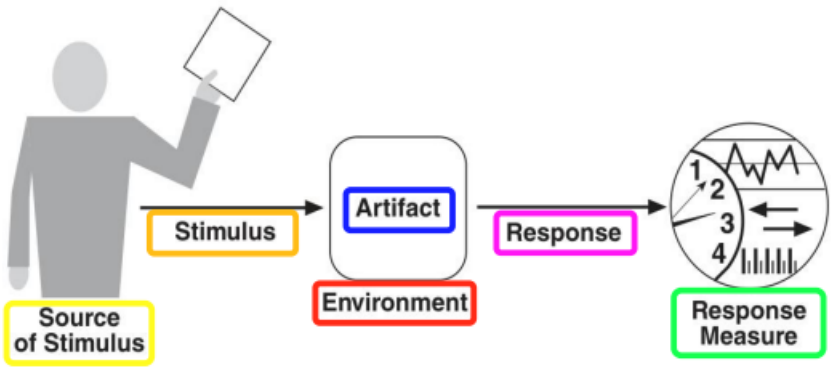
1.4 Quality Attributes

Priority (Combination) =
(Importance(by customer(stakeholder)**),Technical Risk(**by architect**)) => (L/M/H)**

- High (H) and Medium (M) priorities reflect how critical the attribute is from different stakeholder perspectives.

ID	Quality Attribute	Reasoning	Associated Driver/ usecase	Priority (customer,architect) (stakeholder, architect)
QA1	**Usability** End users (individual users and families) UI/UX designers Developers	The platform must be intuitive and user-friendly to allow users to create wishlists, compare prices, and use communication tools effectively.	Users interacting with wishlist, price comparison, and chat/email functionalities. **FR2 , FR4, FR8**	(H, M) **Reason:** H / Essential for customer satisfaction and platform adoption. M / Design flexibility for usability enhancements may not be complex.
QA2	**Performance** Developers, System administrators Performance test specialists	The website must provide quick response times, especially for price comparisons and search functionalities.	Key operations like price comparison, Wishlist filtering, and real-time updates for users. **FR2, FR4, FR5**	(H, H) **Reason**: H / Crucial for a seamless and efficient user experience. H / Requires optimized architecture for speed.
QA3	**Scalability** Developers Cloud architects System administrators	The system should handle increased user traffic during high-demand periods like holiday seasons.	Support for peak traffic and concurrent users, especially during gift-planning times. **FR2, FR4, FR6**	(H, H) **Reason:** H / Ensures reliability during high traffic seasons. H / Requires scalable infrastructure and architectural planning.
QA4	**Security** Security experts System administrators Legal and compliance teams	Ensure data protection for user accounts, Wishlists, and communication, preventing unauthorized access or breaches.	Login management, Wishlist data security, and communication encryption. **FR1,FR6,FR8**	(H, H) **Reason:** H / Critical for user trust and data protection. H / Requires robust security measures integrated into the architecture.

QA5	**Reliability** System administrators Developers DevOps engineers	The platform should avoid crashes and downtime, especially during critical usage times.	Ensures uninterrupted service for Wishlist management, price comparisons, and communication features. **FR2,FR3,FR4**	(H, H) **Reason:** H / Builds trust and ensures continuous service. H / Demands robust system architecture to minimize failures.
QA6	**Maintainability** Developers System administrators Product owners	The system should be easy to update and expand with new features or external integrations like APIs.	Adding new vendors, improving product suggestions, and modifying algorithms. **FR5,FR8**	(M, M) **Reason:** M / Facilitates adaptability and long-term improvements. M / Requires modular and flexible architecture to simplify changes.
QA7	**Interoperability** E-commerce vendors Developers System administrators	The platform should integrate seamlessly with external APIs for price data, discount updates, and product suggestions.	Integration with e-commerce APIs or other data providers for real-time updates. **FR2,FR5**	(M, H) **Reason:** M / Adds functionality but may not be directly visible to customers. H / Requires proper API handling and architecture for extensibility.



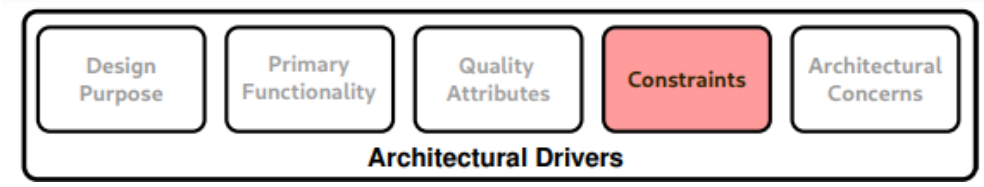
1.5 Quality Scenarios

ID	Quality Attribute	Source of Stimulus	Stimulus	Artifact	Environment	Response	Response Measure	Priority
QAS1	Usability (relevant with ADR8)	End User	Attempts to create or edit a Wishlist (Creating 10 items at 10 seconds)	User Interface (UI)	During normal operations	System responds with clear, intuitive workflows	Tasks are completed within 2 minutes, no errors	High (H, M)
QAS2	Usability (ADR9)	End User	Uses the communication feature (chat/email)	Communication Module	During normal operations	System provides seamless access and interaction	Chat/email features function without delays (Messaging delays above 1 second disrupt user experience.)	High (H, M)
QAS3	Performance (ADR3,ADR4)	End User	Loads price comparison for a Wishlist	Price Comparison Engine	During normal operations	Results displayed quickly	Results appear within 2 seconds	High (H, H)
QAS4	Scalability (ADR1, ADR10, ADR11)	System	High number of concurrent users accessing	Entire System Architecture	Holiday shopping season	System maintains normal performance levels	Handles up to 10,000 users without degradation (Peak holiday traffic requires scalable architecture.)	High (H, H)
QAS5	Scalability (ADR11)	System	Bulks data import from external APIs	External API Integration	During peak operations	System imports data efficiently without impacting users	Completes bulk import within 5 minutes (Importing 1,000 items at 5 seconds each fits the limit.)	Medium (M, H)
QAS6	Security (ADR5, ADR6)	Malicious Actor	Attempts unauthorized access to user data	Authentication System	Under attack	Prevents access and alerts administrators	Unauthorized access attempts are blocked (Intrusion systems act in milliseconds to block attacks.)	High (H, H)

QAS7	Security (ADR5)	End User	Logs in with valid credentials	Authentication System	During normal operations	Grants access securely without delays	Login response within 2 seconds (Authentication systems complete database checks quickly.)	High (H, H)
QAS8	Reliability (ADR7,DR12)	System	Unexpected failure occurs	System Components	Under failure conditions	System recovers and restores operations automatically	Recovery within 30 seconds (Automated failover systems resolve issues rapidly.)	High (H, H)
QAS9	Reliability (ADR1, ADR12, ADR13)	End User	Accesses wishlist during peak usage	Wishlist Module	Holiday shopping season	System provides uninterrupted access	Access with no errors or downtime (High availability ensures uninterrupted access during peak times.)	High (H, H)
QAS10	Maintainability (ADR13)	Developer/ Admin	Updates required for adding new features	System Architecture	During planned maintenance	Updates are implemented without impacting live users	Downtime less than 5 minutes (Modular updates minimize disruption to users.)	Medium (M, M)
QAS11	Interoperability (ADR5)	External System (Online Shop)	Sends request for price data or updates	External API Integration	During normal operations	System (Preisvergleichssystem) processes requests and updates seamlessly	Successful API response rate of 99% (Reliable API integrations ensure data consistency and minimize errors.)	High (M, H)
QAS12	Interoperability (ADR5)	System	Requires integration with new APIs	External API Integration	During planned maintenance	New APIs integrated without affecting existing features	Zero disruption in existing workflows (Planned maintenance avoids affecting current workflows.)	Medium (M, H)

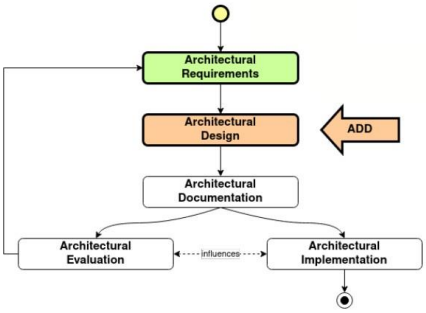
- This section puts the Quality Attributes into measurable scenarios.

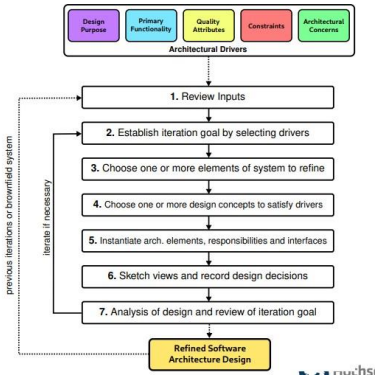
1.6 Constraints



ID	Constraint	DESCRIPTION
CON1 (**Organizational**)	Usability Testing Requirement	The interface must undergo usability testing to ensure it meets the expectations of end users.
CON2 (**Organizational**)	Deadline	The university requests the release as fast as possible.
CON3 (**Technical**)	API Dependency	The platform relies on external APIs for price data and discount updates, requiring stable and reliable integrations.
CON4 (**Technical**)	Scalability During Peak Times	The architecture must handle up to 10,000 concurrent users during the holiday shopping season.
CON5 (**Technical**)	Data Privacy Compliance	The system must comply with GDPR or similar data protection regulations to ensure user data privacy and security.
CON6 (**Technical**)	Limited Server Resources	Hosting and server infrastructure must remain cost-efficient while ensuring reliability and scalability.
CON7 (**Technical**)	Minimal Downtime During Maintenance	System updates and maintenance must not disrupt ongoing user operations, with downtime limited to 5 minutes.
CON8 (**Technical**)	Integration with Standard Authentication	The platform must use a secure and widely accepted authentication system to manage user access and prevent breaches.

Chapter 2: ADD (Attribute Driven Design)

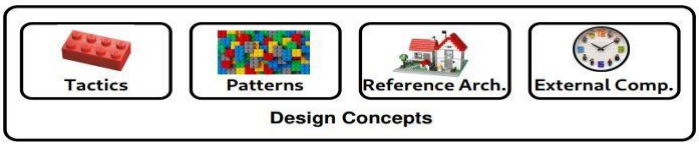


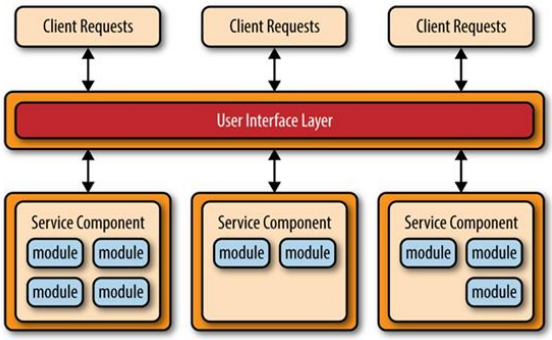


Step 1: Review Inputs
Step 2: Establish Iteration Goal by Selecting Drivers
Step 3: Choose One or More Elements of the System to Refine
Step 4: Choose One or More Design Concepts that satisfy the Selected Drivers
Step 5: Instantiate Arch. Elements, Allocate Responsibilities, and Define Interfaces
Step 6: Sketch Views and Record Design Decisions
Step 7: Analysis of current design and review of iteration goals

Greenfield System: The "Christmas Gift Planning Website" is a **Greenfield System**, meaning it is developed from scratch without dependencies on existing systems or legacy infrastructure. This allows for maximum flexibility to design and implement modern architectures and features tailored to project needs.

Tactic: shows a single QA.
Pattern: shows one or more QAs. In a pattern, LEGO blocks are combined together.





2.1 Architectural Decision Records (ADR)

- This section shows the decision record for the architectural design.

ID – Title :	**DR1 – Microservices Architecture for Main System Structure (Pattern)**
Status:	Approved
Context:	Scalable and Modular System Architecture
Workshop Execution: Who Was Involved?	Developers, System Administrators, Technical Architects
Considered Drivers:	FR2, FR4; QA3, QA5, QA6; CON3; QAS4, QAS9
Decision	• Microservices architecture enables independent scaling, fault isolation, and modularity, ensuring a scalable and adaptable platform. • Each microservice is responsible for a specific function, such as Authentication, Wishlist Management, or Price Comparison, and can be developed, tested, and deployed independently. • Services can scale independently based on demand, allowing features like the Price Comparison Service to handle higher loads during peak seasons. • Failures in one service, such as the Chat Service, do not affect the operation of other services. • New features or bug fixes can be deployed in specific services without disrupting the rest of the system. • Communication between services is managed through HTTP/REST APIs, enabling seamless integration and interaction.
Considered Alternatives:	**Monolithic Architecture:** Rejected because it cannot scale effectively and poses a risk of complete system failure. In monolithic systems, any failure affects the entire application, and scaling the system as a whole is inefficient.
Consequences:	**Enhanced Fault Tolerance**: Only the service with an issue stops working; the rest of the system keeps running. **Scalability and Flexibility**: Each service can be adjusted separately to handle more or less work as needed. **Rapid Deployment**: Changes or fixes can be made to one service at a time without affecting others. **Decoupled Communication**: Services talk to each other using simple, standard methods like HTTP/REST.
Additional Documentation:	**Figure 2**: Microservices Diagram showing service-to-service communication, including API Call **Figure 7**: Physical Deployment View **Figure 8**: Logical/Structural View **Figure 10**: Implementation/Developer View (Component Diagram)

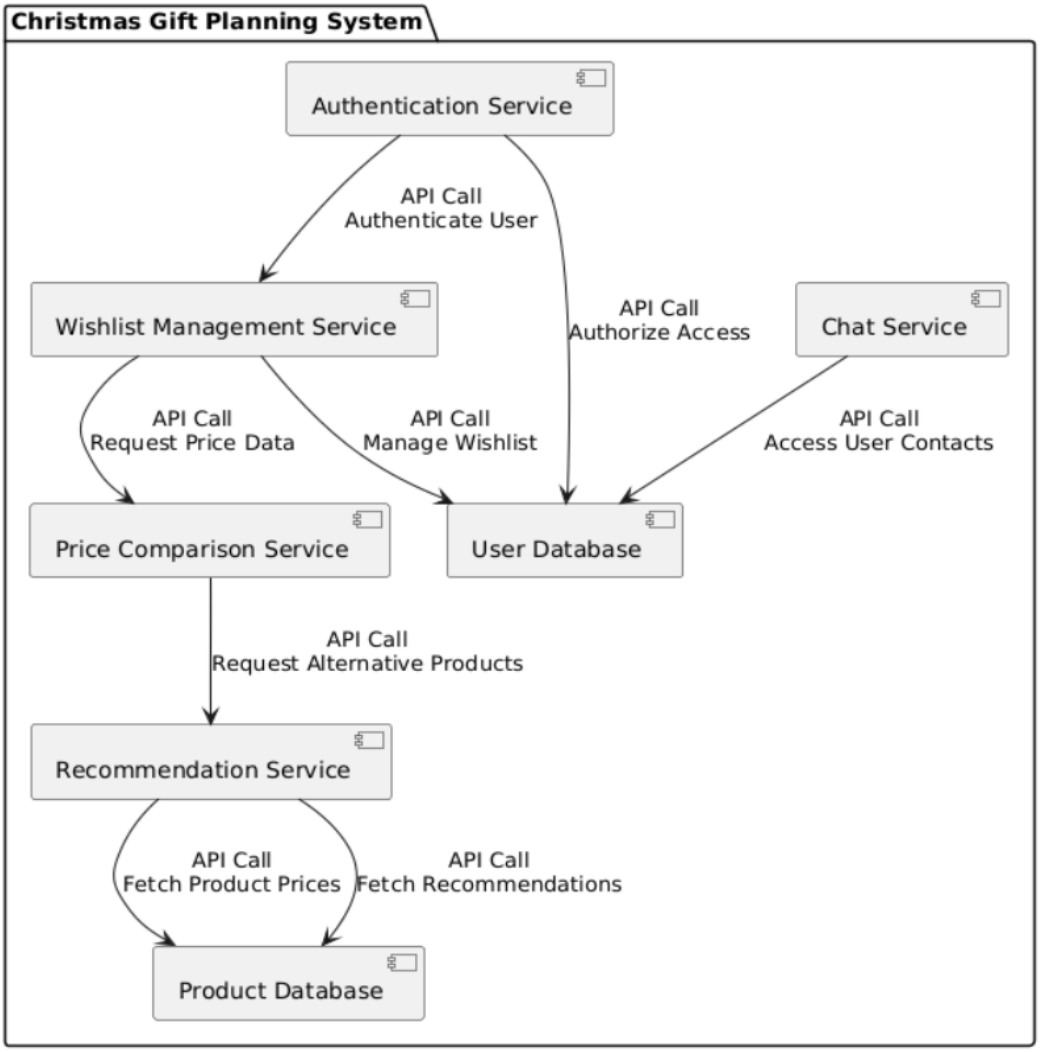


Figure 2

Stakeholders:

- Developers
- System Administrators
- Technical Architects

Shows service-to-service communication for modular scalability and independent service management.

Services and Functions

1. **Authentication Service (AS)**
   Handles user authentication and authorization processes.
2. **Wishlist Management Service (WMS)**
   Manages user wishlists, including creation, modification, and retrieval.

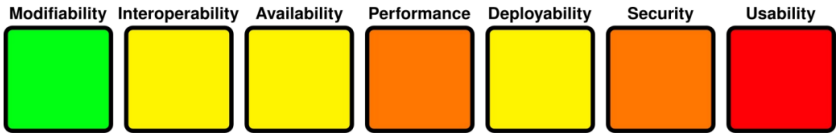
3. **Price Comparison Service (PCS)**
   Retrieves and compares product prices from various sources.
4. **Recommendation Service (RS)**
   Provides alternative product recommendations based on user preferences.
5. **Chat Service (CS)**
   Supports user communication and collaboration through messaging features.

Databases and Connections

1. **User Database (USER_DB)**
   Serves as a repository for user data, including account details and wishlists.
2. **Product Database (PRODUCT_DB)**
   Stores product information, such as pricing and availability.

System Interactions

1. **Authentication Service (AS)**
   Validates user credentials and ensures secure access to other services.
2. **Wishlist Management Service (WMS)**
   Requests price information from the Price Comparison Service (PCS) to provide users with updated data for their wishlists.



ID – Title :	**DR2 – Service Mesh for Inter-Service Communication (Pattern)**
Status:	Approved
Context:	Efficient and Secure Microservice Communication
Workshop Execution: Who Was Involved?	Developers, Network Engineers, System Architects
Considered Drivers:	FR2, FR4; QA2, QA5, QA7; CON2; QAS3, QAS8
Decision	• Service Mesh enhances inter-service communication by equipping each microservice with a sidecar proxy. • Sidecar proxies manage communication, routing, failure handling, and circuit-breaking without altering the internal logic of services. • Circuit breakers prevent system overload by temporarily disabling failing services and rerouting requests as needed. • The Service Mesh provides real-time traffic monitoring and observability, enabling quick issue resolution. • It ensures secure communication through mutual TLS (mTLS), offering authentication, encryption, and consistent reliability under heavy load.
Considered Alternatives:	**Direct Communication:** Services communicate directly without intermediaries but complicates monitoring and error management. **Challenges:** Increases difficulty in detecting and resolving errors, reducing system resilience and making management harder.
Consequences:	**Faster Response Times:** Optimizes communication paths to reduce delays and enhance critical functions like price comparison. **System Resilience:** Features like automatic retries and circuit breakers address temporary service issues, minimizing disruptions and ensuring smooth user experiences. **Simplified Issue Resolution:** Advanced monitoring tools enable quick identification and resolution of system issues, providing detailed insights for effective troubleshooting.
Additional Documentation:	**Figure 3**: The flow of requests through the Service Mesh, highlighting the interaction between sidecar proxies and services **Figure 9**: Process/Behavior View (Sequence Diagram)

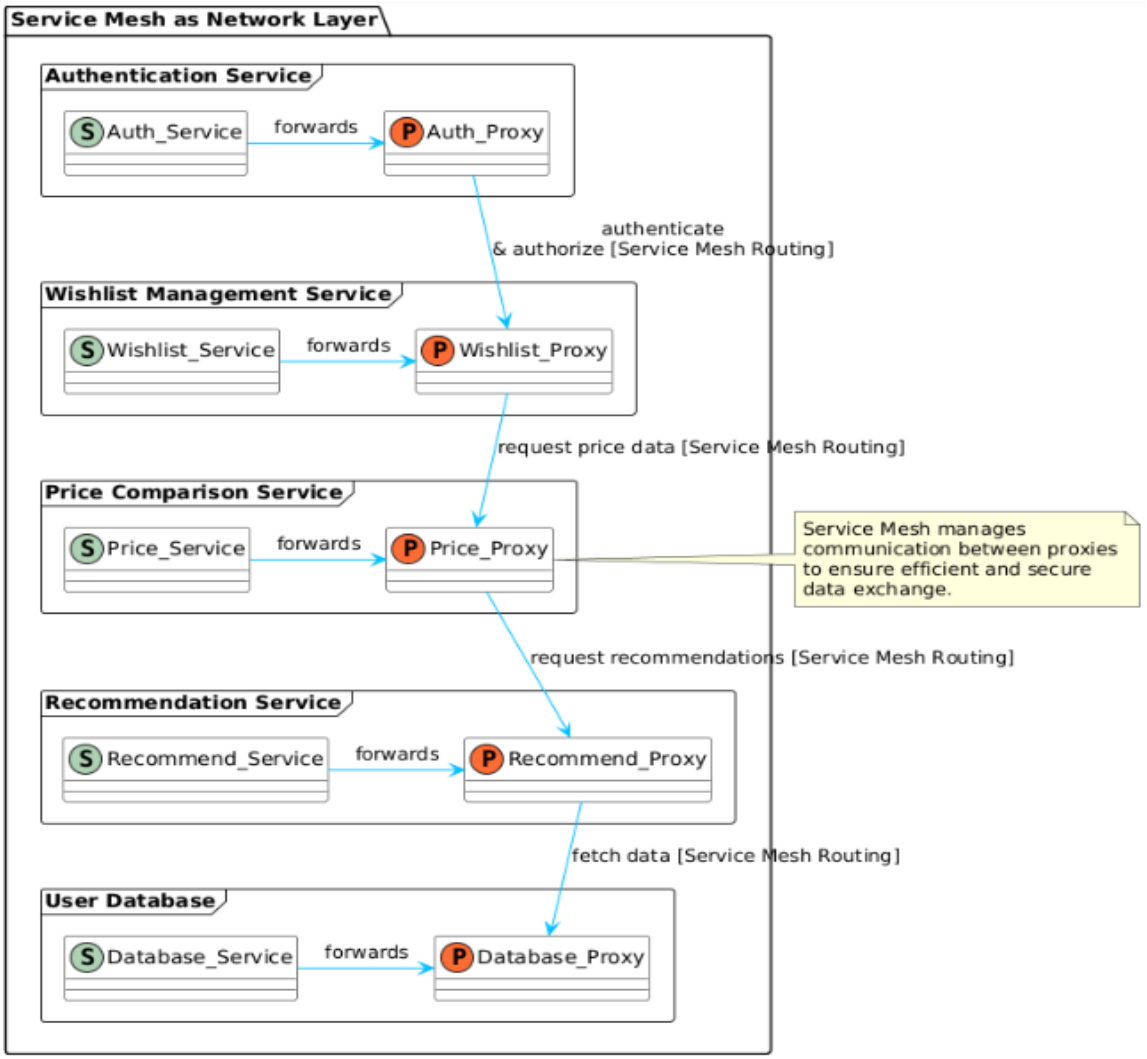


Figure 3

Stakeholders:

- Developers
- Network Engineers
- System Architects

Highlights optimized communication paths and service resilience.

- **Displays the Flow of Requests:**
  Illustrates the flow of requests between microservices using a service mesh.
- **Sidecar Proxies:**
  Manage communication between services, enabling faster and more reliable interactions.
- **Optimized Traffic Routing:**
  Ensures efficient traffic routing and delivers quick response times.

ID – Title :	**DR3 – Redis Caching (Tactic)**
Status:	Approved
Context:	Low Latency Through Data Caching
Workshop Execution: Who Was Involved?	Backend Developers, Performance Testers
Considered Drivers:	FR2, FR5; QA2, QA3; CON5; QAS3, QAS5
Decision	• Redis caching reduces database load and improves response times by caching frequently accessed data. • It enhances system performance by increasing speed and reducing latency for key features like price comparisons and alternative product suggestions. • Redis operates as an independent module, supporting the system's modular design by allowing updates or scaling without affecting other components. • Users benefit from instant responses, ensuring a smoother and more efficient shopping experience.
Considered Alternatives:	**Direct Database Queries:** The option of Direct Database Queries was not chosen because it leads to slower response times due to high latency.
Consequences:	**Faster Response Times:** Redis enhances user experience by providing quicker responses, especially for key features like price comparisons. **Improved Performance:** The system performs efficiently even under heavy traffic by leveraging Redis to handle frequently accessed data. **Reduced Database Load:** Caching reduces the load on the database, ensuring consistent and reliable performance. **Flexibility and Modularity:** Redis operates as a modular component, allowing updates or changes to specific parts of the system without affecting others. **Continuous System Operation:** Since Redis is modular, updates or maintenance can be carried out without stopping the entire system, ensuring smooth and uninterrupted operations.
Additional Documentation:	**Figure 9**: Process/Behavior View (Sequence Diagram) **Figure 8:** Logical/Structural View

ID – Title :	**DR4 – Load Testing Framework - JMeter (Tactic)** (JMeter is a load testing framework designed to test the performance of various services, particularly web applications, by simulating different loads and analyzing the results.)
Status:	Approved
Context:	Testing System Performance Under Heavy Load
Workshop Execution: Who Was Involved?	Performance Testers, System Administrators
Considered Drivers:	FR2, FR4; QA2, QA3; CON6; QAS3, QAS4
Decision	• The JMeter load testing tool is used to validate the system's performance and scalability during peak holiday seasons, particularly Christmas. • It ensures the system can efficiently handle varying traffic levels, from normal to peak usage. • JMeter verifies that the system can accommodate increased demand without disruption. • The tool helps maintain reliability by ensuring the platform remains operational and responsive under high loads. • Users experience consistent performance with no degradation, even during heavy usage. • Observing the system's reliability under stress increases user trust and confidence. • JMeter allows for swift identification and resolution of issues, ensuring a seamless experience during high-demand periods.
Considered Alternatives:	**No load testing**: Rejected because it can't test how the system works under stress.
Consequences:	**Confirms Normal Traffic Handling:** Validates that the system can manage regular traffic levels efficiently. **Identifies Design Weaknesses:** Helps locate areas in the system design that need improvement. **Ensures Smooth User Experience:** Guarantees a seamless user experience, even under heavy usage conditions.
Additional Documentation:	**Figure 4:** Load Testing Framework - JMeter

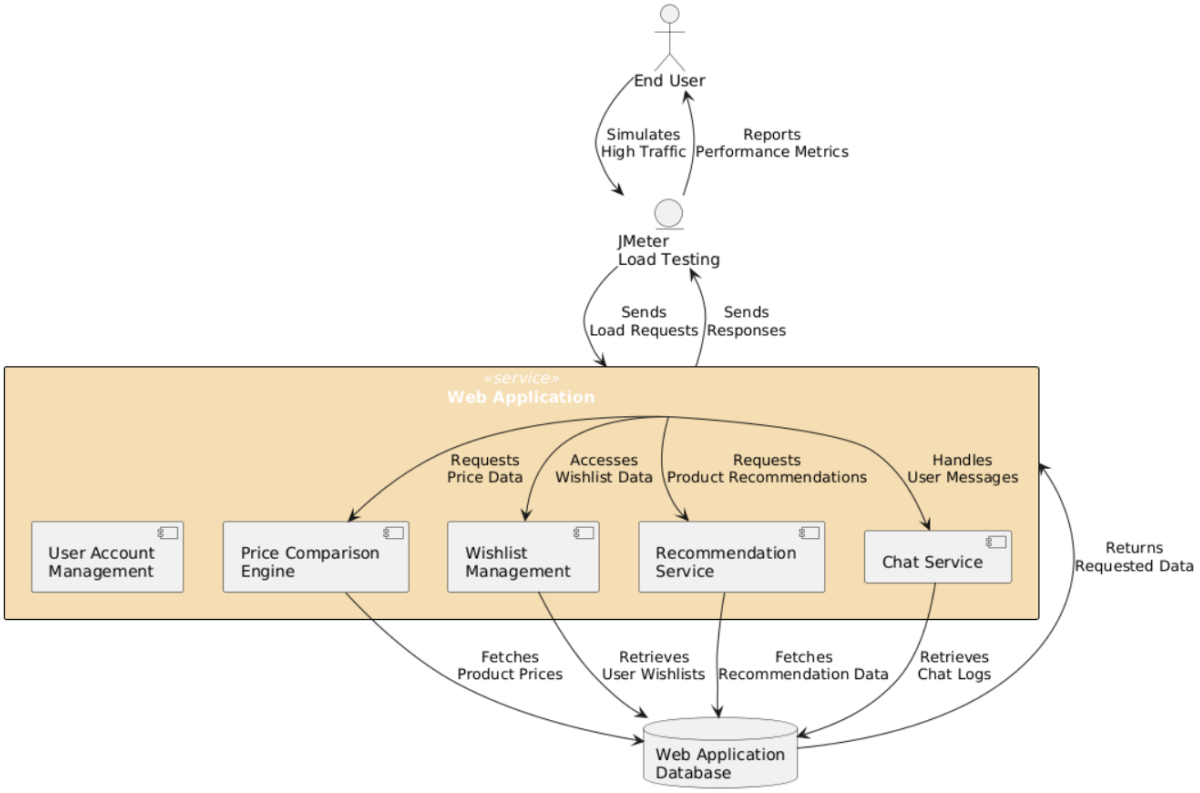


Figure 4

Stakeholders:

- Performance Testers
- System Administrators

Demonstrates system stress testing during peak load conditions.

- **Traffic Simulation with JMeter:**
  Demonstrates how JMeter simulates varying traffic loads.
- **Performance Validation:**
  Validates the system's performance under both normal and peak traffic conditions.
- **Stress Testing:**
  Ensures system stability and reliability during high-load scenarios.

ID – Title :	**DR5 – API Gateway (Tactic)**
Status:	Approved
Context:	Centralized Security for User Data
Workshop Execution: Who Was Involved?	Security Experts, Developers, System Administrators
Considered Drivers:	FR1, FR2, FR6; QA4, QA5, QA7; CON2; QAS6, QAS7, QAS11
Decision	• The API Gateway ensures centralized authentication and security, allowing only authorized users to access the system. • It balances user traffic across services and prevents system overload by managing request limits. • The gateway translates client requests into formats compatible with backend services for smooth communication. • It identifies the correct backend service for each request and efficiently routes it. • Frequently accessed data is cached to reduce backend load and improve response times. • The gateway tracks requests, errors, and performance, providing valuable insights for platform improvement. • It manages interactions with external APIs to ensure reliable data exchange and real-time updates on prices, discounts, and product recommendations. • From the user perspective, it ensures secure accounts, fast system performance, and accurate, up-to-date product information.
Considered Alternatives:	• **Direct Service Authentication:** Each service handles its own authentication, but this complicates managing consistent security policies. It can cause inconsistencies and potential security gaps across the system. Due to these risks, this approach was not chosen.
Consequences:	**Simplified and Strengthened Security:** Centralizes security measures, ensuring consistent and robust protection across all services. **Secure and Reliable User Experience:** Enhances user trust by providing a secure and seamless platform experience. **Centralized Traffic Management:** Manages all API traffic efficiently, preventing backend services from becoming overloaded. **Improved Observability and Performance:** Offers monitoring and caching capabilities to optimize performance and provide actionable insights. **Seamless API Integration:** Enables smooth interactions with external APIs, ensuring accurate and up-to-date pricing and product information for users.
Additional Documentation:	**Figure 8:** Logical / Structural View **Figure 9:** Process/Behavior View: Sequence Diagram **Figure 10:** Implementation/Developer View: Component Diagram

ID – Title :	**DR6 – Intrusion Prevention System (IPS) (Pattern)**
Status:	Approved
Context:	Preventing Unauthorized Access and Threats
Workshop Execution: Who Was Involved?	Security Specialists, Network Engineers
Considered Drivers:	FR6, FR8; QA4, QA5; CON4; QAS6, QAS8
Decision	• Intrusion Prevention Systems (IPS) detect and prevent potential security threats, enhancing overall system security. • They ensure user data, such as personal and wishlist information, remains safe, increasing user trust. • IPS complements the API Gateway by providing network-level protection against threats like DDoS attacks and port scans. • It mitigates DDoS attacks by preventing excessive traffic that could disrupt the system. • IPS blocks unauthorized attempts to exploit open ports through port scans. • It monitors suspicious activities, such as failed logins or unexpected traffic spikes, to prevent potential breaches.
Considered Alternatives:	**Firewall Only:** Rejected because it does not provide detailed intrusion detection or prevention capabilities.
Consequences:	**Improved Security:** Prevents unauthorized access and blocks malicious activities, safeguarding the system. **Enhanced Platform Reliability:** Detects and mitigates threats in real time, ensuring stable and uninterrupted operations. **Increased User Trust:** Protects user data effectively, fostering confidence and trust in the platform
Additional Documentation:	**Figure 7:** Physical Deployment View **Figure 9:** Process/Behavior View: Sequence Diagram

ID – Title :	**DR7 – Enhanced Data Backup and Recovery (Tactic)**
Status:	Approved
Context:	Secure Data Backup and Recovery
Workshop Execution: Who Was Involved?	System Administrators, IT Operations Teams
Considered Drivers:	FR1, FR6; QA4, QA5; CON5; QAS6, QAS8
Decision	• The cloud-based backup solution automatically saves changes and creates multiple backup versions for secure restoration. • It ensures quick and easy recovery of data in case of failures or cyberattacks. • The system safeguards critical user data, enhancing platform reliability and security. • Cloud-based backups allow restoration without reliance on physical infrastructure, providing flexibility and accessibility. • Users feel confident knowing their personal and wishlist data are protected against loss or damage. • Quick recovery ensures minimal disruption to the user experience during system issues.
Considered Alternatives:	**Local backups:** Rejected due to higher risks of data loss (e.g., hardware failures) and slower recovery processes compared to cloud-based solutions.
Consequences:	**Efficient Data Recovery:** Ensures quick and reliable restoration of data during system failures or cyberattacks. **Multiple Backup Versions:** Maintains several backup versions to provide a secure and flexible recovery process. **Increased User Trust:** Builds confidence in the platform by safeguarding user data and ensuring its availability.
Additional Documentation:	**Figure 5:** Backup schedule and recovery process diagram

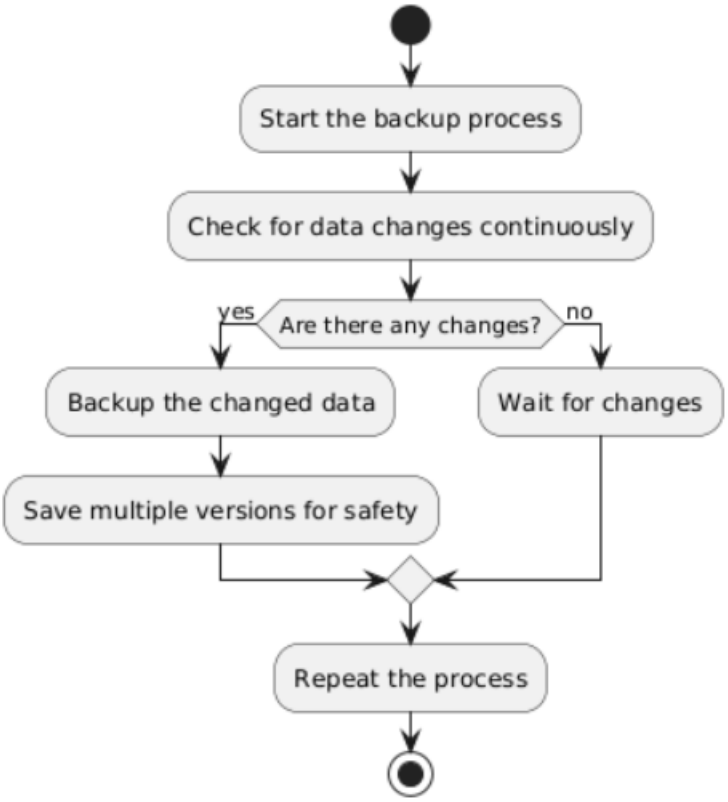


Figure 5

Stakeholders:

- System Administrators
- IT Operations Teams

Visualizes cloud-based backup and recovery mechanisms.

- **Cloud-Based Backup Visualization:**
  Illustrates cloud-based backup processes with automatic scheduling.
- **Versioned Backups:**
  Offers multiple backup versions to enable data recovery in case of failures or attacks.
- **Fast and Secure Restoration:**
  Ensures minimal disruption with quick and secure restoration processes.

ID – Title :	**DR8 – Enhanced Personalization with Maintain User Model (Tactic)**
Status:	Approved
Context:	Tailoring Gift Recommendations to Individual Preferences
Workshop Execution: Who Was Involved?	End-Users, UI/UX Designers, Developers
Considered Drivers:	FR4, FR5; QA1, QA6; CON5; QAS1, QAS2
Decision	• The "Maintain User Model" tactic tracks user interactions, such as items viewed, added to wishlists, or purchased, to create detailed user profiles. • It provides personalized gift suggestions based on past user behaviors and preferences. • The system highlights relevant products, such as frequently searched categories or preferred brands, especially during promotions or sales. • Users receive alerts about items of interest, such as discounts on preferred products, enhancing their shopping experience.
Considered Alternatives:	**Static User Profiles:** This was rejected because it can't adjust to users' changing behaviors and preferences, which would make recommendations less accurate over time. This method wouldn't support the goal to offer a highly personalized and dynamic user experience effectively.
Consequences:	**Relevant Gift Suggestions:** Delivers gift recommendations tailored to the user's past interactions and preferences, enhancing the overall shopping experience. **Personalized User Experience:** Makes the interface and recommendations feel more thoughtful and personal, boosting user satisfaction. **Increased Loyalty:** Encourages repeat visits and builds customer loyalty by understanding and addressing individual user needs effectively.
Additional Documentation:	

ID – Title :	**DR9 – Support User Initiative: Cancel, Undo, Pause/Resume, Aggregate (Tactic)**
Status:	Approved
Context:	Enhance system interactivity by allowing users to control and modify their actions dynamically.
Workshop Execution: Who Was Involved?	End-Users, Designers
Considered Drivers:	FR4, FR7; QA1, QA2; CON5; QAS1, QAS3
Decision	• User control features like cancel, undo, pause/resume, and aggregate are integrated into the system's interface. • These features enable users to manage inputs freely, correct mistakes, and control interactions without delays. • They enhance user autonomy and satisfaction by making the system more adaptable and responsive to user needs. • The platform becomes easier to use, reducing frustration and improving the overall quality of interactions.
Considered Alternatives:	**Static User Interface:** Rejected because it does not allow for flexible user interactions, which might make users unhappy due to a strict and unforgiving interface.
Consequences:	**Increased User Satisfaction:** Users have greater control over their actions, leading to a more personalized and responsive experience. **Reduced User Errors:** Features like undo and cancel allow users to easily correct mistakes, enhancing the usability of the system. **Enhanced Flexibility:** Pause/Resume and Aggregate functions provide users with options to manage their activities more efficiently, suiting varied use cases and preferences.
Additional Documentation:	

ID – Title :	**DR10 – Load Balancers for Traffic Management (Tactic)**
Status:	Approved
Context:	Efficient Traffic Distribution During High Demand
Workshop Execution: Who Was Involved?	Network Administrators, Performance Engineers
Considered Drivers:	FR2, FR4; QA3, QA5; CON3; QAS4, QAS9
Decision	• Load balancers are used to distribute incoming web traffic across multiple servers. • They prevent any single server from becoming overloaded, ensuring stable performance. • This approach allows the system to handle high traffic volumes, especially during peak times like the holiday season. • Users benefit from faster and more reliable website access. • The risk of slow loading times or downtime is minimized, enhancing the shopping experience and user satisfaction.
Considered Alternatives:	**Static Routing:** Rejected because it does not dynamically adjust to changes in traffic volume, potentially leading to traffic overload on certain servers.
Consequences:	**Reliability:** Servers are less likely to crash under high traffic as load is distributed evenly. **Scalability:** The system can handle increases in traffic without performance degradation, ensuring the site remains fast during peak times. **Enhanced User Experience:** Visitors experience reduced load times and smoother interactions, even during heavy usage periods.
Additional Documentation:	**Figure 7:** Physical Deployment View

ID – Title :	**DR11 – Auto-Scaling Mechanism for Traffic Management (Tactic)**
Status:	Approved
Context:	Dynamic Traffic Management with Auto-Scaling
Workshop Execution: Who Was Involved?	Cloud Architects, System Administrators, Developers
Considered Drivers:	FR2, FR4; QA3, QA5; CON3; QAS4, QAS9
Decision	• An auto-scaling solution dynamically adjusts resources based on real-time traffic demands. • Resources are added when user numbers exceed 5,000 and scaled further above 7,000 users. • This ensures consistent performance and prevents service disruptions during high traffic. • Users benefit from fast and uninterrupted service, even under heavy system load. • Auto-scaling optimizes resource usage to maintain a seamless user experience.
Considered Alternatives:	**Fixed infrastructure**: Rejected because it cannot scale quickly and easily.
Consequences:	**Performance:** Maintains system performance during traffic increases, preventing delays for users. **Cost Efficiency:** Reduces costs by scaling down resources during lower usage periods. **Flexibility:** Quickly adapts resource usage as traffic levels change.
Additional Documentation:	**Figure 6**: Web Platform Automatic Scaling Flow Diagram

Stakeholders:

- Cloud Architects
- System Administrators
- Developers

Explains dynamic resource allocation to manage traffic demands.

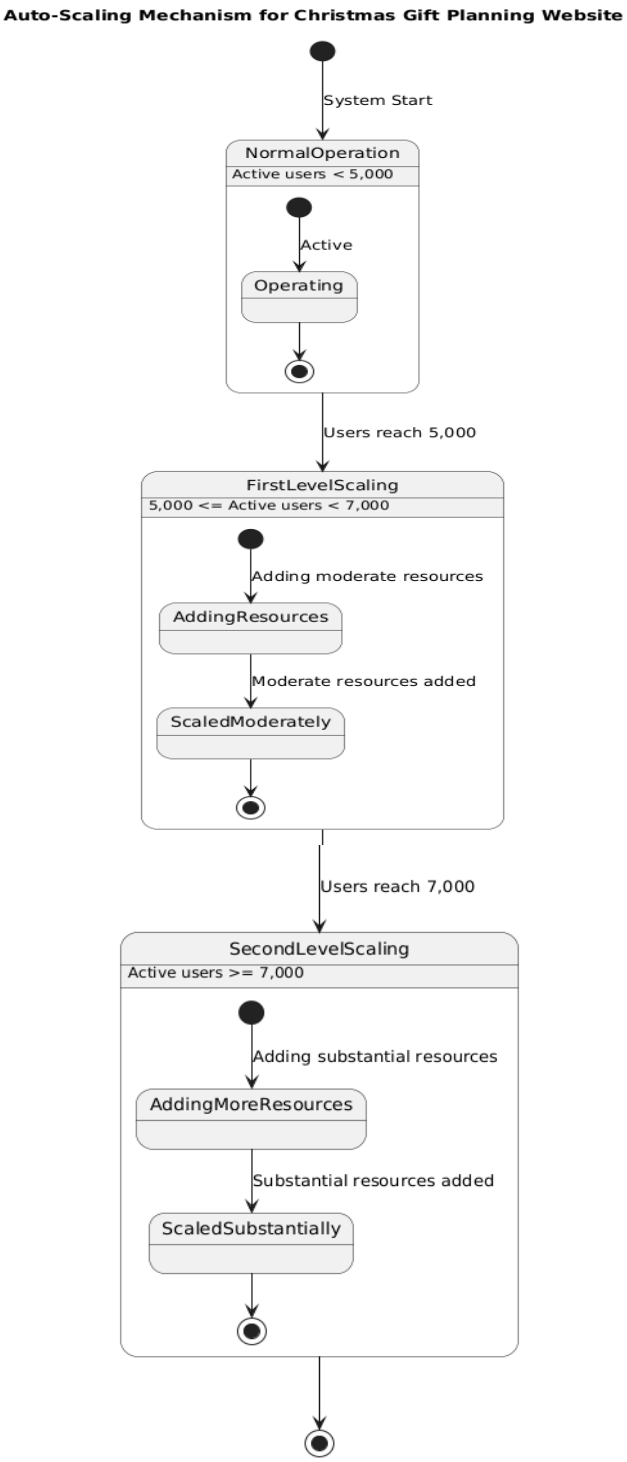


Figure 6

- **Dynamic Resource Scaling:**
  Explains the dynamic adjustment of resources based on traffic demands.
- **Automated Resource Management:**
  Automatically increases or decreases server resources during periods of high or low usage.
- **Optimized Performance and Cost Efficiency:**
  Balances system performance with cost-effective resource utilization.

ID – Title :	**DR12 – Advanced Monitoring and Alerting System (Tactic)**
Status:	Approved
Context:	Real-Time System Monitoring for Reliability
Workshop Execution: Who Was Involved?	System Administrators, DevOps Engineers
Considered Drivers:	FR2, FR6; QA2, QA5; CON5; QAS3, QAS8
Decision	• A centralized monitoring system oversees all platform activities from a single point. • It detects and addresses issues like errors, performance slowdowns, or security threats swiftly. • Performance monitoring alerts teams to optimize when slowdowns occur, such as delays in viewing gift lists. • In case of a system crash, it restarts affected components to ensure uninterrupted service. • Security threats, such as DDoS attacks or unauthorized access attempts, are isolated and addressed immediately. • The system uses real-time data analysis, error logging, and automated alerts to enhance platform reliability and efficiency.
Considered Alternatives:	**Manual monitoring:** It was rejected because it doesn't give real-time information and responds too slowly.
Consequences:	**Real-Time Performance Monitoring:** Instantly detects and addresses performance slowdowns or failures, reducing downtime and minimizing disruptions. **Centralized Metrics Analysis:** Consolidates system metrics for a comprehensive view, simplifying management and improving decision-making. **Enhanced Trust and Reliability:** Boosts confidence in system stability by proactively addressing issues, reassuring users and administrators.
Additional Documentation:	

ID – Title :	**DR13 – Modular Design (Pattern)**
Status:	Approved
Context:	Easily Expandable System Through Modular Design
Workshop Execution: Who Was Involved?	Developers, Technical Architects, System Administrators
Considered Drivers:	FR5, FR8; QA3, QA5, QA6; CON6; QAS9, QAS10
Decision	• Modular architecture enables independent development and updates for each system component. • This approach reduces downtime during maintenance and simplifies system management. • New features, like a "gift delivery tracking" module, can be added without affecting existing functionalities. • Modular design minimizes the impact of updates or fixes, ensuring high system availability during peak times like holiday seasons. • Issues in one module, such as communication, do not disrupt other modules like price comparison or wishlist management, enhancing reliability. • Users experience a seamless platform with uninterrupted service, even during updates. • Faster introduction of new features keeps the platform up-to-date with minimal disruption.
Considered Alternatives:	**Monolithic updates**: Rejected because they cause longer downtime and are harder to manage.
Consequences:	**Faster System Updates:** Allows for quicker implementation of updates and improvements. **Reduced Downtime:** Minimizes service interruptions during maintenance or when adding new features. **Increased Flexibility:** Enables the addition of new functionalities without disrupting existing components. **Improved Error Isolation:** Contains issues within a single module, reducing the risk of system-wide failures.
Additional Documentation:	**Figure 10**: Implementation/Developer View (Component Diagram) **Figure 11**: Implementation/Developer View (Package Diagram)

Chapter 3: Views

3.1 Physical Deployment View :

Stakeholders: End Users , System Engineers, Security Teams, Developers

- Depicts physical system components like servers, databases, and IPS

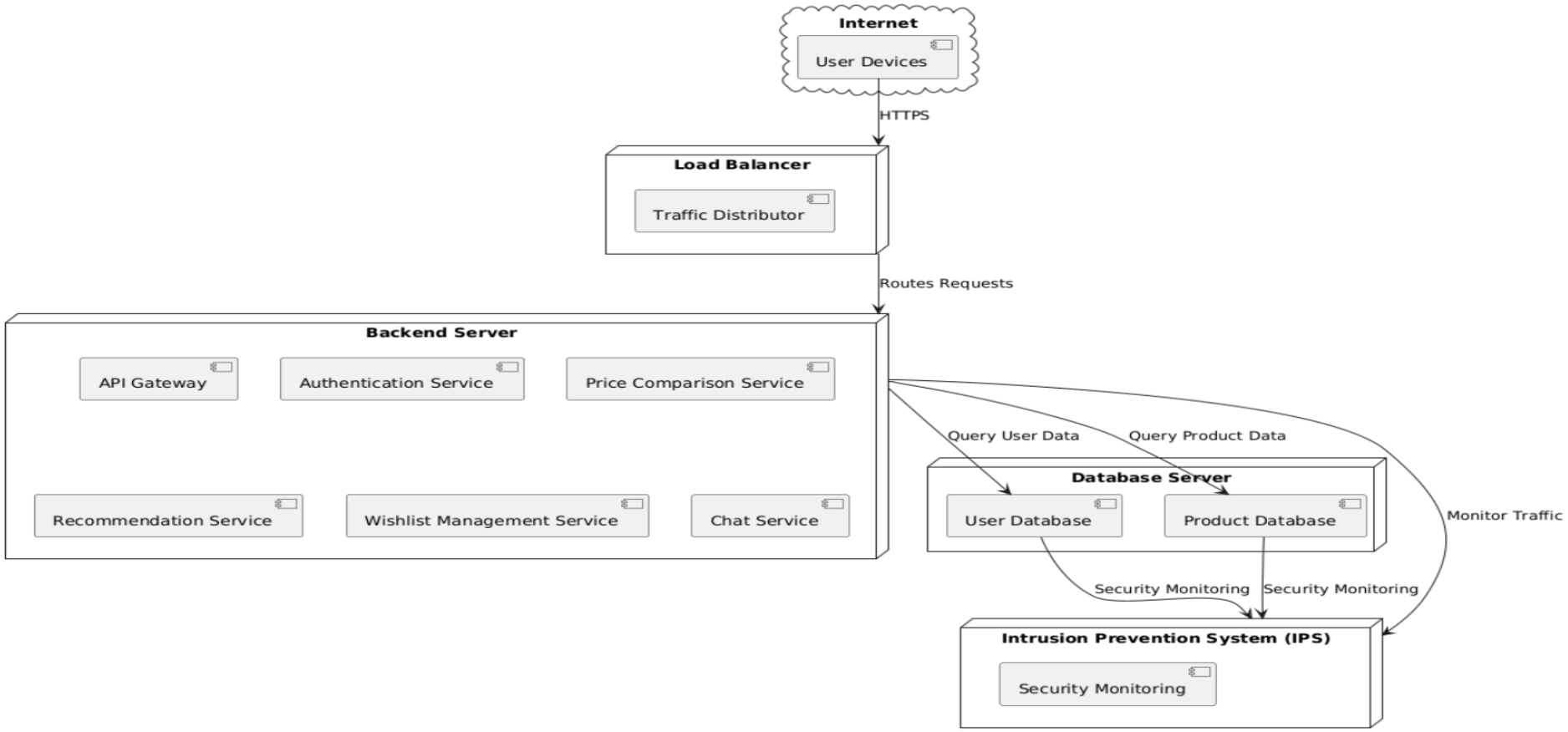


Figure 7

Artifact	Description	Relevant Stakeholders
Internet	Represents the entry point for users interacting with the system via secure HTTPS.	**End Users**: Access the system securely via HTTPS.
User Devices	Devices (e.g., smartphones, desktops) that access the system through a web interface.	**End Users**: Use the platform for Wishlist management and gift planning.
Graphical User Interface (GUI)	The front-end interface through which users interact with the system; it provides visualizations and controls.	**End Users:** Use the platform for wishlist management and gift planning. **UI/UX Designers:** Design an intuitive and user-friendly interface. **Developers:** Implement and maintain the GUI. **Testers/Quality Assurance Teams:** Ensure usability and functionality. **System Administrators:** Maintain GUI reliability and performance. **Product Owners/Managers:** Prioritize GUI features and gather user feedback. **Marketing Teams:** Promote the GUI as part of the platform's strengths.
Load Balancer	Distributes incoming traffic across servers to ensure scalability and fault tolerance.	**System Engineers**: Monitor and configure traffic handling.
Server 1 (API Gateway)	Hosts the API Gateway for routing user requests to appropriate services.	**Developers:** Build and test services; **System Engineers:** Ensure secure and efficient operation.
Server 2 (Authentication Service)	Manages user authentication and ensures secure access to the platform.	**Developers:** Implement authentication mechanisms; **System Engineers:** Ensure security and performance.
Server 3 (Price Comparison Service)	Retrieves and compares product prices to provide users with the best options.	**Developers:** Build algorithms for price comparison; **System Engineers:** Monitor service reliability.
Server 4 (Recommendation Service)	Suggests alternative products based on user preferences and product availability.	**Developers:** Implement recommendation algorithms; **System Engineers:** Ensure performance.
Server 5 (Wishlist Management Service)	Allows users to create, update, and manage wishlists.	**Developers:** Build wishlist functionalities; **End Users:** Organize gift ideas.
Server 6 (Chat Service)	Provides communication tools like chat to coordinate gift planning.	**Developers:** Implement messaging features; **End Users:** Collaborate with group members.

User Database	Stores user account details, preferences, and Wishlists securely.	**System Administrators**: Ensure data security and backups; **Developers**: Query and update user data.
Product Database	Contains product-related data, including prices, discounts, and availability.	**E-commerce Vendors**: Provide product data; **Developers**: Optimize queries for product information.
Intrusion Prevention System (IPS)	Monitors and protects the system against unauthorized access, DDoS attacks, and other threats.	**Security Specialists**: Monitor threats; **System Engineers**: Maintain and upgrade system defenses.
Docker Containers	Encapsulates services on each server for modular deployment and efficient resource management.	**System Engineers**: Deploy and manage containers; **Developers**: Build modular services.

3.2 Logical / Structural View

Stakeholders: End Users, Developers, System Administrators

- Focuses on how components like API Gateway and databases interact structurally.

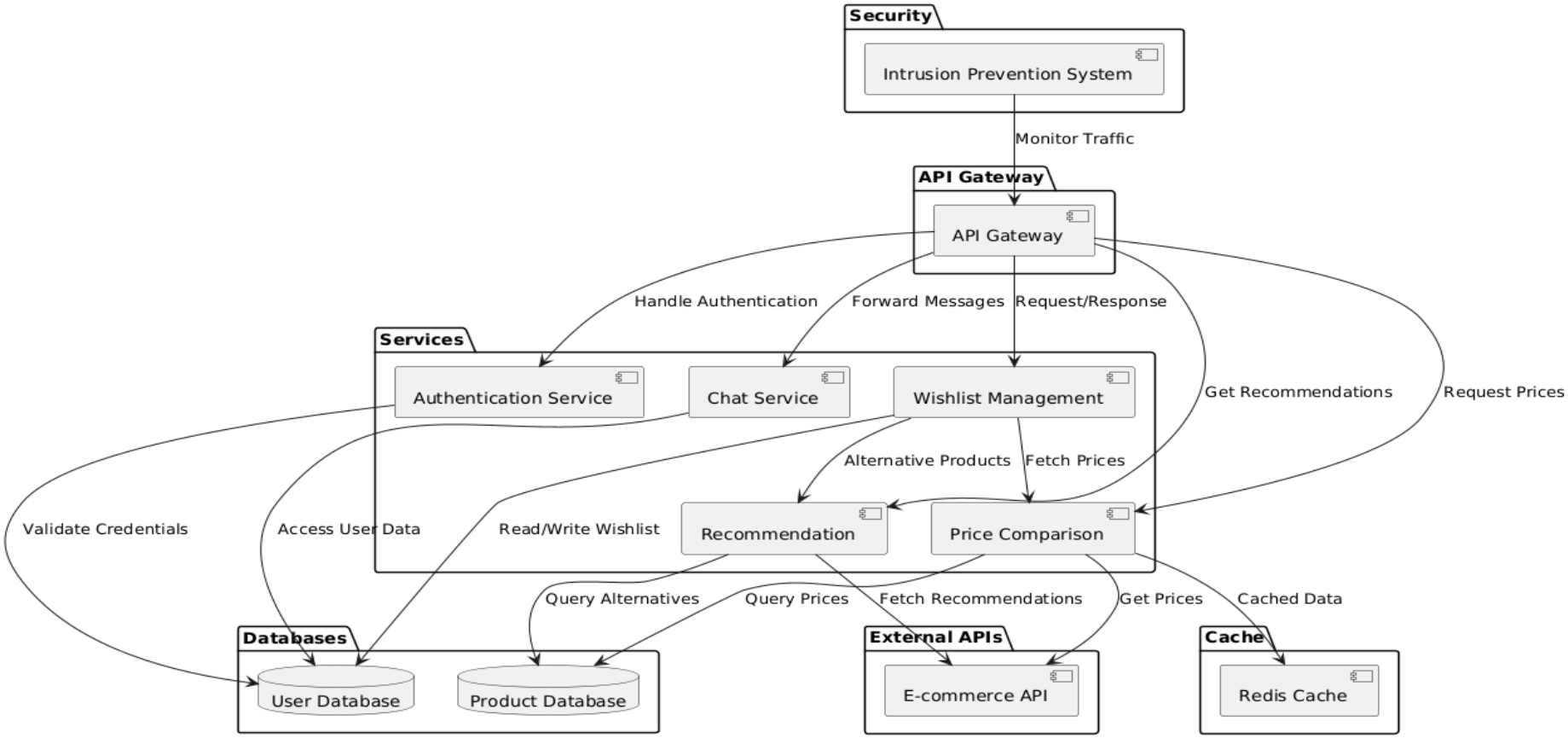


Figure 8

Component	Purpose	Connections	Stakeholders Impacted
API Gateway	Central entry point for requests, handles routing, authentication, and traffic management.	- Connects to all services (e.g., Wishlist Management, Price Comparison). - Routes authentication requests to **Authentication Service.**	**End Users:** Ensures secure and efficient access. **System Administrators:** Monitors traffic and security.
Authentication Service (AS)	Validates user credentials and ensures secure access to other services.	- Communicates with **User Database** to validate credentials. - Connected to **API Gateway** for authentication requests.	**End Users:** Secure login and access. **Security Teams:** Ensures compliance with security protocols.
Wishlist Management (WMS)	Manages user wishlists, including adding, modifying, and reading wishlist data.	- Communicates with **User Database** for storing wishlists. - Requests price data from **Price Comparison Service**. - Requests recommendations from **Recommendation Service.**	**End Users:** Organizes wishlists and gift ideas. **Developers:** Maintains the wishlist module.
Price Comparison Service (PCS)	Retrieves product prices and discounts, using caching for performance.	- Queries **Product Database** for product prices. - Uses **Redis Cache** for frequently accessed data. - Communicates with **E-commerce API** for real-time price updates.	**End Users:** Finds the best prices quickly. **E-commerce Vendors:** Provides real-time pricing.
Recommendation Service (RS)	Suggests alternative products based on user preferences or product availability.	- Queries **Product Database** for product alternatives. - Communicates with **E-commerce API** for recommendations.	**End Users:** Receives tailored suggestions. **Developers:** Implements algorithms for recommendations.

Chat Service (CS)	Facilitates user communication via chat or email.	- Accesses **User Database** for user information. - Connected to **API Gateway** for message forwarding.	**End Users:** Collaborates with group members. **System Administrators:** Manages communication infrastructure.
User Database (USER_DB)	Stores user account details, credentials, and wishlist data securely.	- Accessed by **Authentication Service, Wishlist Management, and Chat Service.**	**System Administrators:** Ensures data security and availability. **End Users:** Trusts their data is secure.
Product Database (PRODUCT_DB)	Stores product information, including prices, stock, and alternatives.	- Accessed by **Price Comparison Service** and **Recommendation Service** for querying product data.	**E-commerce Vendors:** Provides accurate product data. **Developers:** Optimizes product queries.
Redis Cache (REDIS)	Provides caching for frequently accessed price data to reduce database load.	- Connected to **Price Comparison Service** to speed up price-related queries.	**Developers:** Improves system performance. **End Users:** Benefits from faster responses.
E-commerce API (EAPI) **(External API)**	An external service that provides data to the system or adds functionality. External service providing real-time price and availability data.	- Communicates with **Price Comparison Service** and **Recommendation Service.**	**E-commerce Vendors:** Shares pricing data. **End Users:** Accesses real-time product information.
Intrusion Prevention System (IPS)	Monitors and blocks unauthorized access or malicious traffic.	- Secures traffic handled by **API Gateway.** - Monitors communication with **E-commerce API**.	**Security Teams:** Prevents breaches. **System Administrators:** Ensures secure system operations.

3.3 Process/Behavior View: Sequence Diagram

Stakeholders: Developers, System Administrators

- Describes process flows like requests through the service mesh.

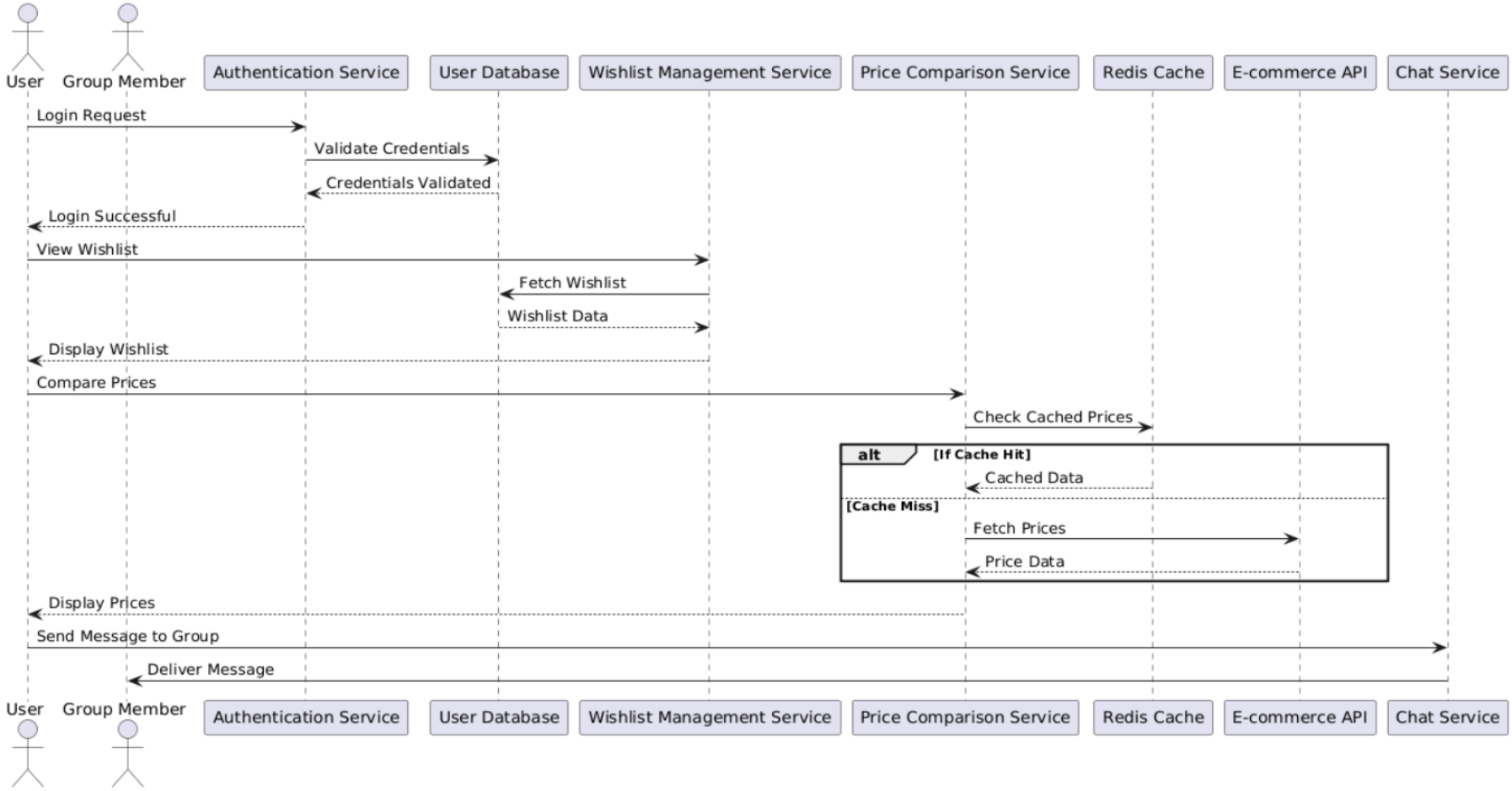


Figure 9

3.4 Implementation/Developer View: Component Diagram

Stakeholders: Developers, Product Owners, Testers/QA Teams

- Displays core components, databases, and API integrations.

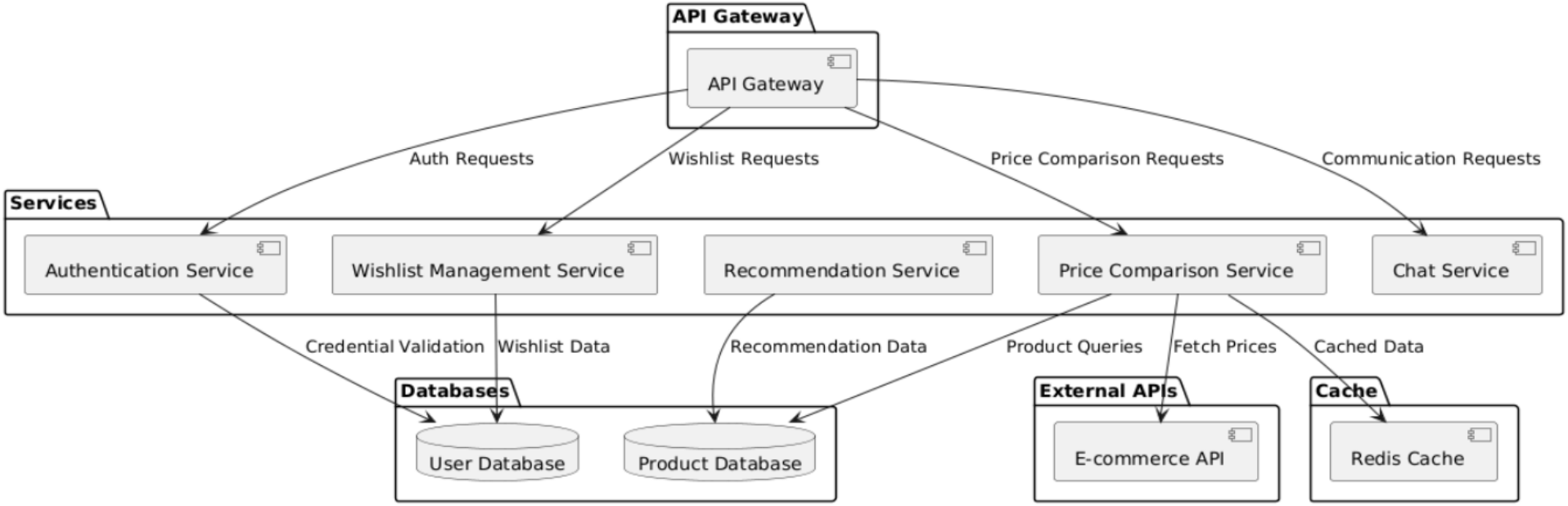


Figure 10

- Core system components, such as Authentication Service, Wishlist Management Service, and Price Comparison Service.
- Databases (User Database, Product Database) and external integrations (E-commerce API).
- The API Gateway, which acts as a central hub for routing user requests to the appropriate services.

3.5 Implementation/Developer View: Package Diagram

Stakeholders: Developers, Product Owners

- Emphasizes system modularity and maintainability for scalability.

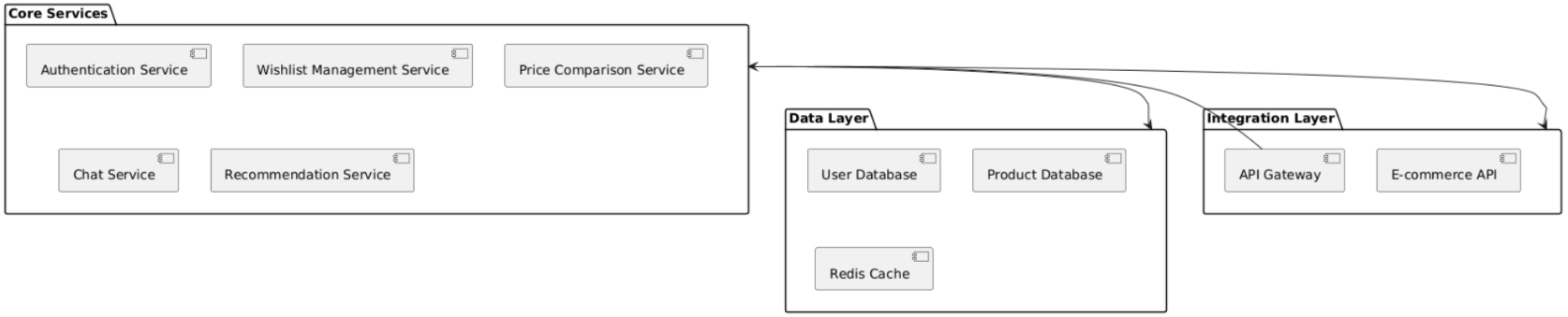


Figure 11

- The Package Diagram organizes the system into logical layers:
- Core Services: Includes the main functionalities like authentication, wishlist management, and price comparison.
- Data Layer: Contains the User Database, Product Database, and Redis Cache.
- Integration Layer: Consists of the API Gateway and s.
- This diagram emphasizes the subsystem decomposition and maintainability of the platform, ensuring scalability and flexibility for future enhancements.

3.6 Architecture Overview

- This section shows an overview of the chosen architecture.

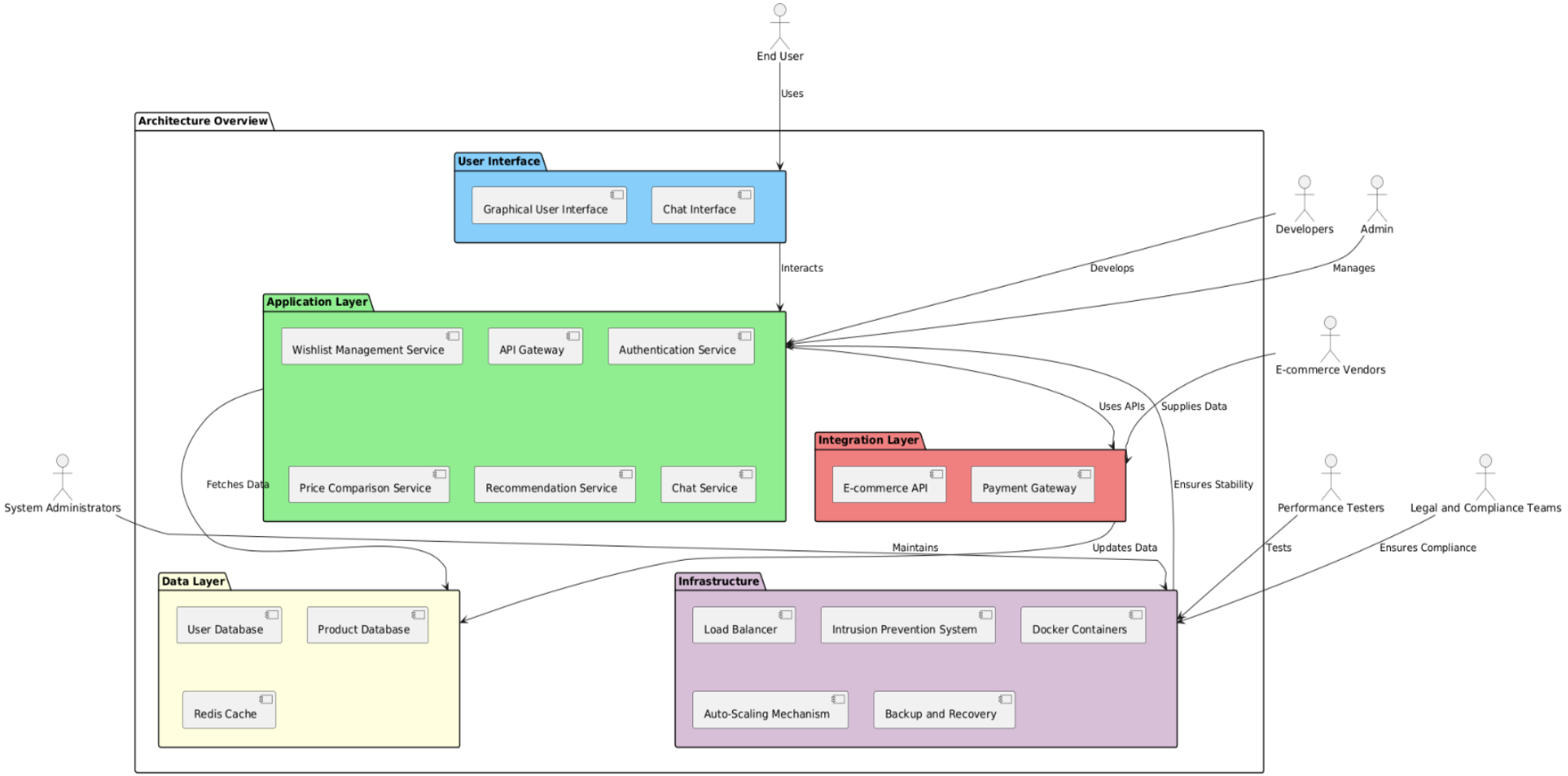


Figure 12: Architecture Overview

Chapter 4: List of Stakeholders

End Users

- Families and friends planning gifts for the holiday season.
- Individual users managing wishlists, comparing prices, and coordinating with others.

Corporate Users:

- Businesses planning bulk gifts for employees or clients (if applicable).

System Administrators

- Responsible for maintaining the platform, ensuring uptime, and managing security.
- Monitoring system performance during peak times (e.g., holidays).

Developers

- Building, testing, and updating the features of the platform, including wishlist management, price comparison, and communication tools.

DevOps Engineers:

- Implementing CI/CD pipelines, managing server infrastructure, and handling system scaling during peak usage.

E-commerce Vendors

- Retailers and online stores providing price data, discounts, and product availability information through APIs.

Small/Local Vendors:

- Providing region-specific products and discounts to enhance diversity.

Product Owners/Managers

- Overseeing the project's development, ensuring requirements are met, and managing feature priorities.
- Gathering user feedback for product iterations and improvements.

Marketing Teams

- Promoting the platform to attract new users and maintain user engagement.
- Running campaigns to boost platform usage during critical holiday seasons.

Legal and Compliance Teams

- Ensuring adherence to regulations like GDPR for data privacy and security.
- Monitoring compliance with regional laws for data transfer and e-commerce operations.

Hosting and Infrastructure Providers

- Offering server resources and scalability for handling traffic, especially during peak times.
- Providing disaster recovery solutions and data redundancy services.

Third-party API Providers

- External systems providing price, discount updates, and alternative product suggestions.

Payment Gateway Providers:

- Facilitating secure online payments if transactions are part of the platform.

Testers/Quality Assurance Teams

- Validating the platform's usability, performance, scalability, and security.
- Performing stress testing and reliability checks under simulated peak traffic conditions.

Design Teams

- Creating a user-friendly interface and improving user experience.
- Ensuring accessibility compliance for users with disabilities.

Investors or Sponsors

- Providing funding for the development and maintenance of the tool.
- Supporting marketing and expansion efforts to attract more users and partners.