SOFTWARE ARCHITECTURES

Ecommerce Application

Assignment - 2

Submitted By:

Mallidi Akhil Reddy

2024TM93056

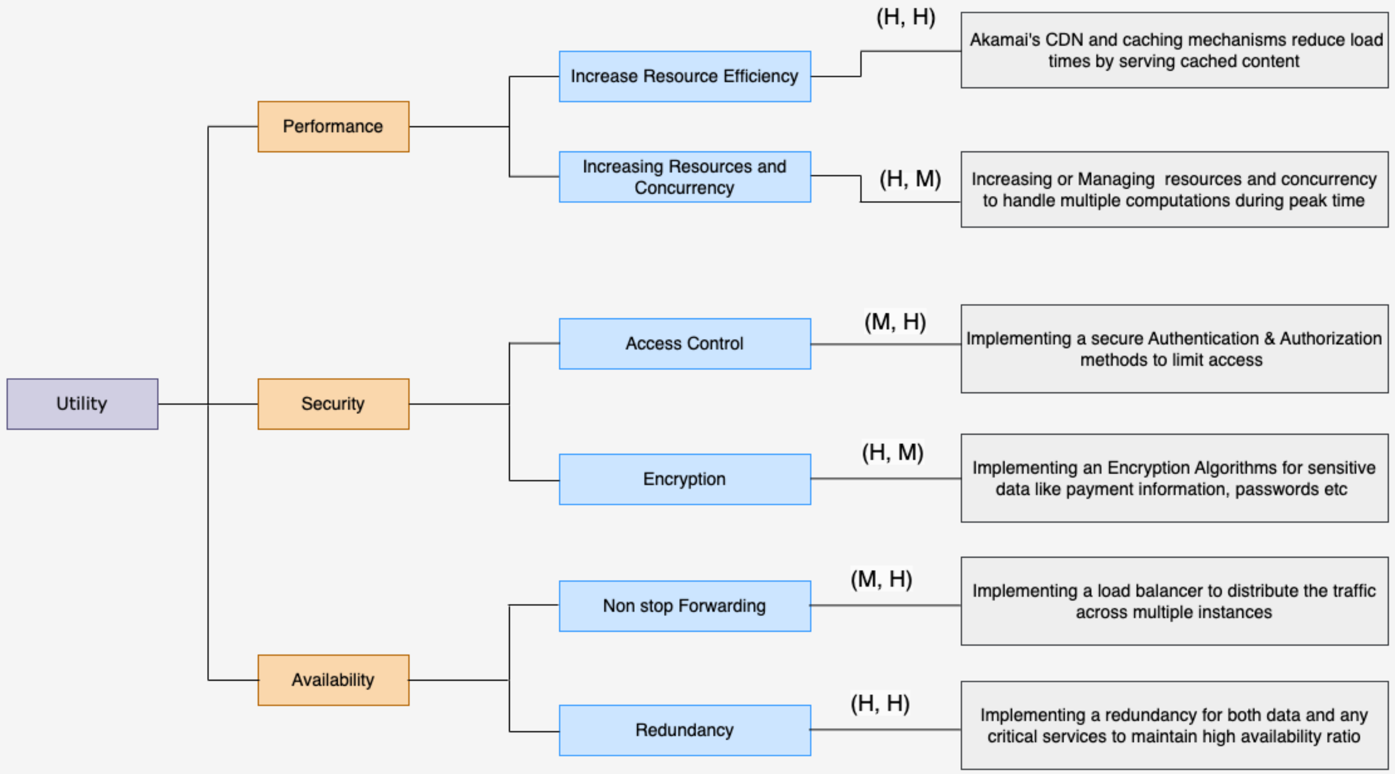
Purpose of the System

- The purpose of this ecommerce application is to provide a digital platform where users can browse, search, and purchase a wide range of products online.
- User-friendly application for viewing and exploring a wide range of products with detailed images, descriptions, and customer reviews.
- The system provides product suggestions based on user behavior, preferences, and past purchases to enhance the user experience.
- This platform allows users to create accounts, save payment details, manage their orders, track deliveries, and review previous purchases.
- The system also allows users to search products using keywords and apply filters like price, brand, category, and ratings to easily find what they're looking for.
- The system safeguards the user data, especially sensitive information like payment details, through robust security measures and compliance with data protection laws.

Key Requirements – Functional & Non-Functional

Functional	Non-Functional
Product Catalogue and Search functionalities	Performance
User Account Management	Availability
User Product Reviews	Security
Personalised Product Reviews	Modifiability
Processing Payments	Usability
Shopping Cart and Order Management	
Notifications and Alerts	
Customer Support	

Utility tree - Architecturally Significant Requirements (ASR)


Utility
Performance
Security
Availability
Increase Resource Efficiency
Increasing Resources and Concurrency
Access Control
Encryption
Non stop Forwarding
Redundancy
(H, H)
(H, M)
(M, H)
(H, M)
(M, H)
(H, H)
Akamai's CDN and caching mechanisms reduce load times by serving cached content
Increasing or Managing resources and concurrency to handle multiple computations during peak time
Implementing a secure Authentication & Authorization methods to limit access
Implementing an Encryption Algorithms for sensitive data like payment information, passwords etc
Implementing a load balancer to distribute the traffic across multiple instances
Implementing a redundancy for both data and any critical services to maintain high availability ratio

Tactics used to achieve the top 3 ASR’s

Performance

- **Increase Resource Efficiency** – Caching frequently accessed data such as product catalogs, reviews etc with Akamai CDN. A caching layer temporarily stores data for quick retrieval. For instance, product details are cached to reduce database reads.
- **Increasing Resources and Concurrency** - Use read replicas to offload read-heavy operations like product data from the primary database and and also replicating the critical services to handle multiple requests by routing among them.

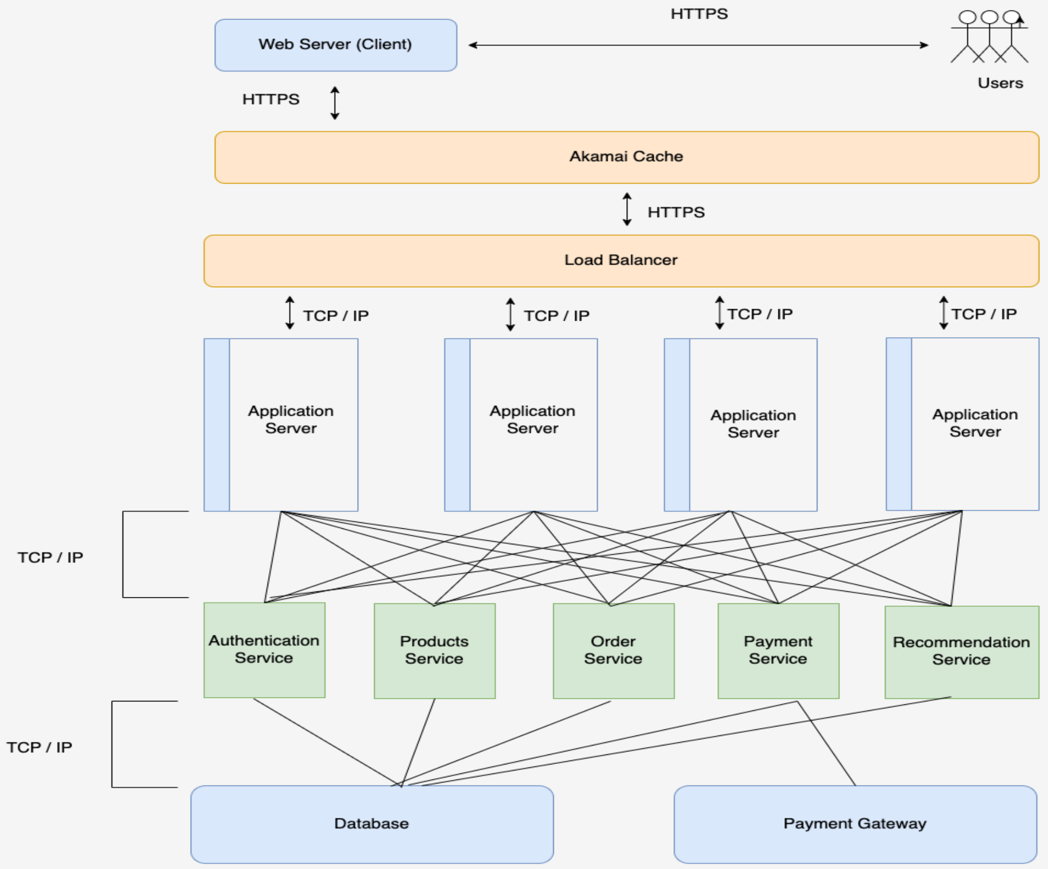
Security

- **Access Control** – Using tools like IAM (Identity Access Management), we can define access polices and limit the access based on the user role and log all access to sensitive resources for audit purposes.
- **Encryption** - Sensitive data like passwords, payment information etc are encrypted in transit and at rest, ensuring only authorized entities access it, using AES and TLS mechanisms.

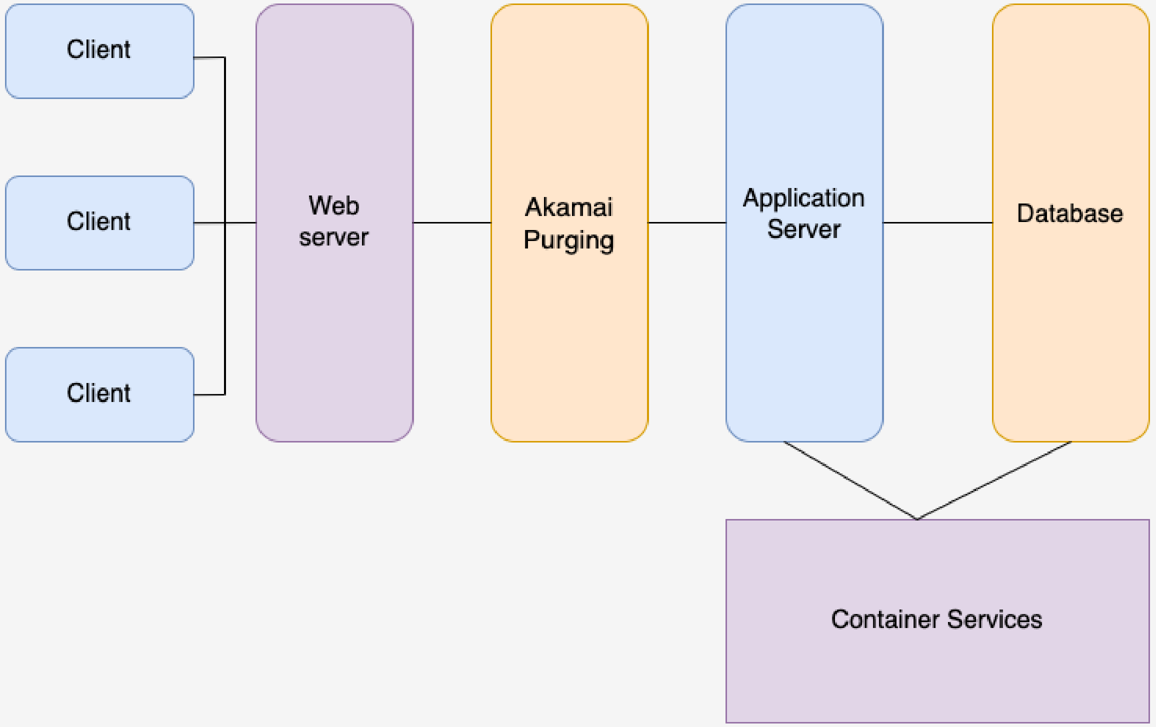
Availability

- **Redundancy -** Redundancy minimizes the risk of total system failure by ensuring there’s always a backup in place for critical components. This is especially important for an e-commerce platform, where downtime can result in lost sales. Deploying multiple instances of application servers and databases across different regions.
- **Non-stop Forwarding** – Load balancer is a strategy used to ensure that user requests are consistently forwarded to available application servers and databases, even in the event of server failures or disruptions.

Component Connection Diagram

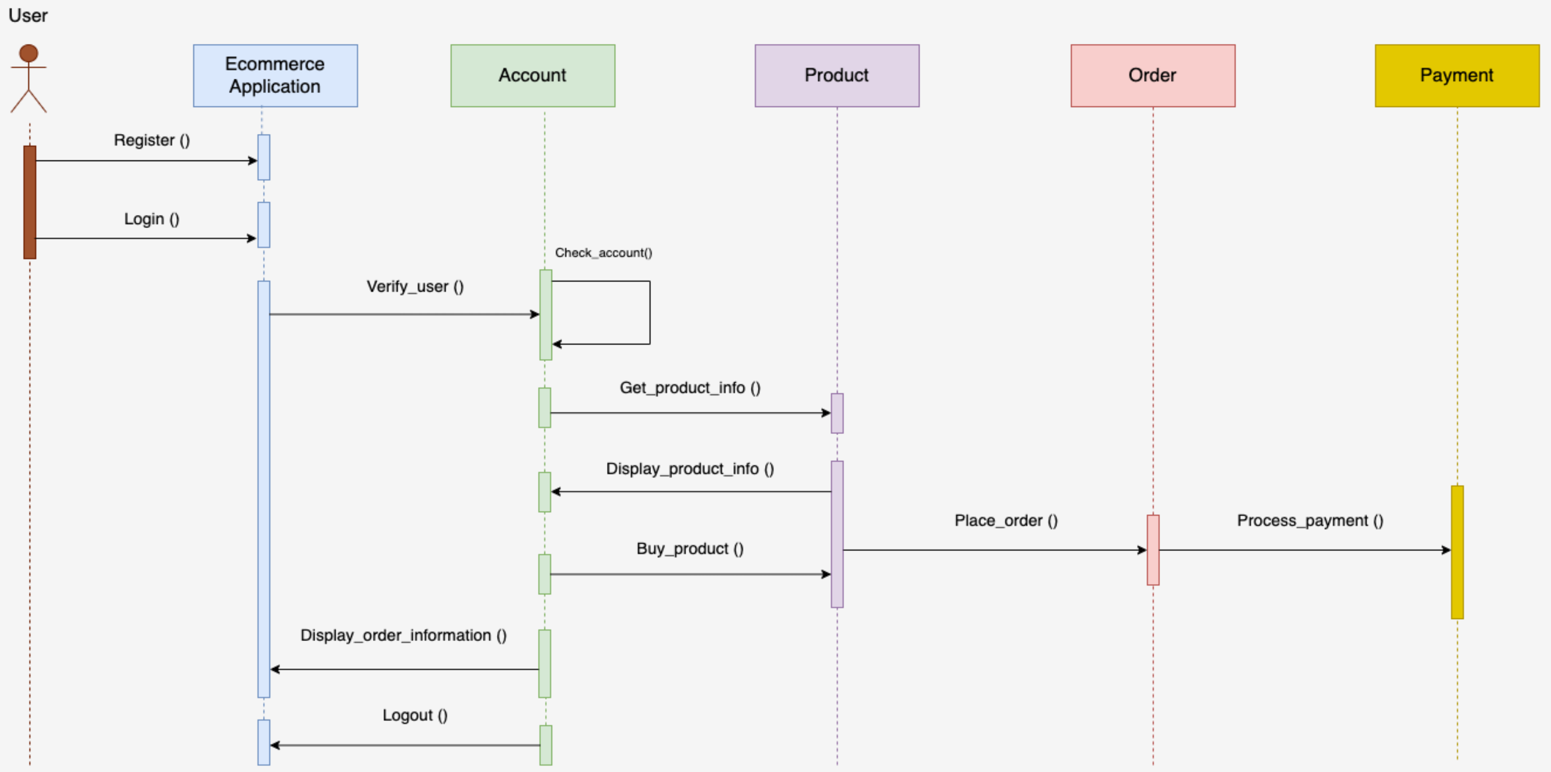


Deployment Diagram



Sequence Diagram

Scenario: Logged in user browsing products and placing an order



Architecture Patterns used

Client Server Pattern:

- The system follows the client server pattern where in which there is clear separation between the client and server and they are communicated using REST API's. When a user performs an action (e.g., searching for vehicles, comparing options), the frontend sends a request to the backend services through communication protocols.
- With the pattern, we can be able to scale up server during peak times with out effecting the client and also server and client can be managed individually, thus by ensuring Performance and Maintainability.
- Sensitive operations (like payment processing) and data storage are kept on the server, ensuring better control over data integrity and security.

Microservices Pattern:

- Backend of this ecommerce application follows micro services pattern, where in which critical services like Order processing, Payments, Inventory, Notifications service etc are developed as an independent services, each responsible for specific functionality.
- Each service can be scaled up/down based on the demand, and updates or failures of one system won't be impacting others thus ensuring Availability, Performance and Maintainability.

Key Learnings

- Got a good understanding on different software architecture diagrams and their need to describe the requirements.
- Planning the non-functional requirements such as scalability, availability, performance, security, modifiability, etc is equally important to planning the functional requirements.
- Understanding ASRs is crucial for defining the critical quality attributes that the system must meet, such as performance, scalability, security, and usability.
- Sequence diagrams for different use cases give developers a clear idea on what is expected from the system and all the players involved in that use cases.
- The component connection diagrams helps to visualize implementation strategies much earlier in system designing and makes planning much easier.
- The deployment view gives a clear image to operation teams on how the application modules are deployed and helps in rough estimation of cost early on and eases planning.

THANK YOU

Mallidi Akhil Reddy

2024TM93056