



SOFTWARE ARCHITECTURES

Assignment - 1

Submitted By:

Mallidi Akhil Reddy

2024TM93056



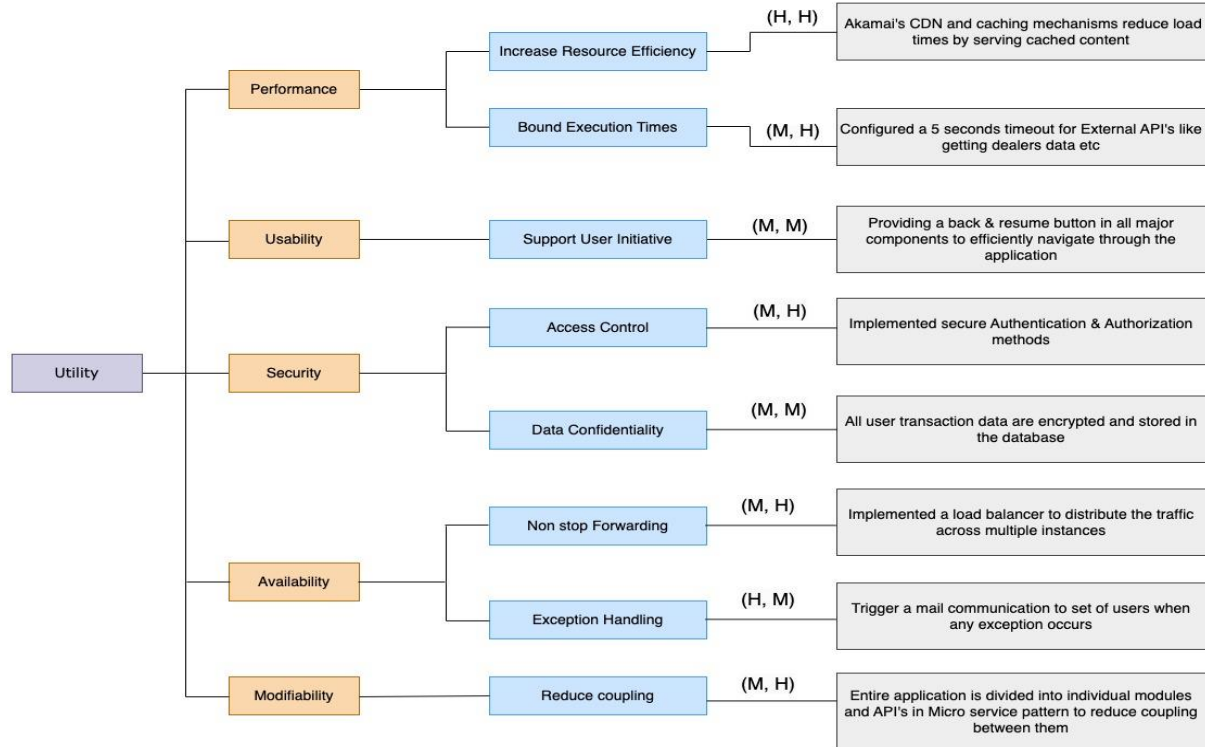
Purpose of the System

- The purpose of this web application is to deliver a highly engaging and user-centric platform for vehicle buyers.
- Users can navigate through a visually rich interface to explore a diverse inventory of vehicles.
- The application allows for easy comparison of different vehicle models, features, and specifications, helping users make informed decisions.
- The system integrates location-based services to help users find and connect with dealers in their locality, facilitating easier access to vehicles, test drives, and service options.
- Users can create an account, save their vehicle configurations, build and price settings, personal preferences, also supports seamless order placement online, and can track its status.
- The system provides a dynamic and interactive interface for visualizing and modifying vehicle features according to user preferences.
- The platform provides real-time information about current offers and promotions based on vehicle trims and configurations.

Key Requirements – Functional & Non-Functional

Functional	Non-Functional
Login to save build & price configurations	Performance
Browse the inventory of vehicles	Usability
Compare different models & trims of vehicles	Security
Search for inventory at nearby dealers	Modifiability
Sign up for updates or test drives	Availability
Check for offers based on trims & models	
Customize vehicles with various configurations	
Place and track orders	

Utility tree - Architecturally Significant Requirements (ASR)



Tactics used to achieve the top 5 ASR's



Performance

- **Increase Resource Efficiency** – Caching with Akamai CDN reduces the load on origin systems by caching static content like images, videos of vehicle etc which often decreases latency.
- **Bound Execution Time** - A timeout helps release resources quickly if any external API call is not successful within the expected timeframe, allowing the server to handle other requests efficiently.



Security

- **Access Control** – Users have assigned roles, and each role has specific access restrictions.

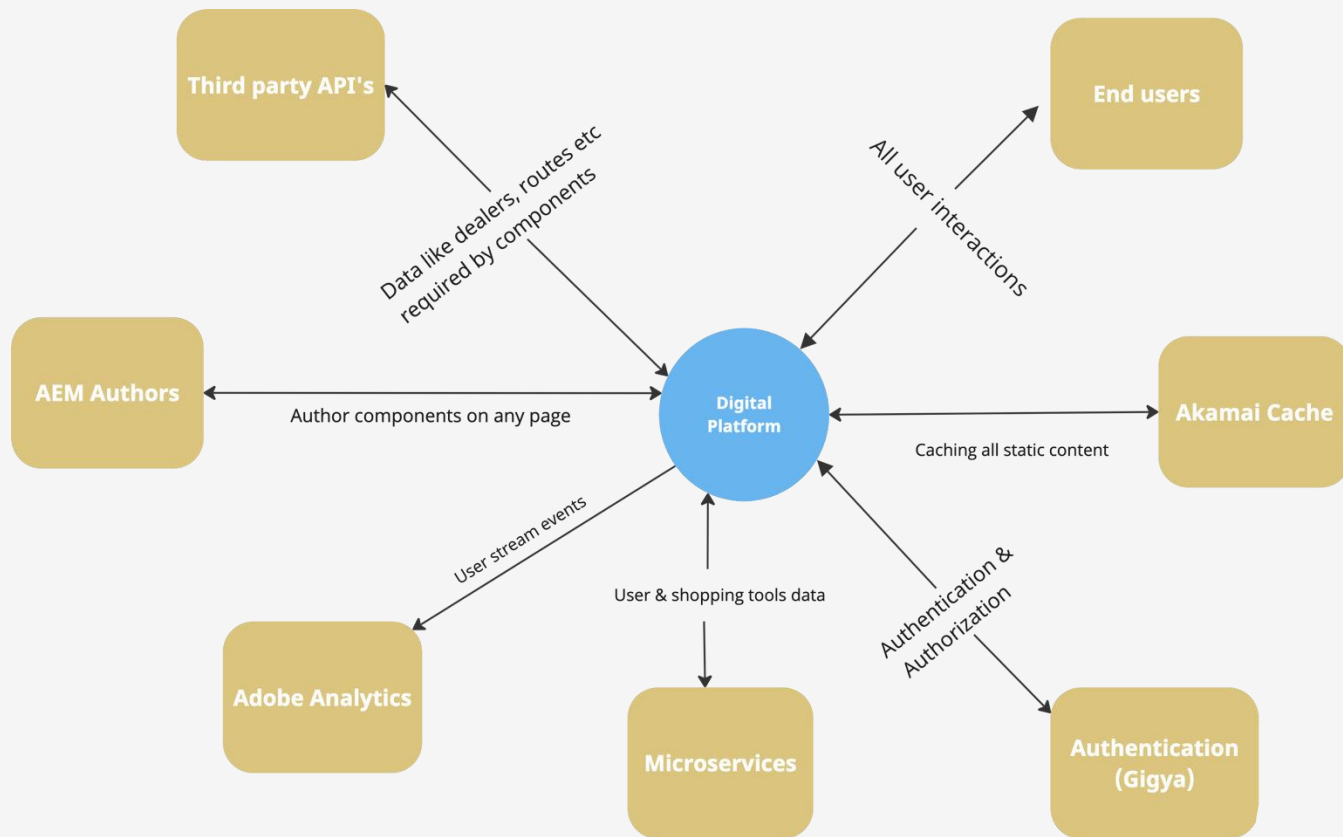
Availability

- **Non-stop Forwarding** – Load balancer is a strategy used to ensure that user requests are consistently forwarded to available backend servers, even in the event of server failures or disruptions.

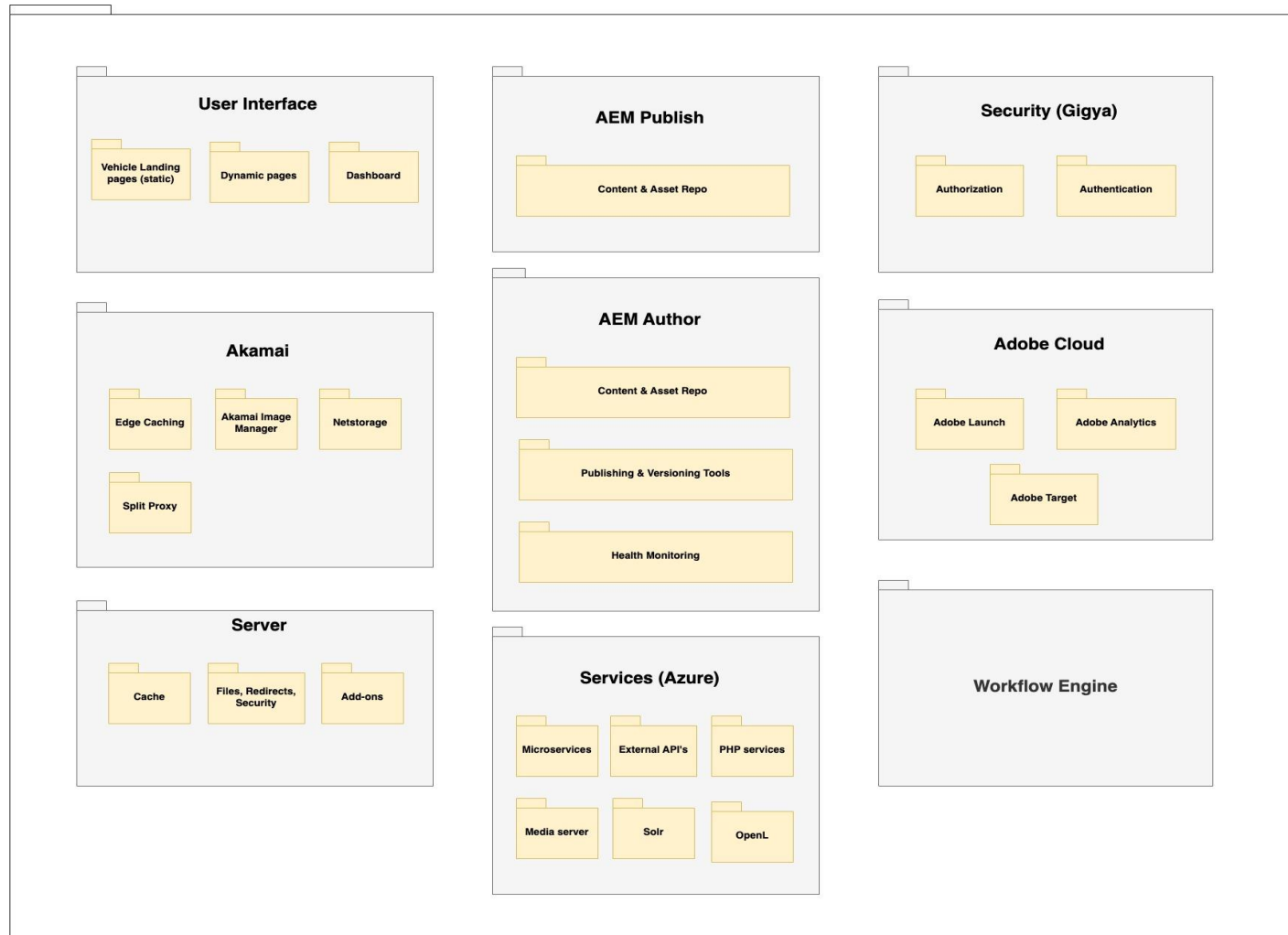
Modifiability

- **Reduce coupling** – Services are implemented in a micro service pattern, so that loose coupling ensures these services to be deployed or scaled without impacting other services.
- 
- 

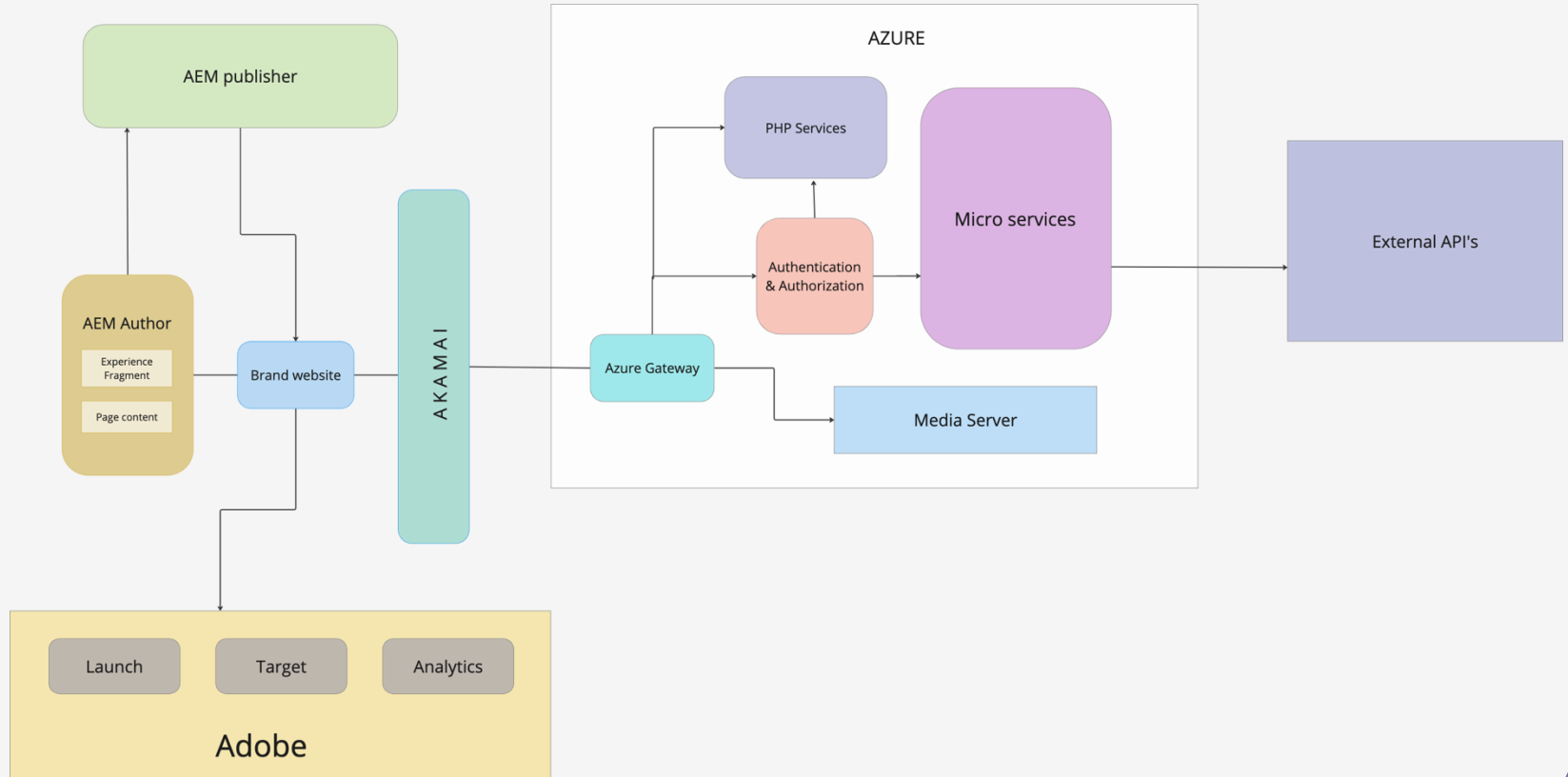
Context Diagram



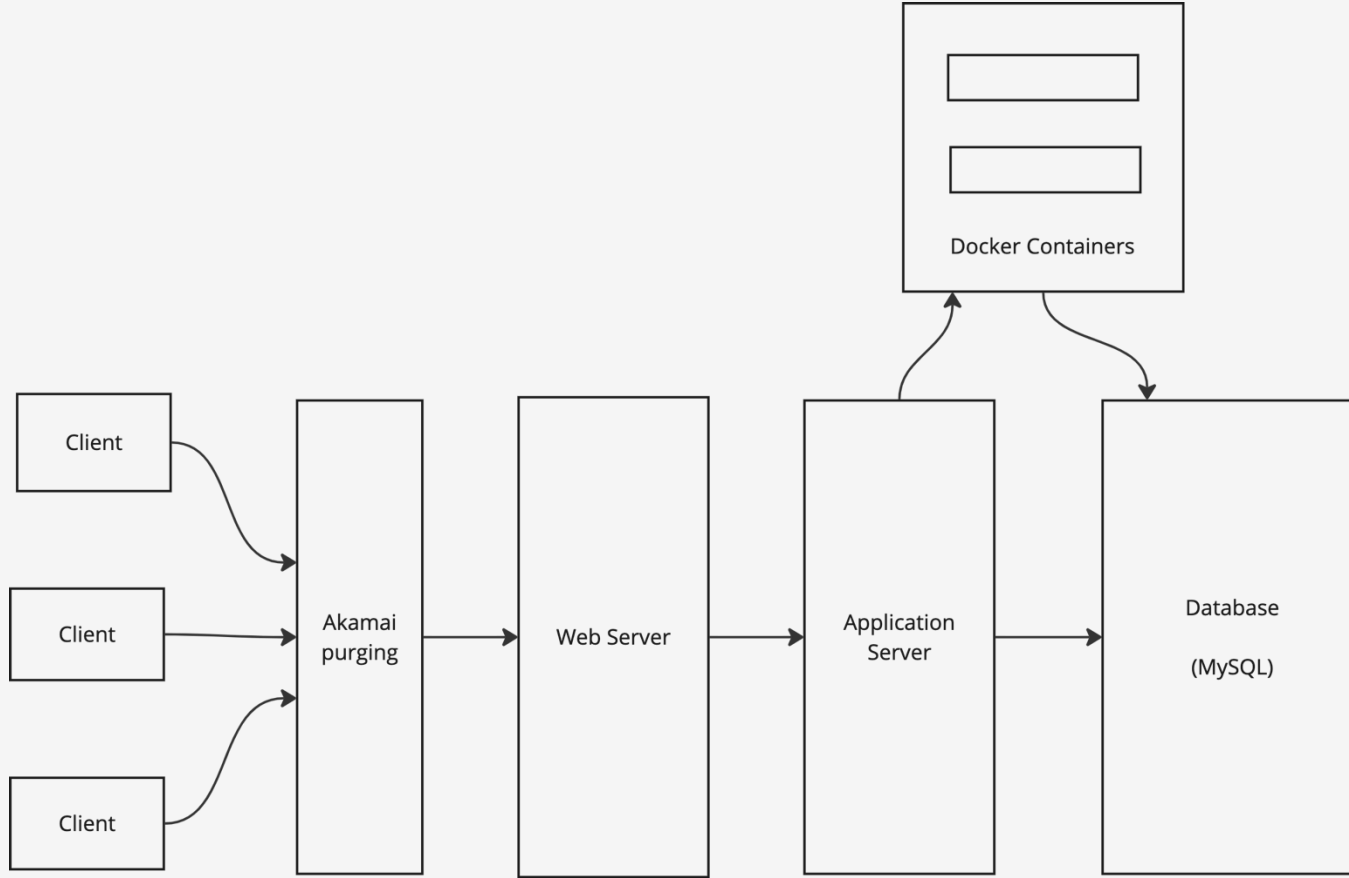
Module Decomposition Diagram



Component Connection Diagram



Deployment Diagram



How the system works

- The UI is designed with Adobe Experience Manager (AEM) components, ensuring a visually appealing and user-friendly experience.
- When a user performs an action (e.g., searching for vehicles, comparing options), the frontend sends a request to the backend services through APIs.
- The backend is composed of microservices, each responsible for a specific aspect of the application like Vehicle Inventory, Comparison service, Customization, Order & User Management services etc.
- So, when user tries to access any static pages, they are served from Akamai cache to reduce to load on server. In case of dynamic pages like Customization, Order, User profile etc, we route the requests to server with load balancer to handle the load.
- Users can create an account, save their vehicle configurations, build and price settings, personal preferences, also supports seamless order placement online, and can track its status.
- The application displays current offers and promotions based on vehicle trims. This data is retrieved from a promotional service and presented to users dynamically.
- Adobe User Stream Analytics collects data on user interactions, which is analysed to improve user experience and functionality.

Key Learnings

- Gathering Functional and Non-Functional requirements of the systems and determine the goal of System.
- Planning the non-functional requirements such as scalability, availability, performance, security, modifiability, etc is equally important to planning the functional requirements.
- Got a good understanding on which tactics we need to use to achieve any quality attributes.
- Understanding ASRs is crucial for defining the critical quality attributes that the system must meet, such as performance, scalability, security, and usability.
- Preparing Utility Tree based on the Architecturally Significant Requirements (ASR's).
- Got a good understanding on how to create a context diagram, component connection diagram, Module decomposition diagram and deployment diagram.
- With Utility diagram, breaking down high-level quality attributes into specific requirements helps in pinpointing which architectural tactics to employ.

The slide features decorative hexagonal patterns in the corners. The top-right and bottom-left corners have light blue and teal hexagons connected by thin lines. The bottom-right corner has teal hexagons connected by thin lines. The text "THANK YOU" is centered in a bold, dark blue font.

THANK YOU

Mallidi Akhil Reddy

2024TM93056