Architecture Document for

TMT East 1 – Ad Sales Insights Provider

AD Sales Insights Provider-AD-1

virtusa

TABLE OF CONTENTS

1 INTRODUCTION – PART-A 1

1.1 PURPOSE 1
1.2 SCOPE 1
1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS 1
1.4 REFERENCES 1

2 ARCHITECTURAL GOALS AND CONSTRAINTS - – PART-B 2

2.1 REUSABILITY 3
2.2 SCALABILITY 3
2.3 CUSTOMIZABILITY 4
2.4 EXTENDIBILITY 4
2.5 USE OF EXISTING BUSINESS LOGIC 4
2.6 TIME TO MARKET 4
2.7 PORTABILITY 4
2.8 AVAILABILITY 4

3 PRODUCTIZATION ASSESSMENT – PART-B 5

3.1 RE-USABLE COMPONENTS 5
3.2 ANALYZE ARCHITECTURAL FRAMEWORKS IN REPOSITORY 5
3.3 IDENTIFY AND ANALYZE OPEN SOURCE AND COTS PRODUCTS 6

4 SYSTEM ARCHITECTURE – PART-A/B 6

4.1 OVERVIEW - – PART-A 6
4.2 LOGICAL/FUNCTIONAL VIEW - – PART-A 7
4.3 USE CASE VIEW – PART-A 8
4.4 DEPLOYMENT VIEW – PART-B 9

5 Alternative Solutions Considered – PART-B 10

Document Revisions

Date	Version	Description	Author
02/05/2023	01	Architecture Document Part - A	Team - Vital Virtuosos
28/05/2023	02	Architecture Document Part - B	Team - Vital Virtuosos

virtusa

1 Introduction – PART-A

1.1 Purpose

This document provides a comprehensive architectural overview of the **Ad Sales Insights Provider** system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions, which have been made on the system.

1.2 Scope

Depending on its unique characteristics and capabilities, Ad Sales Insights Provider system may have a wide range of applications. The goal is to assist organizations in optimizing their advertising sales by offering insights and analytics that may guide strategy creation and decision-making.

The scope of this document includes audience insights, sales forecasting, competitive analysis and campaign optimization. It affects advertisement strategy, data management and user experience. This doesn't include ad creation, ad placement and ad pricing.

1.3 Definitions, Acronyms and Abbreviations

Ad Sales Insights Provider: A system that offers data-driven insights and suggestions to assist companies optimize their advertising campaigns.
Demographic Data: Information about a population's characteristics, such as age, gender, income, level of education, etc.
Ad – Advertisement
ROI – Return on Investment
API – Application Programming Interface
UI – User Interface
UX – User Experience
ETL – Extract Transform Load Data

1.4 References

https://www.decisionanalyst.com/whitepapers/advertisingresearch/
https://www.salesforce.com/blog/artificial-intelligence-advertising-sales/
https://github.com/topics/sales-insights
https://ieeexplore.ieee.org/document/9342304
https://ieeexplore.ieee.org/document/7583967
https://www.ama.org/marketing-news/2020-top-50/
https://www.researchgate.net/publication/324869312_The_Effect_of_House_Ads_on_Multichannel_Sales

2 Architectural Goals and Constraints – PART-B

Architectural Goals:

1. **Use of an off-the-shelf product:** The architecture should leverage existing off-the-shelf products or components to minimize development effort and time to market. This goal aims to take advantage of established solutions that provide ad sales insights capabilities, rather than building everything from scratch.
2. **Portability:** The architecture should be designed to ensure portability across different platforms, such as desktop, web, and mobile. This goal aims to enable users to access ad sales insights from various devices and environments seamlessly.
3. **Distribution:** The architecture should support efficient distribution of ad sales insights to multiple clients or users. This goal ensures that the insights can be disseminated to the intended recipients in a timely and scalable manner.
4. **Reuse:** The architecture should facilitate the reuse of common components and services across different modules or subsystems. This goal aims to promote code reusability, reduce duplication, and enhance maintainability.

Architectural Constraints:

1. **Development tools:** The architecture should be compatible with the development tools and technologies mandated by the organization. This constraint ensures consistency and ease of development across the organization.
2. **Legacy code:** The architecture should be capable of integrating with existing legacy systems or codebases. This constraint acknowledges the presence of legacy systems or code that need to be incorporated into the ad sales insights provider.
3. **Performance requirements**: The architecture should meet specific performance requirements, such as response time, throughput, and scalability. This constraint ensures that the ad sales insights provider can handle the expected workload and deliver insights in a timely manner.
4. **Security and privacy:** The architecture should incorporate robust security measures to protect sensitive data and ensure user privacy. This constraint is particularly important for handling ad-related data, which may contain personal or confidential information.

Architectural Assumptions:

1. The organization has conducted a thorough evaluation of available off-the-shelf products and determined that they meet the necessary requirements for the ad sales insights provider.
2. The ad sales insights provider needs to be accessible on multiple platforms to cater to the diverse needs of users.
3. The ad sales insights provider serves a large number of clients or users who require real-time or near-real-time access to the insights.
4. The ad sales insights provider consists of multiple modules or subsystems that share common functionalities or services.
5. The organization has specific development tools and technologies in place, and the architecture needs to align with those choices.
6. The organization has legacy systems or codebases that are critical for the functioning of the ad sales insights provider and must be integrated into the architecture.
7. The organization has defined performance targets or requirements that the architecture needs to satisfy.
8. The organization operates in a regulated environment with data protection and privacy requirements.

2.1 Reusability

To ensure reusability of UI components for the second phase of the back-end data server enhancement project, the UI system should be designed with modularity in mind. This includes creating separate and independent modules for the UI components that can be reused in different projects. The core UI system should be decoupled from the specific data server implementation, allowing it to be easily integrated with different RDBMS-based enhancements in the future.

2.2 Scalability

The system should be designed to scale well, considering the limitations of the current application and flat file system. This can be achieved by implementing a distributed architecture that allows for horizontal scaling. Components such as data processing and analysis should be designed to handle large volumes of data efficiently. Additionally, the system should be able to dynamically allocate resources based on demand to ensure optimal performance and scalability.

2.3 Customizability

The UI system should provide customization options to meet the specific needs of different users or organizations. This can include features such as configurable layouts, color schemes, branding, and user preferences. The architecture should support flexible customization without compromising the core functionality and usability of the system.

2.4 Extendibility

The system should be designed to accommodate the addition of new features and support modular development. This can be achieved by using a modular architecture, where components can be added or modified independently without impacting the overall system. The architecture should allow for easy integration of new modules or enhancements, facilitating future extensibility and reducing the impact of changes on existing functionality.

2.5 Use of Existing Business Logic

To leverage the existing application's business logic, the system should encapsulate the business logic in reusable components such as a DLL (Dynamic-Link Library). This allows the business logic to be easily integrated into the UI system, promoting code reuse and maintaining consistency with the existing application.

2.6 Time to Market

Considering the time constraints for market release, technology choices should be evaluated based on their ability to expedite development. Technologies such as VB COM (Component Object Model) or ATLs (Active Template Library) that offer rapid development and deployment capabilities can be considered. However, it is important to balance time to market with long-term maintainability and scalability requirements.

2.7 Portability

To ensure portability across different operating systems (OS) and databases, the architecture should be designed with platform-independent technologies and frameworks. This can involve using standards-based protocols and APIs, avoiding OS or database-specific dependencies, and implementing abstraction layers to handle variations in underlying platforms.

2.8 Availability

The system should meet uptime constraints and have fault tolerance mechanisms in place to ensure high availability. This can include implementing redundancy at the infrastructure level, such as load balancing and failover mechanisms. Additionally, fault-tolerant designs and automated monitoring can help detect and recover from failures, minimizing downtime and maximizing system availability.

3 Productization Assessment – PART-B

3.1 Re-Usable Components

Based on the requirements document, the Productization specialist has performed a reuse analysis to identify reusable components. The following components have been identified:

1. **User Interface (UI) Components: The** UI components can be designed in a modular and reusable manner, allowing them to be utilized in the second phase of the back-end data server enhancement project. The core of the UI system should be reusable, enabling easy integration with the RDBMS-based enhancement project.
2. **Business Logic Components**: The existing application's business logic can be extracted and encapsulated into reusable components, such as a DLL (Dynamic Link Library). These components can be reused in the architecture, ensuring consistency and leveraging established business rules and logic.
3. **Data Access Components:** Components responsible for accessing and retrieving data from the database or other sources can be designed as reusable modules. This allows for efficient and consistent data retrieval across the system.

Commonality and Variability Analysis:

The commonality and variability analysis assesses the degree of common functionality shared among the identified reusable components and identifies the points of variation. It helps determine the level of abstraction needed to create reusable modules. For example, in the case of UI components, the common functionality may include basic user interaction elements, while the points of variation could be specific layout requirements or branding customization. By analyzing these commonalities and variabilities, a flexible and reusable UI framework can be established.

3.2 Analyze Architectural Frameworks in Repository

The Productization specialist has analyzed the architectural frameworks available in the repository that can be reused in the current architecture under discussion. This analysis helps identify frameworks that provide pre-defined structures, patterns, and best practices for developing specific components or functionalities.

By leveraging existing architectural frameworks, development effort can be reduced, and consistency can be maintained across different components of the system. The analysis should consider the compatibility of the frameworks with the technology stack and requirements of the ad sales insights provider for TV.

3.3 Identify and Analyze Open Source and COTS Products

The Productization specialist has identified open-source and commercial-off-the-shelf (COTS) products that can be utilized in the project. This includes analyzing the suitability of these products for meeting the project's requirements and their compatibility with the overall architecture.

Additionally, the specialist has documented any copyright or legal approval needs associated with the identified open-source or COTS products. This ensures compliance with licensing and legal requirements, protecting the intellectual property rights of the system and avoiding any potential legal issues.

4 System Architecture – PART-A/B

System Architecture of Ad Sales Insights Provider depends on specific capabilities of the system. It includes data collection, data preprocessing, data analytics using machine learning modes and data visualization tools to provide insights to be reported on the dashboard.

4.1 Overview – PART-A

The application is divided into three tiers: User Interface Services, Business Services and Data Services. Each layer is responsible for handling a well-defined area of the application.

User Interface Services: This layer manages the application's user interface, including user input and output display. It has direct user interaction and converts what the user does into instructions that the other levels may follow. Web pages, user forms, mobile applications, and other user interface elements may be included in this layer.

Business Services: This layer is in charge of managing the application's business logic, including the procedures and rules that direct the behavior of the application. It conducts actions including data validation, computations, and decision-making on the data it receives from the user interface layer. Workflow engines, business logic modules, and other processing elements could be present in this layer.

Data Services: The storage and retrieval of data utilized by the application are handled by this layer. The application may function with many data storage systems without having to change the business logic layer because it offers an abstraction layer between the business logic layer and the underlying data sources. Database management systems, file systems, and other components for data storage could be included in this layer.

As the changes to one layer can be made without having an impact on the other layers, this separation of concerns makes it simpler to develop and maintain the application over time. Each layer can be scaled independently of the others. This makes the application flexible and scalable, enabling it to adapt to changes in user demand and data volume.

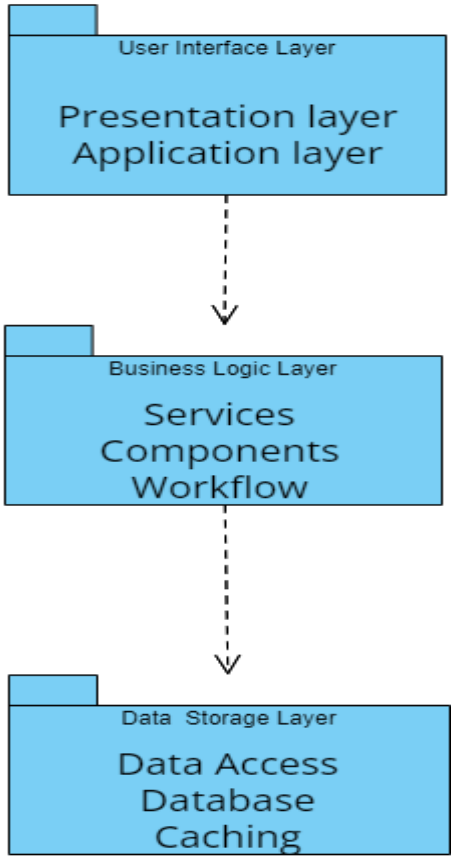
4.2 Logical/Functional View – PART-A

User Interface Layer: The Presentation Layer and the Application Layer are the two logical parts that make up the User Interface Layer. While the Application Layer manages user input and requests, the Presentation Layer is in charge of providing information to the user in a user-friendly manner. Through established interfaces, the User Interface Layer communicates with the Business Logic Layer.

Business Logic Layer: Services, Components, and Workflows are the components of the Business Logic Layer. These logical parts are in charge of handling user input, carrying out business logic, and providing insights and suggestions to advertisers. Through established interfaces, the Business Logic Layer communicates with the User Interface Layer and the Data Storage Layer.

Data Storage Layer: Database(s), Caching, and Data Access make up the Data Storage Layer. These logical parts are in charge of handling and storing the system's data. Through established interfaces, the Data Storage Layer communicates with the Business Logic Layer.

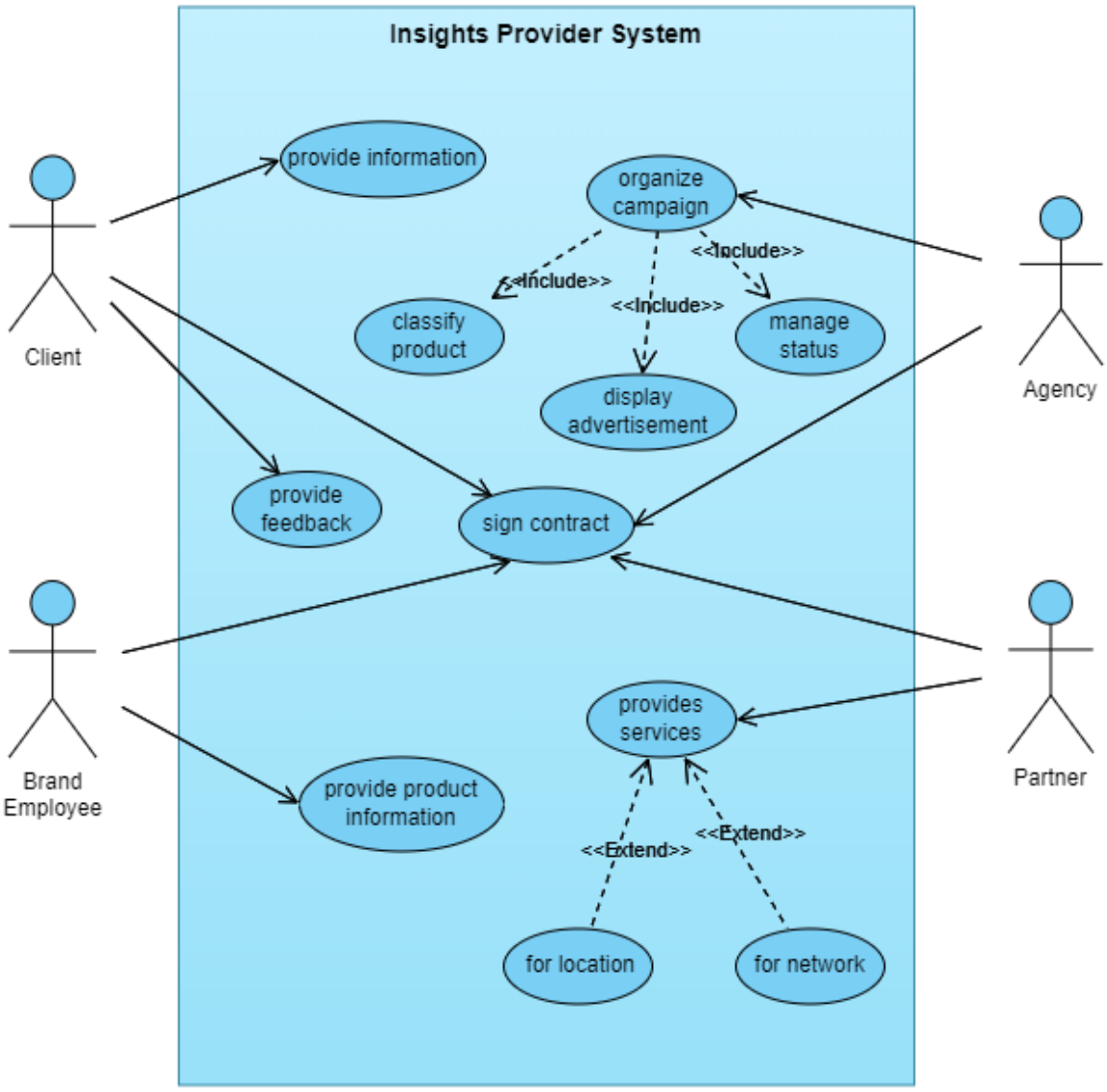
Package and Layering Diagram:



virtusa

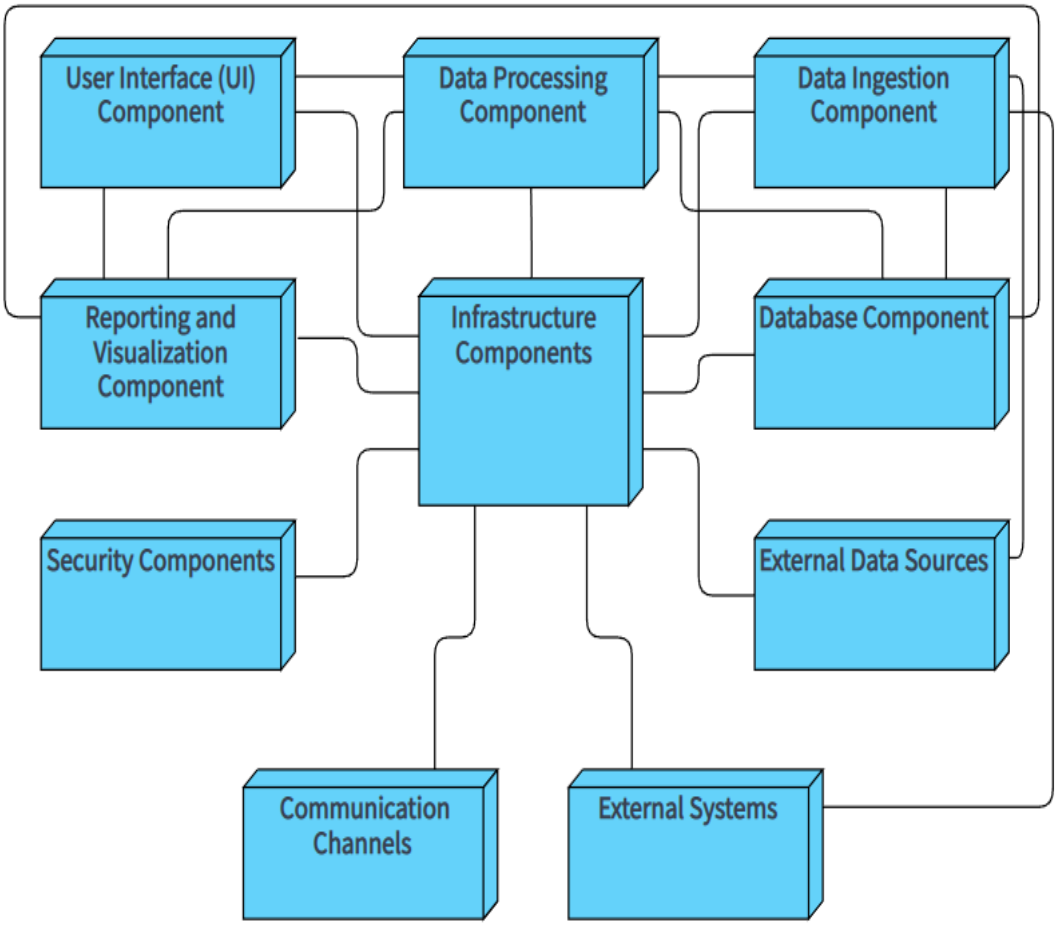
4.3 Use Case View – PART-A

This diagram demonstrates how different stakeholders in the ad sales ecosystem can benefit from using the ad sales insights provider system to gain visibility into their campaigns' performance, identify areas for improvement, and collaborate more effectively with their partners. By leveraging the system's data and analytics capabilities, more informed decisions can be made and better results can be driven for advertising efforts.



The use cases involved are client, agency, brand employee, and partner.
Client: Uses the ad sales insights provider system to track the performance of their advertising campaigns across different platforms and channels.
Agency: Uses the system to monitor the campaigns they are running on behalf of their clients, and to provide them with reports and insights on their performance.
Brand Employee: Uses the system to access the performance metrics of their own campaigns and those of their competitors, as well as to identify opportunities for optimization and growth.
Partner: Uses the system to monitor the performance of campaigns they are running in partnership with the client or agency, and to collaborate with them on optimization strategies and tactics.

4.4 Deployment View – PART-B



5 Alternative Solutions Considered – PART-B

1. **Monolithic Architecture:** One alternative solution considered was a monolithic architecture where all the components of the ad sales insights provider, such as data ingestion, processing, reporting, and visualization, are tightly integrated into a single application. In this approach, the entire system would be deployed and scaled as a single unit.
2. **Microservices Architecture:** Another alternative solution considered was a microservices architecture where the system would be divided into smaller, independent services, each responsible for a specific functionality. Each service would have its own database and communicate with each other through APIs. This approach provides flexibility, scalability, and ease of maintenance.

Rationale for Selecting the Current Solution:

The current solution for the ad sales insights provider for TV is based on a microservices architecture. Several factors contributed to the selection of this architecture:

1. **Scalability:** The microservices architecture allows for scaling individual components based on demand. It enables horizontal scaling of specific services that experience high loads, ensuring optimal performance and resource utilization.
2. **Modularity and Reusability:** By breaking down the system into smaller, independent services, the architecture promotes modularity and reusability of components. Each service can be developed, tested, and deployed independently, enabling agile development practices and facilitating future enhancements.
3. **Fault Isolation and Resilience:** In a microservices architecture, if a particular service fails, it does not bring down the entire system. Fault isolation and resilience are achieved as services can handle failures independently, minimizing the impact on other components.
4. **Technology Diversity:** With a microservices architecture, different services can be developed using different technologies and programming languages, allowing flexibility in choosing the best tools and frameworks for each specific task. This helps optimize performance, utilize specialized skills, and integrate with diverse data sources.
5. **Scalable Data Storage**: The microservices architecture allows for the use of different databases for each service, based on specific requirements. This flexibility enables the selection of appropriate data storage technologies that can efficiently handle the volume and variety of data generated by the ad sales insights provider.

6. **Future Extensibility:** The modular nature of the microservices architecture makes it easier to introduce new features and functionality in the future. Additional services can be developed and integrated without impacting the existing components, promoting extensibility and adaptability to evolving business needs.

Overall, the selection of a microservices architecture for the ad sales insights provider for TV offers the benefits of scalability, modularity, fault isolation, technology diversity, and future extensibility. It aligns with the objectives of the system and provides a flexible and robust foundation for delivering efficient and actionable insights to users in the TV ad sales domain.