Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

Aligning Software Architecture with Cost Structure: A Comparative Study Using ATAM and Lean Canvas in Early Startup Development

Jan falih Fadhillah[*1], Dana Sulistiyo Kusumo[2]

[1,2]Computer Science, Telkom University, Indonesia

Email: [1]janfalih@student.telkomuniversity.ac.id

Received : Jan 10, 2025; Revised : Jan 25, 2025; Accepted : Feb 5, 2025; Published : Dec 23, 2025

Abstract

Startups in the early phase often face challenges in balancing operational efficiency with resource constraints. This research find how startups can choose software architecture to align with cost structures with the Lean Canvas framework and the Architecture Trade-off Analysis Method (ATAM). Lean canvas allows for startups to identify cost structures at an early stage and align with market demands efficiently and ATAM helps to evaluate software architecture systematically by analysing trade-offs and quality attributes. Although microservice architecture offers modularity and scalability, its implementation can lead to higher operational costs making it unsuitable for startups with limited budgets. On the other hand, monolithic architecture is more cost-effective, easy to manage and suitable for the needs of early-stage startups. This research emphasizes that systematic evaluation of software architecture based on business goals and resource limitations is essential for startup growth for sustainability. By combining Lean Canvas for business validation and ATAM for architectural decision making, startups can optimize operational and technical strategies, analyse risks, and identify trade-offs that are implemented according to business development.

Keywords : *Cost Structure, Lean Canvas, Microservices, Monolithic Architecture, Software Architecture, Startup.*

1. INTRODUCTION

This research emphasizes the importance of an architecture that is appropriate to the cost structure in startup development at the early phase. With ATAM and Lean Canvas, this research adapts how software architecture can optimize and support efficiency in business operations. Lean Canvas helps startups identify key elements such as cost structures while ATAM provides a systematic framework for evaluating architectural decisions based on real scenarios and the quality attribute requirements of both methods. enabling startups to not only reduce the risk of architectural decisions but also ensure that architectural choices align with business needs. Early-phase startups often face unique challenges, one of which is the tendency to push new ideas without fully understanding and validating market needs [1], [2], [3]. This approach, while innovative, can result in significant risks and inefficiencies, especially when the idea fails to address a real customer problem [2]. To maximize opportunities and validate ideas, startups adopt the Lean Canvas framework, Lean Canvas is adopted from Business Model Canvas (BMC) [4]. Lean Canvas allowing them to formulate and test assumptions about the business model [4]. Iterating on Lean Canvas allows hypotheses to be tested efficiently, and feedback can be obtained to adjust by aligning the proposed solution with the actual market demand as discussed in these research [2], [4]. Innovative startups face serious difficulties in their innovation processes because of a scarcity of financial resources [5]. According to [6], for a business to have a successful strategy, it must both produce products at the lowest possible costs and control costs continuously at all levels of an organization, including production, marketing, and non-marketing support functions. Lean Canvas is

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

not only for identifying problems or solutions but also for identifying cost structures [3], which is essential for the sustainability of business operations. Cost reduction is a critical factor for the success of any business [7], especially for startups with limited resources. According to [6], "the role of costs is mainly addressed as cost structure in relation to revenue and profitability". Many startups burning money to create brand awareness and generate loyalty [8]. This highlights the importance of understanding cost structure is essential for managing expenses and ensuring operational efficiency. But the success of this process also depends on other processes such as selecting the right software architecture. Software architecture in startups plays an important role in optimizing application attributes such as scalability and application performance in supporting customer demand.

For startups, choosing the right software architecture is quite challenging. Because they develop innovative software-intensive products under time constraints and with a lack of resources [9]. Many organizations, from large tech companies (e.g., Google, Amazon, Netflix) to small startups, have adopted microservices as a best practice [10], [11]. Startups immediately choose a microservice software architecture [11] that may not fit their cost structure. This mismatch can result in operational challenges, including increased costs and preventing startups from growing their business efficiently. Although microservice is popular software architecture according to several studies [10], [12], [13]. Microservices are not always the most cost-effective choice [10]. Startups must be able to carefully evaluate whether the complexity and higher operational costs can accommodate microservices [10],[12]. Or whether other architectures such as monolithic may be more appropriate for startup needs. Indeed, microservice architecture offers clear advantages, such as modularity, scalability, and the ability to develop and deploy components independently [10], [12]. This is attractive to businesses looking for rapid growth. However, as noted in the study. This can result in increased costs for infrastructure, maintenance, and development. For startups operating on a limited budget, these costs can quickly become expensive, resulting in inefficiencies and challenges in growing their business. On the other hand, monolithic architecture can be a consideration. Monolithic architecture, although less flexible and scalable than microservices, offers significant advantages in cost efficiency as discussed in these research [12], [13], [14]. Monolithic combines all code bases into one, making it easier to develop [10],[15]. For early-phase startup development whose needs are still few, monolithic architecture is a good choice. This architecture can reduce operational costs, minimize complexity, and simplify system maintenance, the main factors that are important for early-phase startup development.

To ensure that the chosen architecture is in accordance with the operational needs of the startup, systematic evaluation is very important. ATAM is a systematic way to assess the consequences of architectural decisions considering quality attribute requirements [16],[17]. With the Analysis Tradeoff Architecture Method (ATAM) approach, this approach has been proven to evaluate software architectures based on predetermined scenarios and quality attributes [18], [19], [20]. ATAM allows startups to assess the trade-offs involved in software architecture decisions and understand the implications for cost, scalability, and other important factors. By implementing ATAM, startups can identify risks and prioritize software architecture choices that are in accordance with their business goals and resource constraints. For example, startups can use ATAM to evaluate whether a monolithic or microservice architecture is more suitable for their needs by considering factors such as expected traffic, team expertise, budget constraints and others. ATAM is a method that that involves multiple stakeholders, including architects, developers, testers, and users [20], [21]. In this research, ATAM also facilitates discussions among stakeholders about how architectural decisions address quality attributes [17],[19], [20]. The evaluation detailed analysis of how each software architecture performs in achieving software attributes [22]. Including aspects such as cost structure. The main challenge for startups is to strike the right balance between scalability and cost structure. While scalability is often prioritized to accommodate future growth, it is equally important to ensure the current architecture fits the cost

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

structure. In this context, leveraging cloud infrastructure [23] [24], [25] can be a strategic choice for startups, as it offers flexibility in scaling resources while managing costs effectively. By utilizing cloud services, startups can adjust their infrastructure according to their immediate needs, minimizing upfront investments and aligning costs with their growth trajectory.

2. METHOD

To solve problems in research, logical stages are needed to complete the research. The main steps are interconnected and sustainable. These stages are sequenced according to Figure 1.

Figure 1. Research Methodology

2.1. Data Collection

The data collection stage explains how data will be collected and processed according to research needs. The initial step in the data collection stage is identifying the data needed for research. Identifying data needs can provide guidance for the data collection stage. The required data and collection techniques are shown in Table 1.

Table 1. Data Collection Infromation

Data	Data Type	Data Usage	Data Collection Method
Cost Structure	Secondary	Designing a lean canvas and as a reference for architectural development	Literature Study, Observations, and Questionnaires.

1. Literature Study
   Collect information from various sources such as books, scientific journals, articles, and other publications related to software architecture, business models, ATAM methods, and Lean Canvas. This literature study helps build a strong theoretical basis for the research and identifies relevant previous findings.
2. Observation
   Observation is a data collection method in which researchers directly observe the object or phenomenon being studied. In this context, observations can be made to observe how software architecture is applied in practice, how business models are implemented in company, and how ATAM and Lean Canvas methods are used in project development and evaluation.

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

3. Questionnaire
   A questionnaire consists of a standard set of questions accompanied by a uniform answer format. Individuals who receive questionnaires are known as respondents [26]. Data obtained from the questionnaire was analysed to provide more insight into the suitability of the business model.

2.2. Designing Cost Structure

After collecting data then carried out at the cost structure design stage with a lean canvas. The stages in designing cost structure elements with a lean canvas are shown in Figure 2.

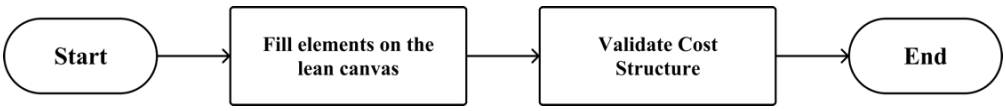

Figure 2. Designing Cost Structure Elements

2.2.1. Lean Canvas

Lean Canvas is an adaptation of the Business Model Canvas (BMC) [1], which is usually used in developing business models in startups. Because the business model canvas takes a long time to design, the business model canvas is not suitable for startup businesses that are still in the product design stage [1]. The Lean Canvas method also focuses more on business development compared to business planning [1]. For these reasons, Ash Maurya formed Lean Canvas. In the Lean Canvas, there are several sections that focus on problems faced by customers. The elements of the Lean Canvas can be seen in Figure 3.

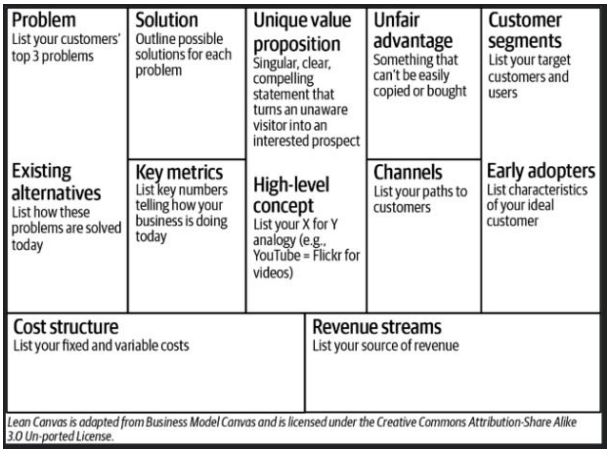

Figure 3. Lean Canvas [3]

2.3. Analysis Trade-offs Architecture Method (ATAM)

Architecture Tradeoff Analysis Method (ATAM) is the most popular architectural evaluation method [18]. ATAM is an evolved version of Scenario-Based Architecture Analysis (SAAM). Unlike SAAM, ATAM focuses on evaluating quality attributes [20]. ATAM is a general design-time architectural evaluation method that uses scenarios to assess architectural design decisions. In this research, ATAM is used as an initial evaluation to see the trade-offs that occur when using a monolithic or microservice architecture.

2.4. Analysis And Implementation

At this stage, the business model is identified to be compared with the software architecture. The development and design steps are carried out based on the chosen architecture:

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

1. Identify
   This step explains the business processes and the relationship between problems in the cost structure and quality factors that are considered scalability and performance.
2. Comparison of Software Architecture (Monolithic and Microservice)
   Compare software architectures, both monolithic and microservices, by paying attention to quality factors. The quality factors that are analysed are factors that support the business model using a utility tree. From these quality factors, a software architecture is then selected that suits the business model.

2.4.1. Monolithic Architecture

Monolithic Architecture is software design where there are different components (such as authorization, business logic, etc.) [12]. Combined into one program on one platform, it can be seen as in Figure 4.

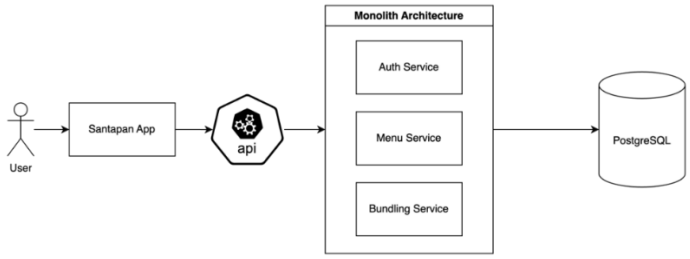

Figure 4. Monolithic Architecture

Figure 4 shows monolithic architecture of an application that provides the business processes of the application. Even though it implements many business processes throughout the software, the application is created as a stand-alone architecture in the program as discussed in these research [12], [11].

2.4.2. Microservice Architecture

Microservice Architecture is an architecture that organizes applications as collections. Each microservices must provide part of the business logic [12]. Microservices architecture is also a popular architecture in recent years [15]. Figure 5 is an example of a microservices application, this application provides business processes. There are 2 core microservices, which serve business logic and act as if it were one application using REST API.

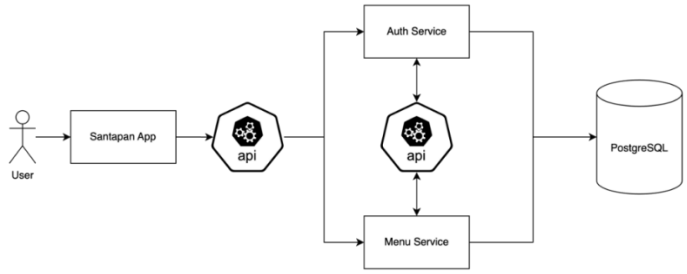

Figure 5. Microservice Architecture

3. RESULT

3.1. Experimental Object

Santapan is a technology startup that operates in healthy food delivery services. With a vision to encourage a healthy lifestyle through nutritious food, Santapan connects users with various menu

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

options provided by MSMEs in Indonesia. The service targets customers who care about health, convenience, and food quality.

Santapan

Figure 6. Santapan Logo

Using an easy-to-use mobile application, customers can order healthy meals tailored to their daily nutritional needs. Dining also offers personalization features, such as diet plans, allergies, or specific food preferences.

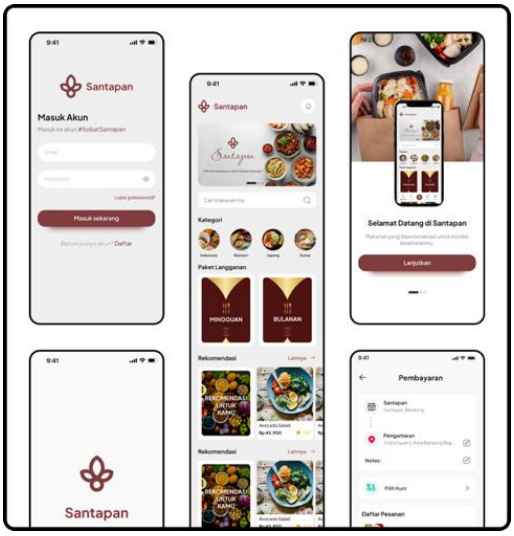

Figure 7. User Interface of Santapan Application

Figure 7 shows the user interface of the Santapan application with various services available in Santapan. Users can easily choose food that suits their needs. The following are the services available in the Santapan application:

1. Weekly and Monthly Meal Plan Subscriptions
   For customer convenience, Santapan provides a subscription service with scheduled deliveries every week or every month, ensuring healthy food is always available without the hassle of ordering.
2. Healthy Menu Choices
   Dining offers a variety of healthy menus that include options for low-calorie, high-protein, and gluten-free diets. Each menu is designed to ensure nutritional balance according to the customer's health goals.

3.2. Cost Structure Design with Lean Canvas

This step explains the business processes in the startup and the relationship between problems in the cost structure and quality factors that are considered scalability and performance. Lean Canvas provides a better framework by focusing on critical business elements, such as customer problems,

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

solutions, and key metrics. For this research, Lean Canvas was chosen for its ability to capture and analyze key business components in the context of a startup. When building an initial startup MVP, thinking about the cost structure is very important [3]. Many startups in the early stages burn their money just for brand awareness and generating loyalty [8]. Focusing on the cost structure is very important because, an early stage of a startup, managing costs effectively can have a significant impact on the sustainability and scalability of the business. Determining costs early, not only reduces risk but can optimize resource allocation, simplifying operations [6]. And ensure its software architecture is aligned with its financial goals. Stakeholders determine the cost structure with planning for the near future as in Table 2.

Table 2. Cost Component and Estimated Cost

Cost Component	Percentage or Estimated Cost	Description
Technology Deployment	50% from total cost	Development application and cloud infrastructure
Operational	30% from total cost	Customer support dan internal management
Marketing	20% from total cost	Promotion of the app to attract new users and maintain the user base

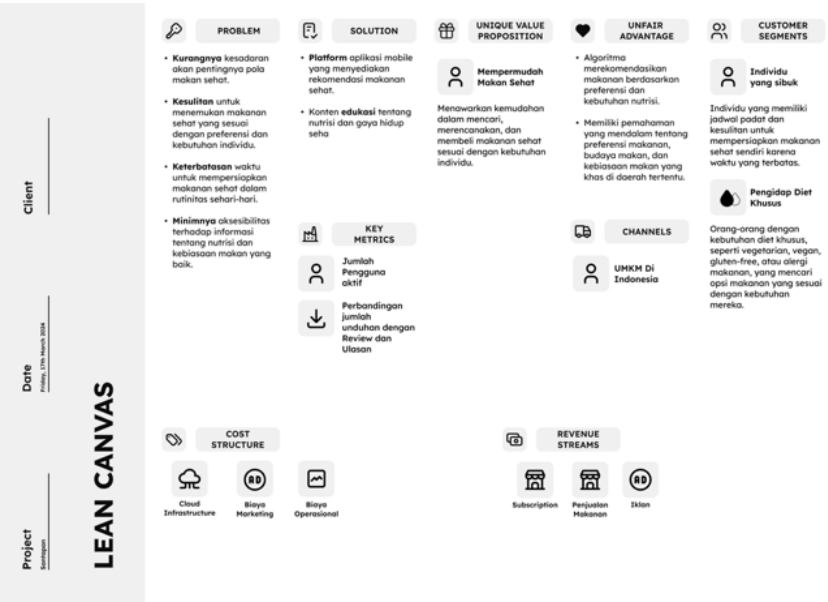

Figure 8. Lean Canvas of Santapan

In Table 4 technology deployment is focused on application development and cloud infrastructure which also affects scalability, security, and application performance. According to [23], Cloud infrastructure also offers many advantages such as scalability, flexibility, and cost efficiency. So, the relationship between cost components is also influenced by cloud infrastructure as in Table 4 and the elements in the Lean Canvas of seen through the grouping of problems, solutions, and unique value propositions as in Figure 8.

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

3.2.1. Cost Structure as a Guide for Architectural Decisions with ATAM

Referring previous research [1] "Cost Framework There are a number of expenses that startup must pay in order to run the business model". The results of the cost structure analysis on the Lean Canvas are used as a basis for evaluating business needs in the Architecture Trade-off Analysis Method (ATAM) as an influence to see trade-offs and risks in accordance with the assessment in the Architecture Trade-off Analysis Method (ATAM) which is adjusted to stakeholder needs. Table 4 shows key cost components, broken down as technology deployment (50%), operations (30%), and marketing (20%), shape the architectural priorities as follows:

1. Technology Deployment
   Technology deployment includes development costs and also cloud infrastructure, which is part of the cost structure. The large cost allocation highlights the importance of scalability, performance, and maintainability in architectural decision making. Technology deployment defined as a fixed cost, Fixed costs are expenses that are incurred consistently based on a specific time [1].
2. Oprational Cost
   Operational costs include customer support and internal management used for system continuity and maintenance. As a result, 30% of income is set aside for operational fees.
3. Marketing
   With the possibility that allocations for technology and operations will have more influence on marketing costs. Marketing costs are defined as variable costs [1]. Variable costs are more susceptible to change. These costs change depending on business performance or on a specific period [1].

3.3. Identify Needs with ATAM

Architecture Trade-off Analysis Method (ATAM) is the main goal of the startup to identify strengths, weaknesses, and risks in architectural designs and see the trade-offs used to ensure the best solution suits business and technical needs. The research was used to compare two choices of monolithic architecture and microservices along with the ATAM stages carried out.

1. Identify Stakeholder
   According to [21], "ATAM is a method that that involves multiple stakeholders, including architects, developers, testers, and users." The stakeholders in question are parties who are interested in the performance of the software architecture. They contribute to identifying architectural needs and priorities [16].

Table 3. Stakeholders Interest and Key Needs

Stakeholder	Interests	Key Needs
End-Users	Smooth, performance, and secure personal data.	Responsive, intuitive interface, data encryption and user authentication.
Developers	Ease of maintenance and development, scalability to adapt to the number of users.	Cloud architecture supports system development, uses technology that suits needs, and supports easy updates.
Management	Risk reduction and minimizes needs with real investment.	Cost-effective architecture, ensures applications match business needs.

Based on this information, non-functional requirements (Non-Functional Requirements) that must be met by the software architecture are obtained and categorized into several categories which are evaluated using the Architecture Trade-off Analysis Method (ATAM) process [19], as in Table 4. What

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

is taken from the needs of the stakeholders, so that the needs of the software architecture are summarized into Non-Functional Requirements.

Table 4. Non-Functional Requirements

Category	Description	Parameter
Scalability	The system must be able to handle an increase in the number of users without affecting performance	Supports up to 1,000 active users per day
Performance	The system must be responsive when serving user requests.	Maximum response time 4 seconds.
Security	The system must maintain the confidentiality and security of user data.	Authentication and encryption of user data.
Availability	The system accessed stably.	90% uptime.
Maintainability	Ease of debugging and architecture updates	Modularity of systems and documentation

According to [5] every startup has limited resources (time, energy, budget, and priorities) which creates serious difficulties in its innovation process. Therefore, choosing an NFR that is in line with the startup's main goals is a strategic step. In this case:

- Costs: keeping the budget under control is essential to keep the project on track without exceeding the specified costs.
- Scalability: this focus is relevant if the application or system is designed to support the growth of users or large amounts of data, which is a priority in long-term projects.
- Performance: performance is an important attribute to ensure a system runs quickly and responsively, especially when dealing with many users or complex operations.

2. Determination of Quality Attributes

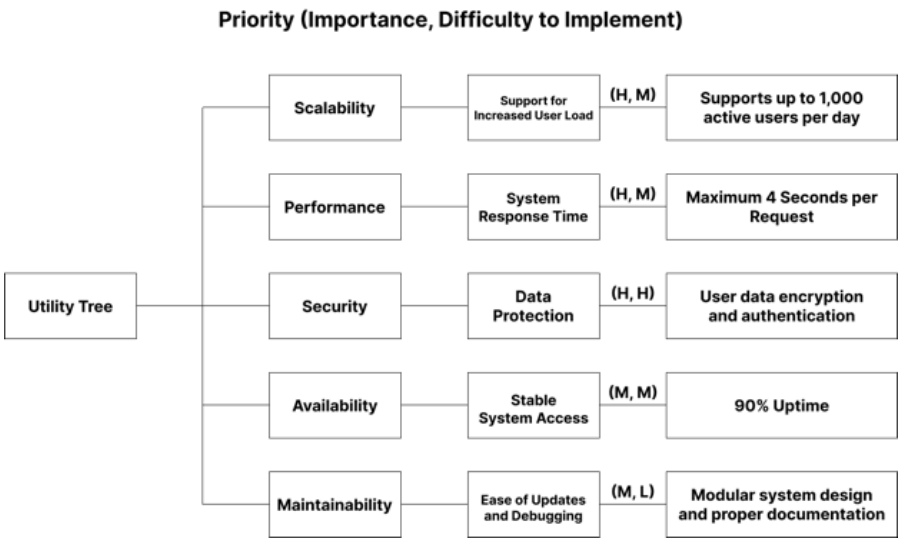

Figure 9. Utility Tree dari Quality Attributes

Setting quality attributes is an important component of the Analysis Trade-off Architecture Method (ATAM). This phase transforms stakeholder requirements and expectations into measurable quality

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

characteristics. These characteristics inform architectural choices and help ensure that the system meets business and technical objectives as in Figure 9 which describes the priority and difficulty of implementing a software architecture.

3. Quality Scenario
   The Quality Scenario phase in ATAM describes specific situations or "scenarios" to evaluate the architecture's alignment with its goals [20]. These scenarios define how the system should react to various settings, ensuring the system meets stakeholder expectations in terms of qualities such as performance and scalability.
4. Trade-off Analysis
   In Trade-off analysis, the authors evaluate the potential advantages and disadvantages of different architectural decisions, especially when two or more solutions conflict in their ability to meet the quality attributes of Non-Functional Requirements (NFR) As done in Table 5.

Table 5. Trade-off Analysis

Quality Attribute	Architecture		Trade-off Analysis
	Monolith	Microservices	
Scalability	Less scalable	More scalable	Microservices provide better scalability but at a more complex cost
Performance	Single codebase can be optimize	Potential for slower due to inter-service communication	Monolith offers faster response time, but microservices allow scaling and flexibility
Security	Easier to manage security within a single codebase	Multiple services, more points to secure and increasing complexity	Monolith may have fewer security risks due to simplicity, but microservices provide better fault isolation
Availability	Downtime affects the whole system	Fault-tolerant, as services can be restarted	Microservices enhance availability by isolating failures to individual services, while monolith has a risk of system-wide failure
Maintainability	Harder to maintain as the system grows, but faster development initially	Easier to maintain since each service is modular, but complexity when setting up	Microservices are easier to maintain due to modularity, but Monolith may be easier for small teams with more straightforward updates

5. Risk Identification
   At this stage, we identify potential risks arising from architectural decisions. Risks may be related to performance, scalability, security, availability, or other factors and may occur during system implementation or operation.

3.4. Software Architecture Comparison

The architectural choices for the application were evaluated by comparing monolithic and microservice designs, as illustrated in Figure 10.

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

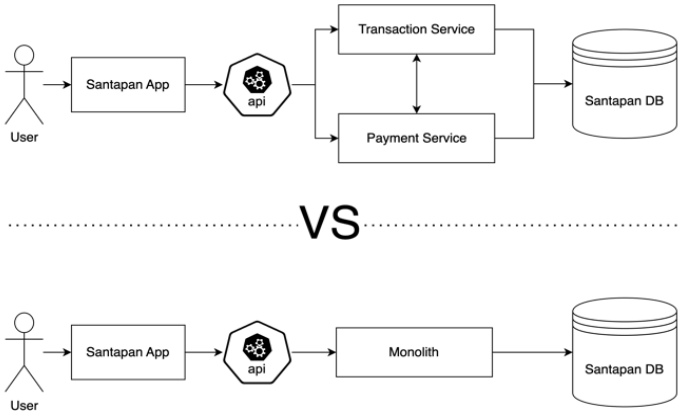

Figure 10. Monolithic vs Microserives: Architecture Design

Monolithic architecture consolidates all components into a single application [12], simplifying initial development but creating challenges in scalability and maintenance as the system grows. On the other hand, technically microservices architecture is a specialty of flexible SOA (Service Oriented Architecture) implementation approaches [24]. Microservices decompose systems into independent services, such as transaction services and payment services, thereby enabling greater scalability, modularity, and fault tolerance. For comparison, the software architecture is built using Golang and PostgreSQL to handle the core functionality. This application exposes three REST endpoints that return JSON objects, as follows:

- Monolithic Service: The service includes menu management, item bundling handling, user authentication, and other important tasks.
- Payment Service: Handles user payment transactions, ensures secure payment processing, and provides feedback regarding payment status (for example, successful, failed, or pending).
- Transaction Service: This service manages user transactions, including order history, payment logs, and transaction status. This ensures that all transaction data is stored securely and retrieved by users.

Implemented using AWS EC2, to ensure scalability and flexibility which is important to support the dynamic growth of applications [24], [25]. EC2 instances allow for horizontal and vertical scaling [27]. Depends on traffic demand and application performance requirements. EC2 makes it possible to manage cloud infrastructure costs while ensuring applications remain responsive and scalable as user traffic increases. See Table 6 for details of EC2 pricing tiers.

Table 6. AWS EC2 Instance Hourly Rate

Instance Name	vCPU	RAM (GB)	On-Demand hourly rate (USD)
t4g.nano	2	0.5 GiB	$0.0042
t4g.micro	2	1 GiB	$0.0084
t4g.small	1	2 GiB	$0.0168

3.4.1. Cost Comparison

To compare costs and monolithic and microservices, analysed the 24-hour effective cost of an EC2 instance deployed in both architectures, as shown in Table 7. The 24-hour effective cost is calculated by multiplying the instance's hourly rate by 24 hours, giving the total cost for usage one full day. In comparison, monolithic architectures and microservices use the same instance type, t4g.small, which provides one vCPU and 2 GiB RAM. Even though they use the same instance, microservices

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

require multiple instances to handle different services (such as transaction and payment services), which increase overall costs compared to a monolithic architecture.

Table 7. AWS EC2 Instance 24H Effective Cost (USD)

Services	Instance Name	RAM (GB)	24H Effective Cost (USD)
monolithic	t4g.small	2 GiB	$0.4032
Transaction (micro)	t4g.small	2 GiB	$0.4032
Payment (micro)	t4g.micro	1 GiB	$0.2016

Referring to previous research [14] , "microservice is more expensive than monolithic". Table 9 shows that although the effective 24-hour cost for both architectures similar when considering only one instance per architecture, the monolithic architecture benefits from fewer instances for its initial deployment. However, as applications scale, the additional instances required for the architecture and microservices may incur higher costs due to the need for more instances to handle increased traffic and scaling of independent services.

3.4.2. Comparison of Performance and Scalability

Apache JMeter is used as a testing tool to compare the performance of applications. JMeter is usually used for load and stress testing [22]. The open-source software tool JMeter can simulate the load on a server [11]. JMeter simulates multiple users simultaneously and measures system response under various conditions. Testing focuses on key performance indicators such as response time, throughput, and the ability to handle many requests per second. In this research, JMeter utilized several parameters, including:

1. Number of threads (users): virtual users or threads to be simulated [11].
2. Ramp-up period (seconds): the time required to increase the number of virtual users or threads [11].
3. Number of loops: the number of times the virtual user repeats the request or scenario.

Aligned with the non-functional scalability requirements in Table 6. Testing leveraged 1,000 virtual users with a ramp-up period of 500 and testing was performed on tasks that were easy to endpoint in microservices and monolithic. Table 8 shows some sample results from JMeter.

Table 8. JMeter Sample Response Time (Simple Task)

Sample #	Label	Sample Time (ms)	Sent Bytes
1	Monolithic	59	632
2	Monolithic	5	632
3	Monolithic	6	632
4	Microservice	86	660
5	Microservice	84	660
6	Microservice	28	660

JMeter with monolithic sent bytes goes to 632B and tries to request an address addition endpoint. Meanwhile, the microservices sent bytes are 660B. The time difference is insignificant, and monolithic

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

architectures even outperform microservices. It is also seen that monolithic systems can remain stable during demand spikes, as shown in Figure 11.

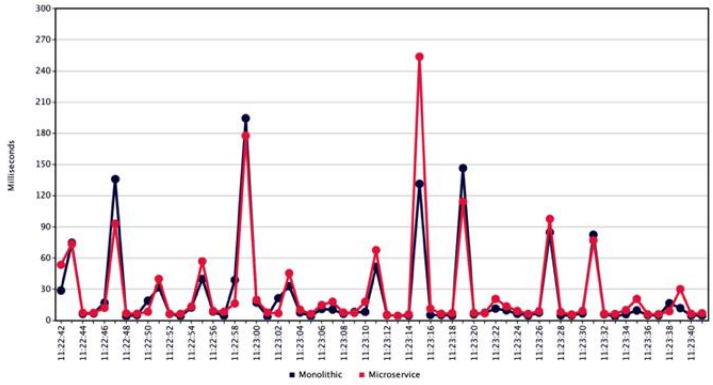

Figure 11. JMeter Graph: Response Time (Simple Task)

A comparison is made between tasks that require more time for monolithic and microservices. Monolithic was tested on JMeter by simultaneously sending 1,000 login requests, while microservices handled 1,000 transaction requests. Sample results from JMeter can be seen in Table 9.

Table 9. JMeter Sample Response time (Heavy Task)

Sample #	Label	Sample Time (ms)	Sent Bytes
1	Monolithic	579	230
2	Monolithic	1065	230
3	Microservice	679	533
4	Microservice	325	533
5	Monolithic	1712	230
6	Microservice	55	533

Monolithic is slower on heavy tasks due to the need to decrypt the salt first, as testing on monolithic systems involves user logins. On the other hand, microservices are slower than before because the transaction service must wait for the payment service before the transaction is created.

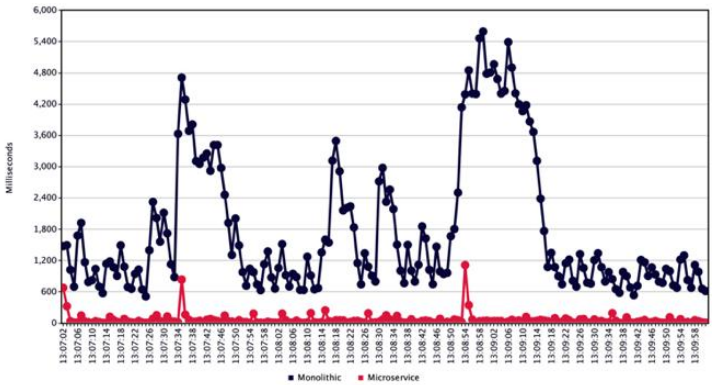

Figure 12. JMeter Graph: Response Time (Simple Task)

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

The higher response time seen in Figure 12, where the monolithic system experienced several spikes exceeding 5,000ms before finally stabilizing again, could be caused by several factors. One of them is the high query load and the decryption process of the salt used for security needs during the user login process. This emphasizes the importance of optimizing authentication mechanisms and database management in monolithic systems to minimize latency and improve overall performance.

3.5. Implications

Startup cost structure depends on the architecture chosen, which must balance scalability, performance, and operational costs. It uses AWS EC2 instances for its deployment and the cost is determined by factors such as instance type, size, RAM (memory), and number of instances for each service

- Monolithic Architecture: Monolithic still provide high performance [15],[14]. Because it meets the important quality attributes of scalability, performance, and availability with relatively low complexity in terms of implementation and monitoring.
- Microservices Architecture: Although microservices offer greater flexibility and scalability, they have higher operational costs. Each service requires its own EC2 instance, leading to more complex infrastructure management and potentially higher cloud costs [15].

Monolithic Architecture remains a cost-effective option [14], especially in the early stages of development with a corresponding cost breakdown in Table 4. The ability to handle moderate traffic without requiring extensive changes makes it a practical choice. However, as the scalability of startup improves and the introduction of more users and services, the transition to a microservices architecture can offer longer-term scalability and better fault tolerance, albeit at a higher cost.

4. DISCUSSIONS

The findings of the research conducted have an impact on architectural decisions and the cost structure of the early-phase startup development. Especially on the decision between two architectural choices monolithic and microservices with cost structure. Where this choice will affect the cost, performance, and scalability [13].

4.1. Implications of Architectural Choices on Cost Structure

The use of monolithic architecture in the early phase of a startup provides many advantages such as low initial costs. As seen in Table 9, the cost implementing monolithic architecture is significantly lower compared to microservice, making it a suitable choice for early phase startups. In accordance with previous research [11], “Test results have shown that client-operated microservices indeed reduce infrastructure costs by 13% in comparison to standard monolithic architectures and in the case of services specifically designed for optimal scaling in the provider-operated cloud environment, infrastructure costs were reduced by 77%”. However, research [11] also concluded “a microservice architecture is not the best suited for every context. A monolithic architecture seems to be a better choice for simple, small-sized systems that do not have to support a large number of concurrent users“.

This research confirms that monolithic architecture has cheaper costs for startups that still have a small number of users [11], [12], [14]. the cost of the AWS instance required is less compared to the microservices architecture [10], and can still meet the quality attributes [19], [20]. And criteria in Figure 9, performance and scalability that can still be done with monolithic make it chosen [10], [12] as a software architecture. This is very important in the early phase of a startup, where budgets are often limited. Monolithic architecture not only reduces cloud infrastructure costs in the cost structure in Table 4. Like AWS instance costs, it also simplifies development and operational processes according to the trade-offs in Table 7. Although less modular than microservice [10], [12], [13] monolithic architecture

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

can still meet quality requirements. Attribute that corresponds to the ATAM process [19], [21]. This quality attribute is very important to fulfill [21], especially in the early stages of a startup which requires a fast development process and cost efficiency [6].

4.2. Scalability and Performance

Comparison between scalability and performance in software architecture was carried out using the JMeter tool [11]. JMeter is used to simulate various workload scenarios by measuring response time, throughput, and system capabilities [11]. This analysis provides a clear picture of how each architecture type responds to scalability and performance needs. Table 10 shows that monolithic performance provide good performance, this architecture shows its limitations. Dependencies between components make it difficult to adjust modularity [10], [14]. On the other hand, microservice architecture offer a superior solution by dividing the application into small, independent services, each service can be upgraded separately as needed. Previous research [11], [12] concluded that microservice architecture is sometimes not suitable for some contexts. Monolithic architecture is a good choice for simple, small-sized systems that cannot handle a lot of users. However, based on this research early phase startup prefer to use a monolithic architecture because of several previous considerations on the quality attributes in Figure 9. In accordance with the Architecture Tradeoff Analysis Method (ATAM) process, which includes the required performance and scalability parameters desired by stakeholders in Table 5.

4.3. Recommendation

Based on the results of the analysis that has been carried out, here are several recommendations regarding the choice of software architecture for startups in the early phase:

1. Startups with limited resources should choose a monolithic architecture in the early stages. Low initial costs, easy management, and the ability to meet basic quality requirements such as performance and scalability make monolithic the optimal choice.
2. Although monolithic architecture is suitable for the early phase, business and technology needs can change over time. It is recommended to conduct periodic evaluations using processes such as the Architecture Trade-off Analysis Method (ATAM) to determine whether a transition to a microservice architecture is necessary.

5. CONCLUSION

Based on the research conducted, it can be concluded that monolithic is more suitable for early-stage startups. This is due to lower implementation costs, simplicity in management and being able to meet the basic quality attributes needed such as performance and scalability. Although microservice architecture has the advantage of flexibility and scalability in the long term, microservices require a larger initial investment and high operational complexity, making it less ideal for startups that are still in the early stages of development and have limited budgets. The study has limitations that need to be considered:

1. The focus of the study is on aspects of cost, performance, and scalability without discussing other factors such as security and modularity.
2. Performance testing data uses simulation with JMeter in certain scenarios. The results do not fully reflect real conditions in the production environment.
3. The study is limited to the early stages of technology-based startups and is less relevant to other industries or startups that are already in the advanced development stage.

In future work, can include testing on systems with larger scales and loads by producing representative ones. In addition, in-depth analysis of other quality attributes such as security, efficiency, and team development. In addition, further research can try evaluations other than using methods other

than ATAM. We hope this research provides insights for the technology-based startup industry, especially for the early stages of development. Some lessons that can be learned by choosing monolithic architecture:

1. Save infrastructure costs and early development, allowing more resources to focus on product development
2. Reduce operational complexity in the early stages, so that the team can focus more on product development.

However, it is important for startups to realize that monolithic architecture has limitations that can affect scalability in the future. Therefore, architectural decisions must be adjusted to the needs required.

CONFLICT OF INTEREST

The authors declares that there is no conflict of interest between the authors or with research object in this paper.

REFERENCES

[1] N. Razabillah, S. R. Putri Junaedi, O. P. Maria Daeli, and N. S. Arasid, "Lean Canvas and the Business Model Canvas Model in Startup Piecework," *Startupreneur Business Digital (SABDA Journal)*, vol. 2, no. 1, pp. 72–85, Feb. 2023, doi: 10.33050/sabda.v2i1.239.

[2] T. Felin, A. Gambardella, S. Stern, and T. Zenger, "Lean startup and the business model: Experimentation revisited," *Long Range Plann*, vol. 53, no. 4, p. 101889, Aug. 2020, doi: 10.1016/j.lrp.2019.06.002.

[3] A. Maurya, *Running lean*, 3rd Edition. " O'Reilly Media, Inc.," 2022.

[4] M. Pellegrini, "The Business Canvas*," in *The 39th ACM International Conference on Design of Communication*, New York, NY, USA: ACM, Oct. 2021, pp. 224–230. doi: 10.1145/3472714.3473645.

[5] F.-L. Noelia and D.-C. Rosalia, "A dynamic analysis of the role of entrepreneurial ecosystems in reducing innovation obstacles for startups," *Journal of Business Venturing Insights*, vol. 14, p. e00192, Nov. 2020, doi: 10.1016/j.jbvi.2020.e00192.

[6] R. G. Chammassian and V. Sabatier, "The role of costs in business model design for early-stage technology startups," *Technol Forecast Soc Change*, vol. 157, p. 120090, Aug. 2020, doi: 10.1016/j.techfore.2020.120090.

[7] Eashwar Sivakumar and Paras Chawla, "Decentralized Lean Business Model Canvas for BlockchainBased Enterprises ," *Journal of Computer Sciences*, vol. 18, pp. 426–440, 2022, doi: 10.3844/jcssp.2022.426.440.

[8] Rachmad and Yoesoep Edhie, "The Influence And Impact of The Money Burning Strategy on The Future of Startups," *Adpebi Science Series*, Jul. 2022.

[9] M. Unterkalmsteiner *et al.*, "Software Startups -- A Research Agenda," *e-Informatica Software Engineering Journal*, vol. 3, no. 1, Aug. 2023, doi: 10.5277/e-Inf160105.

[10] N. C. Mendonca, C. Box, C. Manolache, and L. Ryan, "The Monolith Strikes Back: Why Istio Migrated From Microservices to a Monolithic Architecture," *IEEE Softw*, vol. 38, no. 5, pp. 17–22, Sep. 2021, doi: 10.1109/MS.2021.3080335.

[11] G. Blinowski, A. Ojdowska, and A. Przybyłek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access*, vol. 10, pp. 20357–20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[12] K. Gos and W. Zabierowski, "The Comparison of Microservice and Monolithic Architecture," in *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, IEEE, Apr. 2020, pp. 150–153. doi: 10.1109/MEMSTECH49584.2020.9109514.

[13] R. Su, X. Li, and D. Taibi, "From Microservice to Monolith: A Multivocal Literature Review," *Electronics (Basel)*, vol. 13, no. 8, p. 1452, Apr. 2024, doi: 10.3390/electronics13081452.

Vol. 6, No. 6, December 2025, Page. 5553-5569
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.4302

[14] T. Selivorstova, S. Klishch, S. Kyrychenko, A. Guda, and K. Ostrovskaya, "Analysis of Monolithic and Microservice Architectures Features and Metrics," *Computer systems and information technologies*, no. 3, pp. 59–65, Apr. 2022, doi: 10.31891/CSIT-2021-5-8.
[15] R. Su and X. Li, "Modular Monolith: Is This the Trend in Software Architecture?," in *Proceedings of the 1st International Workshop on New Trends in Software Architecture*, New York, NY, USA: ACM, Apr. 2024, pp. 10–13. doi: 10.1145/3643657.3643911.
[16] M. Sahlabadi, R. C. Muniyandi, Z. Shukur, and F. Qamar, "Lightweight Software Architecture Evaluation for Industry: A Comprehensive Review," *Sensors*, vol. 22, no. 3, p. 1252, Feb. 2022, doi: 10.3390/s22031252.
[17] D. A. S. G. Putra Kusuma, "Designing and Evaluating Representational State Transfer Architecture for School Management Information System," *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 7, pp. 3649–3658, Jul. 2020, doi: 10.30534/ijeter/2020/124872020.
[18] D. Sobhy, R. Bahsoon, L. Minku, and R. Kazman, "Evaluation of Software Architectures under Uncertainty," *ACM Transactions on Software Engineering and Methodology*, vol. 30, no. 4, pp. 1–50, Oct. 2021, doi: 10.1145/3464305.
[19] F. Moshiri, A. Asosheh, and F. Hashembigi, "Introduce an Enhanced Hospital Information System Reference Architecture with ATAM Evaluation," 2024.
[20] S. M. Ågren *et al.*, "Architecture evaluation in continuous development," *Journal of Systems and Software*, vol. 184, p. 111111, Feb. 2022, doi: 10.1016/j.jss.2021.111111.
[21] A. El Murabet and A. Abtoy, "Methodologies of the Validation of Software Architectures," *Journal of Computing Theories and Applications*, vol. 1, no. 2, pp. 78–85, Nov. 2023, doi: 10.33633/jcta.v1i2.9332.
[22] D. de Silva *et al.*, "Evaluating the Effectiveness of Different Software Testing Frameworks on Software Quality," May 19, 2023. doi: 10.21203/rs.3.rs-2928368/v1.
[23] Osinachi Deborah Segun-Falade, Olajide Soji Osundare, Wagobera Edgar Kedi, Patrick Azuka Okeleke, Tochukwu Ignatius Ijomah, and Oluwatosin Yetunde Abdul-Azeez, "Assessing the transformative impact of cloud computing on software deployment and management," *Computer Science & IT Research Journal*, vol. 5, no. 8, pp. 2062–2082, Aug. 2024, doi: 10.51594/csitrj.v5i8.1492.
[24] N. Mateus-Coelho, M. Cruz-Cunha, and L. G. Ferreira, "Security in Microservices Architectures," *Procedia Comput Sci*, vol. 181, pp. 1225–1236, 2021, doi: 10.1016/j.procs.2021.01.320.
[25] A. Choudhary, "A walkthrough of Amazon Elastic Compute Cloud (Amazon EC2): A Review," *Int J Res Appl Sci Eng Technol*, vol. 9, no. 11, pp. 93–97, Nov. 2021, doi: 10.22214/ijraset.2021.38764.
[26] H. Sihotang, "Quantitative research methods" 2023, *UKI Press*.
[27] M. Catillo, U. Villano, and M. Rak, "A survey on auto-scaling: how to exploit cloud elasticity," *International Journal of Grid and Utility Computing*, vol. 14, no. 1, p. 37, 2023, doi: 10.1504/IJGUC.2023.129702.