Journal of
Information Systems Engineering
and Business Intelligence

Vol.11, No.1, February 2025
Available online at: http://e-journal.unair.ac.id/index.php/JISEBI

PlatFab: A Platform Engineering Approach to Improve Developer Productivity

Vaishnavi Srinivasan[1)*], Manimegalai Rajkumar [2)], Srivatsan Santhanam [3)], Arjit Garg [4)]

[1) 2)] *PSG Institute of Technology and Applied Research, Coimbatore, India*

[1)]vaishnavi1731@gmail.com, [2)]drrm@psgitech.ac.in

[3) 4)] *SAP Labs India, Bengaluru, India*

[3)]srivatsan.santhanam@sap.com, [4)] arjit.garg@sap.com

Abstract

Background: Software developers are key players in IT/ITES business in order to drive software development by writing high-quality code quickly. Based on user needs, they must adapt evolving technologies and tools to produce efficient and successful software using Software Development Life Cycle (SDLC) principles. Platform Engineering comprises a set of activities to design, develop and maintain software code, making it a foundation for building software applications.
Objective: This work focuses on reducing the time and effort needed to execute the above tasks that boosts software developer productivity which includes software development workflow automation. The main objective of the proposed work is to lower total cost of ownership, standardize software development practices, help cost optimization and reduce production incidents.
Methods: PlatFab, a Platform Engineering service implemented in Industrial Budgeting System is presented in this work. The methodology involves custom developer portal with Continuous Integration and Continuous Delivery/Continuous Deployment. (CI/CD) pipeline to automate financial workflows and streamline collaborative development. It provides the developers architectural components, containers, infrastructure automation and services orchestration that helps them to concentrate on their quality code irrespective of implementation efforts.
Results: After deploying PlatFab in an organization's software development, build time is reduced by one minute for each service, and 60MB of storage space is saved for each service. Developers can handle vulnerability attacks in one day. Before the use of PlatFab, build time was five minutes, 2 GB was used for each service, and vulnerability handling required five days to resolve. Production downtime issues were 12 before PlatFab and were reduced to almost zero after integrating PlatFab.
Conclusion: The results after implementing PlatFab for a Budgeting System service in an IT Organization help the developers reduce build time, number of days to fix vulnerabilities, and space requirements for the service. PlatFab helps the developers complete their projects with quality code in a shorter time, improving their productivity.

Keywords: Agile Methodology, Budgeting Service, Platform Engineering, Software Development Life Cycle, Service Oriented Architecture.

Article history: Received 28 June 2024, first decision 13 September 2024, accepted 17 March 2025, available online 28 March 2025

I. INTRODUCTION

Software, the word most heard in the recent century, made notable advancements in almost all industries, education sectors, and public sectors, acts as the single reason for the emergence of digital platforms. The role of software is significant to position and advance the technological world to the next level. Solving complex problems, by performing activities such as requirements gathering, designing, creating, implementing and maintaining it, is the key must be carried out to generate the final product with high-quality and as per customers' expectations [1]. SDLC includes both conventional models such as waterfall, iterative, spiral, V shaped, and advanced models such as Agile, process of software development [2]. Software Development Life Cycle (SDLC) covers systematic approaches that that are used for the development activities using a layered or tiered architecture to build the end product [3]. Agile development is a dynamic and incremental software development methodology that places an emphasis on customer input, teamwork, and the delivery of incremental, tiny releases. Agile approaches place a strong emphasis on flexibility, change adaptation, and ongoing development [4]. Software platforms help in efficient development and deployment of software applications. Software platforms furnish software developers with many resources, including frameworks, libraries, and services, which aid them throughout the software development lifecycle. They are of three types, namely, (1) Development Platform, which provides development environments including IDEs, programming

* Corresponding author

ISSN 2443-2555 (online) 2598-6333 (print)
doi: http://dx.doi.org/10.20473/jisebi.11.1.79-90

languages, (2) Platform Engineering, which involves building, designing and maintaining underlying infrastructure and software applications; and (3) Runtime Platforms that include tools for mobile and web applications, analytics process. Applications can be developed and deployed more effectively when software platforms are in place. Software platforms shorten the time and effort needed to develop an application by offering a pre-built foundation. Platforms make it possible to reuse common parts and services, boosting output and lowering costs. Software platforms make it easier for disparate applications to communicate with one another and integrate and exchange data. Service Oriented Architecture (SOA) plays a significant role in making the activities reusable in software platforms. It provides software as services, which are modular components and implement specific business operations. Services are made available to applications in the SOA architecture via an internet-based network call. Every service is a stand-alone full-featured business function which reduces cost of ownership, makes components distributed and heterogeneous.

Services' autonomous and reusable nature encourages flexibility and scalability [5]. SOA in Distributed energy resources provide easy integration of services [6]. Banking applications implement SOA in the payment module of PDAM and optimize the payment procedure to deliver superior services to consumers [7]. SOA streamlines the administration of resources and service delivery, making feature additions and changes easier without affecting the whole architecture. Tests reveal increased application performance, responsiveness, and multi-source service integration and speeds up mobile app development and reduces change effects [8]. With many advantages such as reusability, modularity, flexibility and easy management of services, SOA faces limitations such as implementation cost, interoperability between services, performance overhead due to many layers, high level of complexity when number of services increases and security measures at each layer. The impact of SOA in increasing productivity and efficiency does not rise significantly. Microservices are modern SOA which builds services in distributed environments. These services, orchestration and containerization, became a novel process in software development [9]. While using individual reusable services might be advantageous in software development, it is important to acknowledge the existence of downsides. The obstacles encountered in microservices include latency concerns, network bottlenecks, managing several services in large-scale systems, and service attacks. When undertaking large-scale software developments, it is crucial to consider scalability and the integration of legacy systems. This indicates the necessity for a systematic approach to enable the reuse of services inside the development environment.

PlatFab is a Platform Engineering framework that addresses the specific industrial workflows. Platform Engineering comprises a range of activities to design, build and maintain the infrastructure and services of an organization. They are used to develop scalable, reusable platforms for efficient development of products. The impact of platform engineering is brought about by the development of comprehensive platforms that manage the entire lifecycle and integration of many services, rather than only focusing on individual services. This approach commonly makes use of microservices, containers, and orchestration tools [9]. It offers a flexible ecosystem for an organization to implement collaboration and teamwork. Investing in strong platform engineering processes is recommended by the research for manufacturing organizations. According to the study's findings, platform engineering is also guaranteeing security, adaptability, and efficiency [10]. Platform engineering leverages the ideas of Service-Oriented Architecture (SOA) but operates at a more advanced level of abstraction to achieve smooth integration, scalability, and automation, especially in intricate industrial settings. PlatFab integrates all these activities into a unified ecosystem for seamless integration and enhancement in productivity. In Platform Engineering, all the key activities are separated as layers, in which each layer is coupled with the same functionalities. Layers, Abstraction, Services, available open standards, and APIs are the main components in development platforms. The role of platform engineering in a software development environment is shown in Figure 1. It includes application development and business process, service development providing open standards and APIs, and platform engineering responsible for managing infrastructure components.

Layers and Abstraction: Figure 1 illustrates how platform components are arranged hierarchically, each building on the layer below it. Developers can conceal the internal workings of platform components through abstraction to work at higher levels of abstraction. Layering the architecture using abstraction shows the necessary functionalities to the developers. The development team focuses on the key processes to be carried out to accomplish the solution for the business problem provided. Platform engineers create all the key software components required for the developers to develop software. They connect with the help of APIs provided by the service team [11].

Service Oriented Architecture (SOA): Loosely coupled and reusable services are the foundation of software platforms, and SOA is an architectural style that encourages platform development. Services in Service-Oriented Architecture (SOA) are intended to be autonomous and loosely coupled, facilitating seamless integration [12]. Also, SOA makes it possible to create flexible and adaptable software platforms by allowing services to be designed as reusable units.

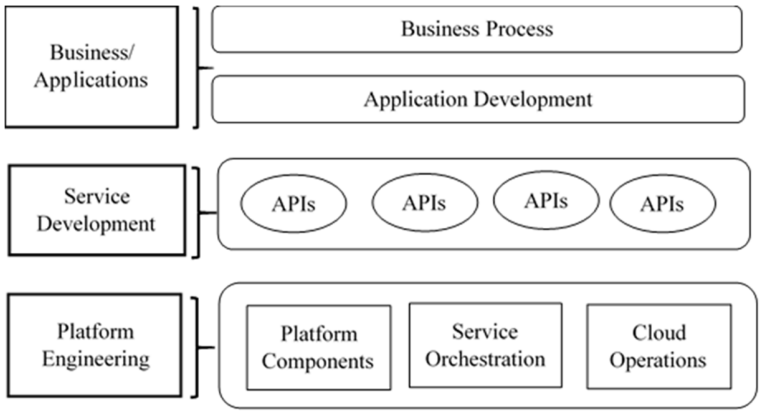

Fig. 1 Role of Platform Engineering in Software Development

Open Standards and APIs: Open-source tools make up the platform that platform engineers typically build. Platform engineers in an organization customize the infrastructure as code tools to meet the requirements of application developers. Open standards and APIs are frequently used by software platforms to guarantee compatibility and interoperability with other systems. Publicly available open standards make it possible to create interoperable software components and systems. The interfaces that allow various software components to connect and communicate with one another are defined by application programming interfaces, or APIs. Some available open standards and APIs include Docker, Kubernetes, Github etc.

Developer productivity and experience, by offering automated infrastructure operations together with self-service features are the main objectives of Platform Engineering. It promises to improve the experience of developers and accelerate the delivery of value to customers by product teams, making it popular. This aims to simplify and eliminate uncertainty in the supply of cutting-edge software. It tackles some of the most significant difficulties in implementing Development Operations at large scale, such as coordinating development procedures with business objectives and reducing the workload associated with overseeing a complex set of tools and infrastructure throughout the development lifecycle of a software application. Although prior research has examined the advantages of platform engineering in a broad sense, there is a clear lack of information regarding its implementation in highly specialized industrial sectors. These sectors face considerable obstacles due to legacy systems and various operating needs. Despite the growth of Platform Engineering across various domains, its application in areas such as industrial budgeting remains unexplored. This discrepancy highlights the necessity for additional investigation into specialized platform engineering approaches designed specifically for these situations. Existing frameworks frequently lack the necessary adaptability and scalability to manage the intricate processes and regulatory demands of financial operations. This study fills this gap by presenting PlatFab, an innovative platform engineering framework aimed at enhancing developer productivity and refining budgeting procedures. PlatFab is a custom designed platform engineering framework developed specifically to enhance developer productivity in the context of Industrial Budgeting Services. This integrates domain specific automation tools and pipelines for centralized workflow management to address inefficiencies unique to budgeting services. Its novelty lies in its ability to streamline collaboration among cross functional teams while reducing deployment time and reducing down time.

The proposed methodology contributes the following (1) PlatFab, a domain specific platform engineering framework deigned for industrial sectors, (2) Demonstrating significant productivity improvements through real world implementation, (3) establishing a scalable methodology that can be extended to other regulated and domain specific industrial operations. This research contributes to the field of industrial sectors by introducing a novel platform engineering framework designed for their specific industrial challenges. Comparing other studies, this approach emphasizes practical implementation and evaluation through developer centric metrics such as build time, vulnerability fix days, production downtime and resource utilization.

II. Literature Review

Every developer relies on the common set of activities such as design, build, code, deploy and test for software development. Looping through the common activities for every software development decreases developer productivity. Several organizations have established platform-based approaches among teams to handle the activities that are repeatable. Platform Engineering is intertwined with SOA in focus of reusability, modularity of software components with increased developer productivity. The report suggests prioritizing services and assessing successful

and unsuccessful implementations using platforms to boost software development team productivity, eliminate duplication, and boost job satisfaction [13]. The Digital Platform operation model has disrupted many sectors and services, improving efficiency, information sharing, and business processes. The proposed design allows conceptual layer classification based on functionality and domain knowledge [14]. Digital Transformation aims to close the gap between operations and engineering [15]. DevOps provides continuous integration of services and deployment pipelines which offers quick and reliably proactive quality assurance by automating testing and monitoring early in the development process [16]. Platform engineering provides all the components for the management of infrastructure and services. Concept Realm introduces methods to balance the distribution of problem domain knowledge among developers that decreases developer turnover increasing productivity [17]. Software professionals employ methodologies such as DevOps, Kubernetes, Scrum, and Waterfall models to develop software of superior quality. Question-and-answer sites are helpful for practitioners to locate useful resources to surmount obstacles, to remain up-to-date on current trends, and to enhance the standard of the software they build. The study describes how few services aid in Continuous Integration, Continuous Delivery, and Automation of a retail-clothe-shop operations using Azure Devops [18]. The development of a medical imaging platform provides reusable components in a shared architecture to capture this shared functionality in the medical domain [19]. The simplified Software delivery process in AWS through continuous integration with low repeatability is presented in [20]. Importance of automating build, test and deploy operations is discussed to provide efficiency in operations and performance. To increase operational efficiency while decreasing time to market, organizations seek ways to integrate information, services, infrastructure, and potential sharing into their environment [21]. The proposed architecture promotes robust system design, scalability, and economic viability, enabling sustainable expansion in competitive environments in organizations which streamlined continuous development and operational operations, boosting productivity and lowering overhead expenses [22]. All of the aforementioned objectives can be greatly advanced with the help of Platform Engineering.

As per Gartner podcasts on steering engineering, by 2026, at least 80% of organizations will adopt platform engineering in their software development process [23]. It is mentioned that platform engineering will be a game changer in the tech world since it addresses the inefficiencies in software development. Platform engineering aims to modernize enterprise software delivery, especially for digital transformation. The engineering platform is developed and maintained by a specialized product team to support the needs of software developers and others by offering common, reusable tools and capabilities to integrate with complex infrastructure. An engineering platform's specific capabilities are determined solely by the demands of its end users. Platform teams have to prioritize tasks, comprehend the needs of their user groups, and create a platform that serves the intended user base. The initial step is to develop an Internal Developer Platform (IDPs) which contains a collection of tools, resources, infrastructure services to be used by organization developers.

Platform Engineering offers advantages such as increased developer productivity, standardization of tools and process, security advancements, pipeline management of series of integration activities, and Infrastructure-as-code concept. It enables self-service to developers so that they can be agile and competitive. Developers can concentrate on their development without engaging in infrastructure and resource allotments [24]. Professionals in software platform engineering concentrate on creating strong, adaptable platforms that can handle various applications. Implementing essential features and functionalities that satisfy the demands of various application domains is a crucial part of developing software platforms. To guarantee platforms' dependability and availability, software platform engineering also entails continuous platform management and maintenance.

Previous studies in platform engineering primarily focus on generic software development environments or domains like e-commerce and DevOps. These studies often lack domain specific customization to address challenges in specialized industries such as financial workflows. PlatFab bridges this gap by introducing a tailored framework designed for highly regulated and dynamic requirements of industrial budgeting. It incorporates financial compliance checks and integrates budgeting specific automation workflows. PlatFab emphasizes cost optimization, standardization, and incident reduction in a domain-specific context. This study develops and validates a framework to improve industrial software platform efficiency, automation, interoperability, and scalability in complex industrial environments, explore the synergy between platform engineering and SOA, and support digital transformations with knowledge sharing.

III. METHODS

Platform teams at every organization perform design, development and maintenance of platform engineering frameworks. This team reduces the complexity in handling operations and development work of an organization. It includes infrastructure handling, orchestration of services, pipelining activities and security aspects. Figure 2 provides a typical architecture of platform engineering implementation at industries. Microservices build applications as small,

loosely coupled and independent entities. The Platform interacts with multiple microservices, each designed for a specific purpose. PlatFab is an engineering framework which is integrated with all the microservices of an organization. Service discovery, load balancing, monitoring, and scaling are available for microservice development, deployment, and management on this platform. The components of PlatFab make it easier for various systems, services, and APIs to connect and work together within a development environment. Standardized Helm chart templates, Renovate Bots, SDP Base image for RPL Scripts, A/B deployments are some of the essential features for PlatFab.

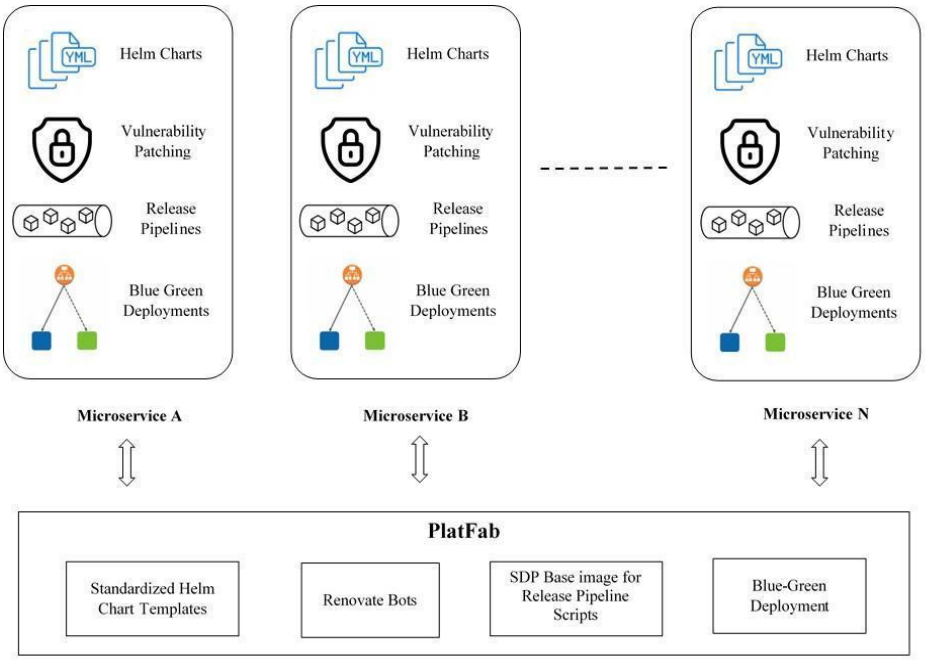

Fig. 2 Typical Architecture of Platform Engineering in an IT/ITES Industry

A. Standardized Helm chart templates

Helm charts are packages of pre-configured Kubernetes (K8s) resources needed to deploy microservices [25]. They are defined as YAML Ain't Markup Language (YAML) files with a directory structure. Helm charts standardize application setups and dependencies, encouraging code reuse and sharing. Developers can use Helm repositories to exchange and contribute charts. Helm charts streamline Kubernetes application deployment and management and simplify Kubernetes cluster deployment for complicated applications, letting developers focus on application development rather than infrastructure management [26]. Helm Charts provide common templates for all microservices. It provides necessary configuration files that are usable by all services that are pre-defined, reusable templates for Kubernetes application package managers. These templates follow best practices and conventions to standardize Kubernetes application deployment. This makes it easier and convenient for the developers to deploy services in their production environments. This helps developers to focus on value added features instead of usual configurations.

B. Renovate Bots

Fixing vulnerabilities in the software and infrastructure is a major concern as new risks are produced daily. Vulnerabilities or attacks arise at software and infrastructure components of an organization [27]. This patching acts as a vulnerability scanner to find microservice security flaws. It comprises security audits, manual code inspections, automated scanning tools, and penetration tests. Vulnerabilities of higher risk are handled by prioritizing them. After identifying a vulnerability, a patch can be applied to keep the system safe. Patching helps the system to be away from the risks. To guarantee that vulnerable systems and applications are patched promptly and effectively, organizations set up patch management methods and procedures. This is collaborated with Common Vulnerabilities and Exposures (CVE) to identify the cyber vulnerabilities prominent in industries. The presented work in [28] developed a tool to identify the vulnerabilities with the CVE platforms. This helps the organization to handle and manage vulnerabilities

efficiently. The study reviews methodical framework, hands-on methods, and list of potential issues to provide source code patch commit mining for a stronger software ecosystem [29]. Vulnerability patching is performed at every microservice to keep it secure from cyber security threats. Renovate Bots provide automated mitigation of threats for microservices. The services do not need to develop their own vulnerability patching. On adopting PlatFab, it provides patching of security threats or vulnerabilities when needed. Also, it performs patch up dependencies to handle the risks. Performing the patching at Dev Fabric will handle all the services of an organization, and thereby increase developer's productivity.

C. SDP Base Image for Release Pipeline Scripts

Release Pipeline scripts automate the deployment process of a service to the production environment. These scripts build the service, test it, package artifacts, and deploy to production servers. It acts as an integration pipeline between the service and product environment. RPL scripts can retrieve the latest service codebase and deployment dependencies from repositories or version control systems. It reduces manual intervention and ensures consistency and reliability in service deployment to product. It standardizes deployment management and allows enterprises to deliver changes to production quickly and confidently. A proposed pipeline strategy improves delivery timelines, test load steps, and benchmarking [30]. Integrating several test processes reduces system disruption and improves process stability and delivery, reduces ambiguity and guessing, ensures quality, and boosts efficiency. The platform team creates the Release Pipeline scripts base images with Secure Development Platform to provide deployment scripts and pipelines. It gives a runtime environment and dependencies for executing Release Pipeline scripts. The developers can access this base image when there is a need in their development.

D. Blue-Green Deployment

Updates may fail when they are deployed to the production environment. When an issue arises, the Blue-Green deployment service assists in rolling back to earlier versions. It aids the stability of microservices. Blue-Green deployments are employed to find problems or irregularities in new releases before they affect the system as a whole. The new version is progressively rolled out to more users or servers if it fits the predetermined criteria and performs well in the initial subset of users. Up until the new version is implemented throughout the infrastructure, this process is carried out gradually. This helps to monitor the deployed versions and reduce the danger of difficulties by restricting their impact radius. By progressively implementing changes and monitoring their impact, firms can identify and resolve issues before they cause widespread service outages or user outrage. It streamlines and automates production environment testing and validation, enabling enterprises to release products confidently, eliminate risks, and optimize user experience through constant evaluation and iteration. The study describes AWS CodeDeploy which gradually shifts from old service to new service [31][32].

E. Architecture of Industrial Budgeting System

PlatFab is implemented in one of the production services named Budgeting system. The typical architecture designed to streamline budgeting activities is illustrated in Figure 3. It demonstrates the approval of budget for the users and monitors the expenses of the organization. The framework consists of three key layers: user interaction, service orchestration and external system integration.

1) User Interaction

The interaction with the user is done through a Web UI. The Profile Service with Profile API validates user credentials with the HostDB i.e., Host database. HostDB acts as a central repository of all user, host and configuration data. Profile service is responsible for managing user and host data. It includes storing, retrieving and updating user profile and host specific information. This component ensures secure access and communication with the database enabling streamlined authentication and authorization.

2) Service orchestration

MidTier acts as intermediary layer handling request from the user and profile service and routing them to appropriate budget services. The budget service layer comprises Microservices such as Obligation Search and Handler. They manage core budgeting workflows including obligation tracking and search operations. When a new expense is submitted, obligation handler updates the corresponding record and synchronizes with external system. Obligation Search service allows user to search for obligations, retrieves appropriate data from the HostDB and sends back to MidTier. This layer implements load balancing for incoming user requests and can be scaled based on workload enabling scalability under high demand. New microservices can be added without disturbing existing workflows.

3) Financial Integration with External System

This facilitates the integration of financial service with the external systems ensuring seamless data exchange for expense tracking and financial reporting. External systems are third party systems that integrate with the platform for tasks such as financial reporting and expense tracking.

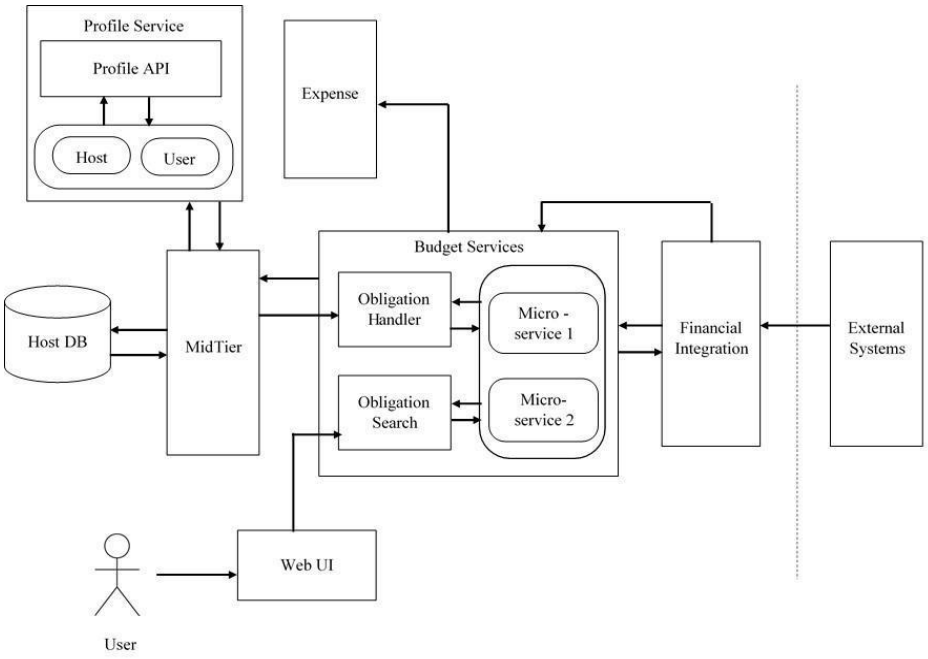

Fig. 3 Architecture of Industrial Budgeting System using PlatFab

The workflow of the Industrial Budgeting System using PlatFab is the user interacts with the system through UI and submits his request form to the finance department in the MidTier. The MidTier routes the request to the appropriate Budget Service. Obligation Search module of Budget Services gets the user request and searches for the profile with user details in the Database. Once a match is received, the user's request for the obligation handler is placed to microservice, then the financial integration module updates the obligation in the external system. User's expense report is also integrated with the budget services to monitor the details of expenses made by the user. The architecture of Industrial Budgeting System is consistent before and after the use of PlatFab since it is integrated into each service as an integrated enhancement rather than changing the overall architecture. It provides workflow optimization and resource utilization. This ensure that the core architecture can remain consistent providing PlatFab's capability to improve overall system performance. Also, using that in domain specific workflows.

F. Customer Interaction - Monitoring and Feedback Loop

The proposed work also provides Customer Interaction - Monitoring and Feedback Loop which initiates the integration of user feedback into platform development. The typical architecture of the Monitoring and Feedback loop of customer interaction in Budgeting service is shown in Figure 6. The customer initiates the process by posting a request for a good or service. The Front End, a representation of the UI, or User Interface, that the client interacts with, is used for the initial interaction with the customer. The API defines the approaches and information formats for interaction between the UI and the backend systems. Using a service messaging system—a queue of messages or a service bus—the API queries the backend services. This enables an asynchronous mode of communication between various services. Microservices, tiny, autonomous services that carry out business tasks, comprise the backend. Microservices maintain and access data with a database, like an RDS. System's performance is monitored with the help of tools such as Kibana, Elastic Stack, Container Monitoring and Infra and DB Metrics. Kibana, a part of Elastic Stack, monitors logs, metrics and other data produced by front end and service messaging. Microservices, when implemented as containers, use container monitoring to monitor the health of containers. Metrics from the Database and Infrastructure are gathered by Infra and DB metrics components to track the overall state and performance of both. Developers use the Kibana Elastic Stack and Container Monitoring component to understand the system's performance, how the application is operating and pinpoint any problems that need to be fixed. Product managers can use the monitoring data to gain insights that help them decide how to improve features, allocate resources, and develop their overall product strategy. The application provides users with immediate feedback through performance metrics or customized interactions based on the data collected during monitoring. These activities act as a Feedback Loop for the customer interaction in this microservice architecture.

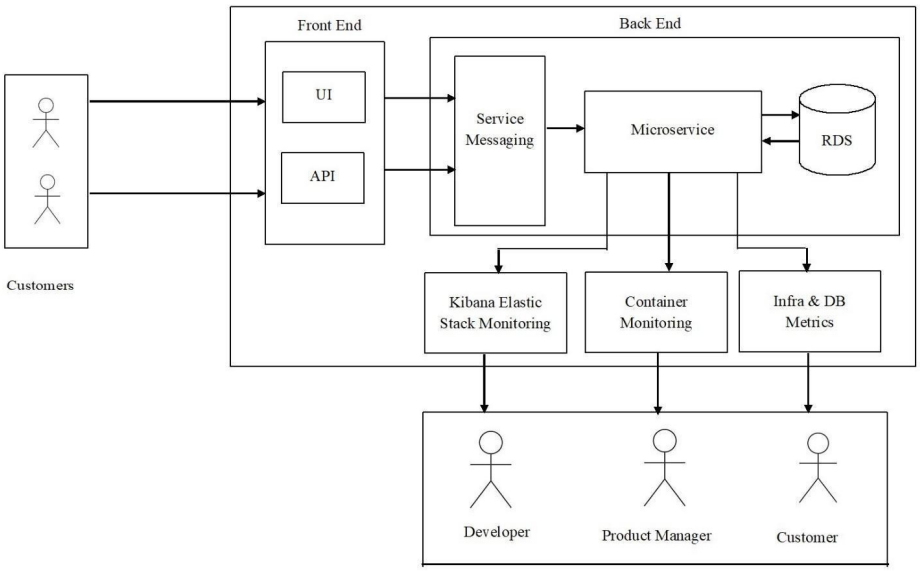

Fig. 4 Flow Diagram of Customer Interaction - Monitoring and Feedback Loop

The novelty of the work is, the architecture is built around modular microservices that can be scaled or replaced independently. Also, it supports growing workloads through load balancing and distributed service design. It integrates platform engineering with industrial-specific automation and orchestration methods, making it new. This study proposes a framework for industrial environments that addresses real-time processing, legacy system integration, and scalability across diverse operational settings, unlike previous studies that focused on generic software platforms. It also facilitates seamless integration with external systems. Integration of customer feedback bridges the gap between developers and users. It ensures that platform is user centric satisfying customer expectations and proactive changes with real time feedback. Case studies in industrial sectors verify the framework's practicality and benefits.

IV. RESULTS

PlatFab incorporates tools and deployment strategies to enhance developer productivity, optimize resource utilization and minimize production downtime. Key tools and techniques include Helm Charts, Renovate Bots, SDP base images and blue-green deployment discussed in section III are used to evaluate the proposed work. These components are selected to address performance metrics such as resource utilization, vulnerability response time, build time and production downtime reduction. The increase in developer productivity was measured by monitoring the space occupied, build time taken, the number of vulnerabilities fixed, and downtime issues. The evaluation dataset consists of real-world data from an industrial budgeting service of an IT industry. It encompasses data of five budgeting service projects with other projects over a 12-month period. Each project data included task allocation, deployment logs, and incident reports. This dataset served as a baseline for assessing the effectiveness of PlatFab framework.

A. Resource Utilization:

Utilization of resources assesses the CPU, memory, storage, and network bandwidth efficiency during deployments. Allocating resources efficiently optimizes performance and cost. 1244 services are running in the organization unit using the Helm chart. Each service has space that occupies close to 2GB. After implementing PlatFab, 60MB of space is saved for each service as helm charts are moved to a common place. Also, the build time is reduced by one minute for each service. For 1244 services around 60 MB of space is saved which can be effectively used for other tasks. The data related to the space occupied is illustrated in Figure 5.

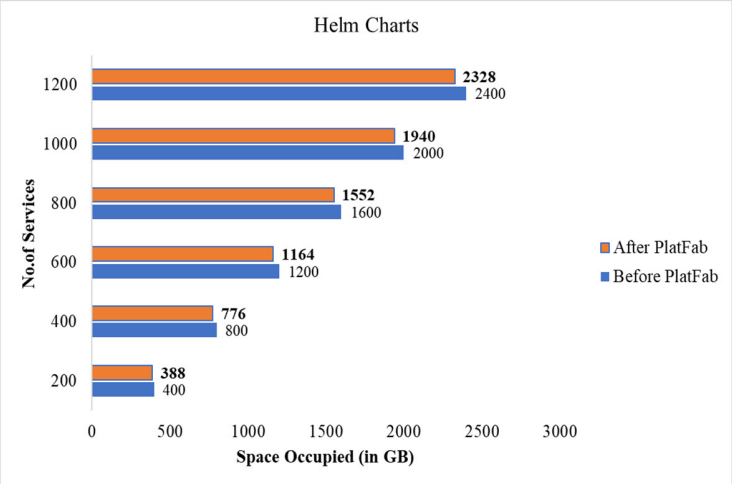

Fig. 5 Space Reduction for Microservice before and after integrating PlatFab

B. Vulnerability Response Time

This metric is used to evaluate the time taken from detecting a vulnerability to its resolution. This helps to evaluate the system's capacity to endure and minimize security risks, such as weaknesses, assaults, and unauthorized intrusion attempts. Security metrics encompass the measurement of both the time taken to respond to security incidents and the adherence to security regulations. A team of 12 members working for budgeting services with five services. Each month the number of vulnerabilities to fix were close to 40. Before implementing PlatFab, the team used to take five days to fix the vulnerability. But after the use of Renovate Presets at PlatFab, it has been reduced to one day. In the last one year, the team saved more than two months of developer productivity doing some minor fixes and used that time in actual development tasks. The increase in productivity after using PlatFab is compared with before using it is shown in Figure 6.

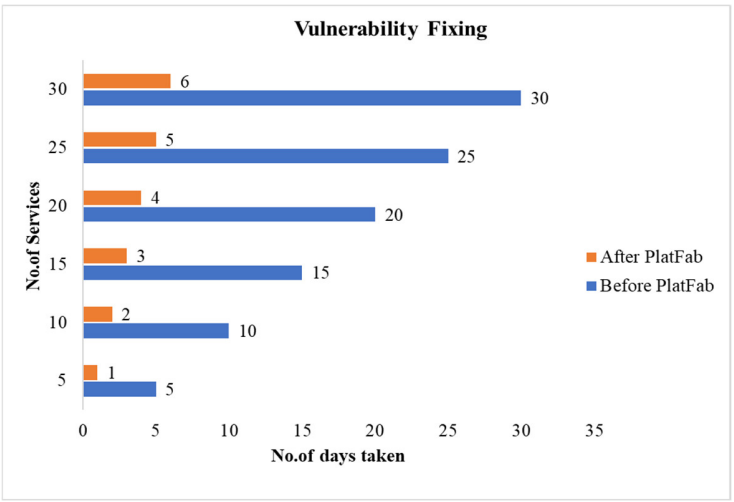

Fig. 6 Number of Days taken to Fix Vulnerability Issues before and after integrating PlatFab

C. Build Time Efficiency

Build Time Efficiency is measured as an average time taken to complete CI/CD pipeline build. After having an SDP base image in PlatFab, the build time for each service is reduced by 30 seconds to 1 minute. If each service runs at least once, it saves more than five hours of build time. If one service build runs 10 times in a day, the team is able to save 10 minutes of developer productivity each day, and for a year, it will be 60 hours saved. An organization unit having 100 developers can save 6000 hours which can be used for development tasks.

D. Production Downtime ITES

Production downtime is measured as the total time the system was unavailable during updates. In 2022, 12 production downtime issues occurred due to new fixes deployed. After the implementation of Blue-Green deployment in 2023, this number is reduced to seven, and in 2024, it is reduced to zero, as shown in Table 1.

TABLE 1
PRODUCTION DOWNTIME ISSUES FOR ALL QUARTERS OF YEAR 2021, 2022, 2023, 2024 AFTER THE USE OF PLATFAB

Quarter /Year	2021	2022	2023	2024
Quarter 1	3	4	1	0
Quarter 2	4	4	3	-
Quarter 3	4	4	2	-
Quarter 4	2	4	1	-
Total	**13**	**12**	**7**	**0**

The results show a significant improvement in the industrial application after implementing PlatFab, in the Budgeting Service. The summary provided in tables and graphical representations using charts illustrates the key benefits of PlatFab such as productivity improvements, deployment time and space reduction.

V. DISCUSSIONS

The results of the work demonstrate the considerable improvement in developer productivity metrics aligning with the platform engineering objectives. The proposed approach records potential benefits in the parameters such as build time, production time issue reduction, vulnerability fixes and resource utilization. It helps the developers to keep ahead of technological advancements by providing them with cutting edge tools, infrastructure deployment, security measures and other best practices. It also helps firms to scale development, react to market demands, and innovate across the software development lifecycle. The Blue-Green deployment approach reduced production downtime from 12 to 7 in a year. This reduction shows how Blue-Green deployment reduces the risks of releasing new patches and upgrades, making the system more stable and reliable. Additionally, the SDP base image in PlatFab significantly reduced build times across services. Each service's build time dropped 30 seconds to one minute. Scaled across the organization, these time saves saved over 6,000 developer hours yearly. This improves operational efficiency and frees up time and resources for product development and innovation. Significantly, the duration needed to address vulnerability concerns was drastically decreased from a span of five days to just 24 hours. The rapid increase in vulnerability management is essential for sustaining safe and resilient systems, particularly in industries where prompt actions against security threats are of utmost importance.

The following benefits are also reaped by the development team in their production when the above-mentioned services are implemented through PlatFab: (1) Teams that adopt PlatFab Helm Templates enjoy the benefits such as they are centrally managed as Kubernetes (K8s) deployment templates, Service Mesh Helm chart version is automatically updated, Optimal deployment configurations are set for services and Automatic enrolment to new features such as automated SDP and Canary deployments, (2) The Renovate Presets offer the functionalities which includes updating Service Mesh version and corresponding sidecars in a single PR, ignoring pre-release versions, applies default configuration for recommended PR scheduling in order to prevent the project being flooded with PRs during the workday, and will hold future presets from all Dev Fabric artifacts, (3) SDP combines tooling and procedures to protect customers from negative impacts during rollout of new software. The advantages of implementing it are rolling out SDP Test Operator for test validation during deployment to production and Leveraging Flagger for automated application analysis, promotion and rollback. Due to cross-org dependencies, rolling out an early version of the SDP Test Operator without Flagger dependency takes place.

The experimental results indicate that average build time for each service is reduced by one-minute which highlights the efficiency of Helm charts. By using SDP base images, 60 MB of space is saved leading to optimal resource utilization. Using Renovate Bots enables to fix the vulnerabilities in one day and down time issues reached to zero by using Blue-Green deployments. These finding highlights PlatFab addresses domain specific challenges. Organizations can speed up their product and service launch by lowering build times, downtime, and security vulnerability resolution. The approach implies how platform engineering can help complex industrial situations. Although build time and downtime were significantly reduced by the framework, resource utilization efficiency varied according to workload intensity, suggesting room for further optimization. Organizations can enhance their operational efficiency, scalability,

security, and reliability by combining classic SOA approaches with new platform engineering methodologies. The finding highlights the unique advantage of the research by emphasizing the importance of integrating platform engineering into industrial workflows. It also establishes a solid basis for future study and practical application in the industry, providing a clear plan for organizations seeking to utilize platform engineering to gain a competitive edge. In contrast to conventional methods, this study incorporates a feedback loop to incorporate user inputs straight into the platform development process, guaranteeing flexibility and ongoing growth.

VI. Conclusions

The proposed approach introduces PlatFab, a Platform engineering approach integrates workflows, tools and services and improves developer productivity in tech companies. They provide automated controls, container orchestration, development environments, necessary infrastructure, deployment pipelines, monitoring and storage solutions, security measures and optimizing techniques. Such an approach has been implemented in one kind of service called Budgeting service which involves budget expenses, approvals, expense tracking services in an organization. PlatFab in Budgeting service, streamlines the workflow, reduces time-to-market and ensures security during development. The efficiency of PlatFab was measured using parameters such as build time, space occupied, days to fix vulnerabilities, production downtime issues. The parameters were recorded before and after implementation of PlatFab in a particular division of the IT industry. The annual productivity i.e., build time savings ranges approximately 42 hours for s single developer ranges 10% – 20% per service. 80% reduction in vulnerability fix time, 3% enhance in resource utilization efficiency and the complete elimination of production downtime showcases the efficiency of PlatFab in optimizing developer productivity and system reliability. The reduction in build time highlights the efficiency of helm charts and SDP base images in streamlining pipeline execution. Similarly, vulnerability response time improvement underscored the importance of automated dependency tools such as Renovate Bots in enhancing security issues. Unlike previous studies that focuses on generic CI/CD optimization, this proposed work emphasizes domain specific challenges in industries. With the help of PlatFab, developers can focus on innovation, value creation and writing quality code and thereby, improving their productivity in the development and deployment of software. PlatFab helps developers to innovate, adapt, and prosper in a constantly changing environment as technology advances. The study can be further extended to other domain datasets.

Author Contributions: *Vaishnavi Srinivasan*: Conceptualization, Investigation Writing – original draft, Writing – review & editing. *Manimegalai Rajkumar*: Writing - Review & Editing, Supervision. *Srivatsan Santhanam*: Conceptualization, Data Curation, Methodology, Supervision, Project administration. *Arjit Garg*: Resources, Investigation, Data Curation, Methodology, Software, Validation,

All authors have read and agreed to the published version of the manuscript.

Funding: This research received no specific grant from any funding agency.

Conflicts of Interest: The authors declare no conflict of interest.

Data Availability: It cannot be shared openly as results are abstained using proprietary tools of an IT firm.

Informed Consent: There were no human subjects.

Institutional Review Board Statement: Not applicable.

Animal Subjects: There were no animal subjects.

ORCID:
Vaishnavi Srinivasan: https://orcid.org/0000-0001-6011-450X
Manimegalai Rajkumar: https://orcid.org/0000-0003-1398-4080
Srivatsan Santhanam: -
Arjit Garg: -

REFERENCES

[1] S. S, "A Study of Software Development Life Cycle Process Models," *SSRN Electronic Journal*, 2017, doi: 10.2139/ssrn.2988291.
[2] S. Ergasheva and A. Kruglov, "Software Development Life Cycle early phases and quality metrics: A Systematic Literature Review," *J Phys Conf Ser*, vol. 1694, no. 1, p. 012007, Dec. 2020, doi: 10.1088/1742-6596/1694/1/012007.
[3] G. Gurung, R. Shah, and D. P. Jaiswal, "Software Development Life Cycle Models-A Comparative Study," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 30–37, Jul. 2020, doi: 10.32628/CSEIT206410.
[4] S. Alsaqqa, S. Sawalha, and H. Abdel-Nabi, "Agile Software Development: Methodologies and Trends," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 14, no. 11, p. 246, Jul. 2020, doi: 10.3991/ijim.v14i11.13269.
[5] S. Suprihadi, S. Wijono, and K. D. Hartomo, "Analysis service architecture utilizing middleware for information services management systems," *International Journal of Applied Science and Engineering*, vol. 21, no. 2, pp. 1–15, 2024, doi: 10.6703/IJASE.202406_21(2).001.
[6] J. Lukkarinen, "Service-Oriented Architecture as an Enabling Technology for Sector Integration," Aalto University School of Electrical Engineering, 2023.
[7] Abd. Halim and S. M. Isa, "Application of Service-Oriented Architecture with Middleware at Bank XYZ to Implement the PDAM Payment," *IIJSE*, vol. 7, 2024.
[8] Afrizal Zein, "Implementation of Service Oriented Architecture in Mobile Applications to Improve System Flexibility, Interoperability, and Scalability," *Journal of Information System, Technology and Engineering*, vol. 2, no. 1, pp. 171–174, Mar. 2024, doi: 10.61487/jiste.v2i1.60.
[9] Z. Seremet and K. Rakic, "Platform Engineering and Site Reliability Engineering: The Path to DevOps Success," 2022, pp. 155–162. doi: 10.2507/daaam.scibook.2022.13.
[10] M. M. El Khatib, H. Alawadhi, and M. Al Mansoori, "Platform Engineering in Manufacturing: Role, Effect, Challenges and Opportunities," *International Journal of Business Analytics and Security (IJBAS)*, 2024.
[11] G. Blokdyk, *Platform engineering The Ultimate Step-By-Step Guide*. 5STARCooks, 2021.
[12] S. Kambhampaty, *SERVICE ORIENTED ARCHITECTURE : FOR ENTERPRISE APPLICATIONS*. Wiley India, 2008.
[13] G. Bergami, "Towards automating microservices orchestration through data-driven evolutionary architectures," *Service Oriented Computing and Applications*, vol. 18, no. 1, pp. 1–12, Mar. 2024, doi: 10.1007/s11761-024-00387-x.
[14] D. Coughlin and B. Lush, "From project to platform: a case study on evolving the software development team," *Library Management*, vol. 45, no. 1/2, pp. 37–45, Feb. 2024, doi: 10.1108/LM-08-2023-0080.
[15] A. Zutshi and A. Grilo, "The Emergence of Digital Platforms: A Conceptual Platform Architecture and impact on Industrial Engineering," *Comput Ind Eng*, vol. 136, pp. 546–555, Oct. 2019, doi: 10.1016/j.cie.2019.07.027.
[16] R. Amaro, R. Pereira, and M. Mira da Silva, "DevOps Metrics and KPIs: A Multivocal Literature Review," *ACM Comput Surv*, vol. 56, no. 9, pp. 1–41, Oct. 2024, doi: 10.1145/3652508.
[17] J. Ross, "Agile and DevOps: Elevating Software Quality through Collaborative Practices," *Journal of Science & Technology*, vol. 5, no. 1, 2024.
[18] S. Shafiq, C. Mayr-Dorn, A. Mashkoor, and A. Egyed, "Balanced knowledge distribution among software development teams—Observations from open- and closed-source software development," *Journal of Software: Evolution and Process*, vol. 36, no. 8, Aug. 2024, doi: 10.1002/smr.2655.
[19] A. A. Khan, J. A. Khan, M. A. Akbar, P. Zhou, and M. Fahmideh, "Insights into software development approaches: mining Q &A repositories," *Empir Softw Eng*, vol. 29, no. 1, p. 8, Jan. 2024, doi: 10.1007/s10664-023-10417-5.
[20] Ž. Grujić, M. Milić, and I. Antović, "Preliminary Experiences of Using the Azure DevOps Platform in Software Development Automation," in *2024 28th International Conference on Information Technology (IT)*, IEEE, Feb. 2024, pp. 1–4. doi: 10.1109/IT61232.2024.10475753.
[21] F. van der Linden and J. G. Wijnstra, "Platform Engineering for the Medical Domain," 2002, pp. 224–237. doi: 10.1007/3-540-47833-7_20.
[22] S. T. Makani and S. Jangampeta, "A Comparative Study of Platform Engineering Tools: Implications for System Design and Scalability," *International Journal of DevOps (IJDO)*, vol. 1, no. 1, 2024.
[23] AWS, "A Roadmap to Continuous Delivery Pipeline Maturity," https://pages.awscloud.com/rs/112-TZM-766/images/A-Roadmap-to-Continuous-Delivery-Pipeline-Maturity-dev-whitepaper.pdf.
[24] K. H. Prakash, "Impact of Platform Engineering Tools on System Design And Performance Metrics," *International Journal of Graphics and Multimedia*, vol. 11, no. 1, 2024.
[25] Yossi Carmon, "Platform Engineering: The 2024 Game-Changer in Tech," https://devops.com/platform-engineering-the-2024-game-changer-in-tech/ (accessed 2 June 2024).
[26] J. Spillner, "Quality Assessment and Improvement of Helm Charts for Kubernetes-Based Cloud Applications," *arXiv manuscript*, 2019.
[27] L. Rice, *Container Security: Fundamental Technology Concepts that Protect Containerized*. O'Reilly Media, 2020.
[28] A. Adithya, V. Vyas, V. Aaswin, S. Lanka, and G. R, "Vulnerability Scanning by CPE-CVE Matching," *Grenze International Journal of Engineering and Technology*, vol. 10, 2024.
[29] S. Buchanan, J. Rangama, and N. Bellavance, "Helm Charts for Azure Kubernetes Service," in *Introducing Azure Kubernetes Service*, Berkeley, CA: Apress, 2020, pp. 151–189. doi: 10.1007/978-1-4842-5519-3_8.
[30] F. Zuo and J. Rhee, "Vulnerability discovery based on source code patch commit mining: a systematic literature review," *Int J Inf Secur*, vol. 23, no. 2, pp. 1513–1526, Apr. 2024, doi: 10.1007/s10207-023-00795-8.
[31] I.-C. Donca, O. P. Stan, M. Misaros, D. Gota, and L. Miclea, "Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects," *Sensors*, vol. 22, no. 12, p. 4637, Jun. 2022, doi: 10.3390/s22124637.
[32] M. Bafana and A. Aabdulaziz, "Immutable Infrastructure in Practice: A Comprehensive Guide to AWS Deployment," *Asian American Research Letter Journal*, vol. 1, no. 1, 2024.