ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Building a Scalable API for Restaurant Online Ordering Systems

Akash Gill

USA

ABSTRACT

The digital transformation of the restaurant industry has made online ordering systems indispensable, enabling convenience for customers and operational efficiency for businesses. This article delves into the pivotal role of scalable APIs (Application Programming Interfaces) in enhancing restaurant operations amidst increasing consumer expectations and technological demands. APIs enable seamless communication between software components, addressing critical challenges such as managing high transaction volumes, integrating online ordering with Point of Sale (POS) systems, providing real-time updates, and ensuring data security. The key features of scalable APIs include real-time inventory synchronization, dynamic menu management, robust error handling, and multi-platform compatibility. These capabilities ensure that restaurants can adapt to high-traffic situations, minimize operational errors, and deliver a unified customer experience. Leveraging modern technologies like microservices architecture, event-driven designs, and programming languages like Go (Golang), APIs facilitate elastic scalability, ensuring operational efficiency even during peak demands. Additionally, the case study of SpotOn demonstrates the tangible benefits of implementing scalable APIs. These include improved transaction handling, seamless system integration, enhanced customer satisfaction, and operational resilience. APIs also support long-term growth by enabling third-party integrations, automating processes, and adapting to emerging technologies such as Artificial Intelligence (AI) and the Internet of Things (IoT). As the industry evolves, scalable APIs will be instrumental in driving innovation, meeting consumer demands, and fostering sustainable growth. This article positions APIs as not just technical enablers but strategic assets essential for restaurants aiming to thrive in a rapidly digitizing and competitive market.

***Corresponding author**

Akash Gill, USA.

Received: April 04, 2022; **Accepted:** April 11, 2022; **Published:** April 30, 2022

Keywords: Scalable APIs, Online Ordering, Point of Sale, Transaction Handling, Microservices Architecture, Inventory Synchronization, Customer Experience, Integration Systems, Data Security, Cloud-Based Solutions, Restaurant Technology, Operational Efficiency

Introduction

There has been a significant change in the restaurant business, especially over the past decade, as a result of digitalization. One of the most crucial shifts is the advent of online ordering systems, which have turned into a crucial tool for restaurants that fight the demands of a technological generation populace [1]. Such systems not only enable customers to order their food comfortably but also open various prospects for restaurants, including the development of already existing standards and increased convenience for consumers. This evolution has brought new questions, especially in coping with the complicated number of transactions and in integrating with an established pre-existing system like the POS. Being insightful about the competitive and rapidly changing environment, restaurants have accentuated the importance of operation efficiency scaling. Even established systems get bogged down under high traffic volume during specific hours of the day or other such demands. However, solutions spawned from large scalable APIs are indeed whopping game-changers. API stands for Application Programming Interface, and it is the means through which two or more software application components can interact effectively. If well developed and efficiently put into practice, they can assist restaurants in handling thousands of transactions efficiently, enhance operational efficiency, and enable the provision of excellent consumer satisfaction [2].

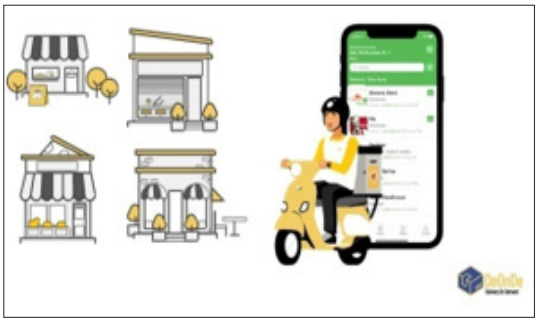

Figure 1: Online Food Delivery System

It is pertinent to emphasize the question of scalability even more. Some consumers change patronage outlets because restaurants that cannot meet the increased demand end up taking longer to attend to their clients. Think of frequent complaints that feature delayed orders, wrong updates for an item, or overall system breakdown during peak times [3]. All these problems not only hinder the flow of work but also blow the restaurant's profile and customer allegiance. It is possible to unsnarl these pain points with the help of scalable APIs. Meanwhile, restaurants' systems are going to be ready for any load.

Scalable APIs also help close the online and in-store gap, or at least diners can contribute to this area. For example, linking online orders with POS systems guarantees up-to-date information on stocks, prices, and order positions. Thus, integration solves input problems, saves time, and minimizes errors that would be

so embarrassing to the restaurant service attendees. APIs can be used to allow restaurants to integrate with any range of services, some integral to a restaurant app and others as third-party delivery services, thus maintaining a consistent customer experience across an app and site. API scaling is not without its pains, especially so if one is a small to medium restaurant with little or no IT support and resources. However, new technologies, especially cloud-based solutions, have overcome most of these system designs. Even small businesses and other establishments can flex their muscles through more easily scalable APIs, opening new and potential markets to them and improving efficiency and effectiveness [4].

In this article, the core issues of constructing a scalable API for restaurant online ordering platforms are discussed. Using practical cases and technology details, it analyses how such APIs may improve the restaurant's functioning, as well as the handling of a large transaction load, POS system integration, and real-time updates [5]. This guideline will ensure that restaurant owner, developer, or people interested in the local commercial real estate industry understand how to best take advantage of scalable APIs as a business tool in the modern age.

Problem Overview

Online food ordering also has become one of the key trends that have transformed the restaurant business and brought several high potentials for business growth and customer participation [6]. However, this has also helped to bring out operational pains that restaurants ought to overcome in light of new shifts. Issues concerning online ordering are critical to a restaurant business. The success of any undertaking depends on how customers are served, how efficiently a restaurant operates, and the amount of profit it is likely to make. To design an efficient workflow, it is essential to know these difficulties to avoid stagnation of the program's further development. A major challenge is the processing of large amounts of transactions within a short period, occasioned by peak time or promotions. For instance, a busy restaurant may get hundreds of orders within a short time, something that is uncomfortable for traditional systems. When such spikes cannot be managed properly, they slow down or even cause systems to freeze or reject orders, which ultimately results in disgruntled customers and fewer sales. This leaves most restaurants open to such interferences mainly because most of the existing solutions offer limited scaling options. This makes it easy for the restaurant to experience such a raw deal during a lunch or dinner rush.

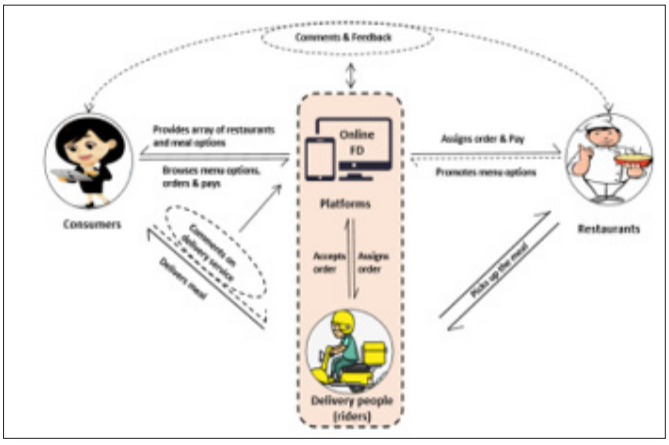

Figure 2: A Review of Online Food Delivery Platforms

Another critical problem is the links between online ordering functionalities and already existing Point of Sale (POS) systems [7]. A POS system is the operational control hub of a restaurant since it deals with orders, stock, cash, and customers. However, most online ordering platforms go outside the POS systems, leading to a dissociation between online and physical stores [8]. Such a structure leads to data duplicities, which in turn lowers efficiency by creating opportunities to input wrong data like inaccurate order details or mismatched inventory. The problems detailed above do more than create operational waste. They also infuriate the customers when their delivery or order is off. Real-time updates are another area of severe discomfort. Consumers and customers expect to receive reliable and real-time details concerning meal availability, order status, and delivery time [9]. There exists no real-time synchronization, which can cause the customer to order the wrong products or have to wait longer without being notified. Such problems can have an unwelcome effect on the level of trust and lead to repeated sales. To restaurants, the lack of a capability to offer initial updates damages their image and masks operational realities.

Figure 3: Benefits of API in Small Businesses

Integration, for instance, enriches scalability and also enhances data security and system reliability. Electronic ordering systems require handling customer information, such as payments and identification data. Large traffic can cause many problems, and an organization's systems can be prone to being hacked, perhaps taken down by denial-of-service attacks, or having valuable information stolen. The ability to guarantee data protection and, at the same time, provide free-flowing functionality provides systems that are scalable yet strong, something that traditional structures often overlook. Other problems stem from operational inefficiencies. Some of the common issues inherent to restaurants today include a lack of integration of online ordering systems with physical outlets. Hence, most eateries have to undertake interventions to ensure that they synchronize their web applications with the physical stores, which is costly and time-consuming [10]. Individual members of staff could find themselves typing in orders into the POS manually, checking stock quantities by hand, or dealing with customer complaints as a result of the inaccuracy that the system presents. Such processes waste a lot of time and resources that would otherwise be useful in important tasks like food preparation and customer relations.

Rather than directly competing with each other, delivery services now feature third-party facilitators, which complicate matters. Even though these channels provide increased exposure, they require interaction with restaurant networks to operate without compromising on flow [11]. For instance, order detail transmission processes to third-party systems may entail some delays or be incongruent with the original. This leads to delayed order deliveries, wrong products ordered, and significantly denting the restaurant's image. The questioned traditional systems reach their most significant disadvantage during periods of the organization's growth or changing customer expectations. For example, a restaurant that begins to add new dishes to its menu or new units must have its systems grow in direct correlation to the changes. However, such structures make it difficult to be flexible and flexible when changing since well-established systems usually develop bottlenecks that slow down growth. Likewise, new functionalities that restaurants pick, like intelligent interfaces or variable pricing, need APIs that would enable and accommodate them.

The COVID-19 pandemic expanded the importance of users with proper ordering processes, more specifically requiring a strong online ordering system [12]. During this period, most restaurants relied so much on online sales in order to survive fully. The companies that had scalable and integrated systems in place were in a better place to change swiftly and address growth in volumes. On the other hand, organizations with fragmented or old systems experienced operational discontinuity, reduced sales, and unsatisfied customers. Clients' expectations are changing rapidly. Today, needy customers are not only concerned about convenience in products and services but also expect speed, accuracy, and personalization [13]. They demand systems that are capable of handling customization, preferences, and payment modes hassle-free. This does foster the current expectations; it requires immense technological support that legacy systems do not possess.

The current issues that affect restaurants that seek to embrace the digital environment include high volume, integration of online and physical, real-time updates, security, and reliability, as well as addressing changing customer needs. This is accentuated by operational problems, challenging third-party connectivity, and the pressures of massive expansion or external shocks. To address these pain points, organizations need an approach to become integrated, work more efficiently, improve their performance, and provide customers with a better experience. The concept of Scalable APIs is the way to go, and it will help restaurants steer around these challenges and succeed in the modern world that is going digital [14].

Key Features of a Scalable API for Online Ordering

A scalable API is the foundational element of today's restaurant online ordering system. It fulfills the role of a middleman between different software parts, which allows them to communicate and exchange data. To address these requirements, the solution with high volume and real-time update frequency and possible integration issues must involve specific options to guarantee scaling, stability, and speed [15]. Here are the key factors that could help determine a healthy and sound API platform for online ordering systems.

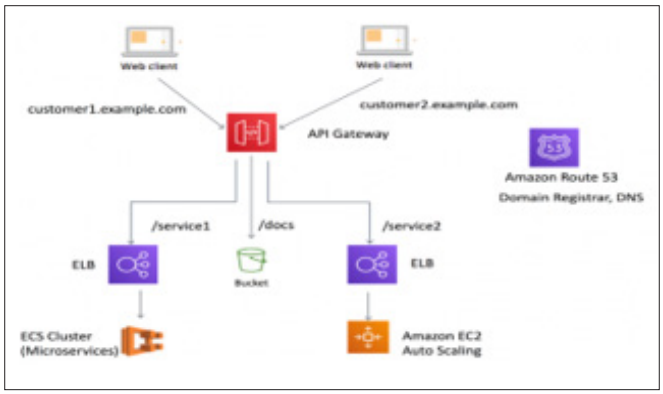

Figure 4: A Scalable API Design

Integration with Point of Sale (POS) Systems

Every restaurant has a Point of Sale (POS) function at the center, where orders, payments, and inventory control take place. An API designed for scalability has to allow integration with POS systems so that the processes within online stores can be in harmony with physical stores [16].

Real-time Data Exchange: POS should be integrated with the online ordering system so that both are connected instantly. For instance, when a customer places an online order, the order should be forwarded and added to the POS system to avoid keying it again and again.

Inventory Management: Integration makes it possible for inventory levels to be updated in real-time to afford customers the solutions that they desire but are currently available in the store. This reduces times such as overselling or even having to refund customers for certain products not available in the store.

Unified Customer Experience: Maintaining the complete integration of online and offline data helps ensure the proper pricing, promotion rates, and bonuses at all possible customer Interfaces.

High Transaction Volume Handling

Scalability is a feature and a key element of the application since restaurants experience a high number of orders during business days, promotions, or holidays.

Load Balancing: Any API that is to be scalable must be able to evenly distribute the incoming traffic so that none of the server's overload. Load balancing eliminates slow response time and system failures.

Concurrency Management: Most customers place multiple orders at the same time, so the API must be designed to handle a large number of requests at once.

Elastic Scalability: As resources can be automatically provisioned up or down depending on use, the API is always ready to service all incoming traffic while inoffensive when traffic dips. For instance, an eatery with a flash sale is most likely to have its orders increase by folds within minutes of posting the sale on its online platforms. A scalable API also ensures that such a surge is managed smoothly without any delays or sellers [17].

Security and Data Integrity

Since a greater number of consumers now order from the comfort of their homes, meal delivery services must protect consumers' data. Personal information, and in particular payment details, are for a customer associated with restaurant consumption; that is why API security is paramount.

Encryption: This information should be encrypted in a way that conforms to industry standards, such as Transport Layer Security (TLS) while using other external systems. This guards information from all forms of eavesdropping or interference.

Authentication and Authorization: Applications should also use complex mechanisms for authorization, such as OAuth 2.0. This allows only specified systems and users to perform specific operations or gain access to specified data.

Compliance: Payment information should be protected by PCI DSS, while personal info should be protected by GDPR, depending on the location.

Monitoring and Alerts: Security requirements must be integrated into an API that is monitored to rapidly identify unusual behavior or malicious attempts to access an application.

Real-Time Updates

Consumers demand timely product information about the menu, the status of an order placed, or the estimated time taken for delivery [18]. A scalable API needs to deliver real-time update capabilities across all the implemented channels.

Dynamic Menu Management: APIs must incorporate real-time changes to the menu, its prices, and menu options. For example, if a particular dish is out of stock, the system must update this information on every channel at once.

Order Status Notifications: APIs can be used to track order processing, shipment, and delivery in real-time, which provides more information to the customer.

Event-Driven Architecture: Web APIs can also push information to users instead of the current practice of users pulling data by offering real-time technologies such as WebSockets or server-sent events.

Multi-Platform Support

Today's consumer engages with restaurants in both human and technological ways through applications, websites, kiosks, and third-party delivery services. The platforms that a scalable API should support include these while at the same time must be consistent [19].

Cross-Platform Compatibility: APIs should perform the same functions on iOS, Android, web browsers, and other platforms so that customers get a similar interface regardless of the device they employ.

Third-Party Integration: Some of the external service providers include DoorDash or Uber Eats, which restaurants frequently use. An API for this kind of scale must flow seamlessly with these platforms, often with real-time order, inventory, and payment synchronization.

Future-Ready Design: APIs can be created that could be extended to support new forms of interfaces in the future, such as voice or AR interfaces for placing orders, without redesigning them from scratch.

Performance Optimization

The functionality of the online ordering system must be fast and reliable for a restaurant [20]. Every delay or mistake becomes apparent to the customer and can cost a company much money.

Caching Mechanisms: APIs can temporarily store other frequently accessed data, eliminating the need for multiple database calls and speeding up the response time.

Asynchronous Processing: A carryout, payment, or order confirmation, for example, should be handled in the background. Then, the managers or other users have to produce and continue with other system operations while waiting for the payment to be processed or the order to be confirmed.

API Gateway: An API gateway can be described as a central, deeply intelligent node between a client and a server that manages traffic, applies protection measures, and maintains uniform performance.

Customizability and Extensibility

Food businesses have always had or sometimes get some specific requirements depending on the size of their kitchens, the type of customers they have, and how they run their business. An API, when intended to be scalable, needs to be adaptable and modular to address all these specifications [21].

Modular Design: Integrated modular APIs also turn on or off the various parts of the services to meet a restaurant's requirements. For example, a restaurant may decide to include loyalty rewards into its business model but reject the third-party delivery service.

Developer-Friendly Documentation: Descriptive and detailed documentation guarantees developers access to enhance or modify the API.

Plug-and-Play Integrations: APIs must also be able to support new components like artificial intelligence recommendation systems or complex analytics as new components without adding to existing system complexity or workload.

Robust Error Handling

It is normal for issues to occur in an API. However, how the API responds to these issues determines the user response and operation flow.

Descriptive Error Messages: APIs should be well documented on the kind of errors to give the developer an easy time in debugging errors.

Retry Mechanisms: In the case of smaller temporary errors, like a temporary lack of connectivity, APIs should attempt to make the requests again as is normal practice.

Failover Support: In case of server failures, which are poignant in API implementations, system redundancy should still provide service continuity.

Analytics and Insights

Using analytics to inform decisions is especially crucial to restaurants that are looking for ways to improve functioning and customer satisfaction. A scalable API should have some

components within it that can support the determination of data gathering and processing [22].

Figure 5: Restaurant Analytics Technologies

Order Trends: APIs can also determine customers' ordering behavior, which would be of particular interest to restaurants. They can use them to find out what menu items are ordered most often or at what times of the day.

Operational Metrics: Measuring performance in areas such as order cycling time, percentage of errors made, and traffic loads speaks to areas that need intervention.

Customer Insights: In the future, APIs should include information such as customer preference and interaction that assists the marketing department in its marketing and even the kitchen in the menu it prepares.

A scalable API is one of the most important and revolutionary tools for restaurants that are willing to operate in the environment of a fast food ordering option. By interfacing with POS systems, processing large volumes of transactions, securing data, offering real-time solutions, and cross-platform compatibility, APIs help restaurant chains perform effectively while providing enhanced value to the customers. Other advantages, such as performance improvement, flexibility, good error handling, and data analysis, also show that the API plays a fundamental role in current restaurants. Effectively handling present issues entails investments in a scalable API plan for embracing the future and adapting to change in a new era driven by the digitization of the food business [23].

Technical Implementation

Constructing a flexible API for restaurant online ordering systems should follow certain steps in design, creation, testing, and implementation. Issues concerning the language to be used, how the system would be structured, and how integration would take place define the viability, robustness, and efficiency of the system [24]. The following section describes the key technical components required to achieve a high-scaling API.

Table 1: Components for Creating a Scalable Restaurant API

Component	Key Features	Examples	Benefits
Programming Language	Go for speed, concurrency, low latency	Goroutines for parallel requests	Scalability, developer productivity
System Architecture	Microservices, containerization	Docker, Kubernetes	Flexible updates, resource efficiency
Database Design	RDBMS, NoSQL, sharding, replication	PostgreSQL, MongoDB	Data consistency, scalability
API Development	REST, GraphQL, endpoint design	GET, POST, flexible data queries	Ease of integration, reliable performance
Real-Time Processing	Event-driven, WebSockets	Apache Kafka, order tracking	Instant updates, customer satisfaction
Security Measures	Encryption, authentication, rate limiting	OAuth 2.0, TLS, input validation	Data protection, misuse prevention
Performance Optimization	Caching, CDN, monitoring	Redis, Prometheus	Faster responses, resource efficiency
Third-Party Integration	Payment and delivery service APIs	Stripe, PayPal, logistics tracking	Smooth transactions, real-time updates
Testing	Unit, load, and regression testing	JMeter, Locust	Stability, high performance
Deployment	CI/CD pipelines, cloud hosting, auto-scaling	AWS, Azure, Docker containers	Reliability, scalability, minimal downtime

Programming Language Choice

The programming language that is used while developing an API does have a powerful impact on it in terms of performance and extensibility. For SpotOn's API, Go (also known as Golang) was selected, and for good reasons:

Performance and Speed: Go is a compiled language, which means it is translated directly into machine code. This makes it faster than interpreted languages like Python or JavaScript. It is perfectly suitable for dealing with large volumes of transactions in real time.

Concurrency Support: Concurrency is an in-built concept implemented using goroutines and channels in Go, which makes grouping multiple tasks at a single time feasible without burdening the systems. This feature is especially important when APIs are dealing with hundreds or thousands of concurrent requests.

Minimal Latency: The Go's lightweight runtime feel and non-invasive memory utilization provide low latency, which is important for customer services.

Developer Productivity: Go is easy, effective, and highly reliable. It helps developers write efficient, easy-to-manage code as soon as possible, which saves time and effort.

Other languages, such as Java or Node.js, can also be used to build APIs. However, few languages have the perfect combination of speed, scalability, and ease of usage as Go does.

System Architecture

That is, how an API is designed defines how it will fare under increasing loads. A microservice architecture is normally preferred for scalable systems over a monolithic one [25].

Microservices Architecture: In this architecture, the system is split into loosely coupled services that can work together, decomposing the executed tasks into processing the order, updating the inventory, or even processing payments. This is the case with each service in that the developers can scale or update one or more services without affecting the whole system. For instance, during high-traffic periods, only the order processing service may be busy, requiring more resources.

Monolithic Challenges: Monolithic systems, on the other hand, have highly integrated components. They are easier to implement initially than physical servers, but when the system has grown complex, it becomes hard to add to or even manage. The use of containerization tools such as Docker and orchestration platforms such as Kubernetes guarantees that microservices are correctly deployed and scaled.

Database Design

Good database design is important to create an efficient way of storing and retrieving data. The database must also be able to handle a large number of reads and writes, remain consistent, and guarantee information integrity.

Relational Databases (RDBMS): Structured data often uses systems such as Pg SQL, PostgreSQL, OR MySQL. Many of them impose strict structures and force data constraints to be rigid across the transaction [26]. For example, the order details with the inventory data can be related to prevent data disconnect throughout modification.

NoSQL Databases: For unstructured and semi-structured data, a wag base, like MongoDB or Cassandra, allows the user more freedom and better scalability. They are most applicable for storing logs, customer behavior data, or data that needs to be frequently retrieved.

Sharding and Replication: Scaled-out measures like sharding, in which data is divided across different servers, and replication, in which data is copied, enhance scalability and reliability to enable the API to accommodate a growing number of transactions without being slowed down.

API Development

Creating APIs should be directed at compliance with necessary standards, such as speed, reliability, and security [27]. This includes matters concerning the choice of endpoints, the fashion of processing the requests, and data transmission.

RESTful APIs: REST has been named architectural style for APIs since it is easy to implement and is based on web technologies. It is important to understand that REST APIs use common HTTP methods such as GET, POST, PUT, and DELETE for operation.

Graph QL: In systems where data can be queried flexibly, Graph QL can be used in place of REST. It enables clients to ask for specific fields, eliminating cases of data over- or under-fetching.

Endpoint Design: Endpoint names like orders or menus with no prefixes or unnecessary numbers like v2 are amazing for the developer. The pathways, such as pagination and filter mechanisms, aim to facilitate data retrieval for big data sets.

Real-Time Processing

Operational data should be processed in real-time, especially to update inventories, order status, and customer notifications. For instance, order status is displayed online for an order management system.

Event-Driven Architecture: The API should be event-driven using ideas such as message brokers RabbitMQ or Apache Kafka. For example, when an order is received, an event requiring a stock update, an alert to the kitchen, and an order confirmation is created for the buyer [28].

Web Sockets: Another feature available with WebSockets is making requests in real time. These can set up long-lived connections with the API client, which is most helpful in tracking order numbers.

Security Measures

Undoubtedly, APIs for the online ordering interface work with significant user information, so security is crucial [29]. Key measures include:

Encryption: Encrypt all the ports using HTTP and TLS to increase the security of the data being transmitted.

Authentication: Use OAuth 2.0 or JWT for the user and application authentication methods since they are secure.

Rate Limiting: To avoid misuse or attacks like DDoS, APIs should set a quota on the number of times a client can make a request in X amount of time.

Data Validation: One is that all incoming data must be validated to avoid injection attacks or invalid inputs getting through. For example, all SQL input, such as entering a malicious command, should be sanitized before engaging the database.

Performance Optimization

Website scalability extends beyond the ability to host more visitors but also maintains efficiency [30].

Caching: Some of the techniques that need to be implemented may include using caching layers from well-known systems Redis or Memcached to store the foods menu most frequently accessed to minimize the use of queries from the database.

CDN Integration: Cache delivery services can improve API response times of static objects, say images of menu options.

Monitoring Tools: Tools like Prometheus or Grafana give real-time information about API usage and can determine issues such as performance bottlenecks and resource usage.

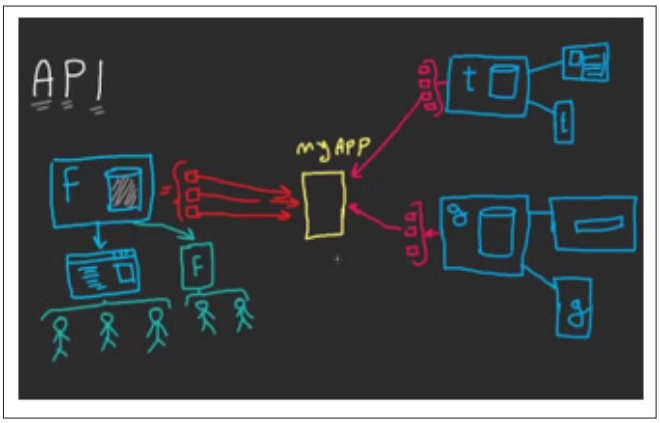

Figure 6: API Development Process

Integration with Third-Party Systems
Online ordering systems need to integrate with external applications, such as delivery service providers or payment processors [31].

Payment Gateway Integration: APIs must be able to integrate with secure gateways such as Stripe, PayPal, and Square to process payments.

Delivery Partner APIs: Order management and tracking depend on synchronizing orders with third-party delivery services. Therefore, APIs enforcing real-time order dispatching and status updates depend on this synchronization.

Testing and Quality Assurance
Several tests provide a guarantee of API stability and efficiency.

Unit Testing: This involves validating individual functions or endpoints to determine whether they are working properly.

Load Testing: Nowadays, there are many performance testing tools, such as JMeter or Locust, that can define high traffic and see how the API responds and whether it is reliable.

Regression Testing: Equally rigorous checks should be conducted to create new updates or features without flaws or distortions of the original performance.

Monitoring in Production: Monitoring after deployment provides developers insights into actual problems in context so problem prevention can be addressed.

Deployment and Scaling
Selecting strategies for the deployment of systems has a huge influence on scalability and reliability.
CI/CD Pipelines: CI/CD pipelines do exactly that – they test and deploy continuously, thus avoiding long outage periods during rollouts.

Cloud Hosting: Deploying the API architecture with the help of AWS Google Cloud or Azure offers automatic scaling and high availability.

Containerization: Docker containers for consistency in deployment make the scalability and maintenance much easier.

Auto-Scaling: Cloud services provide auto-scaling, which means that extra resources are acquired during rush hour.

Post-Implementation Metrics and Optimization
Performance metrics are also analyzed continually after the deployment to maintain and optimize the API [32].

Key Metrics to Monitor: Response times and latency. Failure and success rates involve error pictures and successful requests. Availability and uptime of the server.

Continuous Improvements: API developers should review feedback and analytics to modify API functionalities, solve problems, and extend API capabilities.

When designing online ordering systems for restaurants, it is necessary to make a number of decisions at the micro level, even choosing the programming language to be used. Decisions about deployment options are equally important when designing a scalable API for the new project. Using microservices architecture, getting inspired from Go, helps the developers to build APIs that can result in high TPS, can easily integrate with POS, and can help in real-time updates. API is more than just the capability that serves the currently needed functions. It also turns restaurants into a future-ready organization, making it vital for their digital programs [33].

Case Study: SpotOn's API Implementation
SpotOn has created a revolutionary restaurant online ordering system through the establishment of a limitless API, which provides a perfect illustration of how a technologically sound structure can work wonders for restaurant businesses that thrive in a fast-food culture. From recognizing high transaction volume, POS integration issues, and real-time updates, the implementation of API at SpotOn proved how restaurant performance can be enhanced through sound API design [34]. In this particular case, an analysis of the issues faced by SpotOn, the technologies it adopted, and the benefits derived by the company are discussed.

Challenges Faced by Spot On
The use of online ordering had a positive effect in opening up opportunities for the growth of SpotOn's restaurant clients but, at the same time, posed a threat to that same growth. While the increased demand provided growth potential, it also highlighted several operational bottlenecks.

Handling High Transaction Volumes: Several restaurants noted that they experience many difficulties when buyers place many orders at the same time. In traditional systems, this process clogged up the inadequate systems by overloading them with too much work, which led to delays, mistakes, and even system breakdowns. It was realized that restaurants required a system that could be customarily grown and shrunk to accommodate traffic flow.

Integrating Online Orders with POS Systems: Separate online ordering systems and in-store POS systems were common, and the two systems usually needed to be integrated manually. This not only involved staff time but also negatively impacted order processing, inventory, and reporting systems.

Real-Time Updates: Customers demanded real-time information about the availability of different menus, the status of orders, and the progress of delivery. However, existing systems could not smoothly offer these updates, which resulted in a poor customer experience and a number of operational issues.

That is why Spot On comprehended the necessity of developing a scalable, efficient, and high-performing API to solve these issues.

Technical Solutions Implemented

To overcome these challenges, an API was created that was scalable and improved the online ordering processes for restaurants for SpotOn. The API used current state-of-the-art technology and implemented state-of-the-art industry practices.

Programming Language: Go (Golang): Go was selected as the API's core technology because of the language's performance, concurrency, and minimal overhead. These properties enabled the system to handle large numbers of transactions with a relatively small delay.

Integration with POS Systems: The API integrated the flow of orders from online ordering to the POS from in-store systems without any hitches. Synchronization in real time meant that orders entered on the website immediately appeared in the POS. There was no manual entry, which in turn meant no mistakes. Inventories and menus were real-time, informing customers of the available stocks and avoiding situations where an item's selling ability was outstripped by stock.

System Architecture: Microservices: To enhance Service Scalability and maintainability, SpotOn incorporated and implemented a Microservice design pattern. Each service was designed to perform a specific task, such as order taking or billing and payments. This organization enabled SpotOn to retrieve individual fragments for scaling and updating continuous constituents within the framework without destabilizing the total system.

Event-Driven Design for Real-Time Updates: The API used an event-driven architecture driven by message brokers such as RabbitMQ. Whenever a customer ordered food, events were set off to notify the kitchen, inform the concerned customer, and change the stock immediately.

Performance Optimization: To reduce database queries, SpotOn uses the cache to store regular and random data, such as the details of a particular menu. Moderate and distribute incoming traffic to different servers to ensure that all servers can handle traffic during different periods.

Security Measures: HTTPS and TLS protocols were used to create secure forms of communication. OAuth 2.0 protocols, for example, enabled only certified users or systems to access restricted ports.

Testing and Deployment: Load testing was performed on the API to determine how it would perform at times of high usage and to determine possible causes of delays [35]. SpotOn integrated elaborate CI/CD pipelines to facilitate the timely and efficient delivery of updates and new features.

Outcomes Achieved

The introduction and efficiency of SpotOn's scalable API can increase restaurant efficiency and enhance customer satisfaction.

Enhanced Transaction Handling: The API was able to handle large amounts of transactions during busy times, such as when people order during lunch and dinner. There were fewer cases of slow system and order mistakes, especially during promotion periods and festive seasons.

Seamless Integration: To ensure that the order-taking process was real-time and seamless, SpotOn integrated the online ordering system with the in-store POS systems, eliminating manual entry jobs. This integration also solved some operational overhead, leading staff to concentrate on the areas of customer relations and meal preparation and service.

Improved Real-Time Updates: Customers were informed instantly of the status of their order, right from the time it was placed to the time it was delivered. The constant updating of menus made it possible to avoid processing and serving orders for out-of-stock items, resulting in little to no refunds.

Operational Efficiency: The automatic updates of inventory and order synchronization made a number of differences by allocating much of the time and effort. The event-driven strategy enhanced productivity because there were fewer interruptions, especially at busy restaurants.

Customer Satisfaction: Efficiencies created by faster and more accurate order processing improved the customer's experience. The restaurant owners using the API revealed that the application resulted in better customer loyalty and higher satisfaction levels.

Scalability for Growth: The use of the microservices architecture and the cloud deployment allowed restaurants to expand their operations according to the evolution of their business. That is why the API provided the ability to add new features and integrations without much refactoring.

Lessons Learned

The success of SpotOn's API implementation highlights several key lessons for building scalable systems in the restaurant industry [36].

Choose the Right Technology Stack: To realize scalability and reliability, it is important to learn a high-performance language such as Go and embrace new architectural patterns such as microservices.

Focus on Integration: Compatibility with other systems is important in maintaining operations without necessarily requiring much input from human beings.

Prioritize Real-Time Capabilities: Customers want to receive updates as soon as possible, so real-time processing and notification are critical to modern systems.

Test for Scale: Another critical segment of the testing load is necessary to determine the system's performance at maximum traffic load.

The integration of the SpotOn API is an ideal model for benchmarking efficient and effective online ordering. The integration and real-time update of Go along with Microservices architec-

ture helped Spot On overcome the issues faced by its restaurant clients. The outcome has unveiled the potential benefits such as enhanced transaction processing, integration, and customer satisfaction precipitated by a successful integration of API. This case proves that constant investments in flexible technologies will help the restaurant industry to advance, become more efficient, and satisfy the needs of clients.

Benefits of a Scalable API in the Restaurant Industry

It has now become apparent that combining an Online Ordering System with scalable APIs has significantly changed the functioning and relationships of restaurants. These APIs offer reliable solutions to many operations, including handling large volumes of transactions, improving business efficiency, and improving customer experiences. The advantages are not limited to the technical outcomes but hold vast functional, economic, and strategic impacts on restaurants of all types [37].

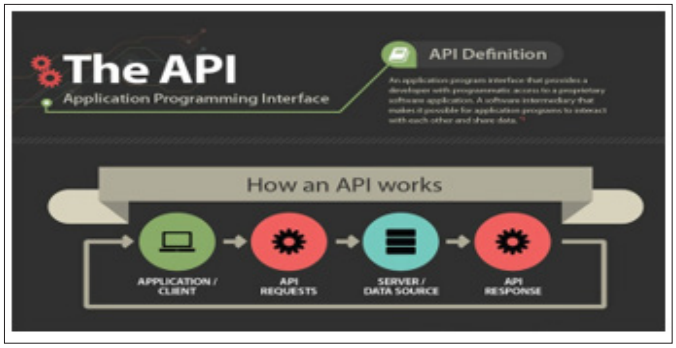

Figure 9: Benefits of Scalable API

New Operational Efficiency

A scalable API can also increase the efficiency of a restaurant's work, depending on the scale of operations implemented by the restaurants.

Seamless Integration with POS Systems: POS systems integrate with online ordering systems and other front-of-house systems and eliminate the need to transpose data. Such integration also makes it possible to promptly update orders at the POS system, which means that personnel will not waste their time and can efficiently work in the kitchen or with customers.

Inventory Management: Integration APIs can be integrated, allowing a business to synchronize its inventory in real-time across multiple channels. Depending on the orders received, the API updates the quantities on hand. This prevents restaurants from offering out-of-stock products and allows customers to see only available products.

Efficient Workflows: Back-of-house operations such as order processing and updating inventories are made easier through mechanized processes, freeing up time during peak hours.

Enhanced Customer Experience

In this particular restaurant industry, customer demand has changed. There is a focus on speed, accuracy, and convenience. Restaurant scaling using an API solution can effectively meet such expectations.

Faster Order Processing: The APIs are created to accommodate a large number of transactions, which leads to customers waiting a shorter time for their orders to be confirmed. Ordering something happens in the blink of an eye if the purchase is made on the Internet or through an application.

Real-Time Updates: Customers are notified as soon as orders are processed, prepared, or out for delivery, thus eliminating the likelihood of the 'dark side of the moon' effect. Real-time synchronization of a menu means that customers can only order what is available in the inventory to eliminate disappointment.

Personalized Experiences: Adaptive APIs provide additional enhanced capabilities such as redeeming points, personalized offers, and recommended products and services to the customer.

Scalability to Handle Growth

Based on customer traffic and sales, growth might come from increasing the number of outlets or simply enhancing the online platform. As this happens, new challenges arise that have to be addressed.

Figure 10: Importance of Scalability

Dynamic Resource Allocation: It is capable of handling high volume, low volume, and everything in between, depending on the traffic, say during a promotion period or holiday. Restaurants can cater to the increase in orders without the systems conking off or tripping.

Support for New Integrations: A scalable API makes it possible to add more functionalities and features, such as delivery partners, payment solutions, or AI integrations, without significant revolutions to the restaurants' systems. Such flexibility is necessary because it facilitates the adjustment process to emerging market conditions.

Facilitating Expansion: This also provides for easy operational scaling for a restaurant, whether it is a single outlet adding more outlets or a chain opening across international markets.

Cost Savings

A scalable API makes the system efficient, which means that one can avoid the high costs of redesigning a system in the future.

Reduced Manual Labor: Automating order synchronization, inventory control, and feedback reduces time and staff effort spent on repetitive tasks.

Optimized Resource Usage: Scalable APIs can efficiently use infrastructure costs by using cloud infrastructure to compute resources during off-peak times.

Error Reduction: Fewer mistakes in purchase orders and stocking mean that there are fewer spoils, fewer returns, and fewer costs involved in addressing consumer grievances.

Competitive Advantage
It shows that restaurants located in a highly competitive environment must embrace various technologies to survive. A scalable API gives a restaurant features that can help it stand out from the competition.

Faster Service Times: Catering organizations with scalable APIs can process more orders in less time, hence meeting customers' time expectations.

Integration with Third-Party Services: Integrated third-party delivery solutions such as Uber Eats and DoorDash are the key enablers, showcasing how restaurants can leverage scalable API solutions and grow clientele.

Data-Driven Insights: Application programming interfaces can learn customer ordering and operational information and offer data on ordering profiles, business hours, and clientele choices. This information enables restaurants to decide on what menu to offer, the right price to charge, and the best marketing approach to use.

Adaptability to Emerging Trends
Situational analysis of restaurant operations must consider the dynamics of an industry experiencing consistent changes in technological and consumer preferences. Scalable APIs put restaurants in a better position to manage these changes as they occur.

AI and Machine Learning Integration: It also indicates that APIs can introduce deep AI mechanisms, including forecast mechanisms for stock control or smart suggestions on what to order next, depending on customers' choices.

IoT Integration: Enterprise-scalable APIs can interact with the Internet of Things (IoT), such as smart kitchen equipment or automated inventories, improving the flow of business.

Support for New Ordering Channels: As the ways to order food change, e.g., voice ordering or AR menu, APIs can normalize with these changes while minimal changes will be needed.

Improved Security and Reliability
Protecting customer information is critical when placing orders online since some required details include payment information. Measures encouraging the increase of security and reliability are associated with the evolution of scalable APIs.

Data Encryption: APIs ensure the security of data communication protocols such as HTTP/SSL and Transport Layer Security for securing customers' and operations' data.

Authentication and Authorization: OAuth or other similar standards allow for APIs to guarantee that only deterministic users and systems can access certain functions.

System Reliability: Load balancing, caching, and failover methods for commonly used systems guarantee systems' availability during peak loads or single failure, ensuring customer loyalty and satisfaction.

Long-Term Technological Sustainability
It results in a sophisticated API that is easily scalable to meet restaurant needs while avoiding expensive and frequent upgrades.

Future-Proofing: The very structure of APIs created in adherence to such standards enables future change and integration with other modules and microservices that make up the service itself.

Developer-Friendly Tools: Documented APIs can be upgraded, modified, or expanded because developers are available to help support these needs as the business progresses.

Reduced Technical Debt: Scalability is less sensitive to obsolescence; in big schemes, this means less expenditure on year-on-year IT eventualities that become obsolete.

Open APIs can solve some of the complex problems today and come up with new possibilities for the modern restaurant business. These APIs exist as an obligatory cornerstone for success in a world that is becoming more digital with each passing day, as they touch upon such aspects as operational optimization and customer satisfaction, support growth, and adaptability. They enhance the daily functioning of restaurants, but from the point of supporting complex integrations, providing cyclical security, and creating a foundation for API scaling, they make restaurants future-proof. Whether the business being run is small and is hoping to increase efficiency or whether it is a large chain getting ready for franchise, integrating a scalable API is a wise practice that reaps ROI in all areas and aspects of the business.

Future of APIs in Restaurant Technology
The restaurant business is ready for technological upheaval, and APIs will help determine its course [38]. APIs will be the key to building, connecting, and sustaining innovation in restaurants to respond to customer preferences, shifts, and growing competition. Developments like AI, IoT, and individualized dining systems require powerful, elastic, and flexible API arrangements to support fast growth.

Table 2: The Future in Restaurant Technology

Aspect	Applications	Examples	Benefits
AI and Machine Learning	Personalized Recommendations, Dynamic Pricing	Reusable drink bottle suggestions, menu price changes	Better customer experience, reduced waste
IoT Integration	Smart Kitchens, Inventory Management	Smart ovens, real-time stock updates	Operational efficiency, cost savings
Omnichannel Experiences	Cross-Platform Integration, Contactless Payments	Mobile apps, QR codes, voice orders	Seamless engagement, enhanced convenience
Advanced Analytics	Real-Time Insights, Predictive Analytics	Sales trends, customer feedback	Intelligent decisions, improved satisfaction

Artificial Intelligence and Machine Learning Integration
Machine learning and artificial intelligence are slowly invading the restaurant business, and API will be the key that will unlock the future of the restaurant and food business.

Personalized Recommendations: Consumers can also be offered a more personalized menu based on the data collected through the AI-integrated APIs, improving consumption. For example, a drink bottle purchased as a reusable item could up-

date its current customer on similar home brew accessories previously purchased.

Dynamic Pricing: Many APIs can be used to implement machine learning algorithms and perform dynamic pricing, where menu prices change based on the day, time, or other factors relevant to a specific restaurant location.

Operational Forecasting: Technology advances in the concept of API, which will assist restaurants in forecasting the inventory requirements, labor, and demand period with the intention of cutting waste and increasing operation efficiency.

Internet of Things (IoT) Integration
With restaurants adopting IoT devices, APIs have become crucial for IoT devices to communicate with each other.

Figure 11: Internet of Things in Commercial Kitchens

Smart Kitchens: Some APIs can be used to interface with IoT devices, connecting kitchen equipment and cookery to facilitate operations, check the operation status, and monitor food quality. For instance, temperature can be increased or decreased in a smart oven depending on an API direction in relation to recipes or orders.

Inventory Management: IoT sensors can monitor inventory levels in real-time and communicate to APIs to change the status of stock in all the platforms used to order stocks simultaneously. This helps avoid waste by eliminating extra portions and also helps customers see the right options on the menu.

Energy Efficiency: These APIs are integrated into IOT systems where they can observe energy use and control equipment operation to help restaurants in their cost-cutting and environmental conservation initiatives.

Omnichannel and Contactless Dining Experiences
The concept of omnichannel where clients can engage with restaurants in various ways will be driven by APIs [39].

Cross-Platform Integration: To ensure customers keep the same experience, a consistent and integrated front-end will cover in-house dining systems, mobile apps, delivery platforms, and kiosks.

Contactless Payments and Ordering: APIs will also extend the possibility of enhancing contactless solutions where customers place orders, pay bills, and divide amounts directly through a smartphone or QR code.

Voice and AR Interfaces: Ordering through voice assistants such as Alexa or Siri will be available for consumption through APIs, in addition to AR views that allow customers to see the menu or visually change it to show how their selected meals look.

Advanced Data Analytics and Reporting
APIs will be crucial in incorporating the various enabling analytics tools that restaurants require to make intelligent decisions.

Real-Time Insights: The APIs associated with analytics platforms will generate timely updated information regarding trends in sales, customers' choices, and operational efficiency.

Predictive Analytics: By adopting these necessary predictive models, APIs will allow restaurants to predict the markets, manage employees, and enhance foods.

Customer Feedback: Consumers are able to share feedback through different modes, and APIs will continue to gather such feedback across the channels to ensure that restaurants rectify issues that may concern customers and enhance customer satisfaction.

APIs are set to be even more influential in the further development of restaurant technology as the restaurant industry needs to catch up with the fast pace of the digital era. APIs are set to become a great enabler of the intelligent integration of AI, IoT, omnichannel experiences, and advanced analytics to help restaurants meet demands, run more efficiently, and embrace innovation. APIs remain a cornerstone of the rapidly evolving restaurant environment, faced with stable growth, increasing effectiveness, and extraordinary client experience. Restaurants relying on API solutions will stand a better chance in the future of technological advancement.

Conclusion
The introduction of scalable APIs into restaurants is not merely a technical innovation but a revolution that affects the hospitality sector by changing the way restaurants do business and interact with their clients and the market. This article focuses on the fact that currently scalable APIs are at the heart of eliminating functional constraints, improving performance, and stimulating new ideas in a more saturated and technological program environment. By facilitating the integration of online and in-store processes, guaranteeing real-time information, and providing safe data processing, APIs enable restaurants to address new customer expectations and retain high stability and speed. Scalable APIs solve basic problems like high transaction rates, POS system compatibility, and customer and staff notifications in real-time. These APIs act as a link between online outlets and physical stores, making sure that the inventory is up to date, orders are processed properly, and the customer experience is seamless. Modern architecture ideas, including microservices architecture and programming languages like Go, help APIs tackle growth challenges while minimizing technical debts and the resulting rigidity. Additionally, security measures that are integrated into the system are in the form of encryption of customer and transactional data, authentication, and meeting industry standards of security.

Reflecting the industry's advancement, these scalable APIs will play a much greater role in future innovations. Implementing AI and ML is expected to be the key to providing customized customer experience and predicting operations. The growth of smart kitchens and inventory systems enabled by the Internet of Things will require APIs for communication and control,

thereby enhancing operational accuracy and resource optimization for sustainability. APIs will also enhance the meaning of the omnichannel approach, which in turn will introduce contactless ordering, vocal commands, and augmented reality menus. Such advancements put APIs at the center of technological solutions and support restaurants to meet new clients' demands and shifts in market trends. The experience of SpotOn shows the practical advantages of a well-built, extended API and its ability to grow and develop. This need was met efficiently through the company's API-centered solution that alleviated some of the most tedious issues, including transaction frequency, POS integration, and real-time synchronization, which, in the long run, enabled the provision of enhanced results, customer satisfaction, and scalability. Using a superior event-driven architecture, caching means, and secure communication, SpotOn showed an awesome example of how scalable APIs could improve operational robustness and foster business evolution. The effectiveness of this model demonstrates how restaurants can redesign their operations with a view of mimicking this model's success as a way of implementing long-term strategies.

APIs provide great opportunities to save costs by automating processes, reducing mistakes, and efficiently using resources. Others can accompany them with third-party software, including delivery services and payments, making these apps even more valuable for restaurants that can capture more clients without compromising on quality. These capabilities not only make operations efficient but also prepare restaurants for rapid changes in an industry marked by new trends in technology and customer preferences. The scalability of APIs can no longer be viewed as a technical innovation but as a strategic direction for restaurants with an ambition to succeed in a rapidly digitalizing environment. They act as a springboard to innovation, improved operations, and customer focus, all of which make it easy for firms to adapt to modern environments. This insight simply indicates how scalable APIs open another front and future for restaurant businesses, allow them to address future developments in an ever-competing field, and lay down a solid structure for continuous future growth [40].

References

1. Ritzer G (2011) The McDonaldization of society. Pine Forge Press 6.
2. Gill A (2018) Developing a real-time electronic funds transfer system for credit unions. International Journal of Advanced Research in Engineering and Technology (IJARET) 9: 162-184.
3. Sivakumar K, Li M, Dong B (2014) Service quality: The impact of frequency, timing, proximity, and sequence of failures and delights. Journal of Marketing 78: 41-58.
4. Rambhadjan M, Schutijser A (2010) SURFnet cloud computing solutions. Universiteit van Amsterdam, System and Network Engineering.
5. Banerjee A (2018) Blockchain technology: supply chain insights from ERP. In Advances in computers. Elsevier 111: 69-98
6. DiPietro R (2017) Restaurant and foodservice research: A critical reflection behind and an optimistic look ahead. International Journal of Contemporary Hospitality Management 29: 1203-1234.
7. Meyer C, Schwager A (2007) Understanding customer experience. Harvard business review 85: 116.
8. Nyati S (2018) Revolutionizing LTL carrier operations: A comprehensive analysis of an algorithm-driven pickup and delivery dispatching solution. International Journal of Science and Research (IJSR) 7: 1659-1666.
9. Handfield R, Linton T (2017) The Living supply chain: The evolving imperative of operating in real time. John Wiley & Sons.
10. Mikkelsen BE, Novotny R, Gittelsohn J (2016) Multi-level, multi-component approaches to community based interventions for healthy living—a three case comparison. International journal of environmental research and public health 13: 1023.
11. Chin E, Felt AP, Greenwood K, Wagner D (2011) Analyzing inter-application communication in Android. In Proceedings of the 9th international conference on Mobile systems, applications, and services 239-252.
12. Hossain ST (2018) Impacts of COVID-19 on the agri-food sector: food security policies of Asian productivity organization members.
13. Kumar A (2019) The convergence of predictive analytics in driving business intelligence and enhancing DevOps efficiency. International Journal of Computational Engineering and Management 6: 118-142.
14. Marr B (2019) Artificial intelligence in practice: how 50 successful companies used AI and machine learning to solve problems. John Wiley & Sons.
15. García-Valls M, Cucinotta T, Lu C (2014) Challenges in real-time virtualization and predictable cloud computing. Journal of Systems Architecture 60: 726-740.
16. Jacobson D, Brail G, Woods D (2012) APIs: A strategy guide. O'Reilly Media, Inc.
17. Zutshi A, Grilo A (2019) The emergence of digital platforms: A conceptual platform architecture and impact on industrial engineering. Computers & Industrial Engineering 136: 546-555.
18. He Z, Han G, Cheng TCE, Fan B, Dong J (2019) Evolutionary food quality and location strategies for restaurants in competitive online-to-offline food ordering and delivery markets: An agent-based approach. International Journal of Production Economics 215: 61-72.
19. Burns B (2018) Designing distributed systems: patterns and paradigms for scalable, reliable services. O'Reilly Media Inc.
20. Blair-Goldensohn S, Hannan K, McDonald R, Neylon T, Reis GA, et al. (2008) Building a sentiment summarizer for local service reviews. In WWW workshop on NLP in the information explosion era 14: 339-348.
21. Zhang Z, Wu C, Cheung DW (2013) A survey on cloud interoperability: taxonomies, standards, and practice. Acm sigmetrics Performance Evaluation Review 40: 13-22.
22. Ali MI, Mileo A, Parreira JX, Fischer M, Kolozali S, et al. (2016) Citypulse: Large scale data analytics framework for smart cities.
23. Nyati S (2018) Transforming telematics in fleet management: Innovations in asset tracking, efficiency, and communication. International Journal of Science and Research (IJSR) 7: 1804-1810.
24. Shen W, Hao Q, Mak H, Neelamkavil J, Xie H, et al. (2010) Systems integration and collaboration in architecture, engineering, construction, and facilities management: A review. Advanced engineering informatics 24: 196-207.
25. Kalske M (2017) Transforming monolithic architecture towards microservice architecture. University of Helsinki.
26. Dar U, Krosing H, Mlodgenski J, Roybal K (2015) PostgreSQL server programming. Packt Publishing Ltd.
27. Ullah KW, Ahmed AS, Ylitalo J (2013) Towards building an

automated security compliance tool for the cloud. In 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications 1587-1593.
28. Palmer A (2013) Ebook: Principles of services marketing. McGraw Hill.
29. Siriwardena P (2014) Advanced API Security. Apress: New York, NY, USA.
30. Bondi AB (2000) Characteristics of scalability and their impact on performance. In Proceedings of the 2nd international workshop on Software and performance 195-203.
31. Au YA, Kauffman RJ (2008) The economics of mobile payments: Understanding stakeholder issues for an emerging financial technology application. Electronic commerce research and applications 7: 141-164.
32. Bezemer CP, Zaidman A (2014). Performance optimization of deployed software-as-a-service applications. Journal of Systems and Software 87: 87-103.
33. Evans JA (2019) Digital Learning: Peril or Promise for Our K-12 Students. National Briefing Paper--Speak Up 2018/19. Project Tomorrow.
34. Teinum A (2013) User testing tool: towards a tool for crowd-source-enabled accessibility evaluation of websites (Master's thesis, Universitetet i Agder; University of Agder).
35. Jiang ZM, Hassan AE (2015) A survey on load testing of large-scale software systems. IEEE Transactions on Software Engineering 41: 1091-1118.
36. Bennett C, Sagmiller DV (2014) Unity AI Programming Essentials. Packt Publishing Ltd.
37. Jogaratnam G (2017) The effect of market orientation, entrepreneurial orientation and human capital on positional advantage: Evidence from the restaurant industry. International Journal of Hospitality Management 60: 104-113.
38. Kranz M (2016) Building the internet of things: Implement new business models, disrupt competitors, transform your industry. John Wiley & Sons.
39. Riikkinen M, Saarijärvi H, Sarlin P, Lähteenmäki I (2018) Using artificial intelligence to create value in insurance. International Journal of Bank Marketing 36: 1145-1168.
40. Buhalis D, Harwood T, Bogicevic V, Viglia G, Beldona S, et al. (2019) Technological disruptions in services: lessons from tourism and hospitality. Journal of service management 30: 484-506.