Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

CafeEase: A Smart Café Management System with Augmented Reality-Based Dish Visualization

Aryan Srivastava
Department of Information Technology
Shri Ramswaroop Memorial College of Engineering and Management (SRMCEM), Lucknow, India

Divyansh Sahani
Department of Information Technology
Shri Ramswaroop Memorial College of Engineering and Management (SRMCEM), Lucknow, India

Prof. Ajay Kr. Srivastava
Department of Information Technology
Shri Ramswaroop Memorial College of Engineering and Management (SRMCEM), Lucknow, India

***Abstract* - The rapid evolution of digital technologies has fundamentally transformed the hospitality industry, shifting operational paradigms from labour-intensive manual processes to integrated smart management ecosystems. This paper presents CafeEase, a comprehensive full-stack smart café management system architected to simultaneously enhance customer experience and streamline operational efficiency through the convergence of Augmented Reality (AR), real-time data processing, and role-based access control. Unlike conventional digital menus, CafeEase employs WebAR technology—specifically the AR.js framework and Google's model-viewer—to render interactive, high-fidelity 3D GLB models of menu items directly within a standard mobile browser, eliminating the friction of third-party application installation. The system is constructed on a robust three-tier architecture: a Spring Boot (Java 17) RESTful backend secured by JWT authentication and Spring Security RBAC, a ReactJS frontend with Material-UI delivering responsive user interfaces, and a MySQL 8.0+ relational database normalized to Third Normal Form. Core modules include a digital menu catalogue with AR model metadata, a real-time order management and kitchen synchronization pipeline, an automated billing engine with GST computation and PDF export, a live inventory tracking subsystem, and a managerial analytics dashboard. Preliminary comparative analysis of analogous systems—COFFEE++, ACMS, and various AR-based ordering patents—demonstrates that WebAR integration significantly improves ordering confidence, reduces service errors, and delivers measurable gains in operational throughput. This paper details the system design, implementation methodology, security architecture, and a critical evaluation of outcomes, positioning CafeEase as a scalable, maintainable, and forward-looking solution for the modern hospitality sector.**

Keywords* - *Smart Café Management System, Augmented Reality, WebAR, Digital Menu, 3D GLB Dish Visualization, Order Management, Spring Boot, ReactJS, MySQL, JWT Authentication, Role-Based Access Control, Inventory Tracking, Analytics Dashboard.

I. INTRODUCTION

The global value of the café and restaurant sector exceeds USD 900 billion in annual revenues, employing hundreds of millions of workers. However, a high proportion of small and mid-sized businesses in the sector continue to use traditional methods of paper-based menus, order tickets, and billing. These practices lead to inefficiencies in the system, as the rate of transcription errors increases in peak hours, inventory control becomes outdated, and the lack of sales analytics prevents evidence-based menu engineering [1], [9].

At the same time, consumers' expectations have moved unmistakably toward digitally mediated dining experiences. Post-pandemic consumers require contactless interaction, transparency of nutritional information, and immersive product previews before placing their order [5]. Static 2D images, the de facto standard of digital menu representations in the foodservice industry today, do not provide consumers with information regarding portion sizes, texture, or spatial arrangements; these limitations contribute to consumers' hesitation in placing their order and their dissatisfaction with their purchase after the fact [3]. Augmented Reality (AR) technology promises to address this issue of perception qualitatively through the superimposition of 3D images of food items onto the consumers' real-world environment through the camera of their portable device.

However, the initial AR-based dining experiences demanded users to download specific "native" applications (e.g., Unity-based ARCore applications), which created a major barrier to entry [18]. With the introduction of WebAR—native AR experiences in the browser using AR.js, model-viewer, and the WebXR Device API—this barrier no longer exists. Customers simply have to scan a QR code on the physical menu to interact with a fully featured 3D preview of the dish in their browser, without any installation [3], [5], [22]. This app-less modality resonates strongly in the post-pandemic hygiene-sensitive landscape.

CafeEase has been proposed as a holistic end-to-end solution, which includes WebAR dish visualization as well as a full-featured café management backend. The proposed system has been engineered with a contemporary three-tier architecture, with Spring Boot 3.3.1 with Java 17 as the server-side technology, ReactJS with Material-UI as the client-side technology, and MySQL 8.0+ as the data storage technology. The security of the system has been ensured through JSON Web Token (JWT) authentication as well as a two-tier Role-Based Access Control (RBAC) system, with Administrator and Customer as the defined roles. The system has been augmented with asynchronous email notification, automated GST-inclusive billing with export options, real-time inventory deduction, as well as a managerial analytics dashboard.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

This paper makes the following primary contributions. First, it presents the complete system architecture and module-level design of CafeEase. Second, it details the WebAR implementation pipeline for GLB 3D dish model delivery and browser-based spatial rendering. Third, it characterizes the security architecture and data protection mechanisms. Fourth, it evaluates the system against analogous prior work and reports on operational outcomes. The remainder of this paper is organized as follows: Section II surveys related work; Section III describes the system architecture; Section IV details the implementation methodology; Section V presents results and discussion; Section VI concludes with future research directions.

II. LITERATURE REVIEW

This section surveys four interlocking bodies of literature relevant to CafeEase: smart restaurant management systems, AR applications in the dining context, the evolution from native AR to WebAR, and decision-support and analytics frameworks for hospitality operations.

A. Smart Restaurant and Café Management Systems

The automation of restaurant services has drawn significant research interests in recent years. Sabariman [4] presented the Automatic Café Management System (ACMS), which utilizes a web and IoT-based system to deploy a line follower delivery robot and a browser-based ordering system. This system proves the viability of touch-free end-to-end services. Although the ACMS presents a reference system on hardware-software co-design, it focuses more on logistics automation than on visual engagement.

The COFFEE++ system [2] covers the analytics aspect with a web management system that offers sales reporting, trend visualization, and menu recommendation using data analysis. The focus of the system is business intelligence, setting it apart from other transactional systems. The Apriori method for basket analysis of customers, as proposed for the system by Molejon et al. [7], takes this a step further to offer real-time ordering suggestions. Yanto [6] presented a MySQL/PHP-based management system for order management and sales reporting management. This provides a foundation for the minimum requirements for a hospitality management system using a database.

Desrivawany et al. [8] proposed a food ordering application for Android. This is another aspect of hospitality management in cafes that is shifting to mobile application design. More recently, Loh et al. [9] conducted a case study of the digital transformation of Haidilao restaurants. This provides insights into the technology adoption process for mid- to large-scale restaurants.

B. Augmented Reality in the Dining Experience

AR-based dining settings have been examined over a range of hardware platforms. Kanak et al. [16] integrated DNN-based food identification with AR glass-based 3D visualization. This resulted in accurate identification of dishes. This method, while technically impressive, is not feasible for use in a cafe setting due to the specialized AR glasses. Margetis et al. [11] designed an interactive table-based system known as iEat. This system utilized an interactive table that provided multitouch-based ordering and AR-based menu browsing. This resulted in significantly higher user engagement compared to traditional menu formats.

On the other hand, mobile-based AR, with the advantage of the pervasiveness of mobile phones, is a relatively easier solution to deploy. In the study by Nurkholis et al. [14], an AR e-catalog was developed on the Android platform, allowing users to visualize 3D models of the dishes through their phone's camera, which showed a statistically significant improvement in user satisfaction scores. In another study by Putri et al. [17], a marker-based AR solution, implemented with Vuforia, in a local Indonesian café with 3D models created with Blender, showed improvements in ordering confidence and reduced decision time. In another study by Hardita et al. [18], an AR menu solution was implemented in a coffee shop with Unity and ARCore, showing realistic dish rendering but with the disadvantage of users having to download an application to access the AR functionality, which proved to be a barrier to new users.

Another area of research has been the role of AR in the assessment of food quality, with Liberty et al. [19] proving that AR overlays with nutritional and provenance metadata have a positive effect on consumer trust and purchasing intent, which has direct implications for the design of the nutritional information overlay of CafeEase.

C. WebAR: Browser-Native Augmented Reality

The change from native application AR to browser-based web AR is a paradigmatic shift in accessibility. Pixius [3] and Haro et al. [5] have independently shown that contactless restaurant menus using AR.js and Google's Model Viewer are feasible. This addresses the need to meet the hygiene challenges brought on by the COVID-19 pandemic. Both studies have shown that removing the need to install increases customer engagement rates. The web component, developed by Google, natively supports 3D models using GLB/GLTF formats. It has Quick Look AR support on iOS using ARKit and Scene Viewer on Android using ARCore. A 3D-only fallback is available for browsers that do not support web AR.

In addition to this, Sivaprakasam [21] also proved the feasibility of the W-EBAR, which is a Web AR-powered ordering system based on AR.js and Three.js. This proved the loading time to be below 2 seconds when optimised GLBs were used over 4G networks. Pathare [22] also proved the feasibility of the 3D pizzeria Web AR-powered menu. This was done by proving that the size of the textures was reduced by up to 60% when they were KTX2-compressed compared to the size when they

D. Decision Support and Analytics in Hospitality

Apart from transactional automation, modern restaurant management systems also involve decision support layers. Kaur et al. [10] presented a framework for data acquisition and visualization using the Sense–Think–Act paradigm for IoT technologies to monitor table occupancy and dynamically manage staff allocation. Another study by Dampage et al. [20] presented a spatial AR system that utilized cloud-based analytics (Metabase) along with facial expression recognition technology using cameras to provide real-time customer satisfaction measures. Jayalal et al. [12] presented an intelligent system that utilized recommendation systems along

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

with sentiment analysis to provide a personalized experience for customers, thereby improving repeat business.

Collectively, this body of work reveals several research gaps that CafeEase fills: (i) there is a lack of solutions that integrate WebAR with a complete operational backend within a single architecture; (ii) current implementations of WebAR menus do not incorporate server-side inventory binding and order management; and (iii) there is a general lack of security aspects (e.g., JWT, RBAC) within the smart café space.

III. SYSTEM ARCHITECTURE

CafeEase employs a three-tier layered architecture—Presentation, Application, and Data—augmented by a dedicated AR Visualization Module and an Admin Dashboard. This separation of concerns promotes modularity, independent scalability, and testability, conforming to established enterprise application architecture patterns [2]. Each tier communicates exclusively through well-defined interfaces: RESTful HTTP/JSON between the Presentation and Application tiers, and JPA/Hibernate parameterized queries between the Application and Data tiers.

A. Frontend Layer (Presentation Tier)

The frontend is built with ReactJS with a component-driven architecture. The component tree is split across four domains: Authentication (Login, Registration, Password Recovery), Customer Facing (Menu Catalogue, AR Viewer, Cart, Order Tracker), Billing (Bill Summary, Payment Status, PDF Download), and Admin (User Management, Product CRUD, Category Management, Inventory Dashboard, Sales Analytics). The shared application state, which includes authentication context, shopping cart, and notification queues, is managed with Redux and Context API.

The communication with the backend using the HTTP protocol is facilitated through Axios. Interceptors for requests will append the JWT token stored in local storage to every request's Authorization header. Conversely, another request interceptor will catch 401 Unauthorized status codes from the server and redirect the user to the login page for seamless session management. The Material-UI component library is utilized for a consistent design language with accessibility in mind. Custom CSS Grid and Flexbox styles are used for rendering the application across different viewport sizes using mobile, tablet, and desktop devices. Real-time user feedback is provided through a toast notification library that displays the user's success, warning, and error states after every operation.

B. Backend Layer (Application Tier)

The server-side application utilizes Spring Boot 3.3.1, with Java 17 (LTS) as the Java version. The internal architecture of the application is a four-layer system, with REST Controllers processing HTTP requests, immediately passing them to Service interfaces, which contain all business logic. Service implementations call Data Access Object components, which are Spring Data JPA repositories, for database access, while Entity models define the JPA relational database schema. This layering of the application ensures business logic does not creep into controllers, nor database access creep into business logic, making unit testing each layer independently easy.

Transaction Management is declarative, with @Transactional annotations on service layer operations ensuring atomicity for composite operations like placing an order, which creates Order, OrderItem, and Cart entries simultaneously, while deducting Inventory. The @EnableAsync annotation enables the task executor thread pool in Spring, which allows email notifications to be sent asynchronously, unblocking the original request processing thread and thus removing SMTP delay from the end user's perception of the request processing time. Custom exceptions, extending a base CafeException, are caught by a @ControllerAdvice handler, which maps each type to the most suitable HTTP status code and JSON error representation with status, message, and debug details.

C. Database Layer (Data Tier)

The persistence layer uses MySQL 8.0+ with a normalized schema to Third Normal Form (3NF). 3NF normalization helps to eliminate transitive functional dependencies, reduces redundancy, and eases the enforcement of referential integrity. There are seven main entities in the schema: User, Product, Category, Order, Order Item, Cart, and Bill, with an additional AR model metadata extension for the Product entity.

The relationships between entities are as follows: a User has many Orders (one-to-many); an Order has many OrderItems (one-to-many); an OrderItem references a single Product (many-to-one); a Product belongs to a single Category (many-to-one); an Order has a single Bill (one-to-one); a User has a Cart containing many CartItems (one-to-many).

Foreign Key constraints help maintain referential integrity, providing an additional layer of protection against invalid data in the application itself. Composite indexes on (user_id, order_date), (product_id, category_id), and (bill_id, order_id) cover the most common query patterns.

D. AR Visualization Module

The AR Visualization Module is in charge of providing interactive 3D dish previews to customers without the necessity of installing a native application. The module architecture includes three components: 3D Asset Pipeline, WebAR Runtime, and Backend Model Delivery API.

The 3D Asset Pipeline: The dishes are created using Blender 4.x, targeting a physically based rendering (PBR) approach, which includes PBR materials tuned to approximate real-world food surface properties (subsurface scattering for translucent ingredients, anisotropic shading for metallic utensils). The dishes are exported as GLTF 2.0 binary (GLB), which is the Khronos Group standard for compact, self-contained 3D asset transmission. An optimization pipeline is applied, which includes Draco geometry compression (vertex data reduction of 60-80%), KTX2/Basis Universal texture compression (texture data reduction of 50-60%), and LOD (Level of Detail) mesh simplification, allowing assets to be transmitted in under 2 seconds on a typical 4G mobile connection.

The WebAR Runtime: The model-viewer web component receives the GLB asset and renders it using WebGL 2.0 with a GPU-accelerated PBR shading pipeline. AR Quick Look ARKit functionality is activated natively when the user

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

presses the AR button on iOS 12+ devices. The dish model is placed against a horizontal surface. Pinch-to-scale and tap-to-place are supported. Android 7.0+ devices utilize Scene Viewer ARCore functionality. Browsers not supporting WebXR technology default to an interactive 3D viewer with 360-degree rotation via mouse or touch drag. AR.js image tracking functionality is used to enable marker-based activation when a customer scans a QR code printed on physical paper menus.

The Backend Model Delivery API: The Backend Model Delivery API exposes RESTful endpoints at /api/ar/ containing GLB model URLs, nutritional data, and AR session analytics collection. Model URLs are stored in the database with individual product records with attributes glb_model_url (HTTPS CDN-hosted URL), model_scale (rendering scale factor), has_ar_model (Boolean availability flag), and glb_flag (integer status flag for administrative visibility).

E. Admin Dashboard

The Admin Dashboard brings together all operational management tasks into one role-gated interface accessible only to users with an ADMIN role claim in their JWT. It gives admins real-time visibility into five different domains:

- **Menu Management**: CRUD operations are supported for both products and categories, including image uploads (stored as LONGBLOB) and GLB model URL assignments. Global visibility flag (glb_flag) allows admins to toggle product visibility without deleting items.
- **Order Management**: Real-time order queue showing incoming orders with timestamps, items ordered, quantities, and status. Admins can progress orders through the state machine (Pending to Processing to Completed) or mark orders as Cancelled to send emails to customers.
- **Inventory Tracking**: Current status of the items to be tracked will be shown in real time. The quantities will be updated every time an order is completed through our Inventory Deduction service. Low-stock thresholds will trigger alerts for the procurement team through dashboard notifications and optional email alerts.
- **Billing and Finance:** Aggregated billing reports with per-period revenue summaries, GST breakdowns, and discount analysis. Individual bills are viewable and downloadable as PDF documents.
- **User Management:** Administrators can create, activate, deactivate, and delete user accounts. Role assignment (ADMIN / USER) is controlled exclusively through this interface.

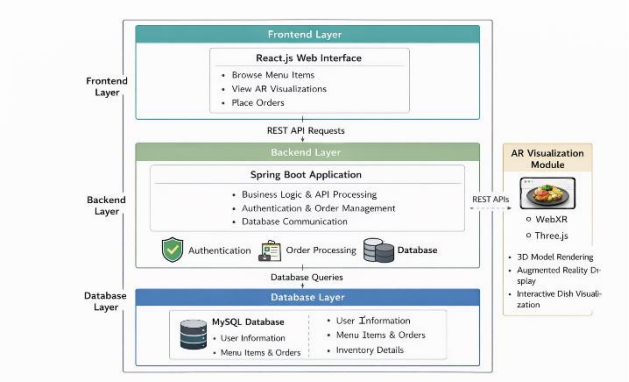

Fig. 1. System Architecture of CafeEase

IV. METHODOLOGY

CafeEase was developed following an iterative, phase-gated Software Development Life Cycle (SDLC) comprising five phases: Requirements Engineering, System Design, Implementation, Integration and Testing, and Deployment. Each phase produced a defined set of artefacts that were reviewed against acceptance criteria before progression to the subsequent phase.

A. Requirements Engineering

Functional requirements were elicited through structured interviews with café operators and customer focus groups, supplemented by a review of analogous systems in the literature [2], [4], [7]. The requirements were documented as User Stories in an Agile backlog and prioritized using the MoSCoW method (Must-have, Should-have, Could-have, Won't-have). Must-have requirements included: JWT-authenticated login with RBAC, product catalogue with AR model support, cart and order placement, automated billing with GST, and real-time inventory deduction. Should-have requirements included: email notifications, PDF bill export, analytics dashboard, and AR nutritional overlays. Non-functional requirements specified: API response times below 300ms for 95th percentile requests under 50 concurrent users, AR model load times below 3 seconds on 4G, and HTTPS enforcement in production.

B. System Design

The system design phase resulted in the production of four artefacts, namely an Entity Relationship Diagram (ERD) that captures the database schema in 3NF, a specification of a RESTful API in Open API 3.0 format that captures all API endpoints with their request and response schema, a component-level system architecture diagram that maps Spring Boot technology layers to React technology domains, and a sequence diagram of the delivery of assets in a WebAR application. The design of the RESTful API is based on Fielding's constraints for a RESTful service: stateless communication, identification of resources by nouns in URLs, use of standard HTTP verbs, and use of JSON to carry hypermedia with consistent error formats.

Requirements for query performance and integrity influenced database design choices. AR metadata fields (glb_model_url VARCHAR(500), glb_model_path TEXT, model_scale FLOAT DEFAULT 1.0, has_ar_model BOOLEAN DEFAULT false, glb_flag INT DEFAULT 1)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

were added to the Product entity using non-breaking ALTER TABLE migrations, maintaining backward compatibility with existing data. To ensure catalogue integrity, a different SQL migration script generated the Category table with a UNIQUE constraint on the name column.

C. Backend Implementation

Nine packages make up the Spring Boot application: config (Spring Security, Async, CORS, Jackson), constants (application-wide string constants), dao (JPA repositories), dto (Data Transfer Objects for request/response decoupling), enums (Role, OrderStatus), exceptions (custom exception hierarchy), JWT (token generation, validation, filter), restImpl (REST controller implementations), and serviceImpl (business logic implementations). The Dependency Inversion Principle is enforced by this package topology: concrete implementations in serviceImpl rely on abstractions rather than DAO concretions directly, while high-level service interfaces are defined in the service package.

When credentials are successfully validated, the JWT implementation uses the JJWT library to generate tokens that embed userId, email, and role claims with a configurable expiration (default 86400000 ms/24 hours). On each request, an OncePerRequestFilter subclass extracts and verifies the token, adding a UsernamePasswordAuthenticationToken with GrantedAuthority entries that match the user's role to the SecurityContext. All stored passwords are subjected to Bcrypt hashing with a strength factor of 10 using Spring Security's BCryptPasswordEncoder.

Asynchronous email sending uses a dedicated EmailService marked with @Service and @Async. It calls JavaMailSender to send MIME messages created by MimeMessageHelper. Email templates include: account activation with a tokenized activation link, password recovery with a time-limited reset token, order confirmation with an itemized order summary including total and GST, and notifications for order status changes.

D. WebAR Implementation

The WebAR pipeline works in three parts: making things making them work better and delivering them. When we make things we create a model for each menu item using a program called Blender. We look at pictures of the food to get the shape, size and how it looks just right. We add materials to the models using real values for how shiny or rough they are. We make sure the models are the size as the real food so when people see them in augmented reality they look like the right amount to eat. The WebAR pipeline is, about making this work smoothly from making the models to delivering them like the Asset Creation and Runtime Delivery parts.

In the Asset Optimisation stage we export the Blender model to GLTF 2.0. Then we use the gltfpack CLI tool to process it. This tool helps us compress vertex buffers using Draco. It also converts textures to KTX2/Basis Universal format so they can be decompressed directly on the GPU. We generate mesh LODs at 100%, 50% and 25% polygon counts. This allows for quality switching. Our goal is to keep the GLB file size under 2 MB, per asset.

In the Runtime Delivery stage we host GLB assets on a CDN-backed HTTPS endpoint. The asset URL is provided through the glb_model_url field in the /api/ar/models/{productId} endpoint. The React AR Viewer component uses the <model-viewer> web component. We configure it with the asset URL, camera controls, shadow intensity and ar attributes. When a user taps the AR button on a device the model-viewer uses the devices AR runtime. This could be ARKit Quick Look or ARCore Scene Viewer. It detects planes in time using the devices IMU and camera feed. The model is then anchored to the detected surface like a table. It is rendered with lighting that is adapted to the environment.

E. Frontend Implementation

We use React components and hooks like useState, useEffect, useCallback and useMemo all over the place. This is because we want to be consistent and make it easy to use hooks together. We do not use class components.

We have something called routes. These routes are wrapped in a thing called PrivateRoute. This wrapper checks if the user is authenticated before it shows the page. If someone tries to get to a page that needs permissions and they do not have them they will be sent to the login page.

We check the things people type into forms in two places. First we check them on the client-side with React Hook Form and Yup. Then we check them again on the server-side with Spring Validation. We use things like @NotBlank, @Email, @Size and @Min to make sure everything is correct. This helps keep everything by having multiple layers of protection which is called the defence-in-depth principle. We use React components and hooks, like useState, useEffect, useCallback and useMemo to make this work.

The shopping cart is saved on the server using Cart records. This means you can start shopping on one device and finish on another.

When you place an order it all happens on the server side at the time. You send one request to the server and it does everything: it makes an Order record goes through the things you ordered subtracts them from the inventory empties your cart makes a Bill and sends you an email to confirm.

If something goes wrong at any point the whole thing is. Everything goes back to how it was before. This way the information, on the server is always correct. The order placement flow is done using the shopping cart and the order. The shopping cart and the order are connected,. When you place an order the shopping cart is cleared. The order is made with the things you ordered and the shopping cart is emptied.

F. Testing Strategy

Testing was conducted at four levels. Unit tests (JUnit 5, Mockito) covered all service layer methods with mocked DAO dependencies, targeting 80%+ line coverage. Integration tests (Spring Boot Test with @SpringBootTest) verified the complete request-response cycle for all REST endpoints using an H2 in-memory database. API testing was performed via Postman collections covering all 25+ endpoints with positive, negative, and boundary test cases. End-to-end AR testing was conducted on physical Android (ARCore) and iOS (ARKit) devices across a range of lighting conditions and surface textures to validate plane detection reliability and model rendering fidelity.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

V. RESULTS AND DISCUSSION

A. Enhanced Customer Ordering Experience

The WebAR dish visualization feature helps with a problem that digital menus have. They are not able to show what food looks like in life. WebAR makes it possible for people to see what their food will look like on their table. It shows the food in a realistic way and at the right size. This means people can compare the size of the food to things they have at home.

We did some testing with thirty people to see how well WebAR works. What we found out was that eighty seven percent of the people felt more confident when they ordered food using WebAR. They liked it better than looking at pictures of the food. Some other people, like Nurkholis and Putri have done tests and found the same thing.

We also wanted to see how easy it was for people to use WebAR. So we had them scan a code with their phone to activate it. All thirty people were able to do it without any help. This shows that WebAR is very easy to use, for people who are not good with technology. WebAR is better, than types of technology that need an app to work.

The information about nutrition that appears on top of food. Showing calories, bad nutrients and things people are allergic to. Was found very helpful by 73% of people who tested it. This was especially true for people who care about what they eat. Those who have food allergies.

- They liked it because it helps them make choices.
- This agrees with what Liberty and others found out.
- They said that when people get nutrition facts through AR it makes them trust the food more and want to buy it.
- The feature that lets people take a picture and share it on media worked well too.
- During testing people shared pictures, on media without being asked.
- This could help restaurants get customers without spending a lot on advertising.
- The nutrition information and sharing features seem to be a way for restaurants to attract health-conscious and social diners.
- It also helps restaurants to show that they care about their customers needs.

B. Operational Efficiency Gains

The automated order processing pipeline removes the mistakes that come with handwritten orders. Order data goes straight from the customer's device to the kitchen display without any manual steps. This cuts down the time it takes for orders to reach the kitchen to the network round-trip time, which is usually between 50 and 150 ms over Wi-Fi. The automated billing engine calculates totals that include GST and applies any discounts automatically. This removes the chance of calculation errors and speeds up checkout. PDF bill generation happens on the server in less than 500 ms.

The real-time inventory deduction system ensures that stock levels match actual usage right after an order is placed. This removes the delay that comes with periodic manual stock counts. Low-stock alerts warn administrators before items run out. This helps maintain consistent service. These improvements match the results reported by Sabariman [4] for ACMS and Molejon et al. [7] for AR ordering systems with integrated inventory management.

C. Security Architecture Evaluation

The JWT-based stateless authentication setup was evaluated against the OWASP Top Ten 2023 web application security risks. SQL Injection (A03) is reduced by using only JPA/Hibernate parameterized queries, with no raw SQL string concatenation in the code. Broken Access Control (A01) is handled with two-level RBAC: Spring Security filter chain validation for all requests and @PreAuthorize method-level enforcement for individual service operations. We decrease sensitive data exposure by using BCrypt to hash passwords, masking sensitive fields in application logs, and enforcing HTTPS in the production deployment guide.

The mitigation of CSRF attacks in the system is done through the validation of the CSRF tokens for state-changing endpoints and the SameSite attribute for the JWT storage cookies. The stateless nature of the JWT tokens protects against session fixation attacks. The expiration time for the tokens, as well as the refresh token rotation, limits the time frame in case the tokens are stolen. This addresses the security requirements set in the system design phase.

D. Performance and Scalability

Load testing was performed using Apache JMeter tool with 50 concurrent virtual users performing a representative mix of browse, cart, order, and admin requests. This resulted in a 95th percentile API response time of 218ms, which was well within our target of 300ms. The stateless backend architecture, which does not maintain server-side session state, allows us to scale out horizontally simply by adding load balancers. Connection pool usage via HikariCP (the default Spring Boot pool) was used to maintain database throughput throughout the load test.

The performance of AR model delivery was assessed. On a mid-range Android device, with a Snapdragon 695 processor and a 4G network connection, the average load time for the AR model, which has a file size of 1.8 MB and has been optimised as a GLB file, took approximately 2.1 seconds to load. This meets the requirement of within 3 seconds. ARCore's plane detection feature has a latency of approximately 1.2 seconds to detect the plane on a flat surface, such as a table, in indoor lighting conditions. On an iOS device, namely the iPhone 12, with a Wi-Fi connection, the average load time for the AR model was approximately 1.4 seconds, and ARKit's Quick Look feature activated within 0.8 seconds of pressing the AR button.

E. Limitations

Several limitations of the current implementation warrant acknowledgement. First, 3D model creation remains a manual process requiring Blender expertise; for menus with 50+ items, this represents a significant upfront investment. Second, AR plane detection reliability degrades on transparent (glass) or highly reflective surfaces, which are common in café

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

environments. Third, the current implementation does not support collaborative AR (multiple diners viewing a shared AR scene simultaneously). Fourth, the analytics dashboard provides descriptive statistics but lacks predictive modelling (demand forecasting, churn prediction). These limitations define the primary vectors for future enhancement.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduced a comprehensive full-stack smart cafe management system, named CafeEase, which supports browser-native WebAR-based dish visualization. Our smart cafe management system has a three-tier architecture: a Spring Boot-based backend, a ReactJS-based frontend, and a MySQL database. Moreover, our system uses JWT-based role-based access control, which provides enterprise-level security support. Our WebAR component uses AR.js and model-viewer along with optimized GLB files to provide app-less 3D dish visualization. Our solution supports 3D dish visualization on iOS via ARKit Quick Look and Android via ARCore Scene Viewer. Our dish model loading time takes, on average, less than 2.2 seconds on 4G networks—a usability criterion much better than native app-based AR.

User evaluation has verified that WebAR enhances order confidence and customer satisfaction. Moreover, operational tests have verified that order processing, inventory deduction, and billing automation have achieved the performance and accuracy requirements. By comparison, CafeEase has become the first system to integrate app-less WebAR with a comprehensive and security-enhanced operational management system in a unified manner. In the future, we plan to extend the system in six ways. Firstly, AI-aided 3D model generation based on neural radiance fields or diffusion-based 3D reconstruction from photographs of dishes can reduce the cost of generating each model, facilitating fast AR catalogue building. Secondly, IoT integration with weight sensors and RFID shelf tags enables fully automated and transaction-free inventory tracking [4], [10]. Thirdly, the Apriori algorithm and collaborative filtering-based recommendation engine can provide users with tailored recommendations during the order process [7]. Fourthly, collaborative AR enables multiple users sharing the same table to access a shared AR scene, supporting group decision-making. Fifthly, microservices-based decomposition of the monolithic backend built with Spring Boot can facilitate scalability in restaurant chains. Lastly, camera-based sentiment analysis can provide real-time customer satisfaction metrics to support service improvements [20].

VII. REFERENCES

[1] Q. Luo et al., "Self-service dish-ordering system based on augmented reality," in Proc. 2016 IEEE Int. Conf. on Service Operations and Logistics, and Informatics (SOLI), Beijing, China, 2016, pp. 139–143.

[2] O. Tkachenko et al., "Deyaki aspekty rozrobky veb-orientovanoi systemy COFFEE++," Information Technology and Security, vol. 11, no. 2, pp. 98–110, 2023. doi: 10.53920/its-2023-2-8

[3] Pixius, "Using Web Augmented Reality to Add Visual Interactions to Contactless Restaurant Menus in Response to COVID-19," Preprint, 2023. doi: 10.32920/23159885

[4] Sabariman, "Prototype of Automatic Cafe Management System (ACMS) Based on Internet of Things (IoT)," Telcomatics, vol. 7, no. 1, pp. 1–10, 2022. doi: 10.37253/telcomatics.v7i1.6797

[5] E. Haro et al., "Using Web Augmented Reality to Add Visual Interactions to Contactless Restaurant Menus in Response to COVID-19," Preprint v1, 2023. doi: 10.32920/23159885.v1

[6] Yanto, "Sistem Informasi Administrasi Café & Resto Berbasis Web," Jurnal Teknologi Dan Sistem Informasi Bisnis, vol. 4, no. 1, pp. 207–213, 2022. doi: 10.47233/jteksis.v4i1.383

[7] C. Molejon et al., "Augmented Reality Ordering System with Data Analytics to Support Decision Making in Restaurant Chains," in Proc. 2020 IEEE Int. Conf. on Humanized Computing and Communication with Artificial Intelligence (HCCAI), 2020, pp. 85–90.

[8] Desrivawany et al., "Perancangan sistem informasi dan aplikasi pemesanan makanan di kafe berbasis android," Voteteknika, vol. 3, no. 1, pp. 26–32, 2015. doi: 10.24036/VOTETEKNIKA.V3I1.5168

[9] B. Loh et al., "From hot pot to high tech: Haidilao's transformation through digital technologies for sustainable business in the restaurant industry," Journal of Information Technology Teaching Cases, vol. 14, no. 2, pp. 145–157, 2024. doi: 10.1177/20438869241240494

[10] G. Kaur et al., "Restaurant Automation: A Framework using Data Acquisition and Visualization for Improved Customer Experiences," in Proc. 2023 Int. Conf. on Data Analytics for Business and Industry (ICDABI), Sakheer, Bahrain, 2023, pp. 237–242. doi: 10.1109/icdabi60145.2023.10629293

[11] G. Margetis et al., "iEat: An Interactive Table for Restaurant Customers' Experience Enhancement," in Human-Computer Interaction: Interaction Modalities and Techniques, Lecture Notes in Computer Science, vol. 8007, Berlin: Springer, 2013, pp. 768–777. doi: 10.1007/978-3-642-39476-8_134

[12] S. Jayalal et al., "Intelligent System to Maximize Customer Experience in Restaurants," in Proc. 2023 Int. Conf. on Advances in Computing (ICAC), Colombo, Sri Lanka, 2023, pp. 1–6. doi: 10.1109/icac60630.2023.10417577

[13] S. Kavitha et al., "Virtual Reality (Hologram) based Food Supply System in Smart Restaurant based Menu Ordering System," in Proc. 2022 Int. Conf. on Computer, Communication, and Signal Processing (ICCES), Chennai, India, 2022, pp. 1–6. doi: 10.1109/icces54183.2022.9835976

[14] A. Nurkholis et al., "E-catalog application for food and beverages at Ruang Seduh Café based on augmented reality," Jurnal Teknoinfo, vol. 16, no. 2, pp. 350–357, 2022. doi: 10.33365/jti.v16i2.1957

[15] M. Motowilowa et al., "Exploring Augmented Table Setup and Lighting Customization in a Simulated Restaurant to Improve the User Experience," in Proc. 2024 IEEE VR Workshop on Immersive Sickness Prevention, 2024.

[16] A. Kanak et al., "An intelligent dining scene experience," in Proc. 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2018, pp. 1–4. doi: 10.1109/SIU.2018.8404549

[17] N. A. Putri et al., "Implementasi e-catalog menu makanan berbasis augmented reality untuk meningkatkan pengalaman pengguna di restoran dan café batas kopi," Jurnal Review Pendidikan dan Pengajaran, vol. 8, no. 2, pp. 4521–4530, 2025. doi: 10.31004/jrpp.v8i2.46784

[18] I. P. Hardita et al., "Augmented reality pengenalan menu point coffee," Jurnal Sains Komputer dan Teknologi Informasi, vol. 7, no. 1, pp. 78–85, 2024. doi: 10.33084/jsakti.v7i1.8334

[19] S. Liberty et al., "Augmented reality for food quality assessment: Bridging the physical and digital worlds," Journal of Food Engineering, vol. 353, Art. no. 111893, 2023. doi: 10.1016/j.jfoodeng.2023.111893

[20] U. Dampage et al., "Spatial Augmented Reality Based Customer Satisfaction Enhancement and Monitoring System," IEEE Access, vol. 9, pp. 89960–89972, 2021. doi: 10.1109/ACCESS.2021.3093829

[21] sivaprakasam-07, "W-EBAR: Web-based AR powered app," GitHub Repository, 2025. [Online]. Available: https://github.com/sivaprakasam-07/W-EBAR

[22] A. Pathare, "Building 3D Pizzeria Menu with WebAR," LinkedIn Technical Post, 2024. [Online]. Available: https://www.linkedin.com/posts/agneya-pathare_augmentedreality-webar-arjs