**Technology Stack Documentation – QuickBite Smart Food Recommendation and Delivery System**

**Project Design Phase-II**

**1. Technical Architecture Overview**

QuickBite utilizes a modern, cloud-based architecture that supports secure, scalable, and responsive food recommendation and ordering services. The architecture facilitates a seamless user experience for three distinct roles: Customers, Restaurant Partners, and Platform Admins. Built with scalability and modularity in mind, the stack is engineered to efficiently handle user authentication, data processing, and personalized recommendation logic.

**2. Architectural Diagram Description**

The high-level architecture encompasses:

* **Frontend:** Accessible through web applications, providing interactive dashboards for all user roles.
* **API Gateway:** Manages and authenticates all external requests, routing them to appropriate backend services.
* **Backend Services:** Includes RESTful APIs for order management, user profiles, and recommendation logic, all built using Node.js and Express.js.
* **Database Layer:** Centralized, cloud-hosted MongoDB Atlas for persistent data storage and retrieval.
* **Authentication:** Secure, role-based access control leveraging JWT tokens and middleware.
* **Deployment:** Cloud-based (AWS or similar) infrastructure supporting CI/CD for rapid feature delivery and updates.
* **Monitoring & Logging:** Integrated tools for health monitoring, error tracking, and performance analytics.

**3. Table 1: Components & Technologies**

|  |  |
| --- | --- |
| Component | Technology/Service |
| Frontend UI | React.js, TailwindCSS, Vanilla CSS |
| Backend API | Node.js, Express.js |
| Database | MongoDB Atlas (Cloud-managed NoSQL) |
| API Gateway | Express.js Routing, Nginx (optional) |
| Authentication | JWT, Role-based Middleware |
| Version Control | Git, GitHub |
| Deployment/Hosting | AWS EC2, AWS Elastic Beanstalk, or similar cloud services |
| Continuous Integration | GitHub Actions, Jenkins |
| Monitoring | CloudWatch, Loggly, or similar |
| Data Processing | Internal Node.js modules for dietary and recommendation logic |

**4. Table 2: Application Characteristics**

|  |  |
| --- | --- |
| Characteristic | Details |
| **Accessibility** | Responsive across devices, WCAG-compliant interfaces |
| **Security** | Encrypted endpoints, secure cloud protocols, JWT access |
| **Scalability** | Cloud-native components enable horizontal and vertical scaling |
| **Maintainability** | Modular microservice approach, easy CI/CD pipeline integrations |
| **Performance** | Optimized APIs, database indexing, low-latency hosting |
| **User Roles** | Customers, Restaurant Partners, Platform Admins (role-based access) |
| **Personalization** | AI-powered recommendation engine, customizable dietary filters |
| **Data Reliability** | Daily cloud backups, real-time data consistency |
| **Monitoring** | Automated log analysis and uptime health checks |

**5. References**

* [C4 Model for Visualizing Software Architecture](https://c4model.com/)
* [IBM Cloud – AI-Powered Back-End Patterns](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)
* [AWS Cloud Architecture Center](https://aws.amazon.com/architecture)
* [Drawing Useful Technical Architecture Diagrams](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)

This document provides a comprehensive summary of the technologies, architecture, and characteristics that form the foundation of the QuickBite platform for robust, scalable, and user-friendly food ordering and recommendation services.

⁂