Project Design Phase-II
Technology Stack (Architecture & Stack)

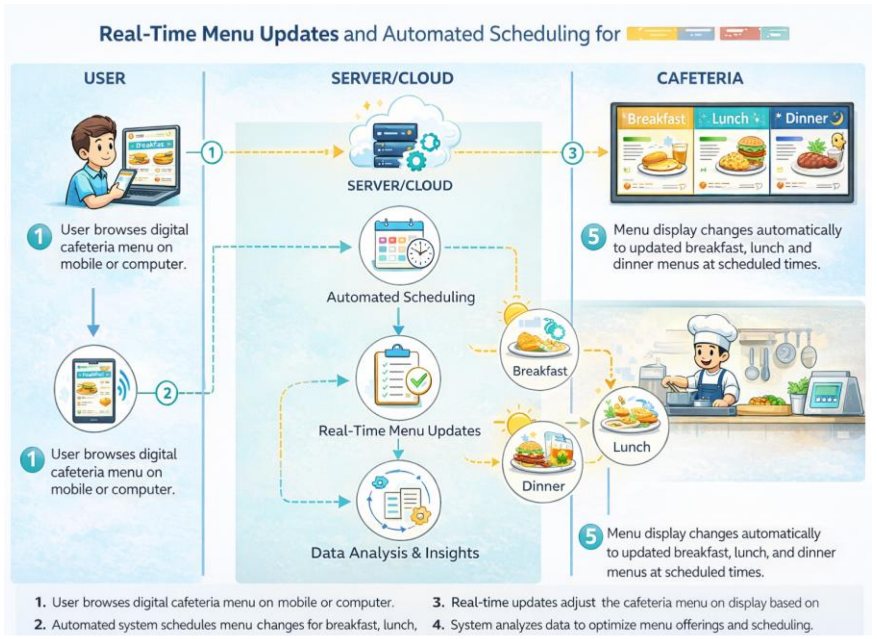
Date	19-02-2026
Team ID	LTVIP2026TMIDS66321
Project Name	Cafeteria Menu Display
Maximum Marks	4 Marks

Technical Architecture:

To provide a dynamic, real-time digital display of menu items, pricing, and availability for cafeteria patrons.

Example: real-time menu updates and automated scheduling for breakfast, lunch and dinner transitions.

Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/


Real-Time Menu Updates and Automated Scheduling for
USER
SERVER/CLOUD
CAFETERIA
Breakfast
Lunch
Dinner
1 User browses digital cafeteria menu on mobile or computer.
1 User browses digital cafeteria menu on mobile or computer.
1
2
3
SERVER/CLOUD
Automated Scheduling
Real-Time Menu Updates
Data Analysis & Insights
5 Menu display changes automatically to updated breakfast, lunch and dinner menus at scheduled times.
5 Menu display changes automatically to updated breakfast, lunch, and dinner menus at scheduled times.
1. User browses digital cafeteria menu on mobile or computer.
2. Automated system schedules menu changes for breakfast, lunch,
3. Real-time updates adjust the cafeteria menu on display based on
4. System analyzes data to optimize menu offerings and scheduling.

Guidelines:

- map out the logic for CMS, menu item scheduling and real-time pricing updates.
- Media players connected to physical LED/LCD displays in cafeteria.
- A cloud-based backend or CMS for remate menu management and content hosting.
- Include APIs for POS integration to automatically sync prices or hide sold-out items.
- Identify storage for high-resolution food imagery, menu databases and digital assets in a Content Delivery Network.
- If applicable, indicate interfaces for recommendation engines that suggest daily specials based on the popular trends or inventory levels.

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Digital signage display for customers and a dashboard for cafeteria staff to update menus.	HTML, CSS3, React Js/Vue Js etc.
2.	Application Logic-1	Core logic for menu management, price updates, and scheduling	Node Js/ Python
3.	Application Logic-2	Real-time synchronization service to ensure all display screens update instantly when a change is made.	WebSockets/Socket.io
4.	Application Logic-3	Image processing service to optimize and display food item photos on the menu.	Cloudinary API/Sharp
5.	Database	Stores menu items, descriptions, prices, nutritional info, and availability status.	PostgreSQL/MongoDB
6.	Cloud Database	Remote backup and synchronization for multi-location cafeteria management.	Firebase real-time database/AWS DB
7.	File Storage	Storage for high-quality food images and promotional videos shown on the displays.	AWS S3/ Google Cloud Storage
8.	External API-1	Integration with a payment gateway if the display includes a QR code for self-checkout.	Stripe API/ Razorpay API
9.	External API-2	Weather or news API to display localized content alongside the menu to engage customers.	OpenWeatherMap API
10.	Machine Learning Model	Recommendation engine to suggest “daily specials” based on historical sales data or time of day.	TensorFlow.js / Scikit-learn
11.	Infrastructure (Server / Cloud)	Hosting for the backend API and the web-based display interface.	AWS EC2/ Heroku / Vercel

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used to build the responsive menu interface and backend.	**Frontend**: React.js or Vue.js with boot strap for responsive design. **Backend**: Node.js or Django
2.	Security Implementations	List all the security / access controls for menu management	**Authentication**: JWT or IAM for role-based access. **Network**: Secure Wi-Fi with firewalls and HTTPS/TLS for data encryption.
3.	Scalable Architecture	Justify the scalability for handling multiple display screens across different cafeteria sections.	**3-Tier Architecture**: Independent presentation, logic , and data tiers. **Microservices**: Separate service for real-time menu updates.
4.	Availability	Justify how the menu stays live during high-traffic lunch hours or server issues.	**Load balancers**: Nginx or AWS ELB to distribute requests across multiple servers. **Offline mode**: Local caching on displays to show the last known menu if the internet fails.
5.	Performance	Design consideration to ensure menu images and prices load instantly on all screens.	**Caching**: Redis for fast retrieval of the current daily menu. **CDN**: Using a Content Delivery Network to serve high-resolution food images with low latency.

References:

https://c4model.com/

https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/

https://www.ibm.com/cloud/architecture

https://aws.amazon.com/architecture

https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d