Bistro 92 - System Features & Design Principles (Quick Fixes)

Q1: List three essential features Bistro 92's system needs for customer satisfaction and efficient order processing.

Answer:

- 1) User-friendly interface that contains User Manual inside:
  Easy browsing of food items including descriptions, images, prices, ratings, and offers on the OLED display. In each step, the display should show how to navigate, how to place orders, how to cancel orders, and how to perform each action.
- 2) Concurrent Order processing:
  This feature helps customers place orders without any delay, ensuring faster service and better customer experience.
- 3) Real-time Order Tracking:
  Customers can see order status updates on the device, such as Preparing, Waiting List, Number of the order, Ready, and Served.

Q2: Describe two design principles to make the smart pad interface intuitive for all users, including tech novices.

Answer:

- **Consistent and Feedback Design:**
  - Consistency: The smart pad's four buttons should have fixed roles throughout the interface. The screen layout should remain uniform, with menu items listed at the top and instructions always displayed at the bottom.

  - Feedback: When a customer interacts with the smart pad, the system should respond instantly. For instance: Selecting 'Spring Rolls' updates the screen to show 'Spring Rolls x1 - $5.99' in the cart. After placing an order, a message like 'Order Placed! Table #5 - Order #123' appears immediately.

- **Minimalist Aesthetic Design:**
  - Use of fixed icons to help customers understand what to do next. A simple and clean interface reduces confusion and enhances usability, especially for customers who may not be tech-savvy. Use large buttons, clear icons, and easily readable fonts.

Q3: Identify three potential security vulnerabilities in Bistro 92's system and suggest one solution for each.

Answer:

1) Unauthorized Access to the Admin Dashboard, Data Theft, Data Manipulation:

Solution:
- *Role-Based Access Control (RBAC):*

Implement Role-Based Access Control (RBAC) to restrict access to sensitive parts of the system based on user roles. Ensure that only authorized personnel (e.g., managers, staff) can access the admin dashboard and related sensitive data.

- *Multi-factor Authentication (MFA):*

Use MFA for admin login to add an extra layer of security.

- *Encryption of Sensitive Data:*

Encrypt sensitive data both in transit and at rest to protect against unauthorized access.

2) Order Tampering (Manipulation of Orders):

Solution:
- *Secure Communication Channels (Encrypted Transmission):*

Use SSL/TLS encryption to ensure that orders are transmitted securely between the smart pad and the cloud system.

- *Unique Order IDs and Audit Logs*:

Assign unique order IDs to each transaction and implement audit logs to track every modification made to an order. This will allow you to trace any changes to orders and hold individuals accountable for tampering.

3) Making the System Unresponsive by DDoS, SQL Injection:

Solution:
- *Cloud-Based DDoS Protection:*

Use a cloud-based DDoS protection service such as Cloudflare, AWS Shield, or Google Cloud Armor to detect and mitigate DDoS attacks.

- *Web Application Firewall (WAF):* Implement a WAF with DDoS protection to filter malicious traffic and prevent system overloads.
- *SQL Injection Prevention:*

In SQL queries like `SELECT * FROM orders WHERE table_id = $`, the `$` is a placeholder, and the actual value is supplied through safe parameterized methods, not directly embedded in the query.

- *Input Validation and Sanitization:*

Validate and sanitize all user inputs to ensure they conform to expected formats. Reject any input that does not match the expected type or format.

Q4: Explain two strategies to keep Bistro 92's system responsive and stable during peak hours.

Answer:

1) Load Balancing

Relevance to Peak Hours:

During peak hours, the Bistro 92 system may experience heavy traffic due to high customer demand. If there is only one server handling all the requests, it could quickly become overloaded, resulting in slower response times or even downtime. Load balancing distributes incoming traffic across multiple servers, ensuring that no single server is overwhelmed, which is crucial during periods of high demand.

Implementation for Bistro 92:

To ensure the system can handle spikes in traffic, Bistro 92 can use AWS Elastic Load Balancing in combination with Node.js servers. Elastic Load Balancing (ELB) automatically distributes incoming web traffic across multiple EC2 instances running Node.js servers. As traffic increases during peak hours, ELB can automatically scale up the number of instances to ensure that the system remains responsive. Once the demand decreases, ELB scales down the number of instances to save costs while maintaining performance.

The use of Elastic Load Balancing helps ensure:

- Improved availability and fault tolerance.
- No single server bears the full burden of customer traffic.
- Automatic scaling, which allows the system to adapt to sudden surges in demand without manual intervention.

2) Caching Frequently Accessed Data

Relevance to Peak Hours:

During busy periods, many customers may access the same data repeatedly, such as browsing the menu or checking the status of their order. If the system constantly queries the database for this frequently requested data, it can create a heavy load on the backend and significantly slow down the response time. By caching frequently accessed data, Bistro

92 can reduce the number of database queries, improving the overall speed and responsiveness of the system.

Implementation for Bistro 92:
Bistro 92 can implement a caching system using Redis, an in-memory data store known for its high performance. This would allow frequently accessed data such as menu details, order statuses, or even customer information to be stored in memory. When a customer requests this data again, the system can retrieve it from the cache instead of querying the database, which is much faster.

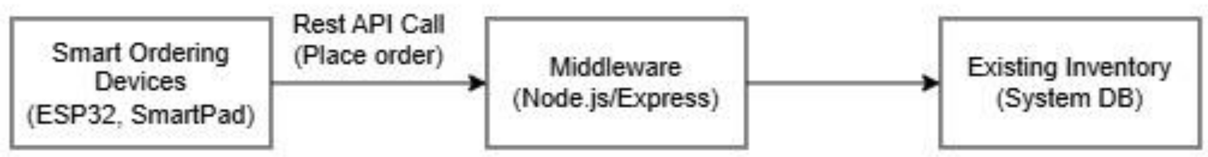
Here's how the caching process would work:
- Menu and order data can be cached in Redis, where it can be quickly retrieved by the smart pads or admin dashboard.
- Redis can be set to automatically refresh cached data at regular intervals or whenever there's a significant update to the menu or orders.
- If data is not found in the cache (a cache miss), the system queries the database and then caches the result for future requests.

By caching data:
- Database load is reduced, allowing for faster response times.
- Faster user experience during peak hours, as customers don't need to wait for the backend to fetch data from the database each time.
- Scalability is improved, especially when dealing with a high volume of repetitive queries.

Q5: Describe one method to integrate the existing inventory system with Bistro 92's new system without disrupting operations.

Answer:

One effective method to integrate the existing inventory system with Bistro 92's new smart ordering system is to use a middleware API layer that acts as a bridge between the smart ordering interface and the inventory database.


Smart Ordering Devices (ESP32, SmartPad)
Rest API Call (Place order)
Middleware (Node.js/Express)
Existing Inventory (System DB)

Flow Explained:

1. Customer places an order on the smart pad.
2. The ESP32 sends the order via REST API to the Middleware/API layer.
3. The middleware parses the order and sends it to the kitchen/dashboard.
4. The inventory Sync API updates stock in the existing inventory system using read-only access, views.
5. A confirmation is sent back to the smart pad and dashboard.

Benefits of this Approach:

- No direct edits to the legacy inventory system.

- Supports fallback caching if inventory DB is unreachable.

- Loose coupling: each component works independently and can be upgraded later.

Why it avoids disruption:

- No changes are made directly to the existing system.
- Ensures both systems remain loosely coupled but data remains synchronized.
- Downtime is minimized through asynchronous updates and caching.