**Artifact 1.3.2 Systemic Textual Analysis (Level 2)**

\*Note: On level 2 analysis only Must and Should functional requirements are included.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Non-Functional Implementation Requirements** | **Functional Requirements** | **Non-Functional Performance Requirements** | **Technical Requirements** | **MoSCoW** | **Comments** |
| Create a “LOG IN” form using HTML, CSS. Add fields for username and password. Implement a PHP script to query the database and validate credentials. | 1.1 Credential validation and logging in (on log in form). | Must validate within 2 seconds. | Implement credential validation using PHP and MySQL with prepared statements to prevent SQL injection. Hash all passwords with the bcrypt algorithm. | M | Ensures role-based access by checking the "role" of the successfully logged-in user and ensuring secure authentication. |
| Add a "Forgot Password" link on LOG IN page using HTML. Backend PHP script sends a password reset link to the registered email. | 1.2 Provide password reset functionality in case the user forgot it (on log in form). | Must execute password reset within 5 seconds. | Use SMTP protocol with PHPMailer to send secure password reset links valid for 15 minutes. | M | Enhances account recovery options for users. |
| Create a “CREATE A CUSTOMER ACCOUNT” form using HTML, CSS. Link it to “LOG IN” page via a hyperlink. Include fields for account details (e.g. Name, Surname, Birthdate, Email, Telephone) Use PHP to update the database with SQL INSERT queries | 1.3 Allow users to create customer accounts (on log in form) | Must create the account within 5 seconds | Use SQL INSERT queries to store customer account details in the database. | M | Enables non existing users to create customer accounts and access the system. \*(Note system automatically assign customer role for account creations through this page) |
| Create an “ACCOUNT MANAGEMENT” form using HTML, CSS. Add fields for account details (e.g. Name, Surname, Birthdate, Email, Telephone) | 1.4 Allow users to view and edit personal information (on account management form). | Must save updates within 3 seconds. | Enable updates using PHP and SQL UPDATE queries, limiting modifiable fields. | M | Enables easy modification of user account details. \*(Fields like Name, Surname and birthday are greyed out and can not be changed) |
| Add a "Change Password" link on ACCOUNT MANAGEMENT page using HTML. Backend PHP script sends a password reset link to the registered email. | 1.5 Provide password reset functionality (on account management form). | Must execute password reset within 5 seconds. |  | M | Enables easy modification of user account password. |
| Create a “ STAFF ACCOUNTS” form using HTML, CSS. Add fields for username, password and account details (e.g. Name, Surname, Birthdate, Email, Telephone, Role). Use PHP to update the database with SQL SELECT, INSERT and UPDATE queries | 1.6 Allow management to create and modify staff accounts (on staff accounts form). | Must create the account within 5 seconds | Enable updates using PHP and SQL UPDATE queries, limiting modifiable fields. | M | Enables management to create, modify staff account and assign roles/accesses |
| Create a “BOOK A TABLE” form using HTML, CSS. Add fields for date, time, and guest count. Use PHP to query the database for availability. | 2.1 Check table availability (on Book a Table form). | Must confirm availability within 3 seconds. | Query the database for available tables using AJAX calls to a RESTful API. | M | |  | | --- | |  |  |  | | --- | | Ensures real-time interaction by checking table availability. \*(Note: status automatically sets to active). | |
| Add a “CONFIRM RESERVATION” button on the form using HTML. PHP script updates the database with SQL INSERT queries. | 2.2 Book a table and save reservation details (on Book a Table form). | Must save reservation within 3 seconds. | Insert reservation data into the database using PHP and SQL INSERT queries. Notifications must be sent via SMTP protocol within 5 seconds | M | Allows customers to finalize bookings and updates the reservation database. \*(Note: status of reservation automatically sets to active). |
| Create a “MANAGE RESERVATIONS” page using HTML, CSS. Add options for modifying or canceling reservations. Use PHP with SQL UPDATE and DELETE queries. | 2.3 View, modify or cancel reservations (on Manage Reservations page). | Must execute changes within 3 seconds. | Implement options for modification and cancellation using SQL UPDATE and DELETE queries. | M | Enables Customers to adjust or cancel their bookings as needed, Waitstaff to know the workflow that they must handle, Management to modify or cancel reservations depending on each account role. |
| Use PHP to send email/SMS confirmation or status updates. Integrate PHP mail() function. | 2.4 Send confirmation/change status/reminder notification to customer (via email). | Must send notifications within 5 seconds. |  | M | Keeps customers informed about their reservation status. |
| Create an “INPUT ORDER” form using HTML, CSS. Include fields for table number, item selection, quantity, and special requests. Use PHP to save data in the database with SQL INSERT queries. | 3.1 Add orders to the system (on Input Order form). | Must save orders within 2 seconds. | Store order details in a MySQL database using indexed fields for efficient querying. Ensure orders save using PHP and SQL INSERT queries. | M | Ensures real-time order input for seamless communication between waitstaff and kitchen. \*(Note: only waitstaff has access to this form also status automatically is submitted). |
| Create a “MANAGE ORDERS” form using HTML, CSS. Include options to update order statuses (e.g., "In Progress", "Ready"). Use PHP and SQL UPDATE queries. | 3.2 Update order statuses (on manage orders form). | Must update statuses within 2 seconds.  Enables real-time updates of order statuses for better coordination between kitchen and waitstaff. | Enable real-time updates using WebSocket connections to broadcast status changes to waitstaff terminals. | M | Enables real-time updates of order statuses for better coordination between kitchen and waitstaff. \*(Note: only kitchen staff has access to this form). |
| Create an “INVENTORY MANAGEMENT” form using HTML, CSS. Add fields for Item Name, Quantity On Hand, Reorder Level and Supplier Information. Use PHP for tole based access with SQL queries to track inventory levels and PHP mail() function for notifying management on low inventory | 4.1 Monitor inventory levels and send notifications for low inventory (on Inventory Management form). | Must alert management on low inventory within 3 seconds. | Track inventory using SQL SELECT queries and send low-stock alerts via SMTP | M | Prevents resource shortages.  \*(Note: kitchen staff update the inventory once per day. Management gets an email notification on work email when an item is on low inventory). |
| Add a “DAILY REPORT” button to the “INVENTORY MANAGEMENT’ form using HTML, CSS. Use PHP to generate usage reports using SQL SELECT and AGGREGATE queries. | 4.2 Generate daily inventory usage reports (on Inventory Management form). | Must generate reports within 5 seconds. | Use SQL AGGREGATE functions to generate reports and provide a download option as PDF files. | M | Provides management with insights into inventory usage for better planning and control. \*(Note: Only management has access) |
| Create a “MODIFY INVENTORY” form using HTML, CSS. Add fields for Item ID, Item Name, Reorder Level and Supplier Information. Link it to “INVENTORY MANAGEMENT” form via a hyperlink using HTML. | 4.3 Modify inventory items (on Inventory Management form). | Must execute changes within 3 seconds. | Use PHP with SQL INSERT, UPDATE and DELETE queries. | M | Ensures inventory is up-to-date and accurate. (\*Note: Only management has access). |
| Create a “SCHEDULE” form using HTML, CSS. Add fields Staff ID, Date, Name, Position, Shift Times and hours worked. Use PHP and SQL SELECT, INSERT, and UPDATE queries for handling schedule data. | 5.1 Add staff schedules (on Schedule form). | Must save schedules within 3 seconds. | Store schedules in a database with fields for staff ID, position, shift times, and hours worked. Use PHP and SQL queries to handle CRUD operations. | M | Ensures proper workforce allocation and operational efficiency. \*(Note: Only management has access) |
| Create a “VIEW SCHEDULE” form using HTML, CSS. Add fields Date, Name, Position, Shift Times and hours worked. Use PHP to implement role-based access and SQL SELECT queries for handling schedule data. | 5.2 Allow staff to view their schedule (on view schedule form) | Must execute within 3 seconds |  | M | Allows staff to view their individual schedule for the current and upcoming week while management has access to all the staff’s schedule |
| Create a “SCHEDULE REQUESTS’ form using HTML, CSS. Add fields Request ID, User ID, Date, Status and Comments. Use PHP to implement role-based access and SQL SELECT, INSERT, and UPDATE queries for handling schedule data. | 5.3 Allow staff to submit schedule requests (time off or changes). (on schedule request form). | Must submit request within 2 seconds | Implement a form with role-based access control (RBAC), allowing staff to view and submit only their own requests. | M | Enables staff members to request schedule changes or time off. Also allows them to track the status of their request. \*(Note: role-based access ensures that each waitstaff or kitchen staff member has access only on their own requests. Also they do not have access to modify the status of the request. Status is automatically set to pending). |
|  | 5.4 Allow management to approve or deny schedule requests. (time off or changes).(on schedule request form). | Must save status changes within 2 seconds | Provide a role-specific dashboard where management can update request statuses using PHP and SQL UPDATE queries. | M | Enables management to approve or deny schedule changes or time off. \*(Note: role-based access ensures that management has access to all staff’s requests). |