2024

singh, Rakesh

Accel Dream

9/28/2024

Acceldream ERP

**ERP Document Index**

**1. Introduction**

* 1.1 Purpose
* 1.2 Scope
* 1.3 Overview of ERP

**2. User Management Module (Login and Sign-up Screen)**

* 2.1 Overview
* 2.2 Functional Requirements
  + 2.2.1 User Registration
  + 2.2.2 User Authentication
  + 2.2.3 Role-Based Access Control (RBAC)
  + 2.2.4 Profile Management
* 2.3 User Interface Design
* 2.4 Technical Requirements
* 2.5 Testing Strategy

**3. Inventory Management Module**

* 3.1 Overview
* 3.2 Functional Requirements
  + 3.2.1 Product Management
  + 3.2.2 Stock Tracking
  + 3.2.3 Low Stock Alerts
  + 3.2.4 Barcode Scanning
* 3.3 User Interface Design
* 3.4 Technical Requirements
* 3.5 Testing Strategy

**4. Sales and Order Management Module**

* 4.1 Overview
* 4.2 Functional Requirements
  + 4.2.1 Customer Management
  + 4.2.2 Order Processing
  + 4.2.3 Payment Processing
  + 4.2.4 Order Tracking
* 4.3 User Interface Design
* 4.4 Technical Requirements
* 4.5 Testing Strategy

**5. Finance and Accounting Module**

* 5.1 Overview
* 5.2 Functional Requirements
  + 5.2.1 General Ledger
  + 5.2.2 Accounts Receivable and Payable
  + 5.2.3 Financial Reporting
* 5.3 User Interface Design
* 5.4 Technical Requirements
* 5.5 Testing Strategy

**6. Human Resources Management Module**

* 6.1 Overview
* 6.2 Functional Requirements
  + 6.2.1 Employee Management
  + 6.2.2 Payroll Processing
  + 6.2.3 Leave Management
  + 6.2.4 Performance Evaluation
* 6.3 User Interface Design
* 6.4 Technical Requirements
* 6.5 Testing Strategy

**7. Customer Relationship Management (CRM) Module**

* 7.1 Overview
* 7.2 Functional Requirements
  + 7.2.1 Lead Management
  + 7.2.2 Customer Interaction History
  + 7.2.3 Campaign Management
* 7.3 User Interface Design
* 7.4 Technical Requirements
* 7.5 Testing Strategy

**8. Reporting and Analytics Module**

* 8.1 Overview
* 8.2 Functional Requirements
  + 8.2.1 Custom Report Generation
  + 8.2.2 Dashboard Creation
  + 8.2.3 Data Visualization Tools
* 8.3 User Interface Design
* 8.4 Technical Requirements
* 8.5 Testing Strategy

**9. Supply Chain Management Module**

* 9.1 Overview
* 9.2 Functional Requirements
  + 9.2.1 Supplier Management
  + 9.2.2 Purchase Order Management
  + 9.2.3 Supply Chain Analytics
* 9.3 User Interface Design
* 9.4 Technical Requirements
* 9.5 Testing Strategy

**10. Implementation Plan**

* 10.1 Development Phases
* 10.2 Timeline
* 10.3 Resource Allocation

**11. Testing Requirements**

* 11.1 Functional Testing
* 11.2 Integration Testing
* 11.3 User Acceptance Testing (UAT)

**12. Conclusion**

* 12.1 Summary of Features
* 12.2 Future Enhancements

**13. Appendices**

* 13.1 Glossary of Terms
* 13.2 References
* 13.3 Contact Information

**2. User Management Module (Login and Sign-up Screen)**

**Login Page Screen Document with OTP Process**

**1. Overview**

The Login Page Screen allows users to securely access their accounts using a username/email and password, followed by a One-Time Password (OTP) verification process to enhance security.

**2. Objectives**

* To provide a secure and user-friendly login process.
* To implement OTP verification as an additional security layer.
* To ensure the system is resilient to unauthorized access.

**3. Scope**

This document covers the following functionalities:

* User Login with Username/Email and Password
* OTP Generation and Sending
* OTP Verification
* Error Handling and User Feedback

**4. Functional Requirements**

**4.1. User Login**

* **Inputs**:
  + Username/Email (string, required)
  + Password (string, required)
* **Processes**:
  + Validate inputs (non-empty, proper format for email).
  + Authenticate user credentials against the database.
  + On successful authentication, generate and send an OTP to the user’s registered email/phone number.

**4.2. OTP Generation and Sending**

* **Processes**:
  + Generate a unique OTP (6-digit code) upon successful login attempt.
  + Send the OTP via the preferred method (SMS, email).
  + Set an expiration time for the OTP (e.g., 5 minutes).

**4.3. OTP Verification**

* **Inputs**:
  + OTP (string, required)
* **Processes**:
  + Validate the entered OTP against the generated OTP.
  + Check if the OTP is within the expiration time.
  + On successful verification, log the user in and redirect them to the dashboard.
  + If verification fails, provide appropriate error messages.

**4.4. Error Handling and User Feedback**

* **Processes**:
  + Display error messages for invalid credentials, expired OTPs, or failed verification attempts.
  + Provide feedback on successful login or OTP submission.

**5. User Interface Design**

**5.1. Login Page Layout**

* **Components**:
  + Input Fields:
    - Username/Email
    - Password
  + Buttons:
    - Login
    - Send OTP (visible after credential validation)
  + Links:
    - Forgot Password
    - Register (for new users)
* **Flow**:
  + User enters username/email and password.
  + User clicks on "Login."
  + On successful authentication, display OTP input field.
  + User enters OTP and clicks on "Verify."

**5.2. Error Messages**

* "Invalid username or password."
* "OTP has expired. Please request a new one."
* "Invalid OTP. Please try again."

**5.3. Success Messages**

* "OTP sent to your registered email/phone."
* "Login successful. Redirecting..."

**6. Technical Requirements**

**6.1. Architecture**

* **Tech Stack**:
  + Frontend: React, Angular, or Vue.js
  + Backend: Node.js (Express) or Python (Django/Flask)
  + Database: PostgreSQL or MongoDB
  + OTP Service: Twilio or an email service provider (e.g., SendGrid)

**6.2. API Endpoints**

* **Authentication**
  + POST /api/login: Authenticate user and send OTP
  + POST /api/verify-otp: Verify the OTP entered by the user

**6.3. Data Models**

* **User Model**

json

Copy code

{

"id": "UUID",

"username": "String",

"email": "String",

"passwordHash": "String",

"otp": "String",

"otpExpiresAt": "DateTime"

}

**7. Implementation Plan**

**7.1. Development Phases**

1. **Phase 1: User Authentication**
   * Implement user login with username/email and password.
2. **Phase 2: OTP Generation and Sending**
   * Integrate OTP generation and sending functionality.
3. **Phase 3: OTP Verification**
   * Develop the OTP verification process.
4. **Phase 4: Testing and Deployment**
   * Conduct thorough testing (unit and user acceptance testing).

**7.2. Timeline**

* Estimated development time: 4-6 weeks.
* Allocate 1 week for testing and revisions.

**7.3. Resources Required**

* Development Team: 1 Backend Developer, 1 Frontend Developer, 1 UI/UX Designer, 1 QA Tester.
* Tools: Git for version control, Postman for API testing.

**8. Testing Requirements**

* **Functional Testing**
  + Validate each feature against the specifications.
* **User Acceptance Testing (UAT)**
  + Engage end-users to ensure usability and functionality.

**9. Acceptance Criteria**

* All specified functionalities are implemented and verified.
* The module passes all testing phases.
* User feedback is incorporated and addressed.

**10. Conclusion**

This document outlines the functional specifications for the Login Page Screen with OTP Process. The successful implementation of this module will enhance security and provide a seamless user experience during the login process.

**Signup Page Screen Document with OTP Process**

**1. Overview**

The Signup Page Screen allows new users to create an account securely by entering their personal information. An additional layer of security is provided through the One-Time Password (OTP) verification process, which ensures the validity of the provided contact information.

**2. Objectives**

* To provide a user-friendly interface for new user registration.
* To implement OTP verification for validating user contact information.
* To ensure secure handling of user data during the signup process.

**3. Scope**

This document covers the following functionalities:

* User Registration with Personal Information
* OTP Generation and Sending
* OTP Verification
* Error Handling and User Feedback

**4. Functional Requirements**

**4.1. User Registration**

* **Inputs**:
  + Username (string, required)
  + Email (string, required, unique)
  + Password (string, required, with validation for strength)
  + Phone Number (string, required)
  + First Name (string, required)
  + Last Name (string, required)
* **Processes**:
  + Validate all input fields (non-empty, proper formats, and uniqueness for email).
  + Hash the password before storing it in the database.
  + On successful registration, generate and send an OTP to the provided phone number.

**4.2. OTP Generation and Sending**

* **Processes**:
  + Generate a unique OTP (6-digit code) upon successful registration.
  + Send the OTP via SMS or email to the registered phone number.
  + Set an expiration time for the OTP (e.g., 5 minutes).

**4.3. OTP Verification**

* **Inputs**:
  + OTP (string, required)
* **Processes**:
  + Validate the entered OTP against the generated OTP.
  + Check if the OTP is within the expiration time.
  + On successful verification, complete the signup process and notify the user.
  + If verification fails, provide appropriate error messages.

**4.4. Error Handling and User Feedback**

* **Processes**:
  + Display error messages for invalid inputs, expired OTPs, or failed verification attempts.
  + Provide feedback on successful registration and OTP submission.

**5. User Interface Design**

**5.1. Signup Page Layout**

* **Components**:
  + Input Fields:
    - Username
    - Email
    - Password
    - Phone Number
    - First Name
    - Last Name
  + Buttons:
    - Sign Up
    - Send OTP (visible after successful registration)
  + Links:
    - Already have an account? (redirect to Login)
    - Privacy Policy (link to privacy policy)
* **Flow**:
  + User enters their personal information and clicks "Sign Up."
  + On successful registration, display OTP input field.
  + User enters OTP and clicks "Verify."

**5.2. Error Messages**

* "Email is already in use."
* "Invalid phone number format."
* "Password does not meet security requirements."
* "OTP has expired. Please request a new one."
* "Invalid OTP. Please try again."

**5.3. Success Messages**

* "Registration successful. An OTP has been sent to your phone."
* "Signup completed. Redirecting to your dashboard..."

**6. Technical Requirements**

**6.1. Architecture**

* **Tech Stack**:
  + Frontend: React, Angular, or Vue.js
  + Backend: Node.js (Express) or Python (Django/Flask)
  + Database: PostgreSQL or MongoDB
  + OTP Service: Twilio or an email service provider (e.g., SendGrid)

**6.2. API Endpoints**

* **User Registration**
  + POST /api/signup: Register a new user and send OTP
  + POST /api/verify-otp: Verify the OTP entered by the user

**6.3. Data Models**

* **User Model**

json

Copy code

{

"id": "UUID",

"username": "String",

"email": "String",

"passwordHash": "String",

"phoneNumber": "String",

"otp": "String",

"otpExpiresAt": "DateTime"

}

**7. Implementation Plan**

**7.1. Development Phases**

1. **Phase 1: User Registration**
   * Implement user registration with personal information.
2. **Phase 2: OTP Generation and Sending**
   * Integrate OTP generation and sending functionality.
3. **Phase 3: OTP Verification**
   * Develop the OTP verification process.
4. **Phase 4: Testing and Deployment**
   * Conduct thorough testing (unit and user acceptance testing).

**7.2. Timeline**

* Estimated development time: 4-6 weeks.
* Allocate 1 week for testing and revisions.

**7.3. Resources Required**

* Development Team: 1 Backend Developer, 1 Frontend Developer, 1 UI/UX Designer, 1 QA Tester.
* Tools: Git for version control, Postman for API testing.

**8. Testing Requirements**

* **Functional Testing**
  + Validate each feature against the specifications.
* **User Acceptance Testing (UAT)**
  + Engage end-users to ensure usability and functionality.

**9. Acceptance Criteria**

* All specified functionalities are implemented and verified.
* The module passes all testing phases.
* User feedback is incorporated and addressed.

**10. Conclusion**

This document outlines the functional specifications for the Signup Page Screen with OTP Process. The successful implementation of this module will enhance user registration security and provide a seamless experience for new users.

**List of essential modules that would be included in a Acceldream ERP Application:**

**1. Inventory Management**

* Product Management
* Stock Tracking
* Warehouse Management
* Barcode Scanning
* Stock Alerts and Reordering

**2. Sales Management**

* Sales Order Processing
* Point of Sale (POS) System
* Payment Processing
* Sales Analytics and Reporting
* Discounts and Promotions Management

**3. Customer Relationship Management (CRM)**

* Customer Profiles and Segmentation
* Purchase History Tracking
* Loyalty and Rewards Programs
* Customer Communication Logs
* Feedback and Support Management

**4. Supplier Management**

* Supplier Profiles and Contacts
* Purchase Order Management
* Supplier Performance Tracking
* Invoice Management

**5. Financial Management**

* General Ledger
* Accounts Receivable and Payable
* Financial Reporting
* Budgeting and Forecasting

**6. Reporting and Analytics**

* Sales Reports
* Inventory Reports
* Customer Insights
* Performance Dashboards
* Custom Report Generation

**7. User Management**

* Role-Based Access Control
* User Authentication and Security
* Activity Logging

**8. Human Resources Management (HRM) (if applicable)**

* Employee Records Management
* Attendance Tracking
* Payroll Management
* Recruitment and Onboarding

**9. E-commerce Integration**

* Online Store Management
* Order Synchronization
* Customer Management
* Payment Gateway Integration

**10. Supply Chain Management (optional)**

* Demand Planning
* Procurement Management
* Logistics and Shipping Management

**11. Mobile Application (optional)**

* Mobile Access to ERP Features
* Inventory Management on the Go
* Sales Processing via Mobile Devices

**12. Help Desk and Support (optional)**

* Ticketing System
* Customer Support Management
* Knowledge Base

**13. Compliance and Audit Management**

* Regulatory Compliance Tracking
* Audit Trails and Reports

**14. Document Management**

* Centralized Document Repository
* Document Sharing and Collaboration

**Functional Document for Inventory Management Module**

**1. Overview**

The Inventory Management module will facilitate the management of stock in a retail environment, ensuring accurate tracking, control, and reporting of inventory levels, movements, and product details.

**2. Purpose**

To provide a comprehensive system that helps users manage inventory efficiently, minimize stock discrepancies, and streamline reordering processes.

**3. Scope**

This module will cover:

* Product Management
* Stock Tracking
* Barcode Scanning
* Stock Alerts and Reordering
* Reporting and Analytics

**4. Functional Requirements**

**4.1. Product Management**

* **Add New Products**
  + **Inputs**:
    - Product Name (string, required)
    - SKU (string, unique, required)
    - Category (dropdown selection, required)
    - Supplier Information (name, contact details)
    - Purchase Price (decimal, required)
    - Selling Price (decimal, required)
    - Initial Stock Quantity (integer, required)
  + **Processes**:
    - Validate inputs (e.g., check for SKU uniqueness).
    - Store product details in the database.
    - Generate a confirmation message.
* **Update Existing Products**
  + **Processes**:
    - Fetch product details based on SKU.
    - Allow modification of product attributes.
    - Save changes and log updates for audit.
* **Delete Products**
  + **Processes**:
    - Verify if the product is not linked to active transactions.
    - Confirm deletion and remove from the database.

**4.2. Stock Tracking**

* **Real-Time Stock Levels**
  + **Processes**:
    - Update stock levels automatically after sales or stock additions.
    - Display current stock levels on the product detail page.
* **Movement Tracking**
  + **Processes**:
    - Log each stock movement (additions, sales, returns) with timestamps.
    - Maintain a transaction history for auditing.

**4.3. Barcode Scanning**

* **Barcode Integration**
  + **Processes**:
    - Use barcode scanning for product identification during sales or stock updates.
    - Allow manual entry as a fallback.

**4.4. Stock Alerts and Reordering**

* **Low Stock Alerts**
  + **Processes**:
    - Set thresholds for minimum stock levels.
    - Automatically trigger alerts via email or in-app notifications when stock is low.
* **Reorder Management**
  + **Processes**:
    - Generate purchase orders based on low stock alerts.
    - Track order status until stock is replenished.

**4.5. Reporting and Analytics**

* **Inventory Reports**
  + **Processes**:
    - Generate reports on current stock levels, movements, and turnover rates.
    - Provide options to export reports in various formats (CSV, PDF).
* **Performance Dashboards**
  + **Processes**:
    - Visualize key metrics with graphs and charts.
    - Allow filters by date range and product categories.

**5. User Interface Design**

* **Dashboard**
  + A user-friendly dashboard displaying key metrics (total stock, low stock alerts, recent movements).
* **Product Management Interface**
  + Form layout for adding/updating products with clear labels and validation messages.
* **Reports Section**
  + Table layout for reports with export options and filtering capabilities.

**6. Non-Functional Requirements**

* **Performance**
  + The system should handle 1,000+ products with a response time of under 2 seconds for all queries.
* **Scalability**
  + Must support an increase to 10,000 products without requiring significant architectural changes.
* **Security**
  + Implement role-based access control (RBAC) to restrict access to sensitive features.
  + Ensure data encryption for sensitive information.
* **Usability**
  + Conduct usability testing to ensure the interface is intuitive and easy to navigate.

**7. Architecture**

* **Tech Stack**:
  + Frontend: React, Angular, or Vue.js
  + Backend: Node.js with Express or Python with Django/Flask
  + Database: PostgreSQL or MySQL
  + Hosting: AWS, Azure, or on-premises server
* **Data Flow Diagram**:
  + Diagram depicting user interactions with the system, showing how data flows between components (frontend, backend, database).

**8. Implementation Considerations**

* **Development Phases**:
  + Phase 1: Product Management features
  + Phase 2: Stock Tracking and Alerts
  + Phase 3: Reporting and Analytics
  + Phase 4: Testing and User Feedback
* **Version Control**: Use Git for version control, with a clear branching strategy (e.g., main, develop, feature branches).
* **Testing Strategy**:
  + Unit Testing: Each function/component should have unit tests.
  + Integration Testing: Test interactions between modules.
  + User Acceptance Testing (UAT): Engage with end-users for feedback.

**9. Acceptance Criteria**

* All functional requirements are implemented and verified.
* The module passes performance and usability testing.
* Users can manage inventory effectively and generate reports without issues.

**10. Conclusion**

This document outlines the functional and development aspects of the Inventory Management module. Successful implementation will enhance inventory control, reduce stock discrepancies, and improve operational efficiency within the retail shop.

**Functional Document for Product Management Module**

**1. Overview**

The Product Management module is designed to allow users to efficiently manage the lifecycle of products within the retail shop. This includes adding new products, updating existing ones, managing product details, and organizing inventory.

**2. Objectives**

* To provide a user-friendly interface for adding, updating, and deleting products.
* To ensure accurate tracking of product information and stock levels.
* To facilitate easy categorization and searching of products.

**3. Scope**

This module will encompass the following functionalities:

* Add New Products
* Update Existing Products
* Delete Products
* View Product Details
* Manage Product Categories
* Search and Filter Products

**4. Functional Requirements**

**4.1. Add New Products**

* **Inputs**:
  + Product Name (string, required)
  + SKU (string, unique, required)
  + Category (dropdown selection, required)
  + Supplier Information (string, optional)
  + Purchase Price (decimal, required)
  + Selling Price (decimal, required)
  + Initial Stock Quantity (integer, required)
  + Product Description (string, optional)
  + Image Upload (file, optional)
* **Processes**:
  + Validate all input fields to ensure required fields are filled and formats are correct.
  + Store product details in the database.
  + Generate a confirmation message upon successful addition.

**4.2. Update Existing Products**

* **Inputs**:
  + Users can search for a product by SKU or name.
  + Modify fields including product name, category, prices, stock quantity, and description.
* **Processes**:
  + Fetch existing product details based on user input.
  + Validate updated fields.
  + Save changes and log the update for audit purposes.
  + Notify the user upon successful update.

**4.3. Delete Products**

* **Processes**:
  + Users can search for products by SKU or name.
  + Confirm deletion through a dialog box to prevent accidental removal.
  + Check if the product is linked to active transactions before allowing deletion.
  + Remove product from the database and notify the user.

**4.4. View Product Details**

* **Processes**:
  + Users can view product details by searching for SKU or name.
  + Display all relevant information including name, SKU, category, stock level, prices, and descriptions.

**4.5. Manage Product Categories**

* **Processes**:
  + Users can add new categories.
  + Users can edit existing categories (name, description).
  + Users can delete categories (with confirmation), ensuring no products are linked to them.

**4.6. Search and Filter Products**

* **Features**:
  + Implement a search bar for product names or SKUs.
  + Provide filters for categories, stock levels, and price ranges.

**5. User Interface Design**

**5.1. Product Management Dashboard**

* **Main Features**:
  + Quick access buttons for adding, searching, and managing products.
  + Display a summary of products (total count, low stock alerts).

**5.2. Add New Product Form**

* **Layout**:
  + Input fields with clear labels.
  + Validation messages below each field for errors.
  + Submit and Cancel buttons at the bottom.

**5.3. Product List View**

* **Table Layout**:
  + Columns: Product Name, SKU, Category, Stock Quantity, Purchase Price, Selling Price, Actions (Edit/Delete).
  + Search bar and filter options at the top.

**5.4. Product Detail View**

* **Details Display**:
  + Display all relevant product information with options to edit or delete.

**6. Technical Requirements**

**6.1. Architecture**

* **Tech Stack**:
  + Frontend: React or Angular
  + Backend: Node.js (Express) or Python (Django/Flask)
  + Database: PostgreSQL or MySQL

**6.2. API Endpoints**

* **Product Management**
  + POST /api/products: Add a new product
  + GET /api/products/:id: Retrieve product details
  + PUT /api/products/:id: Update product
  + DELETE /api/products/:id: Delete product
  + GET /api/categories: Fetch all categories
  + POST /api/categories: Add a new category

**7. Implementation Plan**

**7.1. Development Phases**

1. **Phase 1: Basic Product CRUD Operations**
   * Implement adding, updating, and deleting products.
2. **Phase 2: Category Management**
   * Develop functionalities for managing product categories.
3. **Phase 3: Search and Filter Features**
   * Integrate search and filtering capabilities.
4. **Phase 4: Testing and Deployment**
   * Conduct thorough testing (unit and user acceptance testing).

**7.2. Timeline**

* Estimated development time: 4-6 weeks.
* Allocate 1 week for testing and revisions.

**7.3. Resources Required**

* Development Team: 1 Backend Developer, 1 Frontend Developer, 1 UI/UX Designer, 1 QA Tester.
* Tools: Git for version control, Postman for API testing.

**8. Testing Requirements**

* **Functional Testing**
  + Validate each feature against the specifications.
* **User Acceptance Testing (UAT)**
  + Engage end-users to ensure usability and functionality.

**9. Acceptance Criteria**

* All specified functionalities are implemented and verified.
* The module passes all testing phases.
* User feedback is incorporated and addressed.

**10. Conclusion**

This document outlines the functional specifications for the Product Management module. The successful implementation of this module will enhance product handling and improve inventory accuracy in the retail shop.