**Tally Ho! Inventory System**

1. **Project Overview**

* **Tally Ho! Inventory System:** Tally Ho! is a web-based inventory management system designed to streamline how organizations track and manage physical items. The system provides a structured way for users to add and maintain item records, including descriptions, quantities, and images. Through a permission-based workflow, users can submit deletion requests that require administrator approval, ensuring transparency and accountability. With an emphasis on simplicity, usability, and control, Tally Ho! supports effective inventory oversight for teams of all sizes.
* **Objective**: To develop a centralized system for managing inventory of items, allowing users to add, track, and request deletion of items with admin oversight.
* **Stakeholders:** Admins, Inventory Managers, Employees
* **Scope**:
  + The system manages item records including name, quantity, description, image, and ownership.
  + Users can request deletion of items, and all actions are logged. Admins approve deletions, manage user roles, and review history.

1. **Requirements Gathering**

**Functional Requirements:**

* **Item Management:** Users can add, view, and request deletion of item records.
* **Image Upload:** Each item can include an uploaded photo.
* **Role Management:** Admins manage user accounts and roles.
* **Activity Logs:** All item-related actions are recorded.
* **Delete Requests:** Users submit deletion requests which are reviewed and approved/rejected by admins.

**Non-Functional Requirements:**

* **Scalability:** Capable of handling growth in item volume and users.
* **Security:** Session-based access, input sanitization, and access restrictions by role.
* **Usability:** Responsive and user-friendly web UI.

1. **System Design System Architecture:**

* **Frontend:** HTML/CSS/JavaScript and PHP forms for user interaction.
* **Backend:** PHP handling sessions, form processing, and database access.

**Database Design:**

**Tables:**

* **users** (id, username, password, role, status)
* **items** (id, name, description, quantity, image\_path, added\_by, date\_added)
* **history** (id, item\_id, action, performed\_by, timestamp)
* **delete\_requests** (id, item\_id, user\_id, reason, status, request\_date)

**User Interface Design:**

* User Dashboard: Add/View Items, Submit Delete Requests
* Admin Dashboard: Manage Users, View All Items, Approve Delete Requests, View Logs

1. **Development Plan**

**Frontend:**

* Technologies: HTML, CSS for layout and styling; basic JavaScript for form behavior.
* Forms: Login, Add item, request deletion, login.

**Backend:**

* PHP scripts for form handling, CRUD operations, login, and session management.
* Image upload handling with validation.

**Database:**

* SQL schema to define users, items, history, and requests.
* Queries to support item management and activity logging.

**User Roles:**

* Admin: Full access including approval and user management.
* User: Can add/view their items and request deletions.

**5. Testing Plan Unit Testing:**

* Test PHP functions for item handling, request processing, and form validation.

**Integration Testing:**

* Confirm seamless interaction between forms, PHP logic, and database queries.

**User Acceptance Testing (UAT):**

* End-users and admins test workflows like adding items, submitting requests, and reviewing history.

1. **Deployment Plan Local Testing:**

* XAMPP/WAMP stack for Apache and MySQL testing.

**Production Deployment:**

* Upload project files to a PHP-enabled hosting server.
* Import database SQL and set folder permissions (especially /uploads).

**Permissions:**

* Set correct folder permissions for uploading images.
* Enforce access controls based on session roles.

**7. Maintenance and Support Routine Maintenance**:

* Monitor database growth and clean inactive records.
* Ensure PHP and MySQL versions are up to date.

**Backup Strategy:**

* Weekly backups of the database and uploaded images.

**Log Monitoring:**

* Regular checks of user activity via history table.

**8. Project Closure Final Deliverables:**

* Source code and file structure
* Database SQL script
* User guide for system use

**Lessons Learned:**

* Clear role-based workflows simplify system use.
* Admin approval for deletions adds traceability and control.