**Design Document: Main Inventory System**

**1. Introduction**

This document details the architecture and design for the Main Inventory Management System, reflecting the supplied MariaDB schema. The system manages all tracked IT and lab assets, composite PC setups, employees, and storage, and provides reporting via database views.

**2. System Architecture**

* **Web Stack:** PHP backend, HTML/JavaScript frontend
* **Database:** MariaDB/MySQL, using schema as supplied
* **Authentication:** Employee accounts (with role/type)

**3. Data Model**

**3.1 Core Tables**

* **Component Tables:**
  + accessories
  + graphicscards
  + keyboards
  + mice
  + minipc
  + monitors
  + motherboards
  + powersupplies
  + ramsticks
  + storage\_components
* **Composites:**
  + pcsetups (links components by ID)
* **Meta:**
  + employees (accounts, authentication, role)
  + storage\_slots (locations for storage management)

**3.2 Relationships**

* Each component table has its own primary key (e.g., gpu\_id in graphicscards).
* pcsetups references components using foreign keys (enforced in schema).
* storage\_components references storage\_slots with FK.
* Employees authenticate via employee\_id and password.
* All relationships must honor referential integrity.

**3.3 Views**

* component\_totals – summary by category/cost
* disposed\_parts – list of disposed assets
* stored\_components\_storage – assets currently in storage by category

**4. Main Components**

**4.1 Backend (PHP)**

* **Component Controllers:**
  + CRUD for each asset table, input validation, status and condition logic
* **PC Setup Controller:**
  + CRUD for composite setups, ensures all referenced components exist, prevents orphaned references
* **Employee Controller:**
  + CRUD, authentication, password, role management
* **Storage Controller:**
  + CRUD for storage components and slots
* **Reporting:**
  + Read from database views, export as CSV

**4.2 Frontend (HTML/JS)**

* **Dashboards:**
  + Asset tables per component, composite PC setups, storage, reporting views
* **Detail/Edit Forms:**
  + Per-asset, per-PC setup, per-employee, per-storage slot/component
* **Authentication UI:**
  + Login/logout forms for employees
* **Reporting:**
  + Export buttons, printable views, summary dashboards

**5. Core Flows**

* **Asset Management:**
  + Add/edit/delete assets per type; validate required fields and unique IDs; update status and condition
* **PC Setup Management:**
  + Create/edit/delete composite setups; enforce FK integrity to component tables; prevent deletion if referenced elsewhere
* **Employee Management:**
  + Add/edit/delete employees; enforce unique emails; hash and store passwords; manage employee types/roles
* **Storage Management:**
  + Assign storage components to slots; update storage details as items move in/out
* **Reporting:**
  + Query views for totals, disposed, storage; display in UI and allow CSV export

**6. Validation & Business Logic**

* Unique PKs in each table
* FK integrity for all composite and storage references
* Required fields per table enforced on both client and server
* Only ENUM-allowed values for status/condition in each table
* On PC setup creation/update, all referenced IDs must exist
* Employee authentication uses hashed passwords
* Prevent deletion of components if referenced in pcsetups or other FKs

**7. Security**

* Passwords stored
* SQL injection protection via prepared statements
* Input validation and sanitization for all forms
* Role-based access: restrict sensitive actions to appropriate employee\_type

**8. Error Handling**

* Frontend: Inline user feedback for validation failures and DB errors
* Backend: Logging for failed operations, exceptions, and unauthorized access

**9. Extensibility**

* Adding new component types:
  + Add new table, update controllers and reporting logic
* Adding new composite relationships:
  + Add FKs and update PC setup logic
* Future enhancements:
  + Rental, analytics, notification systems can be added modularly

**10. Non-Functional**

* Responsive, accessible UI
* Modular PHP code with clear separation of concerns
* Efficient, indexed queries for fast lookups and reporting
* Inline code documentation and user/developer guides

**11. Out of Scope**

* Rental or reservation management
* Financial tracking, vendor integration
* Notification/alert system
* Advanced analytics/dashboarding beyond provided views
* External API or system integration

**12. Appendix: Table & Field Reference**

**Component Tables:**

* **accessories:** acc\_id, name, type, condition, cost, status
* **graphicscards:** gpu\_id, name, condition, cost, status, location
* **keyboards:** kb\_id, name, condition, cost, status, location
* **mice:** mouse\_id, name, condition, cost, status, location
* **minipc:** mipc\_id, name, condition, cost, status, location
* **monitors:** monitor\_id, name, width, condition, cost, status, location
* **motherboards:** mobo\_id, name, size, condition, cost, status, location
* **powersupplies:** psu\_id, name, wattage, modular, condition, cost, status, location
* **ramsticks:** ram\_id, name, type, speed, condition, cost, status, location
* **storage\_components:** storage\_id, storage\_slot\_id, name, media, type, capacity, condition, cost, status, location

**Composite & Meta Tables:**

* **pcsetups:** pc\_id, mobo\_id, gpu\_id, ram\_id, storage\_slot\_id, psu\_id, monitor\_id, acc\_id, kb\_id, mouse\_id, tableLocation, PCcondition
* **employees:** employee\_id, password, first\_name, last\_name, email, hire\_date, employee\_type
* **storage\_slots:** storage\_slot\_id, description, location

**Database Views:**

* **component\_totals:** category-level summary (count/cost)
* **disposed\_parts:** disposed/retired asset records
* **stored\_components\_storage:** assets currently in storage by type

**13. Repository Structure Recommendation**

* /backend/ — PHP controllers, models, validation, DB connection
* /frontend/ — HTML, CSS, JS, forms, dashboards, authentication views
* /sql/ — Schema DDL, view definitions, sample data
* /docs/ — Requirements, design, and user documentation
* /tests/ — Unit/integration tests for critical logic
* /exports/ — Generated CSV and report files (ignored in VCS)