**Main Inventory System Requirements**

**1. Functional Requirements**

**1.1 Component Inventory**

* Admin/staff can add, edit, delete, and view:
  + Accessories
  + Graphics Cards (GPUs)
  + Keyboards
  + Mice
  + MiniPCs
  + Monitors
  + Motherboards
  + Power Supplies
  + RAM Sticks
  + Storage Components and Storage Slots
* Each component type has its own set of fields as per schema (e.g., status, condition, location, cost, etc.).

**1.2 PC Setups Management**

* Admin/staff can add, edit, delete, and view composite PC setups.
* PC setups can link existing components (mobo, GPU, RAM, PSU, monitor, keyboard, mouse, storage) via IDs.
* PC setups track overall condition and location.

**1.3 Employee Management**

* Admin/staff can add, edit, and delete employee accounts.
* Employees have roles (employee\_type), authentication (password), and contact information.
* Only employees with valid credentials can access management features.

**1.4 Status & Condition Tracking**

* Track and update physical condition (GOOD/BROKEN) and status (IN\_USE, STORAGE, DISPOSED, etc.) for each item.
* Status determines availability for use or assignment.

**1.5 Location Tracking**

* Track current location for every asset/component and PC setup.
* Locations are free-text (e.g., "Table 2 - 1", "Storage room 212").

**1.6 Reporting (Views)**

* Use database views:
  + component\_totals: count/cost summary by category
  + disposed\_parts: list/category of disposed assets
  + stored\_components\_storage: what's in storage by category
* Admin/staff can view and export these reports.

**1.7 Data Validation**

* Enforce:
  + Unique IDs for all assets (per component table)
  + Mandatory fields (e.g., name, condition, status, etc.)
  + Referential integrity for PC setups and storage components (FK constraints)

**2. Technical Requirements**

**2.1 Database Structure**

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

**Views:**

* component\_totals
* disposed\_parts
* stored\_components\_storage

**Key Constraints:**

* All component IDs are primary keys (VARCHAR)
* Foreign keys for composite setups and storage (see schema)
* Employee emails are unique

**2.2 Backend Logic**

* CRUD operations for each component table and composite PC setups
* Employee CRUD and authentication
* Status and condition updates
* PC setup creation links to component IDs (must exist)
* Export for all views (CSV)

**2.3 Frontend Logic**

* Forms for each asset type and PC setup
* List/detail views per asset type, storage, and disposed reports
* Reporting dashboard (summary, totals, export)
* Employee management and login UI

**2.4 Access Control**

* Only employees with valid credentials can access management
* Role-based access via employee\_type (DBA, FullTime, SoftwareDev)

**2.5 Non-Functional**

* Secure password storage
* Input validation (client & server)
* Prepared statements for SQL
* Responsive UI
* Documentation and code comments

**2.6 Out of Scope**

* No rental/reservation system
* No advanced analytics beyond provided views
* No integrations with external systems

**3. Acceptance Criteria**

* All assets can be managed per-table and as composite setups
* Employees are authenticated and managed via the system
* All relationships and references honored (no orphaned setups)
* Disposed, stored, and totals views accurately reflect DB
* Data entry is validated, errors handled gracefully