

Project: Asset Management Database

GROUP D MEMBERS

Name	Student Number	Registration Number
ATURINZIRE HARGREAVE	2400722603	24/U/22603/PS
AKANKUNDA RITA	2400703072	24/U/03072/PS
NDAGIRE NAIRAH	2400710032	24/U/10032/PS
MUTEBI STUART	2400725953	24/U/25953/PS
NANTALE CECILIA	2400724555	24/U/24555/PS

1. Introduction and Purpose

The **Asset Management Database** is designed as the centralized, authoritative system for all organizational assets. Its primary objective is to move beyond simple record-keeping to provide a **complete, structured repository** that supports all phases of an asset's life—from initial acquisition through to its final disposal. This database is critical for enabling accurate reporting, financial valuation, and strategic decision-making regarding the organization's investments in physical and digital resources.

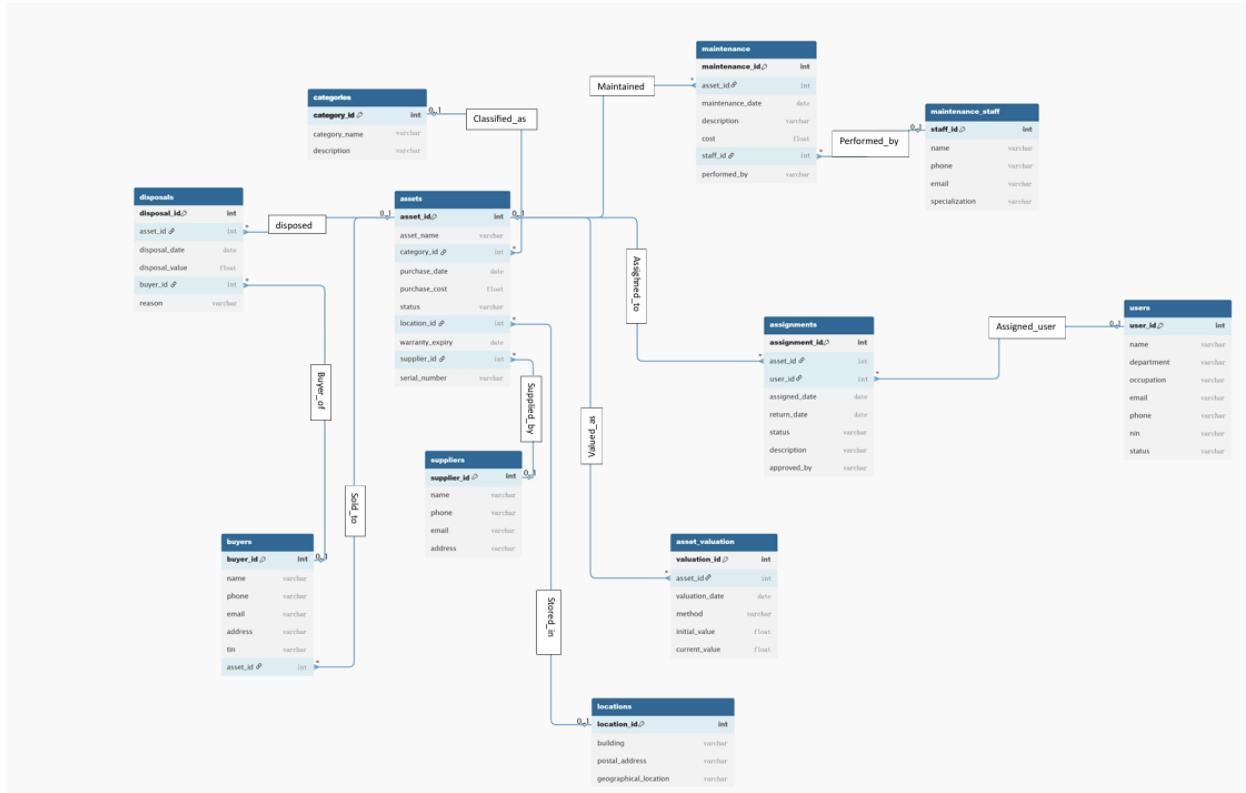
1.1 Scope and Key Functionalities

The scope of the database is comprehensive, covering all data necessary for effective asset governance. It includes core **Asset Master Data** (such as serial numbers, categories, and current status) alongside supporting organizational metadata (details on suppliers, buyers, and geographic locations). Crucially, the system tracks all transactional history, including **user assignments, scheduled and unscheduled maintenance logs, and financial records** related to valuation and disposal events. This integrated approach ensures that data is consistent and provides a single, reliable source of truth for every asset in the inventory.

1.2 Logical Design

The relational structure of the Asset Management Database is formally defined by the **Entity-Relationship Diagram (ERD)**, which serves as the foundational blueprint for the underlying MySQL schema. The design is engineered to prioritize **referential integrity** and maintain a complete historical audit trail for every asset through dedicated transactional tables.

ERD(Entity Relationship Diagram)



Enterprise Rules

The following key **business constraints** govern the data within the system and ensure its integrity:

- **Asset Categorization:** Every asset must be strictly assigned to one and only one category and must be located in exactly one registered location.
- **Lifecycle History:** A single asset can be associated with multiple lifecycle events throughout its lifetime, including assignments to users, maintenance actions, and valuation updates.
- **Maintenance Validation:** All maintenance records must be linked to qualified personnel verified via the `maintenance_staff` table.
- **Disposal Finality:** An asset can undergo the disposal process only once, establishing a final, one-to-one link to the record in the `disposals` table.

1.3 Physical Design

The physical implementation translates the logical design into structured MySQL tables, utilizing **Primary Keys (PKs)** and **Foreign Keys (FKs)** to enforce the mandatory relationships defined in the ERD.

Tables and Primary Keys (PKs)

The master tables and transactional tables, identified by their primary keys, are:

- **Master Data:** `assets` (PK: `asset_id`),
- `categories` (PK: `category_id`),
- `locations` (PK: `location_id`),
- `suppliers` (PK: `supplier_id`),
- `users` (PK: `user_id`),
- `maintenance_staff` (PK: `staff_id`).
- **Transactional Data:** `assignments` (PK: `assignment_id`),
- `maintenance` (PK: `maintenance_id`),
- `asset_valuation` (PK: `valuation_id`),
- `disposals` (PK: `disposal_id`),
- `buyers` (PK: `buyer_id`).

Key Relationships (Foreign Keys — FKs)

Foreign keys establish the links between these tables:

- **Master Data to Asset:** The `assets` table includes FKs to `category_id`, `location_id`, and `supplier_id`.
- **Asset to Lifecycle:** All transactional tables (`assignments`, `maintenance`, `asset_valuation`, `disposals`) contain the FK `asset_id`, establishing the crucial 1-to-N relationships necessary for historical tracking.
- **Personnel Links:** The `assignments` table links to `user_id` (from the `users` table), and the `maintenance` table links to `staff_id` (from `maintenance_staff`).

2. Implementation and Conclusion

2.1 Technology Stack (Simplified)

The core technologies used are:

- **Database (MySQL):** Chosen because it is highly reliable and handles a large volume of data transactions very quickly, which is critical for an organizational system.
- **Backend (Django & DRF):** This Python-based framework manages the logic, security, and data flow. It uses a tool (the ORM) that allows the system to read and write data to the MySQL database using simple Python commands instead of complex SQL code.
- **Frontend (React):** The user interface (UI) is built using **React**. This tool was chosen because it creates a fast, modern user experience and is easy to update. The interface communicates with the Django backend using standardized **APIs** (data exchange rules) to show and update asset information.

2.2 Key Database Construction Command

To illustrate the physical construction, the following `CREATE TABLE` statement defines the core structure of the `assets` table, enforcing data integrity through primary keys, foreign keys, and required fields.

```
CREATE TABLE assets (
    asset_id INT PRIMARY KEY AUTO_INCREMENT,
    serial_number VARCHAR(100) NOT NULL UNIQUE,
    asset_name VARCHAR(255) NOT NULL,
    acquisition_date DATE NOT NULL,
    current_status ENUM('Active', 'In Maintenance', 'Disposed') NOT NULL,

    category_id INT NOT NULL,
    location_id INT NOT NULL,
    supplier_id INT,

    FOREIGN KEY (category_id) REFERENCES categories(category_id),
    FOREIGN KEY (location_id) REFERENCES locations(location_id),
    FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id)
);
```

2.3 Data Integrity Mechanisms

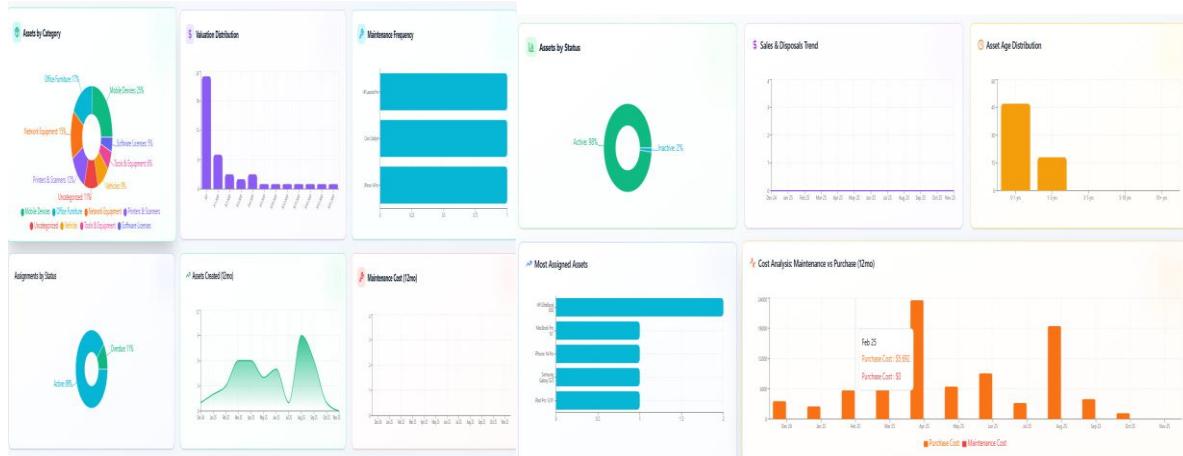
Data integrity is the cornerstone of the Asset Management Database, ensuring that data is accurate, consistent, and reliable. This integrity is maintained at the database level through the strategic use of constraints and specific data types:

- **Referential Integrity (Foreign Keys):** This is the most critical constraint, enforced by **Foreign Keys (FKs)**. These links ensure that an asset cannot be assigned a `category_id` or `location_id` unless that ID already exists in the `categories` or `locations` master tables, preventing "orphaned" or invalid records.
- **Entity Integrity (Primary Keys): Primary Keys (PKs)** guarantee that every asset has a unique identifier (`asset_id`), ensuring that no two records are ever confused. The `UNIQUE` constraint on `serial_number` also prevents duplicate physical assets from being entered.

- **Domain Integrity (Data Types):**
 - **NOT NULL:** Ensures that mandatory fields like `serial_number`, `asset_name`, and `acquisition_date` are always populated.
 - **ENUM:** The `current_status` field uses an `ENUM` type, restricting its values to a predefined list ('Active', 'In Maintenance', 'Disposed'), which eliminates typographical errors and ensures standardized reporting.
 - **DATE:** The `acquisition_date` is strictly stored as a date, guaranteeing data consistency for financial and historical analysis.

2.4 Sample GUI and Reports

The Next.js frontend provides a unified graphical interface for data entry and display. The dashboard aggregates data from the MySQL database in real-time to generate key operational reports.



2.5 Conclusion and Next Steps

Project Success & Testing

The project successfully delivered a **fully operational, multi-tiered Asset Management System** with a robust design. Success was confirmed through **thorough unit and integration testing** conducted via Django, which verified the core ORM functionality and the reliability of relationship constraints. This testing confirmed the accuracy of complex reporting queries and the stability of Foreign Key definitions.

Technical Challenges:

1. **Data Type Mapping:** Converting large Django model text fields to the most efficient and performant MySQL data types.
2. **Report Optimization:** Fine-tuning complex multi-table joins required for time-series and valuation reports. This was resolved through the strategic **indexing** of key Foreign Key columns to enhance query speeds.

Future Work

System Resilience: **Implementing** automated scheduled database backups **for disaster recovery.**

- **Advanced Features:** Developing **predictive maintenance features** using historical data.