Track-Stack Database

✔ **Users** (Admin, Contributor, Viewer)  
✔ **Assets**  
✔ **Inventory**  
✔ **Categories**  
✔ **Storage**  
✔ **Transactions**  
✔ **Reports**  
✔ **Audit logs**  
✔ **Access control**  
✔ **Notifications**

### ****Relationship Check****

|  |  |  |
| --- | --- | --- |
| **Entity 1** | **Entity 2** | **Relationship Type** |
| Users → Assets | One-to-Many | A user can manage multiple assets, but an asset is managed by one user. |
| Users → Inventory | One-to-Many | A user can manage multiple inventory items, but each inventory record has one responsible user. |
| Users → Reports | One-to-Many | A user can generate multiple reports, but each report is associated with one user. |
| Users → Roles | Many-to-One | Multiple users can have the same role, but a user has one role at a time. |
| Users → Departments | Many-to-One | Multiple users belong to one department, but a user is in one department at a time. |
| Assets → Categories | Many-to-One | Multiple assets can belong to one category, but an asset has one category. |
| Assets → Locations | Many-to-One | Multiple assets can be in one location, but an asset is in one location at a time. |
| Assets→ Maintenance | One-to-Many | An asset can have multiple maintenance records, but each maintenance record belongs to one asset. |
| Assets→ Transactions | One-to-Many | An asset can have multiple transaction records (check-in/check-out), but each transaction belongs to one asset. |
| Inventory → Products | One-to-One | Each inventory record corresponds to one product. |
| Inventory → Supplier | Many-to-One | Multiple inventory items can come from one supplier, but each item has one supplier. |
| Inventory→ Transactions | One-to-Many | An inventory item can have multiple transactions (in/out), but each transaction belongs to one inventory item. |
| Reports → Assets | Many-to-Many | A report can include multiple assets, and an asset can appear in multiple reports. |
| Reports → Inventory | Many-to-Many | A report can include multiple inventory items, and an inventory item can appear in multiple reports. |

**Users Schema (Admin, Contributor, Viewer)**

User Management Relationships

**Users → Roles** Many-to-One

- Each user has one role (Admin, Contributor, Viewer)

- Multiple users can have the same role

**Users → Assets** One-to-Many

- Users can create/manage multiple assets

- Each asset has one creator/owner

**Users → AuditLogs** One-to-Many

- Each user's actions are tracked in multiple audit logs

- Each log entry is associated with one user

**📌 Users Collection Schema**

{

"\_id": ObjectId("user123"),

"username": "john.doe",

"email": "john.doe@company.com",

"password": "hashedPassword123",

"role": "Admin", // Can be "Admin", "Contributor", or "Viewer"

"department": "IT",

"profile": {

"firstName": "John",

"lastName": "Doe",

"phone": "+1234567890",

"avatar": "avatar123.jpg"

},

"permissions": ["manage\_assets", "view\_reports", "edit\_inventory"],

"lastLogin": ISODate("2024-03-11T10:00:00Z"),

"createdAt": ISODate("2024-03-11T10:00:00Z"),

"updatedAt": ISODate("2024-03-11T10:00:00Z")

}

- Handles user authentication and authorization

- Role-based access control (Admin, Contributor, Viewer)

- Department assignment for organizational structure

- Tracks user permissions and login history

- Stores basic profile information

### ****🔹 Explanation of Users Fields:****

### Top-Level Fields

### \_id

### Unique identifier for the user, stored as an ObjectId.

### Example: ObjectId("user123")

### username

### The unique username of the user.

### Example: "john.doe"

### email

### The user’s email address.

### Example: "john.doe@company.com"

### password

### The user's password, stored as a hashed value for security.

### Example: "hashedPassword123" (not the actual password but a hashed version).

### role

### Defines the user’s access level.

### Possible values:

### "Admin" → Full access

### "Contributor" → Limited access to edit/create content

### "Viewer" → Read-only access

### Example: "Admin"

### department

### The department the user belongs to.

### Example: "IT"

### Profile Section (profile)

### Contains personal details of the user.

### firstName

### User’s first name.

### Example: "John"

### lastName

### User’s last name.

### Example: "Doe"

### phone

### User’s contact number.

### Example: "+1234567890"

### avatar

### Filename of the user’s profile picture.

### Example: "avatar123.jpg"

### Permissions (permissions)

### Defines the user’s capabilities within the system.

### permissions

### List of allowed actions for the user.

### Example: ["manage\_assets", "view\_reports", "edit\_inventory"]

### Possible permissions:

### "manage\_assets" → Can create, edit, or delete assets

### "view\_reports" → Can view reports

### "edit\_inventory" → Can modify inventory

### Audit Fields

### Track login history and account creation details.

### lastLogin

### Timestamp of the user’s last login.

### Example: ISODate("2024-03-11T10:00:00Z")

### createdAt

### Timestamp when the user account was created.

### Example: ISODate("2024-03-11T10:00:00Z")

### updatedAt

### Timestamp of the last profile update.

### Example: ISODate("2024-03-11T10:00:00Z")

### ****📌 Assets Schema****

Asset Management Relationships

**Assets → Categories** Many-to-One

- Multiple assets can belong to one category

- Each asset must be in one category

**Assets → Tags** Many-to-Many

- Assets can have multiple tags

- Tags can be applied to multiple assets

**Assets → Versions** One-to-Many

- One asset can have multiple versions

- Each version belongs to one asset

### ****📌 Assets Collection Schema****

{

"\_id": ObjectId("asset123"),

"name": "Project Presentation",

"type": "application/pdf",

"category": "Documents",

"tags": ["project", "2024", "presentation"],

"metadata": {

"size": 1024576, // in bytes

"dimensions": null,

"duration": null

},

"storage": {

"path": "/documents/presentation.pdf",

"url": "https://storage.example.com/presentation.pdf"

},

"versions": [

{

"version": 1,

"path": "/documents/presentation\_v1.pdf",

"createdBy": ObjectId("user123"),

"createdAt": ISODate("2024-03-11T10:00:00Z")

}

],

"createdBy": ObjectId("user123"),

"createdAt": ISODate("2024-03-11T10:00:00Z"),

"updatedAt": ISODate("2024-03-11T10:00:00Z")

}

- Manages digital assets and their metadata

- Supports versioning for file changes

- Categorization through tags and categories

- Tracks file storage locations and URLs

- Stores technical metadata (size, dimensions)

### ****🔹 Explanation of Assets Fields:****

**Top-Level Fields**

1. **\_id:**
   * Unique identifier for the asset, represented as an ObjectId.
   * Example: ObjectId("asset123")
2. **name:**
   * Name of the asset.
   * Example: "Project Presentation"
3. **type:**
   * The file type (MIME type).
   * Example: "application/pdf" (indicates a PDF file)
4. **category:**
   * Categorization of the asset.
   * Example: "Documents"
5. **tags:**
   * An array of keywords for better searchability.
   * Example: ["project", "2024", "presentation"]

**Metadata Section (metadata)**

Contains additional information about the file.

1. **size:**
   * File size in bytes.
   * Example: 1024576 (≈1MB)
2. **dimensions:**
   * For image/video assets, this would store width & height.
   * Since this is a PDF, it is null.
3. **duration:**
   * If this were an audio or video file, it would store duration in seconds.
   * Since this is a PDF, it is null.

**Storage Section (storage)**

Contains file storage details.

1. **path:**
   * Internal storage path where the file is located.
   * Example: "/documents/presentation.pdf"
2. **url:**
   * Publicly accessible URL to download/view the file.

**Versions Section (versions)**

Keeps track of previous versions of the file.

1. **Each version object contains:**

* version: The version number (e.g., 1 for first version).
* path: Internal path for the specific version ("/documents/presentation\_v1.pdf").
* createdBy: ID of the user who uploaded this version (ObjectId("user123")).
* createdAt: Timestamp when this version was created (ISODate("2024-03-11T10:00:00Z")).

**Audit Fields**

Track who created/modified the asset and when.

1. **createdBy:**
   * User ID of the creator (ObjectId("user123")).
2. **createdAt:**
   * Timestamp of when the file was created (ISODate("2024-03-11T10:00:00Z")).
3. **updatedAt:**
   * Timestamp of the last update (ISODate("2024-03-11T10:00:00Z")).

### ****📌 Inventory Schema****

**Assets → Suppliers** Many-to-One

- Multiple assets can come from one supplier

- Each asset has one primary supplier

**Assets → StockAlerts** One-to-Many

- One asset can have multiple stock alerts

- Each alert belongs to one asset

**Assets → Transactions** One-to-Many

- One asset can have multiple transactions

### - Each transaction belongs to one asset

### ****📌 Inventory Collection Schema****

{

"\_id": ObjectId("inv123"),

"assetId": ObjectId("asset123"),

"quantity": 50,

"location": {

"warehouse": "Main",

"section": "A1",

"shelf": "S1"

},

"status": "In\_Stock", // Can be "In\_Stock", "Low\_Stock", "Out\_of\_Stock"

"threshold": {

"minimum": 10,

"maximum": 100

},

"transactions": [

{

"type": "check\_out",

"quantity": 5,

"userId": ObjectId("user123"),

"timestamp": ISODate("2024-03-11T10:30:00Z")

}

],

"createdAt": ISODate("2024-03-11T10:00:00Z"),

"updatedAt": ISODate("2024-03-11T10:30:00Z")

}

- Tracks physical inventory levels

- Manages storage locations

- Records stock thresholds for alerts

- Maintains transaction history

- Links to digital assets

### ****🔹 Explanation of Inventory Fields:****

**Top-Level Fields**

1. **\_id**
   * Unique identifier for the inventory record, stored as an ObjectId.
   * Example: ObjectId("inv123")
2. **assetId**
   * Reference to the asset being tracked in inventory.
   * Example: ObjectId("asset123") (links to an asset document)
3. **quantity**
   * The current stock count of the asset.
   * Example: 50 (means 50 units of the asset are available)

**Location Section (location)**

Stores details about where the asset is stored.

1. **warehouse**
   * Name of the warehouse where the asset is stored.
   * Example: "Main"
2. **section**
   * The section within the warehouse.
   * Example: "A1"
3. **shelf**
   * The specific shelf where the asset is located.
   * Example: "S1"

**Stock Status (status)**

Defines the availability of the asset.

1. **status** 
   * Represents the stock level.
   * Possible values:
     + "In\_Stock" → Sufficient stock available
     + "Low\_Stock" → Stock is below the minimum threshold
     + "Out\_of\_Stock" → No stock available
   * Example: "In\_Stock"

**Stock Threshold (threshold)**

Defines the stock limits**.**

1. **minimum**
   * The minimum quantity before stock is considered low.
   * Example: 10
2. **maximum**
   * The maximum quantity to prevent overstocking.
   * Example: 100

**Transaction History (transactions)**

Records stock movements (e.g., check-ins, check-outs).

1. **Each transaction object contains:**

* type → Type of transaction:
  + "check\_out" → Stock taken out
  + "check\_in" → Stock added
* quantity → Number of units moved (e.g., 5 units checked out).
* userId → User responsible for the transaction (ObjectId("user123")).
* timestamp → When the transaction happened (ISODate("2024-03-11T10:30:00Z")).

**Audit Fields**

Track when the inventory record was created or updated.

1. **createdAt**

* Timestamp when the inventory record was created.
* Example: ISODate("2024-03-11T10:00:00Z")

1. **updatedAt**

* Timestamp when the inventory was last updated.
* Example: ISODate("2024-03-11T10:30:00Z")

**📌 AuditLog Schema**

**Assets → SharedAccess** One-to-Many

- One asset can be shared with multiple users

- Each share record belongs to one asset

**Users → Notifications**  One-to-Many

- Users can receive multiple notifications

- Each notification is for one user

### ****📌 AuditLog Collection Schema****

{

"\_id": ObjectId("log123"),

"entityType": "asset", // Can be "asset", "inventory", "user"

"entityId": ObjectId("asset123"),

"action": "update", // Can be "create", "update", "delete"

"changes": {

"before": { "name": "Old Name" },

"after": { "name": "Project Presentation" }

},

"performedBy": ObjectId("user123"),

"timestamp": ISODate("2024-03-11T10:15:00Z"),

"ipAddress": "192.168.1.100"

}

- Records all system changes

- Tracks who made changes

- Stores before/after states

- Maintains security audit trail

- Timestamps all actions

### ****🔹 Explanation of AuditLog Fields:****

### Top-Level Fields

### \_id

### Unique identifier for the log entry, stored as an ObjectId.

### Example: ObjectId("log123")

### entityType

### Specifies what type of entity was affected.

### Possible values:

### "asset" → An asset record was changed

### "inventory" → An inventory record was changed

### "user" → A user record was changed

### Example: "asset" (indicates changes were made to an asset)

### entityId

### The unique ID of the entity that was modified.

### Example: ObjectId("asset123") (refers to a specific asset)

### action

### Describes what kind of change occurred.

### Possible values:

### "create" → A new entity was created

### "update" → An existing entity was modified

### "delete" → An entity was deleted

### Example: "update" (indicates an update was made)

### Change Details (changes)

### Tracks what was modified.

### before

### The state of the entity before the change.

### Example: { "name": "Old Name" }

### after

### The state of the entity after the change.

### Example: { "name": "Project Presentation" }

### User & Timestamp Fields

### performedBy

### The user who made the change.

### Example: ObjectId("user123") (refers to a user document)

### timestamp

### When the action occurred.

### Example: ISODate("2024-03-11T10:15:00Z")

### ipAddress

### The IP address of the user who performed the action.

### Example: "192.168.1.100"