**Cold Storage Management System Project Document**

Name : Mohd Maaz

BITS ID: 2024MT12175

**Version History**

| **Version Number** | **Date** | **Author/Owner** | **Description of Change** |
| --- | --- | --- | --- |
| 1 | 07-09-2024 | Mohd Maaz | Added Problem Statement |

**REQUIREMENTS SPECIFICATION**

**Problem Statement & Requirements Definition**

**To manage the storage of vegetables in a cold storage facility. The system tracks the in boarding and outboarding of vegetable bags by farmers, manages lot numbers for better storage, calculates storage rent based on the duration of storage, and generates rent invoice accordingly. It will help in accounting and replace paper based system. The goal is to ensure efficient management of storage data, accurate rent calculation, and streamlined billing processes for farmers.**

**Requirements Definition:**

1. **Track Inboarding and Outboarding:** Record the number of vegetable bags stored and retrieved by farmers along with the respective dates.
2. **Lot Number Management:** Assign and track lot numbers to groups of vegetable bags for better organization and traceability within the cold storage.
3. **Vegetable Category Management:** Categorize vegetable bags as either "Seed" or "Vegetable" and apply different rent rates accordingly.
4. **Calculate Storage Rent:** Implement a rent calculation mechanism based on the vegetable category, duration of storage, and predefined rent per year.
5. **Manage Farmer Information:** Store and retrieve farmer details such as Farmer ID, Name, Contact Information, and Address.

**Project Features Identified**

| **Feature ID** | **Feature Name** | **Description** |
| --- | --- | --- |
| F01 | Login | Authentication and on success take you to your dashboard |
| F02 | Farmer Management | Manage farmer details including Farmer ID, Name, Contact Information, and Address |
| F03 | Vegetable Bag In boarding and outboarding | Record the In/out-boarding of vegetable bags, including the number of bags, lot number, vegetable category, and the date of outboarding |
| F04 | Lot Number Management | Assign and manage lot numbers for vegetable bags to ensure proper organization and traceability within the cold storage |
| F05 | Vegetable Category Management | Categorize vegetable bags as "Seed" or "Vegetable" and apply different rent rates based on the category |
| F06 | Rent Calculation | Calculate storage rent based on the vegetable category, duration of storage, and the predefined rent per year |
| F07 | Report Generation | Generate reports for farmers, including storage duration, rent calculations, and billing history |

**Software and Hardware Details**

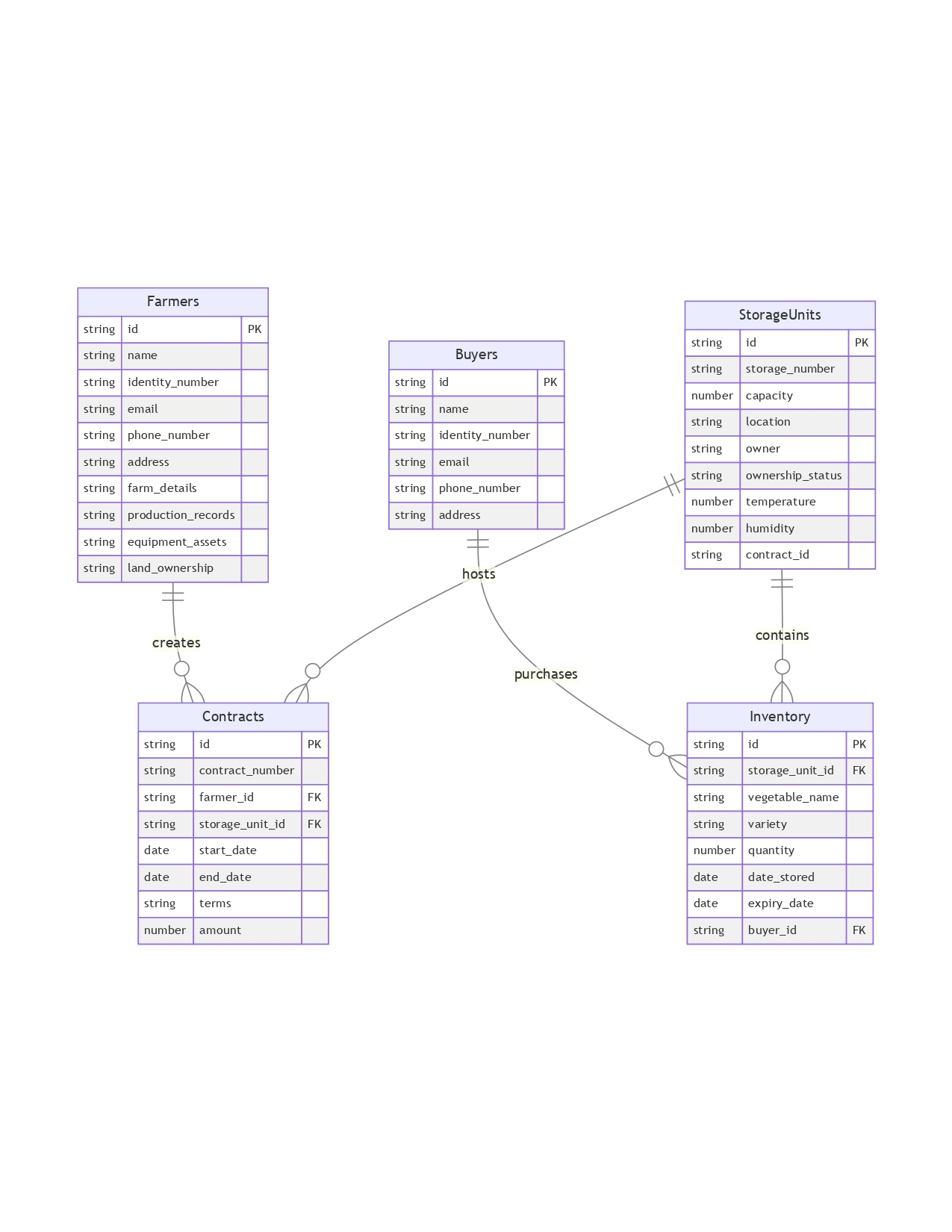
| **Platform** | **Details** |
| --- | --- |
| **Platform** | VS Code (code Editor) |
| **Frontend/console** | Frontend |
| **Backend/server** | Backend (Server) |
| **Database** | MongoDB |
| **Frontend Programming Language** | HTML5, CSS, Javascript |
| **Backend Programming Language** | JavaScript |

**Project Plan**

**=============== END OF DELIVERABLE 1 ====================**

**II. CONCEPTUAL DESIGN:**

ER Diagram



**Object Model:**

**Entities and Attributes**

**1. Farmers**

* Attributes:
  + id (Primary Key)
  + name
  + identity\_number (Unique)
  + email (Unique)
  + phone\_number
  + address
  + farm\_details
  + production\_records
  + equipment\_assets
  + land\_ownership

**2. Buyers**

* Attributes:
  + id (Primary Key)
  + name
  + identity\_number (Unique)
  + email (Unique)
  + phone\_number
  + address

**3. Storage units**

* Attributes:
  + id (Primary Key)
  + storage\_number (Unique)
  + capacity
  + location
  + owner
  + ownership\_status (Enum: "Owned", "Leased")
  + temperature
  + humidity
  + contract\_id (Foreign Key to Contracts.id)

**4. Contracts**

* Attributes:
  + id (Primary Key)
  + contract\_number (Unique)
  + farmer\_id (Foreign Key to Farmers.id)
  + storage\_unit\_id (Foreign Key to Storage units.id)
  + start\_date
  + end\_date
  + terms
  + amount

**5. Inventory**

* Attributes:
  + id (Primary Key)
  + storage\_unit\_id (Foreign Key to Storage units.id)
  + vegetable\_name
  + variety
  + quantity
  + date\_stored
  + expiry\_date
  + buyer\_id (Foreign Key to Buyers.id)

**Relationships**

1. **Farmers ↔ Contracts**
   * A farmer can have multiple contracts
   * Cardinality: 1
2. **Contracts ↔ Storage units**
   * A contract is associated with one storage unit, but a storage unit may not necessarily have a contract
   * Cardinality: 1:1 (optional on the Storage units side)
3. **Storage units ↔ Inventory**
   * A storage unit can store multiple inventory items
   * Cardinality: 1
4. **Buyers ↔ Inventory**
   * A buyer can purchase multiple inventory items
   * Cardinality: 1

**ER Diagram Object Model:**

**Entities:**

* Farmers (Primary Key: id)
* Buyers (Primary Key: id)
* StorageUnits (Primary Key: id)
* Contracts (Primary Key: id)
* Inventory (Primary Key: id)

**Relationships:**

* Farmers (1) ↔ (N) Contracts
* Contracts (1) ↔ (1) StorageUnits
* StorageUnits (1) ↔ (N) Inventory
* Buyers (1) ↔ (N) Inventory

**===============ENDOFDELIVERABLE2====================**

**III LOGICAL DESIGN**

* + 1. Farmers Table

CREATE TABLE Farmers (

farmer\_id VARCHAR(255) PRIMARY KEY,

name VARCHAR(255) NOT NULL,

identity\_number VARCHAR(255) UNIQUE NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

phone\_number VARCHAR(255) NOT NULL,

address VARCHAR(255),

farm\_details TEXT,

production\_records TEXT,

equipment\_assets TEXT,

land\_ownership TEXT

);

* + 1. Buyers Table

CREATE TABLE Buyers (

buyer\_id VARCHAR(255) PRIMARY KEY,

name VARCHAR(255) NOT NULL,

identity\_number VARCHAR(255) UNIQUE NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

phone\_number VARCHAR(255) NOT NULL,

address VARCHAR(255)

);

* + 1. Storage Units Table

CREATE TABLE StorageUnits (

storage\_unit\_id VARCHAR(255) PRIMARY KEY,

storage\_number VARCHAR(255) UNIQUE NOT NULL,

capacity DECIMAL(10,2) NOT NULL CHECK (capacity >= 0),

location VARCHAR(255) NOT NULL,

owner VARCHAR(255) NOT NULL,

ownership\_status ENUM('Owned', 'Leased') NOT NULL,

temperature DECIMAL(5,2),

humidity DECIMAL(5,2),

contract\_id VARCHAR(255)

);

* + 1. Contracts Table

CREATE TABLE Contracts (

contract\_id VARCHAR(255) PRIMARY KEY,

contract\_number VARCHAR(255) UNIQUE NOT NULL,

farmer\_id VARCHAR(255) NOT NULL,

storage\_unit\_id VARCHAR(255) NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

terms TEXT,

amount DECIMAL(10,2) NOT NULL CHECK (amount >= 0),

FOREIGN KEY (farmer\_id) REFERENCES Farmers(farmer\_id),

FOREIGN KEY (storage\_unit\_id) REFERENCES StorageUnits(storage\_unit\_id)

);

* + 1. Inventory Table

CREATE TABLE Inventory (

inventory\_id VARCHAR(255) PRIMARY KEY,

storage\_unit\_id VARCHAR(255) NOT NULL,

vegetable\_name VARCHAR(255) NOT NULL,

variety VARCHAR(255),

quantity DECIMAL(10,2) NOT NULL CHECK (quantity >= 0),

date\_stored DATE NOT NULL,

expiry\_date DATE NOT NULL,

buyer\_id VARCHAR(255) NOT NULL,

FOREIGN KEY (storage\_unit\_id) REFERENCES StorageUnits(storage\_unit\_id),

FOREIGN KEY (buyer\_id) REFERENCES Buyers(buyer\_id)

);

Corresponding **MongoDB commands**:

use cold\_storage\_db;

1. **db.createCollection**("Farmers", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["id", "name", "identity\_number", "email",   
 "phone\_number"],

properties: {

id: {

bsonType: "string"

},

name: {

bsonType: "string"

},

identity\_number: {

bsonType: "string"

},

email: {

bsonType: "string",

pattern: "^.+@.+\\..+$"

},

phone\_number: {

bsonType: "string"

},

address: {

bsonType: "string"

},

farm\_details: {

bsonType: "string"

},

production\_records: {

bsonType: "string"

},

equipment\_assets: {

bsonType: "string"

},

land\_ownership: {

bsonType: "string"

}

}

}

}

});

db.Farmers.createIndex({ identity\_number: 1 }, { unique: true });

db.Farmers.createIndex({ email: 1 }, { unique: true });

1. **db.createCollection**("Buyers", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["id", "name", "identity\_number", "email",   
 "phone\_number"],

properties: {

id: {

bsonType: "string"

},

name: {

bsonType: "string"

},

identity\_number: {

bsonType: "string"

},

email: {

bsonType: "string",

pattern: "^.+@.+\\..+$"

},

phone\_number: {

bsonType: "string"

},

address: {

bsonType: "string"

}

}

}

}

});

db.Buyers.createIndex({ identity\_number: 1 }, { unique: true });

db.Buyers.createIndex({ email: 1 }, { unique: true });

1. **db.createCollection**("StorageUnits", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["id", "storage\_number", "capacity", "location", "owner",   
 "ownership\_status", "temperature", "humidity"],

properties: {

id: {

bsonType: "string"

},

storage\_number: {

bsonType: "string"

},

capacity: {

bsonType: "number",

minimum: 0

},

location: {

bsonType: "string"

},

owner: {

bsonType: "string"

},

ownership\_status: {

enum: ["Owned", "Leased"]

},

temperature: {

bsonType: "number"

},

humidity: {

bsonType: "number"

},

contract\_id: {

bsonType: "string"

}

}

}

}

});

db.StorageUnits.createIndex({ storage\_number: 1 }, { unique: true });

1. **db.createCollection**("Contracts", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["id", "contract\_number", "farmer\_id", "storage\_unit\_id",   
 "start\_date", "end\_date", "amount"],

properties: {

id: {

bsonType: "string"

},

contract\_number: {

bsonType: "string"

},

farmer\_id: {

bsonType: "string"

},

storage\_unit\_id: {

bsonType: "string"

},

start\_date: {

bsonType: "date"

},

end\_date: {

bsonType: "date"

},

terms: {

bsonType: "string"

},

amount: {

bsonType: "number",

minimum: 0

}

}

}

}

});

db.Contracts.createIndex({ contract\_number: 1 }, { unique: true });

1. **db.createCollection**("Inventory", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["id", "storage\_unit\_id", "vegetable\_name", "quantity",   
 "date\_stored", "expiry\_date", "buyer\_id"],

properties: {

id: {

bsonType: "string"

},

storage\_unit\_id: {

bsonType: "string"

},

vegetable\_name: {

bsonType: "string"

},

variety: {

bsonType: "string"

},

quantity: {

bsonType: "number",

minimum: 0

},

date\_stored: {

bsonType: "date"

},

expiry\_date: {

bsonType: "date"

},

buyer\_id: {

bsonType: "string"

}

}

}

}

});

db.Inventory.createIndex({ id: 1 }, { unique: true });

**Normalization Analysis for Each Table**

**Farmers Table**

State: 3NF ✓ Reason:

* Primary Key: farmer\_id
* No multivalued attributes
* No partial dependencies
* No transitive dependencies
* All non-key attributes fully dependent on primary key

**Buyers Table**

State: 3NF ✓ Reason:

* Primary Key: buyer\_id
* No multivalued attributes
* No partial dependencies
* No transitive dependencies
* Each attribute depends directly on primary key

**Storage Units Table**

State: 3NF ✓ Reason:

* Primary Key: storage\_unit\_id
* No multivalued attributes
* No partial dependencies
* No transitive dependencies
* Attributes like temperature and humidity directly related to storage unit

**Contracts Table**

State: 3NF ✓ Reason:

* Primary Key: contract\_id
* No multivalued attributes
* No partial dependencies
* No transitive dependencies
* All attributes directly dependent on contract primary key

**Inventory Table**

State: 3NF ✓ Reason:

* Primary Key: inventory\_id
* No multivalued attributes
* No partial dependencies
* No transitive dependencies
* Attributes like quantity and expiry date directly related to inventory item

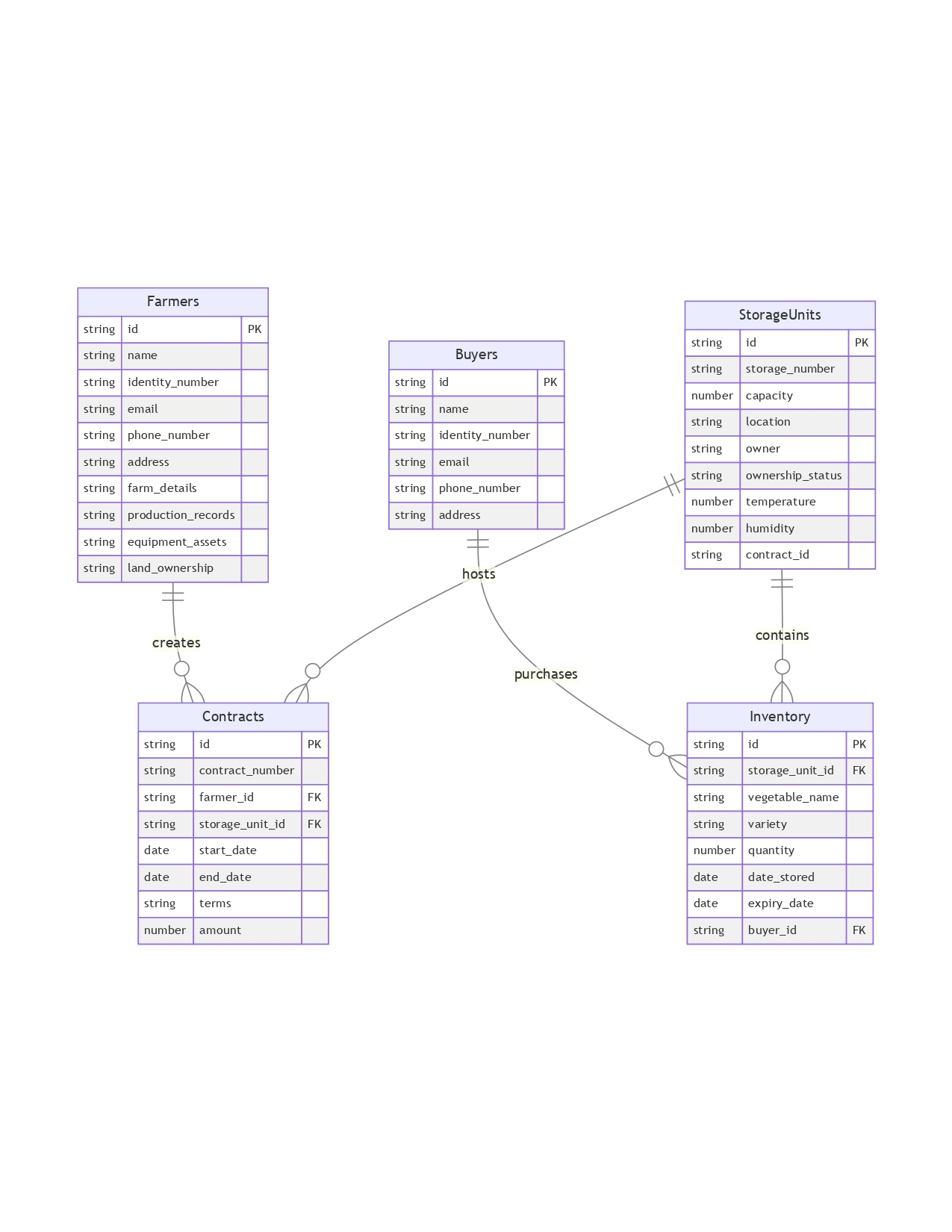
Version Control Table

**State: 3NF ✓ Reason:**

* Primary Key: version\_id
* Tracks schema changes linearly
* No multivalued attributes
* No partial dependencies
* No transitive dependencies

**Normalization Strategy**

* Separated complex attributes
* Eliminated data redundancy
* Maintained referential integrity
* Ensured functional dependencies

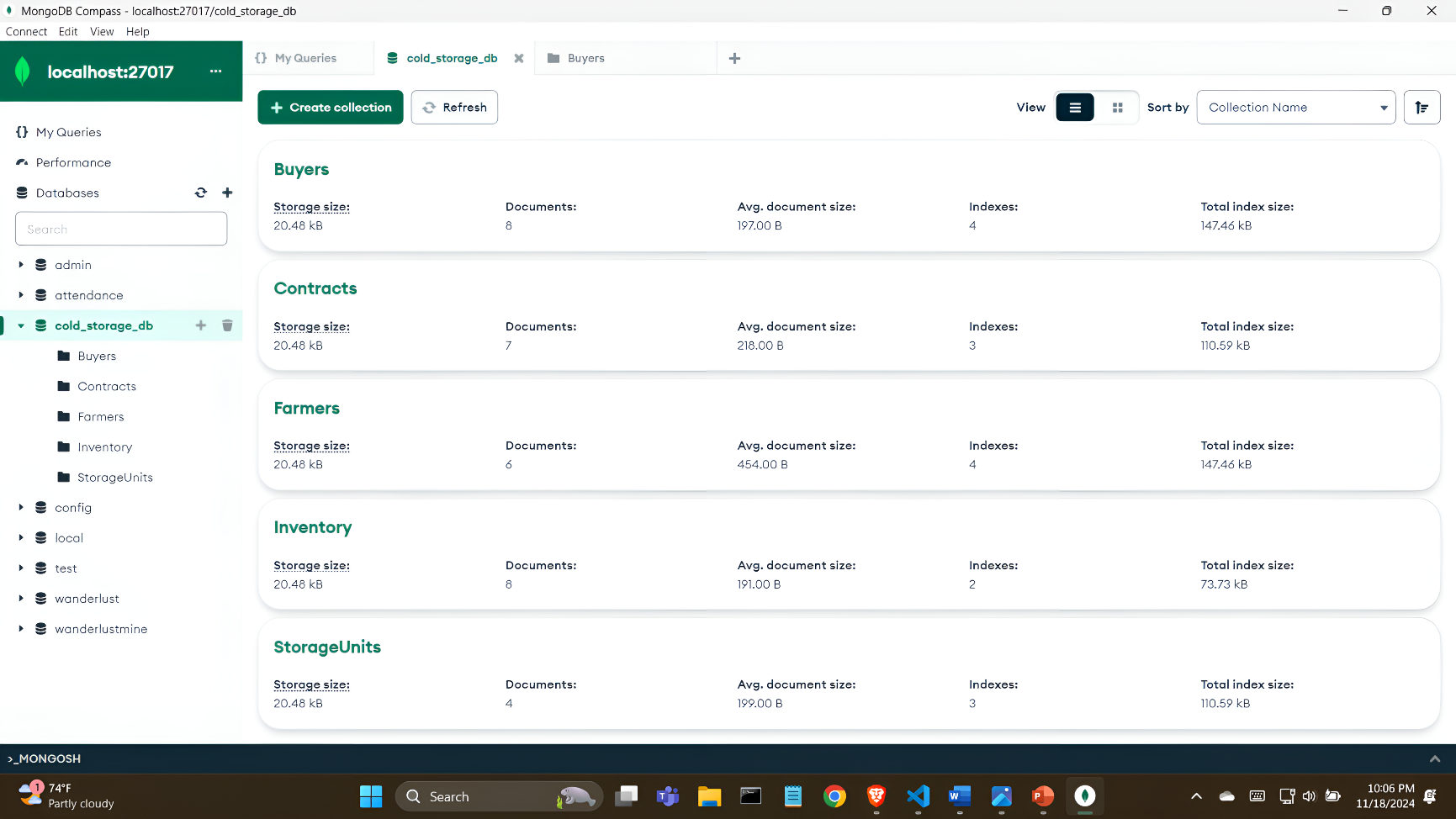
**ER Diagram Object Model**

**Entities:**

* Farmers (Primary Key: id)
* Buyers (Primary Key: id)
* Storage units (Primary Key: id)
* Contracts (Primary Key: id)
* Inventory (Primary Key: id)

**Relationships:**

* Farmers (1) ↔ (N) Contracts
* Contracts (1) ↔ (1) Storage Units
* Storage units (1) ↔ (N) Inventory
* Buyers (1) ↔ (N) Inventory

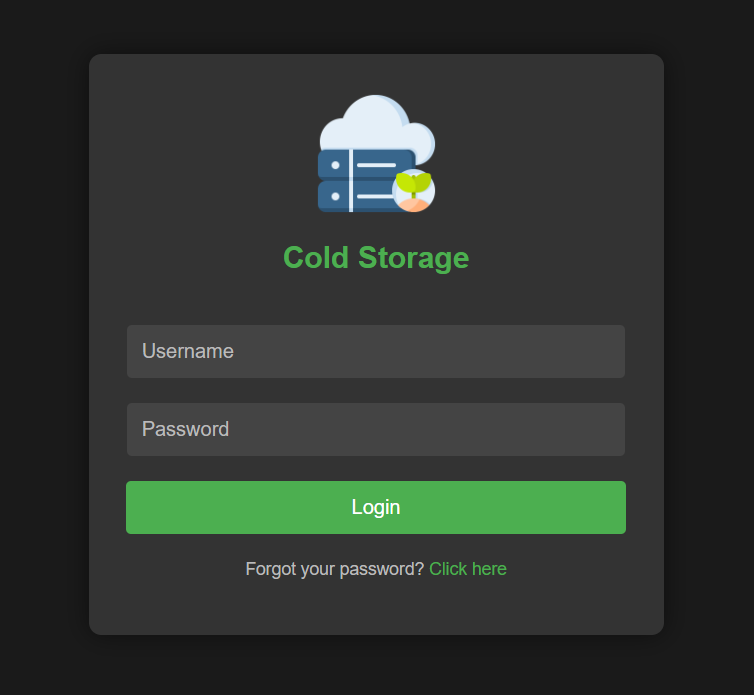


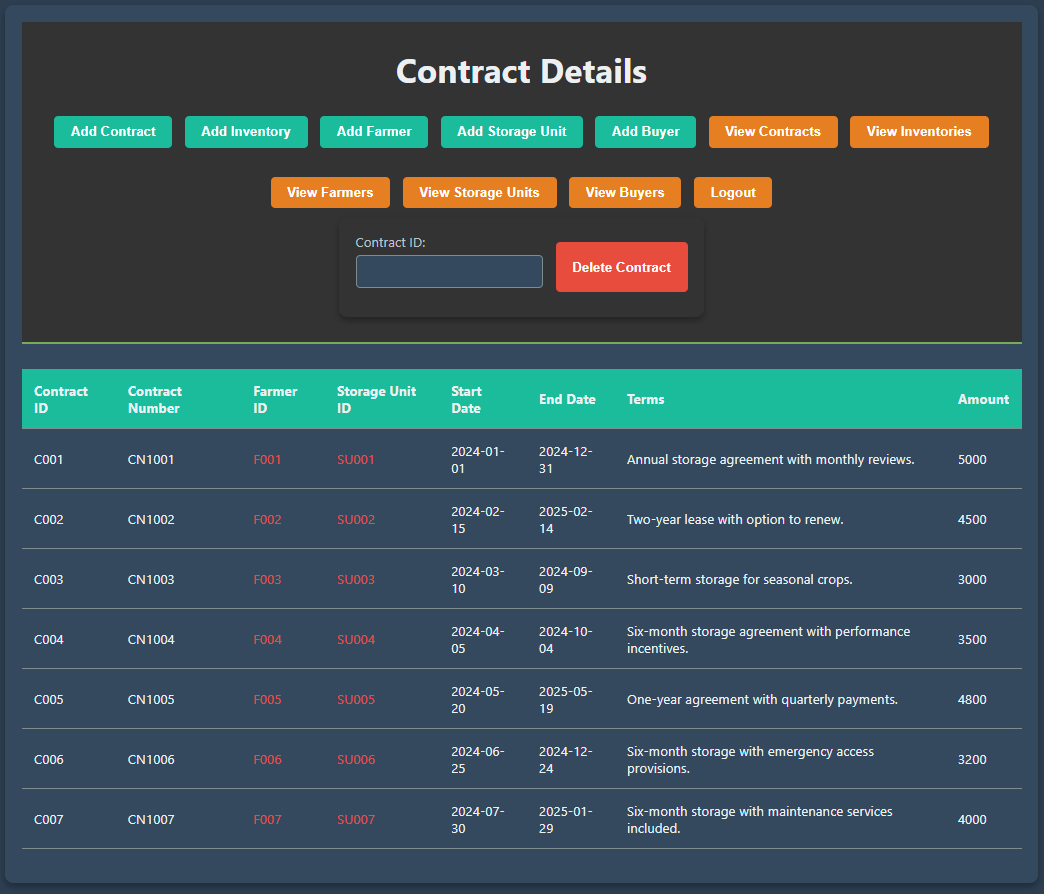
**============= END OF DELIVERABLE 4 ====================**

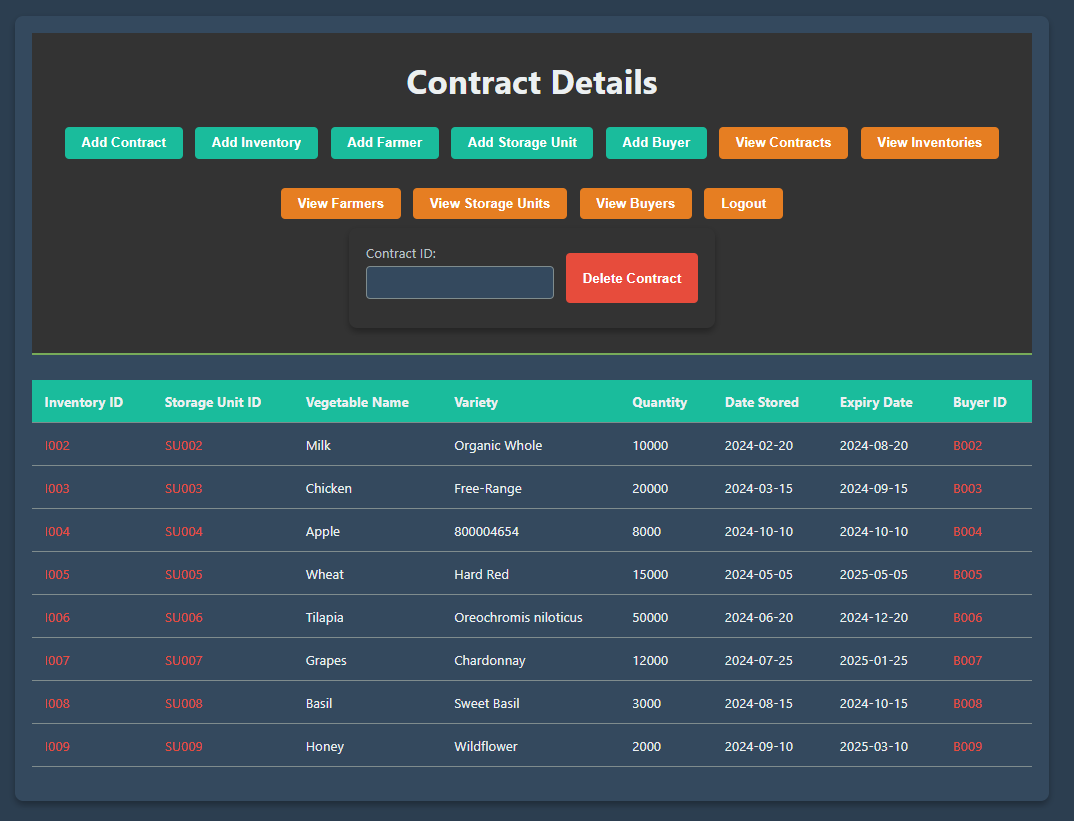
**V. FINAL DOC**

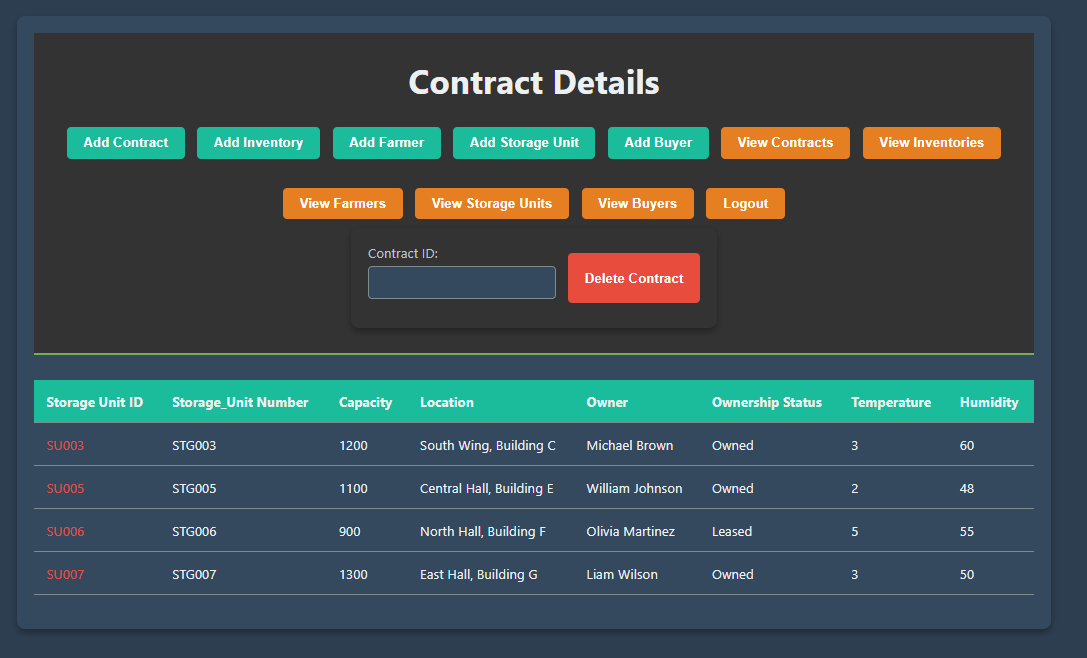
| **Task Name/ID** | **testid** | **TEST DATA** | **Expected OUTPUT** | **Actual OUTPUT** | **TEST PASS/FAIL** |
| --- | --- | --- | --- | --- | --- |
| **Add New Farmer** | **F001** | **Name: John Doe<br>Identity Number: FARM123456<br>Email:** [**john.doe@example.com**](mailto:john.doe@example.com)**<br>Phone: +1234567890** | **Farmer successfully added to database** | **Farmer FARM123456 added with name John Doe** | **PASS** |
| **Unique Email Constraint** | **F002** | **Email:** [**john.doe@example.com**](mailto:john.doe@example.com) **(duplicate)** | **Error - Duplicate email not allowed** | **Error: Duplicate email** [**john.doe@example.com**](mailto:john.doe@example.com) | **PASS** |
| **Register New Buyer** | **B001** | **Name: Alice Smith<br>Identity Number: BUYER789012<br>Email:** [**alice.smith@example.com**](mailto:alice.smith@example.com)**<br>Phone: +0987654321** | **Buyer successfully registered** | **Buyer BUYER789012 registered as Alice Smith** | **PASS** |
| **Unique Identity Number Validation** | **B002** | **Identity Number: BUYER789012 (duplicate)** | **Error - Duplicate identity number rejected** | **Error: Duplicate identity number BUYER789012** | **PASS** |
| **Create Storage Unit** | **SU001** | **Storage Number: COLD-001<br>Capacity: 1000 kg<br>Location: Warehouse A<br>Owner: Cold Storage Corp<br>Ownership Status: Owned** | **Storage unit successfully created** | **Storage unit COLD-001 created with 1000 kg capacity** | **PASS** |
| **Capacity Constraint Validation** | **SU002** | **Capacity: -500 (Invalid negative value)** | **Error - Capacity must be non-negative** | **Error: Capacity cannot be negative** | **PASS** |
| **Create New Contract** | **C001** | **Farmer ID: FARM123456<br>Storage Unit ID: COLD-001<br>Start Date: 2024-01-01<br>End Date: 2024-12-31<br>Amount: 5000.00** | **Contract successfully created** | **Contract ID 1234 created between Farmer FARM123456 and Storage Unit COLD-001** | **PASS** |
| **Contract Date Validation** | **C002** | **Start Date: 2024-12-31<br>End Date: 2024-01-01 (Invalid)** | **Error - Invalid contract dates** | **Error: End date cannot be before start date** | **PASS** |
| **Add Inventory Item** | **I001** | **Storage Unit ID: COLD-001<br>Vegetable Name: Carrots<br>Quantity: 500 kg<br>Date Stored: 2024-01-15<br>Expiry Date: 2024-02-15<br>Buyer ID: BUYER789012** | **Inventory item successfully added** | **Added 500 kg of Carrots to Storage Unit COLD-001** | **PASS** |
| **Quantity Constraint** | **I002** | **Quantity: -100 (Invalid negative value)** | **Error - Quantity must be non-negative** | **Error: Quantity cannot be negative** | **PASS** |
| **Track Schema Version** | **V001** | **Version Number: 1.0<br>Description: Initial database schema** | **Version logged successfully** | **Schema version 1.0 "Initial database schema" logged** | **PASS** |

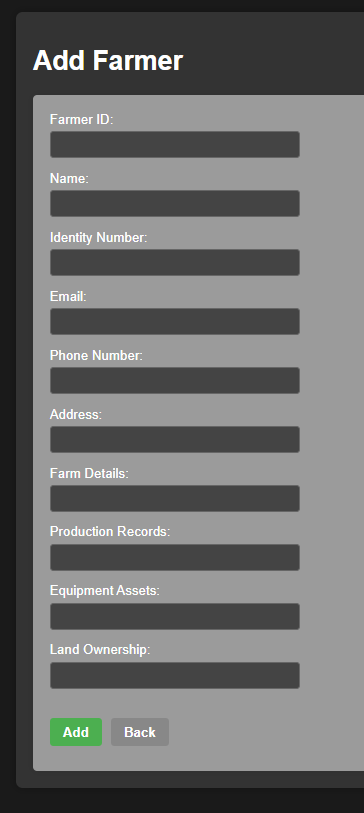
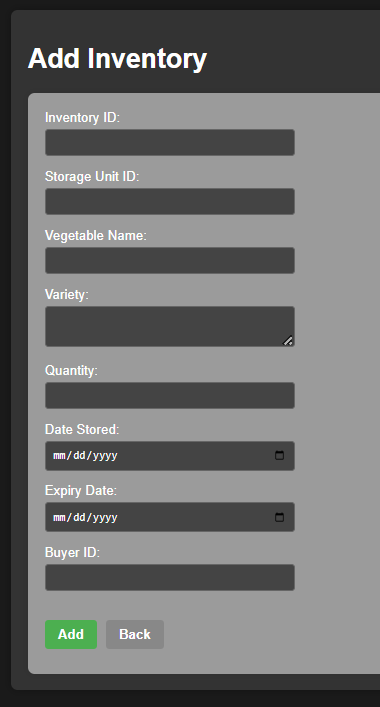
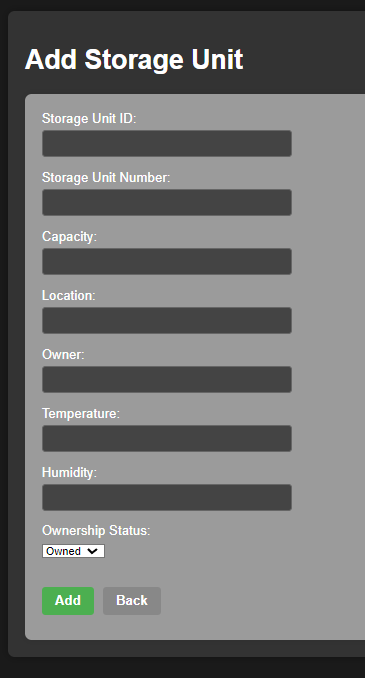
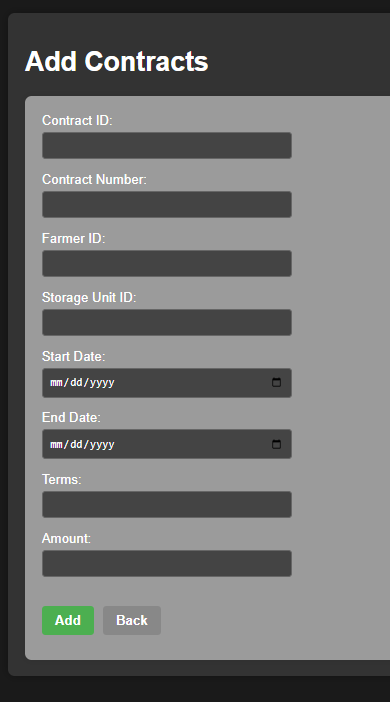
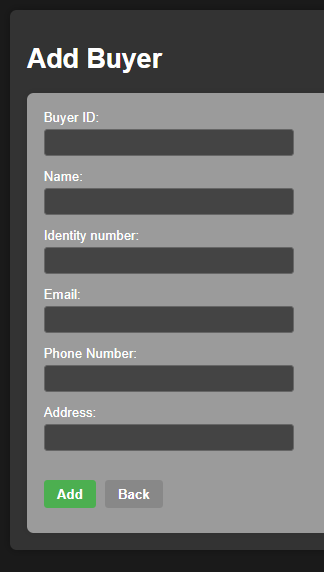
**DEMO VIDEO**

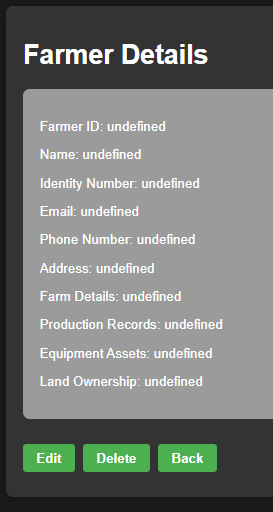
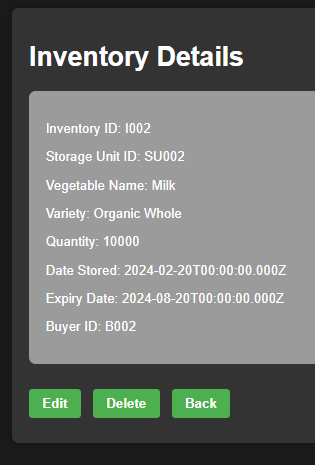
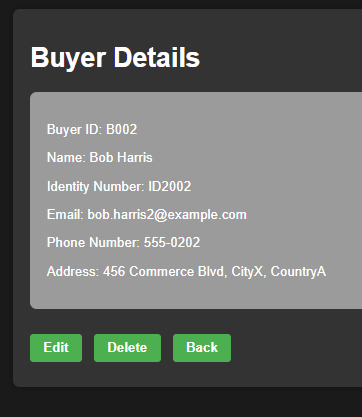
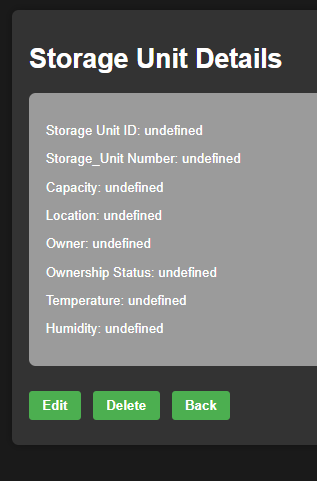
**SCREEN SHOTS OF APPLICATION:**











**Google Drive Link for Sharing Deliverables:**

[**https://drive.google.com/file/d/1FC2QT4hxzAPuYsi3rkIkP8XCkf7OcIRt/view?usp=sharing**](https://drive.google.com/file/d/1FC2QT4hxzAPuYsi3rkIkP8XCkf7OcIRt/view?usp=sharing)