## Notes

**Key requirements:**

- Product name
- Table (main) by Purpose
- Company name
- Dates
    - Start
        - Maybe made vs. received
    - End
- Still in use? (No, yes, never)
- Quantity (string)
- Description
- Test table (child of product)
- Machine table (child of product)
- Product ID (NARA)
- Product ID (Company)
- Product photo (url)
- Test sheet url
- Find a way to differentiate different items under a product
- Test success or fail? Or ongoing?
- Test dates
- Test types
- Test unique ID’s
- Connect machine table to test (test → machine)
- Materials column
- Use null to handle missing values

**Maybe:**

- Hazardous?
- Expiration date

**Additional details info examples:**

- Talk about specific products (ie. one of the items is a defect)
- Explain size metric (ie. 1.5 means that we have one whole sheet and a half sheet since it was cut)
- “Meta data is on item, see in photo”

**NOT in scope or NOT needed:**

- Records (only non-records)
- Do not dispose value
- Standard lab chemicals

## Iteration 1

**Tables:**

- **Product**
    - Product name
    - Company name
    - Quantity (string)
    - Description
    - Product ID (NARA) (Unique)
    - Product ID (Company)
    - Product photo (url)
    - Materials
    - Hazardous
    - Purpose
    - Dates:
        - Start
            - Maybe made vs. received
        - End
    - Still in use? (No, yes, never)
    - Test ID
- **Test**
    - Test ID (Unique)
    - Test name
    - Test type
    - Test dates
    - Test sheet url
    - Test success or fail? Or ongoing?
    - Product ID (reference to connect test to product)
- **Machine**
    - Machine name
    - Machine ID (Unique)
    - Status (still working)
    - Test ID (reference to connect machine to test)

**Questions:**

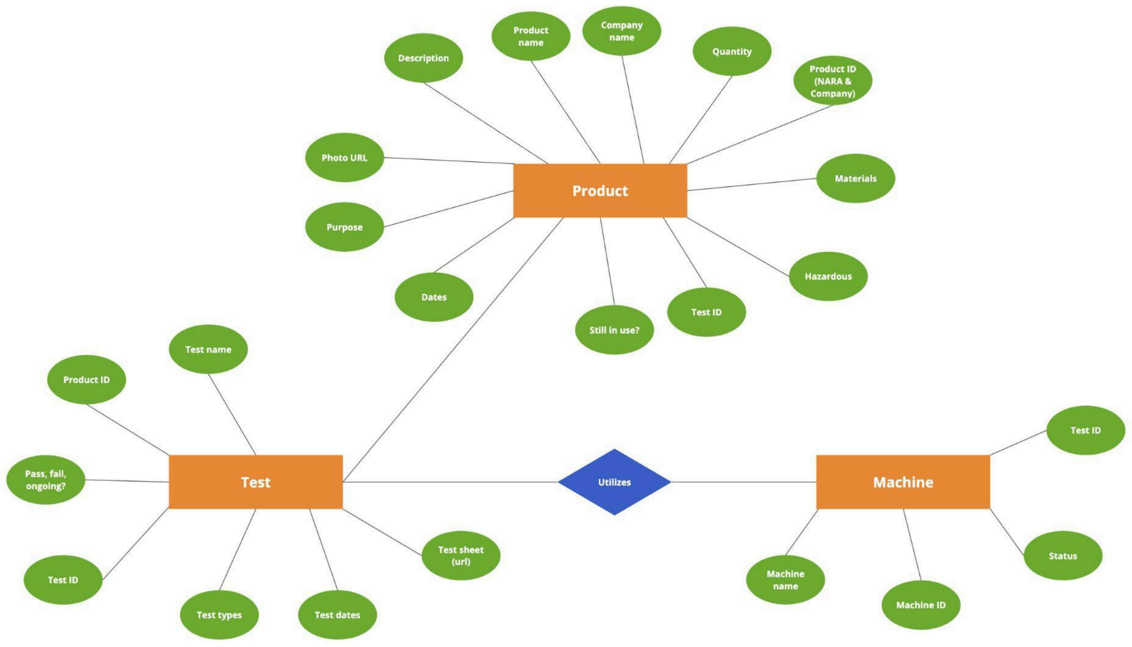
1. Can one test use multiple different products?
2. Can a test utilize multiple machines?

**Feedback:**

1. DB generated unique ID
2. Company product ID
3. Still in use (change to "Approve")
    a. Yes, no, with

        i. Detials inside descritnion (DOCUMENTATION)

4. Date
    a. Created
    b. Recieved for review
5. Change “pass, fail, ongoing, see description” to “result”
6. Add descrption col to Test table
7. Remove “test type”
8. “Machine” rename to “Instruments”
9. Remove “Status” from Machine
10. Add “descrioption” column to Machine
11. Product → test
    a. Many to many
12. Test → Instrument
    a. Many to one
        i. Instrument can be used in MANY test
        ii. Test can use ONE instrument
        iii. Ex:
            1. Company can make MANY products
            2. Product are from ONE company
            3. Arrow points to Company
13. Maybe add “tested by” column (DOCUMENTATION)



## Iteration 2

**Todo:**

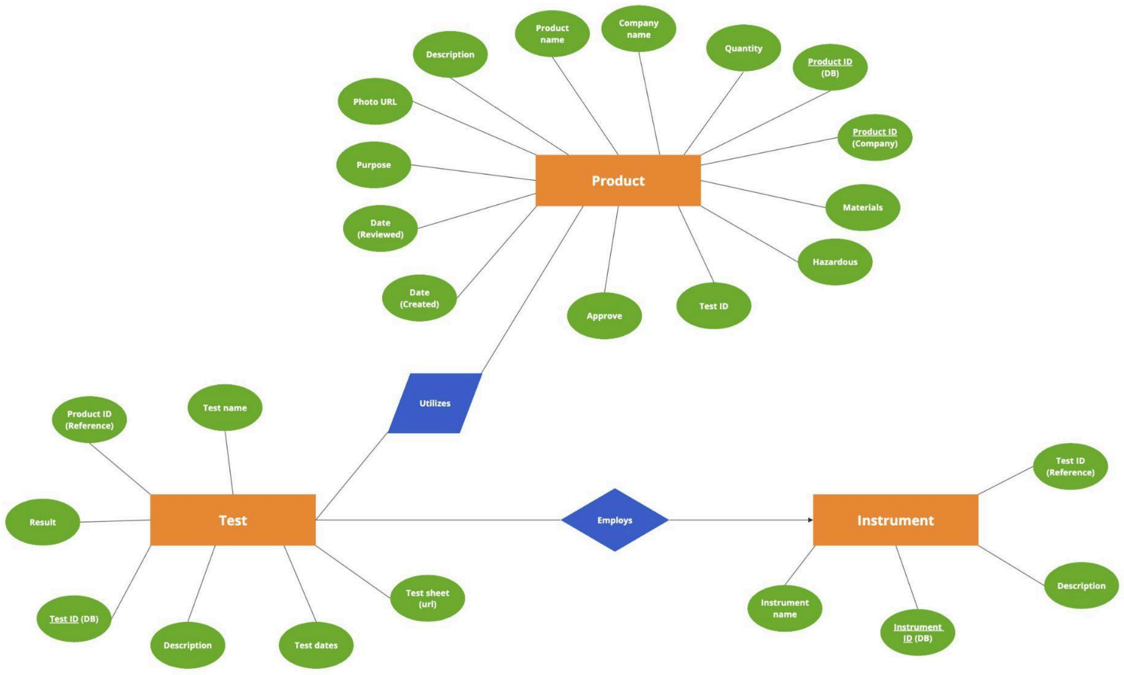
1. Incorporate feedback into ER Diagram
2. Creating tables in SQL
    a. Review feedback

**Table relationships:**

- Product → Test:
    - Verb: utilize
- Test → Machine
    - Verb: employ

**Feedback:**

1. Switch table relationships verb names



## Iteration 3

**Todo:**

1. Creating tables in SQL
    a. Review feedback
2. Identify needs to server-side and front-end

**Needs:**

1. **Server Side:**
    a. Java
    b. Spring Boot
2. **Client Side:**
    a. Javascript
    b. React

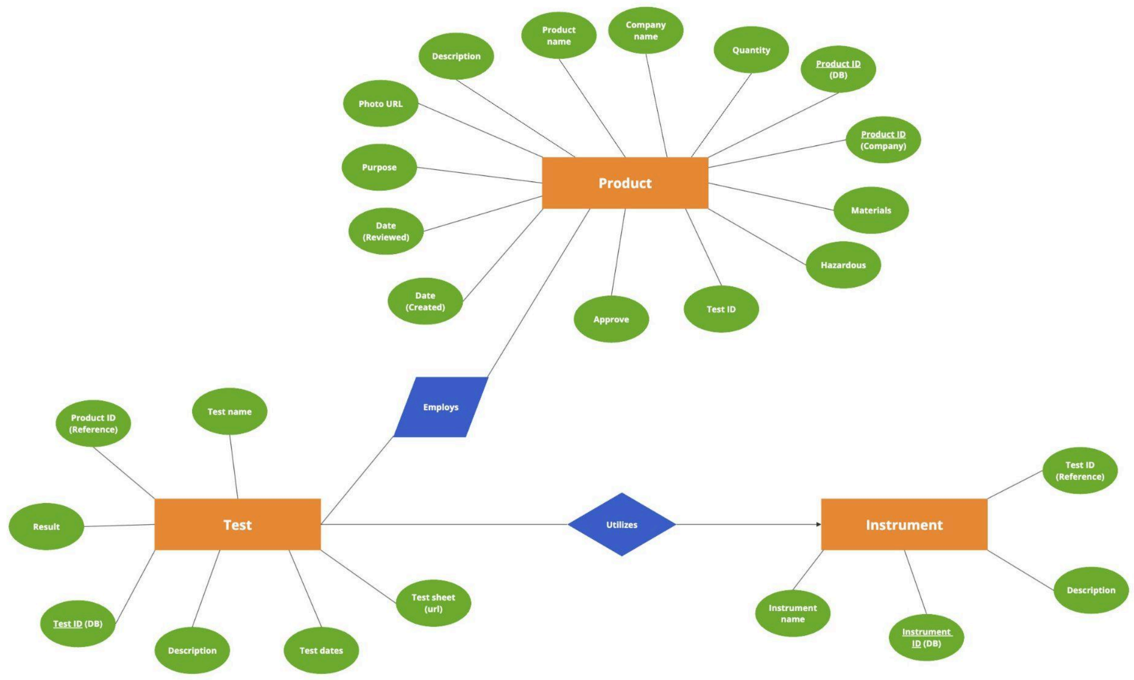
**Timeline:**

1. Create SQL Table Statements
2. **Server-side:**
    a. Import Spring Boot to Java file(s)
    b. Connect Sprint Boot to MySQL
        i. XML Dependencies
    c. Test Connection to SQL (Don't use SQL yet for url ports)
    d. Create basic models and test url ports
        i. Thunderclient or Postman
    e. Implement SQL queries into models
    f. Implement test to see if server-side ports work as intended
3. **Client-side:**
    a. Create react app
    b. Create and organize components
    c. Create navigation
        i. React-router
    d. Async fetch and call url ports to server-side to grab data
    e. Display data to React
        i. useEffect, useState

**Features:**

1. Filter parameters
    a. Product values
        i. Material type, age, manufacturer, testing data
    b. Additional testing data filter
        i. "Does it have testing data"
2. Sort Parameters
    a. Alphabetical or numerical
        i. Ex:

            1. Date
            2. Quantity
            3. Location
3. Visualize image
    a. Local
    b. Future: other computers
4. Export CSV file & filtered tuples (future)
5. Reading csv file (future)
    a. Check if its a certain test file (XRF, FTIR, etc.)
        i. GCMS is very complicated (nested graphs)
    b. Read and plot data



```
CREATE TABLE Product (
 Id INT,
 Product_Id_C VARCHAR(255),
 Materials VARCHAR(255),
 Hazardous BOOLEAN,
 Approve VARCHAR(255),
 Date_Created DATE,
 Date_Reviewed DATE,
 Purpose VARCHAR(255),
 Photo_URL VARCHAR(255),
 Descriptions VARCHAR(1500),
 Product_Name VARCHAR(255),
 Company_Name VARCHAR(255),
 Quantity VARCHAR(255)
);

CREATE TABLE Test (
 Id INT,
 Test_Name VARCHAR(255),
 Test_Date DATE,
 Descriptions VARCHAR(1500),
 Result VARCHAR(255),
 Test_Sheet VARCHAR(255),
 Instrument_Id INT REFERENCES Instrument(Id)
);

CREATE TABLE Employs (
 Product_Id INT REFERENCES Product(Id),
 Test_Id INT REFERENCES Test(Id)
);

CREATE TABLE Instrument (
 Id INT,
 Instrument_Name VARCHAR(255),
 Descriptions VARCHAR(1500)
);
```

## Iteration 4

**Todo:**

1. Prototype UI
2. Finalize server-side/client-side plans

**MVP:**

1. Add
   a. Product, Test, Machine
2. Update
   a. Product, Test, Machine
3. Remove
   a. Product, Test, Machine
4. Search

    a. Product, Test, Machine

**Needs:**

1. **Server Side:**
    a. Java JDK (18.0.1)
    b. Spring Boot (3.3.1)
    c. Maven (3.9.8)
    d. IntelliJ IDEA 2023.3.2 (Community Edition)
        i. https://www.oracle.com/java/technologies/javase/jdk18-archive-downloads.html
2. **Client Side:**
    a. Javascript
    b. React.js (18.3.1)
    c. Node.js (20.9.0)
    d. VSCode (1.90.2)

**Terms of Service:**

1. **Spring Boot**
    a. https://github.com/spring-projects/spring-boot/blob/main/LICENSE.txt
2. **Apache Maven**
    a. https://maven.apache.org/ref/3.0/license.html
3. **Node.js**
    a. https://nodesource.com/legal/product-tos
    b. https://docs.oracle.com/communications/F25434_01/docs.10/Licensing%20Manual/GUID-994D4BFE-FAED-4D42-A3A2-6B90EF444769.htm

**Justification:**

1. **Java**
    a. Java is needed as the language for the server side of the full-stack website for my intern's summer internship project at NARA. This website will be used by the Heritage Lab and will run locally on one of their computers. The server-side aspect of this application will include functionalities such as create, read, update, and delete (CRUD). These functions will manipulate a MySQL database.
    b. **New Justification:**
        i. Java is needed as the language for the server side of the CRUD full-stack website for my intern's summer internship project at NARA. He needs Java JDK 18.0.1 (or any Java JDK 17+) because Spring Boot 3.0, which has been approved in another ticket (Request #14498), requires it.
2. **Spring Boot**
    a. Spring Boot is an open-source framework typically used in server-side development. It simplifies the process of creating REST APIs for my intern's project at NARA. REST is a set of architectural constraints that allows the client side to call the website's server side. The website for my project will be used by the Heritage Science Lab and will run locally on one of their computers.

3. **Maven**
    a. The Apache Software Foundation hosts Maven, a build automation tool used in Java projects. My intern is creating a CRUD website for his summer internship project at NARA, so Maven is necessary to run Spring Boot applications, which is the framework for the server side of his CRUD application. This website will be used by the Heritage Science Lab and will run locally on one of their computers.
4. **Javascript**
    a. Javascript should already be pre-installed into the computer but I'm not too sure. Javascript is needed as the language for the client side of my intern's full-stack CRUD (create, read, update, delete) website for his summer internship project at NARA. This website will be used by the Heritage Science Lab and will run locally on one of their computers.
5. **React.js**
    a. React is an open-source front-end Javascript library. It is needed to create the user interface for the front-end of my intern's CRUD website for his summer internship project. His website will be used by the Heritage Science Lab and will run locally on one of their computers.
6. **Node.js**
    a. Node.js is an open-source runtime environment that connects the client side to the website's server side. Node is needed to use the FETCH API in React.js to call endpoints from the server side of my intern's CRUD (create, read, update, delete) website for his summer internship project at NARA. This website will be used by the Heritage Science Lab and will run locally on one of their computers.
7. **IntelliJ**
    a. IntelliJ is the IDE needed to code Java or the server-side for my intern's CRUD full-stack summer internship project. IntelliJ IDEA 2023.2 (Community Edition) or any newer version would be great to have.
8. **VSCode**
    a. VSCode is the best IDE to code JavaScript or the client-side for my intern's CRUD full-stack summer internship project. VSCode 1.90.2 or any newer version would be perfect.

## Iteration 5

**IntelliJ**

1. **Version:**
    a. IntelliJ IDEA Community Edition 2024.1.4
        i. Or any version that's 2023 or above
2. **Terms and conditions (one of these):**
    a. https://www.jetbrains.com/legal/docs/toolbox/user_community/
    b. https://www.jetbrains.com/legal/docs/toolbox/user/

**Post MVP (DOCUMENTATION):**

1. Add a next page
    a. Ex:
        i. Rule: limit is 20 results per page
        ii. 100 results → 5 pages
2. Website converts CSV test sheets to graph automatically
3. Export searched data
4. Import
5. Booking a material feature
    a. History of booking
        i. New table and database
6. Cloud, concurrency, parralelism
7. Question mark bubble for extra info

**Mock Data**

1. **Product — Test**
    a. Plastic-film — burn-test
    b. Nylon-fabric — burn-test
    c. Plastic-film — age-test
    d. Ice-cub — melt-test
2. **Test — Instrument**
    a. Burn-test — lighter
    b. Melt-test — lighter
    c. Age-test — sensor
3. **Product — Test — Instrument**
    a. Plastic-film — burn-test — lighter
    b. Nylon-fabric — burn-test — lighter
    c. Plastic-film — age-test — sensor
    d. Ice-cube — melt-test — lighter

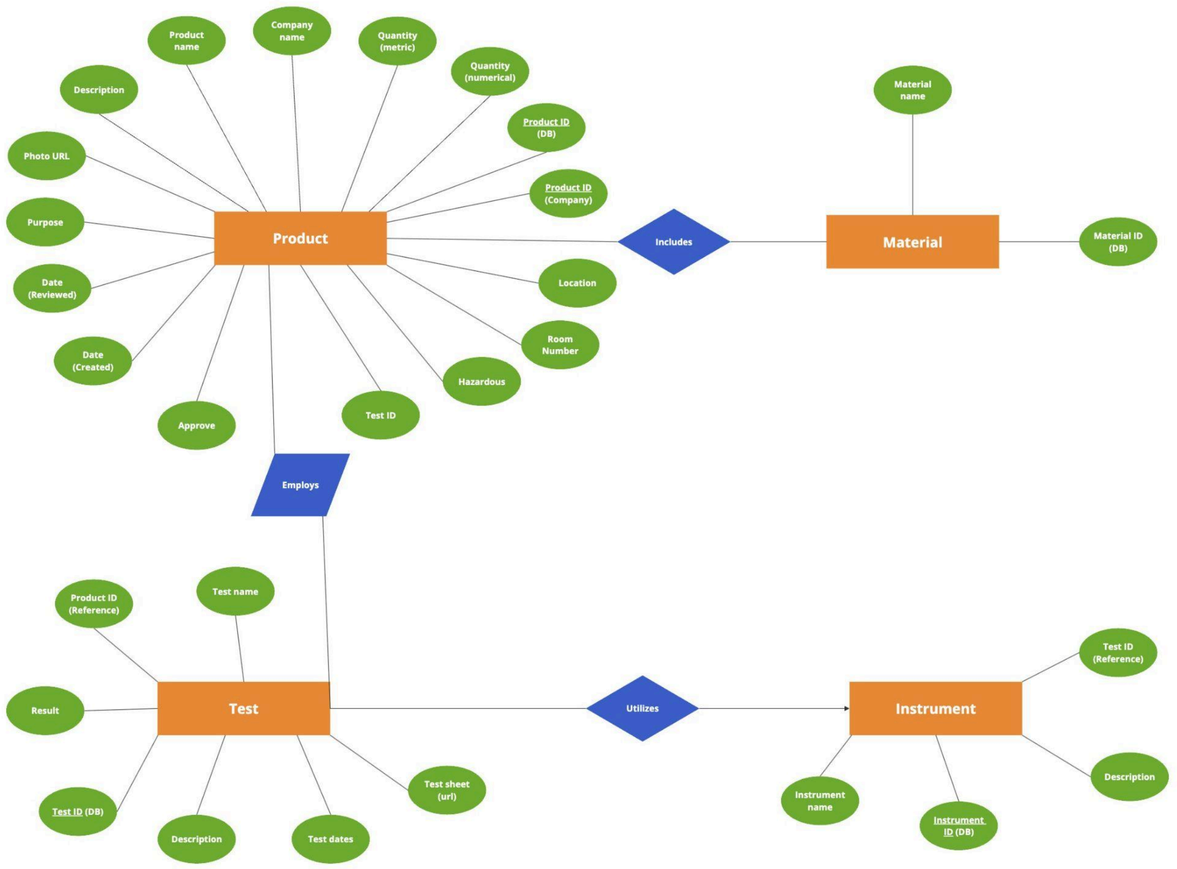
**Feedback:**

1. Two quantity columns
2. Location
    a. Two columns
        i. Room number
        ii. Location (ie. bench)
3. Multiple materials (prob materials table)
    a. Many-to-many → table
4. Change “pass, fail, ongoing, see description” to “result”
    a. Dropdown

```
CREATE DATABASE naraDB;
USE naraDB;
```

```sql
CREATE TABLE Product (
Id INT,
Product_Id_C VARCHAR(255),
Materials VARCHAR(255),
Hazardous BOOLEAN,
Approve VARCHAR(255),
Date_Created DATE,
Date_Reviewed DATE,
Purpose VARCHAR(255),
Photo_URL VARCHAR(255),
Descriptions VARCHAR(1500),
Product_Name VARCHAR(255),
Company_Name VARCHAR(255),
Quantity_Metric VARCHAR(255),
Quantity_Numeric VARCHAR(255),
Position VARCHAR(255),
Room_Number INT
);

CREATE TABLE Test (
Id INT,
Test_Name VARCHAR(255),
Test_Date DATE,
Descriptions VARCHAR(1500),
Result VARCHAR(255),
Test_Sheet VARCHAR(255),
Instrument_Id INT REFERENCES Instrument(Id)
);

CREATE TABLE Employs (
Product_Id INT REFERENCES Product(Id),
Test_Id INT REFERENCES Test(Id)
);

CREATE TABLE Instrument (
Id INT,
Instrument_Name VARCHAR(255),
Descriptions VARCHAR(1500)
);

CREATE TABLE Material (
 Id INT,
 Material_Name VARCHAR(255)
);

CREATE TABLE Includes (
 Product_Id INT REFERENCES Product(Id),
 Material_Id INT REFERENCES Material(Id)
);
```

## Iteration 6



```
CREATE TABLE naraDB;
USE naraDB;

-- Create Product Table
CREATE TABLE Product (
```

```sql
 Id INT AUTO_INCREMENT PRIMARY KEY,
 Product_Name VARCHAR(255) NOT NULL,
 Company_Name VARCHAR(255),
 Product_Id_C VARCHAR(255),
 Hazardous BOOLEAN,
 Approve VARCHAR(255),
 Date_Created DATE,
 Date_Reviewed DATE,
 Purpose VARCHAR(255),
 Photo_URL VARCHAR(255),
 Descriptions VARCHAR(1500),
 Quantity_Metric VARCHAR(255),
 Quantity_Numeric VARCHAR(255),
 Position VARCHAR(255),
 Room_Number INT
);

-- Create Test Table
CREATE TABLE Test (
 Id INT AUTO_INCREMENT PRIMARY KEY,
 Test_Name VARCHAR(255) NOT NULL,
 Test_Date DATE,
 Descriptions VARCHAR(1500),
 Result VARCHAR(255),
 Test_Sheet VARCHAR(255),
 Instrument_Id INT,
 FOREIGN KEY (Instrument_Id) REFERENCES Instrument(Id)
);

-- Create Employs Table
CREATE TABLE Employs (
 Product_Id INT,
 Test_Id INT,
 PRIMARY KEY (Product_Id, Test_Id),
 FOREIGN KEY (Product_Id) REFERENCES Product(Id),
 FOREIGN KEY (Test_Id) REFERENCES Test(Id)
);

-- Create Instrument Table
CREATE TABLE Instrument (
 Id INT AUTO_INCREMENT PRIMARY KEY,
 Instrument_Name VARCHAR(255) NOT NULL,
 Descriptions VARCHAR(1500)
);

-- Create Material Table
CREATE TABLE Material (
 Id INT AUTO_INCREMENT PRIMARY KEY,
 Material_Name VARCHAR(255) NOT NULL
);

-- Create Includes Table
CREATE TABLE Includes (
 Product_Id INT,
 Material_Id INT,
 PRIMARY KEY (Product_Id, Material_Id),
 FOREIGN KEY (Product_Id) REFERENCES Product(Id),
 FOREIGN KEY (Material_Id) REFERENCES Material(Id)
);
```

**Mock Data**

1. https://docs.google.com/spreadsheets/d/1MV663SJNImvUrOofYsP9yWN0YtNjNv2M71mKBaGLu7Y/edit?usp=sharing
    a. Info: each sheet represents a table (6 in total)

```sql
-- Insert data into Product Table
INSERT INTO Product (
 Product_Name,
 Company_Name,
 Product_Id_C,
 Hazardous,
 Approve,
 Date_Created,
 Date_Reviewed,
 Purpose,
 Photo_URL,
 Descriptions,
 Quantity_Metric,
 Quantity_Numeric,
 Position,
 Room_Number
) VALUES (
 'Plastic-film',
 'Google',
 '123nds',
 TRUE,
 'Yes',
 '2008-12-24',
 '2014-12-24',
 'Example record',
 'ksjflkasf.png',
 'Plastic is essential for recycling',
 'sheets',
 12,
 'Cabinet 2',
 1800
);

INSERT INTO Product (
 Product_Name,
 Company_Name,
 Hazardous,
 Approve,
 Date_Reviewed,
 Purpose,
 Descriptions,
 Quantity_Metric,
 Quantity_Numeric,
 Position,
 Room_Number
) VALUES (
 'Nylon-fabric',
 'Apple',
 FALSE,
 'With',
 '2020-09-18',
 'Exhibition',
 'Approved with blah blah blah',
 'in^2',
 30,
 'Drawer 5',
 1800
);

INSERT INTO Product (
 Product_Name,
 Company_Name,
 Product_Id_C,
 Purpose,
 Photo_URL,
 Descriptions,
 Quantity_Metric,
 Quantity_Numeric,
 Position,
 Room_Number
) VALUES (
 'Ice-cube',
 'Antartica',
 '987dfac',
 'Conservation treatment',
 '984289ufasdfas.jpg',
 'This is frozen water',
 'oz',
 24,
 'Shelve 9',
 1600
);

-- Insert data into Test Table
INSERT INTO Test (
 Test_Name,
 Test_Date,
 Descriptions,
 Result,
 Test_Sheet,
 Instrument_Id
) VALUES (
 'Burn-test',
 '2024-07-16',
 'They burnt stuff',
 'pass',
 '282oadfnd.csv',
 1
);

INSERT INTO Test (
 Test_Name,
```

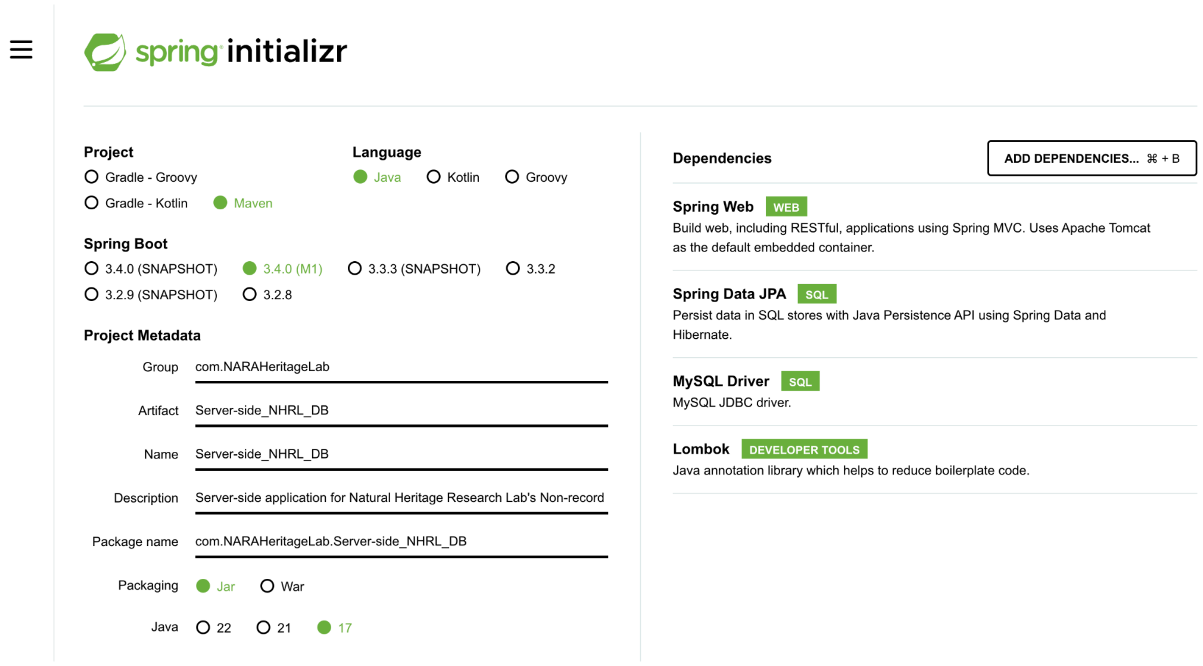
```sql
 Descriptions,
 Result,
 Instrument_Id
) VALUES (
 'Age-test',
 'They aged things',
 'ongoing',
 2
);

INSERT INTO Test (
 Test_Name,
 Test_Date,
 Descriptions,
 Result,
 Instrument_Id
) VALUES (
 'Melt-test',
 '1990-04-01',
 'They melted ice',
 'see description',
 1
);

INSERT INTO Product (
 Product_Name,
 Company_Name,
 Hazardous,
 Approve,
 Date_Created,
 Purpose,
 Descriptions,
 Quantity_Metric,
 Quantity_Numeric,
 Position,
 Room_Number
) VALUES (
 'Cardboard-box',
 'Microsoft',
 FALSE,
 'No',
 '2019-10-11',
 'Holdings maintenance',
 'This is a literal cardboard box',
 'box',
 2,
 'Counter 1',
 1500
);

-- Insert data into ProductTest Table
INSERT INTO ProductTest (Product_Id, Test_Id) VALUES (1, 1);
INSERT INTO ProductTest (Product_Id, Test_Id) VALUES (2, 1);
INSERT INTO ProductTest (Product_Id, Test_Id) VALUES (1, 2);
INSERT INTO ProductTest (Product_Id, Test_Id) VALUES (3, 3);

-- Insert data into Instrument Table
INSERT INTO Instrument (Instrument_Name, Descriptions) VALUES ('Lighter', 'Lights
stuff on fire');
INSERT INTO Instrument (Instrument_Name, Descriptions) VALUES ('Sensor', 'Senses
temperature');

-- Insert data into Material Table
INSERT INTO Material (Material_Name) VALUES ('Plastic');
INSERT INTO Material (Material_Name) VALUES ('Paper');
INSERT INTO Material (Material_Name) VALUES ('Nylon');
INSERT INTO Material (Material_Name) VALUES ('Water');

-- Insert data into ProductMaterial Table
INSERT INTO ProductMaterial (Product_Id, Material_Id) VALUES (1, 1);
INSERT INTO ProductMaterial (Product_Id, Material_Id) VALUES (1, 2);
INSERT INTO ProductMaterial (Product_Id, Material_Id) VALUES (2, 1);
INSERT INTO ProductMaterial (Product_Id, Material_Id) VALUES (2, 3);
INSERT INTO ProductMaterial (Product_Id, Material_Id) VALUES (3, 4);
```

## Iteration 7

**Common errors:**

1. Create database table
   a. CREATE DATABASE naraDB;

2. Unable to access database in MySQL:
    a. USE naraDB;
3. Unable to update or remove from tables in MySQL:
    a. SET SQL_SAFE_UPDATES = 0;
    b. SET FOREIGN_KEY_CHECKS = 0;
        i. SET FOREIGN_KEY_CHECKS = 1;
            1. Run this line after you remove table



Project: ○ Gradle - Groovy ○ Gradle - Kotlin [✓ Maven]

Language: [✓ Java] ○ Kotlin ○ Groovy

Spring Boot: ○ 3.4.0 (SNAPSHOT) [✓ 3.4.0 (M1)] ○ 3.3.3 (SNAPSHOT) ○ 3.3.2 ○ 3.2.9 (SNAPSHOT) ○ 3.2.8

Project Metadata

|  |  |
|---|---|
| Group | com.NARAHeritageLab |
| Artifact | Server-side_NHRL_DB |
| Name | Server-side_NHRL_DB |
| Description | Server-side application for Natural Heritage Research Lab's Non-record |
| Package name | com.NARAHeritageLab.Server-side_NHRL_DB |
| Packaging | [✓ Jar] ○ War |
| Java | ○ 22 ○ 21 [✓ 17] |

Dependencies — ADD DEPENDENCIES... ⌘ + B

**Spring Web** WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

**Spring Data JPA** SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

**MySQL Driver** SQL
MySQL JDBC driver.

**Lombok** DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

**Steps:**

1. Picture above
2. Check dependencies in xml file to confirm
3. Create folders inside com.example
    a. Controller, entity (or model), repository, exception
4. Initilize connection with MySQL in application.properties

## Iteration 8

```
spring.application.name=Server-side_NHRL_DB
spring.datasource.url=jdbc:mysql://localhost:3306/naraDB
spring.datasource.username=root
spring.datasource.password=NaraDB1804#

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```