# CS 121 Final Project Reflection

**Due: Saturday March 16th, 11:59PM PST**

*Note: This specification is currently for 23wi, subject to minor changes for 24wi.*

This reflection document will include written portions of the Final Project. Submit this as **`reflection.pdf`** with the rest of the required files for your submission on CodePost.

For ease, we have highlighted any parts which require answers in **blue**.

**Student name(s): Matthew Casertano, Luke Kottom**

**Student email(s): mcaserta@caltech.edu, lkottom@caltech.edu**

---

## Part L0. Introduction

Answer the following questions to introduce us to your database and application; you may pull anything relevant from your Project Proposal and/or README.

### Database/Application Overview

What application did you design and implement? What was the motivation for your application? What was the dataset and rationale behind finding your dataset?

***Database and Application Overview Answer (3-4 sentences) :***

***Our database powers our application which is a military equipment procurement system, handling users, manufacturers, customers, equipment, and orders, filled largely with real-world data as a demonstration. It keeps everything structured and ensures clients (military branches of the U.S. or other countries) and admins (manufacturers) only have access to what they need. Clients can log in, browse inventory, and place orders, while admins manage stock, approve purchases, and keep things running smoothly. Everything updates in real time, when an order is placed, stock adjusts instantly, and when an admin approves or rejects it, the changes reflect immediately.***

***Data set (general or specific) Answer:***

***The starting point for our data was two Wikipedia pages, specifically https://en.wikipedia.org/wiki/List_of_equipment_of_the_United_States_Army and https://en.wikipedia.org/wiki/List_of_active_United_States_military_aircraft***

***We scraped this data, taking land vehicles from the first link with get_vehicle_data.py (combining the artillery and vehicle tables), and aircraft from the second link with get_air_data.py. We then merged the data and augmented it with AI-generated descriptions for each piece of equipment and did some data manipulation (e.g. removing duplicates) to form our equipment.csv with integrate.py. We also had to use***

***AI to get the manufacturers for each vehicle as it was not provided in the original Wikipedia dataset (unlike for aircraft, where the manufacturer was provided).***

***Next, we generated users, customers, and manufacturers (create_users.py). Users were divided into manufacturers and customers, with manufacturers representing major defense companies and customers simulating government agencies, military branches, and defense contractors. The manufacturers' data was based on the list of unique manufacturers from our Wikipedia-based equipment data, while customers were synthetically generated in Python, assigning them randomized names, email addresses, countries, and security clearances based on real world possibilities. All users were assigned with the same password "password123" and random salts.***

***Finally, we generated orders (create_orders.py). This was also done with Python. We randomly assigned customers to military equipment purchases while ensuring realistic spending behaviors based on equipment price and category. Each order was timestamped and included quantities, pricing, and order status. Additionally, we ensured valid relationships between users, equipment, & orders, preventing inconsistencies in the data.***

***Client user(s) Answer***:

**The client user interaction is designed for military branches of the U.S. or foreign countries to browse and purchase military equipment from manufacturers. Users can log in, view available equipment, and place orders. They can view their order history and have a defined identity so only they can log in and have access/permission over their orders etc.. Additionally, users have restricted permissions, ensuring they can only browse inventory, place orders, and review past transactions. Orders will have to be approved by the companies (admin) and once approved they immediately update for the user.**

***Admin user(s) Answer:***

***The admin interface is built for manufacturers to manage their military equipment inventory. Admins can add, update, and remove equipment, view sales data, and process orders. They have full control over their listings, ensuring accurate pricing and stock levels. It's all for a specific admin identity so you can only manage your own products as an admin. Unlike clients, admins can see order details to approve or reject purchases, and adjust inventory. The system allows quick updates for example when the admin approves the client immediately can see that. Once the client places an order the quantity immediately goes down in anticipation of approval (so no stock is double requested) but if the admin rejects that the quantity immediately goes back up in the equipment database. So, everything updates live.***

## Part A. ER Diagrams

As we've practiced these past few weeks, the ER model is essential for designing your database application, and we expect you to iterate upon your design as you work through the ER and implementation steps. In this answer, you should provide a full ER diagram of your system. Your grade will be based on correct representation of the ER model as well as readability, consistency, and organization.

**Notes:** For this section **only**, we will allow (and encourage) students to share their diagrams on Discord (**#er-diagram-feedback**) to get feedback from other students on their ER diagrams given a brief summary of your dataset and domain requirements. This is offered as an opportunity to test your ER diagrams for accuracy and robustness, as another pair of eyes can sometimes catch constraints that are not satisfied or which are inconsistent with your specified domain requirements.

**Requirements:**

- Entity sets, relationship sets, and weak entity sets should be properly represented (also, do not use ER symbols not taught in class)
- Mapping cardinalities should be appropriate for your database schema, and in sync with your DDL
- Participation constraints should be appropriate and in sync with your DDL (total, partial, numeric)
- Use specialization where appropriate (e.g. *purchasers* and *travelers* inheriting from a *customers* specialization in A6)
- Do not use degrees greater than 3 in your relationships, do not use more than one arrow in ternary relationships.
- Use descriptive attributes appropriately
- Underline primary keys and dotted-underline discriminators
- Expectations from A6 still apply here
- Note: You do not need ER diagrams for views

**ER Diagrams:**

**CS 121 Class Diagram.vsdx**

## Part B. DDL (Indexes)

As mentioned in Part B, you will need to add at least one index at the bottom of your **`setup.sql`** and show that it makes a performance benefit for some query(s).

Here, describe your process for choosing your index(es) and show that it is used by at least one query, which speeds up the performance of the same query on a version of the same table without that index. You may find lec14-analysis.sql and Lecture 14 slides on indexes useful for strategies to choose and test your indexes. **Remember that indexes are already created in MySQL for PKs and FKs, so you should not be recreating these.**

***Index(es):***

***I created an index on the price_usd column to improve performance for price range queries (for customers who want to see military equipment within a certain price range).***

***Justification and Performance Testing Results:***

***Before adding the index, SQL had to scan the entire equipment table to find items within a specific price range, using a scan of type=ALL as shown in the EXPLAIN output below. After adding the index, SQL can use an efficient scan (type=range) that can help enable it to save a lot of time. This optimization is especially important as the database grows, since it uses a B+ Tree (as explained in lecture) which can reduce the complexity from O(n) to O(log n) based on tree structure.***

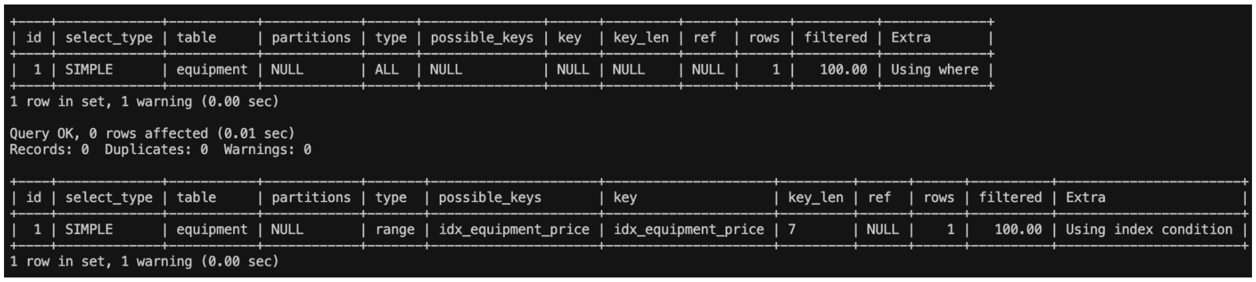
***SQL:***

***EXPLAIN SELECT * FROM equipment WHERE price_usd BETWEEN 10000000 AND 50000000;***

***CREATE INDEX idx_equipment_price ON equipment(price_usd);***

***EXPLAIN SELECT * FROM equipment WHERE price_usd BETWEEN 10000000 AND 50000000;***

***Output:***



```
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table     | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | equipment | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    1 |   100.00 | Using where |
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

+----+-------------+-----------+------------+-------+---------------------+---------------------+---------+------+------+----------+-----------------------+
| id | select_type | table     | partitions | type  | possible_keys       | key                 | key_len | ref  | rows | filtered | Extra                 |
+----+-------------+-----------+------------+-------+---------------------+---------------------+---------+------+------+----------+-----------------------+
|  1 | SIMPLE      | equipment | NULL       | range | idx_equipment_price | idx_equipment_price | 7       | NULL |    1 |   100.00 | Using index condition |
+----+-------------+-----------+------------+-------+---------------------+---------------------+---------+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)
```

## Part C. Functional Dependencies and Normal Forms

**Note:** For 24wi, this part will be optional, but students can earn up to 8 additional points for answers. A comprehensive slide deck is provided on Canvas covering Functional Dependencies and Normal Forms (the most important Normal Form to know about is BCNF; 3NF and 4NF are included for additional material).

**Requirements (from Final Project specification):**

- What is the purpose of a functional dependency? Why is it relevant to database design?
- Why is BCNF ideal for a database schema? Your answer should include a specific reference to the slides and/or supplementary recording (e.g. we don't want to see an answer copied from a Google search; we want you to be able to answer this and ask questions if you are unsure!)
- Identify *at least 2 non-trivial functional dependencies* in your database
- Choose and justify your decision for the normal form(s) used in your database for at least 2 tables (if you have more, we will not require extra work, but will be more lenient with small errors). BCNF and 3NF will be the more common NF's expected, 4NF is also fine (but not 1NF).
  - Your justification will be strengthened with a discussion of your dataset breakdown, which we expect you to run into trade-offs of redundancy and performance.
- For up to two of your relations having at least 3 attributes (each) and at least one functional dependency, prove that they are in your chosen NF, using similar methods from the BCNF assignment.
  - If you have identified functional dependencies which are not preserved under a BCNF decomposition, this is fine

**Purpose/Relevance of Functional Dependencies:**

**Purpose/Relevance of BCNF:**

**Identified Functional Dependencies:**

**Normal Forms Used (with Brief Justifications):**

**NF Proof 1:**

**NF Proof 2:**

---

## Part G. Relational Algebra

**Requirements (from Final Project specification, Part G):**

- Minimum of 3 non-trivial queries (e.g. no queries simply in the form `SELECT <x> FROM <y>`)
- At least 1 group by with aggregation
- At least 3 joins (across a minimum of 2 queries)
- At least 1 update, insert, and/or delete
  - This may be equivalent to said SQL statements elsewhere (e.g. queries or procedural code), but are not required to be; in other words, you can write these independent of other sections
- Appropriate projection/extended projection use
- Computed attributes should be renamed appropriately
- Part of your grade will come from overall demonstration of relational algebra in the context of your schemas; obviously minimal effort will be ineligible for full credit; it is difficult to formally define "obviously minimal", but refer to A1 and the midterm for examples of what we're looking for
- Above each query, briefly describe what it is computing; we will use this to grade for correctness based on what the query is supposed to compute; lack of descriptions will result in deductions, since we have no idea otherwise of what the query is intended to do.

Below, provide each of your RA queries following their respective description.

Query 1: Shows sales performance by equipment from manufacturer 1, showing equipment name, category, total units sold, and revenue for approved orders only.

$$temp1 \leftarrow \sigma_{manufacturer\_id=1}(equipment)$$

$$temp2 \leftarrow \sigma_{status=\text{"approved"}}(orders)$$

$$temp3 \leftarrow temp1 \bowtie temp2$$

$${}_{equipment\_id}\mathcal{G}_{name,\ equipment\_category,\ sum(order\_quantity)\ as\ total\_units,\ sum(total\_amount\_usd)\ as\ total\_revenue}(temp3)$$

Query 2: Lists all orders for equipment from manufacturer 1 with complete details including order ID, date, customer name, equipment name, quantity, amount, and status.

$temp1 \leftarrow orders \bowtie equipment$

$temp2 \leftarrow temp1 \bowtie customers$

$temp3 \leftarrow temp2 \bowtie users$

$temp4 \leftarrow \sigma_{manufacturer\_id = 1}(temp3)$

$\Pi_{order\_id,\ order\_date,\ users.name\ as\ customers\_name,\ equipment.name\ as\ equipment\_name,\ order\_quantity,\ total\_amount\_usd,\ status}(temp4)$

Query 3: Sets the stock level to 8 for equipment ID 1 from manufacturer 1 while preserving all other equipment records.

$equipment \leftarrow \Pi_{equipment\_id,\ manufacturer\_id,\ equipment\_category,\ 8\ as\ stock\ description,\ name,\ specialty,\ price\_usd}$
$(\sigma_{equipment\_id=1\ \cap\ manufacturer\_id=1}(equipment))$
$\cup$
$\sigma_{equipment\_id \neq 1\ \cup\ manufacturer\_id \neq 1}(equipment)$

## Part L1. Written Reflection Responses

### Challenges and Limitations

List any problems (at least one) that came up in the design and implementation of your database/application (minimum 2-3 sentences)

***Answer:***
***The biggest challenge for us was error handling across our interfaces (client and admin). We built a huge system with a lot of different options and a very extensive flow for users, so we had to account for lots of possible user inputs and make sure that we flowed/connected everything correctly between all the different possible interfaces. Part of this challenge was with making sure that the two applications could "talk to each other" and update live based on user interaction and we had to implement triggers to make this happen.***

### Future work

If you are particularly eager for a certain application (have your own start-up in mind?), it is easy to over-scope a final project, especially one that isn't a term-long project. You can list any stretch goals you might have "if you had the time" which staff can help give feedback on prioritizing (2-3 sentences).

***Answer:***

***If we had more time, we would first want to develop a user verification process that could for example require some kind of official government credentials to ensure that only legitimate users can make an account to access our platform (currently, we automatically verify every user so we have the foundation for doing this built out just not the verification process itself). We also would want to expand our web scraper to also pull images of the equipment from Wikipedia that could be integrated into our table such that customers can see photos of the equipment they are interested in. Finally, we would want to build out more sales statistics so that the user could see more information about their sales such as maybe breaking it down by the last month/day/week instead of just total sales.***

### Self-Evaluation

What is your expected grade for the Final Project (out of 100)? Justify what you think are the strongest points, especially pointing to demonstrated improvement in areas that may have had lower scores in assignments throughout the term. Also provide any notes here on areas that could be improved (and what you would do differently next time).

***Answer: 95***
***I think we worked really hard on it and made a great application that completes all requirements needed with solid data and user interaction, but we also think that our ER diagram might be off and maybe a few other small things as well could be off.***

What is your expected grade for the course overall? A

## Collaboration (Required for Partners)

This section is required for projects which involved partner work. Each partner should include 2-3 sentences identifying the amount of time they spent working on the project, as well as their specific responsibilities and overall experience working with a partner in this project.

***Partner 1 Name: Luke Kottom***
***Partner 1 Responsibilities and Reflection:***
***Worked about 25 hours***
***Did the bulk of the work on app_client.py and on setting up the interface and debugging various issues that came up***
***Did most of the initial work on the ER diagram and flowchart***
***Worked closely with Prof. Hovik across many meetings***

***Partner 2 Name: Matthew Casertano***
***Partner 2 Responsibilities and Reflection:***
***Worked about 20 hours***
***Took care of data collection and creating SQL data tables to be imported based on real-world and synthetic data***
***Worked on app_admin.py and creating the admin interface interaction***
***Worked on the ER diagram and flowchart***
***Attended several meetings with Prof. Hovik***

## Other comments

This is the first time CS 121 has had a Final Project, and we would appreciate your feedback on whether you would recommend this in future terms, as well as what you found most helpful, and what you might find helpful to change.

***Answer:***

***We think it was a great opportunity to try our newfound skills in SQL on a real-world project and really enjoyed building out the project.***

***We found most helpful the opportunity to interact and ask questions to Prof. Hovik, which we probably had more of because we started very early.***

***We thought it might be helpful to clarify more clearly the ER diagrams and provide more examples as that was a big source of confusion for us, although we were able to figure it out by working with Prof. Hovik. Also, we spent WAY more time than was allotted on this document so we felt that the time estimates were far too low (although part of it could be that our project was maybe more complicated than was expected).***