**University of British Columbia, Vancouver**
Department of Computer Science

# CPSC 304 Project Cover Page

Milestone #: 4

Date: 11/29/2024

Group Number: 91

| Name | Student Number | CS Alias (Userid) | Preferred E-mail Address |
|---|---|---|---|
| Jerry Qi | 73382533 | s6l7k | zqi07@students.cs.ubc.ca |
| Rapeewit Chanprakaisi | 57529208 | w2g6k | rchanpra@student.ubc.ca |
| Eric Xiang | 90612029 | g9e4y | ericxiang8@hotmail.com |

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

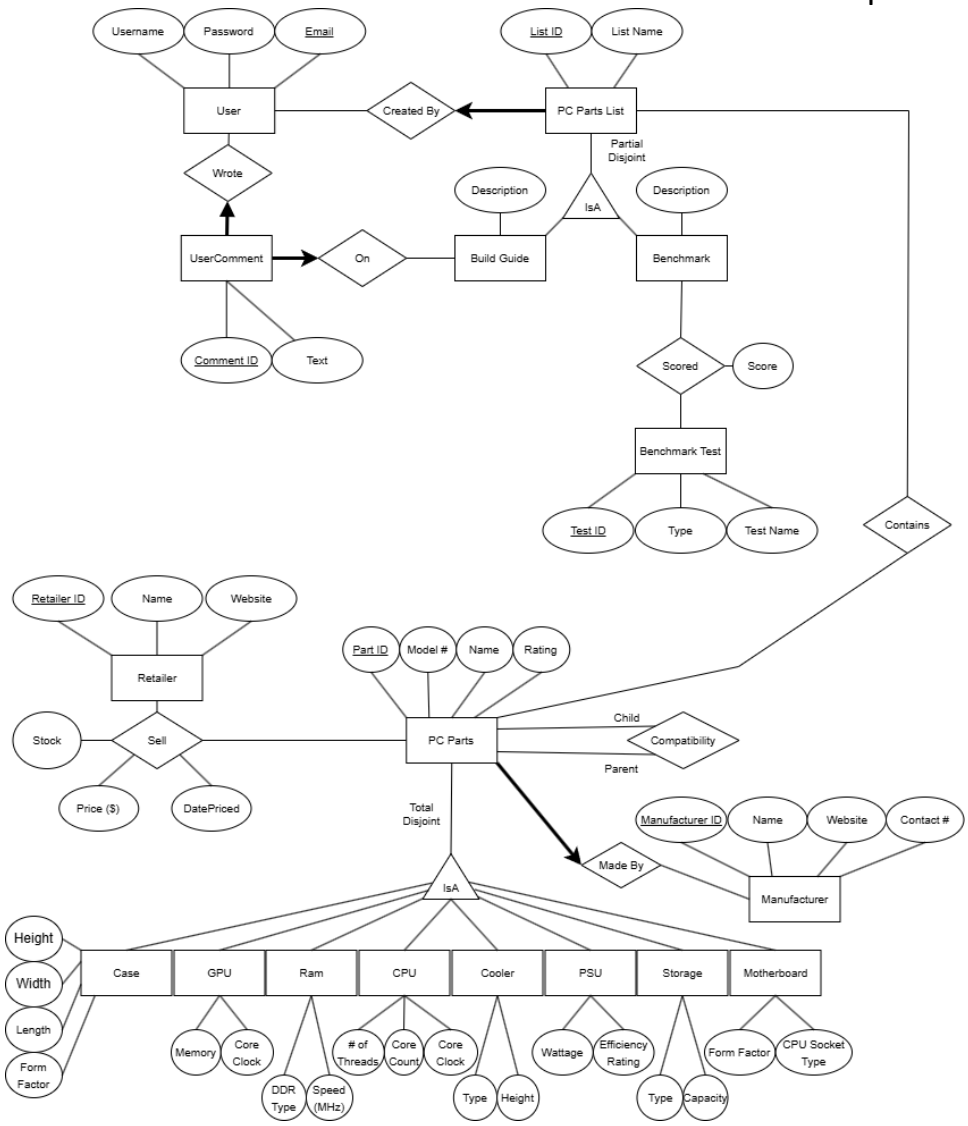
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

**University of British Columbia, Vancouver**
Department of Computer Science

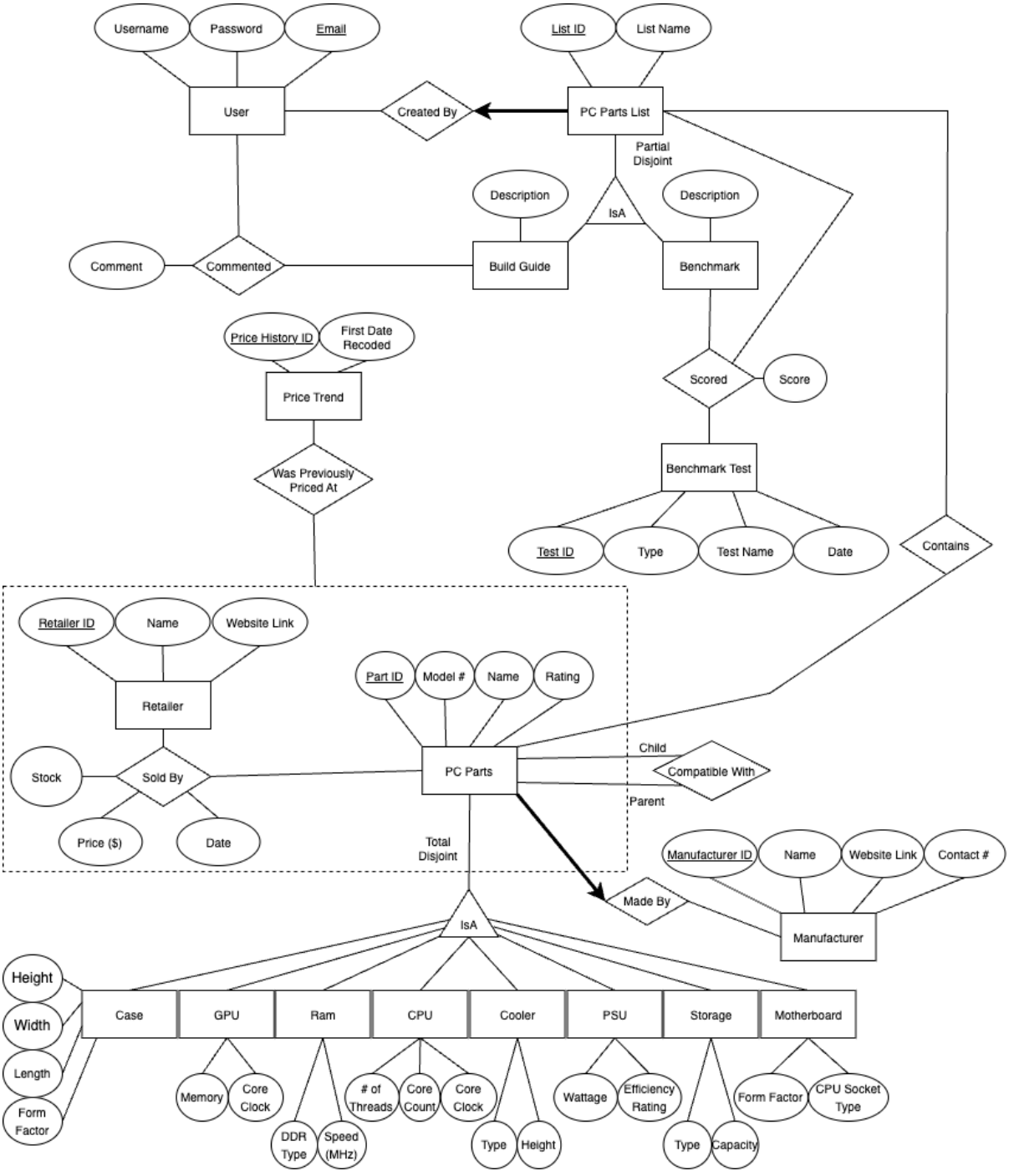
---

1. **Short description of the final project, and what it accomplished.**
   Our project is a PC parts picker that assists users in building a custom PC. The domain of our project is custom PC building/computer hardware management/shopping assistant. The database will provide functionality that allows users to search, filter, and select computer hardware components for custom PC builds based on specific criteria, ensure component compatibility checks, and allow users to comment on each product with availability from various retailers.

2. **Description of how your final schema differed from the schema you turned in. If the final schema differed, explain why.**
   Our final schema differed from the schema we turned in previously.



Below is the old schema:




We removed the “was Previously Priced at” relationship and Price Trend entity because aggregation is bit complicated to handle, we instead extended the comment part of our schema, allowing us to keep records of comments more easily. We also removed Date attribute from Benchmark Test since it’s not allowed.

---

## 3. List of all SQL queries used to satisfy the rubric items and where each query can be found in the code (file name and line number(s)).

All our SQL queries can be found in the appService.js file located in the root folder:

2.1.1 INSERT (Line number 158)

`INSERT INTO Contain (ListID, PartID) VALUES (:ListID, :PartID)`

2.1.2 UPDATE (Line number 183)

`UPDATE PCParts SET Name=:Name, Model=:Model, Rating=:Rating, ManufacturerID=:ManufacturerID where PartID=:PartID`

2.1.3 DELETE (Line number 208)

'Select * FROM PCPartsList c WHERE c.ListID=:ListID'

`DELETE FROM Contain WHERE ListID=:ListID AND PartID =:PartID`

2.1.4 Selection (Line number 235)

`SELECT p.PartID, p.Name, p.Model, p.Rating, p.ManufacturerID, s.Price, s.DatePriced, r.RetailerID, r.Name, r.Website

FROM Sell s

JOIN Retailer r ON r.RetailerID=s.RetailerID

JOIN PCParts p ON p.PartID=s.PartID

WHERE ${string}`

---

2.1.5 Projection (Line number 251)

`

SELECT ${attributes}'

FROM Retailer

`

2.1.6 Join (Line number 427)

`Select cp.PartID, Name, Model, Rating, ManufacturerID

FROM (Select * FROM Contain c WHERE c.ListID=:ListID) cp

JOIN PCParts p ON p.PartID=cp.PartID`

2.1.7 Aggregation with GROUP BY (Line number 286)

`

```
SELECT x.ManufacturerID, x.Name, AVG(x.Rating)
FROM (SELECT p.Rating, p.ManufacturerID, m.Name
    FROM PcParts p
    JOIN Manufacturer m ON p.ManufacturerID=m.ManufacturerID) x
GROUP BY x.ManufacturerID, x.Name
```

`

This query calculates the average rating of products (from the PcParts table) grouped by each manufacturer (identified by ManufacturerID and Name) in the Manufacturer table. It joins the two tables to associate product data with the respective manufacturers.

2.1.8 Aggregation with HAVING (Line number 305)

```
SELECT ManufacturerID, Name, COUNT(PartID), MIN(Rating)
FROM (SELECT p.PartID, p.Rating, p.ManufacturerID, m.Name
    FROM PcParts p
    JOIN Manufacturer m ON p.ManufacturerID=m.ManufacturerID)
GROUP BY ManufacturerID, Name
HAVING MIN(Rating) >= :rating
```

This query retrieves the ManufacturerID, Name, the number of parts (COUNT(PartID)), and the minimum rating (MIN(Rating)) for each manufacturer whose lowest-rated product has a rating greater than or equal to a specified threshold (:rating). It groups the results by manufacturer.

---

2.1.9 Nested aggregation with GROUP BY (Line number 326)

```
SELECT *

FROM PCParts pe

WHERE pe.Rating > ALL(SELECT AVG(p.Rating) FROM PCParts p GROUP BY ManufacturerID)
```

This query selects all records from the PCParts table where the Rating of the part is greater than the average rating of all parts for each manufacturer (as calculated and grouped by ManufacturerID). It filters rows by comparing each part's rating to the maximum of these average ratings.

2.1.10 Division (Line number 341)

```
SELECT * FROM PCParts p

WHERE NOT EXISTS (

  (SELECT r.RetailerID FROM Retailer r)

  MINUS

  (SELECT s.RetailerID FROM Sell s WHERE p.PartID = s.PartID))
```

This query selects all records from the PCParts table where each part is sold by every retailer listed in the Retailer table. It ensures that there are no retailers who do not sell the given part by comparing the set of all retailers (Retailer table) with the set of retailers selling the part (Sell table).

---

4. **For SQL queries 2.1.7 through 2.1.10 inclusive, include a copy of your SQL query and a maximum of 1-2 sentences describing what that query does. You can embed this in your above list of queries. You don't need to include the output of the query. The purpose of this requirement is to allow your TAs time outside of your presentation to verify these more complex queries are well formed.**