Stage 4: Project Report

Project Direction Changes

The Car Dealership Management application final features are very similar to the original proposal, though some slight changes. Originally, the creative component was only adding a smart search functionality that would allow users to add complex or advanced search filters when searching for cars in the inventory. However, we discussed it with our TA and the professor who felt it was not creative enough. Now, our creative component includes a feature that assigns a vehicle score based on reviews, mileage, year, and other factors, offering dealerships valuable insights to optimize their marketing strategies.

Application Usefulness

The application has features that make it very useful for dealerships to both manage their inventory and their needs. The advanced scoring and filtering functionalities allow dealerships to better understand market trends and customer preferences. We also include an authentication/login feature, which means secure data access for dealership staff. While the application is currently highly functional, it could benefit from having a feature that allows users to add large batches of data rather than only one at once, since dealerships could have many cars to add or update at once.

Data Schema / Source Changes

Our schema and source of data is exactly as we proposed during the previous stages. We used datasets from Kaggle, such as the Carvana car sales data and the Craigslist vehicles dataset, for market analysis and inventory insights as we mentioned in our original proposal.

ER / Table Modifications

Slight changes were made to our original UML diagram. We removed the relationship between the Reviews and Warranties tables, as it was deemed unnecessary. We also added an id field to the Details table and set it as the primary key. We ensured uniqueness in the combination of make, model, and year attributes in the Details table by implementing constraints in the database and adding additional backend validation for foreign key relationships to prevent duplicate entries. We think this is a more suitable design because using an id as a primary key in Details allows for better indexing and ensuring uniqueness improves data integrity.

Added / Removed Functionalities

As mentioned earlier, we added the vehicle scoring feature as a creative component, which calculates scores based on reviews and other criteria like mileage and age. Advanced search is still a general feature of the app but is not a standalone feature anymore because we felt it was important to include it as a general feature.

Advanced Database Features

Our advanced database queries play a significant role in enhancing the functionality of the app. The advanced queries for our creative components like vehicle_features_score and sales_trend_score allow dealerships to rank and analyze their vehicles based on multiple parameters. Our procedures like VehicleDisplayOptimizer allowed for dynamic scoring and ranking for better dealership car management.

Technical Challenges / Lessons Learned

Alondra: One technical challenge the team encountered was setting up the development environment and database using Google Cloud Platform. None of us had prior experience with GCP, which made it difficult to configure the environment for development and testing. In the future, I would opt out of using GCP for the database and use something easier to set up locally like SQLite. If GCP is a priority to a future team, I would suggest using Docker containers so that development is consistent for the whole team, which is something we should have considered adopting earlier on.

Pitupoom: One of the technical challenges we encountered was managing unique constraints and primary keys in a database schema while using Django, as Django does not natively support composite primary keys—primary keys composed of multiple columns. This limitation posed a challenge when designing the Details table, where the combination of attributes make, model, and year needed to be unique to prevent duplicate entries. I solved this issue by adding an id attribute as the primary key for the table. I also enforced a unique constraint on the combination of make, model, and year using Django's model-level constraints so that the database functioned just like before.

Matupoom: One technical challenge that we faced was connecting the backend to the frontend to show the results properly. There is complexity from configuring the APIs in Django to handle requests correctly and ensuring that the React app could fetch and display data. And the setting up to connect GCP to Django was complicated because I couldn’t install mysql-server to my personal laptop.

Praise: One challenge we encountered while implementing transactions for our car dealership management system was ensuring correct behavior under and consistency under real-world scenarios. Determining the appropriate isolation level for transactions was a challenge. Lower isolation levels, like READ COMMITTED, were faster but risked issues like non-repeatable reads. Complex transactions involving multiple tables like updating car prices and inventory scores simultaneously could lead to deadlocks. For example, two transactions attempting to update overlapping records in different sequences would block each other. Triggers executed within transactions sometimes caused unintended conflicts. For example, a trigger updating inventory metrics could fail due to constraints, which would invalidate the entire transaction, even though the main operation was successful.

Future Improvements

The Car Dealership Management system has a lot of interconnected data and complex relationships that could have benefitted from a graph database such as Neo4j. We could have also implemented a machine learning model to provide more predictive analytics as opposed to just an advanced query algorithm to generate vehicle scores.

Teamwork / Labor Division

While we all did database work in alignment with the requirements, our team also made sure to divide tasks based on individual skills evenly. We made sure to split up frontend and backend tasks based on knowledge and comfortability with the tech stack. We made sure to have regular check-ins to ensure on time completion and a modular workflow. Overall, everyone contributed significantly to the project’s success.