Team080 - Project Report

Guan-Hong Lin

Chiayu Wang

Chengzhe Li

Nicole Chen

Demo video link: https://youtu.be/Z-wose2pUlA

A Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

Compared to the Stage 1 proposal, the final project had several changes in direction:

1. **Expanded Features Beyond Original Scope**
    a. Added Car Detail Page, User Dashboard Page, and Register/Login Pages with better information display and navigation, which were not explicitly described in the original proposal.
    b. Developed a Preference Analysis Chart based on user ratings and a stored procedure, providing personalized insights beyond the basic recommendation system we initially planned.

2. **Partial Implementation of Advanced Features**
    a. The Hybrid Recommendation System was only partially completed:
        i. Popular car recommendations based on views and favorites were implemented.
        ii. Personalized favorites-based and similar model recommendations were not fully developed due to time constraints.
    b. The Price Prediction Model using linear regression was not implemented, although the Price Trend Chart was successfully delivered.

3. **Additional Functionalities Not Originally Proposed**
    a. Implemented a Search Alert function allowing users to save search criteria, extending beyond basic search and filter capabilities.
    b. Built a full Favorite System (Add/Remove favorites and Favorite Page), enhancing user engagement.

4. **Backend and Database Improvements**
    a. Set up ActivityLog System to record user interactions (such as favoriting cars) systematically.

b. Processed large raw CSV datasets and imported them into the database after cleaning, which required significantly more preprocessing than originally expected.
c. Developed Rating Input and Rating History functionality for users to rate cars and track their own rating activities.

5. **Unimplemented or Simplified Elements**
    a. A secure buyer-seller messaging system was planned but ultimately not implemented.
    b. Admin moderation features were simplified compared to the detailed controls proposed in Stage 1.
    c. Some advanced visualization features (such as multi-brand comparisons in market trends) were not fully implemented.

B Discuss what you think your application achieved or failed to achieve regarding its usefulness

While our platform has demonstrated comprehensive functionality—covering essential features such as data storage, SQL queries, query performance, triggers, stored procedures, constraints, and an aesthetically pleasing front-end—there are still several areas that require further refinement. Below, I outline the key shortcomings and potential enhancements:

1. **Missing Average Rating and Review Count**
    a. Currently, we do not display the average rating or the number of reviews for each car, which are crucial indicators of a vehicle's popularity. To address this, we could:
        i. Add an average rating column to the car card display.
        ii. Introduce two new fields in the `car` table: `avg_rating` (FLOAT) and `review_count` (INT).
        iii. Implement a trigger to automatically update these values whenever a new rating is inserted.
2. **Insufficient Data Validation and Constraints**
    a. Although we have implemented stored procedures, triggers, and constraints, many critical validations are still missing, particularly at the attribute level for example:
        i. Email Format: Currently, emails can be entered without an "@" symbol or in any invalid format.
        ii. Phone Number Length: There are no restrictions on digit count, allowing incomplete or incorrect entries.
        iii. Weak Password Policies: Passwords can be as short as one character, with no checks for common or weak passwords.
    b. These issues can be resolved by enforcing stricter constraints and input validation rules at the database and application levels.

3. **Lack of Security Measures Against XSS and SQL Injection**
    a. Our platform is vulnerable to several security threats:
        i. Cross-Site Scripting (XSS): Input fields and URLs do not filter JavaScript code, allowing potential script injection.
        ii. SQL Injection: Unfiltered SQL queries in user inputs could lead to unauthorized data access or manipulation.
        iii. Command Execution: Malicious inputs might execute unintended commands on the server.
    b. To mitigate these risks, we should:
        i. Implement input filter to block JavaScript and SQL keywords.
        ii. Use parameterized queries to prevent SQL injection.
4. **Missing Session and Authentication Controls**
    a. Inadequate Permission Differentiation: Logged-in and guest users have nearly identical access (e.g., both logged in and unlogged in pages can view car details and historical ratings).
5. **Improvements should include:**
    a. Restrict guest users from accessing certain information (e.g., view history rating, the permissionto rate) to stimulate account creation.

C Discuss if you changed the schema or source of the data for your application.

1. To implement the **user car match search notification feature**, I added two new tables: SavedSearch and Alert. I also added a UserId column to the existing Car table, allowing the system to track how many cars each user has posted.
2. The **SavedSearch** table stores the search criteria saved by users, including:
    a. SearchId (Primary Key)
    b. UserId (references the user)
    c. Make and Model (the car brand and model)
    d. CreatedAt (timestamp when the search was saved)

3. The **Alert** table records matches between saved searches and cars, including:
    a. AlertId (Primary Key)
    b. SearchId (references the saved search)
    c. CarId (references the matched car)
    d. MatchedAt (timestamp when the match occurred)
    e. IsNotified (whether the user has been notified)

4. To implement the **favorite system** and a**ctivity tracking feature**, I added a new table: ActivityLog. I also enhanced the Favorite table by setting a composite primary key to better manage user-car relationships.
5. The **ActivityLog table** records user actions, specifically favoriting cars, including:
    a. LogId (Primary Key)

b. UserId (references the user)
c. Action (description of the activity, such as "Favorited car ID 123")
d. Timestamp (time when the activity occurred)

6. The **Favorite table** was updated to include:
    a. UserId (references the user)
    b. CarId (references the favorited car)

7. Additionally, a composite primary key (UserId, CarId) was added to the Favorite table to ensure that each user can favorite a specific car only once, enforcing data integrity and preventing duplicate entries.

D Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

1. **Added SavedSearch and Alert tables**
    a. I added a SavedSearch table and an Alert table to support saved search and notification functionalities, which were not included in the original ER diagram.
2. **Added UserID foreign key to Car table**
    a. I added a UserID foreign key to the Car table to explicitly associate each car listing with its owner, ensuring better data linkage between users and cars.
3. **Differences and improvements**
    a. Compared to the original simpler design, the final version is more normalized and better supports user-specific features like saved searches and personalized alerts. This design improves data organization, enhances user experience, and strengthens the relationship between users and their data.
4. **Added ActivityLog table**
    a. I added an ActivityLog table to track user actions, such as favoriting cars. This table was not included in the original ER diagram or proposal.
5. **Added composite primary key to Favorite table**
    a. I modified the Favorite table to use a composite primary key (UserId, CarId) to ensure that each user can favorite a specific car only once, enforcing uniqueness at the database level.
6. **Differences and improvements**
    a. Compared to the original simpler design, the final version introduces an ActivityLog system that improves system transparency, allows future development of features like Recent Activity feeds and Admin monitoring, and enhances platform engagement analysis. The use of a composite key in the Favorite table also ensures stronger data integrity and reflects real-world relationship constraints more accurately.

E Discuss what functionalities you added or removed. Why?

1. **Added user authentication to protect data operations**.
    a. Only the owner of a car or advertisement can update or delete their data. This was important to ensure data security and prevent unauthorized access or modifications.
2. **Removed buyer-seller messaging functionality.**
    a. I prioritized completing the core features first, and implementing secure messaging would have required a real-time system that was beyond the project scope.
3. **Removed the hybrid recommendation system**
    a. Including recent views-based suggestions, favorites-based suggestions, and trending cars. These features would require additional tables and complex queries, which were not feasible within the project timeline.
4. **Added ActivityLog system:**
    a. I added an ActivityLog system to record when users favorite cars, enhancing transparency and enabling future analytical capabilities. This feature was not part of the original project proposal but improves system monitoring and potential future insights.
5. **Added Recent Activities section:**
    a. I added a Recent Activities section on the Favorites page, allowing users to view their latest actions related to favoriting cars. This addition enhances user experience by providing immediate feedback and tracking of user engagement.
6. **Removed price prediction model:**
    a. I removed the price prediction model (originally proposed using machine learning) to prioritize completing and polishing core features like the main page, car detail page, favorites system, and price trend analysis. Implementing a reliable predictive model would have required additional data processing and validation efforts beyond the project's timeline.
7. **Reason for changes:**
    a. We focused on ensuring core functionalities were completed to a high standard rather than partially implementing optional features. The ActivityLog system added more meaningful real-world value to the platform than a basic predictive model would have, improving both user engagement and future extensibility.
8. **Add Advanced Filter:**
    a. The Filter Page enables users to specify search criteria, including car brand, manufacturing year, and price range, to efficiently locate vehicles that match their preferences.Upon form submission, the backend processes the criteria, queries the database accordingly, and returns matching results displayed on the Filter Results Page.
9. **Removed the function of calculating the average score for vehicles**
    a. As although it is a very useful feature, our website lacks sufficient users to obtain sufficient ratings to calculate the average score for each vehicle

F Explain how you think your advanced database programs complement your application.

1. **Implemented Saved Searches and Alerts** to allow users to save search criteria and receive notifications when matching cars are posted. This improves user engagement by keeping users informed without needing to manually check for updates.
2. **Implemented transaction-based favorite and activity logging:**
    a. I implemented SQL transactions when users add cars to their favorites, ensuring that both the insertion into the Favorite table and the logging into the ActivityLog table occur atomically. This guarantees database consistency by rolling back both operations if any part fails.
3. **Implemented advanced queries for efficient data retrieval:**
    a. I implemented JOIN operations to fetch detailed information about users' favorite cars, subqueries to calculate how many users favorited each car, and GROUP BY aggregation to generate historical price trend graphs. These advanced queries enhance the richness and efficiency of the data presented to users, allowing for meaningful insights like favorite counts and price fluctuation analysis without compromising application performance.
4. **The stored procedure operation** in the database greatly simplifies the calling of complex procedures. A stored procedure in the database is used to guess the user's preference based on their type, input the user's type, and output the ranking of the preference for various cars. It can be called every time the system guesses the user's preferences and generates a pie chart.
5. Our **advanced database programs** enhance the application's responsiveness, reliability, and user experience by providing real-time, multi-condition search capabilities. Through optimized dynamic SQL queries, users can filter listings instantly by brand, price, year, and mileage, ensuring fast and smooth interaction without needing page reloads. The relational schema, with well-designed primary and foreign key relationships, ensures data integrity across user accounts, car listings, and ratings. Indexing strategies were also implemented to accelerate frequent queries on popular attributes like price and mileage. Together, these backend optimizations make the platform scalable, support high user activity, and maintain a consistent experience as the dataset grows.

G Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.

1. **Chiayu Wang**

a. One technical challenge we faced was ensuring only the data owner could modify or delete their car or advertisement. At first, there were no strict ownership checks, creating security risks. We solved this by adding session-based authentication and verifying ownership before updates or deletions. Future teams should implement strong authorization checks early to avoid similar issues.

2. **Guan-Hong Lin**
    a. One major technical challenge was parsing the ActivityLog to link back to car titles. Initially, we only logged "Favorited car ID 8" in ActivityLog, but to display a human-friendly activity feed ("You favorited Toyota Camry"), we needed to match log entries back to CarTitle. We solved this using a SQL JOIN combined with string parsing (SUBSTRING_INDEX) in MySQL, dynamically extracting the car ID from the action string and joining it to the Car table. This experience taught us that early database design should anticipate future display needs; storing raw IDs rather than just text would have made things cleaner. Advice to future teams: plan Activity Logs with structured fields instead of free-text for easier querying and linking.
3. **Chengzhe Li**
    a. For me, I think the most difficult part is cleaning and importing the raw data. Initially, we had nearly a million pieces of raw data, especially when the advertising table had a huge amount of data. I needed to clean, process, and import this data into the original table, and then split the original table into the final required tables. Massive join operations are difficult to complete with basic SQL, and almost always require timeouts. I can only write Python scripts and use distributed advanced data processing frameworks such as pandas and numpy to process these data, and then import the fully processed data into a table. However, GCP Importing these data is still very slow because our configuration is not advanced enough, so we can only reduce the data volume to 1/10 of the original before importing.
4. **Nicole Chen**
    a. One technical challenge I encountered was ensuring that the dynamic filtering page correctly synchronized with the backend APIs, especially when users applied multiple optional search conditions. Early in development, the frontend sometimes failed to display results accurately because the API did not correctly handle empty or missing filter parameters, causing errors or incomplete queries. I systematically tested and documented each API endpoint, identified cases where unvalidated inputs caused backend issues, and worked with the team to update the backend to handle optional filters more gracefully. I also helped debug frontend-to-backend integration problems on the filter and results pages. My advice to future teams: prioritize early testing of dynamic input APIs and define clear API contracts so that missing or optional fields do not cause instability in user-facing features.

H Are there other things that changed comparing the final application with the original proposal?

There were several changes between the final application and the original Stage 1 proposal:

1. **Expanded Scope**:
    a. We added several new pages and features that were not explicitly planned in the proposal, including a detailed Car Detail page, a User Dashboard, a Favorites Page, and a Recent Activities section.
2. **New Features Introduced**:
    a. We implemented an ActivityLog system to systematically track user actions such as favoriting cars, providing better user engagement insights and system transparency. This was not originally proposed.
3. **Partial Implementation of Recommendations**:
    a. While the proposal included a full hybrid recommendation system (based on recent views, favorites, and trending cars), only popular car recommendations were partially completed. Personalized recommendations based on favorites and recent views were not fully implemented due to time constraints.
4. **Simplified Search Functionality**:
    a. Instead of a fully dynamic multi-condition filter with live updates, a keyword search bar and a basic filter page were implemented to allow users to search by brand, year, and price range. This simplified design improved development speed while maintaining core usability.
5. **Price Prediction Model Not Implemented**:
    a. Although the proposal mentioned a machine learning-based price prediction model, it was not implemented. We prioritized building a working price trend chart and other core features to ensure project stability and completeness.
6. **Database Design Enhancements**:
    a. Additional tables such as ActivityLog, SavedSearch, and Alert were introduced. We also updated existing tables (e.g., adding a composite primary key to Favorite) to better support real-world functionality, improve data integrity, and prepare for future feature expansions.
7. **Feature Removals**:
    a. The secure buyer-seller messaging system and some advanced visualization features (like multi-brand comparison) were not implemented as originally proposed, focusing instead on delivering a stable and usable core system.

Overall, while the core vision remained the same — helping users navigate the used car market efficiently — the final product emphasized system stability, data integrity, and user experience improvements over optional or overly ambitious features.

I Describe future work that you think, other than the interface, that the application can improve on

1. One area for future improvement is adding a map visualization for used car listings. This would help users easily find cars based on location and improve the browsing experience.
2. Another improvement would be implementing a secure buyer-seller messaging system. Allowing direct communication on the platform would make the buying process more convenient and efficient.
3. Implement true Hybrid Recommendation System: suggest cars based on user favorites, platform popularity, or similar models.
4. Introduce price prediction model (using Linear Regression) based on price trends.
5. Better activity logging structure: store actionType, carId, userId as separate fields for easier JOINs and future analysis.
6. Focus on adding background jobs to handle heavy tasks like updating popular car rankings and price trends asynchronously. This would improve API response time and make the system more scalable as data volume grows. Implementing automatic database backups could also strengthen data resilience.

J Describe the final division of labor and how well you managed teamwork.

For the final division of labor, each team member focused on specific parts of the project.

Team Member	Contribution
Guan-Hong Lin	● Car Detail Page ● Main Page ● Favorite System (Add/Remove) ● Favorite Page ● Main Page Keyword Search ● Price Trend Chart ● ActivityLog System
Chiayu Wang	● Set up the development environment ● Built the User Dashboard page ● Implemented car and advertisement-related database operations ● Developed the search alert function ● Managed user sessions ● Implemented user authentication
Chengzhe Li	● Set up data base schema ● Processed raw csv files data and imported to DB instance

	• Car Detail Page • Register and login Page • Rating Input and Rating history • stored procedure • Preference analysis chart
Nicole Chen	• Filter Page and Filter results page • Integrated and tested the Register API functionality • Designed database constraints