EcoRide: A Car Rental with Environmental Rewards System (Stage 4 Checkpoint 2)

Project Overview

Our DBMS project began as a conceptual framework for a car rental system incorporating environmental rewards. By the final stage, we successfully developed an end-to-end application that integrates user roles (buyer or seller), eco-friendly incentives, dynamic pricing, and analytics. Below, we highlight changes, achievements, technical challenges, and future work.

Changes in Project Direction

During the development of the final project, several changes were made compared to the original proposal to better align with technical feasibility, user experience, and practical use cases. Below is a summary of the key shifts:

1. **Focus on EcoPoints as a Reward Mechanism:**

   The original plan included a wider range of environmental reward strategies. However, in the final project, we simplified the EcoPoints system by basing it only on car ratings and mileage. We chose not to add bonuses for things like trip frequency to keep the focus on encouraging good car maintenance and eco-friendly driving habits.

2. **Limited Buyer Cancellation Feature:**

   Originally, the project aimed to allow flexibility for both buyers and sellers by enabling trip cancellation. However, technical constraints and priority changes led to not implementing buyer cancellations at this stage to ensure that the platform's core reward and pricing features were robust.

3. **Data Analytics and Eco-Stats Features as New Priorities:**

   While these features were not explicitly emphasized in the original proposal, they were added during development to provide meaningful insights to users. Data Analytics allows users to view the top 5 most eco-friendly vehicles, while Eco-Stats provides users insights into their driving habits, total trip mileage, and other statistical trends.

4. **Adjustments to UI and Back-End Complexity:**

   The project initially targeted a simpler interface and less integration complexity. However, the dynamic pricing model(fine calculation) and advanced data visualization (analytics dashboards and EcoStats) required more UI/UX considerations and database restructuring to ensure seamless user interactions.

Usefulness of the Application

The application successfully achieved its goal of creating a user-friendly and eco-conscious car rental system. The features like EcoPoints, Data Analytics, and Eco-Stats added value by encouraging fuel-efficient driving and responsible car ownership. Buyers benefit from tools like identifying the Top 5 Cars and Eco-Stats, making it easier to select the most economical and popular rental options. Sellers have also been incentivized to maintain their vehicles in good condition to earn rewards.

However, the application did encounter some limitations. The system does not yet allow buyers to cancel bookings, which could impact user flexibility and convenience.

Despite this area for improvement, the system has implemented a dynamic pricing model and reward strategies that are working effectively to support eco-friendly decisions and transparency between buyers and sellers. The application has successfully met its primary goal of simplifying the car rental process while promoting sustainability.

Schema and Data Source Changes

We changed the schema to support new features and ensure data integrity.

Schema Changes:

1. **Added Constraints:** Unique constraints on the name column in the User table
2. **Extended the Booking Table:** Added tripStatus and tripCost columns for better trip tracking and dynamic pricing.

Data Source Changes

We primarily used the dataset listed in Stage 1 as the foundation for this project. However, we incorporated an additional dataset to supply accurate startDate and endDate information. This improved the application's trip data and allowed for more precise calculations related to bookings and trip durations.

Link to the additional dataset -
https://www.kaggle.com/competitions/nyc-taxi-trip-duration/data?select=test.zip

ER Diagram Design:

The ER Diagram was carefully designed and implemented without major revisions as its initial structure aligned with the desired behavior.

Functionalities Added or Removed

We added features like **Data Analytics** and **Eco-Stats** to provide users with better insights and improve their decision-making. Data Analytics enables users to view the top 5 cars and top 5 eco-friendly cars on the platform, helping users choose better rental options. The Eco-Stats feature provides important insights such as total trips taken, average mileage, longest trip

duration, total money spent on rentals, favorite car companies, and the number of bookings associated with these favorite companies.

We removed or limited certain features to ensure simplicity, relevance, and better user experience:

1. **Buyers Cannot Cancel Their Bookings**

   Reason: This feature was excluded to streamline system operations and maintain stability by avoiding potential resource conflicts. While cancellation could add flexibility, the feature was deprioritized to ensure a smoother and more efficient user experience.

2. **Sellers' EcoPoints are Calculated Solely on Car Ratings, and Not considering Trip Count**

   Reason: We limited calculations to car ratings to ensure that sellers focus on maintaining the condition of their cars (e.g., cleanliness and servicing) rather than prioritizing trip frequency. This approach promotes environmental awareness and aligns with the vision of encouraging sustainable car maintenance behavior.

Benefit of using advanced database programs

Our advanced database programs significantly enhance the functionality, efficiency, and reliability of our application. By implementing features like advanced SQL queries, stored procedures, triggers, and transactions, we ensured that our application operates smoothly under varying loads and provides accurate, real-time data to users.

1. **Stored Procedures for Efficient Data Handling:**

   Stored procedures played a key role in handling the Eco-Stats feature by processing complex calculations related to trip data, average mileage, total trips, and other metrics. These stored procedures encapsulate logic directly in the database, allowing for faster execution of repetitive and resource-intensive operations while reducing the burden on the application layer.

2. **Triggers for Automated State Changes:**

   Triggers were implemented to manage real-time database changes, such as updating car availability status when a booking is made. This removes the need for the application to keep checking for updates, making it quicker and more reliable.

3. **Transaction Handling with Serializable Isolation:**

   To compute EcoPoints without conflicts or inconsistencies, we applied serializable transactions to maintain data accuracy during concurrent access. This ensures that multiple users or actions (e.g., simultaneous trip updates) are handled without interfering with each other.

4. **Data Analytics and Visualization:**

   We used advanced database queries to make operations faster and added data visualization features. This allows users to view meaningful insights, like the most popular or eco-friendly cars, with minimal delay.

Technical Challenges encountered by each team member

1. **Rohan Patil - Triggers:**

   A significant technical hurdle was creating database triggers to handle automatic status updates for cars based on booking events (e.g., availability changes). The challenge stemmed from ensuring that triggers were executed reliably without impacting database performance. Additionally, the data needed to be curated carefully to ensure only relevant, accurate information was being updated. To solve this, we focused on testing triggers under different booking conditions to identify edge cases and optimize their performance. This taught us the value of clear trigger logic, testing edge cases thoroughly, and ensuring database triggers do not interfere with system performance. This can serve as advice for future developers using triggers in similar database workflows.

2. **Chaitanya Deshpande - Stored Procedures:**

   One major technical challenge we encountered was implementing the stored procedures to handle the Eco-Stats feature. The stored procedure is needed to calculate aggregated statistics like total trips, average mileage, total spending,etc efficiently. The challenge arose because integrating these calculations required optimizing queries to handle large datasets without affecting database performance. To solve this, we focused on indexing key fields and testing query performance iteratively. This taught us the importance of database optimization, indexing, and careful planning when implementing stored procedures, especially when dealing with aggregated data. Future teams can use this approach as guidance when faced with similar performance-related database calculations.

3. **Nandini Deore - Transactions with EcoPoints Logic:**

   Managing database transactions while ensuring the correct computation of EcoPoints was a complex technical challenge. The logic needed to account for edge cases, such as simultaneous updates or interrupted bookings, which could compromise data consistency. To address this, we enforced serializable transactions and thoroughly debugged the transaction logic to ensure accurate reward calculations. This experience highlighted the importance of understanding concurrency, database isolation levels, and testing transaction logic in real-world scenarios. Future teams can refer to this experience when implementing financial reward mechanisms or other sensitive transaction-based computations.

4. **Ankit Chavan - Advanced SQL Queries & Data Visualization:**

   Another key challenge was related to advanced SQL queries and data visualization. The first major challenge was creating a query to calculate the average car rating after each booking was completed and ensuring it was rendered correctly on the user interface. The second advanced query focused on identifying the top 5 eco-friendly cars based on their calculated eco performance metrics. Alongside these queries, we faced additional difficulties with data visualization, specifically presenting the data for the top 5 car companies and top 5 eco-friendly cars in a way that was easy to interpret by users. This experience highlighted the importance of writing advanced sql queries for tasks like data visualization. Future teams can use this experience as guidance when dealing with complex SQL queries and visualization challenges to ensure performance and clarity.

Other changes

No, we have mentioned all the changes in the above sections.

Future work

While our application achieves its primary goals effectively, there are several areas of potential improvement and future development beyond the current interface:

1. **Advanced Cancellation and Refund Mechanisms:**

   Currently, buyers cannot cancel their bookings. In the future, implementing a fair and flexible cancellation policy would improve user experience by allowing buyers to make last-minute changes without financial penalties under reasonable conditions.

2. **Integration of Real-Time Analytics and Machine Learning:**

   Machine learning could improve user experience by predicting demand trends, suggesting optimal rental pricing based on factors like time, location, and user behavior, and offering personalized car recommendations. Additionally, it could optimize maintenance schedules by predicting when vehicles are likely to need servicing, reducing downtime, and improving reliability for users.

3. **Image-Based Mileage Verification**

   To enhance transparency and trust between buyers and sellers, we plan on implementing an image-based mileage verification system. This feature would require users to upload clear photos of the car's odometer both before and after each trip. These images would serve as evidence to verify the reported mileage, reducing disputes and ensuring the accuracy of mileage-based rewards and payments. This addition would improve the reliability of the platform and provide a fairer experience for all users.

Work division

We are a group of 4, and our work distribution is as follows:

Rohan Patil: Database Design, SQL Queries, Backend APIs, Triggers.

Chaitanya Deshpande: Indexing, SQL Queries, Stored Procedure, UI Design.

Nandini Deore: Database Design, Indexing, Backend APIs, Transaction.

Ankit Chavan: Data modeling, UI Design, SQL Queries, Data Visualization.

We have mentioned the broad division of the work above. However, effective teamwork played a crucial role in the success of our project. We ensured all team members were on the same page by discussing each functionality and technical detail thoroughly before implementation. This approach allowed us to align our goals, address potential issues early, and avoid misunderstandings. Each team member maintained transparency about their progress, fostering open communication throughout the project. Whenever someone was stuck, whoever was free would step in to help. Being friends outside of the project strengthened this dynamic, as we encouraged each other and collaboratively reviewed each other's work to improve the overall quality of our contributions.