Final Enhancement

This plan outlines how to add a Vehicle Review System to the existing vehicle inventory application. The review system will let authenticated users post ratings and comments for vehicles. It will interact with the database by creating a new reviews table, add a new model to handle review-related queries, introduce a new controller to process review submissions and retrieval, create new routes for handling review operations, and add a new view to display reviews on the vehicle detail page. All changes follow best practices such as using SQL prepared statements, data validation, and proper error handling.

CREATE TABLE IF NOT EXISTS public.review (

review\_id SERIAL PRIMARY KEY,

vehicle\_id INT NOT NULL,

account\_id INT NOT NULL,

rating INT NOT NULL,

review\_text TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT fk\_vehicle FOREIGN KEY(vehicle\_id) REFERENCES public.inventory(inv\_id) ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT fk\_account FOREIGN KEY(account\_id) REFERENCES public.account(account\_id) ON UPDATE CASCADE ON DELETE CASCADE

);

Create a new model for reviews with functions to interact with the database. This file should include:

* A function getReviewsByVehicleId(vehicle\_id) that retrieves all reviews for a given vehicle using a SQL query with a prepared statement.
* A function insertReview(reviewData) that inserts a new review record into the review table. The reviewData object should include vehicle\_id, account\_id, rating, and review\_text.  
  Ensure error handling and logging are in place.

Create a new controller for handling review operations. This file should define:

* A function getVehicleReviews that accepts a request (with vehicle id as a parameter), calls the review model's getReviewsByVehicleId function, and returns the list of reviews (or integrates them into the vehicle detail view).
* A function addReview that processes POST requests to add a review. It should validate the review input (rating and review text), call the review model's insertReview function, and handle success/error responses (using flash messages or JSON responses as appropriate).  
  Include pseudocode comments for each step and ensure authorization (e.g., check if the user is logged in) is enforced.

Create a new route file for review-related endpoints. In this file, set up an Express router with:

* A GET route at '/reviews/:vehicle\_id' to fetch and return all reviews for a particular vehicle.
* A POST route at '/reviews' to handle submission of a new review. Use middleware to verify that the user is authenticated (for example, using utilities.checkLogin).  
  Include validation middleware if desired (for checking rating values and making sure review text is not empty).  
  Attach this router to the main app in server.js if needed or require it from a central routes file.

Create a new view file that displays the list of reviews for a vehicle and includes a form for submitting a new review. The file should:

* Loop through an array of reviews passed from the controller and display details such as rating, review text, reviewer name (if available), and the creation date.
* Include a form (visible for logged-in users) with fields for rating (e.g., a dropdown with numbers 1 to 5) and review text, and a hidden field for the vehicle id.
* Provide instructions or pseudocode comments regarding client-side validation to ensure the required fields are filled out before submission.