Indian Journal of Computer Science and Technology
Volume 5, Issue3 (September-December 2026), PP: 626-632.
www.indjcst.com
ISSN No: 2583-5300

Auto Wise: An AI/ML-Based Vehicle Valuation and Visual Damage Inspection System

Swapnil Harale[1], Owais Chougule[2], Alok Chougule[3], Prathamesh Nandrekar[4], Muzzammil Shaikh[5]

[1,2,3,4,5] *Department of Artificial Intelligence and Machine Learning, Sunjay Bhokare Group of Institutes Miraj, Maharashtra, India.*

To Cite this Article: *Swapnil Harale[1], Owais Chougule[2], Alok Chougule[3], Prathamesh Nandrekar[4], Muzzammil Shaikh[5], "Auto Wise: An AI/ML-Based Vehicle Valuation and Visual Damage Inspection System", Indian Journal of Computer Science and Technology, Volume 05, Issue 03 (September-December 2026), PP: 626-632.*

Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution License; Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: *The used-vehicle market provides an important option for buyers who want affordable transportation and sellers who want to recover value from an existing vehicle. However, estimating a fair price can be difficult because the value of a vehicle depends on several factors, including make, model, manufacturing year, distance travelled, fuel type, transmission, ownership history, maintenance, visible condition, and local market demand. AutoWise is proposed as an integrated AI/ML-based vehicle valuation and inspection platform intended to support more informed used-car decisions. The system combines a machine-learning model for estimating a vehicle's market value with a computer-vision component for identifying visible damage, such as dents and scratches, from suitable images. The platform is also designed to present the results through a readable inspection and valuation report and to support marketplace functions such as user accounts, vehicle listings, seller information, and buyer interest. A Python-based backend can coordinate the prediction model, image-processing workflow, database operations, and web interface. Historical vehicle data is used to train and evaluate the price-estimation model, while image data with appropriate annotations is required for damage detection. The predicted price and damage findings are presented as estimates rather than guarantees; photographs cannot establish every hidden mechanical or structural fault. This paper describes the problem, proposed system, related technical concepts, modules, architecture, development methodology, testing strategy, limitations, and possible future improvements for AutoWise.*

Key Word: *AutoWise, used-car valuation, machine learning, regression, computer vision, vehicle damage detection, Flask, Python, MySQL, inspection report.*

I.INTRODUCTION

Buying or selling a used vehicle requires an assessment of both its price and its condition. Unlike a new vehicle with a published price, a used vehicle may have a different value depending on age, mileage, variant, fuel type, service record, ownership history, location, and visible wear. Two vehicles of the same model and year can therefore have noticeably different market values. Buyers may find it difficult to judge whether an asking price is reasonable, while sellers may struggle to choose a competitive price. In addition, a vehicle can have visible damage that affects its resale value, and a buyer may not know how to interpret that damage or estimate likely repair expenses.

Many people use online listings to compare vehicles, but the information available across listings can be inconsistent. Prices may be asking prices rather than completed sale prices, vehicle details may be incomplete, and photographs may not show every part of the vehicle. A listing alone may not provide a consistent valuation method or a structured inspection summary. These difficulties create an opportunity for a decision-support system that brings vehicle information, price estimation, visual inspection support, and listing information into a single workflow.

AutoWise is proposed to address this need. It uses machine-learning techniques to estimate a market value from selected vehicle attributes and computer-vision techniques to analyse submitted images for visible damage. The system can then present the prediction, detected damage, and any configured repair-cost estimates in a consolidated report. Marketplace features can help users view listings, review seller information, and express interest in a vehicle. A separate physical inspection service may also be offered if an inspection team and operational process are available.

The primary aim is not to replace a professional mechanic or guarantee a final selling price. Instead, AutoWise is intended to provide an accessible first-level estimate and organize useful information for the buyer or seller. The quality of the output depends on the relevance of the training data, the quality of photographs, the performance of the models, and the reliability of the information entered by users.

A. Problem Statement

Used-car buyers and sellers may not have a consistent, transparent method for estimating a vehicle's market value and reviewing visible damage. Manual comparisons can take time, and judgments may differ between individuals. AutoWise

proposes a centralized application that uses vehicle attributes for price estimation, image analysis for visible-damage identification, and structured reporting to help users make better-informed decisions.

B. Objectives

The main objectives of AutoWise are: (1) to collect relevant vehicle attributes through a clear user interface; (2) to estimate a vehicle's market price using a trained regression model; (3) to analyse suitable vehicle images for visible damage using a computer-vision model; (4) to combine the available results in a readable inspection and valuation report; (5) to provide database-backed user and vehicle-listing functions; (6) to protect user information through appropriate authentication and access controls; and (7) to evaluate the system through functional tests and model-performance metrics.

C. Scope

The planned scope includes used-car price estimation, visible damage analysis from images, report generation, user registration and login, vehicle listing management, and buyer-interest tracking. The system may use public datasets and authorised listing data for development and evaluation. Exact capabilities depend on the final implementation, dataset, model, and available infrastructure. Hidden engine faults, accident history, odometer fraud, and legal ownership status cannot be verified from photographs alone and are outside the claims of an image-based inspection feature.

II. LITERATURE REVIEW

Research and established software practice in machine learning, computer vision, and web application development provide the technical foundation for AutoWise. Vehicle-price estimation is a supervised learning problem when examples include input attributes and known target prices. Regression methods learn a relationship between features and a numeric target. Linear models can provide a simple baseline, while tree-based methods can capture non-linear relationships between vehicle attributes. Random Forest, introduced by Breiman, combines predictions from multiple decision trees and is commonly used for structured-data tasks. Its performance still depends on representative data, suitable preprocessing, and careful evaluation.

Computer vision allows software to extract information from images. Image-processing libraries can resize images, normalize input formats, and prepare images for a trained model. Object-detection models can predict both a class and a location within an image. For AutoWise, a damage-detection model would require a labelled image dataset containing relevant visible defects, such as dents or scratches, and examples of undamaged vehicle surfaces. Performance can be affected by lighting, camera angle, image resolution, occlusion, paint colour, and the size of the defect. A system should therefore communicate uncertainty and avoid treating every prediction as definitive.

Web application architecture provides a way to connect the user interface, backend logic, machine-learning models, and database. Flask can expose Python functions through HTTP routes, while a relational database such as MySQL can store structured information about users, vehicles, listings, and reports. Separating the interface, application logic, model inference, and data persistence makes the system easier to test and maintain. Authentication and authorization are particularly important because the application may contain user contact details and listing information.

Online vehicle listings can help users understand market context, but listed prices should not automatically be treated as final transaction prices. Data gathered from external websites must be obtained in accordance with the provider's terms, permissions, and applicable law. A model trained on incomplete or outdated data may produce misleading estimates. The literature and technical documentation therefore support a cautious approach: use a baseline model, measure its performance on held-out data, document limitations, and provide results as decision support rather than as a guarantee.

III. PROPOSED SYSTEM AND MODULES

AutoWise is organized as a set of connected modules. Each module performs a defined task while exchanging data through the application backend. The exact set of features presented to a user should reflect the modules that are implemented and tested in the final project.

A. User Registration and Authentication

This module allows a user to create an account and sign in. The backend validates submitted fields and checks the user's credentials. Passwords should be stored using a suitable password-hashing method rather than as plain text. Session management and authorization rules should prevent one user from editing another user's private information. User-facing error messages should be clear without revealing sensitive account details.

B. Vehicle Information Module

The vehicle information form collects attributes needed for valuation. Typical fields may include make, model, variant, manufacturing year, kilometres driven, fuel type, transmission, ownership count, and service information where available. Validation rules should reject impossible values, missing required fields, and inconsistent combinations. The user should be told which attributes are required and which are optional. Standardized category values and consistent units improve the quality of the data passed to the model.

C. Price Prediction Module

The price prediction module receives validated vehicle attributes and applies the same preprocessing used during model

training. A trained regression model returns an estimated price in Indian rupees. Candidate algorithms include Linear Regression as a baseline and Random Forest Regressor as a non-linear model. Model selection should be based on measured performance rather than on the algorithm name alone. The model and its preprocessing steps should be saved and versioned together so that inference uses the correct feature order and transformations. The interface should identify the output as an estimate and, where supported, show a useful range or confidence-related explanation rather than suggesting absolute certainty.

D. Image Upload and Damage Detection Module

The image module allows a user to upload one or more vehicle photographs. The backend checks the file type and size, stores files safely, and prepares valid images for inference. A trained computer-vision model may identify visible damage classes such as dents and scratches. If the model supports object detection, it may also return bounding boxes or regions of interest to indicate where a suspected defect appears. The system should handle unsupported file types, empty uploads, and model errors gracefully. Since damage recognition depends on image quality and the dataset, the result should be described as a possible visible defect requiring confirmation when necessary.

E. Repair Estimate and Valuation Adjustment

If repair-cost estimates are available from a defined price table or inspection process, the system can display them alongside the predicted market value. A simplified adjusted estimate can be calculated by subtracting an estimated repair cost from the predicted market value. This is only a configurable approximation: the market may already account for some defects, and damage can affect resale value differently depending on severity, part, labour, vehicle age, and local conditions. The application should make the assumptions visible and avoid double-counting the effect of damage.

F. Inspection Report Module

The report module combines the vehicle details, predicted market value, image-analysis findings, estimated repair costs when available, and a summary of limitations. It may include a unique report identifier, date of generation, uploaded image references, and the model version used. Reports should distinguish user-provided information from machine-generated results. If the system has not received images or has not run the damage model, it must not imply that a visual inspection was completed.

G. Marketplace and Listing Module

The marketplace module can allow sellers to create listings and buyers to browse available vehicles. A listing may include a title, vehicle attributes, asking price, photographs, seller profile, and listing status. Buyers can express interest through a controlled workflow. The database should link each listing to its owner and ensure that only authorized users can edit or remove it. External market references can support comparison, but any integration with third-party platforms requires appropriate access and compliance with their terms.

H. Administrator and Physical Inspection Module

An administrator function may support moderation of listings, management of users, and review of reports or inspection requests. If a paid physical-inspection service is included, the workflow should record booking details, requested time, status, and the assigned inspector. The actual inspection service requires trained personnel and an operational process; a software booking screen alone does not establish that a physical inspection has occurred.

IV. SYSTEM DESIGN AND ARCHITECTURE

The proposed architecture separates the client interface, application backend, machine-learning services, and database. A user interacts with the web interface and submits vehicle details or images. The backend validates the request and routes it to the relevant service. The price-prediction service processes structured attributes, while the image-analysis service processes photographs. The backend combines available outputs, stores relevant records, and returns a report to the interface.

The database can contain related tables such as Users, Vehicles, Listings, Valuation Reports, Damage Findings, and Inspection Bookings. A primary key uniquely identifies each record, while foreign keys maintain relationships between users, vehicles, and listings. Sensitive data should be minimized and accessed only when required. Uploaded images should be stored using safe file names and access controls; the database may store file references rather than large binary objects, depending on the chosen design.

A modular design improves maintainability because the prediction model can be updated without rewriting the entire interface. It also allows independent tests for input validation, model inference, database queries, and report generation. API endpoints should validate inputs and return predictable response structures. Errors such as invalid vehicle details, unavailable model files, database connection failures, and unsupported image formats should be handled explicitly rather than causing the whole application to fail.

A. Proposed Workflow

1. The user opens AutoWise and signs in, if authentication is required. 2. The user enters vehicle details and submits the form. 3. The backend validates the values and prepares the model input. 4. The price model generates an estimated market value. 5. The user may upload photographs for visible-damage analysis. 6. The vision model returns any supported damage

findings. 7. If configured repair estimates are available, the backend calculates and labels an adjusted estimate. 8. The report module assembles the results and stores the report record. 9. The user can review the report and, where marketplace features are implemented, view or create listings and express interest.

B. Core Valuation Formula

A regression model estimates the vehicle price from its features. In general form, the model can be expressed as:

$$P=f(X)$$

where $\hat{P}$ is the predicted price, X is the feature vector containing vehicle attributes, and f is the trained prediction function. If a simple repair-cost adjustment is used, the estimate may be represented as:

$$Padj=P-R$$

where Padj is the simplified adjusted estimate and $\hat{R}$ is the estimated repair cost. This equation is a transparent calculation rule, not a claim that every vehicle's resale value changes by exactly the repair cost. The adjustment should only be displayed when the assumptions and repair-cost source are defined.

C. Design Considerations

Usability: The interface should use clear labels, sensible defaults, field validation, and readable reports. Scalability: Model inference, image processing, and database operations should be separated so that each component can be optimized as demand increases. Security: Authentication, authorization, secure password storage, input validation, safe upload handling, and protected configuration values are required. Reliability: Missing data and model failures should produce understandable messages. Explainability: The system should state which details influence the estimate where feasible and disclose important limitations. Maintainability: Code, datasets, model versions, and test cases should be organized so that the system can be reviewed and updated.

Fig. 1. Proposed system architecture of AutoWise

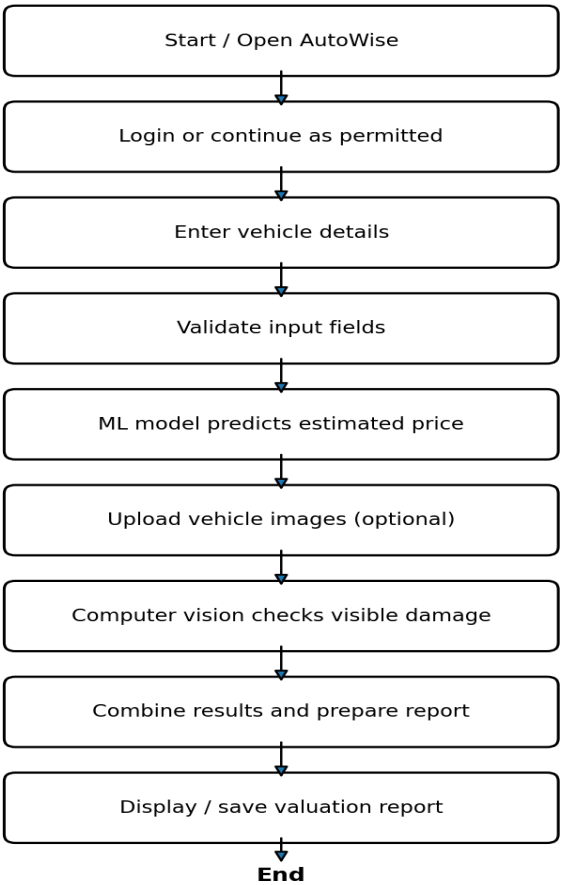


Fig. 1. AutoWise system flowchart

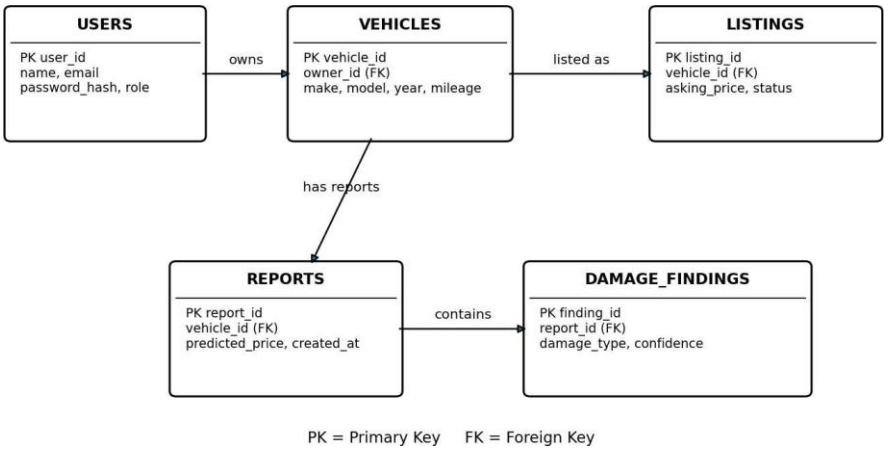


Fig. 2. Proposed entity-relationship (ER) diagram for the AutoWise database.

User / Web Interface → Python Backend / Flask API → (Price Prediction Model + Computer-Vision Model) → Report Builder and MySQL Database. The database may store users, vehicles, listings, reports, and inspection requests.

V. DEVELOPMENT METHODOLOGY

The proposed development process follows an iterative approach. Work is divided into small modules, each of which can be implemented, tested, and improved before full integration. The methodology begins with requirements analysis, followed by data preparation, model development, backend and database implementation, interface development, integration, testing, and deployment preparation.

A. Requirements Gathering

Functional requirements include collecting vehicle details, producing a price estimate, accepting images, displaying damage findings when supported, generating reports, managing listings, and maintaining user records. Non-functional requirements include reasonable response time, clear error handling, maintainable code, data protection, and compatibility with the intended execution environment. Requirements should be prioritized so that core valuation works before optional marketplace or booking features are added.

B. Dataset Preparation

The price dataset should contain relevant vehicle attributes and a target price. Initial preparation includes checking column names, data types, duplicate rows, missing values, inconsistent category labels, and implausible prices or mileage values. Categorical fields need appropriate encoding, and numeric fields may require validation or transformation. Training and test sets should be separated before model evaluation to reduce the risk of reporting performance on data used to fit the model. If data is collected over time, a time-based test split may better reflect future use. Any removal or transformation of records should be documented.

C. Model Training and Evaluation

A baseline model provides a point of comparison. Candidate regression algorithms can then be trained with a consistent preprocessing pipeline. Common metrics include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). MAE reports the average absolute difference between predicted and actual prices in the target unit. RMSE gives larger errors more influence. R^2 describes how much target variation is explained relative to a baseline under its standard interpretation, but it should not be used alone. Evaluation should include inspection of error distribution and performance across important vehicle groups. No accuracy value should be reported until it has been calculated on an appropriate test set.

D. Damage Model Preparation

Damage detection requires images labelled for the classes that the model is expected to recognize. Labels may indicate a defect class and, for object detection, the location of the defect. Training and validation images should be separated to avoid near-duplicate images appearing across splits. The test set should reflect the variety expected in use, including lighting, angles, vehicle colours, and image quality. Precision, recall, and mean average precision may be suitable metrics for an object-detection model, depending on the chosen framework. A model should not be described as complete or accurate without testing it on representative data.

E. Backend and Interface Integration

The backend receives form submissions, validates inputs, loads the correct model artefacts, performs inference, and

communicates results to the interface. A Flask application is one possible implementation for Python-based routing. The interface should display the estimated price, relevant vehicle details, damage findings, and report status. API responses should be consistent, and model-loading failures should be logged without exposing secrets to users. Database operations should use parameterized queries or an appropriate database library to reduce injection risks.

VI. TESTING AND VALIDATION

This paper describes the proposed design and evaluation approach. Statements about completed features, measured accuracy, deployment, or user acceptance should be added only after the corresponding work has actually been completed. For reproducibility, the project should record the dataset source, preprocessing steps, model type, important parameters, train/test split method, software versions, and evaluation results. This makes it easier for supervisors and future developers to understand how a result was obtained.

Testing is necessary to confirm that each module behaves as expected and that the integrated system handles normal and exceptional conditions. Testing should cover both application functionality and the quality of machine-learning outputs. The following categories form a test plan; actual pass/fail results should be filled in after tests are executed.

A. Unit Testing

Unit tests should verify input validation, preprocessing, prediction function calls, report calculations, database helper functions, and upload checks independently. Example cases include a valid vehicle record, a missing required field, an unsupported fuel type, negative mileage, an unsupported image format, and a missing model file. Boundary cases should be tested as well as ordinary inputs.

B. Integration Testing

Integration tests should check whether the interface sends the expected data to the backend, whether the backend calls the correct model, and whether results are stored and returned correctly. Tests should also confirm that a generated report refers to the right vehicle and user, that failed image inference does not create false damage findings, and that listing operations preserve ownership and authorization rules.

C. Model Validation

The price model should be evaluated using held-out examples and appropriate regression metrics. Prediction errors should be reviewed for unusually expensive or inexpensive vehicles and for different age or mileage groups. The damage model should be assessed using labelled images that were not used for training. False positives may incorrectly report damage, while false negatives may miss visible defects. The interface should communicate these limitations and encourage professional confirmation where the result could affect a purchase decision.

D. User Acceptance and Performance Testing

User acceptance testing can involve a small group of intended users completing common tasks, such as entering vehicle details, uploading a photograph, reading a report, and browsing a listing. Feedback can identify confusing labels or missing information. Performance testing should measure response time for price prediction, image processing, report generation, and database queries under expected usage. Image inference may require more processing time than ordinary form submission, so the interface should indicate progress and avoid duplicate submissions.

VII. EXPECTED RESULTS AND DISCUSSION

Security checks should verify that users cannot access another user's private records, that administrative functions are restricted, that invalid inputs are rejected, and that uploaded files cannot be used to execute server-side code. Passwords should not be stored in plain text. Database credentials and secret keys should be kept outside public source files. The project should collect only necessary personal information and define how uploaded images and reports are retained or deleted.

VIII. LIMITATIONS AND FUTURE SCOPE

The expected outcome is a working prototype that demonstrates how machine learning and computer vision can support used-vehicle assessment. For a valid input record, the price-prediction module should return a numeric estimate in rupees. For supported photographs, the damage module should return findings only for the classes it has been trained to recognize. The report should present the source vehicle details, model-generated results, and any repair-cost assumptions in a clear format. Marketplace functions should connect listings with their owners and allow authorized users to express interest.

The usefulness of the system should be evaluated through evidence rather than assumptions. Relevant evidence includes regression error on held-out data, damage-detection metrics on labelled images, successful functional test cases, and feedback from users who complete defined tasks. If the data is limited or does not represent the local used-car market, the valuation model may not generalize well. Similarly, a damage model trained on a narrow image set may fail on unusual angles, reflections, dirt, or very small defects. These risks should be described in the final report.

A major benefit of the proposed workflow is the consolidation of information. Instead of viewing a price estimate and visual findings separately, a user can review both in a single report. However, consolidation does not remove the need for independent verification. Vehicle price depends on market conditions and negotiation, while photographs provide only a partial view of the vehicle. AutoWise is therefore best understood as a support tool that can help users ask better questions and decide whether a professional inspection is needed.

IX. CONCLUSION

The proposed system has several limitations. First, the quality of price estimates depends on the quantity, freshness, and representativeness of the dataset. Asking prices may differ from actual transaction prices. Second, visible-damage detection cannot confirm hidden mechanical issues, internal structural damage, previous repairs, or the authenticity of the odometer. Third, image quality and environmental conditions can affect predictions. Fourth, a repair-cost table may not reflect local labour rates, parts availability, or the severity of a defect. Fifth, marketplace integrations and physical inspection services require operational, legal, and data-access considerations beyond model development.

Future improvements may include adding more representative regional data, updating the model as market conditions change, comparing multiple regression algorithms, providing a transparent prediction range, and improving error analysis.

The vision component could be expanded with additional labelled damage categories and more robust evaluation. A guided photo-capture process could request front, rear, side, and close-up views to improve coverage. Further development may add service-history records, document verification, appointment scheduling, notifications, multilingual support, and mobile-friendly interfaces. Integrations with third-party vehicle listing platforms should be pursued only through permitted and reliable data-access methods. Future versions may also include a human review workflow for uncertain predictions.

Acknowledgements

AutoWise proposes an integrated approach to used-vehicle valuation and visual inspection by combining machine-learning price estimation, computer-vision damage analysis, report generation, and database-backed marketplace features. The system is intended to help buyers and sellers understand a vehicle's estimated market value and review possible visible defects in a structured way. Its modular architecture separates the user interface, backend logic, prediction models, and data storage, allowing each component to be tested and improved independently.

The reliability of AutoWise depends on sound dataset preparation, appropriate model selection, careful evaluation, secure application design, and honest communication of uncertainty. A predicted price should be treated as an estimate, and image-based damage findings should not replace a professional mechanical inspection. The proposed testing plan emphasizes measurable model performance, functional correctness, privacy, and user experience. With further validation and responsible implementation, AutoWise can serve as a useful decision-support prototype for the used-vehicle market.

REFERENCES

The project team would like to express sincere gratitude to the project guide and faculty members for their guidance, suggestions, and encouragement during the planning and development of AutoWise. We also thank our institute for providing an academic environment in which we could study artificial intelligence, machine learning, software development, and database concepts through a practical project. We acknowledge the support of classmates, friends, and family members who encouraged the team during project work. The final version of this section should include the correct names and designations of the guide, department, and institute, and should reflect the support actually received.