

Combining Expert and Automated Property Valuation in a Web-Based Real Estate Platform

Constantin-Alexandru Murgășanu

University of Craiova

Craiova, Romania

murgasanu.constantin.y7g@student.ucv.ro

Paul-Ștefan Popescu

Dept. of Computers and Information Technology

University of Craiova, Craiova, Romania

stefan.popescu@edu.ucv.ro

ABSTRACT

Estimating the market value of a residential property remains a friction point in online real estate transactions: mainstream listing portals aggregate advertisements but leave valuation either to paid, certified experts or to the untrained judgement of the users themselves. This paper presents PropertyEval, a web platform that integrates property and listing management with two complementary valuation channels: a multi-stage appraisal workflow carried out by human evaluators, and an on-demand automated valuation model (AVM) trained at request time on the platform's own history of completed appraisals and asking prices. Every automated estimate is presented together with a confidence interval, the size of the training set and the name of the model, so that the origin of the displayed figure remains inspectable by the user. We describe the role-based interaction design, the valuation module and the validation of the platform through functional testing and heuristic evaluation, and we discuss how transparent instant estimates can complement, rather than replace, expert appraisal in consumer-facing real estate applications.

AUTHOR KEYWORDS

Automated valuation model; real estate platform; human-AI interaction; machine learning; regression; web application; transparency.

DOI: 10.37789/icusi.2026.22

INTRODUCTION

The real estate market is one of the most dynamic and complex economic sectors, and in Romania it has expanded considerably over the last decade, driven by accelerated urbanisation, rising purchasing power and easier access to mortgage credit. This growth has generated a strong demand for digital tools that simplify the processes of buying, selling and evaluating properties. The dominant local platforms – large advertisement aggregators – cover the publication and search of listings well, but the valuation of a property remains largely a process that depends on certified appraisers or on subjective comparisons made by users without domain training. Sellers therefore lack an objective reference for the asking price, buyers lack a second opinion on whether a price is plausible, and this uncertainty is a significant source of friction in the transaction process.

Internationally, the PropTech trend addresses this gap through automated valuation models (AVMs), statistical or machine-learning models that estimate the market value of a property from its characteristics and from historical comparable transactions [1, 2]. On the Romanian market, however, no mainstream portal currently offers an integrated, freely accessible automated estimate, and valuation remains a paid, on-request service.

This paper describes the design, implementation and validation of PropertyEval, a full-stack web platform that closes this gap at prototype scale. The platform combines classic marketplace functionality – property registration, image galleries, listing publication, search and filtering – with two complementary valuation channels: (i) a manual appraisal workflow, modelled as a multi-stage process involving the property owner, an evaluator and an administrator, and (ii) an instant automated estimate produced by a gradient-boosted regression model trained on demand, exclusively on the platform's own data (completed appraisals and asking prices), without external services or datasets.

The contribution is threefold. First, we present an artifact: a working platform whose backend follows Clean Architecture and whose frontend is a modern Angular single-page application, with role-based interaction covering four actor types. Second, we describe an interaction design that treats the automated estimate as transparent, provisional advice rather than an authoritative verdict: the user interface always discloses the confidence interval, the number of training records and the model that produced the figure, and the system explicitly declines to produce an estimate when the available data are insufficient. This design operationalises, in a consumer domain, several of the guidelines for human-AI interaction proposed by Amershi et al. [6], in particular making clear how well the system can do what it does. Third, we report on the validation of the platform through an automated frontend test suite, systematic end-to-end scenarios and a heuristic inspection of the interface. We distinguish the two levels of contribution explicitly: the constituent technologies are established and their integration is an engineering contribution, whereas the research contribution is the interaction pattern that couples a staged human appraisal with a transparent, self-limiting automated estimate inside a single consumer-facing workflow.

The remainder of the paper reviews related work (Section 2), presents the system, its actors and its data model

(Section 3) and the design of its main functional flows (Section 4), details the architecture (Section 5) and the automated valuation module (Section 6), then discusses the user interaction design (Section 7), the testing strategy (Section 8) and a heuristic evaluation of the interface (Section 9), before the limitations (Section 10) and the conclusion (Section 11).

RELATED WORK

Digital real estate platforms

The Romanian digital real estate market is dominated by a few large portals – imobiliare.ro, Storia and OLX Imobiliare – that aggregate hundreds of thousands of listings and offer filtering by location, property type, surface and price. These platforms operate predominantly as advertisement aggregators: they publish market statistics and trend reports, but they do not expose an automated valuation instrument to ordinary users; appraisals are performed by certified partners, on request and for a fee. Globally, the PropTech trend moves in the opposite direction: Zillow and Redfin in the United States, Immowelt and ImmoScout24 in Germany or Idealista in Spain embed AVMs that give sellers and buyers an immediate reference point without replacing the certified expert. In all these cases the estimate is computed from large proprietary transaction databases and is shown as a single figure, with little or no indication of the data behind it. PropertyEval follows this trend at a smaller scale, with a simplified AVM trained on the platform’s internal data.

Property valuation techniques

Traditional appraisal practice distinguishes three main methods [1]: the sales-comparison approach, which adjusts the prices of recently transacted comparable properties; the income approach, used mainly for rental or commercial assets; and the cost approach, which sums land value and depreciated construction cost. The accuracy of the comparative method, which dominates residential valuation, depends directly on the availability of relevant comparables and on the appraiser’s experience. AVMs replace the manual selection of comparables with a statistical or machine-learning model fitted on a large volume of historical data [2]; their main advantages are speed and scalability, their main limitations the dependence on the quantity and quality of the training data and the difficulty of capturing qualitative attributes such as finishing quality or view.

Gradient-boosted regression

Estimating the value of a property is by nature a regression problem: the target variable is continuous. Decision trees partition the feature space through successive if-then rules, but a single tree tends to overfit; ensemble techniques combine many trees to obtain more robust predictions. Gradient boosting builds the ensemble sequentially, each new tree correcting the residual errors of the current ensemble by gradient descent in function space [3]. FastTree, the algorithm used in this work through the ML.NET library [5], is an efficient implementation of gradient-boosted decision trees inspired

by LambdaMART [4], with strong performance on tabular datasets of moderate size. Alternative implementations such as XGBoost [13] and LightGBM [14], or bagging ensembles such as random forests, are often stronger on large tabular datasets, but in a .NET deployment they require an external Python runtime or a native dependency and a separate serving path. FastTree was selected because it is trained and executed in-process, which is what makes the on-demand training strategy of Section 6 possible at all; at the data volumes currently handled by the platform, the difference between these learners is in any case dominated by the size of the training set.

Human-AI interaction and trust

Presenting an algorithmic estimate to a lay user raises interaction design questions that go beyond model accuracy. The guidelines for human-AI interaction of Amershi et al. [6] recommend, among others, making clear what the system can do and how well it can do it. Research on trust in automation shows that trust should be calibrated to actual system reliability, and that communicating uncertainty is a key mechanism for such calibration [7]. In the valuation context this argues for exposing, alongside the point estimate, an explicit error interval and information about the data that produced it – the approach adopted in PropertyEval – rather than a single, authoritative number.

SYSTEM OVERVIEW AND USER ROLES

PropertyEval is organised around five functional modules, mirrored by the services of the backend: account management and authentication, property management, listing management, manual evaluation and automated valuation. Requirements were specified per module, together with non-functional requirements structured along the quality characteristics of ISO/IEC 25010 [12].

Four actors were identified, organised in a generalisation hierarchy in which each more specialised actor inherits all use cases of the more general one: a Visitor may register, authenticate and browse, search and filter active listings, including the full detail page of a property; a Standard user additionally manages their own properties, images and listings, requests a manual appraisal and obtains the instant automated estimate for their own properties or for properties with an active listing; an Evaluator additionally works the queue of pending appraisal requests and may estimate any property in the system; an Administrator additionally manages users and roles and has unrestricted access to all properties, listings and evaluations. Table 1 summarises the access matrix.

Table 1. Actor – use case access matrix (V = Visitor, U = Standard user, E = Evaluator, A = Administrator).

Use case	V	U	E	A
Browse, search and filter active listings	✓	✓	✓	✓
Manage own properties and images	–	✓	✓	✓
Publish and manage own listings	–	✓	✓	✓

Request a manual appraisal	–	✓	✓	✓
Instant automated estimate (own/active)	–	✓	✓	✓
Assume and complete appraisals	–	–	✓	✓
Automated estimate for any property	–	–	✓	✓
Manage users and roles	–	–	–	✓

Data model

The data model gravitates around the Property entity, which holds the physical attributes of a dwelling: type (apartment or house), usable surface, number of rooms and bathrooms, year of construction and a free-text description. The location is factored out into a separate Address entity (street, city, county); besides normalising geographic data, this makes City and County directly available as categorical features for the valuation model. A property may have multiple photographs (PropertyImage, deleted in cascade with the property), while users are represented by a User entity linked to a separate Role entity with three predefined records (Client, Evaluator, Admin), allowing the permission system to be extended without schema changes.

The commercial dimension is deliberately separated from the physical asset through a distinct Listing entity, holding the title, the asking price and a status (Active, Inactive, Expired, Sold). This separation reflects a domain reality: the structural characteristics of a dwelling are stable in time, whereas its sale status changes frequently, and the same property may be the subject of several successive listings at different prices; modelling them separately preserves the commercial history of the asset. Finally, the Evaluation entity models the manual appraisal as a multi-stage process (Pending, InProgress, Completed, Rejected) with two distinct references to User – the requester and the initially unassigned evaluator – which makes the staged workflow possible without an additional assignment entity. Referential integrity is enforced with differentiated delete strategies: listings and evaluations restrict the deletion of their property, protecting transactional history, while images cascade.

DESIGN OF THE MAIN FUNCTIONAL FLOWS

Three flows connect the requirements to the implementation and were designed explicitly before coding: the publication of a listing, the staged manual appraisal and the synchronous automated estimate. Listing publication is guarded on the server: the endpoint extracts the user identifier from the JWT, verifies that the indicated property exists and that the current user owns it – a restriction waived for administrators – and only then persists a Listing with initial status Active; the failure paths terminate with 404 or 403 responses that the interface maps to user-readable messages.

The manual appraisal involves three roles and unfolds in stages separated in time. The owner's request creates an Evaluation with status Pending and no assigned evaluator;

an evaluator consults the pending queue and assumes the request, which moves it to InProgress; the assigned evaluator then submits the appraised value, optional notes and the date, moving it to Completed – or to Rejected when the request is considered unjustified. Assumption is exclusive: a request leaves the queue as soon as one evaluator takes it, so the same request can never produce two competing appraisals, and a second opinion requires a new request, which the owner sees as a separate dated entry in the evaluation history of the property. Evaluators are members of the platform rather than independent appraisers hired by the user, and the prototype treats the appraisal as free of charge; in a deployment it would carry a fee, as it does on the incumbent portals, leaving the owner's asking price and the automated estimate as the two signals available at no cost.

The automated estimate, by contrast, is synchronous and leaves the database unchanged: training data are loaded, the model is fitted, a single prediction is produced and the response is returned within the same request. This difference in temporal structure between the two valuation channels – a long-running social process versus an instantaneous computation – is reflected directly in the interface design discussed in Section 7.

SYSTEM ARCHITECTURE

The system follows the client-server model typical of modern web applications: an Angular single-page application communicates through a REST API [9] with an ASP.NET Core backend that owns business logic and persistence in a relational database (Figure 1). The two components are developed and tested independently and interact only through a well-defined contract of HTTP routes and JSON structures.

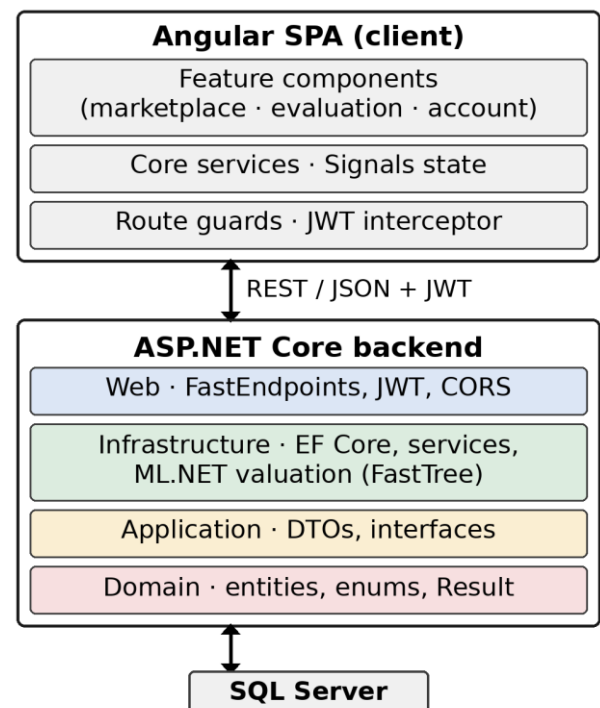


Figure 1. Overall architecture: Angular SPA, layered ASP.NET Core backend and relational persistence.

Backend: Clean Architecture with FastEndpoints

The backend is organised on Clean Architecture principles [8], with the dependency rule oriented strictly towards the core. The Domain layer contains the entities, the status enumerations and cross-cutting elements such as a generic Result type for explicit success/failure propagation. The Application layer defines the data-transfer objects and the service interfaces, decoupling the exposed contract from the internal model. The Infrastructure layer concentrates technical detail: the Entity Framework Core context and migrations, the business services for each resource, and – a distinctive element – the ML.NET-based valuation service described in Section 6. The Web layer is the composition root and exposes the API through FastEndpoints, a minimal-endpoint library in which every operation is a self-contained class declaring its route, HTTP verb, allowed roles and processing logic; compared with classic MVC controllers, this isolates each operation of the API and makes it individually testable.

Frontend: standalone Angular application

The frontend uses exclusively standalone components, organised in three conceptual zones: a core zone with API services, data models, route guards and the authentication interceptor; a feature zone grouping functionality by domain (authentication, marketplace, evaluation, user account, administration); and a shared zone of reusable interface components. Global state – in particular the authentication state – is managed with Angular Signals, so that derived values such as “is authenticated” or “is administrator” are computed automatically; routing uses lazy loading of components, and access to sensitive routes is restricted by guards that check both authentication and role.

Security

Passwords are stored only as PBKDF2–SHA-256 derivations with a unique random salt and a high iteration count [10]; on successful login the server issues a signed JWT [11] carrying the user identity and role, and keeps no session state, each request being autonomous in the stateless REST style. Every API operation declares the roles allowed to execute it, and requests lacking the necessary role are rejected before reaching business logic; on the client, an HTTP interceptor attaches the token to protected requests and logs the user out on invalid-session responses, while a restrictive CORS policy accepts requests only from the origin of the Angular application.

THE AUTOMATED VALUATION MODULE

Training data

When a user requests an estimate for a property, the valuation service assembles a training set from two internal sources: completed manual appraisals (status Completed, with a strictly positive appraised value) and listings with a specified asking price, excluding in both cases the target property itself. The rationale follows from valuation theory: values established by specialists approximate the correct value of a dwelling, while asking prices reflect the market’s perception, and combining the

two approximates, in simplified form, a comparative market analysis carried out over the entire database rather than over a handful of manually selected comparables.

Pipeline and model

If fewer than a minimum number of records are available (five in the current implementation), the request is rejected with a validation error, since a model trained on a very small sample would not produce a trustworthy estimate; the interface communicates this state explicitly instead of returning an unreliable figure. Otherwise the data are split into training and test sets (80/20), and for each dwelling the following features are extracted: surface, number of rooms, number of bathrooms, year of construction, and the categorical variables property type, city and county, encoded with one-hot encoding. The concatenated feature vector feeds a FastTree regression model [4, 5] configured with 100 trees, 20 leaves per tree and a minimum of one example per leaf; performance is measured on the held-out test set with the coefficient of determination (R^2), the mean absolute error (MAE) and the root mean squared error (RMSE). The three metrics are recomputed at every request and returned with the response, so the quality of the model is observable as the internal dataset grows rather than being fixed at deployment time. Table 2 summarises the configuration and the evaluation protocol of the valuation model.

Table 2. Configuration and evaluation protocol of the automated valuation model.

Element	Value
Algorithm	FastTree gradient-boosted regression trees (ML.NET)
Hyperparameters	100 trees; 20 leaves/tree; min. 1 example/leaf
Numerical features	surface (m ²), rooms, bathrooms, year built
Categorical features	property type, city, county (one-hot encoded)
Training data	completed appraisals + priced listings, target property excluded
Minimum data threshold	5 records; below it the request is refused explicitly
Train/test split	80% / 20%, fixed seed
Reported metrics	R^2 , MAE, RMSE on the held-out test set
Returned output	estimate; interval = estimate $\pm$ RMSE (≥ 0); training rows; model name
Example run (Fig. 4)	26 rows $\rightarrow$ €210,000, interval €190,000 – €220,000

Transparent output

The response returned to the user contains the point estimate, a confidence interval derived from the RMSE of the model (bounded below at zero), the number of records used for training and the name of the model. Exposing these elements is a deliberate interaction design choice: the figure presented to the user is inspectable, its provenance disclosed and its precision bounded by an explicit interval, which supports calibrated rather than blind trust in the automated advice [6, 7]. The flow is synchronous and stateless – the model is trained on demand, used for a single prediction and discarded – so every estimate reflects the most recent state of the database.

A worked example

Figure 4 (Section 7) shows a representative interaction. A user creates an appraisal request for a three-room, 85 m² apartment built in 2023 in Craiova, Dolj county. At that moment the database contains 26 usable records – completed appraisals and priced listings – which pass the minimum-data threshold; the FastTree model trained on them returns a point estimate of €210,000 and the range shown in the figure, derived from the error measured on the held-out split, whose metrics are reported in the last row of Table 2. With only 26 records these values characterise the current data volume rather than the accuracy attainable by the approach, and since they are recomputed at every request the reported quality of the model follows the growth of the internal dataset. A user who sees a wide range built from 26 records can weigh the figure accordingly, which is precisely the calibration effect the presentation is designed to produce; the same request issued when fewer than five records exist yields not a number but an explicit message that a reliable estimate cannot yet be generated.

USER INTERACTION DESIGN

Browsing and searching listings

The public entry point of the application is the marketplace page, which presents active listings as cards with the essential facts of each dwelling: type, location, surface, number of rooms and price, together with the derived price per square metre. Optional filters – city, property type, price interval, minimum number of rooms and bathrooms – are translated by the backend into a combined database query with paginated results (Figure 2), and selecting a card opens a detail page with the full information of the property, its image gallery and, when available, its valuation.

The screenshot shows the 'MARKETPLACE Available Properties' interface. It includes a search bar with filters for City (Craiova), Type (Any type), Min price (150000), Max price (250000), Bedrooms (3), and Bathrooms. Below the filters, there is a result card for an 'Apartment' with a price of €210,000, area of 85 sqm, 3 bedrooms, and 1 bathroom. The card also shows a price range and a status of 'Active'.

Figure 2. Marketplace page: search filters and a result card with the essential facts of the listing.

Publishing a listing

To sell a dwelling, an authenticated user registers the property and converts it into an active listing through a single guided form (Figure 3), which collects the address, the physical characteristics and the photographs, validates plausibility constraints at the API level (for instance on

surface and construction year) and creates the address, property and listing records in one flow. Because the listing is an entity separate from the property, its lifecycle – active, inactive, expired, sold – is managed independently, and a property can be re-listed after a previous listing was marked as sold. A dedicated account area lists the user’s own listings with per-status counters and inline editing (Figure 5).

The screenshot shows the 'SELLER WORKSPACE List a Property' form. It includes fields for Street address, City, County, Property type, Area, Bedrooms, Bathrooms, and Year built. There is a section for 'Photos' with an 'Add photos' button and a 'Listing' section with 'Title', 'Asking price', and 'Status' fields. A 'Create listing' button is at the bottom.

Figure 3. Seller workspace: registering a property and creating its listing in a single guided form.

Photographs are uploaded to a dedicated endpoint and referenced in the database; the concrete storage mechanism – currently the local file system of the server – is isolated behind a generic interface, so that it can later be replaced by a cloud storage service without changes to the rest of the application.

Two complementary valuation channels

The platform deliberately offers two ways of obtaining a value for a dwelling, with different temporal and epistemic properties. The manual appraisal, whose stages were described in Section 4, is slow and social: at every step the API checks the role of the caller and, at completion, the identity of the assigned evaluator, administrators being allowed to intervene in any evaluation. The requester follows the state of their requests in a dedicated page, while evaluators have a workbench listing the requests assigned to them.

The automated estimate, in contrast, is instantaneous and provisional. It is offered immediately after an appraisal request is created, so that the user receives a data-driven reference point while waiting for the expert (Figure 4), and it is also available on demand for owned properties and active listings. The presentation follows the transparency principles discussed in Section 6: the estimate is accompanied by its range, the model name and the size of the training set, and when historical data are insufficient the system states so explicitly instead of fabricating a number. The heuristic inspection reported in Section 9 nevertheless showed that the meaning of the range is not self-evident to a lay user; attaching to the estimate card a short contextual explanation – that the interval reflects the error of the model on data it has not seen, and widens when comparable records are scarce – is the first interface

change scheduled for the next iteration. The two channels are thus positioned as complementary: the instant estimate anchors expectations and supports pricing decisions, while the completed expert appraisal remains the authoritative value – and, once completed, feeds back into the training data of the model, gradually improving future estimates.

Figure 4. Instant estimate shown after creating an appraisal request: point value, range, model name and training-set size are all disclosed.

Figure 5. Seller workspace: the user’s own listings with status counters.

Role-adaptive interface

The interface adapts continuously to the authentication state and role stored in the token: the menu and the available actions differ between a visitor, a client, an evaluator and an administrator, unauthorised navigation is intercepted by route guards that redirect to the login page while preserving the originally requested URL, and expired sessions are detected by the HTTP interceptor, which logs the user out automatically. The visible interface therefore remains consistent with the permissions actually enforced by the API.

TESTING AND VALIDATION

The testing strategy treated the two components differently, according to the nature of their risks. The frontend is covered by an automated suite written alongside development: 24 specification files totalling 72 test cases, all passing. They cover the API services (properties, listings, evaluations, authentication, saved properties), executed against a simulated HTTP client to verify that each service builds the correct route, verb and query parameters; the security mechanisms, with the authentication guard tested on both branches and the interceptor tested to attach the authorisation header to protected requests but not to the authentication routes; and the page components, verifying construction, form

validation and the expected service interactions with simulated dependencies.

The backend was validated through systematic manual testing of the API, using the interactive documentation interface generated from the endpoint definitions, and through end-to-end walkthroughs of the complete system: registration and login, property creation with image upload, listing publication and its visibility under the search filters, the full manual appraisal lifecycle with the correct state transitions at each step, and the automated estimate in both its regimes – with sufficient data, and with insufficient data, where the explicit refusal message was verified. Role-based authorisation was tested with accounts of all three roles, verifying that restricted operations return a forbidden response for users without the necessary role and succeed for those with it, and that unauthenticated requests to protected routes are rejected.

The non-functional requirements were validated implicitly by the same scenarios: CRUD operations responded without perceptible blocking of the interface, on-demand training completed within the limits of a synchronous API response for the moderate data volumes involved, all protected endpoints required a valid token with issuer, audience and lifetime validation, and the interface adapted to desktop and mobile viewport sizes. This level of assurance is adequate for a prototype but not for deployment: the backend has no automated regression suite, and building one at the API level – covering the authorisation matrix of Table 1 and both regimes of the valuation endpoint – is a prerequisite for further development of the platform.

HEURISTIC USABILITY EVALUATION

To complement functional testing with a usability perspective, we performed an expert-based heuristic evaluation of the interface against Nielsen’s ten usability heuristics [15, 16]. The inspection was carried out by a single expert evaluator – the second author, who was not involved in the implementation – over four task scenarios covering the roles of the platform: browsing and filtering listings as a visitor; registering a property and publishing a listing as a standard user; requesting an appraisal and reading the instant estimate as an owner; and assuming and completing an appraisal from the pending queue as an evaluator. Each scenario was walked through twice, once to become familiar with the flow and once to record problems; every problem was assigned to the heuristic it violates and rated on Nielsen’s 0–4 severity scale. Because a single evaluator was involved, no inter-evaluator agreement could be computed, and Nielsen recommends three to five evaluators to surface the majority of the problems; the results below are therefore an early prediction of the difficulties users are likely to meet, not a validated inventory. Table 3 reports, for each heuristic, the mechanism of the interface that addresses it and the problem identified, with its severity.

Table 3. Heuristic evaluation: supporting mechanisms and problems identified, rated on Nielsen’s 0–4 severity scale.

Heuristic	Supporting mechanism	Problem identified (severity)
1. Visibility of system status	Status badges and per-status counters; estimate card discloses model, interval, training rows	No progress indication during image upload and on-demand training (2)
2. Match with the real world	Appraisal-practice vocabulary (listing, asking price, evaluation); price per m ²	None identified
3. User control and freedom	Resettable filters; editable listing lifecycle; re-listing after sale	Destructive operations have neither confirmation nor undo (3)
4. Consistency and standards	Uniform card layout, standard form controls, uniform mapping of API errors to messages	None identified
5. Error prevention	Plausibility validation; minimum-data threshold prevents unreliable estimates	Plausibility rules enforced only server-side, so violations surface after submission (2)
6. Recognition rather than recall	Persistent visible filter values; labelled status chips	None identified
7. Flexibility and efficiency of use	Deep-linkable guarded routes preserving the target URL; saved-properties list	None identified
8. Aesthetic and minimalist design	Cards restricted to the essential facts of each dwelling	None identified
9. Error recognition and recovery	Insufficient-data message states the reason; 403/404 mapped to readable messages	None identified
10. Help and documentation	None: the interface relies on the self-evidence of its forms	No onboarding or contextual help; the estimate interval is unexplained where it is shown (3)

Four problems were recorded, all of minor to moderate severity and none of them blocking a main task flow. The two most severe concern the same lay-user assumption: the interface offers no onboarding and no contextual help, so the meaning of the estimate interval – the element on which the interaction design rests – is left unexplained where it is shown, and destructive operations are executed without confirmation or undo. Both are addressed in the next iteration, the first by an explanatory affordance attached to the estimate card, the second by a confirmation dialogue with a short undo window. Progress indication for image upload and on-demand training, and inline validation mirroring the server-side plausibility rules, complete the list; none of the four requires an architectural change. An inspection of this kind predicts where users are likely to struggle, but it cannot show whether the transparent presentation of the estimate actually calibrates their trust – that question requires the user study outlined in Section 10.

DISCUSSION AND LIMITATIONS

Compared with the incumbent Romanian portals, which publish listings and market statistics but delegate valuation to certified partners, on request and for a fee, PropertyEval makes a valuation available to every user at the moment of interest and at no cost, while keeping the expert channel for the cases that justify it. It disposes of incomparably less data than the large international AVMs, but differs from them in two respects that do not depend on data volume: the automated figure is never presented alone – interval, model and training size accompany it, and the system refuses to answer below a data threshold instead of returning a figure of unknown quality – and the two valuation channels are coupled, each completed expert appraisal entering the training set of the model. The automated channel therefore improves as a by-product of the manual one, without external data acquisition.

The on-demand training strategy has an attractive simplicity – no model versioning, no staleness, no serving infrastructure – but it does not scale: the training cost grows with the volume of accumulated data, repeated requests retrain identical models, and the work is performed inside a synchronous HTTP call, so the response time of the estimate degrades exactly as the platform becomes successful. The evolution is clear and does not affect the interaction design: persist the model, retrain it periodically or when a threshold of new records is reached, serve predictions from the persisted model behind a cache, and display the training date alongside the provenance information already shown. Relatedly, the confidence interval is currently derived from the global RMSE of the model rather than from the uncertainty of the individual prediction; quantile regression or conformal prediction would provide per-property intervals that better reflect local data density, while feature-importance or SHAP-based explanations [17] would extend the transparency of the estimate from how much the model can err to which characteristics of the dwelling drove the figure.

A second limitation concerns the data source: the model is trained exclusively on the platform's internal records, so the accuracy of its estimates is bounded by the volume of appraisals and listings accumulated – the classic cold-start problem of an internal AVM, in contrast with mature platforms that dispose of large historical datasets. The consequence for deployment is concrete: early on, or in a city with few records, the model either refuses to answer or answers from a handful of comparables that need not represent the local market, so the coverage of the automated channel is uneven in a way the user cannot see. Weighting comparables by locality, applying the minimum-data threshold per city rather than globally, and integrating external market data are the mitigations we consider. Third, and most important for the interaction claim, the transparency-oriented presentation is grounded in established human-AI interaction guidelines but its effect on users has not been measured: a controlled study comparing three presentation variants – point estimate only, estimate with interval, estimate with full provenance

– on trust calibration and on the asking price users eventually set is the main direction of empirical work, and until it is run the interaction design remains motivated rather than validated.

Beyond its limitations, the design generalises to other consumer domains in which an authoritative human judgement coexists with a cheap algorithmic approximation – used-car pricing, art or collectible appraisal, insurance pre-assessment. The pattern is the same: separate the asset from its commercial lifecycle, feed the model with both expert judgements and market signals, train on demand while data are scarce, and present the automated figure with its uncertainty and provenance, refusing to answer below a data threshold.

CONCLUSION

We presented PropertyEval, a web platform that integrates listing management with a dual valuation mechanism: a staged, role-based workflow for expert appraisal and an instant, transparently presented automated estimate produced by a gradient-boosted regression model trained on demand on the platform's own data. The architecture – an Angular single-page client over a Clean Architecture ASP.NET Core backend with an embedded ML.NET valuation service – supported the complete set of flows, validated through an automated frontend suite, systematic end-to-end testing and a heuristic inspection of the interface. Beyond the artifact itself, the work illustrates a design stance for consumer-facing algorithmic advice: expose the uncertainty and the provenance of the estimate, decline to answer when the data do not support an answer, and position the automated figure as a complement to – not a substitute for – human expertise. Future work targets per-prediction uncertainty, model persistence and caching, external data integration, backend test automation, and the user study that would show whether this presentation actually calibrates trust and pricing behaviour.

REFERENCES

- [1] E. Pagourtzi, V. Assimakopoulos, T. Hatzichristos, and N. French. 2003. Real estate appraisal: a review of valuation methods. *Journal of Property Investment & Finance* 21, 4, 383–401.
- [2] N. Kok, E.-L. Koponen, and C. A. Martínez-Barbosa. 2017. Big data in real estate? From manual appraisal to automated valuation. *The Journal of Portfolio Management* 43, 6, 202–211.
- [3] J. H. Friedman. 2001. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics* 29, 5, 1189–1232.
- [4] C. J. C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: an overview. *Microsoft Research Technical Report MSR-TR-2010-82*.
- [5] Z. Ahmed et al. 2019. Machine learning at Microsoft with ML.NET. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*, 2448–2458.
- [6] S. Amershi et al. 2019. Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*, Paper 3, 1–13.
- [7] J. D. Lee and K. A. See. 2004. Trust in automation: designing for appropriate reliance. *Human Factors* 46, 1, 50–80.
- [8] R. C. Martin. 2017. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.
- [9] R. T. Fielding and R. N. Taylor. 2002. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology* 2, 2, 115–150.
- [10] B. Kaliski. 2000. PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898, IETF.
- [11] M. Jones, J. Bradley, and N. Sakimura. 2015. JSON Web Token (JWT). RFC 7519, IETF.
- [12] ISO/IEC. 2011. ISO/IEC 25010:2011 – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.
- [13] T. Chen and C. Guestrin. 2016. XGBoost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, 785–794.
- [14] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. 2017. LightGBM: a highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30 (NIPS '17)*, 3146–3154.
- [15] J. Nielsen and R. Molich. 1990. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90)*, 249–256.
- [16] J. Nielsen. 1994. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*, 152–158.
- [17] S. M. Lundberg and S.-I. Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30 (NIPS '17)*, 4765–4774.