DSE3101 HDB Decode Technical Documentation

Team Member: Hang Yizhou, Hu Shiqi, Li Minyi, Qin Mulan, Sow Yun Tsing, Su Zhimin

Part I Overview

1.1 Project Overview

Buying or selling an HDB flat in Singapore presents several challenges. For users interested in specific units, it is often difficult to determine a fair market price, as listing platforms may reflect inflated asking prices rather than actual transaction values, particularly in non-seller's markets. For general buyers, the process can be overwhelming due to the number of factors to consider, such as budget, location, flat type, and amenities. With information spread across multiple sources, users may find it difficult to make informed and confident decisions. Therefore, we aim to develop an intuitive and interactive digital platform that helps users confidently navigate the HDB resale market.

HDB Decode is implemented using Streamlit to provide users with a smooth, intuitive experience as they navigate the HDB resale market and make informed, data-driven decisions.

Target users:

1. HDB homeowners looking to sell or assess the value of their current flat
2. Buyers with specific flats in mind who want to evaluate whether the asking price is reasonable
3. General buyers exploring suitable HDB flats based on budget, amenities, and lifestyle needs

Key features include:

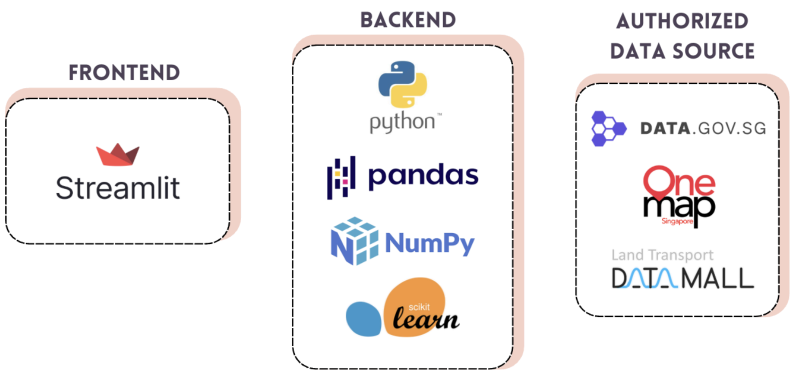
1. Resale price trends for both general market analysis and filtered views based on user-selected criteria such as location, flat type, or remaining lease
2. Resale price prediction for specific HDB units
3. Recommendations of ideal HDBs based on user preferences
4. Map-based exploration to help users evaluate HDBs of interest by location

Our project repository can be found at: https://github.com/yizhouhang/dse3101-hdbdecode

To access our app:

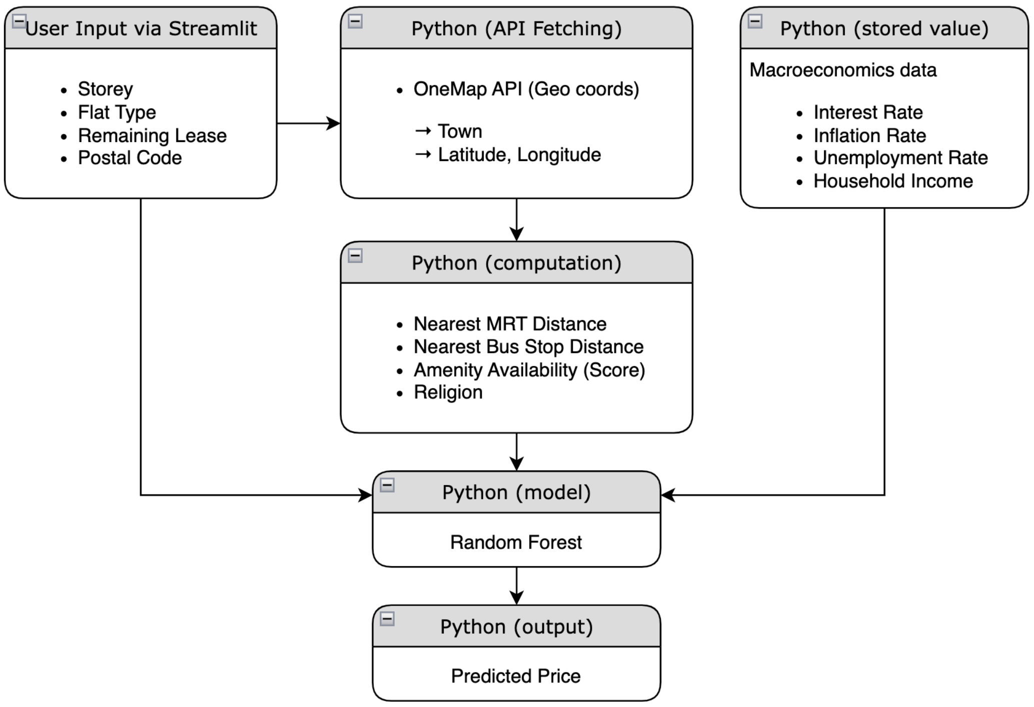
1. Ensure Docker Desktop is open
2. Run the following lines of code:
    - docker pull yuntsingg/hdb-decode-app
    - docker run -p 8501:8501 yuntsingg/hdb-decode-app
3. Open http://localhost:8501 in your browser

Project Tools:



1.2 Data Architecture

Below is an overview of the data flow between user inputs, data processing, modeling, and output display:



1.3 Data and Methodology

Data Collection

Our project utilised a range of datasets sourced from Singapore's open data platforms and APIs:

- **HDB Resale Flat Prices and Room Attributes**: data.gov.sg
- **Temporal Macroeconomic Indicators**: data.gov.sg.
- **Geospatial Data**: OneMap API
- **Demographic Data**: data.gov.sg.

Data Processing

We removed duplicates and missing values and excluded non-predictive identifiers from the dataset for prediction. Categorical features were encoded using One-Hot Encoding, and numerical variables were normalised and standardised via a pipeline using ColumnTransformer.

Feature Engineering
We engineered features across four dimensions: room attributes, temporal, spatial, and demographics. Key transformations included conversion to numeric, inflation computation, and distance-based amenity scoring and transportation accessibility metrics using the Haversine formula. In the end, features with high multicollinearity or low significance were removed via VIF and statistical testing was used to evaluate their predictive power.

Modeling
We tested linear models, machine learning (e.g., Random Forest, CatBoost), and lightweight neural networks. Random Forest outperformed others ($R^2 \approx 0.973$, RMSE ≈ 0.02) and was selected as the final model.

Part II Backend

2.1 Pre-processing

To ensure our resale price prediction model accurately captures the multidimensional factors influencing HDB prices, we curated and engineered a diverse set of features grounded in both domain knowledge and data availability. These features span four key dimensions – room attributes, temporal indicators, spatial context, and demographic composition. Each contributes to a more holistic representation of resale value determinants. This section details the rationale behind feature selection and the methods used to compute or transform them for modeling.

2.1.1 Cleaning

Remove duplicates and missing values
We removed any duplicated rows from the resale transaction dataset and conducted a column-wise check for missing values to eliminate redundant entries.

Feature Reduction
Non-essential identifiers such as block and address were dropped from the modeling dataset to reduce dimensionality, as they do not provide generalizable predictive value.

However, we acknowledge that address-level information is important for user-facing features. Therefore, we retained this information in a separate file — hdb_geospatial.csv—which preserves key location attributes for each unique HDB address. This file includes the address, latitude, longitude, town, distances to the nearest MRT and bus stops, names of those nearest stops, and amenity scores.

2.1.2 Feature Sourcing and Engineering

Room Attributes
Lease Duration Conversion
The `remaining lease` field originally appeared as a string (e.g., "94 years 5 months"). We created a function `convert_to_years()` to convert these values into a float representing total lease years. This standardized the feature for numerical analysis.

Flat Type Mapping
The `flat type` field was converted into numeric codes based on a defined mapping. For types in the format “n ROOM,” the value was directly encoded as n. Two exceptions were handled separately: Executive flats were encoded as 6, and Multi-generation flats as 7.

Storey Range Simplification
The `storey range` field (e.g., "04 TO 06") was simplified by extracting the lower bound as a proxy variable, converting it to an integer (e.g., 4).

Temporal Data

Inflation Rate: (monthly, calculated by CPI)

We obtained the Consumer Price Index (CPI) data with 2019 as the base year from data.gov.sg. To produce an index relative to the most recent data in our timeframe, we rebased the CPI by dividing each month's CPI by the CPI value of December 2024 (the latest data), treating it as the reference point. We then calculated the monthly inflation rate as the percentage change between consecutive months, as shown in the formula:

$$Inflation_t = \frac{CPI_t - CPI_{t-1}}{CPI_{t-1}} \times 100$$

We initially kept both the CPI (based on 2024.12) and the monthly inflation rate as variables. After comparing the model performance under three conditions: (1) including CPI only, (2) including inflation rate only, and (3) including both, it turned out that keeping inflation rate only yielded the highest R-squared, indicating superior explanatory power.

Resident Unemployment Rate

The Resident Unemployment Rate is sourced from the dataset “Resident Long-Term Unemployment Rate (Annual)” available on data.gov.sg, published by the Ministry of Manpower. As of processing, the dataset is current up to January 2025, with an annual update frequency. Although other datasets include total population or citizen-only unemployment rates, the resident unemployment rate, which includes both citizens and permanent residents, was selected. This better represents the demographic relevant to resale HDB flat buyers. Additionally, quarterly short-term unemployment rates were excluded, as they are often affected by seasonal fluctuations and do not accurately reflect longer-term macroeconomic trends. The long-term annual rate serves as a more stable indicator of sustained economic conditions.

Interest Rate

The interest rate data is taken from “Current Banks Interest Rates (End of Period), Monthly”, last updated in March 2025. The data is recorded monthly, capturing interest rates at the end of each month. Among the available indicators, the Singapore Overnight Rate Average (SORA) was selected for its clarity and interpretability. As a widely used benchmark rate, SORA provides a useful macroeconomic signal that reflects the cost of borrowing and overall economic sentiment.

Year

The `year` is derived from the resale transaction date to align each observation with annual macroeconomic indicators such as unemployment rate and property availability.

Month

The `month` is extracted separately from the year and included as a standalone numerical feature (ranging from 1 to 12). This captures seasonality in the housing market, where transaction volumes tend to be higher in certain months, particularly in the second and third quarters of the year.

Private Property

This data, sourced from a quarterly-updated dataset on data.gov.sg, reflects the current market supply of private housing and indirectly captures overall housing demand. Since private properties are a common substitute for resale HDB flats, this feature serves as a useful proxy for estimating demand in the resale market. Its strong performance in our model likely stems from its relevance to buyer decision-making and its ability to capture supply-demand dynamics without introducing multicollinearity.

Spatial Data

Town

To incorporate the important categorical variable `town` into our model, we applied one-hot encoding, which converts each town into a binary feature. To avoid the dummy variable trap and reduce multicollinearity, we dropped the first town alphabetically – "Ang Mo Kio" – as the reference category.

Distance to Nearest MRT/Bus

To quantify public transport accessibility, we engineered two location-based features: `distance to nearest bus stop` and `distance to nearest MRT station`. HDB addresses were geocoded via the OneMap API, and public transport coordinates were sourced from cleaned external datasets. Distances were computed using the Haversine formula, which provides accurate great-circle distances based on latitude and longitude:

$$d = 2r \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

where

- ϕ_1, ϕ_2 are latitudes of two locations (in radians)
- λ_1, λ_2 are longitudes of two locations (in radians)
- r is the Earth's radius (≈ 6,371,000 m)

The shortest distance to each transport mode was stored as a numerical predictor for resale price modeling.

Amenity Scores (Five Categories)

We computed the amenity scores using a customized formula across five categories – healthcare, food, education, shopping, and recreation. For each HDB unit, we computed five corresponding amenity scores. The score increases when amenities are more numerous and located closer to the flat. We came up with the formula:

$$S_{amenity-type} = \sum \frac{1}{max(d_{ij}, \epsilon)}$$

- $S_{amenity-type}$: Total accessibility score for one amenity type
- i: HDB flat
- j: facility
- d_{ij}: Distance from HDB flat i to facility j (meters)
- ϵ: Minimum distance threshold (50 meters) to avoid division by very small values

To compute the distance between each HDB flat and each facility,

1. We first retrieved the coordinates of all the HDB flats using their addresses (available in the publicized HDB resale price dataset) via the Onemap API.
2. For the amenity locations, we obtained the data from various public sources (mainly government databases) and then retrieved the coordinates via the same API.
3. Pairwise distances were calculated using the same Haversine formula (as mentioned above) with the coordinates.
4. To focus on the nearby amenities, we filtered out those located more than 3 km away from the HDB flat. We only considered the amenities within a 3 km radius for the amenity score computation.

Demographic Data

Average monthly household income

Income data was sourced from the General Household Surveys 2015 and 2020, which provide the average monthly household income by planning area. For years between or beyond 2015 and 2020, we used data from the nearest available year. This feature captures the socioeconomic profile of each area, allowing us to account for income-related preferences or affordability constraints in housing selection and pricing analysis.

Ethnicity

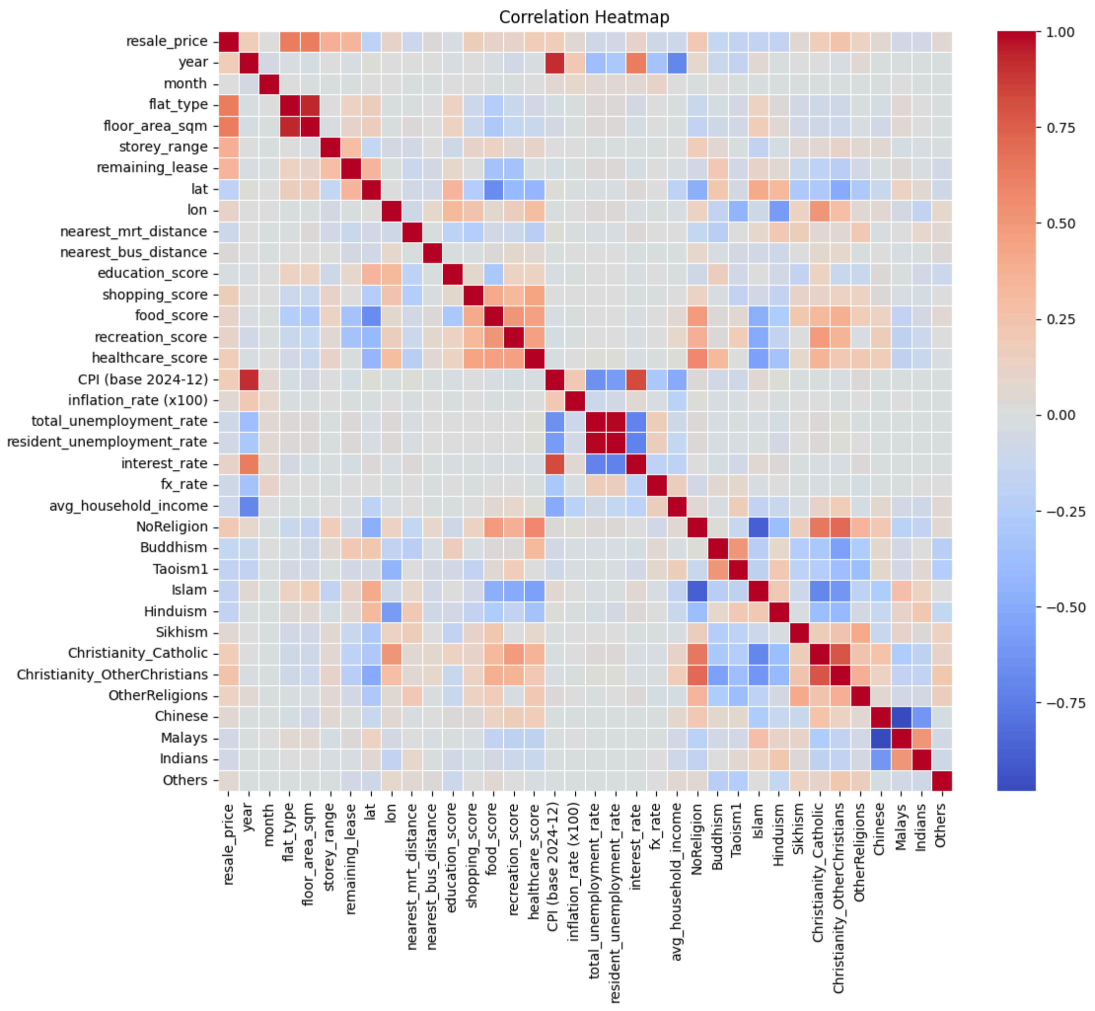
Ethnic composition data was sourced from the General Household Surveys 2015 and 2020, which report the number of residents by race in each planning area. We computed the proportion of each racial group as normalized features. For years between or beyond 2015 and 2020, we used data from the closest available year. This feature reflects Singapore's Ethnic Integration Policy (EIP), which imposes race-based eligibility restrictions on HDB purchases, allowing us to incorporate this regulatory constraint into our analysis.

Religion

This dataset was sourced from the General Household Survey 2015 and 2020, which provides the number of residents affiliated with each religion within each planning area. From this data, we computed the proportion of each religion within every area to create a set of normalized features. By incorporating this feature, we aim to capture the proximity preference to places of worship, community clusterings, and cultural constraints (avoiding specific floor levels or postal codes due to beliefs or superstitions).

2.1.3 Feature Selection

To address potential multicollinearity, we examined the correlation matrix and computed the Variance Inflation Factor (VIF) for each predictor. Here are the predictors with VIF values greater than 10, indicating a high degree of multicollinearity: `remaining_lease`, `floor_area`, `flat_type`, `ethnicity`, `foreign_exchange_rate`, and `CPI`.



From the correlation heatmap, `floor area` and `flat type` were found to be highly correlated. Given that flat type provides more interpretable categorical detail, we retained it and excluded `floor area`. This removal significantly reduced the VIF for flat type. Similarly, `resident unemployment rate` and `total unemployment rate` were highly correlated. Based on domain relevance to HDB housing, we retained `resident unemployment rate` and excluded `total unemployment rate`.

We also conducted **statistical significance testing** using linear regression and evaluated the p-values of each variable. Features such as `ethnicity` and `foreign exchange` rate showed p-values much greater than 0.05, suggesting limited explanatory power. These variables were therefore excluded from the final model. Similarly, we excluded `CPI` as it did not improve the model significantly. We retained `remaining_lease` due to its strong predictive importance.

2.1.4 Final Variables & Data Sources

Category	Variable Name	Type	Description	Data Source
Room Attributes	Flat Type	Categorical	Type of HDB flat (e.g., 3-room, 4-room)	Data.gov.sg (Resale flat prices based on registration date from Jan 2017 onwards)
	Town	Categorical	Estate location of the flat	
	Remaining Lease (years)	Numerical	Number of years left on the lease	
Temporal	Year, Month	Numerical	Resale year and month	
	Inflation Rate	Numerical	Monthly inflation derived from CPI	Data.gov.sg (CPI)
	Interest Rate	Numerical	Prevailing mortgage interest rate	Data.gov.sg (Interest Rate)
	Unemployment Rate	Numerical	Monthly resident unemployment rate	Data.gov.sg (Unemployment)
	Household Income	Numerical	Average monthly household income	Data.gov.sg (income 2015) Data.gov.sg (income 2020)
Spatial	Distance to MRT/Bus	Numerical	Distance to nearest station/stop (meters)	OneMap API reverseGeocode
	Healthcare Score	Numerical	Custom accessibility scores for doctors, dentists, clinics, hospitals, pharmacies	Humanitarian Data Exchange (HDX) sg healthcare
	Food Score	Numerical	Custom accessibility scores for food courts	Data.gov.sg (Hawker Centers)
	Education Score	Numerical	Custom accessibility scores for education (primary, secondary, junior college, centralised institutes, mixed levels)	Data.gov.sg (Schools)
	Shopping Score	Numerical	Custom accessibility scores for shopping malls, supermarkets	Kaggle (Shopping Malls) Data.gov.sg (Supermarkets)
	Recreation Score	Numerical	Custom accessibility scores for parks, gyms, libraries	Onemap API Themes
Demo-graphic	Religion	Numerical	Town-level population of each religion	Data.gov.sg (Religion 2015) Data.gov.sg (Religion 2020)

2.1.5 Categorical Data—One-Hot Encoding

The categorical variables in our dataset, such as `town`, were encoded using One-Hot Encoding. This transformed the single categorical column into multiple binary columns.

2.1.6 Numerical Data—Normalization and Standardization

We applied min-max normalization to all numerical predictors to bring them onto a consistent scale, typically between 0 and 1. The transformation is defined as,

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This step is essential for distance-based models or those sensitive to feature magnitude, such as regularized regressions or neural networks. Normalization ensures that no single feature disproportionately influences the model due to its scale. Normalized numerical features were further standardized using StandardScaler to ensure consistent scaling across features. These preprocessing steps were implemented using a ColumnTransformer, which was then integrated into a Pipeline alongside the machine learning model.

2.2 Modeling and Analysis

2.2.1 Train-Test Split

To ensure a fair and consistent comparison across different modeling strategies—linear regression, machine learning, and deep learning—we applied the same 80/20 train-test split across all experiments. A fixed random seed was used to maintain reproducibility and ensure that all models were trained and evaluated on identical data partitions. This setup allows for objective performance benchmarking across various algorithms.

2.2.2 Linear Regression Models

Multiple Linear Regression (MLR)
We first implemented MLR as a baseline model, using the full set of engineered features. Z-score standardization was applied to all numerical variables. Among three linear models, MLR achieved the best performance in terms of R-squared (0.8743), RMSE (0.1251), and MAE (0.2678). This suggests that regularization was not necessary in this case.

Lasso Regression
Lasso Regression was implemented using LassoCV with cross-validated alpha tuning. It performed automatic feature selection by shrinking some coefficients to zero. However, it slightly underperformed compared to MLR in all evaluation metrics, with an R-squared of 0.9692, an RMSE of 0.1301, and an MAE of 0.2711. It indicates that L1 regularization did not improve model performance.

Ridge Regression
Ridge Regression, applied via RidgeCV, introduced L2 regularization to handle multicollinearity. While it produced more stable coefficient estimates, it did not outperform MLR. The model still showed marginally lower R-squared and higher error metrics. Regularization did not yield performance gains in this case.

2.2.3 Machine Learning

Building upon the statistical baseline models, nine machine learning models have been trained and tested. In this section, three of the representative models, including Random Forest, Support Vector Machine, and CatBoost, will be detailed. They are respectively representative of three distinct learning strategies: bagging, margin optimization, and gradient boosting.

Random Forest—Bagging
Random Forest is an ensemble of decision trees trained using bootstrap samples. It is known for its robustness, resistance to overfitting, and ability to capture non-linear feature interactions. Its R-squared is around 97.3% with an RMSE of 0.02.

Support Vector Machine (SVM)—Margin Optimization
SVM aims to find a function that deviates from actual observations by a value no greater than a defined epsilon. This is especially important when it comes to a dataset with high-dimensional structures. Its R-squared is around 90% with an RMSE of 0.03.

CatBoost—Gradient Boosting
CatBoost is a gradient boosting algorithm that combines multiple shallow and symmetric decision trees built sequentially. In our experiments, CatBoost achieved an R-squared of approximately

96.9% with an RMSE of 0.02. This result was very close to that of Random Forest, but due to its tendency to overfit and longer training time, it was not selected as the best-performing model.

2.2.4 Deep learning Models

We also experimented with three lightweight deep learning architectures using TensorFlow due to computing resource constraints.

Neural Network (NN)
We implemented a feedforward neural network with three hidden layers (128, 64, 32 units, ReLU activation) and a single output node. Features were standardized, and the model was trained using the Adam optimizer with MSE loss for 50 epochs. It achieved strong performance across all evaluation metrics, capturing complex nonlinear patterns in the data effectively. Its R-squared is around 0.8153 with an RMSE of 0.0410.

Recurrent Neural Network (SimpleRNN)
A SimpleRNN model with 50 units and ReLU activation was trained on reshaped input data, using the same settings as the NN. Due to the non-sequential nature of the dataset, the RNN underperformed, showing that temporal modeling offered little advantage for this task. Its R-squared is around 0.6691, with an RMSE of 0.0548.

Long Short-Term Memory (LSTM)
Similar to SimpleRNN, we tested an LSTM with 50 units to assess whether its memory gating improved accuracy. While slightly better than the RNN, it showed no clear advantage over the NN and incurred higher computational cost, making it less suitable for this static tabular dataset. Its R-squared is around 0.6708, with an RMSE of 0.0547.

Model Performance Comparison

Model	R-Squared	RMSE	MAE
Random Forest	0.9731	0.0210	0.0147
CatBoost	0.9694	0.0224	0.0163
Decision Tree	0.9427	0.0307	0.0213
XGBoost	0.9209	0.0361	0.0259
NN	0.9082	0.0289	0.0219
SVM	0.9051	0.0396	0.0319
SGD Regressor	0.8752	0.0454	0.0342
MLR	0.8743	0.1251	0.2678
Ridge	0.8743	0.1251	0.2678
Lasso	0.8692	0.1301	0.2711
Bagging Boost	0.8032	0.0570	0.0437
LSTM	0.7992	0.0427	0.0327
PLS Regression	0.7511	0.0641	0.0490
AdaBoost	0.7511	0.0641	0.0490
RNN	0.7378	0.0488	0.0399

2.2.5 Fitting the Best-Performing Model—Random Forest

After identifying Random Forest as the best-performing model, we performed hyperparameter tuning using Randomized Grid Search. Due to the large size of the dataset, running even a 3-fold cross-validation on the full data was computationally intensive. To address this, we sampled 20% of the dataset for the tuning process. This tuning subset is independent from the testing dataset used during the model selection phase.

The best set of hyperparameters (see Appendix) was then applied to the final model. Subsequently, the model was trained on the full training set and evaluated on the held-out test set. The final performance metrics are as follows:

Average R-Squared	Average RMSE	Average MAE	Average MAPE
0.9681	0.0230	0.0161	6.4150%

To make the model outputs accessible to the frontend, we generated a new dataset that reverts normalized and one-hot encoded features back to their original form for human readability, and includes the predicted resale price for each existing HDB flat. All temporal features (e.g., interest rate, unemployment rate) were set to the most current values, simulating present-day macroeconomic and market conditions. This ensures that users receive predictions that reflect the latest economic environment.

Additionally, the model was saved using the joblib package in a .pkl file, which allows the frontend team to instantly access the model without training the model each time when used. This also ensures that model parameters and scripts remain unchanged during the process of data transfer, eliminating the need to revisit and align the code across teams, which boosts the efficiency of collaboration within the group.

2.3 Challenges & Improvements

2.3.1 Data Limitations

Our development was constrained by several data availability limitations. A complete registry of all HDB flats, including their configurations and availability, was not accessible to us. As a result, we relied solely on resale transaction data, which inherently excludes unsold units, rental flats, and newly launched flats. This limitation directly impacted the design of our "Find Your Ideal Home" feature. We faced two possible approaches: 1. assume that all HDB blocks contain all room types and a wide range of storey heights, and generate synthetic combinations by populating and permuting all possible configurations; 2. restrict predictions only to flat configurations observed in the available resale data, even if it means presenting a more limited selection. After careful consideration, we adopted the second approach. While it may result in a less comprehensive list of options, we believe it provides a more accurate and trustworthy experience. Suggesting hypothetical flats—such as a 3-room unit on a low floor in a block that never offered that configuration—could mislead users and result in disappointment and distrust. Our priority was to ensure that every "ideal home" presented is based on real, verifiable data. To mitigate this limitation, we encourage users to complement our platform with external listing services such as PropertyGuru to access the most up-to-date flat availability.

The "Predict the Price" feature uses macroeconomic data as of December 2024 and demographic indicators, such as household income and religion—from the 2020 census. We currently rely on manually downloaded datasets and have not implemented automated real-time integration.

We also lacked access to forward-looking indicators like projected unemployment, inflation, and interest rates, which limited our ability to incorporate economic expectations into forecasts. Similarly, missing data on BTO launches and future housing supply reduced our ability to reflect upcoming unit availability in user recommendations.

2.3.2 Model limitations

Another challenge was balancing model complexity with computational limitations, especially for deep learning models. Due to constraints in computing resources, we did not extensively experiment with the state-of-the-art architectures. Our deep learning models were simple and lightweight. Future work with stronger hardware or cloud-based infrastructure could enable deeper model exploration like Transformers or high-performing gradient boosting methods, potentially yielding better results.

2.3.3 Improvements

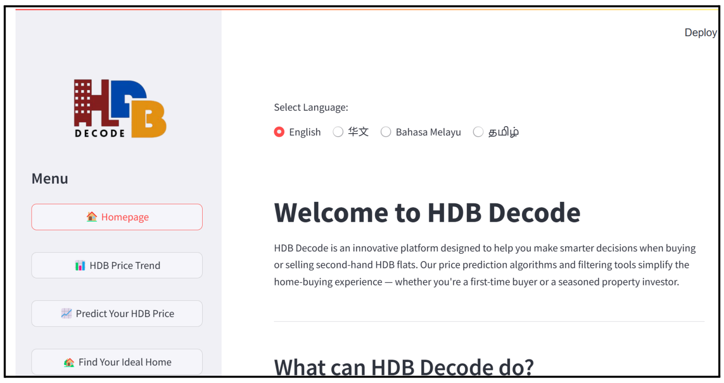
Future improvements may focus on expanding the range of integrated datasets, including those mentioned earlier, through collaboration with third-party platforms. This would enhance the model's contextual awareness and predictive capability. Additionally, an automated, real-time data updating pipeline should be implemented to ensure the system remains accurate and up to date with the latest resale and macroeconomic trends.

From a user experience perspective, the current amenity score feature could be enhanced by displaying a detailed listing of nearby amenities, providing users with more practical and location-specific information during their resale decision-making process.

Part III Frontend

3.1 Overall Structure and Layout

We started by searching for references online to get inspiration on what kinds of features to implement, as well as good UI practices. In coming up with our design methodology, we decided that the overall goal of the frontend team was to ensure that our app is simple, intuitive and accessible for users. To achieve our goal, we prioritised clean visual hierarchies, ensuring that users can easily interpret data without being overwhelmed. Features such as dynamic maps, prediction results, and visual charts were implemented to make interaction both informative and engaging.



Welcome to HDB Decode

HDB Decode is an innovative platform designed to help you make smarter decisions when buying or selling second-hand HDB flats. Our price prediction algorithms and filtering tools simplify the home-buying experience — whether you're a first-time buyer or a seasoned property investor.

What can HDB Decode do?

Figure: Homepage with sidebar

Built using *Streamlit*, the app is divided into four main pages: *Homepage, HDB Price Trend, Predict Your HDB Price,* and *Find Your Ideal Home*. These sections are accessible through the sidebar menu, allowing users to transition seamlessly between the features of HDB Decode. The appendix contains screenshots of each page in HDB Decode.

The ordering from *Homepage* → *HDB Price Trend* → *Prediction* → *Find Your Ideal Home* is intentionally designed to follow a logical progression that accompanies users step-by-step in their HDB resale journey. Users typically begin by exploring market trends, before moving on to fetching a predicted price before making decisions, mirroring a real-world decision-making process.

Our group followed the framework provided by Jakob Nielsen's 10 principles for interaction design.

3.2 Design Decisions

Simplifying Choices: A Miller's Law Approach

To avoid overwhelming users, we applied Miller's Law, which suggests that individuals can hold only about 7 (±2) items in their working memory at a time. By intentionally limiting the number of visible options across all implemented features, we aim to reduce cognitive load.

Layout of Filters Based on Industry Standards

Our initial design allowed users to select up to 5 optional filters before showing any criteria, giving them full autonomy over their search priorities. However, testing revealed that this approach was unfamiliar and unintuitive compared to common user expectations.

According to Jakob's Law, users would expect similar layouts to familiar apps, hence we analysed the frontend design of established property platforms' (e.g., PropertyGuru, 99.co, Zillow) and adapted our layout to adhere to user patterns. Key filters, namely budget and flat type are displayed at the top by default. Additional filters are now tucked into collapsible sections, enabling a cleaner interface while still supporting advanced customisation.

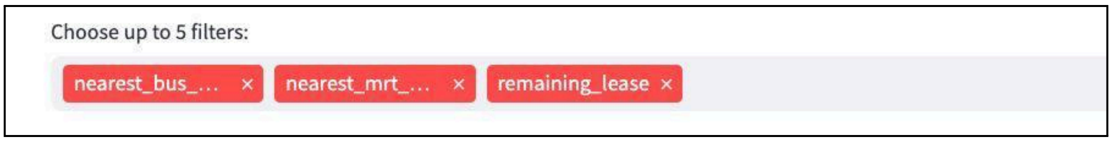

Choose up to 5 filters:

Figure: Initial Design

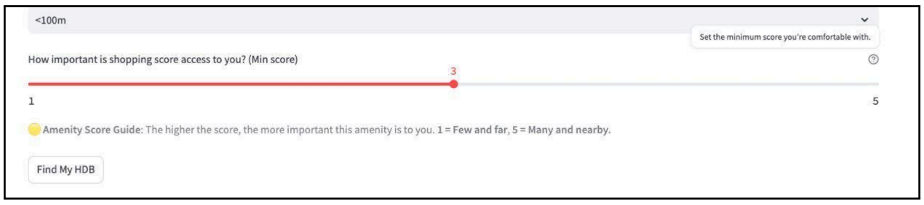
Adapting Sorting Criteria to User Preferences

After filtering, our initial approach is to rank all the selected HDBs by lowest price, so the cheapest HDB that satisfies all the conditions would be the priority. However, usability testing indicates that price isn't always the most important factor, especially when all listings fall within a user's budget.

To accommodate different preferences, we introduced multiple sorting options. Users can now sort results based on what matters most to them: price, amenity score, or flat age. This makes the selection process more personalised and effective.

Making Amenity Filters More Intuitive

Initially, amenity filters used a sliding selector. We intended for the filter to represent a continuous range, since users might be looking for flats with a minimum level of nearby amenities (e.g., a score of 3 and above). However, users often misinterpreted it. For example, selecting a minimum score of 3 looked more like a filter for flats with scores between 1 and 3. To minimise confusion, we replaced the slider with checkboxes, allowing users to explicitly select their desired score, reducing ambiguity.


<100m

Set the minimum score you're comfortable with.

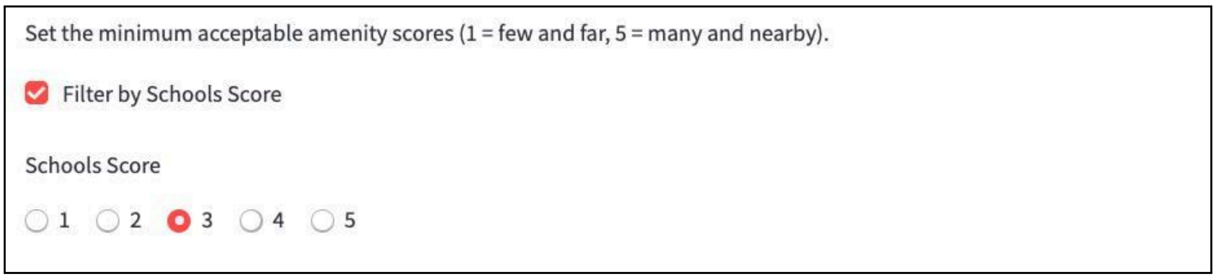
How important is shopping score access to you? (Min score)

3

1 5

Amenity Score Guide: The higher the score, the more important this amenity is to you. 1 = Few and far, 5 = Many and nearby.

Figure: Initial Layout


Set the minimum acceptable amenity scores (1 = few and far, 5 = many and nearby).

☑ Filter by Schools Score

Schools Score

1 2 [✓ 3] 4 5

Figure: Current Layout

Collapsible Additional Information

Understanding the importance of clarity, we included both a simplified amenity score guide and a collapsible “What is Amenity Score?” section. This dual approach supports both casual users (who need a quick understanding of the score) and detail-oriented users (who are interested in the specifics, such as what a score of 5 in schools really means).

Improving Discoverability Through Layout Adjustments

Initially, the *Compare via Postal Code* feature was implemented directly below the *Compare HDBs* feature, based on the assumption that users would want to explore locations immediately after browsing the listings. However, we realised that users may need to scroll for a long time before reaching this section, which could decrease the discoverability of our feature. Consequently, we put the *Compare via Postal Code* feature into a separate tab, while still providing clear visual cues and reminders for users to check it out to support recognition. This preserves the intended user flow without compromising usability.

3.3 Integration of frontend and backend components

In our project, the frontend and backend work closely together to deliver a smooth user experience. The backend provides essential resources, such as the CSV file containing the necessary traits and predicted prices of most HDBs (used by the frontend to find ideal HDB), and the machine learning model in predict_price.py. When a user enters input into the frontend, the input values are passed into the model, and the predicted price is returned and displayed on the interface. The frontend is designed to proactively detect and respond to invalid inputs. If a user inputs invalid data, such as a postal code that does not correspond to an HDB unit or a remaining lease value exceeding 99 years (which exceeds the maximum possible remaining lease in Singapore), a warning message is shown. This prompts the user to correct their input and ensures that only valid parameters are passed to the backend model for prediction. By providing immediate feedback, we indicate the problem to users and guide them to resolve it, helping recover from errors effectively.

Beyond integration of data into user interface, a key part of our development process involved ongoing collaboration between frontend and backend teams, especially when designing features that affect both data and user interaction. One example is the amenity score. Originally, the backend generated this score using internal trait-based logic that was not on a standardised scale. However, from a user's perspective, a 1–5 scale would be more intuitive and easier to understand. After testing, we also found that this simple scaling led to skewed results, with very few HDBs scoring a 5, which made the feature less useful. Through team discussions, we realised we needed a better way to scale the data while keeping it meaningful to users. We collectively decided to adopt a percentile-based approach: a score of 5 now represents the top 20% of HDBs in terms of amenities, resulting in a more balanced and informative distribution. Decisions like this required collaborative problem-solving to align technical data with user expectations. The frontend team ensured the scale was understandable and usable in the UI, while the backend team handled data transformation and accuracy. This collaboration helped make the final feature both functional and intuitive.

3.4 Limitations and Future Improvements

3.4.1 Streamlit Limitations

During implementation, we encountered several limitations with Streamlit. One key issue is its page navigation behavior: when users move between sections (e.g., from *Homepage* to *HDB Price Trends*), the new page retains the previous scroll position instead of resetting to the top. This could possibly disrupt user flow and can confuse first-time users, who expect consistent navigation patterns.

Another limitation involves Streamlit’s restrictions on interactive elements. For instance, components like st.warning do not support embedded hyperlinks, making messages like *Compare via Postal Code* non-clickable and reducing user guidance efficiency.

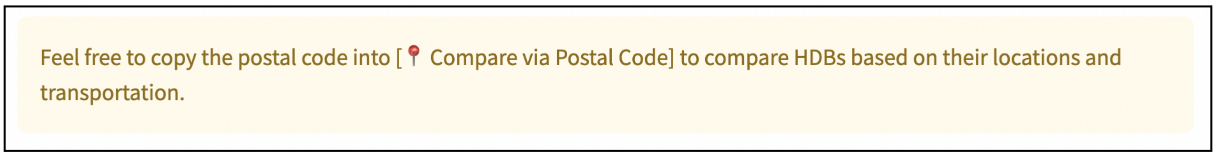
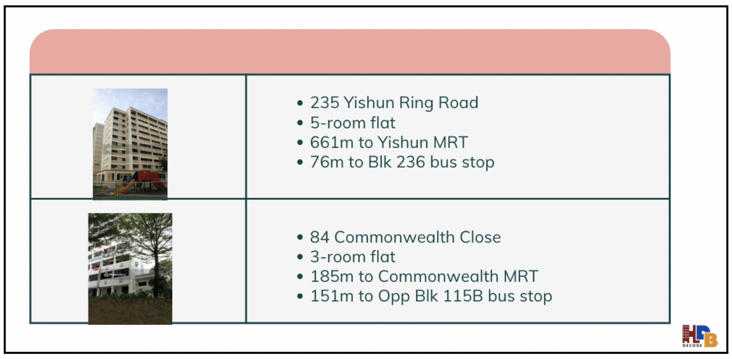

Feel free to copy the postal code into [📍 Compare via Postal Code] to compare HDBs based on their locations and transportation.

Figure: Warning box in Streamlit

We tried injecting JavaScript to manage scroll positioning, but it was unsuccessful. To overcome these limitations, we intend to explore alternative frameworks such as React, Dash, or hybrid solutions for greater frontend control.

3.4.2 Future Improvement

Visual Integration via API or Partnerships
The current map interface lacks rich visual information about HDBs. We aim to integrate real-time property images by connecting to third-party APIs or forming partnerships with platforms like PropertyGuru. This will allow us to present relevant photos within the app, giving users a more immersive and informative view of each HDB.


- 235 Yishun Ring Road
- 5-room flat
- 661m to Yishun MRT
- 76m to Blk 236 bus stop

- 84 Commonwealth Close
- 3-room flat
- 185m to Commonwealth MRT
- 151m to Opp Blk 115B bus stop

Figure: Prototype of integrating images

External Listing Links
Pop-ups on the map will include links to official resale listings, floor plans, and agent contact details, allowing users to easily transition from exploration to decision-making.

Double-Click to Add Postal Code to Map
We propose a new interaction where users can double-click on a postal code in the *Find HDBs* results table to automatically add that HDB to the comparison map. This eliminates the need for manual copying and pasting, creating a smoother and more integrated experience across the *Find HDBs* and *Compare via Postal Code* features.

Adding Filters by Postal Code and MRT
In the future, we plan to introduce additional filters for postal code and nearby MRT stations for the *Find Your Ideal Home* page to help users narrow down their search to more precise locations based on personal preferences. Given that the current interface already offers a wide range of filters, we will conduct A/B testing and User Acceptance Testing before implementation to ensure these new features improve usability without contributing to decision fatigue.

Enhancing Location Information in the *Compare via Postal Code* Feature
In the future, we plan to enhance the *Compare via Postal Code* feature by displaying the actual locations of the nearest MRT station and bus stop directly on the map, instead of showing them only in text that appears when users click the location marker of the postal code entered. We also intend to expand the information provided to include more facilities such as hawker centres, schools, and supermarkets.

Improving Postal Code Input in *Compare via Postal Code* Feature

Currently, users can only remove entered postal codes in the *Compare via Postal Code* feature by manually deleting them one by one or refreshing the page. To enhance user convenience, we plan to introduce a 'Clear All' button that allows users to remove all entered postal codes with a single click.

Part IV Contribution and Reflection

4.1 Contributions

Team Member	Sub-team	Contribution
Li Minyi	Backend	● Transformed macroeconomic and private property data ● Implemented and evaluated machine learning models
Qin Mulan	Backend	● Transformed room attributes; Developed distance to nearest bus stop/MRT station feature ● Implemented and evaluated deep learning models
Su Zhimin	Backend	● Engineered amenity score predictors and developed CPI-derived inflation features ● Implemented and evaluated linear regression models
Hang Yizhou	Frontend	● Designed Homepage layout ● Implemented the *Find Ideal HDBs* feature to filter and sort relevant HDBs on *Find Your Ideal Home* page
Hu Shiqi	Frontend	● Designed and developed the *HDB Price Trends* page using effective data visualisations ● Collaborated with the backend team to integrate the *Predict Your HDB Price* feature
Sow Yun Tsing	Frontend	● Implemented the *Compare via Postal Code* feature under *Find Your Ideal Home* page ● Implemented translation feature

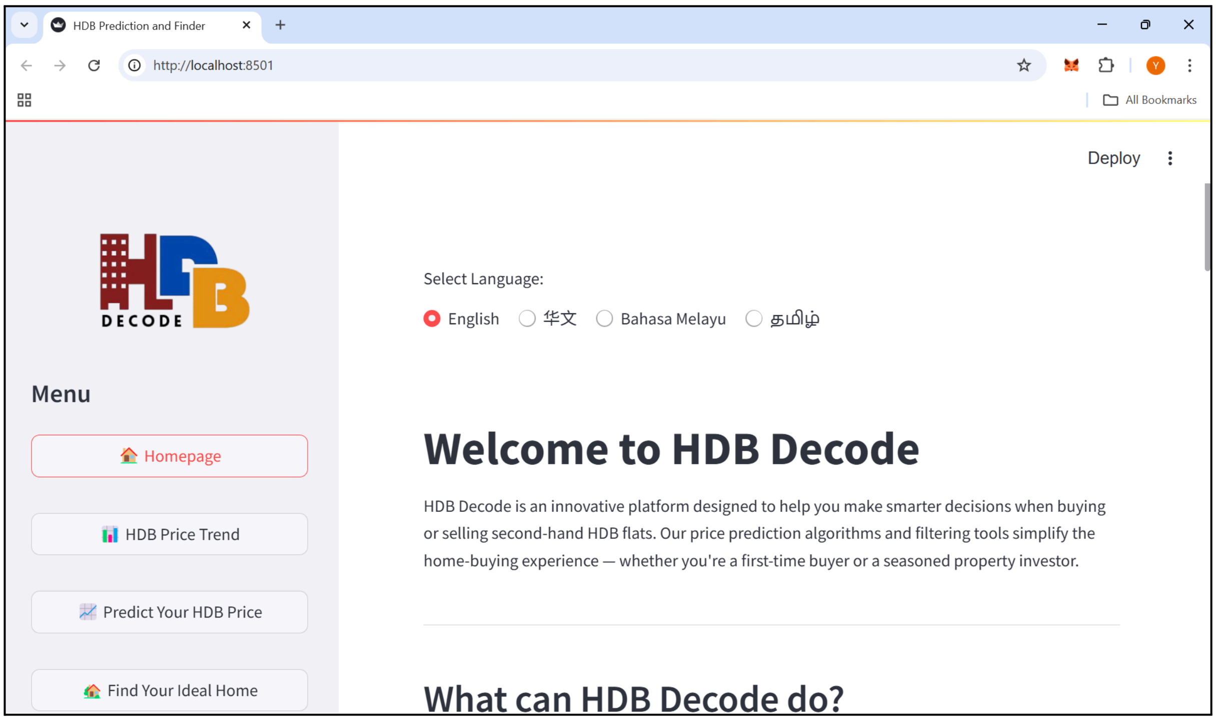
4.2 Backend Reflections

Overall, the backend team stayed largely on track with our planned timeline. However, we spent more time than expected on feature engineering. This phase requires extensive data sourcing, cleaning, transformation, and integration from multiple sources, as well as constructing our own scores. In contrast, the modeling phase progressed more smoothly once the features were finalized. The extra time invested in feature engineering ultimately enabled smoother and more effective modeling, as the prepared dataset required minimal additional adjustment during experimentation.

4.3 Frontend Reflections

We have successfully completed all tasks outlined in the original project timeline. From developing the prototype and setting up the Streamlit frontend to creating data visualisations, implementing interactive features, integrating backend functions, and conducting usability tests, we were able to achieve the planned milestones. As the backend team continued refining the data format and model output, we had to remain flexible in our implementation to continue developing the functionality of our app. To maximise efficiency and save time, we implemented a demo using mock data while awaiting the final CSV and model from the backend team. The demo version allowed us to conduct usability testing, gather valuable feedback, and iterate on our design to improve user experience. Through this process, we gained valuable experience in cross-team collaboration and learned how to effectively coordinate timelines and deliverables across different teams.

Appendix



Select Language:

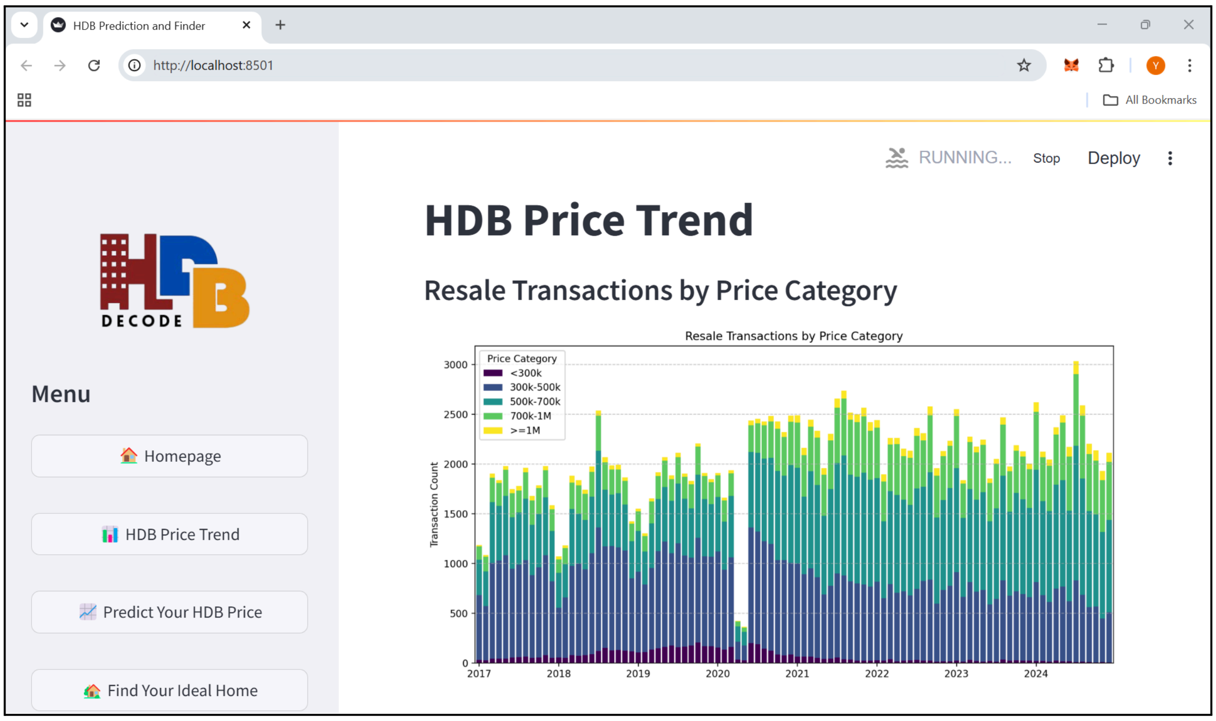
[✓ English] 华文 Bahasa Melayu தமிழ்

Welcome to HDB Decode

HDB Decode is an innovative platform designed to help you make smarter decisions when buying or selling second-hand HDB flats. Our price prediction algorithms and filtering tools simplify the home-buying experience — whether you're a first-time buyer or a seasoned property investor.

What can HDB Decode do?

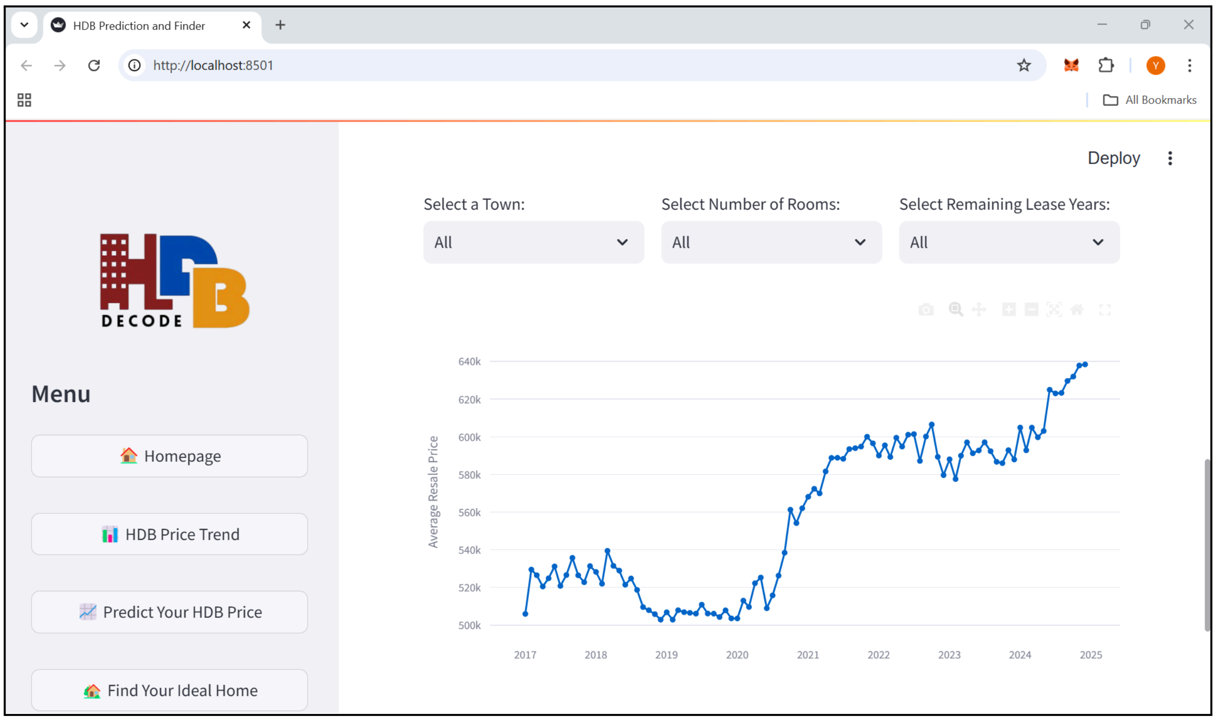
Figure 1: Homepage



HDB Price Trend

Resale Transactions by Price Category

Figure 2: Resale Transactions by Price Category Graph

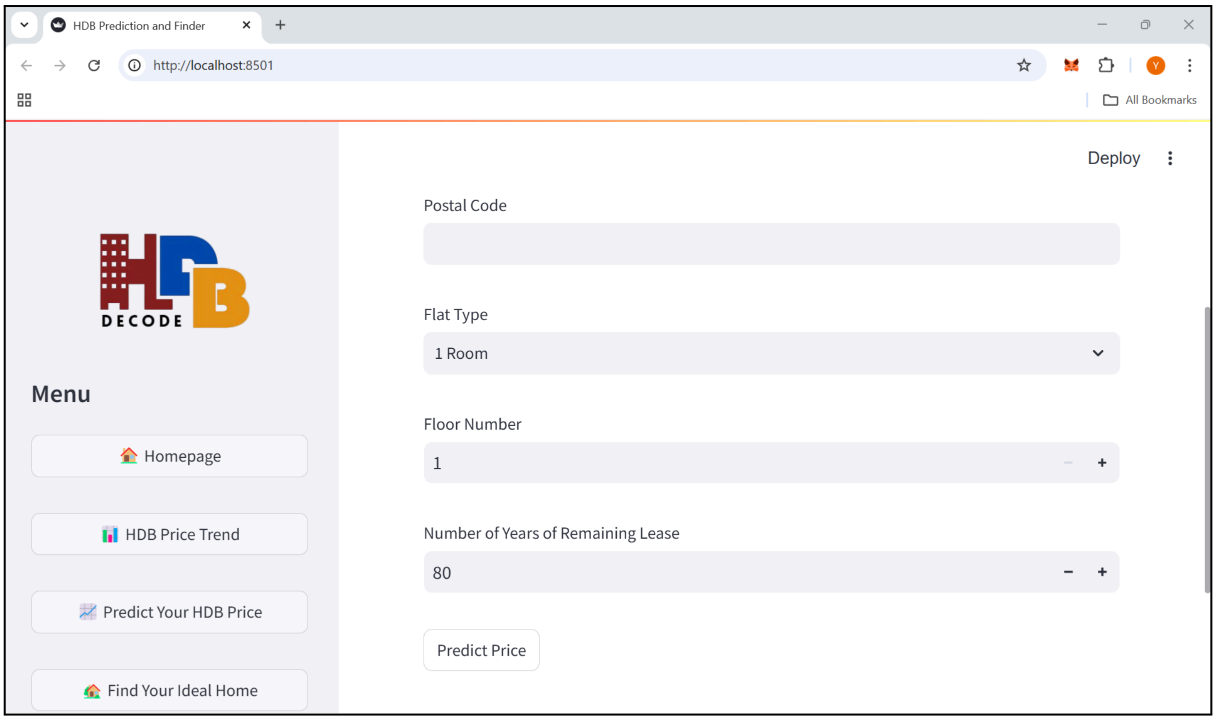


Select a Town: All

Select Number of Rooms: All

Select Remaining Lease Years: All

Figure 3: Filters for Price Trend



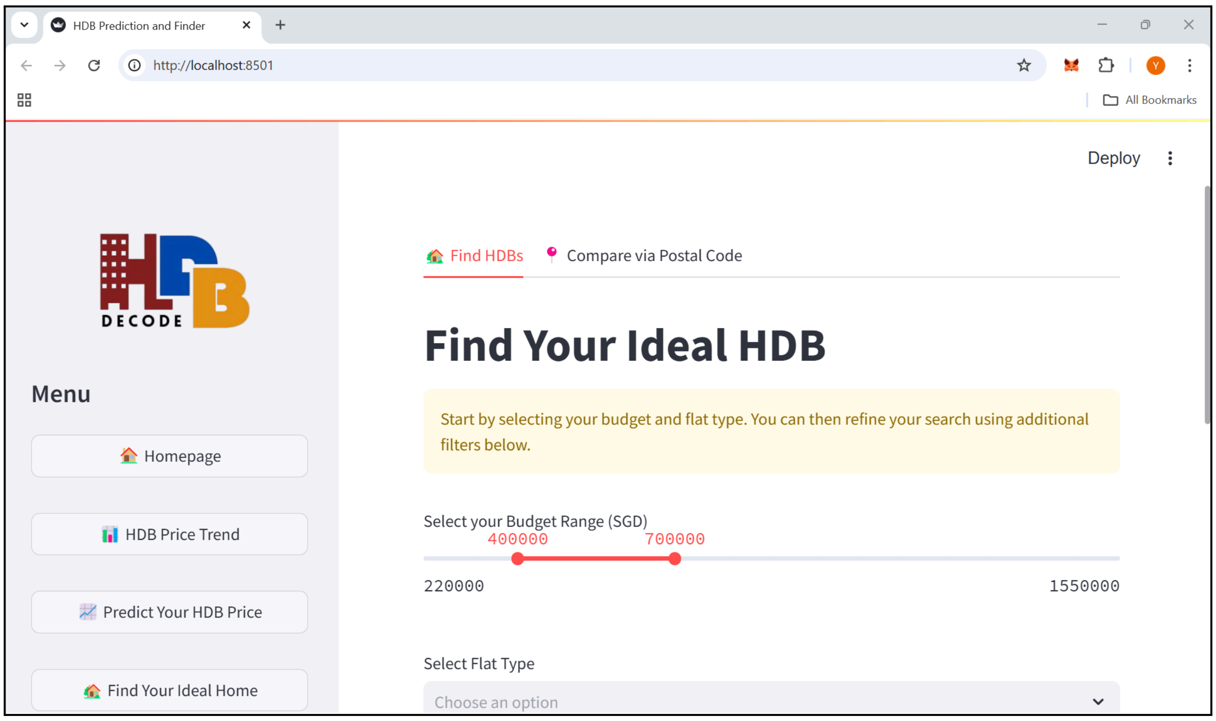
Postal Code

Flat Type: 1 Room

Floor Number: 1

Number of Years of Remaining Lease: 80

Figure 4: Predict Your HDB Price Page



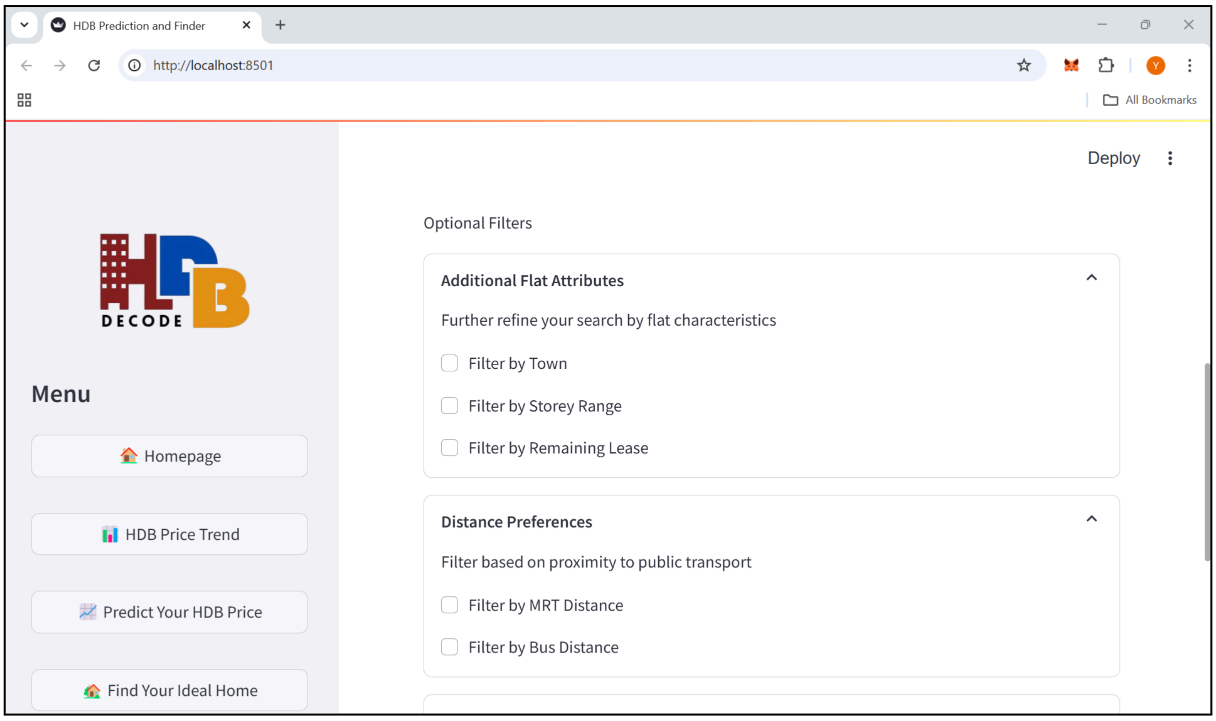
Find Your Ideal HDB

Start by selecting your budget and flat type. You can then refine your search using additional filters below.

Select your Budget Range (SGD)
400000 700000
220000 1550000

Select Flat Type
Choose an option

Figure 5: Find Your Ideal HDB Page



Optional Filters

Additional Flat Attributes

Further refine your search by flat characteristics

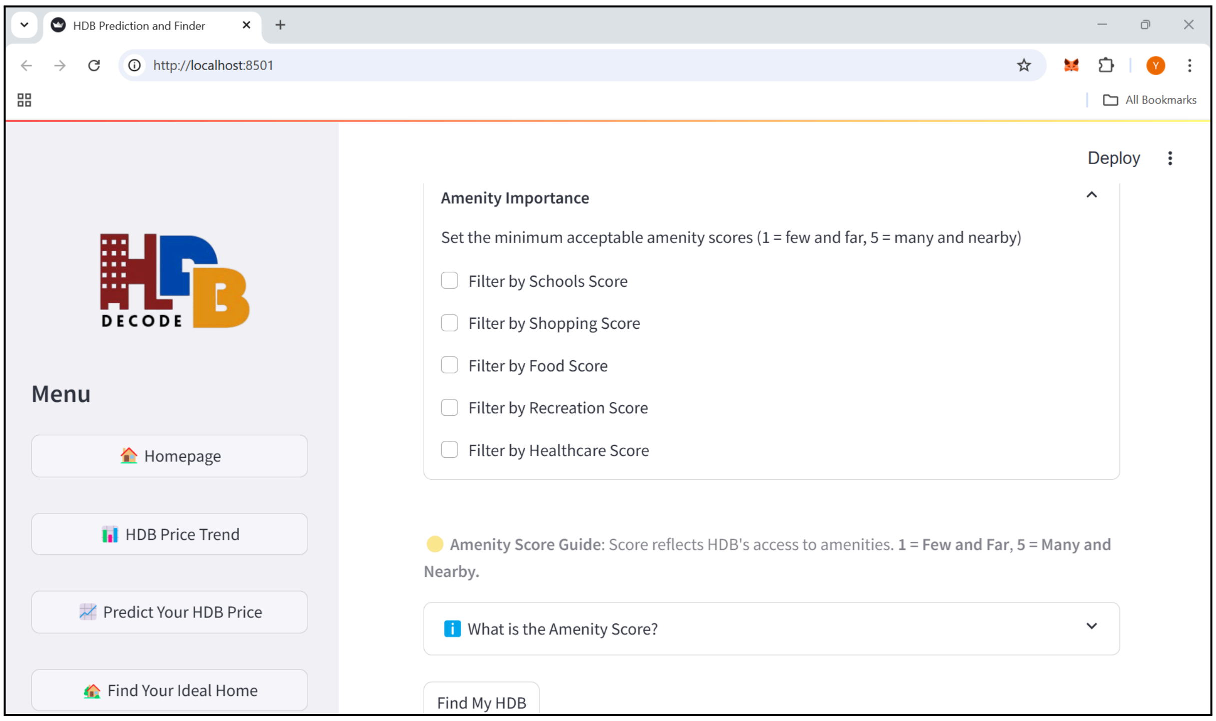
- [ ] Filter by Town
- [ ] Filter by Storey Range
- [ ] Filter by Remaining Lease

Distance Preferences

Filter based on proximity to public transport

- [ ] Filter by MRT Distance
- [ ] Filter by Bus Distance

Figure 6: Optional Filters on Find Your Ideal HDB Page


HDB Prediction and Finder

http://localhost:8501

All Bookmarks

Deploy

DECODE

Menu

Homepage

HDB Price Trend

Predict Your HDB Price

Find Your Ideal Home

Amenity Importance

Set the minimum acceptable amenity scores (1 = few and far, 5 = many and nearby)

Filter by Schools Score

Filter by Shopping Score

Filter by Food Score

Filter by Recreation Score

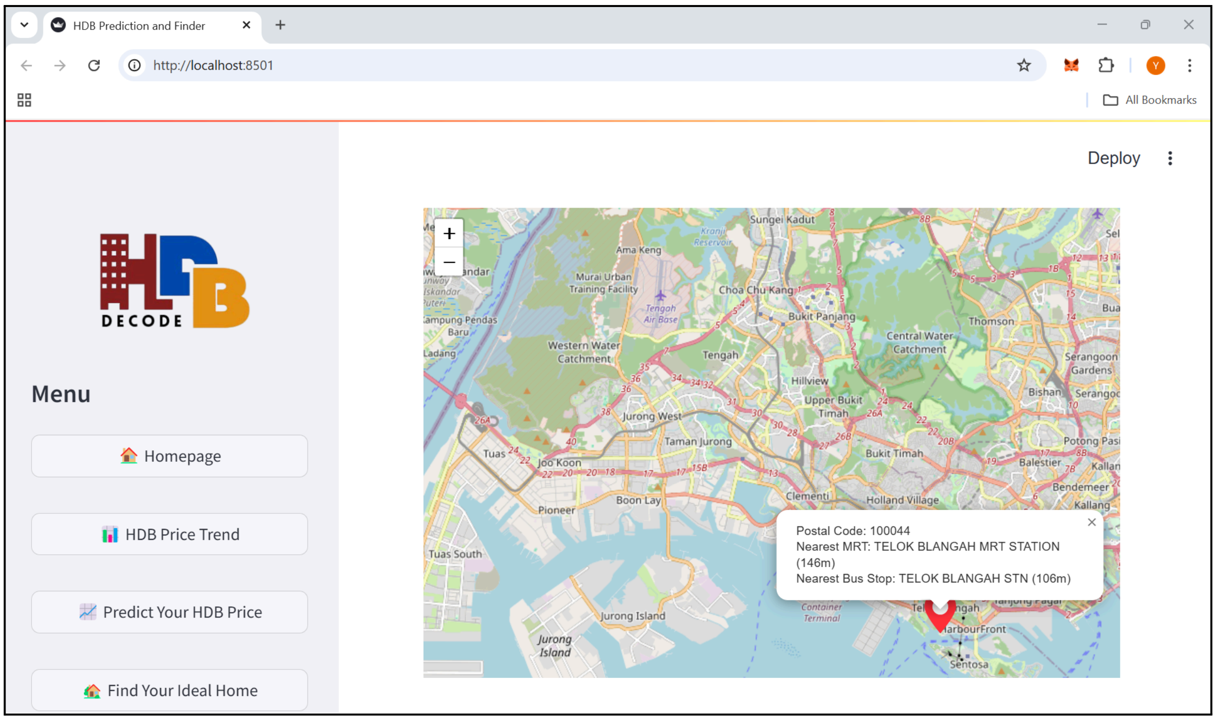
Filter by Healthcare Score

Amenity Score Guide: Score reflects HDB's access to amenities. **1 = Few and Far, 5 = Many and Nearby.**

What is the Amenity Score?

Find My HDB

Figure 7: Optional Amenity Filters for Find Your Ideal HDB Page


HDB Prediction and Finder

http://localhost:8501

All Bookmarks

Deploy

DECODE

Menu

Homepage

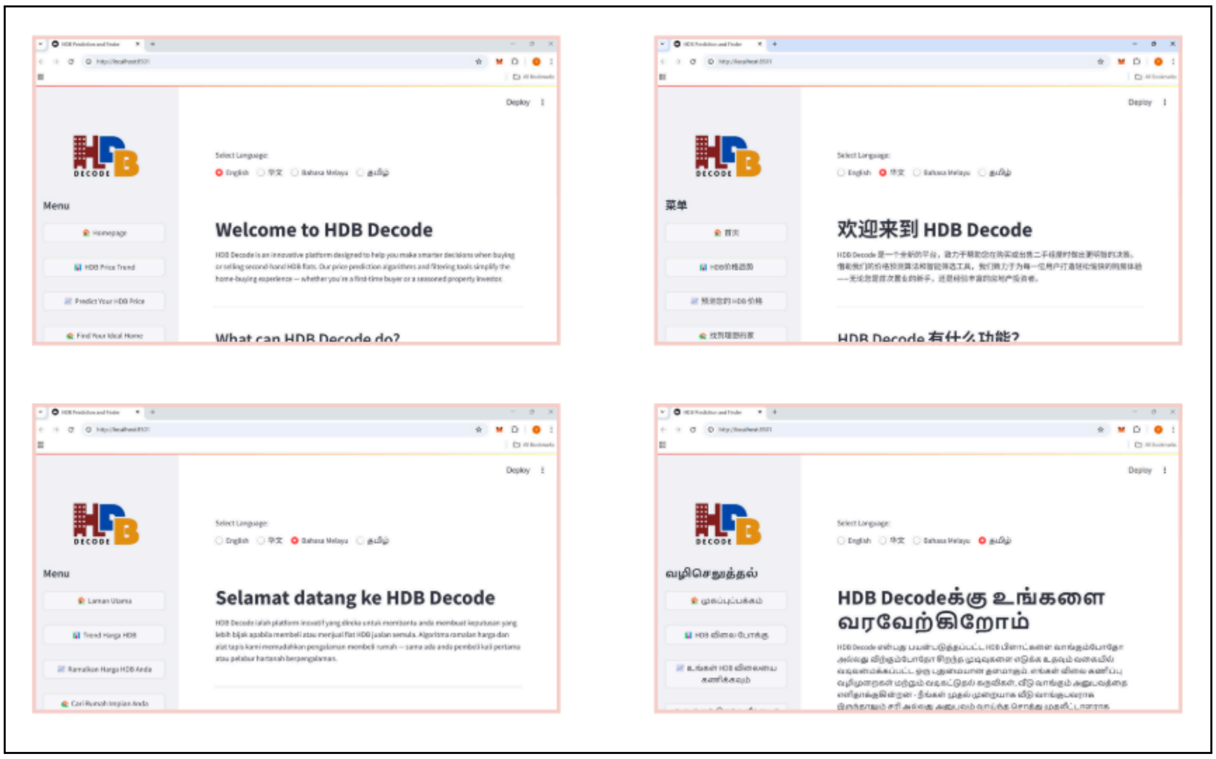
HDB Price Trend

Predict Your HDB Price

Find Your Ideal Home

Postal Code: 100044
Nearest MRT: TELOK BLANGAH MRT STATION (146m)
Nearest Bus Stop: TELOK BLANGAH STN (106m)

Figure 8: Compare via Postal Code Page



Welcome to HDB Decode

HDB Decode is an innovative platform designed to help you make smarter decisions when buying or selling second-hand HDB flats. Our price prediction algorithms and filtering tools simplify the home-buying experience — whether you're a first-time buyer or a seasoned property investor.

What can HDB Decode do?

欢迎来到 HDB Decode

[illegible]

HDB Decode 有什么功能?

Selamat datang ke HDB Decode

HDB Decode ialah platform inovatif yang direka untuk membantu anda membuat keputusan yang lebih bijak apabila membeli atau menjual flat HDB jualan semula. Algoritma ramalan harga dan alat tapis kami memudahkan pengalaman membeli rumah — sama ada anda pembeli kali pertama atau pelabur hartanah berpengalaman.

HDB Decodeக்கு உங்களை வரவேற்கிறோம்

[illegible]

Figure 9: Translation Feature

Data Source

- Data.gov.sg (Resale flat prices based on registration date from Jan 2017 onwards)
  https://data.gov.sg/datasets/d_8b84c4ee58e3cfc0ece0d773c8ca6abc/view
- Data.gov.sg (CPI)
  https://data.gov.sg/datasets/d_8f3660871b62f38609915ee7ef45ee2c/view
- Data.gov.sg (2015)
  https://data.gov.sg/datasets/d_bb771c5189ce18007621533dd36142bb/view
- Data.gov.sg (2020)
  https://data.gov.sg/datasets/d_2d6793de474551149c438ba349a108fd/view
- OneMap API (distance to MRT/ Bus Stop)
  https://www.onemap.gov.sg/apidocs/reverseGeocode
- Humanitarian Data Exchange (HDX)
  https://data.humdata.org/dataset/hotosm_sgp_health_facilities?force_layout=desktop
- Data.gov.sg (Hawker Centers)
  https://data.gov.sg/datasets/d_ccca3606c337a5c386b9c88dc0dd08b6/view
- Data.gov.sg (Schools)
  https://data.gov.sg/datasets/d_688b934f82c1059ed0a6993d2a829089/view
- Kaggle (Shopping Malls)
  https://www.kaggle.com/datasets/karthikgangula/shopping-mall-coordinates?resource=download
- Data.gov.sg (Supermarkets)
  https://data.gov.sg/datasets/d_1bf762ee1d6d7fb61192cb442fb2f5b4/view
- Onemap API (Recreation)
  https://www.onemap.gov.sg/apidocs/themes
- Data.gov.sg (Religion 2015)
  https://data.gov.sg/datasets/d_609f4027e7fdc33251db12f4d3f38d4f/view
- Data.gov.sg (Religion 2020)
  https://data.gov.sg/datasets/d_a58564fbed922609a0f79af96069dd9b/view