FirstHome Advisor

An application designed to help young adults (aged 25–35) assess their financial readiness to purchase their first home using a mortgage loan. The app serves an advisory and educational role, relying on user data and a classification model that evaluates creditworthiness and risk.

# Target Audience

* Young individuals planning to buy their first home
* Users with income from employment contracts, self-employment, freelancing
* Individuals without financial expertise seeking to better understand their options
* Users interested in independently assessing their creditworthiness

# Core Features

## MVP (Minimum Viable Product)

* A form to input personal and financial data (age, income, employment type, monthly liabilities, own contribution, etc.)
* Classify users as ready, almost ready, or not ready for a mortgage
* Calculation of indicators such as DTI (debt-to-income ratio)
* Short, personalized recommendations based on the classification
* Clear presentation of results and risk level

## Extended Version (Optional)

* "What-if" simulation — analyze changes in income, contribution, or liabilities
* History and saving of user scenarios
* Selection of various loan types and impact analysis (interest rate, repayment period)
* Comparison with statistical data (e.g., average DTI, typical installments in a given region)
* PDF report generation with analysis

# Technologies

* Programming language: Python
* Libraries: Pandas, Scikit-learn, Matplotlib, Streamlit
* ML Model: Classifier (e.g., Random Forest, Logistic Regression) to assess risk
* Optional: SHAP or similar tools for prediction interpretability

# Input Data Scope

* User’s age
* Type of employment and income stability
* Monthly income
* Monthly financial liabilities (e.g., loans, credit cards, leasing)
* Amount of own contribution
* Planned property value
* Number of dependents

# Output Data Scope

* Credit readiness assessment (e.g., ready / needs improvement / not ready)
* Risk score (e.g., probability of being classified as high-risk)
* Key indicators: DTI, contribution ratio, projected installment
* Personalized suggestions for improving credit readiness

# User Interface

* Web-based application accessible via browser
* Intuitive form for data input
* Results visualization and guidance section
* Option to save and compare different scenarios

# Development Opportunities

* Integration with banking databases or loan comparison APIs
* Extension to include credit scoring based on actual BIK data (simulated)
* Mobile or PWA version
* Incorporation of economic/inflation indicators into the analysis
* Option to create a user account and track long-term financial status

# Classes

##### UserData

Description: Represents the user's financial and personal data.

Attributes:

* age: int
* employment\_type: str
* monthly\_income: float
* monthly\_expenses: float
* existing\_loans: float
* own\_contribution: float
* property\_value: float
* dependents: int

Methods:

to\_dataframe() -> pd.DataFrame: Prepares the data for the classifier.

##### RiskClassifier

Description: Wrapper for the ML model used to assess user risk.

Attributes:

* model – e.g., a scikit-learn object
* preprocessor – pipeline / scaler / encoder

Methods:

predict(user\_data: UserData) -> str: Returns classification (e.g., "eligible")

predict\_proba(user\_data: UserData) -> float: Returns risk probability

##### CreditCalculator

Description: Performs auxiliary calculations based on user data.

Methods:

calculate\_dti(user\_data: UserData) -> float

calculate\_max\_credit(user\_data: UserData) -> float

recommendations(user\_data: UserData) -> list[str]

##### AssessmentResult

Description: Structure for the assessment result.

Attributes:

* risk\_category: str
* risk\_score: float
* max\_loan: float
* dti: float
* recommendations: list[str]

##### AppController

Description: Coordinates the application logic; connects components and returns the final result to the UI.

Methods:

run\_assessment(user\_data: UserData) -> AssessmentResult

##### MainUI

Description: Interface layer.

Methods:

render\_form() -> UserData

display\_result(result: AssessmentResult)

# Architecture Pattern: MVC with ML logic in the service layer

Model – UserData, AssessmentResult, ML data (RiskClassifier)

Service – CreditCalculator, AppController

View/Interface – MainUI

# Data Flow

User fills out the form → MainUI creates UserData

AppController.run\_assessment():

Uses RiskClassifier for prediction

Uses CreditCalculator for calculations

Creates AssessmentResult

MainUI presents the result to the user

# Diagram UML

Obraz zawierający tekst, diagram, Czcionka, paragon

Opis wygenerowany automatycznie