Solution Architecture

Prosperity Prognosticator: ML For Startup Success Prediction

1. High-Level Architecture

The solution follows a three-layer architecture: Data Layer, ML Processing Layer, and Presentation Layer. All layers interact through Python scripts and the Flask framework.

2. Architecture Diagram (Text Representation)

Layer	Components
User Layer	**Web Browser → Prediction Form → Result Page**
Application Layer	**Flask App (app.py) → Routes: /, /predict → Model Loader → Predictor**
ML Layer	**random_forest_model.pkl → Scikit-learn → Feature Processing → Prediction Output**
Data Layer	**startup_data.csv → Pandas → Cleaned Dataset → Encoded Features**

3. Component Details

3.1 Data Layer

- Source: Kaggle Startup Success Prediction dataset (startup_data.csv)
- Processing: Pandas for loading, cleaning, null handling
- Encoding: LabelEncoder for categorical variables
- Splitting: 80% training, 20% testing (train_test_split)

3.2 ML Processing Layer

- Algorithms: Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, SVM, KNN
- Evaluation: Accuracy score, Classification report, Confusion matrix
- Tuning: GridSearchCV for hyperparameter optimization
- Feature Selection: Top 10 features based on importance scores
- Saving: Pickle serialization to random_forest_model.pkl

3.3 Application Layer

- Framework: Flask (Python micro web framework)
- Routes: / (home), /predict (GET shows form, POST runs prediction)
- Model Loading: pickle.load() on application startup
- Data Passing: Form data → NumPy array → Model → Prediction → Template

3.4 Presentation Layer

- index.html: Dynamic form with input fields for all top features
- result.html: Displays prediction result with return navigation

4. File Structure

File/Folder	Type	Description
startup_data.csv	Data	Raw training/testing dataset from Kaggle
startup-prediction-eda-model.ipynb	Notebook	Full ML pipeline: EDA, training, tuning
random_forest_model.pkl	Model	Saved trained Random Forest model
features.pkl	Model	Saved list of top selected feature names
app.py	Backend	Flask application with routing and prediction logic
templates/index.html	Frontend	Startup metrics input form
templates/result.html	Frontend	Prediction result display page