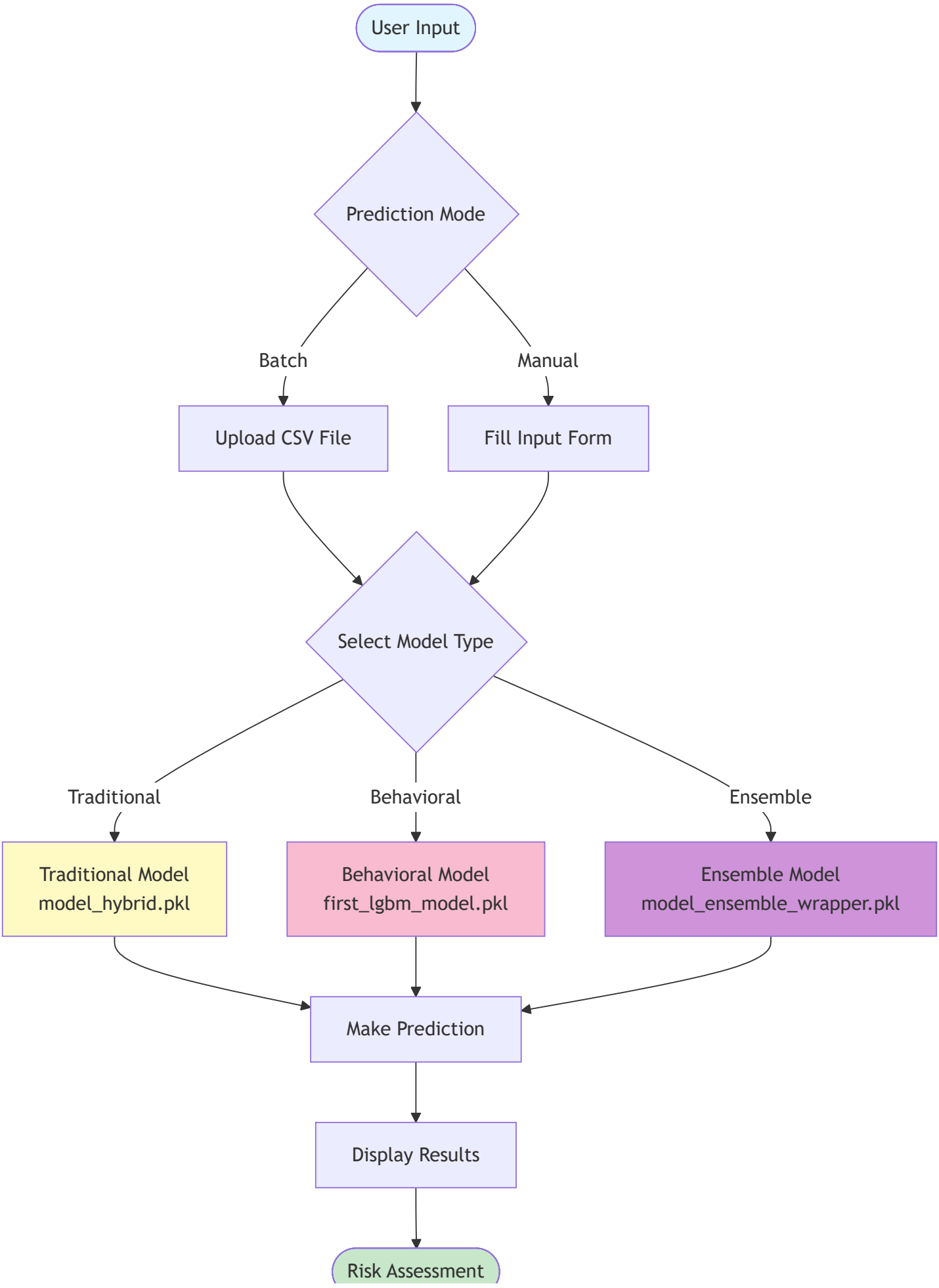
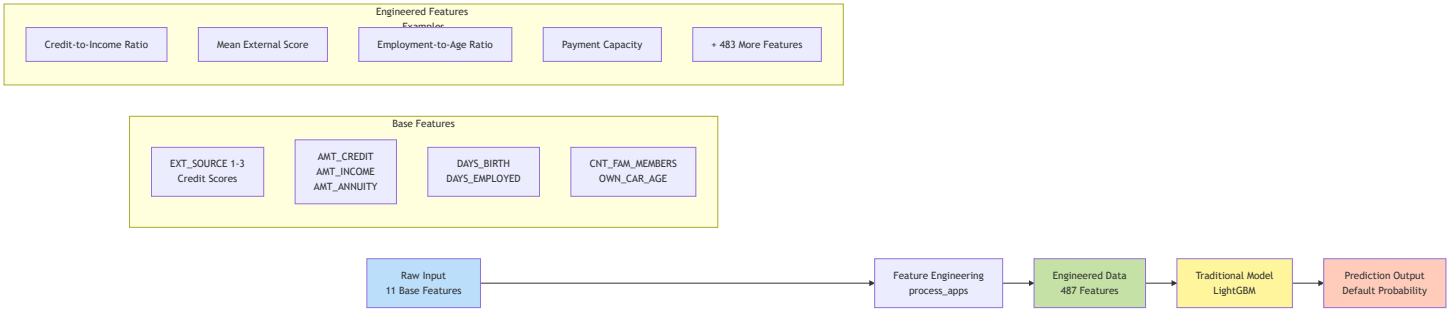
Loan Default Hybrid System - Model Architecture Flowchart

System Overview


User Input
Prediction Mode
Batch
Manual
Upload CSV File
Fill Input Form
Select Model Type
Traditional
Behavioral
Ensemble
Traditional Model
model_hybrid.pkl
Behavioral Model
first_lgbm_model.pkl
Ensemble Model
model_ensemble_wrapper.pkl
Make Prediction
Display Results
Risk Assessment

Detailed Model Pipeline

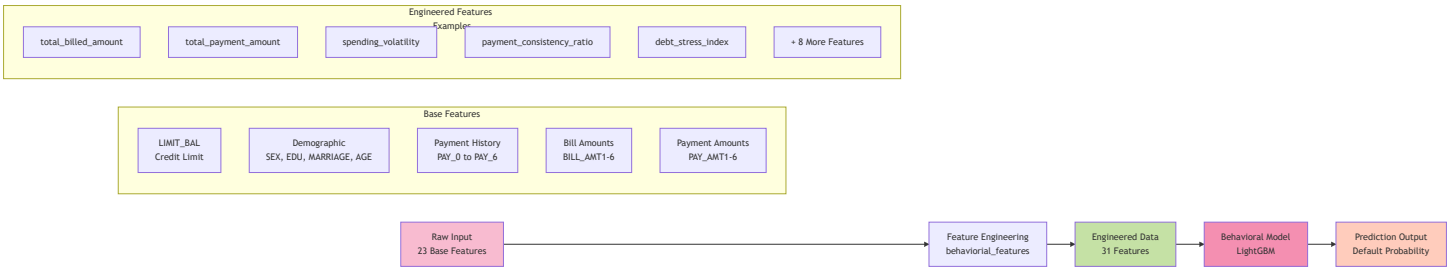
1. Traditional Model Pipeline (Home Credit Features)



Traditional Model Flow:

1. **Input**: 11 base features from Home Credit dataset
2. **Processing**: `process_apps()` function creates 487 engineered features
3. **Model**: LightGBM trained on 487 features
4. **Output**: Default probability (0-1)

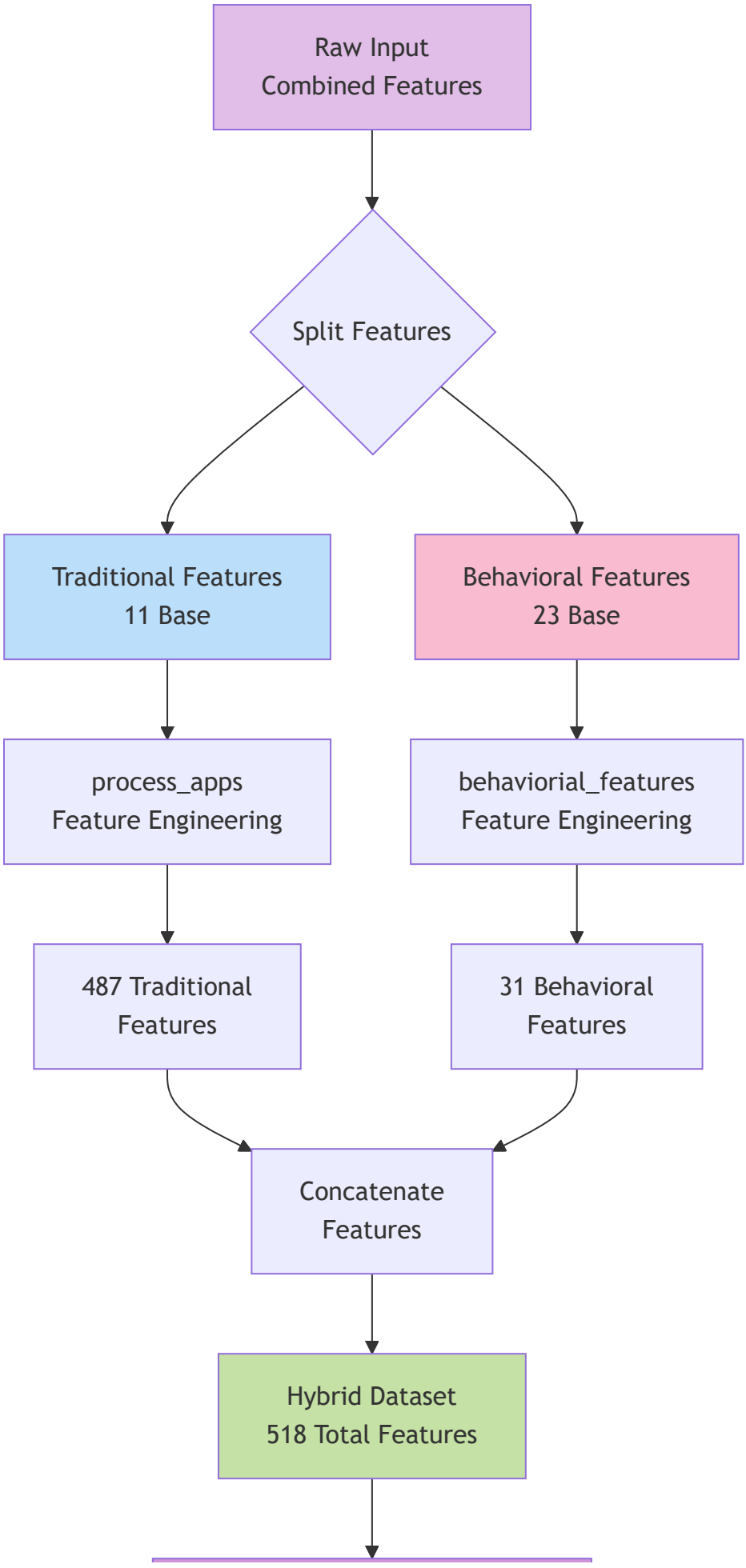
2. Behavioral Model Pipeline (UCI Credit Card Features)

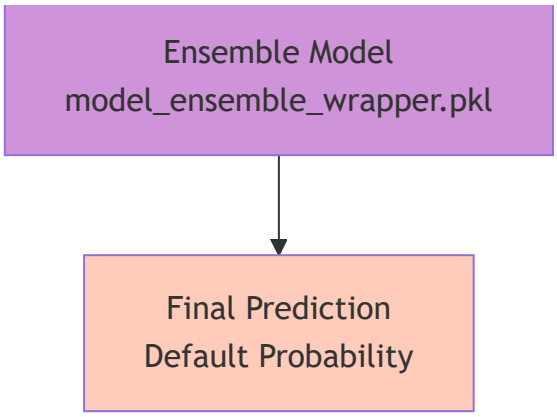


Behavioral Model Flow:

1. **Input**: 23 base features from UCI Credit Card dataset
2. **Processing**: `behaviorial_features()` function creates 31 total features (23 base + 8 engineered)
3. **Model**: LightGBM trained on 31 features
4. **Output**: Default probability (0-1)

3. Ensemble Model Pipeline (Hybrid Features)


Raw Input
Combined Features
Split Features
Traditional Features
11 Base
Behavioral Features
23 Base
process_apps
Feature Engineering
behaviorial_features
Feature Engineering
487 Traditional
Features
31 Behavioral
Features
Concatenate
Features
Hybrid Dataset
518 Total Features

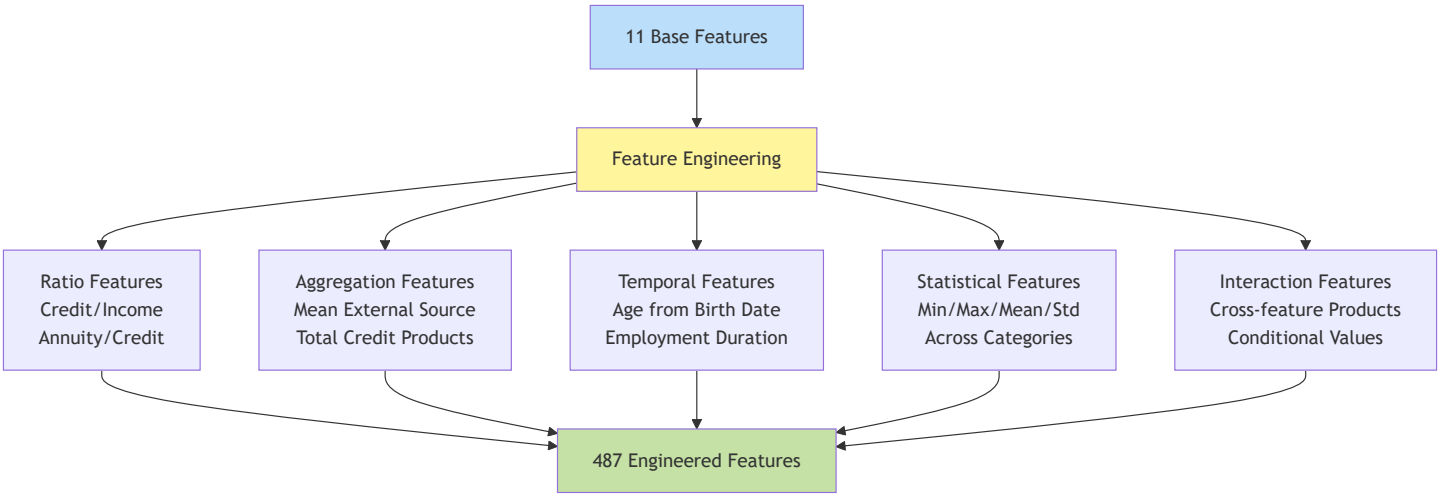


Ensemble Model Flow:

1. **Input**: Combined 34 base features (11 traditional + 23 behavioral)
2. **Split**: Separate features into traditional and behavioral groups
3. **Processing**:
   - Apply `process_apps()` to traditional features → 487 features
   - Apply `behaviorial_features()` to behavioral features → 31 features
4. **Combine**: Concatenate both feature sets → 518 total features
5. **Model**: Ensemble LightGBM trained on hybrid features
6. **Output**: Default probability (0-1)

Feature Engineering Details

Traditional Feature Engineering (`process_apps`)



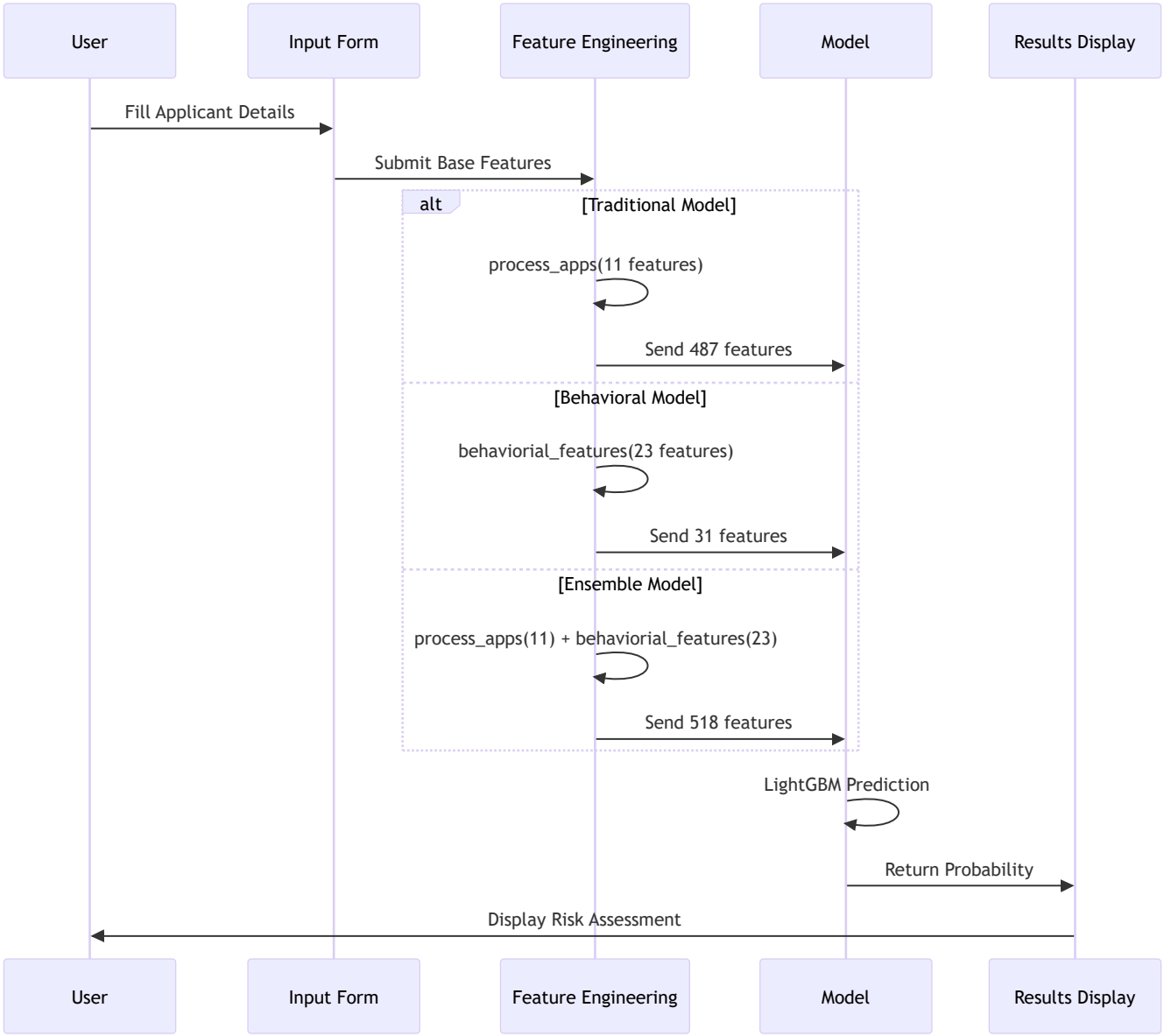
Behavioral Feature Engineering (`behaviorial_features`)

Syntax error in text

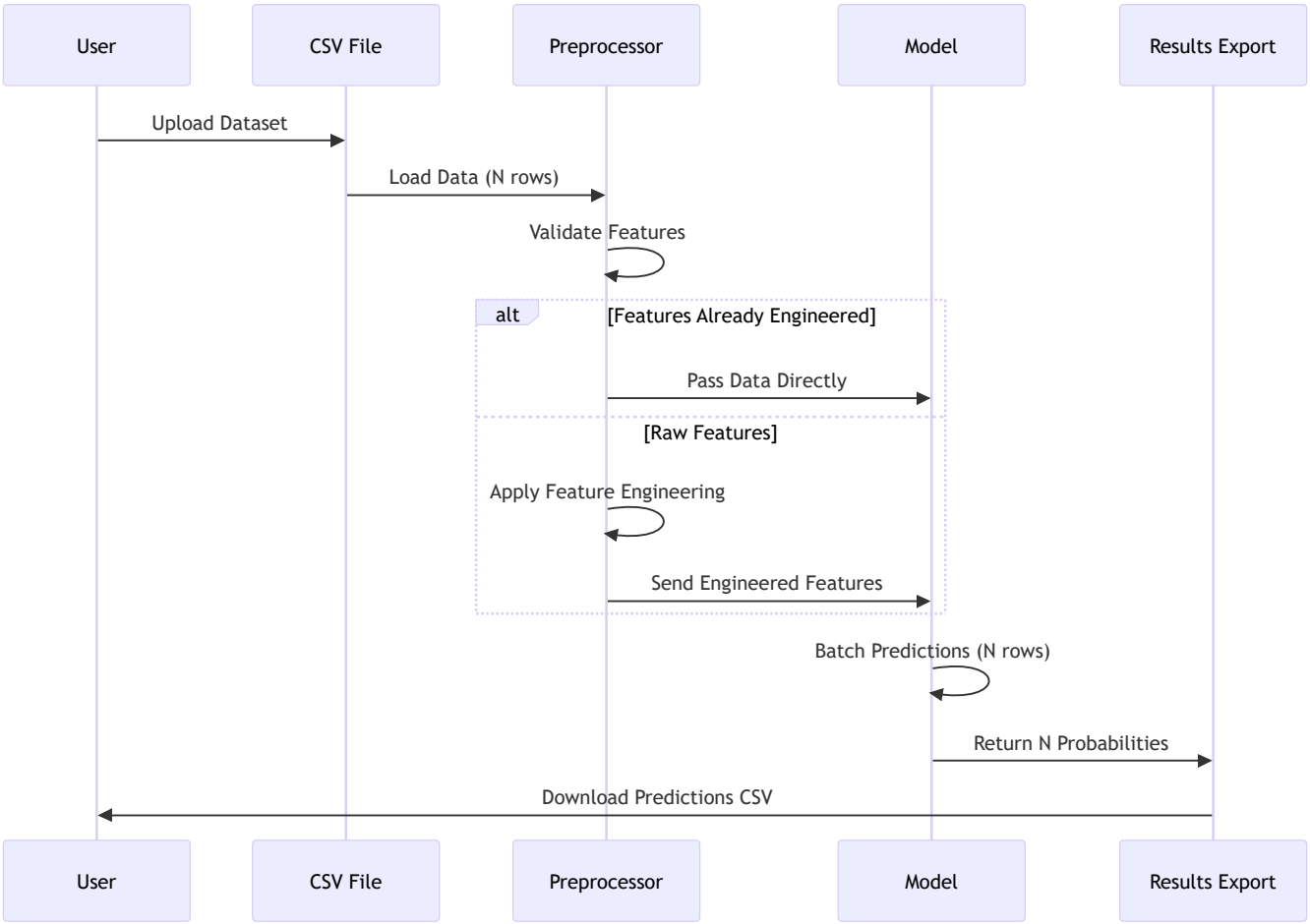
mermaid version 11.12.1

Data Flow Architecture

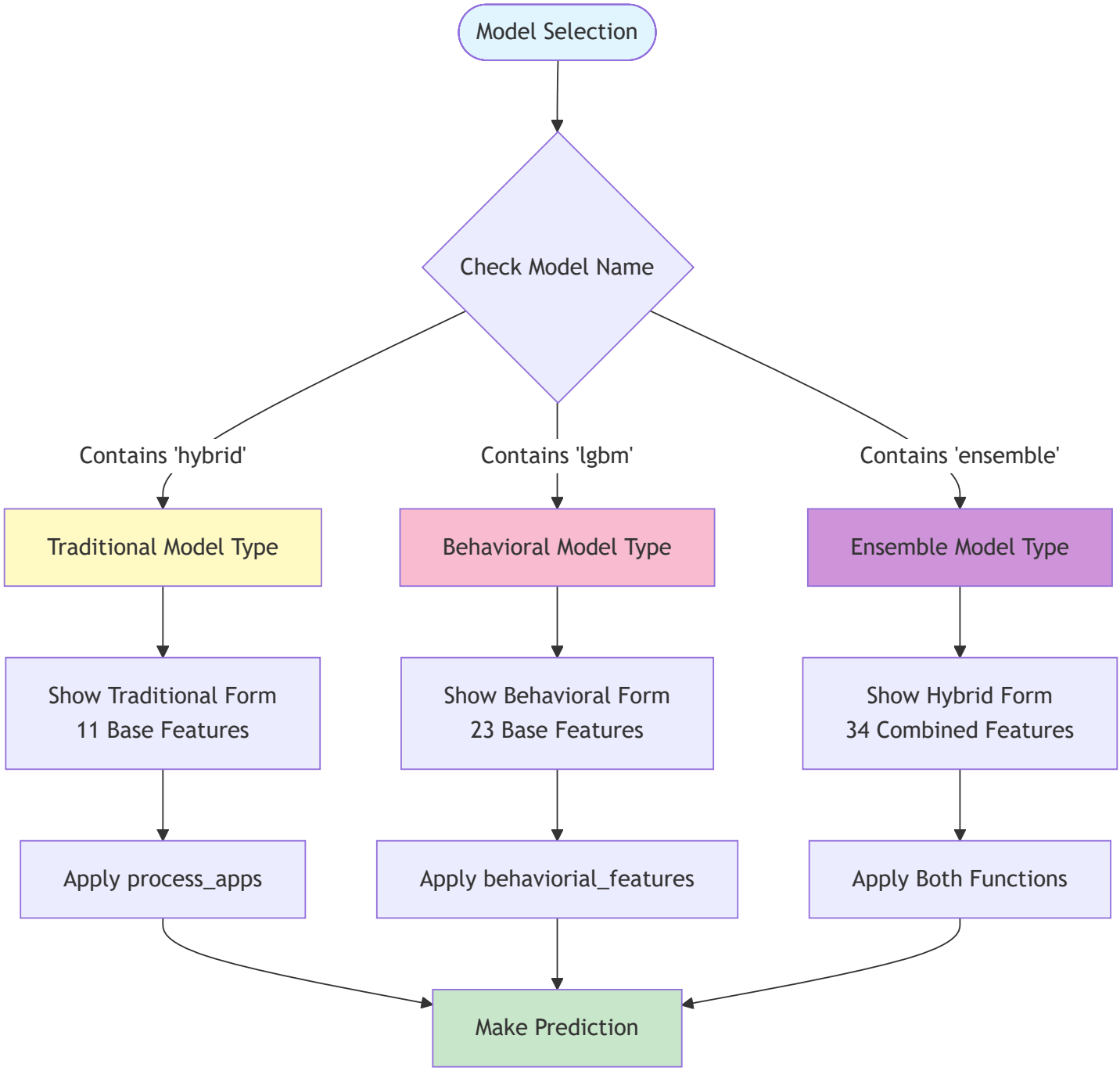
Manual Input (Single Applicant)



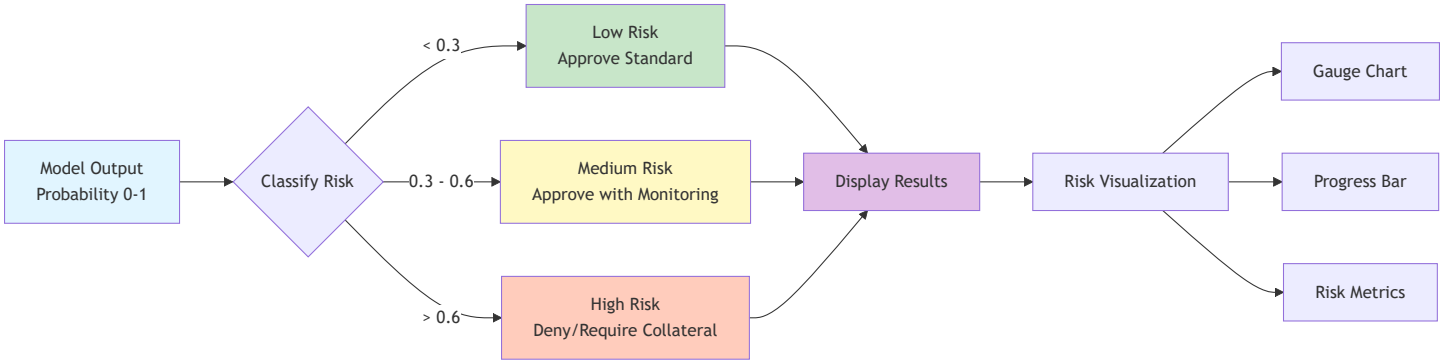
Batch Input (CSV Upload)



Model Selection Logic



Risk Classification Pipeline



Feature Count Summary

Model Type	Base Features	Engineered Features	Total Features
Traditional	11	476	**487**
Behavioral	23	8	**31**
Ensemble	34 (11+23)	484 (476+8)	**518**

Key Engineering Functions

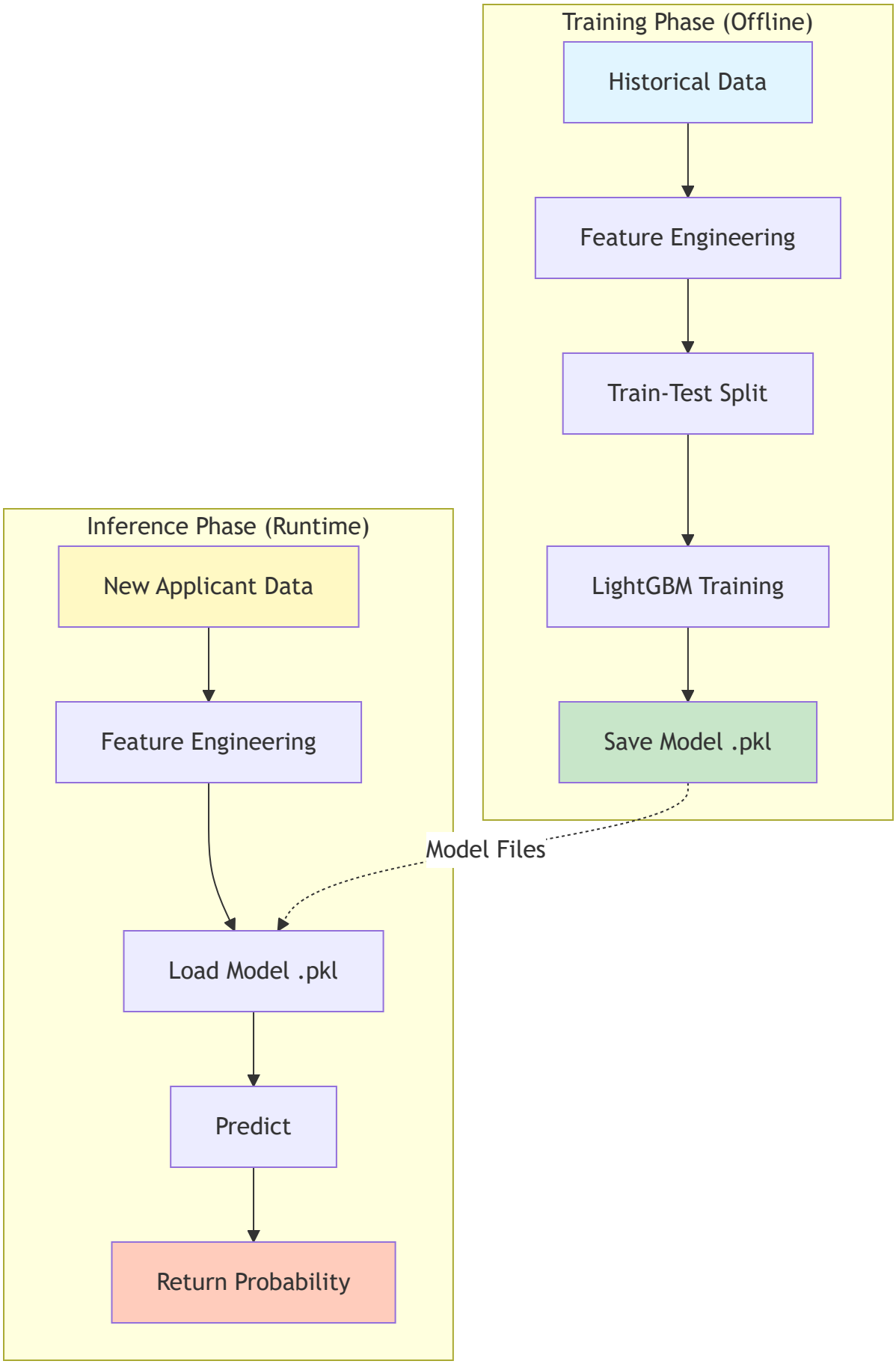
Traditional: `process_apps(df)`

```
Input:  11 base features
        ├── EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3
        ├── AMT_CREDIT, AMT_INCOME_TOTAL, AMT_ANNUITY, AMT_GOODS_PRICE
        ├── DAYS_BIRTH, DAYS_EMPLOYED
        └── CNT_FAM_MEMBERS, OWN_CAR_AGE

Output: 487 engineered features
        ├── Original 11 features
        ├── 476 calculated features:
            ├── Ratios (credit/income, annuity/income, etc.)
            ├── Aggregations (mean, sum, max, min)
            ├── Statistical (std, variance, percentiles)
            └── Interactions (cross-products, conditionals)
```

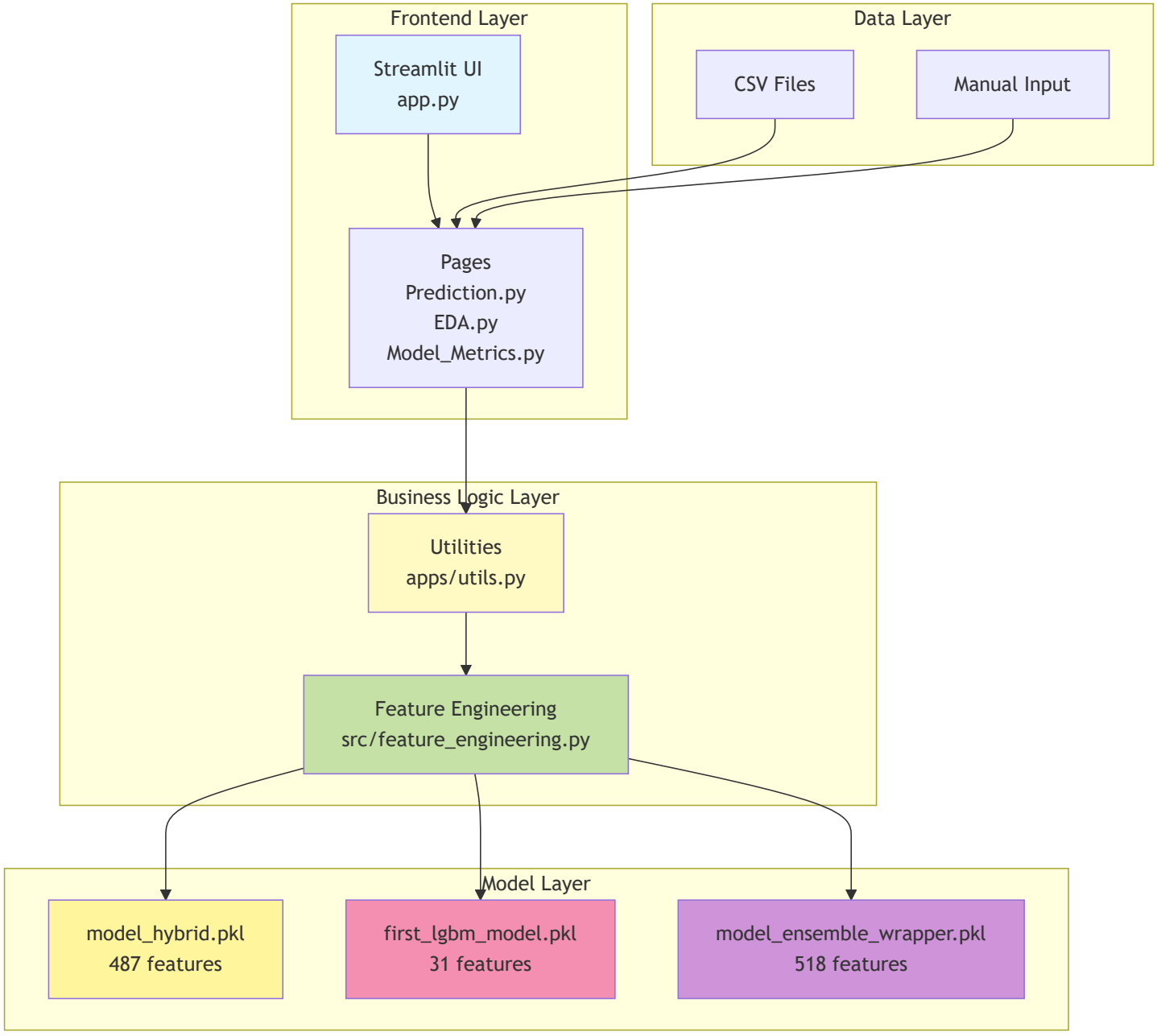
Behavioral: `behaviorial_features(df)`

```
Input:  23 base features
        ├── LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE
        ├── PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6
        ├── BILL_AMT1-6 (6 features)
        └── PAY_AMT1-6 (6 features)

Output: 31 features (23 base + 8 engineered)
        ├── Original 23 features
        ├── 8 calculated features:
            ├── total_billed_amount
            ├── total_payment_amount
            ├── avg_transaction_amount
            ├── spending_volatility
            ├── payment_consistency_ratio
            ├── debt_stress_index
            ├── credit_utilization_trend
            └── missed_payment_count (+ others)
```

Model Training Architecture



System Architecture Overview

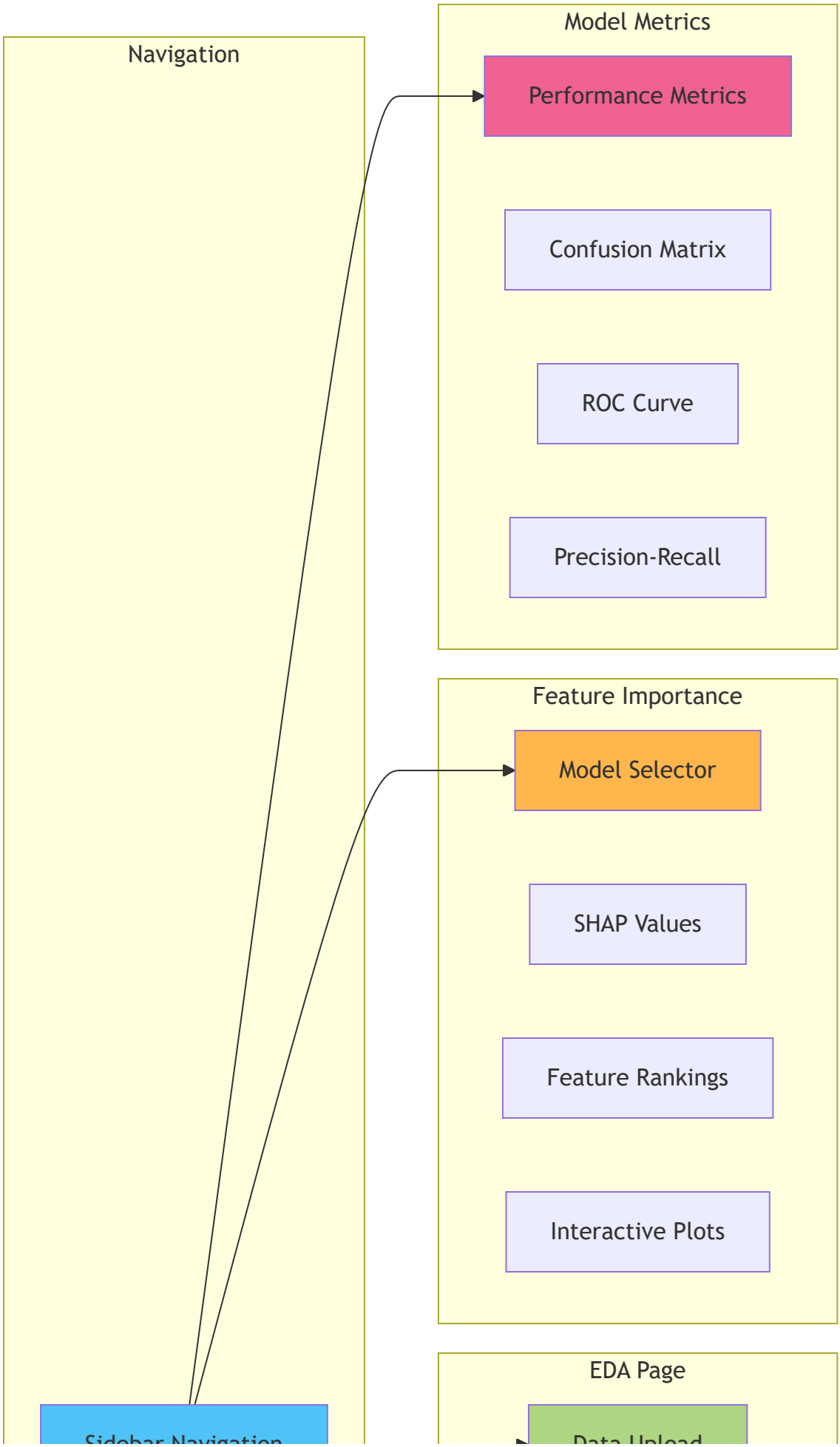


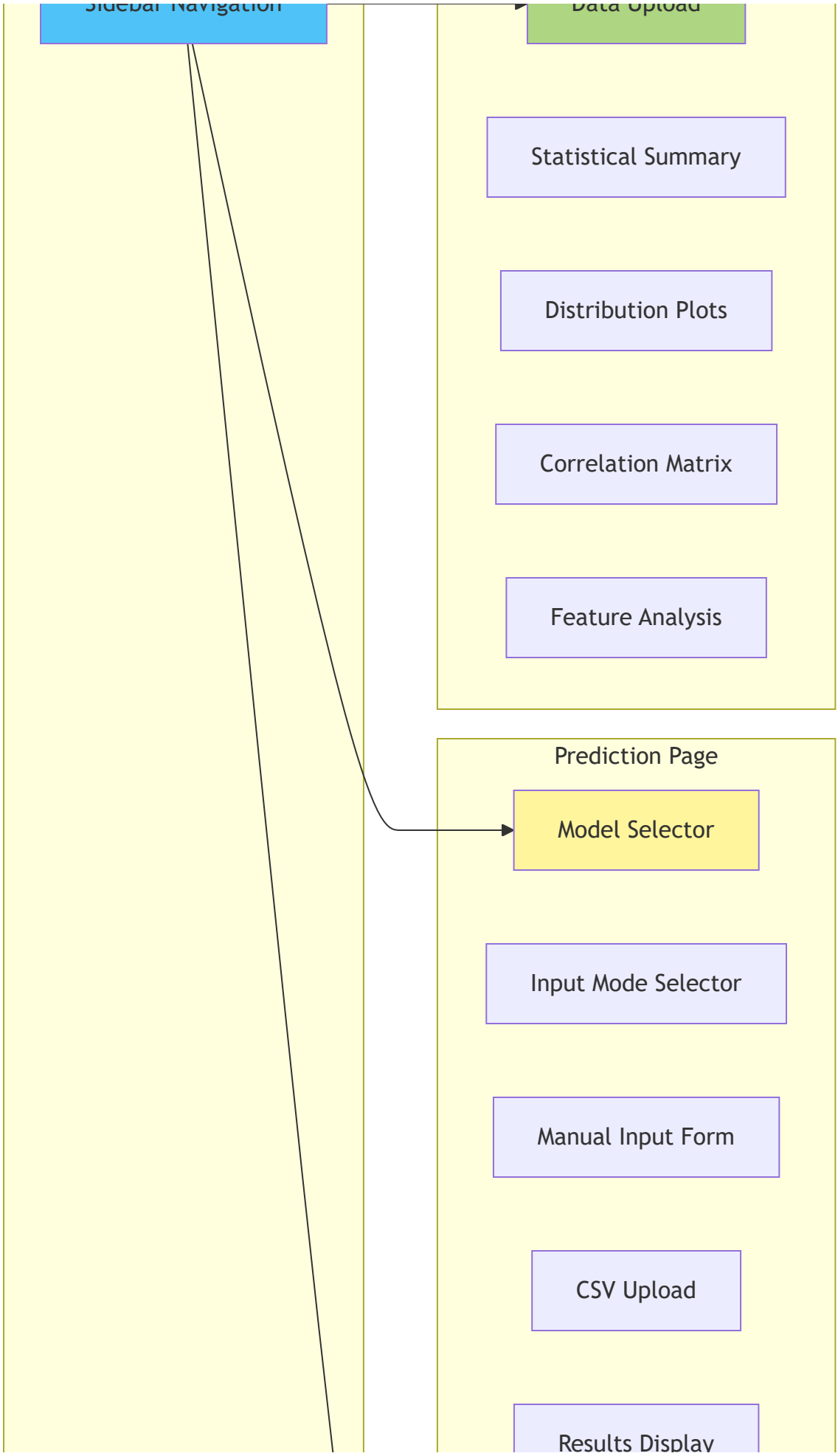
Complete System Architecture

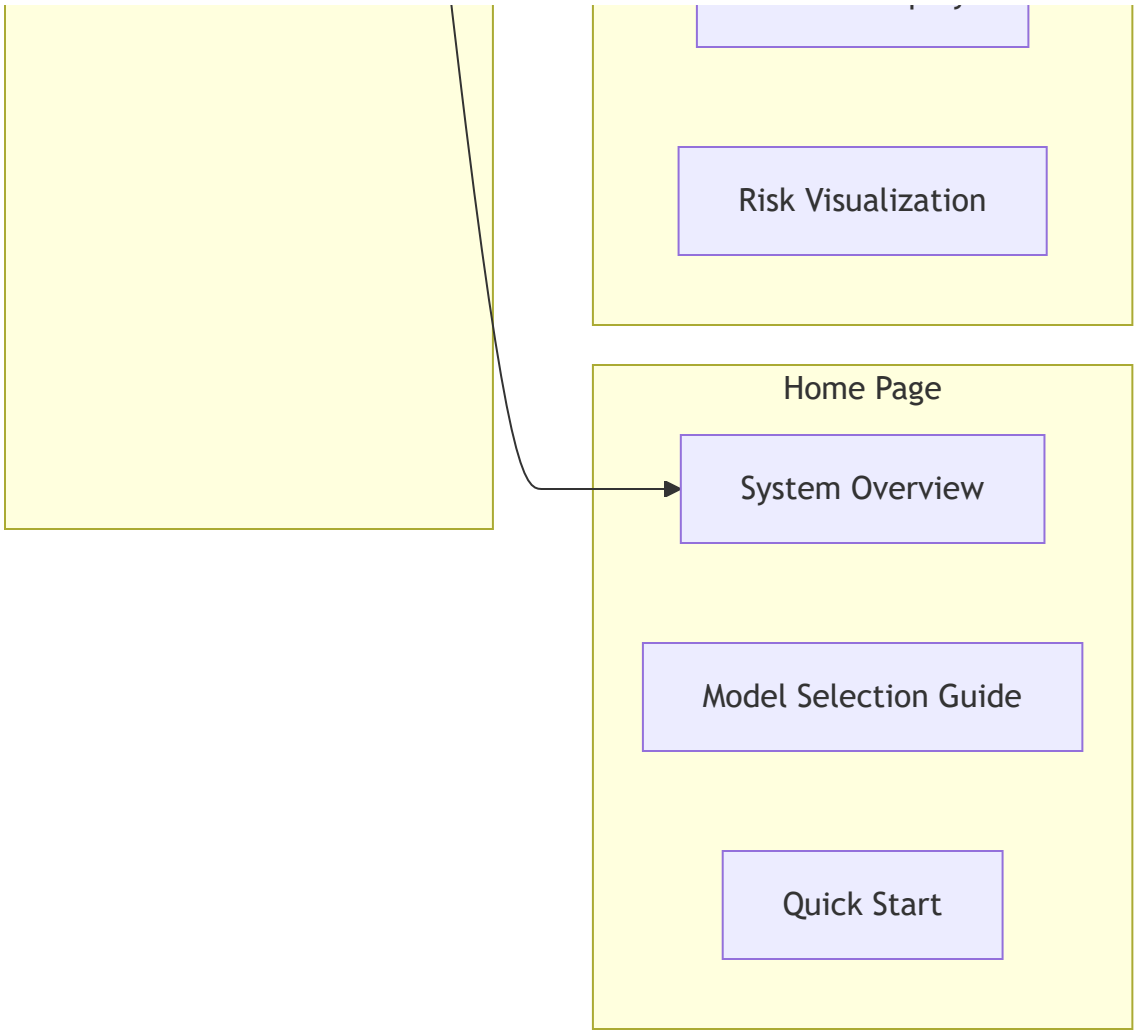
Full Application Structure

Detailed Component Architecture

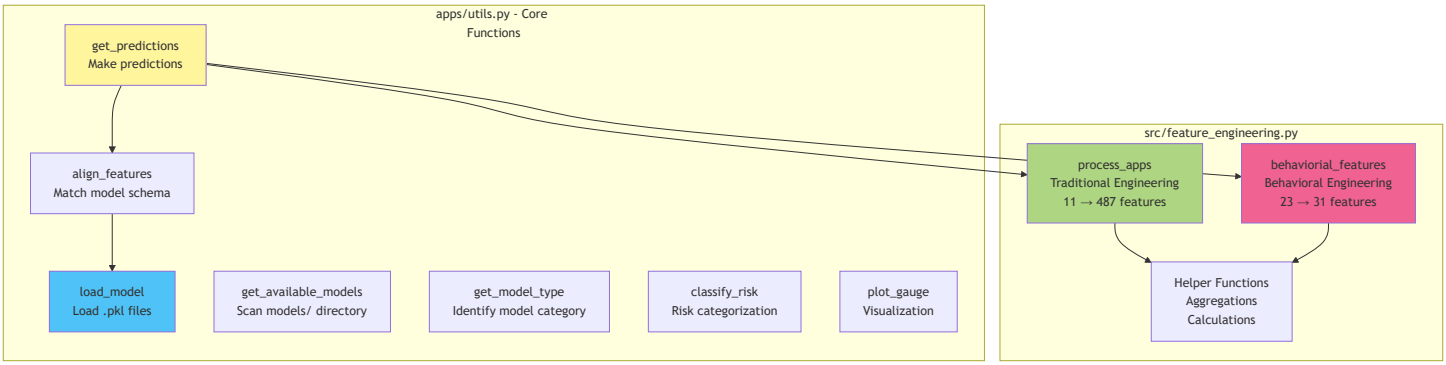
1. Frontend Components (Streamlit Pages)


Navigation
Model Metrics
Performance Metrics
Confusion Matrix
ROC Curve
Precision-Recall
Feature Importance
Model Selector
SHAP Values
Feature Rankings
Interactive Plots
EDA Page
Data Upload


Sidebar Navigation
Data Upload
Statistical Summary
Distribution Plots
Correlation Matrix
Feature Analysis
Prediction Page
Model Selector
Input Mode Selector
Manual Input Form
CSV Upload
Results Display



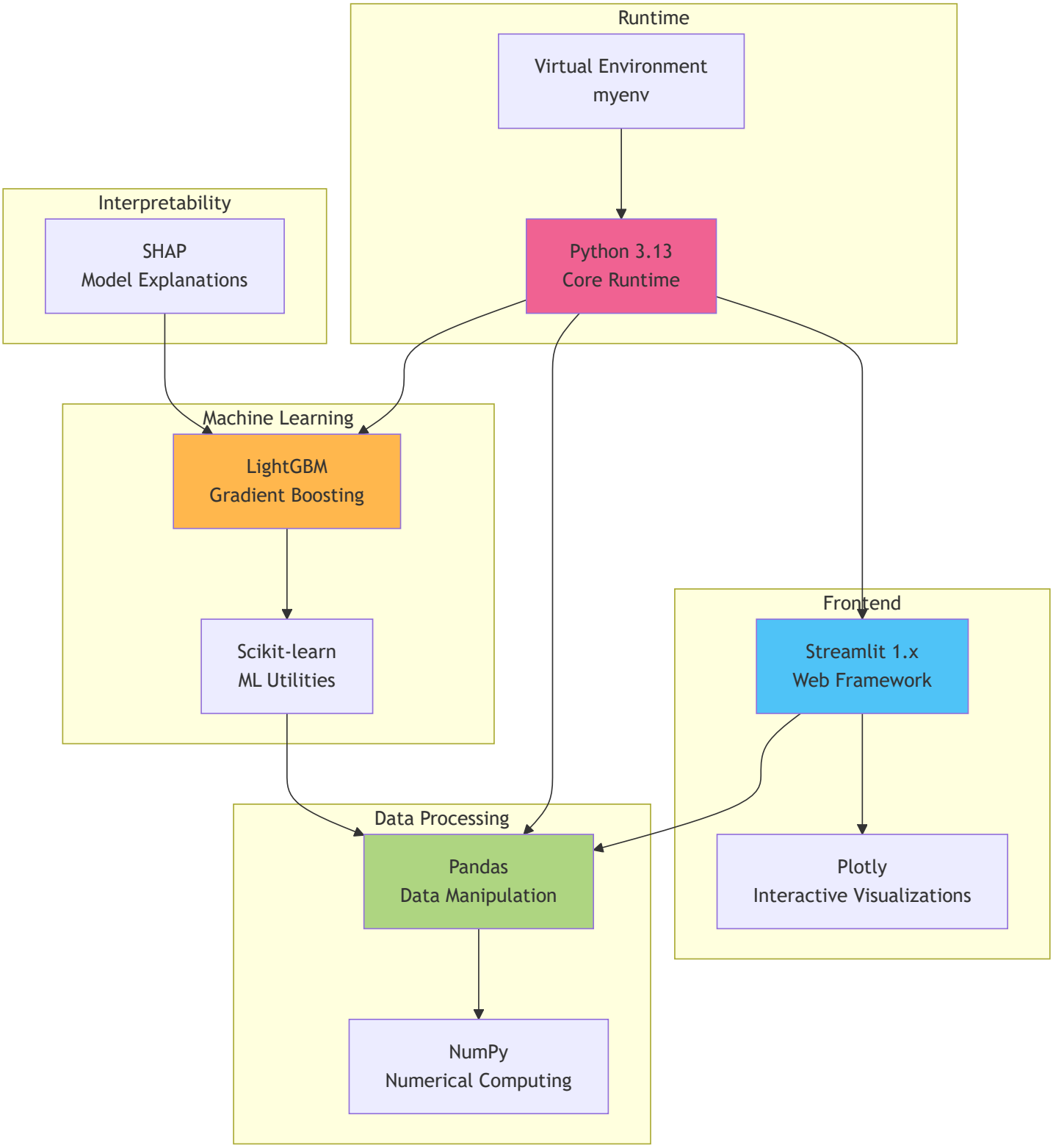
2. Business Logic Components



3. File Structure Tree

```
Loan Default Hybrid System/
│
├── app.py                          # Main Streamlit application
├── requirement.txt                 # Python dependencies
├── MODEL_ARCHITECTURE_FLOWCHART.md # This documentation
├── DATA_FLOW_EXPLANATION.md        # Data flow docs
│
├── myenv/                          # Virtual environment
│   ├── Scripts/                       # Python executables
│   └── Lib/                           # Installed packages
│
├── pages/                          # Streamlit pages
│   ├── Prediction.py                  # Main prediction interface
│   ├── EDA.py                         # Exploratory data analysis
│   ├── Feature_Importance.py          # Feature importance plots
│   └── Model_Metrics.py               # Model performance metrics
│
├── apps/                           # Business logic
│   └── utils.py                       # Core utility functions
│
├── src/                            # Source code
│   └── feature_engineering.py         # Feature engineering functions
│
├── models/                         # Trained models
│   ├── model_hybrid.pkl               # Traditional model (487 features)
│   ├── first_lgbm_model.pkl           # Behavioral model (31 features)
│   └── model_ensemble_wrapper.pkl     # Ensemble model (518 features)
│
└── data/ (optional)                # Training/test data
    ├── smoke_engineered.csv           # Traditional features
    ├── uci_interface_test.csv         # Behavioral features
    └── smoke_hybrid_features.csv      # Hybrid features
```

4. Technology Stack

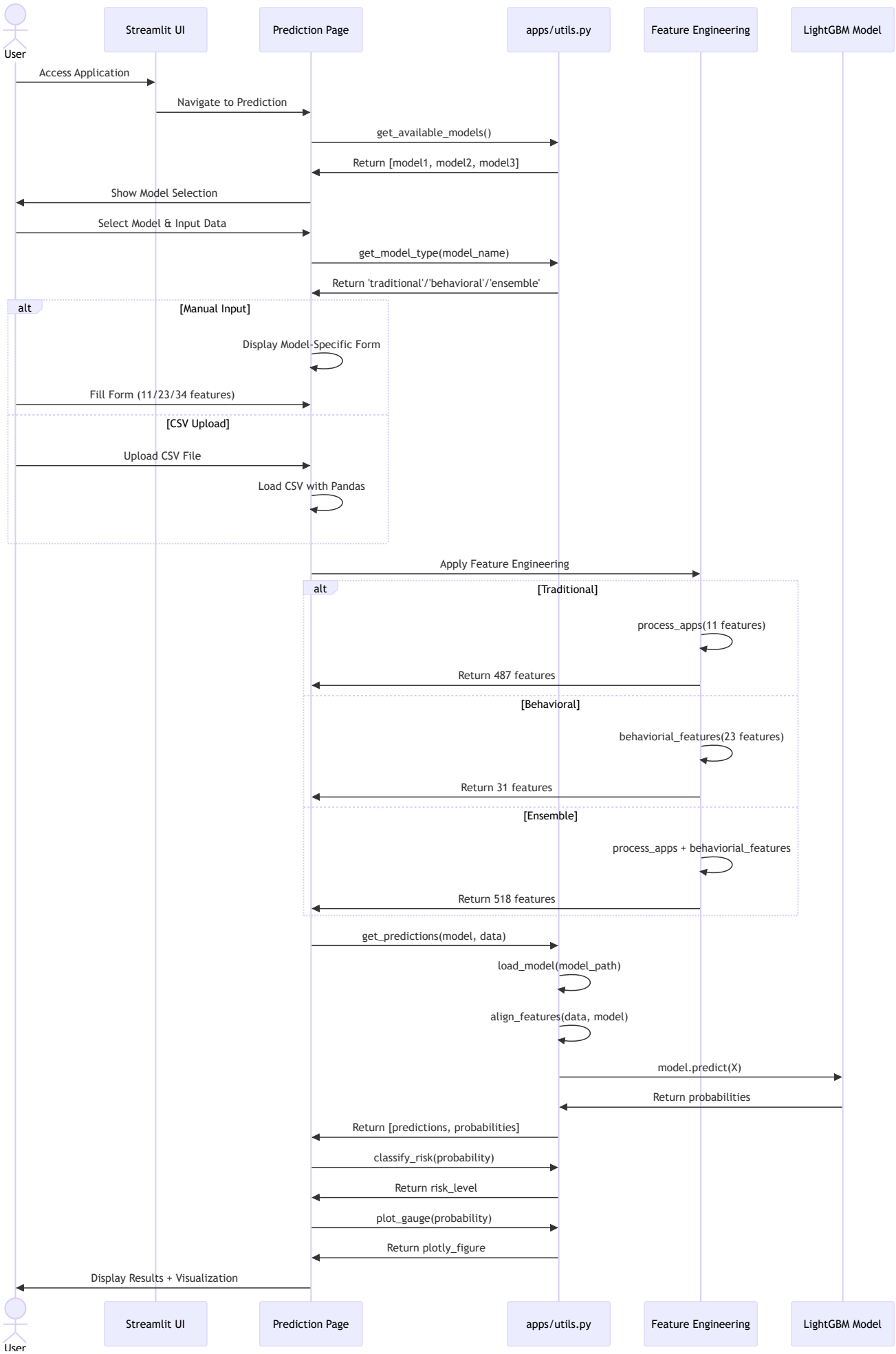


Key Dependencies:

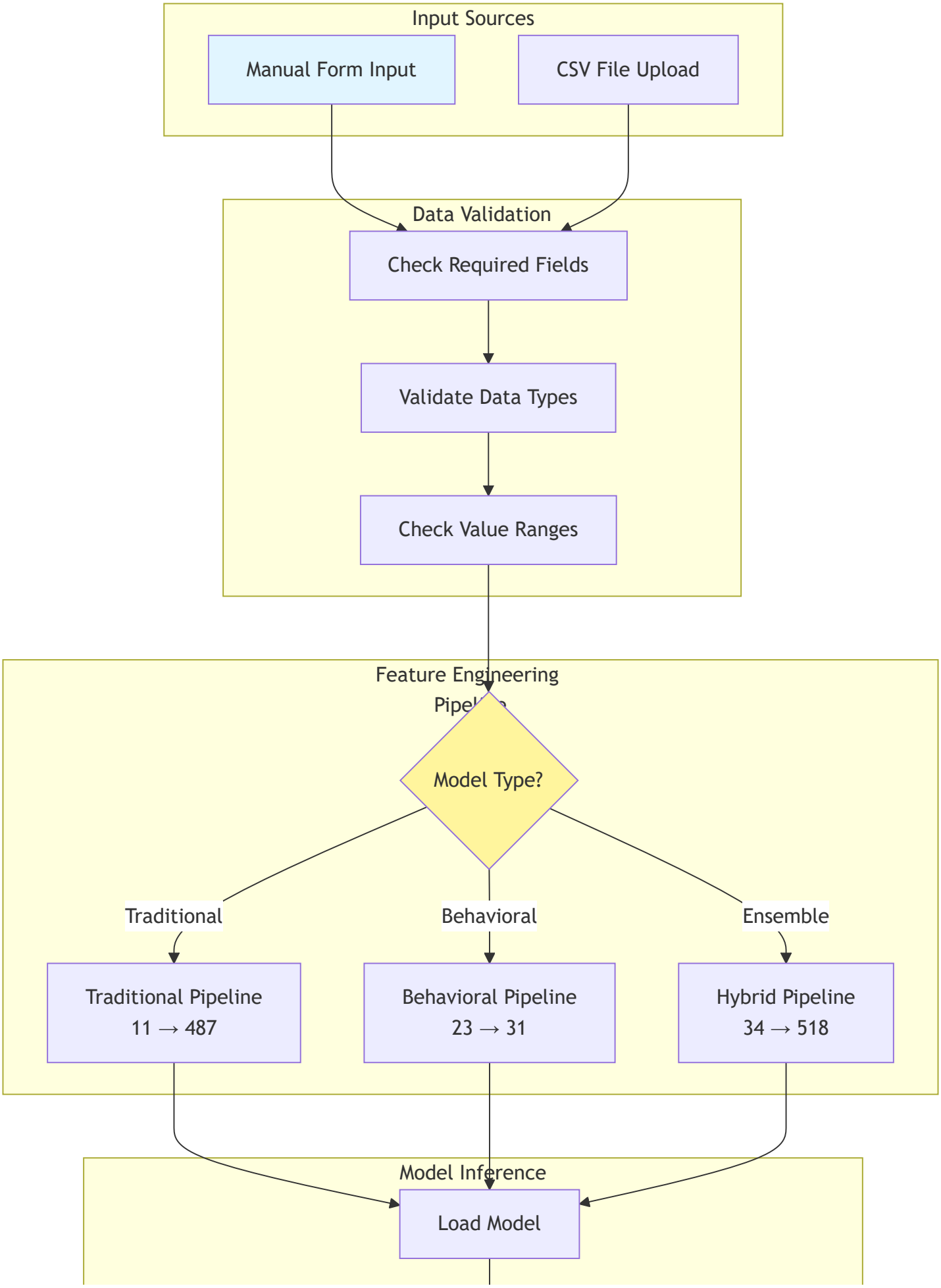
- **Streamlit**: Web application framework
- **LightGBM**: Gradient boosting model
- **Pandas**: Data manipulation
- **NumPy**: Numerical operations

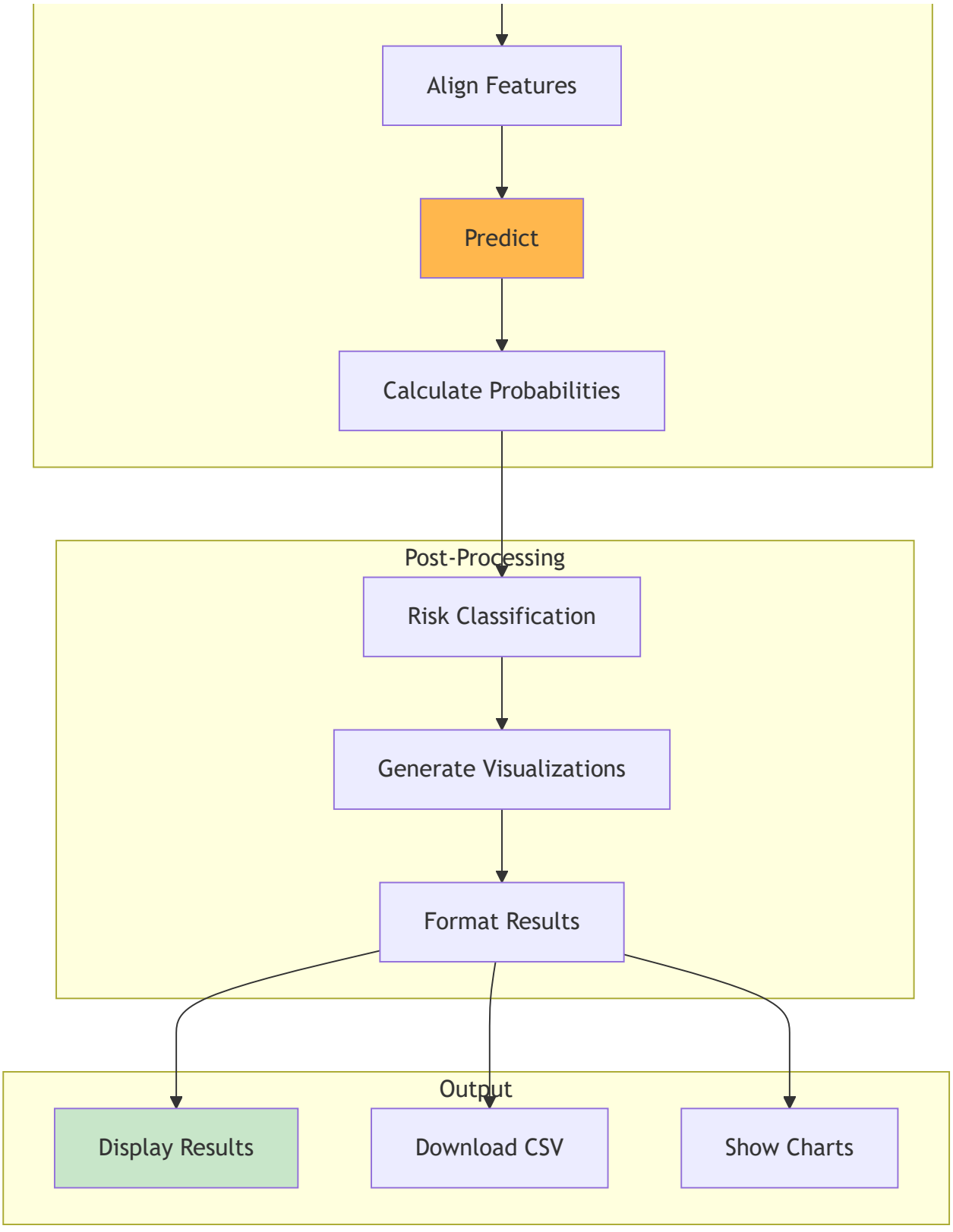
- **Plotly**: Interactive visualizations
- **SHAP**: Model interpretability
- **Scikit-learn**: ML utilities

5. Request-Response Flow

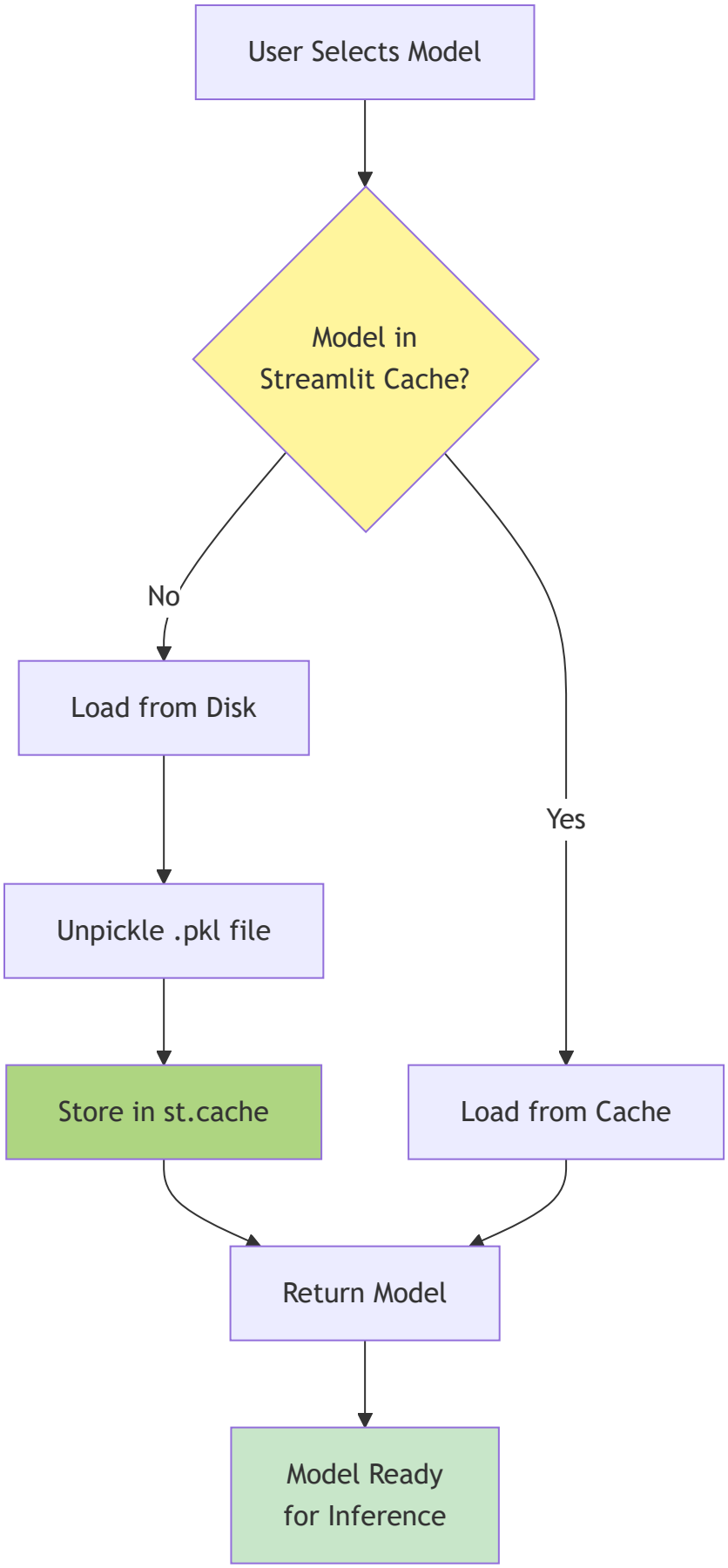

User
Streamlit UI
Prediction Page
apps/utils.py
Feature Engineering
LightGBM Model
Access Application
Navigate to Prediction
get_available_models()
Return [model1, model2, model3]
Show Model Selection
Select Model & Input Data
get_model_type(model_name)
Return 'traditional'/'behavioral'/'ensemble'
alt
[Manual Input]
Display Model-Specific Form
Fill Form (11/23/34 features)
[CSV Upload]
Upload CSV File
Load CSV with Pandas
Apply Feature Engineering
alt
[Traditional]
process_apps(11 features)
Return 487 features
[Behavioral]
behaviorial_features(23 features)
Return 31 features
[Ensemble]
process_apps + behaviorial_features
Return 518 features
get_predictions(model, data)
load_model(model_path)
align_features(data, model)
model.predict(X)
Return probabilities
Return [predictions, probabilities]
classify_risk(probability)
Return risk_level
plot_gauge(probability)
Return plotly_figure
Display Results + Visualization
User
Streamlit UI
Prediction Page
apps/utils.py
Feature Engineering
LightGBM Model

6. Data Flow Through System


Input Sources
Manual Form Input
CSV File Upload
Data Validation
Check Required Fields
Validate Data Types
Check Value Ranges
Feature Engineering
Model Type?
Traditional
Behavioral
Ensemble
Traditional Pipeline
11 → 487
Behavioral Pipeline
23 → 31
Hybrid Pipeline
34 → 518
Model Inference
Load Model


Align Features
Predict
Calculate Probabilities
Post-Processing
Risk Classification
Generate Visualizations
Format Results
Output
Display Results
Download CSV
Show Charts

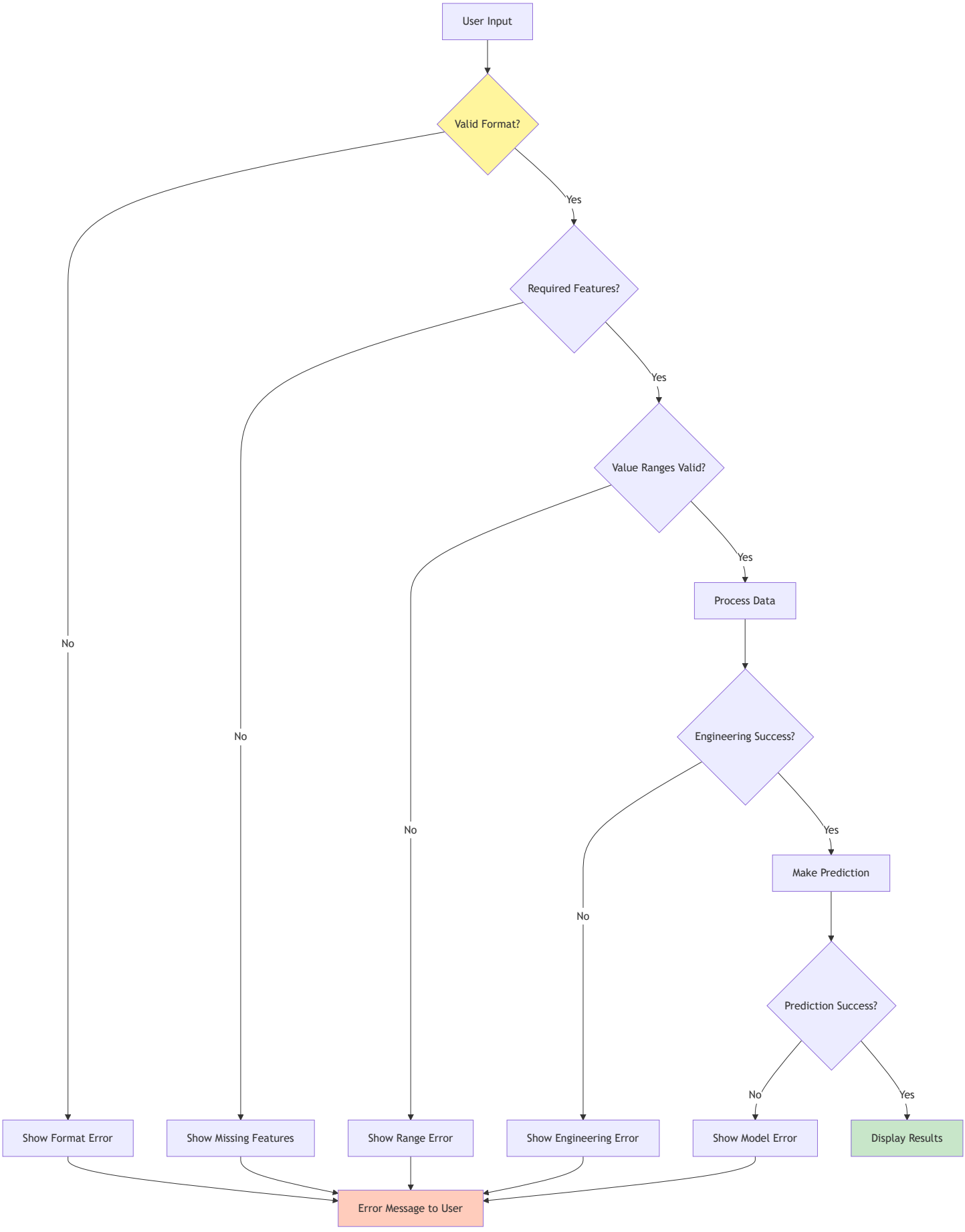
7. Model Loading & Caching Strategy



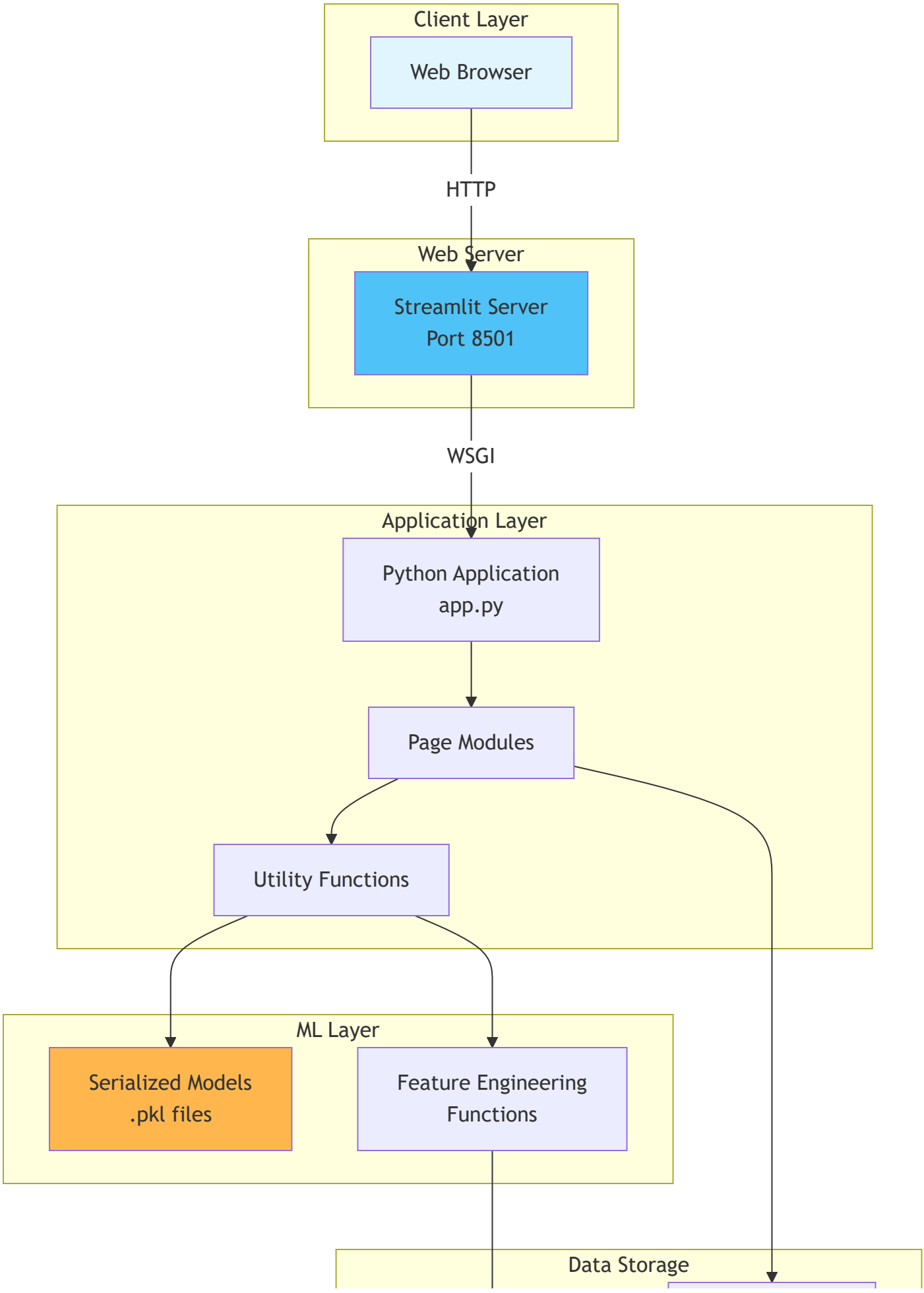
Caching Benefits:

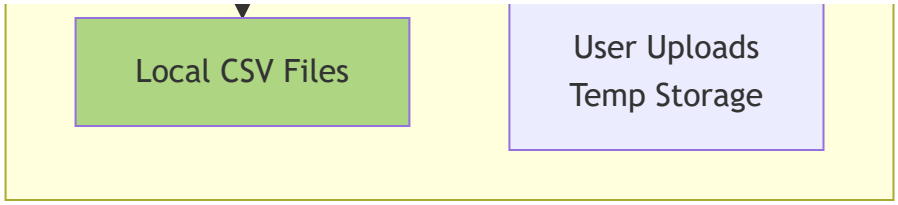
- Models loaded once per session
- Faster subsequent predictions
- Reduced memory overhead
- Better user experience

8. Error Handling & Validation



9. Deployment Architecture (Production Ready)

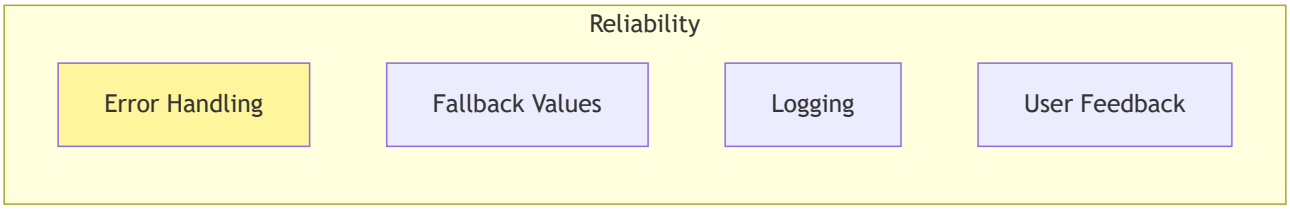

Client Layer
Web Browser
HTTP
Web Server
Streamlit Server
Port 8501
WSGI
Application Layer
Python Application
app.py
Page Modules
Utility Functions
ML Layer
Serialized Models
.pkl files
Feature Engineering
Functions
Data Storage

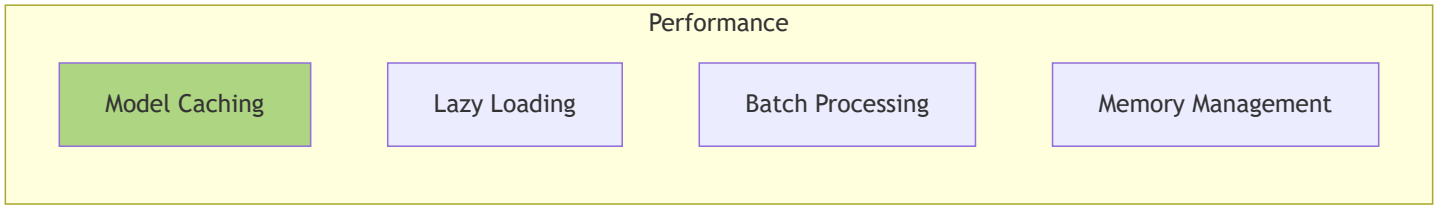


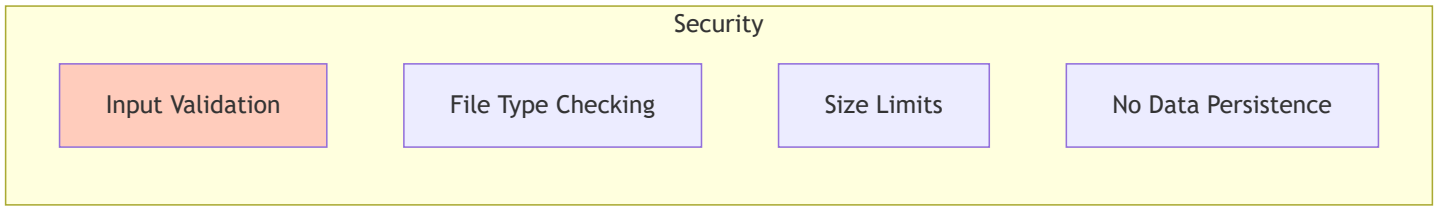
Deployment Options:

- **Local**: `streamlit run app.py`
- **Cloud**: Streamlit Community Cloud, Heroku, AWS, Azure
- **Docker**: Containerized deployment
- **Requirements**: Python 3.13, ~500MB models, 2GB RAM minimum

10. Security & Performance Considerations







Summary

This hybrid system provides three specialized models:

1. **Traditional Model**: Deep feature engineering from 11 base features → 487 features
2. **Behavioral Model**: Payment pattern analysis from 23 base features → 31 features

3. **Ensemble Model**: Combined approach using both feature sets → 518 features

Each model uses LightGBM for predictions and returns a probability (0-1) representing default risk, which is then classified into Low/Medium/High risk categories for actionable insights.

System Highlights:

- **Modular Architecture**: Clean separation of concerns (UI, Logic, Models, Data)
- **Scalable Design**: Easy to add new models or features
- **User-Friendly**: Streamlit provides intuitive interface
- **Production-Ready**: Error handling, caching, validation
- **Well-Documented**: Comprehensive flowcharts and architecture diagrams