# AI-Assisted Design Phase

**Prompt Used**
**AI Tool used: ChatGPT 5**
“Design a loan default probability prediction system for a commercial bank. Describe the system architecture including data inputs, preprocessing, model choice, training pipeline, and how it integrates into a loan approval workflow. Mention data flow, model deployment, and explainability considerations. Provide this in a concise way”

## AI Generated Response

**1. Goal**

Predict the **probability of default (PD)** for loan applicants and integrate it into the **loan approval workflow** to automate, explain, and monitor credit decisions.

**2. Architecture Overview**

[Internal + External Data]->[Data Lake → Feature Store]-> [Training Pipeline → Model Registry]-> [Scoring API]-> [Decision Engine]-> [Loan Approval Workflow + Monitoring]

**3. Data Inputs**

- **Applicant data:** age, employment, income, loan purpose, amount, term
- **Internal data:** past loans, payments, delinquencies, account activity
- **Credit bureau data:** utilization, delinquencies, score bands
- **Macroeconomic data:** unemployment, region risk factors
- **Target:** 90+ days past due or charge-off within 12 months

**4. Preprocessing**

- Clean and standardize data, remove leaks
- Engineer ratios (DTI, utilization), recency, and trend features
- Handle imbalance with class weights
- Apply **Weight of Evidence (WOE)** binning for interpretability
- Split data by time for training, validation, and out-of-time testing

**5. Model Choice**

- **Option A: Logistic Regression (scorecard)** — simple, explainable

- **Option B: Gradient Boosted Trees (LightGBM/XGBoost)** — higher accuracy
- Use **isotonic calibration** and store both champion and challenger models

**6. Training Pipeline**

- Automated ETL → feature selection → cross-validation → hyperparameter tuning
- Metrics: AUC, KS, Gini, Brier, PSI
- Output: trained model, calibration, explainer (SHAP), and documentation in **model registry**

**7. Deployment & Integration**

- **Model Serving:** containerized API (e.g., FastAPI) <150 ms latency
- **Input:** applicant features
- **Output:** PD score + reason codes
- **Decision Engine:** combines model score + policy rules
    - PD < T1 → approve
    - T1 ≤ PD < T2 → manual review
    - PD ≥ T2 → decline

**8. Explainability**

- **Global:** feature importance, SHAP summary
- **Local:** top 3–5 reason codes per applicant
- **Counterfactuals:** “PD would drop if utilization ≤ 30%”
- Used for underwriter UI and regulatory disclosure

**9. Monitoring & Retraining**

- Track AUC, KS, drift (PSI/CSI), latency, and overrides
- Monthly backtests by origination month
- Retrain quarterly or when drift exceeds threshold
- Keep audit logs for all decisions

**10. Workflow Integration**

1. Applicant submits data
2. Features pulled → model scores PD
3. Decision engine applies thresholds
4. Auto-approve / refer / decline
5. Underwriter sees reasons and can override
6. Feedback loop updates future retraining

**11. Security & Governance**

- All PII encrypted; audit trails for each decision
- Document model lineage and performance for compliance
- Role-based access and periodic validation

## Analysis and Improvements

**1. Weaknesses / Risks**

1. Data bias: Historical data may reflect past discrimination or unequal access to credit.
2. Explainability risk: Gradient boosting is not inherently transparent — officers may not understand why a loan was rejected.
3. Data privacy and compliance: Sensitive borrower data (income, credit score) must meet GDPR/ISO standards for protection.

**2. Improvements**

Propose concrete ways to fix or enhance:

| Issue | Suggested Improvement |
|---|---|
| Data bias | Apply fairness-aware preprocessing (reweighting or resampling to balance groups). Include fairness metrics (equal opportunity difference). |
| Explainability | Add an explainable AI layer using SHAP or LIME to visualize influential features for each decision. |
| Data privacy | Use encryption in transit (TLS) and at rest (AES-256). Implement access control and anonymize data during retraining. |

| Issue | Suggested Improvement |
|---|---|
| Model drift | Set up a monitoring pipeline to detect performance degradation and trigger retraining. |
| Human oversight | Ensure final approval remains human-led with clear justification logs. |

### 3. Conclusion

Through this activity, I learned that while AI can quickly propose technically solid architectures, human critical review is essential to identify ethical, security, and fairness gaps. Designing ML systems responsibly requires not just accuracy but explainability, compliance, and continuous monitoring to maintain trust in financial decision-making.