Credit Path AI – Batch Loan Default Predictor

1. Overview

The system is designed to analyze multiple borrowers at once and predict the risk of loan default. It provides actionable recommendations for each borrower and logs all predictions for auditing or future analysis.

Frontend: HTML + JavaScript form to enter multiple borrowers' details.

Backend: FastAPI server handling batch predictions, scoring, and recommendations.

Data Logging: All predictions logged to predictions.log.

Error Handling: Handles invalid or missing fields and connection errors.

2. Actions / Workflow

Input Submission (Frontend)

Users can add multiple borrowers dynamically using "+ Add Borrower".

Each borrower has fields:

- Loan Amount
- Installment (%)
- Annual Income
- Credit Score

Users submit all borrower data by clicking Analyze Risk.

Data Processing (Frontend → Backend)

The frontend collects all borrower inputs and sends a POST request to /predict_batch endpoint as JSON:

```
{"borrowers": [{"loanAmount": 50000, "installment": 10 ,
"income":80000 , "creditScore": 700},
{"loanAmount": 100000, "installment": 15, "income": 120000,
 "creditScore": 650 }]}
```

Batch Prediction (Backend)

For each borrower:

Risk Score Calculation:

risk_score = 0.5*(loanAmount/income) + 0.3*(installment/100) - 0.2*(creditScore/850)
risk_score = min(max(risk_score, 0), 1)

Risk Classification & Recommendation Mapping:

risk_score > 0.5 → High Risk → "Review loan or require collateral"
0.2 < risk_score ≤ 0.5 → Medium Risk → "Consider partial collateral or adjust installment"
risk_score ≤ 0.2 → Low Risk → "Loan likely safe to approve"

Backend returns an array of borrower results, including risk score, prediction, color, and recommendation.

Response Handling (Frontend)

- ➔ Displays color-coded result cards per borrower.
- ➔ Shows a risk bar and recommendation.
- ➔ Computes average risk score across borrowers.
- ➔ Handles server errors and displays user-friendly messages.

3. Logging

All predictions are logged using Python's logging module:

```
import loggingimport jsonfrom datetime import
datetimelogging.basicConfig
(filename="predictions.log", level=logging.INFO, format='%(asctime)s
%(message)s')
logging.info(json.dumps({    "timestamp": str(datetime.now()),
"borrower": idx+1,"input": borrower_data,    "output": result}))
```

Logged Information:

- ➔ Timestamp
- ➔ Borrower index
- ➔ Input data (loan amount, income, installment, credit score)
- ➔ Output (prediction, color, risk score, recommendation)

Purpose: Auditing, traceability, and future analytics.

Example log entry:


```
{"timestamp": "2025-10-12 23:55:12", "borrower": 1, "input":
{"loanAmount":50000,"installment":10,"income":80000,"creditScore":700}
,"output":{"prediction": "Medium Risk of Default", "color":"orange"
,"risk_score":0.34,"recommendation":"Consider partial collateral or
adjust installment"}}
```


4. Error Handling

Backend

- Invalid numeric inputs (negative values, zero income, etc.) handled per borrower:

  ```
  if b.loanAmount <= 0 or b.income <= 0 or b.installment < 0 or
  b.creditScore < 0:
  results.append({"error": f"Invalid input for borrower {idx+1}"})
  continue
  ```

- Catches unexpected exceptions and returns errors for specific borrowers without crashing the batch process.

Frontend

- Field validation: HTML required, min, and max attributes ensure correct numeric input.
- Server errors / connectivity issues : Displays a clear message if backend is unreachable or returns a non-200 response.
- At least one borrower: Prevents deleting all rows to maintain valid input.