✅ Tech Evaluation Use Case: Credit Risk Configuration & Monitoring Portal

Inspired by: Durga Analytics | Credit Risk ML Platform for an NBFC

🎯 Objective

Design and develop a **multi-tenant full stack portal** for configuring, triggering, and monitoring ML-based credit scoring for a fictional NBFC. The portal should allow credit analysts and field officers to manage data integrations, scoring APIs, and view real-time risk predictions and performance monitoring dashboards.

🧩 Core Features to Implement

1. Customer Onboarding & Tenant Management

- Create new tenant (NBFC) with:
  - Tenant ID, NBFC name, admin user
- Isolated view per tenant using RBAC

2. Bureau API Configuration

- UI for configuring credentials and endpoints for 3 bureau APIs
- Store credentials securely (mock calls acceptable)

3. Model Trigger UI

- Upload dummy applicant data (CSV/JSON)
- Trigger scoring API (mocked FastAPI) that returns:
  - Risk score
  - Prediction: approve/reject
  - Top 3 SHAP reasons

4. Model Monitoring Dashboard

- Embed Power BI dashboard (use iframe placeholder)
- View:
  - Delinquency trends
  - Feature drift alerts
  - NPA prediction summary

5. User Roles & Permissions

- `Risk Analyst` : Can configure models, view all tenants
- `Field Officer` : Can only score applications
- `Admin` : Full access per tenant

📦 Technology Expectations (Based on Real Project Stack)

Layer	Preferred Tech Stack
Frontend	React (preferred), TypeScript, HTML/CSS
Backend	Django (admin, ORM), FastAPI
Database	PostgreSQL or SQLite
Authentication	JWT or OAuth2
Dashboard Embed	Power BI iframe/token
Optional	Docker, AWS Lambda-style mocks, SHAP mock JSON viewer

---

🔐 Security Goals

- Multi-tenant RBAC enforcement
- Secure API auth via JWT (token expiry, refresh)
- Protect all endpoints with permissions
- Simulate secure data handling (e.g., redact API credentials on UI)

🧪 Evaluation Flow

1. **Admin logs in**, creates a new NBFC tenant.
2. **Risk Analyst configures** API endpoints for CIBIL, Experian, CRIF (simulated).
3. **Field Officer uploads** sample application JSON and triggers score.
4. Backend returns:
    - Score: 0–100
    - Risk: Approve/Reject
    - SHAP explanation: “Low bureau score”, “High credit utilization”

5. Risk Analyst opens **monitoring dashboard** (Power BI iframe) to inspect delinquency trend.

📊 Sample Output JSON (Mocked API Score Response)

```
{ "score": 43, "decision": "Reject", "shap_explanation": [ "Low bureau score", "Recent delinquency", "High EMI to Income ratio" ] }
```

📁 Suggested Folder Structure

```
project/
├── frontend/ (React app)
├── backend/
│   ├── api/ (FastAPI)
│   ├── admin/ (Django admin)
│   ├── auth/ (OAuth2/JWT)
│   └── models/
├── db/ (PostgreSQL or SQLite setup)
├── dashboard/ (iframe embed)
└── docker-compose.yml
```

✅ Bonus Points

- Simulate SHAP explanations with visuals (bar chart or list)
- Add retry logic on API failures
- Add CSV upload + preview in frontend

🧠 Evaluation Criteria

Skill Area	Criteria
Frontend	Dynamic forms, error handling, role-based visibility
Backend	Modular API, tenant auth, data modeling

Skill Area	Criteria
Security	JWT handling, secure config storage
Monitoring	Dashboard embed, log rendering
Code Quality	Clear structure, reuse, docstrings, tests
Completeness	End-to-end flow, from config to score to dashboard