SpendSense

From Plaid to Personalized Learning

Background

Banks generate massive transaction data through Plaid integrations but struggle to turn it into actionable customer insights without crossing into regulated financial advice.

Your challenge: Build an explainable, consent-aware system that detects behavioral patterns from transaction data, assigns personas, and delivers personalized financial education with clear guardrails around eligibility and tone.

Project Overview

Individual or small team project with no strict deadline.

Deliverables:

- Synthetic Plaid-style data generator (50-100 users)
- Feature pipeline detecting subscriptions, savings, credit, income patterns
- Persona assignment system (5 personas)
- Recommendation engine with plain-language rationales
- Consent and eligibility guardrails
- Operator view for oversight
- Evaluation harness with metrics

Core Requirements

1. Data Ingestion (Plaid-Style)

Create synthetic data matching Plaid's structure:

Accounts:

- account_id
- type/subtype (checking, savings, credit card, money market, HSA)
- balances: available, current, limit
- iso_currency_code
- holder_category (exclude business accounts)

Transactions:

- account_id
- date
- amount
- merchant_name or merchant_entity_id
- payment_channel
- personal_finance_category (primary/detailed)
- pending status

Liabilities:

- Credit cards: APRs (type/percentage), minimum_payment_amount, last_payment_amount, is_overdue, next_payment_due_date, last_statement_balance
- Mortgages/Student Loans: interest_rate, next_payment_due_date

Requirements:

- Generate 50-100 synthetic users
- No real PII—use fake names, masked account numbers
- Diverse financial situations (various income levels, credit behaviors, saving patterns)
- Ingest from CSV/JSON (no live Plaid connection required)

2. Behavioral Signal Detection

Compute these signals per time window (30-day and 180-day):

Subscriptions:

- Recurring merchants (≥3 in 90 days with monthly/weekly cadence)
- Monthly recurring spend
- Subscription share of total spend

Savings:

- Net inflow to savings-like accounts (savings, money market, cash management, HSA)
- Growth rate
- Emergency fund coverage = savings balance / average monthly expenses

Credit:

- Utilization = balance / limit
- Flags for ≥30%, ≥50%, ≥80% utilization
- Minimum-payment-only detection
- Interest charges present
- Overdue status

Income Stability:

- Payroll ACH detection
- Payment frequency and variability
- Cash-flow buffer in months

3. Persona Assignment (Maximum 5)

Assign each user to a persona based on detected behaviors:

Persona 1: High Utilization

Criteria:

- Any card utilization ≥50% OR interest charges > 0 OR minimum-payment-only OR is_overdue = true

Primary Focus:

- Reduce utilization and interest; payment planning and autopay education

Persona 2: Variable Income Budgeter

Criteria:

- Median pay gap > 45 days AND cash-flow buffer < 1 month

Primary Focus:

- Percent-based budgets, emergency fund basics, smoothing strategies

Persona 3: Subscription-Heavy

Criteria:

- Recurring merchants ≥3 AND (monthly recurring spend ≥$50 in 30d OR subscription spend share ≥10%)

Primary Focus:

- Subscription audit, cancellation/negotiation tips, bill alerts

Persona 4: Savings Builder

Criteria:

- Savings growth rate ≥2% over window OR net savings inflow ≥$200/month, AND all card utilizations < 30%

Primary Focus:

- Goal setting, automation, APY optimization (HYSA/CD basics)

Persona 5: [Your Custom Persona]

Create one additional persona and document:

- Clear criteria based on behavioral signals
- Rationale for why this persona matters
- Primary educational focus
- Prioritization logic if multiple personas match

4. Personalization & Recommendations

Output per user per window:

- 3-5 education items mapped to persona/signals
- 1-3 partner offers with eligibility checks
- Every item includes a "because" rationale citing concrete data
- Plain-language explanations (no jargon)

Example rationale format:

"We noticed your Visa ending in 4523 is at 68% utilization ($3,400 of $5,000 limit). Bringing this below 30% could improve your credit score and reduce interest charges of $87/month."

Education Content Examples:

- Articles on debt paydown strategies
- Budget templates for variable income
- Subscription audit checklists
- Emergency fund calculators
- Credit utilization explainers

Partner Offer Examples:

- Balance transfer credit cards (if credit utilization high)
- High-yield savings accounts (if building emergency fund)
- Budgeting apps (if variable income)
- Subscription management tools (if subscription-heavy)

5. Consent, Eligibility & Tone Guardrails

Consent:

- Require explicit opt-in before processing data
- Allow users to revoke consent at any time
- Track consent status per user
- No recommendations without consent

Eligibility:

- Don't recommend products user isn't eligible for
- Check minimum income/credit requirements
- Filter based on existing accounts (don't offer savings account if they have one)
- Avoid harmful suggestions (no payday loans, predatory products)

Tone:

- No shaming language
- Empowering, educational tone
- Avoid judgmental phrases like "you're overspending"
- Use neutral, supportive language

Disclosure:

Every recommendation must include: "This is educational content, not financial advice. Consult a licensed advisor for personalized guidance."

6. Operator View

Build a simple interface for human oversight:

- View detected signals for any user
- See short-term (30d) and long-term (180d) persona assignments
- Review generated recommendations with rationales
- Approve or override recommendations
- Access decision trace (why this recommendation was made)
- Flag recommendations for review

7. Evaluation & Metrics

Build an evaluation system that measures:

- Coverage: % of users with assigned persona and ≥3 detected behaviors
- Explainability: % of recommendations with plain-language rationales
- Relevance: manual review or simple scoring of education-persona fit
- Latency: time to generate recommendations (should be fast on laptop)
- Fairness: basic demographic parity check if synthetic data includes demographics

Output:

- JSON/CSV metrics file
- Brief summary report (1-2 pages)
- Per-user decision traces

Technical Architecture

Modular Structure

Organize code into clear modules:

- ingest/ - Data loading and validation
- features/ - Signal detection and feature engineering
- personas/ - Persona assignment logic
- recommend/ - Recommendation engine
- guardrails/ - Consent, eligibility, tone checks
- ui/ - Operator view and user experience
- eval/ - Evaluation harness
- docs/ - Decision log and schema documentation

Storage

Use local storage:

- SQLite for relational data
- Parquet for analytics
- JSON for configs and logs

API

Build a simple REST API for:

- POST /users - Create user
- POST /consent - Record consent
- GET /profile/{user_id} - Get behavioral profile
- GET /recommendations/{user_id} - Get recommendations
- POST /feedback - Record user feedback
- GET /operator/review - Operator approval queue

AI Integration (Optional)

While not required, you may use:

- LLMs for generating educational content text
- Simple ranking/bandit algorithms for offer selection
- Multimodal models for image/video content (optional)

Rules-based baseline is acceptable. Focus on explainability over sophistication.

Code Quality Requirements

- Clear modular structure (see architecture above)
- One-command setup (requirements.txt or package.json)
- Concise README with setup and usage instructions
- ≥10 unit/integration tests

- Deterministic behavior (use seeds for randomness)
- Decision log in /docs explaining key choices
- Explicit limitations documented
- Standard "not financial advice" disclaimer

Success Criteria

Your project will be evaluated on these metrics:

Category	**Metric**	**Target**
Coverage	Users with assigned persona + ≥3 behaviors	100%
Explainability	Recommendations with rationales	100%
Latency	Time to generate recommendations per user	<5 seconds
Auditability	Recommendations with decision traces	100%
Code Quality	Passing unit/integration tests	≥10 tests
Documentation	Schema and decision log clarity	Complete

Additional Requirements:

- All personas have clear, documented criteria
- Guardrails prevent ineligible offers
- Tone checks enforce "no shaming" language
- Consent is tracked and enforced
- Operator view shows all signals and can override
- Evaluation report includes fairness analysis
- System runs locally without external dependencies

User Experience Requirements

Create a simple, usable end-user experience. Options:

- Web app mock showing personalized dashboard
- Email preview templates
- Chat interface for Q&A
- Content feed (like social media)
- Mobile app mockup (Figma/screenshots acceptable)

Creative formats welcome:

- Generated images/infographics
- Short video content
- Interactive calculators
- Gamified savings challenges

Submission Requirements

Submit the following:

- Code repository (GitHub preferred)
- Brief technical writeup (1-2 pages)
- Documentation of AI tools and prompts used
- Demo video or live presentation
- Performance metrics and benchmarks
- Test cases and validation results
- Data model/schema documentation
- Evaluation report (JSON/CSV + summary)

Technical Contact

For questions or clarifications:

Bryce Harris - bharris@peak6.com

Build Strategy

Recommended phases:

- Data Foundation: Generate synthetic dataset, validate schema
- Feature Engineering: Build signal detection for subscriptions, savings, credit, income
- Persona System: Implement assignment logic and prioritization
- Recommendations: Build engine with rationales and content catalog
- Guardrails & UX: Add consent, eligibility, tone checks; build operator view
- Evaluation: Run metrics harness, document results and limitations

Final Note

Financial AI must be explainable and auditable. Every recommendation needs a clear rationale that cites specific data points.

Core principles:

- Transparency over sophistication
- User control over automation
- Education over sales
- Fairness built in from day one

Build systems people can trust with their financial data.