Project Requirements Document

1. Project Overview

Project Name: KYC-Rysk-Classifier

Purpose: An AI-assisted KYC triage service that classifies customers into Low/Medium/High/Critical risk levels by combining a fine-tuned transformer model (MiniLM-L6) with policy-first rules (e.g., sanctions overrides) and provides explainable reasons for analyst triage and compliance audits.

Key Features:

- Risk classification (single/batch input, with probabilities and reasons)
- Audit logging and search/export (tamper-proof HMAC logs)
- Dashboard for metrics (latency, distribution, charts)
- File upload (CSV/Excel/txt for batch processing)
- Policy overrides (e.g., sanctions → HIGH regardless of model)

Target Users: Internal compliance analysts at banks/financial institutions for AML/KYC onboarding triage.

Technology Stack:

- Backend: Python, FastAPI, PyTorch (MiniLM model), Pydantic for validation
- Frontend: Vanilla HTML/JS, Chart.js (CDN), Nginx for serving
- Database: None (in-memory for batch jobs; flat files for audit logs)
- Infrastructure: Docker, docker-compose, Hetzner CX22 (CPU)
- ML/AI: PyTorch MiniLM-L6 (transformer for text classification), no GPU required

Deployment:

- Environment: Hetzner EU (GDPR-compliant), local dev via Docker
- Architecture: Containerized microservices (UI + API), stateless API

Compliance/Regulatory Context: GDPR (EU-hosted, PII redaction, right to deletion via log retention); FATF AML standards (sanctions/PEP checks); financial industry best practices (audit trails, no personalized advice).

---

2. Functional Requirements

2.1 Core Feature 1: Risk Classification (Single/Batch)

Description: Classifies client risk based on evidence text, with optional policy overrides for sanctions hits.

Input:

- Format: JSON `{text: string, override: bool}` for API; UI textarea or file upload (.txt, .csv, .xlsx, .jsonl)
- Fields: `text` (required with [KYC] and [COUNTRY] tags), `override` (bool)
- Validation: Must include [KYC] and [COUNTRY] tags; server-side length check; API key required

Output:

- Format: JSON
- Fields: `label` (string: LOW/MEDIUM/HIGH/CRITICAL), `probs` (dict: low/medium/high probabilities), `rule` (string: model_only or sanctions_override), `why` (list of reasons), `auto_clear` (bool), `policy_config_hash` (string)
- Example: `{"label": "LOW", "probs": {"low": 0.9, "medium": 0.08, "high": 0.02}, "rule": "model_only", "why": ["country=DE", "no sanctions"], "auto_clear": true}`

Endpoints/Actions:

- `POST /classify` - Single classification
- `POST /classify_batch` - Batch classification (up to 256 items)

Business Logic:

- Model: MiniLM softmax for probabilities; label = argmax probability
- Override: If sanctions code present and policy.enable=true, label = HIGH, rule = sanctions_override
- FATF bump: High-risk countries nudge low/medium to next level if bump_threshold > probs[low/medium]
- Auto-clear: Confidence > thresholds based on configured values

Edge Cases:

- Empty input → 400 error "missing required tags"
- Model load failure → 500 "Model loading failed"
- Sanctions override dry-run → return override_would_apply=true without changing label

---

2.2 Core Feature 2: Batch Processing and File Upload

Description: Handles bulk risk assessments via file upload or item array.

Supported Operations:

- Single processing: Direct UI input or POST /classify
- Batch processing: File upload (CSV/Excel) to POST /classify_batch, processes arrays
- Bulk operations: Maximum 256 items per batch

File Handling:

- Accepted formats: .txt (fills textarea), .csv/.xlsx/.xls/.jsonl (batch)
- Size limits: 10MB max file, 256 item max array
- Row limits: 50 rows preview (client), 256 server process
- Validation: Client parses XLSX via SheetJS; server rejects unsupported MIME types; content blacklist filtering based on codeblacklist.txt

Processing Flow:

1. Upload file → client parse (SheetJS for Excel)
2. Preview table (first 50 rows)
3. Submit to API → batch processing
4. Download results as CSV with id, label, rule, probs, why columns

---

2.3 Core Feature 3: Audit Trail

Description: Logs all decisions for search/export with cryptographic integrity checks.

Data Storage:

- What is logged: timestamp, text_hash, label, rule, probs, why, sanc_codes, country, meta_tags, req_id, HMAC signature
- Format: JSONL (one record per line) in logs/audit.jsonl
- Retention: Configurable, default 90 days

Search Capabilities:

- Query parameters: label, rule, band, country, owner, lang, since, until

- Filters: Case-insensitive matches, time ranges
- Pagination: Cursor-based, next 100-500 items

Export Functionality:

- Export formats: CSV (columns: ts, label, rule, why)
- Export endpoints: `GET /audit_export`
- CLI tools: scripts/search_audit.py (filtering), tools/verify_audit.py (HMAC verification)

Audit Trail:

- Integrity verification: HMAC SHA-256 over stable fields
- Tamper-proofing: Logs redact PII; no raw text stored
- Verification tools: tools/verify_audit.py prints OK/FAIL per log line

---

2.4 Analytics & Reporting

Dashboard Features:

- Risk distribution over time: Stacked bar chart showing LOW/MEDIUM/HIGH counts
- Recent activity list: Latest N audit items with labels and reasons
- Stats: % low/medium/high, total assessments, average response time

Visualizations:

- Bar chart: 7-day risk distribution by label (Chart.js)
- Donut chart: Probability breakdown for single assessments (UI)
- Radar chart: LOW/MED/HIGH probabilities visualization (UI)

Time Ranges:

- Default: 7 days
- Configurable: Hardcoded to 7 days in UI

Metrics Exposure:

- Endpoint: `/metrics`
- Format: Prometheus text format

- Tracked metrics:
  - `request_latency_seconds{route}` - Request duration histogram
  - `decisions_total` - Total classification decisions
  - `pred_total{label,path}` - Predictions by label and endpoint
  - `sanctions_present_total` - Sanctions detected counter
  - `sanctions_override_total` - Override applied counter
  - `autoclear_low_total` - Auto-cleared low-risk counter
  - `redactions_applied{type}` - PII redactions by type
  - `audited_total` - Total audited requests

2.5 User Interface

Navigation Structure:

- Tab 1: Classify - Manual assessment with textarea and sample chips
- Tab 2: Audit - Search/export functionality with filters
- Tab 3: Dashboard - Charts and statistics visualization
- Tab 4: Batch Assess - File upload interface (feature flag gated)

Interactive Elements:

- Textarea for evidence input
- Override toggle (sanctions policy switch)
- Sample chips (predefined test cases)
- File upload dropzone with drag-and-drop
- Search filters and inputs for audit queries

User Actions:

- Paste/assess evidence: Displays label, probabilities, and explainable reasons
- Toggle override: Enables/disables sanctions policy logic
- Upload file: Loads/previews batch data, downloads results CSV
- Search audit: Filters and paginates historical decisions

Visual Feedback:

- Loading states: Skeleton screens during API requests
- Success/error messages: Toast notifications (top-right)
- Progress indicators: Batch upload progress bar

Theming:

- Available themes: Light, dark
- User preferences: LocalStorage persistence with toggle button

---

3. Non-Functional Requirements

3.1 Performance

Response Times:

- Single requests: <500ms target (typical 30-100ms on CPU)
- Batch operations: ~200 items/60 seconds (extrapolated 5k items/min)
- Cold start: <1 second for model loading

Resource Constraints:

- Memory: ~1GB RAM (PyTorch model footprint)
- CPU: 2-4 vCPUs (Hetzner CX22 specifications)
- Storage: 10GB allocation (logs and configuration)

Optimization:

- Caching strategy: Model preloaded at startup; no external cache layer
- Rate limiting: Not implemented (trusted internal users)
- Batch size limits: Maximum 256 items per array submission

3.2 Scalability

Architecture:

- Stateless/Stateful: Stateless API design

- Horizontal scaling: Yes, fully stateless and replicable
- Load balancing: Not implemented (single container deployment)

Resource Requirements:

- Compute: CPU-only (no GPU dependency)
- Memory: 1GB base + additional for batch processing
- Storage: Flat files for logs with 90-day retention

Data Management:

- Retention policy: 90 days (configurable via environment)
- Archival strategy: None (deletion on schedule)
- Cleanup procedures: Manual or cron job (not currently implemented)

3.3 Security

Authentication:

- Method: API key via X-API-Key header
- Endpoints: /classify, /classify_batch, /audit* protected when API_KEY environment variable is set
- Key rotation: Manual rotation via environment configuration

Data Protection:

- PII handling: Redaction to placeholders (email → **EMAIL**, IBAN → last 4 digits)
- Sensitive data: Evidence text hashed, no raw storage in audit logs
- Data at rest: Flat files unencrypted (assumes secure host environment)
- Data in transit: TLS if HTTPS enabled (optional Nginx configuration)

Access Control:

- CORS policy: Restricted to localhost:3000 and 46.62.218.2:3000
- IP whitelisting: Not implemented
- Role-based access: Not implemented (single internal role)

Audit Security:

- Tamper detection: HMAC SHA-256 signature per log entry
- Log integrity: Verification via tools/verify_audit.py
- Backup security: Not applicable (no automated backups)

Compliance:

- Regulations: GDPR (EU hosting requirement)
- Data residency: EU (Hetzner data centers)
- Right to deletion: Delete logs/audit.jsonl file

3.4 Reliability

Uptime Target: 99% SLA (based on Hetzner 99.9% infrastructure uptime)

Error Handling:

- Timeout handling: 60-second timeout for batch operations
- Malformed requests: 400 status with JSON "detail" field
- Service unavailability: 500 status with HTML error page

Graceful Degradation:

- Feature fallbacks: Charts show alternative text if JavaScript fails
- Partial functionality: API remains operational without UI

Backup & Recovery:

- Backup frequency: None automated (manual backups)
- Backup scope: logs/ and config/ directories
- Recovery time objective (RTO): Minutes (container restart)
- Recovery point objective (RPO): Hours (no daily automated backups)

Monitoring:

- Health checks: `GET /health` endpoint (validates model and policies loaded)
- Alerting: Not implemented (external Grafana/Prometheus recommended)
- Logging: JSONL audit logs with configurable log_path

3.5 Usability

Responsive Design:

- Breakpoints: <768px (mobile stack layout), <1024px (tablet side-by-side)
- Mobile-first: No (desktop-first approach)

Accessibility:

- WCAG level: A compliance (ARIA labels, title attributes)
- Screen reader support: Role alerts, aria-live regions for status updates
- Keyboard navigation: Tab, Enter, Escape key support
- Color contrast: High contrast (customizable themes)

User Experience:

- Intuitive navigation: Tab-based interface, quick-action chips, toast notifications
- Help/documentation: Inline tooltips, comprehensive README
- Status messages: Toast notifications for success/error states
- Error messages: Actionable feedback ("empty input", "network error")

Input Constraints:

- File size limits: 10MB maximum (client-side warning)
- Row/item limits: 50-row preview, 256 maximum processing (UI warnings)
- Format validation: Real-time validation via accept attributes

---

3.6 Maintainability

Containerization:

- Platform: Docker
- Base images: nginx:alpine (UI), python:3.11-slim (API)
- Orchestration: docker-compose (1 API container + 1 UI container)

Configuration:

- Format: YAML (config/risk_rules.yaml) + environment variables

- Environment variables: IS_LOCAL, API_KEY, MODEL_CKPT
- Secrets management: .env files in kyc-risk-minilm/ directory

Testing:

- Unit tests: 15+ test files covering sanctions, batch, audit functionality
- Integration tests: API smoke tests
- E2E tests: Not implemented (manual UI testing)
- Test files: tests/ directory (pytest framework)

CI/CD:

- Pipeline stages: Not implemented (manual deployments)
- Quality gates: Unit tests + evaluation (macro F1 >0.90, high recall >0.99)
- Automated checks: Not implemented

Code Quality:

- Code style: No linter enforced
- Documentation: Inline docstrings, comprehensive README
- Version control: Git with remote branch tracking

3.7 Cost

Infrastructure:

- Monthly cost: ~€10 (Hetzner CX22 instance)

Resource Usage:

- Bandwidth: Low (text-only API requests)
- Storage: ~1GB/year (audit logs with 90-day retention)
- Compute: CPU-only, lightweight processing

External Services:

- APIs: None (no external API dependencies)
- Third-party tools: None (Chart.js via CDN only)

3.8 Environment & Deployment

Development Environment:

- Local setup: `docker-compose up` for full stack
- Environment flags: IS_LOCAL=true for development proxy configuration
- Mock data: Not applicable (uses real trained model)

Production Environment:

- Hosting: Hetzner EU data centers
- Networking: CORS allow specific IP, /api nginx reverse proxy
- Domain/DNS: http://46.62.218.2:3000 (no custom domain)
- SSL/TLS: None (HTTP only)

Deployment Process:

- Manual/automated: Manual docker-compose deployment
- Rollback procedure: Stop old containers, start new containers
- Zero-downtime: No (single container limitation)

Environment Parity:

- Dev/staging/prod: Same codebase, different environment variables
- Configuration differences: IS_LOCAL (dev=true, prod=false); API_KEY (prod set, dev optional)

---

4. Constraints & Assumptions

Constraints

- No GPU available (CPU-only MiniLM inference)
- No persistent database (flat files and in-memory logs only)
- Time constraint: 2-week initial build window
- Resource constraint: Single vCPU with 4GB RAM allocation

Assumptions

- Users are internal compliance analysts (not public-facing)
- Data is untrusted but compliant (no banned/illegal content)
- Infrastructure runs Docker on Linux environment
- No external system integrations required

---

5. Dependencies

External Services:

- None (fully self-contained)

Third-Party Libraries:

- fastapi==0.104 - API framework with async support
- torch==2.1.2 - ML model inference engine
- transformers==4.36.2 - Hugging Face model loading
- pandas - CSV/Excel processing
- rapidfuzz - Fuzzy string matching for entity names

Internal Dependencies:

- None (standalone application)

---

6. Future Considerations

Planned Features:

- Name/entity fuzzy matching for sanctions list comparison
- Adverse media integration for enhanced risk signals
- Multi-model support (e5-small alternative)

Scalability Plans:

- Kubernetes deployment for horizontal scaling
- Redis integration for persistent batch state storage

Technical Debt:

- No persistent batch state (in-memory only, lost on restart)
- UI not fully tested for screen reader accessibility
- No automated backup solution for audit logs

---

7. Glossary

Term	Definition
KYC	Know Your Customer compliance process
AML	Anti-Money Laundering regulations
FATF	Financial Action Task Force guidelines
HMAC	Hash-based Message Authentication Code for integrity verification
PII	Personally Identifiable Information
GDPR	General Data Protection Regulation
Sanctions Override	Policy rule to set HIGH risk if sanctions code present
Auto-clear	High-confidence low-risk assessments that skip manual review

---

8. Approval & Sign-off

Role	Name	Date	Signature
Project Manager	N/A	2025-10-21	N/A
Tech Lead	N/A	2025-10-21	N/A
Stakeholder	N/A	2025-10-21	N/A
Compliance Officer	N/A	2025-10-21	N/A

Document Version: 1.0
Last Updated: 2025-10-21
Next Review: 2026-04-21