SecureControl

Banking Maker-Checker Controls

Proof of Concept

Project Documentation

Version 0.1.0 | February 2026

Repository: sanketbagad/maker-checker-poc

Branch: master

Next.js 16 | React 19 | Supabase | TypeScript | Tailwind CSS v4

Generated on February 25, 2026

Table of Contents

1. Executive Summary

SecureControl is a proof-of-concept banking controls platform that demonstrates a dual-authorization (maker-checker) workflow with automated policy analysis, blacklist management, and comprehensive audit trails. The application is designed for risk and compliance teams operating in regulated financial environments where every transaction requires independent verification before execution.

The platform enforces the four-eyes principle: Makers create transactions that must be independently reviewed and approved by Checkers. An automated policy engine scores each transaction in real-time, flagging potential compliance violations based on configurable rules. A SuperAdmin role provides system-wide oversight, user management, and comprehensive audit visibility.

The project also implements a complete KYC (Know Your Customer) onboarding pipeline with Indian identity validation (PAN, Aadhaar), multi-step form wizard, OTP email verification, and checker-based KYC approval gating. Users cannot access the transaction dashboard until their KYC is approved.

Key Capabilities

- Dual-control (maker-checker) transaction approval workflow
- Four user roles: Maker, Checker, Admin, SuperAdmin with strict RBAC
- Automated policy engine with 4 rule types and real-time risk scoring
- Full KYC onboarding flow (5 steps) with Indian identity validation
- OTP-based email verification during registration
- Blacklist management with automatic transaction flagging
- Comprehensive audit log explorer with full action history
- Role-based dashboards with real-time statistics
- Responsive, theme-aware UI (light/dark mode) built on Radix primitives
- Supabase Auth integration with Row Level Security (RLS) enforcement

2. Technology Stack

Frontend

Next.js 16: App Router with React Server Components and Server Actions

React 19: Latest React with concurrent features and use() hook

TypeScript 5: Full type safety across client and server code

Tailwind CSS v4: Utility-first CSS with design tokens and theme support

Radix UI: Accessible, unstyled UI primitives (30+ components)

SWR 2.3: React hooks for data fetching with stale-while-revalidate caching

Recharts 2.15: Composable charting library for dashboard statistics

Lucide React: Icon library integrated across UI components

Backend & Infrastructure

Supabase: Managed Postgres database with Auth, RLS, and real-time capabilities

Supabase Auth: Email/password authentication with session management via SSR cookies

Supabase Admin API: Service-role operations for superadmin user creation

Resend: Transactional email API for OTP delivery and credential notifications

Vercel Analytics: Lightweight usage metrics and web vitals tracking

Zod 3.25: Runtime schema validation for form inputs and API payloads

React Hook Form: Performant form state management with Zod resolver integration

Development Tooling

pnpm 9+: Fast, disk-efficient package manager

ESLint: Code quality and consistency enforcement

PostCSS: CSS transformation pipeline with Tailwind plugin

tsx: TypeScript execution for CLI scripts (e.g., seed-superadmin)

dotenv: Environment variable loading for scripts

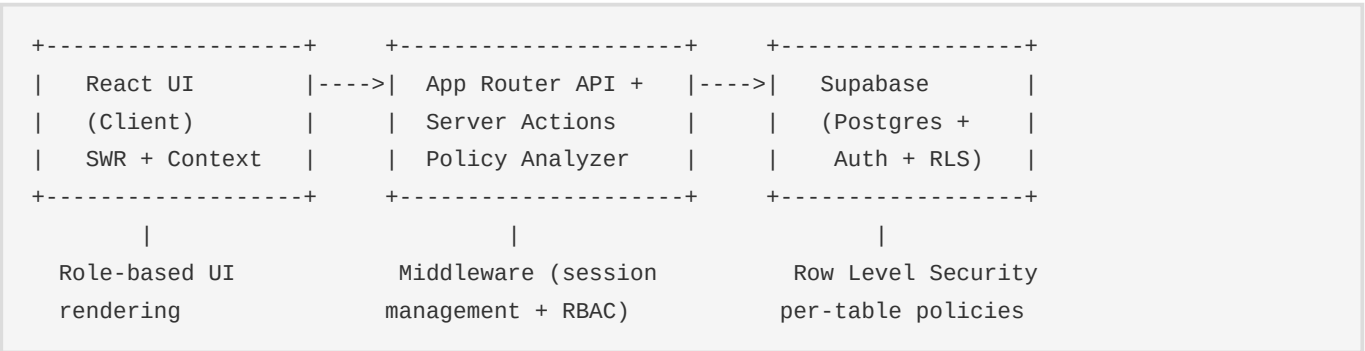
3. Project Architecture

Directory Structure

```
app/                  # App Router: routes, layouts, API handlers
  api/                # REST API endpoints (transactions, admin, kyc, auth)
  auth/               # Authentication pages (login, sign-up, onboarding, kyc-pending)
  dashboard/          # Role-based dashboards (maker, checker, admin, audit, policy)
components/           # Reusable UI and dashboard components
  ui/                 # 50+ shadcn-style atomic components (Radix-based)
  dashboard/          # Domain-specific dashboard modules
contexts/             # React contexts (dashboard state, user profile)
hooks/                # Custom hooks (SWR data, mobile detection, toast)
lib/                  # Core utilities
  supabase/           # Supabase client (browser + server + middleware)
  policy-analyzer.ts  # Compliance rule engine
  types.ts            # TypeScript type definitions
  constants.ts        # Application-wide constants
  email.ts            # Email utilities (Resend integration)
  otp-utils.ts        # OTP generation and cookie-based storage
scripts/              # SQL migrations and seed scripts
styles/               # Global Tailwind layer overrides
```

High-Level Architecture

The application follows a layered architecture separating presentation, business logic, and data access:



Request Flow

1. User hits a route -> Next.js middleware intercepts, calls updateSession() to refresh Supabase auth cookies.
2. Middleware enforces route protection: unauthenticated users redirected to login; authenticated users on login pages redirected to their dashboard; makers without KYC blocked from dashboard routes.
3. Server Components fetch data directly via Supabase server client. Client Components use SWR hooks to fetch from API routes.

4. Mutations go through Server Actions (auth) or API routes (transactions, KYC, admin), which validate permissions, execute business logic, and write audit logs.
5. Policy analyzer runs automatically on new transactions, checking compliance rules and flagging violations.

4. Authentication & Authorization

Authentication Flow

Authentication is handled by Supabase Auth with email/password credentials. Session management uses server-side cookies via @supabase/ssr, refreshed automatically through Next.js middleware on every request.

Registration (Sign-Up)

- Step 1: User fills in registration form (name, email, password)
- Step 2: System sends OTP verification email via Resend API (or displays in dev mode)
- Step 3: OTP stored in base64-encoded cookies with 5-minute expiry, max 3 attempts
- Step 4: On successful OTP verification, Supabase Auth account is created
- Step 5: Profile record auto-created via database trigger with role = 'maker'
- Step 6: User redirected to KYC onboarding wizard

Login

- User authenticates with email/password via Server Action
- System reads role from profiles table (not JWT metadata) for authoritative RBAC
- Role-based redirect: superadmin -> /dashboard/admin, checker/admin -> /dashboard/checker, maker (KYC complete) -> /dashboard/maker, maker (KYC incomplete) -> /auth/onboarding or /auth/kyc-pending

Role-Based Access Control (RBAC)

The system implements four distinct user roles with hierarchical permissions:

Role	Created By	Permissions
Maker	Self-registration	Create transactions, view own data, submit KYC
Checker	SuperAdmin	Review/approve/reject transactions & KYC applications
Admin	SuperAdmin	Same as Checker + policy management visibility
SuperAdmin	Seed script	Full system access, user creation, admin dashboard

RBAC is enforced at three layers: (1) Middleware - route-level access control and KYC gating, (2) API Routes - permission checks before data operations, (3) Database - Supabase Row Level Security policies restrict data visibility per role.

5. KYC Onboarding Flow

The platform includes a comprehensive Know Your Customer (KYC) onboarding pipeline that gates maker access to the transaction dashboard. Users must complete KYC verification before creating transactions.

5-Step KYC Wizard

Step 1: Personal Information

First/last name, date of birth, PAN number (validated as ABCDE1234F format), Aadhaar number (validated as 12 digits)

Step 2: Contact Details

Mobile number (10 digits), email address, current address, permanent address

Step 3: Employment & Account

Account type (savings/current/salary), occupation, annual income bracket, Politically Exposed Person (PEP) declaration

Step 4: Nominee Information

Optional step - nominee name, relationship, date of birth (can be skipped)

Step 5: Review & Submit

Summary of all entered information with edit capability per section before final submission

KYC Review Process

- Submitted applications enter 'pending' status
- Checkers/Admins can view all applications via /api/kyc/list with pagination and status filters
- Reviewer actions: approve, reject, or mark as under_review with notes
- On approval, profiles.kyc_completed flag is set to true via database trigger
- Rejected applications display reason; user can re-apply
- KYC-pending page polls status every 30 seconds and auto-redirects on approval

KYC Application ID Format

Each application receives an auto-generated ID: KYC-YYYY-XXXXXX (e.g., KYC-2026-000001), generated by a database sequence and trigger.

6. Dashboard Layer

The dashboard is a server-rendered layout with role-based content. A shared DashboardProvider context supplies user profile information and role flags to all child components.

Maker Dashboard

- Statistics grid: total transactions, pending, approved today, rejected today
- Recent transactions table with status badges and quick navigation
- 'New Transaction' button linking to transaction creation form
- Transaction types: Fund Transfer, Payment Approval, Account Change, Loan Approval

Checker Dashboard

- Statistics: pending count, flagged count, approved today (by this checker), rejected today
- Pending transactions list ordered oldest-first for FIFO review
- Inline approve/reject actions with required checker notes
- Policy violation details displayed alongside each pending transaction

SuperAdmin Dashboard

- Verified via superadmin role check from database (not JWT)
- User Overview: counts by role (maker, checker, admin, superadmin)
- Transaction Overview: pending, approved, rejected, flagged counts
- KYC & Audit: KYC pending count, total audit events today
- Recent Users list + Recent Audit Logs with links to detail pages
- User management: create checkers/admins, view all users

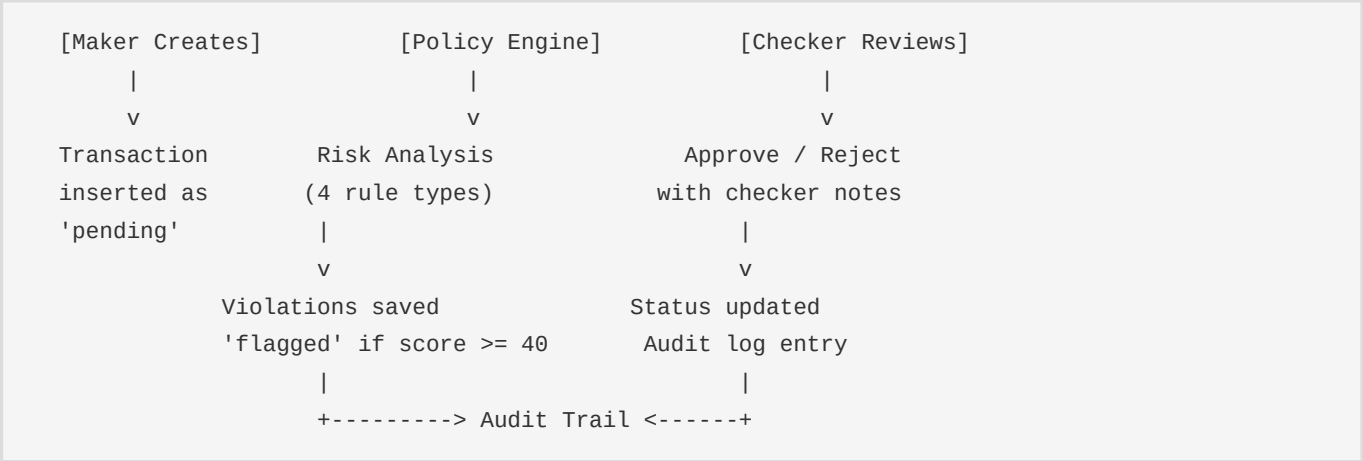
Common Dashboard Features

- Responsive sidebar navigation with role-specific menu items
- Header with user profile info, role badge, and sign-out
- Light/dark theme support via next-themes
- Loading skeletons and Suspense boundaries for progressive data display

7. Maker-Checker Workflow

The core business workflow follows the dual-control principle mandated by banking regulations. No single person can both create and approve a transaction.

Transaction Lifecycle



Transaction Types

Type	Description
Fund Transfer	Transfer of funds between accounts with source/destination tracking
Payment Approval	Authorization of scheduled or one-time payment requests
Account Change	Modifications to account parameters or settings
Loan Approval	Processing and authorization of loan disbursement requests

Transaction Statuses

Status	Description
Pending	Newly created transaction awaiting checker review
Approved	Transaction reviewed and approved by a checker
Rejected	Transaction reviewed and rejected by a checker (with notes)
Flagged	Automatically flagged by policy engine (risk score >= 40)

Key Constraints

- A maker CANNOT approve their own transactions
- All status changes are recorded in the audit log with timestamps and user IDs

- Checker must provide notes when approving or rejecting
- Flagged transactions require extra scrutiny - policy violations are displayed to checkers

8. Policy Engine & Compliance

The policy analyzer (lib/policy-analyzer.ts) is the automated compliance backbone. It runs against every new transaction, evaluating all active policy rules and generating a composite risk score.

Policy Rule Types

Rule Type	Check	Details
Amount Threshold	amount > threshold	Severity scales with ratio: >10x=critical, >5x=high, >2x=medium, else low
Duplicate Detection	Same amount + account in 24h	Flags potential duplicate transactions; >2 matches = high severity
Blacklist Check	Source/dest on blacklist	Critical severity if either account matches active blacklist entry
Time-Based	Outside business hours	Flags transactions created on weekends or outside 9am-6pm, Mon-Fri

Risk Scoring

Each violation contributes to a composite risk score (0-100). Transactions with a risk score of 40 or above are automatically flagged for enhanced review.

Severity	Score	Impact
Critical	40	Immediate attention - blacklist matches, extreme amounts
High	25	Senior management escalation recommended
Medium	15	Enhanced review required by checker
Low	5	Informational - time-based or minor threshold breach

Automated Recommendations

The engine generates contextual recommendations based on detected violations:

- 'Review transaction details carefully before approval' - on any violation
- 'Escalate to senior management for approval' - on critical/high severity violations
- 'Verify beneficiary identity before processing' - on blacklist matches

9. Blacklist Management

The blacklist subsystem maintains a registry of flagged account numbers. During policy analysis, both source and destination accounts are checked against active blacklist entries. Matches generate critical-severity violations.

Blacklist Entry Fields

- account_number - The flagged account identifier
- entity_name - Name of the associated entity (optional)
- reason - Reason for blacklisting (optional)
- is_active - Toggle to enable/disable without deletion
- created_by - User who created the entry (audit trail)
- created_at - Timestamp of creation

UI Management

The BlacklistManager component provides a UI for admins/checkers to view, add, and toggle blacklist entries. New entries are logged to the audit trail. The blacklist check runs as part of the automated policy analysis on every new transaction.

10. Audit Trail System

Every significant action in the system is logged to the audit_logs table, providing a complete chronological record for compliance review and forensic analysis.

Audited Actions

Action	Trigger
TRANSACTION_CREATED	Maker submits a new transaction
TRANSACTION_APPROVED	Checker approves a pending transaction
TRANSACTION_REJECTED	Checker rejects a pending transaction
BLACKLIST_ADDED	Admin adds account to blacklist
BLACKLIST_REMOVED	Admin deactivates a blacklist entry
POLICY_UPDATED	Admin modifies a policy rule
USER_LOGIN	User successfully authenticates
USER_CREATED	SuperAdmin creates a new user

USER_ROLE_UPDATED	SuperAdmin changes a user's role
USER_DEACTIVATED	SuperAdmin deactivates user account

Audit Log Fields

- user_id - The user who performed the action
- action - Type of action (from AUDIT_ACTIONS constants)
- entity_type - Type of entity affected (transaction, user, policy_rule, etc.)
- entity_id - ID of the affected entity
- old_values - JSON snapshot of previous state (for updates)
- new_values - JSON snapshot of new state (for creates/updates)
- ip_address - Client IP address (when available)
- created_at - Timestamp of the action

Audit Explorer UI

The /dashboard/audit route provides a paginated, searchable audit log table displaying the last 100 events by default. Each entry shows the user, action type, entity reference, and timestamp.

11. API Layer

The API is implemented as Next.js App Router route handlers under app/api/. All endpoints authenticate the caller, verify role permissions, and return structured JSON responses.

Transaction APIs

Method	Endpoint	Description
GET	/api/transactions	List transactions with pagination and filters
POST	/api/transactions	Create a new transaction (maker only)
POST	/api/transactions/analyze	Run policy analysis on a transaction

KYC APIs

Method	Endpoint	Description
POST	/api/kyc/submit	Submit KYC application (maker)
GET	/api/kyc/status	Get authenticated user's KYC status
POST	/api/kyc/review	Approve/reject KYC (checker/admin)
GET	/api/kyc/list	Paginated KYC application list (checker/admin)
POST	/api/kyc/update	Update KYC fields (non-identity) (maker)

Admin APIs

Method	Endpoint	Description
GET	/api/admin/users	List users with filters (superadmin)
POST	/api/admin/users	Create checker/admin user (superadmin)
GET	/api/admin/stats	Dashboard statistics (superadmin)

Auth APIs

Method	Endpoint	Description
POST	/api/auth/send-otp	Send OTP email for registration
POST	/api/auth/verify-otp	Verify OTP during registration

12. Database Schema

The database is hosted on Supabase (managed PostgreSQL). Schema is defined in SQL migration scripts under the scripts/ directory. All tables have Row Level Security (RLS) enabled.

Core Tables

Table	Purpose & Key Columns
profiles	User profiles: id, email, full_name, role, is_active, kyc_completed, timestamps
transactions	Transaction records: id, type, amount, currency, source/dest accounts, status, created_by, checked_by, metadata
policy_rules	Compliance rules: id, rule_name, rule_type, threshold_value, is_active, description
policy_violations	Detected violations: id, transaction_id, rule_id, violation_details, severity
blacklist	Flagged accounts: id, account_number, entity_name, reason, is_active
audit_logs	Action history: id, user_id, action, entity_type, entity_id, old/new values, ip_address
kyc_applications	KYC submissions: id, application_id, user_id, personal/contact/employment/nominee fields, kyc_status

Database Triggers

- handle_new_user: Auto-creates a profile record when a new user signs up via Supabase Auth
- update_updated_at: Automatically updates the updated_at timestamp on row modifications
- handle_kyc_approval: Syncs profiles.kyc_completed flag when KYC status changes to 'approved'
- Auto-generated KYC application_id via sequence (KYC-YYYY-XXXXXX format)

Row Level Security (RLS)

All tables have RLS enabled. Key policies include:

- Users can read their own profile; checkers/admins can read all profiles
- Makers can insert transactions; all authenticated users can read transactions
- Only checkers can update transaction status (approve/reject)
- Users can view their own KYC; checkers/admins can view and update all KYC applications
- Audit logs readable by all authenticated users; insertable by authenticated users

Seed Data

The schema initialization includes seed data for testing:

- 5 default policy rules (amount thresholds at various levels, duplicate detection, blacklist check, business hours)

- 3 sample blacklist entries for testing flagging behavior
- seed-superadmin.ts CLI script creates or promotes a superadmin user

13. Security Implementation

Authentication Security

- Server-side session management via encrypted HTTP-only cookies (@supabase/ssr)
- Session refresh on every request through Next.js middleware
- Role read from database (profiles table), NOT from JWT claims - prevents token manipulation
- OTP verification during registration with 5-minute expiry and 3-attempt limit
- Temporary passwords for admin-created users (uppercase + lowercase + digit + special character)

Authorization Security

- Three-layer RBAC: Middleware -> API Route -> Database RLS
- Middleware blocks unauthorized route access before Server Components execute
- API routes verify caller role via Supabase session before processing requests
- RLS policies enforce data isolation at the database level as a final safeguard
- SuperAdmin operations use Supabase service role key (server-side only, never exposed to client)

Data Protection

- All database access via Supabase client with RLS enforcement
- Sensitive operations (user creation, role changes) restricted to superadmin API routes
- KYC identity fields (PAN, Aadhaar) validated on both client and server
- Audit trail provides tamper-evident logging of all state changes
- Environment variables (Supabase keys, Resend API key) kept in .env.local, never committed

Input Validation

- Zod schemas validate form inputs and API request payloads
- Server-side validation in all API routes (never trust client-only validation)
- PAN format: ABCDE1234F (5 letters + 4 digits + 1 letter)
- Aadhaar format: 12 digits
- Mobile format: 10 digits
- Duplicate KYC application prevention at API level

14. UI Component Library

The project includes 50+ reusable UI components following the shadcn/ui pattern - Radix UI primitives wrapped with Tailwind CSS styling. All components support light/dark theme via CSS custom properties.

Atomic Components (components/ui/)

- Accordion, Alert, Alert Dialog, Aspect Ratio, Avatar, Badge
- Breadcrumb, Button, Button Group, Calendar, Card, Carousel
- Chart, Checkbox, Collapsible, Command, Context Menu, Dialog
- Drawer, Dropdown Menu, Empty State, Field, Form, Hover Card
- Input, Input Group, Input OTP, Item, Kbd, Label
- Menubar, Navigation Menu, Pagination, Popover, Progress
- Radio Group, Resizable Panels, Scroll Area, Select, Separator
- Sheet, Sidebar, Skeleton, Slider, Sonner (Toast), Spinner
- Switch, Table, Tabs, Textarea, Toast, Toggle, Toggle Group, Tooltip

Dashboard Components (components/dashboard/)

- Header - Top navigation bar with user info, role badge, sign-out
- Sidebar - Collapsible navigation with role-based menu items
- StatsCards / StatsGrid - Dashboard statistics display
- TransactionTable - Compound component with context-driven rendering
- TransactionFilters - Type, status, date range, and search filters
- PendingTransactionList - Checker-focused review queue
- PolicyRulesManager - Policy rule configuration display
- BlacklistManager - Blacklist entry management interface
- AuditLogTable - Paginated audit event explorer
- Skeleton loaders - Match dashboard layout for progressive loading

15. Deployment & Operations

Prerequisites

- Node.js 20+ and pnpm 9+ (or npm)
- Supabase project with email/password Auth provider enabled
- Resend account for transactional emails (optional for development)

Environment Variables

```
NEXT_PUBLIC_SUPABASE_URL=your-supabase-url
NEXT_PUBLIC_SUPABASE_ANON_KEY=your-anon-key
SUPABASE_SERVICE_ROLE_KEY=your-service-role-key
RESEND_API_KEY=your-resend-api-key
NEXT_PUBLIC_SITE_URL=http://localhost:3000
NEXT_PUBLIC_DEV_SUPABASE_REDIRECT_URL=http://localhost:3000/auth/callback
```

Setup Steps

- 1. Clone repository and install dependencies: pnpm install
- 2. Create Supabase project and enable email authentication
- 3. Apply database schema: Run scripts/001_create_schema.sql in Supabase SQL editor
- 4. Apply KYC schema: Run scripts/002_create_kyc_schema.sql
- 5. (Optional) Create superadmin: npx tsx scripts/seed-superadmin.ts
- 6. Configure .env.local with Supabase and Resend credentials
- 7. Start development server: pnpm dev

Deployment

The application is optimized for Vercel deployment. Configure the same environment variables in Vercel project settings. The build command is 'next build' and the framework preset is automatically detected.

Monitoring & Observability

- Vercel Analytics integrated for web vitals and usage metrics
- Supabase Dashboard provides database logs and auth event monitoring
- Audit logs table serves as an application-level activity ledger

16. Future Scope & Roadmap

Planned Features

- Cheque ingestion pipeline with OCR-backed image/PDF processing and automated authenticity scoring
- Multi-level approval workflows with configurable escalation chains and SLA tracking
- Automated notifications (email, Slack, Teams) on critical policy violations
- Risk trend dashboards with historical analytics and export (CSV/PDF) capabilities
- Maker/checker workload balancing based on domain expertise and availability
- Hardened RLS policies with service-role functions for scheduled background jobs

Architecture Evolution

Data Plane: Nightly ETL from core banking into warehouse (Snowflake/BigQuery) via Fivetran, dbt models for settlement metrics, CDC streaming to Supabase

Compute: Background workers (BullMQ/Temporal/Supabase Edge Functions) for policy re-scoring, OCR, ledger postings, SLA bulk approvals

Scheduling: Vercel Cron or Supabase Scheduled Functions for FX refresh, reconciliation, key rotation, stale draft cleanup

Eventing: Kafka/Redpanda topics for AML screening, alerting, and downstream analytics pipelines

Observability: Sentry + Datadog + Grafana for latency, queue depth, risk rule drift, and policy breach alerts with automated escalation

---

End of Document

SecureControl - Banking Maker-Checker Controls PoC

Generated: February 25, 2026