Access Controls Policy

Effective date: February 2026 **Owner:** Crumbs Money Engineering **Review cadence:** Annually, upon personnel changes, or after any security incident **Related documents:** Information Security Policy · Data Deletion & Retention Policy

1. Purpose

This policy defines the access control measures Crumbs Money uses to limit access to production assets, sensitive data, and infrastructure systems. It ensures that only authorized individuals and services can access resources, and that access is granted on a least-privilege, need-to-know basis.

2. Scope

This policy covers access to:

- Production application servers and deployment pipelines
- Production databases
- Third-party service dashboards (Plaid, Firebase, Railway)
- Source code repositories
- API keys, secrets, and credentials
- End-user financial data

3. Principles

Principle	Description
Least privilege	Every person and service receives the minimum access necessary to perform their function.
Role-based access	Access is assigned based on role, not on an individual ad-hoc basis.
Separation of duties	No single individual has unchecked access to all systems.
Verify, don't trust	Identity is verified on every request. No implicit trust based on network location or prior authentication.
Auditability	Access grants and changes are documented and reviewable.

4. Human access controls

4.1 Production infrastructure (Railway)

Control	Detail
Authentication	Individual accounts with email + password. MFA recommended for all team members.
Authorization	Railway supports team roles (Admin, Member). Only Admin can modify environment variables, database access, and deployment settings. Members have read-only or limited access.
Secrets	Environment variables (Plaid keys, database URL, Firebase credentials) are accessible only through the Railway dashboard — never exposed in logs, source code, or API responses.
Audit	Railway logs all deployments, configuration changes, and team membership changes.

4.2 Database (PostgreSQL on Railway)

Control	Detail
Network access	Database connections route through Railway's proxy. No direct public IP access.
Credentials	Single connection string (username + password) stored as a Railway environment variable. Not committed to source code.
Access scope	Only the backend application connects to the database. No team members connect directly in normal operations.
Query safety	All application queries use parameterized statements to prevent SQL injection.

4.3 Plaid Dashboard

Control	Detail
Authentication	Individual team accounts with email + password. MFA recommended.
Authorization	Plaid Dashboard supports team roles (Admin, Developer). Only Admins can rotate API keys or change product permissions.
API keys	`PLAID_CLIENT_ID` and `PLAID_SECRET` are stored exclusively in Railway environment variables. Never committed to source code.
Audit	Plaid Dashboard logs API key usage, team membership changes, and key rotations.

4.4 Firebase Console (Google Cloud)

Control	Detail
Authentication	Google accounts with Google's built-in MFA support.
Authorization	Firebase uses Google Cloud IAM roles (Owner, Editor, Viewer). Team members are assigned the minimum role needed.
Service account	The Firebase service account key (JSON) is stored on the production server only, excluded from source control via `.gitignore`. Used server-side by Firebase Admin SDK to verify user ID tokens.

Control	Detail
Audit	Google Cloud provides audit logs for all IAM changes, console access, and API usage via Cloud Audit Logs.

4.5 Source code repository

Control	Detail
Authentication	Individual accounts on the Git hosting provider (e.g., GitHub). MFA recommended.
Authorization	Repository access controlled via collaborator permissions (Admin, Write, Read).
Secrets protection	`.gitignore` excludes all secret files (`server/.env` , `firebase-service-account.json`). Pre-commit checks recommended to prevent accidental commits of sensitive files.

---

5. Non-human (service-to-service) access controls

5.1 Backend → Plaid API

Control	Detail
Authentication method	API key headers (`PLAID-CLIENT-ID` , `PLAID-SECRET`) plus per-user OAuth-style `access_token` .
Transport	All calls over HTTPS/TLS.
Scope	Access tokens are scoped to specific Plaid products (Transactions, Investments) and are read-only — no payment initiation or fund transfers.
Token lifecycle	Access tokens are created when a user links an institution and revoked (via Plaid `/item/remove`) when the user disconnects.

5.2 Backend → Firebase Admin SDK

Control	Detail
Authentication method	Service account JSON key file.
Transport	All calls over HTTPS/TLS.
Scope	Used exclusively to verify user ID tokens. No write operations to Firebase.
Key storage	Stored on the production server only, path specified via environment variable. Excluded from source control.

5.3 Backend → PostgreSQL

Control	Detail
Authentication method	Connection string with username and password.
Transport	TLS-encrypted connections through Railway's proxy.
Scope	Application queries are scoped by authenticated user ID (`req.uid`). No cross-user queries exist in the codebase.

5.4 Frontend → Backend API

Control	Detail
Authentication method	Firebase ID token sent as `Authorization: Bearer <token>` header on every request.
Verification	Backend middleware verifies the token with Firebase Admin SDK on every request. Invalid or expired tokens receive a 401 response.
CORS	Backend only accepts requests from the configured frontend origin.

---

6. End-user data access controls

Control	Detail
Identity verification	User identity is derived exclusively from the verified Firebase ID token — never from URL parameters, request body, or query strings.
Data isolation	Every database query and Plaid API call is scoped by the authenticated user's Firebase UID. No endpoint returns another user's data.
No admin endpoints	There are no admin, superuser, or cross-user data access paths in the application.
Read-only access	The application only reads financial data. No endpoints initiate transactions, transfers, or modifications to the user's financial accounts.

---

7. Access provisioning & de-provisioning

7.1 Granting access

1. New team members are granted access only to the systems required for their role.
2. Access is provisioned through each platform's native team/IAM management (Railway, Firebase/GCP, Plaid Dashboard, GitHub).
3. Access grants are documented (who, what system, what role, when).

7.2 Revoking access

1. When a team member leaves or changes role, their access is revoked from all production systems within **24 hours**.
2. Revocation checklist:
   - ☐ Remove from Railway team
   - ☐ Remove from Firebase/Google Cloud IAM
   - ☐ Remove from Plaid Dashboard team
   - ☐ Remove from source code repository
   - ☐ Rotate any shared credentials the individual had access to (database password, Plaid API keys) if warranted
3. Revocation is documented (who, what systems, when).

7.3 Periodic review

- Access rights across all production systems are reviewed **quarterly**.
- Reviews confirm that each team member's access level matches their current role and that no orphaned accounts exist.
- Review findings and any resulting changes are documented.

---

8. Credential & secret management

Practice	Detail
Storage	All secrets stored in Railway environment variables or secure server files — never in source code, logs, or client-side storage.
Source control	`.gitignore` blocks `server/.env`, `firebase-service-account.json`, and `**/firebase-service-account*.json`.
Rotation	Credentials are rotated immediately upon suspected compromise. Routine rotation is performed annually or when team membership changes.
Transmission	Secrets are never transmitted via email, chat, or other unencrypted channels. Shared only through each platform's secure dashboard.

---

9. Monitoring & audit

System	Audit capability
Railway	Deployment logs, environment variable change history, team membership changes.
Firebase / Google Cloud	Cloud Audit Logs for IAM changes, authentication events, Admin SDK usage.
Plaid Dashboard	API key rotation history, team membership changes, API call logs.
Application	Server logs capture authentication failures (401s) and Plaid API errors. No sensitive data (tokens, balances) is logged.

10. Policy violations

Violations of this policy — including unauthorized access attempts, credential sharing, or failure to revoke access — are treated as security incidents and handled per the Incident Response procedure.

11. Review & updates

- This policy is reviewed annually, upon any personnel change, and after any security incident.
- Changes are approved by the engineering lead and documented in the changelog below.

Changelog

Date	Change
February 2026	Initial version.

Last updated: February 2026