APPS BY VIKAS

Meet Vikas



GenzLudo

Ludo with 3D characters and up to 8 players. A first-of-its-kind no-teaming-up feature keeps play fair. Try it - you'll love the difference.

Download on Google Play



Kachu

Too much scrolling straining your neck? Add your addictive apps and complete daily neck exercises to unlock them.

Download on Google Play

A PRACTICAL 10-MINUTE GUIDE

Secure 2FA / MFA that attackers cannot easily bypass

Simple language. Real attack examples. Production-ready checks.

FIRST UNDERSTAND THE REAL PROBLEM

2FA means the user proves identity with two different things: something they know (password) and something they have (phone, authenticator app, passkey or security key).

ATTACK EXAMPLE

An attacker has the user's password but not the 6-digit code. They click Forgot 2FA. If one email link disables 2FA, the attacker never needs the code. The recovery path has silently defeated MFA.

THE GOLDEN RULE

Every path that can add, remove, reset or replace a factor must be at least as secure as normal login.

WEAK DESIGN

Password → TOTP → login

But: email link → disable TOTP

SECURE DESIGN

Strong proof → risk check → delay if needed → reset → revoke sessions → notify

How to implement TOTP correctly

TOTP is a time-based 6-digit code. The app and your server both calculate it from the same secret seed.

IMPORTANT MENTAL MODEL

The QR code contains the secret seed. Anyone who copies that seed can generate valid codes forever. Treat it like a long-lived password - not like a temporary QR image.

ENROLLMENT: WHEN THE USER TURNS TOTP ON

1 Re-authenticate

Ask for the password or an existing strong factor. A stolen logged-in session alone must not be able to add MFA.

2 Create and protect the seed

Generate at least 160 random bits using a CSPRNG. Encrypt the seed with authenticated encryption; keep the key in KMS/HSM, separate from the database.

3 Show the QR once

Build an otpauth URI with issuer + account name. Never send the URI or seed to logs, analytics, crash reports, support tools or third parties.

4 Verify before activation

Ask for one fresh code. Only after it verifies should the factor become active. Then create backup codes and notify the owner.

LOGIN VERIFICATION: WHAT THE SERVER SHOULD DO

- 1. Load the encrypted seed → decrypt only inside the auth service.
- 2. Check codes only for a small window: current 30s step, usually ±1 step for clock skew.
- 3. Compare safely using a mature TOTP library. Never write your own crypto.
- 4. Atomically store the accepted time-step. Reject that same step if it is submitted again.
- 5. On success, complete only the login/action that created this challenge; expire the challenge.

RATE LIMITING

A 6-digit code has only 1,000,000 possibilities. Limit attempts per account, IP/network and device signal. Increase delay after failures, but avoid a permanent lockout attackers can abuse.

REPLAY PREVENTION

Two requests can submit the same valid code at the same time. Use an atomic database compare-and-set so only one request can mark that time-step as used.

The part most teams accidentally weaken

Recovery is not a customer-support convenience. It is another high-risk authentication protocol.

WORKED ATTACK EXAMPLE

Attacker knows the password and controls the mailbox. Normal login stops them at TOTP. They choose Forgot 2FA, use the email link, attach their own authenticator and enter the account. Result: 2FA existed, but email recovery was the real master key.

A SAFE RECOVERY FLOW

1 Use remaining proof

Password plus backup code, an existing passkey/security key, trusted factor, or carefully verified identity evidence. Email alone is weak proof.

2 Evaluate risk

Check new device, unusual location, velocity, recent password/email/phone changes and known-compromised signals. High risk should add delay, review or denial.

3 Reset with containment

Invalidate the old factor and backup codes. Revoke all sessions (or at minimum every other session) and cancel pending challenges.

4 Notify and preserve evidence

Alert every verified channel with time/device details and a safe 'not me' route. Log who approved the reset, evidence used, risk signals and outcome.

BACKUP CODES

- Generate random high-entropy codes with a CSPRNG.
- Show once and let the user download/print safely.
- Store only a slow password hash or keyed hash.
- Each code works once. Regeneration invalidates the full old set.

SUPPORT / MANUAL RECOVERY

- Support must never casually switch MFA off.
- Use a written policy: acceptable evidence, approvals, cooldowns and escalation.
- Separate requester, reviewer and approver for valuable/admin accounts.
- Audit support access and every recovery decision.

ADDING, REMOVING OR REPLACING A FACTOR

Require recent strong re-authentication → verify the new factor → invalidate the old factor and recovery artifacts → revoke sessions → notify the owner. A stolen session cookie must not be enough. For admins or money movement, prefer a security key/passkey and consider a second approver.

Which method should you use?

Prefer phishing-resistant factors. Keep weaker methods only where users genuinely need them.

METHOD	SECURITY	GOOD FOR	MAIN RISK
Passkey / security key	Phishing-resistant	Admins, payments, best default	Device loss; plan multiple keys/devices
TOTP app	Strong baseline	Most apps and developer tools	Can be phished; seed theft
SMS OTP	Fallback only	Users without authenticator apps	SIM swap, number recycling, delivery
Email OTP	Not independent	Low-risk step-up	Mailbox often controls recovery too

FINAL PRODUCTION CHECKLIST

- ☑ TOTP seed created with a CSPRNG and encrypted at rest; encryption key stored separately.
- ☑ Enrollment requires re-authentication and one successful code before activation.
- ☑ Verification has per-account + network/device rate limits and small clock-skew window.
- ☑ Accepted TOTP time-step is stored atomically to prevent replay.
- ☑ Backup codes are random, hashed/keyed-hashed, single-use and replaceable as one set.
- ☑ Secrets, QR URIs, OTPs and recovery evidence never enter logs, analytics or support tools.
- ☑ Recovery never disables MFA from an email link alone; risk checks and cooldowns exist.
- ☑ Factor add/remove/reset requires recent strong re-authentication and verifies the new factor.
- ☑ Factor reset/change revokes sessions, pending challenges and old recovery artifacts.
- ☑ Owner receives useful notifications with a safe 'not me' action.
- ☑ Support recovery has written evidence, approval and audit requirements.
- ☑ Trusted-device tokens are random, server-revocable and expiring; ignored for sensitive actions.
- ☑ Audit logs capture actor, factor, action, time, device/risk signals and result; alerts monitor abuse.

REMEMBER

Your MFA is only as strong as the easiest path that can reset, remove or replace it.

More things worth exploring

Find more free guides at vikasbuilds.com/resources