TECHNICAL DESIGN DOCUMENT TEMPLATE
SwitchUp – Fitness application

MY BENEFIT
One-sentence pitch:
A smart, user-friendly fitness tracking application that helps users monitor workouts, track progress, and build consistent healthy habits using real-time data stored securely in Firebase.
Fitness-tracker-app

1. OVERVIEW

Goal

- Help users track daily workouts and fitness progress in a simple and structured way.
- Provide data-driven insights to improve consistency and performance.
- Encourage healthy habits through measurable goals and progress visualization.

Key features:

- User authentication (Sign up / Login / Logout)
- Create, update, and delete workout entries
- Track workout type, duration, calories burned, and date
- View workout history
- Dashboard with progress overview
- Secure Firestore data storage
- Responsive UI (mobile & desktop friendly)

* Target users & success criteria: \[who benefits and how success is measured]

Target Users

- 👨‍🎓 Students who want a structured way to track workouts.
- 🏋️‍♂️ Gym beginners who need guidance from predefined activity categories.
- 💪 Fitness enthusiasts who want consistent and organized performance tracking.
- 👨‍💼 Busy professionals who prefer quick activity selection instead of manual typing.

Success Criteria

The application will be considered successful if:

- ✅ Users consistently log workouts using the predefined activity list.
- ✅ Workout data remains structured and free from inconsistent naming.
- ✅ Users can clearly track progress over time (frequency, duration, calories).
- ✅ Increased weekly engagement (repeat logins and activity entries).
- ✅ Firestore queries remain efficient due to normalized data structure.
- ✅ Security rules correctly restrict users to only accessing their own workout records.

2. TECH STACK (GOLDEN PATH)

Runtime: Node (Firebase Gen 2 Cloud Functions)
Language: TypeScript (strict)
Front-end: React + Vite
UI kit: shadcn/ui (Radix + Tailwind source-copy model)
Styling: Tailwind CSS (design-token file)
State / data fetching: TanStack Query
Forms & validation: React Hook Form + Zod resolver
Shared validation: Zod (client & server)
Backend services: Firebase Auth · Firestore
Package manager / mono: PNPM workspaces
Build orchestration: Turborepo (remote caching)
Component workshop: Storybook (UI in isolation)
Unit / component tests: Vitest + Testing Library
Visual / interaction: Storybook + @storybook/testing-library
End-to-end tests: Playwright
Linting: ESLint (typescript-eslint) + eslint-plugin-perfectionist
Formatting: Prettier
Type-safe env vars: T3 Env (Zod-validated)
Versioning / publishing: Changesets (monorepo changelogs & releases)
CI / CD: GitHub Actions (Turbo-aware pipeline; see §8)

---

3. MONOREPO LAYOUT (PNPM)

```
.
├── apps/
│   └── web/          ← React front-end (+ .storybook)
├── packages/
│   ├── shared/       ← Zod schemas, utilities, common types
│   └── seeding/      ← Data-seeding helpers (Firestore emulator/Admin SDK)
├── docs/             ← Project docs (this TDD, ADRs, API notes)
└── .github/          ← CI workflows
```

---

4. ARCHITECTURE

Client (React + TanStack Query) ⇄ tRPC HTTPS endpoints (Cloud Functions)
tRPC handlers read/write Firestore documents and interact with Storage.

<!-- Replace or link to a diagram if useful. -->

---

5. DATA MODEL

Entity	Key fields	Notes
User	uid, email, role, …	Auth via Firebase
\[…]	…	…

* Security rules: \[plan or link]
* Index strategy: \[composite indexes]

---

6. API DESIGN (tRPC)

Router	Procedure	Input (Zod schema)	Output
user	getById	uid	User
\[…]	…	…	…

Error-handling conventions: \[auth errors, validation errors, etc.]

---

7. TESTING STRATEGY

Level / focus	Toolset	Scope
Unit	Vitest	Pure functions, hooks
Component	Vitest + Testing Library	React components
Visual / interaction	Storybook + @storybook/testing-library	UI snapshots, interactions
End-to-end	Playwright	Auth flows, happy paths

* Coverage target: \[e.g., 80 % statements]
* Fixtures / seeding: `pnpm seed` → runs scripts in `packages/seeding` against the Firebase emulator.

---

8. CI / CD PIPELINE (GITHUB ACTIONS)

9. Setup PNPM and restore Turbo remote cache

10. `pnpm exec turbo run lint typecheck` – ESLint & `tsc --noEmit`

11. `pnpm exec turbo run test` – Vitest (Turbo skips untouched packages)

12. `pnpm exec turbo run build-storybook` – generates static Storybook

13. `pnpm exec turbo run e2e` – Playwright suite (headless)

14. Deploy preview (Firebase Hosting channel + optional Storybook host)

15. Changesets release & promote to prod on merge to `main`

---

9. ENVIRONMENTS & SECRETS

Env	URL / target	Notes
local	localhost:5173	.env + Firebase emulators; validated by T3 Env
preview-*	Firebase Hosting channel	Auto-created per PR
prod	https://app.example.com	Promote via CI workflow

Secrets handled with `firebase functions:config:set` and GitHub repo secrets.

---

10. PERFORMANCE & SCALABILITY

* Denormalize Firestore data to avoid hot-document writes.
* Tune TanStack Query caching (`staleTime`, prefetch patterns).
* Code-split via Vite dynamic imports.

---

11. MONITORING & LOGGING

Concern	Tool	Notes
Runtime errors	Firebase Crashlytics / Sentry	Front-end error capture
Server logs	Google Cloud Logging	Structured JSON logs
Analytics	GA4 or PostHog	Track funnels & usage

---

12. ACCESSIBILITY & I18N

* shadcn/ui components use Radix primitives (focus, ARIA).
* Storybook a11y addon for quick audits.
* WCAG 2.1 AA checklist (contrast, keyboard nav).
* i18n plan: \[react-intl, language switcher, etc.]

---

13. CODE QUALITY & FORMATTING

* Prettier formats on save / commit.
* ESLint governs rules; perfectionist plug-in auto-sorts imports and object keys.
* Husky pre-commit hook runs `lint-staged`.

---

14. OPEN QUESTIONS / RISKS

Item	Owner	Resolution date
\[e.g., Payment gateway]	—	—
\[…]		

---

15. APPENDICES

* Setup script: `pnpm exec turbo run setup`
* Branching model: Conventional commits + Changesets for versioning.
* Links: product spec, Figma, Storybook URL, ADR index, etc.

Last updated: \[YYYY-MM-DD]

End of template – copy into `docs/technical-design-doc.md` and customize.