Typing Quest (Working Title)

Product Requirements Document (PRD)

Version 1.0 • 2026-01-25

Purpose: Build a fun, game-based typing application for 11-12 year olds that teaches keyboard familiarity, correct hand placement, and measurable speed/accuracy improvement over 2-3 weeks.

Target Users	Kids age 11-12 (beginner to intermediate typists)
Practice Plan	15 minutes/day, 5-6 days/week, for 2-3 weeks
Success Targets	20-30 WPM at 95% accuracy; correct home-row hand placement
Platform	Web app (mobile-friendly; best with physical keyboard)
Build Environment	Replit (full-stack)

1. Problem Statement & Goals

Many kids can "hunt-and-peck" but lack keyboard familiarity, consistent finger placement, and a motivating feedback loop. Typing Quest turns daily practice into short games with visible progress and rewards.

Primary learning goals (must hit):

- Students recognize and confidently use all keys (letters, numbers, punctuation, Shift, Backspace, Enter, Space).
- After 2-3 weeks at ~15 minutes/day: 20-30 WPM with 95% accuracy on age-appropriate passages.
- Correct hand placement and finger use (home row and common reach keys).
- Fun, interactive games to reduce boredom and improve consistency.
- Badges/awards to demonstrate progress and encourage streaks.

Non-goals (out of scope for v1):

- Full LMS integration (Google Classroom / Clever) in v1.
- Real-time multiplayer competition (can be v2).
- Native mobile apps - web-only initially.

2. Users, Personas, and Key Use Cases

Primary user: Kid (11-12) practicing daily.

Secondary users: Parent/guardian (progress visibility), teacher/club leader (optional cohort view).

Contexts: Home laptop, Chromebook, desktop; occasional tablet (external keyboard recommended).

User stories

- As a kid, I want quick games that start instantly so practice feels like play.
- As a kid, I want to earn badges and unlock new levels/skins so I stay motivated.
- As a parent, I want to see weekly progress (WPM, accuracy, streaks) without hovering.
- As a teacher/leader, I want to set a practice plan and see who is improving.

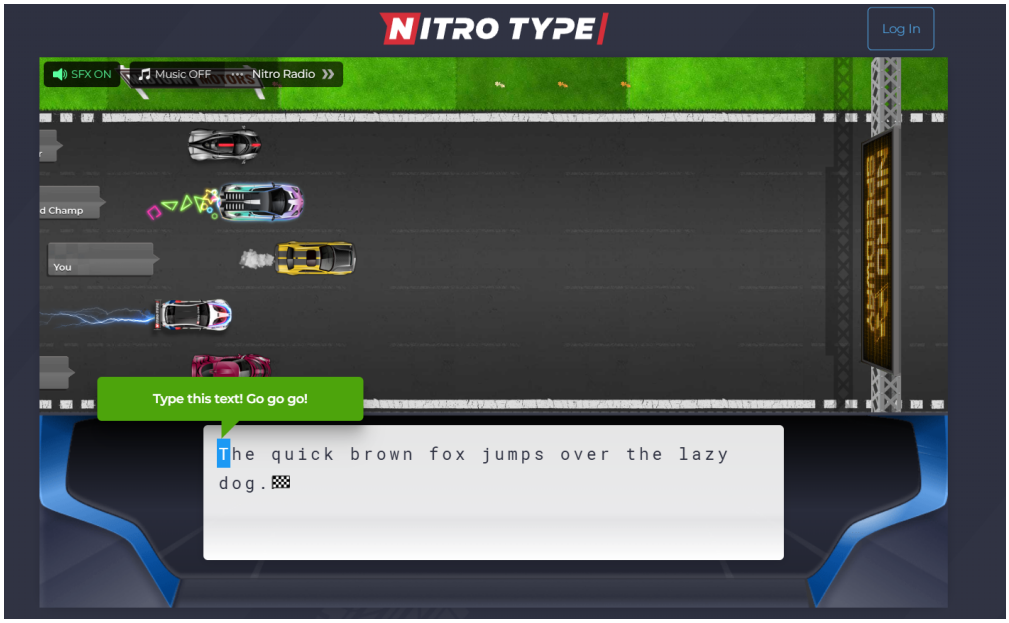
Accessibility & safety

- Kid-safe design: no open chat, no public profiles by default, minimal personal data.
- Readable typography, high contrast, and optional reduced motion mode.
- Audio optional (SFX/music toggles) with default volume low.

3. Game Concepts (with examples)

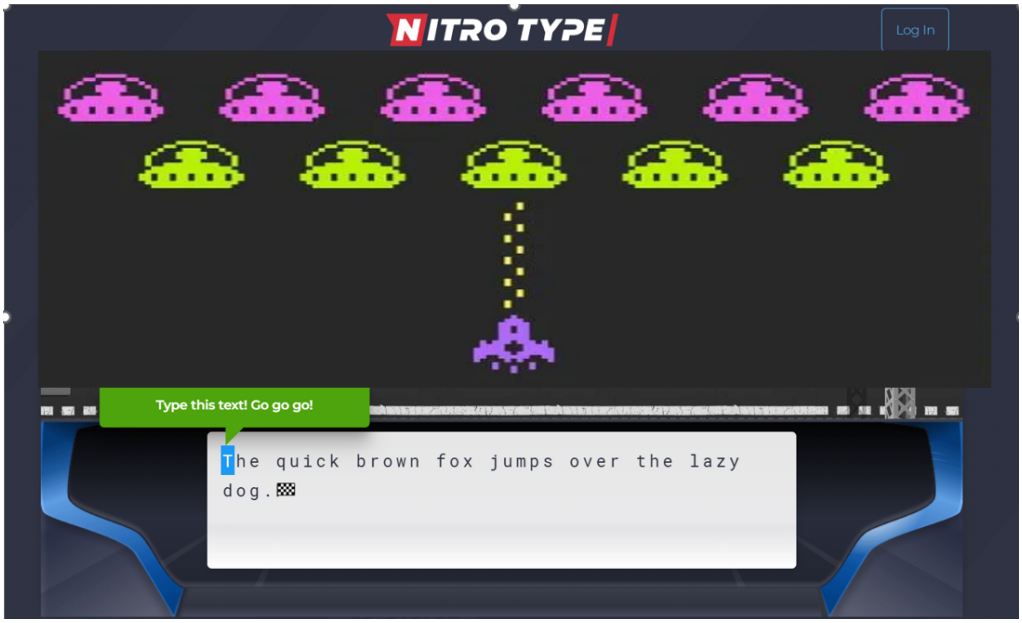
Games are short (1-3 minutes) and map directly to a typing skill. Each session records WPM, accuracy, and keys practiced.

Example Game A: NitroType-style racing



Reference screenshot provided by user: race advances as the student types the prompt.

Example Game B: Space Invaders typing shooter



Reference screenshot provided by user: correct typing fires shots; misses reduce shields/score.

Recommended v1 game modes

- **Race Sprint**: type a short sentence; your car/character moves with each correct character. Focus: rhythm + accuracy.
- **Alien Defense**: enemies show a letter/short word; type to zap them. Focus: key recognition & reaction.
- **Home Row Builder**: guided drills (ASDF JKL;) with hand placement overlay. Focus: technique.
- **Word Dash**: themed word lists with streak multipliers. Focus: common patterns.
- **Boss Level**: 60-second checkpoint at end of each week with a badge reward.

4. Learning Design & Progression (2-3 Weeks)

The app guides students through a structured progression, while still letting them replay favorite games.

Suggested progression

Week	Focus	Daily 15-min Plan (default)
Week 1	Home row + accuracy	5 min guided drills + 8 min game + 2 min recap
Week 2	Reach keys + punctuation	4 min drills + 9 min game + 2 min recap
Week 3 (optional)	Speed + mixed passages	3 min drills + 10 min game + 2 min boss test

Instructional supports

- **Hand placement coach**: optional on-screen keyboard + finger map; prompts when the wrong finger is likely used (heuristic).
- **Error-friendly input**: highlight mistakes immediately; allow backspace; show common error patterns.
- **Recap screen**: WPM, accuracy, keys missed, and one coaching tip.

5. Functional Requirements

This section lists what v1 must do. Requirements are grouped by student experience, progress/rewards, and admin/parent views.

5.1 Student experience

- **Account-less quick start**: "Play as Guest" launches in under 5 seconds.
- **Profiles**: optional kid profiles (nickname + avatar) stored locally or via simple login (emailless code) for parents/teachers.
- **Practice session**: daily 15-minute guided flow with clear start/stop and a recap screen.
- **Typing engine**: real-time character highlighting, error detection, WPM & accuracy calculation, backspace behavior.
- **Game selection**: 3-5 games available; recommend next based on weak keys/skills.
- **Difficulty scaling**: adapt prompt length and target speed to keep challenge in the "fun zone".

5.2 Progress, badges, and rewards

- **Badges**: earnable awards (e.g., "Home Row Hero", "95% Accuracy", "7-Day Streak").
- **Levels**: XP for completing sessions; level gates unlock new themes/ships/cars.
- **Streaks**: track daily practice streak; gentle recovery mechanics (1 "streak save" per week).
- **Goal cards**: simple targets shown each session (e.g., "Hit 22 WPM" or "Practice ; key").

5.3 Parent/teacher view (minimal v1)

- **Dashboard**: weekly chart/table for WPM, accuracy, minutes practiced, streak.
- **Badges**: view earned badges and upcoming badge goals.
- **Export/share**: download a simple progress report (PDF or image).

6. Frontend Requirements (Replit)

Recommended stack: React + Vite + TypeScript (works well on Replit and matches the game-like UI needs).

6.1 Core UI screens

- Landing / Start: Guest play, profile selection, settings (sound, motion).
- Daily Session: drill → game → recap flow with progress bar.
- Game Screen: full-screen canvas/DOM hybrid; shows prompt, typed text, score, timer, pause.
- Badges & Rewards: badge grid, level/XP bar, unlockables.
- Parent/Teacher Dashboard: password/guardian gate, weekly metrics, export.

6.2 Typing engine (frontend)

- Capture keyboard events reliably across browsers; ignore non-character keys unless used in lessons (Shift, Backspace).
- Render prompt with per-character state: pending / correct / incorrect / current cursor.
- Compute real-time metrics: WPM (standard 5 chars/word), raw WPM, accuracy %, error count, backspaces.
- Support different input modes: sentence prompts, word targets, single-key targets.
- Accessibility: focus management, ARIA labels for buttons, keyboard-only navigation outside games.

6.3 Game rendering approach

- v1 can be DOM/CSS sprites (simpler) or HTML5 canvas. Start with DOM sprites + simple animations; switch to canvas if needed.
- Frame loop: requestAnimationFrame for movement; keep logic deterministic and testable.
- Responsive layout: game area scales to viewport; keep text legible; support 1366x768 baseline.

7. Backend Requirements (Replit)

Backend is primarily for saving progress, profiles, and generating reports. Keep it simple and reliable.

Recommended stack

Node.js + Express (TypeScript) with a lightweight DB (SQLite for simplest, or Replit Postgres if you want hosted persistence).

7.1 Data storage

- **Profiles**: id, nickname, avatar/theme, createdAt.
- **Sessions**: profileId, startedAt, durationSec, gameMode, promptId, wpm, accuracy, errors, keysMissed (optional).
- **Badges**: badgeId, profileId, earnedAt.
- **Streaks**: lastPracticeDate, currentStreak, longestStreak.

7.2 API endpoints (v1)

- POST /api/profiles - create profile
- GET /api/profiles/:id - fetch profile + summary stats
- POST /api/sessions - submit session results
- GET /api/profiles/:id/sessions?range=7d|30d - list sessions
- GET /api/profiles/:id/badges - list badges
- POST /api/profiles/:id/export - generate progress report (PDF) (optional v1.1)

7.3 Security & privacy

- Default to minimal PII: nickname only; avoid collecting kid emails in v1.
- Parent gate for dashboard (simple passcode set on device) if accounts are local-only.
- Rate-limit write endpoints and validate payloads (zod/joi) to avoid spam.

8. Success Metrics & Instrumentation

We need to know if kids are improving and if the app keeps them practicing.

Core metrics

- % of students reaching 20 WPM @ 95% accuracy by day 14 and day 21.
- Median WPM and accuracy change per week.
- Practice adherence: sessions/week, average session length, streak distribution.
- Game engagement: most played modes, rage-quit rate (quit < 30 seconds).
- Key mastery: error heatmap by key (letters and punctuation).

Event logging (lightweight)

Store events in your DB with a rolling retention window (e.g., 90 days) for simple reporting.

9. Milestones & Build Plan (Replit)

Milestones are designed for a small team building on Replit. Each milestone should end with a demo.

Milestone	Deliverable
M0 - Setup (Day 1-2)	Repo, Vite+React UI shell, Express API, DB connection; basic deployment on Replit.
M1 - Typing Engine (Week 1)	Prompt renderer, key capture, correctness states, WPM/accuracy; 1 drill mode; local saving.
M2 - Game 1: Race Sprint (Week 1)	Race goals, progress tied to correct characters; recap screen; session recording API.
M3 - Game 2: Alien Defense (Week 2)	Enemy waves, scoring, difficulty scaling; sound toggle; performance tuning.
M4 - Progress & Badges (Week 2)	Badge rules engine, XP/levels, streak logic; badge screen; parent dashboard MVP.
M5 - Content & Curriculum (Week 3)	Week-by-week progression, prompt library, boss tests; key mastery heatmap.
M6 - Polish & Launch (Week 3)	Bug fixes, accessibility pass, kid-safe defaults, basic report export, README/how-to.

Definition of Done for v1

- At least 3 game modes and 1 boss test.
- Accurate WPM/accuracy tracking and recap after every session.
- Badges/levels + streaks working end-to-end.
- Parent dashboard shows last 7/30 days progress.
- Runs smoothly on typical school/home laptops (Chromebook class) and modern browsers.

10. Risks, Open Questions, and Tradeoffs

Key risks to manage early:

- **Typing accuracy measurement**: decide how strict mistakes are (allow backspace vs. force restart).
- **Finger tracking**: browsers cannot truly detect which finger is used; coaching must be heuristic and optional.
- **Kid privacy**: avoid collecting unnecessary data; keep everything local unless a parent enables accounts.
- **Game performance**: keep animations light for Chromebooks; avoid heavy canvas effects in v1.

Open questions (decide before M2)

- Will profiles be **local-only** (simpler) or **cloud-synced** (better for schools)?
- Do we want a **single-player** focus only for v1, or add ghost racing against your past best?
- Which keyboard layouts must be supported (US QWERTY only in v1)?