Tech Personality Quiz Web App Spec

1) Product Summary

A 16Personalities-style quiz that outputs a user's **Tech Personality** based on **5 scored spectrums**, then recommends:

- **Top-fit tech roles** (from your 16)
- **Skill roadmaps**
- **Resources/courses** (including "at your school" mappings)
- Optional: shareable results card + tracking over time

Users answer a questionnaire (1–5 Likert). The app computes **percentages** for each spectrum and generates:

- A **4-letter Tech Type** from the first four scales (16 total types)
- A suffix from the 5th scale, like **-A / -T** (Adaptive vs Structured)

---

2) Core Model: The 5 Scales

Scale 1 — Focus

Builder (B) ↔ Analyzer (A)

Scale 2 — Interface

User-Facing (U) ↔ Systems-Facing (S)

Scale 3 — Change Style

Exploratory (E) ↔ Operational (O)

Scale 4 — Decision Driver

Vision-Led (V) ↔ Logic-Led (L)

Scale 5 — Execution Temperament (Suffix)

Adaptive (A) ↔ Structured (T)
(This is the "Identity" equivalent: appears as a suffix, not a separate type.)

---

3) Quiz Content Structure

Question format

Each question is:

- `id`
- `text`
- `scale` (Focus/Interface/Change/Decision/Execution)
- `direction` (which side agreement supports)
- optional `weight` (default 1.0, adjustable later)
- optional `reverse_scored` (true/false)

Answer format

Likert 1–5:

- 1 Strongly Disagree
- 2 Disagree
- 3 Neutral
- 4 Agree
- 5 Strongly Agree

---

4) Scoring Method (16Personalities-style percentages)

Convert a response to a signed score

Let `r` be 1..5. Convert to centered value:

- `v = r - 3` → yields `-2, -1, 0, +1, +2`

Each question has a `direction`:

- If agreement supports the "left" side (e.g., Builder), then signed contribution is `+v` to Builder.

- If agreement supports the “right” side (e.g., Analyzer), then signed contribution is `+v` to Analyzer.
  (Equivalently: “left” gets `+v`, “right” gets `-v` and you later compare totals.)

Include per-question weight:

- `contribution = v * weight`

Per-scale totals

Compute `sum` across questions in that scale:

- `scale_score = Σ(contribution * side_sign)`
  Where `side_sign` is +1 if statement aligns with “left” side, -1 if aligns with “right” side.

Turn score into a percentage

You want a stable 0–100% leaning.

Max per question is `2 * weight` (when r=5 and v=+2) in magnitude.

So:

- `max_possible = Σ(2 * weight)` per scale
- `normalized = scale_score / max_possible` which lies in [-1, +1]
- `percent_left = round(50 + 50 * normalized)`
- `percent_right = 100 - percent_left`

Then select the letter:

- If `percent_left >= 50` use left letter (B/U/E/V/Adaptive)
- else right letter (A/S/O/L/Structured)

Final Type

- Primary 4 letters: (Focus, Interface, Change, Decision) → 16 types
- Suffix: Execution → `-A` (Adaptive) or `-T` (Structured)
  Example: `B-S-O-L-T`

---

5) Results Content: the 16 Tech Personalities

Store the type definitions as content blocks:

- name (“Architect”, “Operator”, etc.)
- tagline
- description
- strengths
- watch-outs
- best-fit roles (from your 16)
- secondary fits
- recommended learning path (skills + resources)

This content should be CMS-like (editable without code).

---

6) Web App User Experience

Landing Page

- What it is (“Find your Tech Personality”)
- CTA: “Take the quiz”
- Secondary CTA: “How it works”
- Social proof area (optional): “X people have taken it”
- If logged in: show last result + “Retake”

Quiz Flow

- Progress bar (e.g., 1/40)
- One question per page (best completion)
- Or 5 per page (faster) with sticky progress
- Mobile-first big buttons 1–5
- Save progress (local + server if logged in)
- Optional “I’m not sure” (counts as 3/Neutral)

Results Page

Must feel like 16Personalities:

- Big personality name + code (e.g., **The Architect — B-S-O-L-T**)
- Bars for each scale showing %:
    - Focus: 68% Builder / 32% Analyzer
    - Interface: 22% User / 78% Systems
    - Change: 41% Exploratory / 59% Operational
    - Decision: 30% Vision / 70% Logic

    - Execution: 45% Adaptive / 55% Structured
- Description + strengths + watch-outs
- “Top 3 Roles for You” + why
- “Next 5 Skills to Learn”
- “Courses/resources for you” (personalized)
- Buttons:
    - Share result (image card)
    - Download PDF (optional)
    - Retake quiz

Recommendations Area (key differentiator)

“Based on your profile, here’s what to do this month”

- Suggested learning track (e.g., “Cloud Foundations”, “Frontend UX”, “ML Systems”)
- Resource links grouped:
    - “Start Here”
    - “Hands-on”
    - “Deep dive”
    - “Portfolio ideas”

Account (Optional but powerful)

- Email login / OAuth (Google)
- Save results history
- Compare changes over time
- Bookmark resources
- Add “School” & “Goals” preferences

---

7) Personalization Features (what makes it feel “alive”)

A) School-based resources

User selects:

- School (dropdown + “Other”)
- Major interest (AI/ML, SWE, Product, Design, Cyber, Cloud)
- Current level (Beginner/Intermediate/Advanced)

Then show:

- Courses at their school (manual list or scraped list)

- Clubs (e.g., AI club, cybersecurity club)
- Internal resources (tutoring centers, labs)
- Local opportunities (career center links)

Implementation idea:
Create a “Resource Directory” that can be filtered by:

- school_id
- department
- topic tags (ml, cloud, frontend, security, product)
- difficulty
- format (course, article, video, project, club)

B) “If you’re this type, start here” bundles

Pre-built bundles for each personality:

- “Your 2-week starter kit”
- “Your first portfolio project”
- “Your next course”
- “Your biggest trap + how to avoid it”

C) Type-to-role ranking

Each personality maps to roles with weights:
Example:

- Architect (B-S-O-L): Cloud + Systems + Backend high
- Storyteller (A-U-E-V): UX/UI + Product Design high

Use a simple scoring table:

- RoleScore = Σ(weight_of_trait_alignment * user_trait_percent)

---

8) Admin / CMS Requirements

You will want an admin dashboard for iteration.

Admin can:

1. Edit questions:
- text
- scale

- direction
- weight
- active/inactive

2. Edit personality type profiles (16):
- content sections
- role matches
- recommended skills
- recommended resources

3. Edit resources directory:
- title, url, tags, difficulty, school mapping

4. View analytics:
- completion rate by question
- dropout question index
- average time per question
- distribution of types
- role click-through rates

---

9) Data Model (Practical, minimal)

Tables (Postgres example)

users

- id (uuid)
- email
- name
- school_id (nullable)
- created_at

schools

- id
- name
- metadata (json)

quizzes

- id
- version (int)
- created_at
- is_active

questions

- id
- quiz_id
- order_index
- text
- scale (enum)
- direction (enum: LEFT/RIGHT)
- weight (float default 1.0)
- is_active

responses

- id
- user_id (nullable if anonymous)
- session_id (for anonymous)
- quiz_id
- started_at
- completed_at
- result_type_code (e.g., “BSOL”)
- result_suffix (e.g., “T”)
- result_json (stores full %s)

answers

- id
- response_id
- question_id
- value (1..5)
- created_at

type_profiles

- type_code (“BSOL”)
- name
- tagline
- content_json (strengths, risks, etc.)
- role_rankings_json

resources

- id
- title
- url
- description
- tags (array)
- difficulty
- school_id (nullable)

- type_codes (array nullable)
- role_tags (array)

feedback

- id
- response_id
- question_id (nullable)
- user_text
- rating (optional)
- created_at

---

10) Tech Stack Suggestion (simple, modern)

One solid path:

Frontend

- Next.js (App Router)
- Tailwind
- Server actions or API routes
- Chart bars for results

Backend

- Next.js API routes OR separate Express/Fastify
- Postgres + Prisma
- Auth: NextAuth (Google/email)

Analytics

- PostHog or simple internal tracking table

Hosting

- Vercel (frontend)
- Supabase/Neon for Postgres

---

11) Versioning + Iteration

You will iterate on questions a lot. Protect results integrity by:

- Versioned quizzes
- A response always links to the quiz version used
- Type mapping content can evolve, but store the raw percentages so you can re-render.

---

12) Viral/Share Features (optional but big)

- Shareable results card (image)
- “Compare with a friend” link (shows both types)
- “Team view” (for clubs/classes)
- “What role should you explore next?” mini CTA

Bottom Section: Upgrade to an AI/ML Project (Learning + Optimization)

Goal

Use collected data to **improve the quiz’s predictive power**:

- Which questions actually predict the **role someone wants**
- Which questions are noisy or misleading
- How to reweight questions per population (beginners vs experienced, etc.)

Data to collect (with consent)

On the results page, ask:

1. “Which role are you most interested in right now?” (pick 1–3)
2. “How confident are you?” (1–5)
3. “Why?” (free response)
4. Optional: “What’s your current skill level?” + “Past experience”

This turns your app into a dataset:

- Inputs: answers + derived trait percents
- Labels: intended role(s), confidence, text rationale

ML tasks you can do

1) Role prediction model

Train a model to predict role interest from answers.

- Baseline: multinomial logistic regression
- Better: gradient boosting (XGBoost/LightGBM)
- Output: probability distribution over roles

Use this to:

- Compare “hand-designed scoring” vs model prediction
- Find which traits/questions matter most

2) Question quality scoring

For each question:

- Measure information gain / feature importance
- Correlation with correct role labels
- Detect questions that cause confusion (high variance, low predictive contribution)

Outcome:

- Remove weak questions
- Rewrite ambiguous ones
- Adjust weights based on data

3) Personalized weights (“why these weights”)

You can compute:

- Global weights (best overall)
- Segment weights (by school, skill level, major interest)
  Then show a transparent explanation:
- “Your answers suggest Systems + Operational + Logic because you strongly agreed with questions about reliability and infrastructure.”

4) Use free-text explanations (optional advanced)

If users write “why,” you can:

- Embed text (sentence transformers)
- Cluster rationales

- Detect motivations:
    - “I like solving puzzles” (SWE)
    - “I like helping users” (UX/PM)
    - “I want to stop hackers” (security)

Then add a “Motivation layer” to recommendations.

Privacy / ethics must-haves

- Clear consent checkbox for using answers to improve the test
- Let users delete their data
- Store anonymous session ids when not logged in
- Don’t sell personal data; keep it educational

“AI mode” product feature (cool upgrade)

On results page:

- “AI second opinion” (shows top 3 roles predicted by model)
- “Which questions influenced this most?” (top features)
- “Was this accurate?” thumbs up/down feedback loop

If you want, I can also generate:

- The full **JSON structure** for questions + type profiles (ready to seed a database)
- A clean **Next.js folder structure** + API endpoints list
- The exact **role-mapping weight table** (traits → role probabilities) so results feel *sharp* immediately.