AI Study Buddy

Technical Specification

Project: P6: AI Study Buddy

Generated: 2025-12-27

Repository: /workspaces/embodiment-lab

AI Study Buddy: Workflow and User Journey Guide (Owner Edition)

Document purpose

This guide explains, in narrative form, how the AI Study Buddy platform works end to end. It focuses on real user journeys (Participant, Admin, Owner, Viewer/Mentor), how the UI leads each user, and how those interactions are captured by the system. It is intentionally descriptive and story oriented, with concrete workflows, diagrams, and technical detail grounded in the codebase.

Scope

- Participant experience: entry, consent, demographics, pre-test, learning, scenarios, feedback, post-test, completion.
- Admin/Owner experience: sessions, responses, slide and question management, API settings, data quality validation, exports.
- Mentor (viewer) experience: read-only views.
- Technical architecture: client, Supabase edge functions, data model, AI integrations, and guardrails.

Executive overview (plain-language)

The platform is a single-page web app that runs a structured research study on AI literacy. Participants are guided through a linear flow: consent, demographics, pre-test, learning content (slides + tutor), short scenarios, post-test, and completion. The experience supports two learning modes: Text Mode (chat-based tutor) and Avatar Mode (video avatar with speech). Admins and the owner can manage content, view live study data, validate sessions, and export full datasets. Mentors can view high-level research data but cannot change content or export.

From a technical point of view, the app is a React + Vite client backed by Supabase (database, auth, edge functions). The learning content and questions are stored in Supabase tables and rendered dynamically on the front end. Participant responses are saved through a server-side edge function that validates input and writes to the database. AI tutor features are powered by OpenAI GPT-5 mini for text and avatar responses. The system enforces a strict study flow with sessionStorage markers and blocks backward navigation after each stage is completed.

Actors and roles

Participant

Goal: complete the study once, in a clear step-by-step journey.

What they do:

- Start from a study link (either text or avatar mode).
- Give consent.
- Answer demographics and pre-test questions.
- Learn by reading slides and chatting with a tutor.
- Complete learning scenarios and feedback.
- Answer post-test questions.
- Finish and optionally download their responses.

Admin

Goal: manage study content and observe results without breaking data integrity.

What they do:

- View sessions and responses.
- Edit slides and questions.
- Update Anam API key and toggle Anam usage.
- Export responses and review suspicious sessions (request validation).

Restrictions:

- Cannot delete sessions or reset the study.
- Cannot directly accept or ignore suspicious sessions (must request owner approval).

Owner

Goal: full control over study configuration and data quality decisions.

What they do:

- Everything an admin can do.
- Control the master API switch and OpenAI toggle.
- Accept or ignore suspicious sessions.
- Delete sessions (when necessary).
- Download system documentation and workflow guide.

Viewer / Mentor

Goal: read-only monitoring and supervision.

What they do:

- View sessions and overview metrics.

Restrictions:

- No export access.
- No content editing.
- No API settings.
- No audit log.

High-level system architecture

The platform uses a front-end only client paired with Supabase for data, auth, and server-side functions. The core architecture is shown below.

```
[Participant Browser]
   |  React SPA (Vite + TS)
   |  UI flow + sessionStorage
   v
[Supabase Edge Functions]
   - save-study-data
   - complete-session
   - chat (text tutor)
   - anam-session (avatar)
   - generate-image (playground)
   - update-session-validation
   - toggle-api
   - save-avatar-time

   v
[Supabase Postgres]
   - study_sessions
   - demographic_responses
```

```
- pre_test_responses
- post_test_responses
- scenarios
- dialogue_turns
- study_questions
- study_slides
- app_settings
- admin_audit_log
- user_roles / admin_users
```

AI integrations:

- Text tutor: Supabase edge function `chat` -> OpenAI API (GPT-5 mini) -> streaming response.
- Avatar tutor: Supabase edge function `anam-session` -> Anam API -> avatar session.
- Image Playground: Supabase edge function `generate-image` -> OpenAI image endpoint -> image output.

Participant story: from link to completion

This section explains the participant journey as a story, showing exactly how the UI leads the participant and how the system captures their data.

Step 1: Study entry link

Entry routes:

- `/study/text`
- `/study/avatar`

What the user sees:

- A welcome page with study description, time estimate, and privacy notice.
- The selected mode is shown as a badge (Text Mode or Avatar Mode).

System behavior:

- The `StudyEntry` page validates the mode from the URL.
- If an invalid mode is provided, the user sees an error page.
- If the query parameter `?reset=1` is included, the system clears all sessionStorage data and allows a fresh start.
- The page sets a local lock if the study has already been completed on the device (stored in localStorage).

Diagram:

```
/ study / :mode
   -> validate mode
   -> if invalid: error page
   -> if reset=1: clear session storage
   -> show welcome content
```

Step 2: Consent

The participant must review and accept the consent. If they accept, a session is created.

System behavior:

- `save-study-data` is called with action `create_session`.
- A unique `sessionId` is generated by the server and stored in sessionStorage.
- The session is written to `study_sessions`.

Outcome:

- Consent is recorded by the existence of a sessionId in sessionStorage.

Step 3: Demographics

The participant fills demographic questions.

System behavior:

- Answers are stored in sessionStorage as `demographics`.
- When the participant submits, the app calls `save-study-data` with action `save_demographics`.
- Data is written to `demographic_responses` and `demographics` tables.

Quality checks:

- `useBotDetection` tracks how fast questions are answered.
- Suspicious timing flags can be reported to `study_sessions` for later review.

Step 4: Pre-test

The participant completes knowledge questions.

System behavior:

- Answers saved in sessionStorage as `preTest`.
- `save-study-data` called with `save_pre_test` to write responses.
- Responses stored in `pre_test_responses`.

AI personalization:

- The text tutor uses pre-test data to adapt responses. The pre-test is sent to the `chat` edge function and used to build a weak/strong topic profile.

Step 5: Mode assignment

The participant selects Text Mode or Avatar Mode. If they entered via `/study/:mode`, the mode can be pre-assigned.

System behavior:

- Session guard prevents switching modes mid-study.
- If mode switching is attempted, the session is reset and flagged.
- `save-study-data` action `update_mode` stores the selected mode in `study_sessions`.
- The selection is cached in sessionStorage as `studyMode`.

Diagram:

```
User chooses mode
 -> update_mode (edge function)
 -> store studyMode in sessionStorage
 -> navigate /learning/:mode
```

Step 6: Learning (slides + tutor)

The participant enters the learning area. This is where the main instructional content lives.

UI layout:

- Header with logo, mode badge, Finish button, and Exit button.
- Slide viewer (center).
- Tutor panel (right) or avatar panel (right).
- AI Playground toggle (rightmost panel).

Slides:

- Slides are loaded from `study_slides`.
- Only active slides are shown, ordered by `sort_order`.

Tutor (Text Mode):

- A chat panel appears on the right.
- The assistant greets on the first slide and updates context on each slide change.
- The chat uses the `chat` edge function and streams responses.

Tutor (Avatar Mode):

- The avatar panel uses the Anam SDK.
- The system calls `anam-session` to create a session, injecting slide context.
- Avatar time on each slide can be logged to `avatar_time_tracking`.

AI Playground:

- Optional panel that calls `generate-image` to produce an image using the OpenAI image endpoint.

Mobile behavior:

- The chat panel is placed at the bottom for Text Mode.
- Avatar Mode is disabled on mobile to avoid multiple Anam clients.

Step 7: Scenarios

Scenarios are structured conversations based on predefined scripts.

System behavior:

- Scenario content is loaded from `src/data/scenarios`.
- Conversations are persisted in sessionStorage as `scenario-<id>` during the session.
- After each scenario, `ScenarioFeedback` collects ratings and saves results via `save-study-data` with `save_scenario`.
- Scenario responses are written to the `scenarios` table and dialogue entries to `dialogue_turns` (when logged).

Diagram (simplified):

```
Scenario page
  -> scripted AI line
  -> user responds
  -> repeat
  -> feedback
  -> save_scenario
```

Step 8: Post-test

The post-test is split into three pages to reduce overload.

System behavior:

- Each page stores responses in sessionStorage (postTestPage1/2/3).
- Responses are saved to `post_test_responses` through `save-study-data`.
- Flow guards prevent going backwards once a post-test page is completed.

Step 9: Completion

The participant sees a completion page and optional download.

System behavior:

- `complete-session` is called to mark the session as completed.

- A localStorage flag `studyCompleted = true` prevents repeat participation on the same device.
- A response export button is available (client-side export).

Participant journey (end-to-end diagram)

```
[Study Link]
   -> [Consent]
   -> [Demographics]
   -> [Pre-test]
   -> [Mode Assignment]
   -> [Learning: Slides + Tutor + Playground]
   -> [Scenario 1..N]
   -> [Scenario Feedback]
   -> [Post-test 1]
   -> [Post-test 2]
   -> [Post-test 3]
   -> [Completion + Optional Export]
```

Narrative walkthrough: Participant in Text Mode (story)

Imagine a participant named Alex receives the text-mode link. They open `/study/text` and immediately see a polished welcome card with an estimated duration and a privacy reminder. The UI clearly labels the mode so Alex knows this is the text-based study.

Alex clicks to continue and reads the consent. Once they accept, the system creates a new session server-side. This is a crucial moment: a new `sessionId` is minted and saved to sessionStorage, which becomes the spine for every subsequent save. From this point onward, every page the participant completes writes data back to Supabase using the same sessionId.

On the demographics page, Alex answers profile questions. The UI captures each answer and also records time spent and answer speed. When Alex submits, the app sends a validated payload to `save-study-data`. If the answers were completed unusually fast, a suspicion flag can be attached to the session. Alex does not see this; it is a data-quality mechanism for the research team.

Next comes the pre-test. These questions measure baseline knowledge. Alex answers them, and the data is saved to `pre_test_responses`. This is not just a scoring step; the pre-test becomes part of the context the tutor uses to adjust explanations later.

Alex now reaches the mode assignment page. Because they entered through `/study/text`, the system already pre-assigns the mode and auto-navigates forward. The platform intentionally prevents switching mid-session; this protects the study design and ensures clean data.

In the learning section, Alex sees a slide viewer on the left and a chat panel on the right. The tutor greets them on the first slide. As Alex moves through the slides, the tutor injects the current slide title and key points into the system context, ensuring responses stay relevant. When Alex asks a question, the system streams the response from the chat edge function, which calls the OpenAI API (GPT-5 mini). This feels responsive and conversational, but the model is strictly constrained to the slide topic.

If Alex opens the AI Playground, they can generate images with prompts and parameters. This is optional and does not block the study flow, but it enriches the experience and gives a practical, hands-on step.

After learning, Alex enters a scenario. The scenario is a scripted dialogue, so Alex experiences a guided conversation. Once the final dialogue turn is reached, the app collects scenario feedback. The conversation and feedback are saved in the scenarios tables, ensuring that later analysis can compare perceptions against responses.

Finally, Alex completes the post-test over three short pages. Each page saves and locks once completed. Once the final page is submitted, Alex is shown a completion screen and can optionally download their responses. The system marks the session completed in the database and sets a localStorage flag to prevent a second run on the same device.

Narrative walkthrough: Participant in Avatar Mode (story)

Now imagine the same participant starts from `/study/avatar`. The entry page is identical, but the mode badge confirms an avatar-guided experience.

The early flow is the same: consent creates a session, demographics and pre-test are saved, and the system confirms the learning mode. When the participant reaches the learning page, the main difference is the right panel: instead of text chat, the avatar is visible and speaks.

When the learning page loads, the client requests an Anam session via the `anam-session` edge function. This function checks the master API switch and the Anam toggle in `app_settings`, and then retrieves the most recent API key. If the key is stored in the database, it overrides the environment default. This ensures each admin can use their own free Anam key if needed.

The avatar uses the same slide context logic as text mode. Each slide provides a system prompt and key points, ensuring the avatar stays on topic. The transcript is shown in the UI for accessibility and review.

In avatar mode, the AI Playground remains optional but available. The participant can still generate images, but the core differentiator is the visual, spoken tutor experience.

As in text mode, scenarios are completed after the learning section. The scenario dialogues and feedback are stored the same way. The post-test and completion pages are identical between modes to keep the study consistent.

One practical limitation: on smaller screens, avatar mode is intentionally disabled during the learning phase to avoid creating multiple Anam clients at once. The UI shows a message instructing participants to use a larger screen for the avatar experience.

Participant journey: micro-interactions and UI guardrails

Key guardrails the participant experiences (often silently):

- Flow enforcement: the user cannot skip to later steps without completing prerequisites.
- Step locking: once a step is completed, navigation back is blocked.
- Mode switching prevention: attempting to switch mode resets the session and returns to the start.
- Local completion lock: a completed session is stored in localStorage, blocking repeat participation on the same device.

Timing and data quality tracking:

- Each page records entry time and per-question answer times.
- If suspicious timing is detected, a flag is written to the session.
- The participant is never shown these flags, but the admin can review them later.

Component map (participant-facing)

The key participant-facing pages and components are:

- `StudyEntry`: validates mode, shows the initial welcome message, and handles reset flow.
- `Consent`: collects consent and creates the session.

- `Demographics`: renders demographic questions, saves results.
- `PreTest`: renders knowledge questions, saves results.
- `ModeAssignment`: locks the mode and prevents mid-study switching.
- `Learning`: displays slides, tutor panel, and optional AI Playground.
- `SlideViewer`: renders slide content, title, and key points.
- `TextModeChat`: streams text tutor responses.
- `AvatarModePanel`: renders the avatar client and transcript.
- `Scenario`: scripted dialogue flow.
- `ScenarioFeedback`: collects trust, confidence, and engagement ratings.
- `PostTestPage1/2/3`: post-test survey pages.
- `Completion`: marks the session complete and offers optional export.

Admin and owner workflows

Admin login

Route:

- `/admin/login`

System behavior:

- Supabase auth used for admin sign-in.
- Role is verified by checking `user_roles`.
- Access granted for users with researcher/admin role.

Admin dashboard overview

The Overview tab provides a high-level status view:

- Total sessions, completion counts, and trends.
- Data quality alerts and suspicious sessions.
- Mode distribution and summary metrics.

Sessions tab (data quality control)

The Sessions tab is the primary place for reviewing individual sessions.

Key actions:

- View session status and mode.
- See suspicion flags and detailed reasons for suspicious behavior.
- Validate a session: Accept or Ignore (owner can finalize).

Validation logic:

- Suspicious sessions are excluded from analysis unless explicitly accepted.
- Validation status is stored in `study_sessions.validation_status`.

Responses tab (export and audit)

The Responses tab aggregates answers from demographics, pre-test, and post-test tables.

Key actions:

- Export full dataset across sessions.
- Export per-session data with dialogue if available.
- Filter results by validation status.

Slides tab (content management)

Admins and the owner can edit slide content that participants see.

Fields:

- Title
- Content (Markdown supported)
- Key points (bullet summary)
- System prompt context (used for the AI tutor)
- Visibility (active/inactive)

Changes take effect immediately for new sessions.

Questions tab (survey management)

Study questions are stored in `study_questions` and can be edited.

Capabilities:

- Create, edit, and disable questions.
- Mark correct answers for pre-test knowledge items.
- Control question visibility by section.

API Settings tab

This is the operational control panel for AI services.

Capabilities:

- Master switch (owner only).
- Toggle OpenAI chat (owner only).
- Toggle Anam avatar (admin and owner).
- Update Anam API key (admin and owner).

Audit:

- All changes are logged to `admin_audit_log`.
- API key changes store the user email and timestamp.

My Access tab

Shows the logged-in user what they can and cannot do.

Activity Log tab

Displays an audit log of content and API changes.

Viewer / mentor workflow

Viewer accounts are read-only. They can:

- View Overview metrics.
- View Sessions list.

They cannot:

- Edit content.
- Export data.
- View audit logs.
- Access API settings.

This separation is enforced by permission checks in the UI and at the edge function layer.

Data model and storage

The system uses a Supabase Postgres database. Key tables include:

- study_sessions: top-level record for each participant session.
- demographic_responses: per-question demographic answers.
- pre_test_responses: per-question pre-test answers.
- post_test_responses: per-question post-test answers.
- scenarios: per-scenario ratings and metadata.
- dialogue_turns: stored dialogue lines linked to scenario sessions.
- study_slides: learning content displayed in the slides.
- study_questions: the question bank used for all surveys.
- app_settings: API toggles and stored Anam API key.
- admin_audit_log: audit trail for admin/owner actions.
- avatar_time_tracking: time spent per slide in avatar mode.

Session linkage:

- The sessionId used in the UI is a string identifier saved in sessionStorage.
- The database also uses a UUID `id` for internal relations.
- `save-avatar-time` resolves the internal UUID when saving slide durations.

Data flow for saving participant responses

```
UI (Participant Page)
  -> sessionStorage updated
  -> Edge function: save-study-data
      - validates input schema
      - writes to table(s)
      - updates study_sessions
  -> Supabase Postgres tables
```

The `save-study-data` function supports multiple actions:

- create_session
- save_demographics
- save_pre_test
- save_post_test
- save_scenario
- update_mode
- reset_session
- update_activity
- report_suspicious

Each action has its own validation schema to protect data quality and prevent malformed writes.

AI tutor data flow (Text Mode)

```
User -> TextModeChat
  -> streamChat()
     -> fetch /functions/v1/chat
```

```
          -> chat edge function
             -> OpenAI API (GPT-5 mini)
             -> stream response
    -> UI shows streaming reply
```

Text Mode uses slide context as a system message. It injects:

- Slide title
- Slide content context
- Key points to cover

This ensures the tutor stays on the current topic.

Avatar tutor data flow (Avatar Mode)

```
User -> AvatarModePanel
  -> useAnamClient
     -> /functions/v1/anam-session
        -> check app_settings
        -> select API key (db override or env)
        -> create Anam session
  -> Avatar client renders video + transcript
```

Key behavior:

- Avatar sessions are blocked if the master switch or Anam switch is disabled.
- The system logs slide time for avatar sessions using `save-avatar-time`.

Image playground data flow

```
User -> ImagePlayground
  -> /functions/v1/generate-image
     -> OpenAI image endpoint
  -> UI displays image
```

The playground accepts parameters (prompt, negative prompt, CFG, steps, size) and is optional for participants.

Study flow guardrails

The study is intentionally linear. The `useStudyFlowGuard` hook enforces:

- No skipping ahead without completing required steps.
- No returning to already completed steps.
- Forced redirects based on sessionStorage markers.

Key sessionStorage markers:

- sessionId
- demographics
- preTest
- studyMode
- postTestPage1
- postTestPage2
- postTestPage3
- studyCompleted

LocalStorage marker:

- studyCompleted (prevents repeat participation on the same device)

If a user tries to return to a completed step, they are redirected forward with a warning toast.

Suspicious activity detection (data quality)

The platform tracks timing patterns via `useBotDetection`:

- Time spent on a page.
- Answer speed per question.
- Average answer speed.
- Slide reading time.

Suspicion scoring:

- Each rule adds points (for example, completing a page too fast adds a large score).
- A higher score indicates more suspicious behavior.
- Flags and scores are saved to `study_sessions` for review.

Admin/owner workflow:

- Suspicious sessions appear in the Sessions tab.
- Owner can accept or ignore them.
- Accepted sessions are included in exports and analytics.

API settings workflow (owner and admin)

The API settings panel reads and writes entries in `app_settings`.

Key values:

- api_enabled (master switch)
- openai_api_enabled
- anam_api_enabled
- anam_api_key

Rules:

- Only owner can toggle master switch and OpenAI.
- Admins can toggle Anam and update the Anam key.
- Updated_by and updated_at fields are recorded and shown in the UI.

Content editing workflow (slides and questions)

Slides:

- Stored in `study_slides` with fields: title, content, key_points, system_prompt_context, sort_order, is_active.
- Admins can update text and key points without deleting.
- Owner can fully manage slide visibility and order.

Questions:

- Stored in `study_questions` and grouped by section (demographics, pre-test, post-test).
- Admins can edit text, options, and correct answers (for knowledge scoring).
- Questions can be disabled to remove them from live flow.

Export workflow (admin and owner)

Exports are generated in the Responses tab and the Sessions tab.

Export types:

- Session-level CSV or PDF with demographics, pre-test, post-test, scenarios, and dialogue.
- Study-level CSV across all sessions.

Validation rules:

- Suspicious sessions are excluded unless accepted.
- The system applies the validation status before including data in export.

Error and recovery paths

Key recovery points:

- Invalid study link: show error page, no session created.
- Mode switching attempt: session reset and user returned to start.
- Missing required step: flow guard redirects to the correct step.
- API disabled: chat or avatar requests return a 503 response.

Operational notes (owner)

- The platform depends on Supabase edge functions for critical writes. If edge functions fail, data may not be saved.
- The `create-admin-users` function seeds admin accounts based on environment secrets.
- API keys can be rotated in the UI, and the current key owner is shown in API Settings.

Narrative walkthrough: Admin reviewing data quality

An admin logs in and opens the Sessions tab. They see a list of sessions with status, mode, and data quality indicators. The suspicious sessions are visually flagged. The admin clicks into a flagged session to review the details. The system displays a list of flags such as \"page completed too quickly\" or \"average answer time very fast\".

At this point the admin has two paths:

- If the session looks valid despite the flags, they request validation so the owner can accept it.
- If the session is clearly invalid (for example, all answers were completed in seconds), the admin requests that the owner ignore it.

The important detail is that the admin does not make the final decision. The workflow is designed to keep data governance centralized with the owner while still allowing admins to triage the workload.

When the owner later reviews the session, the decision is written to `study_sessions.validation_status`. That status drives whether the session is included in exports and statistics.

Narrative walkthrough: Owner acceptance and deletion

The owner sees the same Sessions list but with full control. If a flagged session is clearly invalid, the owner can ignore it. If the session is legitimate, the owner can accept it. This status is immediate and affects exports and dashboards.

In rare cases (for example, test data or corrupted sessions), the owner can delete sessions. The `update-session-validation` edge function deletes all related records in an ordered sequence (dialogue_turns -> scenarios -> responses -> session). This ensures the database remains consistent and prevents orphaned rows.

Admin and owner UI component map

The most important admin-facing components include:

- `AdminOverview`: summary metrics and data quality alerts.
- `AdminSessions`: session list, filters, and validation workflow.
- `AdminResponses`: aggregate responses and export features.
- `AdminSlides`: content editing for slides.
- `AdminQuestions`: question bank management.
- `ApiToggle`: API settings and key rotation.
- `AdminAuditLog`: change history for admin actions.

Data dictionary (practical, owner-readable)

This dictionary describes what each table is for and what kind of data it stores. Field names may vary, but the purpose is stable.

Study sessions

- `study_sessions`: one row per participant session. Stores mode, status, timing metadata, suspicion flags, validation status, and completion timestamps.

Demographic answers

- `demographic_responses`: one row per demographic question answered (sessionId, questionId, answer).
- `demographics`: structured demographic summary for quick reporting.

Knowledge and survey answers

- `pre_test_responses`: per-question answers for the knowledge pre-test.
- `post_test_responses`: per-question answers for post-test pages.

Scenarios and dialogue

- `scenarios`: per-scenario results (confidence, trust, engagement).
- `dialogue_turns`: line-by-line conversation content for each scenario.

Content tables

- `study_slides`: slide title, content, key points, AI context, ordering, and active flag.
- `study_questions`: question bank for demographics, pre-test, and post-test.

System settings

- `app_settings`: key-value storage for API toggles and Anam API keys.
- `admin_audit_log`: records who changed what and when.
- `user_roles` / `admin_users`: access control for admin panel.
- `avatar_time_tracking`: time spent per slide in avatar mode.

Client-side state and persistence (important for troubleshooting)

The front end stores short-lived state in sessionStorage and long-lived flags in localStorage.

SessionStorage keys (examples):

- sessionId: the current session identifier.
- studyMode: text or avatar.
- demographics, preTest: cached answers before save.
- postTestPage1/2/3: markers for completion.
- currentSlide: last slide index.
- scenario-<id>: conversation history per scenario.

LocalStorage keys:

- studyCompleted: set to true once the completion page is reached.

These keys are how the guardrails know where the participant is in the flow. If a key is missing, the user is redirected back to the last valid step.

AI prompt shaping and context (how the tutor stays on topic)

Text Mode:

- Every request includes a system context message based on the current slide.
- The context contains slide title, key points, and a system prompt summary.
- The chat function uses pre-test results to customize explanation depth.

Avatar Mode:

- The Anam session is created with a system prompt that includes slide context and strict topic boundaries.
- This keeps avatar responses aligned with the instructional content.

The design goal is consistent: the tutor should always talk about the current slide, not drift into unrelated topics.

System limits and constraints (operational reality)

These constraints are intentional and should be considered in study operations:

- Avatar Mode on mobile is disabled in the learning view to avoid duplicate Anam clients.
- Mode switching mid-session is treated as invalid and resets the session.
- API usage depends on app_settings toggles; when disabled, the AI services return 503.
- The `save-study-data` function enforces input validation and rate limiting.

Understanding these constraints helps explain participant behavior and avoids false alarms in data review.

Appendix A: UI route map

```
/                       -> Welcome
/study/:mode            -> StudyEntry
/consent                -> Consent
/demographics           -> Demographics
/pre-test               -> PreTest
/mode-assignment        -> ModeAssignment
/learning/:mode         -> Learning (Slides + Tutor)
/scenario/:mode/:id     -> Scenario
/scenario/:mode/:id/feedback -> ScenarioFeedback
/post-test-1            -> PostTestPage1
/post-test-2            -> PostTestPage2
/post-test-3            -> PostTestPage3
/completion             -> Completion
```

```
/admin/login           -> AdminLogin
/admin                 -> AdminDashboard
```

Appendix B: Edge functions summary

- save-study-data: create sessions, store responses, log suspicious activity, update mode.
- complete-session: mark session as completed.
- chat: text tutor (OpenAI GPT-5 mini), uses pre-test context.
- anam-session: avatar tutor (Anam), injects slide context.
- generate-image: AI playground image generation.
- save-avatar-time: record slide time in avatar mode.
- toggle-api: API toggles and key updates with audit log.
- update-session-validation: accept/ignore suspicious sessions.
- create-admin-users: bootstrap admin accounts.

Appendix C: Example sequence diagram (text tutor)

```
Participant
  -> TextModeChat
  -> streamChat() [client]
  -> /functions/v1/chat [edge]
  -> OpenAI API (GPT-5 mini)
  -> stream response
  -> UI updates
```

Appendix D: Example sequence diagram (avatar tutor)

```
Participant
  -> AvatarModePanel
  -> useAnamClient() [client]
  -> /functions/v1/anam-session [edge]
  -> Anam API session
  -> Avatar video + transcript
```

Appendix E: Example sequence diagram (save responses)

```
Participant submits answers
  -> sessionStorage updated
  -> save-study-data (edge)
     - validate
     - insert into response tables
     - update study_sessions
```

Appendix F: Participant guardrail states

```
Missing sessionId         -> redirect to /consent
Missing demographics      -> redirect to /demographics
Missing preTest           -> redirect to /pre-test
Missing studyMode         -> redirect to /mode-assignment
Missing postTestPage1     -> redirect to /post-test-1
Missing postTestPage2     -> redirect to /post-test-2
Missing postTestPage3     -> redirect to /post-test-3
```

Appendix G: Suspicion rules (plain-language)

Suspicion flags are generated if:

- A page is completed faster than a minimum threshold.
- Many answers are given suspiciously fast.
- The average answer time is below a minimum threshold.
- Slide view time is too short to be realistic.

These flags and scores are stored in `study_sessions` and shown to admins in the Sessions tab for validation.

Closing note

This workflow guide is intentionally narrative. It should help you explain the full story to stakeholders: how a participant is guided, how data is recorded, and how the owner/admin workflows keep the study consistent and trustworthy.