# AI Study Buddy

## Technical Specification

Project: P6: AI Study Buddy

Generated: 2025-12-27

Repository: /workspaces/embodiment-lab

# AI Study Buddy System - Technical Specification

Version: 1.0

Generated: 2025-12-26

Repository: /workspaces/embodiment-lab

Project: P6: AI Study Buddy

Primary owner: jakub.majewski@tum.de

Mentor (viewer access): efe.bozkir@tum.de

## Document purpose and provenance

This document is a full, code accurate, end to end technical specification of the AI Study Buddy platform. It describes the system from four perspectives: participant, admin, owner, and mentor. It documents architecture, data flow, session state, AI integration, and export logic, with enough detail to support maintenance, audits, and research reporting.

The content is derived from the current codebase in /workspaces/embodiment-lab and the deployed Supabase edge functions. Any earlier auto generated documentation is known to be incomplete or outdated. This specification supersedes all earlier drafts and should be treated as the authoritative reference.

## Table of contents

## 1. Executive summary

AI Study Buddy is a research platform built to compare two learning modalities: a text based AI tutor and an avatar based AI tutor. The system guides participants through consent, demographics, pre test, learning, post test, and feedback, while collecting structured responses and optional qualitative data. The same instructional content is used for both modes to ensure a fair comparison.

The platform is implemented as a React 18 + TypeScript single page application, bundled with Vite and styled with Tailwind and shadcn UI. The backend uses Supabase for authentication, Postgres storage, realtime updates, and serverless edge functions. AI services use OpenAI GPT-5 mini for tutoring and OpenAI image generation for the playground, while avatar streaming is handled by the avatar SDK.

The study flow is guarded by a client side state machine that prevents step skipping and controls repeat participation per device. Data quality checks automatically flag suspicious behavior based on timing heuristics. Admin and owner tooling supports content editing, validation workflows, and comprehensive exports for analysis.

## 2. Research objectives and study design

The study is designed to measure how the presence of an embodied AI tutor changes learner outcomes and perceptions. The design is a standard pre test / post test experiment with additional perception and feedback measures.

Primary objectives

- Measure baseline knowledge of AI image generation concepts.
- Provide a consistent curriculum across both learning modes.
- Measure post learning knowledge gain.

- Quantify trust, engagement, satisfaction, and expectations.
- Collect open ended feedback for qualitative analysis.

Secondary objectives

- Track engagement through interaction time and slide viewing time.
- Observe usage patterns for optional tools (image Playground).
- Flag low effort or anomalous submissions to protect data integrity.

Data categories

- Demographics
- Pre test knowledge
- Post test knowledge
- Likert scale perception measures
- Open feedback text
- Session metadata (time, duration, mode)
- Data quality signals and validation status
- Avatar slide time tracking (avatar mode)

## 3. System scope and non goals

In scope

- A single study with a defined questionnaire set
- Two learning modes: text and avatar
- Live content updates by admins
- Data quality and validation workflow
- Export capability for analysis

Out of scope

- Participant login or identity management
- Automated randomization across multiple experiments
- Multi study routing within a single admin panel
- Storage of participant video or audio
- Detailed moderation tools for chat content

## 4. Architecture overview

The system is organized into three layers: client UI, backend services, and external AI providers.

### 4.1 Client application

Core stack

- React 18 with TypeScript
- Vite bundler and dev server
- Tailwind CSS and shadcn UI components
- TanStack Query for data access where needed
- Recharts for analytics visuals
- jsPDF and jspdf-autotable for PDF exports

Key directories

- src/pages: route level pages (Welcome, Consent, Demographics, PreTest, Learning, PostTest, Completion, Admin)
- src/components: reusable UI components and admin widgets
- src/components/modes: Text mode chat and Avatar mode panels
- src/hooks: slide loading, question loading, session guard, bot detection, Anam integration
- src/lib: permissions, audit logging, study data persistence
- src/utils: chat streaming and PDF generation utilities
- supabase/functions: backend edge functions

### 4.2 Backend services

**Supabase provides:**

- Postgres database for all study data
- Supabase Auth for admin and owner login
- Edge functions for privileged operations and AI calls
- Realtime channels for live admin updates

Edge functions use the service role key and bypass RLS. Client operations use the anon key and are restricted by RLS or by function level authorization.

### 4.3 AI integrations

Text tutor

- The client sends chat history to /functions/v1/chat
- The edge function builds a system prompt and calls the OpenAI API (GPT-5 mini)
- The gateway uses openai/gpt-5-mini
- Responses are streamed back via SSE

Avatar tutor

- The client requests a session token from /functions/v1/anam-session
- The edge function builds a detailed system prompt and retrieves the Anam API key
- The Anam SDK streams the avatar video and audio to the client

Image Playground

- The client calls /functions/v1/generate-image
- The edge function forwards to the OpenAI image generation endpoint
- The gateway uses google/gemini-2.5-flash-image-preview

### 4.4 Deployment and environment

Frontend is deployed via the Lovable platform, connected to GitHub. Supabase hosts the database and edge functions. Environment variables configure API keys and secrets.

Key environment variables

- VITE_SUPABASE_URL
- VITE_SUPABASE_PUBLISHABLE_KEY
- SUPABASE_SERVICE_ROLE_KEY

- LOVABLE_API_KEY
- ANAM_API_KEY
- ADMIN_CREATION_SECRET
- OWNER_PASSWORD
- DEFAULT_ADMIN_PASSWORD
- MENTOR_PASSWORD

## 5. User roles and permissions

Roles are defined by email and by membership in Supabase user_roles. Permissions are derived in src/lib/permissions.ts and are enforced in both UI and edge functions.

### 5.1 Owner

Capabilities

- Full access to all admin tabs and exports
- Create, edit, hide, and delete slides
- Create, edit, disable, and delete questions
- Toggle master API and OpenAI API switches
- Toggle Anam API and update Anam API key
- Validate sessions directly
- Delete sessions and cascade data
- View full audit log

### 5.2 Admin

Capabilities

- Create, edit, and hide slides (cannot delete)
- Create, edit, and disable questions (cannot delete)
- Toggle Anam API
- Update Anam API key
- Export data
- Request validation decisions (pending approval)
- View audit log

### 5.3 Viewer (mentor)

Capabilities

- View sessions and overview metrics
- No access to exports, settings, or content editing

### 5.4 Permission enforcement

- UI gating hides tabs and buttons when permissions are not granted.
- toggle-api and update-session-validation edge functions verify user roles.
- Viewer emails are explicitly blocked in sensitive functions.

## 6. Participant end to end flow

The participant experience is a strictly ordered flow. Each step has a defined entry requirement enforced by

useStudyFlowGuard, and completion is tracked in sessionStorage.

### 6.1 Study entry (Welcome and StudyEntry)

Welcome (/)

Purpose

- Provide a public entry point
- Summarize study goals and duration
- Encourage consent and participation

Key behaviors

- If localStorage contains studyCompleted, the CTA is disabled.
- CTA clears sessionStorage and routes to /consent.

StudyEntry (/study/:mode)

Purpose

- Allow direct entry via mode specific links

Key behaviors

- Validates the mode param (text or avatar)
- Sets preAssignedMode in sessionStorage
- Provides a mode specific intro screen
- Supports reset via ?reset=1 query

### 6.2 Consent (/consent)

Purpose

- Present informed consent
- Require explicit confirmation

UI elements

- Consent sections with icons
- Checkbox to confirm consent and age
- Continue and Decline buttons

Logic

- On consent, create_session is invoked with a default mode of text
- sessionId is stored in sessionStorage
- Participant is routed to /demographics

### 6.3 Demographics (/demographics)

Purpose

- Collect demographic data
- Ensure all questions are answered
- Enforce minimum age

UI elements

- Question cards with radio inputs
- Optional text inputs for Other and accessibility follow up
- Progress bar and vertical navigation
- ConsentSidebar for quick access to study summary

Logic

- Questions loaded via useStudyQuestions (demographic)
- Snapshot stored as demographicQuestionsSnapshot
- Bot detection tracks timing
- Under age participants are redirected to a termination screen
- Responses saved via save_demographics

Validation

- Each question is marked complete only if required follow ups are filled
- Questions containing language or accessibility keywords trigger additional input

### 6.4 Pre test (/pre-test)

Purpose

- Capture baseline knowledge

UI elements

- Multiple choice lists with checkboxes
- Progress bar and vertical navigation
- Sticky Continue button with gating

Logic

- Questions loaded via useStudyQuestions (pre_test)
- Snapshot stored as preTestQuestionsSnapshot
- Multiple selections are allowed and stored using |||
- Bot detection collects timing
- Responses saved via save_pre_test

### 6.5 Mode assignment (/mode-assignment)

Purpose

- Select learning mode

UI elements

- Two large cards with mode details
- Loading overlay on selection

Logic

- If preAssignedMode exists, it is auto selected
- If a user attempts to switch modes mid session, the session is reset and blocked
- Mode is stored in sessionStorage and persisted via update_mode

### 6.6 Learning module (/learning/:mode)

Purpose

- Deliver slide based content
- Provide AI tutor support
- Offer optional Playground

Layout

- Header with mode label and Finish button
- SlideViewer center panel
- Tutor panel on right (desktop)
- Playground panel optional (desktop)
- Mobile: chat panel in bottom half

Logic

- Slides loaded from study_slides
- Slide changes update AI tutor context
- Finish button becomes prominent after 60 seconds
- Playground button pulses after 3 seconds to encourage discovery

### 6.7 Post test page 1 (/post-test-1)

Purpose

- Collect perceptions via Likert scales

Logic

- Filters post test questions by category in perception set
- Filters by mode_specific
- For text mode, replaces avatar wording with AI chatbot
- Stores responses in postTestPage1

### 6.8 Post test page 2 (/post-test-2)

Purpose

- Knowledge check after learning

Logic

- Filters post test questions by category = knowledge
- Supports multiple selections
- Stores responses in postTestPage2

### 6.9 Post test page 3 (/post-test-3)

Purpose

- Open feedback collection

Logic

- Filters post test questions by category = open_feedback

- Enforces 10 to 50 characters
- Stores responses in postTestPage3

### 6.10 Completion (/completion)

Purpose

- Final acknowledgement
- Optional participant export
- Lock the study on the device

Logic

- Combines post test responses and saves via save_post_test
- Marks session complete via complete-session
- Sets studyCompleted in sessionStorage and localStorage

### 6.11 Withdrawal and exit

- ExitStudyButton allows participants to withdraw at any time
- Sets session status to withdrawn in study_sessions
- Clears sessionStorage

### 6.12 Inactivity timeout

- 30 minute inactivity timer
- 5 minute warning overlay
- Resets session on timeout and returns to start

## 7. Learning experience deep dive

### 7.1 Slide system and grammar

Slides are stored in study_slides and rendered by SlideViewer. The content supports a custom markdown style grammar to allow rich formatting without HTML.

Supported syntax

- Headings: #, ##, ###
- Bullets: -, *, unicode bullets
- Numbered items: 1. 2. 3.
- Code blocks: ```
- Tables: pipe delimited rows
- Quotes: >
- Inline formatting: **bold**, *italic*, `code`
- Flow lines: -> or -->
- Images: ![alt](url)

Rendering details

- Each list section becomes a card with a rounded border and subtle background
- Headings are promoted into section titles
- Tables are rendered with a styled header row
- Inline code uses a monospaced style with a light background

Sanitization

- HTML output is sanitized via DOMPurify
- Allowed tags: strong, em, code, span
- Allowed attributes: class

### 7.2 SlideViewer layout

- Header with slide index and title
- Decorative corner borders for a presentation look
- Scrollable body area
- Key takeaways panel with numbered bullets
- Navigation buttons and dot indicators
- Automatic scroll to top on slide change

### 7.3 Text mode tutor

Message flow

- A system context message is injected with the current slide title and AI context.
- The chat function uses SSE for streaming responses.
- Messages are appended to the conversation and rendered in a bubble layout.

Behavioral rules

- Short responses (2 to 4 sentences)
- Friendly, conversational tone
- Encourage questions and experimentation
- Strict focus on AI image generation

### 7.4 Avatar mode tutor

Session creation

- Client calls anam-session with slide context.
- Edge function constructs a large system prompt including slide topic.
- The returned session token is used by the Anam SDK.

Runtime behavior

- The avatar is silent unless user speaks.
- Slide changes trigger a reconnection to refresh context.
- Camera auto starts on first connection, then hides after 45 seconds.
- Mic is initially muted until user enables it.

Transcript

- Captures user and avatar utterances
- Filters out JSON and system events
- Records only when mic is on

### 7.5 Image Playground

Features

- Prompt and negative prompt input
- Advanced settings: CFG scale, steps, seed, width, height
- Quick start prompt templates
- Recent images list

Backend constraints

- CFG scale is capped at 15 in the function
- Image dimensions are validated server side
- Errors are mapped to user friendly messages

## 8. Session storage and client side state

sessionStorage is the primary storage for in progress study data. It supports flow guard, export, and transient state.

Key sessionStorage keys

- sessionId
- studyMode
- demographics
- preTest
- postTestPage1
- postTestPage2
- postTestPage3
- demographicQuestionsSnapshot
- preTestQuestionsSnapshot
- postTestQuestionsSnapshot
- studyCompleted
- dialogueLog
- scenarioFeedback

Local storage keys

- studyCompleted

The storage model is intentionally simple but implies that participant exports are only as complete as the current browser state. A reload or cleared storage may produce incomplete local exports even if the database has the full data.

## 9. Admin and owner workflows

The admin panel is the operational control center for the study. Tabs and features are permission gated.

### 9.1 Overview

Metrics

- Total completed and incomplete sessions
- Mode distribution (text, avatar, both)
- Average session duration
- Average avatar time
- Sessions per day

Knowledge gain

- Pre test score calculated as percent correct
- Post test knowledge score calculated as percent correct
- Gain calculated as post minus pre
- Per mode averages

Likert analysis

- Mean, median, std for trust, engagement, satisfaction items
- Distribution counts for 1 to 5 responses
- By mode analysis

Suspicious sessions

- Total suspicious count
- Average suspicion score
- Breakdown by validation status
- Top flags and counts

Exports

- Comprehensive CSV and JSON
- Publication ready PDF
- Section CSVs for demographics, knowledge gain, correlations, and question performance

### 9.2 Sessions

Table

- Session ID, mode, data completeness, timestamps, duration, status, flags, validation

Filters

- Date range, mode, status, validation status, data completeness

Actions

- View session details in a modal
- Export per session PDF
- Bulk accept, request accept, ignore, hide, restore
- Delete sessions (owner only)

Session detail modal

- Demographics (legacy and flexible)
- Pre test responses
- Post test responses
- Dialogue transcripts
- Data quality flags and requirements

### 9.3 Responses

- Aggregated distributions for each question
- Open feedback responses extracted from post test

- CSV export of raw responses

### 9.4 Slides

- Edit slide title, content, key points, AI context
- Add new slides
- Hide or show slides via is_active
- Audit log entries for each change

### 9.5 Questions

- Edit question text, options, correct answers
- Set category and question_meta.type
- Enable or disable questions
- Set mode_specific (text, avatar, both)

### 9.6 API settings

- Master switch (owner only)
- OpenAI switch (owner only)
- Anam switch (admins allowed)
- Anam key updates

### 9.7 My Access

- Permission matrix and descriptions

### 9.8 Activity log

- Admin action history with timestamp and changes

## 10. Data exports and reporting

Exports are generated client side based on current filters and data loaded into the UI.

### 10.1 Participant CSV

- Built on the Completion page
- Category, Question, Response format
- Includes demographics, pre test, post test, feedback, and dialogue

### 10.2 Sessions CSV

- Summary of sessions with flags and duration

### 10.3 Per session PDF

- Session metadata
- Demographics, pre test, post test
- Data quality flags and requirements
- Dialogue transcript

### 10.4 Comprehensive CSV

- One row per session
- Includes all responses and derived scores

- Correctness columns for pre and post knowledge

### 10.5 Comprehensive JSON

- Contains all raw data and computed metrics

### 10.6 Publication PDF

- Contains charts and narrative summary
- Intended for research reporting

## 11. Data quality and validation

### 11.1 Timing rules

- Demographics >= 15s
- Pre test >= 30s
- Post test >= 45s
- Learning >= 3 slides * 8s
- Average slide view >= 8s
- Fast answers ratio < 50 percent
- Average answer time >= 1.5s

### 11.2 Suspicion scoring

- Page too fast: +30
- Fast answer ratio: +25
- Average answer too fast: +20
- Average slide view too fast: +25

### 11.3 Validation states

- pending
- pending_accepted
- pending_ignored
- accepted
- ignored

## 12. Database schema and data model

The following tables are critical for the main flow.

### 12.1 study_sessions

- id (UUID)
- session_id (public UUID)
- mode
- modes_used
- started_at
- completed_at
- status
- suspicion_score
- suspicious_flags

- validation_status
- validated_by
- validated_at

### 12.2 demographic_responses

- id
- session_id
- question_id
- answer
- created_at

### 12.3 pre_test_responses

- id
- session_id
- question_id
- answer
- created_at

### 12.4 post_test_responses

- id
- session_id
- question_id
- answer
- created_at

### 12.5 study_questions

- id
- question_id
- question_text
- options
- correct_answer
- question_type
- category
- question_meta
- allow_multiple
- mode_specific
- is_active
- sort_order

### 12.6 study_questions_public

- View exposing question_text and options only

### 12.7 study_slides

- id
- slide_id
- title

- content
- image_url
- key_points
- system_prompt_context
- sort_order
- is_active

### 12.8 scenarios and dialogue_turns

Legacy scenario tables used by optional routes.

### 12.9 avatar_time_tracking

- id
- session_id
- slide_id
- slide_title
- duration_seconds
- started_at
- ended_at

### 12.10 admin_audit_log

- admin_email
- action_type
- entity_type
- entity_id
- entity_name
- changes
- created_at

### 12.11 app_settings

- key
- value
- updated_at
- updated_by

### 12.12 user_roles and admin_users

- user_roles: user_id, role
- admin_users: email list

## 13. Edge functions and backend services

### 13.1 save-study-data

Actions

- create_session
- save_demographics
- save_pre_test

- save_post_test
- save_scenario
- update_mode
- reset_session
- update_activity
- report_suspicious

### 13.2 complete-session

- Marks session completed and sets completed_at

### 13.3 chat

- Validates API switches
- Sends request to OpenAI API (GPT-5 mini)
- Streams responses

### 13.4 anam-session

- Validates API switches
- Uses Anam API key from app_settings or env
- Returns session token

### 13.5 generate-image

- Validates prompt, negative prompt, CFG, steps, seed, size
- Returns generated image URL

### 13.6 toggle-api

- Role based toggle of API settings
- Logs changes to admin_audit_log

### 13.7 update-session-validation

- Owner can accept, ignore, or delete sessions
- Admin can request acceptance or ignore

### 13.8 save-avatar-time

- Stores per slide time tracking entries

### 13.9 create-admin-users

- Provisioning endpoint with secrets

## 14. AI integrations and prompt policy

### 14.1 Text mode

- System prompt defines Alex persona
- Short responses and topic focus
- Pre test results used to adapt explanations

### 14.2 Avatar mode

- Strict topic boundaries
- Slide context embedded in system prompt
- Silent system events for camera and mic

### 14.3 Playground

- Prompt includes dimension hints and negative prompt

## 15. Security and privacy

- No PII is collected for participants
- Session IDs are random UUIDs
- Consent text states no video recording
- Admin auth via Supabase Auth
- API keys stored in app_settings, masked for non owners

## 16. Reliability, error handling, and recovery

- useStudyFlowGuard enforces order and redirects
- Session timeout resets after inactivity
- Mode switching resets sessions
- Errors show toasts and allow retry

## 17. Performance and scalability

- Streaming chat reduces latency
- Supabase functions offload writes
- Admin dashboards are paginated and filtered

## 18. Content operations and maintenance

- Admins can edit slides and questions live
- Changes are instant and recorded in audit logs
- Stable question IDs are required for long term comparisons

## 19. Testing and QA posture

- Vitest tests cover selected components
- Manual testing is required for full flow
- Recommended: run full text and avatar studies after each release

## 20. Known limitations and legacy paths

- Scenario routes exist but are not in the main flow
- Voice mode exists in database but not exposed
- Participant CSV relies on sessionStorage
- Repeat blocking is local to device

## 21. Roadmap and recommended improvements

- Add server side participant export
- Add versioning for questions and slides

- Add automated E2E tests
- Improve repeat participation controls if required

## 22. Glossary

- Session: a single participant run
- Pre test: baseline knowledge assessment
- Post test: after learning assessment
- Likert: 1 to 5 rating scale
- Open feedback: short text responses
- Suspicion score: data quality score

## Appendix A: Route map

- /: Welcome
- /study/:mode: Study entry
- /consent: Consent
- /demographics: Demographics
- /pre-test: Pre test
- /mode-assignment: Mode selection
- /learning/:mode: Learning
- /post-test-1: Post test page 1
- /post-test-2: Post test page 2
- /post-test-3: Post test page 3
- /completion: Completion
- /admin/login: Admin login
- /admin: Admin dashboard
- /scenario/:mode/:scenarioId: Legacy scenario

## Appendix B: Client side storage keys

- sessionId
- studyMode
- demographics
- preTest
- postTestPage1
- postTestPage2
- postTestPage3
- demographicQuestionsSnapshot
- preTestQuestionsSnapshot
- postTestQuestionsSnapshot
- studyCompleted
- dialogueLog
- scenarioFeedback

## Appendix C: Export schema details

Participant CSV

- Category, Question, Response

Admin sessions CSV

- Session ID, Mode, Started, Completed, Duration, Status, Flags

Comprehensive CSV

- Session metadata
- Demo columns
- Pre test columns
- Post test columns
- Correctness columns

## Appendix D: Data quality rules and requirements

Rules

- Demographics >= 15s
- Pre test >= 30s
- Post test >= 45s
- Learning >= 24s
- Average slide view >= 8s
- Fast answer ratio < 50 percent
- Average answer >= 1.5s

Suspicion score bands

- 0 to 19: normal
- 20 to 39: low risk
- 40 to 59: medium risk
- 60 or more: high risk

## Appendix E: Edge function request and response examples

Example: create session

Request body

- action: create_session
- mode: text

Response

- success: true
- sessionId: UUID

Example: save demographics

Request body

- action: save_demographics
- sessionId: UUID
- demographics: { question_id: answer }

Example: save post test

Request body

- action: save_post_test
- sessionId: UUID
- postTestResponses: [{ questionId, answer }]

## Appendix F: Analytics formulas and metrics

Knowledge gain

- Pre score = correct_pre / total_pre
- Post score = correct_post / total_post
- Gain = post score - pre score

Mode counts

- Modes used derived from modes_used array or mode field
- both = text and avatar present

## Appendix G: Detailed page specifications

This appendix provides a detailed, page by page specification. Each page includes purpose, UI elements, state dependencies, data writes, and failure modes. The intent is to provide an operational view that can be used by developers, testers, or auditors.

### G1. Welcome page (/)

Purpose

- Provide a public entry point with a concise overview of the study.
- Reinforce privacy and anonymity and the estimated time (~20 minutes).
- Prevent repeat participation on the same device.

UI layout

- A glassmorphic header with the TUM logo.
- A large hero title and summary paragraphs.
- A feature grid that highlights time, Playground, interaction format, and anonymity.
- A CTA button to start the consent flow.
- Footer with project title and mentor name.

State dependencies

- localStorage.studyCompleted is checked on load.

State changes

- If CTA is clicked, resetStudyState clears sessionStorage and localStorage keys, then navigates to /consent.

Failure and edge cases

- If studyCompleted is set, CTA is disabled and a warning message appears.
- A reset query parameter (?reset=1) clears state and unlocks the study.

### G2. StudyEntry (/study/:mode)

Purpose

- Support distinct study links that assign mode in advance.

UI layout

- Similar layout to Welcome, but includes a mode badge indicating Text Mode or Avatar Mode.
- Contains a CTA that starts the consent flow with the pre assigned mode.

State dependencies

- URL param mode must be text or avatar.
- localStorage.studyCompleted is checked to prevent repeat participation.

State changes

- Stores preAssignedMode in sessionStorage for later use in ModeAssignment.
- Reset query parameter clears all study state.

Failure and edge cases

- If mode is invalid, an error page is shown with no CTA.
- If localStorage.studyCompleted is true, CTA is disabled.

### G3. Consent (/consent)

Purpose

- Present informed consent and allow the participant to opt in.

UI layout

- Consent sections: purpose, procedures, data collection, risks, voluntary participation.
- Checkbox that confirms consent and age >= 18.
- Continue and Decline buttons.

State dependencies

- None, this is the entry to the guarded flow.

State changes

- On consent, create_session is invoked with a default mode of text.
- A new sessionId is stored in sessionStorage.

Data writes

- Edge function save-study-data with action create_session.

Failure and edge cases

- If session creation fails, a toast error is displayed and the user can retry.

### G4. Demographics (/demographics)

Purpose

- Collect demographic data for segmentation and analysis.

UI layout

- Header with progress summary and Exit Study button.
- ConsentSidebar button pinned to the bottom right.
- VerticalProgressBar to show completion status and enable navigation.
- Question cards with radio options and optional text inputs.

State dependencies

- Requires sessionId from the consent step.
- useStudyFlowGuard enforces this requirement.

State changes

- Saves responses to sessionStorage.demographics.
- Saves a snapshot of question metadata to demographicQuestionsSnapshot.

Data writes

- save_demographics action writes to demographic_responses.

Validation

- Each question is considered complete if:
- A primary answer is selected.
- If answer is Other, an additional text input is provided.
- If answer is Yes for accessibility, a follow up text is provided.

Failure and edge cases

- Under age participants (age < 18) are redirected to a termination screen.
- If saving fails, a toast error is displayed and the participant stays on the page.

### G5. Pre test (/pre-test)

Purpose

- Capture baseline knowledge before learning.

UI layout

- Progress bar with completion percentage.
- VerticalProgressBar for navigation.
- Question cards with checkbox answers.

State dependencies

- Requires demographics to be completed.

State changes

- Saves responses to sessionStorage.preTest.
- Saves a snapshot of question metadata to preTestQuestionsSnapshot.

Data writes

- save_pre_test action writes to pre_test_responses.

Validation

- All questions must have at least one selected answer.
- Multiple selections are stored as answer1|||answer2.

Failure and edge cases

- If save fails, a toast error is displayed and the participant can retry.

### G6. Mode assignment (/mode-assignment)

Purpose

- Allow the participant to choose text or avatar mode.

UI layout

- Two mode cards with icon, title, and feature list.

State dependencies

- Requires preTest completion and sessionId.

State changes

- Writes selected mode to sessionStorage.studyMode.
- Clears preAssignedMode if it was used.

Data writes

- save-study-data action update_mode updates study_sessions.mode.

Failure and edge cases

- Attempted mode switching after selection triggers reset_session with reason mode_switch.
- Reset clears sessionStorage and returns to /.

### G7. Learning module (/learning/:mode)

Purpose

- Deliver slide content and AI tutor support.

UI layout

- Fixed header with mode badge and Finish button.
- SlideViewer centered with scrollable content.
- TextModeChat or AvatarModePanel on the right (desktop).
- Playground panel toggled from a right edge button.
- On mobile, chat is a fixed bottom panel.

State dependencies

- Requires sessionId, demographics, preTest, and studyMode.

State changes

- Slide changes update currentSlide in component state.

Data writes

- Avatar mode: save-avatar-time called on slide change and unmount.

Failure and edge cases

- If slides fail to load, a retry button is displayed.
- Avatar mode is not rendered on mobile to prevent multiple Anam clients.

### G8. Post test page 1 (/post-test-1)

Purpose

- Capture Likert responses about trust, engagement, satisfaction, expectations, realism, and avatar qualities.

UI layout

- Section headers for each category.
- LikertScale component per question.
- Progress bar and vertical navigation.

State dependencies

- Requires completion of learning module.

State changes

- Writes responses to sessionStorage.postTestPage1.
- Snapshot stored in postTestQuestionsSnapshot.

Validation

- All Likert questions must have a selected value (1 to 5).

Failure and edge cases

- If questions fail to load, the page shows an error and retry option.

### G9. Post test page 2 (/post-test-2)

Purpose

- Capture post learning knowledge.

UI layout

- Checkbox based multi selection questions.
- Progress bar and vertical navigation.

State dependencies

- Requires post test page 1 completion.

State changes

- Writes responses to sessionStorage.postTestPage2.

Validation

- All knowledge questions must have at least one answer.

### G10. Post test page 3 (/post-test-3)

Purpose

- Capture open feedback about the learning experience.

UI layout

- Textarea per open feedback question.
- Character counter and minimum indicator.
- Progress bar and vertical navigation.

State dependencies

- Requires post test page 2 completion.

State changes

- Writes responses to sessionStorage.postTestPage3.

Validation

- Each answer must be between 10 and 50 characters.
- If no open feedback questions exist, the page is auto complete.

Data writes

- save_post_test writes post test responses to post_test_responses.
- complete-session marks study_sessions as completed.

### G11. Completion (/completion)

Purpose

- Thank the participant and provide a personal data export.

UI layout

- Thank you hero, summary of what was learned, optional export button.

State changes

- Sets studyCompleted in sessionStorage and localStorage.
- Clears post test data in sessionStorage.

Data writes

- None (data already written in PostTestPage3).

### G12. Admin login (/admin/login)

Purpose

- Authenticate admins and owners using Supabase Auth.

UI layout

- Email and password login form

State changes

- Supabase Auth session stored in localStorage.

### G13. Admin dashboard (/admin)

Purpose

- Provide analysis, content management, validation, and exports.

UI layout

- Header with role badge and user email
- Tabs for Overview, Sessions, Responses, Slides, Questions, API Settings, My Access, Activity Log

State changes

- Realtime channels update data when other admins change content.

## Appendix H: Data dictionary (full)

This appendix lists the full set of tables and columns as defined in src/integrations/supabase/types.ts. Descriptions are based on usage in the application.

### H1. admin_audit_log

Purpose: audit trail of admin actions.

Columns

- id: UUID primary key
- admin_email: user email
- action_type: create, update, delete, toggle
- entity_type: slide, question, setting, api_setting
- entity_id: identifier of the entity
- entity_name: display name
- changes: JSON diff of changed fields
- created_at: timestamp

### H2. admin_users

Purpose: list of admin emails used by validation endpoints.

Columns

- id
- email
- created_at

### H3. app_settings

Purpose: key value store for runtime settings.

Columns

- id
- key: api_enabled, openai_api_enabled, anam_api_enabled, anam_api_key
- value: JSON payload

- updated_at
- updated_by

### H4. avatar_time_tracking

Purpose: record time spent on each slide in avatar mode.

Columns

- id
- session_id (FK to study_sessions.id)
- slide_id
- slide_title
- started_at
- ended_at
- duration_seconds
- created_at

### H5. demographic_responses

Purpose: flexible demographic responses with question IDs.

Columns

- id
- session_id
- question_id
- answer
- created_at

### H6. demographics (legacy)

Purpose: legacy demographics table with fixed columns.

Columns

- id
- session_id
- age_range
- education
- digital_experience
- created_at

### H7. dialogue_turns

Purpose: store scenario dialogue messages.

Columns

- id
- scenario_id (FK to scenarios.id)
- role
- content
- timestamp

- created_at

### H8. post_test_responses

Purpose: post test responses for all categories.

Columns

- id
- session_id
- question_id
- answer
- created_at

### H9. pre_test_responses

Purpose: pre test responses.

Columns

- id
- session_id
- question_id
- answer
- created_at

### H10. question_change_requests

Purpose: queue for future question change review.

Columns

- id
- question_id
- change_type
- proposed_changes
- requested_by
- requested_at
- reviewed_by
- reviewed_at
- review_notes
- status

### H11. scenarios

Purpose: legacy scenario based learning data.

Columns

- id
- session_id
- scenario_id
- generated_images
- confidence_rating

- trust_rating
- engagement_rating
- completed_at
- created_at

### H12. study_questions

Purpose: source of truth for all questions.

Columns

- id
- question_id
- question_text
- question_type
- options
- correct_answer
- category
- question_meta
- allow_multiple
- mode_specific
- sort_order
- is_active
- created_at
- updated_at
- created_by
- approved_by
- approved_at
- pending_changes

### H13. study_sessions

Purpose: central session record.

Columns

- id
- session_id
- mode
- modes_used
- status
- started_at
- completed_at
- created_at
- last_activity_at
- suspicion_score
- suspicious_flags
- validation_status
- validated_by
- validated_at

- browser_fingerprint (currently unused)

### H14. study_slides

Purpose: source of truth for slides.

Columns

- id
- slide_id
- title
- content
- image_url
- key_points
- system_prompt_context
- sort_order
- is_active
- created_at
- updated_at

### H15. user_roles

Purpose: map Supabase auth users to roles.

Columns

- id
- user_id
- role
- created_at

### H16. study_questions_public (view)

Purpose: participant facing question view without correct answers.

Columns

- id
- question_id
- question_text
- question_type
- category
- question_meta
- options
- allow_multiple
- sort_order
- is_active

## Appendix I: UI component reference

### I1. SlideViewer

Purpose: render a single learning slide with navigation.

Key props

- slides: array of slide objects
- currentSlide: slide object to render
- onSlideChange: callback when next or previous is selected

Behavior

- Scrolls to top on slide change
- Renders formatted content using the custom grammar
- Displays header with slide number and title
- Shows key takeaways panel if key points exist

### I2. LikertScale

Purpose: capture Likert ratings in post test.

Behavior

- Desktop: horizontal 5 option grid
- Mobile: vertical list for better tapping
- Values stored as strings 1 to 5

### I3. VerticalProgressBar

Purpose: show question completion status and allow navigation.

Behavior

- Displays a dot for each question
- Completed answers are highlighted
- Clicking jumps to question position

### I4. ConsentSidebar

Purpose: provide quick access to consent text during the study.

Behavior

- Fixed button opens a right side sheet
- Scrollable consent content

### I5. ExitStudyButton

Purpose: allow participants to withdraw from the study.

Behavior

- Opens a confirmation dialog
- On confirmation, sets session status to withdrawn and clears sessionStorage

### I6. TextModeChat

Purpose: text based AI tutor.

Behavior

- Maintains local chat history
- Streams responses from chat edge function
- Inserts slide context system message

### I7. AvatarModePanel

Purpose: avatar tutor experience.

Behavior

- Initializes Anam client
- Shows avatar video and transcript
- Toggles mic and camera
- Tracks slide time and saves via save-avatar-time

### I8. TranscriptPanel

Purpose: show user and avatar messages in avatar mode.

Behavior

- Updates in real time
- Distinguishes user vs avatar messages

### I9. ImagePlayground

Purpose: optional image generation tool.

Behavior

- Supports prompt input and advanced parameters
- Sends requests to generate-image
- Displays recent results

### I10. AdminSessions table

Purpose: detailed session management UI.

Behavior

- Filtering, sorting, pagination
- Suspicion indicators
- Validation actions

### I11. AdminOverview charts

Purpose: analytics dashboard.

Behavior

- Multiple charts and metrics
- Export actions for CSV and PDF

## Appendix J: Edge function contracts (expanded)

### J1. save-study-data

Endpoint

- POST /functions/v1/save-study-data

Common headers

- Content-Type: application/json
- Authorization: Bearer VITE_SUPABASE_PUBLISHABLE_KEY

Actions

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

Example: create_session

Request

- action: create_session
- mode: text

Response

- success: true
- sessionId: <uuid>

Example: save_pre_test

Request

- action: save_pre_test
- sessionId: <uuid>
- preTestResponses: [{ questionId, answer }]

Response

- success: true

### J2. complete-session

Endpoint

- POST /functions/v1/complete-session

Request

- sessionId: <uuid>

Response

- success: true

### J3. chat

Endpoint

- POST /functions/v1/chat

Request

- messages: [{ role, content }]
- preTestData: { questionId: answer }

Response

- SSE stream of chat completion chunks

### J4. anam-session

Endpoint

- POST /functions/v1/anam-session

Request

- slideContext: { id, title, keyPoints, systemPromptContext }

Response

- sessionToken

### J5. generate-image

Endpoint

- POST /functions/v1/generate-image

Request

- prompt
- negativePrompt
- cfgScale
- steps
- seed
- width
- height

Response

- imageUrl

### J6. toggle-api

Endpoint

- POST /functions/v1/toggle-api

Actions

- get
- toggle

- update_key

Response

- settings object and user info for get
- success messages for updates

### J7. update-session-validation

Endpoint

- POST /functions/v1/update-session-validation

Request

- sessionId or sessionIds
- status or approve

Response

- success and updated status

### J8. save-avatar-time

Endpoint

- POST /functions/v1/save-avatar-time

Request

- sessionId
- slideId
- slideTitle
- startedAt
- endedAt
- durationSeconds

Response

- success, id

### J9. create-admin-users

Endpoint

- POST /functions/v1/create-admin-users

Request

- secret

Response

- list of created or updated users

## Appendix K: Admin analytics and calculations

This appendix describes how key metrics in AdminOverview are computed.

### K1. Session filtering

- Reset sessions (status = reset) are excluded from analysis.
- Ignored sessions (validation_status = ignored) are excluded.
- Suspicious sessions are excluded unless validation_status = accepted.

### K2. Knowledge scoring

- Pre test scores are computed from pre_test_responses where correct_answer is set.
- Post test knowledge scores are computed from post_test_responses with question_id starting with knowledge-.
- Multi answer correctness requires exact set match (all correct answers selected, no extra answers).

### K3. Knowledge gain

- Pre score = correct_pre / total_pre * 100
- Post score = correct_post / total_post * 100
- Gain = post score - pre score

### K4. Mode distribution

- modes_used is used if present, otherwise mode
- If both text and avatar appear, the session is classified as both

### K5. Avatar time

- Avatar time is aggregated from avatar_time_tracking by session_id.
- Average avatar time is total time / number of sessions with avatar time.

### K6. Demographics

- Age breakdown is computed from demo-age responses.
- Education breakdown is computed from demo-education.
- Digital experience breakdown from demo-digital-experience.

### K7. Likert analysis

- Responses are parsed as integers 1 to 5.
- Mean, median, and standard deviation are computed per item.
- Likert analysis is computed for trust, engagement, and satisfaction.

### K8. Correlation analysis

- Avatar time vs knowledge gain (x = avatar minutes, y = gain).
- Session duration vs knowledge gain (x = duration minutes, y = gain).

### K9. Statistical tests

- Welch's t-test is used to compare mean gains between modes.
- Cohen's d effect size is computed for differences.
- Confidence intervals are reported when sufficient data exists.

## Appendix L: Export field maps and interpretation notes

### L1. Participant CSV

- Category: logical group
- Question: question text from snapshots or DB fallback
- Response: raw response string (multi answers separated by '; ')

### L2. Sessions CSV

- Status: completed, incomplete, reset
- Flags: suspicious flags joined by '; '

### L3. Comprehensive CSV

- PRE: question text for pre test
- PRE_CORRECT: Yes or No based on exact match to correct_answer
- POST: question text for post test
- POST_CORRECT: Yes or No based on exact match to correct_answer
- If no correct_answer exists, PRE_CORRECT and POST_CORRECT show N/A

### L4. JSON export

- Contains all raw response arrays
- Contains summary metrics and per session aggregates
- Suitable for archival or analysis pipelines

## Appendix M: Slide formatting guide and examples

This appendix provides a reference for the slide content formatting system. It is intended for owners and admins who edit slide content in the Slides tab.

### M1. Overview of the formatting grammar

Slide content uses a markdown like grammar implemented in SlideViewer. The parser is not a full markdown engine; it recognizes specific patterns and renders them into styled blocks. Unsupported HTML is sanitized and stripped.

Supported constructs

- Headings: #, ##, ###
- Bullet lists: -, *, or unicode bullets
- Numbered lists: 1. 2. 3.
- Code blocks: ```
- Tables: pipe delimited rows
- Quotes: >
- Inline formatting: **bold**, *italic*, `code`
- Flow lines: -> or -->
- Images: ![alt](url)
- Horizontal rules: --- or ***

### M2. Headings

Headings mark a new section within a slide. They are rendered with a colored left bar and larger font size.

Example

# Title of the section

## Subsection title

### Small section title

Notes

- A heading line should start at the beginning of the line.
- A heading becomes the title for the next text or list block.

### M3. Bullet lists

Bullet items are rendered in a card with a muted background and a bullet icon.

Example

- This is a bullet item
- This is another bullet item

Notes

- The parser recognizes - or * at the start of the line.
- Each bullet item becomes a list item in the card.

### M4. Numbered lists

Numbered items are rendered in a card with numeric badges.

Example

1. First step
2. Second step
3. Third step

Notes

- The parser recognizes digits followed by a dot and space.
- The number displayed is auto generated and does not need to match the input.

### M5. Code blocks

Code blocks are displayed in a monospaced style with a dark background.

Example

```
Prompt: "a sunset over mountains"
Negative: "blurry, low quality"
```

Notes

- Code blocks start and end with ``` on separate lines.
- The code block preserves line breaks.

### M6. Tables

Tables are rendered when a series of lines starts with a pipe character. The first row is treated as a header.

Example

| Category | Examples |
| -------- | -------- |
| Lighting | golden hour, rim light |
| Style | watercolor, oil painting |

Notes

- The separator row (| ---- |) is optional but recommended.
- Each row becomes a table row in the UI.

### M7. Quotes

Lines beginning with > are rendered as a callout block.

Example

> *This is a quote or highlighted remark.*

### M8. Inline formatting

Inline formatting applies inside text paragraphs.

Examples

- **Bold** becomes emphasized text.
- *Italic* becomes emphasized text.
- `code` becomes monospaced inline code.

Notes

- Nested or mixed formatting is limited; keep formatting simple.
- Any HTML tags you type will be sanitized and removed.

### M9. Flow lines

Lines containing -> or --> are rendered as a centered formula style row.

Example

Text Prompt --> AI Model --> Image Output

### M10. Images

Images can be embedded with markdown image syntax.

Example

![Landscape example](https://example.com/landscape.png)

Notes

- The image URL must be publicly accessible.
- Images are rendered with a maximum height to avoid layout overflow.

### M11. Spacing and blank lines

Blank lines create spacing between sections. Use blank lines to separate logical blocks.

### M12. Key points and AI context

In addition to content, each slide has two separate fields in the database:

- key_points: a list of short takeaways shown in a highlighted panel
- system_prompt_context: a summary provided to the AI tutor

These should align with the slide content but be concise. The AI tutor uses system_prompt_context as the highest priority context for the current slide.

## Appendix N: Session state machine and status transitions

This appendix describes the lifecycle of a study session and the states used to track progress and data quality.

### N1. Session creation

- The session is created when the participant consents.
- create_session inserts into study_sessions with session_id and mode.
- modes_used is initialized with the default mode.

### N2. Active session

During the study, the session remains active. No explicit status indicates in progress, but the absence of completed_at implies the session is incomplete.

### N3. Completion

- On PostTestPage3, save_post_test writes post test responses.
- complete-session sets completed_at and status = completed.
- studyCompleted is set in localStorage and sessionStorage.

### N4. Withdrawal

- ExitStudyButton sets status = withdrawn and clears completion markers.
- Withdrawn sessions are treated as incomplete and excluded from analysis.

### N5. Reset

Reset occurs when a participant attempts to switch modes or when a session times out.

- reset_session sets status = reset and may include a reason.
- The session is excluded from analysis by default.

### N6. Validation statuses

Suspicious sessions are marked with a suspicion_score and suspicious_flags. Validation status controls whether the session is included in analysis.

Validation values

- pending: default
- pending_accepted: admin requested acceptance
- pending_ignored: admin requested ignore
- accepted: owner approved

- ignored: owner ignored

### N7. State transition summary

- create_session -> incomplete
- incomplete -> completed (complete-session)
- incomplete -> withdrawn (ExitStudyButton)
- incomplete -> reset (timeout or mode switch)
- suspicious -> pending_accepted or pending_ignored (admin request)
- pending_* -> accepted or ignored (owner approval)

## Appendix O: Data flow narratives

This appendix provides narrative descriptions of how data travels through the system for each major data category.

### O1. Session creation

Flow

1. Participant consents.
2. Client calls save-study-data with action create_session.
3. Edge function inserts study_sessions record.
4. sessionId returned to the client and stored in sessionStorage.

### O2. Demographics responses

Flow

1. Questions loaded from study_questions_public.
2. Participant selects answers and optional text.
3. Responses normalized and stored in sessionStorage.
4. save_demographics sends question_id and answer pairs.
5. Edge function inserts rows into demographic_responses.

### O3. Pre test responses

Flow

1. Participant answers pre test questions.
2. Responses stored with ||| separators for multi choice.
3. save_pre_test converts responses to array of questionId/answer objects.
4. Edge function inserts rows into pre_test_responses.

### O4. Post test responses

Flow

1. Page 1 stores Likert responses in postTestPage1.
2. Page 2 stores knowledge responses in postTestPage2.
3. Page 3 stores open feedback in postTestPage3.
4. On completion, responses are merged and sent to save_post_test.
5. Edge function inserts rows into post_test_responses.

### O5. Completion state

Flow

1. complete-session updates study_sessions status and completed_at.
2. studyCompleted stored locally to block repeats.

### O6. Avatar time tracking

Flow

1. AvatarModePanel tracks slide entry time.
2. On slide change or unmount, save-avatar-time is called.
3. Edge function inserts a row into avatar_time_tracking.

### O7. Suspicious activity

Flow

1. useBotDetection captures timing data on each page.
2. On completion of a page, analyzeTimingData creates flags and a score.
3. logSuspiciousActivity sends report_suspicious to save-study-data.
4. Edge function updates study_sessions with flags and suspicion_score.

### O8. Admin content changes

Flow

1. Admin edits slides or questions in the dashboard.
2. Updates are written directly to study_slides or study_questions.
3. logAdminAction writes an audit entry to admin_audit_log.
4. Realtime subscriptions update other admin clients.

### O9. API settings changes

Flow

1. Admin toggles Anam API or updates Anam key.
2. toggle-api edge function validates the user and updates app_settings.
3. Changes are recorded in admin_audit_log.

## Appendix P: Admin operational playbooks

This appendix provides step by step procedures for common admin tasks.

### P1. Review data quality alerts

1. Open the Admin dashboard.
2. Go to the Sessions tab.
3. Filter by status = suspicious or validation = pending.
4. Click the score indicator to review flags.
5. Open the session detail modal for context.
6. Decide to accept or ignore.

### P2. Request acceptance of suspicious sessions (admin)

1. Select one or more sessions in the Sessions table.
2. Click Request Accept.
3. Status changes to pending_accepted.
4. Notify the owner for final approval.

### P3. Approve acceptance (owner)

1. Filter Sessions by validation = pending.
2. Open the session detail modal if needed.
3. Click Approve Accept or Approve Ignore.
4. Status changes to accepted or ignored.

### P4. Update slide content

1. Go to Slides tab.
2. Expand the slide in the accordion.
3. Click Edit.
4. Modify title, content, key points, and AI context.
5. Save and verify changes in preview.

### P5. Update questions and correct answers

1. Go to Questions tab.
2. Select the question to edit.
3. Update text and options.
4. Mark correct answers (multi select allowed).
5. Save and verify in preview mode.

### P6. Update Anam API key

1. Go to API Settings.
2. Enter the new key in the Anam API Key field.
3. Click Save.
4. Verify that the status banner shows the new key suffix.

### P7. Export study data

1. Go to Overview tab.
2. Choose date range and status filter.
3. Export comprehensive CSV or JSON.
4. For per session PDF, use Sessions tab.

### P8. Generate publication PDF

1. Go to Overview tab.
2. Click Export Publication Report.
3. Review the PDF output for charts and summary metrics.

## Appendix Q: QA test scripts and checklists

This appendix provides manual QA scripts to verify system correctness.

### Q1. Participant flow test (text mode)

1. Navigate to /study/text.
2. Accept consent.
3. Complete demographics and verify progress counts.
4. Complete pre test with at least one multi answer question.
5. Select Text mode.
6. Navigate through slides and interact with text tutor.
7. Open Playground and generate one image.
8. Complete post test pages 1 to 3.
9. Verify completion page and export CSV.
10. Confirm localStorage studyCompleted is set.

Expected

- All pages progress only after required answers.
- CSV export contains all answered questions.
- Chat responses stream correctly.

### Q2. Participant flow test (avatar mode)

1. Navigate to /study/avatar.
2. Accept consent and complete demographics and pre test.
3. Select Avatar mode.
4. Verify avatar loads and microphone toggles.
5. Speak a short question and verify transcript captures it.
6. Navigate slides and confirm avatar context changes.
7. Complete post test pages and finish.

Expected

- Avatar session token is created.
- Mic toggle controls user transcript.
- Avatar remains on topic.

### Q3. Data quality detection test

1. Complete demographics and pre test as quickly as possible.
2. Finish the study in under 1 minute.
3. Open Sessions tab and verify suspicious flags appear.

Expected

- Suspicion score > 0.
- Flags mention fast page completion and fast answers.

### Q4. Admin permissions test

1. Login as viewer.
2. Verify Slides and Questions tabs are hidden.
3. Verify export buttons are hidden.
4. Login as admin and verify content editing is possible.
5. Login as owner and verify delete actions are available.

### Q5. API settings test

1. As admin, toggle Anam API and verify status change.
2. As owner, toggle master API and OpenAI API.
3. Verify AI services respond with service unavailable when disabled.

### Q6. Export validation test

1. In Sessions tab, export CSV and verify columns.
2. In Overview, export comprehensive CSV and JSON.
3. Open a session detail and export PDF.
4. Verify that responses match the database.

### Q7. Repeat participation test

1. Complete the study once.
2. Return to Welcome and verify CTA is disabled.
3. Use reset query to clear state and start again.

## Appendix R: Codebase map and file reference

This appendix maps the most important files in the repository to their purpose. It is intended to help developers or reviewers quickly locate functionality.

### R1. Application entry and routing

- src/main.tsx: React entry point, mounts App and loads global CSS.
- src/App.tsx: main route table for all study and admin routes.
- src/index.css: global styles and Tailwind base layers.

### R2. Pages (participant flow)

- src/pages/Welcome.tsx: welcome and entry CTA with repeat blocking.
- src/pages/StudyEntry.tsx: mode specific entry with preAssignedMode.
- src/pages/Consent.tsx: consent form and session creation.
- src/pages/Demographics.tsx: demographic questionnaire and age gate.
- src/pages/PreTest.tsx: baseline knowledge assessment.
- src/pages/ModeAssignment.tsx: mode selection and lock.
- src/pages/Learning.tsx: slides plus tutor and Playground.
- src/pages/PostTestPage1.tsx: Likert perception items.
- src/pages/PostTestPage2.tsx: knowledge check items.
- src/pages/PostTestPage3.tsx: open feedback and completion.
- src/pages/Completion.tsx: thank you page and participant CSV export.
- src/pages/Scenario.tsx: legacy scenario flow (not used in main path).
- src/pages/ScenarioFeedback.tsx: legacy scenario feedback.
- src/pages/AdminLogin.tsx: admin login page.
- src/pages/AdminDashboard.tsx: admin interface shell.

### R3. Core participant components

- src/components/SlideViewer.tsx: slide rendering and navigation.
- src/components/VerticalProgressBar.tsx: question navigator for surveys.
- src/components/LikertScale.tsx: Likert scale UI for desktop and mobile.

- src/components/ImagePlayground.tsx: image generation tool.
- src/components/ConsentSidebar.tsx: quick access consent panel.
- src/components/ParticipantFooter.tsx: study footer for participant pages.
- src/components/ExitStudyButton.tsx: withdrawal control.

### R4. Tutor mode components

- src/components/modes/TextModeChat.tsx: text mode chat UI and streaming.
- src/components/modes/AvatarModePanel.tsx: avatar mode panel with video, mic, transcript.
- src/components/TranscriptPanel.tsx: transcript viewer for avatar mode.

### R5. Admin components

- src/components/admin/AdminOverview.tsx: analytics dashboard and exports.
- src/components/admin/AdminSessions.tsx: session list, validation, per session export.
- src/components/admin/AdminResponses.tsx: aggregated response analytics.
- src/components/admin/AdminSlides.tsx: slide management.
- src/components/admin/AdminQuestions.tsx: question management.
- src/components/admin/ApiToggle.tsx: API settings UI.
- src/components/admin/AdminAuditLog.tsx: audit log display.
- src/components/admin/PermissionsInfo.tsx: permission matrix display.

### R6. Hooks

- src/hooks/useStudyFlowGuard.ts: flow guard and session reset helpers.
- src/hooks/useStudyQuestions.ts: fetch questions from study_questions_public or study_questions.
- src/hooks/useStudySlides.ts: fetch slide data from study_slides.
- src/hooks/useBotDetection.ts: timing based data quality detection.
- src/hooks/useSessionTimeout.ts: inactivity timeout and reset.
- src/hooks/useAnamClient.ts: Anam SDK integration and transcript management.

### R7. Lib and utils

- src/lib/permissions.ts: role based permission config.
- src/lib/studyData.ts: client side wrappers for edge functions.
- src/lib/auditLog.ts: admin audit logging helpers.
- src/lib/suspicion.ts: suspicion thresholds and rule descriptors.
- src/utils/aiChat.ts: SSE chat streaming parser.
- src/utils/generateSystemDocumentation.ts: legacy system docs generator.

### R8. Supabase functions

- supabase/functions/save-study-data: core persistence actions.
- supabase/functions/complete-session: mark completion.
- supabase/functions/chat: text tutor gateway.
- supabase/functions/anam-session: avatar session creation.
- supabase/functions/generate-image: Playground integration.
- supabase/functions/toggle-api: API settings control.
- supabase/functions/update-session-validation: validation workflow.
- supabase/functions/save-avatar-time: avatar slide time tracking.
- supabase/functions/create-admin-users: admin provisioning.

## Appendix S: Prompt policy details and tutor behavior

This appendix provides an expanded view of the tutor behavior, based on system prompts used in text and avatar modes. The content is summarized, not quoted verbatim, but reflects the intent and structure of the prompts.

### S1. Text mode tutor policy

Persona

- Name: Alex
- Role: friendly AI tutor focused on AI image generation
- Tone: conversational, warm, and encouraging

Behavior constraints

- Responses are short (2 to 4 sentences).
- Avoid overly formal language.
- Provide concrete examples and analogies.
- Encourage follow up questions.
- Avoid technical jargon unless explained.

Scope constraints

- Topic scope: AI image generation only.
- Avoid discussion of unrelated topics.
- Ethical reminders included when relevant.

Knowledge adaptation

- Pre test data is evaluated against correct answers.
- Weak areas are highlighted in the system context.
- The tutor provides extra clarification on weak topics.

### S2. Avatar mode tutor policy

Persona

- Same Alex persona but with additional constraints for video and audio interaction.
- Strong emphasis on being a tutor rather than a general chat assistant.

Topic boundaries

- Strictly limited to AI image generation topics.
- Multi stage redirection for off topic attempts.
- Escalation from gentle to firm redirection if needed.

Camera and mic behavior

- The avatar must not admit lack of camera vision.
- Responses to camera questions are vague and redirect back to the topic.
- Camera and mic toggles trigger silent or short acknowledgments.

Slide context handling

- Slide context is included in the system prompt when the session is created.

- On slide change, the client reconnects to refresh the context.
- The avatar is instructed to stay silent on system context updates.

### S3. Differences between text and avatar modes

- Text mode can use longer chat history; avatar mode focuses on immediate interaction.
- Text mode inserts slide context as a system message in the conversation.
- Avatar mode uses a large system prompt, with slide context injected directly.
- Avatar mode treats camera and mic toggles as system events.

### S4. Safety and compliance notes

- Both modes emphasize ethical use of AI imagery.
- Both modes discourage deepfakes and misuse.
- The system does not store camera video or audio.

## Appendix T: Question and slide authoring guidelines

This appendix provides operational guidance for owners and admins who edit questions and slides.

### T1. Question identifiers

- question_id is the stable key used in database responses and exports.
- Changing question_id breaks historical comparisons.
- Always create new questions with new IDs rather than reusing old IDs.

### T2. Question types and categories

- question_type determines which page the question appears on: demographic, pre_test, post_test.
- category groups questions in the UI and analytics, for example:
- trust, engagement, satisfaction
- expectations, avatar-qualities, realism
- knowledge
- open_feedback

### T3. Correct answers

- correct_answer is stored as a string.
- For multiple correct answers, separate with |||.
- Correctness checks require exact match of the answer set.
- If correct_answer is empty, the question is treated as non scored.

### T4. allow_multiple and multi select

- allow_multiple in the public view determines if multiple answers can be selected.
- The UI uses allow_multiple for pre test and post test knowledge questions.

### T5. question_meta

- question_meta can include type (likert, open, text).
- question_meta can include placeholder text for open fields.
- question_meta may include prefer_not_to_say for demographics.

### T6. mode_specific

- mode_specific controls whether a question is shown to text users, avatar users, or both.
- Use both for questions that should be universal.

### T7. Slide content

- Use the formatting grammar outlined in Appendix M.
- Keep paragraphs concise and use lists for clarity.
- Avoid raw HTML; it will be sanitized.

### T8. Slide key points

- Key points should be short and declarative.
- They are displayed in a highlighted panel at the end of each slide.

### T9. Slide AI context

- system_prompt_context should summarize the slide in a few sentences.
- The AI tutor relies on this field to stay on topic.

### T10. Testing after changes

- After editing slides or questions, run a full study pass in text and avatar mode.
- Verify exports include the updated question text.
- Verify that post test category filters still align with question definitions.

## Appendix U: Admin charts and analytics interpretation

This appendix explains the analytics views in AdminOverview and how to interpret each chart or metric.

### U1. Session summary tiles

- Total completed: count of valid sessions with completed_at set.
- Total incomplete: valid sessions without completed_at.
- Text mode completed: sessions where modes_used includes text only.
- Avatar mode completed: sessions where modes_used includes avatar only.
- Both modes: sessions where modes_used includes both text and avatar.
- No mode: sessions with no mode data (should be rare).

Interpretation

- A high incomplete count may indicate drop off or technical friction.
- Both modes indicates mode switching in past versions or legacy flows.

### U2. Sessions per day

- Built from completed_at or created_at timestamps.
- Displays a count per date.

Interpretation

- Peaks may correspond to recruitment waves or classroom sessions.

### U3. Mode comparison

- Displays the distribution of text, avatar, and both.

Interpretation

- Helps verify the balance across modes.

### U4. Knowledge gain analysis

- Pre test score: percent correct on pre test items with correct answers.
- Post test score: percent correct on knowledge items with correct answers.
- Gain: post minus pre.

Interpretation

- Positive gain indicates learning improvement.
- Compare averages across modes for primary hypothesis.

### U5. Avatar time by slide

- Aggregates avatar_time_tracking durations.
- Shows average time per slide and total time.

Interpretation

- Slides with very low time may indicate skimming.
- Slides with high time may indicate difficulty or engagement.

### U6. Correlation charts

- Avatar time vs gain: x axis is avatar minutes, y axis is gain.
- Session duration vs gain: x axis is minutes, y axis is gain.

Interpretation

- Positive slope suggests more time is associated with higher gains.

### U7. Question performance analysis

- Each question shows correct rate and total responses.
- Missing correct answers are counted separately.

Interpretation

- Low correct rate may indicate confusing questions or slide gaps.
- Missing correct answers reduce the reliability of scoring.

### U8. Likert perception analysis

- Mean, median, and standard deviation for trust, engagement, satisfaction items.
- Distribution of 1 to 5 responses.

Interpretation

- Mean closer to 5 indicates positive perception.
- High std indicates divergent opinions.

### U9. Suspicion and validation summary

- Suspicious count and average suspicion score.
- Pending and awaiting approval counts.

- Top flags list.

Interpretation

- High suspicious count suggests rushed participation or friction.
- Many pending validations indicates admin backlog.

## Appendix V: Error handling and user messaging catalog

This appendix lists common error cases and user facing messages. Exact text may change with future revisions, but the triggers remain consistent.

### V1. Consent and session creation

- Trigger: create_session fails.
- Message: "Failed to start study session. Please try again."

### V2. Demographics save

- Trigger: save_demographics fails.
- Message: "Failed to save your responses. Please try again."

### V3. Pre test save

- Trigger: save_pre_test fails.
- Message: "Failed to save your responses. Please try again."

### V4. Post test save

- Trigger: save_post_test fails.
- Message: "Failed to save your responses. Please try again."

### V5. Session timeout

- Trigger: inactivity >= 30 minutes.
- Message: "Session expired due to inactivity. Please start the study again."

### V6. Mode switching attempt

- Trigger: selecting a different mode after studyMode is set.
- Message: "Mode switching is not allowed during an active session. Your session has been reset."

### V7. Chat failures

- Trigger: chat edge function returns error or non 200.
- Message: "Failed to get response" or "Service temporarily unavailable."

### V8. Image generation errors

- Rate limit: "Rate limit exceeded. Please wait a moment before generating another image."
- Payment required: "Credits required. Please add credits to continue generating images."
- Generic: "Generation failed. Please try again."

### V9. Admin data fetch

- Trigger: Supabase queries fail in admin tabs.
- Message: "Failed to load questions" or "Failed to load slides" with retry options.

### V10. API disabled

- Trigger: API master switch or specific API switch disabled.
- Message: "Service temporarily unavailable. Please try again later."

## Appendix W: Privacy and compliance notes

This appendix clarifies privacy statements and data handling practices.

### W1. Participant privacy

- Participants are not required to create accounts.
- No names, emails, or personal identifiers are collected.
- The only identifier is a random session_id UUID.

### W2. Camera and audio

- The consent form explicitly states that video is not recorded.
- Camera usage in avatar mode is local only and not stored.
- Audio is processed by Anam for the live session but not stored in the platform database.

### W3. Data minimization

- Responses are stored only as question_id and answer pairs.
- No raw device identifiers are stored; browser_fingerprint exists but is unused.

### W4. Exports and sharing

- Admin exports may include all responses and session IDs.
- Exports should be handled under research data policies.
- Participant exports are limited to their own session data.

### W5. Compliance intent

- The consent text references GDPR compliance and anonymity.
- Data is stored in Supabase with standard security practices.

## Appendix X: Mobile and responsive behavior notes

This appendix summarizes the mobile specific UI behaviors.

### X1. LikertScale

- On mobile, LikertScale renders a vertical list of buttons.
- Each option includes a numeric label and text.
- This improves tap targets and readability.

### X2. Learning page

- On mobile, the right side panel is replaced by a fixed bottom chat panel.
- Avatar mode is disabled on mobile to avoid multiple Anam clients.
- Playground panel is hidden on mobile.

### X3. SlideViewer

- SlideViewer occupies full width on mobile.

- Slide content is scrollable with touch gestures.
- Navigation buttons remain visible at the bottom.

### X4. Admin dashboard

- Admin tabs are horizontally scrollable on small screens.
- Data tables use horizontal scroll when needed.
- Export buttons are compacted or hidden for viewers.