



SECURITY ARCHITECTURE DOCUMENT

Word Labs Education

Version 1.0 · March 2026

Author	Nicholas Deeney, Word Labs Education
Contact	nick@wordlabs.app
Website	wordlabs.app

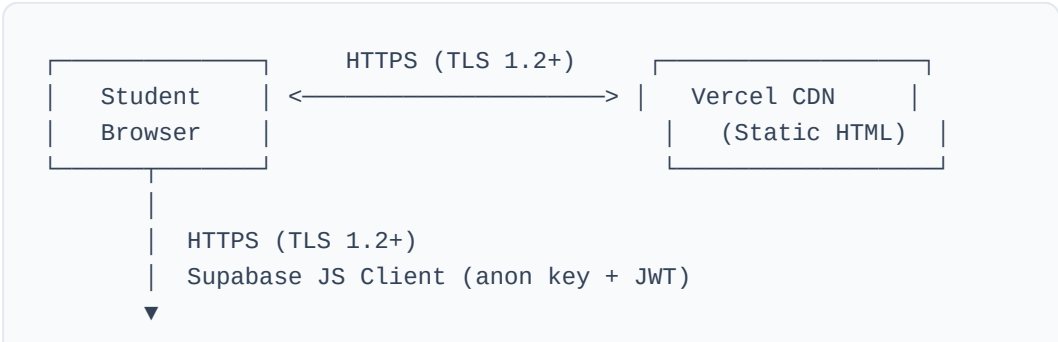
1. SYSTEM OVERVIEW

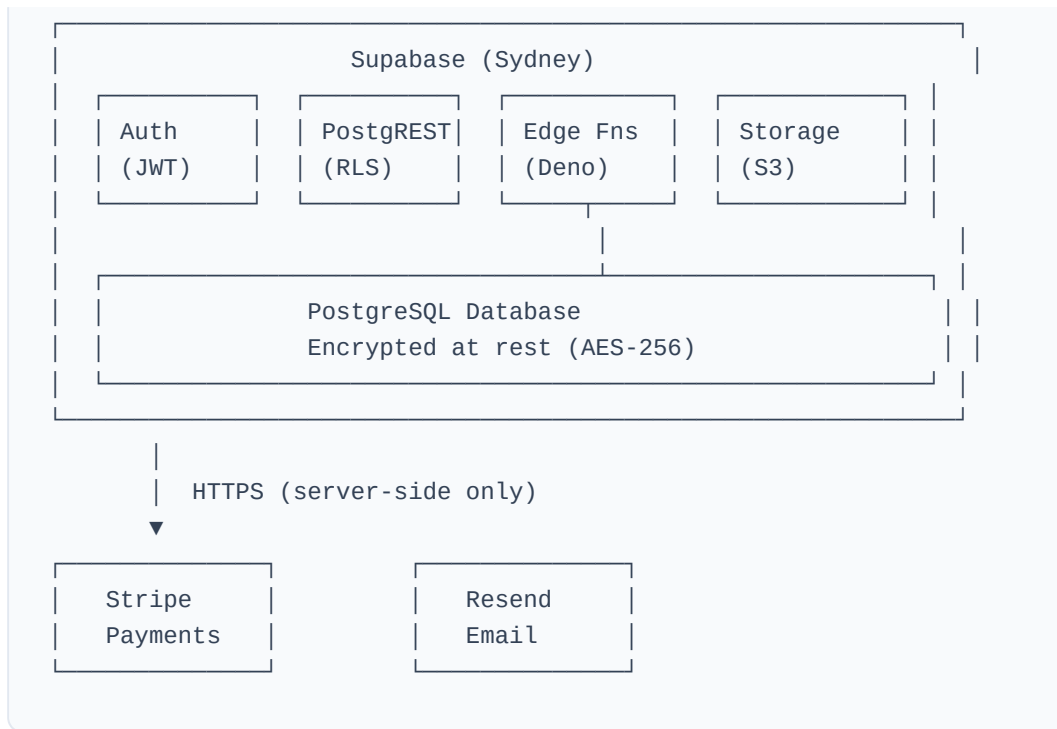
Word Labs is a browser-based educational application for morphology and phonics instruction, targeted at upper primary students (ages 9–12). The application consists of:

- **Frontend:** Static HTML/CSS/JavaScript pages served via Vercel CDN
- **Backend:** Supabase (hosted PostgreSQL with REST API, Auth, Edge Functions, Storage)
- **Payments:** Stripe (PCI DSS Level 1 compliant payment processor)
- **Transactional email:** Resend (domain-verified, sends from notifications@wordlabs.app)

No user data is processed or stored on the Vercel frontend servers. All data operations are handled by Supabase in the **Sydney (ap-southeast-2)** region.

2. DATA FLOW DIAGRAM





Teacher flow

- Teacher signs up via email and password — Supabase Auth creates account with bcrypt-hashed password
- Teacher creates classes and adds students — data written via Supabase REST API with RLS enforcement
- Teacher views dashboard — data read via Supabase REST API, scoped to their school by RLS

Student flow

- Student enters class code — Supabase REST API lookup (public, read-only via RLS)
- Student selects name and enters 3-character code — verified against students table
- Student plays games — progress recorded via atomic RPC function (increment_progress)
- Session stored in browser sessionStorage (cleared on tab close)

3. AUTHENTICATION MODEL

Teacher authentication

Provider	Supabase Auth (built on GoTrue)
Method	Email + password
Password storage	bcrypt hash (cost factor 10)

Session	JWT tokens issued by Supabase Auth, stored in browser memory
Token refresh	Automatic via Supabase JS client
Password reset	Email-based reset flow via Supabase Auth + Resend

Student authentication

Method	Class code (6-char alphanumeric) + student name + 3-character student code
Password	None — students are children (ages 9–12); codes are generated and managed by teachers
Verification	Codes are verified server-side via the <code>verify_student_login</code> Postgres RPC. Student codes are never sent to the client for comparison — the client passes the entered code to the RPC, which performs a constant-time lookup and returns a session token if valid. This prevents enumeration of student codes via direct table access.
Session	Stored in <code>sessionStorage</code> (not persistent across browser sessions)
PII	Student records contain first name only — no email, no date of birth, no photo

CSRF protection

- Supabase uses JWT bearer tokens in the Authorization header (not cookies), which is inherently CSRF-resistant
- Student sessions use `sessionStorage` (not cookies), providing CSRF protection
- All state-changing operations require a valid session

4. AUTHORISATION — ROW LEVEL SECURITY (RLS)

All database tables have Row Level Security (RLS) enabled. Policies enforce:

Table	SELECT	INSERT / UPDATE / DELETE
schools	Own school only (via <code>teachers.school_id</code>)	Own school only

Table	SELECT	INSERT / UPDATE / DELETE
teachers	Own record only	Own record only
classes	Open (students need to find their class)	Teacher's school only (via <code>auth_user_id</code>)
students	Open (students need to see classmates for login)	Teacher's school only
student_progress	Open (students view own progress)	Via <code>increment_progress</code> RPC only
student_character	Open (students view own character)	Open (students update own character)
shop_items	Open (all students browse shop)	Teacher's school only
class_word_lists	Open (students load word lists)	Teacher's school only (via <code>get_my_school_id()</code>)
class_spelling_sets	Open (students load assigned sets)	Teacher's school only (via <code>get_my_school_id()</code>)
spelling_set_assignments	Open (students load their assignments)	Teacher's school only (via <code>get_my_school_id()</code>)
spelling_check_in_results	Open (students view own results)	Teacher's school only (via <code>get_my_school_id()</code>)
feedback	Own record only	Insert-only via edge function

Key RLS principles

- Teachers can only modify data belonging to their own school
- Students can read class-level data but cannot modify other students' records

- All cross-school access is denied at the database level — even a compromised client cannot read another school's data
- School scoping for spelling sets, word lists, and their assignments is enforced via the `get_my_school_id()` helper which chains `teacher` → `school` in a single RLS check
- The Supabase anon key is intentionally public — RLS is the security boundary, not key secrecy

Atomic write operations (race condition prevention)

State-changing operations that could race under concurrent requests are implemented as Postgres RPCs rather than client-side read-modify-write sequences. This guarantees atomicity at the database level and eliminates lost-update bugs.

RPC	Purpose	Mechanism
<code>increment_progress</code>	Game answer recording (most frequent write)	Single atomic UPSERT with SQL arithmetic (<code>correct = correct + 1</code>)
<code>purchase</code>	Shop item purchase (debit quarks, grant item)	<code>SELECT FOR UPDATE</code> row lock on <code>student_character</code>
<code>saveScientist</code>	Character customisation saves	Targeted <code>jsonb_set</code> update (no full-row overwrite)
<code>atomic_delete_class</code>	Teacher deletes a class	Server-side transaction deleting class + students + progress in one statement
<code>verify_student_login</code>	Student login code verification	Server-side code comparison; codes never exposed to client

5. ENCRYPTION

In transit

- All client-server communication uses HTTPS (TLS 1.2+)
- Vercel enforces HTTPS with automatic certificate provisioning
- Supabase enforces HTTPS for all API endpoints
- WebSocket connections (Supabase realtime) use WSS

At rest

- **Supabase PostgreSQL:** AES-256 encryption at rest (AWS managed encryption)
- **Supabase Storage:** AES-256 encryption at rest (AWS S3 server-side encryption)
- **Stripe:** PCI DSS Level 1 (no card data touches Word Labs infrastructure)

Secrets management

- Stripe secret key — stored in Supabase Edge Function secrets (environment variables)
- Stripe webhook secret — stored in Supabase Edge Function secrets
- Resend API key — stored in Supabase Edge Function secrets
- Anthropic API key (for AI word analysis) — stored in Supabase Edge Function secrets
- **No secrets are stored in frontend code or version control**

6. HTTP SECURITY HEADERS

The following security headers are deployed via Vercel configuration:

Header	Value	Purpose
X-Content-Type-Options	nosniff	Prevents MIME type sniffing
X-Frame-Options	SAMEORIGIN	Prevents clickjacking
X-XSS-Protection	1; mode=block	Legacy XSS filter (defence in depth)
Referrer-Policy	strict-origin-when-cross-origin	Limits referrer leakage
Content-Security-Policy	See below	Restricts resource loading

Content Security Policy

```
default-src 'self';
script-src 'self' 'unsafe-inline'
  https://cdn.jsdelivr.net/npm/@supabase/
  https://cdn.jsdelivr.net/npm/jszip@3/
  https://cdn.jsdelivr.net/npm/pptxgenjs@3.12.0/
  https://cdnjs.cloudflare.com/ajax/libs/fabric.js/5.3.1/
  https://cdnjs.cloudflare.com/ajax/libs/qrcodejs/1.0.0/;
```

```
style-src 'self' 'unsafe-inline' https://fonts.googleapis.com;
font-src 'self' https://fonts.gstatic.com;
img-src 'self' data: blob: https://kdpavfrzmmzknqfpodrl.supabase.co;
connect-src 'self' https://kdpavfrzmmzknqfpodrl.supabase.co
      wss://kdpavfrzmmzknqfpodrl.supabase.co;
form-action 'self';
frame-ancestors 'none';
object-src 'none';
base-uri 'self'
```

Note: 'unsafe-inline' is required for script-src because the application uses inline script blocks (no build system or bundler). This is mitigated by: (1) HTML-escaping all user-supplied data before DOM insertion, (2) no dynamic code evaluation anywhere in the codebase, and (3) inline event handlers removed from security-sensitive pages (login, signup, account) in favour of `addEventListener()`. Script sources are restricted to specific CDN package paths rather than broad CDN wildcards.

7. INPUT VALIDATION AND XSS PREVENTION

Server-side

- Supabase RLS policies validate data types and ownership
- Edge functions validate request bodies and authentication
- PostgreSQL column types enforce data constraints

Client-side

- All user-supplied data (student names, class names, word list names) is HTML-escaped via a dedicated `escapeHtml()` function before insertion into the DOM
- Template literals in `innerHTML` use escaped values to prevent script injection
- URL parameters are validated (e.g., `returnTo` only allows relative alphanumeric paths — prevents open redirect)
- Form inputs use appropriate HTML5 types (`type="email"`, `required`, etc.)

8. DATA MINIMISATION

Data	Stored?	Purpose
Student first name	Yes	Display in-game and on teacher dashboard
Student surname	No	Not collected

Data	Stored?	Purpose
Student email	No	Not collected
Student date of birth	No	Not collected
Student photo	No	Not collected
Student IP address	No	Not logged by application
Game progress (scores)	Yes	Teacher reporting and student feedback
Teacher email	Yes	Authentication and communication
School name	Yes	Multi-tenancy and display
Payment details	No	Handled entirely by Stripe

9. SUB-PROCESSORS AND DATA SOVEREIGNTY

Sub-processor	Purpose	Data accessed	Location
Supabase Inc.	Database, auth, storage, edge functions	All application data	Sydney, Australia (ap-southeast-2)
Vercel Inc.	Static file hosting (CDN)	No user data (static files only)	Global CDN, Sydney edge
Stripe Inc.	Payment processing	Teacher email, school name (billing only). No student data.	Global (PCI DSS Level 1)
Anthropic PBC	AI word analysis (via edge function)	Word lists only. No student data.	United States
Google Cloud (Google LLC)	Text-to-speech audio	Word text only. No student data.	United States / global
Resend Inc.	Transactional email	Teacher email only. No student data.	United States

Sub-processor	Purpose	Data accessed	Location
Cloudflare Inc.	DNS and domain registration	No user data	Global

All student and teacher data is stored in Supabase Sydney (ap-southeast-2), Australia. No student data ever leaves Australia. Stripe processes payment data under PCI DSS compliance. Anthropic and Google Cloud only receive word content for analysis and text-to-speech respectively — no student names, progress, or personal data.

10. ACCESS CONTROL

Production database

- Only the application owner (Nicholas Deeney) has Supabase dashboard access
- No shared admin accounts
- Supabase management access requires email + password + MFA

Code repository

- GitHub repository: NickD135/morphology-builder
- Only the owner has push access to the main branch
- Vercel auto-deploys from the main branch

Edge functions

- Deployed via Supabase CLI
- Only the owner has deployment credentials
- Functions run in Deno isolates with limited permissions

11. EDGE FUNCTION SECURITY

Word Labs runs 10 Supabase Edge Functions for server-side operations that cannot safely run in the client (Stripe checkout, AI word analysis, TTS generation, email delivery, etc.). Every client-facing edge function is hardened to the following standard.

Authentication

Every edge function that handles teacher-scoped or school-scoped data validates the caller's Supabase JWT before executing any work. Functions that

accept public requests (student-facing TTS, for example) still validate that the request originates from an allowed origin via the CORS check below.

CORS allowlist

Client-facing edge functions restrict `Access-Control-Allow-Origin` to an explicit allowlist of five origins (production, Vercel preview deployments, and local development). Previously these returned the wildcard `*`, which was tightened to an allowlist as part of a full security audit.

Rate limiting

The `send-feedback` function enforces application-level rate limiting (1 submission per email address per 5 minutes) on top of Supabase's built-in platform limits. This prevents abuse of the feedback channel for spam or denial-of-wallet on the Resend mail delivery provider.

Redirect validation

The `create-checkout` and `create-portal-session` functions validate the Stripe redirect URL against an allowlist before constructing the checkout session. This prevents open-redirect vulnerabilities where an attacker could construct a checkout URL pointing at a phishing page.

Secrets hygiene

API keys and webhook secrets are stored in Supabase Edge Function environment variables, never in source code or version control. Logs are scrubbed of key prefixes — previously the `analyze-words` function logged the first few characters of its API key for debugging, which was removed to prevent key-prefix exposure through log aggregation.

Email template safety

The `send-feedback` function's email template HTML-escapes all user-supplied content before interpolating it into the email body, preventing HTML injection in feedback notifications delivered via Resend.

12. VULNERABILITY MANAGEMENT

Current mitigations

- HTML escaping of all user-supplied data in DOM operations
- Content Security Policy headers restricting resource loading
- Row Level Security enforcing data access boundaries
- HTTPS everywhere (Vercel + Supabase)
- Open redirect protection on login flow

Ongoing practices

- Regular review of Supabase security advisories
- Dependency updates for Supabase JS client (loaded via CDN — always latest)
- Code review before deployment to production

Known limitations

- 'unsafe-inline' in CSP script-src (required by architecture — no build system)
- No external penetration testing conducted yet (planned)
- No Web Application Firewall (WAF) in front of the application
- Rate limiting is applied to the public feedback channel but otherwise relies on Supabase's built-in platform limits

13. COMPLIANCE SUMMARY

Requirement	Status
Australian Privacy Act 1988	Compliant — privacy policy published, APPs addressed
Australian Privacy Principles (APPs)	Compliant — data minimisation, access, correction, deletion
NSW PPIP Act 1998	Compliant — IPP alignment documented in privacy policy
Notifiable Data Breaches scheme	Documented — see Incident Response Plan
Data stored in Australia	Yes — Supabase Sydney (ap-southeast-2)
HTTPS / encryption in transit	Yes — TLS 1.2+ everywhere
Encryption at rest	Yes — AES-256 (Supabase/AWS)
Children's data protection	Yes — first names only, no PII, school is data controller
WCAG 2.1 AA accessibility	Compliant — skip links, ARIA landmarks, keyboard alternatives, colour contrast
PCI DSS (payments)	Handled by Stripe (Level 1 certified)

14. CONTACT

Security contact	Nicholas Deeney
Email	nick@wordlabs.app
Privacy policy	wordlabs.app/privacy
Incident response plan	wordlabs.app/incident-response

*This document should be reviewed and updated whenever the architecture changes.
Last reviewed: April 2026.*