Pollivu — Product & Architecture Document

> **Privacy-First, Real-Time Polling Platform** Version 2.0 · February 2026

■ Deliverable Summary

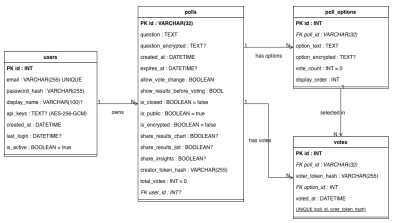
#	Deliverable	Location
1	**Product & Architecture Document**	This file (`PRODUCT_ARCHITECTURE.md`)
2	**Wireframes / Prototype**	`pollivu_wirefreme.svg`
3	**Database Tables Visualization**	**tables.drawio.png** 
4	**Working Application (Live URL)**	https://jayachandranpm.pythonanywhere.com

Table of Contents

---

1. Problem Definition & Target Users

The Problem

Existing polling tools (Google Forms, Strawpoll, Typeform) suffer from one or more of these issues:

Pain Point	Impact
Require sign-up to vote	Drops participation by 40–60%
Collect PII / track users	Privacy-aware audiences avoid them
No real-time feedback	Creators can't see momentum as it happens
Overly complex UIs	Takes 5+ clicks to create a simple poll
No AI assistance	Users struggle to craft good questions and options

Our Thesis

> A poll that can be **created in 30 seconds**, **voted on without sign-up**, and shows **live results** — all while collecting **zero personal data** — will see significantly higher engagement than existing tools.

Success Metrics

Metric	Target
Idea → live shareable poll	**< 60 seconds**
Voter casts a vote from opening link	**< 10 seconds**
Personal data stored for voters	**Zero PII**

Target Users

Primary: Team Leads & Community Managers

- **Who:** Slack workspace admins, Discord moderators, classroom teachers, startup founders
- **Need:** Quick decision-making with group input (e.g., "Where should we hold the offsite?")
- **Behavior:** Creates 2–5 polls per week; shares via link in chat/email

Secondary: Anonymous Voters

- **Who:** Anyone with the poll link
- **Need:** Cast a vote quickly without creating an account or revealing identity
- **Behavior:** One-time visitor; arrives via shared link; leaves after voting

Persona Comparison

Attribute	Creator ("Priya")	Voter ("Arjun")
Tech comfort	Moderate	Low–High
Account needed	■ Yes (to manage polls)	■ No
Core action	Create → Share → Analyze	Click link → Vote → See results
Time budget	60 seconds	10 seconds

---

2. Core User Flows

Flow 1 — Poll Creation (Authenticated)

```
Landing Page → Register / Login → Dashboard → "Create Poll" → Enter question + options → Set expiration, visibility, vote-change rules → Submit → Redirected to Results page (with share link + QR code)
```

Flow 2 — AI-Assisted Poll Creation

```
Dashboard → "Create with AI" → Enter topic + choose provider (Gemini/OpenAI/Claude) → AI generates question + options → User edits/accepts → Submit → Live poll
```

Flow 3 — Voting (Anonymous, No Auth)

```
Open shared link → See question + options → Tap an option → Vote recorded (session-hashed, anonymous) → See live results (animated bar chart + doughnut)
```

Flow 4 — Real-Time Results

```
Creator opens Results page → Sees live bar/doughnut charts + word cloud + timeline → As voters vote, charts update via short polling (every 3s) → Export CSV / Generate QR code / Share link → Creator changes settings → All viewers see changes within 3 seconds
```

Flow 5 — Poll Management

```
Dashboard → See all polls (active, expired, closed) → Edit question / Add-Remove options / AI-suggest new options
→ Close / Reopen / Delete / Toggle public-private → Toggle results sharing: chart, vote list, insights
(individually)
```

Flow 6 — Poll Embedding

```
Creator copies embed code (iframe) from poll or results page → Pastes into website / blog / CMS → Lightweight
embeddable widget renders in iframe → Visitors vote directly in the embed (AJAX, no redirect)
```

---

3. High-Level Architecture Diagram

Component Overview

Component	Technology	Purpose
Cloud Platform	PythonAnywhere	Hosting, managed MySQL, SSL, Python-native
Web Server	Gunicorn (2 workers, 4 threads)	WSGI server for Flask
Framework	Flask 3.1.2	Lightweight Python web framework

Templating	Jinja2	Server-side HTML rendering
Real-Time	Short polling (3s `setInterval` + `fetch`)	Near real-time updates without blocking threads
Database	MySQL 8.0	Persistent relational data storage
ORM	SQLAlchemy + Flask-Migrate (Alembic)	DB abstraction & version-controlled migrations
Caching	Flask-Caching (SimpleCache / Redis)	Response caching & rate-limit backend
Authentication	Flask-Login + PBKDF2-SHA256	Session-based user auth
Encryption	AES-256-GCM (cryptography lib)	API key & sensitive data encryption at rest
AI	Gemini / OpenAI / Claude	Multi-provider poll generation & suggestions
Charts	Chart.js (CDN)	Client-side data visualization
CSS	Custom CSS (no framework)	Lightweight, fast-loading styles

---

4. Key Technical Decisions

4a. Why PythonAnywhere? (Cloud Platform)

Factor	Decision	Rationale
Deployment model	PaaS (Platform-as-a-Service)	No server management, focus on application code
MySQL	Built-in managed MySQL	One-click database provisioning, automatic backups
SSL	Free auto-TLS certificates	HTTPS by default for all web apps
Python-native	Purpose-built for Python apps	Pre-installed Python versions, `pip` support, WSGI configuration
Cost	Free tier available; affordable paid plans	Ideal for early-stage products
Git workflow	`git pull` → reload → live	Deploy in under 60 seconds

Trade-off acknowledged: PythonAnywhere does **not** support WebSockets. This led us to adopt short polling for real-time updates — which actually turned out to be simpler, more portable, and more scalable than

WebSocket/SSE alternatives (see Section 10).

When we'd migrate: At ~10,000+ concurrent users, we'd move to AWS ECS or GCP Cloud Run for horizontal scaling and WebSocket support.

4b. Why Gunicorn + Threads? (Compute Model)

Factor	Decision	Rationale
Server	Gunicorn 24.x	Industry-standard Python WSGI server
Workers	2 workers × 4 threads = 8 concurrent requests	Matches PythonAnywhere's resource limits
Why threads	`--threads 4` instead of `eventlet`	Flask is synchronous; threads are simpler, compatible with all Python versions
Why not eventlet	Removed	Eventlet is incompatible with Python 3.13+; causes green-thread issues
Why not async (Uvicorn/FastAPI)	Overhead	Flask ecosystem (Flask-Login, Flask-WTF, Flask-Limiter) is mature and battle-tested

Procfile:

```
web: gunicorn -w 2 -b 0.0.0.0:$PORT --timeout 120 --threads 4 app:app
```

Scaling path:

```
Current: 2 workers × 4 threads = 8 concurrent requests ↓ (~5K users) Phase 2: 4 workers × 4 threads + Redis caching ↓ (~50K users) Phase 3: Multiple containers + Load balancer + Redis + Read replicas
```

4c. Why MySQL? (Database)

Factor	Decision	Rationale
Data model	Relational (SQL)	Polls → Options → Votes is inherently relational with foreign keys
Engine	MySQL 8.0	Excellent read performance for vote-count aggregations
Why not PostgreSQL	Familiarity + PaaS support	MySQL is natively available on PythonAnywhere

Why not NoSQL	Constraints needed	Vote uniqueness (`UNIQUE(poll_id, voter_hash)`) is critical — easier in SQL
Migrations	Flask-Migrate (Alembic)	Version-controlled schema changes

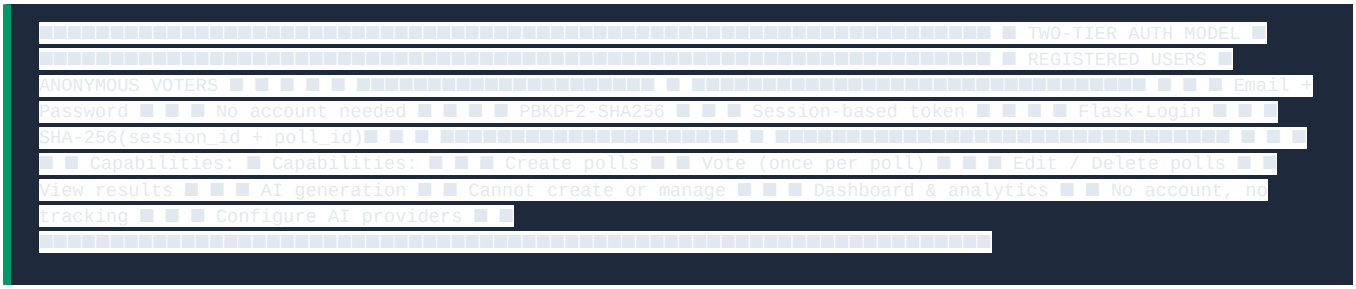
Query patterns:

Pattern	Example	Frequency
Write-heavy	`INSERT INTO votes` (with unique constraint check)	Every vote
Read-heavy	Aggregate vote counts per option	Every page load + every 3s poll
Indexed lookups	`WHERE poll_id = ?`, `WHERE user_id = ?`	Every request

Indexes:

- `polls.user_id` — fast dashboard loading
- `votes.poll_id + voter_token_hash` — composite index for vote dedup
- `poll_options.poll_id` — fast option retrieval

4d. How Authentication Works



Authentication Mechanism Details

Component	Implementation	Why
Password hashing	`PBKDF2-SHA256` via Werkzeug	NIST-recommended; built into Flask — no extra dependency
Session management	Server-side Flask sessions	`HTTPONLY=True`, `SAMESITE=Lax`, `SECURE=True` (production)
Vote deduplication	`SHA-256(session_id + poll_id)`	Prevents double-voting **without storing any personal data** — no IP, no fingerprint, no cross-poll tracking

API key storage	AES-256-GCM encrypted in MySQL	Users configure their own AI provider keys; encrypted at rest with PBKDF2-derived keys from app secret
CSRF protection	Flask-WTF CSRF tokens	Every POST form includes a cryptographic token to prevent cross-site request forgery

Session Cookie Configuration

```
SESSION_COOKIE_SECURE = True # HTTPS only (production) SESSION_COOKIE_HTTPONLY = True # Not accessible via
JavaScript SESSION_COOKIE_SAMESITE = 'Lax' # Prevents CSRF via third-party sites
```

4e. How Data is Stored and Accessed

Data Flow — Complete Vote Lifecycle

```
■■■■■■■■■■■■■■■■■■■■ ■ Voter opens ■ ■ shared link ■ ■■■■■■■■■■■■■■■■■■■■ ■ ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ GET /poll/<id> ■ ■ → Poll.query.get_or_404(poll_id) ■ ■
→ Check session for existing vote ■ ■ → Render poll.html with options ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ Voter selects an option ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ POST /poll/<id>/vote ■ ■ → Rate Limiter: 30
requests/minute ■ ■ → CSRF token validation ■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ ▼
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ PollService.vote() ■ ■ ■ ■ 1. Generate voter_token_hash:
■ ■ SHA-256(session_id + poll_id) ■ ■ ■ ■ 2. Check existing vote: ■ ■ SELECT FROM votes ■ ■ WHERE poll_id=? AND
voter_hash=? ■ ■ ■ ■ 3a. No existing vote: ■ ■ INSERT INTO votes (...) ■ ■ UPDATE poll_options SET ■ ■
vote_count = vote_count + 1 ■ ■ UPDATE polls SET ■ ■ total_votes = total_votes + 1 ■ ■ ■ ■ 3b. Existing +
allow_change: ■ ■ UPDATE old option count - 1 ■ ■ UPDATE new option count + 1 ■ ■ UPDATE vote record ■ ■ ■ ■ 3c.
Existing + no change allowed: ■ ■ → 403 "Already voted" ■ ■ ■ ■ 4. db.session.commit() ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ ▼ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ Short Polling (other clients) ■ ■ ■ ■ Every 3 seconds: ■ ■ GET /api/poll/<id>/live_stats ■ ■ → Returns:
total_votes, results[], ■ ■ is_active, updated_at ■ ■ → Client compares with previous state ■ ■ → Updates
charts/bars in-place ■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

Storage Architecture

Data	Storage	Encryption	Access Pattern	Retention
User credentials	`users` table	Password: PBKDF2-SHA256 hash	Login authentication	Permanent
AI API keys	`users.api_keys` column	AES-256-GCM encrypted	Decrypt on AI request	Until user deletes
Poll questions	`polls` table	Optional AES-256-GCM	Read on poll view	Until poll deleted
Poll options	`poll_options` table	Optional AES-256-GCM	Read on poll view	Until poll deleted
Poll settings	`polls` table (booleans)	Plaintext	Read on every request	Until poll deleted

Votes	`votes` table	Voter identity: SHA-256 hash only	Write once; read for counts	Until poll deleted
Sessions	Server-side (signed cookie)	HMAC-signed	Every request	Browser session
API cache	In-memory / Redis	N/A	Every 2–3 seconds (short polling)	2–5 min TTL

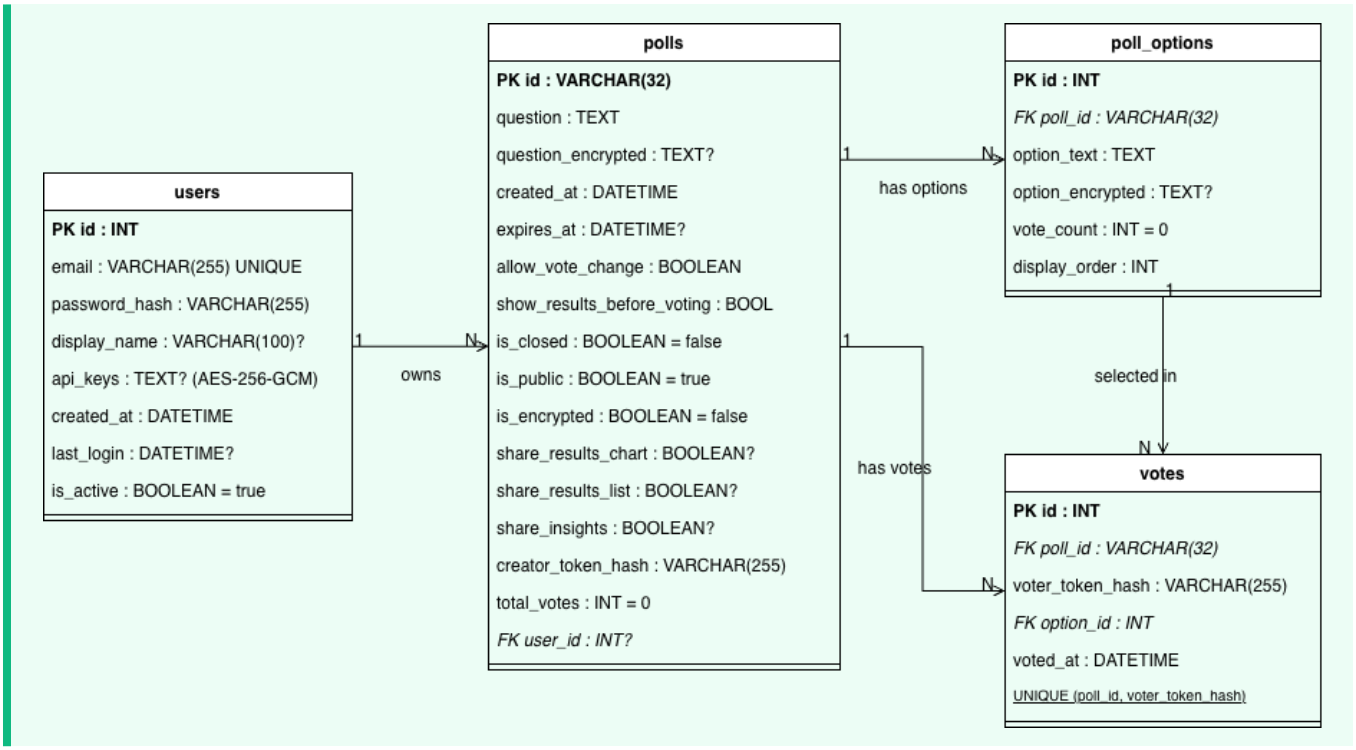
Encryption Architecture

```
┌─────────────────────────────────────────────────────────────────┐
│                    ENCRYPTION LAYERS                            │
├─────────────────────────────────────────────────────────────────┤
│ Transport: HTTPS (TLS 1.3) — enforced via HSTS                  │
│ Passwords: PBKDF2-SHA256 — Werkzeug generate_password_hash()    │
│            600,000 iterations (default)                         │
│ API Keys: AES-256-GCM ──────────────────────────────────────    │
│           App Secret Key → PBKDF2-HMAC-SHA256 → 256-bit AES Key │
│           + Random 96-bit Nonce → AES-256-GCM Encrypt           │
│           → Base64-encoded ciphertext → MySQL ──────────────    │
│ Voter IDs: SHA-256(session_id + poll_id) — One-way hash — cannot be reversed │
│ No PII stored — Poll Data: Optional AES-256-GCM for sensitive questions and option text │
└─────────────────────────────────────────────────────────────────┘
```

---

5. Database Tables Visualization

> ■ **Full visual diagram:** See
>
> **tables.drawio.png**



Entity Relationship Diagram (Text)

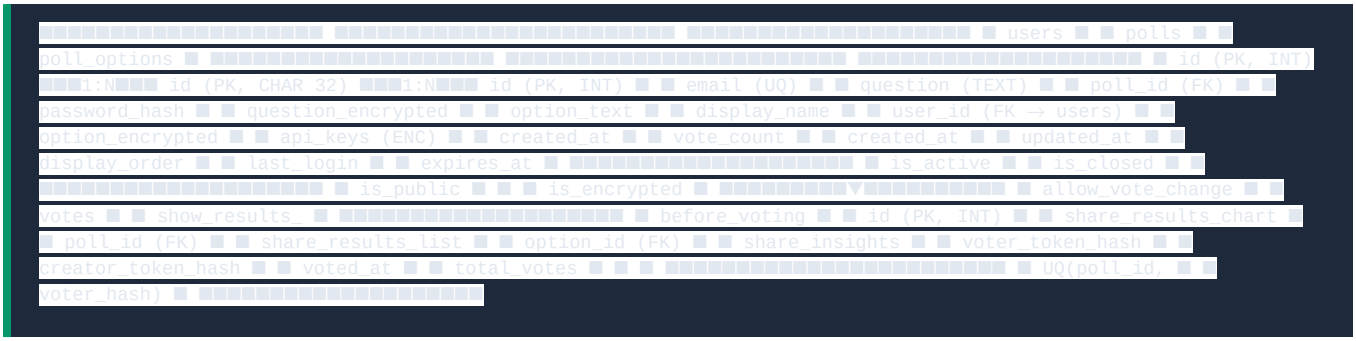


Table Details

`users` — Registered poll creators

Column	Type	Constraints	Description
`id`	INT	PK, AUTO_INCREMENT	Unique user ID
`email`	VARCHAR(255)	UNIQUE, NOT NULL, INDEXED	Login identifier
`password_hash`	VARCHAR(255)	NOT NULL	PBKDF2-SHA256 hash
`display_name`	VARCHAR(100)	NULLABLE	User's display name
`api_keys`	TEXT	NULLABLE	AES-256-GCM encrypted JSON of AI provider keys
`created_at`	DATETIME	DEFAULT NOW()	Registration timestamp
`last_login`	DATETIME	NULLABLE	Last login time
`is_active`	BOOLEAN	DEFAULT TRUE	Account status

`polls` — Poll questions and settings

Column	Type	Constraints	Description
`id`	CHAR(32)	PK	32-char random hex (128 bits of entropy)
`question`	TEXT	NOT NULL	Poll question text
`question_encrypted`	TEXT	NULLABLE	AES-256-GCM encrypted question (optional)
`user_id`	INT	FK → users.id, INDEXED	Creator (nullable for anonymous)
`created_at`	DATETIME	DEFAULT NOW()	Creation timestamp

`updated_at`	DATETIME	DEFAULT NOW(), ON UPDATE NOW()	Last modification (used for change detection)
`expires_at`	DATETIME	NULLABLE	Auto-close time (null = never)
`is_closed`	BOOLEAN	DEFAULT FALSE	Manually closed by creator
`is_public`	BOOLEAN	DEFAULT TRUE	Visible in public listings
`is_encrypted`	BOOLEAN	DEFAULT FALSE	Whether question is encrypted
`allow_vote_change`	BOOLEAN	DEFAULT FALSE	Allow voters to change their vote
`show_results_before_voting`	BOOLEAN	DEFAULT FALSE	Show results before casting vote
`share_results_chart`	BOOLEAN	DEFAULT TRUE	Share doughnut chart with non-creators
`share_results_list`	BOOLEAN	DEFAULT TRUE	Share vote list with non-creators
`share_insights`	BOOLEAN	DEFAULT TRUE	Share analytics with non-creators
`creator_token_hash`	VARCHAR(255)	NOT NULL	SHA-256 hash for anonymous ownership verification
`total_votes`	INT	DEFAULT 0	Denormalized vote counter for fast reads

`poll_options` — Poll choices

Column	Type	Constraints	Description
`id`	INT	PK, AUTO_INCREMENT	Unique option ID
`poll_id`	CHAR(32)	FK → polls.id, ON DELETE CASCADE	Parent poll
`option_text`	TEXT	NOT NULL	Option display text
`option_encrypted`	TEXT	NULLABLE	AES-256-GCM encrypted option text
`vote_count`	INT	DEFAULT 0	Denormalized vote counter
`display_order`	INT	NOT NULL	Rendering order

`votes` — Anonymous vote records

Column	Type	Constraints	Description
`id`	INT	PK, AUTO_INCREMENT	Unique vote ID
`poll_id`	CHAR(32)	FK → polls.id, ON DELETE CASCADE	Parent poll
`option_id`	INT	FK → poll_options.id, ON DELETE CASCADE	Chosen option
`voter_token_hash`	VARCHAR(255)	NOT NULL	SHA-256 hash — **no PII stored**
`voted_at`	DATETIME	DEFAULT NOW()	Vote timestamp
—	—	**UNIQUE(poll_id, voter_token_hash)**	Enforces one vote per session per poll
—	—	**INDEX(poll_id, voter_token_hash)**	Fast duplicate check

Key Constraints & Design Choices

Design Choice	Rationale
32-char hex poll ID	Unguessable (128 bits entropy); doubles as an access token for unlisted polls
Denormalized `total_votes` & `vote_count`	Avoids `COUNT(*)` queries on every page load; updated atomically on vote
`CASCADE DELETE` on all FKs	Deleting a poll removes all options and votes automatically
`UNIQUE(poll_id, voter_token_hash)`	Database-level enforcement of one vote per session per poll
`updated_at` with `ON UPDATE`	Short polling clients compare this to detect any settings change

---

6. Wireframes / Prototype

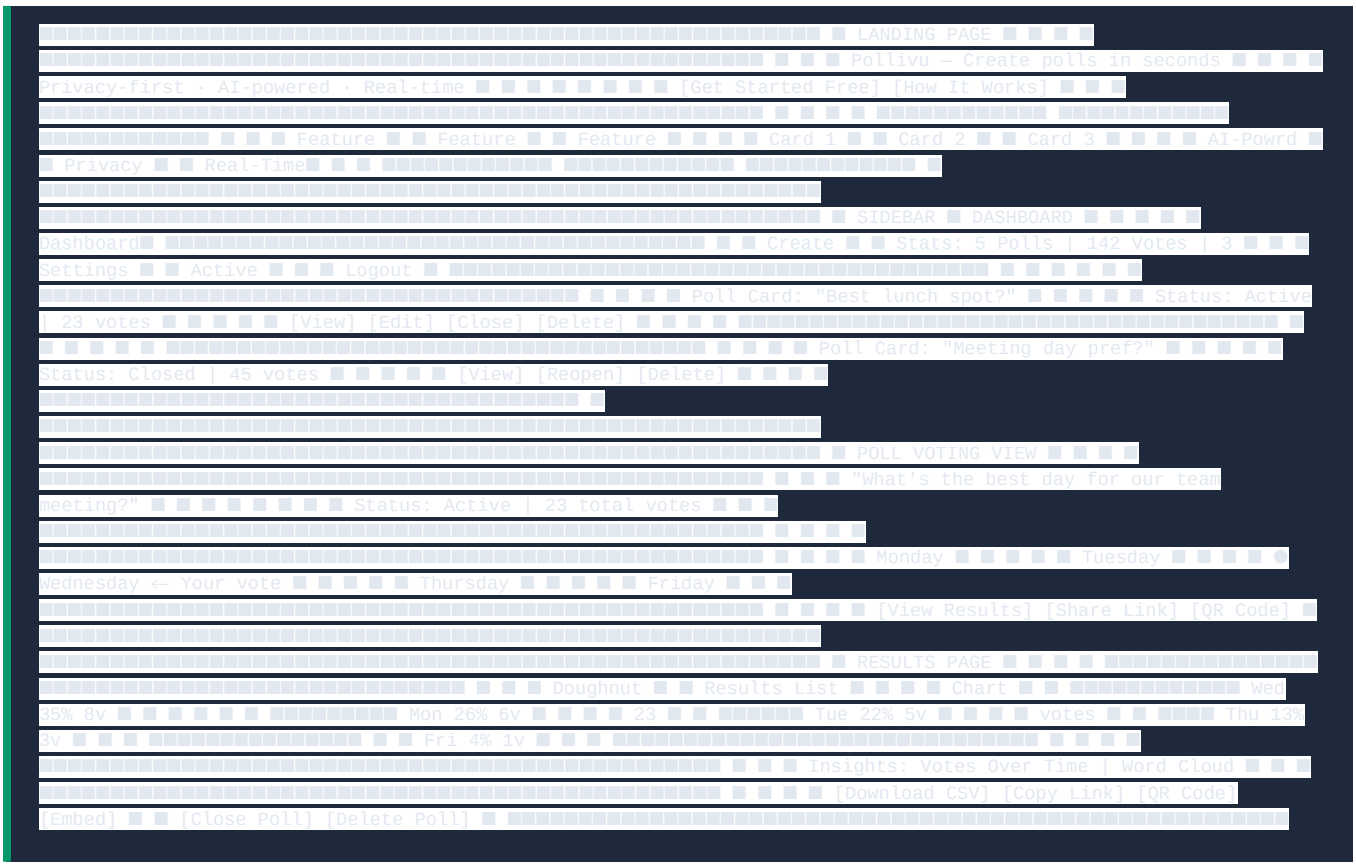
> ■ **Full wireframes:** See `pollivu_wirefreme.svg`

Screen Inventory

Screen	Purpose	Auth Required

Landing Page	Product overview, CTA to register/login	■
Register / Login	Email + password authentication	■
Dashboard	List all user's polls, stats, quick actions	■
Create Poll	Manual poll creation form (question + 2–10 options + settings)	■
Create with AI	AI-assisted poll generation (topic → poll)	■
Poll View	Voting interface — see question, tap to vote	■
Results	Live charts (doughnut + bar), insights (timeline + word cloud), export	■
Edit Poll	Modify question, options, settings, AI-suggest new options	■
Settings	Configure AI provider API keys (Gemini/OpenAI/Claude)	■

Key Screen Layouts



7. Feature Scope

Shipped Features (v2.0)

Feature	Status	Details
User registration & login	■	Email + password, PBKDF2-SHA256
Create poll (2–10 options)	■	Manual form with validation
Poll expiration	■	1h, 24h, 7d, 30d, never
Anonymous voting	■	No login required, session-hashed dedup
Near real-time updates	■	Short polling, 3s interval, `updated_at` change detection
Live charts	■	Doughnut + bar via Chart.js
AI poll generation	■	Gemini, OpenAI, Claude
AI option suggestions	■	Smart suggestions on edit page
QR code generation	■	Emerald-themed QR for poll sharing
CSV export	■	Download results as spreadsheet
Poll editing	■	Question, options, all settings
Close / Reopen / Delete	■	Creator actions with real-time sync
Public / Unlisted toggle	■	Unlisted = only accessible via link
Allow vote changes	■	Per-poll toggle
Granular results sharing	■	Chart / vote list / insights toggles
Poll embedding	■	iframe widget with isolated CSP
AES-256-GCM encryption	■	API keys + optional poll data
Rate limiting	■	Per-endpoint (30 votes/min, 10 AI/min)
CSRF protection	■	Flask-WTF on all POST forms
Security headers	■	CSP, HSTS, X-Frame, X-Content-Type
Responsive design	■	Mobile-first, custom CSS

Dashboard with analytics	■	Votes timeline + word cloud

Future Roadmap

Feature	Priority	Complexity
OAuth sign-in (Google, GitHub)	High	Medium
Poll templates & categories	Medium	Low
Webhook notifications	Medium	Medium
Team workspaces	Medium	High
Multi-question surveys	Low	High

---

8. API Design

RESTful Endpoints

Method	Endpoint	Auth	Rate Limit	Purpose
`GET`	`/poll/<id>`	None	Exempt	View poll for voting
`POST`	`/poll/<id>/vote`	None	30/min	Cast a vote
`GET`	`/poll/<id>/results`	None	Exempt	View results
`POST`	`/poll/<id>/close`	Creator	Default	Close poll
`POST`	`/poll/<id>/reopen`	Creator	Default	Reopen poll
`POST`	`/poll/<id>/delete`	Creator	Default	Delete poll
`POST`	`/poll/<id>/toggle-public`	Owner	Default	Toggle visibility
`GET/POST`	`/poll/<id>/edit`	Owner	Default	Edit poll
`POST`	`/poll/<id>/option/add`	Owner	Default	Add option
`POST`	`/poll/<id>/option/<oid>/delete`	Owner	Default	Remove option
`POST`	`/poll/<id>/options/suggest`	Owner	Default	AI suggest options
`GET`	`/poll/<id>/export/csv`	None	Default	Download CSV
`GET`	`/poll/<id>/qr`	None	Default	Generate QR code

`GET`	`/poll/<id>/embed`	None	Exempt	Embeddable iframe widget

AI API Endpoints

Method	Endpoint	Auth	Rate Limit	Purpose
`POST`	`/api/ai/generate`	User	10/min	Generate poll via AI
`POST`	`/api/ai/suggest`	User	10/min	Get AI improvements
`POST`	`/api/ai/test`	User	10/min	Test provider connection
`GET`	`/api/ai/providers`	User	Default	List configured providers

Live Data API (Short Polling)

Method	Endpoint	Cache	Rate Limit	Purpose
`GET`	`/api/poll/<id>/live_stats`	2s	60/min	Votes, status, `updated_at`
`GET`	`/api/poll/<id>/status`	10s	60/min	Quick status check
`GET`	`/api/poll/<id>/analytics`	—	30/min	Timeline + word cloud

Example `/api/poll/<id>/live_stats` response:

```
{ "success": true, "poll_id": "a1b2c3d4e5f6...", "total_votes": 42, "is_active": true, "is_closed": false, "question": "What's your favorite color?", "updated_at": "2026-02-20T16:30:00", "results": [ {"id": 1, "option_text": "Red", "vote_count": 15, "percentage": 35.7}, {"id": 2, "option_text": "Blue", "vote_count": 27, "percentage": 64.3} ] }
```

---

9. Security Architecture

Defense in Depth (7 Layers)

```
Layer 1: Transport → HTTPS only (HSTS max-age=31536000 in production) Layer 2: Application → CSP, X-Frame-Options, X-Content-Type-Options Layer 3: Input → CSRF tokens (Flask-WTF) + Bleach HTML sanitization Layer 4: Rate Limiting → Flask-Limiter (30 votes/min, 10 AI/min, 60 stats/min) Layer 5: Authentication → PBKDF2-SHA256 passwords, HttpOnly session cookies Layer 6: Data at Rest → AES-256-GCM for API keys; SHA-256 for voter IDs Layer 7: Query Safety → SQLAlchemy ORM (parameterized queries, zero raw SQL)
```

Security Headers (Every Response)

```
X-Frame-Options: SAMEORIGIN X-Content-Type-Options: nosniff X-XSS-Protection: 1; mode=block Referrer-Policy:
strict-origin-when-cross-origin Permissions-Policy: geolocation=(), microphone=(), camera=()
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' ... Strict-Transport-Security:
max-age=31536000; includeSubDomains (production only) Cache-Control: no-store (on /login, /register, /settings,
/dashboard)
```

Embed exception: `/poll/<id>/embed` removes `X-Frame-Options` and sets `frame-ancestors *` so the widget can be iframed from any origin.

---

10. Real-Time Update Mechanism

Evolution

Version	Approach	Problem
v1.0	Flask-SocketIO + Eventlet (WebSockets)	PythonAnywhere doesn't support WebSockets; Eventlet breaks on Python 3.13+
v1.1	Server-Sent Events (SSE)	SSE holds threads open indefinitely → exhausts worker pool → dashboard won't load
v2.0	**Short Polling (current)**	■ Works everywhere, no thread blocking, simple

How Short Polling Works

```
Client (poll.html / results.html) ■ ■ setInterval(3000ms) ■ ■■■→ GET /api/poll/<id>/live_stats (server cache:
2s) ■ Returns: { total_votes, is_active, is_closed, ■ question, updated_at, results[] } ■ ■ Client-side logic: ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ ■ if updated_at changed → reload entire page ■ ■ ■
if total_votes changed → update bars/charts ■ ■ ■ if is_active changed → Toast + reload ■ ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ ■■■→ Repeat every 3 seconds (stops on
window.beforeunload)
```

Why This Works

Concern	Solution
Thread exhaustion	Each request completes instantly (~5ms); no held-open connections
Platform compatibility	Standard HTTP GET — works on PythonAnywhere, Render, Vercel, shared hosts
Settings detection	`updated_at` timestamp changes on edit/close/reopen/toggle → client detects and reloads

Vote count updates	`total_votes` comparison → update charts in-place without full reload
Server load	2-second server-side cache means even 100 clients polling = ~1 DB query/2s per poll

---

11. Scalability & Growth Path

Phase	Users	Changes
Current	0 – 1,000	PythonAnywhere, MySQL (single), in-memory cache, 3s polling
Phase 2	1,000 – 10,000	+ Redis caching, 4 workers × 4 threads, MySQL connection pooling, CDN for static assets, 5s polling interval
Phase 3	10,000 – 50,000	AWS ECS / Cloud Run, MySQL read replicas, Redis Cluster, Load Balancer, Celery for async AI, WebSocket upgrade, S3 + CloudFront

Performance Characteristics

Metric	Current	Target at 10K
Poll creation	< 200ms	< 300ms
Vote casting	< 100ms	< 150ms
Update detection	≤ 3s	≤ 5s
AI generation	2–8s (provider-dependent)	2–8s (async queue)
Page load (first paint)	< 1s	< 1.5s

---

12. Trade-offs & Decisions Log

Decision	Alternative Considered	Why We Chose This
Flask over Django	Django (built-in admin, ORM)	Flask is lighter; Blueprint structure gives organization without overhead

Jinja2 (server-rendered) over SPA	React/Vue SPA	Faster initial load, better SEO, no build step, simpler deployment
Short polling over WebSockets	Flask-SocketIO + Eventlet	Works on all platforms; no thread blocking; 3s latency is acceptable
Short polling over SSE	Server-Sent Events	SSE holds threads open → exhausts workers on limited hosting
PythonAnywhere over AWS	EC2, ECS, Lambda	No infra management; built-in MySQL; deploy in 60 seconds
MySQL over PostgreSQL	PostgreSQL	Available on PythonAnywhere; excellent read performance for vote counts
Session-hash over IP tracking	IP-based dedup	IPs are PII; shared IPs block legitimate votes; session hash is anonymous
AES-256-GCM over env vars	Environment variables for API keys	Per-user keys need per-row encryption; env vars are per-deployment
Multi-provider AI	Hardcode Gemini only	Users choose provider for cost, privacy, or capability
`updated_at` for change detection	Hash-based or event-driven	Simple timestamp comparison; one column covers all settings changes
Custom CSS over Bootstrap/Tailwind	CSS frameworks	Zero dependencies, sub-100KB pages, full design control
32-char hex IDs over UUIDs	UUID v4	Shorter URLs; 128 bits entropy; no hyphens
Denormalized vote counts	`COUNT(*)` on read	Avoids expensive aggregation on every page load / poll cycle

---

13. Working Application

Live URL

■ **https://jayachandranpm.pythonanywhere.com**

Deployment Details

Item	Value

Platform	PythonAnywhere
Python version	3.12
WSGI server	Gunicorn (2 workers, 4 threads)
Database	MySQL 8.0 (PythonAnywhere managed)
SSL	Auto-provisioned TLS certificate
Domain	`jayachandranpm.pythonanywhere.com`

Dependencies (16 packages)

Package	Version	Purpose
Flask	3.1.2	Web framework
Flask-SQLAlchemy	3.1.1	ORM
Flask-WTF	1.2.2	Forms & CSRF
Flask-Limiter	4.1.1	Rate limiting
Flask-Migrate	4.1.0	DB migrations
Flask-Login	0.6.3	Authentication
Flask-Caching	2.3.1	Response caching
gunicorn	24.1.1	WSGI server
cryptography	46.0.3	AES-256 encryption
bleach	6.3.0	Input sanitization
qrcode	8.2	QR code generation
requests	2.32.5	HTTP client (AI APIs)
python-dotenv	1.2.1	Environment variables
email_validator	2.3.0	Email validation
mysql-connector-python	9.5.0	MySQL driver
redis	7.1.0	Cache backend

Project Structure

```
POLL PAL/ ■■■ app.py # Flask app factory, middleware, security headers ■■■ config.py # Environment-based config (Dev/Production) ■■■ extensions.py # Flask extension initialization ■■■ models.py # SQLAlchemy: User, Poll, PollOption, Vote ■■■ forms.py # Flask-WTF form definitions ■■■ utils.py # ID generation, hashing, sanitization ■■■ encryption.py # AES-256-GCM encryption module ■■■ ai_service.py # Multi-provider AI (Gemini/OpenAI/Claude) ■■■ ai_prompts.py # Centralized AI prompt templates ■■■ tasks.py # Background task utilities ■■■ Procfile # Gunicorn deployment command ■■■ requirements.txt # 16 pinned Python dependencies ■■■ PRODUCT_ARCHITECTURE.md # this document ■■■ pollivu_wireframe.svg # Wireframes / prototype ■■■ tables.drawio.png # Database tables visualization ■ ■■■ blueprints/ # Modular route handlers ■ ■■■ auth/routes.py # login, register, settings ■ ■■■ dashboard/routes.py # User dashboard ■ ■■■ polls/routes.py # Poll CRUD, voting, embed, AI suggest ■ ■■■ api/routes.py # JSON API (AI, live stats, analytics) ■ ■■■ main/routes.py # Landing page, public routes ■ ■■■ services/ # Business logic layer ■ ■■■ poll_service.py # Poll creation, voting, management ■ ■■■ config_validation.py # Environment variable validators ■ ■■■ templates/ # Jinja2 HTML templates (20+ files) ■■■ static/css/ # Custom stylesheets (11 files, no framework) ■■■ static/js/ # Client-side JavaScript ■■■ static/images/ # Logo and assets ■■■ migrations/ # Alembic database migrations ■■■ tests/ # Test suite (pytest)
```

Pollivu v2.0 · Product & Architecture Document · February 2026 Prepared for technical review and project evaluation