Design Document

Mads S. Balto

June 10, 2025

1 Introduction

`#below-emerald` is a Discord channel devoted to focused League of Legends VOD questions. Users typically provide

- champion played,
- current learning objective,
- a concise question,
- an optional self-assigned *division* role.

This project collects questions *only from users who have explicitly opted-in.* The long-term goal is to surface descriptive statistics such as "which champions Gold players most frequently struggle with" or "how objectives differ by rank."

2 GDPR Compliance

With explicit consent, we store *encrypted message text* and a *hashed user ID.* No direct identifiers (plain user IDs, usernames, or message IDs) are stored. Processing follows the principles of the General Data Protection Regulation (GDPR).

2.1 Data-Protection Principles

- **Lawfulness, Fairness, Transparency:** Users invoke `/consent`; the bot presents a clear summary of data use before they opt-in.
- **Purpose Limitation:** Data is used solely to analyse question trends by champion and rank.
- **Data Minimisation:** Only three fields are stored: encrypted message text, SHA-256 user-ID hash, and a deduplication hash (`row_hash`).
- **Accuracy:** A live lookup against the consent registry ensures we never store messages from users who have revoked consent.

- **Storage Limitation:** All data is deleted on study completion or individual withdrawal.
- **Integrity & Confidentiality:** Messages are encrypted with AES-256-CBC using a 32-byte key loaded from `.env`. Data never leaves the local machine.
- **Accountability:** The data controller is Mads S. Balto, who maintains this document and the audit log.

2.2 Data-Subject Rights

Participants may: be informed, access their data, withdraw consent, or request deletion at any time.

2.3 Legal Basis

Processing is based on freely given, explicit consent (GDPR Art. 6(1)(a)). Under-13 users are excluded by policy.

2.4 Security Measures

- Local SQLite file (`project_data.db`) with file-system ACLs.
- AES-256 encryption (random IV per message, Base64 encoding for storage).
- `ENCRYPTION_KEY` kept in `.env`; never committed.
- Audit trail in `consent_log` (encrypted user ID, action, timestamp).
- Breach notification within 72 h if ever required.

3 Design

3.1 Consent System

`/consent` displays a summary plus context-sensitive buttons:

- **Not yet consented:** *Opt In*, *Dismiss.*
- **Already consented:** *Retract Consent*, *Dismiss.*

Consent Registry

```
Table: consent_registry
Columns:
  user_id_hash TEXT PRIMARY KEY         -- SHA-256(user_id)
```

Logging

```
Table: consent_log
Columns:
  id          INTEGER PRIMARY KEY
  user_id_enc TEXT    -- AES-256 encrypted user ID
  action      TEXT    -- 'gave consent' | 'retracted consent'
  timestamp   TEXT
```

3.2 Message Collection

`/collect` is a *manual* command limited to one channel. For each message in history:

1. Hash author ID → `user_id_hash`.
2. If hash not in `consent_registry` ⇒ skip.
3. Encrypt message text → `message_enc`.
4. Compute `row_hash` = SHA-256(`user_id_hash` ∥ *plaintext message*).
5. Insert into `data`. Duplicate `row_hash` is ignored.

Fail-closed: if the consent table is missing/corrupted, nothing is collected.

3.3 Data Schema

```
Table: data
Columns:
  id            INTEGER PRIMARY KEY
  user_id_hash  TEXT    -- SHA-256(user ID)
  message_enc   TEXT    -- Base64(iv | ciphertext)
  row_hash      TEXT UNIQUE  -- SHA-256(user_id_hash || plaintext)
```

No message IDs or division roles are stored; rank and champion are inferred later from the encrypted text once decrypted during analysis.

4 Technical Stack

- **Python 3.10+** with `discord.py` 2.x
- AES-256 (`cryptography` library) and SHA-256 (`hashlib`)
- SQLite 3.41 (autocommit connections, 10 s timeout)
- Minimal Discord permissions: scope `bot` only; access confined via a zero-permission role granted *read* on a single channel.