1. Demo Scenario Overview

Describe what your final demo will include.

* What specific features will you present?

In the final demo, we'll walk through the main features that bring PiConfess+ to life as an anonymous confession platform, **including:**

* + **Anonymous Confession Submission:** Users can submit text-based confessions with optional emoji reactions.
  + **Search Functionality:** Users can search for names or keywords within previously posted confessions.
  + **Emoji Reaction System:** Users can react to confessions using one of four emojis: ❤️, 😂, 😭, 😳.
  + **Trending Confessions display:** A dynamic section displays the top 3 confessions with the highest total reactions.
  + **NSFW content filtering:** Basic profanity filtering ensures confessions do not include inappropriate language.
* What user actions will be shown?

During the demo, the user interactions will be demonstrated **as:**

* + **Submitting a Confession**
    - A user types a confession and clicks the “Submit” button.
    - Example: “I secretly like Ryan, but I can never tell him 😭”
  + **Searching Confessions**
    - The user enters a keyword like “Ryan” in the search bar and clicks “Search.”
  + **Reacting to a Confession**
    - A user clicks an emoji (e.g., ❤️) on a displayed confession to react to it.
  + **Viewing Trending Confessions**
    - The user scrolls to or clicks into the **Trending Confessions** section to view the most popular posts.
* What parts of the application will be functional (even if other parts are not yet finished)?

| **Feature** | **Functionality** |
| --- | --- |
| Confession Submission | Implemented using PHP; checks for NSFW words before saving to MariaDB. |
| Keyword Search | PHP fetches confessions from MariaDB based on keyword matches. |
| Emoji Reactions | PHP increments reaction count in MariaDB and updates in real-time. |
| Trending Confessions | PHP fetches and displays top 3 confessions based on total reaction counts. |
| NSFW Filtering | Basic text filter implemented in PHP to reject inappropriate content. |

1. Planned URL Endpoints

List all the web pages or API endpoints your application will support for the demo.

For each endpoint, provide:

* URL path (e.g., /login.php, /submit\_form.php)
* Supported HTTP method(s) (GET, POST)
* Expected HTTP variables (from form or query string)
* Session variables that are used or required
* Database tables/operations involved (e.g., fetch user info, insert new record)

| **URL Path** | **Method** | **What It Does** | **Input Variables** | **Session Needed** | **Database Actions** |
| --- | --- | --- | --- | --- | --- |
| /submit\_confession.php | POST | Save a new anonymous confession | confession\_text | No | NSFW check → Insert into confessions table |
| /search.php | GET | Search confessions by name or keyword | query | No | SELECT from confessions WHERE text includes keyword |
| /react.php | POST | Add a reaction (❤️, 😂, 😭, 😳) to a confession | confession\_id, reaction\_type | No | UPDATE reaction count in confessions table |
| /trending.php | GET | Show most reacted (trending) confessions | None | No | SELECT top 3 from confessions ORDER BY total reactions |
| (part of submit) NSFW Filter | (internal) | Check confession for bad words | confession\_text | No | No DB action (text is filtered before saving) |

1. Database Design
2. **Entity-Relationship Diagram (ERD)**

* Draw an ERD that represents the entities and their relationships involved in your application.

### **📘 Entity-Relationship Diagram (ERD)**

**Entities:**

* Confession
* Reaction

**Relationships:**

* One Confession can have **many** Reactions
* Each Reaction is linked to one Confession

1. **Relational Model**

Translate your ERD into relational tables, including:

* Table names
* Column names and data types
* Primary keys and foreign keys

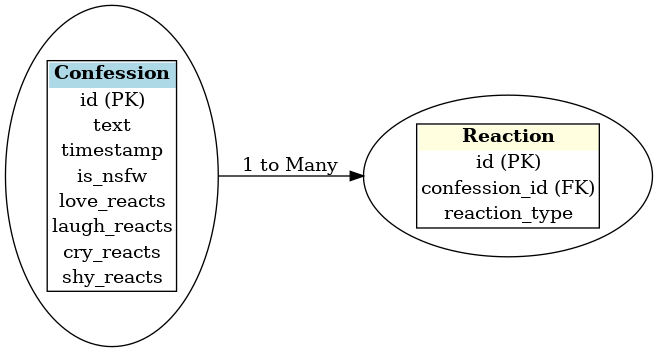
🗃️ **Relational Model (Tables & Fields)**

**🟦 confessions table**

| **Column Name** | **Data Type** | **Notes** |
| --- | --- | --- |
| id | INT AUTO\_INCREMENT | Primary Key |
| text | TEXT | The full confession text |
| timestamp | DATETIME | When the confession was made |
| is\_nsfw | BOOLEAN | True if flagged as NSFW |
| love\_reacts | INT DEFAULT 0 | ❤️ count |
| laugh\_reacts | INT DEFAULT 0 | 😂 count |
| cry\_reacts | INT DEFAULT 0 | 😭 count |
| shy\_reacts | INT DEFAULT 0 | 😳 count |

**🟦 reactions table**

| **Column Name** | **Data Type** | **Notes** |
| --- | --- | --- |
| id | INT AUTO\_INCREMENT | Primary Key |
| confession\_id | INT | Foreign Key → confessions.id |
| reaction\_type | ENUM('love', 'laugh', 'cry', 'shy') | Reaction emoji |



1. **Normalization**

Briefly explain how your relational model meets Third Normal Form (3NF).

* Identify any design decisions you made to remove redundancy or ensure data integrity.

### **Normalization (Up to Third Normal Form - 3NF)**

#### **💡 What is 3NF?**

**Third Normal Form means:**

1. No repeating groups (1NF)
2. All columns depend on the primary key (2NF)
3. No transitive dependencies (3NF)

#### **✔ How PiConfess+ meets 3NF:**

* **1NF:** Every column stores atomic values (e.g., each reaction type is a separate field).
* **2NF:** All non-key fields (like text, timestamp, emoji counts) depend on the id of the confession.
* **3NF:** There are no derived or calculated fields stored directly (e.g., total reactions can be calculated on the fly), and there's no redundant data like storing user identity (since it’s anonymous).

#### **🔧 Design Decisions for 3NF:**

* Choose to store reaction counts directly in the **confessions** table for performance (no JOIN needed to get counts).
* Added **is\_nsfw** as a flag instead of storing filtered/conflicting content in a separate table — this keeps moderation simple.
* Kept user info out of the schema (since the app is anonymous by design).