**Database Schema Design**

**1. Users Table (website)**

Stores user account information.

| **Column** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| user\_id | UUID | Primary Key, NOT NULL | Unique identifier for each user. |
| username | VARCHAR(100) | NOT NULL | User's unique display name. |
| Email | VARCHAR(255) | NOT NULL, UNIQUE | User's email address. |
| password\_hash | VARCHAR(255) | NOT NULL | Hashed password for security. |
| auth\_provider | VARCHAR(50) | DEFAULT 'local' | Authentication type (e.g., local, OAuth). |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Account creation timestamp. |
| updated\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | Last update timestamp. |

**2. Queries Table (website)**

Stores user-submitted queries.

| **Column** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| query\_id | UUID | Primary Key, NOT NULL | Unique identifier for each query. |
| user\_id | UUID | Foreign Key (Users.user\_id), NOT NULL | User who submitted the query. |
| query\_text | TEXT | NOT NULL | User's input query or code snippet. |
| response\_text | TEXT |  | LLM-generated response. |
| status | ENUM('pending', 'completed', 'failed') | DEFAULT 'pending' | Current processing status. |
| submitted\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | When the query was submitted. |
| processed\_at | TIMESTAMP |  | When the query was processed. |

**3. Problems Table (model)**

Stores metadata about competitive programming problems.

| **Column** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| problem\_id | UUID | Primary Key, NOT NULL | Unique identifier for each problem. |
| Title | VARCHAR(200) | NOT NULL | Problem title. |
| difficulty | ENUM('easy', 'medium', 'hard') | DEFAULT 'medium' | Problem difficulty level. |
| Tags | JSONB |  | Tags (e.g., ["dynamic programming", "greedy"]). |
| description | TEXT |  | Detailed problem description. |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | When the problem was added. |

**4. Solutions Table (model)**

Stores solutions to problems.

| **Column** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| solution\_id | UUID | Primary Key, NOT NULL | Unique identifier for each solution. |
| problem\_id | UUID | Foreign Key (Problems.problem\_id), NOT NULL | Related problem ID. |
| solution\_code | TEXT | NOT NULL | Submitted solution code. |
| language | VARCHAR(50) | NOT NULL | Programming language used. |
| submitted\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | Submission timestamp. |
| is\_correct | BOOLEAN | DEFAULT NULL | Whether the solution passed validation. |

**5. Chat Logs Table (website)**

Logs chat interactions for debugging and analytics.

| **Column** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| chat\_id | UUID | Primary Key, NOT NULL | Unique identifier for the chat log. |
| user\_id | UUID | Foreign Key (Users.user\_id) | Related user. |
| query\_id | UUID | Foreign Key (Queries.query\_id) | Associated query. |
| message | TEXT | NOT NULL | Message content (from user or system). |
| sender | ENUM('user', 'system') | NOT NULL | Message sender. |
| timestamp | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | When the message was logged. |

**6. Knowledge Base Table (website & model)**

Stores competitive programming knowledge snippets and resources.

| **Column** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| kb\_id | UUID | Primary Key, NOT NULL | Unique identifier for each entry. |
| Title | VARCHAR(200) | NOT NULL | Title of the resource or snippet. |
| content | TEXT | NOT NULL | Knowledge content or explanation. |
| Tags | JSONB |  | Tags (e.g., ["binary search", "tips"]). |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | When the entry was created. |

**7. System Metrics Table (website monitoring)**

Stores data for monitoring and analytics.

| **Column** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| metric\_id | SERIAL | Primary Key, NOT NULL | Unique identifier for each metric. |
| metric\_name | VARCHAR(100) | NOT NULL | Metric name (e.g., API latency). |
| metric\_value | FLOAT | NOT NULL | Value of the metric. |
| timestamp | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | When the metric was logged. |

**8. Settings Table (website)**

Stores user-specific and global settings.

| **Column** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| setting\_id | UUID | Primary Key, NOT NULL | Unique identifier for the setting. |
| user\_id | UUID | Foreign Key (Users.user\_id), NULL | If null, global setting. |
| key | VARCHAR(100) | NOT NULL | Setting name (e.g., theme). |
| value | JSON |  | Value of the setting (e.g., dark). |
| updated\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | Last update timestamp. |

**Relationships**

1. **Users → Queries** (1-to-Many): A user can submit multiple queries.
2. **Users → Chat log** (1-to-Many): A user has multiple chat logs.
3. **Users → Settings(1-to-1):** each user has only one settings set
4. **Queries → Chat Logs** (1-to-Many): Each query generates multiple chat messages.
5. **Problems → Solutions** (1-to-Many): A problem can have multiple solutions.
6. **Knowledge Base**: Independent, queried by LLM for context.

This design is modular, scalable, and ensures data integrity with foreign key constraints and structured schema.