Poetry Enjoy Life

Database Design Document

1. Overview of Database Design

This database is designed to provide user management functions for the "Poetry Enjoy Life" webpage, supporting user registration, login, and interactions with a poetry robot. The design also encompasses aspects of security and scalability.The core requirements include: :

User Information Management: Storing basic user information (username, email, etc.) and encrypted passwords.

Security: Ensuring the security of user passwords by avoiding storing plaintext passwords.

Scalability: The design should facilitate future additions such as user role management and activity tracking.

2. Selected Database

Considering the scale and requirements of the project, we have chosen MySQL as the backend database system. MySQL is a widely used relational database management system known for its high performance, reliability, and ease of use, making it suitable for small to medium-sized applications.

3. E-R Diagram

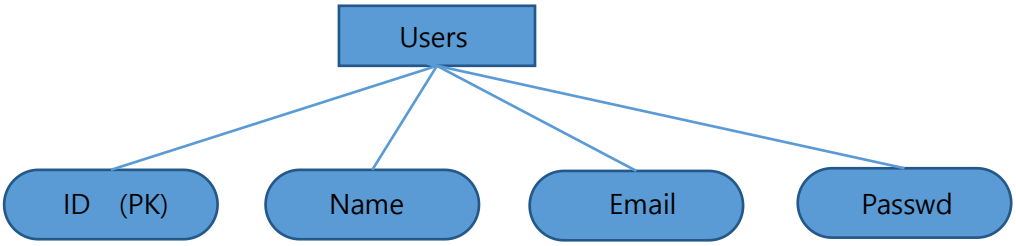
In this simple user management system, the main entity is the User, which includes the following attributes:

ID (Primary Key, Auto Increment)

Username (Unique)

Email (Unique)

Password (Encrypted Storage)



4. Implementation Details and Table Structure

4.1 Database Initialization

First, create the database file:

```
CREATE DATABASE poetry_life;
USE poetry_life;
```

4.2 User Table Structure

Next, create the users table to store user information:

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

列名	数据类型	属性	描述
id	INT	PRIMARY KEY, AUTO_INCREMENT	用户 ID，主键
username	VARCHAR(50)	NOT NULL, UNIQUE	用户名，唯一
email	VARCHAR(10)	NOT NULL, UNIQUE	邮箱，唯一
password	VARCHAR(255)	NOT NULL	密码，加密存储
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间戳

4.3 Password Encryption Handling

For security purposes, user passwords should be encrypted before being stored in the database. Algorithms such as bcrypt can be used to hash the passwords before storing them.

```
// 注册接口
app.post('/register', async (req, res) => {
    const { username, email, password } = req.body;

    // 检查邮箱是否已存在
    pool.execute('SELECT * FROM users WHERE email = ?', [email], async (err, results) => {
        if (err) {
            return res.json({ status: 'error', message: 'Database error' });
        }

        if (results.length > 0) {
            return res.json({ status: 'error', message: 'Email already in use' });
        }

        // 哈希密码
        const hashedPassword = await bcrypt.hash(password, 10);

        // 插入新用户
        pool.execute('INSERT INTO users (username, email, password) VALUES (?, ?, ?)',
            [username, email, hashedPassword], (err, results) => {
            if (err) {
                return res.json({ status: 'error', message: 'Database error' });
            }
            res.json({ status: 'success', message: 'User registered successfully' });
        });
    });
});
```

4.4 Security Considerations

SQL Injection Protection: Use parameterized queries or ORM frameworks to prevent SQL injection attacks.

Password Encryption: As mentioned above, use strong hash functions to encrypt passwords, ensuring that even if the database is compromised, the passwords are not easily cracked.

Data Validation: Strictly validate user input at the application layer to ensure data

legality and security.

Through the above design and implementation, we can provide a secure, efficient, and scalable user management system for the "Poetry Enjoy Life" webpage.