The User Manual of Liru Online Forum

Date: December 4th, 2024

Chapter 1 Overview

1.1 Software System Overview

This system is a proof-of-concept (PoC) campus blogging platform developed using Django, aimed at providing a secure, convenient, and user-friendly platform for the campus community. Through this platform, users can freely share academic insights, campus news, and personal stories. The system's core functionalities include user registration and login, blog post creation and management, comment interaction, and content search. Developed with Django, the system leverages its powerful ORM (Object-Relational Mapping), templating engine, and user authentication mechanisms to ensure data security and optimize user experience.

The PoC version is designed to validate the feasibility of the system's basic functionalities, with a focus on demonstrating the operation of core modules such as database interactions, frontend-backend communication, and user management. The system's architecture, built on the Django framework, ensures scalability and maintainability. Through this proof-of-concept, the technical requirements for the system are clarified, laying the groundwork for future feature expansions.

The platform is designed with a simple and intuitive user interface, making it accessible to non-technical users. Users can access the platform through a web browser and complete tasks such as registration, login, blog posting, and commenting. The backend, powered by Django, provides robust management capabilities, ensuring data integrity and user information security.

1.2 Target Users

The primary users of this system include students, teachers, and other staff within the campus community. Teachers can share academic resources via the blog platform, while students can document and share their campus life. The entire campus community can engage in knowledge sharing and discussions through this platform. The system is particularly suited for users interested in content creation and knowledge dissemination, providing an open and secure environment that fosters intellectual exchange.

Although this PoC system is mainly intended for technical evaluation and feature testing, special attention has also been given to user experience and interaction design. The target users at this stage include developers and testers who will assess whether the system's core functionalities meet expectations and provide feedback for further development and enhancements.

Chapter 2 System Summary

2.1 System Operating Environment Requirements

2.1.1 Operating System Requirements

The system supports multiple operating systems to ensure maximum compatibility and user coverage:

Windows 10 or higher: For Windows users, ensuring the platform’s broad availability.

macOS 10.15 or higher: Provides smooth experiences for Mac users, supporting applications within the Apple ecosystem.

Linux (e.g., Ubuntu 20.04 or higher): Supports open-source operating systems, catering to the needs of developers and technical users.

Supporting these operating systems ensures that the platform runs stably across different platforms, providing a consistent user experience regardless of the operating system used.

2.1.2 Browser Requirements

To ensure that the system's user interface performs consistently across devices, we recommend accessing the platform using the following browsers:

Google Chrome (recommended): Known for its strong compatibility and support for modern web standards.

Mozilla Firefox: A secure and reliable alternative, popular among technical users and developers.

Safari: Optimized for macOS and iOS, ensuring the best experience for Apple devices.

Microsoft Edge: A robust choice for Windows system users with strong compatibility.

Using these modern browsers ensures smooth loading speeds and seamless user interaction. We advise users to keep their browsers up to date to maintain optimal performance and security.

2.1.3 Development Tools Requirements

The system was built using several development tools and environment configurations to ensure efficient coding and debugging capabilities:

Pycharm Professional Edition: This serves as the primary development environment, offering strong support for Django development. Its built-in debugging tools and database management features simplify both the debugging process and data handling.

MySQL 8.0: The system uses MySQL as the database management system for storing and managing data, including blog posts, user information, and comments. MySQL's efficiency and broad compatibility make it ideal for this platform.

Navicat: To simplify database management, developers can use tools like Navicat for visualizing SQL queries and performing operations on database tables. Navicat offers an intuitive interface, greatly enhancing development efficiency.

Git: The Git version control system ensures traceability and collaboration during development. Development teams can manage code remotely through platforms like GitHub or GitLab, ensuring team members are in sync.

These development tools together form the system's development environment, providing developers with a streamlined and efficient workflow. For developers looking to expand or modify the system, it is recommended to use the tools mentioned above for optimal development and maintenance.

2.2 User Access Levels

To ensure the system's security and stability, different access levels are assigned based on user roles, ensuring that users can only perform actions within their scope of permissions.

2.2.1 Visitors

Access Rights: Visitors are unregistered users who can freely browse the blog content on the site, including posts and comments.

Restrictions: Visitors cannot publish blog posts, comment on articles, or access registered users' personal profiles or management features.

Use Case: Suitable for temporary users interested in reading blog content without creating an account.

2.2.2 Registered Users

Access Rights: Registered users, who have verified their email, have elevated permissions that allow them to publish, edit, and manage their own blog posts, view and comment on others' posts, and manage their own profiles.

Management Capabilities: Registered users can modify or delete their own content and interact with other users via the comment system.

Use Case: Ideal for frequent users who want to share their personal insights or engage with others.

2.2.3 Administrators

Access Rights: Administrators have the highest level of access in the system, with full control over all users, posts, and comments. They can delete inappropriate content, review articles, and manage user accounts.

Management Capabilities: Administrators are responsible for the platform's daily maintenance, ensuring stable operation, and can monitor system performance through the backend.

Use Case: Suitable for system administrators or the development team responsible for content moderation and user management.

By designing this multi-tiered user access structure, the system ensures that each user role is assigned the necessary functionality while maintaining platform security and stability.

Chapter 3 User Guide

This chapter provides detailed instructions on how to use the system, including user registration, blog post creation, commenting, and environment setup. Since our blog system relies on a local database, we acknowledge certain limitations in our technical knowledge, and invite you to follow the steps below to ensure the system is properly installed and configured.

3.1 Environment Setup

Before you can enjoy the features of posting blogs and interacting with others, it’s necessary to configure the system environment. This includes installing MySQL, Navicat for visual database management, and setting up the necessary Python packages and modules. Please follow the steps below:

3.1.1 Installing MySQL

Our system uses MySQL as the backend database. Follow these steps to install it:

Step 1: Visit the MySQL official site and download MySQL version 8.0.

Step 2: Select the appropriate installer for your operating system (Windows, macOS, etc.).

Step 3: Follow the installation wizard, and we recommend choosing the "Server Only" mode and setting a root password to ensure database security.

Step 4: Once installation is complete, start the MySQL service and verify that you can access the database via the command line.

3.1.2 Installing Navicat

Navicat is a tool we recommend for managing the MySQL database. It simplifies operations such as table management and queries.

Step 1: Visit the Navicat official site and download the version suited for your operating system.

Step 2: After installation, start Navicat and log in to MySQL using the root account to ensure a successful connection.

Step 3: Create a new database within Navicat for use with the Django system.

3.1.3 Installing Python Packages

To ensure the Django system operates correctly, you will need to install the necessary Python packages:

Step 1: Ensure that Python 3.9 or a higher version is installed on your machine.

Step 2: Use pip to install Django and the MySQL client by running the following commands:

```
pip install django mysqlclient
```

Step 3: Verify the installation of Django and the MySQL client by running the following commands:

```
django-admin --version
python -c "import MySQLdb"
```

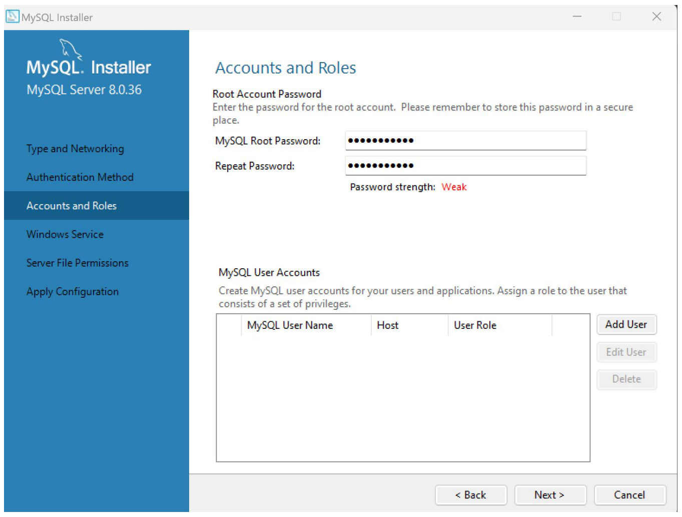
Once these steps are completed, you are ready to proceed with using the system.

3.1.4 Detailed Instructions

To ensure the local database is correctly connected to the Django project, please follow these steps to configure MySQL and Navicat, and establish a connection with the Django project.

I.MySQL Database Setup

During the MySQL installation process, you will be prompted to set a password for the root user. Please remember this password, as it will be used in subsequent steps to connect the database to the Django project. If you have already set the password, proceed to the next steps.


MySQL Installer

MySQL. Installer
MySQL Server 8.0.36

Accounts and Roles

Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password: ●●●●●●●●●●●
Repeat Password: ●●●●●●●●●●●
Password strength: Weak

MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role

II.Connecting MySQL with Navicat

Navicat will be used to manage the MySQL database, and it must be configured with the correct connection details:

Open Navicat, click on "Connection," and select "MySQL."

In the connection setup window, enter the following information:

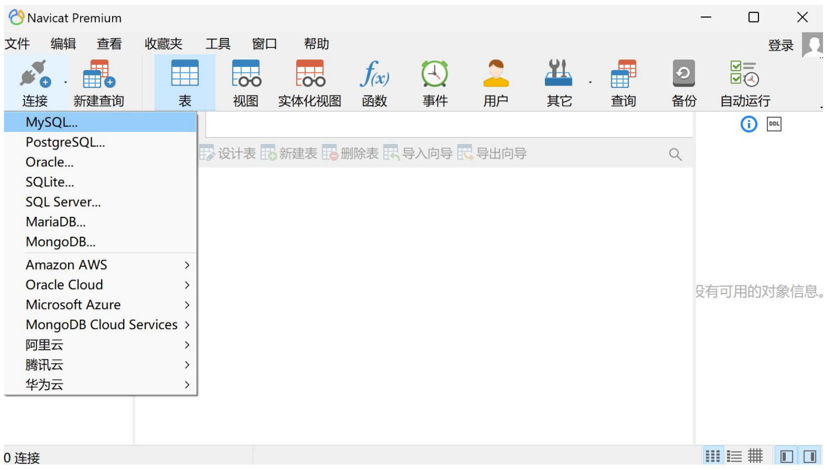
Host: localhost

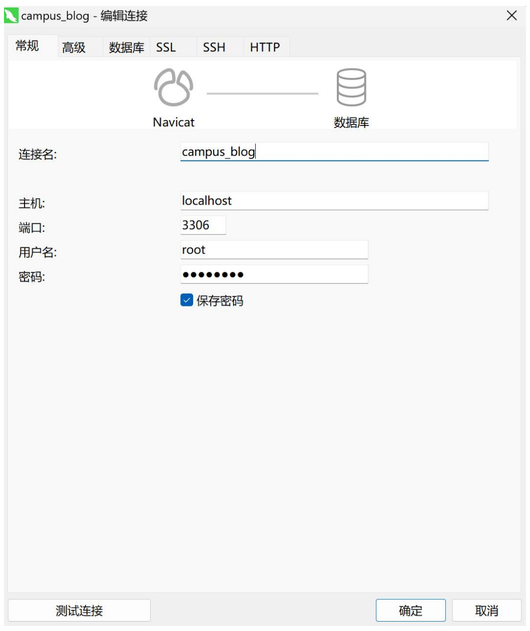
Port: 3306 (default)

Username: root

Password: Enter the password you set during the MySQL installation process (if using the Django project's default configuration, the password is 15728393542).

Click "Test Connection" to ensure Navicat successfully connects to the MySQL database.





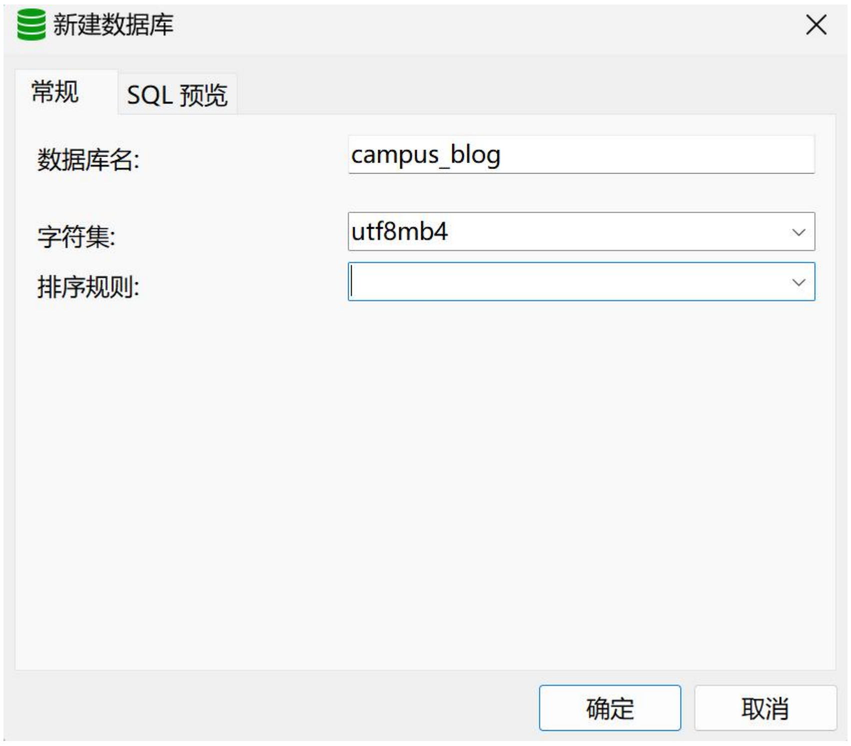
连接名: campus_blog
主机: localhost
端口: 3306
用户名: root
密码: ••••••••
☑ 保存密码

III.Creating a New Database

Once connected, you need to create a database named campus_blog in Navicat for use with the Django project:

In Navicat, right-click on the connection name and select "New Database."

Enter the database name campus_blog and confirm the creation.



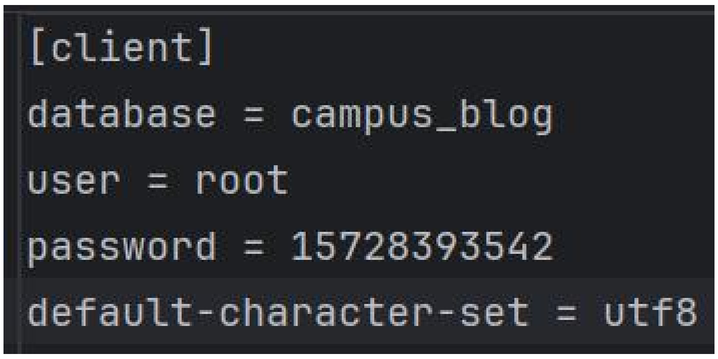
数据库名: campus_blog

字符集: utf8mb4

排序规则:

IV.Django Project Configuration

In the Django project's configuration file (cnf file), the username is set to root and the password to 15728393542 by default. If you set a different password during the MySQL installation, please update the relevant database connection details in the Django configuration file to match the MySQL settings.



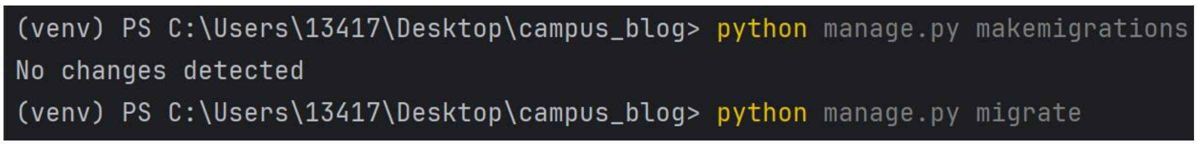
```
[client]
database = campus_blog
user = root
password = 15728393542
default-character-set = utf8
```

V.Database Migrations

After completing the above database setup, you need to run database migration commands in the Pycharm terminal to synchronize the Django project with the database structure:

First, navigate to the project's root directory and run the following command to generate the migration files:

Next, apply the migration changes to the database:



```
(venv) PS C:\Users\13417\Desktop\campus_blog> python manage.py makemigrations
No changes detected
(venv) PS C:\Users\13417\Desktop\campus_blog> python manage.py migrate
```

By completing these steps, your local database will be successfully connected to the Django project, and you will be able to use the system's features, such as blog posting and commenting.

3.2 User Registration and Login

3.2.1 User Registration

To publish blog posts or comment on articles, you will first need to register an account:

Step 1: Click the "Register" button located at the top right corner of the homepage.

Step 2: Fill in the required fields such as username, email address, and password.

Step 3: Complete the CAPTCHA to ensure the security of the registration process.

Step 4: After submitting, you will receive a verification email. Please click the link in the email to activate your account.

3.2.2 User Login

After registration, you can log in using the following steps:

Step 1: Click the "Login" button.

Step 2: Enter your registered email and password, and then click "Login." After successful login, the system will redirect you to the homepage where you can begin using the platform.

3.2.3 Logout

You can log out at any time by clicking the "Logout" button located in the top right corner of the interface.

3.3 Blog Post Creation and Publishing

3.3.1 Create and Publish a Blog Post

Once logged in, you can create and publish a blog post using the following steps:

Step 1: Click the "Publish Blog" button in the navigation bar.

Step 2: Use the integrated rich text editor to write your post. The editor supports adding images, links, and formatted text.

Step 3: Fill in the title and content of the post, and optionally assign categories or tags to help readers find your content.

Step 4: Click "Publish" to display your post on the blog homepage.

3.4 Commenting and Interaction

3.4.1 Post a Comment

After reading a blog post, you can post a comment as follows:

Step 1: Find the comment box located at the bottom of the post.

Step 2: Enter your comment and click "Submit Comment." The comment will appear in real-time below the post.

3.4.2 Reply to a Comment

Users can interact by replying to others' comments:

Step 1: Click the "Reply" button below a comment.

Step 2: Enter your reply and click "Submit."

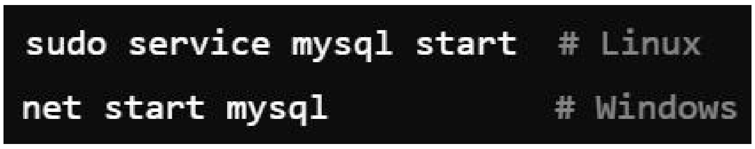
Chapter 4 Frequently Asked Questions (FAQ)

Since this blog system is based on a local database, users may encounter issues related to database installation and connection during use. Below are solutions to common problems to help users resolve technical issues quickly.

4.1 Issues with MySQL Installation and Connection

4.1.1 I cannot start or connect to the database after installing MySQL locally.

Solution: Since the system relies on a local MySQL database, ensure that the MySQL service is correctly running. You can manually start the service with the following commands:



```
sudo service mysql start  # Linux
net start mysql           # Windows
```

Ensure that the root password set during installation is correct. If the password is forgotten, you can reset it via the command line.

Check firewall settings to ensure that MySQL's default port 3306 is not blocked. If it is, unlock this port to allow database connections.

4.1.2 Navicat cannot connect to the local MySQL database.

Solution: Verify that the connection information you entered (such as localhost, port 3306, root password) is correct.Ensure that the MySQL service is running and try testing the connection via the command line:

```
mysql -u root -p
```

4.2 Issues with Python Package Installation

4.2.1 Errors occur when installing Django or mysqlclient.

Solution: When setting up the local environment, ensure that Python 3.9 or higher is installed, and that pip is updated. You can update pip with the following command:

```
python -m pip install --upgrade pip
```

Install Django and mysqlclient using the following commands:

```
pip install django mysqlclient
```

4.2.2 Unable to import mysqlclient.

Solution: mysqlclient is the interface library required to connect the system to the local database. Ensure that the necessary system dependencies are installed before installing mysqlclient. On Windows, you may need to install system libraries manually. On Linux, you can install the dependencies using:

```
sudo apt-get install python3-dev libmysqlclient-dev  # Linux
```

4.3 Issues with Blog Posting

4.3.1 My blog post is not appearing on the homepage.

Solution: Make sure that all required fields, such as title and content, are filled in, and that you clicked the "Publish" button.If the post still does not appear, try refreshing the page or check if the local database is running properly.

4.3.2 Changes made while editing a blog post are not saved.

Solution: Since the system relies on a local database, unstable network conditions may cause saving issues. Ensure a stable network connection and verify that the MySQL service is running.

4.4 Issues with Comments

4.4.1 My comment is not showing under the post.

Solution: Ensure that the comment was successfully submitted and refresh the page to check. If it still doesn’t appear, there may be a local server delay or database processing issue. Check if MySQL is running properly.

4.4.2 I want to delete or edit a published comment.

Solution: Due to system design limitations, currently, published comments cannot be deleted or edited. We recommend reviewing your comments carefully before submitting.