Introduction to Software Engineering

AI Group 18: Liru Online Forum

White Paper

Abstract

This white paper introduces the Liru Online Forum, a campus blog platform developed using the Django framework. The platform is designed to provide a comprehensive solution for campus community members, offering features such as academic resource sharing, blog publishing, campus news, and interactive commenting. Liru Online Forum enables users to easily publish personal blogs, access academic materials, and engage in discussions on campus hot topics. The platform ensures the security and authenticity of content through a rigorous user authentication and content review mechanism, preventing the spread of inappropriate information.

Liru Online Forum employs a modular design with core features including user management, blog publishing, comment interaction, and content search. The backend uses a MySQL database to ensure secure data storage and efficient retrieval, while the frontend incorporates responsive design, ensuring smooth operation across different devices. Additionally, the system supports future expansions such as personalized recommendations and user behavior analysis modules to continuously enhance the user experience.

The key advantages of Liru Online Forum lie in its campus-focused design, featuring strict authentication and content review mechanisms that ensure the reliability of user-published content, fostering a positive campus communication environment. The platform is highly flexible and scalable, allowing for continuous upgrades based on user feedback. The project benefits from the use of open-source technologies to control development costs, while long-term profitability is anticipated through multiple channels, including advertisements, paid content, and premium user services. It is expected that the platform will experience rapid user growth within six months of project completion, reaching profitability within a year.

Context

Chapter 1 Project Overview

1.1 Project Background

With the rapid advancement of the information age, the way academic communication and daily information dissemination take place on campus has gradually shifted from traditional face-to-face interactions to online platforms. However, most existing social media platforms have not fully considered the unique needs of campus communities. Students and faculty members need a centralized platform that not only meets the requirements for academic resource sharing but also allows them to easily publish personal stories and campus news. Although current social media platforms offer information publishing and interaction functionalities, they are not customized for campus environments, leading to severe information fragmentation and an inability to meet the needs for effective knowledge management and dissemination. Moreover, issues related to content security and authenticity further hinder normal communication within the campus community. Therefore, a well-integrated blog platform specifically designed for the campus is essential.

1.2 Project Objectives

The main goal of this project is to create a campus blog platform based on the Django framework, enabling campus community members to publish and manage a variety of content. The platform will support users in creating blog articles, accessing academic resources, and participating in discussions and interactions. Additionally, the project aims to integrate a strict user authentication and content review mechanism to ensure that published content is reliable, guaranteeing the safety of information exchange and ultimately enhancing academic atmosphere and interaction within the campus community.

1.3 Pain Points Analysis

The primary pain points of existing social media and forum platforms are as follows: First, the lack of systematic content review mechanisms leads to frequent occurrences of misinformation and inappropriate content, affecting user experience. Second, the decentralized functions of these platforms make it difficult for users to efficiently manage personal published content, thus reducing the efficiency of information retrieval and sharing. Lastly, the existing platforms have limited interactive features, which cannot meet the needs of campus users for academic discussions and in-depth communication. Campus users urgently need a platform that can effectively integrate these features, address the aforementioned pain points, and enhance information dissemination and interaction efficiency.

1.4 Solution

To address these pain points, this project developed a campus blog platform based on the Django framework. The platform is designed using a modular approach, with core functions including user management, blog publishing, comment interaction, and content search, ensuring that users can smoothly publish and manage blog articles.

The system uses MySQL as the backend database to ensure the secure and efficient storage of data. The frontend employs a responsive design to allow users to seamlessly access and operate the system across different devices. Moreover, the system supports future functionality extensions, such as personalized recommendations and user behavior analysis modules, ensuring continuous optimization based on user needs.

1.5 Incremental Model in Project Development

The incremental model is a development approach where the system is built in several increments (or phases), each delivering part of the functionality. Liru Online Forum adopted this model to implement core features while refining and expanding based on user feedback. The application of the incremental model in this project is as follows:

Initial Increment - User Management Module
The team first developed core functions like user registration, login, and logout, ensuring secure access to the platform using Django's built-in authentication.

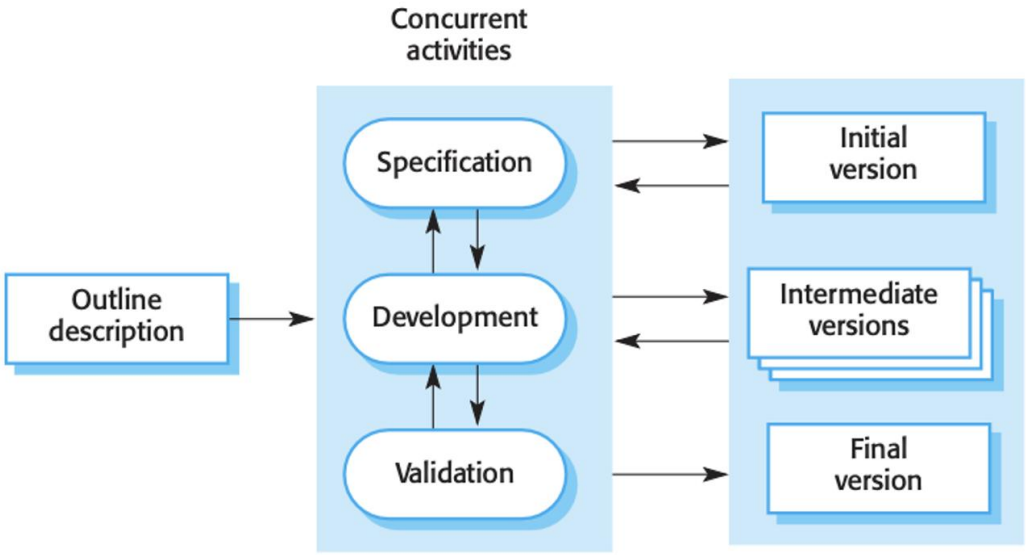
Second Increment - Blog Publishing and Management
After the user management module, the team added blog publishing and management, allowing users to create, edit, and organize content using categories and tags.

Third Increment - Commenting and Interaction
Next, the commenting and interaction features were implemented, enabling users to comment on blog posts with nested replies, enhancing engagement.

Future Increments - Personalized Recommendations and Behavior Analysis
Future increments will include personalized recommendations and user behavior analysis, providing customized content based on user interests.

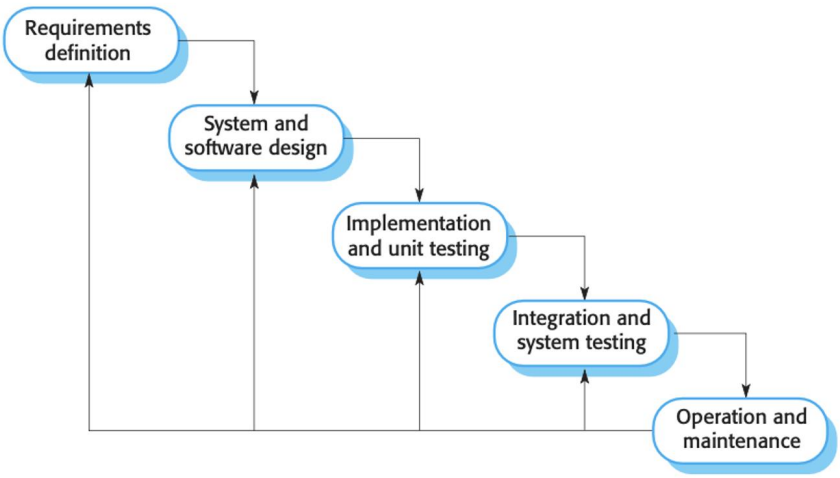
Advantages of the Incremental Model: The incremental model allows independent development, testing, and deployment of each increment, reducing risks and enabling early delivery of core functionalities while responding to user feedback.

Incremental development model



We did not choose the Waterfall Model: Because it follows a rigid, sequential process where each phase must be fully completed before moving to the next. This limits flexibility, making it difficult to incorporate feedback or adapt to changing requirements during development. The incremental model, by contrast, allows for iterative improvements and quicker delivery of core features, better aligning with the dynamic needs of the Liru Online Forum project.

Waterfall model



UNIVERSITY OF ABERDEEN

1.6 Project Advantages

Compared to existing platforms, this project is specifically tailored for campus communities, offering content publishing and management features that align better

with campus culture. First, the platform's strict user authentication and content review mechanisms effectively enhance the authenticity and security of information, ensuring a positive user experience. Second, the modular design and incremental development model ensure flexibility and scalability, enabling the platform to upgrade continuously based on user feedback and evolving needs. Additionally, the interaction design emphasizes user experience, providing rich interactive features that strengthen user engagement and ease of information sharing.

1.7 Cost Overview

The major costs of project development include the labor costs of the development team, server rental and maintenance fees, as well as resources needed for ongoing platform operations and content management. To reduce costs, the project makes full use of existing open-source technologies and frameworks, such as Django and MySQL, which not only accelerate development but also ensure the platform's stability and security. During the operational phase, the team will incrementally increase server resources in response to user growth to maintain high efficiency as the user base expands.

1.8 Profit Outlook

The long-term development plan of the project includes achieving profitability through multiple channels. As the campus blog platform continues to grow and attract more users, revenue can be generated through advertisements, paid content, and advanced user privileges. The platform will also explore partnerships with campus and external institutions to provide customized services for additional revenue. It is expected that within six months after the project is completed, the platform will experience significant user growth, reaching profitability within a year.

Chapter 2 Functional Requirements

2.1 User Stories

In this project, user stories help describe how different system modules meet user needs. Below are some major user scenarios:

User Registration and Login: Users can create an account through the registration feature by providing a username, password, and email address. After successful registration, users can use the login feature to access other platform functions. Users can choose to log out when needed. To enhance security, the system will verify the user's email address and require a sufficiently complex password to ensure account safety.

Publish Blog: Logged-in users can publish blog posts through the blog module. Users can enter the title and content and submit it for publication. Additionally, users can add tags to their blog posts to make it easier for others to search for related content. Once published, the blog post will be displayed on public pages, allowing other users to browse and interact with it.

Post Comments: Users can post comments after reading other users' blog posts, and the comments will be displayed below the articles. The comment system supports nested replies, allowing users to reply directly to other comments, forming discussion threads and enhancing user interaction and community atmosphere.

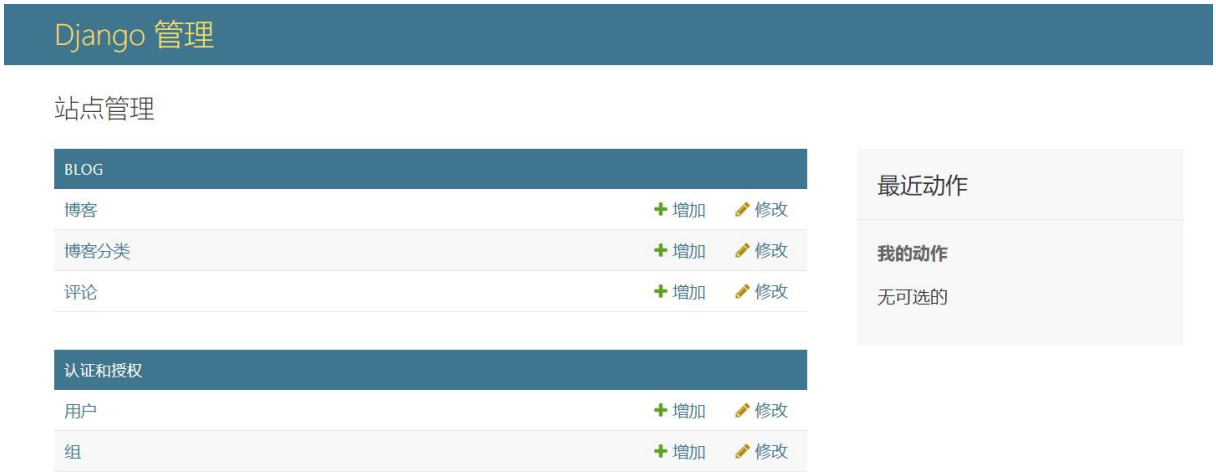
2.2 System Features

System features cover the main functional requirements of the system, ensuring that users can easily use the services provided by the platform:

User Management: Users can register, log in, and log out, ensuring that only verified users can access the core features of the platform. During registration, the system sends a verification email to confirm the user's identity, and login provides secure password encryption and verification to ensure data security.

Blog Publishing and Management: Users can publish, edit, and manage their blog posts, ensuring timely publication of information and the ability to modify content. Users can also add categories and tags to their posts for better content organization and easier searching by other users. Additionally, the system provides a draft-saving feature, allowing users to save drafts while writing and continue editing later.

Comment Feature: Users can comment on blog posts, and the comments are displayed in real-time under the corresponding blog post to facilitate user interaction. The comment feature supports multi-level nesting, allowing users to reply to other users' comments, forming complete discussion threads. Additionally, the system provides a reporting feature, enabling users to report inappropriate comments to maintain a healthy and harmonious community environment.



Chapter 3 Non-Functional Requirements

3.1 Performance Requirements

The system should have a high response speed and efficient resource utilization to ensure stable and reliable operation, even under high user access. The system should optimize memory and processor resource usage to improve response time, particularly during user registration, blog publishing, or commenting, ensuring smooth operation.

3.2 Security

The platform must have high data security and privacy protection mechanisms to safeguard users' personal information and blog content. The system should use strict user authentication and permission management to ensure that only authorized users can access sensitive data. In addition, the platform will be equipped with firewalls, encryption mechanisms, and anti-fraud measures to protect the system from malicious attacks and cybersecurity threats.

3.3 Compatibility

The system should have high compatibility and be able to run normally on different operating systems (e.g., Windows, macOS, Linux) and browsers (e.g., Chrome, Firefox, Safari), providing a consistent user experience for various usage scenarios.

3.4 Reliability

The system should have data backup and recovery mechanisms to ensure quick recovery and continued normal operation in case of system failures or data loss. The system should also maintain stability under high load and emergency situations, ensuring reliable platform operation.

3.5 Maintainability

The platform adopts a modular design, making it easier for developers to maintain and extend the system in the future. The code should follow standardized conventions to ensure readability and maintainability. Additionally, the system should support logging to quickly locate and fix problems when failures occur.

3.6 Usability

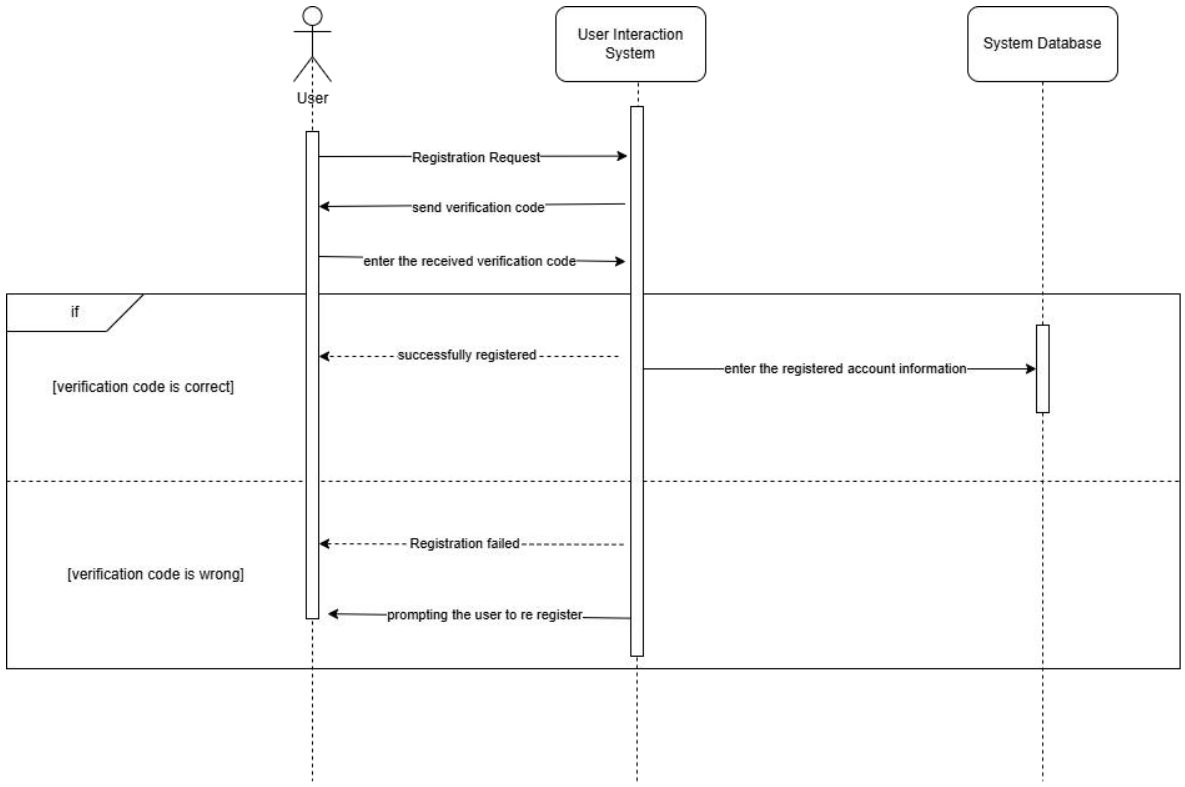
The system should provide a user-friendly interface and detailed help documentation to reduce the learning curve and help users quickly become familiar with the platform's features. The interface design should align with user habits to ensure ease of operation.

3.7 Legal Compliance

The system must comply with relevant laws, regulations, and industry standards, such as data protection regulations, to ensure its legality and compliance.

Chapter 4 Sequence Diagram

4.1 Sequence Diagram



4.2 Sequence Diagram Description

This section describes the sequence diagram of the project, illustrating the interactions between different actors and objects within the system. The sequence diagram helps in visualizing the sequence of processes involved in the operations of the campus blog platform. Below are the key use cases represented in the sequence diagram:

User Registration: When a user attempts to register, the system verifies the provided information, stores the data in the user database, and confirms registration.

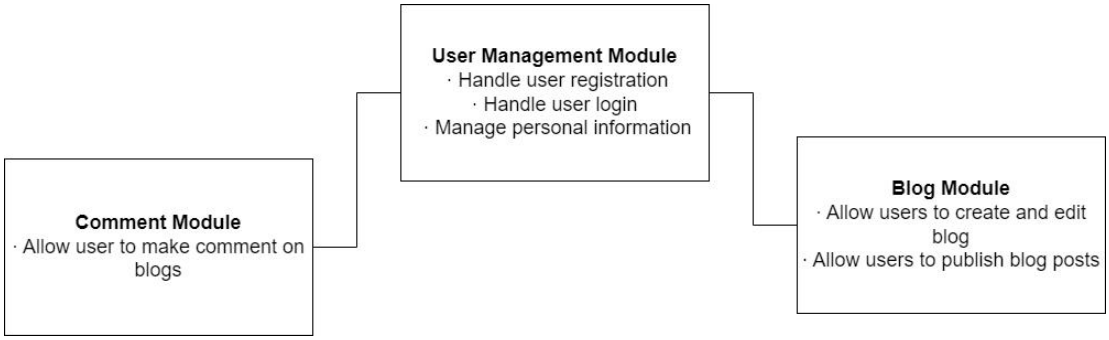
Blog Publishing: Once logged in, a user can publish a blog post. The sequence diagram shows the interaction between the user, the blog module, and the backend for processing and storing the post.

Comment Interaction: Users can comment on published blog posts. The sequence diagram details the flow of data between the user, the comment module, and the database to display the comments.

User Authentication: The sequence diagram also represents the authentication process where a user logs in by providing credentials that are verified against the stored data.

Chapter 5 Context Model and Business Process Model

5.1 Context Model



5.2 Context Model Description

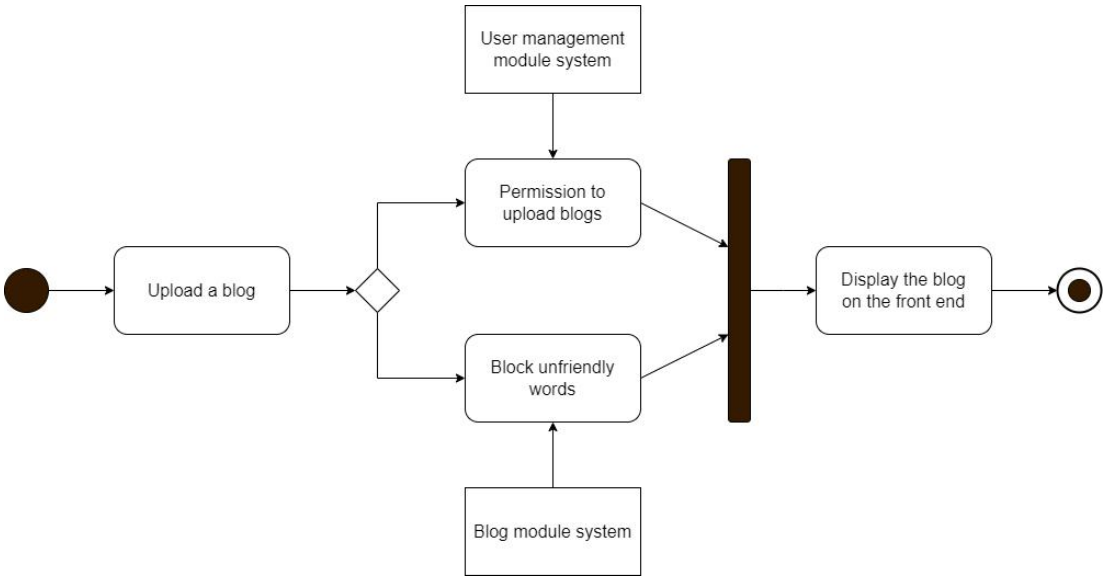
The context model illustrates the relationships and interactions between different modules of the system. In this project, the user management module, blog module, and comment module are the primary components, and their interactions are described as follows:

User Management Module: Handles user registration, login, and management of personal information. This module interacts with the blog and comment modules, ensuring smooth data flow between user profiles and published content.

Blog Module: Allows users to create, edit, and publish blog posts. Data from the blog module is linked to user profiles, ensuring a cohesive user experience across the platform.

Comment Module: Users can comment on blog posts, and these comments are managed through the comment module, which also stores data related to each user's interactions.

5.3 Business Process Model



5.4 Business Process Model Description

The business process model represents the core processes involved in the system's operations. In this project, the key business processes include user registration, blog publishing, and commenting:

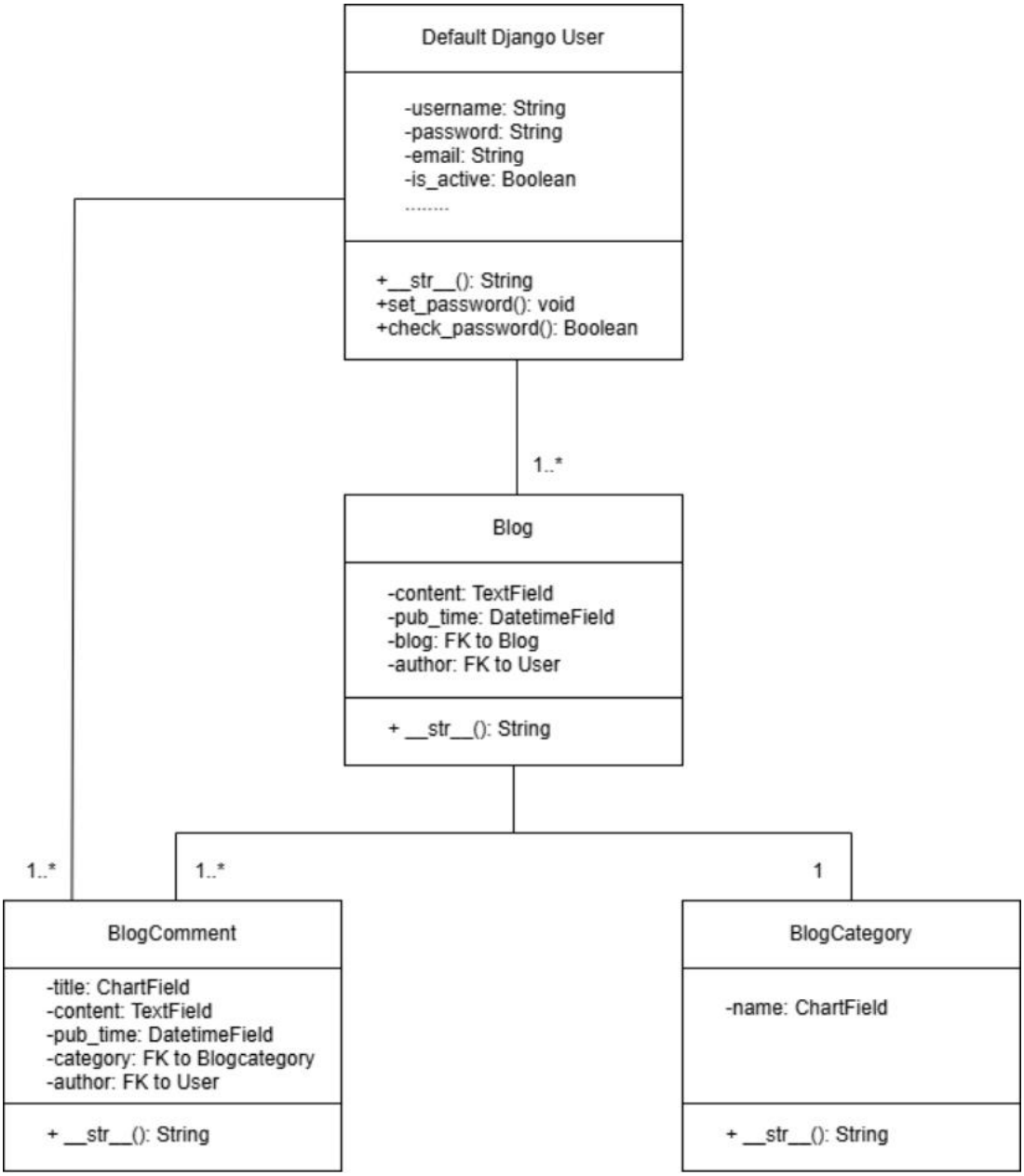
User Registration Process: A user provides the necessary information for registration. The system verifies the provided details and stores them in the user database. Upon successful registration, the user can access the platform.

Blog Publishing Process: A logged-in user creates a blog post by entering the content and any associated media. The system validates the content and stores it in the blog database. The blog post is displayed on the user's profile and accessible to others.

Commenting Process: Users can comment on existing blog posts.The comment is linked to the relevant blog and user, and displayed in real-time.

Chapter 6 Class Diagram

6.1 Class Diagram



6.2 Class Diagram Description

The class diagram shows the different classes in the system and their relationships, representing the object structure within the system. In this project, the primary classes include Default Django User, Blog, BlogComment, and BlogCategory, ensuring the clarity and maintainability of the system logic:

Default Django User: This class represents the default user model provided by Django, which includes fields like username, password, email, and is_active. It provides methods like set_password() to securely set a user's password and

check_password() to verify the password.

Blog: Each blog post is created by a user and contains fields such as content, pub_time, and references to its author. The Blog class is linked to the Default Django User class (one-to-many relationship).

BlogComment: Each comment is associated with a specific blog post. Users can comment on blogs, and each blog can have multiple comments (one-to-many relationship). This class includes fields like title, content, pub_time, and references to the blog and author.

BlogCategory: This class represents different categories that a blog post can belong to, making it easier for users to filter and search blog posts. It includes a field name and links to multiple blogs (one-to-many relationship).

Illustration Suggestion:

A class diagram should be inserted here to show the relationships between the Default Django User, Blog, BlogComment, and BlogCategory classes. This diagram will help readers better understand the interactions between the objects and classes in the system.

6.3 Class Relationship Analysis

Default Django User and Blog Relationship: Each user can publish multiple blog posts, but each blog post belongs to only one user.

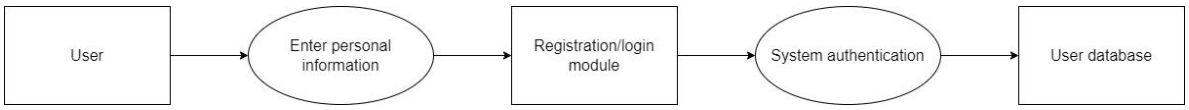
Blog and BlogComment Relationship: A blog can have multiple comments, but each comment is associated with only one blog.

Blog and BlogCategory Relationship: Each blog belongs to a specific category, and each category can contain multiple blogs.

Chapter 7 Data Flow Diagram

7.1 Data Flow Diagram Description

The Data Flow Diagram (DFD) is used to describe the flow and processing of data within the system. In this project, the primary data flows involve user management, blog publishing, and comment modules, ensuring smooth and secure data transmission throughout the system.



User Data Flow: Users enter personal information through the registration and login modules. The system verifies user identity and saves the data to the user database. After successful login, users can publish blogs and post comments.

Blog Data Flow: When users publish a blog, the system stores the blog content in the database through the blog module and displays it on the front end. The published blog is visible on both public and personal pages, allowing other users to view and comment on it.

Comment Data Flow: When a user posts a comment on a blog, the comment is associated with the corresponding blog. The system stores the comment data in the comment database and updates the article page in real-time to display the latest comments.

Chapter 8 Use Case Diagram and Description of Use Cases

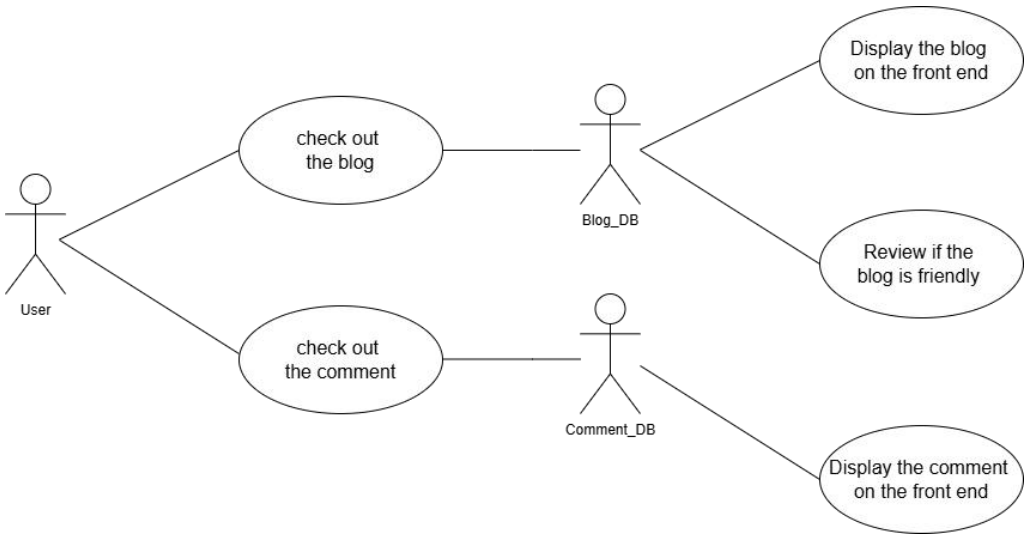
8.1 Use Case Diagram of User Management Module System

8.2 Description

Use Case Name	User Management Module System
Description	The user management module includes user registration, login, and logout functionalities.
Data	Blog content, publication time, user comments,

Trigger Events	1.Users enter a username, password, and email address on the registration page and submit it. 2.The system verifies the user's input and stores it in the user database. 3.Users enter their username and password on the login page, and upon successful verification, the system grants access to platform features. 4.Users can choose to log out, and the system updates their status to logged out.
Actors	User, User Database
Interaction	1.User: Register, log in, and log out. 2.User Database: Store and verify user information.

8.3 Use Case Diagram of Blog Module System



8.4 Description

Use Case Name	Blog Module System
Description	The blog module is one of the core functional modules of this project, where users can publish blog articles and interact with other users.
Data	Blog content, publication time, user comments,
Trigger Events	1.Users create a new article through the blog publishing feature by entering the title and content and submitting it. 2.The system receives the user's submission, verifies the data, and stores it in the blog database. 3.Once published successfully, the system updates the blog list and displays the new article on both public and personal pages. 4.Other users can comment on the blog post, and the comment content is displayed in real-time below the article.
Actors	User, Blog Database, Comment Database
Interaction	1.User: Publish and view blog articles, interact through comments. 2.Blog Database: Store and retrieve blog content. 3.Comment Database: Store user comments and display them in real-time on the page.