



UE21CS352B - Object Oriented Analysis & Design using Java

Mini Project Report

Online Blogging platform Management System

Submitted by:

**T H Manoj
Sunil k
Srinivasa k
Sundeeep**

**PES2UG22CS612
PES2UG22CS599
PES2UG22CS577
PES2UG22CS598**

6th Semester F Section

Prof. Ruby Dinakar

January - May 2024

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY**

**(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, Indi**

Problem Statement : Online Blogging Platform Management System

The **Online Blogging Platform Management System** is a comprehensive solution designed to streamline content creation and enhance reader engagement. The platform enables users to register, create, edit, and publish blog posts seamlessly, while providing administrators with tools to manage posts, users, categories, and platform settings efficiently. Specifically, administrators will have control over content moderation, category management, user roles, and system analytics.

Key Features:

1. User-Friendly Interface

- Developed an intuitive and responsive web interface for users to compose, edit, and publish posts easily with support for rich text formatting and media embedding.
- Includes a clean and organized dashboard for administrators to oversee blogs, manage categories, and moderate content effectively.

2. Post and Category Management

- Enables users to create new blog posts, update content, and delete posts.
- Administrators can create, edit, or remove categories and tags to organize blog posts and improve discoverability.
- Implements search and filter functionalities to allow readers to find posts based on categories, tags, authors, or keywords.

3. Commenting and Moderation

- Allows readers to comment on blog posts, promoting interaction and feedback.
- Enables administrators to moderate comments, remove inappropriate content, and ban spammers.
- Supports nested or threaded comments for better conversational flow.

4. Admin Content and User Management

- Empowers administrators to view, approve, or reject blog submissions (if required).

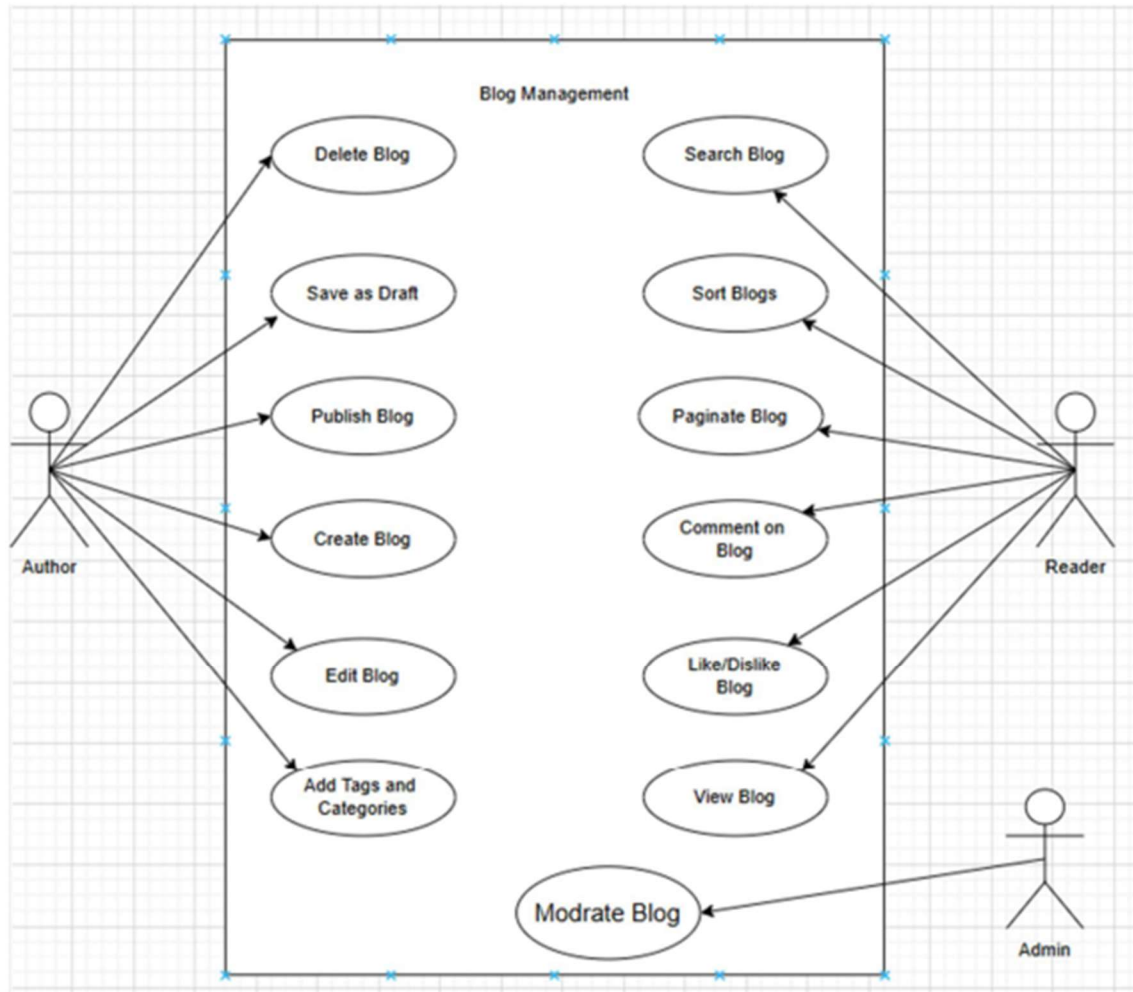
- Supports role-based access control to assign different permissions (e.g., Admin, Editor, Contributor).
- Administrators can manage user accounts, reset passwords, and oversee user-generated content.

5. User Authentication and Security

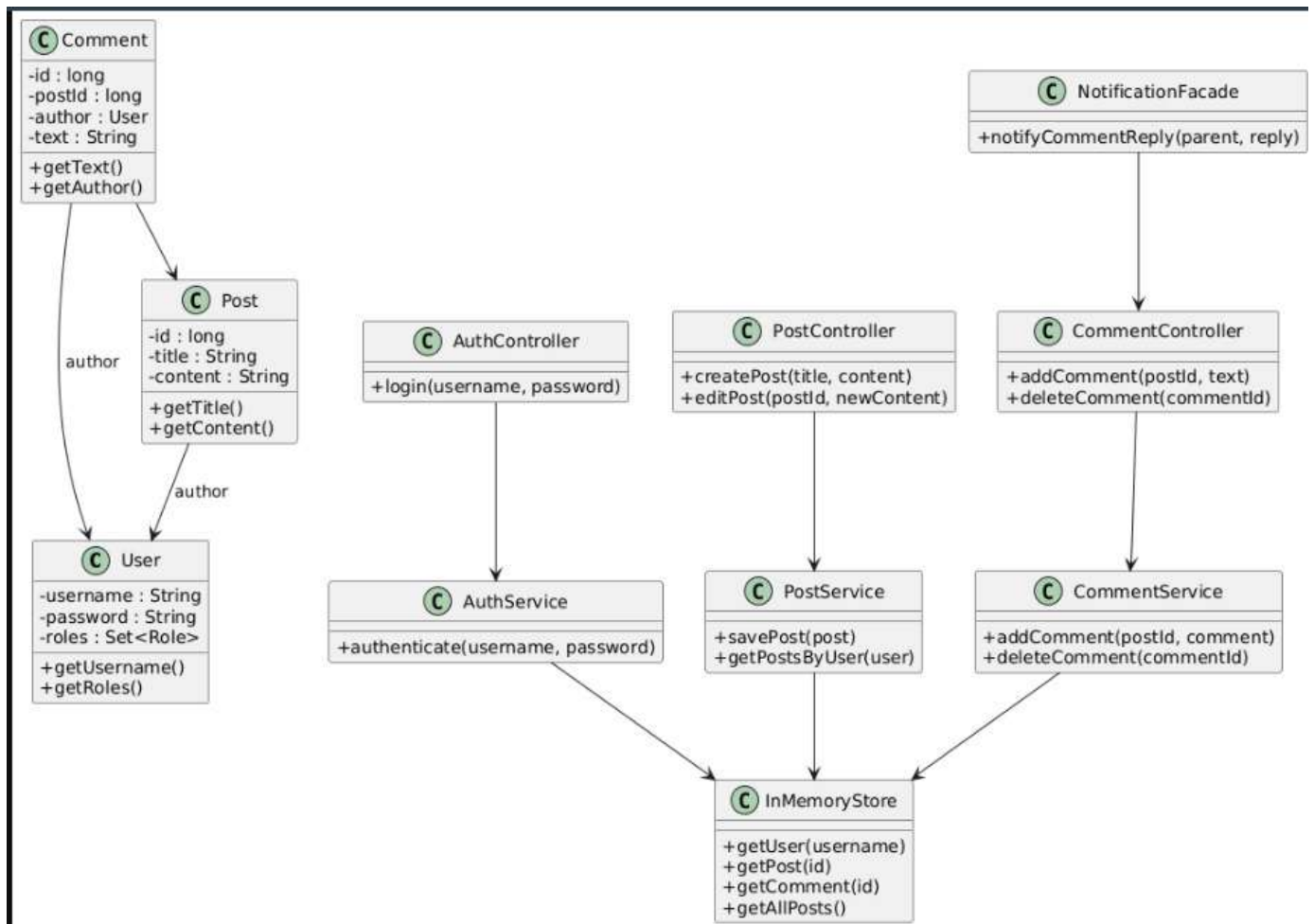
- Implements secure user registration and login functionality with password encryption and session handling.
- Provides distinct login portals for regular users and administrators, ensuring restricted access to sensitive admin functionalities.
- Allows users to manage profiles, change passwords, and track their published and draft posts

Models:

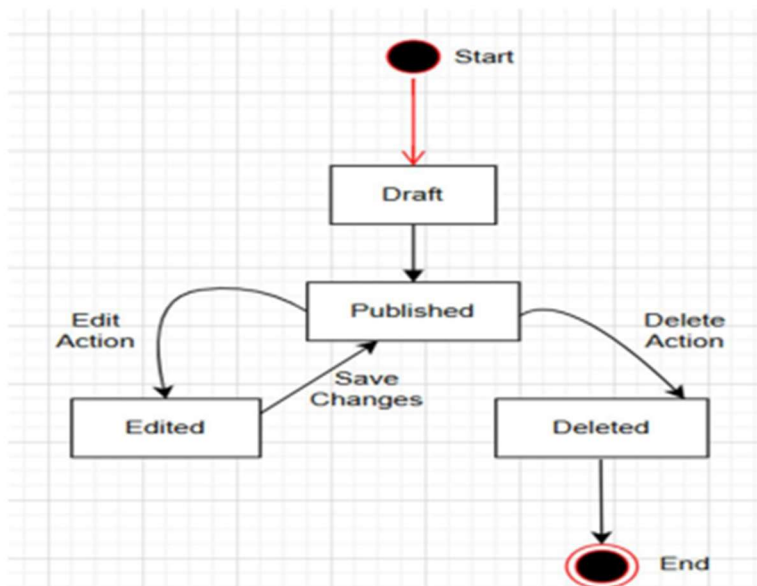
Use Case Diagram:



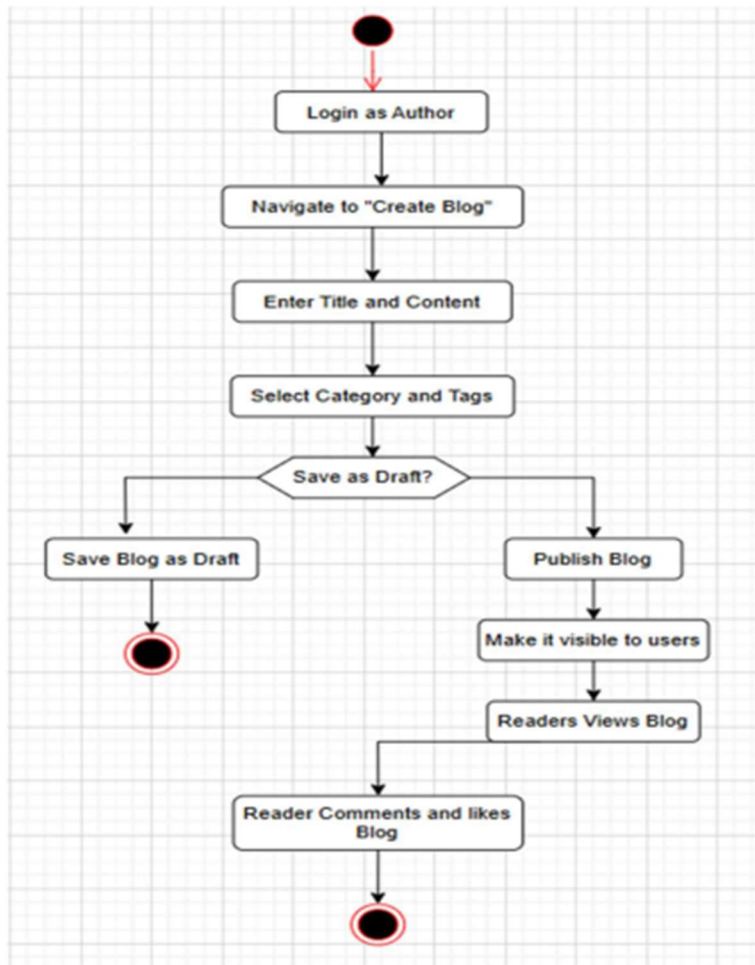
Class Diagram:



State Diagram:



Activity Diagrams:



Architecture Patterns, Design Principles, and Design Patterns:

Architecture Patterns

1. Adapter Pattern:

Implementation: PasswordHasherAdapter class in util package

Description: This pattern converts the interface of a third-party hashing library (Java's MessageDigest) into the compatible interface that this application expects (Hasher interface). It allows the system to work with incompatible interfaces by creating a bridge between them.

2.Facade Pattern:

Implementation: NotificationFacade class in util package

Description: This pattern provides a unified interface to a set of interfaces in the notification subsystem. It defines a higher-level interface that makes the subsystem easier to use by hiding the complexities of notification logic (logging, displaying dialogs, potentially sending emails) behind a simple interface.

3. Proxy Pattern:

Implementation: CommentServiceProxy class in service package

Description: This pattern controls access to the actual CommentService implementation. The proxy acts as a surrogate, adding security checks before allowing operations on comments. It ensures that only users with appropriate permissions can perform certain operations, like editing their own comments or (for admins) deleting any comment.

4. Flyweight Pattern:

Implementation: In InMemoryStore class for managing Tag and Category instances

Description: This pattern shares common Tag and Category objects across the application instead of creating new instances every time. It minimizes memory usage by reusing existing objects with the same underlying data (tag/category name), which is particularly useful for a blog system where the same tags may be used across multiple posts.

Components of MVC

1. Model

- The **Model** represents the application's data and core business logic. It encapsulates operations related to data access, manipulation, validation, and persistence, independently of how the data is presented to users.
- In the **Blog Application Platform**, the Model includes entities such as **User**, **Post**, **Comment**, **Category**, and **Tag**, along with corresponding services and repositories that handle operations like creating, updating, deleting, and retrieving data from the database.

2. View

- The **View** constitutes the presentation layer of the application. It is responsible for displaying data to users and capturing user input. It formats and renders the data provided by the Model in a user-friendly way.
- In our system, the View comprises the **user interface components**, such as web pages for post creation, post listings, user login/registration, comment sections, and admin dashboards. These interfaces can be built using technologies like **JSP**, **Thymeleaf**, or frontend frameworks (if applicable), ensuring an interactive and responsive experience for both regular users and administrators

3. Controller

- The **Controller** acts as a mediator between the Model and the View. It handles incoming requests from users, processes them, interacts with the Model to retrieve or update data, and then determines the appropriate View to render.
- In our Blog Application Platform, the Controller contains logic to handle actions such as **user authentication**, **post creation and editing**, **comment submission**, **category management**, and **administrative approvals**. It routes user actions to the correct services and ensures the application responds appropriately to different user interactions.

Design principles:

1. Single Responsibility Principle (SRP)

- Each class has one primary responsibility
- AuthService handles only authentication concerns
- PostController handles only post-related user actions
- LoginView handles only login UI elements and interactions

2. Open/Closed Principle (OCP)

- The CommentService interface is open for extension but closed for modification
- New implementations like CommentServiceImpl extend functionality without changing existing code

3. Liskov Substitution Principle (LSP)

- The CommentServiceProxy can be used anywhere a CommentService is expected
- It maintains the same contract while adding authorization checks

4. Interface Segregation Principle (ISP)

- Small, focused interfaces like Hasher with single purposes
- No large, catch-all interfaces forcing unneeded method implementations

5. Dependency Inversion Principle (DIP)

- Controllers depend on service interfaces, not concrete implementations
- Dependencies are injected through constructors
- Facilitates unit testing and component replacement

6. Separation of Concerns

- Authentication logic is separate from post management
- UI rendering is separate from business rules
- Data persistence is separate from data presentation

MVC LAYERS:

1. Model Layer

- Contains domain objects (User, Post, Comment, etc.)
- Pure data representations with minimal business logic
- Implements Serializable to support persistence

2. View Layer

- Swing-based UI components (LoginView, MainView, etc.)
- Responsible only for displaying data and capturing user inputs
- No direct data manipulation or business logic

3. Controller Layer

- Mediates between Model and View (AuthController, PostController, etc.)
- Handles user actions from the View
- Invokes appropriate Service methods
- Updates the View based on results

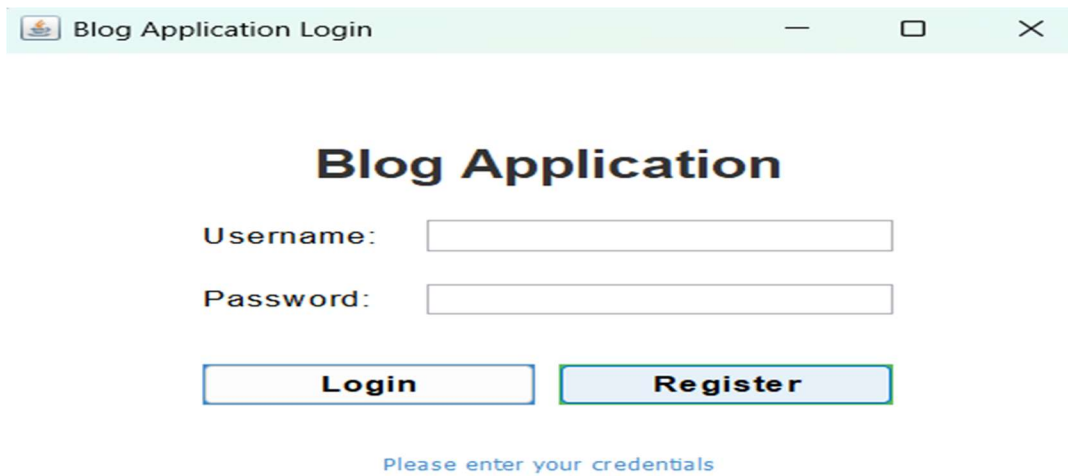
4. Service Layer

- Additional layer between Controllers and data
- Contains business logic (AuthService, PostService, etc.)

- Implements validation and domain rules

ScreenShots :-

UI :



The screenshot shows a web browser window with the title 'Blog Application Login'. The page content includes a heading 'Blog Application' in bold. Below the heading are two input fields: 'Username:' and 'Password:'. Underneath these fields are two buttons: 'Login' and 'Register'. At the bottom of the form, there is a message: 'Please enter your credentials'.

Blog Application Login

Blog Application

Username:

Password:

Please enter your credentials

Register New User

Username:

Password:

Confirm Password:

Please enter your credentials

Users Posts Comments

Username	Roles
pratik	[READER]
author	[AUTHOR]
admin	[ADMIN]

Add User

Edit User

Delete User

Logout

Blog Application - admin

Users

Posts

Comments

ID	Title	Author	Status	Created At
1	Welcome to the Blog	pratik	DRAFT	2025-04-22T10:10:04.03106...
2	Getting Started	pratik	DRAFT	2025-04-22T10:10:04.03948...
3	Forest	currentUser	DRAFT	2025-04-22T10:10:25.27600...

Add Post

Edit Post

Delete Post

Logout

Users

Posts

Comments

Comments

pratik: #saveforest

Add Comment

Submit Comment

Logout

Blog Application - admin

UsersPostsComments

ID	Title	Author	Status	Created At
1	Welcome to the Blog	pratik	DRAFT	2025-04-22T10:10:04.0310
2	Getting Started	pratik	DRAFT	2025-04-22T10:10:04.0394
3	Forest	currentUser	DRAFT	2025-04-22T10:10:25.2760

Add Post

Title:

Content:

Status:

DRAFT

SaveCancel

Add PostEdit PostDelete Post

Logout

Individual contributions of the team members:

T H Manoj – Core System & Integration

Sunil K – Comment System & Access Control

Shrinivas K – Post Management & Content System

Sundeeep M – User Management & Authentication