Multi-User Blog Platform

JAVA PROGRAMMING LAB

Submitted

In partial fulfillment of the requirements for completion of

Bachelor of Technology IVth Semester

in

Computer Science Engineering (Artificial Intelligence/Machine Learning)

by

KHYATI CHINTHA 23261A6620

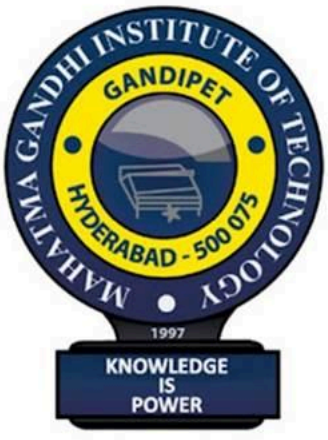
Under the guidance of

Ms. Deshmukh Deepika

ASSISTANT PROFESSOR

DEPARTMENT OF EMERGING TECHNOLOGIES

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY



Department of Emerging Technologies

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY, GANDIPET,

HYDERABAD-500075, INDIA

June 2025

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY
GANDIPET, HYDERABAD-500075, Telangana

CERTIFICATE

This is to certify that the software engineering entitled "**Multi-User Blog Platform**" is being submitted by

KHYATI CHINTHA 23261A6620

in partial fulfilment of the requirement for the **Laboratory Project** in **EMERGING TECHNOLOGIES** is a record of bona fide work carried out by them. The results of the investigations enclosed in this report have been verified and found satisfactory.

Ms. Deshmukh Deepika
Assistant Professor
ET Department
MGIT

INDEX

ABSTRACT

This project presents a comprehensive, user-friendly command-line blogging platform developed in Java, designed to enable multiple users to securely create, manage, and interact with blog content. The application incorporates essential blogging features such as user registration with unique usernames and emails, secure login with SHA-256 password hashing, post creation, editing, deletion, and threaded commenting.

The platform features an adaptive, menu-driven interface that changes based on the user's authentication status, ensuring intuitive navigation and minimizing user errors. Authenticated users can create new blog posts, view all posts or just their own, search for posts by keyword, and comment on any post. Strict access control ensures that only the original author of a post can edit or delete it, preserving content integrity and ownership.

All posts and comments are time stamped for clarity and chronological tracking. The system provides real-time feedback and robust error handling to guide users through all operations. The codebase is structured using clear object-oriented principles, with dedicated classes for users, posts, comments, and the core blog system logic. This modular design not only enhances readability and maintainability but also allows for easy future expansion, such as adding persistent storage or advanced user roles.

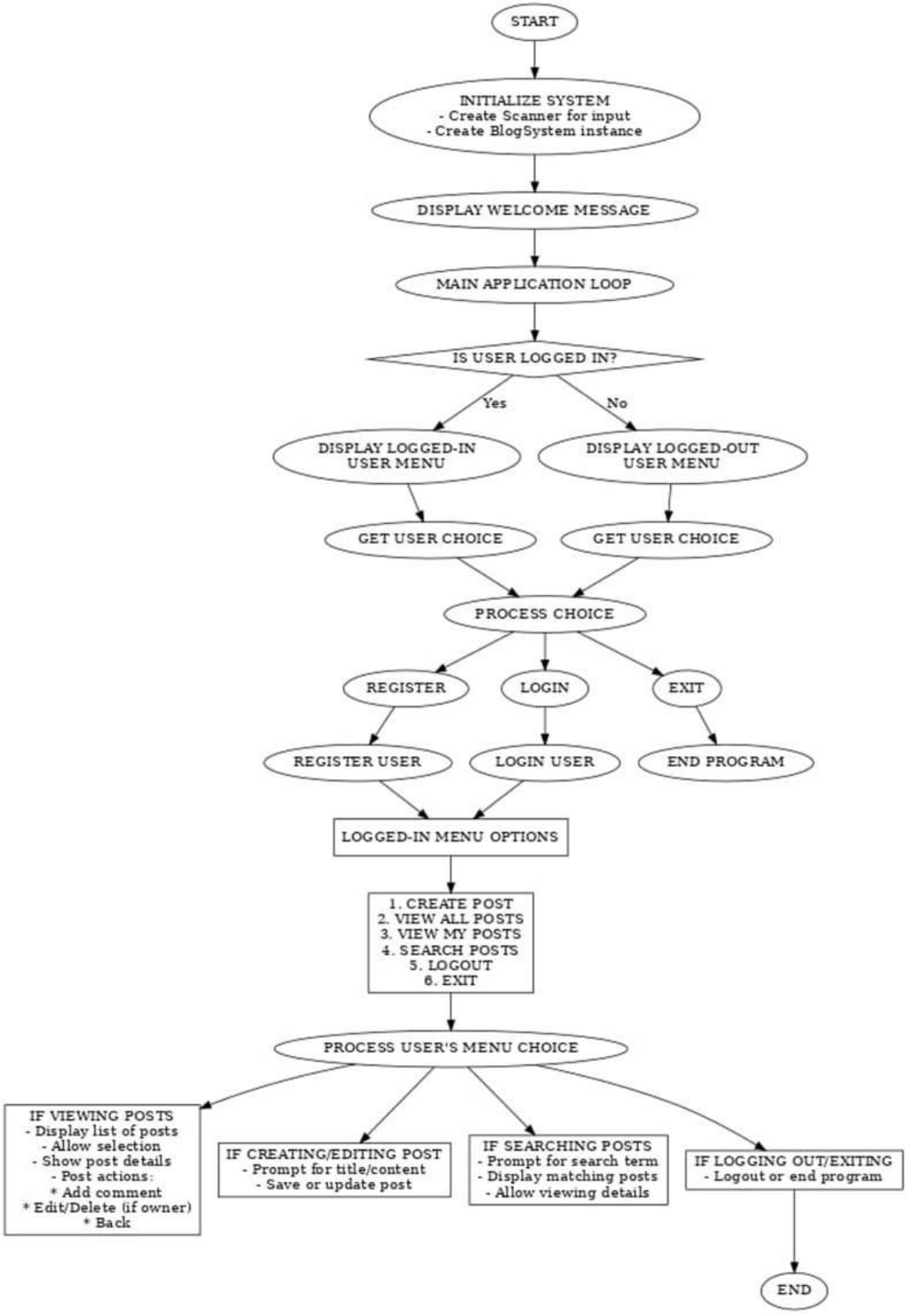
Overall, this platform serves as a practical and educational example of building a secure, interactive, and multi-user application in Java. It demonstrates best practices in authentication, input validation, error handling, and modular programming, making it ideal for both learning and as a foundation for more advanced blogging solutions.

ALGORITHM

1. **Start**
2. **Initialize System**
    - Create a Scanner for user input.
    - Instantiate the BlogSystem to manage users and posts.
3. **Main Application Loop**
    - Display a welcome message.
    - Set an exit flag to false.
    - While exit is false:
        - Display the appropriate menu based on whether a user is logged in.
        - Read the user's menu choice.
4. **Handle Menu Choices**
    - **If no user is logged in:**
        - If choice is "1": Call registerUser() to register a new user.
        - If choice is "2": Call loginUser() to log in an existing user.
        - If choice is "3": Set exit to true and print a goodbye message.
        - Otherwise: Print an invalid option message.
    - **If a user is logged in:**
        - If choice is "1": Call createPost() to create a new post.
        - If choice is "2": Call viewAllPosts() to display all posts.
        - If choice is "3": Call viewMyPosts() to display the current user's posts.
        - If choice is "4": Call searchPosts() to search for posts.
        - If choice is "5": Call blogSystem.logout() to log out the current user.
        - If choice is "6": Set exit to true and print a goodbye message.
        - Otherwise: Print an invalid option message.
5. **User Registration (registerUser())**
    - Prompt for username, password, and email.
    - Attempt to register the user in BlogSystem.
    - On success, inform the user; on failure, display the error.
6. **User Login (loginUser())**
    - Prompt for username and password.
    - Attempt to log in via BlogSystem.
    - Inform the user if login fails.
7. **Post Creation (createPost())**
    - Prompt for post title.
    - Prompt for multi-line post content, ending with "END".
    - Attempt to create the post in BlogSystem.
    - On success, inform the user; on failure, display the error.
8. **Viewing Posts**
    - **All Posts (viewAllPosts()):**
        - Retrieve and display all posts.
        - If none exist, inform the user.
        - Allow the user to select a post to view details or return.
    - **User's Posts (viewMyPosts()):**
        - Retrieve and display posts by the current user.

        - If none exist, inform the user.
        - Allow the user to select a post to view details or return.
9. **Search Posts (searchPosts())**
    - Prompt for a search term.
    - Retrieve and display matching posts.
    - If none is found, inform the user.
    - Allow the user to select a post to view details or return.
10. **Post Details and Actions (displayPostDetails())**
    - Display the post's title, author, timestamp, and content.
    - Display all comments on the post.
    - Present options:
        - Add a comment.
        - If the current user is the author: Edit or delete the post.
        - Go back.
    - Handle the user's choice:
        - **Add Comment:** Prompt for comment text and add it to the post.
        - **Edit Post:** If author, prompt for new title and/or content, then update the post.
        - **Delete Post:** If author, confirm deletion and remove the post if confirmed.
        - **Back:** Return to the previous menu.
11. **Logout**
    - Clear the current user session.
12. **Exit**
    - Close the Scanner.
    - End the program.
13. **End**

FLOWCHART


START
INITIALIZE SYSTEM
- Create Scanner for input
- Create BlogSystem instance
DISPLAY WELCOME MESSAGE
MAIN APPLICATION LOOP
IS USER LOGGED IN?
Yes
No
DISPLAY LOGGED-IN USER MENU
DISPLAY LOGGED-OUT USER MENU
GET USER CHOICE
GET USER CHOICE
PROCESS CHOICE
REGISTER
LOGIN
EXIT
REGISTER USER
LOGIN USER
END PROGRAM
LOGGED-IN MENU OPTIONS
1. CREATE POST
2. VIEW ALL POSTS
3. VIEW MY POSTS
4. SEARCH POSTS
5. LOGOUT
6. EXIT
PROCESS USER'S MENU CHOICE
IF VIEWING POSTS
- Display list of posts
- Allow selection
- Show post details
- Post actions:
* Add comment
* Edit/Delete (if owner)
* Back
IF CREATING/EDITING POST
- Prompt for title/content
- Save or update post
IF SEARCHING POSTS
- Prompt for search term
- Display matching posts
- Allow viewing details
IF LOGGING OUT/EXITING
- Logout or end program
END

CODE

```
import java.util.*;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.nio.charset.StandardCharsets;

/**
 * OptimizedBlogPlatform: Main class that serves as the entry point for the blog application
 * Combines the best features from both original implementations in a streamlined way
 */
public class OptimizedBlogPlatform {
    private static final Scanner scanner = new Scanner(System.in); // Scanner for user input
    private static final BlogSystem blogSystem = new BlogSystem(); // Core blog system

    public static void main(String[] args) {
        System.out.println("Welcome to the Optimized Blog Platform!"); // Welcome message
        boolean exit = false; // Flag to control main program loop

        // Main application loop
        while (!exit) {
            displayMenu(); // Show appropriate menu based on login status
            String choice = scanner.nextLine().trim(); // Get user's menu choice

            try {
                // Process user input based on login status
                if (blogSystem.getCurrentUser() == null) {
                    // Menu options for logged-out users
                    switch (choice) {
                        case "1": registerUser(); break; // Register new user
                        case "2": loginUser(); break; // Login existing user
                        case "3": // Exit application
                            exit = true;
                            System.out.println("Thank you for using Blog Platform. Goodbye!");
                            break;
                        default:
                            System.out.println("Invalid option. Please try again.");
                    }
                } else {
                    // Menu options for logged-in users
                    switch (choice) {
                        case "1": createPost(); break; // Create new post
                        case "2": viewAllPosts(); break; // View all posts
                        case "3": viewMyPosts(); break; // View own posts
                        case "4": searchPosts(); break; // Search for posts
                        case "5": blogSystem.logout(); break; // Logout current user
                        case "6": // Exit application
                            exit = true;
                            System.out.println("Thank you for using Blog Platform. Goodbye!");
```

```
                    break;
                default:
                    System.out.println("Invalid option. Please try again.");
            }
        }
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage()); // Display any errors
    }
  }
  scanner.close(); // Close scanner to prevent resource leak
}

/**
 * Display the appropriate menu based on login status
 */
private static void displayMenu() {
  if (blogSystem.getCurrentUser() == null) {
    // Display menu for logged-out users
    System.out.println("\n===== MENU =====");
    System.out.println("1. Register");
    System.out.println("2. Login");
    System.out.println("3. Exit");
  } else {
    // Display menu for logged-in users
    User currentUser = blogSystem.getCurrentUser();
    System.out.println("\n===== MENU =====");
    System.out.println("Logged in as: " + currentUser.getUsername());
    System.out.println("1. Create Post");
    System.out.println("2. View All Posts");
    System.out.println("3. View My Posts");
    System.out.println("4. Search Posts");
    System.out.println("5. Logout");
    System.out.println("6. Exit");
  }
  System.out.print("Choose an option: ");
}

/**
 * Handle user registration process
 */
private static void registerUser() {
  System.out.print("Enter username: "); // Prompt for username
  String username = scanner.nextLine().trim(); // Get username input
  System.out.print("Enter password: "); // Prompt for password
  String password = scanner.nextLine().trim(); // Get password input
  System.out.print("Enter email: "); // Prompt for email
  String email = scanner.nextLine().trim(); // Get email input

  try {
    blogSystem.registerUser(username, password, email); // Register with provided info
    System.out.println("Registration successful! Please login.");
```

```
    } catch (IllegalArgumentException e) {
        System.out.println("Registration failed: " + e.getMessage()); // Display validation errors
    }
}

/**
 * Handle user login process
 */
private static void loginUser() {
    System.out.print("Enter username: "); // Prompt for username
    String username = scanner.nextLine().trim(); // Get username input
    System.out.print("Enter password: "); // Prompt for password
    String password = scanner.nextLine().trim(); // Get password input

    boolean success = blogSystem.login(username, password); // Attempt login
    if (!success) {
        System.out.println("Invalid username or password."); // Inform about failed login
    }
}

/**
 * Handle post creation process
 */
private static void createPost() {
    System.out.print("Enter post title: "); // Prompt for title
    String title = scanner.nextLine().trim(); // Get title input
    System.out.println("Enter post content (type END on a new line to finish):"); // Prompt for content

    // Collect multi-line content until END marker
    StringBuilder content = new StringBuilder();
    String line;
    while (!(line = scanner.nextLine()).equals("END")) {
        content.append(line).append("\n");
    }

    try {
        blogSystem.createPost(title, content.toString()); // Create post with provided info
        System.out.println("Post created successfully!");
    } catch (IllegalArgumentException e) {
        System.out.println("Failed to create post: " + e.getMessage()); // Display validation errors
    }
}

/**
 * Display all posts in the system
 */
private static void viewAllPosts() {
    List<Post> posts = blogSystem.getAllPosts(); // Get all posts

    if (posts.isEmpty()) { // Check if there are posts
        System.out.println("No posts available.");
```

```
            return;
        }

        displayPosts(posts); // Display the posts
    }

    /**
     * Display posts by the current user
     */
    private static void viewMyPosts() {
        User currentUser = blogSystem.getCurrentUser(); // Get current user
        List<Post> posts = blogSystem.getUserPosts(currentUser.getUsername()); // Get user's posts

        if (posts.isEmpty()) { // Check if user has posts
            System.out.println("You haven't created any posts yet.");
            return;
        }

        displayPosts(posts); // Display the posts
    }

    /**
     * Search posts by keyword
     */
    private static void searchPosts() {
        System.out.print("Enter search term: "); // Prompt for search term
        String searchTerm = scanner.nextLine().trim(); // Get search input

        List<Post> posts = blogSystem.searchPosts(searchTerm); // Search for posts

        if (posts.isEmpty()) { // Check if any posts match
            System.out.println("No posts found matching your search term.");
            return;
        }

        displayPosts(posts); // Display matching posts
    }

    /**
     * Display a list of posts and handle post selection
     * @param posts List of posts to display
     */
    private static void displayPosts(List<Post> posts) {
        // Show list of posts
        for (int i = 0; i < posts.size(); i++) {
            Post post = posts.get(i);
            System.out.println("\n" + (i + 1) + ". " + post.getTitle() +
                               " by " + post.getAuthor() +
                               " on " + post.getFormattedDate());
        }
```

```
    // Handle post selection
    System.out.print("\nEnter post number to view details (or 0 to go back): ");
    try {
        int choice = Integer.parseInt(scanner.nextLine().trim());
        if (choice > 0 && choice <= posts.size()) {
            displayPostDetails(posts.get(choice - 1)); // Show selected post
        }
    } catch (NumberFormatException e) {
        System.out.println("Invalid selection.");
    }
}

/**
 * Display details of a specific post and handle post interactions
 * @param post The post to display
 */
private static void displayPostDetails(Post post) {
    boolean back = false;

    while (!back) {
        // Display post header
        System.out.println("\n----- " + post.getTitle() + " -----");
        System.out.println("By: " + post.getAuthor() + " | " + post.getFormattedDate());
        System.out.println("---------------------------------------");
        System.out.println(post.getContent()); // Display content
        System.out.println("---------------------------------------");

        // Display comments
        System.out.println("\nComments:");
        List<Comment> comments = post.getComments();
        if (comments.isEmpty()) {
            System.out.println("No comments yet.");
        } else {
            for (Comment comment : comments) {
                System.out.println(comment.getAuthor() + " (" +
                                   comment.getFormattedDate() + "): " +
                                   comment.getContent());
            }
        }

        // Display available actions
        System.out.println("\n1. Add comment");
        boolean isAuthor = post.getAuthor().equals(blogSystem.getCurrentUser().getUsername());
        if (isAuthor) {
            System.out.println("2. Edit post");
            System.out.println("3. Delete post");
        }
        System.out.println("0. Back");

        System.out.print("Choose an option: ");
        String choice = scanner.nextLine().trim();
```

```
            // Process user choice
            switch (choice) {
                case "1": // Add comment
                    System.out.print("Enter your comment: ");
                    String commentText = scanner.nextLine().trim();
                    try {
                        post.addComment(blogSystem.getCurrentUser().getUsername(), commentText);
                        System.out.println("Comment added successfully!");
                    } catch (IllegalArgumentException e) {
                        System.out.println("Failed to add comment: " + e.getMessage());
                    }
                    break;
                case "2": // Edit post (author only)
                    if (isAuthor) {
                        editPost(post);
                    } else {
                        System.out.println("Invalid option.");
                    }
                    break;
                case "3": // Delete post (author only)
                    if (isAuthor) {
                        if (deletePost(post)) {
                            back = true; // Return to previous menu after deletion
                        }
                    } else {
                        System.out.println("Invalid option.");
                    }
                    break;
                case "0": // Go back
                    back = true;
                    break;
                default:
                    System.out.println("Invalid option. Please try again.");
            }
        }
    }

    /**
     * Handle post editing process
     * @param post The post to edit
     */
    private static void editPost(Post post) {
        System.out.print("Enter new title (or press Enter to keep current): ");
        String newTitle = scanner.nextLine().trim();
        if (!newTitle.isEmpty()) {
            post.setTitle(newTitle); // Update title if provided
        }

        System.out.println("Enter new content (type END on a new line to finish, or just END to keep
current):");
```

```
        StringBuilder newContent = new StringBuilder();
        String line;
        while (!(line = scanner.nextLine()).equals("END")) {
            newContent.append(line).append("\n");
        }

        if (newContent.length() > 0) {
            post.setContent(newContent.toString()); // Update content if provided
        }

        System.out.println("Post updated successfully!");
    }

    /**
     * Handle post deletion process
     * @param post The post to delete
     * @return true if post was deleted, false otherwise
     */
    private static boolean deletePost(Post post) {
        System.out.print("Are you sure you want to delete this post? (y/n): ");
        String confirm = scanner.nextLine().trim().toLowerCase();
        if (confirm.equals("y")) {
            try {
                blogSystem.deletePost(post); // Delete the post
                System.out.println("Post deleted successfully!");
                return true;
            } catch (IllegalStateException e) {
                System.out.println("Failed to delete post: " + e.getMessage());
            }
        }
        return false;
    }
}

/**
 * User class for authentication and user information management
 */
class User {
    private final String username; // Unique identifier for the user
    private final String password; // Hashed password for authentication
    private final String email; // User's email address

    /**
     * Constructor to initialize a User with required information
     * @param username Unique username
     * @param password Hashed password
     * @param email User's email
     */
    public User(String username, String password, String email) {
        this.username = username;
        this.password = password;
```

```
        this.email = email;
    }

    // Getters with concise implementation
    public String getUsername() { return username; }
    public String getPassword() { return password; }
    public String getEmail() { return email; }
}

/**
 * Post class for blog posts management
 */
class Post {
    private String title; // Title of the post
    private String content; // Content of the post
    private final String author; // Username of post author
    private final LocalDateTime createdAt; // Creation timestamp
    private final List<Comment> comments; // Comments on this post

    /**
     * Constructor to initialize a Post with required information
     * @param title Post title
     * @param content Post content
     * @param author Username of post creator
     */
    public Post(String title, String content, String author) {
        this.title = title;
        this.content = content;
        this.author = author;
        this.createdAt = LocalDateTime.now();
        this.comments = new ArrayList<>();
    }

    // Getters and setters
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public String getContent() { return content; }
    public void setContent(String content) { this.content = content; }
    public String getAuthor() { return author; }
    public LocalDateTime getCreatedAt() { return createdAt; }

    /**
     * Format the creation date for display
     * @return Formatted date string
     */
    public String getFormattedDate() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        return createdAt.format(formatter);
    }

    /**
```

```
     * Get all comments on this post
     * @return List of comments
     */
    public List<Comment> getComments() {
        return new ArrayList<>(comments); // Return copy to prevent external modification
    }

    /**
     * Add a new comment to this post
     * @param author Username of comment creator
     * @param content Comment text
     */
    public void addComment(String author, String content) {
        if (content == null || content.trim().isEmpty()) {
            throw new IllegalArgumentException("Comment content cannot be empty");
        }
        Comment comment = new Comment(author, content);
        comments.add(comment);
    }
}

/**
 * Comment class for post comments management
 */
class Comment {
    private final String author; // Username of comment creator
    private final String content; // Comment text
    private final LocalDateTime createdAt; // Creation timestamp

    /**
     * Constructor to initialize a Comment with required information
     * @param author Username of comment creator
     * @param content Comment text
     */
    public Comment(String author, String content) {
        this.author = author;
        this.content = content;
        this.createdAt = LocalDateTime.now();
    }

    // Getters with concise implementation
    public String getAuthor() { return author; }
    public String getContent() { return content; }

    /**
     * Format the creation date for display
     * @return Formatted date string
     */
    public String getFormattedDate() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        return createdAt.format(formatter);
```

```
    }
}

/**
 * BlogSystem class for managing the entire blog platform
 */
class BlogSystem {
    private final Map<String, User> users; // Map of username to User objects
    private final List<Post> posts; // List of all posts
    private User currentUser; // Currently logged in user

    /**
     * Constructor to initialize the blog system
     */
    public BlogSystem() {
        this.users = new HashMap<>();
        this.posts = new ArrayList<>();
        this.currentUser = null;
    }

    /**
     * Register a new user in the system with validation
     * @param username User's username
     * @param password User's password (will be hashed)
     * @param email User's email
     */
    public void registerUser(String username, String password, String email) {
        // Validate inputs
        if (username == null || username.trim().isEmpty()) {
            throw new IllegalArgumentException("Username cannot be empty");
        }
        if (password == null || password.length() < 6) {
            throw new IllegalArgumentException("Password must be at least 6 characters");
        }
        if (email == null || !email.contains("@")) {
            throw new IllegalArgumentException("Invalid email format");
        }

        // Check for duplicate username/email
        if (users.containsKey(username)) {
            throw new IllegalArgumentException("Username already exists");
        }

        for (User user : users.values()) {
            if (user.getEmail().equalsIgnoreCase(email)) {
                throw new IllegalArgumentException("Email already registered");
            }
        }

        // Create and store new user with hashed password
        User newUser = new User(username, hashPassword(password), email);
```

```
    users.put(username, newUser);
}

/**
 * Authenticate and login a user
 * @param username User's username
 * @param password User's password (will be hashed for comparison)
 * @return true if login successful, false otherwise
 */
public boolean login(String username, String password) {
    User user = users.get(username);
    if (user != null && user.getPassword().equals(hashPassword(password))) {
        currentUser = user;
        System.out.println("Login successful! Welcome, " + username + "!");
        return true;
    }
    return false;
}

/**
 * Logout the current user
 */
public void logout() {
    if (currentUser != null) {
        System.out.println("Goodbye, " + currentUser.getUsername() + "!");
        currentUser = null;
    }
}

/**
 * Get the currently logged in user
 * @return Current user or null if none
 */
public User getCurrentUser() {
    return currentUser;
}

/**
 * Create a new blog post
 * @param title Post title
 * @param content Post content
 */
public void createPost(String title, String content) {
    if (currentUser == null) {
        throw new IllegalStateException("You must be logged in to create a post");
    }

    if (title == null || title.trim().isEmpty()) {
        throw new IllegalArgumentException("Post title cannot be empty");
    }
```

```
    if (content == null || content.trim().isEmpty()) {
        throw new IllegalArgumentException("Post content cannot be empty");
    }

    Post newPost = new Post(title, content, currentUser.getUsername());
    posts.add(newPost);
}

/**
 * Get all posts in the system
 * @return List of all posts
 */
public List<Post> getAllPosts() {
    return new ArrayList<>(posts); // Return copy to prevent external modification
}

/**
 * Get posts by a specific user
 * @param username Username of the user
 * @return List of posts by the user
 */
public List<Post> getUserPosts(String username) {
    List<Post> userPosts = new ArrayList<>();
    for (Post post : posts) {
        if (post.getAuthor().equals(username)) {
            userPosts.add(post);
        }
    }
    return userPosts;
}

/**
 * Search posts by keyword
 * @param searchTerm Term to search for
 * @return List of matching posts
 */
public List<Post> searchPosts(String searchTerm) {
    if (searchTerm == null || searchTerm.trim().isEmpty()) {
        return new ArrayList<>();
    }

    List<Post> results = new ArrayList<>();
    String term = searchTerm.toLowerCase();

    for (Post post : posts) {
        if (post.getTitle().toLowerCase().contains(term) ||
            post.().toLowerCase().contains(term) ||
            post.getAuthor().toLowerCase().contains(term)) {
            results.add(post);
        }
    }
```

```
        return results;
    }

    /**
     * Delete an existing post
     * @param post Post to delete
     */
    public void deletePost(Post post) {
        if (currentUser == null || !post.getAuthor().equals(currentUser.getUsername())) {
            throw new IllegalStateException("You can only delete your own posts");
        }

        posts.remove(post);
    }

    /**
     * Hash a password for secure storage
     * @param password Raw password to hash
     * @return Hashed password
     */
    private String hashPassword(String password) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hash = digest.digest(password.getBytes(StandardCharsets.UTF_8));

            StringBuilder hexString = new StringBuilder();
            for (byte b : hash) {
                String hex = Integer.toHexString(0xff & b);
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);
            }
            return hexString.toString();
        } catch (NoSuchAlgorithmException e) {
            // Fallback to simple hash if SHA-256 is not available
            return String.valueOf(password.hashCode());
        }
    }
}
```

OUTPUT

```
Welcome to the Optimized Blog Platform!

===== MENU =====
1. Register
2. Login
3. Exit
Choose an option: 1
Enter username: alice
Enter password: alicepass
Enter email: alice@example.com
Registration successful! Please login.

===== MENU =====
1. Register
2. Login
3. Exit
Choose an option: 1
Enter username: bob
Enter password: bobpass
Enter email: bob@example.com
Registration successful! Please login.

===== MENU =====
1. Register
2. Login
3. Exit
Choose an option: 2
Enter username: alice
Enter password: alicepass
Login successful! Welcome, alice!

===== MENU =====
Logged in as: alice
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
6. Exit
Choose an option: 1
Enter post title: Alice's First Post
Enter post content (type END on a new line to finish):
Hello, this is Alice's first post!
END
Post created successfully!

===== MENU =====
Logged in as: alice
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
```

6. Exit
Choose an option: 1
Enter post title: Alice's Second Post
Enter post content (type END on a new line to finish):
Second post content.
END
Post created successfully!

===== MENU =====
Logged in as: alice
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
6. Exit
Choose an option: 5
Goodbye, alice!

===== MENU =====
1. Register
2. Login
3. Exit
Choose an option: 2
Enter username: bob
Enter password: bobpass
Login successful! Welcome, bob!

===== MENU =====
Logged in as: bob
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
6. Exit
Choose an option: 1
Enter post title: Bob's Post
Enter post content (type END on a new line to finish):
Hi, Bob here!
END
Post created successfully!

===== MENU =====
Logged in as: bob
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
6. Exit
Choose an option: 2

1. Alice's First Post by alice on 2025-05-21 21:07

2. Alice's Second Post by alice on 2025-05-21 21:07

3. Bob's Post by bob on 2025-05-21 21:08

Enter post number to view details (or 0 to go back): 1

----- Alice's First Post -----
By: alice | 2025-05-21 21:07

Hello, this is Alice's first post!

Comments:
No comments yet.

1. Add comment
0. Back
Choose an option: 1
Enter your comment: Welcome, Alice!
Comment added successfully!

----- Alice's First Post -----
By: alice | 2025-05-21 21:07

Hello, this is Alice's first post!

Comments:
bob (2025-05-21 21:08): Welcome, Alice!

1. Add comment
0. Back
Choose an option: 0

===== MENU =====
Logged in as: bob
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
6. Exit
Choose an option: 3

1. Bob's Post by bob on 2025-05-21 21:08

Enter post number to view details (or 0 to go back): 1

----- Bob's Post -----
By: bob | 2025-05-21 21:08

Hi, Bob here!

Comments:
No comments yet.

1. Add comment
2. Edit post
3. Delete post
0. Back
Choose an option: 2
Enter new title (or press Enter to keep current): Bob's Edited Post
Enter new content (type END on a new line to finish, or just END to keep current):
Hi, Bob here! I've edited my post.
END
Post updated successfully!

----- Bob's Edited Post -----
By: bob | 2025-05-21 21:08

Hi, Bob here! I've edited my post.

Comments:
No comments yet.

1. Add comment
2. Edit post
3. Delete post
0. Back
Choose an option: 3
Are you sure you want to delete this post? (y/n): y
Post deleted successfully!

===== MENU =====
Logged in as: bob
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
6. Exit
Choose an option: 4
Enter search term: Second

1. Alice's Second Post by alice on 2025-05-21 21:07

Enter post number to view details (or 0 to go back): 1

----- Alice's Second Post -----
By: alice | 2025-05-21 21:07

Second post content.

Comments:
No comments yet.

1. Add comment
0. Back
Choose an option: 0

===== MENU =====
Logged in as: bob
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
6. Exit
Choose an option: 5
Goodbye, bob!

===== MENU =====
1. Register
2. Login
3. Exit
Choose an option: 2
Enter username: alice
Enter password: alicepass
Login successful! Welcome, alice!

===== MENU =====
Logged in as: alice
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
6. Exit
Choose an option: 3

1. Alice's First Post by alice on 2025-05-21 21:07

2. Alice's Second Post by alice on 2025-05-21 21:07

Enter post number to view details (or 0 to go back): 1

----- Alice's First Post -----
By: alice | 2025-05-21 21:07

Hello, this is Alice's first post!

Comments:
bob (2025-05-21 21:08): Welcome, Alice!

1. Add comment
2. Edit post
3. Delete post
0. Back
Choose an option: 2

Enter new title (or press Enter to keep current): Alice's Welcome Post
Enter new content (type END on a new line to finish, or just END to keep current):
Welcome to my blog!
END
Post updated successfully!

----- Alice's Welcome Post -----
By: alice | 2025-05-21 21:07

Welcome to my blog!

Comments:
bob (2025-05-21 21:08): Welcome, Alice!

1. Add comment
2. Edit post
3. Delete post
0. Back
Choose an option: 3
Are you sure you want to delete this post? (y/n): n

----- Alice's Welcome Post -----
By: alice | 2025-05-21 21:07

Welcome to my blog!

Comments:
bob (2025-05-21 21:08): Welcome, Alice!

1. Add comment
2. Edit post
3. Delete post
0. Back
Choose an option: 0

===== MENU =====
Logged in as: alice
1. Create Post
2. View All Posts
3. View My Posts
4. Search Posts
5. Logout
6. Exit
Choose an option: 6
Thank you for using Blog Platform. Goodbye!

REFERENCES

1. **Java Official Documentation and Tutorials**
    - www.oracle.com/java/technologies/javase-documentation.html
    - www.tutorialspoint.com/java/index.htm
2. **Java Scanner, Collections, and OOP**
    - www.geeksforgeeks.org/scanner-class-in-java/
    - www.geeksforgeeks.org/arraylist-in-java/
    - www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/
3. **Password Hashing and Security**
    - www.baeldung.com/java-password-hashing
    - www.baeldung.com/java-security
4. **Date and Time API**
    - www.baeldung.com/java-8-date-time-intro
    - www.geeksforgeeks.org/localdatetime-in-java/
5. **Command-Line Interface (CLI) Programming**
    - www.baeldung.com/java-command-line-arguments
    - www.geeksforgeeks.org/command-line-interface-in-java/
6. **Best Practices and Java Community**
    - www.stackoverflow.com/questions/tagged/java
    - www.geeksforgeeks.org/best-practices-in-java/