



UE22CS352B - Object Oriented Analysis & Design

Mini Project Report

HelpDesk Management System

Submitted by:

Dhruthan M N (PES2UG22CS181)

Sanket B. Muttur (PES1UG23CS846)

Veeresh Amaragatti (PES1UG23CS847)

Vineet Goel (PES1UG22CS697)

6th Sem -L Section

Sowmya Shree P

January - May 2025

**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, India

Problem Statement:

In many organizations and online communities, addressing technical issues and sharing knowledge efficiently can be challenging due to the lack of a centralized platform. Existing systems often lack flexibility, are difficult to manage, or do not offer proper moderation and collaboration tools. There is a need for a robust, secure, and user-friendly platform that enables users to report problems, seek solutions, and contribute to a growing knowledge base—all while maintaining administrative control and content integrity.

The **HelpDesk Management System** is designed to solve this by providing a centralized, feature-rich web application where users can post technical issues, receive help from the community, and contribute to the resolution of problems through a structured and moderated platform.

Key Features

Major Use Cases

1. User Authentication and Profile Management:

- Registration
- Login (Normal user/Admin)
- View Profile
- Update Profile

This is a major use case because it's fundamental to the system - users need to authenticate before accessing most features, and profile management is essential for user identity.

2. Post Management:

- View Posts
- Create Posts
- Edit Posts
- Delete Posts
- Search Posts (by Title/Category/Tag/Content)

Post management represents the core functionality of the HelpDesk system. It's the primary way users interact with the platform and share information.

3. Comment and Interaction System:

- Write comments on posts
- Reply to comments
- Upvote/Downvote posts and comments

This is a major use case as it enables user engagement and interaction, which is central to a community-based help desk system.

4. Admin Content Moderation:

- View/Hide/Delete Posts
- View/Hide/Delete Comments
- Manage Categories
- Handle Reports

Content moderation is critical for maintaining quality and appropriate content on the platform, making it a major use case for administrators.

Minor Use Cases

1. User Role Management:

- Change user roles
- Suspend users
- Ban users

While important for administration, these features are used less frequently than the core content management functions.

2. Reporting System:

- Report posts
- Report comments

The reporting system is a supporting feature that helps maintain content quality but is used less frequently than the main interaction features.

3. Category and Tag Management:

- Create categories
- Edit categories
- Delete categories
- View posts by category/tag

This is classified as minor because it's primarily an organizational feature that supports the main content system.

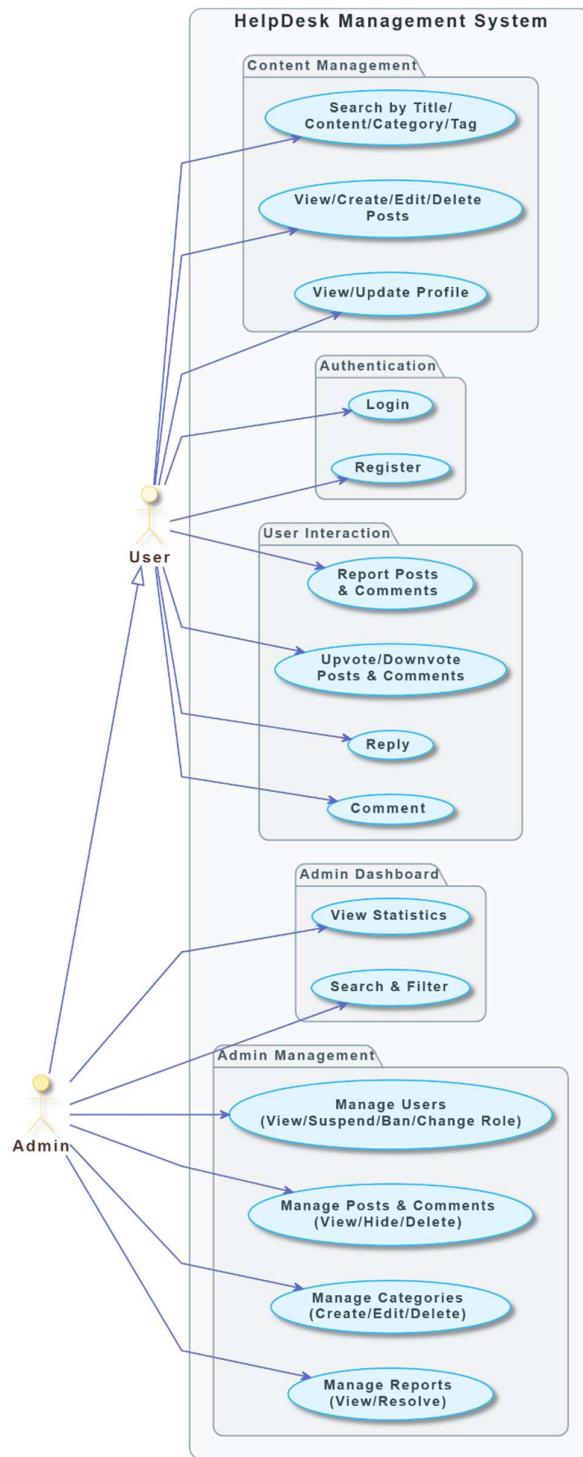
4. Admin Dashboard and Analytics:

- View statistics (users, posts, comments, reports)
- Search and filter
- Categorize handling of system entities

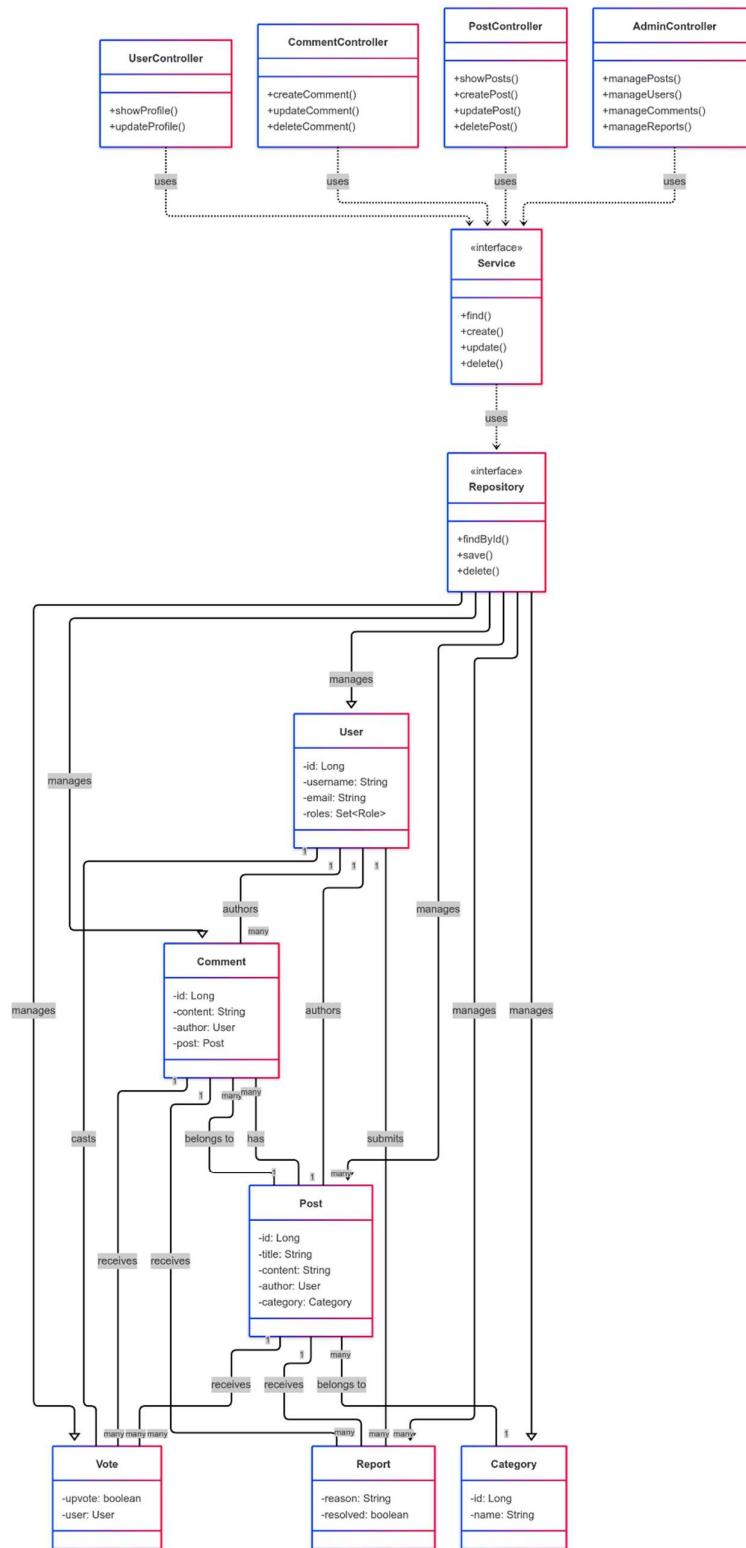
While valuable for administrators, these analytical features support decision-making rather than providing core functionality.

Models

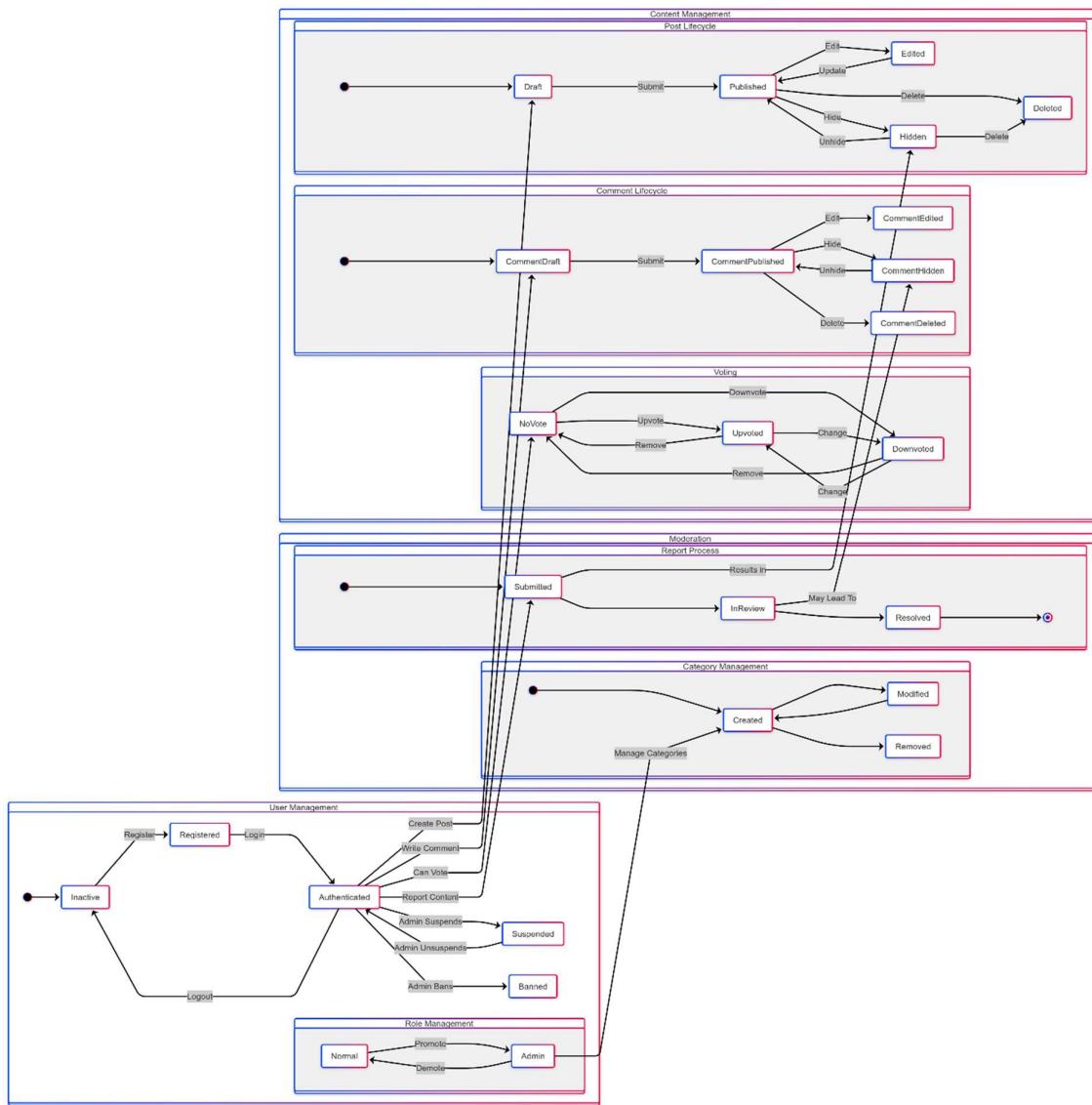
Use Case Diagram:



Class Diagram:

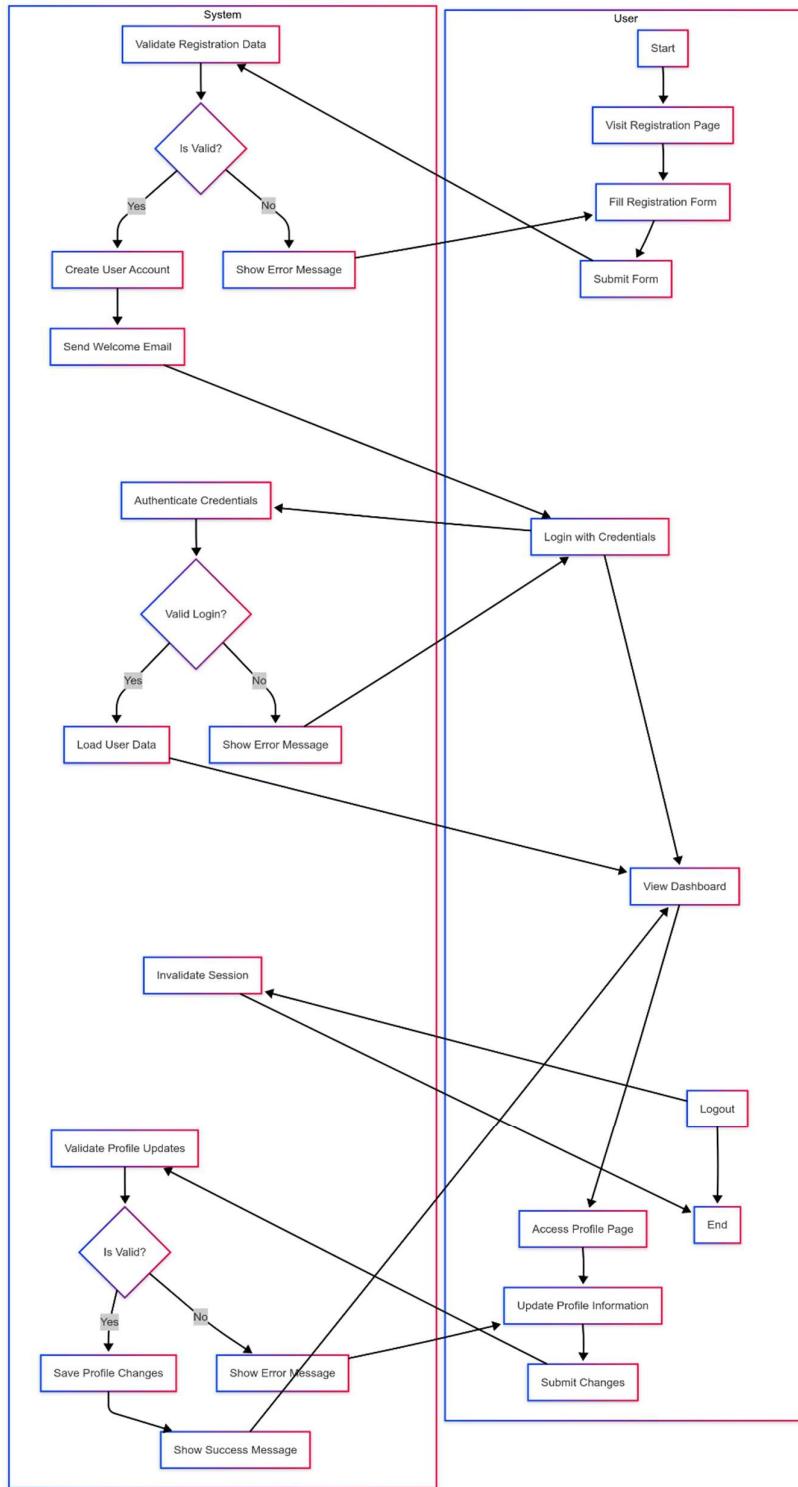


State Diagram:

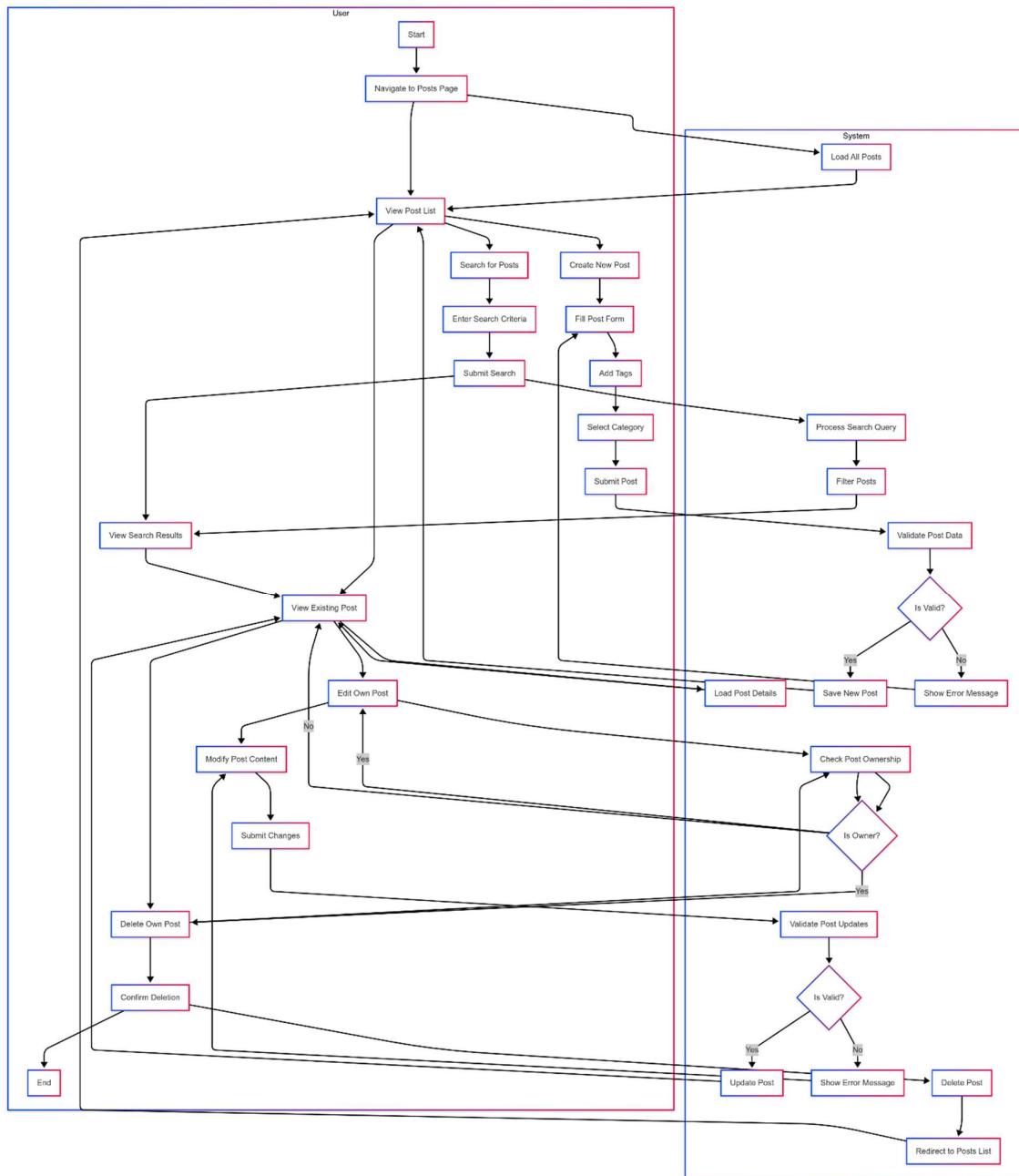


Activity Diagrams:

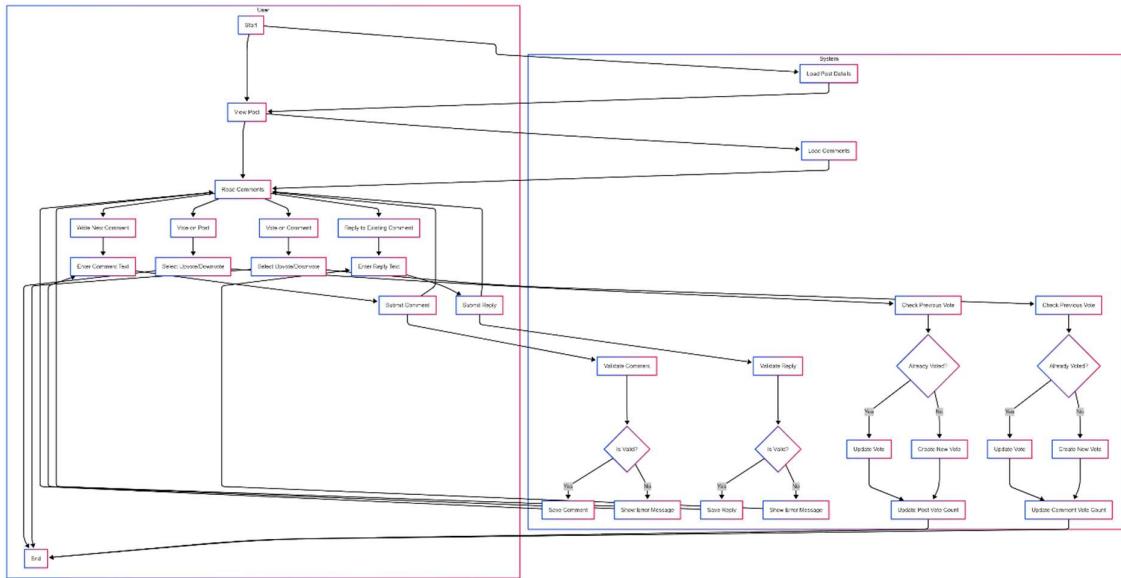
Major Use case 1 - User Authentication and Profile Management:



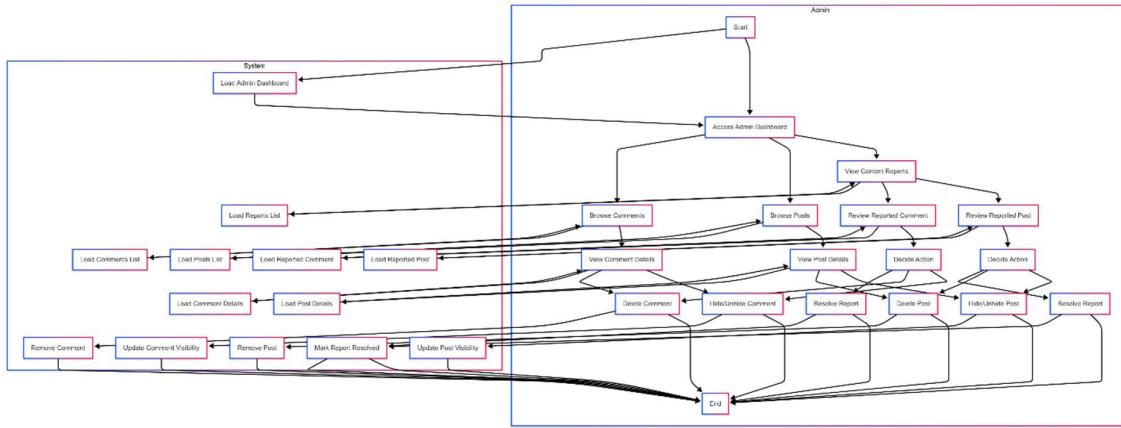
Major Use case 2 - Post Management:



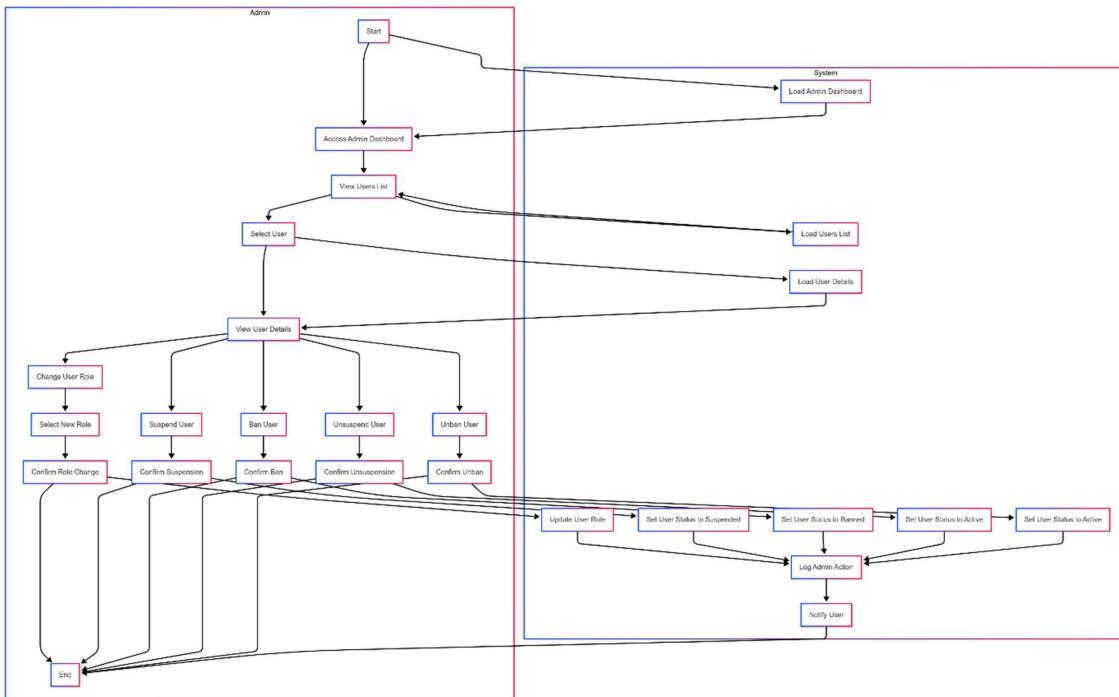
Major Use case 3 - Comment and Interaction System:



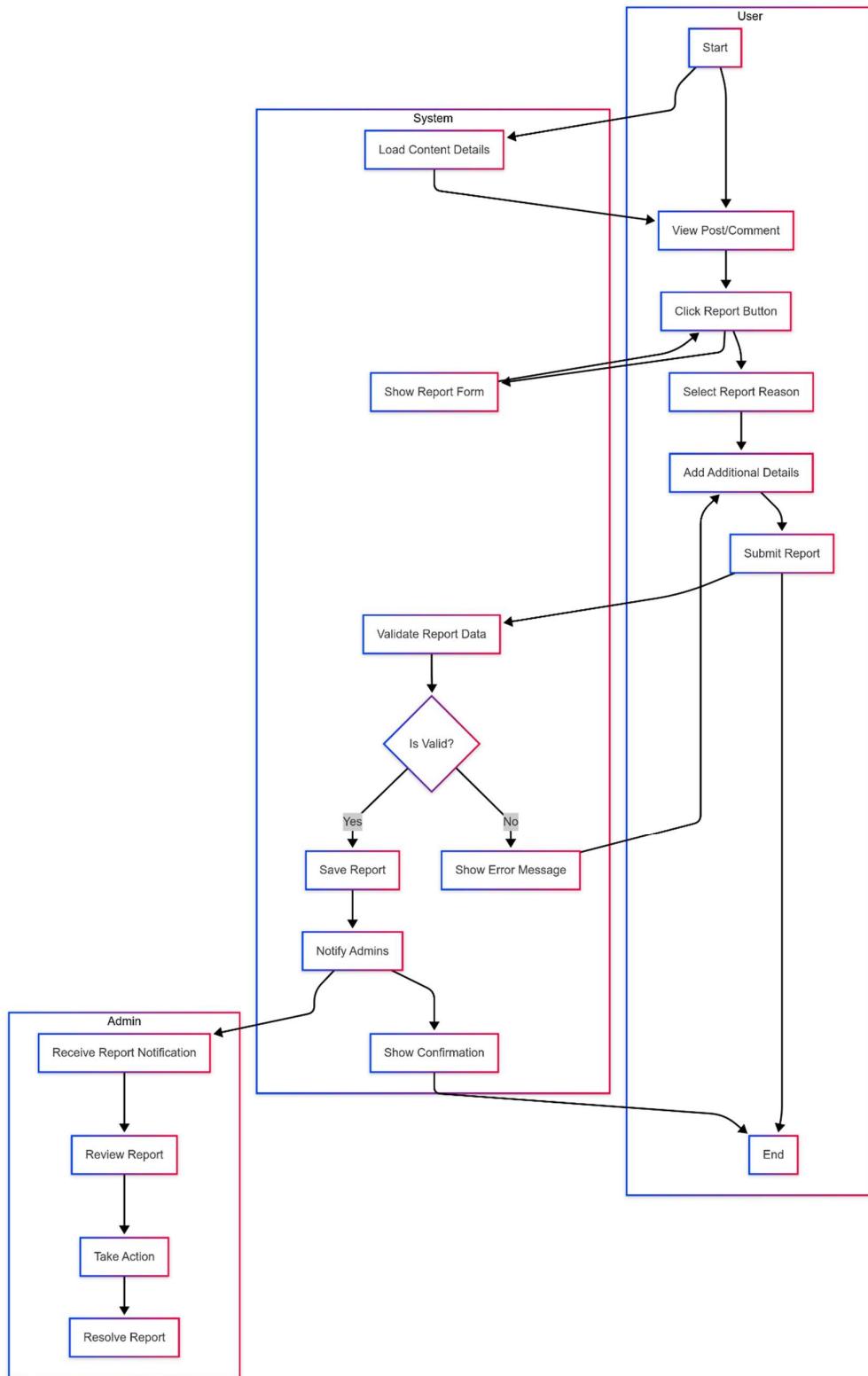
Major Use case 4 - Admin Content Moderation:



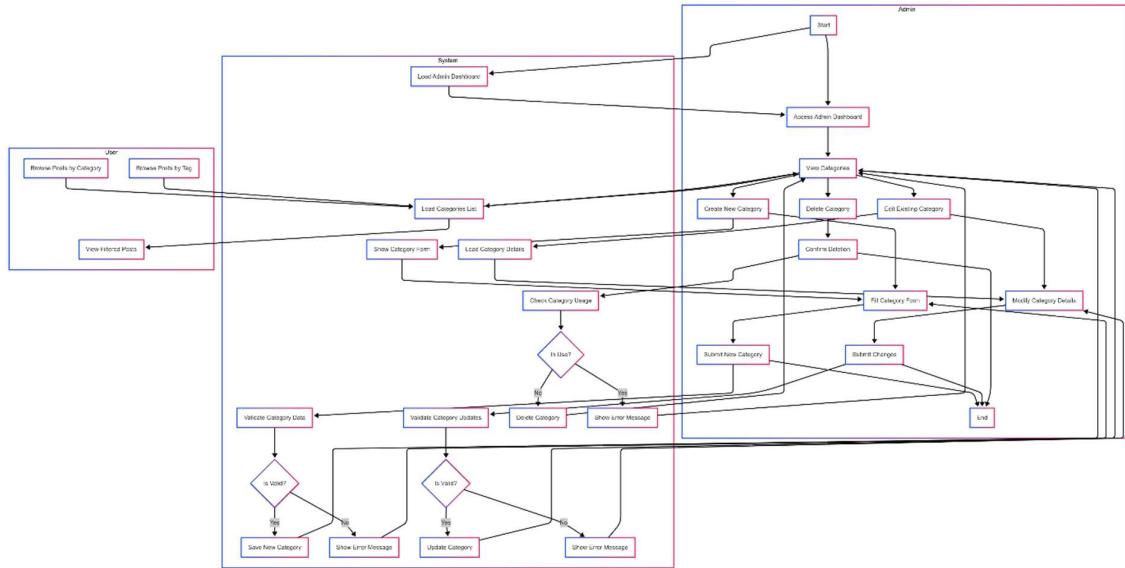
Minor Use case 1 - User Role Management:



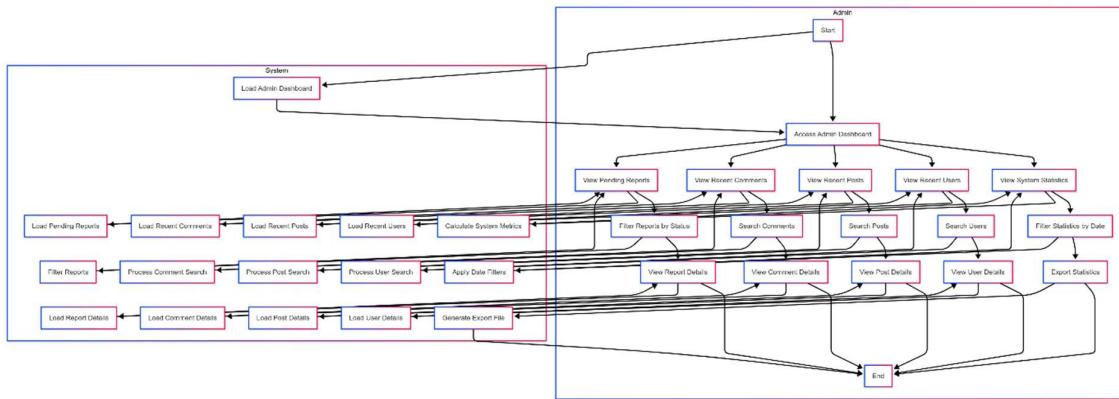
Minor Use case 2 - Reporting System:



Minor Use case 3 - Category and Tag Management:



Minor Use case 4 - Admin Dashboard and Analytics:



Architecture Patterns: Model-View-Controller (MVC) in HelpDesk Management System

Overview of MVC Architecture

The HelpDesk Management System implements the Model-View-Controller (MVC) architectural pattern, which separates the application into three interconnected components. This separation enhances maintainability, scalability, and testability of the application while promoting code reusability and parallel development.

Components of MVC in the HelpDesk System

1. Model Layer

The Model layer represents the application's data structure and business logic. It is responsible for retrieving, processing, and storing data.

Key Components:

1. **Domain Entities:** Java classes that represent the core business objects.
 - **User:** Represents system users with attributes like username, email, and roles
 - **Post:** Represents help desk posts created by users
 - **Comment:** Represents user comments on posts
 - **Category:** Represents post categories for organization
 - **Vote:** Represents user votes on posts and comments
 - **Report:** Represents content reports submitted by users
 - **Role:** Represents user roles for authorization
2. **Repositories:** Interfaces that extend Spring Data JPA's **JpaRepository** to provide data access operations.
 - **UserRepository:** Manages user data persistence
 - **PostRepository:** Handles post-related database operations
 - **CommentRepository:** Manages comment data
 - **CategoryRepository:** Handles category data
 - **VoteRepository:** Manages voting data
 - **ReportRepository:** Handles report data
 - **RoleRepository:** Manages role data

3. **Services:** Classes that implement business logic and act as an intermediary between controllers and repositories.

- **UserService:** Handles user-related operations like registration and profile management
- **PostService:** Manages post creation, retrieval, and moderation
- **CommentService:** Handles comment operations
- **CategoryService:** Manages category operations
- **VoteService:** Handles voting functionality
- **ReportService:** Manages content reporting

4. **Data Transfer Objects (DTOs):** Objects used to transfer data between subsystems.

- **UserDto, UserProfileDto, UserRegistrationDto:** Transfer user data
- **PostDto:** Transfers post data
- **CommentDto:** Transfers comment data
- **CategoryDto:** Transfers category data
- **VoteDto:** Transfers vote data
- **ReportDto:** Transfers report data

2. View Layer

The View layer is responsible for rendering the user interface and presenting data to users. In the HelpDesk system, this is implemented using Thymeleaf templates.

Key Components:

1. **Thymeleaf Templates:** HTML templates with Thymeleaf expressions for dynamic content rendering.

- **login.html, register.html:** Authentication views
- **dashboard.html:** Main user dashboard
- **profile.html:** User profile view
- **posts/list.html, posts/view.html, posts/create.html, posts/edit.html:** Post-related views
- **admin/dashboard.html, admin/users.html, admin/posts.html:** Admin views

2. **Static Resources:**

- CSS stylesheets for visual presentation
- JavaScript files for client-side functionality
- Images and other media assets

3. Controller Layer

The Controller layer handles user requests, processes them (often by interacting with the Model layer), and returns the appropriate View.

Key Components:

1. **Web Controllers:** Spring MVC controllers that handle HTTP requests.
 - o **AuthController:** Manages authentication operations
 - o **UserController:** Handles user profile operations
 - o **DashboardController:** Manages dashboard views
 - o **PostController:** Handles post-related operations
 - o **CommentController:** Manages comment operations
 - o **CategoryController:** Handles category operations
 - o **VoteController:** Manages voting functionality
 - o **ReportController:** Handles content reporting
 - o **AdminController:** Manages administrative functions
2. **REST Controllers:** Controllers that provide RESTful API endpoints.
 - o API versions of the controllers for programmatic access

Benefits of MVC in the HelpDesk System

1. **Separation of Concerns:** Each component has a specific responsibility, making the codebase more organized and maintainable.
2. **Code Reusability:** Models and business logic can be reused across different views and controllers.
3. **Parallel Development:** Different team members can work on models, views, and controllers simultaneously.
4. **Testability:** Components can be tested in isolation, facilitating unit testing.
5. **Flexibility:** Changes to one component have minimal impact on others, allowing for easier modifications and enhancements.

Design Principles in the HelpDesk Management System

1. SOLID Principles

Single Responsibility Principle (SRP)

The HelpDesk system demonstrates SRP by ensuring each class has only one reason to change:

- Model Classes: Each entity class (User, Post, Comment, etc.) represents a single domain concept
- Repository Interfaces: Each repository is focused on data access for a specific entity
- Service Classes: Services handle specific business logic for their respective domains
- Controllers: Each controller manages requests related to a specific feature area For example, the UserRepository focuses solely on user data access operations, while the UserService handles user-related business logic.

Open/Closed Principle (OCP)

The system is designed to be open for extension but closed for modification:

- Repository Interfaces: Spring Data JPA repositories allow extending functionality without modifying existing code
- Service Interfaces: Services are defined through interfaces, enabling different implementations
- DTOs: Data Transfer Objects facilitate extension of data structures without affecting core entities

Liskov Substitution Principle (LSP)

The system adheres to LSP through proper inheritance hierarchies:

- User Roles: Different user roles (Admin, Regular User) can be substituted without affecting the system's behavior
- Repository Interfaces: All repositories extend JpaRepository, ensuring consistent behavior

Interface Segregation Principle (ISP)

The system implements ISP by providing focused interfaces:

- Service Interfaces: Each service interface contains only methods relevant to its domain
- Repository Interfaces: Repositories define only methods needed for their specific entity

Dependency Inversion Principle (DIP)

The system follows DIP through dependency injection and abstraction:

- Constructor Injection: Dependencies are injected via constructors
- Interface-based Design: High-level modules depend on abstractions, not concrete implementations

2. Separation of Concerns (SoC)

The HelpDesk system clearly separates different concerns:

- Presentation Logic: Controllers handle HTTP requests and responses
- Business Logic: Services implement domain-specific operations
- Data Access Logic: Repositories manage database interactions
- Domain Logic: Entity classes encapsulate domain concepts and rules This separation makes the system easier to maintain and test, as changes to one concern don't affect others.

3. Don't Repeat Yourself (DRY)

The system avoids code duplication through:

- Common Base Classes: Shared functionality is extracted into base classes
- Utility Classes: Common operations are centralized in utility classes
- Template Fragments: Thymeleaf template fragments for reusable UI components

4. KISS (Keep It Simple, Stupid)

The system maintains simplicity through:

- Clear Naming Conventions: Classes, methods, and variables have descriptive names
- Focused Components: Each component has a clear, single purpose
- Straightforward Workflows: User journeys follow intuitive paths

5. YAGNI (You Aren't Gonna Need It)

The system avoids unnecessary complexity by:

- Implementing Required Features: Only features specified in requirements are implemented
- Minimal Dependencies: Only necessary libraries are included
- Focused Functionality: Each component provides only what's needed

6. Law of Demeter (Principle of Least Knowledge)

The system follows the Law of Demeter by:

- Encapsulated Access: Objects interact only with their immediate collaborators
- DTOs for Data Transfer: DTOs provide a clean interface between layers
- Service Abstraction: Controllers don't access repositories directly

7. Composition Over Inheritance

The system favors composition over inheritance:

- Service Composition: Services use other services through composition
- Feature Modules: Functionality is composed of smaller, focused components

8. Programming to Interfaces

The system implements programming to interfaces:

- Repository Interfaces: Data access is defined through interfaces
- Service Interfaces: Business logic is accessed through interfaces
- Dependency Injection: Components depend on interfaces, not implementations

9. Model-View-Controller (MVC)

As previously discussed, the system follows the MVC architectural pattern:

- Model: Entity classes, repositories, and services
- View: Thymeleaf templates
- Controller: Spring MVC controllers

10. Defensive Programming

The system employs defensive programming techniques:

- Input Validation: User inputs are validated before processing
- Error Handling: Exceptions are properly caught and handled
- Null Checks: Potential null values are checked before use

11. Fail-Fast Principle

The system implements the fail-fast principle:

- Early Validation: Inputs are validated as early as possible
- Immediate Error Reporting: Errors are reported immediately when detected
- Constraint Enforcement: Database constraints enforce data integrity

12. Principle of Least Surprise

The system follows predictable patterns:

- Consistent Naming: Similar operations have similar names
- Standard Workflows: Common operations follow standard patterns
- Familiar Interfaces: User interfaces follow common web patterns

13. Secure by Design

The system incorporates security principles:

- Authentication: Spring Security for user authentication
- Authorization: Role-based access control
- Input Sanitization: User inputs are sanitized to prevent attacks
- CSRF Protection: Cross-Site Request Forgery protection

14. Persistence Ignorance

The domain model is designed to be persistence-ignorant:

- Clean Domain Model: Entity classes focus on business concepts
- JPA Annotations: Persistence details are added through annotations
- Repository Pattern: Data access is abstracted through repositories

15. Convention over Configuration

The system leverages Spring Boot's convention over configuration:

- Standard Project Structure: Following Spring Boot conventions
- Default Configurations: Using sensible defaults
- Annotation-Based Configuration: Using annotations instead of XML

Conclusion

The HelpDesk Management System demonstrates a comprehensive application of modern software design principles. By adhering to these principles, the system achieves:

1. Maintainability: Code is organized, focused, and follows clear patterns
2. Extensibility: New features can be added with minimal changes to existing code
3. Testability: Components can be tested in isolation
4. Scalability: The system can grow without significant architectural changes
5. Security: Security concerns are addressed throughout the design
6. Usability: The system follows intuitive patterns for users These design principles work together to create a robust, flexible, and maintainable system that meets the needs of both users and administrators while providing a solid foundation for future enhancements.

Design Patterns in the HelpDesk Management System

1. Repository Pattern

The Repository pattern is extensively used to abstract the data access layer from the rest of the application.

Implementation:

- Each entity has a corresponding repository interface that extends JpaRepository
- Examples include UserRepository, PostRepository, CommentRepository, etc.

Benefits:

- Centralizes data access logic
- Provides a clean API for domain objects
- Facilitates unit testing through mock repositories
- Decouples business logic from data access details

2. Dependency Injection Pattern

Spring's dependency injection is used throughout the application to manage component dependencies.

Implementation:

- Constructor injection in controllers and services
- Autowired dependencies

Benefits:

- Loose coupling between components
- Easier unit testing through dependency mocking
- Centralized configuration of components
- Promotes the use of interfaces over concrete implementations

3. Data Transfer Object (DTO) Pattern

DTOs are used to transfer data between subsystems of the application.

Implementation:

- DTO classes for various entities (UserDto, PostDto, CommentDto, etc.)

- Lombok annotations (@Data, @NoArgsConstructor, @AllArgsConstructor) to reduce boilerplate

Benefits:

- Decouples the presentation layer from the domain model
- Allows for different data representations for different use cases
- Prevents exposing sensitive domain information
- Facilitates API versioning and evolution

4. Service Layer Pattern

The Service Layer pattern is used to encapsulate business logic.

Implementation:

- Service interfaces and their implementations
- Business logic contained within service classes

Benefits:

- Centralizes business logic
- Provides a clear API for controllers
- Facilitates transaction management
- Promotes separation of concerns

5. Front Controller Pattern

Spring MVC implements the Front Controller pattern through DispatcherServlet.

Implementation:

- Controller classes with request mapping annotations
- Centralized request handling

Benefits:

- Centralizes request handling logic
- Provides consistent handling of cross-cutting concerns
- Simplifies security implementation
- Facilitates view resolution

6. Model-View-Controller (MVC) Pattern

The entire application follows the MVC architectural pattern.

Implementation:

- Models: Entity classes and DTOs
- Views: Thymeleaf templates
- Controllers: Spring MVC controllers

Benefits:

- Separation of concerns
- Parallel development of components
- Improved maintainability
- Enhanced testability

7. Template Method Pattern

The Template Method pattern is used in service implementations for operations that follow a similar sequence but differ in specific steps.

Implementation:

- Base methods that define the algorithm structure
- Specialized methods that implement specific steps

Benefits:

- Reuse of common algorithm structure
- Customization of specific steps
- Reduced code duplication
- Consistent behavior across similar operations

8. Observer Pattern

Spring's event system implements the Observer pattern for loosely coupled event handling.

Implementation:

- Event publishers
- Event listeners
- Application events

Benefits:

- Loose coupling between components
- Simplified event-driven programming
- Scalable event handling
- Separation of concerns

9. Factory Method Pattern

Factory methods are used to create objects, particularly in service implementations.

Implementation:

- Methods that create and return objects
- Encapsulated object creation logic

Benefits:

- Encapsulation of object creation logic
- Centralized object creation
- Flexibility in object creation process
- Consistent object initialization

10. Builder Pattern

Lombok's `@Builder` annotation implements the Builder pattern for complex object construction.

Implementation:

- `@Builder` annotation on entity and DTO classes
- Fluent API for object construction

Benefits:

- Simplified construction of complex objects
- Improved readability of object creation code
- Optional parameters handling
- Immutable object creation

11. Singleton Pattern

Spring beans are singletons by default, implementing the Singleton pattern.

Implementation:

- Spring-managed beans with singleton scope
- Single instance per application context

Benefits:

- Controlled access to shared resources
- Reduced memory footprint
- Consistent state across the application
- Centralized configuration

12. Composite Pattern

The comment system implements a form of the Composite pattern for handling comment hierarchies.

Implementation:

- Comments can contain child comments (replies)
- Recursive processing of comment hierarchies

Benefits:

- Unified treatment of individual and composite objects
- Simplified client code
- Natural representation of hierarchical structures
- Extensibility for new types of components

13. Strategy Pattern

Different strategies for handling various types of content (posts, comments) implement the Strategy pattern.

Implementation:

- Different service implementations for different content types
- Common interfaces for similar operations

Benefits:

- Runtime selection of algorithms
- Encapsulation of algorithm details
- Simplified client code
- Extensibility for new strategies

14. Decorator Pattern

Spring Security filters implement the Decorator pattern to add security functionality to HTTP requests.

Implementation:

- Security filters that wrap HTTP requests
- Additional functionality added to base behavior

Benefits:

- Dynamic addition of responsibilities
- Flexible alternative to subclassing
- Composable behavior modifications
- Separation of concerns

15. Proxy Pattern

Spring's transaction management and security use the Proxy pattern to add cross-cutting concerns.

Implementation:

- Transaction proxies for service methods
- Security proxies for protected resources

Benefits:

- Transparent addition of functionality
- Separation of cross-cutting concerns
- Controlled access to resources
- Simplified client code

Conclusion

The HelpDesk Management System effectively implements numerous design patterns that contribute to its maintainability, flexibility, and adherence to best practices. These patterns work together to create a robust architecture that:

1. **Separates concerns:** Each component has a clear responsibility
2. **Promotes reusability:** Common patterns are applied consistently
3. **Enhances testability:** Components can be tested in isolation
4. **Facilitates maintenance:** Changes can be made with minimal impact
5. **Supports extensibility:** New features can be added without major restructuring

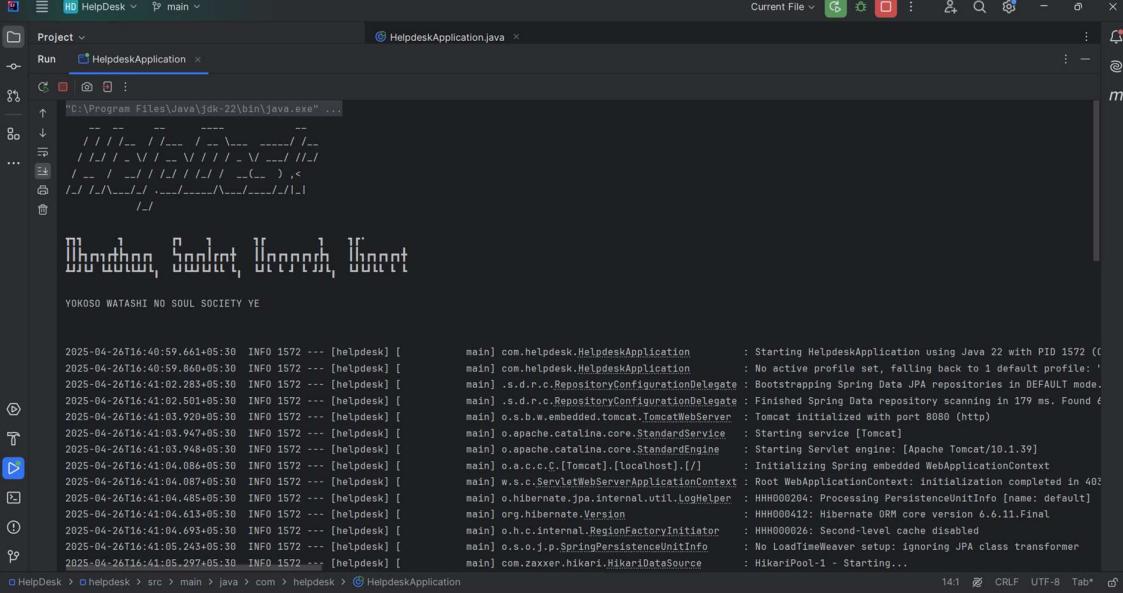
The consistent application of these design patterns demonstrates a well-architected system that follows industry best practices for enterprise application development.

GitHub Repo Link

GitHub [link](#) for [HelpDesk Management System](#)

Screenshots

Starting the Application:



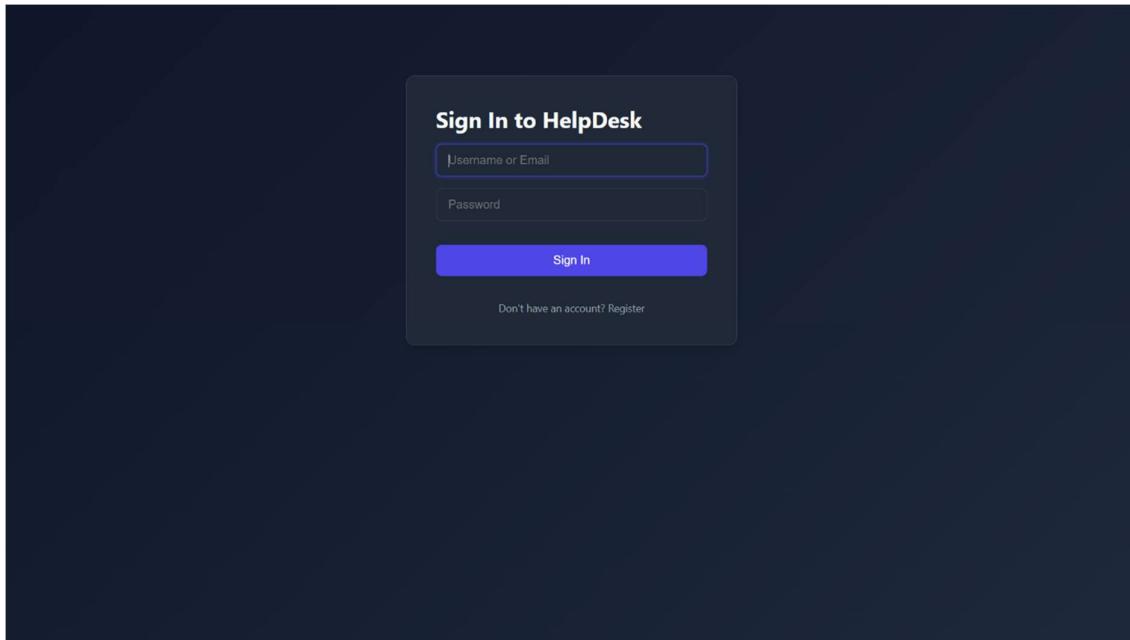
```

HelpDesk HelpDeskApplication.main()
"java -jar HelpdeskApplication.jar"
YOKOSO WATASHI NO SOUL SOCIETY YE

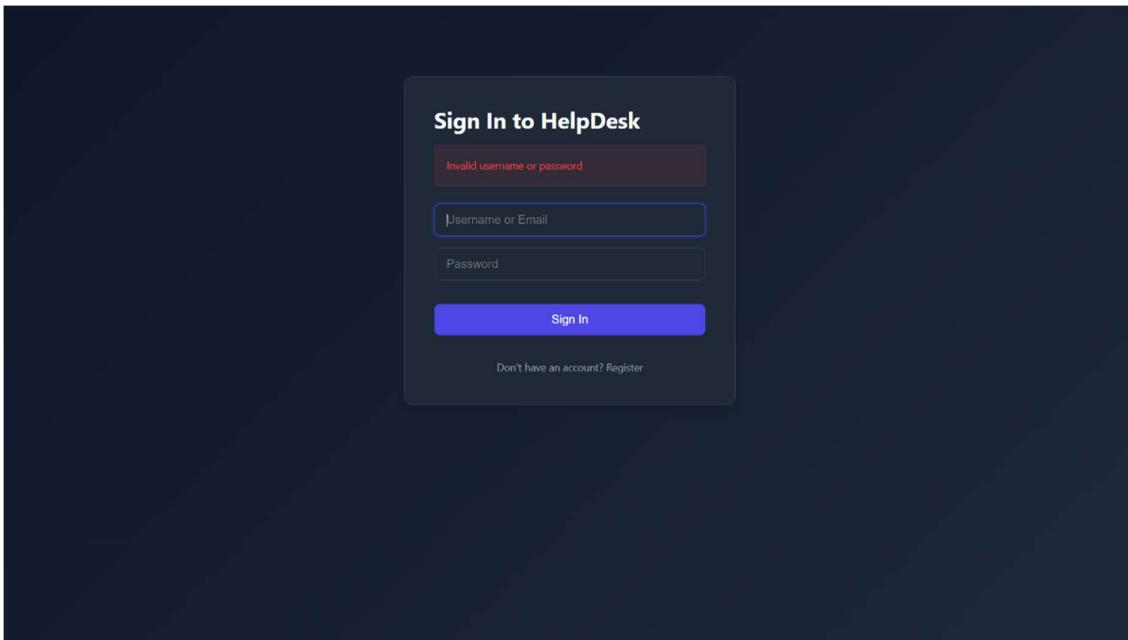
2025-04-26T16:40:59.661+05:30 INFO 1572 --- [helpdesk] [main] com.helpdesk.HelpdeskApplication : Starting HelpdeskApplication using Java 22 with PID 1572 ((...)
2025-04-26T16:40:59.866+05:30 INFO 1572 --- [helpdesk] [main] com.helpdesk.HelpdeskApplication : No active profile set, falling back to 1 default profile: '...'
2025-04-26T16:41:02.283+05:30 INFO 1572 --- [helpdesk] [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-04-26T16:41:02.503+05:30 INFO 1572 --- [helpdesk] [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 179 ms. Found 4
2025-04-26T16:41:03.920+05:30 INFO 1572 --- [helpdesk] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-04-26T16:41:03.947+05:30 INFO 1572 --- [helpdesk] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-04-26T16:41:03.948+05:30 INFO 1572 --- [helpdesk] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.39]
2025-04-26T16:41:04.086+05:30 INFO 1572 --- [helpdesk] [main] o.a.c.c.Tomcat@localhost{/} : Initializing Spring embedded WebApplicationContext
2025-04-26T16:41:04.087+05:30 INFO 1572 --- [helpdesk] [main] w.s.c.WebServerApplicationContext : Root WebApplicationContext: initialization completed in 403
2025-04-26T16:41:04.485+05:30 INFO 1572 --- [helpdesk] [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-04-26T16:41:04.613+05:30 INFO 1572 --- [helpdesk] [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.11.Final
2025-04-26T16:41:04.693+05:30 INFO 1572 --- [helpdesk] [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2025-04-26T16:41:05.243+05:30 INFO 1572 --- [helpdesk] [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-04-26T16:41:05.297+05:30 INFO 1572 --- [helpdesk] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...

```

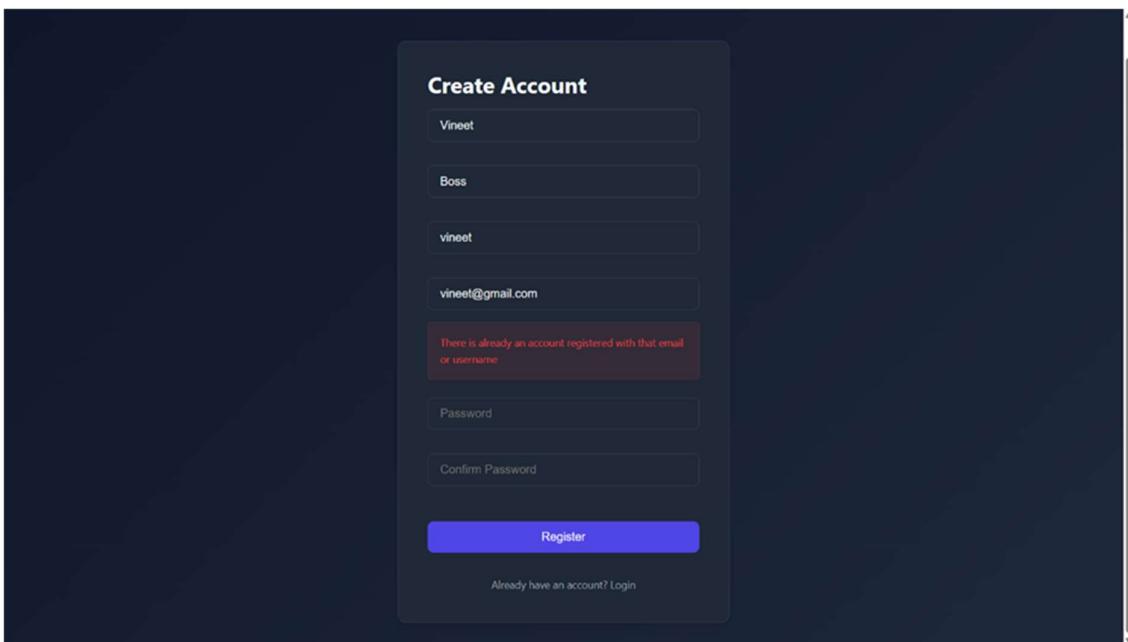
Sign In



Authenticaion



Invalid Registration



Registering

The screenshot shows a registration form titled "Create Account". The fields filled in are:

- First Name: Sanket
- Last Name: Muttur
- Username: sanket
- Email: sanket@gmail.com
- Password: (obscured)
- Confirm Password: (obscured)

A blue "Register" button is at the bottom, and a link "Already have an account? Login" is at the bottom right.

Successful Registration

The screenshot shows the same registration form as above, but with a green success message at the top: "Registration successful! Please login." The other fields and layout remain the same.

User Dashboard after Sign In

The screenshot shows the user dashboard for 'HelpDesk'. At the top, there's a navigation bar with links for Dashboard, Posts, Categories, Profile, and Logout. Below the navigation, a welcome message reads 'Welcome to HelpDesk, vineet!'. A 'Quick Actions' section contains buttons for 'Create New Post', 'Browse Posts', 'Browse Categories', and 'View Profile'. The main content area is titled 'Recent Posts' and displays two posts:

- One Piece**
One of the most awesome things about One Piece is how it seamlessly blends epic adventure with deep storytelling, all while keeping its world rich and vibrant. Take the Grand Line, for example. It...
Category: Anime Tags: OnePiece GOAT Posted by vineet on 23 Apr 2025
- Solo Leveling**
In Solo Leveling, aura farming is a process by which Sung Jinwoo uses his unique abilities to absorb aura or energy from monsters he defeats, which enhances his powers and allows him to level up. T...
Category: Anime Tags: SoloLeveling auraFarming Posted by vineet on 23 Apr 2025

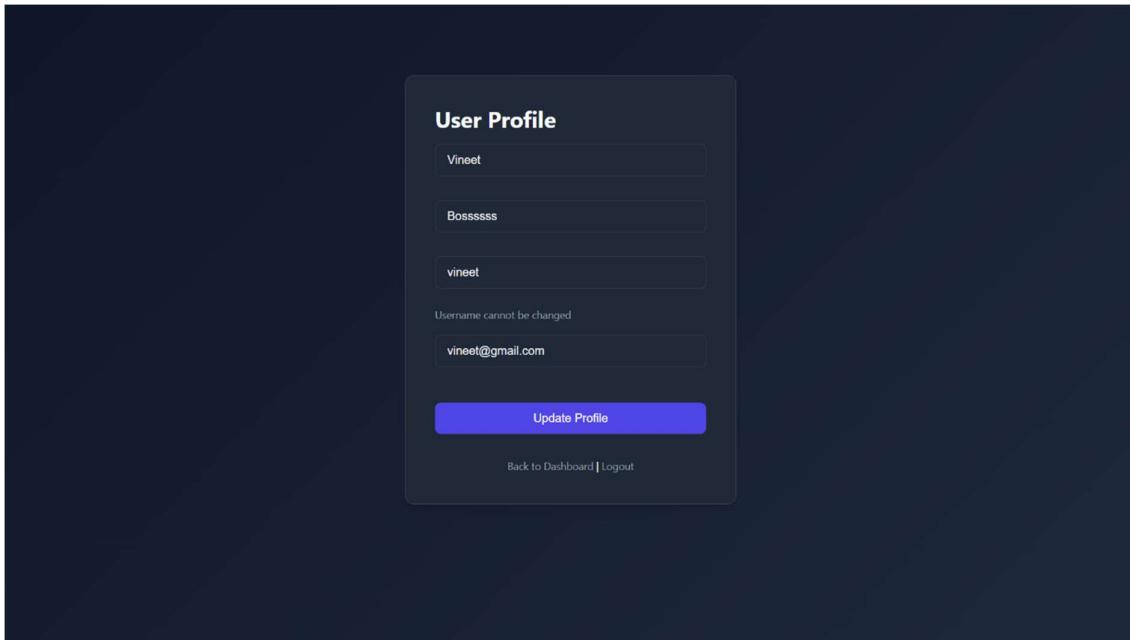
User Profile

The screenshot shows the 'User Profile' edit screen. It features a form with the following fields:

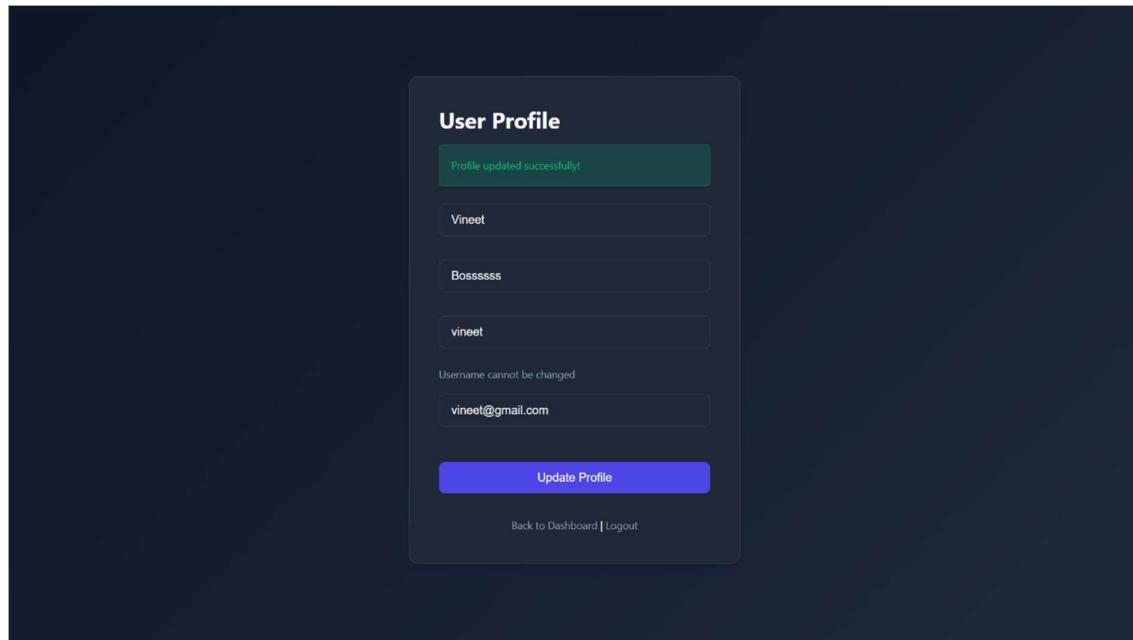
- First Name: Vineet
- Last Name: Boss
- Email: vineet@gmail.com

A note below the first name field states: 'Username cannot be changed'. At the bottom of the form is a blue 'Update Profile' button. Below the form, there are links for 'Back to Dashboard' and 'Logout'.

Updating User Profile



User Profile Update Successful



Viewing Posts

The screenshot shows a dark-themed web application interface. At the top, there's a navigation bar with the title "HelpDesk" and links for "Dashboard", "Posts", "Categories", "Profile", and "Logout". Below the navigation, a post card is displayed. The post has a title "Cloud Computing: Revolutionizing Scalability and Flexibility" with a small icon above it. It was posted by "Dhruthan Boss" in the "Networking" category on April 23, 2025. The post includes a short description about cloud computing's scalability and flexibility, mentioning auto-scaling, storage flexibility, and geographic distribution. Below the post, there's a section titled "Comments" with a "Add a Comment" button.

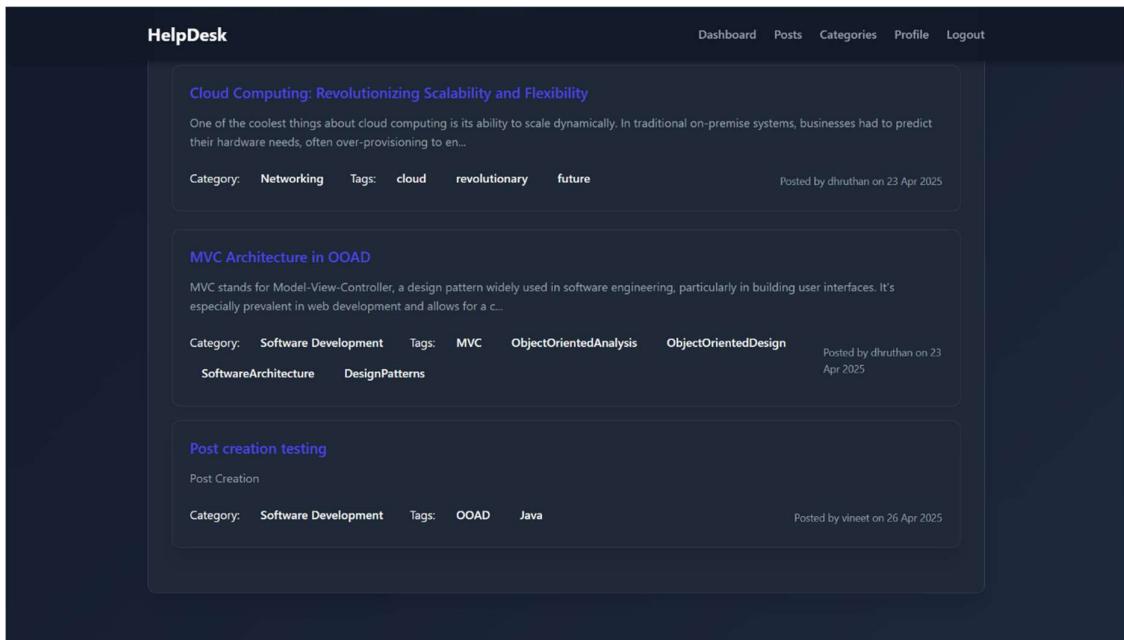
Post Creation

The screenshot shows the "Create New Post" form. The title is "Create New Post". The form fields include:

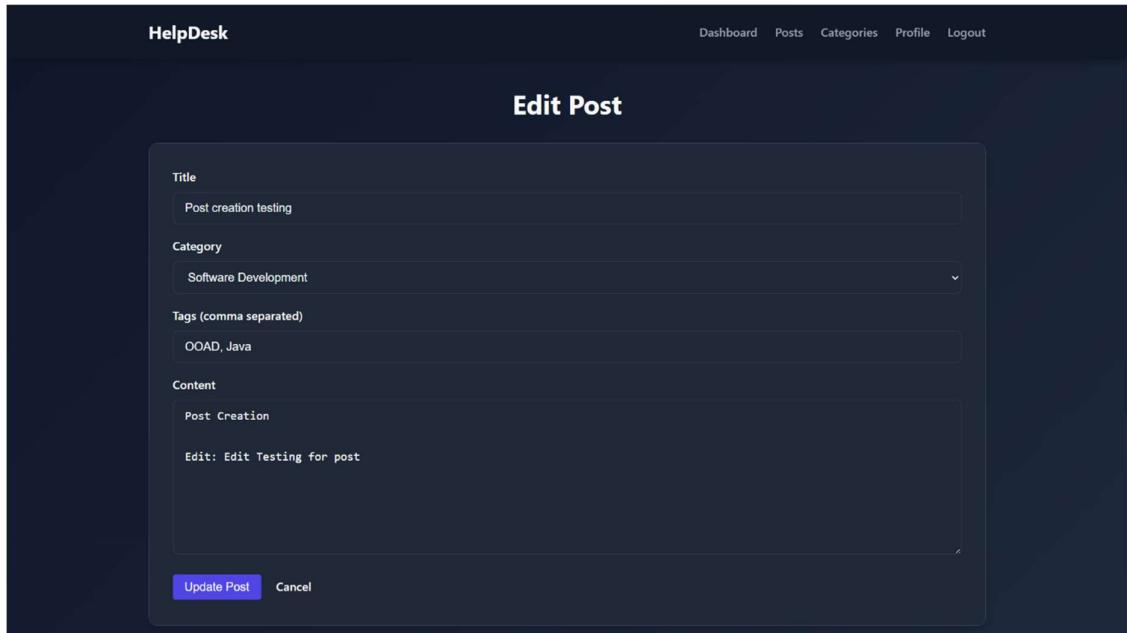
- Title:** Post creation testing
- Category:** Software Development
- Tags (comma separated):** OOAD, Java
- Content:** Post Creation

At the bottom of the form are two buttons: "Create Post" and "Cancel".

Post Created Successfully



Editing the Post



Successfully edited the post

The screenshot shows a dark-themed web application interface. At the top, there's a navigation bar with links for Dashboard, Posts, Categories, Profile, and Logout. Below the navigation, a post card is displayed for a post titled "Post creation testing". The post has 0 upvotes and 0 downvotes. It was posted by "Vineet Bossssss" in "Software Development" on April 26, 2025, and was updated on the same day. The post content is "Post Creation", and there's a link "Edit: Edit Testing for post". Below the post card, there's a "Comments" section with a placeholder "Add a Comment" and a "Submit Comment" button. A small modal window is open in the center, asking "localhost:8080 says Are you sure you want to delete this post?" with "OK" and "Cancel" buttons.

Deleting the Post

This screenshot shows the same application interface after the deletion process has been initiated. The post card for "Post creation testing" is still visible, but the modal window from the previous screenshot is now prominent in the center. It displays the message "localhost:8080 says Are you sure you want to delete this post?" with "OK" and "Cancel" buttons. The URL "localhost:8080/posts/delete/5" is visible at the bottom of the page.

Search Post by Title

The screenshot shows a search results page titled "Search Results for: OOAd". At the top, there are two buttons: "Back to All Posts" and "Create New Post". Below them is a search bar containing the tag "OOAd" and a "Search" button. The main content area displays a single post card. The title of the post is "MVC Architecture in OOAD". The post content is: "MVC stands for Model-View-Controller, a design pattern widely used in software engineering, particularly in building user interfaces. It's especially prevalent in web development and allows for a c...". Below the content, the post details are listed: Category: Software Development, Tags: MVC, ObjectOrientedAnalysis, ObjectOrientedDesign, SoftwareArchitecture, DesignPatterns. To the right, it shows "Posted by dhruthan on 23 Apr 2025".

Search Post by Tag

The screenshot shows a search results page titled "Search Results for: goat". At the top, there are two buttons: "Back to All Posts" and "Create New Post". Below them is a search bar containing the tag "goat" and a "Search" button. The main content area displays a single post card. The title of the post is "One Piece". The post content is: "One of the most awesome things about One Piece is how it seamlessly blends epic adventure with deep storytelling, all while keeping its world rich and vibrant. Take the Grand Line, for example. It'...". Below the content, the post details are listed: Category: Anime, Tags: OnePiece, GOAT. To the right, it shows "Posted by vineet on 23 Apr 2025".

Search Post by Category

The screenshot shows a dark-themed web application interface. At the top, there is a navigation bar with the title "HelpDesk" on the left and links for "Dashboard", "Posts", "Categories", "Profile", and "Logout" on the right. Below the navigation bar, the main content area has a heading "Search Results for: networking". There is a search bar with the query "networking" and a "Search" button. Below the search bar, a post card displays the following information:
Category: Networking Tags: cloud, revolutionary, future Posted by dhruthan on 23 Apr 2025
The post content is: "Cloud Computing: Revolutionizing Scalability and Flexibility
One of the coolest things about cloud computing is its ability to scale dynamically. In traditional on-premise systems, businesses had to predict their hardware needs, often over-provisioning to ensure..."

Search Post by Content

The screenshot shows a dark-themed web application interface. At the top, there is a navigation bar with the title "HelpDesk" on the left and links for "Dashboard", "Posts", "Categories", "Profile", and "Logout" on the right. Below the navigation bar, the main content area has a heading "Search Results for: Sung Jinwoo". There is a search bar with the query "Sung Jinwoo" and a "Search" button. Below the search bar, a post card displays the following information:
Category: Anime Tags: SoloLeveling, auraFarming Posted by vineet on 23 Apr 2025
The post content is: "Solo Leveling, aura farming is a process by which Sung Jinwoo uses his unique abilities to absorb aura or energy from monsters he defeats, which enhances his powers and allows him to level up. T..."

View Post by Category

The screenshot shows a dark-themed web application interface. At the top, there's a navigation bar with the title "HelpDesk" on the left and links for "Dashboard", "Posts", "Categories", "Profile", and "Logout" on the right. Below the navigation, the main content area has a heading "Posts in Category: Anime". There are two posts listed in a card format:

- One Piece**
One of the most awesome things about One Piece is how it seamlessly blends epic adventure with deep storytelling, all while keeping its world rich and vibrant. Take the Grand Line, for example. It...
Tags: [OnePiece](#) [GOAT](#) Posted by vineet on 23 Apr 2025
- Solo Leveling**
In Solo Leveling, aura farming is a process by which Sung Jinwoo uses his unique abilities to absorb aura or energy from monsters he defeats, which enhances his powers and allows him to level up. T...
Tags: [SoloLeveling](#) [auraFarming](#) Posted by vineet on 23 Apr 2025

View Post by Tag

The screenshot shows a dark-themed web application interface. At the top, there's a navigation bar with the title "HelpDesk" on the left and links for "Dashboard", "Posts", "Categories", "Profile", and "Logout" on the right. Below the navigation, the main content area has a heading "Posts tagged with: cloud". There is one post listed in a card format:

- Cloud Computing: Revolutionizing Scalability and Flexibility**
One of the coolest things about cloud computing is its ability to scale dynamically. In traditional on-premise systems, businesses had to predict their hardware needs, often over-provisioning to en...
Category: [Networking](#) Tags: [cloud](#) [revolutionary](#) [future](#) Posted by dhruthan on 23 Apr 2025

Upvoting the Post/Comment

HelpDesk

Dashboard Posts Categories Profile Admin Logout

One Piece Edit Delete

2 ↑ ↓

Posted by Vineet Bossssss in [Anime](#) on Apr 23, 2025

Tags: [OnePiece](#) [GOAT](#)

One of the most awesome things about One Piece is how it seamlessly blends epic adventure with deep storytelling, all while keeping its world rich and vibrant. Take the Grand Line, for example. It's this mysterious, dangerous, and incredibly diverse sea where pirates go to find the One Piece, the legendary treasure. What's mind-blowing is how the Grand Line is filled with unique islands, each with its own culture, wildlife, and way of life. From the tropical paradise of Skypiea to the icy, heart-pounding waters of the New World, every arc feels like a completely different world, yet they all tie into the overarching narrative.

Also, the way the series develops its characters is insane! Luffy and the crew are all deeply passionate about their dreams, and their individual backstories—like Zoro's loyalty to his late friend, or Nami's tragic history with Arlong—make them so much more than just adventurous pirates. It's hard not to get emotionally invested, especially when One Piece weaves themes like freedom, friendship, and overcoming adversity throughout the story. There's just so much depth behind the fun and wacky moments! What's your favorite part of One Piece?

Comments

[Add a Comment](#)

Downvoting the Post/Comment

HelpDesk

Dashboard Posts Categories Profile Admin Logout

Solo Leveling Edit Delete

-2 ↑ ↓

Posted by Vineet Bossssss in [Anime](#) on Apr 23, 2025 (updated on Apr 23, 2025)

Tags: [SoloLeveling](#) [auraFarming](#)

In Solo Leveling, aura farming is a process by which Sung Jinwoo uses his unique abilities to absorb aura or energy from monsters he defeats, which enhances his powers and allows him to level up. This process is vital for his progression throughout the series.

Here's how aura farming works for Jinwoo:
The System: After Jinwoo is given the Player System, he gains the ability to level up in a way that mimics a video game. This system tracks his progress, stats, and even his abilities, allowing him to grow exponentially compared to other hunters. This is a key element of his aura farming process, as the system automatically detects and absorbs energy from defeated monsters.

Aura from Monsters: Each time Jinwoo defeats monsters in dungeons or gates, he can absorb their aura or spiritual energy. The monsters' energy is essentially left behind when they die, and Jinwoo's system allows him to collect it. This aura can be used to enhance his strength, agility, magic, and other attributes.

Aura as a Power Source: The aura Jinwoo collects doesn't just passively increase his stats. It's a powerful resource that helps him grow in strength faster than any other hunter. Unlike other hunters who rely on their inherent strength or external magical abilities, Jinwoo uses the monsters' aura to fuel his personal growth.

High-Level Dungeons and Gates: The more powerful the monsters Jinwoo faces, the stronger the aura he can farm. In some high-level dungeons, the aura Jinwoo collects is extremely potent, making these locations prime spots for him to farm powerful energy. This becomes especially useful as he

Commenting

The screenshot shows the 'Comments' section of the HelpDesk application. At the top, there's a form for 'Add a Comment' with a text area containing 'Comment Testing'. Below it is a 'Submit Comment' button. The main area displays a comment from 'Vineet Bossssss' posted on April 23, 2025, at 12:22 PM. The comment reads: 'I don't care 😐😐'. It has 0 upvotes and 0 downvotes. Below this is a reply from 'Dhruthan Boss' posted on April 23, 2025, at 12:26 PM. The reply reads: 'Oh, Vineet, I don't care? Wow, that's the most effort you've put into anything all day. You should seriously consider getting a medal for 'Most Useless Comment of the Year.' Maybe if you cared a little, you wouldn't sound like an extra from a motivational speaker's worst nightmare. It's impressive how much effort you put into not caring. Maybe next time you could not care in silence.' It also has 0 upvotes and 0 downvotes. Both comments have 'Reply' and 'Report' links.

Replies to the Comment

This screenshot shows a reply to the previous comment from 'Vineet Bossssss'. The reply was posted on April 26, 2025, at 05:02 PM and reads: 'Comment Testing'. It has 0 upvotes and 0 downvotes. Below this is another reply from 'Vineet Bossssss' posted on April 26, 2025, at 05:02 PM, which reads: 'Reply to the Comment Testing'. This reply also has 0 upvotes and 0 downvotes. Both replies have 'Reply' and 'Report' links.

Reporting Post

The screenshot shows a dark-themed web application interface. At the top, there's a navigation bar with links for Dashboard, Posts, Categories, Profile, and Logout. Below the navigation, a post card for "MVC Architect" is displayed. The post is by "Dhruthan Boss" in the "Software Dev" category, with 0 upvotes and 0 downvotes. It has tags for MVC and ObjectOrientedAnalysis. A reporting modal is open over the post, titled "Report Post". The modal has a dropdown menu for "Reason for reporting" set to "Misinformation". At the bottom of the modal are "Cancel" and "Report" buttons. The main content area below the post contains a brief explanation of MVC and OOA, followed by a numbered list of components (1. Model) and examples.

Reporting Comment

This screenshot shows the same application interface as the previous one. It displays a thread of comments under the "MVC Architect" post. The first comment is from "Vineet Bossssss" with the text "I don't care 😐 😐". A reporting modal is open over this comment, titled "Report Comment". The "Reason for reporting" dropdown is set to "Harassment or bullying". The modal includes "Cancel" and "Report" buttons. The main content area shows the full comment and a reply from "Dhruthan Boss" where he mocks the comment. Other comments in the thread include "Comment Testing" from "Vineet Bossssss" and a timestamp of April 26, 2025, 05:02 PM.

Admin Dashboard

The screenshot shows the Admin Dashboard interface. At the top, there's a navigation bar with links for Dashboard, Posts, Categories, Profile, Admin, and Logout. Below the navigation is a title 'Admin Dashboard' and a sub-navigation bar with tabs for Dashboard, Users, Posts, Comments, and Reports. The main area displays four summary cards: 'Total Users' (5), 'Total Posts' (4), 'Total Comments' (8), and 'Reports' (3). Below these cards is a section titled 'Recent Users' containing a table with three rows of user data:

Username	Email	Status	Created	Actions
admin	admin@helpdesk.com	ACTIVE	2025-04-23	View
vineet	vineet@gmail.com	ACTIVE	2025-04-23	View
dhruthan	dhruthan@gmail.com	ACTIVE	2025-04-23	View

User Management (Can View individual User profile and Suspend the User)

The screenshot shows the User Management page. At the top, there's a navigation bar with links for Dashboard, Posts, Categories, Profile, Admin, and Logout. Below the navigation is a title 'User Management' and a sub-navigation bar with tabs for Dashboard, Users, Posts, Comments, and Reports. There are search and filter inputs at the top. The main area displays a table of user data:

ID	Username	Email	Name	Status	Roles	Posts	Comments	Created	Actions
1	admin	admin@helpdesk.com	Admin User	ACTIVE	ADMIN USER	0	0	2025-04-23	View Suspend
2	vineet	vineet@gmail.com	Vineet Bossssss	ACTIVE	USER	2	4	2025-04-23	View Suspend
3	dhruthan	dhruthan@gmail.com	Dhruthan Boss	ACTIVE	USER	2	4	2025-04-23	View Suspend
4	veeresh	veeresh@gmail.com	Veeresh Boss	ACTIVE	USER	0	0	2025-04-26	View Suspend
5	sanket	sanket@gmail.com	Sanket Muttur	ACTIVE	USER	0	0	2025-04-26	View Suspend

Post Management (Can View/Hide/Unhide/Delete the particular Post)

The screenshot shows the 'Post Management' section of the HelpDesk application. At the top, there's a navigation bar with links for Dashboard, Posts, Categories, Profile, Admin, and Logout. Below that is a sub-navigation bar with links for Dashboard, Users, Posts (which is highlighted in purple), Comments, and Reports. A search bar labeled 'Search posts...' is followed by a category dropdown set to 'All Categories' and a status dropdown set to 'All'. The main area contains a table with the following data:

ID	Title	Author	Category	Created	Status	Votes	Comments	Actions
1	One Piece	Vineet Bossssss	Anime	2025-04-23	Visible	1	1	
2	Solo Leveling	Vineet Bossssss	Anime	2025-04-23	Visible	-1	1	
3	Cloud Computing: Revolutionizing Scalability and Flexibility	Dhruthan Boss	Networking	2025-04-23	Visible	0	2	
4	MVC Architecture in OOAD	Dhruthan Boss	Software Development	2025-04-23	Visible	0	4	

At the bottom of the page, a copyright notice reads: © 2023 HelpDesk. All rights reserved.

Comment Management (Can View/Hide/Unhide/Delete the particular Comment)

The screenshot shows the 'Comment Management' section of the HelpDesk application. The layout is similar to the Post Management page, with a navigation bar at the top and a sub-navigation bar below it. A search bar labeled 'Search comments...' is followed by a status dropdown set to 'All'. The main area contains a table with the following data:

ID	Content	Author	Post	Created	Status	Votes	Actions
1	Weebs 😊😊	Dhruthan Boss	Post #1	2025-04-23	Visible	0	
2	I don't watch anime 😞	Dhruthan Boss	Post #2	2025-04-23	Visible	0	
3	Nerd 😊😊	Vineet Bossssss	Post #3	2025-04-23	Visible	0	
4	I don't care 😐😐	Vineet Bossssss	Post #4	2025-04-23	Visible	0	
5	Don't want to hear it coming from a weeb 😊😊	Dhruthan Boss	Post #3	2025-04-23	Visible	0	
6	Oh, Vineet, I don't care? Wow, that's the mos...	Dhruthan Boss	Post #4	2025-04-23	Visible	0	
7	Comment Testing	Vineet Bossssss	Post #4	2025-04-26	Visible	0	
8	Reply to the Comment Testing	Vineet Bossssss	Post #4	2025-04-26	Visible	0	

Report Management (Can View/Hide/Unhide/Mark as Resolved for individual Reported Post/Comment)

The screenshot shows the 'Reports Management' section of the HelpDesk application. At the top, there are navigation links: Dashboard, Users, Posts, Comments, Reports, and a purple button labeled 'Logout'. Below this, a sub-navigation bar has 'Post Reports (2)' selected (highlighted in blue) and 'Comment Reports (1)'. A table lists two reported posts:

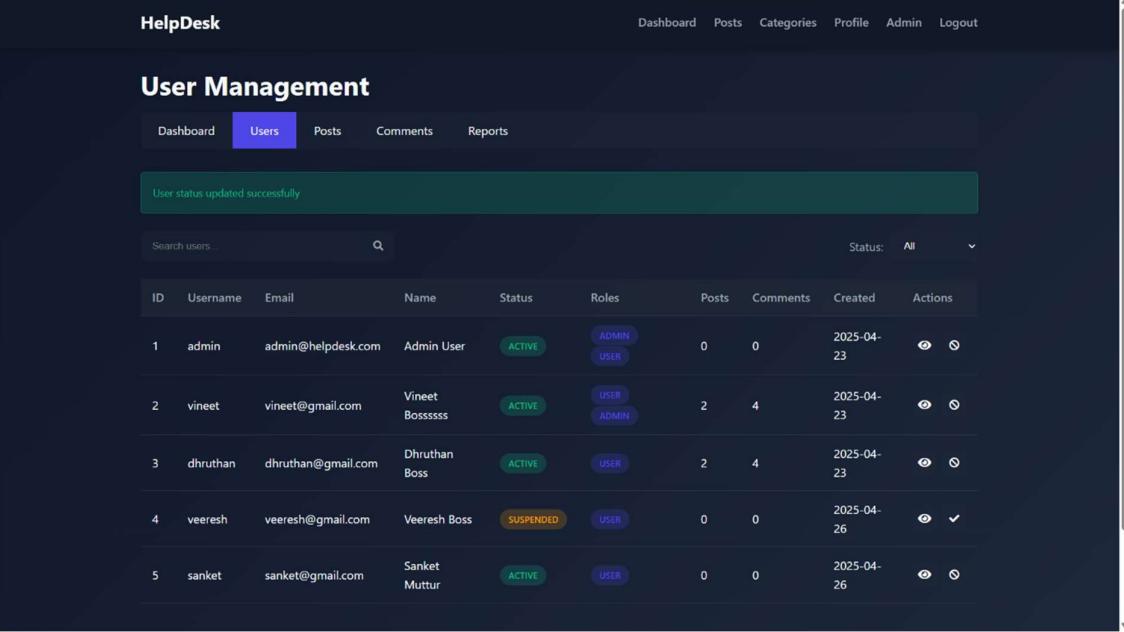
Post ID	Title	Author	Reported By	Reason	Reported At	Actions
3	Cloud Computing: Revolutionizing Scalability and Flexibility	Dhruthan Boss	vineet	Misinformation	2025-04-23 13:54	
4	MVC Architecture in OOAD	Dhruthan Boss	vineet	Misinformation	2025-04-26 17:03	

At the bottom of the page, a copyright notice reads: © 2023 HelpDesk. All rights reserved.

Viewing Individual User Profile (Can see every Post/Comment by the User)

The screenshot shows the 'User Details' section of the HelpDesk application. At the top, there are navigation links: Dashboard, Users, Posts, Comments, and Reports. Below this, a sub-navigation bar has 'Users' selected (highlighted in blue) and 'Posts' and 'Comments' also present. The user profile for 'veeresh' is displayed, including a profile picture, the name 'veeresh', the email 'veeresh@gmail.com', and the title 'Veeresh Boss'. Status indicators show 'ACTIVE' and 'USER'. Account details show 'Account Created: 2025-04-26 16:46' and 'Last Login: Never'. Below these are buttons for 'Suspend User', 'Ban User', and 'Change Role'. A summary at the bottom shows 'Posts (0)' and 'Comments (0)'. A note states: 'This user hasn't created any posts yet.'

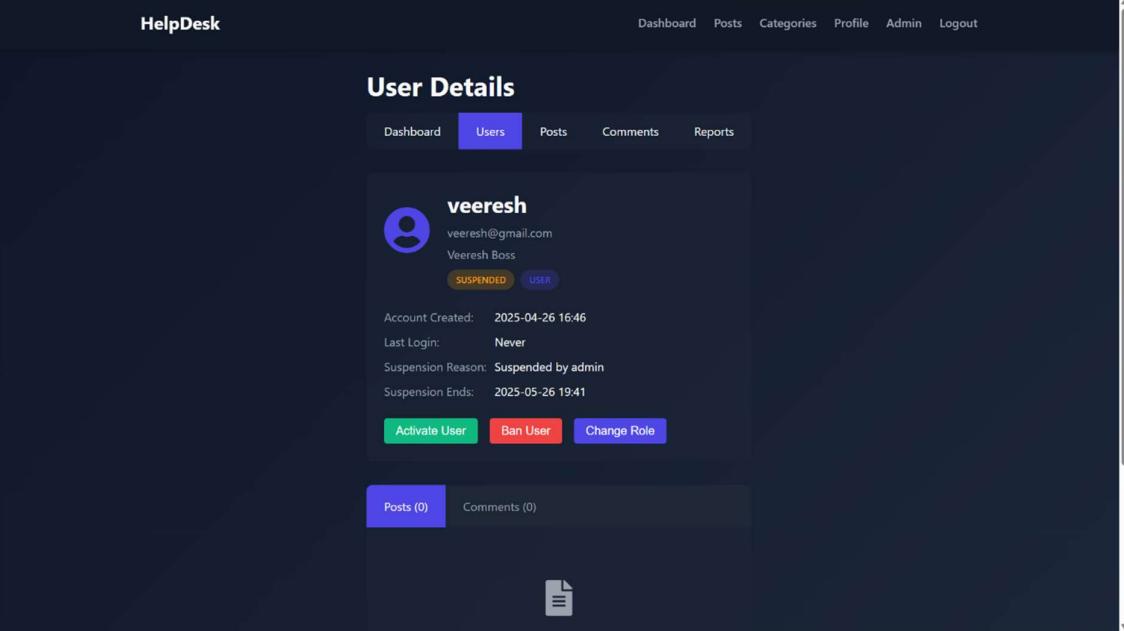
Suspended the User



The screenshot shows the 'User Management' section of the HelpDesk application. At the top, there is a navigation bar with links for Dashboard, Posts, Categories, Profile, Admin, and Logout. Below the navigation is a sub-navigation bar with links for Dashboard, Users (which is highlighted in blue), Posts, Comments, and Reports. A green success message box displays 'User status updated successfully'. The main area features a table with user data. The columns are ID, Username, Email, Name, Status, Roles, Posts, Comments, Created, and Actions. The rows show five users: admin, vineet, dhruthan, veeresh, and sanket. The 'Status' column indicates that 'veeresh' is suspended ('SUSPENDED') while others are active ('ACTIVE'). The 'Actions' column contains edit and delete icons. The table also includes a search bar and a dropdown for filtering by status.

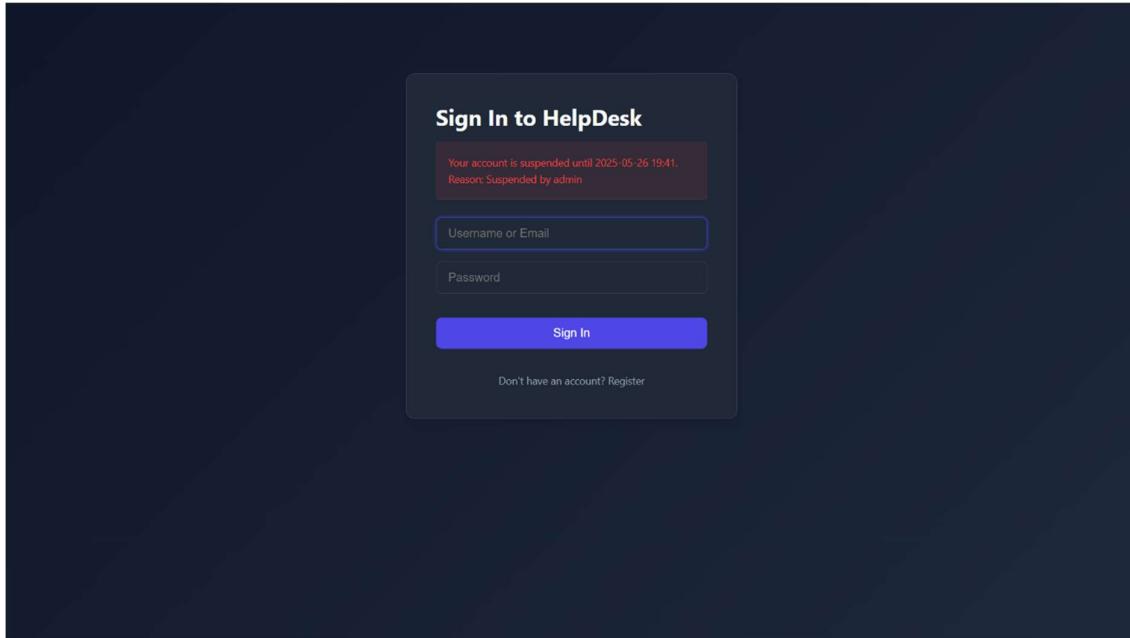
ID	Username	Email	Name	Status	Roles	Posts	Comments	Created	Actions
1	admin	admin@helpdesk.com	Admin User	ACTIVE	ADMIN USER	0	0	2025-04-23	
2	vineet	vineet@gmail.com	Vineet Bossssss	ACTIVE	USER ADMIN	2	4	2025-04-23	
3	dhruthan	dhruthan@gmail.com	Dhruthan Boss	ACTIVE	USER	2	4	2025-04-23	
4	veeresh	veeresh@gmail.com	Veeresh Boss	SUSPENDED	USER	0	0	2025-04-26	
5	sanket	sanket@gmail.com	Sanket Muttur	ACTIVE	USER	0	0	2025-04-26	

Suspended the User



The screenshot shows the 'User Details' page for the user 'veeresh'. At the top, there is a navigation bar with links for Dashboard, Posts, Categories, Profile, Admin, and Logout. Below the navigation is a sub-navigation bar with links for Dashboard, Users (which is highlighted in blue), Posts, Comments, and Reports. The main area displays user details: name 'veeresh', email 'veeresh@gmail.com', and title 'Veeresh Boss'. The 'Status' is listed as 'SUSPENDED' and 'USER'. Below this, account information is provided: Account Created: 2025-04-26 16:46, Last Login: Never, Suspension Reason: Suspended by admin, and Suspension Ends: 2025-05-26 19:41. There are three buttons at the bottom: 'Activate User' (green), 'Ban User' (red), and 'Change Role' (blue). At the bottom of the page, there are links for 'Posts (0)' and 'Comments (0)'.

Suspended User can't login or use the account for one month



Banning the User

A screenshot of a "User Details" page for a user named "Ban User". A modal window titled "Ban User" is open, showing a text area for "Reason for Ban" with the word "Bullying" typed in. Below the modal are buttons for "Ban User" and "Cancel". Underneath the modal, the user's status is listed: "Last Login: Never", "Suspension Reason: Suspended by admin", and "Suspension Ends: 2025-05-26 17:17". At the bottom of the page are buttons for "Activate User" (green), "Ban User" (red), and "Change Role" (purple). Below these buttons are links for "Posts (0)" and "Comments (0)".

User Banned

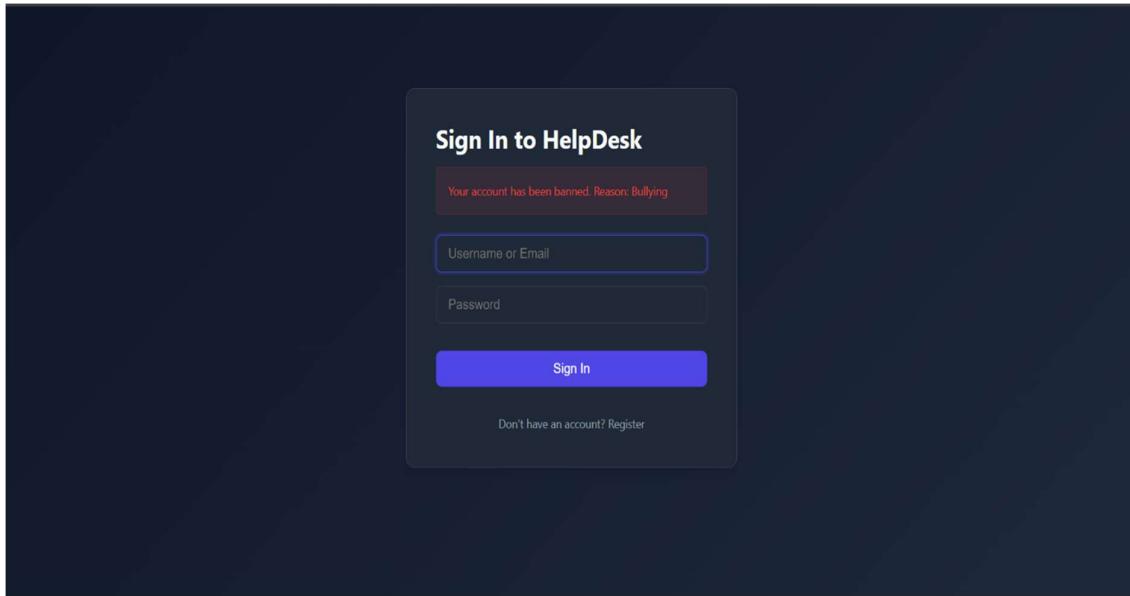
The screenshot shows the 'User Details' page for a user named 'veeresh'. The user's profile picture is a blue placeholder icon. The name 'veeresh' is displayed above the email 'veeresh@gmail.com' and the title 'Veeresh Boss'. Below the profile, there are two status buttons: 'BANNED' (red) and 'USER' (blue). Account details show 'Account Created: 2025-04-26 16:46', 'Last Login: Never', and 'Ban Reason: Bullying'. Two buttons at the bottom are 'Activate User' (green) and 'Change Role' (blue). Below the main card, there are sections for 'Posts (0)' and 'Comments (0)', followed by a note: 'This user hasn't created any posts yet.'

User Banned

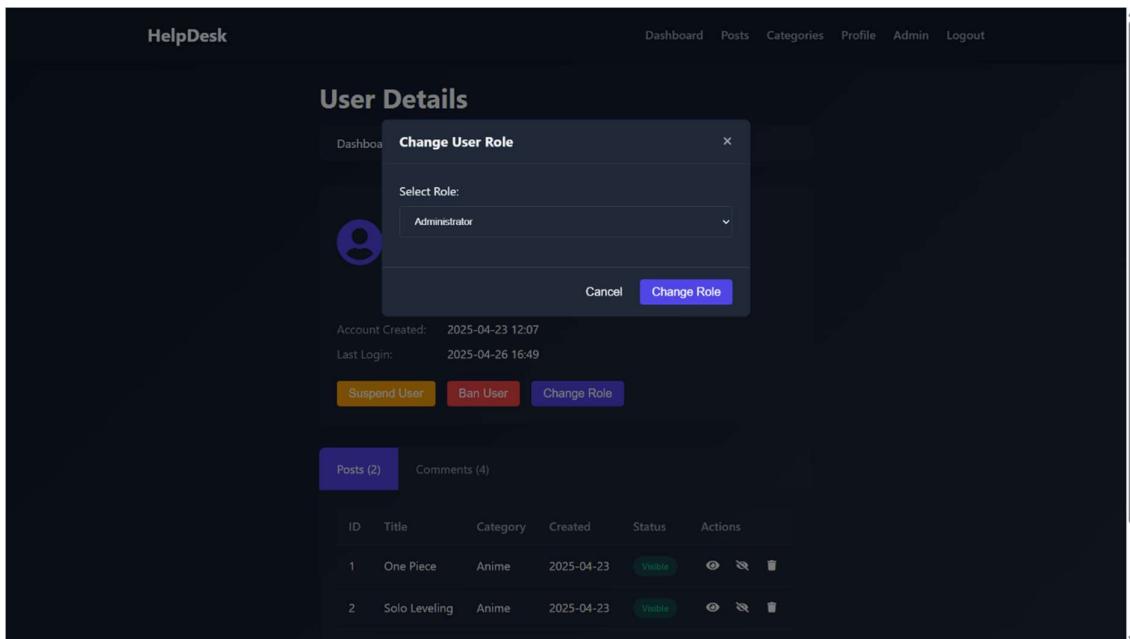
The screenshot shows the 'User Management' page. The 'Users' tab is selected in the navigation bar. A search bar and a status filter 'Status: All' are present. The main table lists five users:

ID	Username	Email	Name	Status	Roles	Posts	Comments	Created	Actions
1	admin	admin@helpdesk.com	Admin User	ACTIVE	ADMIN USER	0	0	2025-04-23	
2	vineet	vineet@gmail.com	Vineet Bossssss	ACTIVE	USER ADMIN	2	4	2025-04-23	
3	dhruthan	dhruthan@gmail.com	Dhruthan Boss	ACTIVE	USER	2	4	2025-04-23	
4	veeresh	veeresh@gmail.com	Veeresh Boss	BANNED	USER	0	0	2025-04-26	✓
5	sanket	sanket@gmail.com	Sanket Muttur	ACTIVE	USER	0	0	2025-04-26	

Banned User can't login or use the account



Changing the Role



HelpDesk

Dashboard Posts Categories Profile Admin Logout

User Details

Change User Role

Select Role:

Administrator

Cancel Change Role

Account Created: 2025-04-23 12:07
Last Login: 2025-04-26 16:49

Suspend User Ban User Change Role

Posts (2) Comments (4)

ID	Title	Category	Created	Status	Actions
1	One Piece	Anime	2025-04-23	Visible	
2	Solo Leveling	Anime	2025-04-23	Visible	

Role changed for the user “vineet” who now has same roles as “admin”

User Management

ID	Username	Email	Name	Status	Roles	Posts	Comments	Created	Actions
1	admin	admin@helpdesk.com	Admin User	ACTIVE	ADMIN USER	0	0	2025-04-23	
2	vineet	vineet@gmail.com	Vineet Bossssss	ACTIVE	USER ADMIN	2	4	2025-04-23	
3	dhruthan	dhruthan@gmail.com	Dhruthan Boss	ACTIVE	USER	2	4	2025-04-23	
4	veeresh	veeresh@gmail.com	Veeresh Boss	ACTIVE	USER	0	0	2025-04-26	
5	sanket	sanket@gmail.com	Sanket Muttur	ACTIVE	USER	0	0	2025-04-26	

Hiding Comment

Comment Management

ID	Content	Author	Post	Created	Status	Votes	Actions
1	Weebs 😊😊	Dhruthan Boss	Post #1	2025-04-23	Visible	0	
2	I don't watch anime 😊	Dhruthan Boss	Post #2	2025-04-23	Visible	0	
3	Nerd 😊😊	Vineet Bossssss	Post #3	2025-04-23	Visible	0	
4	I don't care 😊😊	Vineet Bossssss	Post #4	2025-04-23	Visible	0	
5	Don't want to hear it coming from a weeb 😊😊	Dhruthan Boss	Post #3	2025-04-23	Visible	0	
6	Oh, Vineet, I don't care? Wow, that's the mos...	Dhruthan Boss	Post #4	2025-04-23	Visible	0	
7	Comment Testing	Vineet Bossssss	Post #4	2025-04-26	Hidden	0	
8	Reply to the Comment Testing	Vineet Bossssss	Post #4	2025-04-26	Visible	0	

Hiding Post

The screenshot shows the 'Post Management' section of the HelpDesk application. The 'Posts' tab is selected. A search bar and filters for 'Category' (All Categories) and 'Status' (All) are present. The table lists four posts:

ID	Title	Author	Category	Created	Status	Votes	Comments	Actions	
1	One Piece	Vineet Bossssss	Anime	2025-04-23	Visible	1	1		
2	Solo Leveling	Vineet Bossssss	Anime	2025-04-23	Visible	-1	1		
3	Cloud Computing: Revolutionizing Scalability and Flexibility	Dhruthan Boss	Networking	2025-04-23	Hidden	0	2		
4	MVC Architecture in OOAD	Dhruthan Boss	Software Development	2025-04-23	Visible	0	4		

At the bottom, a copyright notice reads: © 2023 HelpDesk. All rights reserved.

Comment Hiding from Report

The screenshot shows the 'Reports Management' section of the HelpDesk application. The 'Reports' tab is selected. It displays two reports: 'Post Reports (2)' and 'Comment Reports (1)'. The 'Comment Reports (1)' tab is active, showing a single report for a comment:

Comment ID	Content	Author	Reported By	Reason	Reported At	Actions		
6	Oh, Vineet, 'I don't care'? Wow, that's the mos...	Dhruthan Boss	vineet	Harassment	2025-04-26 17:03			

At the bottom, a copyright notice reads: © 2023 HelpDesk. All rights reserved.

Comment Hidden

The screenshot shows the 'Comments' section of the HelpDesk Management System. A success message 'Comment visibility updated successfully' is displayed at the top. Below it is a search bar and a status filter set to 'All'. A table lists eight comments, each with columns for ID, Content, Author, Post, Created, Status, Votes, and Actions. Comment 6 is marked as 'Hidden'.

ID	Content	Author	Post	Created	Status	Votes	Actions
1	Weebs 😊😊	Dhruthan Boss	Post #1	2025-04-23	Visible	0	👁️ 🔍 🗑️
2	I don't watch anime 😊	Dhruthan Boss	Post #2	2025-04-23	Visible	0	👁️ 🔍 🗑️
3	Nerd 😊😊	Vineet Bosssss	Post #3	2025-04-23	Visible	0	👁️ 🔍 🗑️
4	I don't care 😊😊	Vineet Bosssss	Post #4	2025-04-23	Visible	0	👁️ 🔍 🗑️
5	Don't want to hear it coming from a weeb 😊😊	Dhruthan Boss	Post #3	2025-04-23	Visible	0	👁️ 🔍 🗑️
6	Oh, Vineet, I don't care? Wow, that's the mos...	Dhruthan Boss	Post #4	2025-04-23	Hidden	0	👁️ 🔍 🗑️
7	Comment Testing	Vineet Bosssss	Post #4	2025-04-26	Hidden	0	👁️ 🔍 🗑️
8	Reply to the Comment Testing	Vineet Bosssss	Post #4	2025-04-26	Visible	0	👁️ 🔍 🗑️

Individual contributions of the team members:

Name	Module Worked on
Dhruthan M N (PES2UG22CS181)	Major Use case 1
	Minor Use case 1
	Front Controller Pattern
	Model-View-Controller (MVC) Pattern
	Template Method Pattern
	Observer Pattern
Sanket B. Muttur (PES1UG23CS846)	Major Use case 4
	Minor Use case 4
	Repository Pattern
	Dependency Injection Pattern
	Data Transfer Object (DTO) Pattern
	Service Layer Pattern
Veeresh Amaragatti (PES1UG23CS847)	Major Use case 3
	Minor Use case 2
	Factory Method Pattern
	Builder Pattern
	Singleton Pattern
	Composite Pattern
Vineet Goel (PES1UG22CS697)	Major Use case 2
	Minor Use case 3
	Strategy Pattern
	Decorator Pattern
	Proxy Pattern