BLOGGING PLATFORM

CONNECTCO

SOFTWARE ENGINEERING (IT - 314)

Prof. Saurabh Tiwari

Group 23

- 202201287 GHORI ZEEL JIVRAJBHAI **(Group Leader)**
- 202201291 PATEL NAISARGI NILANGBHAI
- 202201272 SIDDHANT GUPTA
- 202201322 ADITYA IYER
- 202201338 KRITARTH JOSHI
- 202201312 DESAI PRITISH
- 202201299 AKSHAT JINDAL
- 202201324 KASHYAP GAJERA
- 202201267 CHAUHAN HEMAN JASHVANTBHAI
- 202201265 CHAUDHARI PRACHI HASMUKHBHAI

SYSTEM DESIGN

System Design Approach

We follow a **Top-Down Design Approach**, as recommended in the uploaded guidelines:

1. **High-level functionalities** are defined first, with a focus on modular decomposition.
2. Subsystems are progressively refined into modules.
3. Ensure **modularity** (independent functionality) and **scalability** (easy horizontal/vertical scaling).
4. Design goals prioritize **usability**, **efficiency**, **robustness**, and **reusability**.

Design Goals

1. High maintainability and flexibility for evolving requirements.
2. Scalability to support increased user base and content volume.
3. Secure and robust handling of user authentication and data.
4. Optimized performance with caching and indexing.
5. User-friendly interfaces for students and admins.
6. Accessibility across devices (web).

Interface Design (Black Box View)

Objective

Understand how the system interacts with the environment by treating the system as a black box, focusing on input and output.

External Interfaces

- **User Interface**: Website
- **Admin Interface**: Dashboard for administrative operations.

- **Email/Notification System**: For alerts and updates.
- **Content Delivery Network (CDN)**: Efficient content delivery.
- **Authentication System**: Secure login and registration.
- **Storage System**: Data storage and media file handling.

Inputs

- User credentials.
- Blog content.
- Media files.
- Forum discussions.
- Feedback submissions.

Outputs

- Authentication responses.
- Blog posts.
- Notifications.
- Analytics data.
- Announcements.

Key Relationships

- **Users** ↔ **Platform**: Users submit requests (e.g., login, create blogs, provide feedback) and receive responses.
- **Admins** ↔ **Platform**: Admins manage notices, handle reports, and moderate forums.

Subsystem Decomposition

Principles

Subsystems are identified based on **high cohesion** and **low coupling**, following partitioning and layering guidelines.

Subsystems and Their Interfaces

1. **Authentication Subsystem**:
    a. Manages user login, registration, and password resets.
    b. Ensures secure access using college-specific emails.
    c. Provides APIs for user token generation and verification.
2. **User Management Subsystem**:
    a. Manages user profiles and preferences.
    b. Interfaces with storage for profile pictures and data persistence.
3. **Blog Management Subsystem**:
    a. Supports blog creation, editing, deletion, and media embedding.
    b. Provides APIs for managing drafts, publishing posts, and privacy settings.
4. **Search and Interaction Subsystem**:
    a. Enables keyword-based searches using Elasticsearch.
    b. Handles user interactions such as likes, comments, and shares.
5. **Collaboration and Forums Subsystem**:
    a. Provides forums for clubs and committees.
    b. Enables collaborative blogging and poll creation.
6. **Analytics Subsystem**:
    a. Tracks user engagement metrics such as views, comments, and likes.
    b. Generates reports for admin analysis.
7. **Notification Subsystem**:
    a. Sends push notifications and emails for updates and alerts.
    b. Provides APIs for triggering notifications across services.
8. **Admin Panel Subsystem**:
    a. Offers tools for moderation, handling complaints, and managing announcements.

Relationships Between Subsystems

- Subsystems interact via **defined interfaces**:
    - Data Coupling: Shared resources (e.g., user data, blog metadata).
    - Message Coupling: Asynchronous communication (e.g., event-driven notifications).
- The architecture uses **closed layering** (modules only communicate with adjacent layers).

Architectural Design

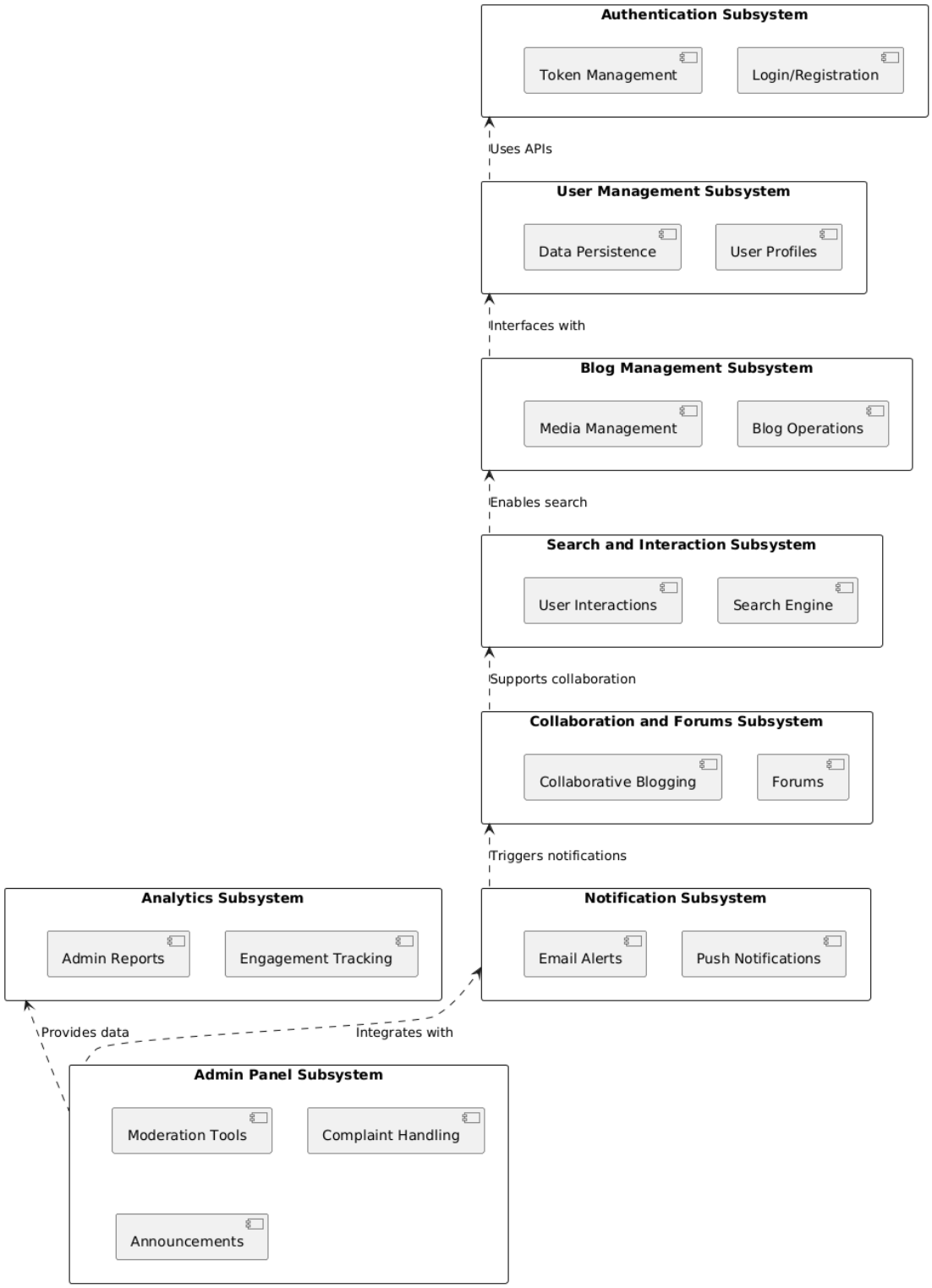
System Layers

We adopt a **three-layer architecture**:

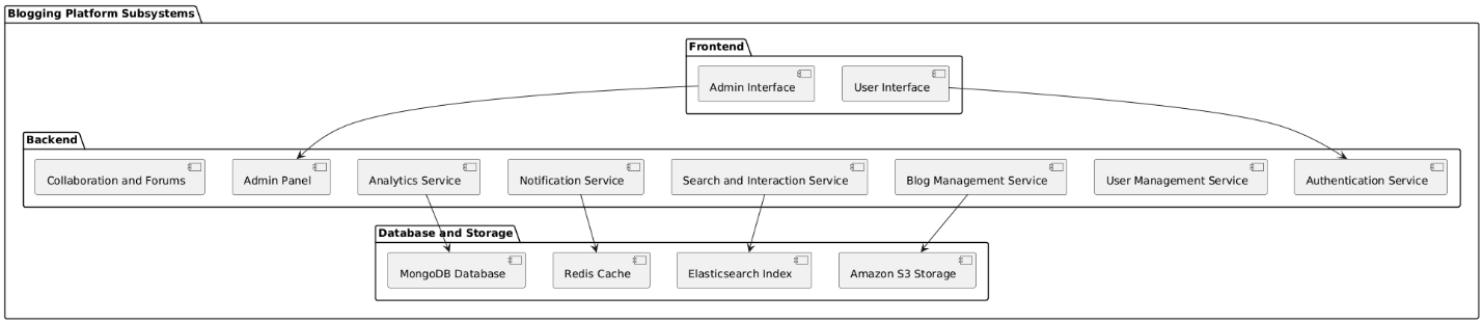
1. **Presentation Layer (Frontend)**:
    a. Web Client: React.js.
    b. Admin Dashboard: A dedicated web interface for admins.
2. **Application Layer (Backend)**:
    a. Services implemented in Node.js/Express.
    b. Modular services for authentication, blogs, search, analytics, notifications, and admin tools.
    c. Authentication Service : Login/Registration, Password Management, Email Verification.
    d. Blog Service : Post Management, Media Handling, Privacy Controls.
    e. Search Service : Content Indexing, Real-time Search, Filters.
    f. Collaboration Service : Co-authoring, Forums and Comments.
    g. Notification Service : Real-time Updates, Email Notifications, Notice Board Integration.
    h. Analytics Service : Engagement Metrics, User Statistics, Performance Monitoring.

3. **Data Layer (Database)**:
    a. **Primary Database**: MongoDB for structured data.
    b. **Cache**: Redis for frequently accessed data.
    c. **Search Index**: Elasticsearch for full-text search.
    d. **File Storage**: Amazon S3 for media.

Module Relation Diagram



Subsystem decomposition diagram:



System context diagram: