**Whistle Social Website Documentation**

**SDD**

* **Micro-services Overview**:

The application is built using a microservices architecture, where different components of the system are developed and deployed independently as separate services.

Each micro-service is responsible for a specific domain or functionality within the application, allowing for better scalability, maintainability, and flexibility.

* **Services:**

**User Authentication Service**: Handles user authentication and authorization, including login, registration, password management, and access control.

**Post Service**: Manages the creation, retrieval, updating, and deletion of posts made by users. Handles functionalities related to posts, such as viewing posts, editing posts, and deleting posts.

**Comment Service**: Responsible for managing comments on posts, including adding comments, retrieving comments and deleting comments.

**Like Service**: Manages the liking functionality for posts, allowing users to like or unlike posts. Tracks the number of likes for each post.

**Friends Service**: Handles friend-related functionalities, such as sending friend requests, approving and disapproving friend requests, removing friends, viewing friends list, viewing requests list and suggesting friends.

**Bookmark Service**: Manages bookmarking functionality, allowing users to bookmark posts. Provides functionality for adding bookmarks, retrieving bookmarked posts, and removing bookmarks.

**Search Service**: Provides simple search functionality across user entity by first and last names.

**Notifications Service**: Notification generation is done by other services and delivery to this service. Sends notifications for events such as new friend requests, new comments on posts, likes on posts.

* **Communication Between Services**:

Microservices communicate with each other via well-defined APIs using standard protocols such as HTTP/REST or message queues.

Services interact to achieve loose coupling and scalability. For example, when a user adds a comment to a post, the Comment Service publishes a message indicating the new comment, which is then consumed by the Post Service to update the post's comment count.

* **Data Management**:

Each microservice manages its own data store.

Data transaction is between microservices through well-defined REST APIs.

* **Deployment**:

Microservices are deployed independently using containerization technology which is Docker.

**Let's break down each microservice in your application and outline its responsibilities:**

**1. User Authentication Service:**

* **Responsibilities:**

Manages user authentication and authorization processes.

Handles user registration, login, and logout functionalities.

Enforces access control policies and permissions.

* **Description:**

This microservice is responsible for managing user identities and ensuring secure access to the application.

Implements security measures such as encryption, session management, and token-based authentication to protect user accounts and sensitive data.

* **Interfaces:**

Provides RESTful APIs for user registration, login, logout, and token management.

Integrates with external identity providers (e.g., OAuth providers) for social login and federated authentication.

**2. Post Service:**

* **Responsibilities:**

Manages the lifecycle of posts created by users.

Allows users to create, retrieve, update, and delete posts.

* **Description:**

This microservice facilitates the creation and management of user-generated content in the form of posts.

It provides endpoints for creating new posts, fetching posts by ID or user, updating post content, and deleting posts.

Manages relationships between posts and other entities such as users, comments, likes, notifications and bookmarks.

* **Interfaces:**

Exposes RESTful APIs for CRUD operations on posts.

Integrates with other microservices for handling related actions such as commenting, liking, bookmarking posts and sending notifications for each of the last functions.

**3. Comment Service:**

* **Responsibilities:**

Manages comments associated with posts.

Allows users to add, retrieve and delete comments.

Handles notifications for new comments on posts.

* **Description:**

This microservice focuses on managing the interaction and engagement around posts through comments.

It provides endpoints for adding comments to posts, fetching comments by post ID and deleting comments.

Sends notifications to users when new comments are added to their posts.

* **Interfaces:**

Exposes RESTful APIs for CRUD operations on comments.

Integrates with the notification service to send comment-related notifications to users.

**4. Like Service:**

* **Responsibilities:**

Manages the liking functionality for posts.

Allows users to like or unlike posts.

Tracks the number of likes for each post.

* **Description:**

This microservice handles the interaction of users with posts through likes.

It provides endpoints for liking and unliking posts, as well as retrieving the number of likes for a given post.

Sends notifications to users when new likes are added to their posts.

* **Interfaces:**

Exposes RESTful APIs for liking and unliking posts, as well as retrieving like counts.

**5. Friends Service:**

* **Responsibilities:**

Manages friend relationships between users.

Allows users to send, accept, reject friend requests and remove accepted friends.

Provides functionality for viewing friends lists and suggesting new friends.

* **Description:**

This microservice facilitates social interactions between users by managing friend connections.

It provides endpoints for sending friend requests, accepting or rejecting friend requests, and retrieving friends lists.

Manages the state of friend relationships and sends notifications to users for friend-related activities.

* **Interfaces:**

Exposes RESTful APIs for managing friend requests, connections, and lists.

Integrates with the notification service to send friend request notifications.

**6. Bookmark Service:**

* **Responsibilities:**

Manages bookmarking functionality for posts.

Allows users to bookmark posts for later reference.

Provides functionality for adding, retrieving, and removing bookmarks.

* **Description:**

This microservice enables users to save posts for later viewing or reference.

It provides endpoints for adding bookmarks to posts, retrieving bookmarked posts for a user, and removing bookmarks.

* **Interfaces:**

Exposes RESTful APIs for managing bookmarks on posts.

Integrates with the post service to retrieve post content and metadata.

**7. Search Service:**

* **Responsibilities:**

Facilitates searching and querying of content across the application.

Allows users to search for users.

* **Description:**

This microservice enables users to discover content within the application efficiently.

It indexes and catalogs data from various sources, including users.

* **Interfaces:**

Exposes RESTful APIs for executing search queries and retrieving search results.

Integrates with other microservices to fetch additional details and context for search results.

**8. Notifications Service:**

* **Responsibilities:**

Manages the generation and delivery of notifications to users.

Sends notifications for various events such as new friend requests, comments on posts, likes on posts, etc.

* **Description:**

This microservice handles the delivery of real-time notifications to users, keeping them informed about relevant activities within the application.

It listens for events triggered by other microservices (e.g., new comments, likes, friend requests) and generates corresponding notifications.

* **Interfaces:**

Exposes APIs for subscribing to specific notification types and managing notification preferences.

Integrates with event-driven architectures and message brokers to receive and process notification events.