1. Overview

The Social Media Aggregator is composed of several key components:

1. **User Management**
2. **Influencer Management**
3. **Social Media Feed Aggregation**
4. **Data Storage (In-memory Database using H2)**
5. **External Social Media API Integration**

2. Classes and Components

2.1 User Management

Class: User

- **Attributes**:
  - Long id: Unique identifier for the user.
  - String username: Username of the user.
  - String password: User's password.
  - List<Influencer> influencers: List of influencers followed by the user.
- **Methods**:
  - getters and setters for each field

Class: UserService

- **Methods**:
  - registerUser(UserCreationDTO userCreationDTO): Registers a new user.
  - findByUsername(String username): Retrieves a user by their username.
  - followInfluencers(String username, List<String> uniqueHashes): Allows a user to follow multiple influencers.

Class: UserRepository

- **Methods**:
  - save(User user): Saves a user to the database.
  - findByUsername(String username): Finds a user by their username.

2.2 Influencer Management

Class: Influencer

- **Attributes**:

    - Long id: Unique identifier for the influencer.
    - String name: Name of the influencer.
    - String uniquehash: Unique hash representing the influencer.
    - List<SocialMediaProfile> profiles: List of social media profiles associated with the influencer.
    - List<User> followers: List of users following this influencer.
- **Methods**:
    - getters and setters for each field

Class: InfluencerService

- **Methods**:
    - addInfluencer(InfluencerCreationDTO influencerDTO): Adds a new influencer with their social media profiles.
    - getAllInfluencers(): Retrieves all influencers.

Class: InfluencerRepository

- **Methods**:
    - save(Influencer influencer): Saves an influencer to the database.
    - findByUniquehash(String uniquehash): Finds an influencer by their unique hash.
    - findByFollowers_Id(Long userId): Finds influencers followed by a user.

2.3 Social Media Profile Management

Class: SocialMediaProfile

- **Attributes**:
    - Long id: Unique identifier for the profile.
    - String platform: The platform name (e.g., Twitter, Facebook).
    - String profileUrl: URL to the profile.
    - Influencer influencer: The influencer to whom this profile belongs.
    - List<SocialMediaFeed> feeds: List of social media feeds related to this profile.
- **Methods**:
    - getters and setters for each field

Class: SocialMediaProfileRepository

- **Methods**:
    - findByPlatformAndUsername(String platform, String username): Retrieves profiles by platform and username.

2.4 Social Media Feed Aggregation

Class: SocialMediaFeed

- **Attributes**:
    - Long id: Unique identifier for the feed.
    - String content: The content of the feed.
    - String mediaUrl: Media URL (if any).
    - LocalDateTime timestamp: Timestamp of the feed.
    - int likes: Number of likes.
    - int shares: Number of shares.
    - int comments: Number of comments.
    - SocialMediaProfile socialMediaProfile: The associated social media profile.
- **Methods**:
    - getters and setters for each field

Class: FeedAggregationService

- **Methods**:
    - aggregateFeeds(String username, List<String> platforms): Aggregates feeds based on the platforms and the influencers a user follows.
    - getAllFeeds(): Retrieves all feeds from the database.
    - getLatestFeeds(List<String> platforms): Retrieves the latest feeds for the specified platforms.

Class: SocialMediaFeedRepository

- **Methods**:
    - save(SocialMediaFeed feed): Saves a feed to the database.
    - findBySocialMediaProfile(SocialMediaProfile profile): Finds feeds by social media profile.
    - findTopBySocialMediaProfileOrderByTimestampDesc(SocialMediaProfile profile): Finds the latest feed by social media profile.

2.5 External Social Media API Integration

This we haven't implemented. This can be the future scope to get the feeds at runtime. The Current requirement was to return mock data but anyways I have added the sample classes for the same.

Class: TwitterClient

- **Methods**:
    - getLatestTweet(String username): Fetches the latest tweet of the user.

Class: InstagramClient

- **Methods**:

    - getLatestInstagramPost(String userId): Fetches the latest Instagram post of the user.

Class: FacebookClient

- **Methods**:
    - getLatestFacebookPost(String pageId): Fetches the latest Facebook post of the page.

3. Database Design

- **User Table (users)**: Contains user information.
- **Influencer Table (influencers)**: Contains influencer information.
- **SocialMediaProfile Table (social_media_profiles)**: Contains social media profile information linked to influencers.
- **SocialMediaFeed Table (social_media_feed)**: Contains social media feeds linked to profiles.
- **UserInfluencer Table (user_influencer)**: Manages the many-to-many relationship between users and influencers.

4. Sequence Flows

4.1 User Registration Flow

1. The user sends a request to the UserController to register.
2. The UserController calls UserService.registerUser().
3. UserService creates a new User entity and saves it using UserRepository.
4. The new user is stored in the users table in the database.

4.2 Follow Influencers Flow

1. The user sends a request to the UserController to follow multiple influencers.
2. The UserController calls UserService.followInfluencers().
3. UserService retrieves the User and Influencer entities.
4. It creates a UserInfluencer entry and saves it using the UserInfluencerRepository.
5. The many-to-many relationship is updated in the user_influencer table.

4.3 Feed Aggregation Flow

1. The user requests aggregated feeds from FeedAggregationController.
2. FeedAggregationController calls FeedAggregationService.aggregateFeeds().
3. FeedAggregationService retrieves the User and Influencer entities.
4. It fetches the feeds using SocialMediaFeedRepository.

5. The feeds are returned and displayed to the user.

5. Future Scope

- **Real-Time Feed Aggregation**: Implement WebSocket or a similar technology for real-time updates.
- **User Authentication**: Implement OAuth2 or JWT-based authentication.
- **Scalability**: Migrate the in-memory database to a distributed database like PostgreSQL or MongoDB.
- **Additional Platforms**: Integrate more social media platforms like LinkedIn, YouTube, etc.
- **Front-End Development**: Develop a front-end UI using Angular or React for better user interaction.

This LLD provides a detailed breakdown of the application's components, their interactions, and the underlying database structure. You can visualize this using diagrams to represent the relationships and sequence flows more effectively