Bounded context identification and defining of microservices.

1. UserService Microservice
Bounded Context: User Management
Responsibilities:
• User Registration: Allows users to register.
• User Authentication: Supports login and identity verification.
• Profile Management: Manages user details such as UserId, FullName, and Email.
• User Information Retrieval: Provides endpoints to retrieve user information, including fetching UserId based on Username.
• Database: Uses a UserDbContext backed by SQLite to store user data.
Interactions:
• Exposes an API that PostService can call to validate users by UserId or retrieve user details.

2. PostService Microservice
Bounded Context: Post Management
Responsibilities:
• Post Creation: Allows users to create posts with attributes like Content and UserId.
• Post Retrieval: Provides access to a list of posts or a specific post by PostId.
• Post Deletion: Allows users to delete posts.
• Database: Uses a PostDbContext to store post data, with UserId as a foreign key to associate posts with users.
Interactions:
• Calls the UserService to validate user existence and retrieve UserId based on Username when creating posts.

3. APIGateway Microservice
Bounded Context: API Gateway
Responsibilities:
• Request Routing: Routes incoming client requests to appropriate backend services (e.g., UserService, PostService).
• External Interfaces: Exposes a unified API to external clients and handles routing to internal services.
Interactions:
• Exposes an API that the client consumes.
• Routes requests to UserService for user-related actions (e.g., registration, user details).
• Routes requests to PostService for post creation, retrieval, and management.

4. Messaging and Event System Microservice
Bounded Context: Messaging/Events
Responsibilities:
• Event Publishing: Publishes events (e.g., UserCreatedMessage, PostCreatedMessage) to notify other services of significant business events.
• Message Queuing: Manages message queues (e.g., RabbitMQ) to enable asynchronous communication between services.
• Event Subscription: Allows services (e.g., PostService) to subscribe to relevant events for reacting to changes in other services.
• Event Handling: Listens for events, triggers relevant actions, and ensures eventual consistency across microservices.
Interactions:
• UserService publishes events like UserCreatedMessage when a new user is registered. These events can be consumed by PostService or other services to perform actions related to the user creation.

• PostService subscribes to events from UserService, such as when a new user is registered, to handle post-creation actions or updates based on the user state.