Explanatory Notes for Task 0: Sequence Diagram

1. Presentation Layer

- **Responsibilities**: This layer is responsible for handling the interaction between the user and the system, including any UI components or API services. It collects user inputs, sends requests to the Business Logic Layer, and displays outputs back to the user. This layer contains the `ServiceAPI` and `WebService`, which act as entry points for external communication.
- **Facade Pattern**: The `Facade Pattern` simplifies the interaction between this layer and the Business Logic Layer by providing a unified interface (`ServiceAPI`). This pattern abstracts the complexity of the underlying systems, so the Presentation Layer does not need to understand the detailed workings of the Business Logic.

2. Business Logic Layer

- **Responsibilities**: The Business Logic Layer encapsulates the core functionality of the application. It defines how the `User`, `Place`, `Review`, and `Amenity` entities behave and interact. This layer processes user inputs from the Presentation Layer, applies the necessary business rules, and communicates with the Persistence Layer to retrieve or update data.
- **Facade Pattern**: The Business Logic Layer interacts with the Persistence Layer by abstracting database operations. Using the facade pattern, the Presentation Layer can call business logic methods without knowing the details of database operations, allowing for flexibility and maintainability.

3. Persistence Layer

- **Responsibilities**: This layer manages all interactions with the database. It contains components like the `DatabaseManager` and handles storage, retrieval, and updates of data. It provides a structured way to manage queries and ensure data consistency. The Persistence Layer is responsible for the storage of all entities like `User`, `Place`, `Review`, and `Amenity`.
- **Facade Pattern**: The Persistence Layer is hidden behind the facade provided by the Business Logic Layer. This ensures that the higher layers are unaware of how data is persisted, reducing coupling and simplifying communication.

How the Facade Pattern Facilitates Communication

The Facade Pattern is used in the diagram to simplify the interaction between the Presentation Layer and the Business Logic Layer. It creates a single interface (the `ServiceAPI`) through which all interactions occur. This reduces complexity, as the Presentation Layer does not need to handle the intricate details of the underlying logic. Similarly, in the Business Logic Layer, the complexity of database operations is hidden from other layers by the persistence facade. This pattern promotes loose coupling and helps maintain separation of concerns between layers, making the system easier to maintain and extend.

By using the facade pattern, the system can evolve without drastically affecting the communication between layers. For example, changes in the database structure or business rules would not require changes in the Presentation Layer, ensuring greater flexibility.