UML DIAGRAM (USER CASE)

Explanation of UML Diagram: Music Recommendation System using Sentiment Analysis

1. Overview

This UML **Class Diagram** represents the architecture of a **Music Recommendation System using Sentiment Analysis**. It showcases the key components, their attributes, and methods, illustrating how they interact to provide personalized song recommendations based on user sentiment.

---

2. Classes and Their Responsibilities

1. User Class

- Represents a user interacting with the system.

- Attributes:

  - `+userID: int` → Unique ID for each user.

  - `+username: string` → Name of the user.

  - `+email: string` → Email for authentication.

- Methods:

  - `+getSentiment(text: string): string` → Takes user input (text) and returns sentiment (positive, negative, neutral).

---

2. SentimentAnalyzer Class

- Responsible for processing user input and determining sentiment.

- Attributes:

  - `-model: AI_Model` → The machine learning model used for sentiment analysis.

- Methods:

  - `+analyzeSentiment(text: string): string` → Analyzes user input and classifies it as positive, negative, or neutral.

---

3. MusicRecommendationEngine Class

UML DIAGRAM (USER CASE)

- Core logic for recommending music based on sentiment.

- Attributes:

  - `-musicDatabase: List<Song>` → Stores song data for recommendations.

- Methods:

  - `+getRecommendedSongs(sentiment: string): List<Song>` → Fetches songs based on sentiment type.

---

4. Song Class

- Represents individual songs in the database.

- Attributes:

  - `+songID: int` → Unique identifier for each song.

  - `+title: string` → Song title.

  - `+artist: string` → Artist name.

  - `+genre: string` → Genre classification.

---

5. Database Class

- Manages interactions with the music database.

- Attributes:

  - `-connection: DB_Connection` → Establishes and maintains database connections.

- Methods:

  - `+fetchSongs(): List<Song>` → Retrieves all available songs for recommendation.

---

3. Class Relationships

1. **User → SentimentAnalyzer**: The `User` provides input text, which is analyzed by the `SentimentAnalyzer` to extract sentiment.

2. **SentimentAnalyzer → MusicRecommendationEngine**: The extracted sentiment is passed to `MusicRecommendationEngine`, which finds suitable songs.

UML DIAGRAM (USER CASE)

3. **MusicRecommendationEngine → Database**: The recommendation engine fetches songs from the `Database`.

4. **Database → Song**: The `Database` maintains a collection of `Song` objects.

---

4. How the System Works

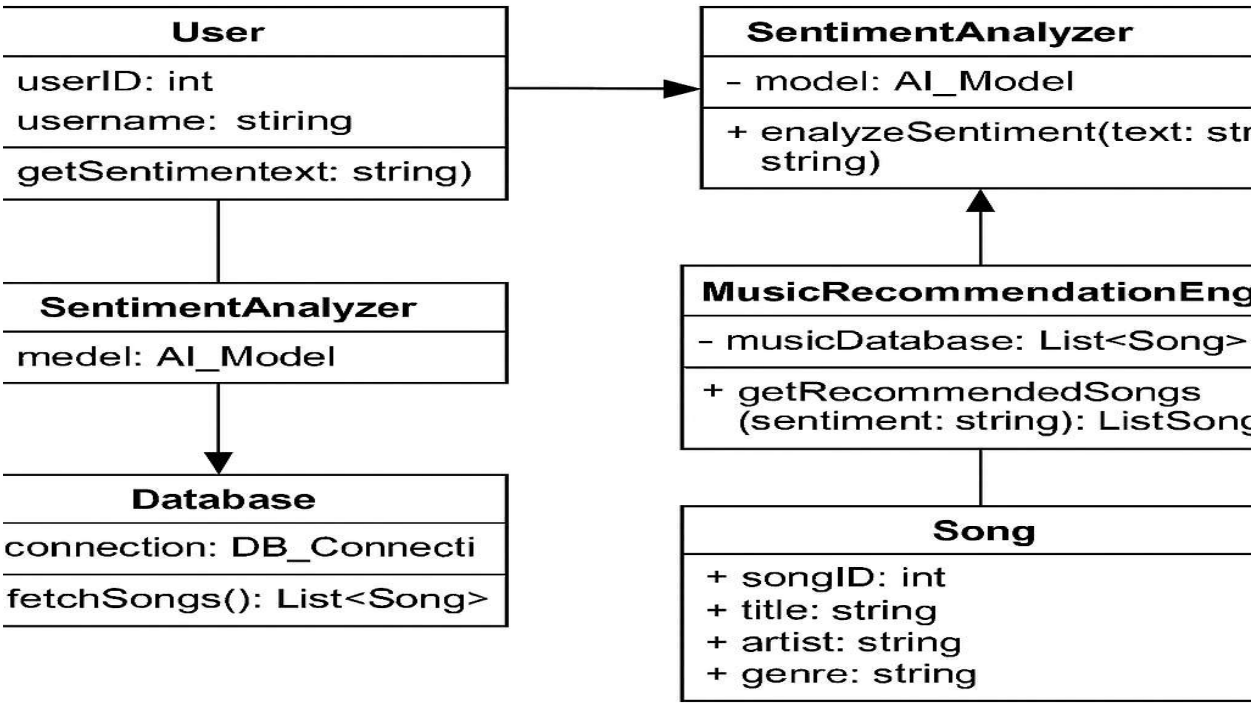
1. A **User** enters a text message (e.g., "I'm feeling happy today!").

2. The **SentimentAnalyzer** processes the text and determines the sentiment (e.g., "Positive").

3. The **MusicRecommendationEngine** fetches songs that match the detected sentiment.

4. The system queries the **Database** for songs of a similar mood.

5. The system returns a playlist of **recommended songs** to the user.



UML DIAGRAM (USER CASE)