User Table

· **Class Naming:** Changed users to User as it's a Java convention to use singular, PascalCase names for classes.

· **Explicit Table Name:** Added @Table(name = "users") to ensure the table name matches your database schema, even if Hibernate defaults change in the future.

· **Field Type:** Changed id to Long for compatibility with databases where primary keys are often BIGINT.

· **Column Constraints:**

* Added @Column(nullable = false) for username, password, email, and role to ensure they cannot be null.
* Added unique = true for username and email to enforce uniqueness at the database level.

· **Lombok Annotations:**

* Added @NoArgsConstructor for creating instances without parameters.

· **~~Validation (Optional):~~** ~~Consider using the javax.validation package for additional field validation, like @Email for email or @Size for password~~

Task Table

· **Class Name and Table Name:** The class name was kept as Task, and the table is explicitly named tasks using the @Table(name = "tasks") annotation for consistency.

· **Field Type:**

* Changed id to Long for better compatibility with databases where the primary key is usually BIGINT.
* Changed DueDate to LocalDate for better date handling rather than String (using LocalDate for date fields is the recommended approach in modern Java).

· **Column Annotations:**

* Made the fields title, description, status, priority, category, and dueDate non-nullable with @Column(nullable = false).
* Renamed DueDate to due\_date to follow common database naming conventions.

· **Foreign Key Relationship:**

* Used @ManyToOne with fetch = FetchType.LAZY for the userId relationship, which allows efficient loading of related data when needed.
* @JoinColumn(name = "user\_id", nullable = false) links this table to the User table, ensuring the foreign key relationship is established properly.

Auditlog

· **Field Type:**

* Changed id to Long for better compatibility with databases where the primary key is usually BIGINT.
* Changed userID to a User entity reference (ManyToOne relationship) rather than just an integer, which provides better object-relational mapping and avoids having a raw foreign key.

· **Timestamp Handling:**

* Added a timestamp field of type LocalDateTime to store the time when the action occurred. This is a better approach than storing the date as a String.

· **Foreign Key Relationship:**

* Used @ManyToOne(fetch = FetchType.LAZY) to map the relationship with the User entity. This will lazily load the user associated with the audit log entry when needed.
* @JoinColumn(name = "user\_id", nullable = false) to specify that the foreign key column will be user\_id, ensuring referential integrity.

@Table

The @Table(name = "audit\_log") annotation specifies the name of the database table that this entity will be mapped to. By default, the table name will be derived from the class name, which in this case would be AuditLog. However, it's often good practice to follow specific naming conventions or ensure consistency between class names and table names in your database schema.

\* Spring take care of parseing client data coming as json