1. Our application's UI Controller (activity/fragment) component would be the MainActivity class. The LiveData ViewModel component would be the HeartRateViewModel class, or MyViewModel if looking at what’s on GitHub. The Repository component would be the HeartRateRepository class. Finally, the Heartrate, HeartRateDao, and HeartRate database classes make up the Room Database component.
2. The annotations that we input into our Heartrate class are the @Entity and @PrimaryKey annotations, which are the same keys in the user class in the video. Our code had the primary key automatically generated whereas the video did not. After importing the proper libraries, these annotations represent entity tables in the SQL database.
3. The UserDao class and the HeartrateDao class are present to define the queries that the app needs to make on the database. The UserDao included a SELECT query that searches for a user by their name. The UserDao class also had an INSERT and INSERT command. These either add or remove specified entities to/from the existing table. Our HeartrateDao class did not have a parameter for the SELECT query, but it returned all results from the table. Our Heartrate Dao also had INSERT and DELETE commands like the UserDao class.
4. Entity annotations are used to define the objects in the relational database modeled by the Object-Relational Mapping (ORM) framework. The specific annotations relate to the specific SQL objects to represent the class. The required annotations would be the @Entity annotation and the @PrimaryKey annotation. Annotations that we’ve used that were not in the video were the @NonNull annotation and the @Ignore annotation to give further metadata about the entity table.
5. The UserDao class had an insert, update, delete, query, and raw query annotation. This differs from the HeartrateDao class as we only implemented an insert and delete annotation. Errors are found in the SQL commands in the Dao class. If there was an error in the SQL format the compiler would give an error when the program tries to run.
6. The Room database class combines the Entity and the Dao for the database for full utilization of the database. The class must be declared abstract and extend the RoomDatabase class. Additionally, this class must include the @Database annotation at the start to specify the components it is putting together for the relational database. The HeartRateDatabase code included the getDatabase() method and the DatabaseWriteExecutor which is not included in the video.
7. Our HeartRateDatabase class contains a similar code to the example from the SingletonClass. However, in the SingletonClass, the new instance of the database is instantiated from the program call. The HeartRateDatabase class uses a database builder that is lifecycle aware and wont lose data due to deletion/creation of a new instance.
8. I made 3 prompts to ChatGPT which resulted in code that would be submissible for this assignment.
   1. For Android, create a database to store a person's heart rate, including pulse and age.
   2. rewrite the code in Java instead of Kotlin
   3. expand the HeartRateDao class to incorporate a Delete Query

The only real differences between ChatGPT’s code and the code that I wrote were names for the attributes and private attributes vs. public attributes. I don’t think that these differences are mistakes, more just different naming conventions and different organizational methods.