System Design, Planning & Assumptions

Objects:

User - has: name, password, catalogue
- Assumption: only 1 catalogue per user

Song - has: song name, artist name, album name, release year, song duration, genre, details, catalogue
Catalogue - has: song(s)

^ All must have unique (auto-generated) ID field!

Artist & Album - potentially just a variable for scope of this task but better as their own objects for future

Methods:
- Add Song to Catalogue
- Fetch Catalogue from User (results should show in a table)
- Filter Song(s) in Catalogue (by release year & song name)
  - Assumption: filter means retrieve songs with matching filter value NOT sort or order by
- See more details of Song (click on song and see details in **new** page/tab)
- Extras: likely that Users also need to be able to Register, Sign in and Sign out

*Production-ready means:
- An implemented architecture
  - I opted for Spring MVC
- Tracking for analytics & usage (i.e. monitoring)
- Crash & stability reporting (i.e. logs)
  - Spring Actuator dependency allows for easy monitoring and reporting of web vitals. Have also added logging and error reporting.
- App needs to be released to production
  - In my case, Spring MVC comes with Tomcat embedded server so the web app can be deployed easily using the .jar file locally (can also use .war file for external servers)
- Unit tests for business logic and critical use cases at least
- Sufficient documentation and commenting where necessary