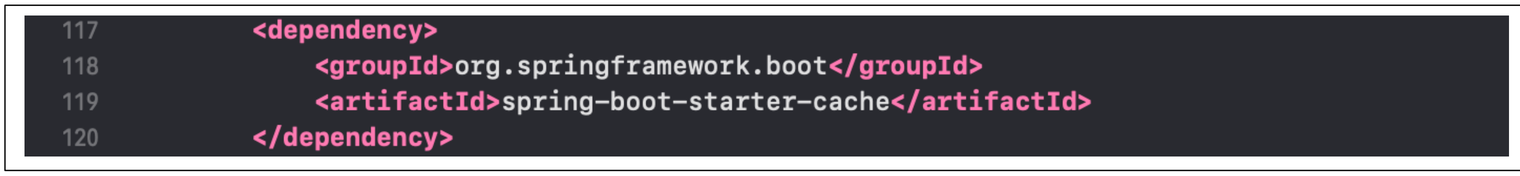
Development Prep for Backend Server / MVP for Disliked Songs List

Building Backend Server

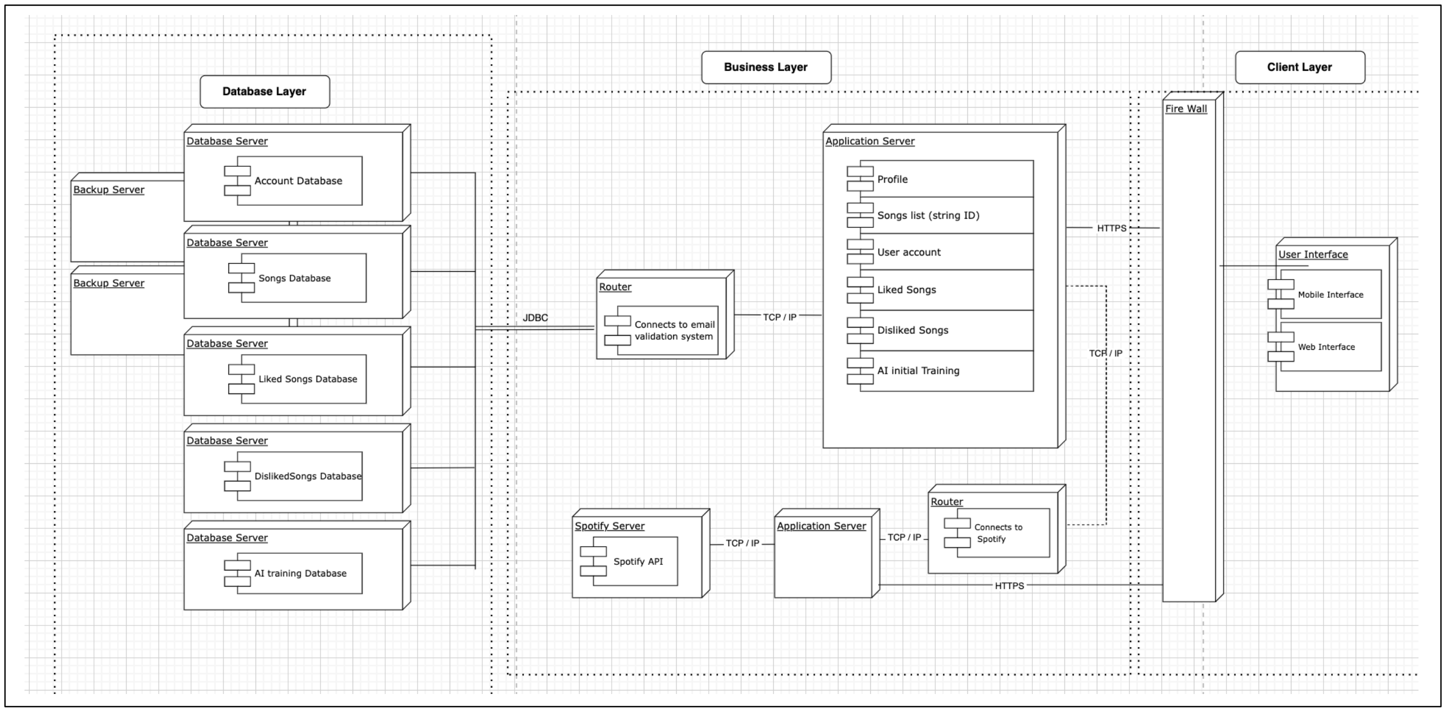
As an initial setting up of using Spring Boot:

- Install JDK as Spring Boot is Java based framework.
- Create project with Spring Initializer.
- Add dependency with Spring Boot Starter web into pom.xml file as shown below.
-

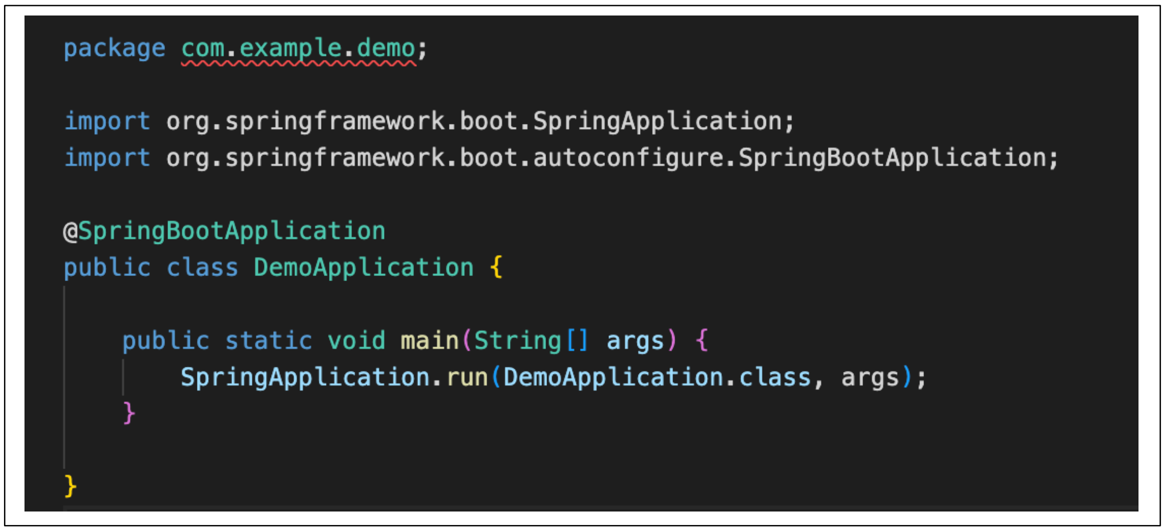


```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-cache</artifactId>
        </dependency>
```

The Back End Server architecture will be following. 3 layers architecture is much easier to build and maintain.
In fact, Spring Boot is good at building 3 layers architecture by the modular design. Also, to make scalable and RESTful server, 3 layer architecture can use strength of Spring Boot.



Rough skeleton code for building Backend Server in Boot Spring below:



```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

In addition, to include endpoints, add

@RestController (to include REST API endpoint)
@GetMapping (to include GET API endpoint)
@PostMapping (to include POST API endpoint)

Creating Disliked Songs List

The steps of deploying disliked songs list and Back End Server would be following.

1. **Define the disliked songs list API endpoints**

   Determine the API endpoints required to manage the disliked songs list using RESTful API design principles

2. **Define the disliked songs list data model**

   Determine the data model required to store the disliked songs list using an ORM tool such as Hibernate

3. **Implement the disliked songs list API endpoints**

   Implement the API endpoints defined in step I using Spring Boot's RESTful API features, such as @RestController and @RequestMapping

4. **Implement the disliked songs list data access layer**

   Implement the data access layer using Spring Boot's ORM features

5. **Implement the functionality to add and remove songs from the disliked songs list**

   Use Spring Boot's validation features to ensure the API endpoints behave as expected

   Use Spring Boot's validation features to enforce data integrity

6. **Implement the functionality to use the disliked songs list for AI training to personalize user preference**

   Use machine learning frameworks

7. **Test the backend server using frameworks such as JUnit and Mockito**

   Ensure that the backend server behaves as expected and handles errors gracefully

8. **Deploy the backend server to a production environment using Spring Boot's deployment features**