<u>Spring boot File structure</u>

In a Spring Boot application, the folder structure is designed to follow the Model-View-Controller (MVC) pattern, which helps to separate concerns within the application. Here is a typical folder structure:

```
src/
├── main/
│   ├── java/
│   │   └── com/
│   │       └── example/
│   │           └── demo/
│   │               ├── DemoApplication.java
│   │               ├── controller/
│   │               │   └── MyController.java
│   │               ├── service/
│   │               │   └── MyService.java
│   │               ├── repository/
│   │               │   └── MyRepository.java
│   │               └── model/
│   │                   └── MyModel.java
│   └── resources/
│       ├── application.properties
│       └── templates/
│           └── mytemplate.html
└── test/
    └── java/
        └── com/
            └── example/
                └── demo/
                    └── DemoApplicationTests.java
```

1. DemoApplication.java

This is the main class annotated with @SpringBootApplication. It serves as the entry point for the Spring Boot application.

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

2. Controller

Controllers handle incoming HTTP requests and return responses. They are typically annotated with @RestController or @Controller.

controller/MyController.java

```
@RestController
@RequestMapping("/api")
public class MyController {

    @Autowired
    private MyService myService;

    @GetMapping("/greeting")
    public String greet() {
        return myService.getGreeting();
    }
}
```

3. Service

Services contain the business logic. They are usually annotated with @Service.

service/MyService.java

```
@Service
public class MyService {

    public String getGreeting() {
        return "Hello, World!";
    }
}
```

4. Repository

Repositories handle data access, typically interacting with databases. They are often interfaces annotated with @Repository.

repository/MyRepository.java

```
@Repository
public interface MyRepository extends JpaRepository<MyModel, Long> {
    // Custom query methods can be defined here
}
```

5. Model

Models represent the data structure, often annotated with @Entity if they correspond to database tables.

model/MyModel.java

```
@Entity
public class MyModel {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;


    // Getters and setters
}
```

6. Resources

The resources folder contains configuration files, templates, static resources, etc.

application.properties: Configuration file for the application.

templates: Contains template files for rendering views (e.g., Thymeleaf templates).

7. Test

The test folder contains unit and integration tests for the application.

DemoApplicationTests.java

```
@SpringBootTest
class DemoApplicationTests {


    @Test
    void contextLoads() {
    }
}
```