Case Study 2: Annotation-Based

Configuration Project Title: Smart Home Automation System

Configuration Type: Annotation-based Spring

Configuration POJO Classes: Device and User

Scenario:

A smart home system manages various IoT devices like lights, fans, and ACs. Users can control these devices through an application. Each user can register and manage multiple devices. Spring annotations like @Component, @Autowired, and @Service are used to auto-wire dependencies and manage components

Components:

• User.java: Annotated with @Component, contains user details like name and home ID.

• Device.java: Annotated with @Component, represents smart devices with attributes like device type and status.

• AutomationService.java: Annotated with @Service, uses @Autowired to inject both User and Device beans to manage device control logic.

• AppConfig.java: A minimal @Configuration class with @ComponentScan to auto-detect components in the package.

• MainApp.java: Loads the context and triggers methods to control devices.

Why Annotation-Based Config?

• Reduces boilerplate and simplifies bean wiring.

• Ideal for component-based development where classes are self-contained and annotated.

• Encourages cleaner separation of concerns with automatic scanning and DI

User.java

Code:

```
package com.SmartHome;
import org.springframework.stereotype.Component;
@Component
public class User {
    private String name = "Sravani";
    private String homeId = "HOME123";
    public String getName() {
        return name;
    }
    public String getHomeId() {
        return homeId;
    }
}
```

Device.java

Code:

```
package com.SmartHome;
import org.springframework.stereotype.Component;
@Component
public class Device {
    private String deviceType = "Air Conditioner";
    private String status = "OFF";
    public String getDeviceType() {
        return deviceType;
    }
    public String getStatus() {
        return status;
    }
    public void turnOn() {
```

```
        status = "ON";
        System.out.println(deviceType + " is turned ON");
    }
    public void turnOff() {
        status = "OFF";
        System.out.println(deviceType + " is turned OFF");
    }
}
```

AutomationService.java

Code:

```
package com.SmartHome;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
@Service
public class AutomationService {
    @Autowired
    private User user;
    @Autowired
    private Device device;
    public void controlDevice() {
        System.out.println("User: " + user.getName() + ", Home ID: " + user.getHomeId());
        device.turnOn();
        System.out.println("Current Device Status: " + device.getStatus());
    }
}
```

AppConfig.java

Code:

```
package com.SmartHome;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@Configuration
@ComponentScan(basePackages = "com.smarthome")
public class AppConfig {
}
```

MainApp.java

Code:

```
package com.SmartHome;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.beans.BeansException;
public class MainApp {
   public static void main(String[] args) throws BeansException {
      ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
      AutomationService service = context.getBean(AutomationService.class);
      service.controlDevice();
   }
}
```

Output:

User: Sravani, Home ID: HOME123

Air Conditioner is turned ON

Current Device Status: ON