

Capstone Project Part-2 (API Implement)

PROJECT PACKAGE STRUCTURE

```
src/main/java/com/example/refundsapi/  
|  
├─ RefundsApiApplication.java  
|  
├─ controller/  
|   └─ RefundController.java  
|  
├─ model/  
|   ├── Refund.java  
|   ├── RefundCreateRequest.java  
|   └─ RefundStatusRequest.java  
|  
├─ service/  
|   └─ RefundService.java  
|  
└─ security/  
    ├── ApiKeyFilter.java  
    └─ SecurityConfig.java
```

1. Spring Initializr (recommended)

Open <https://start.spring.io/>

- Project: Maven Project
- Language: Java
- Spring Boot: latest 3.x (e.g. 3.2.x)
- Group: com.example
- **Artifact: refundsapi**
- Dependencies:
 - Spring Web
 - Spring Security
 - lombok

2. Model Classes

2.1 Refund.java

```
package com.example.refundsapi.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Refund {

    private String id;
    private double amount;
    private String currency;
    private String reason;
    private String status;
}
```

2.2 RefundCreateRequest.java

```
package com.example.refundsapi.model;

import lombok.Data;

@Data
public class RefundCreateRequest {

    private double amount;
```

```
private String currency;  
private String reason;  
}
```

2.3 RefundStatusRequest.java

```
package com.example.refundsapi.model;  
  
import lombok.Data;  
  
@Data  
public class RefundStatusRequest {  
  
    private String refundId;  
}
```

3. Service Layer

3.1 RefundService.java

```
package com.example.refundsapi.service;  
  
import com.example.refundsapi.model.Refund;  
import com.example.refundsapi.model.RefundCreateRequest;  
import com.example.refundsapi.model.RefundStatusRequest;  
import org.springframework.stereotype.Service;  
  
import java.util.*;  
  
@Service  
public class RefundService {
```

```
private final Map<String, Refund> store = new HashMap<>();

public Refund createRefund(RefundCreateRequest req) {
    String id = UUID.randomUUID().toString();

    Refund refund = new Refund(
        id,
        req.getAmount(),
        req.getCurrency(),
        req.getReason(),
        "PENDING"
    );

    store.put(id, refund);
    return refund;
}

public Refund getRefund(String id) {
    return Optional.ofNullable(store.get(id))
        .orElseThrow(() -> new RuntimeException("Refund Not Found"));
}

public Refund checkStatus(String id) {
    return getRefund(id);
}

public List<Refund> getAll() {
    return new ArrayList<>(store.values());
}
}
```

4. Controller Layer

4.1 RefundController.java

```
package com.example.refundsapi.controller;

import com.example.refundsapi.model.Refund;
import com.example.refundsapi.model.RefundCreateRequest;
import com.example.refundsapi.model.RefundStatusRequest;
import com.example.refundsapi.service.RefundService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/refunds")
public class RefundController {

    @Autowired
    private RefundService service;

    @GetMapping
    public List<Refund> getAllRefunds() {
        return service.getAll();
    }

    @PostMapping
```

```
public ResponseEntity<Refund> createRefund(@RequestBody RefundCreateRequest req) {  
    return ResponseEntity.status(201).body(service.createRefund(req));  
}  
  
@GetMapping("/{id}")  
public Refund getRefundById(@PathVariable String id) {  
    return service.getRefund(id);  
}  
  
@PostMapping("/status")  
public Refund getStatus(@RequestBody RefundStatusRequest req) {  
    return service.checkStatus(req.getRefundId());  
}  
}
```

5. Security Layer

5.1 ApiKeyFilter.java

```
package com.example.refundsapi.security;  
  
import jakarta.servlet.FilterChain;  
import jakarta.servlet.ServletException;  
import jakarta.servlet.http.HttpServletRequest;  
import jakarta.servlet.http.HttpServletResponse;  
  
import org.springframework.stereotype.Component;  
import org.springframework.web.filter.OncePerRequestFilter;  
  
import java.io.IOException;
```

```
@Component
public class ApiKeyFilter extends OncePerRequestFilter {

    private static final String TOKEN = "my-secret-token";

    @Override
    protected void doFilterInternal(
        HttpServletRequest request,
        HttpServletResponse response,
        FilterChain chain
    ) throws ServletException, IOException {

        String header = request.getHeader("Authorization");

        if (header == null || !header.equals("Bearer " + TOKEN)) {
            response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
            response.getWriter().write("Unauthorized");
            return;
        }

        chain.doFilter(request, response);
    }
}
```

5.2 SecurityConfig.java

```
package com.example.refundsapi.security;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```

import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFil
ter;

@Configuration
public class SecurityConfig {

    @Autowired
    private ApiKeyFilter apiKeyFilter;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

        return http
            .csrf(csrf -> csrf.disable()) // ← FIX CSRF for POST
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/refunds/**").permitAll() // Allow API calls but
protected by filter
                .anyRequest().authenticated()
            )
            .addFilterBefore(apiKeyFilter, UsernamePasswordAuthenticationFilter.class)
            .build();
    }
}

```

A. pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.7</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>refundsapi</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>refundsapi</name>
  <description>Demo project for Spring Boot</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
```

```
<properties>
  <java.version>17</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
```

```
<artifactId>maven-compiler-plugin</artifactId>
<configuration>
  <annotationProcessorPaths>
    <path>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
    </path>
  </annotationProcessorPaths>
</configuration>
</plugin>
<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
  <configuration>
    <excludes>
      <exclude>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
      </exclude>
    </excludes>
  </configuration>
</plugin>
</plugins>
</build>

</project>
```