**Exercise 1: Setting Up RESTful Services**

mvn archetype:generate -DgroupId=com.example.bookstoreapi -DartifactId=BookstoreAPI Dversion=1.0.0 -DinteractiveMode=false

#pom.xml

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<scope>provided</scope>

</dependency>

</dependencies>

**Exercise 2: Creating Basic REST Controllers**

@RestController

@RequestMapping("/books")

public class BookController {

@GetMapping

public List<Book> getAllBooks() {

}

@PostMapping public Book addBook(@RequestBody Book book) {

//imp

}

@PutMapping("/{id}") public Book updateBook(@PathVariable Long id, @RequestBody Book book) {

//imp

}

@DeleteMapping("/{id}") public void deleteBook(@PathVariable Long id) {

// Imp

}

}

@Data @Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id; private String title; private String author; private Double price; private String isbn;

}

**Exercise 3: Handling Path Variables and Query Parameters**

@GetMapping("/{id}") public Book getBookById(@PathVariable Long id) {

// Imp

}

@GetMapping("/search") public List<Book> searchBooks(@RequestParam(required = false) String title,

@RequestParam(required = false) String author) {

// Imp

}

**Exercise 4: Processing Request Body and Form Data**

@PostMapping("/customers") public Customer createCustomer(@RequestBody Customer customer) {

// Imp

}

@PostMapping("/register") public String registerCustomer(@RequestParam String name, @RequestParam String email) {

// Imp

}

**Exercise 5: Customizing Response Status and Headers**

@ResponseStatus(HttpStatus.CREATED)

@PostMapping("/books") public ResponseEntity<Book> createBook(@RequestBody Book book) {

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "foo");

return new ResponseEntity<>(book, headers, HttpStatus.CREATED);

}

**Exercise 6: Exception Handling in REST Controllers**

@ControllerAdvice public class GlobalExceptionHandler {

@ExceptionHandler(ResourceNotFoundException.class)

@ResponseStatus(HttpStatus.NOT\_FOUND) public ResponseEntity<String> handleNotFound(ResourceNotFoundException ex) {

return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT\_FOUND);

}

// exceptions

}

**Exercise 7: Introduction to Data Transfer Objects (DTOs) Topic:**

@Data

public class BookDTO { private Long id; private String title; private String author; private Double price;

}

BookDTO convertToDTO(Book book) {

ModelMapper modelMapper = new ModelMapper(); return modelMapper.map(book, BookDTO.class);

}