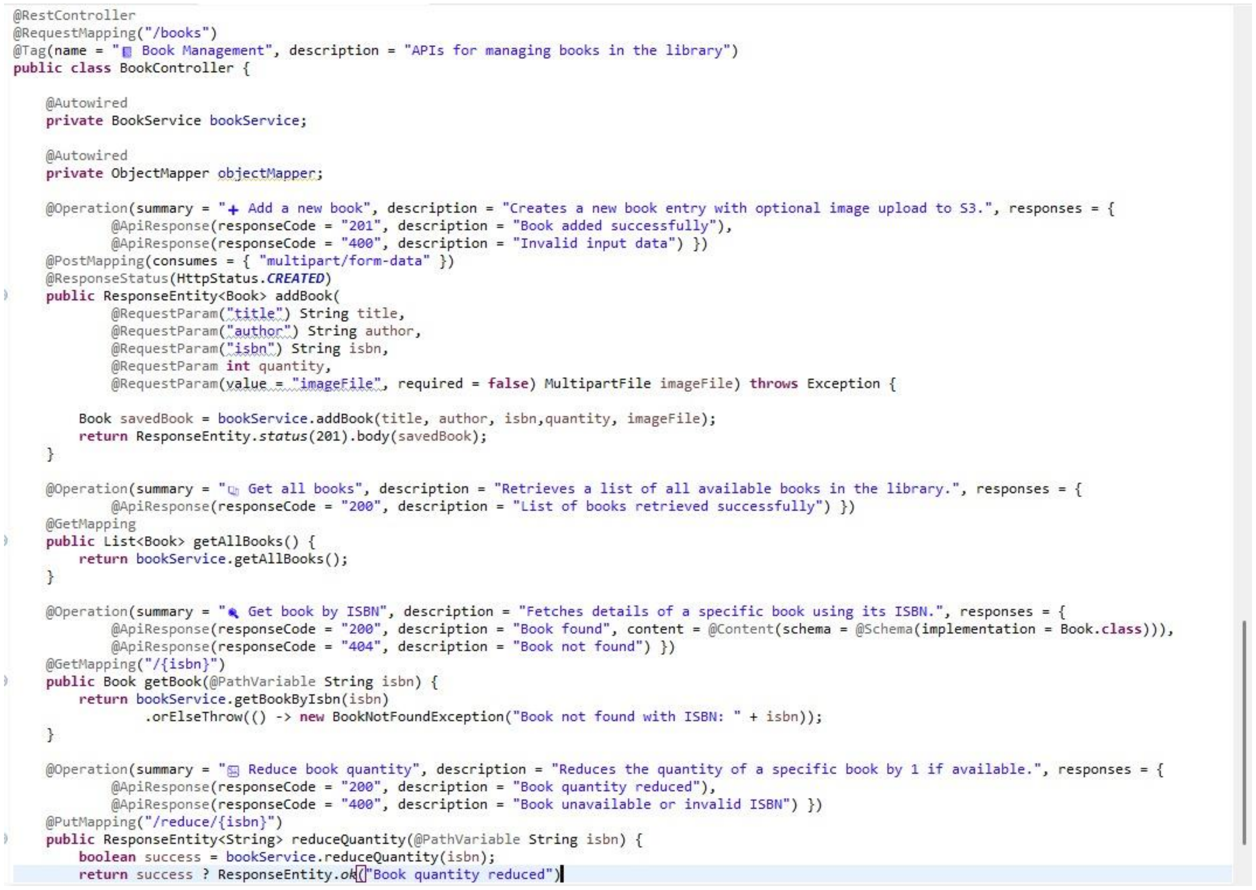
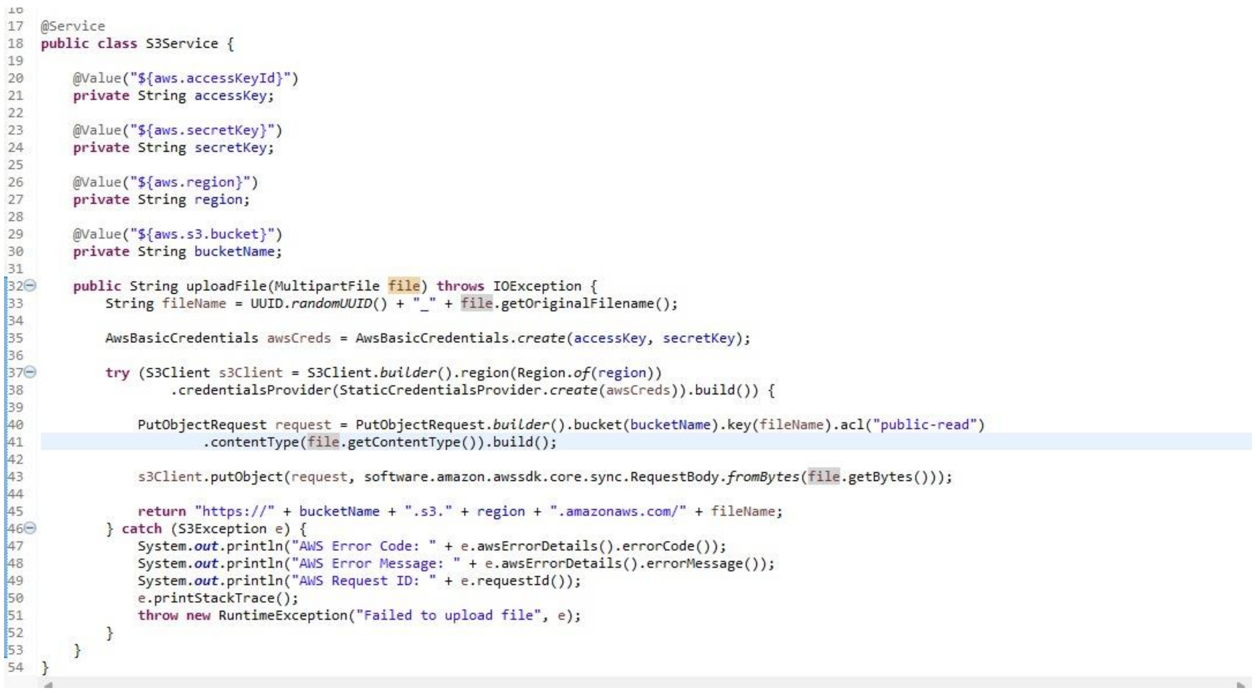
Daily task – 17/10/2025

Controller:



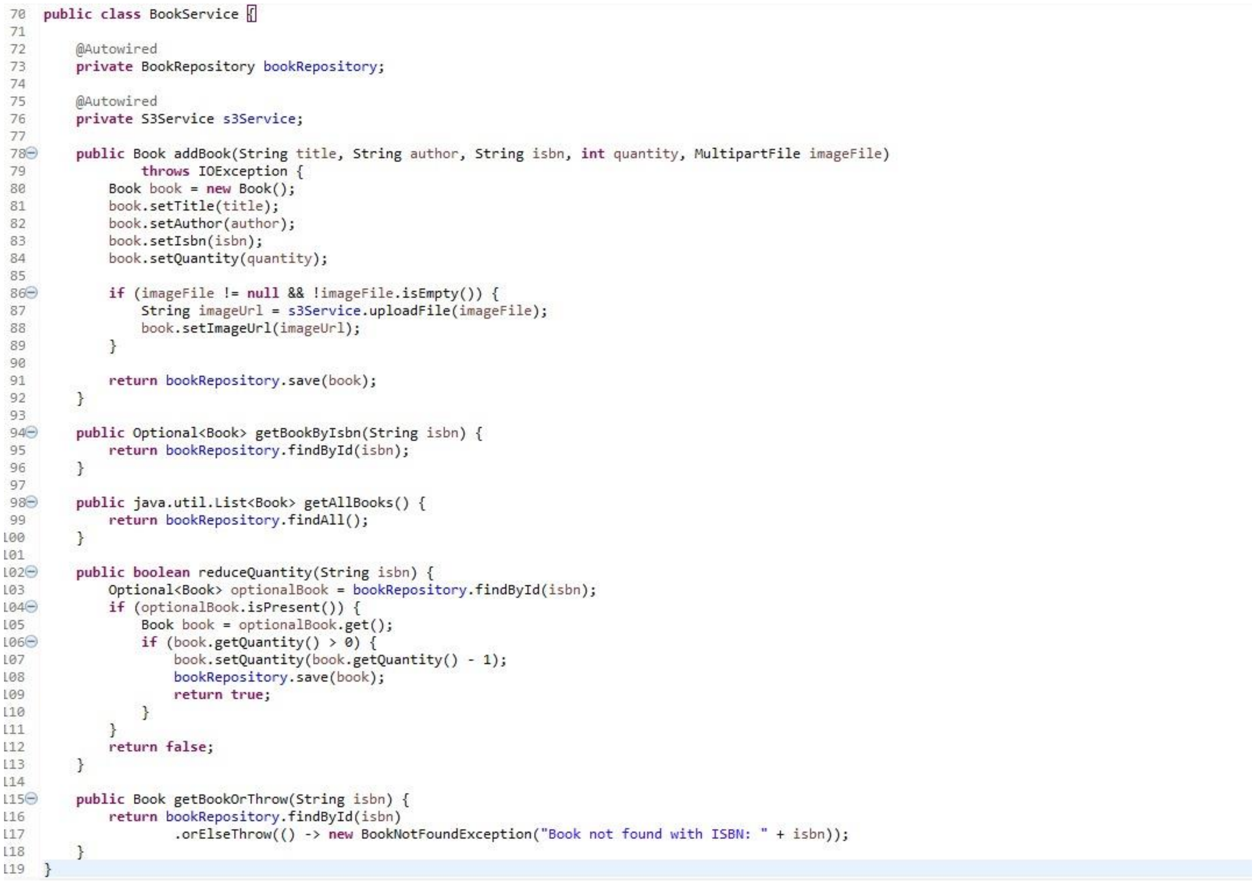
```java
@RestController
@RequestMapping("/books")
@Tag(name = "📘 Book Management", description = "APIs for managing books in the library")
public class BookController {

    @Autowired
    private BookService bookService;

    @Autowired
    private ObjectMapper objectMapper;

    @Operation(summary = "➕ Add a new book", description = "Creates a new book entry with optional image upload to S3.", responses = {
            @ApiResponse(responseCode = "201", description = "Book added successfully"),
            @ApiResponse(responseCode = "400", description = "Invalid input data") })
    @PostMapping(consumes = { "multipart/form-data" })
    @ResponseStatus(HttpStatus.CREATED)
    public ResponseEntity<Book> addBook(
            @RequestParam("title") String title,
            @RequestParam("author") String author,
            @RequestParam("isbn") String isbn,
            @RequestParam int quantity,
            @RequestParam(value = "imageFile", required = false) MultipartFile imageFile) throws Exception {

        Book savedBook = bookService.addBook(title, author, isbn,quantity, imageFile);
        return ResponseEntity.status(201).body(savedBook);
    }

    @Operation(summary = "📚 Get all books", description = "Retrieves a list of all available books in the library.", responses = {
            @ApiResponse(responseCode = "200", description = "List of books retrieved successfully") })
    @GetMapping
    public List<Book> getAllBooks() {
        return bookService.getAllBooks();
    }

    @Operation(summary = "🔍 Get book by ISBN", description = "Fetches details of a specific book using its ISBN.", responses = {
            @ApiResponse(responseCode = "200", description = "Book found", content = @Content(schema = @Schema(implementation = Book.class))),
            @ApiResponse(responseCode = "404", description = "Book not found") })
    @GetMapping("/{isbn}")
    public Book getBook(@PathVariable String isbn) {
        return bookService.getBookByIsbn(isbn)
                .orElseThrow(() -> new BookNotFoundException("Book not found with ISBN: " + isbn));
    }

    @Operation(summary = "📉 Reduce book quantity", description = "Reduces the quantity of a specific book by 1 if available.", responses = {
            @ApiResponse(responseCode = "200", description = "Book quantity reduced"),
            @ApiResponse(responseCode = "400", description = "Book unavailable or invalid ISBN") })
    @PutMapping("/reduce/{isbn}")
    public ResponseEntity<String> reduceQuantity(@PathVariable String isbn) {
        boolean success = bookService.reduceQuantity(isbn);
        return success ? ResponseEntity.ok("Book quantity reduced")
```

S3 service:



```java
@Service
public class S3Service {

    @Value("${aws.accessKeyId}")
    private String accessKey;

    @Value("${aws.secretKey}")
    private String secretKey;

    @Value("${aws.region}")
    private String region;

    @Value("${aws.s3.bucket}")
    private String bucketName;

    public String uploadFile(MultipartFile file) throws IOException {
        String fileName = UUID.randomUUID() + "_" + file.getOriginalFilename();

        AwsBasicCredentials awsCreds = AwsBasicCredentials.create(accessKey, secretKey);

        try (S3Client s3Client = S3Client.builder().region(Region.of(region))
                .credentialsProvider(StaticCredentialsProvider.create(awsCreds)).build()) {

            PutObjectRequest request = PutObjectRequest.builder().bucket(bucketName).key(fileName).acl("public-read")
                    .contentType(file.getContentType()).build();

            s3Client.putObject(request, software.amazon.awssdk.core.sync.RequestBody.fromBytes(file.getBytes()));

            return "https://" + bucketName + ".s3." + region + ".amazonaws.com/" + fileName;
        } catch (S3Exception e) {
            System.out.println("AWS Error Code: " + e.awsErrorDetails().errorCode());
            System.out.println("AWS Error Message: " + e.awsErrorDetails().errorMessage());
            System.out.println("AWS Request ID: " + e.requestId());
            e.printStackTrace();
            throw new RuntimeException("Failed to upload file", e);
        }
    }
}
```

Service:



```java
public class BookService {

    @Autowired
    private BookRepository bookRepository;

    @Autowired
    private S3Service s3Service;

    public Book addBook(String title, String author, String isbn, int quantity, MultipartFile imageFile)
            throws IOException {
        Book book = new Book();
        book.setTitle(title);
        book.setAuthor(author);
        book.setIsbn(isbn);
        book.setQuantity(quantity);

        if (imageFile != null && !imageFile.isEmpty()) {
            String imageUrl = s3Service.uploadFile(imageFile);
            book.setImageUrl(imageUrl);
        }

        return bookRepository.save(book);
    }

    public Optional<Book> getBookByIsbn(String isbn) {
        return bookRepository.findById(isbn);
    }

    public java.util.List<Book> getAllBooks() {
        return bookRepository.findAll();
    }

    public boolean reduceQuantity(String isbn) {
        Optional<Book> optionalBook = bookRepository.findById(isbn);
        if (optionalBook.isPresent()) {
            Book book = optionalBook.get();
            if (book.getQuantity() > 0) {
                book.setQuantity(book.getQuantity() - 1);
                bookRepository.save(book);
                return true;
            }
        }
        return false;
    }

    public Book getBookOrThrow(String isbn) {
        return bookRepository.findById(isbn)
                .orElseThrow(() -> new BookNotFoundException("Book not found with ISBN: " + isbn));
    }
}
```