Software Requirement Specification

For

Library Management System

Course Module: (VLC-SCFSD10-25-0650)

Year: March2025

Author: Karen Chan

[1. Introduction 3](#_Toc193909308)

[2. General Overview: 4](#_Toc193909309)

[3. Functional Requirements 4](#_Toc193909310)

[4. Coding Specific Guidelines 4](#_Toc193909311)

[1. Java (Spring Boot) 4](#_Toc193909312)

[2. JavaScript (React) 4](#_Toc193909313)

[3. CSS 5](#_Toc193909314)

[4. Database Styles (MySQL) 5](#_Toc193909315)

# Introduction

The document you've provided is a **Full-Stack Web Development Capstone Coding Style Guide**, which outlines coding conventions and best practices for the development of a Library Management System project. Here's a summary of the key points:

# General Overview:

* The guide ensures that all code adheres to a set of consistent conventions to improve **quality, readability,** and **maintainability**.
* It includes specific guidelines for **Java (Spring Boot)**, **JavaScript (React)**, **CSS**, and **MySQL**.

# Functional Requirements

Refer to < Business Requirements Document.docx>

# Coding Specific Guidelines

## 1. Java (Spring Boot)

* **Naming Conventions:**
  + **Classes:** PascalCase (e.g., BookController, BookService)
  + **Methods:** camelCase (e.g., getBookById, addBook)
  + **Variables:** camelCase (e.g., bookTitle, bookId)
  + **Constants:** UPPER\_SNAKE\_CASE (e.g., MAX\_BOOK\_COPIES)
* **Indentation:** 4 spaces.
* **Comments:**
  + Use **Javadoc** for class and method documentation.
  + Add comments for explaining complex logic.
* **Imports:** Organize imports and avoid using wildcard imports.
* **Lombok:** Use annotations like **@Getter**, **@Setter**, **@NoArgsConstructor**, **@AllArgsConstructor** to reduce boilerplate code.
* **Error Handling:** Use try-catch blocks to handle exceptions and log errors properly.

## 2. JavaScript (React)

* **Naming Conventions:**
  + **Components:** PascalCase (e.g., BookList, BookForm)
  + **Functions:** camelCase (e.g., handleInputChange, handleSubmit)
  + **Variables:** camelCase (e.g., bookTitle, bookId)
* **Indentation:** 2 spaces.
* **Comments:**
  + Add comments for explaining complex logic.
* **JSX:** Ensure consistent formatting for JSX code.
* **State Management:** Use **React hooks** (e.g., useState, useEffect) for managing state in components.
* **Arrow Functions:** Use arrow functions for concise function definitions.

## 3. CSS

* **Naming Conventions:** Use meaningful class names.
* **Organization:** Separate CSS rules into individual files.
* **Specificity:** Avoid overly specific CSS selectors.
* **Consistent Styling:** Maintain consistent fonts, colors, and spacing across the project.

## 4. Database Styles (MySQL)

* **Naming Conventions:**
  + **Table names:** Use plural, lowercase, underscore\_case (e.g., book\_details).
  + **Column names:** Use lowercase, underscore\_case (e.g., book\_title, book\_id).
  + **Primary keys:** table\_name\_id (e.g., book\_id).
  + **Foreign keys:** referenced\_table\_name\_id (e.g., author\_id).
* **Data Types:**
  + Use appropriate data types for columns.
  + Use **VARCHAR** with reasonable limits for text-based data.
  + Use **DECIMAL** for monetary values.
  + Choose appropriate date/time types (e.g., DATE, DATETIME).

This coding style guide provides a structured approach for the development of the Library Management System, ensuring all team members adhere to the same conventions for a smooth and efficient development process.