# Comprehensive Reference Document for Assignments

## Table of Contents

1. [Introduction](#introduction)

2. [Step-by-Step Instructions](#step-by-step-instructions)

- [Assignment 1: Event Management System](#assignment-1-event-management-system)

- [Assignment 2: Task Management System with Secure Authentication](#assignment-2-task-management-system-with-secure-authentication)

- [Assignment 3: Library Management System with Audit Logging](#assignment-3-library-management-system-with-audit-logging)

3. [Best Practices](#best-practices)

4. [Submission Guidelines](#submission-guidelines)

## Introduction

This reference document provides clear instructions and guidelines for completing three assignments, each focusing on developing backend APIs using Node.js and Express.js frameworks to handle different management systems. These assignments aim to enhance your understanding of backend development, including database interactions, dependency injection, security, and code quality.

### Objectives

- Effectively manage complex data and relationships.

- Ensure secure access and role-based permissions.

- Implement essential backend features such as PDF generation and audit logging.

- Maintain high standards of code quality and best practices.

## Step-by-Step Instructions

### Assignment 1: Event Management System

#### Task Objectives:

- Create, update, and delete events with nested data structures.

- Register participants with validation.

- Retrieve detailed event reports.

#### Steps:

1. **Set Up Project Environment:**

- Install Node.js and Express.js.

- Set up an Express.js server.

2. **Define Endpoints:**

- Create routes for handling event operations (create, update, delete).

- Ensure these routes are capable of managing nested data (e.g., multiple sessions, speakers).

3. **Database Interaction:**

- Use TypeORM or Drizzle to define the schema for events, sessions, and participants.

- Set up complex relationships between these entities.

4. **Dependency Injection:**

- Implement DI for various services, including database operations, event logic, and PDF generation.

5. **Participant Registration:**

- Create an endpoint for registering participants.

- Add validation logic to enforce the maximum number of participants per event.

6. **Event Reports:**

- Develop functionality to generate detailed event reports, incorporating nested lists of sessions and participants.

- Implement PDF generation with specific formatting requirements.

7. **Code Quality:**

- Utilize appropriate design patterns to ensure a clean and maintainable code base.

### Assignment 2: Task Management System with Secure Authentication

#### Task Objectives:

- User registration and role-based access control using JWT.

- Manage tasks with CRUD operations, permissions based on user roles.

- Filter tasks by priority, status, and assigned user.

#### Steps:

1. **Set Up Project Environment:**

- Install Node.js and Express.js.

- Set up an Express.js server.

2. **Define User Endpoints:**

- Create routes for user registration and authentication.

- Implement JWT-based authentication and role-based access control.

3. **Task Management Endpoints:**

- Develop CRUD operations for tasks.

- Enforce permissions based on user roles (e.g., admin, user).

4. **Database Interaction:**

- Use TypeORM or Drizzle to define the schema for users, roles, and tasks.

- Ensure data integrity and relationships.

5. **Dependency Injection:**

- Implement DI for authentication services, role management, and task handling.

6. **Security Measures:**

- Add route guards and middleware for access control and input validation.

7. **Code Quality:**

- Follow best practices and use appropriate design patterns for code organization.

### Assignment 3: Library Management System with Audit Logging

#### Task Objectives:

- CRUD operations for books, authors, and users.

- Manage borrowing and returning of books with constraints.

- Implement audit logging for all user actions.

#### Steps:

1. **Set Up Project Environment:**

- Install Node.js and Express.js.

- Set up an Express.js server.

2. **Define CRUD Endpoints:**

- Create routes for managing books, authors, and users.

- Implement borrowing and returning books with constraints like max borrow limit and due dates.

3. **Audit Logging:**

- Develop functionality to track user actions (borrowing, returning books) with timestamps.

- Implement filtering options to retrieve specific logs.

4. **Database Interaction:**

- Use TypeORM or Drizzle to define the schema and manage relationships between books, authors, users, and logs.

5. **Dependency Injection:**

- Implement DI for database interactions, audit logging, and borrowing logic.

6. **Code Quality:**

- Ensure high code quality by leveraging design patterns and following best practices.

## Best Practices

- **Database Schema Design:**

- Structure your schema to support complex relationships and ensure data integrity.

- **Error Handling:**

- Implement robust error-handling mechanisms throughout your API endpoints.

- **Modularization:**

- Organize your code into modular components for better readability and maintainability.

- **Security:**

- Apply best practices in authentication, authorization, and secure coding to protect your application.

- **Documentation:**

- Document your code and APIs for easy understanding and maintenance.

## Submission Guidelines

1. **Format:**

- Submit your code in a Git repository.

- Ensure the repository includes a README file explaining the setup and usage instructions.

2. **Files:**

- Include all necessary files such as server scripts, configuration files, and database schemas.

3. **Instructions:**

- Provide clear instructions on how to run your project, including any prerequisite installations and initial setups.

4. **Deadline:**

- Submit your assignment on or before the specified deadline.

By following these comprehensive instructions and best practices, you will be well-equipped to complete your assignments effectively and maintain high standards of code quality.