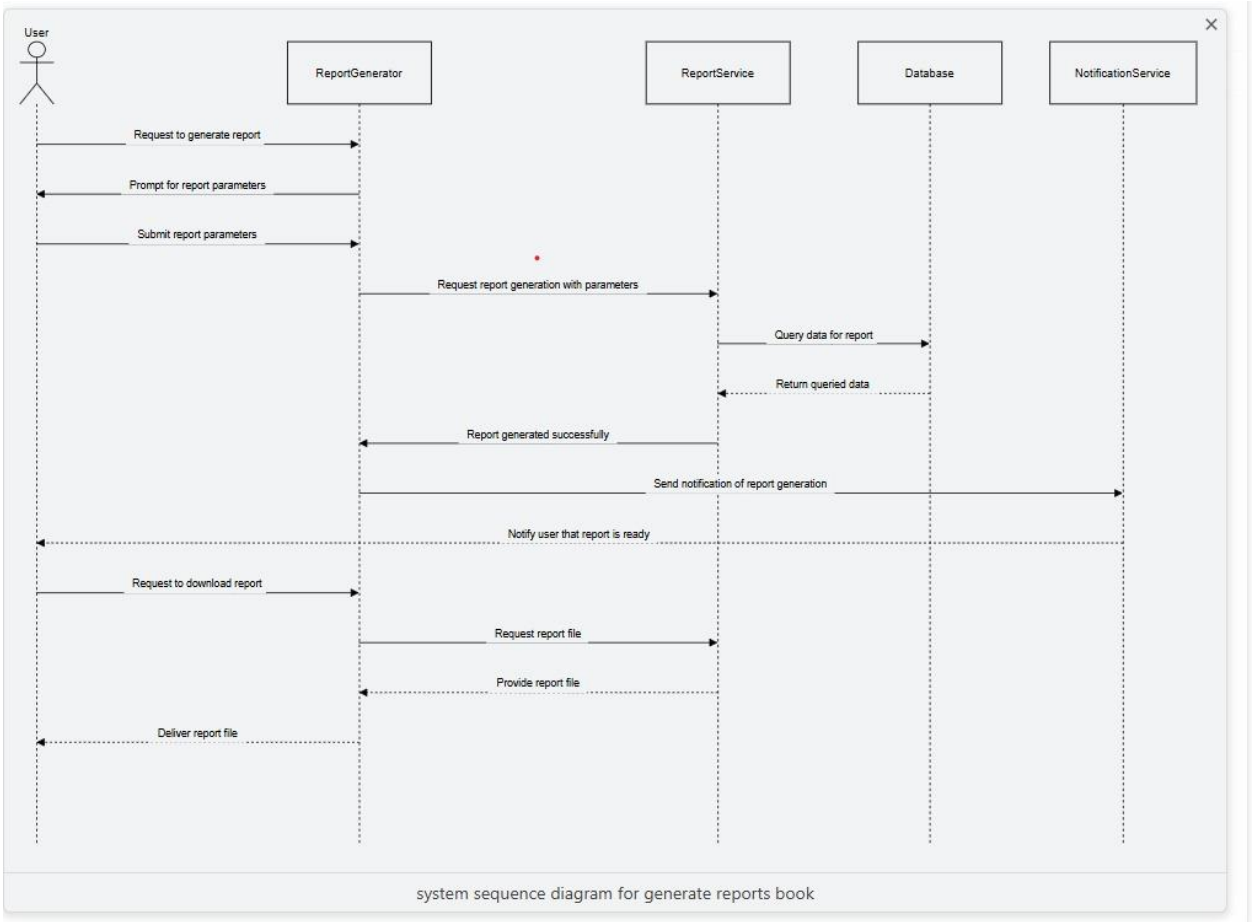
Khadija zamir SP23-BSE-025

Fully Dressed Use case: generate report

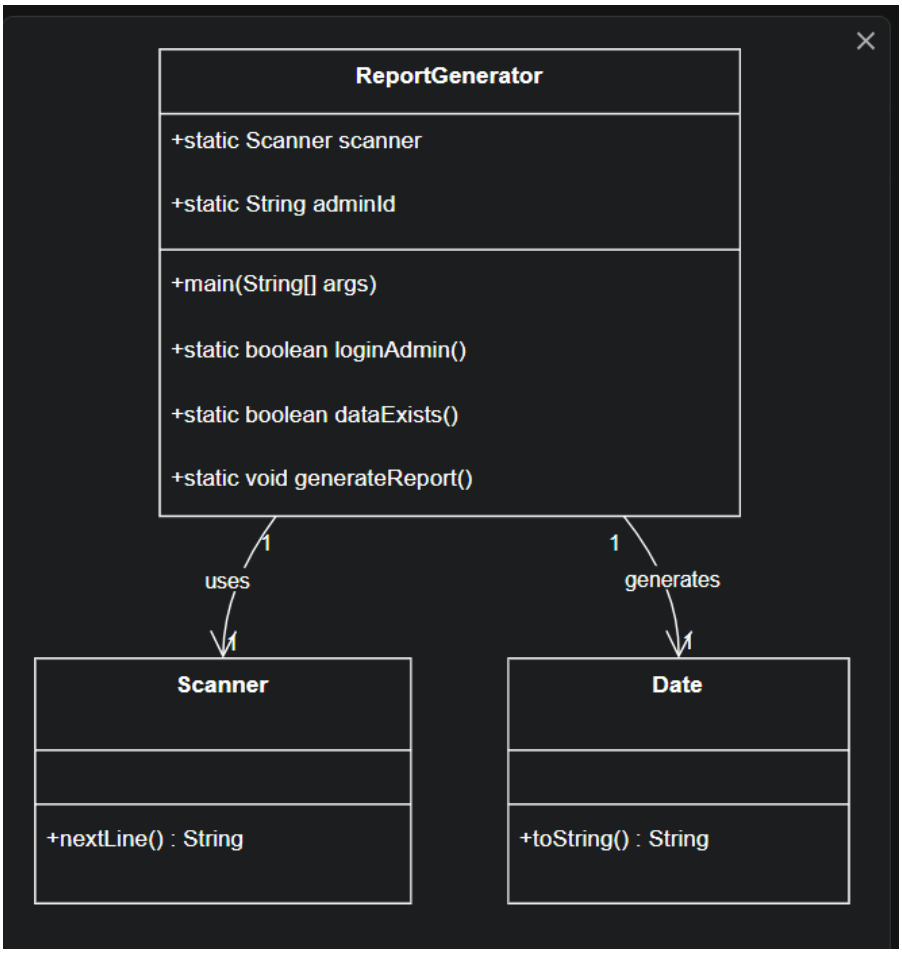
Section	Details
Use Case Name	Generate Reports
Primary Actor	Admin
Trigger	Admin clicks on the "Generate Report" button
Preconditions	1. Admin must be logged into system 2. Relevant data (e.g., borrowing records, returns, catalog) must exist
Main Success Scenario	1. Admin selects the “Generate Reports” function 2. Specifies the type of report (e.g., borrowing frequency, overdue books, catalog additions) 3. Choosing report format (PDF, Excel) 4. System processes the data. 5. The system generates and displays the report. 6. Admin reviews and downloads it
Postconditions	1. Report is generated and available for download/view 2. Report is stored for future reference
Alternate Flows	**2a:** If type or format is not specified, system prompts for missing info **4a:** If there is no data, the system shows error "No data available for the selected criteria"
Special Requirements	1. Report generation must be completed within 10 seconds. 2. Downloadable in PDF and Excel. 3. Stored with timestamp and admin ID. 4. Filter options: date range, department, or category

System Sequence Diagram



system sequence diagram for generate reports book

Class diagram:



Coding Standards

Chapter 7: Coding Standards – Generate Reports

The "Generate Reports" feature is implemented in Java using standard object-oriented practices. The following coding standards were followed:

- **Class names use PascalCase (e.g., ReportGenerator)**
- **Method and variable names use camelCase (e.g., generateReport, reportType)**
- **Indentation is set to 4 spaces**
- **Braces are placed on the same line as the statement**
- **Comments use // for single-line and /* */ for method or block explanations**
- **Each class is placed in a separate file, and methods are kept small and specific**
- **Error handling includes checking for missing input or data, using try-catch for exceptions**