

Library Management System

Phases in Software Development Project – Overview

Introduction

The Software Development Life Cycle (SDLC) is a systematic and structured approach to software development that ensures the delivery of high-quality software products. It consists of multiple phases such as requirement analysis, design, development, testing, deployment, and maintenance. Each phase plays a crucial role in transforming user requirements into a fully functional system.

This project focuses on understanding and implementing the SDLC phases through the development of a Library Management System using the Iterative Development Model. The system is designed to manage library operations such as book management, issuing and returning books, and maintaining records efficiently using modern web technologies.

Objective

The main objectives of this project are:

1. To understand and apply the phases of the SDLC using the Iterative model.
2. To design and develop a functional Library Management System.
3. To automate basic library operations such as book entry, issue, and return.
4. To implement a backend using FastAPI and a frontend using HTML, CSS, and JavaScript.
5. To ensure data consistency, usability, and ease of maintenance.

Theoretical Background

The Software Development Life Cycle (SDLC) provides a framework for systematic software development. Different SDLC models such as Waterfall, Spiral, and Iterative models define various development strategies.

The Iterative Model emphasizes incremental development where the system is built in small modules, tested, and refined continuously. This approach is well-suited for systems like a Library Management System, where features such as book management, issuing, and returning can be developed and improved step-by-step based on feedback.

Key Phases in SDLC

Requirements Gathering and Analysis: Identify functional and non-functional requirements such as managing books, issuing records, and maintaining availability status.

Design: Create system architecture, database schema, and user interface layouts.

Development: Implement backend APIs and frontend interfaces incrementally.

Testing: Validate system functionality and correctness through different testing methods.

Deployment: Make the system available for end users.

Maintenance: Improve performance, fix bugs, and add new features over time.

Procedure

This project follows the Iterative SDLC methodology, where each module is developed, tested, and refined independently. Feedback is incorporated at each stage to ensure reliability and correctness of the Library Management System.

Implementation Steps

Phase 1: Requirements Gathering and Analysis

Objective: Understand the needs of the library system and define clear requirements.

Activities:

- Interact with stakeholders (librarian and users) to understand system needs.
- Identify core functionalities: Add and manage books, View available books, Issue books to students, Return issued books.
- Prioritize features for the first iteration focusing on basic book management and issue/return operations.

Deliverables:

- Requirements specification document
- List of functional and non-functional requirements
- Use case diagrams

Phase 2: Design

Objective: Create a blueprint for the system architecture and user interface.

Activities:

1. Database Design:

- Design the database schema using an Entity-Relationship Diagram (ERD).
- Define key entities: Book (id, book_id, title, author, category, quantity), IssueRecord (id, student_name, book_id, issue_date, return_date, status)

2. API Design:

- Design REST API structure with endpoints for adding books, viewing books, issuing books, and returning books.

3. UI Design:

- Design user interfaces for book entry, issue, and return pages.
- Create wireframes for each page.

Tools Used: Database design tools, Basic UI wireframes, API documentation tools

Deliverables:

- Entity-Relationship Diagram (ERD)
- Database schema
- API endpoint specifications
- UI wireframes and mockups

Phase 3: Development

Objective: Implement the system based on the design specifications.

3.1 Database Setup

Activities:

- Create tables using SQLite database.
- Define relationships using SQLAlchemy ORM.
- Implement database connection and session management.

Files Created:

- database.py - Database configuration and session management
- models.py - SQLAlchemy ORM models for Book and IssueRecord

3.2 Backend Development

Activities:

- Use FastAPI to develop REST APIs.
- Implement CRUD operations for books: POST /api/books/, GET /api/books/, GET /api/books/{book_id}
- Implement book issue and return logic: POST /api/issue/, GET /api/issued-books/, PUT /api/return/{issue_record_id}

- Validate data using Pydantic schemas.
- Enable CORS for frontend communication.

Files Created:

- main.py - FastAPI application with all endpoints
- schemas.py - Pydantic validation schemas
- requirements.txt - Python dependencies

3.3 Frontend Development

Activities:

- Create web pages using HTML5: Home page, Add Book page, View Books page, Issue Book page, Return Book page
- Style pages using CSS3: Modern gradient design, Responsive layout, Smooth animations, Status badges
- Use JavaScript Fetch API: Form submission handling, Dynamic data rendering, Error handling

Files Created:

- index.html - Home page
- add-book.html & add-book.js - Add book functionality
- view-books.html & view-books.js - View books functionality
- issue-book.html & issue-book.js - Issue book functionality
- return-book.html & return-book.js - Return book functionality
- styles.css - Global styling

3.4 Integration

- Connect frontend pages with FastAPI endpoints.
- Test APIs using browser and Postman.
- Ensure proper error handling and validation.
- Implement automatic quantity management.

Tools Used:

- FastAPI - Backend framework
- Python 3.8+ - Programming language
- SQLite - Database
- SQLAlchemy - ORM
- Pydantic - Data validation
- HTML, CSS, JavaScript - Frontend
- VS Code - Code editor
- Postman - API testing
- Uvicorn - ASGI server

Phase 4: Testing

Objective: Ensure the system works correctly and meets all requirements.

4.1 Unit Testing

Activities:

- Test individual API endpoints for correct responses.
- Validate Pydantic schema validations.
- Test database operations (CRUD).

4.2 Integration Testing

Activities:

- Verify interaction between frontend and backend.
- Test complete workflows: Add book → View in books list, Issue book → Quantity decreases, Return book → Quantity increases

4.3 System Testing

Activities:

- Ensure correct behavior for book issue and return scenarios.
- Test edge cases: Issuing when quantity is 0, Returning already returned book, Invalid date inputs

4.4 Performance Testing

Activities:

- Check system performance with multiple records.
- Test response times for API calls.
- Verify database query efficiency.

4.5 Security Testing

Activities:

- Validate input handling and prevent invalid data entry.
- Test SQL injection prevention (ORM protection).
- Verify CORS configuration.

Testing Results:

- ✓ All API endpoints functioning correctly
- ✓ Frontend-backend integration successful
- ✓ Data validation working as expected
- ✓ Automatic quantity management operational
- ✓ Error handling implemented properly

Phase 5: Deployment

Objective: Make the system available for end users.

Activities:

- Backend Deployment: Run FastAPI server using Uvicorn (Command: `uvicorn main:app --reload`), Server runs on `http://localhost:8000`
- CORS Configuration: Ensure CORS is enabled for frontend access, Allow all origins for development
- Frontend Deployment: Deploy frontend as static files, Open `index.html` in web browser
- Verification: Verify API accessibility at `http://localhost:8000/docs`, Test system stability, Ensure all pages load correctly

Deployment Environment:

- Local development environment
- Windows OS
- Python 3.8+
- Modern web browser (Chrome, Firefox, Edge)

Phase 6: Maintenance

Objective: Ensure long-term system reliability and continuous improvement.

Activities:

- Monitoring: Monitor system performance, Track error logs, Analyze user feedback
- Bug Fixes: Fix bugs reported by users, Improve error handling, Optimize database queries
- Feature Enhancements: User authentication, Fine calculation for late returns, Book reservation system, Search and filter, Report generation, Email notifications
- Future Upgrades: Online access with cloud deployment, Admin dashboard with analytics, Mobile application, Barcode scanning, Multi-library support

Outcomes and Evaluation

Achievements

- Requirements Analysis: Clear requirements were analyzed and implemented effectively. All core functionalities identified and delivered.
- Database Design: A structured database ensured efficient data handling. Proper relationships established between entities.
- Automation: Core library operations were automated successfully. Automatic quantity management implemented.

- **Testing:** Testing ensured reliability and correctness of the system. All test cases passed successfully.
- **Deployment:** The deployed system simplified library management tasks. User-friendly interface with modern design.
- **Development Methodology:** Iterative development allowed flexibility and continuous improvement. Modular approach facilitated easy maintenance.

Learning Outcomes

- **SDLC Understanding:** Gained practical experience with all SDLC phases. Understood the importance of iterative development.
- **Technical Skills:** Learned FastAPI framework, Mastered SQLAlchemy ORM, Implemented RESTful API design, Enhanced frontend development skills.
- **Best Practices:** Code organization and modularity, Separation of concerns (MVC pattern), Data validation and error handling, API documentation and testing.

Conclusion

The Library Management System project successfully demonstrates the practical application of the Software Development Life Cycle using the Iterative Development Model. The system effectively automates library operations, provides a user-friendly interface, and ensures data integrity through proper validation and error handling.

The project achieved all its objectives:

- Implemented all SDLC phases systematically
- Developed a fully functional Library Management System
- Automated book management, issue, and return operations
- Created a robust backend with FastAPI and frontend with vanilla JavaScript
- Ensured data consistency and ease of maintenance

The iterative approach proved beneficial, allowing for continuous refinement and improvement at each stage. The system is scalable, maintainable, and ready for future enhancements.


References

6. FastAPI Documentation - <https://fastapi.tiangolo.com/>
7. SQLAlchemy Documentation - <https://www.sqlalchemy.org/>
8. Software Development Life Cycle (SDLC) Models
9. Iterative Development Methodology
10. RESTful API Design Principles
11. Web Development Best Practices

Project Completed: January 2026

Development Model: Iterative SDLC

Technologies: FastAPI, Python, SQLite, HTML, CSS, JavaScript

Status:  Fully Functional and Deployed